Computational Methods for Interpretable Analysis of Uncertain and Incomplete High-Dimensional Biological Data

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Susanne Zabel
aus Viersen

Tübingen
2025

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	17.10.2025
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter:	Prof. Dr. Kay Nieselt
2. Berichterstatter:	Dr. Dmitry Kobak

Disclaimer: this thesis uses Federico Marotta's `kaobook` template based on Ken Arroyo Ohori's doctoral thesis. I am grateful to Frank Schneider and Felix Dangel who added many stylistic features.

Acknowledgments

First and foremost, I would like to thank Kay Nieselt for her guidance and support throughout my PhD. I highly value the positive work environment you are creating, as well as your readily open office door, which facilitates valuable conversations about research, academic direction, and personal concerns. Further, I wish to thank Philipp Hennig for his supervision and valuable advice. Also, I am grateful to Dmitry Kobak for evaluating this thesis, as well as to Nico Pfeifer for joining my examination committee.

The University of Tübingen and the Institute for Bioinformatics and Medical Informatics have provided a great and diverse research environment. In particular, I thank my fantastic colleagues from the *Integrative Transcriptomics* group, André Hennig, Aydin Can Polatkan, Theresa Harbig, Mathias Witte Paz, Dilek Tuncbilek-Dere, Simon Hackl, Wolfgang Fuhl, Meret Häusler, Maik Wolfram-Schauerte, and Thea for a productive and friendly atmosphere. Thank you and also Axel Fehrenbach, Nicolas Brich, Marco Schäfer, Julia Fischer, and Martin Lang for the many shared lunches, coffee breaks, research discussions, and Friday beers. Further, I would like to thank the entire *Methods of Machine Learning* group for welcoming me during occasional visits. A special thanks also goes to Helen Donath and Sabine Gebert-Rudolph for not only sorting every administrative matter, but also for their ongoing care and support of the group. I would also like to thank the students I had the pleasure of supervising and working with during my PhD. In particular, I am grateful to Jennifer Müller and Samira Breitling for their contributions to this work. Furthermore, I would like to thank Wolfgang Fuhl, Cosimo Posth, and Fritz Götz for the engaging research collaborations.

A big thank you—and treat—goes to Findus, the most loyal office companion, for the emotional support he gave during challenging and happy times. Beyond him, there are many human beings who made my PhD journey not only enjoyable but truly special, and who helped turn Tübingen into a home: Thanks to Diego Agudelo España for taking over the alpha role when everyone else had left; to Eric Lacosse for his intriguing thoughts about 'the unknown'; to Luigi Gresele for the best pizza parties; and to Sabrina Klotz for always taking care. I'm grateful to all the Tübingen cyclists for exploring *the Länd*, to the *Meisterkleister* for good times in the choir, and to the *Strickeclub* for so much knitwear "handmade in Österberg". My thanks also go to everyone from the *Kurrende* choir for the music and special sense of community.

Special appreciation goes to Immanuel Schneeberger for his long-standing friendship, to Franziska Steeb for being the best friend of a friend, to Anna von Gagern for not being *dröge*, and to Lieven D'Hondt for the ongoing special connection and friendship.

Finally, I wish to thank my family and extended family simply for their existence and the grounding presence they provide. Thank you, Ulrich, for our shared enthusiasm for science, and thank you, Margrit, for your ever-welcoming warmth. Thank you, John, for sharing everyday life with me, for the calm you bring, your open mind, and for keeping me grounded—especially on weekends, when we leave the desk behind for real work in the fields. Thank you, Andrea, for being a uniquely different Zabel sister. Thank you, Papa, for reminding me to see things from different perspectives ("muss man auch mal so sehen"), and thank you, Mama, for being someone I can talk to about everything—and to you both for being such great parents.

Thank you!

Susanne Zabel
Tübingen, July 10, 2025

Abstract

The rapidly growing availability of extensive "omics" data—including genomics, transcriptomics, proteomics, and metagenomics—opens new perspectives for systems-oriented biological research. Concurrently, characteristic properties of these data, such as high dimensionality, measurement noise, and incomplete datasets, pose significant analytical challenges as they can obscure biological signals and complicate the application of statistical methods. Although machine learning provides effective methods for modeling complex relationships, its efficacy is often limited by these data challenges. Issues like the "curse of dimensionality", problems with model interpretability, and susceptibility to systematic errors further exacerbate this situation. The reliable identification of biologically relevant patterns therefore ranks among the central challenges of modern bioinformatics.

This dissertation addresses these challenges through the development and application of novel computational strategies based on the fundamental principle of targeted information refinement and reduction, aiming to enable robust biological conclusions from complex omics data. A particular focus lies on accounting for measurement uncertainties and missing values during the dimensionality reduction (DR) of high-dimensional data. Common DR methods, often primarily used for visualization, typically neglect uncertainties in the input data, even though biological measurements are frequently affected by technical noise or missing data. Acknowledging and propagating these uncertainties is therefore crucial for ensuring that conclusions drawn from low-dimensional representations, particularly visual interpretations of low-dimensional scatter plots, are robust and trustworthy.

First, this work presents extensions to established DR methods: VIPurPCA and an uncertainty-aware t-SNE framework facilitate the error propagation of measurement uncertainties through Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE), respectively. Both approaches rely on approximate Gaussian error propagation for computational efficiency; the necessary derivatives are computed using automatic differentiation. To handle the iterative nature of t-SNE, the implicit function theorem is additionally employed. Complementary intuitive visualizations, such as animated scatter plots, are developed to enable a direct assessment of the reliability of low-dimensional embeddings.

Beyond measurement noise, the complete absence of data—missing values—presents a distinct and often more severe challenge for analysis. A domain where sparse data is particularly pronounced is ancient genomics. Ancient DNA (aDNA) samples often exhibit low quality and quantity, resulting in significant data gaps and thus incomplete genotype information. However, the impact of these missing data points on the stability and reliability of the resulting PCA projection is often overlooked or not formally quantified. To address this, TrustPCA, a specialized probabilistic method and web tool, is introduced. TrustPCA quantifies and visualizes the uncertainties in PCA projections that specifically arise from such missing genotypes. By providing confidence ellipses for these PCA projections, TrustPCA directly enhances the conclusiveness and trustworthiness of population genetic analyses conducted on sparse aDNA.

Beyond individual data uncertainties, the overall structure and distribution of training data also significantly influence the performance of machine learning methods in general. This work addresses a previously underappreciated problem in the taxonomic classification of DNA sequences: despite a balanced distribution of classes in the training data, imbalances can occur within the feature space—i.e., densely and sparsely covered regions—which particularly impair the performance of classification models. To address this issue, a method for feature space balancing is proposed. This involves a targeted, strategic subsampling of the training data with the aim of achieving a more uniform distribution in the feature space. This approach significantly improves the generalization capability and classification performance of simple, resource-efficient ML models—in specific use cases, even surpassing more complex deep learning approaches.

Complementing such sequence classification, understanding the broader functional and evolutionary context of specific genes often requires dedicated analytical tools. Therefore, BLASTphylo is presented as an interactive web tool that automates the extensive pipeline from one of the most commonly used methods in

bioinformatics, BLAST, to the refined visualization of taxonomic distributions and phylogenetic relationships of bacterial homologous genes. This enables more efficient, accessible, and insightful comparative analyses.

In their entirety, the methods and tools developed herein demonstrate how the guiding principle of strategic information refinement and reduction can be effectively applied to overcome central bioinformatics challenges. By improving the trustworthiness, reproducibility, and interpretability of computational analyses, this dissertation contributes to more precise and insightful biological discoveries from diverse high-dimensional datasets.

Zusammenfassung

Die rasch wachsende Verfügbarkeit umfangreicher "Omics"-Daten—darunter Genomik, Transkriptomik, Proteomik und Metagenomik—eröffnet neue Perspektiven für eine systemorientierte biologische Forschung. Gleichzeitig stellen charakteristische Eigenschaften dieser Daten, wie hohe Dimensionalität, Messrauschen und unvollständige Datensätze, erhebliche analytische Herausforderungen dar, da sie biologische Signale überdecken und die Anwendung statistischer Methoden erschweren. Obwohl maschinelles Lernen effektive Methoden zur Modellierung komplexer Zusammenhänge bereitstellt, ist dessen Wirksamkeit durch die schwierige Datenlage oft begrenzt. Herausforderungen wie der „Fluch der Dimensionalität“, Probleme bei der Modellinterpretierbarkeit und die Anfälligkeit für systematische Fehler verschärfen diese Problematik zusätzlich. Die zuverlässige Identifikation biologisch relevanter Muster zählt daher zu den zentralen Herausforderungen der modernen Bioinformatik.

Diese Dissertation begegnet diesen Herausforderungen durch die Entwicklung und Anwendung neuer computergestützter Strategien, die auf dem Grundprinzip der gezielten Informationsverfeinerung und -reduktion basieren, um robuste biologische Schlussfolgerungen aus komplexen Omics-Daten zu ermöglichen. Ein besonderer Fokus liegt dabei auf der Berücksichtigung von Messunsicherheiten und fehlenden Messwerten bei der Dimensionsreduktion (DR) hochdimensionaler Daten. Gängige DR-Methoden, die oft primär der Visualisierung dienen, vernachlässigen in der Regel die Unsicherheiten in den Eingangsdaten, obwohl biologische Messwerte häufig durch technisches Rauschen oder Datenlücken beeinträchtigt sind. Die Erwägung und Fortpflanzung dieser Unsicherheiten ist daher entscheidend, um sicherzustellen, dass Schlussfolgerungen aus niedrigdimensionalen Repräsentationen, insbesondere visuellen Interpretationen von PCA-Diagrammen, robust und vertrauenswürdig sind.

Zunächst präsentiert diese Arbeit Erweiterungen etablierter DR-Methoden: VIPurPCA und ein unsicherheitsbewusstes t-SNE-Framework ermöglichen die Fehlerfortpflanzung von Messunsicherheiten durch die Hauptkomponentenanalyse (*Principal Component Analysis*, PCA) bzw. *t-Distributed Stochastic Neighbor Embedding* (t-SNE). Beide Ansätze beruhen auf approximativer Gaußscher Fehlerfortpflanzung für eine effiziente Berechnung; die erforderlichen Ableitungen werden mittels automatischer Differenzierung ermittelt, wobei für t-SNE zur Handhabung seiner iterativen Natur zusätzlich der Satz über implizite Funktionen angewandt wird. Ergänzend werden intuitive Visualisierungen, etwa animierte Streudiagramme, entwickelt, die eine Bewertung der Zuverlässigkeit niedrigdimensionaler Einbettungen ermöglichen.

Über Messrauschen hinaus stellt das vollständige Fehlen von Daten—also fehlende Messwerte—eine eigenständige und oft gravierendere Herausforderung in der Datenanalyse dar. Besonders ausgeprägt ist dieses Problem in der genomischen Analyse antiker DNA (aDNA), bei der die Qualität und Quantität der extrahierten DNA häufig stark eingeschränkt ist. Dies führt zu erheblichen Datenlücken und unvollständigen Genotypinformationen. Die Auswirkungen dieser fehlenden Daten auf die Stabilität und Aussagekraft der resultierenden PCA-Projektionen werden jedoch bislang selten systematisch erfasst oder quantifiziert. Zur Schließung dieser Lücke wird TrustPCA vorgestellt, eine spezialisierte probabilistische Methode sowie ein Web-Tool zur Quantifizierung und Visualisierung der durch fehlende Genotypen bedingten Unsicherheiten in PCA-Projektionen. Durch die Bereitstellung von Konfidenzellipsen für die projizierten Positionen im PCA-Raum erhöht TrustPCA unmittelbar die Aussagekraft und Vertrauenswürdigkeit populationsgenetischer Analysen auf Basis unvollständiger aDNA-Daten.

Über individuelle Datenunsicherheiten hinaus beeinflussen auch die Gesamtstruktur und die Verteilung der Trainingsdaten maßgeblich die Leistungsfähigkeit maschineller Lernverfahren. In dieser Arbeit wird ein bislang wenig beachtetes Problem bei der taxonomischen Klassifikation von DNA-Sequenzen adressiert: Trotz einer ausgewogenen Verteilung der Klassen in den Trainingsdaten können Unausgewogenheiten im Merkmalsraum auftreten—also dicht und spärlich abgedeckte Regionen—, die insbesondere die Leistung einfacher Klassifikationsmodelle stark beeinträchtigen. Zur Adressierung dieses Problems wird ein Verfahren zum Ausgleich des Merkmalsraums vorgeschlagen. Dabei erfolgt ein gezieltes, strategisches *Subsampling* der Trainingsdaten mit dem Ziel, eine gleichmäßigere Verteilung im Merkmalsraum herzustellen. Durch

diesen Ansatz wird die Generalisierbarkeit und Klassifikationsleistung einfacher, ressourcenschonender ML-Modelle deutlich verbessert—bis hin zur Überlegenheit gegenüber komplexeren *Deep-Learning*-Ansätzen in spezifischen Anwendungsfällen.

Über die reine Sequenzklassifikation hinaus erfordert das Verständnis des funktionellen und evolutionären Kontexts spezifischer Gene häufig den Einsatz spezialisierter Analysewerkzeuge. Abschließend wird daher BLASTphylo als interaktives Web-Tool vorgestellt. Es automatisiert die umfangreiche Pipeline von Homologiesuchen mittels BLAST—eine der gängigsten Anwendungen der Bioinformatik—bis hin zur verfeinerten, interaktiven Visualisierung taxonomischer Verteilungen und phylogenetischer Beziehungen, insbesondere für bakterielle homologe Gene. Diese Automatisierungs- und Visualisierungsstrategie ermöglicht effizientere, zugänglichere und aufschlussreichere vergleichende Genomanalysen.

In ihrer Gesamtheit demonstrieren die hier entwickelten Methoden und Softwareprogramme, wie das Leitmotiv der strategischen Informationsverfeinerung und -reduktion effektiv zur Bewältigung zentraler bioinformatischer Herausforderungen eingesetzt werden kann. Durch die Verbesserung der Vertrauenswürdigkeit, Reproduzierbarkeit und Interpretierbarkeit rechnergestützter Analysen leistet diese Dissertation einen Beitrag zu präziseren und aufschlussreicheren biologischen Erkenntnissen aus vielfältigen hochdimensionalen Datensätzen.

Table of Contents

List of Figures

List of Tables

Notation

Mathematical Objects and Operations

a	A scalar (lowercase)
$\boldsymbol{a}$	A column vectors (bold lowercase)
$\boldsymbol{A}$	A matrix (bold uppercase)
$\boldsymbol{a}_i$ or $[\boldsymbol{a}]_i$	The i-th entry (element) of vector $\boldsymbol{a}$
$\boldsymbol{A}_{ij}$ or $[\boldsymbol{A}]_{ij}$	The (i, j)-th entry of matrix $\boldsymbol{A}$ (row i, column j)
$\boldsymbol{A}_{i,:}$ (or $\boldsymbol{A}_{:,j}$)	The i-th row (or j-th column) of matrix $\boldsymbol{A}$
$\boldsymbol{I}$ or $\boldsymbol{I}_D$	Identity matrix (of size $D \times D$ if specified)
$\boldsymbol{0}$	Vector of all zeros
$\boldsymbol{A}^\top, \boldsymbol{a}^\top$	Transpose of a matrix $\boldsymbol{A}$ or vector $\boldsymbol{a}$
$\boldsymbol{A}^{-1}$	Inverse of a square matrix $\boldsymbol{A}$
$\boldsymbol{A}^\dagger$	Moore-Penrose pseudoinverse of matrix $\boldsymbol{A}$
$\text{vec}(\boldsymbol{A})$	Vectorization of matrix $\boldsymbol{A}$ by stacking its columns/rows
$\text{vec}^{-1}(\boldsymbol{a})$	Inverse vectorization, reshaping vector $\boldsymbol{a}$ into a matrix
$\text{diag}(\boldsymbol{a})$	Square matrix with vector $\boldsymbol{a}$ on its main diagonal
$\text{diag}(\boldsymbol{A})$	Vector containing the main diagonal elements of matrix $\boldsymbol{A}$
$\text{Tr}(\boldsymbol{A})$	Trace of a square matrix $\boldsymbol{A}$
$\|\|\boldsymbol{a}\|\|$ or $\|\|\boldsymbol{a}\|\|_2$	Euclidean norm (or L_2 norm) of vector $\boldsymbol{a}$
$\|\|\boldsymbol{A}\|\|_F$	Frobenius norm of matrix $\boldsymbol{A}$
$\boldsymbol{AB}$	Matrix multiplication
$\boldsymbol{a} \cdot \boldsymbol{b}$ or $\boldsymbol{a}^\top \boldsymbol{b}$	Dot product (inner product) of vectors $\boldsymbol{a}, \boldsymbol{b}$
$\boldsymbol{A} \otimes \boldsymbol{B}$	Kronecker product of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$

Derivatives and Optimization

$\nabla_{\boldsymbol{a}} f(\boldsymbol{a})$	Gradient of a scalar function f with respect to vector $\boldsymbol{a}$
$\boldsymbol{H}_{\boldsymbol{aa}} f(\boldsymbol{a})$ or $\nabla^2_{\boldsymbol{a}} f(\boldsymbol{a})$	Hessian matrix of scalar function f with respect to vector $\boldsymbol{a}$
$\boldsymbol{J}_{\boldsymbol{a}} \boldsymbol{f}(\boldsymbol{a})$ or $\boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{a})$	Jacobian matrix of vector function $\boldsymbol{f}$ with respect to vector $\boldsymbol{a}$
$\boldsymbol{J}_{\boldsymbol{ab}} \boldsymbol{f}(\boldsymbol{a}, \boldsymbol{b})$	Jacobian matrix of mixed partial derivatives, $\frac{\partial^2 f}{\partial \boldsymbol{a} \partial \boldsymbol{b}}$
$\boldsymbol{J}_{\boldsymbol{b}} \boldsymbol{f}(\boldsymbol{a}, \boldsymbol{b})$	Jacobian matrix of $\boldsymbol{f}$ w.r.t. $\boldsymbol{b}$, holding $\boldsymbol{a}$ constant
$\boldsymbol{J}_{\boldsymbol{a}} \boldsymbol{f}(\boldsymbol{a}_0)$ or $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{a}}\Big\|_{\boldsymbol{a}_0}$	Jacobian matrix of vector function $\boldsymbol{f}$ with respect to vector $\boldsymbol{a}$, evaluated at $\boldsymbol{a}_0$
$\boldsymbol{J}_{\boldsymbol{a}} \boldsymbol{f}(\boldsymbol{a}_0, \boldsymbol{b}_0)$ or $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{a}}\Big\|_{(\boldsymbol{a}_0, \boldsymbol{b}_0)}$	Jacobian matrix of $\boldsymbol{f}$ w.r.t. $\boldsymbol{a}$, holding $\boldsymbol{b}$ constant, evaluated at $(\boldsymbol{a}_0, \boldsymbol{b}_0)$
$\arg\min_{\boldsymbol{a}} f(\boldsymbol{a})$	Argument $\boldsymbol{a}$ that minimizes function $f(\boldsymbol{a})$, often denoted as $\boldsymbol{a}^*$
$C(\cdot)$	General cost or objective function

Probability and Statistics

$p(x)$	Probability density function (PDF) of a continuous random variable X
$p(x \mid y)$	Conditional PDF of X given $Y = y$
$\mathbb{E}[X]$ or $\mathbb{E}_{\boldsymbol{a}}[\cdot]$	Expectation of random variable X or w.r.t. distribution of $\boldsymbol{a}$
$\text{Var}[X]$ or $\mathbb{V}[X]$	Variance of random variable X

$\mathrm{Cov}[X, Y]$	Covariance between random variables X and Y
$\mathcal{N}(\boldsymbol{a}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	Multivariate Gaussian (Normal) distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$
$\delta(\cdot)$	Dirac delta function
$D_{\mathrm{KL}}(\boldsymbol{P} \parallel \boldsymbol{Q})$	Kullback-Leibler divergence between distributions $\boldsymbol{P}$ and $\boldsymbol{Q}$
σ^2	Variance
p_{ij}	Joint probability (similarity) between items i, j in high-dim (t-SNE)
q_{ij}	Joint probability (similarity) between items i, j in low-dim (t-SNE)
$\mathrm{TP}_i, \mathrm{FP}_i$	True Positives, False Positives for class i

Acronyms & Abbreviations

AD	Automatic Differentiation
API	Application Programming Interface
BGC	Biosynthetic Gene Cluster
BLAST	Basic Local Alignment Search Tool
CA	Carbonic Anhydrase
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DL	Deep Learning
DNA	Deoxyribonucleic Acid
DNN	Deep Neural Network
DR	Dimensionality Reduction
FFT	Fast Fourier Transform
FIt-SNE	FFT-accelerated Interpolation-based t-SNE
GPU	Graphics Processing Unit
GTDB	Genome Taxonomy Database
HGT	Horizontal Gene Transfer
HMC	Hamiltonian Monte Carlo
HVP	Hessian-Vector Product
IFT	Implicit Function Theorem
i.i.d.	Independent and Identically Distributed
JAX	Just After eXecution (Python library)
JVP	Jacobian-Vector Product
KL	Kullback-Leibler (divergence)
KNN	k-Nearest Neighbors
LCA	Lowest Common Ancestor
LSTM	Long Short-Term Memory
MAP	Macro Average Precision (for classification)
MC	Monte Carlo
MCMC	Markov Chain Monte Carlo
ML	Machine Learning
NCBI	National Center for Biotechnology Information
NGS	Next-Generation Sequencing
NLP	Natural Language Processing
PCA	Principal Component Analysis
PC	Principal Component
PDF	Probability Density Function
PPCA	Probabilistic Principal Component Analysis
RefSeq	Reference Sequence Database
RNA	Ribonucleic Acid

RNN	Recurrent Neural Network
SNE	Stochastic Neighbor Embedding
SPD	Symmetric Positive Definite (matrix)
SVD	Singular Value Decomposition
SVM	Support Vector Machine
taxID	Taxonomic Identifier
TrustPCA	Tool for Reliability and Uncertainty in SmartPCA
t-SNE	t-Distributed Stochastic Neighbor Embedding
TPU	Tensor Processing Unit
UCI	University of California, Irvine (Machine Learning Repository)
VIPurPCA	Visualizing and Propagating Uncertainty in Principal Component Analysis
VJP	Vector-Jacobian Product
XLA	Accelerated Linear Algebra

1. Introduction

The advent of low-cost, high-throughput technologies, the ease of publishing data, and increasing computational power, have brought the field of bioinformatics into the *big data* era [73, 202]. Central to this transformation are the so-called *omics* disciplines, including genomics, transcriptomics, and proteomics, which enable the simultaneous and comprehensive study of entire sets of biomolecules, such as DNA, RNA, or proteins[1].

Since the breakthrough in DNA sequencing by Fredrick Sanger in 1977 [242], sequencing methods have evolved into massively parallel technologies, known as next- (or second-)generation sequencing (NGS)[2] [79]. NGS enables the efficient determination of nucleotide sequences for DNA and RNA molecules, from individual genes to entire genomes, and provides the basis for modern genomics, transcriptomics, and population genetics. For example, since 2002, 3.7 Billion[3] whole genome shotgun sequences have been deposited to GenBank [22] from all domains of life, a number that has risen exponentially driven by NGS innovations.

The advent of NGS and the resulting explosion of sequencing data have expanded the scope of questions that can be addressed in genomics and transcriptomics. By enabling the comprehensive (and simultaneous) assessment of different classes of biomolecules in individuals and groups of organisms, these technologies have transformed our understanding of the composition, regulation, and evolution of living organisms. The publication of the human genome sequence in 2001 [44, 278]—achieved using Sanger sequencing—laid the foundation for gaining insight into the structure of the human genome, the biological functions encoded in the human genome, genetic variation among individuals, the identification of disease-associated genetic features, and the evolution and history of the human species [138]. Subsequent advances, such as single-cell RNA sequencing, now allow for the profiling of millions of individual cells, capturing molecular heterogeneity across diverse cell states, types, and developmental or disease stages [238]. Large-scale initiatives like the Human Cell Atlas initiative [231] aim to create comprehensive, openly accessible datasets that offer a detailed map of the human body, with far-reaching implications for diagnostics and drug development [238]. Beyond these applications in understanding contemporary human biology and disease, NGS technologies have also facilitated studying human history; specifically, analyzing ancient DNA (aDNA) has revolutionized population genetics by offering direct glimpses into past genetic diversity, migration patterns, and admixture events, allowing researchers to reconstruct complex demographic histories [191, 220, 289].

Beyond human biology, genomics and transcriptomics have also significantly advanced our understanding of microbial diversity and function, as well as their relevance for humans and entire ecosystems. For example, the genome taxonomy database (GTDB) [204] currently comprises nearly 600,000 genomes from over 100,000 bacterial species[4]. The availability and analysis of numerous microbial genome sequences has led to several

1: In biology, the suffix *-ome* denotes the totality of a specific class of biomolecules within an organism or system, e.g., genome, transcriptome or proteome.

2: More recently, third-generation sequencing technologies, characterized by longer read lenghts, have further expanded capabilities. [261].

3: Statistics are available at https://www.ncbi.nlm.nih.gov/genbank/statistics/, last accessed May 30, 2025.

4: Statistics are available at https://gtdb.ecogenomic.org/stats/r220, last accessed May 30, 2025

discoveries related to human health, such as identifying new antimicrobial drug targets [100] and detecting novel biosynthetic gene clusters for antimicrobial natural products [9], both crucial in combating increasing antibiotic resistance. The COVID-19 pandemic further underscored the importance of efficient sequencing and global sharing[5] of viral genomes, which facilitated rapid research and vaccine development [98]. In this context, phylogenies—a description of the evolutionary relationships among a group of organisms—were the basis for identifying, defining, and monitoring various viral variants of concern during the pandemic [12]. These approaches also extend to reconstructing the evolutionary history of all known species, an idea first formulated by Charles Darwin in 1859 [49], and have led to modern hierarchical classification systems and formal nomenclatures of all organisms (taxonomies) based on genotypes rather than phenotypes [99].

5: For example, Nextstrain has maintained an up-to-date analysis of SARS-CoV-2 at https://nextstrain.org/sars-cov-2 since January 20, 2020.

The classification of prokaryotes, in particular, has greatly benefited from genetic approaches due to the limitations of phenotype-based methods, the immense genetic diversity of prokaryotes, the impact of horizontal gene transfer, and the challenges in culturing many bacterial species [99]. Genotype-based classification facilitates the characterization and identification of microbial sequences and genomes that are rapidly populating databases. A significant source for this data is environmental DNA derived from naturally occurring microbial populations, studied through *metagenomics*. While 16S rRNA amplicon sequencing[6] has long been used for assessing bacterial diversity [43, 139, 197], its limited taxonomic resolution and inability to provide functional insights have led to the rise of genome-resolved metagenomics. This approach reconstructs microbial genomes from whole-metagenome sequencing data [123, 203], enabling inference of both taxonomic composition and functional potential (gene and pathway abundance) of microbiomes [262]. This, in turn, deepens our understanding of phenomena such as the human microbiome's role in health and disease [171] and the planet's overall microbial diversity.

6: 16S amplicon sequencing targets the 16S ribosomal RNA gene, a phylogenetic marker.

1.1 Mathematical Models for Biological Data

The comprehensive analysis of entire biological systems, enabled by omics technologies, necessitates rigorous mathematical frameworks for data representation and modeling. Biological data are typically conceptualized as observations (samples or objects), each described by a set of variables or features. A *variable* or *feature* is any measurable characteristic of an object, such as genomic variants, gene expression levels, protein abundances, or clinical attributes. The number of variables D defines the *dimensionality* of the data. These D features are usually measured across N observed samples drawn from a population of interest (e.g., patients, cells, or environmental samples).

Two primary goals in bioinformatics and medical informatics are inference and prediction. *Inference* aims to understand underlying biological mechanisms by fitting a mathematical model to the data, allowing for hypothesis testing about how a system behaves [36]. For example, one might infer ancestral relationships between populations based on genomic data. *Prediction*, in contrast, focuses on forecasting unobserved

outcomes or future events [36], such as an individual's predisposition to a disease based on genetic markers.

In principle, both *statistics* and *machine learning* (ML) provide tools for these goals. Historically, statistics has been preferred for inference, while ML has been more commonly applied to prediction tasks. Statistical inference often involves fitting explicit probability models, which enables the quantification of confidence in observed effects. These methods were traditionally designed for scenarios with relatively few variables. ML prediction methods, on the other hand, often make fewer assumptions about the data-generating process and can perform exceptionally well in high-dimensional settings where $D \gg N$ [86]. In recent years, ML has become a powerful tool for analyzing complex biological data [81], adept at detecting patterns in large datasets [59, 107]. However, the lack of an explicit model in many ML approaches can make biological interpretation challenging [36].

ML prediction methods are considered *supervised learning algorithms*[7], which learn a mapping $f(\boldsymbol{x})$ from input features $\boldsymbol{x}$ to an output $\boldsymbol{y}$ (or a probability distribution over $\boldsymbol{y}$) [110]. This encompasses *classification* tasks (qualitative $\boldsymbol{y}$) and *regression* tasks (quantitative $\boldsymbol{y}$) [110]. Examples include linear regression, support vector machines, decision trees, neural networks, and Bayesian classifiers [86]. Their performance often depends on abundant, high-quality labeled training data but can be affected by biases in data or model limitations [280]. Another major class of ML algorithms is *unsupervised learning*, which deals with unlabeled data, aiming to discover inherent structure. For instance, if data are assumed to lie on a low-dimensional manifold, methods like Principal Component Analysis (PCA), manifold learning, or autoencoders can be used for *dimensionality reduction* [110]. *Clustering* is another unsupervised task that partitions data based on similarity.

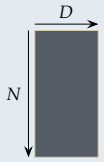
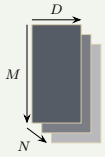
7: ML algorithms can be broadly categorized into four types: supervised, unsupervised, semi-supervised, and reinforcement learning.

Statistical inference[8], itself follows different paradigms, primarily *frequentist* and *Bayesian* inference. Frequentists treat model parameters as fixed and unknown constants and interpret probability as the long-run frequency of events. Bayesian statistics, on the other hand, treats parameters as random variables with associated probability distributions (degrees of belief) [225]. A *prior* belief about a parameter is updated with observed data (via a *likelihood*) to form a *posterior* distribution, providing a natural framework for quantifying *uncertainty* [225].

8: Statistical inference infers properties of whole populations from a limited set of observations, and goes beyond *descriptive statistics*, which is solely concerned with properties of the observed data [132].

Both, statistical and ML methods, offer inference and prediction methods of varying complexity. The simplest form of data analysis involves *univariate* methods, which consider a single variable at a time, even when multiple variables are observed (Table 1.1, first column). More complex tasks—such as predicting a variable from multiple independent variables, or estimating the relationship between a set of dependent variables—require the use of *multivariate* methods (Table 1.1, second column). These methods involve analyzing multiple variables simultaneously and are commonly used in bioinformatic analyses. *Matrix-variate* statistics (Table 1.1, third column) extend multivariate approaches to matrix-valued data, such as spatial, time-series or image data. While multivariate and matrix-variate approaches offer richer analytical capabilities, they also introduce greater complexity.

Table 1.1: Typical data formats in bioinformatics. Univariate data involves a single variable a. Multiple observations (N) are stored in a vector $\boldsymbol{x}$. Multivariate data consists of multiple (D) variables a_i and is represented as a matrix $\boldsymbol{X}$, where rows represent observations and columns represent variables. Matrix-variate data deals with matrix-valued observations, which, when multiple independent identically distributed (i.i.d.) observations are considered, result in tensors.

	Univariate	Multivariate	Matrix-variate
One sample	$x_i = a$ $\in \mathbb{R}$	$\boldsymbol{x}_i = [a_1, \ldots, a_D]^\intercal$ $\in \mathbb{R}^D$	$\boldsymbol{X}_i = \begin{bmatrix} a_{11} & \ldots & a_{1D} \\ \vdots & \ddots & \vdots \\ a_{M1} & \ldots & a_{MD} \end{bmatrix}$ $\in \mathbb{R}^{M \times D}$
N i.i.d. samples	$\boldsymbol{x} = [x_1 \ldots, x_N]^\intercal$  $\in \mathbb{R}^N$	$\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^\intercal$  $\in \mathbb{R}^{N \times D}$	$\mathbf{X} = [\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N]^\intercal$  $\in \mathbb{R}^{N \times M \times D}$

1.2 Challenges of Biological Data Analysis

As highlighted previously, the exponential growth in the volume (N) and scale (e.g., dimensionality, D) of biological data presents both immense opportunities for advancing our understanding of complex biological systems and significant analytical hurdles. Effectively exploiting this data requires addressing challenges related to data management, computational resources, algorithmic scalability, and, crucially, the interpretability of both the data and the models applied.

Technically, managing big data involves robust strategies for storage, curation, and dissemination, ensuring quality and accessibility often through standardized protocols and integrated platforms [107, 134, 202]. Computationally, complex models applied to large datasets demand significant resources, including specialized hardware and high-performance computing environments [202].

From a computational and algorithmic perspective, methods must be scalable with respect to both, the number of features (D) and the number of samples (N) to remain efficient. For instance, algorithms that used to be considered efficient—such as those with a quadratic runtime—may now be too slow in the era of big data [265]. Not only the efficiency but also the effectiveness of inference and prediction models can be compromised when analyzing data with a large number of features D. When analyzing high-dimensional data, a phenomenon known as the *curse of dimensionality* arises, which manifests itself in a variety of ways[9] [21, 26]. Feature selection and dimensionality reduction are common strategies to mitigate this by identifying relevant features or creating lower-dimensional representations.

9: The term describes multiple phenomenons about high-dimensional data. For more information, see Chapter 2.

Another omnipresent issue is the imperfection of biological data. Beyond inherent biological stochasticity, experimental measurements are subject to noise and technical artifacts. Furthermore, data can be incomplete due to missing values or features, a particularly acute problem in fields like ancient genomics where DNA is often degraded and yields sparse genotype information [92]. Ignoring such measurement uncertainty or data missingness can lead to biased analyses and unreliable conclusions.

A related challenge in biomedical data analysis is class imbalance, where certain categories (e.g, healthy controls or model organisms) are over-

represented, potentially biasing models [7]. This imbalance can lead to poor model performance, particularly in the underrepresented classes. To address this, techniques such as under-sampling the majority class or over-sampling the minority class are commonly used [147]. However, as explored in this thesis (Chapter 11), imbalances can also occur within the feature space itself, even with balanced classes, affecting model generalization.

The technical, computational, and algorithmic challenges of high-dimensional data analysis are exacerbated by the difficulty of interpreting and explaining[10] both the data and the models applied. Ensuring interpretability of analytical results, especially when using complex "black box" models[11] like deep neural networks is often non-trivial [59, 60, 215, 272]. Moreover, some models produce probabilistic rather than deterministic outputs, which requires a solid understanding of statistical inference and uncertainty quantification. When model outputs inform critical decisions (e.g., in healthcare, criminal justice, or finance), accountability and transparency are essential [240]. A significant amount of research has focused on *explainable ML*, where a second (post hoc) model is created to explain the first black box model [111]. Besides that, there is a growing focus on developing models that are inherently interpretable, offering transparency and insight into their decision-making processes from the outset [240].

10: The terms *interpretability* and *explainability* are not universally defined and their meanings can vary depending on the field of application [277].

11: The reasoning behind the prediction of "black box" models often remains opaque.

Various approaches have been developed to enhance the interpretability of data and models. In general, visualization is a central tool for data analysis that facilitates inductive reasoning in a natural manner [277]. The concept of *visual analytics* involves presenting information visually in a comprehensible format, enabling the discovery of insights, the drawing of conclusions, and the making of more informed predictions [120]. During the analysis of large-scale (raw) data, visualizations can aid in communicate findings, detecting outliers and anomalous data, understanding the underlying data distribution, and guiding subsequent analysis steps such as normalization and model selection [277].

The visualization of high-dimensional data and probability distributions is non-trivial. Therefore, dimensionality reduction is often a prerequisite for visualizing multivariate data reducing the dimensionality to two (or three). The low-dimensional features extracted through these techniques preserve different aspects of the original high-dimensional data, depending on the method used. Both, the representations themselves and their visualization influence subsequent analyses, such as outlier detection, clustering, and hypothesis generation. It is therefore crucial to assess the reliability of these representations to avoid drawing misleading conclusions, especially when dealing with noisy or incomplete input data [166].

1.3 Contributions

This dissertation contributes novel computational methods that enable more interpretable, reliable, and efficient analysis of large-scale biological data. A central theme uniting these contributions is the principle of strategic information refinement. This principle is explored across three primary areas, each addressing distinct challenges in bioinformatics: (1) quantifying and visualizing uncertainty in dimensionality reduction, (2) improving the robustness of machine learning models for taxonomic classification, and (3) developing an interactive visual analytics tool for exploring complex biological datasets.

Biological data often carries inherent uncertainty, whether from experimental measurements or upstream computational modeling [93]. Standard dimensionality reduction (DR) techniques typically disregard this uncertainty, potentially yielding misleading low-dimensional representations. To addresses this gap, this thesis first introduces **VIPurPCA (Visualizing and Propagating Uncertainty in Principal Component Analysis)**. This method propagates input Gaussian uncertainty through PCA using automatic differentiation. It provides a closed-form approximation for the covariance of the resulting principal components and introduces an intuitive animation-based technique to visualize the stability of the PCA embedding, drawing on existing concepts for uncertainty visualizations of probability distributions [285]. Recognizing the need for similar uncertainty awareness in non-linear DR, we then extend this framework by developing a method for **Uncertainty Propagation through t-Distributed Stochastic Neighbor Embedding (t-SNE)**. This work leverages the implicit function theorem to handle the iterative optimization nature of t-SNE, allowing for the computation of the embedding's sensitivity to input perturbations. Approximate Gaussian error propagation is then used to quantify the uncertainty in the low-dimensional t-SNE map.

Furthermore, recognizing that missing data is a profound source of uncertainty in specific domains, we developed **TrustPCA, A Probabilistic Approach to Visualize the Effect of Missing Data on PCA in Ancient Human Genomics**. TrustPCA adapts error propagation principles to the projection method used for sparse ancient data to quantify and visualize projection uncertainties specifically caused by missing genotypes in ancient human DNA samples, and thereby refining the interpretation of population genetic structures.

The resulting methods for uncertainty quantification and visualization can assist scientists in assessing the reliability of low-dimensional embeddings by addressing key questions, such as

- **(Q1)** Are there specific data points that have an outsized effect on the reducing map?
- **(Q2)** Which aspects of the low-dimensional embedding are sensitive to even minor perturbations of the data?
- **(Q3)** And specifically for analyses involving sparse ancient DNA: What is the confidence region for an ancient sample's position in a PCA-derived genetic map, considering the impact of its missing genotype data?

thereby facilitating robust interpretations.

Beyond enhancing interpretability via uncertainty-aware dimensionality reduction, this thesis explores another facet of information refinement: improving machine learning model robustness through targeted data subsampling, specifically in the context of taxonomic classification. To characterize the taxonomic profile of metagenomic samples, the accurate and efficient taxonomic classification of DNA sequences is crucial. While deep learning models have shown good performance in this task, they can be resource-intensive and lack interpretability. This thesis contributes a novel approach for **Improving Taxonomic Classification with Feature Space Balancing**. We demonstrate that simple k-mer based features, when combined with a novel training data balancing strategy that promotes uniformity in the feature space (distinct from class balancing), can significantly enhance the performance and generalization of simpler, more resource-efficient classifiers (e.g., bagged decision trees, subspace KNNs). This approach outperforms state-of-the-art deep learning methods on challenging classification tasks, particularly for novel organisms.

The final contribution applies the principle of information refinement to visual analytics, facilitating biological discovery from large databases. Interpreting complex biological datasets, such as those arising from homology searches, often requires tools that can effectively manage information overload. This thesis introduces **BLASTphylo, an Interactive Web Tool for Taxonomic and Phylogenetic Analysis of Genes**. This tool is designed to infer and visualize the distribution of a protein across a taxonomy, based on a list of putative homologs identified by BLAST. BLASTphylo automates the reduction of raw, extensive BLAST hit lists by, for instance, retaining only the best-scoring homolog per species. Furthermore, it facilitates the exploration of protein occurrence and evolutionary histories through interactive tree visualizations that allow users to dynamically reduce visual complexity by collapsing clades or focusing on specific taxonomic ranks. This refinement of displayed information makes large-scale bioinformatic analyses more efficient, manageable, and accessible.

Collectively, these contributions aim to provide bioinformaticians with more robust, interpretable, and accessible methods for analyzing the increasingly complex and uncertain data characteristic of modern biological research.

1.4 Outline

This thesis consists of four main parts: Part I introduces mathematical concepts and provides the biological background for this thesis. Part II and Part III comprise the achieved scientific contributions. Part IV summarizes and discusses the impact of the achieved scientific contributions and discusses future directions. The main part is complemented by an appendix (Part V).

Part I provides the foundational background for this thesis. Chapter 2 explores the motivation for using dimensionality reduction (DR), focusing on a key challenge inherent to high-dimensional data—the curse of dimensionality. Furthermore, the mathematical concepts behind two dimensionality reduction methods are introduced in detail:

- **Principal Component Analysis (PCA):** Section 2.1 presents the linear DR technique PCA [95, 209], which computes a low-dimensional subspace where the data is embedded. This section demonstrates how the optimal projection is derived by maximizing the variance of the projected data. Additionally, different methods for computing the axes (principal components) for the optimal projection from the high-dimensional data matrix are discussed. This section concludes with different visualization strategies of PCA embeddings for visual analytics.
- **t-distributed stochastic neighbor embedding (t-SNE):** Section 2.2 introduces the non-linear DR technique t-SNE [161], which iteratively optimizes the coordinates of the low-dimensional data representation to preserve neighborhoods between individual samples. This section begins by introducing probability distributions that model the neighborhoods between two samples in high- and low-dimensional spaces. It then defines the optimization problem and discusses the specifics of the gradient descent algorithm used for optimization.

Chapter 3 introduces key concepts of Bayesian inference, including the prior, likelihood, and posterior distributions, and their relation as defined by Bayes' theorem. Central to this chapter are Gaussian distributions and their properties (Section 3.1.1) since this distribution type makes Bayesian inference computationally very practical. We demonstrate how Gaussian distributions are closed under marginalization, conditioning, and affine transformations. Section 3.1.3 extends these principles to matrix-variate normal distributions, showing how Gaussian inference can be applied in this broader context. Section 3.2 discusses approximations for inference problems where closed-form solutions are intractable. Specifically, when Gaussian distributions are passed through non-linear functions, Taylor approximations can be used to approximate the posterior, involving the computation of derivatives (Section 3.2.1). Monte Carlo methods are introduced as a second method for approximate inference, relying on random sampling (Section 3.2.2).

Chapter 4 introduces automatic differentiation (AD) as a technique for evaluating partial derivatives, which is essential for approximate inference using Taylor series. The chapter explains how AD utilizes the chain rule (Section 4.2) repeatedly to derive the derivative of a composite function represented by a computation graph, illustrated through an example (Section 4.1). Section 4.3 further discusses two specific approaches to AD: forward mode and reverse mode, detailing the scenarios in which each mode is preferred. Furthermore, this section introduces the computation of higher-order derivatives using AD. Lastly, we introduce JAX in Section 4.4, an ML framework that offers an efficient implementation of AD.

Chapter 5 introduces implicit functions, implicit differentiation, and the implicit function theorem, which provides conditions under which an implicitly defined function can be locally expressed as a differentiable explicit function.

Chapter 6 introduces different types and sources of uncertainty, and presents an overview of visualization strategies for both, summary statistics and distributions.

In Chapter 7, classification—the grouping of objects into classes—is discussed from two perspectives. First, concepts and methods for the classification of organisms are discussed in Section 7.1. It defines key terms of evolutionary biology, such as taxonomy, homology and phylogeny, and outlines the steps of a phylogenetic analysis pipeline resulting in a phylogenetic tree. Visualizations of the resulting tree are also presented. Second, Section 7.2 introduces classification as a machine learning method, and provides mathematical details of different classifiers, including k-nearest neighbors classifiers, support vector machines, neural networks, and decision trees.

Part II addresses this manuscript's goal to quantify uncertainties using approximate Gaussian error propagation through various dimensionality reduction techniques. It also introduces a novel approach for visualizing these quantified uncertainties, specifically through the visualization of an uncertain two-dimensional map. Several 'omics'-datasets are used as case studies to demonstrate the importance of uncertainty quantification in this field.

Chapter 8 presents Gaussian error propagation through principal component analysis. Since PCA involves non-linearity when computing the eigenvectors, a first-order Taylor approximation is applied to linearize the function around the input mean. This approach allows for the derivation of closed-form solutions for the mean and covariance matrix of the eigenvector matrix. The resulting covariance matrix captures the uncertainty of the PCA outcomes. The computation of this covariance matrix involves Jacobian matrices, which consist of first-order partial derivatives of the PCA outcome (eigenvectors) with respect to the input data. To efficiently compute these Jacobians, we leverage the automatic differentiation capabilities of the ML framework JAX.

Chapter 8: VIPurPCA—Gaussian error propagation through PCA.

Github repository:
`Integrative-Transcriptomics/VIPurPCA`

Given the high dimensionality of the input data, explicitly computing and storing the large Jacobian matrices may exceed the memory capacity of standard compute machines. To address this, we provide an efficient implementation that computes the covariance matrix using Jacobian-vector products and vector-Jacobian products, thereby avoiding the explicit computation of large Jacobians. We compare our estimates of mean and covariance with those derived from Monte Carlo sampling, analyzing their accuracy relative to the number of Monte Carlo samples.

Finally, having inferred a Gaussian distribution of the PCA output (eigenvectors), we explore potential visualization strategies to communicate the spread of the low-dimensional representations implicitly through samples.

Chapter 9 presents Gaussian error propagation through t-distributed stochastic neighbor embedding. We demonstrate how Gaussian error propagation, using a first-order Taylor approximation, can be applied to t-SNE, similar to the approach used for PCA, to obtain estimates for the mean and covariance matrix of the t-SNE embedding. Unlike PCA, which relies solely on linear algebra routines, t-SNE produces its two-dimensional embedding through an iterative optimization algorithm. This iterative process poses challenges for standard automatic differentiation, as differentiating through all unrolled iterations of the optimization procedure is infeasible for Jacobian computation. To address this, we show how the implicit function theorem can be utilized to

Chapter 9: Gaussian error propagation through t-SNE.

Github repository:
`Integrative-Transcriptomics/tsne`

differentiate the optimization process, making the computation of the Jacobian feasible.

Chapter 10: TrustPCA: A Probabilistic Approach to Visualize the Effect of Missing Data on PCA in Ancient Human Genomics.

Available via trustpca-tuevis.cs.uni-tuebingen.de

Chapter 10 addresses the challenge of interpreting PCA results for ancient DNA samples, which often suffer from extensive missing genotype data. It introduces TrustPCA, a method and web tool that quantifies and visualizes the resulting projection uncertainty. The chapter first demonstrates through simulations how missing data impacts the reliability of standard aDNA projection techniques. It then presents a novel probabilistic framework to predict this uncertainty, visualized as a Gaussian density around the projected ancient sample, thereby enabling more robust interpretations in population genetic studies.

Chapter 11: Improving Taxonomic Classification with Feature Space Balancing.

Part III comprises advances to improve interpretability in the context of taxonomic analysis. Chapter 11 presents a novel approach for taxonomic classification of DNA sequences. Consistent with previous methods, we frame taxonomic classification as a machine learning (ML) task, where a classifier is trained on a set of sequences and evaluated on an independent test set. We demonstrate how the performance of simple ML methods, trained on basic features such as k-mer counts, can be significantly improved through a novel technique we call *feature space balancing*, which involves balancing the training data. We show that selectively omitting certain training data enhances the classifier's generalization capability, enabling it to outperform more complex ML models.

Chapter 12: BLASTphylo—an interactive web tool for taxonomic and phylogenetic analysis of genes.

Available via https://tuevis.cs.uni-tuebingen.de/blastphylo/

Chapter 12 introduces BLASTphylo, a web tool designed to visualize the occurrence of homologous proteins within a taxonomy at various taxonomic ranks, alongside their phylogenetic relationships displayed as interactive trees. The extensive list of putative homologous proteins generated by a BLAST search in large databases can often be overwhelming and difficult to interpret. BLASTphylo addresses this issue by processing these lists, selectively omitting redundant information, mapping the results onto a given taxonomy, and visualizing the taxonomy as an interactive tree. This approach allows users to inspect the occurrence of a protein across different taxonomic ranks. We demonstrate how automated processing, the integration of sequential bioinformatics tools, and interactive visualizations enhance the visual analytics of large-scale bioinformatics data.

Part IV concludes the dissertation by summarizing its key findings and their impact on addressing the posed research questions and advancing the field. It further outlines future research directions for the developed methodologies in uncertainty-aware dimensionality reduction and taxonomic analysis.

1.5 Additional Contributions

As part of my doctoral studies I have also contributed to the following peer-reviewed articles, which are not relevant to this dissertation.

- S.-H. Fan, P. Ebner, S. Reichert, T. Hertlein, S. Zabel, A. K. Lankapalli, K. Nieselt, K. Ohlsen, and F. Götz. "MpsAB is important for *Staphylococcus aureus* virulence and growth at atmospheric CO_2 levels". *Nature communications* 10.1 (2019) [62]

- A. Luqman, M. Z. Muttaqin, S. Yulaipi, P. Ebner, M. Matsuo, S. Zabel, P. M. Tribelli, K. Nieselt, D. Hidayati, and F. Götz. "Trace amines produced by skin bacteria accelerate wound healing in mice". *Communications Biology* 3.1 (2020) [159]
- S. Raue, S.-H. Fan, R. Rosenstein, S. Zabel, A. Luqman, K. Nieselt, and F. Götz. "The genome of *Staphylococcus epidermidis* O47". *Frontiers in Microbiology* 11 (2020) [229]
- A. Luqman, S. Zabel, S. Rahmdel, B. Merz, N. Gruenheit, J. Harter, K. Nieselt, and F. Götz. "The neuromodulator-encoding sadA gene is widely distributed in the human skin microbiome". *Frontiers in Microbiology* 11 (2020) [160]
- W. Fuhl, S. Zabel, and K. Nieselt. "Scanpath Classification with an n-mer Deep Neural Network Architecture". *Proceedings of the 2025 Symposium on Eye Tracking Research and Applications*. ETRA '25. New York, NY, USA: *Association for Computing Machinery*, 2025 [69]

Part I.

Preliminaries & Background

2. Dimensionality Reduction

The following sections introduce the general concept of dimensionality reduction (DR) methods and motivate their necessity when working with high-dimensional data. Two commonly used DR techniques—Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE)—are described in detail.

High-dimensional data $\boldsymbol{x} \in \mathbb{R}^D$ is described by an D-dimensional feature vector. The size of D, for which the data is considered high-dimensional, depends on the area of expertise and is ever-expanding due to technical inventions. A common characteristic of high-dimensional data is however that the number of dimensions exceeds the number of samples or observations by far. High-dimensional data offers the possibility for exhaustive data analysis. In bioinformatics, for example, gene expression levels are measured simultaneously for all transcripts of an organism—the transcriptome—, revealing complex gene expression patterns and regulatory mechanisms (e.g., [192]). Analyzing high-dimensional data can be challenging for several reasons:

- The phenomenon *curse of dimensionality*—first mentioned by Richard Bellman [21]—describes the exponential increase in the volume of Euclidean space associated with adding extra dimensions. Consider a D-dimensional unit hypercube (Figure 2.1a), which is partially covered by another hypercube of edge length s. The expected edge length s needed to cover a fraction r is given by $s = r^{1/D}$ (Figure 2.1b). Thus, to cover 10 % of the volume, an edge length of 0.1 is required for $D = 1$, while in the 10-dimensional case, the edge length must be about 0.8. [86] The exponential growth of the space results in high-dimensional, often sparse datasets, due to the limited amount of training data. The original meaning of the curse of dimensionality manifests itself in several problems in statistics and machine learning.
- Sampling from discrete high-dimensional spaces exhaustively can easily become infeasible. For example, a 10-dimensional unit cube with a uniform spacing of $s = 1/10$ results in $(1/s)^D = 10^{10}$ points. This is relevant when training ML models on a set of training samples that ideally contain all possible feature combinations to generalize accurately. Furthermore, high-dimensional parameter spaces are potentially harder to search for the optimal parameter combination.
- Some distance functions, such as the Euclidean distance, can become meaningless in high-dimensional space. Under certain assumptions about the data distribution, the ratio of the distances of the nearest and farthest neighbors of a reference point approaches 1 as $D \to \infty$ [24]. This can cause problems in ML algorithms that are based on distance. For example, the k-nearest neighbors algorithm uses a distance function to identify the k nearest neighbors of each input, which becomes ill-defined in high dimensions for some distance functions [3].

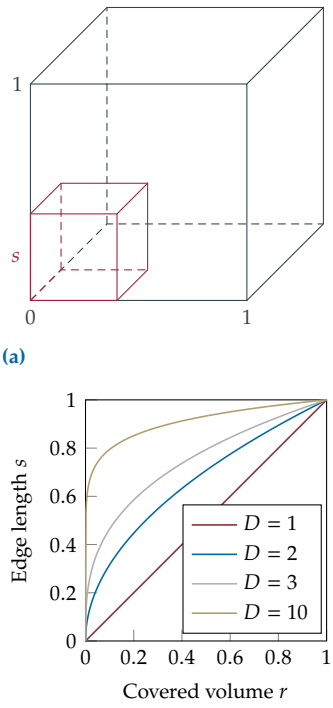


Figure 2.1: Illustration of the curse of dimensionality based on [86]. **(a)** D-dimensional unit hypercube cube (gray, $D = 3$). A fraction of its volume (r) is covered by a hypercube (red) of equal dimensionality with edge length s. **(b)** Edge length s needed to cover a certain fraction r of the volume for different dimensionalities D.

As discussed, concepts like 'local' neighborhoods, sampling, or distances, which are intuitive in low dimensions, often fail to generalize to high-dimensional spaces. One solution to the problem—if it exists—is to tailor methods to the characteristics of high-dimensional data. On the other hand, it can be the structure of the high-dimensional data itself that mitigates the curse of dimensionality: high-dimensional data may have an intrinsic low-dimensional structure due to redundant, correlated, or uninformative features. Capturing (part of) this structure to represent the data in a low-dimensional subspace is known as *dimensionality reduction*.

There is a large variety of dimensionality reduction techniques that retain different properties of the high-dimensional data structure in the low-dimensional representation. In this work, Principal Component Analysis (PCA, [95, 209]) and t-distributed stochastic neighbor embedding (t-SNE, [161]) are introduced in detail. PCA, a linear dimensionality reduction method, identifies directions of maximum variance in the original feature space (principal components) and projects the data onto these directions to define new features as linear combinations of the original features. This approach effectively preserves the global structure of the data by maximizing the amount of variance retained in the reduced representation. In contrast, t-SNE is a nonlinear DR technique that focuses on preserving the local structure of the data. It achieves this by converting pairwise similarities between objects in the high-dimensional space into probabilities, and then optimizing a two-dimensional embedding to reflect these probabilities.

2.1 Principal Component Analysis

1: also known as Karhunen–Loève transform

Principal component analysis[1] (PCA), was introduced by Karl Pearson [209] and Harold Hotelling [95], and is a widely used DR method enabling the visualization of multi-dimensional data. Let $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ be a high-dimensional data set, where each of the N independent and identically distributed (i.i.d.) samples is described by a D-dimensional feature vector, and where all features are assumed to be centered, i.e., have zero mean. PCA seeks to find a low-dimensional projection

$$\boldsymbol{z}_i = \boldsymbol{W}^\intercal \cdot \boldsymbol{x}_i, \tag{2.1}$$

where $\boldsymbol{W} \in \mathbb{R}^{D \times M}$, $\boldsymbol{z}_i \in \mathbb{R}^M$ and $M < D$. The columns $\boldsymbol{w}_i$ of the matrix $\boldsymbol{W}$ are constrained to be orthonormal and form the basis of the M-dimensional subspace in which the projected data is embedded. The optimal choice for the projection matrix $\boldsymbol{W}$ can be motivated by both maximizing the variance of the projections and minimizing the average reconstruction error (Figure 2.2). Here, we will focus on the former and follow the derivation given in [52]. Assuming $z_1, \ldots, z_M$ being the coordinates of the projected data, the variance of the first coordinate z_1 can be estimated as

$$V_1 := \mathbb{V}[z_1] \approx \frac{1}{N} \sum_{n=1}^{N} z_{n1}^2. \tag{2.2}$$

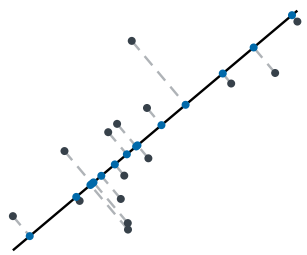


Figure 2.2: PCA projects data (grey) onto a lower-dimensional subspace (black line). The choice of the subspace is such that the projections (blue) are maximally spread along the line and that the mean squared Euclidean distance (light grey) between the original and projected coordinates is minimized.

The first coordinate of z_n can be substituted by the projection of the original data $\boldsymbol{x}_n$ onto the first basis vector $\boldsymbol{w}_1$:

$$V_1 = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{w}_1^\intercal \boldsymbol{x}_n)^2 = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{w}_1^\intercal \boldsymbol{x}_n \boldsymbol{x}_n^\intercal \boldsymbol{w}_1 \tag{2.3}$$

$$= \boldsymbol{w}_1^\intercal \left(\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}_n \boldsymbol{x}_n^\intercal\right)\boldsymbol{w}_1 = \boldsymbol{w}_1^\intercal \boldsymbol{S}\boldsymbol{w}_1, \tag{2.4}$$

where $\boldsymbol{S}$ is the covariance matrix of the original features. The direction of maximum variance is therefore given by the constrained optimization problem

$$\max_{\boldsymbol{w}_1} \boldsymbol{w}_1^\intercal \boldsymbol{S}\boldsymbol{w}_1 \tag{2.5}$$

$$\text{subjected to} \quad ||\boldsymbol{w}_1||^2 = 1. \tag{2.6}$$

The constrained optimization problem can be solved using the Lagrangian (not shown) and results in the relations

$$\boldsymbol{S}\boldsymbol{w}_1 = \lambda_1 \boldsymbol{w}_1 \tag{2.7}$$

$$\boldsymbol{w}_1^\intercal \boldsymbol{w}_1 = 1. \tag{2.8}$$

$\boldsymbol{w}_1$ turns out to be an eigenvector of the covariance matrix $\boldsymbol{S}$, with λ_1 being the corresponding eigenvalue. Using that, Equation (2.4) can be rewritten as

$$V_1 = \boldsymbol{w}_1^\intercal \boldsymbol{S}\boldsymbol{w}_1 = \lambda_1 \boldsymbol{w}_1^\intercal \boldsymbol{w}_1 = \lambda_1. \tag{2.9}$$

The variance of the projected data along the basis vector $\boldsymbol{w}_1$, which is an eigenvector of the covariance matrix, equals the associated eigenvalue. Therefore, to maximize the variance, the eigenvector of $\boldsymbol{S}$ associated with the largest eigenvalue is chosen as the first basis vector and called *first principal component*. The following $M - 1$ principal components are given by the eigenvectors associated with the $M - 1$ next largest eigenvalues, which is a valid orthonormal eigenbasis according to the spectral theorem.

Given the centered data matrix $\boldsymbol{X} \in \mathbb{R}^{D\times N}$, the eigenvectors $\boldsymbol{W}$ that form the principal axes can be computed in two ways:

- **Eigendecomposition of the covariance matrix $\boldsymbol{S}$**, which involves the computation of the covariance matrix $\boldsymbol{S} \in \mathbb{R}^{D\times D}$:

  $$\boldsymbol{S} = \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\intercal,$$

  followed by an eigendecomposition of $\boldsymbol{S}$

  $$\boldsymbol{S} = \boldsymbol{W}\boldsymbol{\Lambda}\boldsymbol{W}^\intercal,$$

  where eigenvectors are in the columns of $\boldsymbol{W} \in \mathbb{R}^{D\times D}$ and eigenvalues are the diagonal elements of $\boldsymbol{\Lambda} \in \mathbb{R}^{D\times D}$.
- **Singular value decomposition (SVD) of the data matrix $\boldsymbol{X}$**, which

is given by

$$\boldsymbol{X} = \boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{V}^\intercal,$$

where $\boldsymbol{W} \in \mathbb{R}^{D\times D}$ and $\boldsymbol{V}^\intercal \in \mathbb{R}^{N\times N}$ are orthogonal matrices and the diagonal entries $\boldsymbol{\Sigma}_{ii}$ of $\boldsymbol{\Sigma} \in \mathbb{R}^{D\times N}$ contain the singular values of $\boldsymbol{X}$. The number of positive singular values equals the rank r of the matrix $\boldsymbol{X}$, which is given by $r \leq \min\{D, N\}$. It can be shown that the columns of $\boldsymbol{W}$ are the eigenvectors of $\boldsymbol{X}\boldsymbol{X}^\intercal$ (and therefore $\boldsymbol{S}$) and singular values of $\boldsymbol{X}$ and eigenvalues of $\boldsymbol{S}$ are related by

$$\boldsymbol{\Lambda}_{ii} = \frac{\boldsymbol{\Sigma}_{ii}^2}{N}.$$

In high-dimensional settings ($D \gg N$) the rank r of the matrix $\boldsymbol{X}$ is given by $r \leq N$. Therefore, we obtain at most $r \leq N$ positive eigenvalues or singular values. Only eigenvectors associated with positive eigenvalues point in directions in the data that contain variance and need to be considered. It can be shown [52], that the N nonzero eigenvalues of $\boldsymbol{X}\boldsymbol{X}^\intercal$ equal the nonzero eigenvalues of $\boldsymbol{X}^\intercal\boldsymbol{X}$ and that the eigenvectors of $\boldsymbol{X}\boldsymbol{X}^\intercal$ can be recovered from the eigenvectors of $\boldsymbol{X}^\intercal\boldsymbol{X}$.[2] However, the matrix $\boldsymbol{X}^\intercal\boldsymbol{X} \in \mathbb{R}^{N\times N}$ is much smaller in size than the matrix $\boldsymbol{X}\boldsymbol{X}^\intercal \in \mathbb{R}^{D\times D}$, and the time complexity to compute the eigenvectors improves from $\mathcal{O}(D^3)$ to $\mathcal{O}(N^3)$.

Having computed the relevant M eigenvectors, the data can be projected onto the principal subspace. In matrix notation, the coordinates of the projected points $\boldsymbol{Z} \in \mathbb{R}^{M\times N}$ are given by

$$\boldsymbol{Z} = \boldsymbol{W}^\intercal\boldsymbol{X}. \tag{2.10}$$

2: The eigenvector equation (Equation (2.8)) can be reformulated as

$$\begin{aligned}\boldsymbol{S}\boldsymbol{w}_i &= \lambda_i\boldsymbol{w}_i\\ {}^1/_N\boldsymbol{X}\boldsymbol{X}^\intercal\boldsymbol{w}_i &= \lambda_i\boldsymbol{w}_i\\ {}^1/_N\boldsymbol{X}^\intercal\boldsymbol{X}\boldsymbol{X}^\intercal\boldsymbol{w}_i &= \lambda_i\boldsymbol{X}^\intercal\boldsymbol{w}_i,\end{aligned}$$

which yields

$${}^1/_N\boldsymbol{X}^\intercal\boldsymbol{X}\boldsymbol{v}_i = \lambda_i\boldsymbol{v}_i,$$

with $\boldsymbol{v}_i = \boldsymbol{X}^\intercal\boldsymbol{w}_i$.

2.1.1 PCA Visualizations

Visualizations of PCA embeddings often take the form of scatterplots, where each axis represents a principal component, typically the first two or three components. For visualizing more than three principal components, parallel coordinates plots can be used [108]. These visualizations facilitate the identification of clusters, trends, and outliers within the data. However, interpreting the observed patterns and the meaning of the principal axes themselves is challenging because the principal components are linear combinations of the original variables. The influence of an original variable on a principal component is determined by the assigned weight (the respective entry of the eigenvector), which defines its contribution to the linear combination. Typically, the columns of the weight matrix are mulitplied by the square root of the corresponding eigenvalue, resulting in what are known as loadings. The loading matrix is often visualized as a heat map to illustrate the significance of the original variables in the principal components [142]. Biplots [71] combine the visualization of principal component scores of the samples and the loadings of the variables into one plot. In this visualization method, the classical scatterplot is superimposed with a set of vectors representing the loadings of the variables. Thus, biplots provide a comprehensive way to visualize the relationships between samples and variables in a PCA.

2.2 t-Distributed Stochastic Neighbor Embedding

Compared to PCA, t-Distributed Stochastic Neighbor Embedding (t-SNE) is a more recent dimensionality reduction technique, that was published in 2008 by Laurens van der Maarten and Geoffrey Hinton [161] as a variation of the previously developed method Stochastic Neighbor Embedding (SNE) [91]. Given a high-dimensional data set $\boldsymbol{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$, t-SNE finds a two- or three-dimensional representation $\boldsymbol{Y} = \{\boldsymbol{y}_1, \dots, \boldsymbol{y}_N\}$, where $\boldsymbol{y} \in \mathbb{R}^2$ or $\boldsymbol{y} \in \mathbb{R}^3$, that can be displayed as a scatter plot. As a first step, high-dimensional Euclidean distances between pairs of data points (Figure 2.3a) are converted into conditional probabilities $p_{j\,|\,i}$ that represent similarities. The conditional probability $p_{j\,|\,i}$ represents the probability that $\boldsymbol{x}_i$ picks $\boldsymbol{x}_j$ as its neighbor if neighbors were picked proportional to a Gaussian centered at $\boldsymbol{x}_i$ (Figure 2.3b) and is defined as

$$p_{j\,|\,i} = \frac{\exp\left(-||\boldsymbol{x}_i - \boldsymbol{x}_j||^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-||\boldsymbol{x}_i - \boldsymbol{x}_k||^2 / 2\sigma_i^2\right)}, \tag{2.11}$$

where σ_i^2 is the variance of the Gaussian centered around $\boldsymbol{x}_i$. Similarly, neighbor probabilities in low-dimensional space are computed under the heavy-tailed Student's t-distribution with one degree of freedom (Figure 2.3c), which are defined by

$$q_{j\,|\,i} = \frac{\left(1 + ||\boldsymbol{y}_i - \boldsymbol{y}_j||^2\right)^{-1}}{\sum_{k \neq i} \left(1 + ||\boldsymbol{y}_i - \boldsymbol{y}_k||^2\right)^{-1}}. \tag{2.12}$$

Instead of using conditional probabilities $p_{j\,|\,i}$ and $q_{j\,|\,i}$, t-SNE uses the joint probabilities p_{ij} and q_{ij}, which are symmetric, i.e. $p_{ij} = p_{ji}$ and $q_{ij} = q_{ji}$. The joint probabilities are given by

$$p_{ij} = \frac{p_{j\,|\,i} + p_{i\,|\,j}}{2N}, \quad \text{and} \tag{2.13}$$

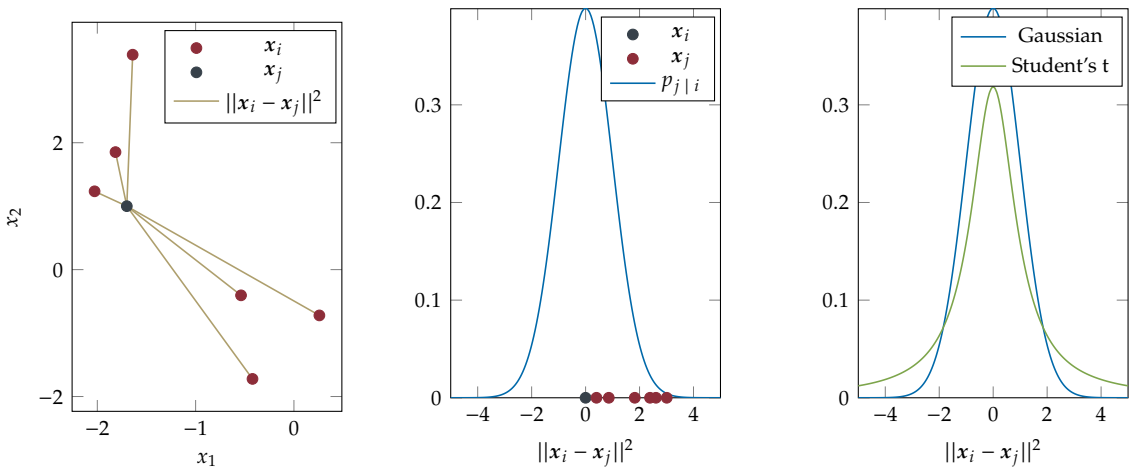
$$q_{ij} = \frac{\left(1 + ||\boldsymbol{y}_i - \boldsymbol{y}_j||^2\right)^{-1}}{\sum_{k \neq l} \left(1 + ||\boldsymbol{y}_k - \boldsymbol{y}_l||^2\right)^{-1}}. \tag{2.14}$$

t-SNE aims to model the coordinates $\boldsymbol{Y}$ in such a way that the joint probability distributions $\boldsymbol{P} \in \mathbb{R}^{N \times N}$ and $\boldsymbol{Q} \in \mathbb{R}^{N \times N}$ are equal. In particular, $\boldsymbol{Y}$ is optimized to minimize the Kullback-Leibler divergence D_{KL} between $\boldsymbol{P}$ and $\boldsymbol{Q}$:

$$\arg\min_{\boldsymbol{Y}} C = \arg\min_{\boldsymbol{Y}} D_{KL}(\boldsymbol{P}||\boldsymbol{Q}) = \arg\min_{\boldsymbol{Y}} \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{2.15}$$

The cost function is minimized using gradient descent. The coordinates $\boldsymbol{Y}$ are initialized as samples from an isotropic Gaussian centered around the origin and updated in each iteration t by

$$\boldsymbol{Y}^{(t)} = \boldsymbol{Y}^{(t-1)} + \eta \frac{\partial C}{\partial \boldsymbol{Y}} + \alpha(t) \left(\boldsymbol{Y}^{(t-1)} - \boldsymbol{Y}^{(t-2)}\right), \tag{2.16}$$



(a) High-dimensional space **(b)** Conditional probability $p_{j\,|\,i}$ **(c)** $p_{j\,|\,i}$ vs. $q_{j\,|\,i}$

Figure 2.3: t-SNE intuitions. **(a)** Pairwise Euclidean distance (gold) between the data point $\boldsymbol{x}_i$ (grey) and all other data points $\boldsymbol{x}_j$ (red) in high-dimensional space (here $D = 2$). **(b)** Conditional probabilities $p_{j\,|\,i}$ that $\boldsymbol{x}_i$ picks $\boldsymbol{x}_j$ as its neighbor. The Gaussian is centered around $\boldsymbol{x}_i$. All data points $\boldsymbol{x}_j$ are plotted at their respective Euclidean distance to $\boldsymbol{x}_i$. **(c)** Conditional probabilities $q_{j\,|\,i}$ in low-dimensional space are modeled by the Student's t-distribution, which has wider tails than the Gaussian distribution.

where η is the learning rate, $\alpha(t)$ the momentum at iteration t, and $\frac{\partial C}{\partial \boldsymbol{Y}}$ the gradient given by

$$\frac{\partial C}{\partial \boldsymbol{y}_i} = 4 \sum_j (p_{ij} - q_{ij})(\boldsymbol{y}_i - \boldsymbol{y}_j) \left(1 + ||\boldsymbol{y}_i - \boldsymbol{y}_j||^2\right)^{-1}. \tag{2.17}$$

The momentum term $\alpha(t)(\boldsymbol{Y}^{(t-1)} - \boldsymbol{Y}^{(t-2)})$ is added to ensure fast convergence and to avoid poor local minima. The user must choose the parameters $\alpha(t)$, η, and the total number of iterations T. The variance σ_i of the Gaussian kernel in Equation (2.11) is not selected directly by the user but determined using a binary search. σ_i is chosen to match a specified perplexity, which is defined by

$$Perp(\boldsymbol{P}_i) = 2^{H(\boldsymbol{P}_i)}, \tag{2.18}$$

where $\boldsymbol{P}_i$ represents the conditional probability distribution over all datapoints given the datapoint x_i, and $H(\boldsymbol{P}_i)$ is the Shannon entropy of $\boldsymbol{P}_i$. The perplexity describes the effective number of neighbors of each data point. Therefore, σ_i is chosen to match that number, which is different for each data point as it depends on its neighborhood.

The introduction of the Student's t-distribution was a major improvement over the previously known SNE method, which uses a Gaussian distribution for neighbor probabilities in low-dimensional space. This modification addresses the so-called *crowding problem*: in two-dimensional space, the area that is available to model high-dimensional distances of neighbors adequately might not be big enough. As a result, moderately distant data points have to be placed too far away in the low-dimensional space. Using the Gaussian kernel, the probability assigned to these moderately distant data points would be very low and they would be attracted to the center of the map, and crowd together. Using the Student's t-distribution, the loss function does not punish the distant placement as much since the probability does not drop as fast as the Gaussian probability.

Algorithm 1: t-SNE algorithm. [91]

Input: data set $\boldsymbol{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$
perplexity $Perp$
number of iterations T
learning rate η
momentum $\alpha(t)$
Output: low-dimensional data $\boldsymbol{Y}^{(T)} = \{\boldsymbol{y}_1, \dots, \boldsymbol{y}_N\}$
begin
  compute pairwise affinities $p_{j\,|\,i}$ (Equation (2.11)) using $Perp$;
  set $p_{ij} = \frac{p_{j\,|\,i} + p_{i\,|\,j}}{2N}$;
  initialize $\boldsymbol{Y}^{(0)} = \{\boldsymbol{y}_1, \dots, \boldsymbol{y}_N\}$ from $\mathcal{N}(0, 10^{-4}I)$;
  for $t = 1$ **to** T **do**
    compute affinities q_{ij} (Equation (2.14));
    compute gradient $\frac{\partial C}{\partial \boldsymbol{Y}}$ (Equation (2.17));
    set $\boldsymbol{Y}^{(t)} = \boldsymbol{Y}^{(t-1)} + \eta \frac{\partial C}{\partial \boldsymbol{Y}} + \alpha(t)\left(\boldsymbol{Y}^{(t-1)} - \boldsymbol{Y}^{(t-2)}\right)$;
  end
end

A simplified summary of the t-SNE algorithm is given in Algorithm 1.

2.2.1 t-SNE Accelerations

The standard implementation of the t-SNE algorithm (Algorithm 1) computes all pairwise similarities in both the high-dimensional and low-dimensional spaces, i.e., p_{ij} and q_{ij}, resulting in a runtime complexity of $\mathcal{O}(N^2)$. This quadratic scaling makes the method computationally expensive for datasets with many samples. Consequently, much of the research following the introduction of t-SNE has focused on heuristics to reduce runtime and improve scalability.

To this end, the gradient is typically splitted into two parts. Rewriting Equation (2.17), the gradient becomes:

$$\frac{\partial C}{\partial \boldsymbol{y}_i} = 4(F_{attr} + F_{rep}) = 4\left(\sum_j p_{ij} q_{ij} Z(\boldsymbol{y}_i - \boldsymbol{y}_j) - \sum_j q_{ij}^2 Z(\boldsymbol{y}_i - \boldsymbol{y}_j)\right), \tag{2.19}$$

where $Z = \sum_j (1 + ||\boldsymbol{y}_i - \boldsymbol{y}_j||^2)^{-1}$ acts as a normalizing factor. Here, F_{attr} denotes the sum of all attractive forces and includes p_{ij} encouraging nearby points to remain close to each other in the embedding. F_{rep} denotes the sum of all repulsive forces inducing global repulsion and preventing collapse of the embedded points.

To accelerate the computation of the attractive forces, heuristics approximate the similarities p_{ij} using only the k nearest neighbors for each sample $\boldsymbol{y}_i$. This is justified by the observation that distant samples have negligible small p_{ij} and thus do not contribute to F_{attr} [148, 273][3]. By employing approximate k-nearest neighbor algorithms, the time complexity for computing the p_{ij} can be reduced to $\mathcal{O}(N \log N)$ or better, depending on the method and dimensionality of the input space [148].

3: These strategies are implemented in Barnes-Hut t-SNE [273] and the FFT-accelerated Interpolation-based SNE (FIt-SNE) [148]

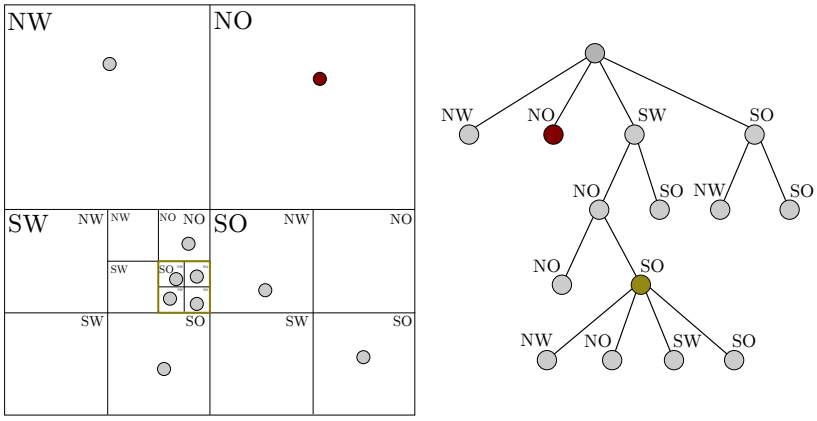


Figure 2.4: Illustration of the quadtree data structure used in Barnes-Hut t-SNE for efficient computation of repulsive forces. **(Left)** A two-dimensional dataset where space is recursively partitioned into four equal quadrants. Regions containing multiple data points are further subdivided. **(Right)** The resulting hierarchical tree structure representing the spatial partitioning. For a given query point (e.g., red), the Barnes-Hut approximation allows repulsive forces from distant groups of points (e.g., those within the golden area) to be approximated by a single force calculated using the center of mass of that group. This significantly reduces pairwise computations needed for the t-SNE gradient.

4: Implemented in Barnes-Hut t-SNE [273]

Computing repulsive forces requires the evaluation of all pairwise low-dimensional similarities q_{ij}, which forms an *N-body problem*. To mitigate this cost, van der Maaten proposed using the Barnes-Hut approximation originally developed for astrophysical simulations [15]. This method approximates interactions with distant points by treating them as a single aggregate at their center of mass, reducing the computational complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$ [273][4]. The algorithm constructs a quadtree (for 2D embeddings) or an octree (for 3D) to efficiently partition space and determine when to aggregate forces (Figure 2.4).

An alternative and even faster method is FIt-SNE, which uses interpolation and the Fast Fourier Transform (FFT) to approximate gradient computations on a grid, achieving near-linear complexity $\mathcal{O}(N)$ under typical conditions [148]. Although not detailed in this thesis, this approach provides one of the most scalable solutions for large-scale t-SNE.

Together, these acceleration strategies significantly reduce the runtime of t-SNE, making it feasible to apply the method to datasets containing hundreds of thousands or even millions of points—far beyond the practical limits of the original algorithm.

3. Probabilistic Inference

The theory of probability formulates the concept of uncertainty mathematically. Probabilities describe the degree of belief about an event. For a continuous variable $X \in \mathbb{R}$, the value x of X is assumed to not be known precisely. Instead, the probability density function (pdf) $p(x)$—a nonnegative function that sums to one—specifies the relative likelihood that X falls into a particular interval; the probability is given by the integral of the pdf over that range. Given two random variables, x and y, that are jointly distributed with $p(x, y)$, two fundamental rules of probability theory can be applied [88]. The sum rule relates the joint distribution to a marginal distribution, e. g. $p(y)$, which describes the likelihood of one variable independent of the other one, and is defined by

$$p(y) = \int p(x, y)\, \mathrm{d}x. \tag{3.1}$$

The product rule relates the joint distribution to a conditional distribution, e.g. $p(x \mid y)$, which describes the likelihood of one variable given the value of the other one, and is defined by

$$p(x, y) = p(x \mid y)\, p(y). \tag{3.2}$$

Conditional and marginal distributions are exemplarily shown in Figure 3.1. Bayes' theorem[1] combines both concepts; prior knowledge about an unobservable variable x, and observable data y generated according to the conditional density $p(y \mid x)$ (likelihood) are combined [88]. The posterior distribution describes our knowledge about x after considering the data, and is defined as

$$\overbrace{p(x \mid y)}^{\text{posterior}} = \frac{\overbrace{p(y \mid x)}^{\text{likelihood}}\, \overbrace{p(x)}^{\text{prior}}}{\underbrace{\int p(y \mid x)\, p(x)\, \mathrm{d}x}_{\text{evidence}}}. \tag{3.3}$$

The evidence ensures that the posterior distribution is normalized. Bayes' theorem provides the basis for probabilistic reasoning under uncertainty.

Closed-form solutions of the posterior distribution are usually only available if prior and likelihood are conjugate. The intractability of the integral describing the evidence (Equation (3.3)) is the main reason why exact inference is a hard problem. Therefore, approximate inference methods, such as Monte Carlo techniques [131], variational inference [25] or Laplace approximation, are applied.

In this chapter, we will firstly introduce Gaussian distributions and their properties, which allow closed-form solutions of various inference tasks (Section 3.1). Section 3.2 introduces some methods for approximate inference.

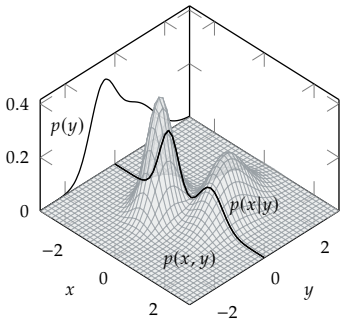


Figure 3.1: Sketch of a joint probability distribution $p(x, y)$, with conditional distribution $p(x \mid y)$ and marginal distribution $p(y)$ [88].

1: Named after Rev. Thomas Bayes [18].

3.1 Gaussian Inference

3.1.1 The Gaussian Distribution

The *Gaussian* or *normal* distribution is of central importance in probability theory and statistics. The *multivariate* normal distribution over $\mathbb{R}^D$ has a probability density function which is fully characterized by a *mean vector* $\boldsymbol{\mu} \in \mathbb{R}^D$ with elements $\mathbb{E}[\mathsf{x}_i] = \mu_i$ and a symmetric, positive semidefinite (SPD) *covariance matrix* $\boldsymbol{\Sigma} \in \mathbb{R}^{D\times D}$. It is defined by

$$\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\intercal \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right). \tag{3.4}$$

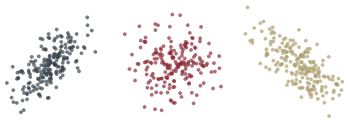


Figure 3.2: From left to right: observed values of positively correlated ($\boldsymbol{\Sigma}_{ij} > 0$), uncorrelated ($\boldsymbol{\Sigma}_{ij} = 0$) and negatively correlated ($\boldsymbol{\Sigma}_{ij} < 0$) Gaussian random variables.

The normal distribution is often referred to as $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ or $\mathsf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Diagonal elements of the covariance matrix $\boldsymbol{\Sigma}_{ii} = \sigma_i^2 = \mathbb{V}[\mathsf{x}_i]$ are the variances of the variables x_i; off-diagonal elements $\boldsymbol{\Sigma}_{ij} = \boldsymbol{\Sigma}_{ji} = \mathbb{C}[\mathsf{x}_i, \mathsf{x}_j], i \neq j$ are the cross-covariance terms which measure the degree to which x_i and x_j are linearly related (Figure 3.2). $\boldsymbol{\Sigma}_{ij} > 0$ indicates that the two variables are positively correlated, while for $\boldsymbol{\Sigma}_{ij} < 0$ they follow an opposite trend. $\boldsymbol{\Sigma}_{ij} = 0$ implies that the two variables are uncorrelated.

The popularity of normal distributions is due in part to the computationally convenient property that Gaussians reproduce under all linear operations. Any *affine transformation* of a Gaussian random variable $\mathsf{x}_1 \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \mathsf{x}_1 \in \mathbb{R}^D$, also has a Gaussian distribution,

$$\mathsf{x}_2 = \boldsymbol{A}\mathsf{x}_1 + \boldsymbol{b} \sim \mathcal{N}(\boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{b},\ \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\intercal) \tag{3.5}$$

for $\boldsymbol{A} \in \mathbb{R}^{M\times D}$ and $\boldsymbol{b} \in \mathbb{R}^M$.

Furthermore, the *product* of two Gaussian probability density functions $\mathcal{N}(\boldsymbol{x}; \boldsymbol{a}, \boldsymbol{A})\mathcal{N}(\boldsymbol{x}; \boldsymbol{b}, \boldsymbol{B})$ yields another Gaussian density function scaled by a factor $c \in \mathbb{R}$, which is given by

$$\mathcal{N}(\boldsymbol{x}; \boldsymbol{a}, \boldsymbol{A})\mathcal{N}(\boldsymbol{x}; \boldsymbol{b}, \boldsymbol{B}) = c\mathcal{N}(\boldsymbol{x}; \boldsymbol{c}, \boldsymbol{C}), \tag{3.6}$$

$$\text{where} \quad \boldsymbol{C} = (\boldsymbol{A}^{-1} + \boldsymbol{B}^{-1})^{-1}, \tag{3.7}$$

$$\text{and} \quad \boldsymbol{c} = \boldsymbol{C}(\boldsymbol{A}^{-1}\boldsymbol{a} + \boldsymbol{B}^{-1}\boldsymbol{b}). \tag{3.8}$$

It is important to note that the product of two Gaussian variables does *not* necessarily yield a Gaussian random variable.

Another useful property of normal distributions is that they are closed under *marginalization* and *conditioning*. Given a multivariate Gaussian variable partitioned in $\mathsf{x}_1 \in \mathbb{R}^D$ and $\mathsf{x}_2 \in \mathbb{R}^M$, as

$$p(\boldsymbol{x}_1, \boldsymbol{x}_2) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{bmatrix}; \begin{bmatrix}\boldsymbol{\mu}_1\\ \boldsymbol{\mu}_2\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12}\\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22}\end{bmatrix}\right) \tag{3.9}$$

both the conditionals $p(\boldsymbol{x}_1 \mid \boldsymbol{x}_2)$ and $p(\boldsymbol{x}_2 \mid \boldsymbol{x}_1)$, and the marginals $p(\boldsymbol{x}_1)$ and $p(\boldsymbol{x}_2)$ are normally distributed according to

$$p(\boldsymbol{x}_1 \mid \boldsymbol{x}_2) = \mathcal{N}(\boldsymbol{x}_1; \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2), \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}(\boldsymbol{\Sigma}_{22})^{-1}\boldsymbol{\Sigma}_{21}) \tag{3.10}$$

$$p(\boldsymbol{x}_1) = \int p(\boldsymbol{x}_1, \boldsymbol{x}_2)\, \mathrm{d}\boldsymbol{x}_2 = \mathcal{N}(\boldsymbol{x}_1; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11}), \tag{3.11}$$

and vice versa for x_2.

The aforementioned properties make Bayesian inference with Gaussians computationally very practical as inference amounts to linear algebra routines. Let $\mathsf{x} \in \mathbb{R}^D$ be a random variable with a Gaussian prior and let $\mathsf{y} \in \mathbb{R}^M$ be another random variable, such that $\mathsf{y} = \boldsymbol{A}\mathsf{x} + \boldsymbol{b}$, distributed according to

$$\begin{aligned} p(\boldsymbol{x}) &= \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \\ p(\boldsymbol{y} \mid \boldsymbol{x}) &= \mathcal{N}(\boldsymbol{y}; \boldsymbol{A}\mathsf{x} + \boldsymbol{b}, \boldsymbol{\Lambda}), \end{aligned}$$

respectively. The posterior distribution $p(\boldsymbol{x} \mid \boldsymbol{y})$ is defined as

$$p(\boldsymbol{x} \mid \boldsymbol{y}) = \mathcal{N}(\boldsymbol{x}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}), \tag{3.12}$$

$$\text{with} \quad \tilde{\boldsymbol{\Sigma}} = (\boldsymbol{\Sigma}^{-1} + \boldsymbol{A}^\intercal \boldsymbol{\Lambda}^{-1} \boldsymbol{A})^{-1} \tag{3.13}$$

$$= \boldsymbol{\Sigma} - \boldsymbol{\Sigma}\boldsymbol{A}^\intercal (\boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\intercal + \boldsymbol{\Lambda})^{-1} \boldsymbol{A}\boldsymbol{\Sigma}, \tag{3.14}$$

$$\text{and} \quad \tilde{\boldsymbol{\mu}} = \tilde{\boldsymbol{\Sigma}}(\boldsymbol{A}^\intercal \boldsymbol{\Lambda}^{-1}(\boldsymbol{y} - \boldsymbol{b}) + \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}) \tag{3.15}$$

$$= \boldsymbol{\mu} + \boldsymbol{\Sigma}\boldsymbol{A}^\intercal (\boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\intercal + \boldsymbol{\Lambda})^{-1}(\boldsymbol{y} - (\boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{b})). \tag{3.16}$$

The marginal $p(\boldsymbol{y})$ (evidence) is distributed as

$$p(\boldsymbol{y}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{b}, \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\intercal + \boldsymbol{\Lambda}) \tag{3.17}$$

3.1.2 Vectorized Matrices and the Kronecker Product

This chapter introduces some useful matrix tools when working with matrix-valued objects: the *Kronecker product* and the *vec operator* [165]. Let $\boldsymbol{A} \in \mathbb{R}^{N \times M}$ and $\boldsymbol{B} \in \mathbb{R}^{P \times Q}$ be two matrices of arbitrary size. The Kronecker product $\boldsymbol{A} \otimes \boldsymbol{B}$ defines a block-matrix of size $NP \times MQ$ with elements defined by

$$[\boldsymbol{A} \otimes \boldsymbol{B}]_{ij,kl} = [\boldsymbol{A}]_{ij}[\boldsymbol{B}]_{kl}. \tag{3.18}$$

Let $\boldsymbol{A} \in \mathbb{R}^{N \times M}$ be a matrix with a_j being its jth column. Applying the vec operator to $\boldsymbol{A}$ results in a vector of size NM

$$\operatorname{vec}(\boldsymbol{A}) = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}, \tag{3.19}$$

where the columns of $\boldsymbol{A}$ are stacked one underneath the other (column-stacking).[2] The operator vec^{-1} denotes the inverse of the vec operator, such that $\boldsymbol{A} = \operatorname{vec}^{-1}(\operatorname{vec} \boldsymbol{A})$. The Kronecker product and the vec operator are connected via

$$\operatorname{vec}(\boldsymbol{a}\boldsymbol{b}^\intercal) = \boldsymbol{b} \otimes \boldsymbol{a}. \tag{3.20}$$

The Kronecker product has some useful features [165] that support efficient calculations with matrices. Firstly, the *mixed product property* addresses the interchangeability of the conventional matrix product and

2: Literature often assumes column stacking. However, many programming languages default to row-stacking vectorization concatenating individual rows of the matrix, e.g. Numpy's `A.ravel()`.

the Kronecker product and is given by the equation

$$(\boldsymbol{A} \otimes \boldsymbol{B})(\boldsymbol{C} \otimes \boldsymbol{D}) = (\boldsymbol{AC}) \otimes (\boldsymbol{BD}) \tag{3.21}$$

assuming the matrices have compatible dimensions. Secondly, the *vectorization property* is applied to efficiently compute matrix-vector products when the matrix has a Kronecker structure. It states that for the three matrices $\boldsymbol{A}, \boldsymbol{B}$, and $\boldsymbol{C}$ the following equations hold if assuming

$$\text{column-stacking} \quad (\boldsymbol{A} \otimes \boldsymbol{B})\,\mathrm{vec}(\boldsymbol{C}) = \mathrm{vec}(\boldsymbol{BCA}^\intercal), \tag{3.22}$$

$$\text{row-stacking} \quad (\boldsymbol{A} \otimes \boldsymbol{B})\,\mathrm{vec}(\boldsymbol{C}) = \mathrm{vec}(\boldsymbol{ACB}^\intercal). \tag{3.23}$$

3.1.3 Gaussian Distributions over Matrices

As described in Section 3.1.1, several properties of multivariate Gaussian distributions make them convenient for inference tasks. Similarly, we can define Gaussian probability distributions over matrices [84], which are as well closed under marginalization, conditioning, and affine transformations. To this end, a matrix-valued random variable $\mathsf{X} \in \mathbb{R}^{N \times K}$ is vectorized using the vec operator (Equation (3.19)) and distributed using the multivariate Gaussian distribution for vectors, which is defined by

$$\mathcal{N}(\boldsymbol{X}, \boldsymbol{X}_0, \boldsymbol{\Sigma}_0) = \frac{\exp\left(-\frac{1}{2}(\mathrm{vec}(\boldsymbol{X}) - \mathrm{vec}(\boldsymbol{X}_0))^\intercal \boldsymbol{\Sigma}_0^{-1}(\mathrm{vec}(\boldsymbol{X}) - \mathrm{vec}(\boldsymbol{X}_0))\right)}{(2\pi)^{NK/2}|\boldsymbol{\Sigma}_0|^{1/2}}, \tag{3.24}$$

where $\boldsymbol{\Sigma}_0$ is a symmetric, positive semi-definite matrix in $\mathbb{R}^{NK \times NK}$.

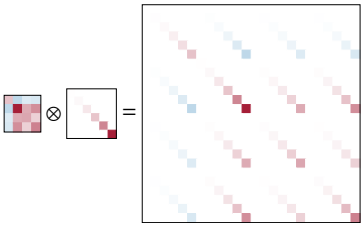


Figure 3.3: Heatmap visualization of a potential Kronecker product $\boldsymbol{W}_0 \otimes \boldsymbol{V}_0 = \boldsymbol{\Sigma}_0$. Matrix values are colored by matrix element value (negative: blue, positive: red).

Applying the aforementioned inference tasks (Section 3.1.1) to matrix-valued variables involves applying linear algebra routines, e.g. matrix inversion, to potentially large matrices. Therefore, to make Gaussian inference computationally feasible, it is inevitable to make assumptions about the covariance matrix. A suitable choice is Kronecker-factored covariance matrices, which model the covariance matrix $\boldsymbol{\Sigma}_0$ by the product of two smaller SPD matrices (see Figure 3.3 for an example):

$$\boldsymbol{\Sigma}_0 = \boldsymbol{W}_0 \otimes \boldsymbol{V}_0 \quad \text{with } \boldsymbol{W}_0 \in \mathbb{R}^{K \times K}, \boldsymbol{V}_0 \in \mathbb{R}^{N \times N}. \tag{3.25}$$

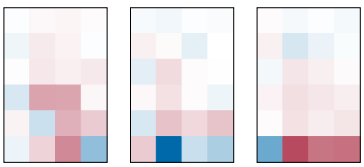


Figure 3.4: Three i.i.d. samples from the distribution $\mathcal{N}(\boldsymbol{I}, \boldsymbol{W}_0 \otimes \boldsymbol{V}_0)$ for the choices of $\boldsymbol{W}_0$ and $\boldsymbol{V}_0$ as shown in Figure 3.3. Matrix values are colored by matrix element value (negative: blue, positive: red).

$\boldsymbol{W}_0$ and $\boldsymbol{V}_0$ are often considered to capture the covariances among the columns and rows of the matrix, respectively. Figure 3.4 shows three samples $\boldsymbol{X}_i$ drawn from the distribution $\mathcal{N}(\boldsymbol{I}, \boldsymbol{W}_0 \otimes \boldsymbol{V}_0)$ according to

$$\boldsymbol{X}_i = \boldsymbol{I} + \mathrm{chol}(\boldsymbol{V}_0)\boldsymbol{S}_i\,\mathrm{chol}^\intercal(\boldsymbol{W}_0), \tag{3.26}$$

where $\mathrm{chol}(\boldsymbol{X})$ represents the Cholesky decomposition of $\boldsymbol{X}$, returning a lower tirangular matrix. The matrices $\boldsymbol{W}_0$ and $\boldsymbol{V}_0$ are chosen according to Figure 3.3, and $\boldsymbol{S}_i \in \mathbb{R}^{N \times K}$ contains N standard normally distributed samples.

3.2 Approximations

A common task in machine learning is computing expectations, i.e. solving integrals of the form

$$\mathbb{E}_{\boldsymbol{x}}[f(\boldsymbol{x})] = \int f(\boldsymbol{x})p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}. \tag{3.27}$$

An analytical solution of the integral is in general not available, even if $p(\boldsymbol{x})$ is Gaussian [52], if $f(\boldsymbol{x})$ is non-linear. Since the non-linearity is the source of the difficulty, a common strategy is to approximate with a simpler, linear function. A first-order Taylor series expansion of f provides such an approximation by locally linearizing the nonlinear function f. The mean and the covariance of $f(x)$ can then be inferred exactly if $p(x)$ is Gaussian distributed (see Equation (3.5)). For a more detailed description of this approach, see Section 3.2.1. A second-order Taylor approximation is often used to approximate an intractable posterior distribution around its mode—the maximum a posteriori estimate—as a Gaussian [163]. This method is known as the Laplace approximation [51]. However, if a Gaussian approximation to the probability distribution of interest is inappropriate due to its structure (e.g. multimodality) and therefore, Laplace's method is not applicable, Monte Carlo methods can be applied. These methods make use of repeated random sampling and are applied to estimate expectations of functions (Section 3.2.2).

3.2.1 Taylor Approximation

The *Taylor series*[3] of a function f is an infinite sum of terms, which are determined by using derivatives of f evaluated at x_0. For a smooth infinitely differentiable function $f : \mathbb{R} \mapsto \mathbb{R}$, it is defined by

3: Named after Brook Taylor [263].

$$T_\infty(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k, \tag{3.28}$$

where $f^{(k)}(x_0)$ is the kth derivative of f at x_0. Generally, a Taylor polynomial of degree n is an approximation of a function around x_0. Higher-order polynomials approximate the function better and more globally (see Figure 3.5 for an example) but require the evaluation of higher-order derivatives making them computationally expensive. To locally linearly approximate a function f a first-order Taylor series expansion is used. As seen in Figure 3.5 (grey, dashed), the first-order approximation T_1 is accurate around x_0, but diverges from the true function the farther we move away from x_0.

If $f : \mathbb{R}^N \mapsto \mathbb{R}^M$ is a vector field, the first-order multivariate Taylor approximation of f around $\boldsymbol{x}_0$ is defined as

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}_0) + (\nabla_{\boldsymbol{x}} f)(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0), \tag{3.29}$$

where $(\nabla_{\boldsymbol{x}} f)(\boldsymbol{x}_0) \in \mathbb{R}^{M \times N}$ is the gradient of f with respect to $\boldsymbol{x}$ evaluated at $\boldsymbol{x}_0$.

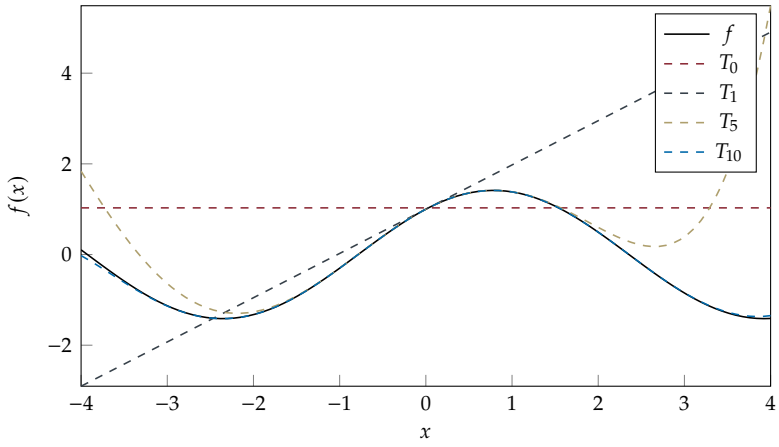


Figure 3.5: Taylor polynomials (dashed) of different degrees approximating the function $f = \sin(x) + \cos(x)$ (black, solid) around $x_0 = 0$.

We can use linearizations via Taylor expansions to approximate the posterior distribution by a Gaussian. An intuition of this approach is illustrated in Figure 3.6. Suppose we have a non-linear function f, such that $y = f(x)$. Projecting a Gaussian distribution $p(x)$ through this function is typically non-Gaussian. Linearizing f at the mean of $p(x)$ approximates the transformed distribution $p(y)$ as a Gaussian. As seen in Figure 3.6, top left, in the given example, the Gaussian posterior provides a good approximation to both the first and second moment of $p(y)$ at a lower cost than a Monte Carlo approximation [184].

To formalize this approach of error propagation, we consider a differentiable nonlinear function $\boldsymbol{f} : R^N \mapsto \mathbb{R}^M$ and a Gaussian distribution $p(\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x)$ over the input $\boldsymbol{x}$. The function $\boldsymbol{f}$ is approximated using a first-order Taylor series expanded at the mean $\boldsymbol{\mu}_x$:

$$\boldsymbol{f}(\boldsymbol{x}) \approx \hat{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{\mu}_x) + \boldsymbol{J}(\boldsymbol{x} - \boldsymbol{\mu}_x), \tag{3.30}$$

where $\boldsymbol{J} = (\nabla_x \boldsymbol{f})(\boldsymbol{\mu}_x)$. The first and second moment of the output distribution can be derived (Appendix A.1) as

$$\mathbb{E}[\hat{\boldsymbol{y}}] = \boldsymbol{f}(\boldsymbol{\mu}_x) \tag{3.31}$$

$$\mathbb{V}[\hat{\boldsymbol{y}}] = \boldsymbol{J}\boldsymbol{\Sigma}_x\boldsymbol{J}^\top \tag{3.32}$$

and used to define the Gaussian distribution $p(\hat{\boldsymbol{y}}) \sim \mathcal{N}(\boldsymbol{f}(\boldsymbol{\mu}_x), \boldsymbol{J}\boldsymbol{\Sigma}_x\boldsymbol{J}^\top)$.

A good approximation is achieved if the function is not too far from linear within the region that is within one standard deviation of the mean [30].

3.2.2 Monte Carlo Methods

Monte Carlo (MC) methods are a class of stochastical algorithms that use random samples to provide approximate solutions for various problems, including numerical integration, error estimation, optimization, and numerical simulations. In this chapter, the basic principle of MC methods is presented. A more comprehensive summary can be found in [131, 164].

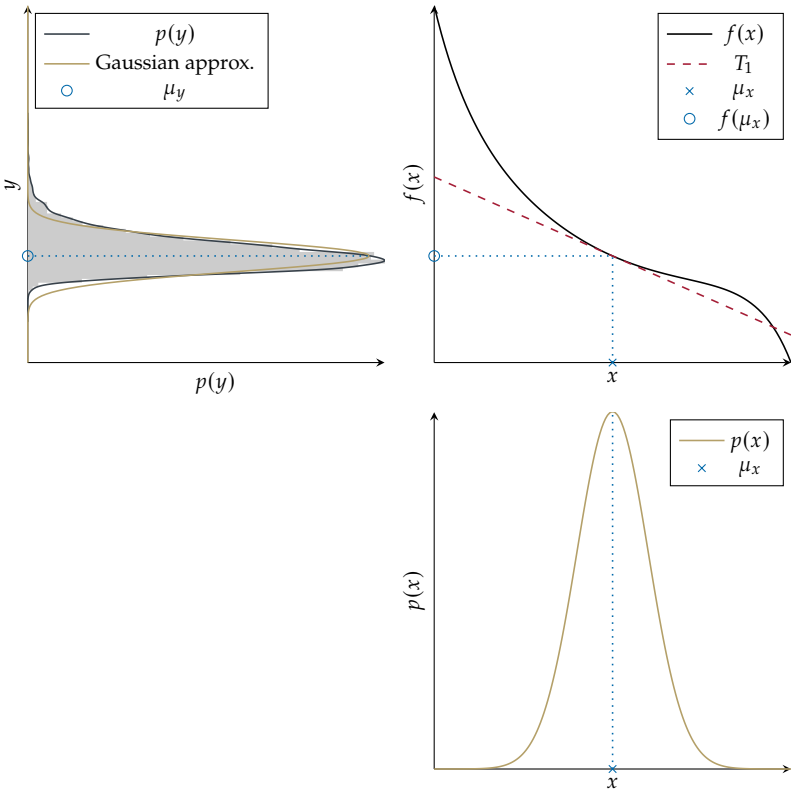


Figure 3.6: Passing a Gaussian distribution $p(x)$ (bottom right) through a nonlinear function f (top right, black line) results in a non-Gaussian distribution $p(y)$, which was generated by 1000 Monte Carlo draws (top left, grey line). The best Gaussian approximation of $p(y)$ is achieved by approximating the function with a first-order Taylor expansion T_1 at the mean μ_x of $p(x)$ and projecting the Gaussian along the straight line (top right, red dashed line). This figure was reconstructed from [184].

In general, MC methods compute expectations in the following way:

$$\mathbb{E}[f] = \mu \coloneqq \int f(x)p(x)\,\mathrm{d}x \approx \frac{1}{S}\sum_{i=1}^{S} f(x_i) \coloneqq \hat{\mu}, \tag{3.33}$$

where $x_i \sim p(x),\ i = 1,\ldots,S$, are independent and identically distributed samples.[4] $\hat{\mu}$—which is itself a random number—is called *Monte Carlo estimator*. It can be shown that this estimator is *unbiased* and therefore a good estimator of μ as its expected value $\mathbb{E}[\hat{\mu}]$ is equal to μ:

$$\mathbb{E}[\hat{\mu}] = \int \frac{1}{S}\sum_{i=1}^{S} f(x_i)p(x_i)\,\mathrm{d}x = \frac{1}{S}\sum_{i=1}^{S}\int f(x_i)p(x_i)\,\mathrm{d}x \tag{3.34}$$

$$= \frac{1}{S}\sum_{i=1}^{S}\mathbb{E}[f(x_i)] = \mu. \tag{3.35}$$

The expected square error (variance) of $\hat{\mu}$ can be derived as

$$\mathbb{E}[\hat{\mu} - \mathbb{E}[\hat{\mu}]]^2 = \mathbb{V}[\hat{\mu}] = \frac{\mathbb{V}[f]}{S} \tag{3.36}$$

and drops with $\mathcal{O}(S^{-1})$; the square root of the expected square error (root mean sqare deviation, RMSD) drops with $\mathcal{O}(S^{-1/2})$. It is important to notice that the accuracy of the Monte Carlo estimate does not depend on the dimension of the input argument x. The simple Monte Carlo method is therefore well suited for high-dimensional problems when closed-form solutions are not available. However, high dimensionality can cause other difficulties for Monte Carlo methods: Drawing i.i.d. samples from $p(x)$ might be difficult, especially in high-dimensional spaces [164]. Nevertheless, sampling is easy for some high-dimensional

4: The presented concepts also apply to multidimensional vectors.

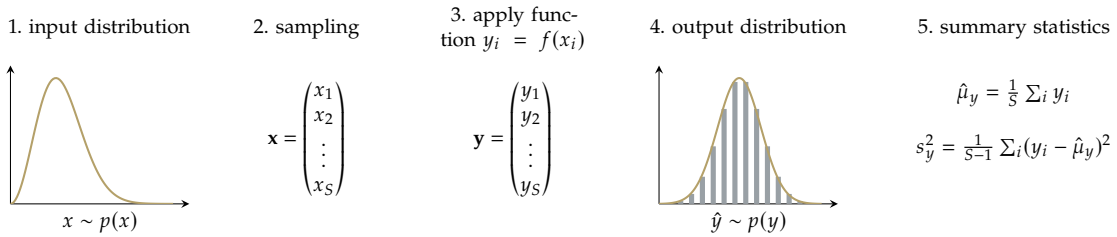


Figure 3.7: Steps involved in error analysis using the simple Monte Carlo method.

densities, such as Gaussian distributions.

When MC methods are used for error estimation, the second quantity of interest is the variance $\mathbb{V}[f]$ for which an unbiased estimator exists:

$$\mathbb{V}[f] = \sigma^2 := \int (f(x) - \mathbb{E}[f])^2 p(x) \mathrm{d}x \approx \frac{1}{S-1} \sum_{i=1}^{S} (f(x_i) - \mathbb{E}[f])^2 := s^2. \tag{3.37}$$

The sample variance s^2 is itself a random quantity with

$$\mathbb{E}[s^2] = \sigma^2 \tag{3.38}$$

$$\mathbb{V}[s^2] = \frac{1}{S} \left(\mu_4 - \frac{S-3}{S-1} \sigma^4 \right), \tag{3.39}$$

where $\mu_4 = \mathbb{E}[f - \mathbb{E}[f]]^4$ refers to the kurtosis of f [182]. Figure 3.7 shows an overview of the steps involved in error analysis using Monte Carlo methods.

4. Automatic Differentiation

We have seen in Section 3.2 that approximate inference methods require the evaluation of first- and second-order derivatives. This chapter introduces a family of techniques—called automatic differentiation (AD)—that can be used to efficiently and accurately compute derivatives of functions specified by computer programs. Besides AD, alternative methods exist for this task, for example manually deriving derivatives and coding them, numerical differentiation, and symbolic differentiation. However, those methods are often time-consuming, inaccurate, not very user-friendly, or do not scale [17]. Although the idea behind AD has been known for decades [193], recent developments in deep learning (DL) led to extensive usage and development of AD techniques in the field of machine learning (ML). By now, AD is part of many modern DL frameworks like TensorFlow [1] or PyTorch [205], where it is mainly used for backpropagation during the training of neural networks.

4.1 General Idea & an Example

The key idea of AD is to repeatedly apply the chain rule of calculus to estimate derivatives of a function given in a high-level programming language. To this end, the function is decomposed into a set of elementary operations for which derivatives are known. This includes for example addition, multiplication, and other elementary functions like sin or cos, linear algebra routines like matrix decompositions, but also control flow statements like loops or conditional branches. By combining the constituent derivatives of a function using the chain rule, the derivative of the entire function can be calculated with respect to its input arguments.

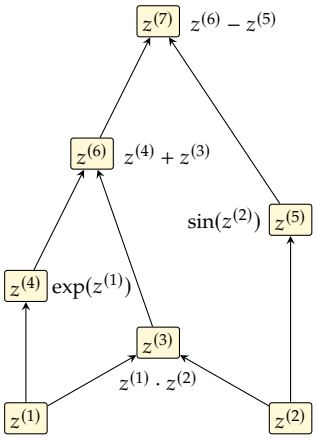
The following example and Figure 4.1 introduce the basic concept of AD, drawing on ideas and illustrations from [48, 195]. Figure 4.1a shows a representation of the function $z^{(7)} = \exp(z^{(1)}) + z^{(1)}z^{(2)} - \sin(z^{(2)})$ as a computer program. By applying basic operations, intermediate variables are sequentially built from the input variables and the result is returned. This trace of elementary operations can also be represented as a computational graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ [16], which is a directed acyclic graph with a set of nodes $\mathcal{V} = \{z^{(1)}, z^{(2)}, \ldots\}$ as variables and a set of edges $\mathcal{E} = \{(z^{(i_1)}, z^{(j_1)}), (z^{(i_2)}, z^{(j_2)}), \ldots\}$. A directed edge $(z^{(i)}, z^{(j)})$ is drawn from $z^{(i)}$ to $z^{(j)}$ when $z^{(j)}$ depends on $z^{(i)}$ (Figure 4.1b). Each edge $(z^{(i)}, z^{(j)})$ can be labelled by the local derivative $\partial z^{(j)}/\partial z^{(i)}$ (Figure 4.1c), which are accumulated according to the chain rule. To compute the partial derivative $\partial z^{(j)}/\partial z^{(i)}$ between any two nodes, the edges of all paths connecting them must be considered. Let $[z^{(i)} \to z^{(j)}]$ be the set of all paths connecting nodes $z^{(i)}$ and $z^{(j)}$, the partial derivative $\partial z^{(j)}/\partial z^{(i)}$ is computed as the sum of path products of local derivatives (Figure 4.1d),

$$\frac{\partial z^{(j)}}{\partial z^{(i)}} = \sum_{p \in [z^{(i)} \to z^{(j)}]} \prod_{(z^{(k)}, z^{(l)}) \in p} \frac{\partial z^{(l)}}{\partial z^{(k)}}, \tag{4.1}$$

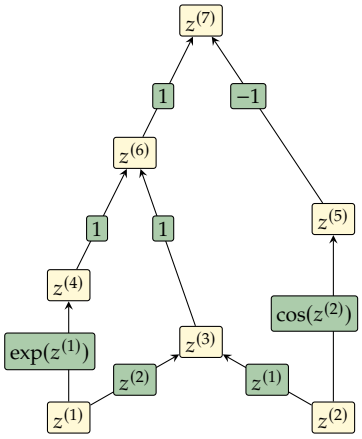
as introduced by Bauer [16].

```python
import numpy as np

def z7(z1: float, z2: float):
    '''Example Function.'''

    # intermediate variables
    z3 = z1 * z2
    z4 = np.exp(z1)
    z5 = np.sin(z2)
    z6 = z4 + z3

    # output variable
    z7 = z6 - z5

    return z7
```

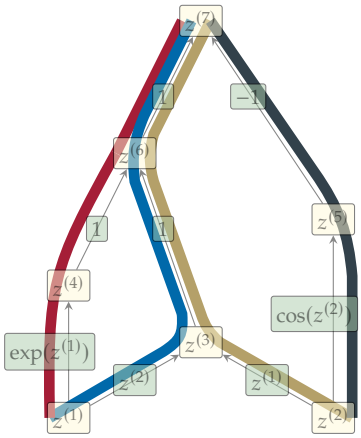
(a) Python program



(b) Computation graph



(c) Local derivatives



(d) Bauer paths

Figure 4.1: Basic concept of AD [48, 195]. **(a)** Computer program written in Python representing the function

$$z^{(7)} = \exp(z^{(1)}) + z^{(1)}z^{(2)} - \sin(z^{(2)}).$$

Intermediate results are represented by the variables $z^{(3)}$, $z^{(4)}$, $z^{(5)}$, and $z^{(6)}$. **(b)** Graph representation of the computer program showing the dependencies (edges) of the variables (vertices). **(c)** Graph showing local derivatives. Directed edges $(z^{(i)}, z^{(j)})$ connect two dependent variables $z^{(i)}$ and $z^{(j)}$ and are labeled by the value of their local derivative $\partial z^{(j)}/\partial z^{(i)}$. **(d)** Computation of the partial derivative of a node w.r.t. another node as a sum over all paths of products of local derivatives connecting both nodes. In this example, partial derivates of the output w.r.t the inputs are computed as follows:

$$\begin{aligned}\frac{\partial z^{(7)}}{\partial z^{(1)}} &= \frac{\partial z^{(7)}}{\partial z^{(6)}}\frac{\partial z^{(6)}}{\partial z^{(4)}}\frac{\partial z^{(4)}}{\partial z^{(1)}} \\ &\quad + \frac{\partial z^{(7)}}{\partial z^{(6)}}\frac{\partial z^{(6)}}{\partial z^{(3)}}\frac{\partial z^{(3)}}{\partial z^{(1)}} \\ &= \exp(z^{(1)}) + z^{(2)} \\ \frac{\partial z^{(7)}}{\partial z^{(2)}} &= \frac{\partial z^{(7)}}{\partial z^{(6)}}\frac{\partial z^{(6)}}{\partial z^{(3)}}\frac{\partial z^{(3)}}{\partial z^{(2)}} + \frac{\partial z^{(7)}}{\partial z^{(5)}}\frac{\partial z^{(5)}}{\partial z^{(2)}} \\ &= z^{(1)} - \cos(z^{(2)}).\end{aligned}$$

4.2 The Jacobian & its Chain Rule

Given a vector-valued differentiable function $f : \mathbb{R}^N \to \mathbb{R}^M$, the *Jacobian* $\boldsymbol{J}_{\boldsymbol{x}} f(\boldsymbol{x})$ of f w.r.t. $\boldsymbol{x}$ is an $M \times N$ matrix containing the partial derivatives of all the outputs w.r.t. all the inputs [52]:

$$\boldsymbol{J}_{\boldsymbol{x}} f(\boldsymbol{x}) = \frac{d\boldsymbol{f}}{d\boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_N} \\ \vdots & & \vdots \\ \frac{\partial f_M}{\partial x_1} & \cdots & \frac{\partial f_M}{\partial x_N} \end{bmatrix}. \tag{4.2}$$

When f is a composite differentiable function $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{h} \circ \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{h}(\boldsymbol{g}(\boldsymbol{x}))$, with $\boldsymbol{x} \in \mathbb{R}^N$, $\boldsymbol{g} : \mathbb{R}^N \to \mathbb{R}^K$, $\boldsymbol{h} : \mathbb{R}^K \to \mathbb{R}^M$, the chain rule is applied to obtain the Jacobian $\boldsymbol{J}_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^{M \times N}$

$$\boldsymbol{J}_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}) = [\boldsymbol{J}_{\boldsymbol{g}} \boldsymbol{h}(\boldsymbol{g}))] \cdot \boldsymbol{J}_{\boldsymbol{x}} \boldsymbol{g}(\boldsymbol{x}), \tag{4.3}$$

where

$$[\boldsymbol{J}_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x})]_{ij} = \frac{\partial f_i}{\partial x_j} = \frac{\partial h_i}{\partial g_1}\frac{\partial g_1}{\partial x_j} + \frac{\partial h_i}{\partial g_2}\frac{\partial g_2}{\partial x_j} + \ldots + \frac{\partial h_i}{\partial g_K}\frac{\partial g_K}{\partial x_j}. \tag{4.4}$$

Hence, the Jacobian is obtained as a sum over chained matrix products for vector-valued functions.

Equation (4.1) becomes [195]:

$$J_{z^{(i)}}(z^{(j)}) = \sum_{p \in [z^{(i)} \to z^{(j)}]} \prod_{(z^{(k)}, z^{(l)}) \in p} J_{z^{(k)}}(z^{(l)}) \tag{4.5}$$

The Jacobian (Equation (4.2)) and its chain rule (Equation (4.3)) can be generalized to matrix-valued functions [165]. Let $\boldsymbol{F} : \mathbb{R}^{N \times Q} \to \mathbb{R}^{M \times P}$ be a differentiable matrix-valued function. Using the vectorization operator from Equation (3.19), the Jacobian $J_{\boldsymbol{X}} \boldsymbol{F}(\boldsymbol{X})$ of $\boldsymbol{F}$ w.r.t. $\boldsymbol{X}$ is an $MP \times NQ$ matrix

$$J_{\boldsymbol{X}} \boldsymbol{F}(\boldsymbol{X}) = \frac{\partial \operatorname{vec}(\boldsymbol{F}(\boldsymbol{X}))}{\partial \operatorname{vec}(\boldsymbol{X})^{\intercal}}, \tag{4.6}$$

with entries

$$[J_{\boldsymbol{X}} \boldsymbol{F}(\boldsymbol{X})]_{ij} = \frac{\partial [\operatorname{vec}(\boldsymbol{F}(\boldsymbol{X}))]_i}{\partial [\operatorname{vec}(\boldsymbol{X})]_j}. \tag{4.7}$$

Let $\boldsymbol{F}$ be a composite differentiable matrix-valued function $\boldsymbol{F} = \boldsymbol{H} \circ \boldsymbol{G}$, where $\boldsymbol{F}(\boldsymbol{X}) = \boldsymbol{H}(\boldsymbol{G}(\boldsymbol{X}))$, $\boldsymbol{F} : \mathbb{R}^{N \times Q} \to \mathbb{R}^{M \times P}$, $\boldsymbol{H} : \mathbb{R}^{R \times S} \to \mathbb{R}^{M \times P}$, $\boldsymbol{G} : \mathbb{R}^{N \times Q} \to \mathbb{R}^{R \times S}$. The generalized Jacobian is defined as

$$J_{\boldsymbol{X}} \boldsymbol{F}(\boldsymbol{X}) = [J_{\boldsymbol{G}} \boldsymbol{H}(\boldsymbol{G})] J_{\boldsymbol{X}} \boldsymbol{G}(\boldsymbol{X}). \tag{4.8}$$

From this follows that the Jacobian for a composition can be interatively built from any number of component functions. In particular, given the component functions

$$\boldsymbol{f}_N : \mathbb{R}^{D_{N-1}} \to \mathbb{R}^{D_N} \tag{4.9}$$

$$\boldsymbol{x}_{N-1} \to \boldsymbol{x} = \boldsymbol{f}_N(\boldsymbol{x}_{N-1}) \tag{4.10}$$

and their composition

$$\boldsymbol{f} = \boldsymbol{f}_N \circ \boldsymbol{f}_{N-1} \circ \ldots \circ \boldsymbol{f}_2 \circ \boldsymbol{f}_1, \tag{4.11}$$

the Jacobian is computed by

$$J_{\boldsymbol{f}}(\boldsymbol{x}) = J_{\boldsymbol{f}_N}(\boldsymbol{x}_{N-1}) \cdot J_{\boldsymbol{f}_{N-1}}(\boldsymbol{x}_{N-2}) \cdot \ldots \cdot J_{\boldsymbol{f}_2}(\boldsymbol{x}_1) \cdot J_{\boldsymbol{f}_1}(\boldsymbol{x}_0), \tag{4.12}$$

where the notation of each term $J_{\boldsymbol{x}_{i-1}} \boldsymbol{f}_i(\boldsymbol{x}_{i-1})$ is simplified to $J_{\boldsymbol{f}_i}(\boldsymbol{x}_{i-1})$.

4.3 AD & its Main Modes

Automatic differentiation computes derivatives through an accumulation of values during code execution to generate numerical derivative evaluations rather than derivative expressions [17]. Different Jacobian accumulation schedules distinguish existing AD modes:

- **Forward accumulation (forward mode AD)**, propagates derivatives in the same order—from input(s) to output(s)—as function values are evaluated.
- **Reverse accumulation (reverse mode AD)**, starts at the function's output after a complete function evaluation and propagates the derivatives backward.

The next paragraphs introduce forward and reverse mode AD and are based on [23, 74, 169]. [82] offers a more comprehensive description.

4.3.1 Forward mode AD

Although the Jacobian $\boldsymbol{J}_{\boldsymbol{x}}\boldsymbol{f}(\boldsymbol{x})$ of a function $\boldsymbol{f} : \mathbb{R}^N \to \mathbb{R}^M$ evaluated at $\boldsymbol{x} \in \mathbb{R}^N$ is often represented as a matrix of size $\mathbb{R}^{M\times N}$, we can also think of it as a linear map, which maps the tangent space of the domain of $\boldsymbol{f}$ at the point $\boldsymbol{x}$ to the tangent space of the codomain of $\boldsymbol{f}$ at the point $\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^M$ (*pushforward map*):

$$\boldsymbol{J}_{\boldsymbol{x}}\boldsymbol{f}(\boldsymbol{x}) : \mathbb{R}^N \to \mathbb{R}^M. \tag{4.13}$$

So, given a tangent vector $\boldsymbol{v}_0 \in \mathbb{R}^N$, we obtain an output tangent vector $\boldsymbol{v} \in \mathbb{R}^M$ as

$$\boldsymbol{v} = \boldsymbol{J}_{\boldsymbol{x}}\boldsymbol{f}(\boldsymbol{x}) \cdot \boldsymbol{v}_0. \tag{4.14}$$

The mapping $(\boldsymbol{x}, \boldsymbol{v}_0) \to \boldsymbol{J}_{\boldsymbol{x}}\boldsymbol{f}(\boldsymbol{x}) \cdot \boldsymbol{v}_0$ is known as the *Jacobian-vector product* (JVP).

The computation of the output tangent vector can be decomposed due to the composite structure of $\boldsymbol{f}$ as defined in Equation (4.11):

$$\begin{aligned}
\boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{x}) \cdot \boldsymbol{v}_0 &= \boldsymbol{J}_{\boldsymbol{f}_N}(\boldsymbol{x}_{N-1}) \cdot \ldots \cdot \boldsymbol{J}_{\boldsymbol{f}_2}(\boldsymbol{x}_1) \cdot \boldsymbol{J}_{\boldsymbol{f}_1}(\boldsymbol{x}_0) \cdot \boldsymbol{v}_0 \\
&= \boldsymbol{J}_{\boldsymbol{f}_N}(\boldsymbol{x}_{N-1}) \cdot \ldots \cdot \boldsymbol{J}_{\boldsymbol{f}_2}(\boldsymbol{x}_1) \cdot \boldsymbol{v}_1 \\
&= \boldsymbol{J}_{\boldsymbol{f}_N}(\boldsymbol{x}_{N-1}) \cdot \ldots \cdot \boldsymbol{v}_2 \\
&= \ldots \\
&= \boldsymbol{J}_{\boldsymbol{f}_N}(\boldsymbol{x}_{N-1}) \cdot \boldsymbol{v}_{N-1} \\
&= \boldsymbol{v}_N
\end{aligned} \tag{4.15}$$

By that, we can propagate the input tangent vector by a series of composite Jacobian-vector products, which is called *forward mode* automatic differentiation. Intermediate vectors are denoted as *tangents* or *perturbations*. Intuitively, forward mode AD quantifies how much the output changes under small perturbations of the input.

If the JVP is applied to an i-th unit tangent vector, the i-th column of the Jacobian is obtained as the output tangent vector. Therefore, the Jacobian can be built one column at a time. This is advantageous for tall Jacobians of functions $\boldsymbol{f} : \mathbb{R}^N \to \mathbb{R}^M$, where $M \gg N$. When $N \gg M$, reverse mode AD is often preferred.

4.3.2 Reverse mode AD

Similar to considering the action of the Jacobian on a vector in the input space (Equation (4.14)), we can compute the action of the transposed Jacobian matrix on a vector $\boldsymbol{w}_0$ in the output space, $\mathbb{R}^N$,

$$\boldsymbol{w} = \boldsymbol{J}_{\boldsymbol{x}}^{\intercal}\boldsymbol{f}(\boldsymbol{x}) \cdot \boldsymbol{w}_0, \tag{4.16}$$

where $\boldsymbol{w} \in \mathbb{R}^M$ and $\boldsymbol{J}_{\boldsymbol{x}}^\intercal : \mathbb{R}^M \to \mathbb{R}^N$. Hence, given a perturbation in the output of the composite function, we can compute a vector that quantifies the change in input space needed to achieve that perturbation [23]. This computation can be performed sequentially similar to Equation (4.15):

$$\begin{aligned} \boldsymbol{J}_f^\intercal(\boldsymbol{x}) \cdot \boldsymbol{w}_0 &= \boldsymbol{J}_{f_1}^\intercal(\boldsymbol{x}_0) \cdot \ldots \cdot \boldsymbol{J}_{f_{N-1}}^\intercal(\boldsymbol{x}_{N-2}) \cdot \boldsymbol{J}_{f_N}^\intercal(\boldsymbol{x}_{N-1}) \cdot \boldsymbol{w}_0 \\ &= \boldsymbol{J}_{f_1}^\intercal(\boldsymbol{x}_0) \cdot \ldots \cdot \boldsymbol{J}_{f_{N-1}}^\intercal(\boldsymbol{x}_{N-2}) \cdot \boldsymbol{J}_{f_N}^\intercal(\boldsymbol{x}_{N-1}) \cdot \boldsymbol{w}_1 \\ &= \boldsymbol{J}_{f_1}^\intercal(\boldsymbol{x}_0) \cdot \ldots \cdot \boldsymbol{J}_{f_{N-1}}^\intercal(\boldsymbol{x}_{N-2}) \cdot \boldsymbol{w}_2 \\ &= \ldots \\ &= \boldsymbol{J}_{f_1}^\intercal(\boldsymbol{x}_0) \cdot \boldsymbol{w}_{N-1} \\ &= \boldsymbol{w}_N \end{aligned} \tag{4.17}$$

Opposed to before, the composition is traversed from the outside to the inside of the function and therefore called *reverse mode AD*. Intermediate vectors are referred to as *adjoints* or *sensitivities*. As seen, reverse mode AD requires a two-phase process, in which function values (primals) are computed in a forward pass, followed by a reverse pass propagating adjoints backward.

An alternative representation of Equation (4.16) is

$$\boldsymbol{w}^\intercal = \boldsymbol{w}_0^\intercal \cdot \boldsymbol{J}_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}), \tag{4.18}$$

where $\boldsymbol{w}_0^\intercal$ and $\boldsymbol{w}^\intercal$ are cotangent vectors from the input and output space, respectively. The mapping from the output to the input cotangent space is called *pullback*. With that, we can define a map $(\boldsymbol{x}, \boldsymbol{w}) \to \boldsymbol{w}^\intercal \cdot \boldsymbol{J}_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x})$, the *vector-Jacobian product* (VJP).

Reverse mode AD builds Jacobians row-by-row and is, therefore, superior to forward mode AD for functions $\boldsymbol{f} : \mathbb{R}^N \to \mathbb{R}^M$, when $N \gg M$. Reverse mode AD requires the computation graph to be stored in memory for the reverse pass, whereas the computation of primals and derivatives can be interleaved for forward mode AD. Therefore, when $N \approx M$, forward mode AD can be more efficient than reverse mode AD.

4.3.3 Higher-order Derivatives

The principles of forward and reverse mode automatic differentiation, used to compute first-order derivatives such as gradients, Jacobians, Jacobian-vector products (JVPs), and vector-Jacobian products (VJPs), can be recursively applied to obtain higher-order derivatives. This capability is particularly useful for computing Hessians (matrices of second-order partial derivatives) or their products with vectors (Hessian-vector products, HVPs). These are essential in many optimization algorithms, for example, in Newton's method to find local optima [169].

Consider a scalar-valued function $f : \mathbb{R}^N \to \mathbb{R}$. Its gradient $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$ is a vector-valued function mapping $\nabla f_{\boldsymbol{x}}(\boldsymbol{x}) : \mathbb{R}^N \to \mathbb{R}^N$. The *Hessian matrix*

$\boldsymbol{H}_{\boldsymbol{x}} f(\boldsymbol{x})$ (or $\nabla^2_{\boldsymbol{x}} f(\boldsymbol{x})$) is the Jacobian of this gradient function,

$$\boldsymbol{H}_{\boldsymbol{x}} f(\boldsymbol{x}) = \boldsymbol{J}_{\boldsymbol{x}}(\nabla_{\boldsymbol{x}} f)(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_N} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_N \partial x_1} & \frac{\partial^2 f}{\partial x_N \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_N^2} \end{bmatrix}, \tag{4.19}$$

where the Hessian is a symmetric $N \times N$ matrix.

The full Hessian matrix or specific Hessian-vector products can be computed by applying AD twice, using various combinations of forward and reverse modes. While the detailed strategies for these combinations are beyond the scope of this introductory section, interested readers are referred to comprehensive texts on automatic differentiation such as [17, 82] or practical manuals on computing HVPs [47].

AD can also readily compute *mixed partial derivatives*. Let $f : \mathbb{R}^P \times \mathbb{R}^N \to \mathbb{R}$ be a scalar-valued function of two vector arguments $\boldsymbol{x} \in \mathbb{R}^P$ and $\boldsymbol{y} \in \mathbb{R}^N$. The matrix of mixed partial derivatives, $\boldsymbol{J}_{\boldsymbol{xy}} f(\boldsymbol{x}, \boldsymbol{y}) = \frac{\partial^2 f}{\partial \boldsymbol{x} \partial \boldsymbol{y}}$, can be obtained by first computing the gradient of f with respect to one variable, for example, $\nabla_{\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y})$. This yields a vector-valued function that still depends on $\boldsymbol{x}$. Subsequently, the Jacobian of this resulting function with respect to the other variable $\boldsymbol{x}$ is computed, yielding $\boldsymbol{J}_{\boldsymbol{xy}} f(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{J}_{\boldsymbol{x}}(\nabla_{\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y})) \in \mathbb{R}^{N \times P}$. The order of differentiation (first with respect to $\boldsymbol{y}$ then $\boldsymbol{x}$, or vice-versa) does not alter the result for smooth functions. Modern AD frameworks typically handle such nested differentiation calls efficiently to produce these mixed Jacobian matrices or their products with vectors [17].

The ability to compute these higher-order derivatives and their vector products efficiently and accurately is a key strength of AD, enabling the implementation of sophisticated numerical methods that would be very challenging or error-prone to derive and code manually.

4.4 JAX—a Machine Learning Framework for AD

JAX (Just After eXecution) [29] is a relatively recent machine learning framework developed by Google that gained popularity in the last couple of years. It combines a new version of Autograd—a framework that can automatically differentiate native Python and Numpy code—and TensorFlow's XLA (Accelerated Linear Algebra) to just-in-time compile and run code on accelerators, like GPUs and TPUs. JAX includes a system of composable function transformations, including[1]:

- `jit()`: this function decorator allows users to transform their own functions into just-in-time compiled versions, which potentially increases computation speed.
- `grad()`, `hessian()`, `jacfwd()`, `jacrev()`, `vjp()`, `jvp()`: JAX supports automatic differentiation of arbitrary numerical functions. The mentioned function transformations evaluate gradients, Hessians, Jacobians in forward and reverse mode AD, VJPs, and JVPs.

1: https://jax.readthedocs.io/en/latest/index.html, last acessed 21 May, 2025

- `vmap()`: Applying a single function to a lot of data is a common task in ML research, for example when processing batches. The vectorization map provides a function transformation for efficient automatic batching, which simplifies programming and improves performance.

5. The Implicit Function Theorem

The Implicit Function Theorem (IFT) is a fundamental result in calculus and analysis with profound implications across various scientific and engineering disciplines, including machine learning and optimization. It provides conditions under which an equation or a system of equations implicitly defining a relationship between variables can be locally expressed as an explicit function. Furthermore, it offers a way to compute the derivative of this implicitly defined function without needing to find its explicit form. This chapter introduces the concept of implicit functions, implicit differentiation, and the formal statement of the IFT, with a particular focus on its application to stationary points of optimization problems, a scenario directly relevant to the uncertainty propagation method developed for t-SNE in Chapter 9.

5.1 Implicit Functions

Many relationships between variables are not expressed in an explicit form like $y = f(x)$, where y is directly given as a function of x. Instead, they are often defined *implicitly* by an equation of the form $G(x, y) = 0$. Here, y is not isolated on one side of the equation.

A classic example is the equation of a unit circle: $x^2 + y^2 - 1 = 0$. This equation defines a relationship between x and y. For any x in the interval $(-1, 1)$, there are generally two values of y that satisfy the equation (e.g., if $x = 0$, then $y = 1$ or $y = -1$). We cannot write a single function $y = f(x)$ that describes the entire circle globally. However, *locally*, around a point (x_0, y_0) on the circle (where $y_0 \neq 0$), we often can express y as an explicit function of x. For instance, near a point on the upper semi-circle, $y = \sqrt{1 - x^2}$, and near a point on the lower semi-circle, $y = -\sqrt{1 - x^2}$ (Figure 5.1). The IFT formalizes when such a local explicit function $y = f(x)$ exists and is differentiable.

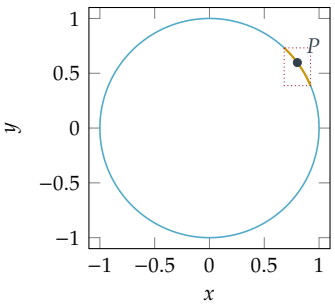


Figure 5.1: The unit circle, defined by the implicit equation $x^2 + y^2 - 1 = 0$. Near a point $P = (x_0, y_0)$ on the circle where the tangent is not vertical (i.e., $y_0 \neq 0$), y can be locally expressed as an explicit function of x. The red rectangle illustrates such a neighborhood.

5.2 Implicit Differentiation

Even if we cannot (or do not want to) find an explicit form $y = f(x)$ for an implicitly defined function $G(x, y) = 0$, we can often still find its derivative $\frac{dy}{dx}$. This process is called *implicit differentiation*. It involves differentiating both sides of the implicit equation $G(x, y) = 0$ with respect to x, treating y as a function of x (i.e., $y(x)$), and then applying the chain rule.

For the unit circle example, $G(x, y) = x^2 + y^2 - 1 = 0$: Differentiating

with respect to x:

$$\frac{d}{dx}(x^2 + y(x)^2 - 1) = \frac{d}{dx}(0)$$
$$\frac{d}{dx}(x^2) + \frac{d}{dx}(y(x)^2) - \frac{d}{dx}(1) = 0$$
$$2x + 2y(x) \cdot \frac{dy}{dx} - 0 = 0.$$

Solving for $\frac{dy}{dx}$, we get:

$$\frac{dy}{dx} = -\frac{x}{y}. \tag{5.1}$$

This gives the slope of the tangent to the circle at any point (x, y) where $y \neq 0$. Notice that this derivative is undefined when $y = 0$ (i.e., at $(1, 0)$ and $(-1, 0)$), which are precisely the points where the tangent to the circle is vertical and y cannot be locally expressed as a differentiable function of x. The IFT, introduced next, provides the precise conditions under which such an implicit derivative exists and is well-defined.

5.3 The Implicit Function Theorem

The Implicit Function Theorem is a cornerstone of mathematical analysis, with early ideas traceable to Newton and Leibniz, and a more formal statement by Cauchy [37, 129]. It rigorously establishes conditions under which an implicitly defined relationship $G(\boldsymbol{x}, \boldsymbol{y}) = \mathbf{0}$ can be locally solved to express $\boldsymbol{y}$ as an explicit function of $\boldsymbol{x}$, say $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$, and provides a formula for the derivative (Jacobian) of this function $\boldsymbol{f}$.

Formal Statement (Vector Case): Let $\boldsymbol{G} : \mathbb{R}^P \times \mathbb{R}^N \to \mathbb{R}^N$ be a continuously differentiable vector-valued function. That is, $\boldsymbol{G}$ takes an input $(\boldsymbol{x}, \boldsymbol{y})$ where $\boldsymbol{x} \in \mathbb{R}^P$ and $\boldsymbol{y} \in \mathbb{R}^N$, and outputs a vector in $\mathbb{R}^N$. We write this as $\boldsymbol{G}(\boldsymbol{x}, \boldsymbol{y}) = (G_1(\boldsymbol{x}, \boldsymbol{y}), \ldots, G_N(\boldsymbol{x}, \boldsymbol{y}))^\intercal$. Suppose there exists a point $(\boldsymbol{x}_0, \boldsymbol{y}_0)$ such that:

1. $\boldsymbol{G}(\boldsymbol{x}_0, \boldsymbol{y}_0) = \mathbf{0}$ (the point satisfies the implicit equation system).
2. The Jacobian matrix of $\boldsymbol{G}$ with respect to $\boldsymbol{y}$, evaluated at $(\boldsymbol{x}_0, \boldsymbol{y}_0)$, denoted $\boldsymbol{J}_{\boldsymbol{y}}\boldsymbol{G}(\boldsymbol{x}_0, \boldsymbol{y}_0) \in \mathbb{R}^{N \times N}$, is non-singular (i.e., its determinant is non-zero, so it is invertible). This Jacobian is:

$$\boldsymbol{J}_{\boldsymbol{y}}\boldsymbol{G}(\boldsymbol{x}_0, \boldsymbol{y}_0) = \begin{bmatrix} \frac{\partial G_1}{\partial y_1} & \cdots & \frac{\partial G_1}{\partial y_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial G_N}{\partial y_1} & \cdots & \frac{\partial G_N}{\partial y_N} \end{bmatrix}_{(\boldsymbol{x}_0, \boldsymbol{y}_0)}.$$

Under these conditions, the theorem guarantees that:

1. There exist open neighborhoods $S_{\boldsymbol{x}_0} \subset \mathbb{R}^P$ around $\boldsymbol{x}_0$ and $S_{\boldsymbol{y}_0} \subset \mathbb{R}^N$ around $\boldsymbol{y}_0$.
2. There exists a *unique* continuously differentiable function $\boldsymbol{f} : S_{\boldsymbol{x}_0} \to S_{\boldsymbol{y}_0}$ such that:
   - $\boldsymbol{f}(\boldsymbol{x}_0) = \boldsymbol{y}_0$.

- $\boldsymbol{G}(\boldsymbol{x}, \boldsymbol{f}(\boldsymbol{x})) = \mathbf{0}$ for all $\boldsymbol{x} \in S_{x_0}$ (i.e., $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$ locally solves the implicit equation).

3. The Jacobian matrix of this implicitly defined function $\boldsymbol{f}$ with respect to $\boldsymbol{x}$, evaluated at any $\boldsymbol{x} \in S_{x_0}$ (and corresponding $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$), is given by:

$$\boldsymbol{J}_{\boldsymbol{x}}\boldsymbol{f}(\boldsymbol{x}) = -\left[\boldsymbol{J}_{\boldsymbol{y}}\boldsymbol{G}(\boldsymbol{x}, \boldsymbol{f}(\boldsymbol{x}))\right]^{-1} \boldsymbol{J}_{\boldsymbol{x}}\boldsymbol{G}(\boldsymbol{x}, \boldsymbol{f}(\boldsymbol{x})). \tag{5.2}$$

Here, $\boldsymbol{J}_{\boldsymbol{x}}\boldsymbol{G}(\boldsymbol{x}, \boldsymbol{f}(\boldsymbol{x})) \in \mathbb{R}^{N \times P}$ is the Jacobian of G with respect to $\boldsymbol{x}$.

Revisiting the Unit Circle Example: The unit circle is defined by the scalar-valued function $G(x, y) = x^2 + y^2 - 1 = 0$. We can further define the partial derivatives $\frac{\partial G}{\partial y} = 2y$, and $\frac{\partial G}{\partial x} = 2x$.

Consider the point $P_1 = (x_0, y_0) = (0, 1)$.

1. $G(0, 1) = 0^2 + 1^2 - 1 = 0$. (Condition 1 satisfied)
2. $\boldsymbol{J}_y G(0, 1) = 2(1) = 2$, which is non-singular (invertible, as $2 \neq 0$). (Condition 2 satisfied)

Thus, the IFT guarantees that near $x = 0$, there's a unique function $y = f(x)$ such that $f(0) = 1$ and $x^2 + f(x)^2 - 1 = 0$. Its derivative is:

$$\frac{df}{dx} = -\left[\frac{\partial G}{\partial y}(x, f(x))\right]^{-1} \frac{\partial G}{\partial x}(x, f(x)) = -[2f(x)]^{-1}(2x) = -\frac{x}{f(x)}.$$

At $(0, 1)$, $\frac{df}{dx} = -\frac{0}{1} = 0$, which matches the slope of the tangent to $y = \sqrt{1 - x^2}$ at $x = 0$.

Now, consider $P_2 = (x_0, y_0) = (1, 0)$.

1. $G(1, 0) = 1^2 + 0^2 - 1 = 0$. (Condition 1 satisfied)
2. $\boldsymbol{J}_y G(1, 0) = 2(0) = 0$, which is singular. (Condition 2 fails)

The IFT does not apply at P_2 to express y as a differentiable function of x, which aligns with our earlier observation of a vertical tangent.

5.4 Implicit Functions Defined by Stationary Points

A particularly important application of the IFT in machine learning and optimization arises when a variable $\boldsymbol{y}$ is defined as the solution to an optimization problem that depends on another variable $\boldsymbol{x}$. Let $C : \mathbb{R}^P \times \mathbb{R}^N \to \mathbb{R}$ be an objective function $C(\boldsymbol{x}, \boldsymbol{y})$. We seek to find $\boldsymbol{y}^* \in \mathbb{R}^N$ that minimizes (or maximizes) this objective for a given $\boldsymbol{x}$:

$$\boldsymbol{y}^*(\boldsymbol{x}) = \underset{\boldsymbol{y} \in \mathbb{R}^N}{\arg\min}\, C(\boldsymbol{x}, \boldsymbol{y}). \tag{5.3}$$

If C is differentiable with respect to $\boldsymbol{y}$, a necessary condition for $\boldsymbol{y}^*$ to be a local extremum is that the gradient of C with respect to $\boldsymbol{y}$ must vanish at $\boldsymbol{y}^*$:

$$\nabla_{\boldsymbol{y}} C(\boldsymbol{x}, \boldsymbol{y}^*(\boldsymbol{x})) = \mathbf{0}. \tag{5.4}$$

This equation provides an implicit definition of the optimal $\boldsymbol{y}^*$ as a function of $\boldsymbol{x}$. We can now apply the IFT to determine the sensitivity of the optimal solution, $\boldsymbol{y}^*$, to changes in the input, $\boldsymbol{x}$, by calculating the Jacobian $\frac{\partial \boldsymbol{y}^*(\boldsymbol{x})}{\partial \boldsymbol{x}}$.

To do this, we map the stationary point condition to the formal structure of the IFT. We define the implicit function $\boldsymbol{G}$ from the theorem to be the gradient of our cost function:

$$\boldsymbol{G}(\boldsymbol{x}, \boldsymbol{y}) \triangleq \nabla_{\boldsymbol{y}} C(\boldsymbol{x}, \boldsymbol{y}).$$

With this definition, the stationary condition $\nabla_{\boldsymbol{y}} C(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{0}$ becomes the implicit system $\boldsymbol{G}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{0}$. The conditions for the IFT to guarantee a differentiable solution map $\boldsymbol{y}^*(\boldsymbol{x})$ are:

1. $\nabla_{\boldsymbol{y}} C(\boldsymbol{x}_0, \boldsymbol{y}_0^*) = \boldsymbol{0}$ for some specific $(\boldsymbol{x}_0, \boldsymbol{y}_0^*)$ (i.e., $\boldsymbol{y}_0^*$ is a stationary point for input $\boldsymbol{x}_0$).
2. The Jacobian of our implicit function $\boldsymbol{G}$ with respect to $\boldsymbol{y}$, evaluated at $(\boldsymbol{x}_0, \boldsymbol{y}_0^*)$, must be non-singular. This Jacobian is:

$$\boldsymbol{J}_{\boldsymbol{y}} \boldsymbol{G}(\boldsymbol{x}_0, \boldsymbol{y}_0^*) = \frac{\partial}{\partial \boldsymbol{y}} \left(\nabla_{\boldsymbol{y}} C(\boldsymbol{x}, \boldsymbol{y}) \right) \Big|_{(\boldsymbol{x}_0, \boldsymbol{y}_0^*)} = \frac{\partial^2 C(\boldsymbol{x}, \boldsymbol{y})}{\partial \boldsymbol{y}^2} \Big|_{(\boldsymbol{x}_0, \boldsymbol{y}_0^*)}.$$

   This is precisely the Hessian matrix of the objective function C with respect to $\boldsymbol{y}$, denoted $\boldsymbol{H}_{\boldsymbol{y}\boldsymbol{y}} C(\boldsymbol{x}_0, \boldsymbol{y}_0^*)$. The condition is that this Hessian must be invertible at the optimum.

If these conditions hold, we apply the IFT formula (Equation (5.2)) to find the Jacobian of the optimal solution map $\boldsymbol{y}^*(\boldsymbol{x})$. Substituting our definitions, the Jacobian $\boldsymbol{J}_{\boldsymbol{y}} \boldsymbol{G}$ is the Hessian with respect to $\boldsymbol{y}$, and the Jacobian $\boldsymbol{J}_{\boldsymbol{x}} \boldsymbol{G}$ is the mixed partial derivative:

$$\boldsymbol{J}_{\boldsymbol{y}} \boldsymbol{G}(\boldsymbol{x}, \boldsymbol{y}^*) = \frac{\partial^2 C(\boldsymbol{x}, \boldsymbol{y}^*)}{\partial \boldsymbol{y}^2} \quad \text{and} \quad \boldsymbol{J}_{\boldsymbol{x}} \boldsymbol{G}(\boldsymbol{x}, \boldsymbol{y}^*) = \frac{\partial^2 C(\boldsymbol{x}, \boldsymbol{y}^*)}{\partial \boldsymbol{x} \partial \boldsymbol{y}}.$$

This gives the final result for the sensitivity of the optimal solution:

$$\frac{\partial \boldsymbol{y}^*(\boldsymbol{x})}{\partial \boldsymbol{x}} = - \left[\frac{\partial^2 C(\boldsymbol{x}, \boldsymbol{y}^*)}{\partial \boldsymbol{y}^2} \right]^{-1} \frac{\partial^2 C(\boldsymbol{x}, \boldsymbol{y}^*)}{\partial \boldsymbol{x} \partial \boldsymbol{y}}. \tag{5.5}$$

This equation allows us to compute how an optimal solution changes with respect to $\boldsymbol{x}$, using only second-order derivatives of the objective function C at the optimum, without needing to differentiate through the optimization process itself. This is precisely the approach used in Chapter 9 to find the sensitivity of the t-SNE embedding to changes in the input data.

6. Uncertainty Visualization

Quantifying and visualizing uncertainties inherent in the data or arising during its analysis is inevitable to improve interpretability and prevent erroneous decisions. This is especially important when making decisions concerning, for example, health and safety. Incorporating uncertainty can be time-consuming, or might require the use of capable analysis techniques and additional knowledge to be interpreted correctly. Appropriate uncertainty visualizations facilitate the evaluation of reliability and are the primary concern of this chapter.

6.1 Types and Sources of Uncertainty

Traditionally, uncertainty is divided into two types, *aleatoric*[1] and *epistemic*[2] uncertainty [259]. Aleatoric—sometimes called stochastic—uncertainty arises from an inherently random process; epistemic uncertainty comes from a lack of knowledge and can theoretically be reduced given more information.

1: from the Latin *alea*, which translates to *dice*.

2: from the Greek ἐπιστήμη meaning knowledge.

Both types of uncertainty arise at different stages of data analysis pipelines. In literature [27, 42, 130], the main sources of uncertainty fall into the following categories:

1. Data uncertainty
   - Incompleteness: Missing or sparse data
   - Inaccuracy and imprecision: Measurement errors
   - Data acquisition through sampling
   - Data ambiguities
2. Model uncertainty
   - Wrong or simplified model or model parameters
   - Algorithmic uncertainties, e.g. approximations
3. Computational uncertainty
   - Simplified implementations
   - Numerical imprecisions

The data visualization process also affects how information is perceived and might introduce visualization uncertainty. This includes the visualization of the uncertainty itself, which requires additional visual channels to be interpreted by the user. The human perception of uncertainty and its impact on the decision-making process were evaluated in several studies [218, 295]. This has led to frameworks for quantitatively assessing uncertainty visualization approaches and providing guidance for their development [102].

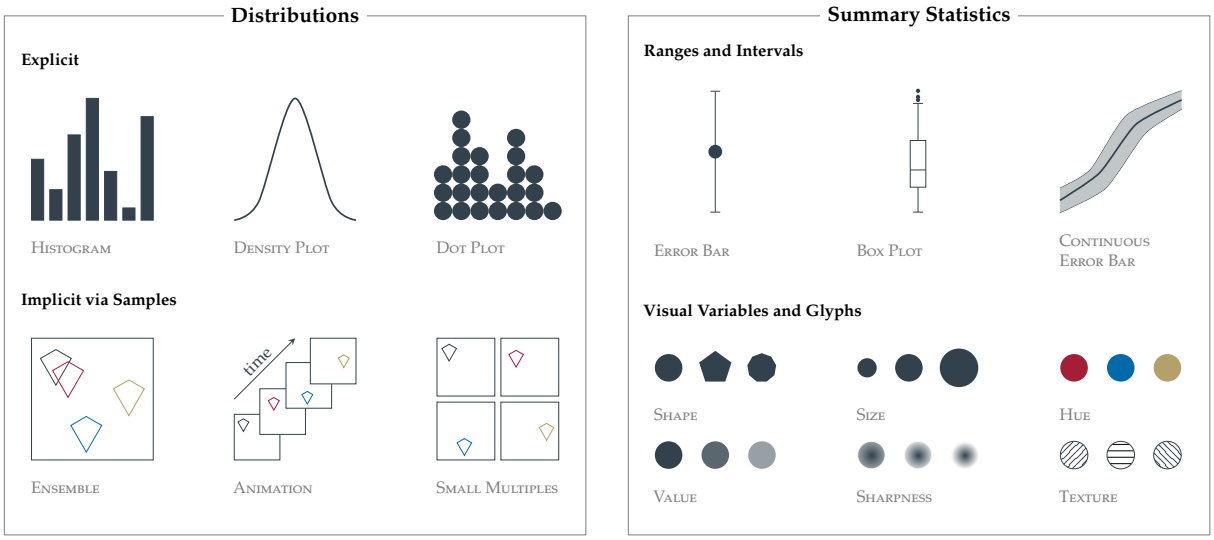


Figure 6.1: Uncertainty visualization strategies. Distributional visualization approaches (left) communicate uncertainty either by explicitly visualizing the distribution or by providing an intuition about the distribution via samples drawn from it. Summary statistics (right) of distributions can be visualized as ranges or encoded as visual variables and glyphs. The shown techniques serve as example visualizations and do not provide a comprehensive list of existing approaches. The figure is based on [113, 200, 285].

6.2 Encoding Uncertainty Visually

There is a large variety of visualization approaches used to communicate uncertainty. Several recent surveys categorize and structure existing methods from different perspectives: Assuming a probabilistic model of uncertainty, Weiskopf [285] distinguishes between methods, that show the full characteristics of the underlying distribution and methods, that visualize statistical summaries (Figure 6.1), or combinations of both. Padilla et al. [200] provide a similar taxonomy discriminating between distributional visualizations, visual encodings of uncertainty, and hybrid approaches. Kamal et al. [113] define the categories geometry, attributes, animation, visual variables, graphical techniques, and glyphs as potential uncertainty visualization channels. Which of the uncertainty visualization strategies is most appropriate depends on the data, the type of data visualization, the visualization channels already occupied, and the target audience who may perceive uncertainty differently. Figure 6.1 shows a non-exhaustive list of different visual representations of uncertainty, each with unique properties, advantages, and disadvantages, which are briefly discussed in the following. Histograms, density plots or dot plots (Figure 6.1, left) visualize distributions explicitly and provide a comprehensive view of the distribution at the price of an additional axis perpendicular to the data axis. Implicit visualizations do not require additional axes but use the same space as uncertainty-unaware data visualization, which makes them understandable to the user. Samples are drawn from the target distribution and treated as if not affected by uncertainty. To get an intuition about the structure of the samples—and therefore the structure of the distribution—they are visualized as an ensemble or individually using time (via animation) or space (small multiples [271]) to traverse through the samples one by one. Overlaying all samples within one plot results in a compact static image allowing users to easily compare different samples. The potential for overplotting and visual clutter with large sample sizes or large amounts of data are

disadvantages of this approach [285]. Animations avoid overplotting and can facilitate uncertainty interpretation [112], but are unusable for print. Small multiples show samples next to each other avoiding overplotting, but making it hard to compare differences between individual samples.

In many cases, it might be sufficient to visualize summary statistics like the mean and standard deviation of the distribution (Figure 6.1, right). Measures of central tendency and measures of variability are typically visualized as e.g. error bars or boxplots, which are well-known visual representations and can therefore be interpreted by the majority of users without additional learning. Alternatively, visual variables and glyphs provide a large design space to include uncertainty in visualizations. For example, it has been shown that an intuitive visualization approach is to encode uncertainty through the sharpness of objects [162]. However, it may be difficult to quantify the level of uncertainty. Additionally, the design and choice of the visual channel must not interfere with the actual meaning of the visualization.

7. Classification

Classification is the process by which objects are grouped into classes based on their traits. This section introduces the concepts and terms used in organism classification and the machine-learning classification methods used in this thesis.

7.1 Classification of Organisms

7.1.1 Taxonomy

Taxonomy[1] is the scientific discipline concerned with the identification, description, nomenclature, and classification of living organisms [249]. The central objects of taxonomic studies are *taxa* (singular: *taxon*), which are delimited groups of organisms that are given a taxonomic rank. Several taxa of a given rank can be aggregated to form a more inclusive taxon at a higher taxonomic rank. The hierarchical classification system that is primarily used today dates back to Carl Linnaeus, who developed a ranked system known as the Linnaean taxonomy, as well as a binomial nomenclature [149]. The primary taxonomic ranks in modern use are *kingdom*, *phylum*, *class*, *order*, *family*, *genus*, and *species*.

1: from the Greek τάξις [taxis] meaning order or arrangement, and -νόμος [-nomos], which translates to law or habit.

There are two main types of classification: phenetic classification, which groups taxa based on their overall similarity, and phylogenetic classification, which groups organisms according to their evolutionary history [249].

7.1.2 Taxonomic Classification

Assigning a DNA sequence to its taxon of origin is called taxonomic classification and is most relevant in metagenomics, where the origin of a large number of reads must be identified to infer the composition of a microbial community [248]. The taxonomic classification of sequences can be performed at different taxonomic ranks, and becomes more difficult at lower taxonomic ranks due to an increased sequence similarity of the taxa.

7.1.3 Homolgy

Homology is defined as similarity resulting from common ancestry. Homologous traits provide evidence of common ancestry and are critical to the construction of phylogenies. Morphological homology describes structural similarities between organisms, while molecular homology refers to similarities at the genetic level.

7.1.4 Phylogeny

Phylogeny[2] describes the evolutionary history and relation among a set of taxa. Species are considered related if they share a recent common ancestor [106]. In the field of evolutionary biology, there are three terms used to describe the relationship between organism based on their evolutionary history and shared ancestry [89]: A *monophyletic group*, or clade, includes an ancestor and all its descendants, making it a natural group where all members are derived from a single common ancestor. Conversely, a *paraphyletic group* contains a common ancestor and some, but not all, of its descendants, thereby excluding some lineages that share the same ancestor. A *polyphenetic group* contains a group of organisms, but not their common ancestor [249].

Section 7.1.5 outlines how sequencing data is used to derive a phylogeny for a set of taxa.

2: from the Greek $\phi\upsilon\lambda\acute{\eta}$ [phylé], meaning tribe, clan, or race, and $\gamma\varepsilon\nu\varepsilon\tau\iota\kappa\acute{o}\zeta$ [genetikós], translating to origin, source or birth.

7.1.5 Phylogenetic Analysis

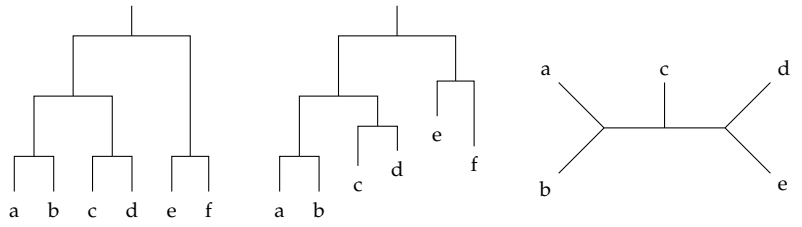
Phylogenetic analysis aims to infer the phylogeny for a set of taxa and typically results in a *phylogenetic tree* (see Section 7.1.6), a diagram reflecting the evolutionary history of the given taxa. Phylogenetic inference is based on heritable traits, which can be either morphological features, such as the morphology or anatomy, or molecular features derived from sequence data, such as DNA sequences [249].

In molecular phylogenetics, a set of taxa is typically represented by a set of DNA sequences derived from homologous genes or loci. These sequences differ from each other due to evolutionary events such as mutations, insertions, and deletions. To ensure that the corresponding features are compared, the sequences are aligned prior to the phylogenetic analysis, forming a *multiple sequence alignment* (MSA). Multiple sequence alignment involves arranging the sequences in such a way that homologous positions across different sequences are aligned. This alignment process often requires the introduction of gaps into the sequences to account for insertions and deletions, resulting in sequences of equal length. The goal is to optimize an alignment score that reflects the overall similarity of the sequences. [106] For more details on different alignment methods, see for example [115, 253].

Finding the optimal tree for a given set of taxa is difficult due to the number of possible bifurcating trees that exist[3]. Therefore, heuristics and optimization methods are used to find a tree that fits the data reasonably well.

3: For n taxa, $(2n-3)!!$ rooted phylogenetic trees can be constructed [65]

Phylogenetic inference methods can be broadly categorized into distance-based and character-based methods. Distance-based methods iteratively construct a phylogenetic tree from genetic distances between pairs of sequences, with the neighbor joining [241] and the UPGMA [251] algorithm being prominent examples. Character-based methods, such as those based on maximum parsimony, maximum likelihood, and Bayesian inference, rely on the sequence characters themselves to build the phylogenetic tree [66].



(a) Cladogram (b) Phylogram (c) Unrooted tree

Figure 7.1: Phylogenetic tree types.

Most phylogenetic analyses focus on the phylogeny of individual homologous genes. Different genes can give rise to different *gene trees*, which may be partially or entirely inconsistent with the *species tree*—a phylogenetic tree which aims to depict the overall lineage relationships among the given taxa [66].

7.1.6 Phylogenetic Trees

In evolutionary biology, the relationships among a group of organisms are represented by a phylogenetic tree. Given a set of taxa $\mathcal{X}$, a phylogenetic tree T on $\mathcal{X}$ is a connected, undirected, acyclic graph $T = (V, E)$, where V is a set of vertices (nodes), which all have degree $\neq 2$, and E is a set of edges connecting the nodes. A taxon labelling $\lambda : \mathcal{X} \to V$ assigns exactly one taxon to every leaf node and none to any internal node [106]. *Cladograms* (Figure 7.1a) display the branching patterns of evolutionary relationships but do not represent the amount of evolutionary change; their branch lengths are arbitrary. A weight, or length, may be assigned to the edges, indicating the number of mutations that have happend along an edge or representing the evolutionary time that has passed. *Phylograms* (Figure 7.1b) incorporate these branch lengths in their visualizations of phylogenetic trees [201].

In general, phylogenetic trees can be either rooted or unrooted. *Rooted trees* (for example, Figure 7.1a) have a root node representing the ancestor from which all other nodes descent. Therefore, rooted trees have a direction, corresponding to evolutionary time. *Unrooted trees* (Figure 7.1c) lack a root node and do not represent ancestor–descandant relationships. However, the edge lengths in unrooted trees still represents genetic distance or evolutionary time. [201]

7.2 Machine Learning Classification Methods

In statistics, classification algorithms predict one of several (discrete) outcomes (i.e., classes) for an observation based on its features. Classifiers are trained on a set of training samples for which the class label is known and which are used to construct the prediction rule (supervised learning). In this section, input variables—scalars or vectors—are denoted as X, quantitative outputs as Y, and qualitative outputs as G. Observed values of these variables are written in lowercase. A set of N input p-vectors is represented by $\boldsymbol{X} \in \mathbb{R}^{N \times p}$. The columns of this matrix—all N observations for the variable X_j—are written in bold as $\boldsymbol{x}_j$, while the rows of this matrix—all variables of one observation—are denoted as

$x_i^\intercal$. Thus, classification refers to the prediction of the output G, denoted by $\hat{G}$, where $\hat{G} \in \mathcal{G}$ (the set of classes), given the input vector X. In the following sections, we will introduce some classification methods. A more detailed description can be found in [86]. Figure 7.2 to Figure 7.5 show the decision boundaries of the introduced classifiers on an example dataset. The default models of python's `scikit-learn` [213] were used.

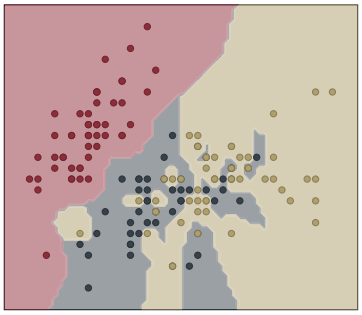


Figure 7.2: Decision boundaries of a KNN classifier ($k = 1$). Regions of the feature space are colored by the class that would be assigned to test samples falling in that region. Training samples are shown as scatterpoints.

7.2.1 k-Nearest Neighbors

The k-nearest neighbor (KNN) classifier [46, 67] is a memory-based model that does not need to be trained. For a given test point x_0, its k nearest training samples are identified based on a distance function (e.g., Euclidean distance). x_0 is classified based on a majority vote of the k nearest neighbors for which the class label is known. The parameter k is a hyperparameter to be optimized. Figure 7.2 shows the decision boundaries of a KNN model ($k = 1$) of a 3-class classification problem.

7.2.2 Support Vector Machines

Support vector machines (SVMs, [45]) in their original form are binary linear classifiers that construct a hyperplane that separates two classes such that the margin—the distance between the hyperplane and the nearest training sample of both classes—is maximal. A large margin is chosen to minimize the generalization error of the classifier. Given a set of training data $(x_1, y_1), \ldots, (x_N, y_N)$, where $x_i \in \mathbb{R}^p$ and $y \in \{-1, 1\}$ and the definition of a hyperplane

$$\{x : f(x) = x^\intercal \beta + \beta_0 = 0\}, \tag{7.1}$$

where $||\beta|| = 1$. The classification rule is given by

$$G(x) = \text{sign}[x^\intercal \beta + \beta_0], \tag{7.2}$$

since $f(x)$ gives the signed distance of x to the hyperplane. For a linearly separable dataset, $f(x)$ can be defined such that $y_i f(x_i) > 0 \, \forall i$. Finding the hyperplane with the biggest margin M is formalized by the following optimization problem:

$$\max_{\beta, \beta_0, ||\beta||=1} M \tag{7.3}$$

$$\text{subject to} \quad y_i(x_i^\intercal \beta + \beta_0) \geq M, \, \forall i, \tag{7.4}$$

which can be rephrased to

$$\min_{\beta, \beta_0} ||\beta|| \tag{7.5}$$

$$\text{subject to} \quad y_i(x_i^\intercal \beta + \beta_0) \geq 1, \, \forall i. \tag{7.6}$$

For data that is not linearly separable (see for example Figure 7.3a) we need to introduce the slack variables $\xi = \{\xi_1, \ldots, \xi_N\}$ that measure the proportional amount by which the prediction $f(x_i)$ is on the wrong side

of its margin. The constraint (Equation (7.4)) is modified to

$$y_i(x_i^\intercal \beta + \beta_0) \geq M(1 - \xi_i), \forall i, \tag{7.7}$$

$\forall i$, $\xi_i \geq 0$, and $\sum_{i=1}^{N} \xi_i \leq K$, where K is a constant that binds the number of misclassifications. The constrained optimization problem can be equivalently expressed by

$$\min_{\beta,\beta_0} \frac{1}{2}||\beta||^2 + C \sum_{i=1}^{N} \xi_i \tag{7.8}$$

$$\text{subject to} \quad \begin{cases} y_i(x_i^\intercal \beta + \beta_0) \geq 1 - \xi_i, \forall i, \\ \xi_i \geq 0, \end{cases} \tag{7.9}$$

where the cost parameter C replaces K.

The minimum of the quadratic function subject to inequality constraints can be found using the Lagrange multiplier and the Karush–Kuhn–Tucker (KKT) conditions [116, 133]. By solving the Lagrangian dual of the above problem (for details see [86]), it is simplified to

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} \alpha_i \alpha_{i'} y_i y_{i'} x_i^\intercal x_{i'} \tag{7.10}$$

$$\text{subject to} \quad \begin{cases} \sum_{i=1}^{N} \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C, \end{cases} \tag{7.11}$$

which can be solved by quadratic programming algorithms. Once the optimal Lagrange multipliers $\hat{\alpha}_i$ are obtained, $\hat{\beta}$ is given by

$$\hat{\beta} = \sum_{i'=1}^{N} \hat{\alpha}_i y_i x_i, \tag{7.12}$$

where $\hat{\alpha}_i = 0$ for observations that lie on the correct side of the margin, $0 < \hat{\alpha}_i < C$ for observations that lie on the margin boundary ($\xi_i = 0$), and $\hat{\alpha}_i = C$ for the remainder ($\xi_i > 0$). Observations x_i for which $\hat{\alpha}_i \neq 0$ are called support vectors. Assume x_s to be a support vector that lies on the edge of the margin and x_s its corresponding class label, $\hat{\beta}_0$ can then be derived as

$$\hat{\beta}_0 = y_s - \sum_{i'=1}^{N} \alpha_i y_i x_i^\intercal x_s. \tag{7.13}$$

Kernel Trick

To learn a nonlinear decision boundary, linear classifiers can be applied to feature vectors obtained by feature mapping $\varphi(x_i)$, that is mapping the data x_i in a higher-dimensional feature space, where the data is linearly separable. When applying the kernel trick [28], the data does not need to be explicitly transformed into feature vectors but requires only a kernel function

$$K(x, x') = \langle \varphi(x) \, , \, \varphi(x') \rangle \tag{7.14}$$

that computes inner products in the transformed space. A real-valued function $K(x, x')$ is a valid kernel function if and only if it is a symmetric and positive semi-definite function. These conditions are known as Mercer's conditions [178].

The solution function $\hat{f}(x)$ can be written as

$$\hat{f}(x) = \varphi(x)^\intercal \hat{\beta} + \hat{\beta}_0 \tag{7.15}$$

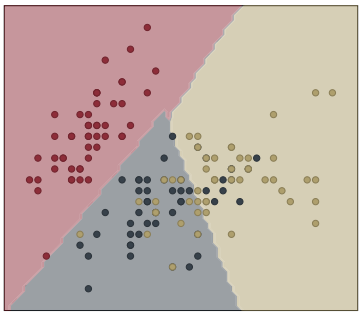
$$= \sum_{i'=1}^{N} \hat{\alpha}_i y_i \varphi(x)^\intercal \varphi(x_i) + \hat{\beta}_0, \tag{7.16}$$

$$= \sum_{i'=1}^{N} \hat{\alpha}_i y_i K(x, x_i) + \hat{\beta}_0. \tag{7.17}$$

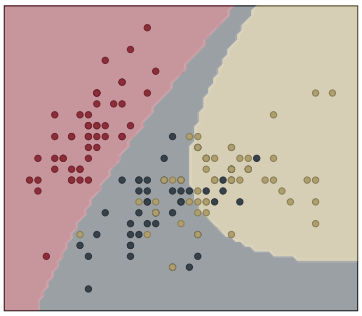
Popular choices for K in SVMs are:

$$\begin{aligned} \text{linear:} \quad & K(x, x') = xx', \\ d\text{th-degree polynomial:} \quad & K(x, x') = (\gamma xx' + c)^d, \\ \text{radial basis function (rbf):} \quad & K(x, x') = \exp(-\gamma ||x - x'||^2) \end{aligned}$$

Figure 7.3b shows the decision boundary of an SVM classifier using a polynomial kernel of degree $d = 3$.



(a)



(b)

Figure 7.3: Decision boundaries of a SVM classifier using a linear **(a)** and a cubic polynomial kernel **(b)**, respectively. Regions of the feature space are colored by the class that would be assigned to test samples falling in that region. Training samples are shown as scatterpoints.

7.2.3 Neural Networks

In this section, we introduce single-layer perceptrons [236], the simplest form of feedforward neural networks. Perceptrons consist of an input layer, one hidden layer, and an output layer. For a K-class classification problem, the probability for each class k is modeled by one output neuron, $Y_k \in [0, 1]$. The number of input neurons equals the number of features, p, that describe the samples. The M derived features in the hidden layer, Z_m, are computed as a linear combination of the inputs:

$$Z_m = \sigma(w_{0m} + w_m^\intercal X), \tag{7.18}$$

where w_m is a vector of weights for the connections between the inputs and the corresponding feature Z_m, w_{0m} is an additional bias, and σ is an activation function—typically the sigmoid function[4].

The output Y_k for each class k is computed as a linear combination of the features Z_m,

$$T_k = w_{0k} + w_k^\intercal Z, \tag{7.19}$$

$$Y_k = g_k(T) \tag{7.20}$$

where w_k is a vector of weights for the connections between the derived features $Z = (Z_1, \ldots, Z_M)$ and the corresponding output Y_k, w_{0k} is an additional bias, and g_k is the output function—typically the softmax function[5].

In general, neural networks can consist of more than one hidden layer (multilayer perceptrons), allowing the model to discern data that is not linearly separable.

4: The sigmoid function is defined as

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

5: The softmax function is defined as

$$g_k(T) = \frac{e^{T_k}}{\sum_{l=1}^{K} e^{T_l}}$$

Learning the unknown parameters of the neural network—the weights and biases—from the training data is an iterative process. After processing each sample from the training data, the weights are updated based on the discrepancy between the predicted output and the expected output. A loss function this error, providing a numerical measure of the discrepancy. An optimization algorithm, often involving gradient descent [38], adjusts the model parameters in a direction that minimizes this error. Gradient descent involves the computation of the gradient of the loss function with respect to the weights and biases of the neural network for a single input–output example. The gradients are efficiently computed using the backpropagation algorithm [121, 237], which repeatedly applies the chain rule, propagating the error backward through the neural network. Backpropagation in its modern version is also known as a reverse mode of automatic differentiation [150, 151] (see Section 4.3.2). For more information on neural networks, see for example [78].

Figure Figure 7.4 shows the decision boundaries of a multi-layer perceptron classifier of a 3-class classification problem.

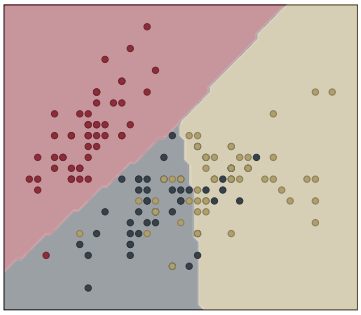


Figure 7.4: Decision boundaries of a multi-layer perceptron classifier with 100 hidden layers. Regions of the feature space are colored by the class that would be assigned to test samples falling in that region. Training samples are shown as scatterpoints.

7.2.4 Decision Trees

Decision trees are a non-parametric classification method, that recursively partitions the feature space into two (or more) subspaces according to a variable's value (see for example Figure 7.5) [31]. The objective is to obtain partitions that contain samples from only one class. Recursive partitions can be represented as trees, hence their name. Starting from the root, internal nodes split the feature space into two subspaces, and leaf nodes are assigned to the most appropriate class label, e.g. the majority vote of all samples within the respective subspace. Training decision trees is a matter of constructing the tree (top-down) and deciding on the split variables and points. To find an optimal split during training, the quality of different splits is evaluated by an impurity measure. Two popular choices are the Gini impurity [31] and the information gain [227]. The feature space is split recursively until a stopping criterion is met.

Decision tree classifiers are popular for their simplicity and interpretability.

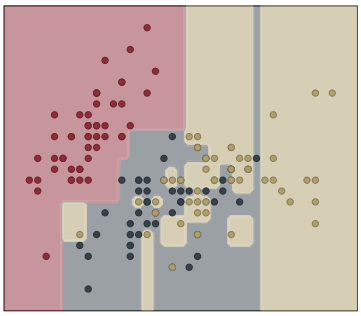


Figure 7.5: Decision boundaries of a decision tree classifier. Regions of the feature space are colored by the class that would be assigned to test samples falling in that region. Training samples are shown as scatterpoints.

Part II.

Uncertainty Propagation in Dimensionality Reduction

VIPurPCA: Visualizing and Propagating Uncertainty in Principal Component Analysis

8.

Disclaimer 8.1 Chapter 8 is based on the peer-reviewed publication with the following co-author contributions:

S. Zabel, P. Hennig, and K. Nieselt. "VIPurPCA: Visualizing and Propagating Uncertainty in Principal Component Analysis". *IEEE Transactions on Visualization and Computer Graphics* (2023) [291]

	Ideas	Analysis & Interpretation	Code	Writing
S. Zabel	33.3 %	80 %	80 %	85 %
P. Hennig	33.3 %	10 %	20 %	10 %
K. Nieselt	33.3 %	10 %	0 %	5 %

Abstract

Variables obtained by experimental measurements or statistical inference typically carry uncertainties. When an algorithm uses such quantities as input variables, this uncertainty should propagate to the algorithm's output. Concretely, we consider the classic notion of principal component analysis (PCA): If it is applied to a finite data matrix containing imperfect (i.e. uncertain) multidimensional measurements, its output—a lower-dimensional representation—is itself subject to uncertainty. We demonstrate that this uncertainty can be approximated by appropriate linearization of the algorithm's nonlinear functionality, using automatic differentiation. By itself, however, this structured, uncertain output is difficult to interpret for users. We provide an animation method that effectively visualizes the uncertainty of the lower dimensional map. Implemented as an open-source software package, it allows researchers to assess the reliability of PCA embeddings.

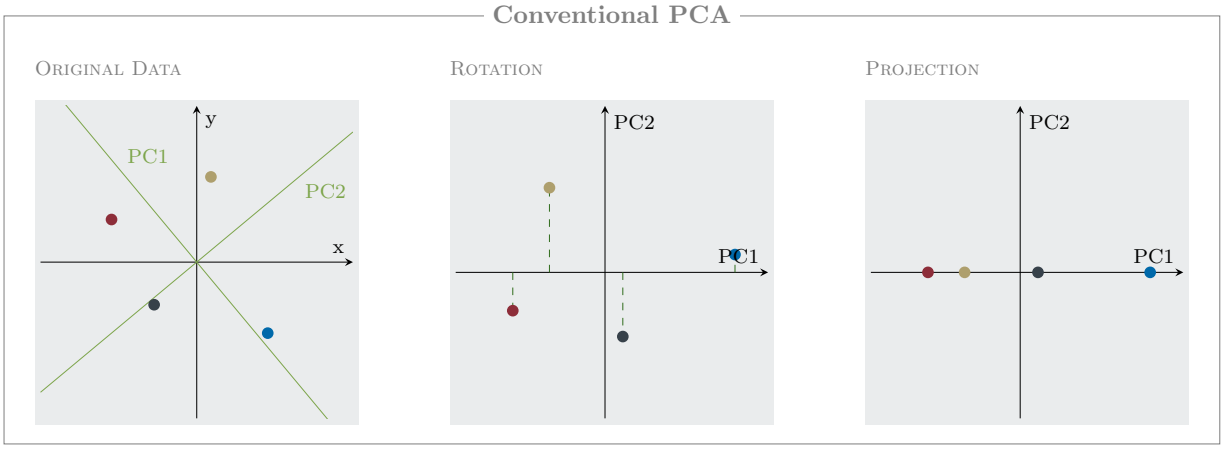
8.1 Introduction

High-dimensional data is encountered in all fields of science. In medicine, the health status of a patient is described by over hundreds of measured records [179], in bioinformatics, high-throughput methods provide large-scale omics data [85], in computer vision, high-resolution images of thousands of pixels are processed for image recognition [153], to just name some prominent examples. The availability of these large data sets provides lots of potential for data analysts, but at the same time raises big challenges. Working on high-dimensional datasets is often limited by computational constraints, therefore, dimensionality reduction techniques are commonly applied. These methods project the data to a lower dimensional subspace, such that properties of the data in high-dimensional space are preserved in their low-dimensional representation.

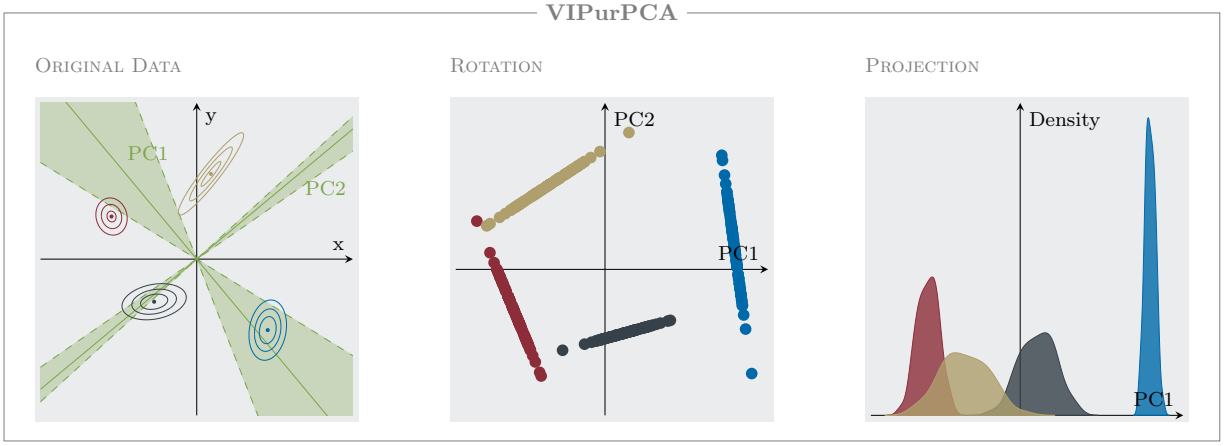
Code, experiments, and animated visualizations are available at the Github repository `Integrative-Transcriptomics/VIPurPCA`, and at `https://integrative-transcriptomics.github.io/VIPurPCA/examples/`

Methods for dimensionality reduction can be divided into linear and nonlinear approaches. The most prominent linear dimensionality reduction is principal component analysis (PCA; [209]), which produces new features as linear combinations of the original variables (Section 2.1). The respective coefficients are formed by a new set of orthonormal basis vectors, which are called principal components (PCs). The PCs point into orthogonal directions of maximum variance in the data and can be found by computing an eigenvalue decomposition of the data's covariance matrix (Figure 8.1a). Data is transformed and projected according to the principal components. Dimensions are reduced by only taking a small subset of all principal components into account. However, conventional PCA is unaware of uncertainty as it uses finite input sets and results in a definite low-dimensional map. It is not common practice to analyze the stability of the reducing map with respect to (minor) imprecisions in the data (uncertainty analysis) or to evaluate how much each input contributes to the output uncertainty (sensitivity analysis). PCA is often applied for a general visual overview of the data but is also used to identify characteristics in the data that explain certain patterns (e.g. clusters), like for example in the field of archeogenomics [221]. Therefore, PCA influences the user's reasoning early on in the data analysis pipeline. Hence, it is relevant to incorporate uncertainty information if available to improve the interpretability of the low-dimensional map, to prevent invalid conclusions, as well as to strengthen robust evidence.

As seen in Figure 8.1b, the uncertainty of the input highly influences the outcome of the PCA. Therefore, we developed a method for visualizing and propagating uncertainty in PCA (VIPurPCA, pronounced 'vip your PCA'). As an input, VIPurPCA expects a dataset carrying uncertainties obtained by experimental measurements or statistical estimates (Figure 8.1b). The output uncertainty is approximated by linearizing PCA's nonlinear functionality to allow for Gaussian error propagation. Here, nonlinearity refers to the computation of the eigenvectors, which is nonlinear in the input. Derivatives as part of the linearization are efficiently computed using automatic differentiation and represent the influence of individual data points on the principal components. In this work, uncertainty is modeled probabilistically which is difficult to visualize explicitly. Hence, we have developed an intuitive visualization of the stability of the lower dimensional map itself and integrated it into VIPurPCA: Equipotential samples drawn along an orbit of the probability distribution build frames of an animation of the resulting lower dimensional map. This visualization technique provides valuable insights into the amount and structure of the uncertainty of PCA's output. In this work, we mathematically formulate error propagation through PCA and show its computational advantage compared to iterative Monte Carlo sampling. We also introduce a method for visualizing embedding uncertainties, which provides additional value to alternative visualizations of uncertain embeddings. In three real-world examples, we show how different sources of input uncertainty influence the stability of the PCA embedding and how this is visually identified using our visualization approach.



(a)



(b)

Figure 8.1: Comparison of conventional PCA **(a)** and VIPurPCA **(b)**. **(a)** Left: Scatterplot of four two-dimensional samples used as input for PCA. The directions of the two computed principal components (PCs) are shown in green. Middle: Dataset in terms of the new coordinate system. Right: Dimensionality reduction by projecting the data onto the first PC. **(b)** Left: Input data with (correlated) Gaussian uncertainties are used as input to VIPurPCA, which propagates the uncertainty to the PCA's output. The mean and 1σ-interval of the computed PCs are shown in green. 100 samples from the computed distribution over the principal components are used to transform (middle) and project (right) the data. The distribution of the one-dimensional projections is represented by a probability density to reduce visual clutter.

8.2 Related Work

There exists a large variety of dimensionality reduction techniques, for reviews see for example [141, 194, 243, 274]. Traditionally, linear techniques like principal component analysis (PCA) [209], factor analysis [252] and classic multidimensional scaling [270] were used, which more recently have been complemented by a large number of nonlinear techniques to handle complex nonlinear data [274]. PCA is in fact the most popular linear dimensionality technique. It computes a new set of features (principal components) as a linear combination of the original variables, such that the amount of variance in the data is maximal when reducing its dimensions. In this work, we extend PCA to handle input data containing known correlated Gaussian errors, which are propagated to the algorithm's output. Methods like factor analysis or probabilistic principal component analysis (PPCA) [269] are generalizations of PCA and model the observed variables as linear combinations of latent factors and added Gaussian noise. This Gaussian noise is unknown, uncorrelated, has zero mean, is even isotropic for PPCA, and is estimated by the method. Similarly, we also assume the observed variables to have observational errors, which however can be correlated, are known, and are entered as input by the user. Recently, Görtler et al. [80] proposed an uncertainty-aware version of PCA working on probability distributions rather than a set of input points. In this method, a closed-form solution for the covariance matrix of the input distributions is provided which is then fed to the standard PCA procedure. This procedure resembles sampling from the input distribution and computing the PCA on the concatenated set of samples. Opposed to that, our method propagates the uncertainty to PCA's final output. Despite being an approximation our result converges to the ground truth obtained by Monte-Carlo-based uncertainty propagation. Görtler et al. [80] visualize the uncertainty in the low-dimensional map by showing samples drawn from their propagated distributions as overlayed scatter points. In this work, we suggest that in some scenarios the overlay visualization is less appropriate and propose instead to visualize showing samples in an animation.

There have been many attempts to incorporate uncertainty and sensitivity analysis into visual analytics tools, i.e., understanding high-dimensional data via low-dimensional representations, to increase the interpretability of the resulting visualizations. One approach adds sensitivity information in terms of flow-lines to the scatterplot to give insights into how one variable changes with respect to another variable in the original space. This approach was implemented in flow-based scatterplots [40] and Generalized Sensitivity Scatterplots [41] using derivatives to determine the sensitivities. Rather than showing sensitivities of one variable with respect to another variable, Faust et al. [64] display sensitivities of projected data with respect to the original data in a tool called DimReader. Using automatic differentiation to compute derivatives they show how infinitesimal perturbation in the data affect the outcome of nonlinear projections. Isolines of the scalar field spanned by these derivatives are computed indicating how projected points move if the input was perturbed in a specific way. Their proposed method focuses on the visualization of the sensitivity analysis considering input parameters as independent and perturbations not to follow any probability distribution, which gives an intuition on how perturbing an input affects the output.

VIPurPCA also provides sensitivity analysis in form of derivatives, but in addition, quantifies the uncertainty of model outputs based on the uncertainty of model inputs and provides an eligible visualization of the output uncertainty.

As the focus of this work is on information visualization, related work on uncertainty visualization for scientific visualizations such as isosurfaces [10, 222] or volume rendering [11] is not covered in detail.

Communicating uncertainties visually is a big challenge [254]. Levontin and Walton [144] recently reviewed available approaches to tackle this task and remaining challenges. There are a variety of approaches to classify uncertainty visualization methods [53, 124, 172, 250]. In this work, uncertainty is modeled using a probabilistic approach which requires the visualization of distributions (Figure 6.1). Visualizing high-dimensional distributions explicitly (e.g., plotting a frequency or probability) is not feasible. Alternatively, implicit approaches are available that explore the distribution via samples drawn from the distribution. [285]. Those potential realizations are visualized in a second step for which different views exist. Individual samples can be overlaid, aggregated in the form of a density representation [246], shown in hypothetical outcome plots [103] one after another in an animation [293] or next to each other as small multiples. In this work, we discuss the advantages and disadvantages of the visualization approaches applied to uncertain low-dimensional maps. Additionally, we implement an animation approach that provides smooth transitions between individual frames.

Introduced by Hennig [87] on animating samples from Gaussian processes, we adopted the approach to visualize the uncertainty of the PCA result (i.e., the distribution over the eigenvectors). We use time to traverse through trajectories of equipotential samples of the output distribution, where at each time point a sample is drawn and used to project the input data accordingly. By that, the visualization provides an intuition about the stability of the outcome in terms of the extent of motion of the points in the two-dimensional map. Furthermore, it gives valuable information about the potential structure of samples in contrast to static visualization of sample distributions (e.g. density plots).

8.3 Problem Setting

Let the high-dimensional input data be represented by a matrix $\boldsymbol{Y} \in \mathbb{R}^{N \times D}$, where N is the number of samples and D is the number of features. We assume this input data is uncertain. We model its vectorized form, $\boldsymbol{y} = \mathrm{vec}(\boldsymbol{Y}) \in \mathbb{R}^{ND}$, as a Gaussian random vector:

$$p(\boldsymbol{y}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{y}_0, \boldsymbol{\Sigma}_y), \tag{8.1}$$

where $\boldsymbol{y}_0 = \mathrm{vec}(\boldsymbol{Y}_0)$ is the mean of the high-dimensional data (typically, $\boldsymbol{Y}_0$ is assumed to be centered or is centered as a first step in PCA), and $\boldsymbol{\Sigma}_y \in \mathbb{R}^{ND \times ND}$ is its covariance matrix.

Principal Component Analysis, when applied to a data matrix (e.g., the mean $\boldsymbol{Y}_0$), computes a set of M principal components, denoted by the columns of a matrix $\boldsymbol{W}_0 \in \mathbb{R}^{D \times M}$. These principal components are the

eigenvectors of the sample covariance matrix $\boldsymbol{S} = \frac{1}{N-1}\boldsymbol{Y}_0^\intercal \boldsymbol{Y}_0$ and form an orthonormal basis for the principal subspace (Section 2.1).

The primary goal of VIPurPCA is to estimate the distribution of these principal components, $\boldsymbol{W}$ (or its vectorization $\boldsymbol{w} = \text{vec}(\boldsymbol{W}) \in \mathbb{R}^{DM}$), that arises from the uncertainty in the input data $\boldsymbol{y}$. A secondary goal is to visualize the impact of this uncertainty on the stability of the low-dimensional map $\boldsymbol{Z} = \boldsymbol{Y}\boldsymbol{W}$.

8.4 Theory & Implementation

8.4.1 Gaussian Error Propagation for PCA

We can frame PCA as a deterministic function f that maps an input data vector $\boldsymbol{y}$ to the corresponding M principal components (eigenvectors) $\boldsymbol{w}$:

$$\boldsymbol{w} = f(\boldsymbol{y}).$$

This function $f : \mathbb{R}^{ND} \to \mathbb{R}^{DM}$ is inherently *nonlinear* due to the computation of the eigenvectors via, for example, singular value decomposition of the input data matrix $\boldsymbol{Y}$. However, a crucial property of the PCA algorithm is that it can be formulated as such an *explicit function*. This explicit nature allows us to directly analyze the sensitivity of the output to the input by computing the derivatives of f. We therefore proceed by approximating the output distribution by linearizing this function around the mean input $\boldsymbol{y}_0$ using a first-order Taylor expansion (as detailed in Section 3.2.1):

$$\boldsymbol{w} = f(\boldsymbol{y}) \approx f(\boldsymbol{y}_0) + \left.\frac{\partial f(\boldsymbol{y})}{\partial \boldsymbol{y}}\right|_{\boldsymbol{y}_0} (\boldsymbol{y} - \boldsymbol{y}_0). \tag{8.2}$$

This linearization implies that if the input $\boldsymbol{y}$ is Gaussian, the output $\boldsymbol{w}$ will be approximately Gaussian. The mean of this distribution is the PCA result on the mean input, $\boldsymbol{w}_0 = f(\boldsymbol{y}_0)$, and its covariance matrix $\boldsymbol{\Sigma}_w \in \mathbb{R}^{DM \times DM}$ is given by the law of Gaussian error propagation:

$$\boldsymbol{\Sigma}_w \approx \left(\boldsymbol{J}_y f(\boldsymbol{y}_0)\right) \boldsymbol{\Sigma}_y \left(\boldsymbol{J}_y f(\boldsymbol{y}_0)\right)^\intercal, \tag{8.3}$$

where $\boldsymbol{J}_y f(\boldsymbol{y}_0) = \left.\frac{\partial f(\boldsymbol{y})}{\partial \boldsymbol{y}}\right|_{\boldsymbol{y}_0}$ is the Jacobian of the PCA mapping evaluated at the mean input. The approximate distribution of the vectorized principal components is therefore:

$$p(\boldsymbol{w} \mid \boldsymbol{y}) \approx \mathcal{N}(\boldsymbol{w}; \boldsymbol{w}_0, \boldsymbol{\Sigma}_w). \tag{8.4}$$

The covariance $\boldsymbol{\Sigma}_w$ quantifies the uncertainty in these principal components, derived from the input uncertainty $\boldsymbol{\Sigma}_y$ and the sensitivity encoded in the Jacobian $\boldsymbol{J}_y f$. This Jacobian can be computed efficiently using automatic differentiation (Chapter 4). The sensitivities of the eigenvalues can be computed analogously. However, we focus on the eigenvectors as they directly determine the projection.

8.4.2 Enhancing Efficiency using JVPs and VJPs

The Jacobian matrix $\boldsymbol{J}_y f(\boldsymbol{y}_0) \in \mathbb{R}^{DM \times ND}$ can be very large. Explicitly forming this matrix and then performing the multiplications in Equation (8.3) is often computationally prohibitive. Similarly, if $\boldsymbol{\Sigma}_y = \boldsymbol{D}_0 \otimes \boldsymbol{N}_0$, forming the full $ND \times ND$ matrix $\boldsymbol{\Sigma}_y$ might be prohibitive.

To circumvent this, we compute the output covariance matrix $\boldsymbol{\Sigma}_w$ one column at a time using matrix-free methods. The k-th column of $\boldsymbol{\Sigma}_w$ can be computed by multiplying the full matrix by the k-th standard basis vector $\boldsymbol{e}_k \in \mathbb{R}^{DM}$:

$$\boldsymbol{\Sigma}_w[:, k] = \Big(\boldsymbol{J}_y f(\boldsymbol{y}_0)\Big)\, \boldsymbol{\Sigma}_y \Big(\boldsymbol{J}_y f(\boldsymbol{y}_0)\Big)^\intercal \boldsymbol{e}_k. \tag{8.5}$$

This is computed via a right-to-left sequence of vector-matrix products[1], which can be implemented efficiently using Jacobian-vector products (JVPs) and vector-Jacobian products (VJPs) provided by AD frameworks as introduced in Section 4.3:

$$\boldsymbol{\Sigma}_w[:, k] = \Big(\boldsymbol{J}_y f(\boldsymbol{y}_0)\Big)\, (\boldsymbol{D}_0 \otimes \boldsymbol{N}_0) \overbrace{\Big(\boldsymbol{J}_y f(\boldsymbol{y}_0)\Big)^\intercal \cdot \boldsymbol{e}_k}^{\text{VJP}}}^{\boldsymbol{v}_a} \tag{8.6}$$

$$= \Big(\boldsymbol{J}_y f(\boldsymbol{y}_0)\Big) \overbrace{(\boldsymbol{D}_0 \otimes \boldsymbol{N}_0) \cdot \boldsymbol{v}_a}^{\text{Equation (3.23)}} \tag{8.7}$$

$$= \Big(\boldsymbol{J}_y f(\boldsymbol{y}_0)\Big) \overbrace{\mathrm{vec}(\boldsymbol{D}_0 \cdot (\mathrm{vec}^{-1}(\boldsymbol{v}_a)) \cdot \boldsymbol{N}_0^\intercal)}^{\text{matrix product}}}^{\boldsymbol{v}_b} \tag{8.8}$$

$$= \overbrace{\Big(\boldsymbol{J}_y f(\boldsymbol{y}_0)\Big) \cdot \boldsymbol{v}_b}^{\text{JVP}}}^{\boldsymbol{v}_c} \tag{8.9}$$

$$= \boldsymbol{v}_c \tag{8.10}$$

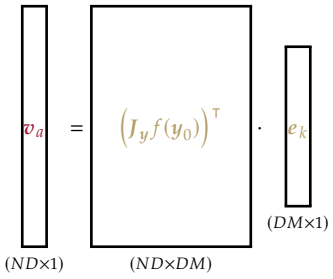
This sequence allows us to construct $\boldsymbol{\Sigma}_w$ column-by-column without ever instantiating the full Jacobian $\boldsymbol{J}_y f$ or the full $\boldsymbol{\Sigma}_y$. If $\boldsymbol{\Sigma}_y$ is not Kronecker-structured (e.g., a full or diagonal matrix), the computation of $\boldsymbol{v}_b = \boldsymbol{\Sigma}_y \boldsymbol{v}_a$ is adapted accordingly.

8.4.3 Implementation

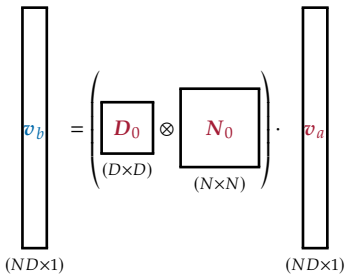
We implemented VIPurPCA using JAX (Section 4.4) for its automatic differentiation capabilities. The core of the implementation is a JAX-compatible function that computes the PCA mapping, $\boldsymbol{w} = f(\boldsymbol{y})$. This function finds the principal axes (eigenvectors $\boldsymbol{W}$) via a Singular Value Decomposition (SVD) of the centered input matrix $\boldsymbol{Y}_0$.

Figure 8.2 outlines the key steps of the implementation. The core function `f` implements the PCA mapping (Figure 8.2a). It takes the matrix form of the data $\boldsymbol{Y}$ as input for simplicity in this example, but a robust

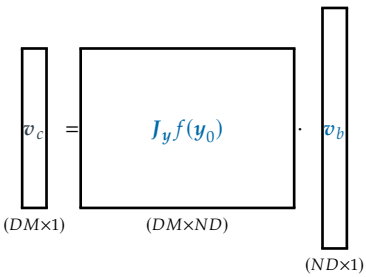
1: Diagrams depicting matrix dimensions: For $\boldsymbol{v}_a = \Big(\boldsymbol{J}_y f(\boldsymbol{y}_0)\Big)^\intercal \boldsymbol{e}_k$:



For $\boldsymbol{v}_b = \boldsymbol{\Sigma}_y \boldsymbol{v}_a$, assuming $\boldsymbol{\Sigma}_y = \boldsymbol{D}_0 \otimes \boldsymbol{N}_0$:



For $\boldsymbol{\Sigma}_w[:, k] = \Big(\boldsymbol{J}_y f(\boldsymbol{y}_0)\Big)\, \boldsymbol{v}_b$:



```python
import jax.numpy as np

def f(Y, m):
    '''Computes the first M eigenvectors.
        Y_matrix is an N x D matrix, assumed centered.
        Returns an M x D matrix of eigenvectors.'''

    # SVD of Y
    _, _, W = np.linalg.svd(Y, full_matrices=False)

    return W[0:m, :]
```

(a) Function to be differentiated

```python
from jax import vjp, linearize
from functools import partial

# Lambda expression fixing the second argument
fun = partial(f, m=2)

# vector-Jacobian product
_, f_vjp = vjp(fun, Y)

# Jacobian-vector product
_, f_jvp = linearize(fun, Y)
```

(b) Function transformations

```python
from jax import nn.one_hot

def cvp(f_jvp, f_vjp, D, N, d, n, m, i):
    '''Computes a J(kron(D, N))J.T - vector product'''

    e_k = nn.one_hot([i], d*m)
    v_a = f_vjp(e_k)[0]
    v_b = np.ravel((N @ np.reshape(v_a, (n, d), 'C')
       @ D), 'C')
    return f_jvp(v_b)
```

(c) Function to compute $\Sigma_w[i]$

```python
from jax import vmap

# Lambda expression fixing all arguments but
    index i
f_cvp = lambda i: cvp(f_jvp, f_vjp, D, N, d, n,
      m, i)

# Compute Sigma
Sigma = np.vstack([vmap(f_cvp)(b) for b in
    batches])
```

(d) Batching & looping

Figure 8.2: Simplified JAX implementation to compute the covariance matrix Σ_w of the principal components. Variable names are chosen to reflect their mathematical counterparts (e.g., `Y_matrix` $\rightarrow \boldsymbol{Y}$, `m_components` $\rightarrow M$). **(a)** PCA function computing principal axes via SVD. **(b)** Obtaining JAX function transformations for VJPs and JVPs of the PCA function. **(c)** Function implementing the sequence of JVP/VJP operations to compute one column of Σ_w. **(d)** Using batching and looping to compute the full covariance matrix Σ_w. Note that the definition of `batches`—each batch is a set of indices—is not shown.

implementation would handle the vectorized form $\boldsymbol{y}$ to align with our mathematical setup. We use JAX's function transformations (`vjp` for vector-Jacobian products and `linearize` for Jacobian-vector products) to obtain efficient, matrix-free functions for computing products with the Jacobian $\boldsymbol{J}_{\boldsymbol{y}} f(\boldsymbol{y}_0)$ and its transpose (Figure 8.2b). The function `cvp` implements the right-to-left sequence of operations from Equation (8.5) to compute a single column of the final covariance matrix Σ_w (Figure 8.2c). The full covariance matrix Σ_w is constructed by applying `vmap` to efficiently map the `cvp` function over the standard basis vectors. For efficiency, this is often done in batches over a set of indices, balancing memory usage and computational parallelism. The choice of batch size depends on available memory. This JAX implementation is primarily optimized for CPU execution. For larger datasets, JAX's `pmap` transformation can be used for parallel execution across multiple GPUs or TPUs, potentially offering memory and speed advantages over `vmap` for very large numbers of output dimensions.

In this section, we have established a method to propagate input uncertainty through the explicit PCA function f by linearizing it around the mean input. This yields an approximate Gaussian distribution over the principal components, $\mathcal{N}(\boldsymbol{w}; \boldsymbol{w}_0, \Sigma_w)$, where the mean $\boldsymbol{w}_0$ is the standard PCA result and the covariance Σ_w quantifies the resulting uncertainty. Furthermore, we outlined an efficient, matrix-free implementation using JAX. Having quantified this uncertainty mathematically, the next section addresses the crucial challenge of visualizing its impact on the final low-dimensional map to enable robust scientific interpretation.

8.5 Uncertainty Visualization of the Low-Dimensional Map

Having derived the approximate Gaussian distribution over the principal axes,

$$p(\boldsymbol{w} \mid \boldsymbol{y}) \approx \mathcal{N}(\boldsymbol{w}; \boldsymbol{w}_0, \boldsymbol{\Sigma}_w),$$

this section discusses strategies for visualizing its impact on the low-dimensional data projections. Understanding user objectives when performing uncertainty-aware PCA guides our visualization design. Typically, PCA is used for:

1. Visualizing data as a 2D scatterplot to get an overview, check for quality (e.g., replicate clustering), or identify patterns.
2. Pre-processing for dimensionality reduction before applying other algorithms (e.g., clustering) to mitigate the curse of dimensionality.

While both use cases benefit from uncertainty awareness, the first explicitly requires visualization of this uncertainty. Key user questions include:

- What is the overall stability of the low-dimensional map? How trustworthy are conclusions drawn from it?
- How stable are specific structures like clusters or trajectories observed in the map?
- How stable are the relative positions of individual samples? Do samples exhibit correlated or uncorrelated uncertainties affecting their pairwise distances?

These questions pertain to the positions and relationships of data points in the projected scatterplot. As introduced in Section 6.2, explicit visualization of the high-dimensional distribution $p(\boldsymbol{w} \mid \boldsymbol{y})$ is infeasible. Instead, we use implicit visualization by examining the effect of the uncertainty on the projected data $\boldsymbol{Z} = \boldsymbol{Y}\boldsymbol{W}$. We draw n samples $\boldsymbol{w}_i \sim p(\boldsymbol{w} \mid \boldsymbol{y})$, reshape them into matrices $\boldsymbol{W}_i$, and generate a set of n corresponding low-dimensional embeddings $\boldsymbol{Z}_i = \boldsymbol{Y}_0\boldsymbol{W}_i$. This set of potential embeddings $Z = \{\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_n\}$ can be visualized using various approaches like overlay, animation, or small multiples (as introduced in Figure 6.1). Section 8.6.3 further discusses these options.

While these methods are all valid, animation presents a unique challenge and opportunity. Displaying independently drawn random samples $\boldsymbol{Z}_i$ as frames can result in a jerky, disconnected visual experience, making it difficult to perceive stable structures and correlated movements. To create a smoother and more interpretable animation, we adopted an approach from Hennig [87] that draws samples along a structured trajectory.

The standard procedure for sampling $\boldsymbol{w}_i \sim \mathcal{N}(\boldsymbol{w}_0, \boldsymbol{\Sigma}_w)$ involves three steps:

1. Computing the Cholesky decomposition $\boldsymbol{L}$ of $\boldsymbol{\Sigma}_w$, such that $\boldsymbol{\Sigma}_w = \boldsymbol{L}\boldsymbol{L}^\intercal$.
2. Drawing a sample $\boldsymbol{s} \in \mathbb{R}^{DM} \sim \mathcal{N}(\boldsymbol{s}; \boldsymbol{0}, \boldsymbol{I})$ from a standard multivariate Gaussian.
3. Computing the final sample $\boldsymbol{w} = \boldsymbol{w}_0 + \boldsymbol{L}\boldsymbol{s}$.

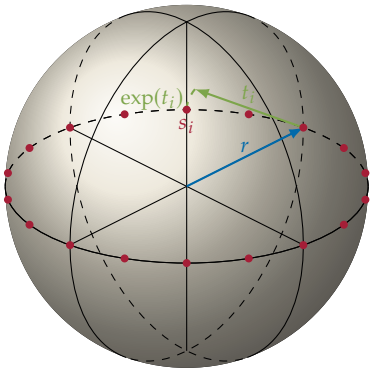


Figure 8.3: Schematic view of drawing equipotential samples from a standard Gaussian distribution. The golden sphere (2-manifold)—defined by the radius r—describes equipotential samples of a three-dimensional standard Gaussian distribution. A one-dimensional orbit is defined along this direction by randomly choosing the direction of the tangent vector t_i from the tangent space at the point r. The exponential map is used to map from the tangent bundle (Lie algebra) to the manifold (Lie group), e.g. $s_i = \exp(t_i)$. By that, equipotential samples (red) are drawn in equidistant intervals.

Our smooth animation approach modifies step 2. The standard Gaussian $\mathcal{N}(\boldsymbol{s}; \boldsymbol{0}, \boldsymbol{I})$ is spherically symmetric. One way to draw a sample $\boldsymbol{s}$ can be to sample a radius r from a Gaussian (Figure 8.3, blue vector) and then choose $\boldsymbol{s}$ uniformly at random on the $(DM-1)$-sphere (Figure 8.3, golden sphere). All points lying on this $(DM-1)$-manifold, which is a Lie group, are equally likely. Instead of randomly picking one point, it would be beneficial to show all of them to get an impression of the degeneracy. As it is impossible to plot an $(DM-1)$-dimensional space, time could be used as an extra dimension to traverse through the space. Therefore, for a smooth animation, we fix a radius r (e.g., corresponding to a specific probability contour) and then trace a path (an orbit) on this $(DM-1)$-sphere of equiprobable points. This is achieved by randomly choosing an initial point $\boldsymbol{s}_0$ on the sphere and by constructing an orthogonal tangent vector $\boldsymbol{t}_0$ at $\boldsymbol{s}_0$ (Figure 8.3, green vector). The tangent is part of the Lie algebra of the Lie group's manifold. We sample n points $\boldsymbol{s}_i$ by choosing tangents of equidistant lengths from 0 to 2π, which are then mapped through the exponential map to the orbit (Figure 8.3, red points). Each $\boldsymbol{s}_i$ is then transformed via step 3 to produce $\boldsymbol{w}_i$, which is reshaped into a projection matrix $\boldsymbol{W}_i$. The set of embeddings $\{\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_n\}$ generated from these structured samples forms the smooth frames of our animation, which aids in perceiving correlated movements and the overall extend of uncertainty.

8.6 Evaluation & Benchmarks

In this section, we evaluate the performance and computational characteristics of VIPurPCA. Specifically, we assess the accuracy of our Gaussian error propagation approach by comparing it against Monte Carlo (MC) sampling, and we analyze the runtime and memory consumption of our JAX-based implementation. Additionally, we discuss various strategies for visualizing the resulting uncertainty in the low-dimensional map.

8.6.1 Runtime & Space Complexity

The runtime and memory requirements of VIPurPCA were evaluated on simulated datasets using a CPU server[2]. We consider high-dimensional input data $\boldsymbol{Y}_0 \in \mathbb{R}^{N \times D}$ with its vectorization $\boldsymbol{y}_0 \in \mathbb{R}^{ND}$ and associated input covariance $\boldsymbol{\Sigma}_y$. The evaluation focuses oncomputing the mean

2: Intel® Xeon® CPU E5-2698 v3 @ 2.30 GHz, 64 CPUs, 378 GB memory

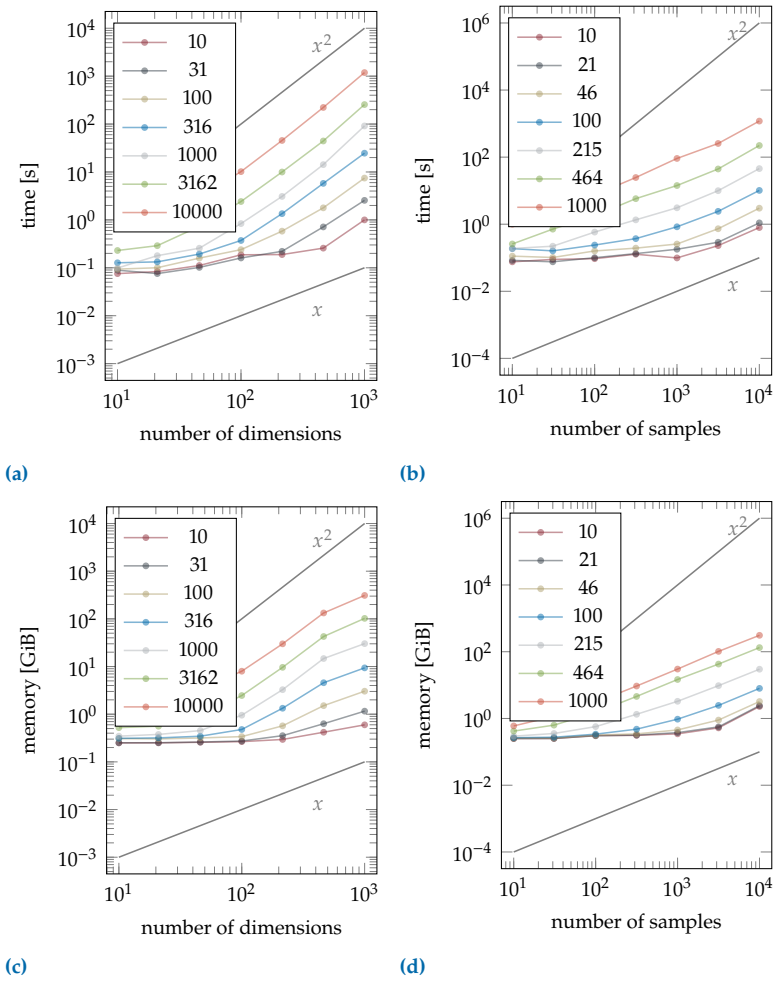


Figure 8.4: Runtime and space complexity of VIPurPCA for propagating uncertainty to the principal components. Grey lines indicate reference complexities. **(a)** Runtime and **(c)** memory usage as D (features) increases, for different N (samples, colors). **(b)** Runtime and **(d)** memory usage as N increases, for different D (colors). Mean values of 10 runs are shown.

$\boldsymbol{w}_0 = f(\boldsymbol{y}_0)$ and the covariance matrix $\boldsymbol{\Sigma}_w \in \mathbb{R}^{DM \times DM}$ of the first M principal components (where $M = 2$ was chosen for 2D visualization). The construction of $\boldsymbol{\Sigma}_w$ was performed column-wise in batches of 1000 using our JVP/VJP-based approach.

Figure 8.4 illustrates the empirical scaling of our method. The total cost of VIPurPCA comprises two main parts: first, computing the mean embedding $\boldsymbol{w}_0$ via standard PCA, and second, computing the covariance $\boldsymbol{\Sigma}_w$ via our linearization method. While the initial PCA involves an SVD with a cost of traditionally $\mathcal{O}(\min(ND^2, N^2D))$ [4], the subsequent uncertainty propagation dominates the overall complexity. Our empirical results show that the runtime and memory consumption for computing $\boldsymbol{\Sigma}_w$ scale approximately quadratically with the number of input features D (i.e., $\mathcal{O}(D^2)$) and linearly with the number of samples N (i.e., $\mathcal{O}(N)$).

8.6.2 Performance Comparison with Monte Carlo Sampling

To assess the accuracy of the Gaussian approximation derived from our linearization, we validate it against Monte Carlo (MC) sampling (Section 3.2.2). MC sampling provides an empirical estimate of the true output distribution by repeatedly propagating samples through

the exact, nonlinear PCA function. As the number of MC samples increases, this empirical estimate converges to the true moments of the output distribution. Our goal is to show that VIPurPCA's analytical approximation accurately matches this ground truth.

The validation proceeds as follows. Given an input distribution $p(\boldsymbol{y}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{y}_0, \boldsymbol{\Sigma}_y)$, we first compute the parameters of our approximate output distribution, $\boldsymbol{w}_0$ and $\boldsymbol{\Sigma}_w$, using VIPurPCA. We then compare these to the MC estimates. For the MC estimation, S samples $\boldsymbol{y}_i \sim p(\boldsymbol{y})$ are drawn. For each sample, the exact PCA function is applied to obtain an output $\boldsymbol{w}_i = f(\boldsymbol{y}_i)$. The MC estimates for the mean ($\boldsymbol{m}_w$) and covariance ($\boldsymbol{C}_w$) are the standard sample mean and covariance of this output set:

$$\boldsymbol{m}_w = \frac{1}{S}\sum_{i=1}^{S} \boldsymbol{w}_i, \tag{8.11}$$

$$\boldsymbol{C}_w = \frac{1}{S-1}\sum_{i=1}^{S} (\boldsymbol{w}_i - \boldsymbol{m}_w)(\boldsymbol{w}_i - \boldsymbol{m}_w)^\intercal. \tag{8.12}$$

We quantify the difference between VIPurPCA's analytical result and the MC estimate using the relative Frobenius norm, E_{rel}:

$$E_{\text{rel}} = \frac{||\boldsymbol{\theta}_{\text{MC}} - \boldsymbol{\theta}_{\text{VIPurPCA}}||_{\text{F}}}{||\boldsymbol{\theta}_{\text{VIPurPCA}}||_{\text{F}}}, \tag{8.13}$$

where $\boldsymbol{\theta}$ represents either the mean vector ($\boldsymbol{w}_0$ vs. $\boldsymbol{m}_w$) or the covariance matrix ($\boldsymbol{\Sigma}_w$ vs. $\boldsymbol{C}_w$).

The MC estimation accuracy and runtime depend on the number of samples S. We implemented MC sampling in JAX using `jax.vmap` for efficient batching of PCA computations over the S samples. Figure 8.5 shows the runtime of MC sampling (solid blue line), which increases linearly with S. VIPurPCA's runtime (dashed blue line) is independent of S. The relative Frobenius norm E_{rel} (red lines) for both the mean (dashed red) and covariance (solid red) estimates decreases as S increases, indicating that the MC estimates converge towards VIPurPCA's analytical approximation.

For each dataset size, there is a critical number of MC samples S beyond which MC sampling becomes more time-consuming than VIPurPCA (intersection of blue lines in Figure 8.5). At these crossover points, the MC estimate of the mean has often converged reasonably well for many datasets. However, the MC estimate of the covariance matrix typically requires significantly more samples S to converge to a similar level of accuracy as VIPurPCA's direct computation, often taking much longer. This highlights VIPurPCA's efficiency, particularly for estimating the output covariance $\boldsymbol{\Sigma}_w$.

8.6.3 Visualization Strategies

This subsection discusses various approaches for visualizing the uncertainty in the low-dimensional PCA map, derived from the propagated distribution $p(\boldsymbol{w} \mid \boldsymbol{y}) \approx \mathcal{N}(\boldsymbol{w}_0, \boldsymbol{\Sigma}_w)$. As detailed in Section 8.5, we generate a set of n sampled projection matrices $\boldsymbol{W}_i$ (from $\boldsymbol{w}_i \sim p(\boldsymbol{w} \mid \boldsymbol{y})$) and use them to project the mean input data $\boldsymbol{Y}_0$, yielding a set of n

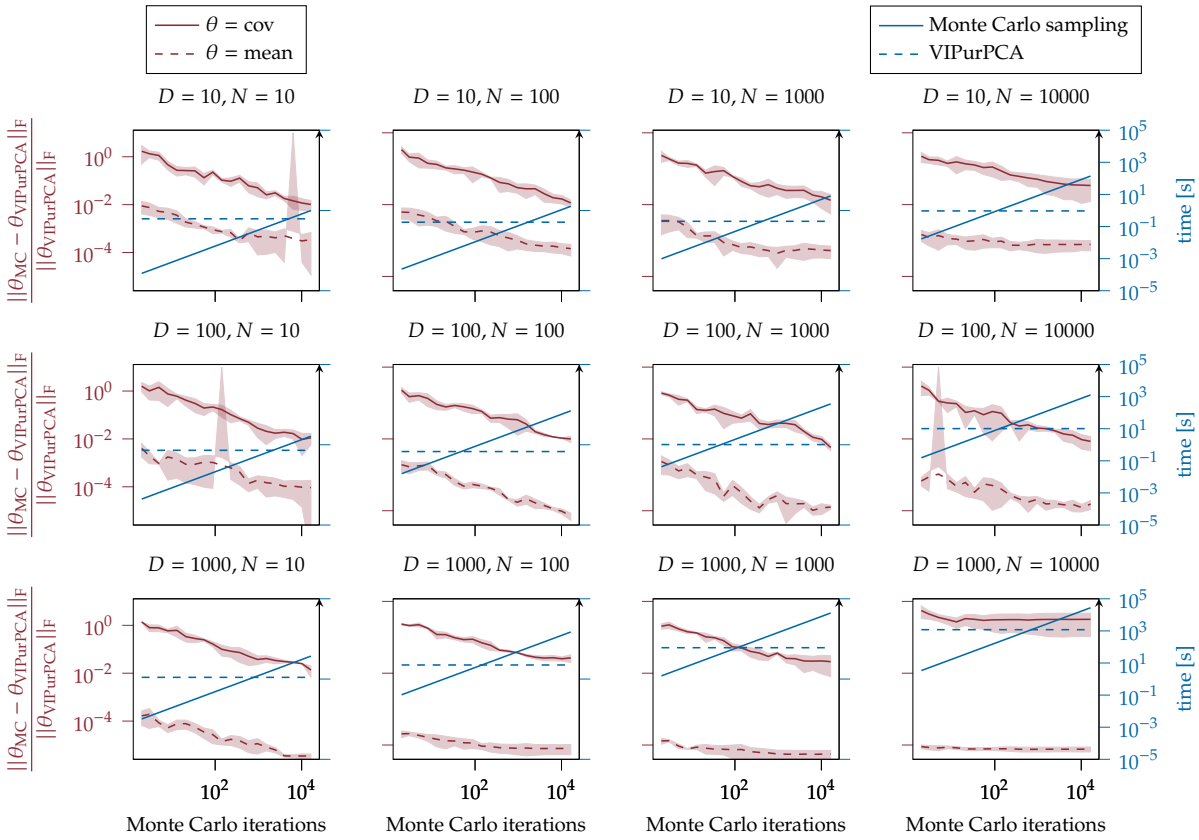


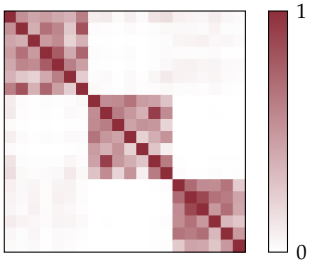
Figure 8.5: Performance evaluation of VIPurPCA compared to Monte Carlo sampling. For simulated datasets of an increasing number of samples (columns) and dimensions (rows), the runtime (blue) of VIPurPCA and Monte Carlo sampling is evaluated for an increasing number of Monte Carlo iterations. The runtime of VIPurPCA is not dependent on the Monte Carlo iterations and is indicated as a blue dashed line as a reference. In addition, the relative Frobenius norms (red) between the first (dashed) and second (solid) moment, respectively, of the output distribution over eigenvectors computed by VIPurPCA and Monte Carlo sampling are shown. The mean values of 10 repeats are shown, respectively.

embeddings $Z = \{\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_n\}$. These embeddings can be visualized via animation, overlay, or small multiples.

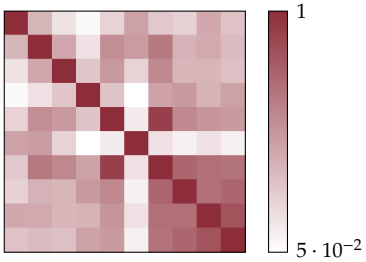
Section 8.5 introduced our animation approach, which uses structured sampling along equipotential orbits to ensure smooth transitions between frames. We now compare this with other methods using a simulated dataset $\boldsymbol{Y}_0 \in \mathbb{R}^{20\times10}$ with three classes. The input uncertainty $\boldsymbol{\Sigma}_y = \boldsymbol{D}_0 \otimes \boldsymbol{N}_0$ is designed such that samples within the same class have correlated uncertainties (Figure 8.6a), and a subset of features also have correlated uncertainties (Figure 8.6b). This models realistic scenarios, e.g., where similar experimental treatment introduces correlated errors for a group of samples, or where unobserved factors affect multiple features simultaneously.

Figure 8.7a shows the standard PCA projection of the data mean $\boldsymbol{Y}_0$. Figure 8.7b (overlay of 100 sampled projections $\boldsymbol{Z}_i$) and Figure 8.7c (density summarization of these samples) illustrate the uncertainty. Both show potential overlap between classes A and C in the PC1-PC2 subspace. However, these static ensemble views do not readily reveal correlated movements between samples or the dynamic nature of their relationships under uncertainty. Visual clutter can also be an issue, even with only 20 data points.

In contrast, the small multiples display (Figure 8.7d, showing 10 sampled

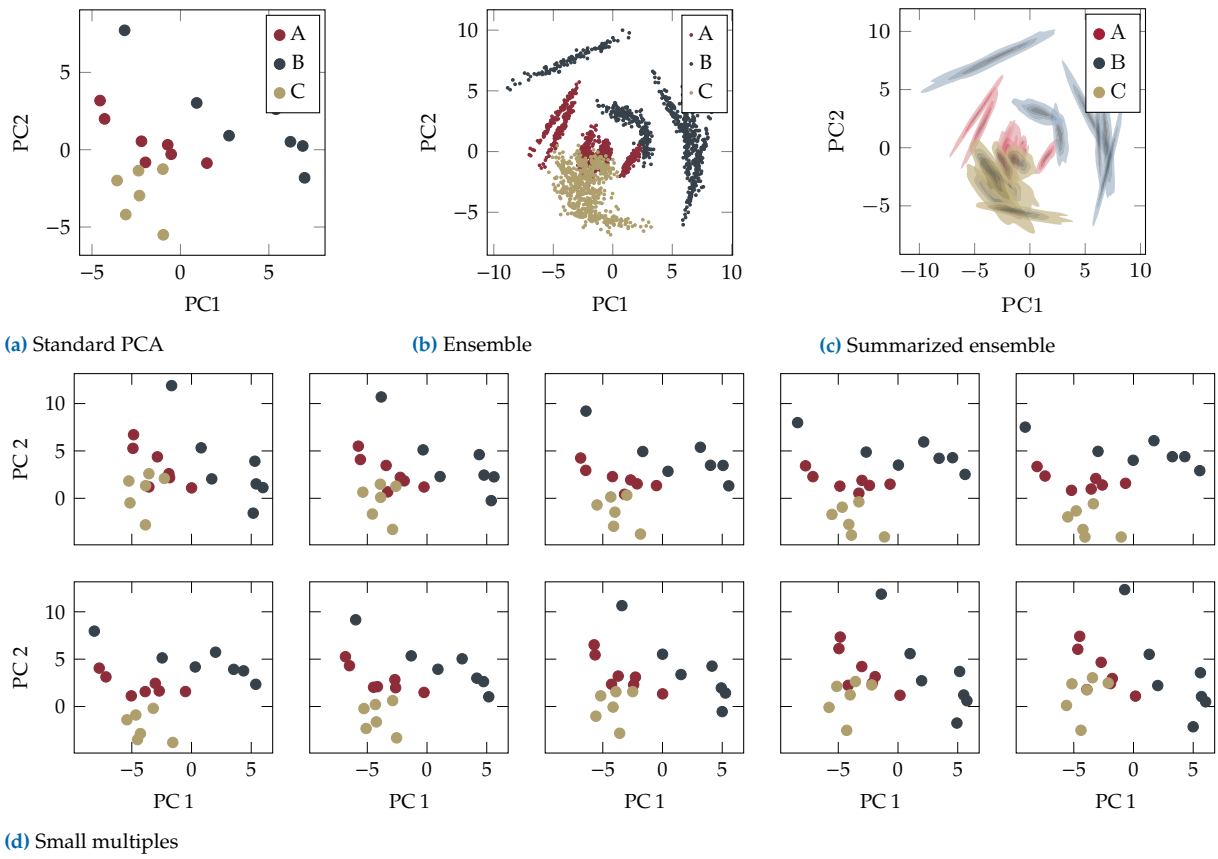


(a) Sample covariance matrix $\boldsymbol{N}_0$



(b) Feature covariance matrix $\boldsymbol{D}_0$

Figure 8.6: Covariance structures for the input distribution $p(\boldsymbol{y}) = \mathcal{N}(\boldsymbol{y}_0, \boldsymbol{D}_0 \otimes \boldsymbol{N}_0)$ used to evaluate visualization approaches.



(a) Standard PCA (b) Ensemble (c) Summarized ensemble

(d) Small multiples

Figure 8.7: Visualization strategies for uncertain low-dimensional PCA maps for a simulated dataset (20 samples, 3 classes). The distribution $p(\boldsymbol{w} \mid \boldsymbol{y})$ computed by VIPurPCA is used to generate the data for the visualizations. **(a)** Standard PCA result using the mean $\boldsymbol{w}_0$. **(b)** Overlay ensemble of 100 sampled projections. **(c)** Density summarization of the ensemble in **(b)**. **(d)** Small multiples showing 10 sampled projections using structured sampling for smooth animation.

projections) and our animation approach (example at `https://integrative-transcriptomics.github.io/VIPurPCA/examples/viseval uation/`) maintain a common scale and axes, facilitating comparison and revealing the structure of potential realizations. Both clearly show the A-C class overlap and highlight specific samples whose positions are particularly variable. The animation, with its smooth transitions, excels at conveying correlated movements and the overall structure of the uncertainty. While small multiples are better for static media (like print), animations provide a more dynamic and often more intuitive understanding of the embedding's stability and how different parts of the projection co-vary.

8.7 Experiments

In this section, we demonstrate the application of VIPurPCA to three diverse example datasets, each exhibiting different sources and structures of input uncertainty. These examples showcase VIPurPCA's ability to reveal insights into the stability of PCA embeddings and how these insights can inform data interpretation. The first dataset, student grades data from Denoeux and Masson [54], features uncorrelated uncertainties arising from an imprecise grading scheme. The second, protein expression levels

	M1	M2	P1	P2
Tom	15	fairly good	$\mathcal{N}(14, 5.7^2)$	[14, 16]
David	9	good	fairly good	10
Bob	6	[10, 11]	[13, 20]	good
Jane	fairly good	very good	19	[10, 12]
Joe	very bad	fairly bad	[10, 14]	14
Jack	1	[4, 6]	9	[6, 9]

Table 8.1: Test results of six students, with some scores provided as real numbers, intervals, or qualitative statements.

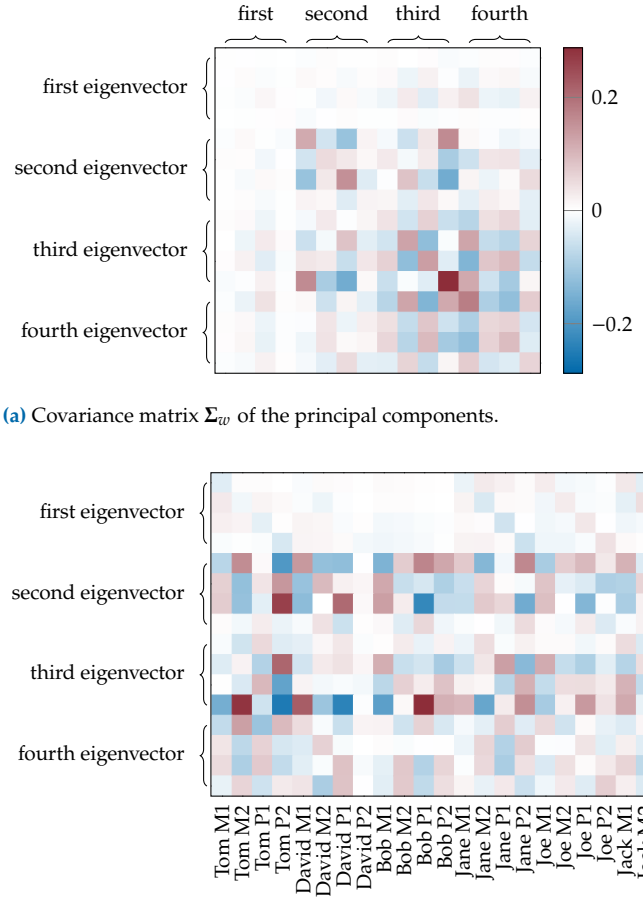
in the mouse cortex by Higuera et al. [90], involves uncertainties derived from technical replicates, which are potentially correlated. The third dataset [210] comprises human gene expression levels with uncertainties inferred from Affymetrix microarrays using Bayesian methods.

8.7.1 Student Grades Data

This simulated dataset, introduced by Denoeux and Masson [54], consists of four marks (ranging from 0 to 20) obtained by six students in mathematics (M1, M2) and physics (P1, P2), as detailed in Table 8.1. Some marks are imprecise, provided as intervals (modeled as uniform distributions) or linguistic labels like "fairly good" (modeled as trapezoidal distributions, see Table B.1b for definitions). One score (P1 for Tom) is given as $\mathcal{N}(14, 5.7^2)$ as suggested by [80]. For each score, we determined its mean and variance based on its representation (see Table B.1c in the Appendix for details). Precise real-valued scores were assigned a very small variance (e.g., 0.1^2). This dataset primarily exemplifies uncorrelated uncertainties for individual scores. The input to VIPurPCA is the 6×4 matrix of mean scores $\boldsymbol{Y}_0$, and a diagonal input covariance matrix $\boldsymbol{\Sigma}_y$. The diagonal entries of $\boldsymbol{\Sigma}_y$ are the variances of the corresponding scores in the vectorized $\boldsymbol{y}_0 = \text{vec}(\boldsymbol{Y}_0)$. Specifically, $(\boldsymbol{\Sigma}_y)_{ii} = \text{Var}((\boldsymbol{y}_0)_i)$, and off-diagonal elements are zero, reflecting the assumption of independent uncertainties between scores after their initial specification.

Applying VIPurPCA yields the mean principal components $\boldsymbol{w}_0$ and their covariance $\boldsymbol{\Sigma}_w$. Figure 8.8a displays $\boldsymbol{\Sigma}_w$ (for the first few PCs). The relatively small (co-)variances associated with the first principal component (PC1) indicate its high stability. This is expected, as PC1 captures a large proportion of the total data variance (74%), making its direction less susceptible to small input uncertainties. Conversely, PCs corresponding to smaller data variances (e.g., PC2, PC3) exhibit larger uncertainties in $\boldsymbol{\Sigma}_w$. The Jacobian $\boldsymbol{J}_{\boldsymbol{y}} f(\boldsymbol{y}_0)$, visualized in Figure 8.8b, reveals the sensitivity of each PC coordinate to changes in each input score. The top rows (corresponding to PC1 elements) show low sensitivity values, confirming PC1's robustness. Rows corresponding to PC2 elements show higher sensitivities, indicating PC2's direction is more influenced by input perturbations.

Figure 8.9a and Figure 8.9c show standard PCA projections onto different PC pairs. Figure 8.9b and Figure 8.9d display the corresponding VIPurPCA results as traces (representing the animation frames from `https://integrative-transcriptomics.github.io/VIPurPCA/examples/studentgrades/`). The shapes of the traces in Figure 8.9b (PC1 vs PC2) indicate greater uncertainty along the PC2 direction than PC1, consistent with $\boldsymbol{\Sigma}_w$. The PC2 vs PC3 projection (Figure 8.9d) reveals



(a) Covariance matrix $\mathbf{\Sigma}_w$ of the principal components.

(b) Jacobian matrix $\boldsymbol{J}_{\boldsymbol{y}} f(\boldsymbol{y}_0)$ (sensitivities).

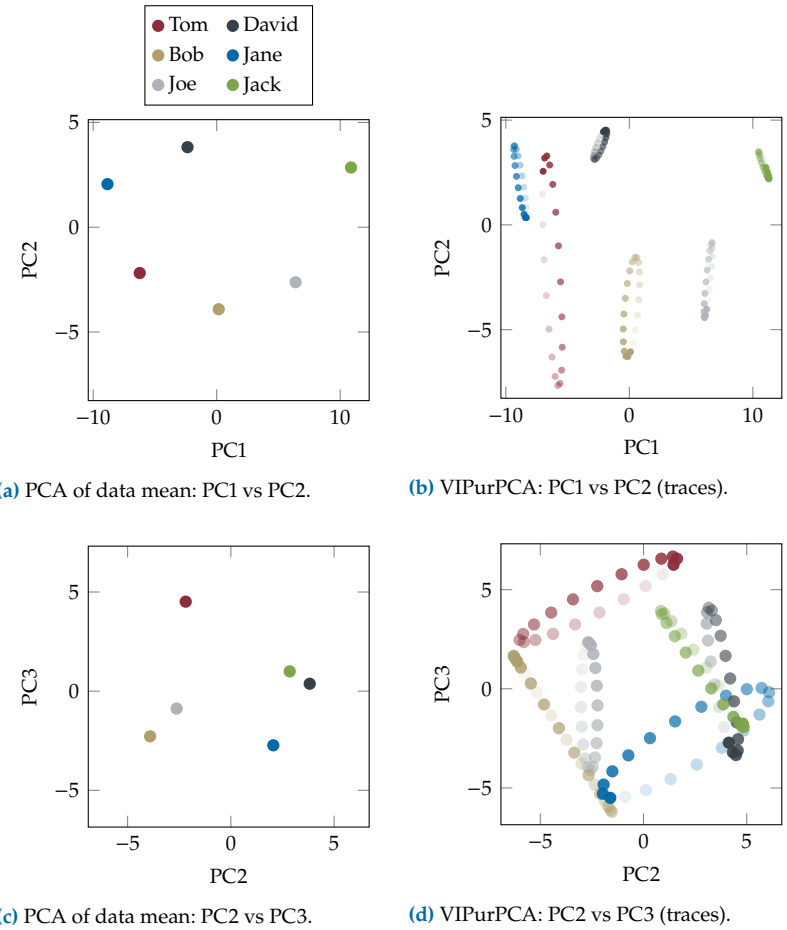
Figure 8.8: Sensitivities for the student grades dataset. **(a)** Covariance matrix $\mathbf{\Sigma}_w$ of the (vectorized) principal components, representing propagated uncertainties. **(b)** Jacobian matrix $\boldsymbol{J}_{\boldsymbol{y}} f(\boldsymbol{y}_0)$ showing the sensitivity of PC elements (rows) to input scores (columns).

much larger uncertainties, reflecting the smaller difference between the second and third eigenvalues and higher sensitivities for these components. The animation (and its trace representation) also reveals correlated movements; for example, the projections of Tom and Jane appear to anti-correlate in the PC2-PC3 subspace. This example underscores that the reliability of PCA projections can vary significantly depending on which PCs are chosen for visualization, an insight not apparent from standard PCA alone.

8.7.2 Mice Protein Expression Data

This dataset from the UCI Machine Learning Repository [170], originally analyzed by Higuera et al. [90], contains expression levels of 77 proteins (features) from the mouse cortex for 72 mice (samples). The experiment included 15 technical replicates for each mouse under specific conditions. Mice were categorized into 8 classes based on genetics and treatment; for visualization, we focus on a binary label: whether mice were stimulated to learn (context shock, CS) or not (no context shock, SC).

After preprocessing as suggested by Higuera et al. [90], replicates were used to estimate the uncertainty of each measurement. This was done by vectorizing the matrix of each replicate and stacking the resulting vectors



(a) PCA of data mean: PC1 vs PC2.

(b) VIPurPCA: PC1 vs PC2 (traces).

(c) PCA of data mean: PC2 vs PC3.

(d) VIPurPCA: PC2 vs PC3 (traces).

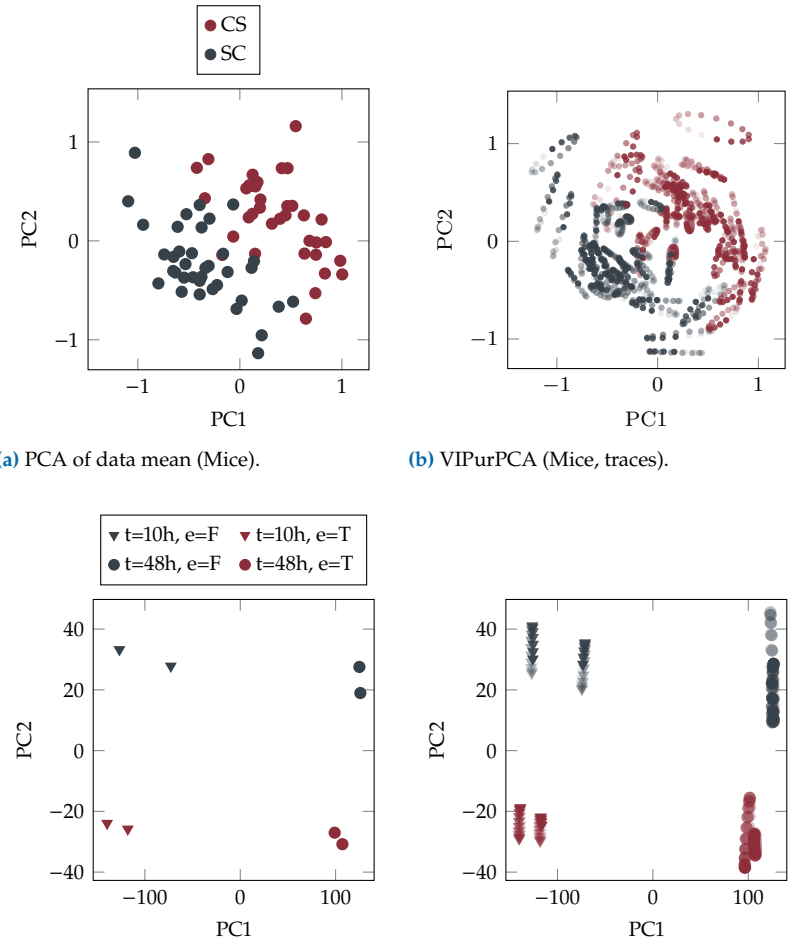
Figure 8.9: Projections of the student grades dataset obtained by standard PCA (**(a)**, **(c)**) and VIPurPCA (**(b)**, **(d)**). Standard PCA was applied to the data mean $\mathbf{Y}_0$. The projection uncertainties as computed byVIPurPCA are visualized as traces (see Section 8.5). Different PCs are shown in subfigure **(a)**, **(b)** and **(c)**, **(d)**, respectively.

as columns in a matrix, where the rows are the variables and the columns are the replicates, to compute a covariance matrix. An important property of the resulting covariance matrix is that it has covariance entries other than zero.

Standard PCA on the mean expression data (Figure 8.10a) shows that the first two PCs can nearly linearly separate the CS and SC classes. The VIPurPCA animation traces (Figure 8.10b, representing the animation available at `https://integrative-transcriptomics.github.io/VIPurPCA/examples/mice/`) illustrate the uncertainty in this PC1-PC2 subspace. While individual samples show some variability, and some CS samples (red) have traces that extend towards the SC samples (grey), the overall class separation appears largely robust to the input uncertainties derived from technical replicates. Correlated uncertainties are evident for some subsets of samples within each class.

8.7.3 Human Gene Expression Data

This microarray gene expression dataset [210] is available via the Bioconductor package `puma` [211]. It comprises data from eight Affymetrix Human Genome U95Av2 Arrays, from a 2×2 factorial experiment investigating two factors: estrogen (absent/present) and time (10h/48h), with two replicates per condition. The `puma` package uses Bayesian methods [154] to derive gene expression levels along with their uncertainty



(a) PCA of data mean (Mice).

(b) VIPurPCA (Mice, traces).

(c) PCA of data mean (Human Gene Exp.).

(d) VIPurPCA (Human Gene Exp., traces).

Figure 8.10: Results of VIPurPCA applied to the mice protein expression data (**(a)** and **(b)**) and human gene expression data (**(c)** and **(d)**). Mice samples are labeled by CS (context shock) and SC (no context shock). Human samples are labeled by timepoint and estrogen presence (T) or absence (F). **(a)** and **(c)** Standard PCA projections onto PC1-PC2. **(b)** and **(d)** VIPurPCA results showing animation traces for the PC1-PC2 projections.

estimates (variances) from the raw microarray data. We selected the 10% most variant genes (across all conditions, based on mean expression) out of 12,625 genes for analysis. The input to VIPurPCA is the matrix of mean expression levels $\boldsymbol{Y}_0$ for these selected genes across the 8 samples (2 replicates × 4 conditions), and a diagonal input covariance matrix $\boldsymbol{\Sigma}_y$ where diagonal elements are the reported variances for each gene in each sample.

Standard PCA (Figure 8.10c) shows that PC1 primarily separates the time points, while PC2 separates by estrogen availability. The VIPurPCA animation traces (Figure 8.10d, representing the animation available at `https://integrative-transcriptomics.github.io/VIPurPCA/examples/human/`) indicate that the projections are more uncertain along the PC2 direction than PC1. Despite this, the separation of experimental conditions by PC1 and PC2 appears reliable, and replicates for each condition generally cluster closely even when considering the propagated input uncertainty. This example demonstrates how VIPurPCA can confirm the robustness of observed structures when uncertainty is propagated from upstream probabilistic analyses.

8.8 Discussion & Conclusion

Dimensionality reduction techniques enable the classification, visualization, and compression of high-dimensional data. Including uncertainty quantification and visualization of the lower dimensional map improves its interpretability and trustworthiness.

The primary contributions of VIPurPCA include:

- **Principled Uncertainty Propagation:** We formulated the propagation of Gaussian input uncertainty ($\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{y}_0, \boldsymbol{\Sigma}_y)$) through PCA using a first-order Taylor approximation. This yields an approximate Gaussian distribution for the principal components ($\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{w}_0, \boldsymbol{\Sigma}_w)$), where the mean $\boldsymbol{w}_0$ is the standard PCA result on $\boldsymbol{y}_0$, and the covariance $\boldsymbol{\Sigma}_w$ quantifies the propagated uncertainty.
- **Efficient Implementation:** Leveraging automatic differentiation (AD) within the JAX framework, we demonstrated an efficient method to compute the necessary Jacobians and construct $\boldsymbol{\Sigma}_w$ using Jacobian-vector products (JVPs) and vector-Jacobian products (VJPs), avoiding the explicit formation of potentially very large intermediate matrices.
- **Handling Correlated Input Uncertainties:** VIPurPCA is capable of handling structured input covariances $\boldsymbol{\Sigma}_y$, including correlated uncertainties between input features or samples (e.g., via Kronecker products or block-diagonal structures), as demonstrated in the mice protein expression dataset.
- **Novel Visualization of Embedding Uncertainty:** We introduced an animation-based technique that visualizes the uncertainty of the low-dimensional PCA map by smoothly transitioning through equipotential samples drawn from $p(\boldsymbol{w} \mid \boldsymbol{y})$. This method effectively conveys the stability of the embedding, including correlated movements of projected points.

A common approach to uncertainty propagation is to linearly approximate a function's outcome using a Taylor series expansion. It provides valid results when the function of interest is not too far from linear within one standard deviation of the mean, and when the input variabilities are relatively small [244]. Therefore, if the input uncertainties become too large, the computed output distribution could be inaccurate as a matter of principle. The use of higher-order terms could improve the approximation, but this involves computing a Hessian matrix of a vector-valued function, which is not feasible in the case of PCA. Directions corresponding to eigenvalues close to zero are extremely sensitive to input variability and their order may also be uncertain. Therefore, their distribution is more likely to be incorrectly approximated by VIPurPCA. However, these low-variant dimensions are usually not retained because a few dimensions already explain most of the variance due to an underlying principle explaining the input data.

For the simulated data, when comparing the first and second moments of the final distribution computed by Monte Carlo sampling and VIPurPCA, the relative error decreases with an increasing number of Monte Carlo iterations (Figure 8.5). The magnitude of the relative error indicates the number of significant decimal digits excluding leading zeros, e.g. if $E_{\text{rel}} = 10^{-p}$, the estimator has at least p correct significant digits. As

can be seen in Figure 8.5 (solid red line), VIPurPCA approximates the uncertainties to about two significant decimal digits in the majority of cases. The propagation of the uncertainties did not work only for the largest data set ($p = 10^3, n = 10^4$), probably because floating point errors occur during the calculation, which could be avoided by allowing JAX to use double (64bit) precision. As shown in Figure B.1, our analytical VIPurPCA method produces an uncertain PCA map that is visually identical to the one generated by a computationally intensive Monte Carlo simulation. This demonstrates that our method effectively and accurately propagates the input uncertainty without the need for repeated sampling. As it depends on the number of iterations, the runtime of Monte Carlo sampling is difficult to compare with that of VIPurPCA. Depending on the number of dimensions p and samples n, Monte Carlo sampling becomes computationally more expensive than VIPurPCA for a certain number of iterations (Figure 8.5, intersection of blue lines). In general, runtime and memory consumption are highly implementation dependent and interdependent. For VIPurPCA, the time (Figure 8.4a, Figure 8.4b) and space complexity (Figure 8.4c, Figure 8.4d) are linear in the number of samples and quadratic in the number of input dimensions. For datasets with sample sizes up to 10^4 and dimensions of 10^3, the compute time is still in the order of minutes.

Another focus of this work was to visualize the uncertainty of the lower dimensional map in an appropriate and intuitive way. It is possible to visualize the covariance matrix of the eigenvectors directly for example as a heatmap [145]. However, the effect of individual entries of the covariance matrix on the stability of the lower dimensional map is not obvious. We therefore decided to rather show the structure of the samples drawn from the distribution over the eigenvectors by using these samples to transform the input data accordingly. Overall, the presentation of the uncertainty of the lower dimensional map as an animation is intended to complement an already existing visualization of a PCA result, namely a scatter plot of two principal components, with uncertainty information. Therefore, if the scatter plot itself fails as an appropriate visualization, i.e. is not clearly arranged, the animation cannot remedy this. Looking at the different visualization options (Figure 8.7b, Figure 8.7c, Figure 8.7d and `https://integrative-transcriptomics.github.io/VIPurPCA/examples/visevaluation/`), various individual advantages and disadvantages become apparent. The point cloud (Figure 8.7b) and the density plot (Figure 8.7c) are capable of displaying many samples so that the full distribution of potential embedding locations for each subject is visible. While the point cloud can represent outliers, the density plot reduces visual clutter, although not completely. Nevertheless, both visualizations show the union of all samples, and the structure of individual samples is lost as a result. However, this is particularly important with overlapping point clouds/density curves to determine whether the objects are actually overlapping or oscillating together. The small multiples visualization (Figure 8.7d) or the animation (`https://integrative-transcriptomics.github.io/VIPurPCA/examples/visevaluation/`) can resolve this issue as well as the problem of visual clutter. By using our proposed method of drawing equipotential samples following an orbit in the distribution (adapted from [87]), a smooth transition between frames is possible and individual data points can be tracked, especially in the animation. While the small multiples visualization requires more space,

the animation has the disadvantage that it cannot be printed. How well the uncertainty of the low-dimensional map is perceived by the user in any of the visualization techniques mentioned depends on several properties of the data, including the number of samples, the amount of uncertainty being visualized, the structure of the data (e.g. if it is clustered), and the structure of the uncertainty (e.g. if it is correlated). Future work could evaluate in detail the perception of uncertainty for differently structured datasets. VIPurPCA also visualizes the Jacobian matrix, which shows the influence of individual features and samples on the outcome of the PCA, as a heatmap. This makes it easy to identify input features or samples that have a large impact on the result.

In this work, we used three real-world data sets with different sources of uncertainty and calculated and visualized the stability of the embeddings. The impact of the input uncertainties on the stability of the embedding depends not only on their magnitude and structure, but also on the dataset itself, and can affect the confidence of the embedding to varying degrees. Dimensionality reduction is often used to visualize potential clusters in the data. VIPurPCA helps to assess whether the observed clusters are stable under the input uncertainties, or whether the cluster membership of individual samples may be questionable (see for example Figure 8.10b). For the human gene expression dataset, VIPurPCA helped to identify that the first two PCs reliably separated experimental conditions and replicates were very similar.

To simplify the use of VIPurPCA for a wide range of scientists, we plan to release VIPurPCA as a web tool. This will provide additional ways of interacting with the data and visualizations that could further improve the interpretability of the data. Although PCA is effectively one of the most commonly used dimensionality reduction techniques, nonlinear methods such as t-SNE [161] and UMAP [173] are favored when the data is distributed near a low-dimensional nonlinear manifold. Consequently, future work could involve extending our uncertainty propagation and visualization approach to these nonlinear methods. In Chapter 9, we demonstrate how Gaussian error propagation can be applied to t-SNE.

Uncertainty Propagation through t-SNE

9.

Disclaimer 9.1 Chapter 9, which has not been published previously, is introduced for the first time in this thesis.

Abstract

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a widely adopted dimensionality reduction technique for visualizing high-dimensional biological data, yet standard implementations typically disregard inherent input data uncertainty. This omission can lead to overconfident interpretations of the resulting low-dimensional embeddings. This work introduces a framework to propagate input uncertainty through the t-SNE algorithm. By leveraging probabilistic approximation methods in conjunction with automatic differentiation for gradient computation within the t-SNE optimization process, we estimate the uncertainty associated with each point in the low-dimensional embedding. We demonstrate our approach on synthetic and real-world biological datasets, showcasing how visualizing this propagated uncertainty provides a more nuanced understanding of the embedding, revealing the stability of clusters and the reliability of individual point placements. While computationally more intensive than standard t-SNE, our method offers a crucial means to assess the robustness of t-SNE visualizations, thereby enabling more informed scientific conclusions in bioinformatics.

9.1 Introduction

t-Distributed Stochastic Neighbor Embedding (t-SNE, Section 2.2) [161] is a commonly used method for visualizing high-dimensional biological datasets. Its strength lies in revealing local data structures and cluster separations in low-dimensional embeddings. Applications in bioinformatics range from exploring cellular populations in single-cell RNA sequencing [126, 157] and cytometry data [20] to assessing relationships between samples in proteomics [245]. However, standard t-SNE implementations treat input data as fixed point estimates, thereby disregarding any inherent uncertainty associated with these measurements.

Code, experiments, and animated visualizations are available at the Github repository `Integrative-Transcriptomics/tsne`

In numerous bioinformatics scenarios, data are intrinsically noisy or uncertain [118]. This uncertainty can stem from various sources: measurement errors in experimental apparatus, stochasticity inherent in biological processes, or statistical uncertainty propagated from upstream computational models that generate the input features. Ignoring such input uncertainty can lead to brittle or misleading interpretations of t-SNE embeddings, where the perceived structure or separation of clusters might be an artifact of noise rather than a true biological signal. Consequently, the stability and reliability of the visualized structures are often unknown.

This chapter directly addresses this limitation by introducing a methodology to propagate input uncertainty through the t-SNE algorithm. Drawing upon the principles of probabilistic inference (Chapter 3) and automatic differentiation (Chapter 4), and leveraging the Implicit Function Theorem (Chapter 5), we extend the t-SNE framework to quantify how uncertainty in the high-dimensional input space translates to uncertainty in the low-dimensional embedding. This approach allows for a more robust interpretation of t-SNE visualizations, offering insights into the confidence of individual point positions and the stability of observed clusters. Our methodology parallels the uncertainty propagation framework developed for Principal Component Analysis in VIPurPCA (Chapter 8).

We begin by formally defining the problem of uncertainty propagation in t-SNE, considering input data points characterized by probability distributions rather than fixed vectors (Section 9.2). We then detail the methodology and practical implementation of our uncertainty-aware t-SNE approach (Section 9.3). This involves using the Implicit Function Theorem to derive the sensitivity of the optimal t-SNE embedding to input perturbations and employing Gaussian error propagation. We address the challenge of the t-SNE cost function's invariance to translation and rotation by utilizing the Moore-Penrose pseudoinverse of its Hessian. The efficacy and interpretability of our method are demonstrated through showcases on synthetic and real-world biological datasets (Section 9.4), highlighting how visualizing uncertainty can prevent over-interpretation and reveal ambiguities. A runtime evaluation assesses the computational characteristics of the proposed method (Section 9.5). Finally, we discuss the implications of our findings and current limitations (Section 9.6).

9.2 Problem Setting

Remark 9.1 Throughout this chapter, we denote matrices by bold uppercase letters (e.g., $\boldsymbol{Y}$) and vectors by bold lowercase letters (e.g., $\boldsymbol{y}$). The vec$(\cdot)$ operator transforms a matrix into a column vector by stacking its columns or rows. For notational simplicity in some contexts, we may use $\boldsymbol{y}$ to represent vec$(\boldsymbol{Y})$ directly where the meaning is clear.

Let the high-dimensional input data be represented by a matrix $\boldsymbol{Y} \in \mathbb{R}^{N\times D}$, where N is the number of samples and D is the number of features. We assume that this input data is uncertain and model its vectorized form, $\boldsymbol{y} = \text{vec}(\boldsymbol{Y}) \in \mathbb{R}^{ND}$, as a Gaussian random vector:

$$p(\boldsymbol{y}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{y}^*, \boldsymbol{\Sigma}_y), \tag{9.1}$$

where $\boldsymbol{y}^* = \text{vec}(\boldsymbol{Y}^*)$ is the mean of the high-dimensional data, and $\boldsymbol{\Sigma}_y \in \mathbb{R}^{ND\times ND}$ is its covariance matrix. In many practical applications, $\boldsymbol{\Sigma}_y$ may possess a specific structure, such as a Kronecker product $\boldsymbol{\Sigma}_y = \boldsymbol{D}_0 \otimes \boldsymbol{N}_0$ (with $\boldsymbol{D}_0 \in \mathbb{R}^{D\times D}$ and $\boldsymbol{N}_0 \in \mathbb{R}^{N\times N}$ being symmetric positive definite (SPD) matrices) or a diagonal matrix, to ensure computational tractability.

t-SNE (Section 2.2) is applied to a point estimate of the input data, typically its mean $\boldsymbol{Y}^*$ (or $\boldsymbol{y}^*$), to obtain a low-dimensional representation $\boldsymbol{Z}^* \in \mathbb{R}^{N \times P}$. For a typical two-dimensional embedding ($P = 2$), its vectorized form is $\boldsymbol{z}^* = \text{vec}(\boldsymbol{Z}^*) \in \mathbb{R}^{2N}$. The optimal embedding $\boldsymbol{z}^*$ is found by minimizing the t-SNE cost function, $C : \mathbb{R}^{ND} \times \mathbb{R}^{2N} \to \mathbb{R}$:

$$\boldsymbol{z}^* = \arg\min_{\boldsymbol{z}} C(\boldsymbol{y}^*, \boldsymbol{z}), \tag{9.2}$$

where $C(\boldsymbol{y}, \boldsymbol{z})$ is the Kullback-Leibler (KL) divergence between the pairwise similarity distributions computed from the matrix forms of $\boldsymbol{y}$ and $\boldsymbol{z}$ respectively.

The core objective of this project is to approximate the distribution of the optimal low-dimensional embedding $\boldsymbol{z}^*$ conditional on the distribution of the input data $\boldsymbol{y}$. To propagate the input uncertainty $\boldsymbol{\Sigma}_y$ through the t-SNE mapping using a first-order Taylor approximation (Gaussian error propagation, as detailed in Section 3.2.1), we require the Jacobian of the optimal solution map, $\frac{\partial \boldsymbol{z}^*(\boldsymbol{y})}{\partial \boldsymbol{y}} \in \mathbb{R}^{2N \times ND}$. This Jacobian quantifies the sensitivity of the optimal embedding $\boldsymbol{z}^*$ to infinitesimal changes in the input $\boldsymbol{y}$.

We can define an implicit function $\boldsymbol{z}^* : \mathbb{R}^{ND} \to \mathbb{R}^{2N}$ that maps an input $\boldsymbol{y}$ to its corresponding optimal t-SNE embedding $\boldsymbol{z}^*(\boldsymbol{y})$. To propagate uncertainty from $\boldsymbol{y}$ to $\boldsymbol{z}^*$ using first-order Taylor approximation (Gaussian error propagation, as detailed in Section 3.2.1), we require the Jacobian of this mapping, $\boldsymbol{J}_{\boldsymbol{z}^*(\boldsymbol{y})} = \frac{\partial \boldsymbol{z}^*(\boldsymbol{y})}{\partial \boldsymbol{y}} \in \mathbb{R}^{2N \times ND}$. This Jacobian quantifies the sensitivity of the optimal embedding $\boldsymbol{z}^*$ to infinitesimal changes in the input $\boldsymbol{y}$.

Computing this Jacobian by differentiating through the entire iterative optimization process of t-SNE using automatic differentiation (Section 4.3) is generally intractable due to the large number of iterations. This would create an excessively large computation graph, leading to prohibitive memory consumption and potential accumulation of floating-point errors. Therefore, the central technical challenge addressed in this chapter is to find an efficient method for computing this Jacobian.

9.3 Theory & Implementation

To overcome the intractability of differentiating through the t-SNE optimization process, we leverage the implicit function theorem (IFT, Chapter 5). This allows us to compute the desired Jacobian $\frac{\partial \boldsymbol{z}^*(\boldsymbol{y})}{\partial \boldsymbol{y}}$ using only derivatives of the t-SNE cost function evaluated at the optimal solution $\boldsymbol{z}^*$.

9.3.1 Applying the Implicit Function Theorem

Let $C(\boldsymbol{y}, \boldsymbol{z})$ be the t-SNE cost function. Given a specific high-dimensional input $\boldsymbol{y}^*$, a t-SNE implementation finds an optimal (or locally optimal) embedding $\boldsymbol{z}^*$ that minimizes $C(\boldsymbol{y}^*, \boldsymbol{z})$. A necessary condition for this

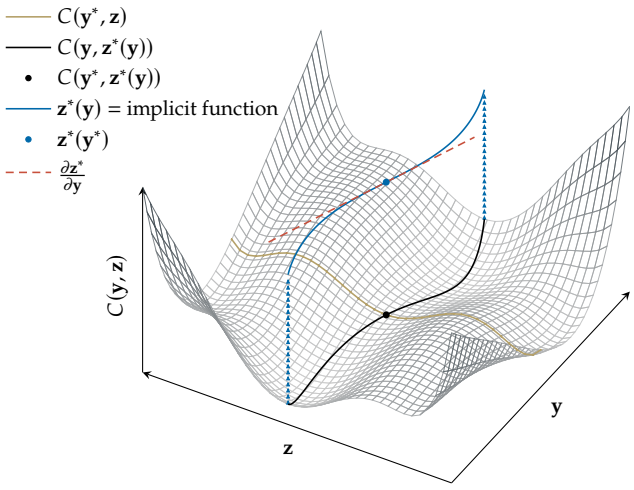


Figure 9.1: Simplified representation of the t-SNE loss function $C(\boldsymbol{y}, \boldsymbol{z})$. The gold line depicts the cross-section of the cost function for a fixed input $\boldsymbol{y}^*$, with its minimum at $\boldsymbol{z}^*$ (black dot) found by the t-SNE optimizer. The blue line, projected onto the $(\boldsymbol{y}, \boldsymbol{z})$-plane, represents the implicit function $\boldsymbol{z}^*(\boldsymbol{y})$, which maps any input $\boldsymbol{y}$ to its optimal embedding $\boldsymbol{z}$. The derivative of interest, $\frac{\partial \boldsymbol{z}^*(\boldsymbol{y})}{\partial \boldsymbol{y}}$ evaluated at $\boldsymbol{y}^*$, is the slope of the tangent to this implicit function (red dashed line) at the point $(\boldsymbol{y}^*, \boldsymbol{z}^*(\boldsymbol{y}^*))$. Note that $\boldsymbol{y}$ and $\boldsymbol{z}$ are high-dimensional in practice.

optimum $(\boldsymbol{y}^*, \boldsymbol{z}^*)$ is that the gradient of the cost function with respect to the embedding $\boldsymbol{z}$ must be zero:

$$\nabla_z C(\boldsymbol{y}^*, \boldsymbol{z}^*) = \mathbf{0}. \tag{9.3}$$

This stationary point condition is the key to our method. It implicitly defines the optimal embedding $\boldsymbol{z}^*$ as a function of the input $\boldsymbol{y}$. Figure 9.1 provides a visual intuition for this principle: while a standard t-SNE optimizer finds a single point minimum (black dot) for a fixed input $\boldsymbol{y}^*$, the condition $\nabla_z C = \mathbf{0}$ defines an entire curve or surface of optimal solutions, $\boldsymbol{z}^*(\boldsymbol{y})$ (blue line), that traces how the minimum shifts as the input $\boldsymbol{y}$ changes.

The implicit function theorem gives a formula for the Jacobian of this solution map, $\boldsymbol{z}^*(\boldsymbol{y})$, in terms of the derivatives of the function that defines the implicit system—in this case, the gradient function $\nabla_z C(\boldsymbol{y}, \boldsymbol{z})$.

The theorem requires the Jacobians of this gradient function with respect to both $\boldsymbol{y}$ and $\boldsymbol{z}$:

- The Jacobian with respect to $\boldsymbol{z}$ is the Hessian of the cost function: $\frac{\partial}{\partial \boldsymbol{z}}(\nabla_z C) = \frac{\partial^2 C}{\partial \boldsymbol{z}^2}$.
- The Jacobian with respect to $\boldsymbol{y}$ is the mixed-partial derivative matrix: $\frac{\partial}{\partial \boldsymbol{y}}(\nabla_z C) = \frac{\partial^2 C}{\partial \boldsymbol{y} \partial \boldsymbol{z}}$.

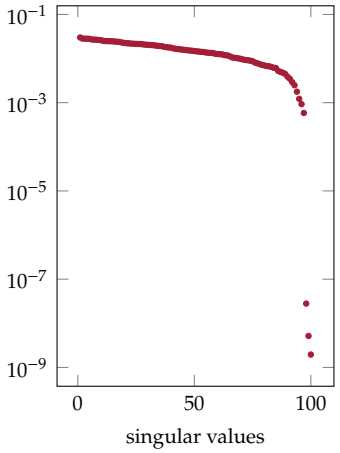
Substituting these components into the IFT formula for stationary points (Equation (5.5)) yields:

$$\frac{\partial \boldsymbol{z}^*(\boldsymbol{y})}{\partial \boldsymbol{y}} = -\left[\frac{\partial^2 C(\boldsymbol{y}, \boldsymbol{z}^*)}{\partial \boldsymbol{z}^2}\right]^{-1} \frac{\partial^2 C(\boldsymbol{y}, \boldsymbol{z}^*)}{\partial \boldsymbol{y} \partial \boldsymbol{z}}. \tag{9.4}$$

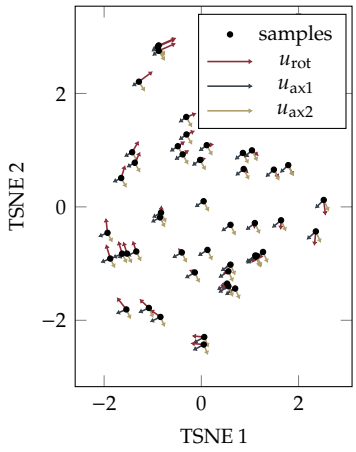
However, a critical challenge arises: the Hessian matrix $\boldsymbol{H}_{zz} = \frac{\partial^2 C}{\partial \boldsymbol{z}^2}$ is singular for the standard t-SNE cost function, which means its inverse does not exist.

Singularity of the t-SNE Cost Function Hessian

The t-SNE cost function $C(\boldsymbol{y}, \boldsymbol{z})$ depends on the low-dimensional coordinates $\boldsymbol{z}$ only through the pairwise Euclidean distances $||\boldsymbol{z}_i - \boldsymbol{z}_j||^2$



(a) Singular values of the Hessian.



(b) Directions of invariance.

Figure 9.2: ***(a)*** The singular value spectrum of the Hessian matrix $\boldsymbol{H}_{zz}$ computed at an optimal embedding z^*. Three singular values are effectively zero. ***(b)*** Visualization of the left singular vectors (reshaped from $\mathbb{R}^{2N}$ to $\mathbb{R}^{N\times 2}$ as per Equation (9.6)) corresponding to the zero singular values. These vectors depict infinitesimal transformations of the embedding: two translations (along orthogonal axes) and one rotation around the embedding's centroid, under which the t-SNE cost function value remains invariant.

used to compute the similarities q_{ij} (Equation (2.14)). Consequently, the cost function is invariant to transformations of the embedding $\boldsymbol{Z}$ (and thus z) that preserve these pairwise distances, namely global translations and rotations of the entire embedding. This invariance implies that the cost function is flat along the directions corresponding to these transformations, leading to a singular Hessian $\boldsymbol{H}_{zz}$.

To illustrate, consider an example dataset with $\boldsymbol{Y}^* \in \mathbb{R}^{50\times 50}$ (so $\boldsymbol{y}^* \in \mathbb{R}^{2500}$) and its t-SNE embedding $\boldsymbol{Z}^* \in \mathbb{R}^{50\times 2}$ (so $z^* \in \mathbb{R}^{100}$). We compute the Hessian $\boldsymbol{H}_{zz}|_{(\boldsymbol{y}^*,z^*)} \in \mathbb{R}^{100\times 100}$ and its singular value decomposition (SVD):

$$\left.\frac{\partial^2 C}{\partial z^2}\right|_{(\boldsymbol{y}^*,z^*)} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top. \tag{9.5}$$

Figure 9.2a shows that the diagonal matrix $\boldsymbol{\Sigma}$ contains three singular values that are numerically zero (up to floating-point precision), while the remaining 97 are positive. This indicates that the cost function has zero curvature in a three-dimensional subspace at $(\boldsymbol{y}^*, z^*)$. The left singular vectors in $\boldsymbol{U}$ corresponding to these zero singular values, denoted $\boldsymbol{u}_{\text{trans1}}$, $\boldsymbol{u}_{\text{trans2}}$, and $\boldsymbol{u}_{\text{rot}}$, span this null space. When reshaped from $\mathbb{R}^{2N}$ to $\mathbb{R}^{N\times 2}$ (where each row i forms a 2D displacement vector for point $\boldsymbol{Z}_i^*$), these vectors represent:

- $\boldsymbol{u}_{\text{trans1}}$: Translation of the entire embedding $\boldsymbol{Z}^*$ along one coordinate axis.
- $\boldsymbol{u}_{\text{trans2}}$: Translation of $\boldsymbol{Z}^*$ along the perpendicular coordinate axis.
- $\boldsymbol{u}_{\text{rot}}$: Rotation of $\boldsymbol{Z}^*$ around its sample mean $\frac{1}{N}\sum_i \boldsymbol{Z}_i^*$.

Figure 9.2b visually illustrates these directions of invariance. The reshaping operation is:

$$\operatorname{vec}^{-1}\left(\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{2N} \end{bmatrix}\right) = \begin{bmatrix} u_1 & u_2 \\ u_3 & u_4 \\ \vdots & \vdots \\ u_{2N-1} & u_{2N} \end{bmatrix} \in \mathbb{R}^{N\times 2}. \tag{9.6}$$

This degeneracy means the standard IFT formula (Equation (9.4)) cannot be directly applied due to the non-invertible Hessian.

Applying the IFT using the Moore-Penrose Pseudoinverse

In typical interpretations of t-SNE embeddings, the absolute position and orientation of the entire point cloud are irrelevant; only the relative positions of points and the structure of clusters matter. The uncertainty associated with these global transformations (translations, rotations) is therefore not of primary interest.

To address the singular Hessian while focusing on meaningful variations, we employ the Moore-Penrose pseudoinverse (†) of the Hessian matrix in the IFT formula. The pseudoinverse $\boldsymbol{H}_{zz}^{\dagger}$ inverts the transformation on the subspace of meaningful variations (the range space of $\boldsymbol{H}_{zz}$) while correctly mapping the irrelevant invariance directions (the null space of $\boldsymbol{H}_{zz}$) to zero. This yields our final expression for the Jacobian:

$$\left.\frac{\partial \boldsymbol{z}^*(\boldsymbol{y})}{\partial \boldsymbol{y}}\right|_{\boldsymbol{y}^*} = -\left[\left.\frac{\partial^2 C}{\partial \boldsymbol{z}^2}\right|_{(\boldsymbol{y}^*,\boldsymbol{z}^*)}\right]^{\dagger} \left.\frac{\partial^2 C}{\partial \boldsymbol{y}\partial \boldsymbol{z}}\right|_{(\boldsymbol{y}^*,\boldsymbol{z}^*)}. \tag{9.7}$$

The pseudoinverse is computed via SVD: if $\boldsymbol{H}_{zz} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\top}$, then $\boldsymbol{H}_{zz}^{\dagger} = \boldsymbol{V}\boldsymbol{\Sigma}^{\dagger}\boldsymbol{U}^{\top}$, where $\boldsymbol{\Sigma}^{\dagger}$ is formed by taking the reciprocal of the non-zero singular values in $\boldsymbol{\Sigma}$.

Applying the IFT offers a significant practical advantage: the Jacobian $\frac{\partial \boldsymbol{z}^*(\boldsymbol{y})}{\partial \boldsymbol{y}}$ can be computed using only second-order derivatives of the cost function C evaluated at the single solution point $(\boldsymbol{y}^*, \boldsymbol{z}^*)$. This avoids differentiating through the entire optimization algorithm and is agnostic to the specific solver used to find $\boldsymbol{z}^*$.

9.3.2 Gaussian Error Propagation through t-SNE

With the Jacobian in hand, we can propagate the input uncertainty. Given the input data $\boldsymbol{y}$ distributed as $p(\boldsymbol{y}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{y}^*, \boldsymbol{\Sigma}_y)$, we approximate the distribution of the embedding $p(\boldsymbol{z}^* \mid \boldsymbol{y})$ using a first-order Taylor expansion (Equation (3.29)) of the solution map $\boldsymbol{z}^*(\boldsymbol{y})$ around $\boldsymbol{y}^*$:

$$\boldsymbol{z}^*(\boldsymbol{y}) \approx \boldsymbol{z}^*(\boldsymbol{y}^*) + \left.\frac{\partial \boldsymbol{z}^*(\boldsymbol{y})}{\partial \boldsymbol{y}}\right|_{\boldsymbol{y}^*} (\boldsymbol{y} - \boldsymbol{y}^*). \tag{9.8}$$

This linear approximation implies that if $\boldsymbol{y}$ is Gaussian, then $\boldsymbol{z}^*(\boldsymbol{y})$ is approximately Gaussian. The mean of this distribution is $\mathbb{E}[\boldsymbol{z}^*(\boldsymbol{y})] \approx \boldsymbol{z}^*(\boldsymbol{y}^*)$, and its covariance matrix $\boldsymbol{\Sigma}_z \in \mathbb{R}^{2N\times 2N}$ is given by Gaussian error propagation:

$$\boldsymbol{\Sigma}_z \approx \left(\left.\frac{\partial \boldsymbol{z}^*(\boldsymbol{y})}{\partial \boldsymbol{y}}\right|_{\boldsymbol{y}^*}\right) \boldsymbol{\Sigma}_y \left(\left.\frac{\partial \boldsymbol{z}^*(\boldsymbol{y})}{\partial \boldsymbol{y}}\right|_{\boldsymbol{y}^*}\right)^{\top}, \tag{9.9}$$

where the Jacobian term is computed via the IFT (Equation (9.7)).

9.3.3 Efficient Computation using Jacobian-Vector and Vector-Jacobian Products

For notational convenience in this section, let $\boldsymbol{H}_{zz} = \frac{\partial^2 C}{\partial z \partial z}\Big|_{(\boldsymbol{y}^*, \boldsymbol{z}^*)}$ be the Hessian with respect to the embedding and $\boldsymbol{J}_{yz} = \frac{\partial^2 C}{\partial \boldsymbol{y} \partial z}\Big|_{(\boldsymbol{y}^*, \boldsymbol{z}^*)}$ be the mixed-partial derivative matrix. The Jacobian of the embedding is then $\boldsymbol{J}_{z^*} = -\boldsymbol{H}_{zz}^{\dagger} \boldsymbol{J}_{yz}$.

The input covariance $\boldsymbol{\Sigma}_y$ and the mixed Jacobian $\boldsymbol{J}_{yz} \in \mathbb{R}^{2N \times ND}$ can be very large, making the explicit formation of $\boldsymbol{J}_{z^*}$ or the final output covariance $\boldsymbol{\Sigma}_z = \boldsymbol{J}_{z^*} \boldsymbol{\Sigma}_y \boldsymbol{J}_{z^*}^{\intercal}$ computationally prohibitive. Instead, we can compute $\boldsymbol{\Sigma}_z$ (or just its diagonal) more efficiently by leveraging matrix-free methods. To compute the k-th column of $\boldsymbol{\Sigma}_z$, we compute its product with the k-th standard basis vector $\boldsymbol{e}_k$:

$$\begin{aligned} \boldsymbol{\Sigma}_z \boldsymbol{e}_k &= (\boldsymbol{J}_{z^*} \boldsymbol{\Sigma}_y \boldsymbol{J}_{z^*}^{\intercal}) \boldsymbol{e}_k \\ &= (\boldsymbol{H}_{zz}^{\dagger} \boldsymbol{J}_{yz}) \boldsymbol{\Sigma}_y (\boldsymbol{J}_{yz}^{\intercal} (\boldsymbol{H}_{zz}^{\dagger})^{\intercal}) \boldsymbol{e}_k. \end{aligned} \tag{9.10}$$

This expression can be computed from right to left via a sequence of matrix-vector products, leveraging automatic differentiation frameworks like JAX to provide Jacobian-vector products (JVPs) and vector-Jacobian products (VJPs) without explicitly forming the large Jacobian matrices:

1. $\boldsymbol{v}_a = (\boldsymbol{H}_{zz}^{\dagger})^{\intercal} \boldsymbol{e}_k$ (pseudoinverse-transpose-vector product)
2. $\boldsymbol{v}_b = \boldsymbol{J}_{yz}^{\intercal} \boldsymbol{v}_a$ (VJP: multiply by transpose of mixed Jacobian)
3. $\boldsymbol{v}_c = \boldsymbol{\Sigma}_y \boldsymbol{v}_b$ (input covariance-vector product). If $\boldsymbol{\Sigma}_y = \boldsymbol{D}_0 \otimes \boldsymbol{N}_0$, then $\boldsymbol{v}_c = \mathrm{vec}(\boldsymbol{D}_0 \cdot \mathrm{vec}^{-1}(\boldsymbol{v}_b) \cdot \boldsymbol{N}_0^{\intercal})$, efficiently computed.
4. $\boldsymbol{v}_d = \boldsymbol{J}_{yz} \boldsymbol{v}_c$ (JVP: multiply by mixed Jacobian)
5. $\boldsymbol{\Sigma}_z[:, k] = \boldsymbol{H}_{zz}^{\dagger} \boldsymbol{v}_d$ (pseudoinverse-vector product)

Each step involves operations that are more tractable than forming the full matrices. The entire matrix $\boldsymbol{\Sigma}_z$ can be constructed by repeating this process for each basis vector $\boldsymbol{e}_k$. Note that $\boldsymbol{v}_a$ is just the k-th column of $\boldsymbol{H}_{zz}^{\dagger}$.

9.3.4 Implementation Details

The IFT-based approach decouples the computation of the optimal embedding $\boldsymbol{z}^*$ from the computation of its sensitivity $\boldsymbol{J}_{z^*}$. This modularity allows us to use any standard, efficient t-SNE solver to obtain $\boldsymbol{z}^*$ from $\boldsymbol{y}^*$.

Our subsequent implementation centers on a JAX function that computes the gradient of the t-SNE cost function with respect to the embedding, $\nabla_z C(\boldsymbol{y}, \boldsymbol{z})$. This single function is the cornerstone for all required derivative calculations. Figure 9.3 outlines this process.

- Figure 9.3a: The process starts by defining a function returning an optimal embedding $\boldsymbol{z}^*$ using an external solver. Separately, we define a JAX-compatible function, that implements $\nabla_z C(\boldsymbol{y}, \boldsymbol{z})$.
- Figure 9.3b: Using the solution $(\boldsymbol{y}^*, \boldsymbol{z}^*)$, the Hessian $\boldsymbol{H}_{zz}$ of the cost function and its pseudoinverse are computed and evaluated at the solution point. Furthermore, we define function transformations to compute VJPs and JVPs of the mixed-partial derivative $\boldsymbol{J}_{yz}$.

- Figure 9.3c: The `cvp`-function implements the right-to-left sequence of matrix-vector products described in Section 9.3.3 (Equation (9.10) and subsequent steps) to compute a single column of the output covariance matrix $\mathbf{\Sigma}_z$.
- Figure 9.3d: The full covariance matrix $\mathbf{\Sigma}_z$ is constructed by applying `cvp` to each column of $\boldsymbol{H}_{zz}^{\dagger}$ using JAX's `vmap` for efficient batching over columns.

This JAX-based implementation is primarily designed for CPU execution but can be adapted for GPU acceleration or distributed computation using `jax.pmap` for larger datasets, which can also offer memory benefits over `jax.vmap` for very large numbers of columns.

```python
import jax.numpy as np
from utils import y2p, z2q

def get_optimal_embedding_Z_star(Y_star_matrix, perp, tsne_kwargs):
    '''Computes the optimal t-SNE embedding Z_star from high dimensional data Y_star.
     This function would call an external t-SNE library (e.g., sklearn.manifold.TSNE
        or openTSNE) and return the N x P matrix Z_star.'''
    ...
    return Z_star

def cost_dz(Y, Z, perp):
    '''Computes gradient of t-SNE cost C(Y,Z) w.r.t. Z (embedding).
        This is g_Z(Y,Z) = dC/dZ. Returns an N x P matrix..'''

    # compute neighbor probabilities P and Q from coordinates
    P = y2p(Y, perp)
    Q = z2q(Z)

    # gradient of cost function w.r.t. Z
    return 4 * np.sum((P - Q) * (Y_diffs) * (1 / 1 + Y_dists), axis=1)

```

(a) t-SNE solver & gradient of t-SNE cost function $\frac{\partial C}{\partial y}$

```python
from jax import grad, vjp, jacrev
from functools import partial

# Optimal embedding
Z_star = Z_star(Y)

# fix perplexity argument
fun = partial(cost_dz, perp=30.0)

# Compute Hessian and its pseudoinverse
H = jacrev(fun, argnums=1)(Y, Z_star)
H_pinv = np.linalg.pinv(H)

# fix Z argument and perplexity argument
f = partial(cost_dz, Z=Z_star, perp=30.0)

# Vector-mixed Jacobian product
_, f_vjp = vjp(f, Y)

# Mixed Jacobian-vector product
_, f_jvp = jax.linearize(f, Y)
```

(b) Hessian pseudoinverse, JVP & VJP

```python
def cvp(f_vjp, f_jvp, v_a, D, N, d, n, H_pinv):
    '''Computes the Sigma_Z-vector product'''

    v_b = f_vjp(-v_a)
    v_c = np.ravel((D @ np.reshape(v_b, (d, n), 'C') @ N), 'C
     ')
    v_d = f_jvp(v_c)

    return np.dot(-H_pinv, v_d)
```

(c) Function to compute $\mathbf{\Sigma}_Z[i]$

```python
from jax import vmap

#Lambda expression fixing all arguments but index i
f_cvp = lambda i: cvp(f_vjp, f_jvp, i, D, N, d, n, H_pinv)

# Compute Sigma_z
Sigma_z = vmap(f_cvp)(H_pinv)
```

(d) Looping

Figure 9.3: Simplified JAX implementation for computing the embedding covariance $\mathbf{\Sigma}_z$. Variables often correspond to their mathematical counterparts (e.g., `Y_star` $\rightarrow \boldsymbol{Y}^*$, `Z_star` $\rightarrow \boldsymbol{Z}^*$). **(a)** Obtaining $\boldsymbol{z}^*$ via an external solver and defining $\nabla_z C$ in JAX. **(b)** Computing the pseudoinverse of the Hessian $\boldsymbol{H}_{zz}$ and setting up functions for JVPs/VJPs related to the mixed Jacobian $\boldsymbol{J}_{yz}$. **(c)** Function implementing the sequence of matrix-vector products to compute a single column of $\mathbf{\Sigma}_z$. **(d)** Using `jax.vmap` to efficiently compute all columns of $\mathbf{\Sigma}_z$.

9.4 Showcases

To demonstrate the validity and interpretability of the sensitivities computed for t-SNE via the implicit function theorem (IFT), we present two illustrative examples: a controlled, small-scale simulation and a real-world, large-scale gene expression time-series experiment. These showcases highlight both the theoretical soundness and the practical utility of our approach.

9.4.1 Simulations

To evaluate whether the sensitivities $\boldsymbol{J}_{z^*} = \left.\frac{\partial z^*(y)}{\partial y}\right|_{y^*}$ derived via the IFT are interpretable and geometrically meaningful, we designed a set of controlled simulations. Each simulated dataset comprises three 5-dimensional data points ($\boldsymbol{Y} \in \mathbb{R}^{3\times5}$), intentionally configured such that the points differ only along a single dimension (the fifth dimension), effectively lying in a one-dimensional affine subspace.

We define three example mean data matrices, $\boldsymbol{Y}^{*(k)} \in \mathbb{R}^{3\times5}$, each with distinct geometric properties:

$$\boldsymbol{Y}^{*(1)} = \begin{bmatrix} 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\boldsymbol{Y}^{*(2)} = \begin{bmatrix} 0 & 0 & 0 & 0 & -5 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 \end{bmatrix},$$

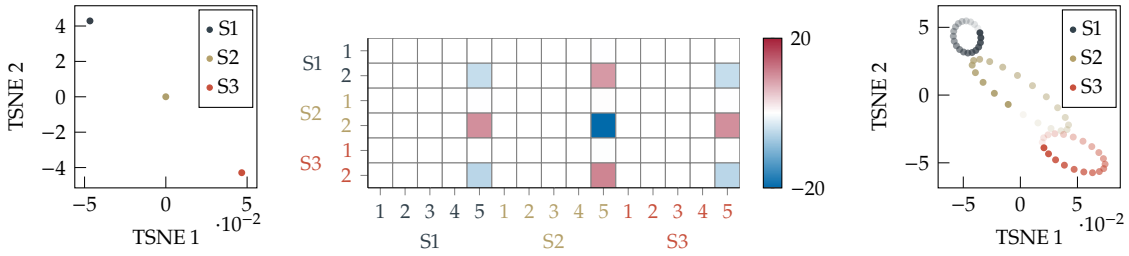
$$\boldsymbol{Y}^{*(3)} = \begin{bmatrix} 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0.3 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

These configurations are:

- $\boldsymbol{Y}^{*(1)}$: Three samples, equidistant along the fifth dimension, centered at zero.
- $\boldsymbol{Y}^{*(2)}$: Same relative geometry as $\boldsymbol{Y}^{*(1)}$, but with distances between samples scaled by a factor of 5.
- $\boldsymbol{Y}^{*(3)}$: Asymmetric placement of the central sample along the fifth dimension, breaking the symmetry of $\boldsymbol{Y}^{*(1)}$.

By applying our implicit differentiation method (Section 9.3.1) to these datasets, we assess whether the computed sensitivities and propagated uncertainties align with intuitive expectations based on the data geometry. For uncertainty propagation, we assume an isotropic input covariance $\boldsymbol{\Sigma}_y = \sigma^2 \boldsymbol{I}_{ND}$ with $\sigma^2 = 0.01$ for the vectorized input $\boldsymbol{y}^{*(k)}$, for $k \in \{1, 2, 3\}$.

Figure 9.4a, Figure 9.5a, and Figure 9.6a show the 2D t-SNE embeddings $\boldsymbol{Z}^{*(k)}$ for the three synthetic datasets. Notably, the embeddings $\boldsymbol{Z}^{*(1)}$ and $\boldsymbol{Z}^{*(2)}$ are structurally almost identical, preserving the relative linear arrangement of samples despite the differing scales of input distances. However, the associated sensitivities (Jacobians $\boldsymbol{J}_{z^{*(k)}}$), visualized in panel (b) of these figures, differ substantially in magnitude. The derivatives for $\boldsymbol{Y}^{*(1)}$ are approximately five times larger in magnitude than those for $\boldsymbol{Y}^{*(2)}$ (compare Figure 9.4b and Figure 9.5b).



(a) t-SNE plot **(b)** Sensitivities **(c)** Propagated uncertainties

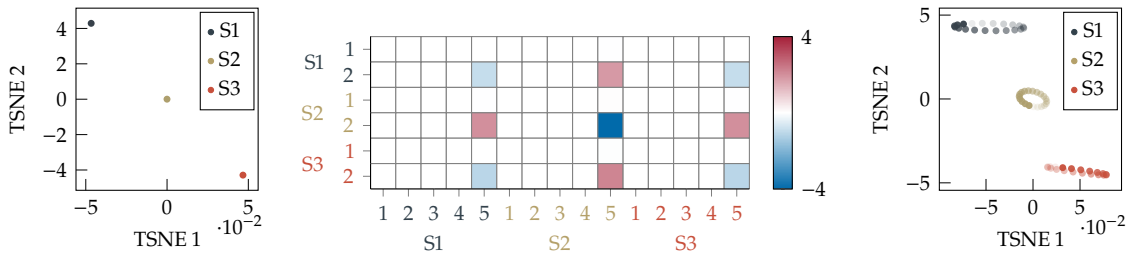
Figure 9.4: Results for simulated dataset $\boldsymbol{Y}^{*(1)}$ (equidistant samples). **(a)** t-SNE embedding $\boldsymbol{Z}^{*(1)}$. **(b)** Visualization of the Jacobian $\boldsymbol{J}_{\boldsymbol{z}^{*(1)}}$, showing the sensitivity of embedding coordinates to perturbations in the input features. **(c)** Hypothetical outcomes of the embedding distribution illustrating the embedding uncertainties derived via approximate Gaussian error propagation.

This observation highlights a key property: while the t-SNE embedding's qualitative geometric structure can be invariant to uniform rescaling of input distances, its sensitivity to perturbations (and thus its stability) is not. Although the relative sensitivity patterns across samples are preserved (i.e., which input dimensions affect which embedding coordinates), the absolute magnitude of the derivatives reciprocally scales with the input distances. Consequently, small perturbations in $\boldsymbol{y}^{*(1)}$ induce qualitatively similar but quantitatively amplified changes in its embedding $\boldsymbol{z}^{*(1)}$ compared to $\boldsymbol{z}^{*(2)}$.

A detailed inspection of the Jacobians for datasets 1 and 2 (Figure 9.4b, Figure 9.5b) reveals that non-zero derivatives arise exclusively from perturbations in the fifth input dimension of each sample. This is consistent with the experimental design, as samples only differ in that coordinate. The structure of these derivatives offers an intuitive geometric interpretation: for example, for dataset $\boldsymbol{Y}^{*(1)}$, consider a small positive shift in the fifth coordinate of the central sample (gold). The tenth column of the Jacobian (Figure 9.4b) indicates that this shift will produce a negative change in its own TSNE 2 coordinate, while producing positive changes for the two outer samples. Geometrically, this means the central sample's embedding moves down as the outer samples' embeddings move up. Referencing the t-SNE plot (Figure 9.4a), this movement pushes the central sample closer to the third (red) sample and further from the first (gray) sample. Perturbing either of the outer samples induces symmetrical effects (see the sensitivities in columns 5 and 15 of the Jacobian in Figure 9.4b). These effects also align with expectations.

In contrast, the sensitivity analysis for dataset 3 (Figure 9.6b) reveals Jacobians that are nearly zero across all input dimensions. This suggests that the corresponding embedding $\boldsymbol{z}^{*(3)}$ is particularly robust to small input perturbations. The asymmetric placement of the central point in $\boldsymbol{Y}^{*(3)}$ appears to stabilize this particular embedding configuration.

Panel (c) in Figure 9.4, Figure 9.5, and Figure 9.6 shows the propagated uncertainty in the embeddings, visualized as hypothetical outcomes of the infered distributions $p(\boldsymbol{Z} \mid \boldsymbol{Y})$. For dataset 1 (Figure 9.4c), the combination of high sensitivities and input noise yields relatively large uncertainty ellipses. The orientation of these ellipses reflects the sensitivity patterns: the central point's uncertainty is elongated along the axis connecting the three points, indicating higher uncertainty in its relative position to the outer points. Dataset 2 (Figure 9.5c) exhibits smaller uncertainty ellipses



(a) t-SNE plot **(b)** Sensitivities **(c)** Propagated uncertainties

Figure 9.5: Results for simulated dataset $\boldsymbol{Y}^{*(2)}$ (equidistant, larger separation). Panels are analogous to Figure 9.4. Note the reduced magnitude of sensitivities **(b)** and smaller uncertainty ellipses **(c)** compared to $\boldsymbol{Y}^{*(1)}$, despite identical input uncertainty levels.

(a) t-SNE plot **(b)** Sensitivities **(c)** Propagated uncertainties

Figure 9.6: Results for simulated dataset $\boldsymbol{Y}^{*(3)}$ (asymmetric placement). Panels are analogous to Figure 9.4. The sensitivities **(b)** are nearly zero, leading to very small uncertainty ellipses **(c)**, indicating a highly stable embedding for this configuration.

due to its reduced sensitivity magnitudes, despite the same input noise level. Conversely, the embedding for dataset 3 (Figure 9.6c) shows very small uncertainty ellipses, confirming its high stability.

These controlled experiments demonstrate that sensitivities computed via implicit differentiation and the subsequent uncertainty propagation provide geometrically intuitive and quantifiable insights into the stability of t-SNE embeddings. In the following subsection, we apply this analysis to a real-world gene expression dataset to assess the robustness of t-SNE embeddings under biologically plausible measurement noise.

9.4.2 Gene Expression Data

RNA-Seq Time Series of *Streptomyces coelicolor* M145

Streptomyces coelicolor (*S. coelicolor*) is the *de facto* model organism for the heterologous expression[1] of biosynthetic gene clusters (BGCs) including those involved in antibiotic production. The suitability of *S. coelicolor* as a host for heterologous expression is largely due to its metabolic capacity, which supplies the precursor molecules necessary for BGC-derived compound synthesis. Many of these precursor molecules are generated under nutrient-limiting conditions, which trigger growth arrest and extensive metabolic reprogramming [258].

1: Heterologous expression refers to the process in which a foreign gene, introduced from another species, is expressed in a host cell, enabling the production of the corresponding protein.

In this study, we analyze a time-series gene expression dataset from *S. coelicolor* M145 cultivated under progressive phosphate depletion (unpublished data), a condition known to trigger a metabolic switch towards secondary metabolism, including antibiotic synthesis [258].

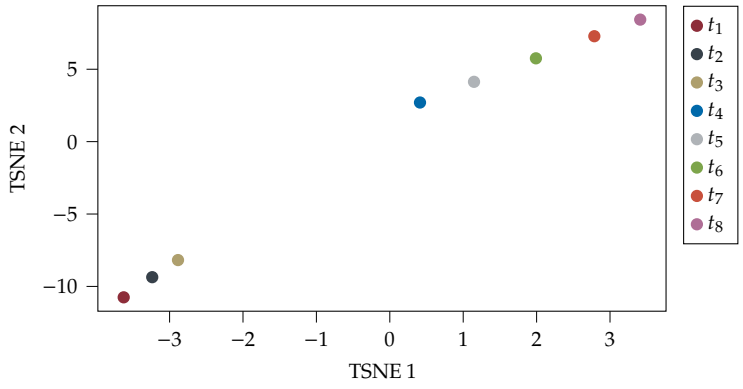


Figure 9.7: t-SNE embedding of mean gene expression profiles from *S. coelicolor* M145 across eight timepoints (t_1–t_8) during progressive phosphate depletion.

RNA sequencing was performed at eight timepoints (t_1 to t_8), with three biological replicates per timepoint. The primary metabolic switch is reported to occur between t_3 and t_4.

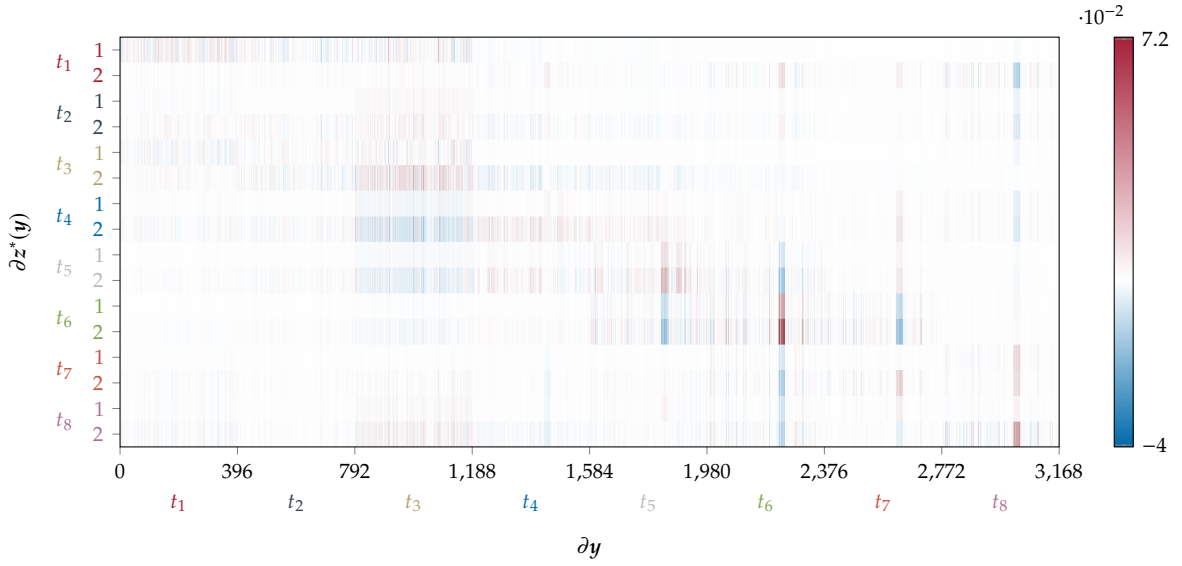
For data preprocessing, we first focused on the most variable genes. From the initial 7908 genes, we calculated the variance of each gene's expression across the eight timepoints (using the mean expression per timepoint from replicates) and retained the top 5% most variant genes, resulting in 396 genes for analysis. The mean expression matrix is thus $\boldsymbol{Y}^* \in \mathbb{R}^{8\times 396}$ (8 timepoints, 396 genes).

To model input uncertainty, for each of these 396 genes at each of the 8 timepoints, we estimated the variance of its mean expression value from the three biological replicates. Specifically, if $x_{g,t,r}$ is the expression of gene g at timepoint t in replicate r, then $\bar{x}_{g,t} = \frac{1}{3}\sum_r x_{g,t,r}$ is the entry in $\boldsymbol{Y}^*$. The variance of this mean is $\mathrm{Var}(\bar{x}_{g,t}) = s^2_{g,t}/3$, where $s^2_{g,t}$ is the sample variance of $x_{g,t,r}$ across replicates. These variances form the diagonal elements of the input covariance matrix $\boldsymbol{\Sigma}_y \in \mathbb{R}^{(8\cdot 396)\times(8\cdot 396)}$, assuming independence of uncertainties between gene measurements. All off-diagonal elements of $\boldsymbol{\Sigma}_y$ are zero.

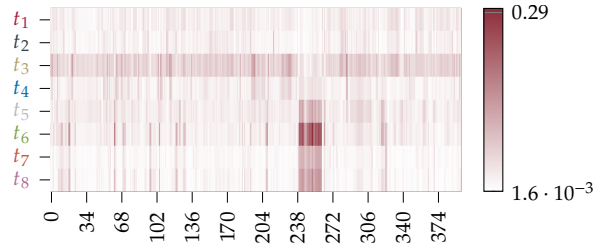
Applying standard t-SNE to the mean expression matrix $\boldsymbol{Y}^*$ yields the embedding $\boldsymbol{Z}^*$ shown in Figure 9.7. The embedding clearly separates early timepoints (t_1–t_3) from later stages (t_4–t_8), reflecting the known metabolic switch. We confirmed that the gradient $\frac{\partial C(y,z)}{\partial z}$ evaluated at $(\boldsymbol{y}^*, \boldsymbol{z}^*)$ is effectively zero, satisfying the prerequisite (Equation (5.4)) for applying the IFT in this setting.

Using our IFT-based method, we computed the Jacobian $\boldsymbol{J}_{z^*} = \left.\frac{\partial z^*(y)}{\partial y}\right|_{y^*}$. This matrix, of size $(8 \cdot 2) \times (8 \cdot 396)$, quantifies how each of the 16 embedding coordinates (2 per timepoint) responds to changes in each of the 8×396 input gene expression values. Figure 9.8a shows this Jacobian as a heatmap. Many embedding coordinates are largely insensitive to gene expression changes at early timepoints t_1 and t_2 (columns 0–791, corresponding to genes at t_1 and t_2). Some sensitivities are observed for genes at t_3 and t_4, particularly affecting the second t-SNE dimension of samples t_3–t_5. Additionally, several coordinates show higher sensitivity to a specific subset of genes at timepoints t_5–t_8. Overall, the magnitudes of the derivatives are relatively low, suggesting a generally stable embedding.

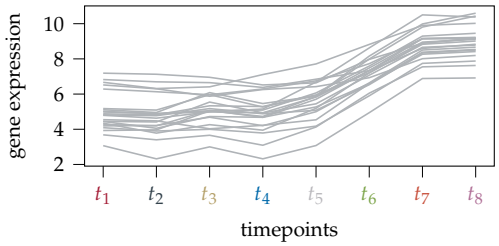
To identify genes and timepoints with the most significant influence on



(a) Jacobian of sensitivities.



(b) Summarized sensitivities.



(c) Profile plots of BCG genes for actinorhodin.

Figure 9.8: (a) Heatmap of the Jacobian J_{z^*}. Rows correspond to the $8 \times 2 = 16$ output embedding coordinates $(z_1^*(t_1), z_2^*(t_1), \ldots, z_2^*(t_8))$. Columns correspond to the $8 \times 396 = 3168$ input features (expression of gene 1 at t_1, gene 2 at $t_1, \ldots$, gene 396 at t_8). **(b)** Summarized sensitivities: for each input feature (gene at a specific timepoint), the sum of absolute values of its derivatives across all 16 output embedding coordinates is shown, reshaped into an 8×396 matrix (timepoints × genes). **(c)** Mean expression profiles (log-scale) of genes identified as highly influential, belonging to the actinorhodin biosynthetic gene cluster.

the embedding, we summarized the Jacobian. For each input feature (gene g at timepoint t), we summed the absolute values of its derivatives across all 16 embedding coordinates: $\sum_{k=1}^{16} |(J_{z^*})_{k,(g,t)}|$. This yields a per-feature total sensitivity score. Figure 9.8b visualizes these scores, reshaped as an 8×396 matrix. A distinct subset of genes (annotated as genes 240–261 in our filtered list) at later timepoints (t_5–t_8) exhibit high total sensitivity. These genes correspond to the biosynthetic gene cluster (BGC) for the antibiotic actinorhodin (BGC0000194). Their expression profiles (Figure 9.8c) show a characteristic sharp upregulation from t_5 onwards, consistent with the onset of secondary metabolism.

Using the computed Jacobian J_{z^*} and the input covariance Σ_y, we propagated the uncertainty via Equation (9.9) to obtain Σ_z. Figure 9.9 visualizes hypothetical outcomes from the resulting approximate Gaussian distribution $p(Z \mid Y)$ following the procedure described in Section 8.5, illustrating the uncertainty in the embedding. The respective animation can be found at `https://github.com/Integrative-Transcriptomics/tsne`. As anticipated from the sensitivity analysis, most timepoint embeddings are relatively stable. However, t_6 displays the largest uncertainty. Notably, the embeddings for t_4 and t_5 tend to co-vary, as do t_7 and t_8, main-

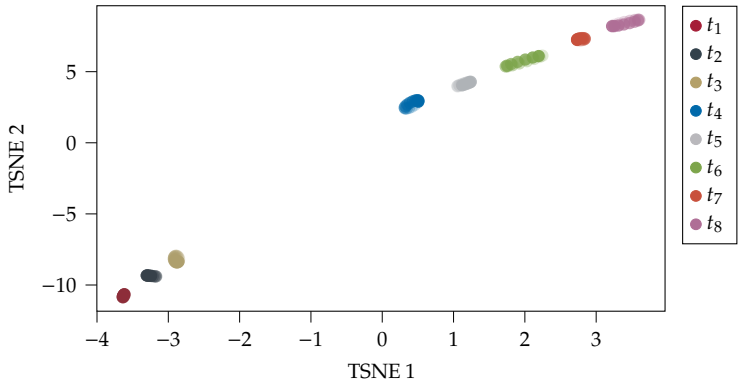


Figure 9.9: Visualization of uncertainty in the t-SNE embedding of *S. coelicolor* M145 data. Hypothetical outcomes of the embedding distribution $p(\boldsymbol{Z} \mid \boldsymbol{Y})$ illustrate the uncertainty for each timepoint's embedding.

taining their relative positions. An interesting dynamic is the apparent anti-correlation in uncertainty between the t_4, t_5 subcluster, the t_7, t_8 subcluster, and t_6. This results in t_6 appearing to shift its proximity, sometimes closer to (t_4, t_5) and sometimes closer to (t_7, t_8). This behavior is biologically plausible: t_6 represents a transitional state. For instance, actinorhodin BGC expression at t_6 lies between t_5 and t_7 (Figure 9.8c). Small perturbations in these genes at t_6 can thus shift its embedding towards either neighbor, a pattern reflected in the Jacobian structure for these specific genes and timepoints (see for example columns 2.220 to 2.241 of the Jacobian in Figure 9.8a).

In summary, this analysis demonstrates how propagating input uncertainty, guided by IFT-derived sensitivities, can link embedding stability to specific biological features (e.g., gene clusters) and provide interpretable insights into the robustness of t-SNE visualizations.

Microarray Time Series of *Streptomyces coelicolor* M145

To further evaluate our uncertainty propagation framework, we analyzed a second dataset from *Streptomyces coelicolor* M145, similar to that described in Section 9.4.2, but with key differences. This dataset features a more fine-grained temporal resolution, with samples taken at 32 distinct timepoints (labeled, for instance, t_{20} through t_{60}) during progressive phosphate depletion, allowing for a detailed analysis of the metabolic switch. However, unlike the RNA-Seq dataset, no biological replicates were available for these microarray measurements. Gene expression levels were measured for 7893 genes, from which the top 5 % most variant genes across the 32 conditions were selected for analysis. This resulted in an expression matrix $\boldsymbol{Y}^* \in \mathbb{R}^{32\times 395}$, which we treat as the mean expression in the absence of replicates[2], where each row corresponds to a timepoint and each column to a gene.

2: With no replicates, $\boldsymbol{Y}^*$ represents the observed point estimates, around which uncertainty is subsequently modeled.

Due to the absence of replicates, we modeled input uncertainties for all measurements and assumed independent measurement uncertainties for each gene at each timepoint, assigning variances drawn uniformly from the range $[0, 0.01]$. This resulted in a diagonal input covariance matrix $\boldsymbol{\Sigma}_y \in \mathbb{R}^{(32\cdot 395)\times(32\cdot 395)}$ for the vectorized mean data $\boldsymbol{y}^* = \mathrm{vec}(\boldsymbol{Y}^*)$. While these uncertainties are synthetic, they allow us to explore the sensitivity of the t-SNE embedding under a plausible noise model.

Applying standard t-SNE to the mean expression matrix $\boldsymbol{Y}^*$ yields the embedding $\boldsymbol{Z}^*$ shown in Figure 9.10. The embedding distinctly separates

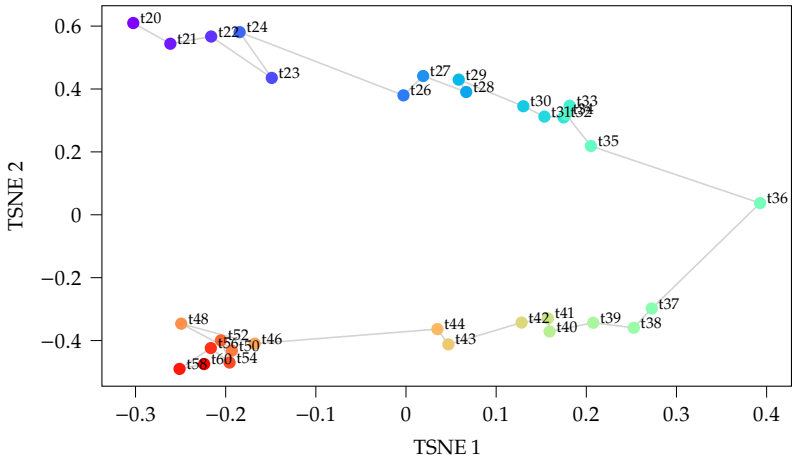
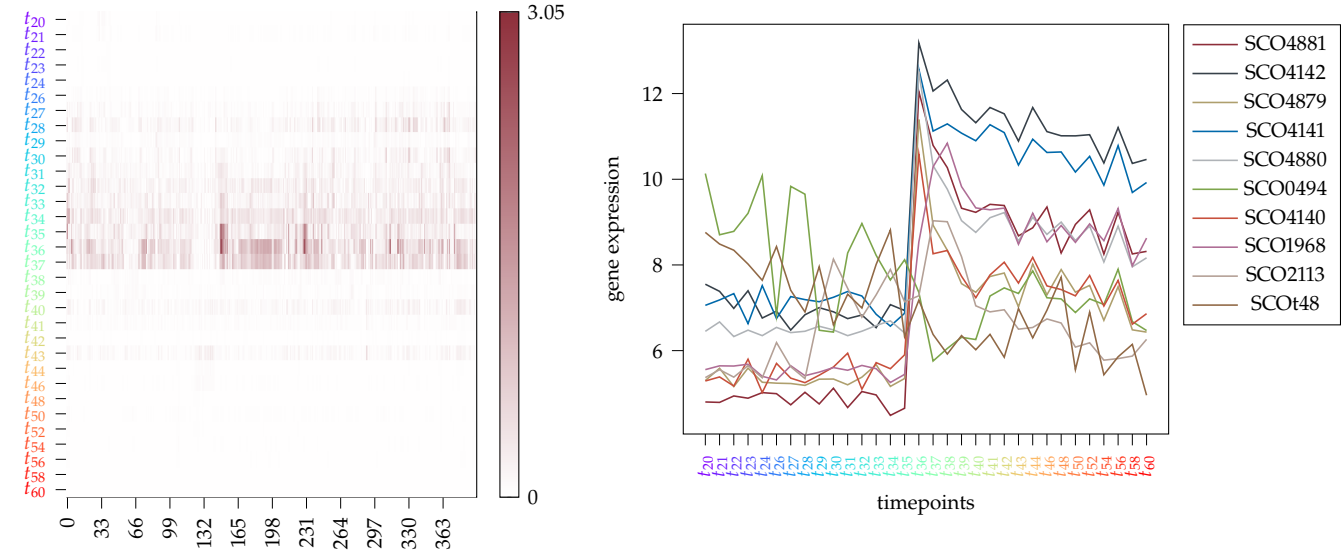


Figure 9.10: t-SNE embedding of mean gene expression profiles from *S. coelicolor* M145 across 32 timepoints ($t_{20}, \ldots, t_{60}$) during progressive phosphate depletion. Consecutive timepoints are connected via a gray line.

earlier timepoints (e.g., t_{20} through t_{35}) from later stages (e.g., t_{37} through t_{60}), with a transition period around t_{36}, consistent with the expected metabolic switch. We verified that the gradient $\frac{\partial C(y^*, z^*)}{\partial z}$ was effectively zero, satisfying the prerequisite (Equation (9.3)) for applying the IFT.

Using our IFT-based method, we computed the Jacobian $\boldsymbol{J}_{z^*} = \left.\frac{\partial z^*(y)}{\partial y}\right|_{y^*}$, which is visualized as a heatmap in Appendix C. Visual inspection of this Jacobian (Appendix C) revealed that the overall embedding is most sensitive to changes in gene expression values around the metabolic switch, particularly at timepoints t_{36} and t_{37}. Many embedding coordinates across all timepoints showed non-zero derivatives with respect to inputs from this transitional period. Specifically, the embedding coordinates of samples $t_{35}, t_{36}, t_{37}, t_{38}$, and t_{43} exhibited the highest sensitivities to changes in the gene expression levels at t_{36} and t_{37}. Compared to the sensitivities computed for the RNA-Seq dataset (Section 9.4.2), the sensitivity magnitudes observed for this microarray dataset were notably higher (up to approximately fourfold) and appeared more sparsely distributed across the Jacobian. This increased sensitivity could be attributed to the use of single point estimates for the microarray data, in contrast to the averaged profiles from the replicate-based RNA-Seq data. Single measurements inherently retain more experimental noise, which might obscure the underlying smooth temporal progression of gene expression. Consequently, the high-dimensional manifold represented by these point estimates may be not that smooth, requiring the t-SNE mapping to be locally more sensitive to capture the slightly more disordered neighborhood relationships. Small perturbations in such noisy input features could then necessitate larger adjustments in the low-dimensional embedding to maintain fidelity to the input similarities, leading to higher observed Jacobian values.

To identify genes and timepoints exerting the most significant influence on the embedding, we summarized the Jacobian by summing the absolute values of derivatives for each input feature (gene g at timepoint t) across all $32 \times 2 = 64$ embedding coordinates: $\sum_{k=1}^{64} |(\boldsymbol{J}_{z^*})_{k,(g,t)}|$. Figure 9.11a visualizes these total sensitivity scores, reshaped into a 32×395 matrix. Several genes (annotated as genes 1, 149–152, 229–231, 308, 312, 369 in our filtered list) exhibited particularly high total sensitivity, primarily influenced by their expression changes around t_{36}. The expression profiles of many of these influential genes (Figure 9.11b) show relatively



(a) Summarized sensitivities.

(b) Profile plots of influential genes.

Figure 9.11: (a) Summarized sensitivities: for each input feature (gene at a specific timepoint), the sum of absolute values of its derivatives across all 64 output embedding coordinates is shown, reshaped into an 32×395 matrix (timepoints $\times$ genes). **(b)** Mean expression profiles (log-scale) of genes identified as most influential on the embedding.

constant expression in early timepoints, followed by a characteristic sharp upregulation around t_{36}, and then a gradual decrease. These genes are involved in diverse cellular functions, including the biosynthesis of teichulosonic acid (e.g., SCO4879-SCO4881), phosphate transport (e.g., SCO4140-SCO4142), and coelichelin biosynthesis (e.g., SCO0494), all plausible candidates for involvement in the metabolic reprogramming during phosphate depletion.

Using the computed Jacobian $\boldsymbol{J}_{z^*}$ and the input covariance $\boldsymbol{\Sigma}_y$, we propagated the uncertainty via Equation (9.9) to obtain $\boldsymbol{\Sigma}_z$. Figure 9.12 visualizes hypothetical outcomes from the resulting approximate Gaussian distribution $p(\boldsymbol{Z} \mid \boldsymbol{Y})$, illustrating the uncertainty in the embedding. The respective animation can be found at `https://github.com/Integrative-Transcriptomics/tsne`. While the synthetically introduced input uncertainties were modest, each embedded sample displays a degree of positional uncertainty. However, these variations primarily manifest as local displacements, and the overall global structure of the t-SNE embedding appears largely stable under this modeled noise.

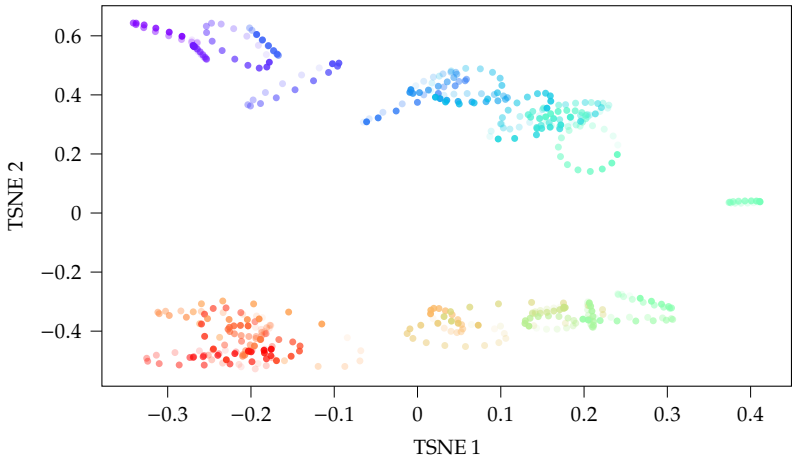


Figure 9.12: Visualization of uncertainty in the t-SNE embedding of *S. coelicolor* M145 data. Hypothetical outcomes of the embedding distribution $p(\boldsymbol{Z} \mid \boldsymbol{Y})$ illustrate the uncertainty for each timepoint's embedding. Timepoints are colored from purple (t_{20}) to red (t_{60}) according to Figure 9.10.

9.5 Runtime Evaluation

The computational cost of propagating uncertainty through t-SNE using our IFT-based approach was evaluated on simulated datasets. Experiments were performed on a CPU server equipped with an Intel Xeon CPU E5-2698 v3 @ 2.30 GHz (64 cores) and 378 GB of memory. We consider input data $\boldsymbol{Y} \sim \mathcal{N}(\boldsymbol{Y}^*, \boldsymbol{\Sigma}_y)$, where $\boldsymbol{Y}^* \in \mathbb{R}^{N \times D}$ is the mean data matrix (N samples, D features). The primary computational steps are: (1) obtaining the optimal embedding $\boldsymbol{z}^*$ using a standard t-SNE solver (cost not included in this evaluation, as it's pre-computation); (2) computing the Hessian $\boldsymbol{H}_{zz}$ and its pseudoinverse $\boldsymbol{H}_{zz}^{\dagger}$; (3) computing the mixed Jacobian terms (implicitly via JVPs/VJPs); (4) constructing the output covariance $\boldsymbol{\Sigma}_z$.

The Hessian $\boldsymbol{H}_{zz}$ is a $(2N) \times (2N)$ matrix (assuming a 2D embedding). Its pseudoinversion using SVD typically costs $\mathcal{O}((2N)^3) = O(N^3)$. The computation of each of the $2N$ columns of $\boldsymbol{\Sigma}_z$ involves JVPs and VJPs related to $\boldsymbol{J}_{yz}$ (mixed Jacobian of size $2N \times ND$) and multiplications by $\boldsymbol{\Sigma}_y$ (size $ND \times ND$). Empirically, as shown in Figure 9.13, the runtime for computing $\boldsymbol{\Sigma}_z$ (after $\boldsymbol{z}^*$ is found and $\boldsymbol{H}_{zz}^{\dagger}$ is computed) scales approximately as $\mathcal{O}(N^2 D)$ when N is moderately large. For fixed D, runtime scales roughly as $\mathcal{O}(N^2)$ (Figure 9.13b), and for fixed N, as $\mathcal{O}(D)$ (Figure 9.13a). The memory usage shows similar trends. The $\mathcal{O}(N^3)$ cost of `jnp.linalg.pinv(H_zz)` is a one-time cost for a given $\boldsymbol{z}^*$. If N is very large, this N^3 term for pseudoinversion will dominate. The $N^2 D$ scaling observed likely reflects the cost of the 'vmap'ped operations for constructing $\boldsymbol{\Sigma}_z$ when D is substantial or when the N^3 term is not yet the bottleneck.

This analysis indicates that while propagating uncertainty adds computational overhead compared to standard t-SNE, the IFT-based approach is feasible for datasets of moderate size commonly encountered in bioinformatics, especially when the number of samples N is not excessively large due to the N^3 pseudoinversion step and N^2 scaling in subsequent parts. For very large N, approximations for the pseudoinverse-vector products or focusing on diagonal elements of $\boldsymbol{\Sigma}_z$ might be necessary.

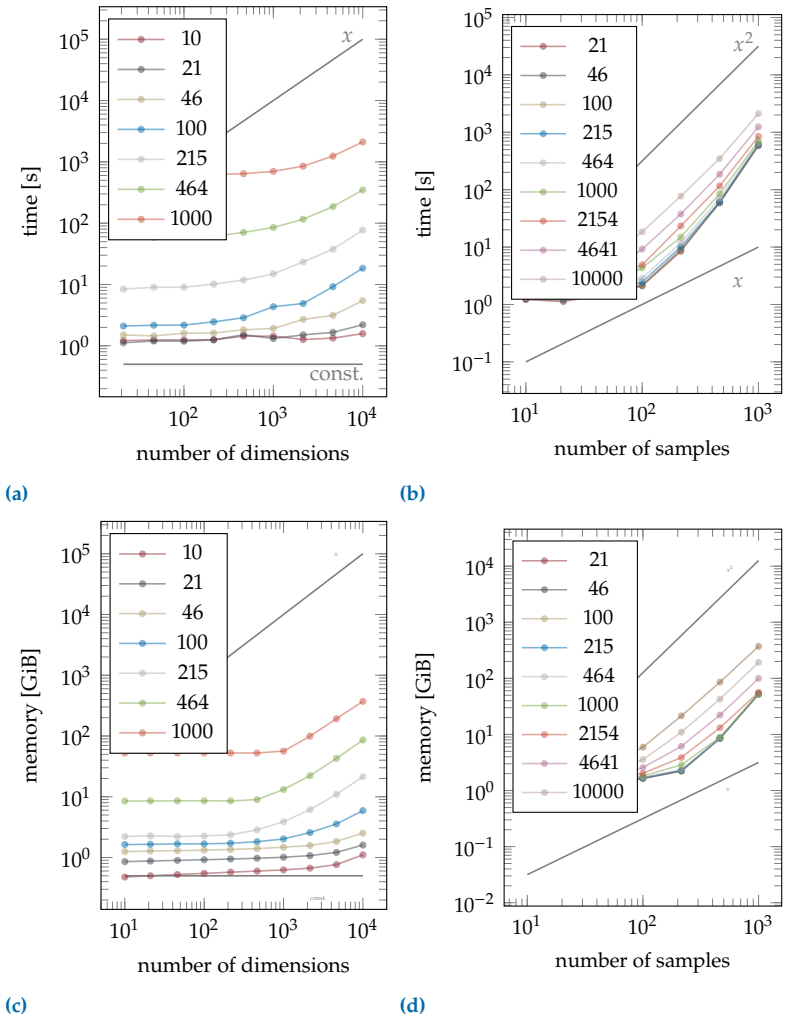


Figure 9.13: Runtime and memory requirements for propagating uncertainty through t-SNE, excluding the initial computation of z^*. Grey lines indicate reference complexities. **(a)** Runtime and and **(c)** memory usage as the number of input dimensions D increases, for different numbers of samples N (colors). **(b)** Runtime and **(d)** memory usage as the number of samples N increases, for different input dimensions D (colors).

9.6 Discussion & Conclusion

In this chapter, we have introduced a novel framework for propagating and visualizing input uncertainty through the t-Distributed Stochastic Neighbor Embedding (t-SNE) algorithm [161]. Our work provides a significant extension to this widely adopted visualization technique. By modeling input data points with associated uncertainty and leveraging the Implicit Function Theorem (IFT) coupled with automatic differentiation, we have developed a principled method to estimate and visualize the uncertainty in the resulting low-dimensional t-SNE embeddings.

Our primary contribution is the development and practical implementation of an uncertainty-aware t-SNE method. The showcases presented in Section 9.4 demonstrated the practical utility of this approach. We have shown that visualizing uncertainty in t-SNE plots, for instance via hypothetical outcome plots, provides an enhanced understanding of the embedding. This allows users to distinguish robustly placed points and stable cluster structures from those whose positions or separations might be heavily influenced by input noise. Furthermore, our framework enables a quantitative assessment of the stability of individual embedded points and the relationships between them, revealing how input uncertainty translates to positional variance in the low-dimensional

map. By comparing our uncertainty-aware embeddings with those from standard deterministic t-SNE, we highlighted instances where structures appearing clear in the latter are revealed to be less certain or even ambiguous once input variability is rigorously propagated and visualized. The runtime evaluation (Section 9.5) indicated that while propagating uncertainty introduces computational overhead, particularly due to the Hessian pseudoinversion and subsequent covariance propagation, the method remains feasible for many bioinformatics datasets of practical interest.

Despite its advantages, our proposed methodology has certain limitations. The most significant is the computational expense, especially for datasets with a very large number of samples (N), which can make the Hessian pseudoinversion and full covariance matrix construction demanding. Our current framework also relies on Gaussian assumptions for both the input uncertainty and the propagated uncertainty in the embedding, which may not hold for all biological data or types of uncertainty; highly non-linear mappings in t-SNE can further challenge the accuracy of such first-order approximations. It is also important to note that the IFT provides sensitivities locally around a specific optimal embedding $\boldsymbol{z}^*$ obtained for a mean input $\boldsymbol{y}^*$. Since t-SNE's cost function is non-convex, different initializations can yield different local optima; our method quantifies uncertainty for one such given optimum, not the uncertainty arising from the choice between multiple optima. The validity of the IFT also depends on $\boldsymbol{z}^*$ being a true stationary point, meaning poor convergence of the t-SNE optimizer could weaken the theoretical basis. Finally, while we used hypothetical outcome plots, effectively visualizing multivariate uncertainty remains a general challenge, particularly for many data points.

In future research, scalability could be enhanced by exploring iterative methods for pseudoinverse-Hessian-vector products, or by developing methods to approximate the output covariance matrix $\boldsymbol{\Sigma}_z$, perhaps by only computing its diagonal or a sparse representation. Extending the framework to handle non-Gaussian input uncertainties, would broaden its applicability. Addressing the global variability of t-SNE embeddings by combining input uncertainty propagation with analyses across multiple optimization runs could provide a more comprehensive picture of embedding stability. The development of interactive visualization tools, allowing users to dynamically explore uncertainty and query points for sensitivity information, would greatly improve usability. Lastly, investigating how uncertainty-aware t-SNE embeddings can inform downstream tasks is another important direction.

In conclusion, the propagation of input uncertainty through t-SNE, as presented in this chapter, represents a critical advancement towards more robust and reliable interpretation of visualizations derived from high-dimensional biological data. By making the uncertainty in the low-dimensional embeddings explicit, our methodology empowers bioinformaticians and biologists to move beyond point estimates, to better assess the confidence in observed patterns, and to acknowledge the inherent ambiguities present in their data. This work, alongside the uncertainty-aware PCA developed in VIPurPCA (Chapter 8), underscores the importance of uncertainty quantification in dimensionality reduction for modern bioinformatics.

A Probabilistic Approach to Visualize the Effect of Missing Data on PCA in Ancient Human Genomics

10.

Disclaimer 10.1 Chapter 10 is based on the peer-reviewed publication with the following co-author contributions:

S. Zabel, S. Breitling, C. Posth, and K. Nieselt. "A Probabilistic Approach to Visualize the Effect of Missing Data on PCA in Ancient Human Genomics". *BMC genomics* 26.1 (2025) [290]

	Ideas	Analysis & Interpretation	Code	Writing
S. Zabel	25 %	60 %	50 %	50 %
S. Breitling	25 %	40 %	50 %	35 %
C. Posth	25 %	0 %	0 %	5 %
K. Nieselt	25 %	0 %	0 %	10 %

Abstract

Background: Principal Component Analysis (PCA) is widely used in population genetics to visualize genetic relationships and population structures. In ancient genomics, genotype information may in parts remain unresolved due to the low abundance and degraded quality of ancient DNA. While methods like SmartPCA allow the projection of ancient samples despite missing data, they do not quantify projection uncertainty. The reliability of PCA projections for often very sparse ancient genotype samples is not well understood. Ignoring this uncertainty may lead to overconfident conclusions about the observed genetic relationships and population structure.

Results: This study systematically investigates the impact of missing loci on PCA projections using both simulated and real ancient human genotype data. Through extensive simulations with high-coverage ancient samples, we demonstrate that increasing levels of missing data can lead to less accurate SmartPCA projections, highlighting the importance of considering uncertainty when interpreting PCA results from ancient samples. To address this, we developed a probabilistic framework to quantify the uncertainty in PCA projections due to missing data. By applying our methodology to modern and ancient West Eurasian genotype samples from the Allen Ancient DNA Resource database, we could show a high concordance between our predicted projection and empirically derived distributions. Applying this framework to real-world data, we demonstrate its utility in predicting and visualizing embedding uncertainties for ancient samples of varying SNP coverages.

Conclusion: Our results emphasize the importance of accounting for projection uncertainty in ancient population studies. We therefore make our probabilistic model available through TrustPCA, a user-friendly web tool that provides researchers with uncertainty estimates alongside PCA projections, facilitating data exploration in ancient human genomic studies and enhancing transparency in data quality reporting.

Code and webtool are available at the Github repository `Integrative-Transcriptomics/trustpca`, and at `https://trustpca-tuevis.cs.uni-tuebingen.de/`

10.1 Background

Human curiosity about our origins has driven research into fundamental questions about where we come from and how we are related to each other. Population genetics, the study of genetic composition and variation within and between populations, aims to understand how genetic diversity evolves over time. The analysis of ancient DNA has amplified this effort, offering insights into past genetic diversity and human evolution. A central challenge in this field is to interpret the ancestral relationships between individuals and populations sampled, especially when working with high-dimensional genotype data. To address this challenge, dimensionality reduction techniques and the resulting visualizations have become a central part of analyses in population genetics [39].

Among these techniques, principal component analysis (PCA) [95, 209] stands out as the most widely used method for reducing the dimensionality of genotype data. PCA works by projecting samples onto a subspace defined by principal components (PCs), which capture the directions of maximum variance in the data. The coordinates of the samples in the reduced space, computed as linear combinations of their original allelic or genotypic values, offer clear genetic interpretations [175, 216]. Typically, only the first two or three PCs are visualized in scatter plots, as they capture most of the variance and effectively reveal patterns of population structure, ancestry, and relatedness among individuals [175, 216].

A significant challenge in analyzing ancient genotype data is the low abundance and degraded quality of ancient DNA [92], which results in sparse data that make direct application of PCA impractical [76]. To overcome this, PCA is often performed on complete modern datasets to define the principal axes, onto which ancient samples are projected using algorithms capable of handling missing data [187]. The most popular PCA software in population genetics that implements this approach is SmartPCA [208], which is part of the EIGENSOFT software suite [208, 223].

However, the reliability, robustness, and replicability of PCA in population genetic studies have been called into question in recent years due to

- the potential for unreliable and misleading projections that do not accurately reflect genetic distances [58],
- the inability to infer admixture levels or directions from PCA plots [58, 281],
- the heavy dependence of results on the choice of reference populations [58, 175, 266, 267],

For studies involving ancient samples, these issues are exacerbated, as the interpretation of admixture and the origins of populations often hinges on the selection of modern reference datasets [58]. Despite these challenges, PCA remains the most widely used method for dimensionality reduction in population genetics.

Modern and ancient genomes are often genotyped at ~600,000 autosomal single nucleotide polymorphisms (SNPs) using the Human Origins array [207]. However, for ancient samples, the proportion of observed positions

can be lower than 1 % of the array [167]. The reliability of PCA projections for such highly sparse samples is not well understood, leaving their placement on PCA plots to the experience of researchers or subjective judgment.

In this work, we systematically analyze how varying levels of missing SNPs influence the reliability of SmartPCA projections through simulations. These simulations aim to raise awareness of how far SmartPCA estimates can deviate from the true embedding. Building on this, we introduce a probabilistic model to quantify embedding uncertainty, offering users of SmartPCA insights into the reliability of projections based on the observed genotypes of ancient individuals. While the embedding generated by SmartPCA represents the best estimate given the limited number of observed SNPs [187], our model provides a probability distribution around this estimate. This distribution indicates the likelihood of the sample being projected to a different place on the map if all SNPs were known, offering a more comprehensive understanding of the uncertainty in PCA results. Since SmartPCA differs methodologically from conventional PCA, existing uncertainty-aware PCA methods (e.g., [80, 291]) cannot be directly applied in this setting. Our approach uniquely addresses this gap by specifically modeling projection uncertainty in SmartPCA. To facilitate accessibility, we developed TrustPCA, a user-friendly tool that implements our probabilistic model, allowing researchers to evaluate and visualize projection uncertainty in their analyses.

10.2 Material & Methods

10.2.1 Data, code, and software availability

The data that support the findings of this study are openly available from the Allen Ancient DNA Resource (AADR), version v54.1.p1: `https://doi.org/10.7910/DVN/FFIDCW)` [167]. The source code for the proposed methodology and for the analyses described in this paper are available at `https://github.com/Integrative-Transcriptomics/TrustPCA/tree/theory_and_paper_code`. TrustPCA is openly available at `https://trustpca-tuevis.cs.uni-tuebingen.de/`, with the source code accessible at `https://github.com/Integrative-Transcriptomics/TrustPCA/tree/main`.

10.2.2 Data

In this study, we utilized genotype data from the *Allen Ancient DNA Resource* (AADR) [167], Version v54.1.p1, released on March 6, 2023. Specifically, we used the *1240K+HO* dataset, which includes genotype data for 20,503 unique human individuals (9,990 ancient and 10,513 present-day) based on the Human Origins array [207], covering 597,573 sites. The genotype data is provided in EIGENSTRAT format [208, 223], consisting of a tabular genotype file where columns represent individuals and rows represent genotype sites. The table entries can have one of four characters (0, 1, 2, or 9), representing zero (0), one (1), or two (2) copies of the reference allele. Missing genotypes are encoded as 9 [208, 223].

From the AADR database, all individuals originating from 67 commonly used modern West Eurasian populations (see Additional file 1 & 2, [140]) were selected, totaling 1,433 individuals. These individuals were used to infer genetic variation. Additionally, all ancient West Eurasian individuals from the Mesolithic or later periods were included in the analysis, resulting in a dataset of 6,627 ancient individuals (see Additional file 4).

It is important to note that genotype calls for all ancient individuals are pseudo-haploid, whereas those for modern individuals are diploid. Additionally, SNP coverage—defined as the proportion of successfully genotyped single nucleotide polymorphisms (SNPs) per individual relative to the total of 540,247 variant SNPs analyzed in this study (see Section 10.3.1)—is uniformly high for modern individuals (100 %), but varies widely among ancient individuals, ranging from 1 % to 100 %. The full distribution of SNP coverage across all ancient individuals is shown in Figure D.1.

10.2.3 SmartPCA: Applying PCA to Genotype Data

Several software tools, including EIGENSOFT [208, 223] and PLINK [226], have been developed to implement PCA (Section 2.1) specifically for genetic data. Among these, we focus on SmartPCA [208], a module within the EIGENSOFT suite that efficiently performs PCA on genotype data and has become the standard tool in population genetic studies.

In this context, the features of the data matrix $\boldsymbol{M}$ represent genotype information across various loci. Prior to PCA application, the data is centered and optionally scaled. In this work, data was scaled to account for the genetic drift, reflecting the rate of allele frequency change at each locus [208]:

$$\boldsymbol{X}_{ij} = \frac{\boldsymbol{M}_{ij} - \boldsymbol{\mu}_j}{\sqrt{\boldsymbol{p}_j(1-\boldsymbol{p}_j)}} \quad \text{with} \quad \boldsymbol{p}_j = \frac{\boldsymbol{\mu}_j}{2}, \tag{10.1}$$

where $\boldsymbol{\mu}_j$ is the mean of feature j. Please note that the "shrinkmode: YES" option of the SmartPCA program was not selected.

Following this normalization, PCA is performed on the centered and scaled data matrix $\boldsymbol{M}$ to compute principal components and to project the samples into a low-dimensional space.

This PCA methodology is predominantly applied to high-quality genotype data obtained from modern populations, where complete genotypic information is generally available across all loci of interest. For ancient samples, which often lack full genotype resolution, direct PCA application is challenging. Instead, a common approach is to project ancient samples onto the top principal components (typically the first two) derived from modern populations, thus enabling comparative analyses without requiring complete genotypic data. This method, known as *projection to the model plane* [187], is implemented in SmartPCA and is described below.

Given the eigenvector matrix $\boldsymbol{V} \in \mathbb{R}^{D\times Q}$ derived from a modern population, an ancient genotype sample $\boldsymbol{z} \in \mathbb{R}^D$ can be divided into observed genotypes $\boldsymbol{z}^* \in \mathbb{R}^M$, where $0 > M \leq D$, and unobserved genotypes

$z^{\#} \in \mathbb{R}^{D-M}$. To project z onto the principal component space defined by V, the following procedure is applied: First, z^* is centered and scaled using the mean and genetic drift calculated from the modern population (Equation (10.1)). With the centered and scaled z^*, the projection scores $\hat{\tau} \in \mathbb{R}^P$ are computed, representing the coordinates of the sample in the principal component space. Since z contains missing data, $\hat{\tau}$ is obtained as a least-square estimate [187]:

$$\hat{\tau} = (V^{*\intercal} V^*)^{-1} V^{*\intercal} z^*, \tag{10.2}$$

where $V^* \in \mathbb{R}^{M \times Q}$ is the subset of rows of V corresponding to the observed loci in z (thus z^*). The result, $\hat{\tau}$, provides an approximation of the true embedding τ that would have been obtained if all genotypes were observed. The factor $(V^{*\intercal} V^*)^{-1}$ minimizes the sum of the quadratic differences between the observed values and the predicted model values. This ensures an effective regression of the observed parts of the sample onto the PCA model plane and takes the entire structure into account.

10.2.4 Evaluating the Impact of Missing Data

To assess the effect of missing genotype information in ancient samples on the projection using the SmartPCA approach as described above, a comprehensive simulation experiment was performed. Fifteen high-coverage ancient samples $z^{(i)}$, $i = 1, \ldots, 15$, each with less than 10 % missing genotypes, were selected. These samples were harmonized—aligned to a common set of SNPs (union)—to ensure consistency in the proportion of missing loci. Random omissions were then introduced to simulate varying levels of missing data across the samples. Each ancient sample was projected onto a two-dimensional subspace derived from modern West Eurasian samples using SmartPCA, yielding embeddings $\tau^{(i)}$, which were considered accurate representations of the samples' true positions in the subspace (Figure 10.1a). To simulate the impact of missing data, genotypes were randomly omitted from each ancient sample at a specified rate r. This process was repeated n times, generating multiple realizations $\{z^{(i,j)}\}_{j=1,\ldots,n}$ of each sample with varying patterns of missing data (Figure 10.1b). The modified samples $z^{(i,j)}$ were then projected using SmartPCA, resulting in a set of embeddings, $\{\hat{\tau}^{(i,j)}\}_{j=1,\ldots,n}$. Each embedding $\hat{\tau}^{(i,j)}$ serves as an approximation of the original embedding $\tau^{(i)}$. To quantify the effect of missing data on the projection, statistics about the difference $\hat{\tau}^{(i,j)} - \tau^{(i)}$ were computed and analyzed for each principal component of the projection.

10.2.5 Modeling the Deviation Between Estimated and True Embeddings in PCA

This section introduces a method to estimate a probability distribution over the true position—if all genotypes were observed—of a sample with unobserved genotypes in the principal component (PC) space. This distribution quantifies the potential deviation of the estimated embedding obtained through SmartPCA from the true embedding, capturing the uncertainty introduced by missing genotypes. The degree of uncertainty

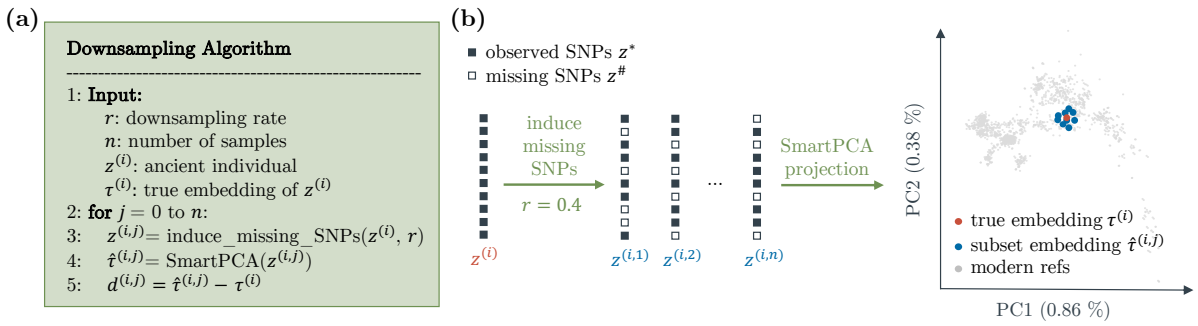


Figure 10.1: Simulation of missing genotypes. (a) Algorithm to simulate missing genotypes in ancient individuals. Starting with a high-coverage sample $z^{(i)}$ and its SmartPCA embedding $\tau^{(i)}$, considered the true embedding of $z^{(i)}$, the algorithm generates n realizations $z^{(i,j)}$ by randomly omitting genotypes at a specified rate r. Each realization $z^{(i,j)}$ is projected onto a predefined PCA subspace, yielding n estimators $\hat{\tau}^{(i,j)}$ for $\tau^{(i)}$. The distribution of $\hat{\tau}^{(i,j)}$ around $\tau^{(i)}$ is analyzed both visually and statistically. (b) Schematic representation of the downsampling algorithm, illustrating the process of random genotype omission and SmartPCA projection.

depends on the amount of missing data and the strength of the signals provided by the observed loci.

The original publication of the '*projection to the model plane*' method [187] provides an error analysis for the PC score estimation arising from missing data, serving as the foundation for the proposed methodology to estimate embedding uncertainty.

In the following, let $\mathcal{P} = \{1, \ldots, P\}$ be the set of indices of the principal components of interest, and let $\mathcal{Q} = \{P+1, \ldots, Q\}$ be the set of indices of the remaining components. For instance, $\boldsymbol{V}_\mathcal{P} \in \mathbb{R}^{D \times P}$ is a matrix containing the first P eigenvectors, and $\boldsymbol{\tau}_\mathcal{P} \in \mathbb{R}^P$ represents the principal component scores of a sample in this subspace. The discrepancy between the true embedding $\boldsymbol{\tau}_\mathcal{P}$ and its estimate by SmartPCA $\hat{\boldsymbol{\tau}}_\mathcal{P}$ is given by [187]

$$\begin{aligned}
\boldsymbol{\tau}_\mathcal{P} - \hat{\boldsymbol{\tau}}_\mathcal{P} &= \boldsymbol{\tau}_\mathcal{P} - (\boldsymbol{V}_\mathcal{P}^{*\intercal}\boldsymbol{V}_\mathcal{P}^*)^{-1}\boldsymbol{V}_\mathcal{P}^{*\intercal}\boldsymbol{z}^* && (10.3)\\
&= \boldsymbol{\tau}_\mathcal{P} - (\boldsymbol{V}_\mathcal{P}^{*\intercal}\boldsymbol{V}_\mathcal{P}^*)^{-1}\boldsymbol{V}_\mathcal{P}^{*\intercal}\boldsymbol{V}^*\boldsymbol{\tau} && (10.4)\\
&= \boldsymbol{\tau}_\mathcal{P} - [(\boldsymbol{V}_\mathcal{P}^{*\intercal}\boldsymbol{V}_\mathcal{P}^*)^{-1}\boldsymbol{V}_\mathcal{P}^{*\intercal}(\boldsymbol{V}_\mathcal{P}^*\boldsymbol{\tau}_\mathcal{P} + \boldsymbol{V}_\mathcal{Q}^*\boldsymbol{\tau}_\mathcal{Q})] && (10.5)\\
&= \boldsymbol{\tau}_\mathcal{P} - (\boldsymbol{V}_\mathcal{P}^{*\intercal}\boldsymbol{V}_\mathcal{P}^*)^{-1}\boldsymbol{V}_\mathcal{P}^{*\intercal}\boldsymbol{V}_\mathcal{P}^*\boldsymbol{\tau}_\mathcal{P} - (\boldsymbol{V}_\mathcal{P}^{*\intercal}\boldsymbol{V}_\mathcal{P}^*)^{-1}\boldsymbol{V}_\mathcal{P}^{*\intercal}\boldsymbol{V}_\mathcal{Q}^*\boldsymbol{\tau}_\mathcal{Q} && (10.6)\\
&= -(\boldsymbol{V}_\mathcal{P}^{*\intercal}\boldsymbol{V}_\mathcal{P}^*)^{-1}\boldsymbol{V}_\mathcal{P}^{*\intercal}\boldsymbol{V}_\mathcal{Q}^*\boldsymbol{\tau}_\mathcal{Q}. && (10.7)
\end{aligned}$$

Note that $\boldsymbol{z}^* = \boldsymbol{V}^*\boldsymbol{\tau}$ follows directly from rearranging $\boldsymbol{\tau} = \boldsymbol{V}^{*\intercal}\boldsymbol{z}$ (Equation (2.10)), leveraging the fact that $\boldsymbol{V}^*$ is an orthogonal matrix (Equation (10.3) to Equation (10.4)). Additionally, observe that $(\boldsymbol{V}_\mathcal{P}^{*\intercal}\boldsymbol{V}_\mathcal{P}^*)^{-1}\boldsymbol{V}_\mathcal{P}^{*\intercal}\boldsymbol{V}_\mathcal{P}^*$ reduces to the identity matrix since $\boldsymbol{V}_\mathcal{P}^{*\intercal}\boldsymbol{V}_\mathcal{P}^*$ is an invertible square matrix (Equation (10.6) to Equation (10.7)).

The derived expression for the discrepancy, $\boldsymbol{\tau}_\mathcal{P} - \hat{\boldsymbol{\tau}}_\mathcal{P}$, shows a linear dependence on $\boldsymbol{\tau}_\mathcal{Q}$. Therefore, we can use the simple rules of moment propagation that can be applied in linear settings to obtain an analytical exact solution [264]. The expectation of the PC scores $\mathbb{E}[\boldsymbol{\tau}_\mathcal{Q}]$ is zero, i.e., $\mathbb{E}[\boldsymbol{\tau}_\mathcal{Q}] = \boldsymbol{0}$, as a consequence of centering the data feature-wise prior to performing PCA. Propagating this expectation through the linear

function yields the expectation of the discrepancy:

$$\mathbb{E}[\boldsymbol{\tau}_{\mathcal{P}} - \hat{\boldsymbol{\tau}}_{\mathcal{P}}] = -(\boldsymbol{V}_{\mathcal{P}}^{*\intercal}\boldsymbol{V}_{\mathcal{P}}^{*})^{-1}\boldsymbol{V}_{\mathcal{P}}^{*\intercal}\boldsymbol{V}_{\mathcal{Q}}^{*}\mathbb{E}[\boldsymbol{\tau}_{\mathcal{Q}}] \tag{10.8}$$

$$= -(\boldsymbol{V}_{\mathcal{P}}^{*\intercal}\boldsymbol{V}_{\mathcal{P}}^{*})^{-1}\boldsymbol{V}_{\mathcal{P}}^{*\intercal}\boldsymbol{V}_{\mathcal{Q}}^{*}\mathbf{0} \tag{10.9}$$

$$= \mathbf{0}. \tag{10.10}$$

The variances of the PC scores, $\mathbb{V}[\boldsymbol{\tau}_{\mathcal{Q}}] \in \mathbb{R}^{Q\times Q}$, are determined by the eigenvalues corresponding to the components, which are equal to the variances of the principal components. Because the PCs are uncorrelated, all covariances between components are zero. Therefore, the covariance matrix $\mathbb{V}[\boldsymbol{\tau}_{\mathcal{Q}}]$ is diagonal and can be expressed as $\boldsymbol{\Lambda}_{\mathcal{Q}}$, where $\boldsymbol{\Lambda}_{\mathcal{Q}}$ contains the eigenvalues of the Q corresponding components. Propagating the variance through the linear function yields the variance of the discrepancy:

$$\mathbb{V}[\boldsymbol{\tau}_{\mathcal{P}} - \hat{\boldsymbol{\tau}}_{\mathcal{P}}] = -(\boldsymbol{V}_{\mathcal{P}}^{*\intercal}\boldsymbol{V}_{\mathcal{P}}^{*})^{-1}\boldsymbol{V}_{\mathcal{P}}^{*\intercal}\boldsymbol{V}_{\mathcal{Q}}^{*}\mathbb{V}[\boldsymbol{\tau}_{\mathcal{Q}}](-(\boldsymbol{V}_{\mathcal{P}}^{*\intercal}\boldsymbol{V}_{\mathcal{P}}^{*})^{-1}\boldsymbol{V}_{\mathcal{P}}^{*\intercal}\boldsymbol{V}_{\mathcal{Q}}^{*})^{\intercal} \tag{10.11}$$

$$= -(\boldsymbol{V}_{\mathcal{P}}^{*\intercal}\boldsymbol{V}_{\mathcal{P}}^{*})^{-1}\boldsymbol{V}_{\mathcal{P}}^{*\intercal}\boldsymbol{V}_{\mathcal{Q}}^{*}\boldsymbol{\Lambda}_{\mathcal{Q}}(-(\boldsymbol{V}_{\mathcal{P}}^{*\intercal}\boldsymbol{V}_{\mathcal{P}}^{*})^{-1}\boldsymbol{V}_{\mathcal{P}}^{*\intercal}\boldsymbol{V}_{\mathcal{Q}}^{*})^{\intercal}. \tag{10.12}$$

Assuming the discrepancy is Gaussian distributed, it is fully specified by its expectation and variance. From the expectation and the variance of the discrepancy $\boldsymbol{\tau}_{\mathcal{P}} - \hat{\boldsymbol{\tau}}_{\mathcal{P}}$, we can derive those of the true embedding $\boldsymbol{\tau}_{\mathcal{P}}$. Since the expectation of the discrepancy, $\mathbb{E}[\boldsymbol{\tau}_{\mathcal{P}} - \hat{\boldsymbol{\tau}}_{\mathcal{P}}]$, is zero, the expectation for the true embedding $\boldsymbol{\tau}_{\mathcal{P}}$ can be expressed as:

$$\mathbb{E}[\boldsymbol{\tau}_{\mathcal{P}} - \hat{\boldsymbol{\tau}}_{\mathcal{P}}] = \mathbf{0} \tag{10.13}$$

$$\mathbb{E}[\boldsymbol{\tau}_{\mathcal{P}}] - \mathbb{E}[\hat{\boldsymbol{\tau}}_{\mathcal{P}}] = \mathbf{0} \tag{10.14}$$

$$\mathbb{E}[\boldsymbol{\tau}_{\mathcal{P}}] - \hat{\boldsymbol{\tau}}_{\mathcal{P}} = \mathbf{0} \tag{10.15}$$

$$\mathbb{E}[\boldsymbol{\tau}_{\mathcal{P}}] = \hat{\boldsymbol{\tau}}_{\mathcal{P}} \tag{10.16}$$

Since $\hat{\boldsymbol{\tau}}_{\mathcal{P}}$ is a known (observed) point estimate, we have $\mathbb{E}[\hat{\boldsymbol{\tau}}_{\mathcal{P}}] = \hat{\boldsymbol{\tau}}_{\mathcal{P}}$ (Equation (10.14) to Equation (10.15)).

Furthermore, since $\hat{\boldsymbol{\tau}}_{\mathcal{P}}$ is a point estimate of zero variance, the variance of the discrepancy equals the variance of the true embedding:

$$\mathbb{V}[\boldsymbol{\tau}_{\mathcal{P}} - \hat{\boldsymbol{\tau}}_{\mathcal{P}}] = \mathbb{V}[\boldsymbol{\tau}_{\mathcal{P}}] \tag{10.17}$$

Consequently, the distribution of the true embedding $\boldsymbol{\tau}_{\mathcal{P}}$ can be expressed as:

$$\boldsymbol{\tau}_{\mathcal{P}} \sim \mathcal{N}(\hat{\boldsymbol{\tau}}_{\mathcal{P}}, \boldsymbol{\Sigma}_{\mathcal{P}}), \tag{10.18}$$

where $\boldsymbol{\Sigma}_{\mathcal{P}} = \mathbb{V}[\boldsymbol{\tau}_{\mathcal{P}} - \hat{\boldsymbol{\tau}}_{\mathcal{P}}]$ is the covariance matrix of the discrepancy.

The assumption of a Gaussian distribution for the discrepancy between estimated and true embeddings in PCA is justified by the Central Limit Theorem (CLT). In this context, the discrepancy arises as a linear combination of contributions from individual loci, with each locus contributing independently (or nearly independently) to the total variance in the embedding space. Specifically, the variance propagation formula in Equation (10.12) aggregates these independent contributions through the eigenvector projections, resulting in a summation of random variables corresponding to the variance at each locus. According to the CLT, the sum (or weighted sum) of a sufficiently large number of independent

random variables with finite variance converges to a Gaussian distribution, irrespective of the original distributions of the variables. Given that PCA typically operates on genotype datasets with thousands or more loci, the high dimensionality ensures that the conditions for the CLT are met, leading to a reliable approximation of the discrepancy as Gaussian distributed.

A key prerequisite for accurately estimating the distribution of the true embedding using the described approach, along with the provided estimates for $\mathbb{E}$ and $\mathbb{V}$, is that the sample of interest, z, originates from a data distribution similar to the one used to compute the eigenvectors and eigenvalues. If this condition is not met, alternative methods for estimating $\mathbb{V}$ may be more appropriate.

10.3 Results

10.3.1 Computation of the PC space

The principal component (PC) space was derived from commonly used modern West Eurasian populations [140] (see Additional file 1 & 2) obtained from the Allen Ancient DNA Resource (AADR) [167]. This PC space is used for all SmartPCA projections in this work. SmartPCA reported that 540,247 variant SNPs were included in the computation of the PC space, while 57,326 invariant loci were excluded. As a result, only the variant loci were retained and used consistently throughout all analyses.

10.3.2 Impact of Missing Loci on Ancient Sample Projections

To evaluate the effect of missing loci on the SmartPCA projections of ancient genotype individuals, a simulation experiment was performed. Fifteen high-coverage West Eurasian ancient individuals, each with at least 90 % of known genotypes, were selected from the AADR database (see Additional file 3). The selection criteria also ensured that the projections of these embeddings were distributed across the two-dimensional PC space. This distribution allowed us to investigate whether certain regions of the PC space are more sensitive to missing data than others.

For each ancient individual, loci were randomly omitted at varying rates to simulate different levels of missing data. This process was repeated 20,000 times for each rate, generating a set of downsampled variants for each ancient individual. Both the downsampled and the original high-coverage individuals were then projected onto the first two dimensions of the PC space using SmartPCA [223]. These projections enabled a comparative analysis to quantify the effect of missing loci on the embedding positions. We will refer to the projections of the high-coverage (original) samples as reference projections and the projections of the downsampled variants as subset projections.

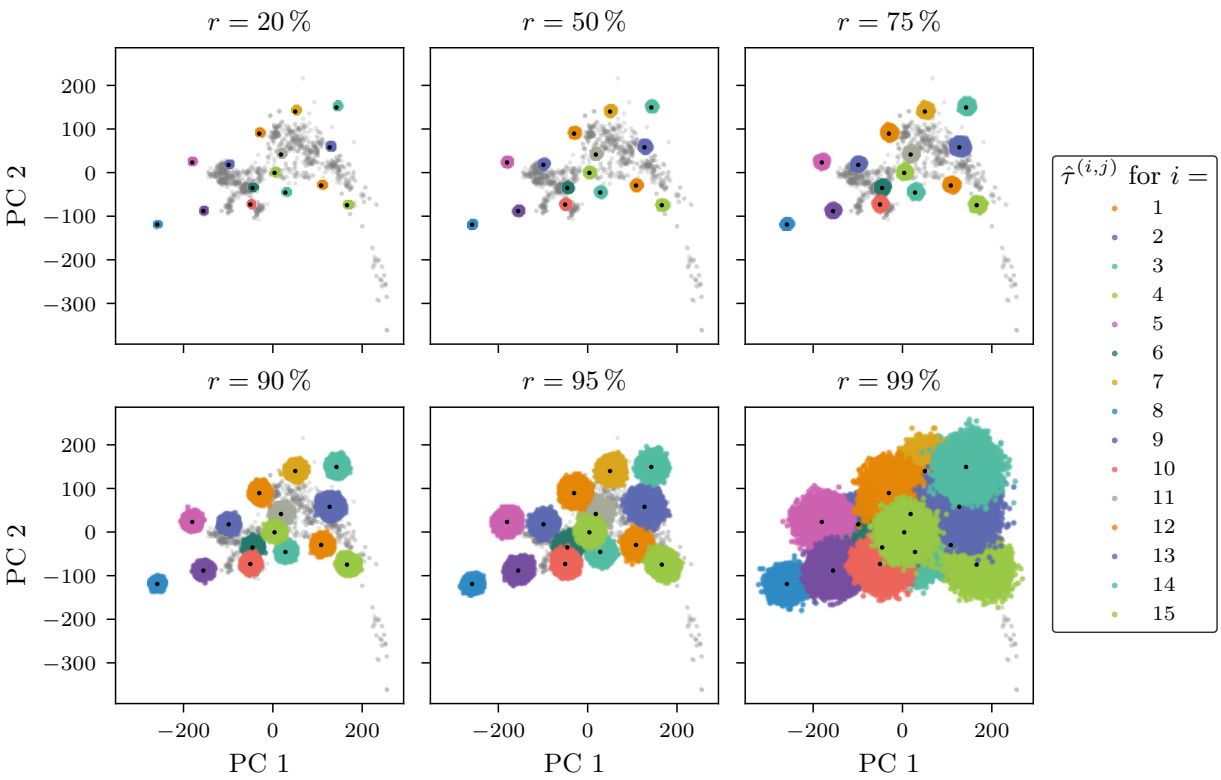


Figure 10.2: Impact of missing loci on SmartPCA projections of ancient genotype samples. The projections of the modern West Eurasian samples, used to compute the principal component (PC) subspace, are shown in gray. The black points represent the projections of 15 high-coverage ancient samples, denoted as the *true* projections $\{\boldsymbol{\tau}^{(i)}\}_{i=1,\dots,15}$. To evaluate the impact of missing loci, loci were randomly omitted from the ancient samples at an increasing rate r. This process was repeated 20,000 times for each sample i. The resulting SmartPCA projections, $\{\hat{\boldsymbol{\tau}}^{(i,j)}\}_{j=1,\dots,20000}$, are shown in their respective colors for each sample i.

Figure 10.2 shows that as the proportion of missing genotypes increases, the spread of the subset projections around their corresponding reference projections becomes larger. Moreover, the subset projections are isotropically distributed around the reference projections, indicating that the uncertainty introduced by missing loci does not exhibit directional bias in the PC space. While slight differences in the extent of the spread are observed at different rates of missing data, the location of the reference projections within the PC space does not appear to systematically influence the degree of spread.

To quantify the impact of missing loci on SmartPCA projections, we examined the discrepancies in PC1 and PC2 for the downsampled ancient samples across varying downsampling rates (r). Figure D.2 shows violin plots of these discrepancies, revealing a clear trend: as r increases, the distributions of discrepancies in both PC1 and PC2 broaden significantly, indicating that the spread of subset projections around their respective reference projections grows with the proportion of missing loci. The median of the discrepancies remains consistently at zero for all r, reflecting the isotropic nature of the projection spread and the absence of systematic directional bias in the PC space. However, the spread, as captured by the width of the violin plots, increases steadily with growing r.

10.3.3 Uncertainty Prediction of SmartPCA Embeddings

The simulation results presented in Section 10.3.2 demonstrate that missing loci can lead to SmartPCA subset projections that deviate substantially from their corresponding reference projections, as illustrated in Figure 10.2. The principal component (PC) scores are derived as a linear combination of eigenvectors and genotypes. Consequently, the influence of missing loci on a projection is determined by two factors: the corresponding eigenvector entries and the (unknown) genotypes at those loci.

In real-world scenarios, the genotypes at missing loci are unavailable, making it impossible to directly compute their contributions to the projection. To address this, we developed a probabilistic model that predicts a distribution for the true embedding location $\boldsymbol{\tau}$, given the SmartPCA embedding $\hat{\boldsymbol{\tau}}$ (for details, see Section 10.2.5). In this model, the error analysis by Nelson *et al.* [187] is extended by incorporating assumptions on the variance structure of the PC scores. We applied variance propagation to predict distributions for true projection locations. These predictions were evaluated against empirical distributions derived from simulations with varying levels of downsampled loci, enabling a direct comparison of predicted and observed effects of missing data.

Uncertainty Predictions for Modern Genotype Samples

In this experiment, we simulated missing data by selecting a single set of loci to be omitted for each downsampling rate r. These loci were removed from all 1,433 full-coverage modern West Eurasian individuals, which were subsequently projected using SmartPCA. The differences between the resulting embedding estimates, $\hat{\boldsymbol{\tau}}$, and the reference embeddings, $\boldsymbol{\tau}$, were computed for the first two principal components and plotted as scatter points (Figure 10.3, blue points). From this distribution of discrepancies, we calculated the mean and covariance, which were used to define an empirical Gaussian distribution.

To predict the distribution of the discrepancy between the true and estimated projections, we leveraged the fact that the same 1433 modern West Eurasian individuals were used to compute the PC axes. This ensures that the eigenvalues of the axes provide valid estimates for the variances of the PC scores. Furthermore, assuming that the mean of the PC scores is zero aligns with the PCA framework. These moments were propagated through the discrepancy function (Equation (10.8)-Equation (10.12)) to derive a predicted Gaussian distribution for the discrepancy.

As illustrated in Figure 10.3, the empirical distributions, visualized as blue covariance ellipses centered on zero, show strong agreement with the predicted distributions (depicted in gold) across all downsampling rates r. Notably, as the proportion of missing loci increases, the empirical spread of the subset projections widens, consistent with the predictions derived from the variance propagation method.

To quantify the similarity between the empirically observed and predicted distributions, we computed the Kullback-Leibler (KL) divergence between the two distributions obtained from modern genotype samples for multiple sets of omitted loci at varying downsampling rates. The

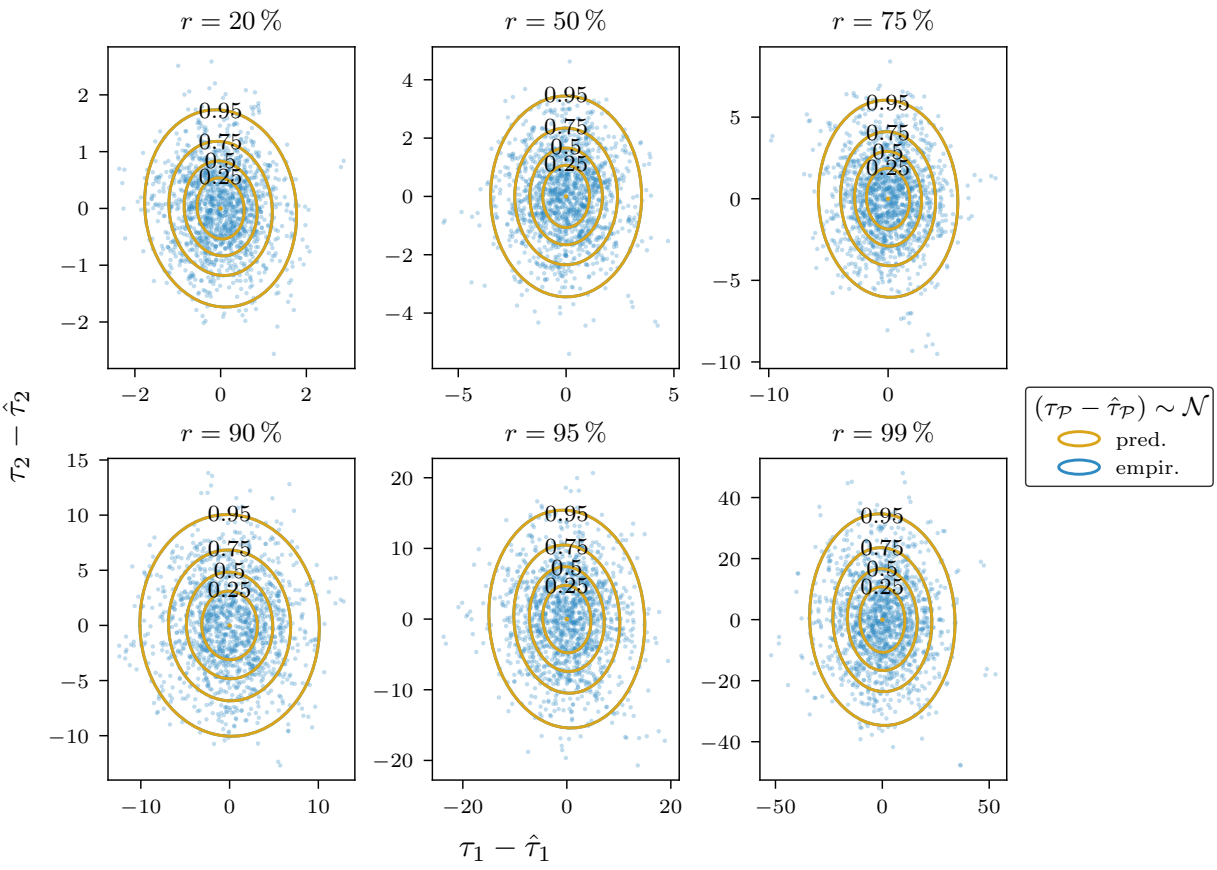


Figure 10.3: Comparison of empirically determined and predicted Gaussian discrepancy distributions for modern genotype sample embeddings. For each of 1,433 high-coverage modern West Eurasian genotype samples, a random set of loci was omitted at the specified downsampling rate r. The discrepancy in the first two principal components between the resulting SmartPCA sample embeddings and the reference embeddings was calculated and plotted as scatter points. From this data, the empirical Gaussian distribution was estimated and represented by blue contours at the specified quantiles. Using the corresponding sets of omitted loci, the Gaussian discrepancy distribution was predicted based on Equation (10.8)-Equation (10.12), with the predicted contours shown in gold overlaying the blue contours.

results demonstrate that the mean KL divergence remains close to zero across all rates of missing data, indicating a strong concordance between the empirical and predicted distributions (Figure D.3).

The results obtained serve as a proof of concept, as the mean and variance structure of the PC scores for the modern individuals used in this experiment were known. This is because the modern samples under investigation originated from the same distribution that was used to compute the PCA itself. In the next section, we extend this approach to predict the discrepancy distribution for out-of-distribution samples, such as ancient genotype samples.

Uncertainty Predictions for Ancient Genotype Samples

Across the examined 1,433 West Eurasian modern and 6,627 ancient samples, we observed higher genotype variation at most loci in ancient samples compared to modern ones (Figure 10.4a). This difference arises from the pseudohaploid SNP calling approach for ancient data, which randomly samples one allele at each locus due to the low coverage of degraded genomic material. The binary state (homozygous reference or

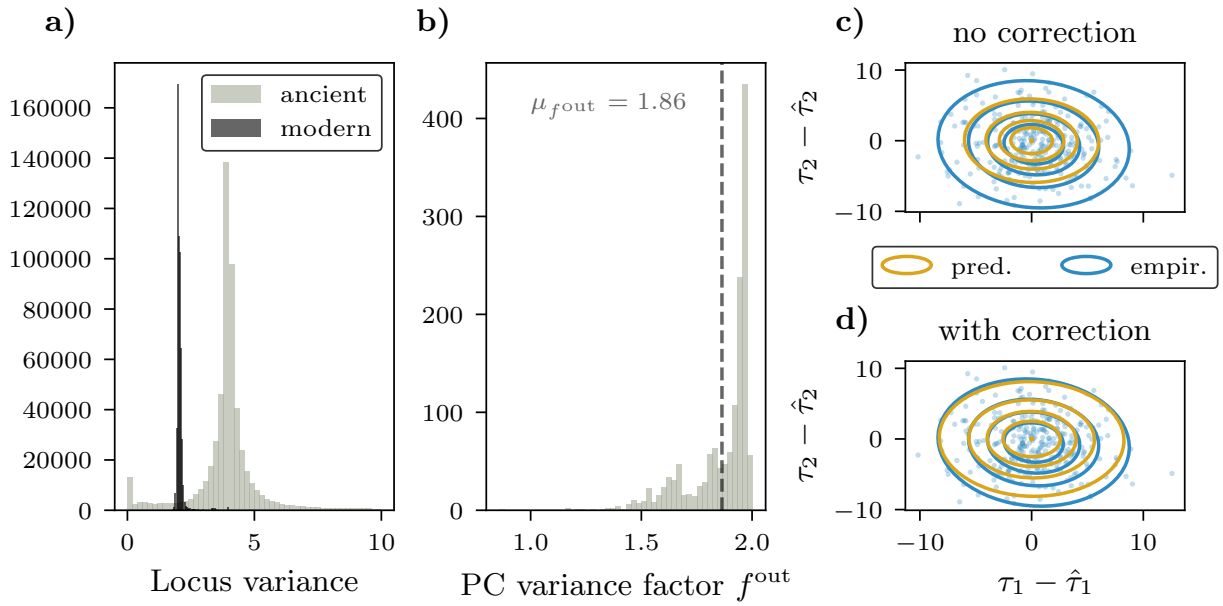


Figure 10.4: Locus variances for ancient and modern samples and their influence on prediction accuracy. (a) Distribution of 540,247 locus variances across 1,433 modern and 6,627 ancient samples. (b) Distribution of principal component (PC) variance factors, quantifying the relative variance of PC scores in ancient embeddings compared to modern ones. (c, d) Comparison of empirically determined and predicted Gaussian discrepancy distributions for ancient genotype sample embeddings. For 304 high-coverage ancient West Eurasian genotype samples, a random set of loci was omitted at the specified downsampling rate r, with results shown here for $r = 0.75$. Discrepancies in the first two principal components between the resulting SmartPCA sample embeddings and the reference embeddings were calculated and plotted as scatter points. Empirical Gaussian distributions were estimated from these discrepancies and visualized as blue contours at the specified quantiles. (c) Gaussian discrepancy distributions were predicted using the corresponding set of omitted loci and the model described in Equation (10.8)-Equation (10.12). (d) Variance adjustments (Equation (10.21)) were incorporated into the prediction framework to refine the predictions.

homozygous alternative) naturally increases the variance compared to the EIGENSTRAT encoding of modern genotype samples [14].

The predictive framework developed for modern samples provides a baseline for modeling PCA embedding uncertainty. However, ancient samples introduce additional challenges due to their higher genotype variance and distinct data characteristics. To address these challenges, we extend the variance propagation approach by incorporating adjustments specific to ancient samples. In the following, we outline these adjustments and evaluate their impact on predicting the discrepancy distribution.

To this end, we selected 304 high-coverage out of the 6,627 ancient samples and simulated missing data by omitting a single set of loci for each downsampling rate r. As shown in Figure 10.4c (for a downsampling rate of 0.75), the discrepancies between the SmartPCA embeddings and the reference embeddings are significantly larger for ancient samples compared to modern ones (see upper right subplot in Figure 10.3 for comparison). When using the same assumptions about the PC score means and variances as those applied to modern samples to predict the discrepancy distribution, the variance of the predicted distribution is notably underestimated (Figure 10.4c, golden contours). These findings underscore the importance of incorporating the increased genotype variation observed in ancient samples to accurately model their discrepancy distributions.

To account for the increased genotype variation observed in ancient samples, we introduced a factor f_i^{in} for each locus i, which quantifies

the relative variance of genotypes in ancient versus modern samples. Specifically, f_i^{in} is defined as:

$$f_i^{\text{in}} = \frac{\mathbb{V}_{\text{ancient}}[i]}{\mathbb{V}_{\text{modern}}[i]} \tag{10.19}$$

where $\mathbb{V}_{\text{ancient}}[i]$ and $\mathbb{V}_{\text{modern}}[i]$ denote the variances of genotypes at locus i, estimated across all 6,627 ancient and 1,433 modern West Eurasian samples, respectively.

To translate this locus-specific variance scaling into the PCA space, we projected the vector of factors $\boldsymbol{f}^{\text{in}} \in \mathbb{R}^D$ (where D is the total number of loci) onto the PC axes using the eigenvector matrix $\boldsymbol{V}$. The resulting projected factors $\boldsymbol{f}^{\text{out}}$ estimate the relative variance of PC scores in ancient embeddings relative to modern ones:

$$\boldsymbol{f}^{\text{out}} = \text{diag}(\boldsymbol{V}^\intercal \boldsymbol{f}^{\text{in}} \boldsymbol{V}) \tag{10.20}$$

Here, diag(·) represents the operation of forming a diagonal matrix from a vector. The resulting $\boldsymbol{f}^{\text{out}} \in \mathbb{R}^Q$ provides the variance adjustments required to account for ancient samples' increased genotype variation when predicting their discrepancy distributions.

The distribution of these variance factors (Figure 10.4b) reveals that ancient embeddings exhibit up to a two-fold increase in variance for their PC scores and a mean of 1.86. Incorporating this variance adjustment into the prediction framework,

$$\mathbb{V}[\boldsymbol{\tau}_Q] = \boldsymbol{f}^{\text{out}} \boldsymbol{\Lambda}_Q, \tag{10.21}$$

yields predicted contours that align closely with the empirical discrepancy distributions (Figure 10.4d). This approach markedly improves prediction accuracy compared to using unadjusted variance assumptions, as illustrated by the misalignment of contours without correction (Figure 10.4c). Furthermore, the improvement is consistent across various downsampling rates, as detailed in Figure D.4 and Figure D.5. The effectiveness of the variance correction is quantitatively supported by a substantial reduction in the Kullback-Leibler (KL) divergence between the empirically derived and predicted distributions, as shown in Figure D.3. These findings highlight the critical role of variance adjustments in accurately modeling PCA projection uncertainties for ancient genotype samples.

10.3.4 Real-World Data Analysis: Uncertainty Predictions for Ancient Samples

To evaluate the performance of the proposed framework on real-world data, eight ancient individuals from the Mesolithic until early Middle Ages from the AADR database were analyzed. These individuals, spanning a range of SNP coverage levels, were specifically selected because they project to diverse locations within the West Eurasian PCA map (Table 10.1). The principal component (PC) space was constructed from modern West Eurasian populations, and the selected ancient samples were projected onto this space using SmartPCA. For each sample, the discrepancy distributions were predicted based on the available and missing

Table 10.1: Properties of ancient test samples from the AADR. SNP coverage refers to the proportion of observed single nucleotide polymorphisms (SNPs) in the sample relative to the total of 540,247 SNPs considered in this work.

Nr.	Master ID	Locality	Period	SNP Coverage [in %]	Reference
1	STR266b	Germany	Middle Ages	4.0	[276]
2	I2014	Germany	Neolithic	1.7	[152]
3	VK470	Russia	Middle Ages	2.1	[168]
4	cth842	Turkey	Neolithic	2.8	[288]
5	I7207	Czechia	Bronze Age	70.2	[185]
6	I4884	Czechia	Bronze Age	76.3	[196]
7	I6531	Poland	Bronze Age	70.8	[196]
8	I11710	Slovakia	Iron Age	77.9	[206]

genotypes. These predictions, centered on the reference projections, were visualized as uncertainty regions in the PCA plot.

The results reveal a clear distinction between samples with high SNP coverage and those with low SNP coverage. Four of the samples, characterized by high coverage (less than 30 % missingness), exhibit narrow discrepancy distributions, indicating high confidence in their embeddings. In contrast, the other four samples, which have substantially lower SNP coverage (at least 96 % missingness), display broader discrepancy distributions. This increased uncertainty reflects the reduced reliability of projections for sparse samples.

The PCA plot in Figure 10.5 overlays the predicted discrepancy distributions onto the modern sample projections. The narrow distributions of high-coverage samples appear tightly clustered around their reference projections, whereas the low-coverage samples show a visibly expanded spread. Notably, these broader distributions intersect with multiple regions of the PC space, highlighting the potential ambiguity in interpreting the ancestry of sparse samples. This analysis underscores the importance of accounting for embedding uncertainty when interpreting PCA projections of ancient genotype samples with missing data.

10.3.5 TrustPCA: a User-friendly Tool to Assess SmartPCA Projection Uncertainties

As ancient genomics advances and PCA remains central in human population genetics, addressing uncertainties in SmartPCA projections caused by missing data is crucial. To this end, **TrustPCA**, a **T**ool for **R**eliability and **U**ncertainty in **S**mart**PCA**, was developed to apply the prediction framework from Section 10.2.5 in a user-friendly way.

Implemented in Python (v3.9) with the Streamlit framework [257], TrustPCA features an intuitive web interface. Users can upload genotype data in EIGENSTRAT format or explore preloaded datasets. Summary statistics, such as the number of missing loci per sample, are calculated and presented.

Using the PC space derived from modern West Eurasian genotype data (Section 10.3.1), TrustPCA enables users to project ancient samples similarly to SmartPCA. It predicts projection uncertainties based on missing loci in the provided samples, visualizing these as uncertainty ellipses on PC plots, representing different confidence levels for sample placements.

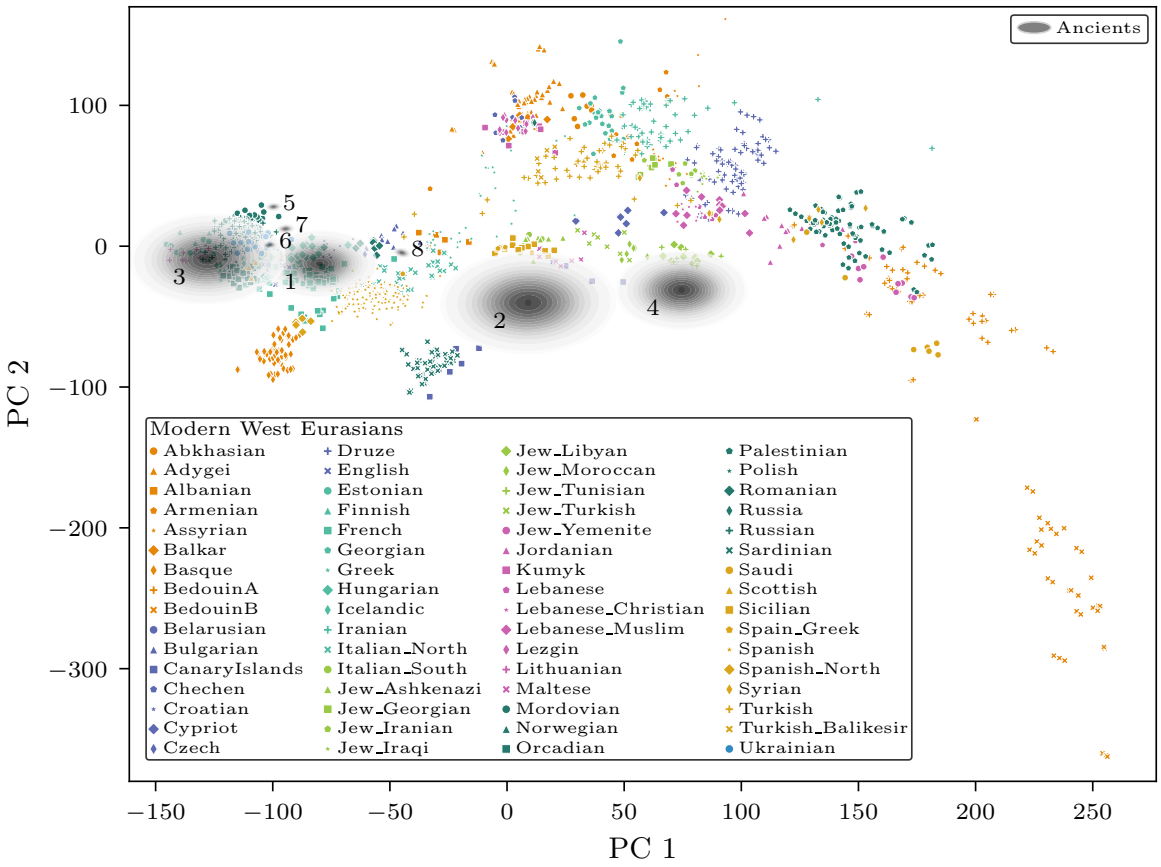


Figure 10.5: West Eurasian PCA map showing modern individuals, color-coded by their locality, projected onto the first two principal components. Predicted embedding distributions (Section 10.2.5) are displayed in gray for eight ancient samples with varying SNP coverage (detailed in Table 10.1). These distributions illustrate the uncertainty in the projections, with narrower distributions corresponding to higher SNP coverage and broader distributions indicating lower coverage. The figure was generated using TrustPCA (Section 10.3.5), which implements the proposed prediction framework.

All outputs, including statistical summaries and visualizations, can be exported as PDFs for easy integration into research workflows. TrustPCA is openly available at `https://trustpca-tuevis.cs.uni-tuebingen.de/`, with the source code accessible at `https://github.com/Integrative-Transcriptomics/TrustPCA/tree/main`.

10.4 Discussion

In this study, we systematically examined the effects of missing data on the reliability of SmartPCA projections for ancient human genotype samples and developed a probabilistic framework to quantify the resulting embedding uncertainty. Through simulations, we demonstrated that as the proportion of missing SNPs increases, the accuracy of SmartPCA projections decreases (see Figure 10.2). This spread becomes more pronounced in cases of extreme SNP sparsity (see Figure D.2). As illustrated in Figure D.1, a substantial number of West Eurasian ancient individuals in the AADR dataset exhibit low SNP coverage. Specifically, approximately 15.2 % of the ancient individuals have a SNP coverage below 10 %, making them especially susceptible to inaccurate SmartPCA projections.

When repeating the simulation with out-of-distribution archaic human individuals—such as Neanderthals or Denisovans—the spread of subset projections is significantly larger compared to typical ancient samples (Figure D.6). This result highlights a higher degree of uncertainty in embedding positions for individuals with little genetic overlap with the modern reference population.

By analyzing real-world ancient samples from the AADR database [167], we have shown that our probabilistic model provides reliable estimates of the discrepancy distributions, offering a quantitative measure of the uncertainty in PCA projections (see Figure 10.5). Importantly, the predictions highlight that high-SNP-coverage samples exhibit tight and reliable embedding distributions, while low-coverage samples display broader and less precise embeddings. These results underscore the limitations of interpreting PCA projections of sparse ancient samples without accounting for the inherent uncertainty introduced by missing data.

As PCA is frequently used in ancient DNA population genetics to generate hypotheses about population shifts or continuity for further testing with quantitative methods like f-statistics, inaccuracies in the assumptions derived from PCA placements can significantly impact subsequent analyses (e.g., [75, 283]). Specifically, misinterpretation of PCA results may lead to certain hypotheses being overlooked or inadequately tested.

The framework we present not only highlights these limitations but also offers a practical solution: with TrustPCA, a tool that integrates this probabilistic model, we provide researchers with uncertainty estimates alongside PCA projections. This approach enables more robust interpretations of PCA results and helps mitigate the risk of misinterpretations in studies of ancient population structure and ancestry.

The concerns raised by Elhaik [58] regarding the reliability of PCA in population genetics highlight the broader challenges in using this method. These include misleading projections, dependence on reference populations, and possibly incorrect interpretations. Our work addresses some of these challenges by quantifying and visualizing the uncertainty in PCA projections. By explicitly modeling the variability introduced by missing data, our probabilistic framework reduces the risk of overinterpretation and enables more robust conclusions. However, it does not resolve all previously mentioned issues. Notably, the choice of reference populations remains a critical factor influencing PCA results, and the interpretation of projections can still be subjective. Future research could investigate how reference population biases interact with data sparsity to influence PCA outcomes.

It is important to note that this study focused exclusively on the West Eurasian PCA map, which forms the basis of the majority of current human population genetic studies. While this choice allowed us to focus on a well-characterized dataset, the conclusions drawn here may not generalize to other geographic regions or datasets with distinct profiles of genetic structure. Future research should explore whether observed uncertainty patterns persist in other genetic landscapes and reference populations, particularly in regions with greater levels of genetic diversity or admixture.

A key prerequisite for accurately estimating the distribution of the true embedding using our probabilistic approach is that the sample of interest originates from a data distribution similar to the one used to compute the eigenvectors and eigenvalues. If this condition is not met, the variance of the predicted embedding distribution may be underestimated (Figure D.4) or overestimated (Figure D.6). For ancient individuals with a genotype variance structure different from that of the modern population, a correction for this variance disparity was applied, resulting in well-aligned predicted distributions (Figure D.3 & Figure D.5).

The issue of missing data has also led to the widespread use of genotype imputation to fill in gaps in ancient datasets [33, 97, 239]. A recent study by Allentoft et al. [5] demonstrated that using fully imputed ancient genomes to construct the PCA—rather than relying solely on modern reference populations—can improve the resolution of ancient individuals in PCA space. However, while imputation can significantly improve the number of successfully genotyped SNPs, it introduces its own challenges. Imputed genotypes are probabilistic estimates based on linkage disequilibrium patterns in reference panels, and their accuracy depends heavily on the quality and compatibility of the reference panel with the sample being studied [13]. Errors in imputation, particularly for SNPs with low read coverage or in regions not well represented in the reference panel, can lead to biased PCA projections [183]. Future work could integrate imputation uncertainty into PCA models, enabling researchers to differentiate between uncertainty arising from imputed versus directly observed SNPs.

Another source of uncertainty arises from genotype calling, particularly in ancient DNA studies where low read coverage and contamination can result in ambiguous base calls [198]. Furthermore, the level of accuracy with which ancient individuals—called using pseudo-haploid genotypes—can be projected onto a map constructed from modern individuals with diploid genotype calls remains unclear. These uncertainties are compounded by the encoding of genotypes in population genetics analyses, which often assume fixed values (e.g., 0, 1, or 2 for diploid genotypes). This deterministic encoding overlooks the probabilistic nature of genotype likelihoods derived from sequencing data [128, 189, 190], introducing hidden uncertainties into downstream PCA projections. Incorporating genotype likelihoods directly into PCA models or using probabilistic representations of genotypes could enhance the robustness of PCA projections, particularly for low-SNP-coverage ancient samples.

10.5 Conclusions

In conclusion, this work emphasizes the importance of considering projection uncertainty when applying PCA to sparse ancient genotype data. By quantifying this uncertainty, we offer researchers a powerful tool for making more reliable and reproducible inferences. However, addressing the broader challenges of PCA in population genetics, including reference population dependency and imputation bias, will require continued methodological innovation and critical evaluation of current practices.

Part III.

Taxonomic Analysis: Advances in Classification and Visual Analytics

11. Improving Taxonomic Classification with Feature Space Balancing

Disclaimer 11.1 Chapter 11 is based on the peer-reviewed publication with the following co-author contributions:

W. Fuhl, S. Zabel, and K. Nieselt. "Improving taxonomic classification with feature space balancing". *Bioinformatics Advances* 3.1 (2023) [68]

	Ideas	Analysis & Interpretation	Code	Writing
W. Fuhl	50 %	45 %	80 %	15 %
S. Zabel	35 %	45 %	20 %	80 %
K. Nieselt	15 %	10 %	0 %	5 %

Abstract

Modern high-throughput sequencing technologies, such as metagenomic sequencing, generate millions of sequences that need to be assigned to their taxonomic rank. Modern approaches either apply local alignment to existing databases, such as MMseqs2, or use deep neural networks, as in DeepMicrobes and BERTax. Due to the increasing size of datasets and databases, alignment-based approaches are expensive in terms of runtime. Deep learning-based approaches can require specialized hardware and consume large amounts of energy. In this paper, we propose to use *k*-mer profiles of DNA sequences as features for taxonomic classification. Although *k*-mer profiles have been used before, we were able to significantly increase their predictive power significantly by applying a feature space balancing approach to the training data. This greatly improved the generalization quality of the classifiers. We have implemented different pipelines using our proposed feature extraction and dataset balancing in combination with different simple classifiers such as bagged decision trees or feature subspace KNNs. By comparing the performance of our pipelines with state-of-the-art algorithms such as BERTax and MMseqs2 on two different datasets, we show that our pipelines outperform these in almost all classification tasks. In particular, sequences from organisms that were not part of the training were classified with high precision.

11.1 Introduction

Metagenomic analysis of environmental or clinical samples has proven to be highly relevant for characterizing biodiversity [94, 96] and has shown that the microbiome has a significant impact on the health of organisms [233, 282]. The advent of metagenomic sequencing required the development of efficient algorithms to classify the taxonomic origin of sequences and to infer the composition of a microbial community. Taxonomic classification describes the process of assigning a sequence to a specific node in the taxonomy on a pre-defined taxonomic rank, for example, genus (Section 7.1.2) [177].

Taxonomic classification of whole-genome shotgun data is computationally challenging for two main reasons: first, millions of sequences need to be classified, and second, the number of available classes, i.e., reference genomes in a database, is constantly growing. Therefore, fast and resource-efficient algorithms are essential to cope with the amount of data available.

A vast variety of methods have been released, which classify algorithmically into traditional alignment-based and *k*-mer approaches, and more recent methods that apply machine learning classifiers. BLAST [6] and Diamond [34, 35] are famous methods that use seed-and-extend approaches to align the query sequences with a database of microbial genomes. Although precise, these alignment-based tools suffer from a low recall and are relatively slow. To increase speed, methods using local alignments of *k*-mers have been developed, such as Kraken2 [286] or MMseqs2 [256]. For a more comprehensive review and benchmarking of modern taxonomic classifiers, see [248].

The above approaches rely heavily on a well-curated database and taxonomy. Machine learning (ML) classification methods attempt to circumvent this dependency. *k*-mer profiles have been used as features for simple classifiers, such as the Naïve Bayes Classifier (NBC, [234, 235, 294], or Support Vector Machines (SVMs, [279]). However, they have been shown to underperform in terms of precision or to be inefficient in terms of memory or runtime [174].

Recently, deep neural network (DNN) architectures were developed to capture more complex dependencies between data (sequences) and target variables (taxonomic class). DeepMicrobes [146] implements a deep learning architecture consisting of many long short-term memory (LSTM) cells followed by a self-attention mechanism and a multilayer perceptron for classification. DeepMicrobes uses a 12-mer embedding as the first layer of the DNN. BERTax [181] uses a transformer model from natural language processing (NLP), trained on large amounts of DNA sequences, to extract a feature representation of the DNA sequence. This feature representation is then fed to a small, fully connected neural network for the final classification. Both approaches, BERTax and DeepMicrobes, show a similar accuracy as alignment-based methods and can be executed extremely fast on GPUs. However, DL methods are often considered black-box approaches that can be difficult to interpret. In addition, the need for specific hardware, for example, to efficiently train the DNNs, makes these approaches inapplicable for some users.

In this work, we evaluate several pipelines for the taxonomic classification of DNA sequences at three different taxonomic levels (superkingdom, phylum, and genus). All pipelines extract *k*-mer profiles from the DNA sequences and use simple machine learning approaches, such as bagged decision trees or subspace KNNs, for classification. A significant increase in performance could be achieved by applying an algorithm that balances the distribution of training samples across the feature space prior to training.

Compared to alternative approaches, such as BERTax, DeepMicrobes, and MMseq2, our pipelines improved classification performance (macro average precision) at phylum level, especially for sequences originating from unseen genera, by ~15 % compared to BERTax. When using

closely related species for training, our pipeline significantly outperformed MMseqs2 taxonomy at genus level, improving the performance by ~11 %.

11.2 Material & Methods

11.2.1 Data

Data provided by the developers of BERTax [181] was used in this work. The datasets include sequences of length 1500 nt retrieved from archaeal, eukaryotic, viral, and bacterial genomes (see Dataset S1 in [181] for accession numbers). We used a subset of datasets that the authors in [181] referred to as the *distantly related* and the *final model* dataset. Data was downloaded via https://osf.io/qg6mv/ (10.09.22). Training and test set splits were adopted. Table 11.1 summarizes the characteristics of the datasets. Generally, all classes are covered by at least 10,000 DNA sequences of length 1500 nt.

As summarized in Figure 4 in [181], the distantly related and final model datasets were created to stress different aspects of taxonomic classification. In the final model dataset, sequences from the same genus are part of the training as well as the test set. Predicting the genus level is therefore easy because the classifier has seen similar sequences during training. On the other hand, in the distantly related dataset, all sequences originating from one genus are used for either training or testing. When predicting the phylum level of a sequence, the classifier has seen sequences from the same phylum during training, but not from the same genus. This classification task is much more difficult and resembles the classification of novel organisms during metagenomic analysis. In summary, the datasets are designed in a way that test data are more similar to the training data in the final model dataset than in the distantly related dataset due to the selection of samples for each subset.

11.2.2 Code & Data Availability

Code and data are available at https://es-cloud.cs.uni-tuebingen.de/d/8e2ab8c3fdd444e1a135/?p=%2FTaxonomyClassification&mode=list.

11.2.3 Feature Extraction

From each sample—a DNA sequence of length 1500 nt—we extracted a set of features by computing the distribution over occurring k-mers in the sequence. To this end, the relative frequency of each of the potentially occurring 4^k k-mers is calculated as the quotient of the absolute count and the sum over all k-mer counts of the sample. The relative frequency of each k-mer is used as a feature.

Table 11.1: Characteristics of the distantly related and the final model dataset provided by [181] that were used in this study. [seqs] refers to the number of sequences of length 1500 nt.

	Distantly related	Final model
Nr. of classes for		
superkingdom	4	4
phylum	30	43
genus	0	155
Size of		
training set [seqs]	2,245,416	5,311,920
test set [seqs]	53,400	88,000

11.2.4 Dataset Balancing

After computing the relative k-mer distribution from all sequences from the training dataset, the training set is imbalanced since some feature combinations are more prevalent than others. This leads to areas in the feature space that are densely covered, whereas other regions are sparsely occupied (see Figure 11.1a). Our approach to dataset balancing aims to establish a more uniform distribution of samples across the feature space. This must not be confused with the commonly known problem of imbalanced classes. To this end, the feature space is discretized using a high-dimensional equidistant grid ($D = 4^k$). The number of grid cells G per feature dimension is a parameter to be learned. The user sets the final number of training samples N. After initially adding $0.3 \cdot N$ randomly selected samples from all classes to the empty feature space, samples are added to the feature space until N samples have been added. Initialization is done to accelerate the balancing algorithm. Adding more samples to the feature space after initialization is shown in Figure 11.1b and works as follows: Let S_i be a randomly selected potential new sample from the training data, let g be the grid cell of S_i in the feature space, and let C_g be the number of samples already present in the grid cell g of S_i. Further, let $C_{\max}$ be the highest count over all grid cells observed so far. The sample S_i is accepted to be added to the grid cell g if $C_g < C_{\max}$ and rejected if $C_g = C_{\max}$. By that, empty areas in the feature space are preferably filled. To prevent the algorithm to get stuck, a sample is accepted independent of $C_{\max}$ after ten consecutive unsuccessful trials.

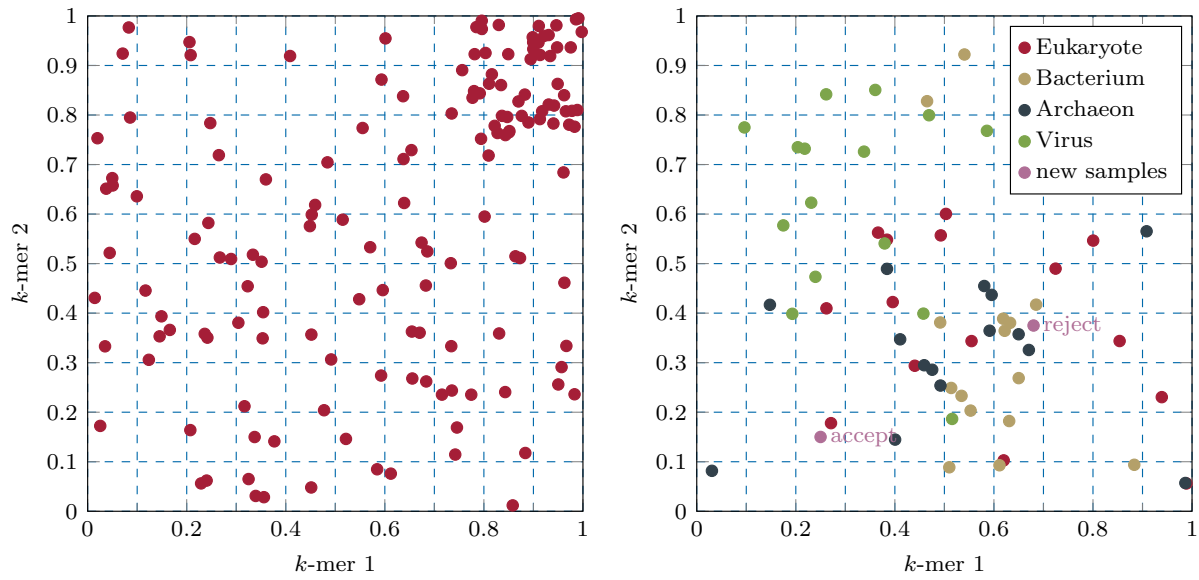
11.2.5 Machine Learning Classifiers

For classification, we used different machine learning models as ensembles and applied the built-in functions of Matlab 2022b, including bagged decision trees, random subspace k-nearest neighbors (random subspace KNNs), random subspace discriminant analysis, support vector machines (SVMs) with different kernels (linear, Gaussian, polynomial), and a small neural network with one hidden layer of size 512 (Section 7.2). In general, hyperparameters were not optimized and default parameters were used.

To evaluate classification performance macro average precision (MAP) was used as a metric, which is given by the class-wise average precision

$$\mathrm{MAP} = \frac{\sum_{i \in \mathbb{C}} \frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FP}_i}}{|\mathbb{C}|}, \tag{11.1}$$

where $\mathbb{C}$ is the set of all classes, TP_i the true-positives of class $i \in \mathbb{C}$, and FP_i the false-positives of class $i \in \mathbb{C}$.



(a) Imbalanced data **(b)** Data balancing

Figure 11.1: Simplified visualization of the dataset balancing approach. The feature space is considered two-dimensional, where each dimension represents the relative frequency of a specific k-mer. Note: for visualization reasons, we neglected that frequencies must add up to one. (a) Samples are not uniformly distributed across the feature space. The upper right area contains relatively more samples than the rest of the feature space, thus the dataset is imbalanced. (b) Balancing the feature space distribution. The feature space is initialized with 15 samples from all four superkingdom classes. The number of grid cells G (per dimension) is set to ten. Due to the current maximal cell count $C_{\max}$ of 5, a new potential next sample is accepted ($1 < C_{\max}$) or rejected ($5 = C_{\max}$), respectively.

11.2.6 Used Hardware

We used an AMD Ryzen 9 3950 with a 16-Core (32 Threads) processor (3.50 GHz) and 64 GB DDR4 memory to evaluate runtime and resource consumption.

11.3 Results

We propose a pipeline for the taxonomic classification of DNA sequences, with the focus of being resource-saving in terms of runtime and memory, while at the same time performing equally well as existing accurate but resource-demanding methods such as BERTax and DeepMicrobes. To achieve this, simple classification algorithms need to be used instead of complex deep learning methods. However, these simple methods can not handle unstructured data like for example text data. DNA sequences, as typically observed in shotgun sequencing, are highly unstructured because they originate from random locations in the genome, so positional information is not comparable between sequences. We, therefore, extract meaningful features from the DNA sequences to be used as input for the classifiers. Given DNA sequences of length 1500 nt from four superkingdoms— archaea, bacteria, eukaryotes, and viruses— (Figure 12.1a), we extracted the relative frequency of all k-mers, e.g. if $k = 3$ the frequencies of all possible $4^3 = 64$ 3-mers, resulting in 64 features (Figure 12.1b). The choice of the parameter k influences the dimensionality D of the input data—$D = 4^k$—and thereby the efficiency of the pipeline.

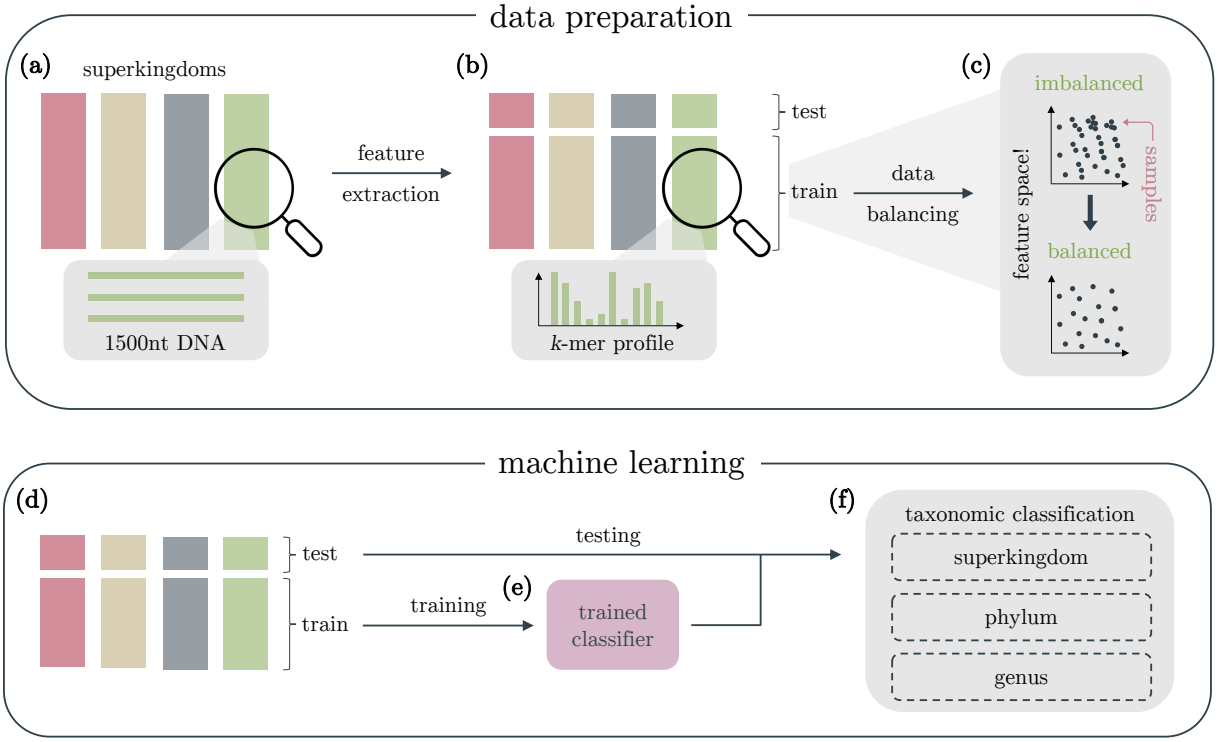


Figure 11.2: Proposed pipeline. (a) Sequences of length 1500 nt originating from four superkingdoms are used as input. (b) From each sequence, k-mer profiles—the relative frequency of all 4^k possible words of length k—are extracted and used as features. (c) Training data is balanced using an undersampling approach. Dense regions of the feature space are thinned out. This reduces the size of the training set. (d, e) The balanced and curated training data is used to train simple supervised learning classifiers. (f) Depending on the taxonomic rank of the given label, the test sequences are taxonomically classified at the superkingdom, phylum, or genus level.

When inspecting the distribution of the training samples across the 4^k dimensional feature space, it became apparent that some areas of the feature space are densely covered, whereas other regions are sparsely occupied. As a result, by construction, classifiers focus mainly on the dense regions by fitting complex decision boundaries to the data, which can lead to overfitting in these areas. At the same time, sparse regions become less important, potentially leading to underfitting there. To advance a uniform distribution of samples across the feature space we applied an undersampling approach keeping most of the data in sparse regions and decreasing the number of samples in dense regions (Figure 12.1c). It is important to note that empty regions remain empty. A high-dimensional equidistant grid is used to discretize the feature space since the features (frequencies) are continuous variables. To approximate a uniform distribution, the number of samples in each cell could be counted and samples removed from cells where the cell count exceeds a certain threshold. While being a valid approach, it is unfeasible for large training sets. Therefore, we implemented a time-efficient approximation that allows the user to specify the final training set size N in advance. For details see Section 11.2.4.

The reduced and balanced training set is subsequently used to train simple classifiers and classifier performance is evaluated in terms of macro-average precision (MAP) on the test dataset (Figure 12.1d-f). In various experiments, taxonomic classification is performed at different taxonomic levels, including in order of decreasing difficulty, superkingdom, phylum, and genus.

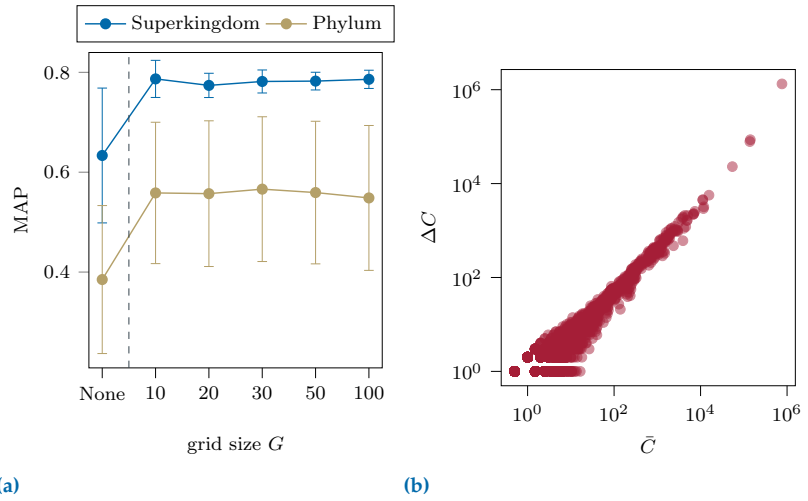


Figure 11.3: (a) Performance evaluation of a classifier (ensemble of bagged decision trees) trained on unbalanced (None) or balanced training data of the distantly related dataset using different grid sizes G. Relative k-mer frequencies were used as features. Mean MAP values and 1σ-intervals over different choices of $k \in \{1, 2, 3, 4, 5\}$ are shown. (b) Effect of data balancing ($N = 6 \cdot 10^5$, $G = 10$) on the sample distribution of the distantly related training set. $\bar{C} = \frac{C_{\text{before}} + C_{\text{after}}}{2}$ describes the average sample count per grid cell before and after data balancing. $\Delta C = C_{\text{after}} - C_{\text{before}}$ describes the number of samples that were removed per grid cell.

In the following sections, the pipeline is applied to and evaluated on two datasets—called distantly related and final model—that were previously published by [181] and contain 1500 nt DNA sequences from four different superkingdoms. For details see Section 11.2.1.

11.3.1 Dataset Balancing Improves Classification Performance

In this section, the effect of dataset balancing and different parameter choices of the balancing algorithm—the grid size G and the maximal number of training samples N kept—on the performance of taxonomic classification is evaluated. For that, classifiers are trained on the (un)balanced distantly related dataset, and performance is assessed on the test set. All three classifiers (bagged decision tree, subspace KNN, small neural network) increased their classification performance on phylum level as more training samples were added but plateaued at more than $6 \cdot 10^5$ training samples (see Figure S1). Therefore, N was set to $6 \cdot 10^5$ for further experiments.

Figure 11.3a shows that data balancing improves the classification performance of an ensemble of bagged decision trees. In all experiments, regardless of the choice of G, MAP values could be increased by up to 0.18 on average by balancing the training data (see Table S1 for exact performance values). Since larger grid sizes increase the complexity of the balancing algorithm, but do not significantly improve the performance, $G = 10$ was chosen for subsequent experiments.

As shown, our approach to data balancing has a positive effect on the performance of taxonomic classification. Reasonable choices have been identified for both G and N, despite the lack of an exhaustive search for optimal parameters. We used these parameters to balance the distantly related training set. Figure 11.3b shows a linear increase in the number of training samples removed ΔC per grid cell for increasingly dense grid cells $\bar{C}$ in a log–log plot. This confirms that more training samples are removed in dense regions of the feature space than in sparse regions.

Table 11.2: Performance evaluation of different ensemble classifiers trained on the distantly related dataset's balanced training data ($G = 10, N = 6 \cdot 10^5$). Relative 3-mer frequencies were used as features. Classification was performed at the superkingdom and phylum, and performance values are given as MAP. The best results appear in bold.

Classifiers	Superkingdom level	Phylum level
Bagged decision trees	0.8164	**0.6971**
Subspace KNN	**0.8823**	0.6756
Subspace discr. analysis	0.6020	0.4894
Linear SVM	0.6462	0.4505
Gaussian SVM	0.6635	0.5705
Polynomial SVM	0.6737	0.6013
Neural network	0.8121	0.6598

Table 11.3: Mean performance of the classifiers subspace KNN and ensemble of bagged decision trees using 5-fold cross validation on the balanced training data ($G = 10, N = 6 \cdot 10^5$) of the distantly related dataset. Relative k-mer frequencies were used as features. The first column shows the respective choice of k. Classification was performed at the superkingdom and phylum level, and performance values are given as MAP. The best results appear in bold.

	Subspace KNN		Bagged decision trees	
k	Superkingdom	Phylum	Superkingdom	Phylum
1	0.5870	0.2588	0.6328	0.3448
2	0.8584	0.6537	0.8664	0.7124
3	0.9438	0.8578	**0.9250**	0.8418
4	**0.9557**	**0.8809**	0.9103	**0.8753**
5	0.9235	0.8326	0.8955	0.8194

11.3.2 Evaluation of Machine Learning Methods

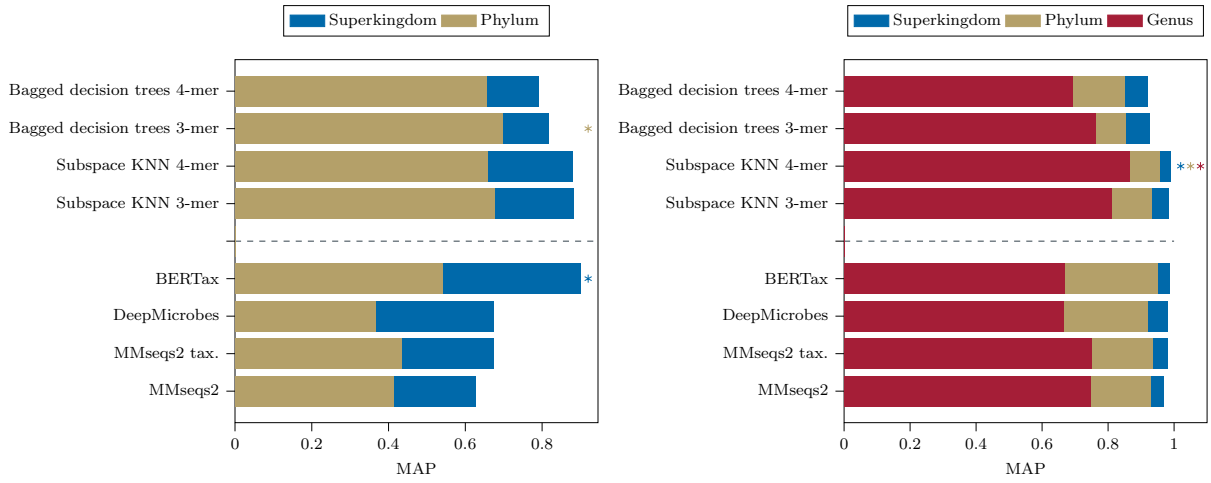
To ensure a resource-efficient pipeline, only simple classifiers were considered and their performance was compared. Using default parameters, ensembles of bagged decision trees and ensembles of subspace KNNs classified test data from the distantly related dataset best (Table 11.2), and were therefore considered for the final evaluations.

11.3.3 Evaluation of Different k-mer Sizes

In this section, we show how the size of k used to extract relative k-mer frequencies from DNA sequences affects classifier performance. To this end, different ks and their combinations were evaluated by a 5-fold cross-validation on the training data of the distantly related dataset using ensembles of bagged decision trees and subspace KNNs. As seen in Table 11.3, the relative frequencies of 3-mers or 4-mers classified sequences best at both taxonomic levels.

11.3.4 Final Performance Evaluation & Comparison to Existing Methods

In the final evaluation, we compared the accuracy of our pipeline with state-of-the-art approaches for taxonomic classification on both the distantly related and the final model dataset. In the paper of [181], MMseqs2 in two different modes [256], DeepMicrobes [146], and BERTax [181] applied to the same datasets performed best, and performance values are shown in terms of macro average precision in the bottom rows of Figure 11.4. Exact values are listed in Table S2. It should be noted, that the performance values for these methods were not reproduced but copied from [181] due to execution problems. However, the same data and dataset splits in training and test sets have been used to make the results comparable. We propose four pipelines that use k-mer distributions as features and perform our proposed balancing approach (Figure 11.4, Tab. S2). Overall, all methods performed better on the final model dataset



(a) Distantly related data

(b) Final model data

Figure 11.4: Performance comparison in terms of MAP of several pipelines implementing our approach (upper four bars) with state-of-the-art methods. Note that performance values for state-of-the-art methods were taken from [181]. Results for the distantly related (a) and final model dataset (b) are shown. The best classification performances are labeled by asterisks.

Methods	Total memory consumption [GB]	Execution time [s]
MMseqs2	30	55
BERTax	15	3,297
Subspace KNN (*)	19	2,486
Bagged decision trees (*)	1	335

Table 11.4: Runtime and memory requirements to classify the phylum level of all test set sequences of the distantly related dataset. Ensemble subspace KNN and ensemble bagged decision trees were used to classify 3-mer profiles. (*) Converted into C++ functions.

(Figure 11.4b) compared to the distantly related dataset (Figure 11.4a) on both superkingdom and phylum levels. Test data sequences from the final model dataset are easier to classify because both test and training contain sequences originating from the same genus, whereas in the distantly related dataset sequences from one genus are assigned to either test or training data. It can be further observed that a more specific taxonomic classification is generally more difficult and performance values of all methods drop for phylum (and genus) level classification. When comparing all methods, our proposed ensemble of subspace KNN classifiers on 4-mer distributions outperformed all state-of-the-art methods on the final model dataset (Figure 11.4b). Especially, on the genus level a performance increase of 0.1134 is denoted compared to the taxonomy mode of MMseqs2. On the distantly related data, all proposed pipelines performed superior to the state-of-the-art methods at phylum level classification with an increase in MAP of up to 0.1561. On the superkingdom level, BERTax outperformed our best result by 0.0183 MAP.

We compared the runtime and the memory requirements of MMseq2, BERTax, and two of our proposed pipelines to classify the phylum level of all sequences in the distantly related test set on a CPU server with 16 cores, 32 threads, and 64 GB of memory (Table 11.4). Our pipelines took more time than MMseqs2, but less time than BERTax. The ensemble of bagged decision trees was particularly economical in terms of memory consumption.

11.4 Discussion

Machine learning, and in particular deep learning approaches, have emerged as promising alternatives to alignment-based methods for taxonomic classification. The number of unclassifiable DNA sequences could be reduced by not relying on well-curated databases. Inexplicability and the need for specific hardware (GPUs) to run efficiently are known drawbacks of deep learning approaches. Many tools like Deep Microbes that perform taxonomic classification rely on k-mers or their frequencies. More recently, BERTax has been introduced, a DNN using NLP that takes the whole sequence as input. BERTax—as claimed by the authors—has learned the 'language' of DNA and outperforms alternative methods in several classification tasks.

This paper presents several exemplary pipelines for the taxonomic classification of sequences. Classification at several taxonomic ranks (superkingdom, phylum, genus) is considered. The strength of the pipelines was most evident on a dataset, where samples in the test and training sets were not allowed to be from the same taxonomic genus. All four of our best-performing pipelines significantly outperformed BERTax, DeepMicrobes, and MMseqs2 at the phylum level. Therefore, our pipelines seem to work particularly well on sequences from novel organisms, i.e., sequences from low-rank taxa that were not part of the training. In addition, our pipelines were superior to alternative methods for classifying closely related sequences at the genus level, which is considered a more difficult task than classifying higher taxonomic ranks.

Although k-mer frequencies have been used by other methods before as features, we could substantially improve their predictive power by combining a training data balancing approach with subspace KNNs and bagged decision trees. We believe that if the training data is more uniformly distributed across the feature space will prevent overfitting in dense areas and underfitting in sparse areas. Undersampling in dense areas significantly increased performance (see Figure 11.3a). In addition, reducing the size of the training set makes training faster. Both classifiers, subspace KNNs, and bagged decision trees, use subsampling of features and training samples, respectively, and contribute further to a model that generalizes well. Using k-mer frequencies as features has the additional advantage that sequences of any length can be handled within one experiment.

In terms of resources, our proposed pipelines classified sequences faster than BERTax, but slower than MMseq2. Although fast, MMseqs2 required most memory and did not perform well in many classification experiments. Our pipeline that uses bagged decision trees was the most economical in terms of memory. In general, DL models, such as transformer models, are known to be time-consuming to train. However, our pipelines could be trained within minutes (subspace KNNs: $\sim$ 14 min, bagged decision trees: $\sim$ 9 min).

So far, our pipelines are more of a proof of concept that simple machine learning can be superior to deep learning approaches for taxonomic classification tasks, and there is room for improvements: Future work could include a more efficient implementation and hyperparameter optimization of the classification methods to improve performance further.

In this paper, we show that *k*-mer profiles are predictive features for taxonomic classification, and when used in combination with dataset balancing and simple machine learning models outperform DL methods. The fact that they are independent of the sequence length and do not require particular hardware are additional features that make the pipelines a promising prototype for further development.

BLASTphylo—an Interactive Web Tool for Taxonomic and Phylogenetic Analysis of Genes

12.

Disclaimer 12.1 Chapter 12 is based on the peer-reviewed publication with the following co-author contributions, and was presented at the 18th International Symposium on Bioinformatics Research and Applications (ISBRA), 2022, in Haifa, Israel:

S. Zabel, J. Müller, F. Götz, and K. Nieselt. "BLASTphylo: An Interactive Web Tool for Taxonomic and Phylogenetic Analysis of Prokaryotic Genes". *Journal of Computational Biology* (2023) [292]

	Ideas	Analysis & Interpretation	Code	Writing
S. Zabel	30 %	65 %	30 %	70 %
J. Müller	10 %	30 %	70 %	25 %
F. Götz	30 %	0 %	0 %	0 %
K. Nieselt	30 %	5 %	0 %	5 %

Abstract

Identifying the function of a protein is crucial to uncovering its role in the cellular complex. Computationally, the most common approach is to search for homologous proteins in a large database of proteins of known function using `BLAST`. One goal of such an analysis is the identification and visualization of the protein in the taxonomy of interest. Another goal is the reconstruction of the phylogenetic history of the protein. However, the `BLAST` result provides information about the occurrence of the protein in the taxonomy and its putative function mainly in a tabular format. This requires manual intervention and makes taxonomic identification laborious. Although various tools exist to visualize and annotate large-scale trees, none of them intuitively and interactively visualize the protein's occurrence in the taxonomy for different taxonomic ranks. To target this gap we developed `BLASTphylo`, a web tool that combines `BLAST` with automatic taxonomic mapping and phylogenetic analysis and presents the results in interactive visualizations. We demonstrate the functionalities of `BLASTphylo` in two case studies. `BLASTphylo` is available via https://tuevis.cs.uni-tuebingen.de/blastphylo/.

12.1 Introduction & Motivation

Characterizing a protein involves the exploration of its taxonomic occurrence as well as its evolutionary relationship to homologous proteins (Section 7.1.3) [212]. With the advent of high-throughput sequencing techniques, databases have accumulated many sequenced and possibly annotated genomes. The availability of these huge collections of predicted and functionally annotated genes allows for large-scale taxonomic and phylogenetic analyses (Section 7.1.5), potentially facilitating the prediction of protein functions [56, 70] or providing evidence for horizontal

Code is available at the Github repository `Integrative-Transcriptomics/BLASTphylo`, the webtool is available at `https://tuevis.cs.uni-tuebingen.de/blastphylo/`

gene transfer[1] (HGT) events [57, 127]. Although they provide useful information about gene distributions, large libraries such as UniProtKB or NCBI RefSeq require automated examination and make manual analysis quite time-consuming.

1: Horizontal gene transfer refers to the movement of genetic material across the barriers that normally restrict mating. This is in contrast to the vertical transmission of genetic material from parent to offspring [119]

Typically, a taxonomic or phylogenetic analysis pipeline starts with the search for homologous proteins in a database (e.g., RefSeq) with the help of `BLAST` [6]. The resulting unstructured list of putative orthologs and paralogs requires additional manual intervention to compute and visualize the taxonomic distribution and phylogenetic relation of these homologs. This includes the selection of one BLAST hit for each taxon on different taxonomic ranks, i.e. selecting one sequence for each species, the taxonomic visualization of the selected hits, the computation of a multiple sequence alignment, and the subsequent reconstruction of the phylogeny (Section 7.1.4). Automating the processing of a BLAST run would highly facilitate the analyses. There have been several attempts to summarize the occurrence and genome-wide comparison of genes, domains, protein clusters, or functional units across the taxonomy in databases such as Pfam [180], Entrez Protein Clusters, TIGRFAM [50] or KEGG (taxonomy mapping) [114]. In addition, tools that come along with these databases such as `PhyloBLAST` [32] or `AnnoTree` [176] have been developed to visually present search results from these databases. However, these efforts are very task-specific and may not provide the level of interactivity required.

Here, we present `BLASTphylo`, an interactive web tool for visualizing taxonomic occurrences and phylogenetic relationships of homologous proteins found by `BLAST`. The taxonomic nomenclature is taken from the NCBI Taxonomy and the sequences are derived from the respective bacterial database (nr or nt). `BLASTphylo` is highly interactive and allows the manipulation of huge taxonomic and phylogenetic trees into a well-arranged, publication-ready visualization. Since all computations are performed automatically for a given gene or protein sequence, `BLASTphylo` is also suitable for non-computationally expert users. We demonstrate the usability of `BLASTphylo` for two case studies.

12.2 Related Work

At some point during the study of proteins possibly large-scale tree structures are required to visualize the result of phylogenetic analyses or taxonomic classifications (Section 7.1.6). Here, we distinguish between pure tree visualization tools and tree analysis tools. Tree visualization tools are mainly used to show and potentially also manipulate and annotate taxonomies or phylogenies, like for example `SpaceTree` [217], `iTOL` [143], and `GraPhlAn` [8]. Tree analysis tools like `MEGA` [135], `AnnoTree` [176], and `PhyloBLAST` [32] were developed to explicitly answer specific biological questions.

Tree visualization tools usually offer a broad variety of tree visualizations, e.g. phylograms or cladograms, rooted or unrooted trees, and rectangular or radial arrangements, to provide an appropriate and compact visualization for different types of tree structures and sizes. Furthermore, some allow for the upload of complex annotations to show meta information

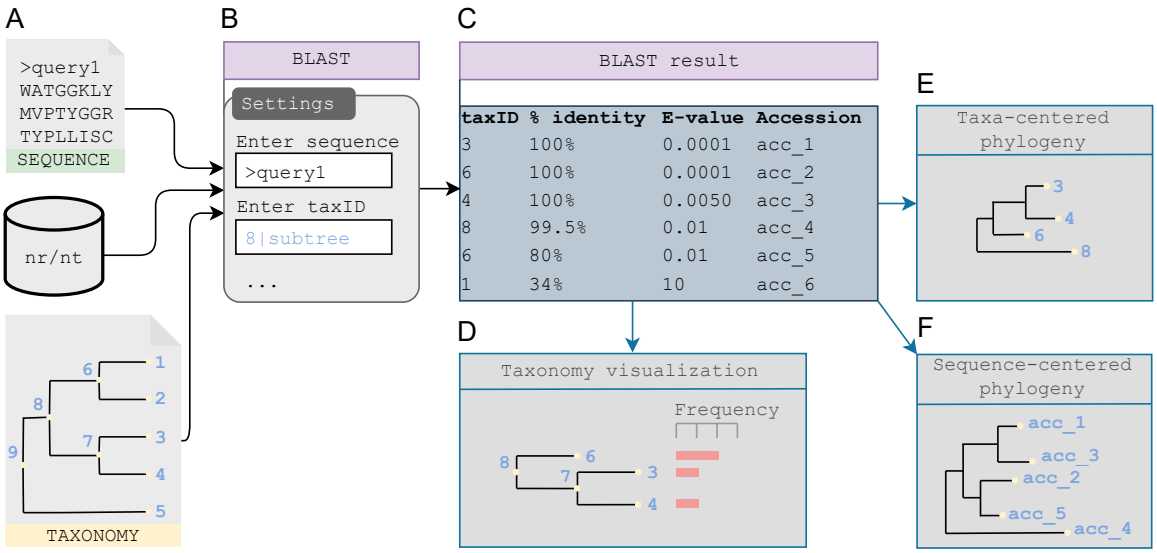


Figure 12.1: Workflow of BLASTphylo: **(A)** BLASTphylo requires a user-defined query sequence, a taxonomy specification (e.g. the NCBI taxonomy or user-defined), and a database (e.g. nr) as an input. Alternatively, a previously generated BLAST result can be used. **(B)** The BLAST search can be restricted to only parts of the taxonomy (here to the subtree rooted at tax 8). **(C)** Only significant hits are kept based on multiple metrics (here the last hit is not kept). **(D)** Next, those branches of the taxonomy that had a significant hit in the BLAST search are shown (here, taxID 1 and 2 are discarded from the taxonomy; note that taxID 7 is shown because its children had significant hits). The number of BLAST hits per leaf taxon is shown as a bar plot. In addition, a taxa **(E)**- or sequenced **(F)**-centered phylogeny is calculated and visualized.

along with the tree (e.g. GraPhlAn). On the other hand, tree analysis tools combine visualizations and annotations that help to answer an intended question. AnnoTree, for example, highlights the occurrence of KEGG orthologs, Pfam domains, TIGRFAM protein families, and taxa in the taxonomy, while PhyloBLAST focuses on calculating and statically visualizing phylogenies. As tree analysis tools target specific questions, they typically come along with search tools and databases.

In this work, we address the automatic taxonomic placement and phylogenetic reconstruction of homologs found by BLAST. The results of the taxonomic and phylogenetic analysis are visualized as interactive trees allowing views at different levels of detail. Especially, collapsing the taxonomic tree to different taxonomic ranks on demand makes BLASTphylo a unique web tool to explore a proteins' occurrence across a taxonomy.

12.3 Functionalities of BLASTphylo

In this section, we will describe the analysis pipeline and the design of BLASTphylo in detail. In general, applying BLASTphylo involves a BLAST search of a protein or DNA sequence of interest, the processing and filtering of the hits, the computation of the resulting taxonomy and phylogeny, and their interactive visualizations, all in an automated pipeline (Figure 12.1). To make it accessible to a broad community of scientists, BLASTphylo was implemented as a web application. The back-end of BLASTphylo is a Python Flask application [83], which allows seamless communication with the front-end via the *web server gateway interface* (WSGI) of Flask. All major computations and tool executions, as described in detail below, are performed in the backend. The front-

end of BLASTphylo implements the graphical user interface and all visualizations, for which D3 [2], React [3], and bootstrap [4] are used.

2: https://d3js.org/
3: https://reactjs.org/
4: https://react-bootstrap.github.io/

12.3.1 Implementation

A BLASTphylo run first starts with a BLAST search for one or two given query sequences, followed by two analysis pipelines: (i) a taxonomic mapping and (ii) a phylogenetic analysis. In this section, we will focus on the main steps and tools that form the back-end of both pipelines. Details of the front-end visualizations are described in Section 12.3.2.

BLAST

BLASTphylo offers three flavors of BLAST: BLASTP, BLASTN, and BLASTX (v2.6.0+). The user can query up to two protein or DNA sequences and can limit the BLAST search to a subset of taxa from the prokaryotic NCBI taxonomy. The user can also specify a taxonomy in Newick notation. BLAST is then executed against the nr or nt database, and subsequently hits are automatically filtered by query coverage, E-value, alignment identity, and subject coverage. It is also possible to enter a pre-computed BLAST result. In the resulting table, each hit has assigned all taxonomic IDs (taxIDs) that carry this hit. Multiple taxa will be assigned to the same hit if they all contain the exact same protein or gene, and the same taxon may be assigned to multiple hits in the case of paralogous sequences.

Taxonomic mapping

Given a list of significant BLAST subjects found for the specified taxonomy, along with their assigned taxIDs, taxonomic mapping is the process of mapping the subjects onto the taxonomy. If a subject's taxID exists in the taxonomy, a counter for that taxon is incremented. If the taxID of the subject is not present in the taxonomy, the taxIDs of the entire lineage of the subject are checked for their presence in the taxonomy. Finally, the visualized taxonomy is restricted to taxa and their respective lineages that have a count greater than 0.

Phylogenetic analysis

Since the BLAST search can lead to the mapping of multiple taxa to one subject and an assignment of several subjects to one taxon two types of phylogeny calculations may be of interest, which we call taxa-centered phylogeny and sequence-centered phylogeny. For the taxa-centered phylogeny, a representative subject is chosen for each taxon to which more than one hit was mapped. This representative sequence is selected based on a score that equally considers the E-value, percent identity, subject coverage, and query coverage. The resulting phylogenetic tree is then labeled by taxa. For the sequence-centered phylogeny, the set of all BLAST subjects is taken into account. This type of phylogeny offers valuable information about the degeneracy of the protein in the taxonomy. Protein accession numbers are used as leaf nodes.

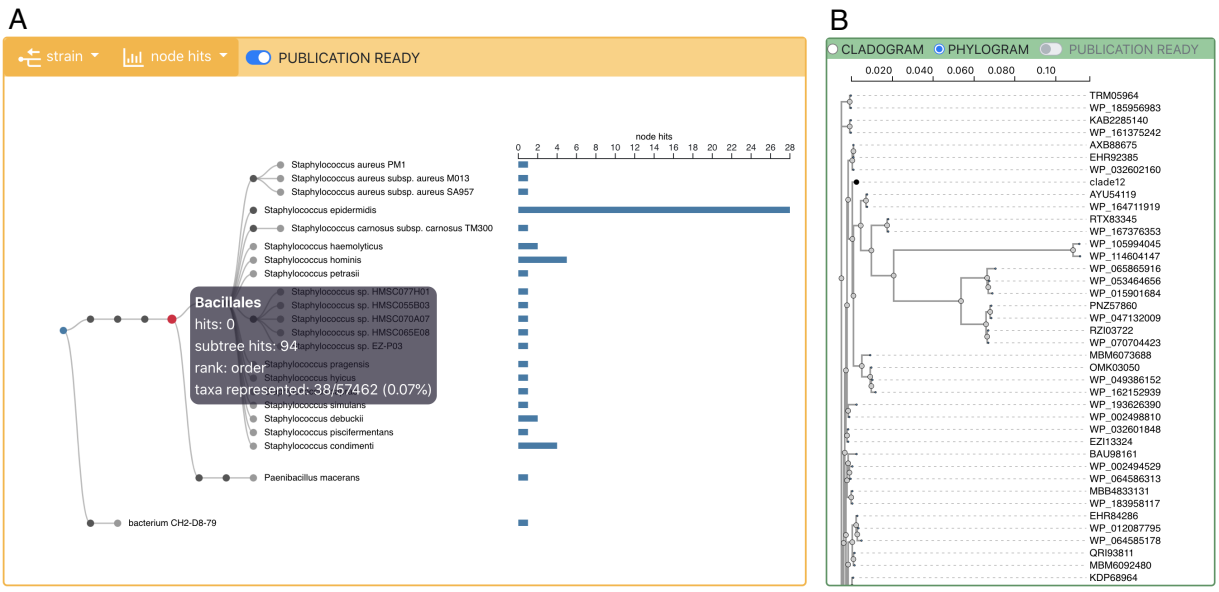


Figure 12.2: Functionalities of `BLASTphylo`: **(A)** Taxonomy visualization of taxa carrying the example protein in the bacterial kingdom. The taxonomy is expanded to strain level and the number of node hits, e.g. the number of different proteins per taxa, is indicated by an adjacent bar plot. Hovering over a node (red node) reveals more information about the taxon. Dark gray nodes indicate internal nodes and the possibility to collapse or expand the underlying subtree. Strains of the species *Staphylococcus epidermidis* have been collapsed. The checked publication ready button ensures an optimized visualization with respect to space for figure export. **(B)** Partial phylogram visualization of the union of BLAST hit sequences. Phylogenetic distances between proteins are indicated by a distance legend. Subtrees can be collapsed by the user (black node).

For both approaches, `BLASTphylo` collects the respective sequences in a FASTA file, and calculates the multiple sequence alignment using `MAFFT` (v7.310) [117]. Subsequently, the phylogenetic tree is computed with `FastTree` (v2.1.10) [224].

12.3.2 BLASTphylo frontend design

The initial page

Due to its popularity, we have designed the initial web page of `BLASTphylo` similar to the NCBI web interface [186]. The user can choose between different flavors of `BLAST`, enter the query sequence(s) in text format or as a FASTA file, or upload a pre-calculated BLAST result. The NCBI taxonomy can be restricted to a list of scientific names or taxonomic IDs. The entire subtree of a taxon will be included when the keyword *|subtree* is attached. Taxa or clades of the taxonomy can also be excluded by using for example the keyword *|!(taxID|subtree)*. A search bar helps to find scientific names or taxonomic IDs of the taxonomy. Finally, `BLAST` parameters, such as the expected threshold or the minimum alignment identity can be customized.

Tree visualizations and interactions

The core of `BLASTphylo`'s frontend is the tree visualization for the taxonomic mapping and the phylogenetic analysis. Cladograms are used to visualize taxonomies, while for the phylogenetic tree, the user can choose between a phylogram, to include branch lengths, or a cladogram,

to conveniently collapse large trees. The latter option is especially useful when the branching pattern in the tree shows one or more very densely populated clades. In BLASTphylo, cladograms are generated using the d3-package d3-flextree[5], and phylograms are generated using the d3-library Phylotree[6]. Both tree types offer the possibility to collapse inner nodes on click, enabling the user to handle large-scale trees.

5: https://github.com/Klortho/d3-flextree

6: https://github.com/veg/phylotree.js

The ability to collapse the tree to a specific taxonomic rank is the most striking feature of the taxonomy visualization (see Figure 12.2A for an example). By selecting one of the predefined taxonomic ranks, ranging from domain to strain level, all taxa below the selected rank are collapsed. This gives the user a quick and nice overview of the distribution of the protein across the taxonomy. When hovering over individual nodes, tooltips indicate the number of hits found for the respective taxon itself as well as for the entire subtree rooted at that taxon. In addition, the user can simultaneously visualize different counts associated with nodes (e.g. BLAST hits) by selecting a bar chart, which is displayed aligned to the tree. A stacked bar chart is used when two queries are used as input.

The taxa-centered phylogeny shows a phylogenetic tree of the best-scoring BLAST hit of each taxon. The user has the option to visualize the phylum level of a taxon as a heatmap attached to the tree. Furthermore, Boolean metadata can be uploaded as a single CSV file and are visualized as a presence/absence heatmap. An example of a sequence-based phylogeny is shown in Figure 12.2B. When examining such a tree, clicking on the leaf label (accession number) opens the corresponding NCBI sequence record in a new tab.

File export

Publication-ready tree visualizations can be exported as SVG elements or JPEG files. Furthermore, trees can be downloaded in the Newick format. Results obtained by taxonomic mapping can be exported as a CSV file including BLAST hit counts and taxonomic rank information for all taxa.

12.4 Showcases

In the following, we will present two case studies that we consider typical examples of the use of BLASTphylo to gain valuable insights into the taxonomic distribution and the phylogenetic relationship of protein homologs. While the first case study deals with the automated phylogenetic analysis of a single protein, the second example outlines the usefulness of BLASTphylo when studying the co-occurrence or mutual exclusion of two proteins across a taxonomy.

12.4.1 Case Study 1: Phylogenetic Analysis of Staphylococcal Lipase

When characterizing a protein, its occurrence and phylogenetic relationship to homologs of the same species or higher ranks are often of great interest. As an example, we analyzed a previously described lipase

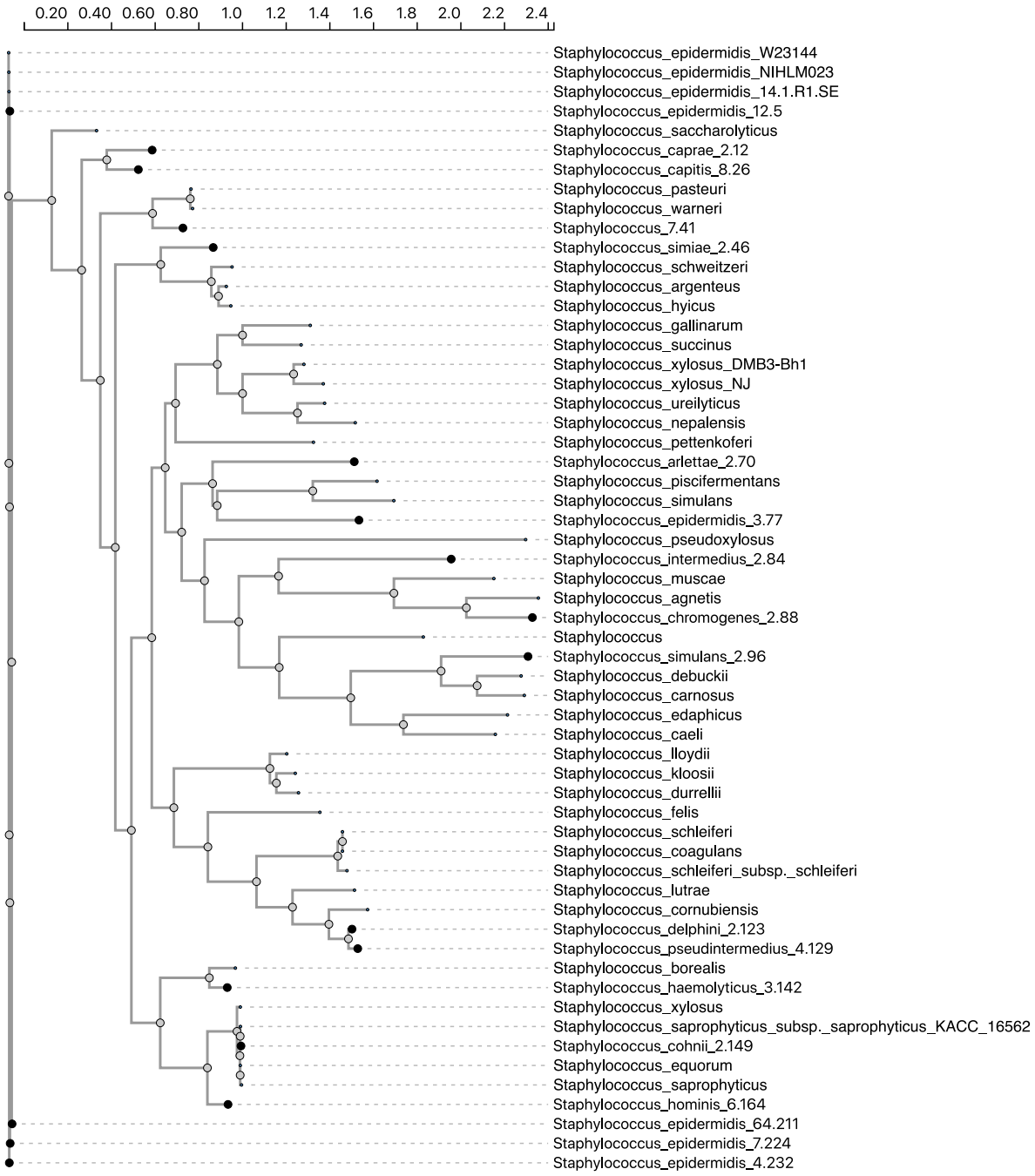


Figure 12.3: Phylogenetic tree of *gehC* homologs in the Staphylococcal genus. *S. aureus* and *unclassified staphylococcus* were excluded from the `BLASTp` search (E-value: 0.05, min. alignment identity: 0.3, min. query coverage: 0.8, min. hit coverage: 0.8). Black leaf nodes indicate a collapsed subtree at this node, which is labeled by the smallest common taxonomic ancestor of all taxa of the subtree, the number of leaf nodes and a random number to ensure unique leaf labels.

encoded by the gene *gehC* (GenBank: M95577.1) from *Staphylococcus epidermidis S2* [287] and show how `BLASTphylo` facilitates and accelerates the process of computing a phylogenetic tree. Similar to the authors of [287], we found homologs of the lipase using `BLASTp`. `BLASTphylo` automatically screens the subject hits found by `BLAST` and extracts the highest scoring hit for each strain for the taxa-based phylogeny and the union of all hits for the sequence-centered phylogeny. On-demand `BLASTphylo` then computes the multiple sequence alignment and phylogeny of the respective sequences and visualizes the phylogenetic tree (Figure 12.3).

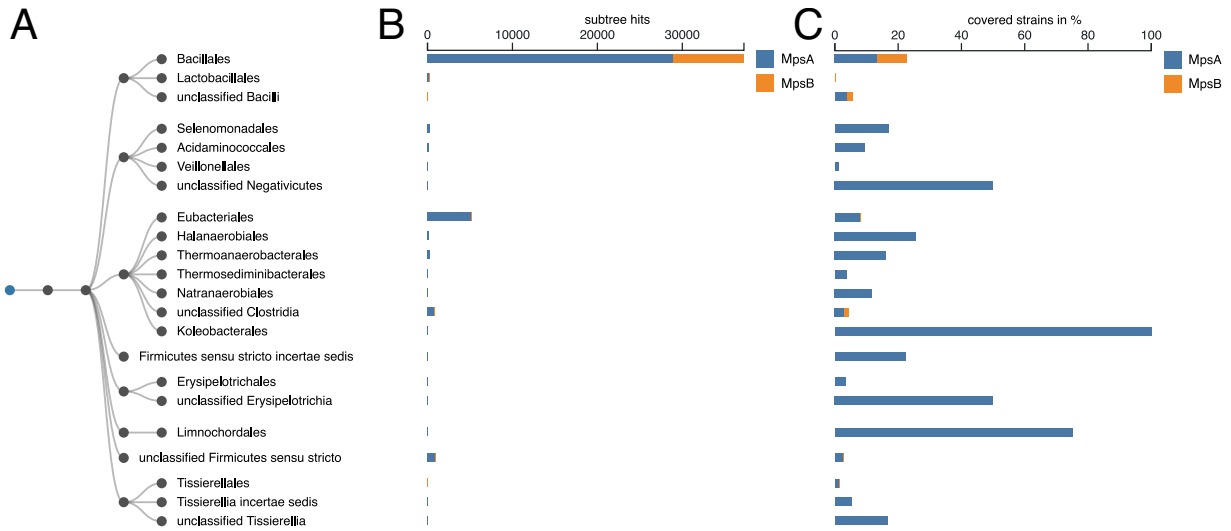


Figure 12.4: Occurrence of MpsA and MpsB across the phylum *Firmicutes* as computed by `BLASTphylo` (`BLASTp` with E-value threshold of 0.05). **(A)** Taxonomic mapping cladogram. **(B)** Barchart indicates the number of hits found within the subtree rooted at the respective taxon. **(C)** Alternative bar chart indicates the percentage of strains from the respective taxon that carry a homolog of MpsA or MpsB.

Starting from the query sequence, the use of `BLASTphylo` does not require any further intervention to obtain a visualization of the phylogeny. The resulting tree includes lipase homologs from 11 out of 12 species of the phylogeny presented in [287] with similar phylogenetic relations, but additionally includes many more species.

12.4.2 Case Study 2: Co-occurrence

CO_2 has been shown to promote bacterial growth, as several essential metabolic processes require either CO_2 or bicarbonate [77]. Due to changes in atmospheric CO_2 and O_2 levels, cytoplasmic levels of CO_2 became too low for CO_2 fixation and assimilation. Therefore, autotrophic bacteria evolved CO_2-concentrating mechanisms in which inorganic carbon transporters and carbonic anhydrases elevate the CO_2 concentration around the primary CO_2 fixing enzyme within the carboxysomes [230]. [63] recently proposed that MpsAB, a protein complex known to play a role in membrane potential generation in *S. aureus*, is also an inorganic carbon transporter, although *Firmicutes* are non-autotrophic and do not possess carboxysomes. However, MpsAB appeared to be important for bacterial growth at atmospheric CO_2 levels and was able to complement an *Escherichia coli* carbonic anhydrase (CA) mutant. Given the results described from wet-lab experiments, we would like to show how `BLASTphylo` can be used to gain additional insights when exploring a protein's function. In particular, we use `BLASTphylo` to address the following questions: How are MpsA and MpsB distributed across the taxonomy? Do both proteins always co-occur? Do organisms possess either mechanism, MpsAB or CA, and if so, is this consistent across clades in the taxonomy?

To show the distribution of the protein complex MpsAB across the phylum *Firmicutes*, the protein sequences of the genes *mpsA* (YP_498998.1) and *mpsB* (YP_498999.1) from *S. aureus* were used as an input for `BLASTphylo`. As shown in Figure 12.4A, MpsA and MpsB are widespread throughout

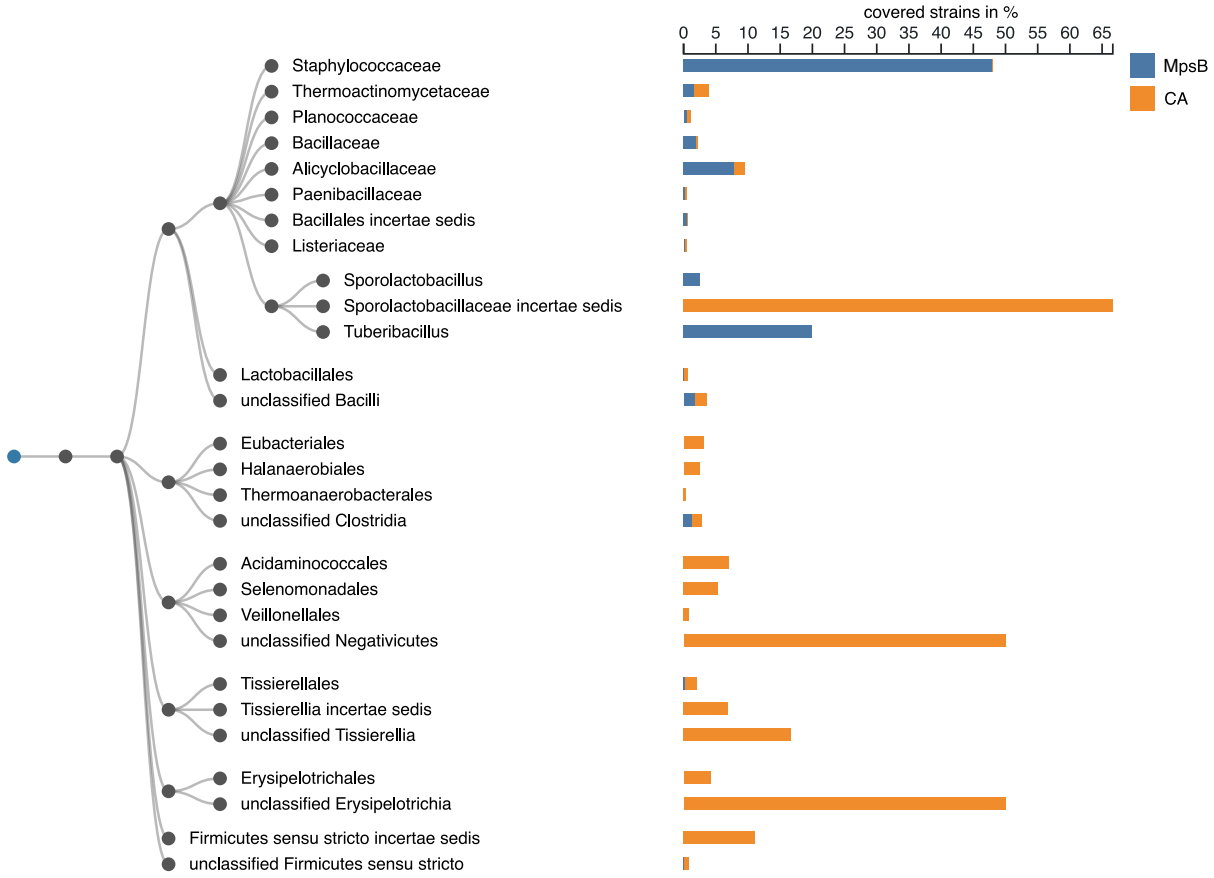


Figure 12.5: Occurrence of MpsB and CA across the phylum *Firmicutes* as computed by `BLASTphylo` (`BLASTp` with E-value threshold of 0.05). The bar chart displays the percentage of strains forming the respective taxon that carry a homolog of MpsB or CA.

the phylum *Firmicutes*. The subtree counts (Fig. Figure 12.4B) show that the order *Bacillales* (and in its subtree the *S. aureus* species) has the highest absolute counts for MpsA and MpsB. It can also be seen that more hits were found for MpsA than for MpsB across all orders. When uncollapsing the taxonomy to strain level (not shown due to the size of the tree), it becomes evident that most of the strains have multiple MpsA paralogs, but not as frequently for the MpsB gene. Furthermore, many strains do not carry an MpsB but only an MpsA homolog. As we have seen, absolute hit counts can be useful to investigate certain questions, but they can also be misleading because they are highly dependent on the number of strains stored in the database. Therefore, we provide another visualization in `BLASTphylo` to get an overall view of the distribution of hits across the taxonomy: for an internal node in the taxonomy (a higher-ranked taxon), the ratio of strains carrying a homologous protein to the total number of strains in the subtree rooted at that node is computed. For example, Figure 12.4C shows that 13% and 10% of the order *Bacillales* carry an MpsA and an MpsB homolog, respectively. Not only the co-occurrence of proteins but also the mutual exclusion of proteins can be of interest to researchers. In the work of [63], the proposed inorganic carbon transporter MpsAB from *S. aureus* was able to complement an *E. coli* carbonic anhydrase (CA) mutant and vice versa. `BLASTphylo` was used to investigate whether there is a tendency across the *Firmicutes* phylum for either of the interchangeable proteins MpsAB and CA to be present. Since the current version of `BLASTphylo` allows only two proteins

to be examined simultaneously, we decided to use MpsB to represent the MpsAB transporter, since MpsB always co-occurs with MpsA, while MpsA often also occurs alone. The taxonomic mapping reveals that at strain level, most of the time either of MpsB or CA is present and only in a few cases the respective strain carries homologs of both proteins. Furthermore, it becomes apparent that in many clades of the taxonomy, one of the proteins seems to be predominant (Figure 12.5). In particular, biofilm-forming bacteria such as *B. anthracis* or *S. aureus*, for which CO_2 diffusion is hampered, use the membrane transporter MpsAB, rather than CA.

12.5 Discussion & Conclusion

Incorporating taxonomic or phylogenetic analysis into the process of characterizing a protein is a common practice to gain insight into the evolutionary origin of the protein or to potentially infer its function. Giving users the flexibility to interactively explore taxonomic or phylogenetic trees helps discover biologically interesting results, especially when little is known about the protein. Until now, finding and filtering homologs, computing phylogenies, and visualizing taxonomic and phylogenetic trees has been time-consuming. It involved the execution of sequential programs, required manual intervention, or tools did not provide the necessary interactivity to gain biologically relevant knowledge. In this work, we present the web tool `BLASTphylo`, which allows users to automatically infer the occurrence of protein homologs across a taxonomy from a BLAST report and to study both their taxonomic distribution and their phylogenetic relationship. Restricting the taxonomy to taxa or clades of interest, automatically filtering the `BLAST` results, and having the option to (un)collapse nodes on-demand results in well-organized trees that can be exported as figures. The strength of `BLASTphylo` is its interactivity with the ability to collapse tree nodes on click, and the potential to show `BLAST` hits in the taxonomy collapsed to any taxonomic rank. In contrast to `AnnoTree`, only taxa that have a hit are displayed instead of showing the entire taxonomy and highlighting the observed hits. This greatly reduces the visual clutter when displaying low taxonomic ranks, making `BLASTphylo` uniquely capable of visualizing the taxonomy at strain-level resolution. Similar to `AnnoTree`, `BLASTphylo` provides detailed information by hovering over tree nodes and leaves indicating the number of hits found for the specific taxon and found in the subtree rooted at that taxon. Additionally, all counts per leaf node are shown as a bar plot, which is displayed aligned with the tree. `AnnoTree` and `BLASTphylo` can handle multiple queries simultaneously. However, `AnnoTree` only highlights the intersection of hits, while `BLASTphylo` visualizes the union of hits along with an indication of which of the queries were found in each taxon, providing an overview of the joint and discrete distribution of the queries across the taxonomy. As a typical use case, we showed that `BLASTphylo` can be used to study the co-occurrence or mutual exclusion of two proteins. Providing both a broad overview visualization at high order ranks, as well as the ability to explore details at strain level resolution on demand, has helped to gain interesting biological results and makes `BLASTphylo` unique among taxonomy visualizations. In its

current version, BLASTphylo allows studying two proteins simultaneously. However, more proteins may be of interest when studying larger protein complexes or multiple proteins and will therefore be included in future versions of BLASTphylo. While currently focused on prokaryotic genes, BLASTphylo will be extended to other domains such as eukaryotes. The phylogenetic tree is displayed completely by default and requires the user to collapse nodes individually as needed. This can be tedious, especially for large trees where hundreds to thousands of lineages exist for a species. Semantic aggregation techniques, e.g. collapsing a clade if all leaves belong to the same species, could be applied in the future to overcome this problem.

Part IV.

Conclusion & Future Directions

13. Discussion & Conclusion

Biological research is increasingly driven by large and complex datasets facilitated by high-throughput technologies. These high-dimensional data offer new opportunities to better understand cellular processes, organismal systems, and their interactions within ecosystems. They have also led to important advances in applied fields such as drug discovery and sustainable food technologies. Yet, the potential of these data comes with significant challenges. The volume and scale of the data can overwhelm computational resources and require scalable computational methods that not only identify patterns but also present them in ways that are both interpretable and actionable for scientists. Moreover, the inherent variability of biological data, the noise introduced by measurements, and the common issue of missing information raise concerns about the reliability of data and model outcomes. Overcoming these challenges is key to fully realizing the value of high-dimensional biological data.

This thesis contributed to meeting these challenges by developing computational methods, visualizations, and analysis pipelines aimed at enhancing the interpretability, reliability, and usability of high-dimensional biological data and the models applied to them. While the contributions span diverse topics, a unifying element across the projects is the strategic use of data reduction techniques—targeted reductions in samples or features. This approach, inspired by the "less is more" philosophy, serves as a foundational strategy to achieve computational efficiency, uncover meaningful patterns, and improve model performance. Beyond this shared emphasis on data reduction, the individual projects tackle distinct challenges, demonstrating a diversity of methodologies and applications.

The following sections highlight the key contributions of this thesis, focusing on the methods developed, their applications, and their impact on advancing the understanding and utility of high-dimensional biological data. The contributions are divided into two main parts: uncertainty propagation in dimensionality reduction and advancements in taxonomic analysis. Each part further outlines the specific challenges associated with the introduced methodologies, followed by a discussion of broader challenges in the field and potential future directions.

13.1 Uncertainty Propagation in Dimensionality Reduction

Dimensionality reduction methods, such as PCA and t-SNE, simplify high-dimensional data but fail to account for uncertainties caused by measurement errors and missing information, leaving the stability of their representations unaddressed. To bridge this gap, this thesis has introduced probabilistic frameworks and visualization approaches to quantify and interpret the impact of these uncertainties.

Since PCA and t-SNE are nonlinear with respect to their input, exact uncertainty propagation is infeasible. We addressed this using function linearization via first-order Taylor expansions. A major contribution of this thesis is the efficient implementation of uncertainty propagation using Jacobian-vector and vector-Jacobian products. This approach avoids the need to explicitly compute large Jacobian matrices, which typically arise in the biological applications addressed in this thesis, leveraging automatic differentiation to maintain computational efficiency. While PCA relies on eigenvalue decomposition, t-SNE involves an iterative optimization process that complicates gradient-based uncertainty propagation. Differentiating through the full optimization graph is computationally prohibitive. To overcome this, we employed the Implicit Function Theorem (IFT), which enables the computation of the embedding's sensitivity to input changes without backpropagating through all optimization steps, significantly improving numerical stability and feasibility. Our probabilistic framework ultimately produces a distribution over possible low-dimensional embeddings. To communicate this effectively, we have utilized implicit visualization methods such as overlay plots, small multiples, and particularly animated trajectories of projected data points, which help assess the stability of clusters and relative sample positions.

Beyond addressing uncertainties stemming from measurement noise, this thesis has also addressed missing data in PCA, particularly within the context of ancient human genomics. Ancient DNA samples are often highly degraded, leading to sparse genotype datasets. Standard PCA projection techniques, like those implemented in SmartPCA, can project these sparse samples onto a reference PC space, but the reliability of such projections is often unknown. To address this, we have developed TrustPCA, a probabilistic framework and user-friendly web tool. TrustPCA quantifies the uncertainty in PCA projections that arises directly from missing genotypes. It builds upon the established error analysis for projecting incomplete data by incorporating a model for the variance of unobserved PC scores, thereby predicting a Gaussian distribution for the true embedding location of an ancient sample. This allows researchers to visualize a confidence region around the SmartPCA projection, offering a quantitative measure of its stability based on the extent of missing data and the information content of the observed genotypes.

The development of these uncertainty-aware dimensionality reduction techniques holds significant biological relevance, particularly in an era where high-throughput experiments generate vast yet inherently noisy or incomplete datasets. Observed variations between samples or conditions can be subtle and easily confounded by experimental noise or biological stochasticity. Standard DR methods, by presenting a single, deterministic embedding, can lead to over-interpretation of transient patterns or an underestimation of ambiguity. Our contributions have directly addressed this by enabling researchers to visualize the 'confidence' associated with low-dimensional structures. This thesis has demonstrated the practical utility of such approaches across diverse biological scenarios. For instance, our methods for handling measurement uncertainty in PCA and t-SNE have shown how to assess the stability of sample groupings and identify influential features whose variability impacts embedding structures, thereby refining interpretations of, for example, time-series experiments or comparative omics studies. Furthermore, by specifically tackling miss-

ing data in ancient genomics, TrustPCA has illustrated how to quantify and visualize the substantial projection uncertainty for sparse genotype samples, therby informing the reliability of inferred genetic relationships. Collectively, these usecases underscore how the presented approaches empower researchers to distinguish robust biological patterns from those potentially confounded by measurement noise or data incompleteness. Such insights lead to more cautious and reliable biological hypothesis generation. Ultimately, by providing a clearer picture of what is truly signal versus what might be an artifact of data imperfections in reduced dimensions, these tools contribute to more rigorous and reproducible biological discovery.

13.1.1 Challenges & Future Directions

Availability of Uncertainty Information in Bioinformatics: A fundamental challenge is the availability of meaningful uncertainty estimates. In bioinformatics, such information is often not readily available or is challenging to infer. Conventional data processing pipelines focus on point estimates, thereby neglecting inherent measurement variability or uncertainties propagated from upstream analyses, such as those in sequence base-calls or read mapping positions [19, 199]. Ideally, bioinformatic workflows would systematically integrate uncertainty estimation arising from experimental errors and biases, allowing uncertainties to be propagated to downstream analyses in a statistically sound manner [136]. For instance, in genomics and transcriptomics, Bayesian methods could propagate uncertainties from raw sequencing data through preprocessing and downstream analysis, similar to the approach of [211] for microarray analyses. This is particularly valuable for noisy and sparse datasets, such as single-cell RNA-seq or ancient genomic data, where incorporating uncertainties into the analysis pipeline can enhance interpretation and guide future decision-making. Some probabilistic models have already been suggested to infer gene counts and their uncertainty estimates in single-cell RNA-seq data analyses [122, 255]. The probabilistic frameworks for dimensionality reduction proposed in this thesis could seamlessly be integrated into such workflows. For example, in single-cell analysis, where t-SNE embeddings are commonly used for clustering or trajectory inference, embedding uncertainties could be propagated through subsequent analytical steps, enriching biological insights and improving interpretability. Specifically for ancient DNA (aDNA) analyzed by TrustPCA, future work could incorporate uncertainty arising from genotyping at low coverages [190], and explore more sophisticated models of DNA degradation and damage patterns to inform the prior likelihood of observing certain alleles at missing loci [101], potentially refining the uncertainty estimates beyond simple presence/absence of data.

Violation of Gaussian Error Assumption: In this thesis, we assumed normally distributed errors to apply Gaussian error propagation. However, this assumption may be violated for some biological datasets. It is important to distinguish that the distribution of errors is not necessarily identical to the distribution of the data itself; for example, while overdispersed count data in bioinformatics—such as gene expression levels in (single-cell) RNA-seq—often follow a negative binomial distribution [156,

232] , the associated measurement errors might not. The error distribution depends on the measurement process, data generation model, and the methods used for normalization. Imposing Gaussian assumptions in such cases can lead to inaccurate uncertainty estimates. While certain non-Gaussian distributions that are closed under linear transformations [247] could extend our linear approximation framework, more general approaches like advanced Monte Carlo methods (discussed below) offer robust alternatives for handling arbitrary noise distributions.

Scalability to Large Datasets: The computational cost of uncertainty propagation in high-dimensional settings using the proposed approach remains a challenge for extremely large datasets, such as those involving millions of single-cell transcriptomes, due to runtime and memory constraints. To address this, leveraging hardware acceleration (e.g., GPUs or TPUs) could improve efficiency. Additionally, exploring alternative approximate inference methods, such as more sophisticated Monte Carlo techniques, could further alleviate the computational burden.

Comparison with Alternative Approaches: The aforementioned challenges underscore the value of exploring robust alternative inference methods. While simple Monte Carlo methods often struggle with the curse of dimensionality, advanced techniques like Markov Chain Monte Carlo (MCMC) and Hamiltonian Monte Carlo (HMC) offer strategies to efficiently sample high-dimensional target distributions by focusing on high-probability regions [164]. MCMC methods construct a Markov chain that efficiently explores the target distribution by generating samples that are more likely in high-probability regions, rather than uniformly sampling the entire space. HMC further improves efficiency by leveraging gradient information to guide the sampling trajectory, enabling faster convergence and more effective exploration of high-dimensional distributions [164]. By focusing on regions of interest and avoiding exhaustive random sampling, these methods can provide accurate uncertainty estimates while remaining computationally feasible, addressing both the limitations of Gaussian assumptions and the scalability challenges posed by large bioinformatics datasets.

For t-SNE specifically, its iterative nature complicates standard MC approaches for full uncertainty propagation. However, exploring methods like Laplace approximation [51] to directly estimate the stability of the t-SNE embedding around a local minimum of its cost function could be a fruitful research direction, provided the local landscape is sufficiently Gaussian-like.

Interpretability and Perception of Uncertainty: Effectively quantifying and visualizing uncertainty is only part of the solution; how users perceive and act upon this information is equally critical. While visualization techniques such as overlays, small multiples, and animations (as evaluated for VIPurPCA), or density plots (as used in TrustPCA) can effectively communicate uncertainty, their precise impact on scientific reasoning and decision-making remains underexplored. User studies are needed to understand how researchers interact with different uncertainty visualizations and interpret stability, structure, and relationships in uncertain embeddings, which can, in turn, inform the design of even more effective visualization techniques.

Ultimately, continued research into these areas—from improving uncertainty availability and propagation methods to understanding user perception—will solidify the role of uncertainty management in refining dimensionality reduction. This refinement is essential for extracting genuinely robust insights from complex biological data, a goal that extends to other analytical domains such as those discussed next.

13.2 Taxonomic Analysis: Advances in Classification and Visual Analytics

Next-generation sequencing has enabled comprehensive genomic studies of individual microbes and entire microbiomes, significantly advancing our understanding of microbial diversity and function. However, despite these technological advances, only a small fraction of the predicted global microbial diversity has been sequenced [1], and existing databases exhibit taxonomic biases due to uneven sequencing efforts across the microbial tree of life. These biases can influence downstream analyses. This thesis addresses two scenarios where such biases pose challenges:

1: Estimates suggest that the total number of microbial species on Earth ranges from 10^{11} to 10^{12}, while sequenced taxa are on the order of 10^5 [155].

First, for the taxonomic classification of sequences, a fundamental step in metagenomics, we investigated the performance of simple machine learning models using k-mer distributions as features. A key challenge arose from taxonomic bias in training data, where overrepresentation of certain taxa created densely populated feature space regions, skewing classifier decision boundaries and often leading to overfitting in these areas while underfitting in sparser regions. To mitigate this, we implemented a heuristic feature space balancing approach, selectively removing training examples from overly dense areas to approximate a more uniform distribution. This targeted reduction improved classifier performance and generalization.

Second, for the taxonomic and phlyogenetic analysis of genes, identifying and filtering homologous genes is a common step in characterizing gene functions across taxa. We developed `BLASTphylo`, an interactive pipeline that aids users in assessing the taxonomic distribution and phylogeny of putative homologs. Homology search tools like BLAST often return extensive, redundant lists dominated by sequences from overrepresented species. `BLASTphylo` addresses this by reducing redundancy (e.g., selecting the most significant hit per species) and providing interactive visualizations that allow users to dynamically explore taxonomic distributions by adjusting ranks and filtering, thus enhancing interpretability. In both projects, a targeted refinement or reduction of data led to improved analytical outcomes.

13.2.1 Challenges & Future Directions for Taxonomic Analysis

The advancements in taxonomic classification and visual analytics presented in this thesis open several avenues for future research, aimed at further enhancing the accuracy, scalability, and interpretability of analyzing microbial genomic data.

Microbial research is increasingly shifting its focus toward species- and even strain-level resolution, a trend mirrored in metagenomics where accurate taxonomic classification of sequencing reads at these finer ranks represents a significant current and future challenge [268]. This difficulty stems from several factors, including the inherently higher sequence conservation among closely related taxa, which complicates their differentiation. Furthermore, technical constraints, such as the short fragments produced by prevalent short-read sequencing technologies, often preclude unambiguous assignment to a specific species or strain [137]. Within the domain of machine learning-based approaches, on which this work focused, several interconnected factors influence classifier efficacy, including the composition of training data, the choice of input features, and the architecture of the classification model itself. Addressing these aspects holistically is key to advancing the field.

A primary concern is the establishment of taxonomically balanced and representative training data. Current genomic databases exhibit significant biases towards well-studied, human-associated, and pathogenic microbes. This skew distorts ML models that rely on such data for training [68, 146, 181]. As demonstrated in this thesis, merely balancing class distributions is insufficient, because underlying taxonomic relationships naturally create structural imbalances in the feature space: training examples from closely related, well-sampled taxa tend to cluster densely, whereas those from underrepresented or phylogenetically distant groups remain sparse. While the heuristic feature space balancing approach introduced in this thesis offers a pragmatic solution, it can be computationally intensive and risks discarding potentially valuable information. Although end-to-end deep learning models that learn features directly from sequences might inherently mitigate some aspects of feature-space density imbalance, a more fundamental limitation persists: many current ML classifiers treat taxonomic classes as independent entities, disregarding their inherent hierarchical structure. Future work should focus on developing models that explicitly incorporate this hierarchy, which could lead to more robust, biologically meaningful classifications and naturally address issues of imbalance related to phylogenetic density.

The choice of input features is also critical. While the k-mer counts used in this work proved effective, this manual feature engineering approach becomes computationally demanding in terms of time and memory as k increases—a necessity for achieving finer, species- or strain-level classification. Moreover, not all k-mers are equally informative, suggesting a need for sophisticated feature selection or feature learning through deep learning. Architectures such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Transformers have shown great promise in automatically extracting relevant discriminative features directly from raw DNA sequences for various genomic tasks, including taxonomic classification [69, 146, 181]. These models could enhance classification performance while reducing the burden of manual feature engineering and selection. Their typical batch-wise training regimen also offers natural points for integrating strategies to counteract class imbalance. As metagenomics increasingly incorporates long-read sequencing technologies, which produce reads of variable and often greater length but with potentially different error profiles [55, 61, 104, 219], future classifiers must adapt.

Deep learning models, with their inherent flexibility regarding input length, are well-suited for this, though their susceptibility to the specific error patterns of long reads requires further investigation.

Further innovation is also needed in the structural design of taxonomic classifiers. Current approaches often frame taxonomic assignment as a flat, single-label classification problem. However, due to sequence conservation, a read might not be unambiguously assignable to a single terminal taxon; classical methods often address this using lowest common ancestor (LCA) algorithms, e.g., [105]. Future ML models could be designed to predict labels across multiple taxonomic ranks simultaneously, enforcing consistency across the hierarchy. A significant challenge here is the representation of hierarchical data. Euclidean space, commonly used for embeddings in ML, struggles to faithfully represent tree-like structures without distortion or crowding, particularly at lower taxonomic ranks: two child nodes placed close together may appear similar in Euclidean space, but this doesn't reflect their true hierarchical distance (Figure 13.1). This is, in part, a manifestation of the curse of dimensionality, where maintaining meaningful separations in Euclidean space requires increasingly high-dimensional embeddings, leading to sparsity and inefficient representations. An interesting future research direction could be the exploration of hyperbolic spaces to model hierarchically structured data (Figure 13.2), which has recently gained attention also in the machine learning community [72, 188, 214, 284]. Hyperbolic spaces address the limitations of Euclidean space by naturally encoding hierarchical structures with lower-dimensional embeddings, leveraging their exponential volume growth to better accommodate tree-like relationships [284]. While successfully applied to other hierarchical data in bioinformatics [125], their potential for improving taxonomic classification specifically remains largely unexplored.

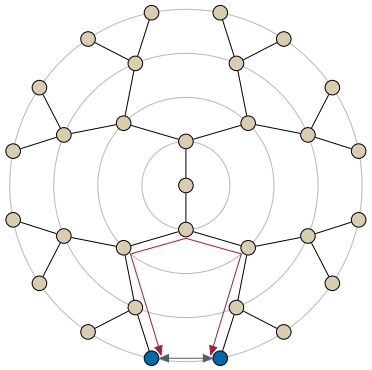


Figure 13.1: A binary tree drawn in the Euclidean space. ↔: Euclidean distance between two nodes. ↔: Desired distance between two nodes.

Beyond designing a high-performing ML model for taxonomic classification, ensuring the interpretability and explainability of classification results remains a significant challenge. Alignment-based methods like DIAMOND [34, 35] offer inherent transparency, as classifications can be traced back to specific sequence alignments. Deep learning models, in contrast, do not inherently offer a clear explanation for their predictions. Making these models more transparent—understanding why a sequence is assigned to a particular taxon—requires dedicated post-hoc explanation techniques, such as feature attribution methods (e.g., SHAP [158], integrated gradients [260]) or attention mechanism visualization [109]. Continued development of such methods is crucial for model validation and building user trust in ML-driven taxonomic assignments.

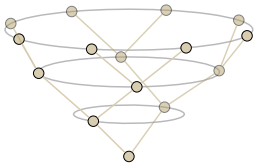


Figure 13.2: Embedding of the binary tree on a hyperboloid.

Another aspect of making taxonomic classifiers more trustworthy and useful is the explicit modeling and utilization of predictive uncertainty associated with taxonomic assignments. Current classifiers often provide a single "best-guess" taxon or, at best, a set of scores from which the top prediction is chosen. However, these scores (e.g., softmax outputs from a neural network) do not always directly translate to well-calibrated probabilities or a full representation of uncertainty. Future research could focus on developing taxonomic classifiers that not only predict a label but also provide a robust measure of confidence. This uncertainty could arise from various sources: ambiguity in the input sequence itself, regions of the feature space where multiple taxa overlap, the model's

own epistemic uncertainty due to limited training data for certain clades, or the inherent difficulty of classifying sequences from truly novel organisms. The benefits of incorporating such uncertainty measures are manifold. Firstly, it would provide a crucial quality control metric; assignments with high uncertainty could be flagged for manual review or targeted re-analysis, preventing the propagation of potentially erroneous classifications into downstream ecological or functional studies. Secondly, uncertainty estimates could inform a more nuanced interpretation of metagenomic community profiles, allowing researchers to differentiate between confidently identified taxa and those whose presence is more speculative. Finally, in the context of identifying novel organisms, high uncertainty coupled with a prediction to a broader taxonomic rank could signal the discovery of a sequence that does not neatly fit existing classifications.

Beyond the initial classification of sequences within metagenomic samples, researchers often seek to understand the broader evolutionary and functional context of specific genes or proteins identified. For instance, after performing taxonomic profiling of a microbiome and identifying key taxa or differentially abundant sequences, a common next step is to investigate particular genes of interest—perhaps those encoding enzymes for a novel metabolic pathway or virulence factors. This involves exploring how these specific genes are distributed across the wider Tree of Life, identifying homologs in diverse organisms, and reconstructing their evolutionary history to infer functional conservation, divergence, or potential horizontal gene transfer events. Such investigations bridge the gap from community-level composition to gene-centric functional and evolutionary analysis, highlighting the need for tools that facilitate this transition and the subsequent in-depth exploration of individual gene families.

Building upon `BLASTphylo`'s framework for accessible gene-centric taxonomic and phylogenetic analysis, future developments can target even more complex biological scenarios, such as the evolution of multi-gene clusters and the integration of structural homology. Bacterial proteins rarely function in isolation; they often require interactions with other proteins to perform their roles effectively. Bacterial genes encoding functionally related proteins are frequently located in close proximity on the genome and are co-regulated [228]. Future developments could thus move beyond analyzing individual query proteins to encompass entire gene clusters, such as those involved in antibiotic biosynthesis or other secondary metabolite production. Automating the identification of homologs for all genes within a defined cluster and visualizing their collective presence, absence, and conservation patterns across different taxonomic ranks could provide powerful insights into the evolution, diversity, and functional modularity of these complex systems.

Another promising direction is to expand homology searches beyond sequence-based approaches by incorporating protein structure similarity. With an increasing number of experimentally determined and computationally predicted protein structures (e.g., via AlphaFold 3 [2]), structural information is becoming an essential resource for studying protein function and evolution .Expanding homology searches beyond sequence similarity to include comparisons of three-dimensional protein structures, for instance, by querying against comprehensive resources

like the AlphaFold Protein Structure Database [275], could uncover distant evolutionary relationships and instances of structural conservation despite sequence divergence. Visualizing the taxonomic distribution of structurally similar proteins, even those with low sequence identity, could offer a more profound understanding of protein family evolution, functional adaptation, and the delineation of superfamilies.

13.3 Conclusion

This thesis has addressed key challenges in bioinformatics related to interpreting large, complex, and often uncertain biological datasets. In particular, computational methods and tools have been introduced for uncertainty-aware dimensionality reduction, taxonomic classification of sequences, and the taxonomic and phylogenetic analysis of genes. While each of these areas presents unique challenges, recurrent themes have been the strategic refinement and reduction of information, alongside the crucial role of visualization, to enhance scalability and interpretability of models and data, paving the way for continued discovery in the data-rich landscape of modern biology.

Part V.

Appendix

Additional Material for Chapter 3 A.

A.1 Derivation of the First and Second Moment of a First-order Taylor Approximation

Let $\boldsymbol{x}$ be a random vector with mean $\mathbb{E}[\boldsymbol{x}] = \boldsymbol{\mu}$. Let $\boldsymbol{f}$ be a vector-valued function differentiable at $\boldsymbol{\mu}$. The first-order Taylor series expansion of $\boldsymbol{f}(\boldsymbol{x})$ around $\boldsymbol{\mu}$ is

$$\boldsymbol{f}(\boldsymbol{x}) \approx \boldsymbol{f}(\boldsymbol{\mu}) + \boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu}),$$

where $\boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{\mu})$ is the Jacobian matrix of $\boldsymbol{f}$ evaluated at $\boldsymbol{\mu}$. For brevity, we denote it as $\boldsymbol{J}$.

We take the expectation of the approximation:

$$\begin{aligned}
\mathbb{E}[\boldsymbol{f}(\boldsymbol{x})] &\approx \mathbb{E}\left[\boldsymbol{f}(\boldsymbol{\mu}) + \boldsymbol{J}(\boldsymbol{x} - \boldsymbol{\mu})\right] \\
&= \mathbb{E}[\boldsymbol{f}(\boldsymbol{\mu})] + \mathbb{E}[\boldsymbol{J}(\boldsymbol{x} - \boldsymbol{\mu})] && \text{(Linearity of Expectation)} \\
&= \boldsymbol{f}(\boldsymbol{\mu}) + \boldsymbol{J}\,\mathbb{E}[\boldsymbol{x} - \boldsymbol{\mu}] && (\boldsymbol{f}(\boldsymbol{\mu}) \text{ and } \boldsymbol{J} \text{ are constant}) \\
&= \boldsymbol{f}(\boldsymbol{\mu}) + \boldsymbol{J}(\mathbb{E}[\boldsymbol{x}] - \mathbb{E}[\boldsymbol{\mu}]) && \text{(Linearity of Expectation)} \\
&= \boldsymbol{f}(\boldsymbol{\mu}) + \boldsymbol{J}(\boldsymbol{\mu} - \boldsymbol{\mu}) && \text{(By definition of } \boldsymbol{\mu}) \\
&= \boldsymbol{f}(\boldsymbol{\mu}) + \boldsymbol{J} \cdot \boldsymbol{0} \\
&= \boldsymbol{f}(\boldsymbol{\mu})
\end{aligned}$$

Thus, to a first-order approximation, the expected value is $\mathbb{E}[\boldsymbol{f}(\boldsymbol{x})] \approx \boldsymbol{f}(\boldsymbol{\mu})$.

Let $\boldsymbol{x}$ be a random vector with mean $\boldsymbol{\mu}$ and covariance matrix $\mathbb{V}(\boldsymbol{x}) = \boldsymbol{\Sigma}_{\boldsymbol{x}}$. The covariance of the vector function $\boldsymbol{f}(\boldsymbol{x})$ is defined as:

$$\mathbb{V}[\boldsymbol{f}(\boldsymbol{x})] = \mathbb{E}\left[(\boldsymbol{f}(\boldsymbol{x}) - \mathbb{E}[\boldsymbol{f}(\boldsymbol{x})])(\boldsymbol{f}(\boldsymbol{x}) - \mathbb{E}[\boldsymbol{f}(\boldsymbol{x})])^T\right]$$

From our previous result, we have the first-order approximation for the deviation term:

$$\boldsymbol{f}(\boldsymbol{x}) - \mathbb{E}[\boldsymbol{f}(\boldsymbol{x})] \approx \left(\boldsymbol{f}(\boldsymbol{\mu}) + \boldsymbol{J}(\boldsymbol{x} - \boldsymbol{\mu})\right) - \boldsymbol{f}(\boldsymbol{\mu}) = \boldsymbol{J}(\boldsymbol{x} - \boldsymbol{\mu})$$

Substituting this into the covariance definition:

$$\begin{aligned}
\mathbb{V}[\boldsymbol{f}(\boldsymbol{x})] &\approx \mathbb{E}\left[\left(\boldsymbol{J}(\boldsymbol{x} - \boldsymbol{\mu})\right)\left(\boldsymbol{J}(\boldsymbol{x} - \boldsymbol{\mu})\right)^T\right] \\
&= \mathbb{E}\left[\boldsymbol{J}(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^T\boldsymbol{J}^T\right] && \text{(Transpose of a product)} \\
&= \boldsymbol{J}\,\mathbb{E}\left[(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^T\right]\boldsymbol{J}^T && (\boldsymbol{J} \text{ is a constant matrix}) \\
&= \boldsymbol{J}\,\boldsymbol{\Sigma}_{\boldsymbol{x}}\,\boldsymbol{J}^T && \text{(By definition, } \boldsymbol{\Sigma}_{\boldsymbol{x}} = \mathbb{E}[(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^T])
\end{aligned}$$

Thus, the first-order approximation for the covariance matrix is $\mathbb{V}[\boldsymbol{f}(\boldsymbol{x})] \approx \boldsymbol{J}\boldsymbol{\Sigma}_{\boldsymbol{x}}\boldsymbol{J}^T$. This is the formula for linear propagation of uncertainty.

Additional Material for Chapter 8 | B.

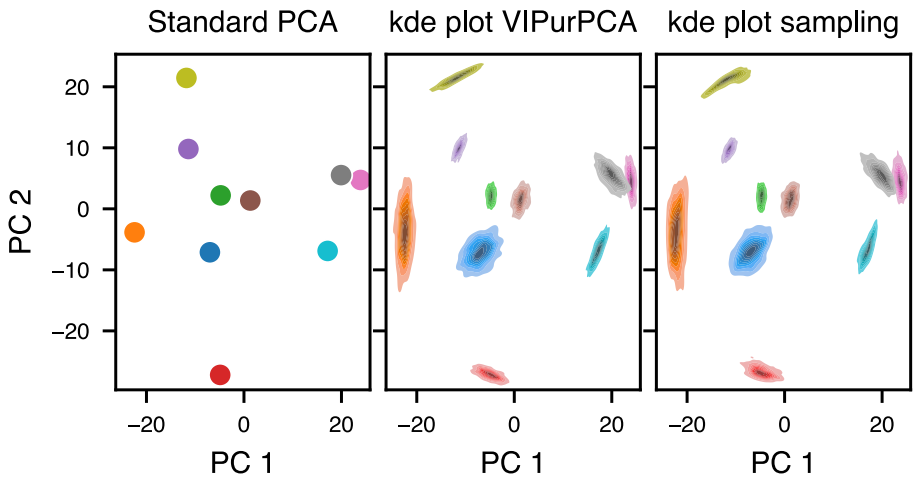


Figure B.1: Comparison of `VIPurPCA` with Monte-Carlo sampling. The left plot represents the lower dimensional map obtained by applying standard PCA to the mean of the input distribution. The middle plot shows a kde (kernel density estimation) plot for 500 random samples of eigenvectors drawn from the distribution computed by `VIPurPCA`. The right plot shows a kde plot from 500 iterations of Monte-Carlo sampling from the input distribution.

Table B.1: Representation of student test results using precise, interval, or qualitative data, mapped to trapezoidal distributions to model the associated uncertainties. **(a)** The test results of six students [54] are provided as real numbers, intervals or qualitative statements. **(b)** Mapping from linguistic labels to trapezoidal distributions, which are defined by four parameters. **(c)** Uncertainties associated with student grades.

(a) Student Grades

	M1	M2	P1	P2
Tom	15	fairly good	$\mathcal{N}(14, 5.7^2)$	[14, 16]
David	9	good	fairly good	10
Bob	6	[10, 11]	[13, 20]	good
Jane	fairly good	very good	19	[10, 12]
Joe	very bad	fairly bad	[10, 14]	[14]
Jack	1	[4, 6]	9	[6, 9]

(b) Mapping

Linguistic label	Trapezoidal distribution
very bad	$T(0, 0, 2, 6)$
bad	$T(2, 4, 6, 7)$
fairly bad	$T(5, 7, 9, 10)$
fairly good	$T(10, 11, 13, 14)$
good	$T(13, 14, 16, 18)$
very good	$T(14, 18, 20, 20)$

(c) Means and Variances

	M1		M2		P1		P2	
	μ	σ^2	μ	σ^2	μ	σ^2	μ	σ^2
Tom	15	0.1	12.29	1.23	14.1	33.33	15	0.33
David	9	0.1	15.29	1.23	12.29	1.23	10	0.1
Bob	6	0.1	10.5	0.08	16.5	4.08	15.3	1.23
Jane	12.29	1.23	17.8	1.97	19	0.1	11	0.33
Joe	2.17	1.97	7.71	1.23	12	1.33	14	0.1
Jack	1	0.1	5	0.33	9	0.1	7.5	0.75

Additional Material for Chapter 9

C.

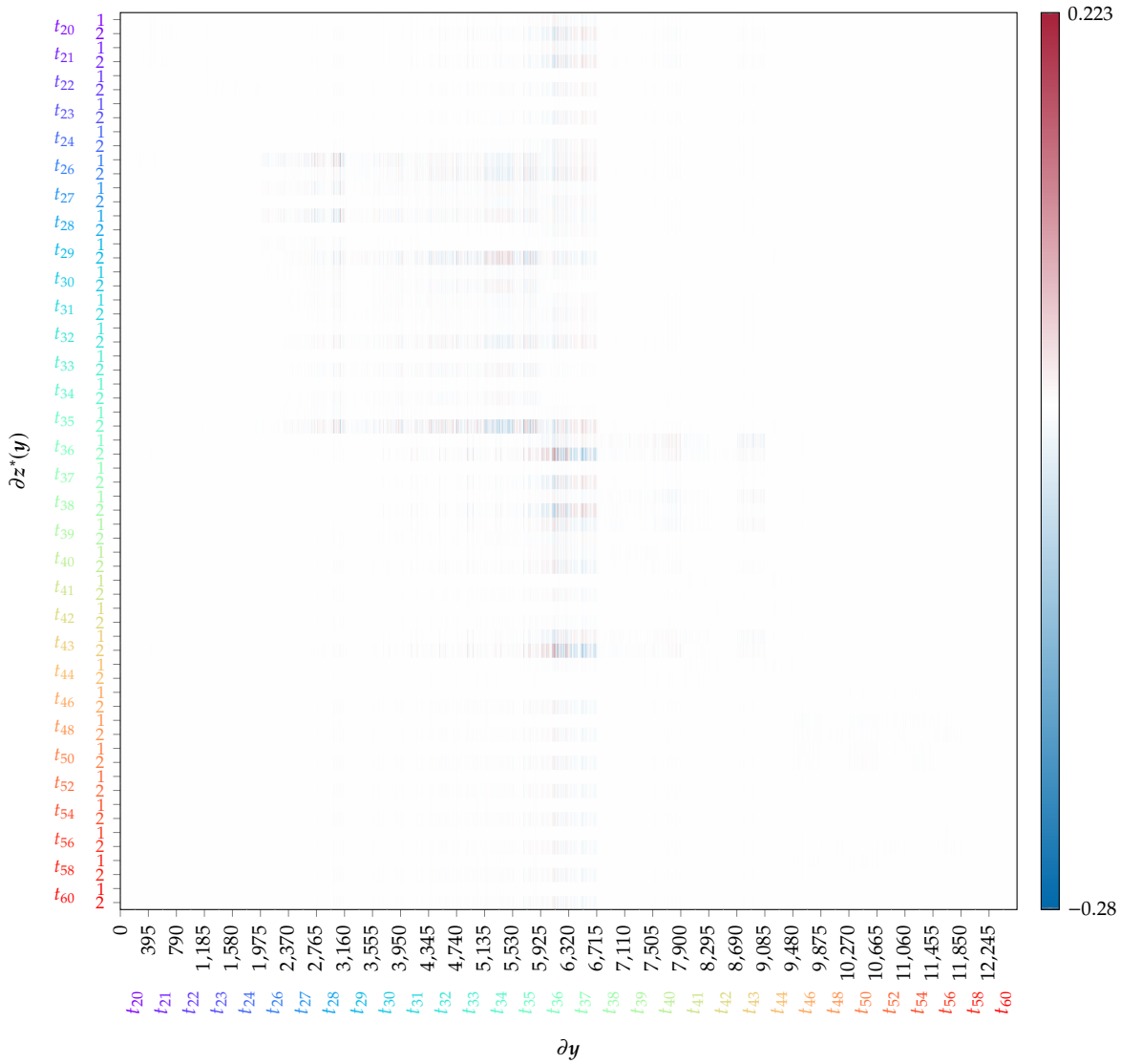


Figure C.1: Heatmap of the Jacobian J_{z^*}. Rows correspond to the $32 \times 2 = 64$ output embedding coordinates $(z_1(t_1), z_2(t_1), \ldots, z_2(t_{60}))$. Columns correspond to the $32 \times 395 = 12,640$ input features (expression of gene 1 at t_1, gene 2 at $t_1, \ldots$, gene 395 at t_{60}).

Additional Material for Chapter 10 D.

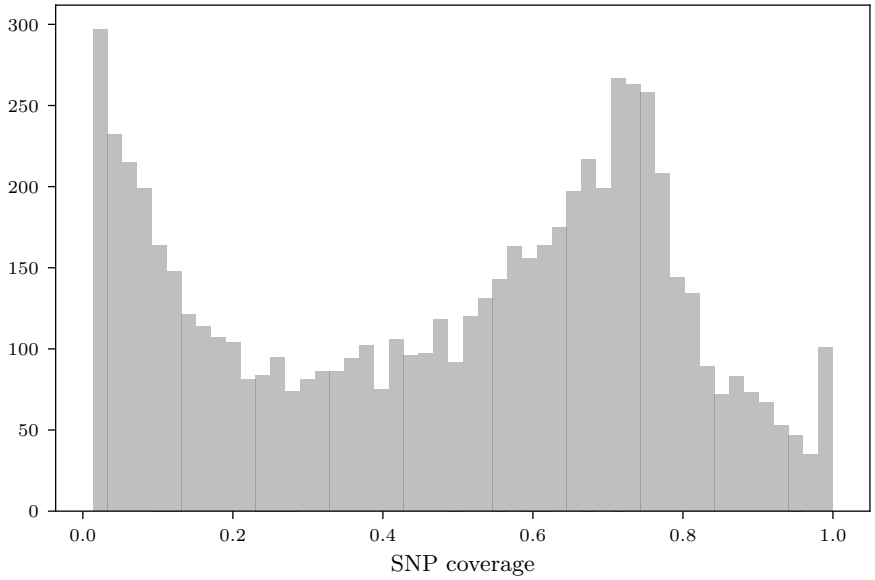


Figure D.1: Histogram of SNP coverage across the 6,627 West Eurasian human individuals from the Allen Ancient DNA Resource included in this study. SNP coverage is defined as the proportion of successfully genotyped single nucleotide polymorphisms (SNPs) per individual, relative to the total of 540,247 SNPs analyzed in this work.

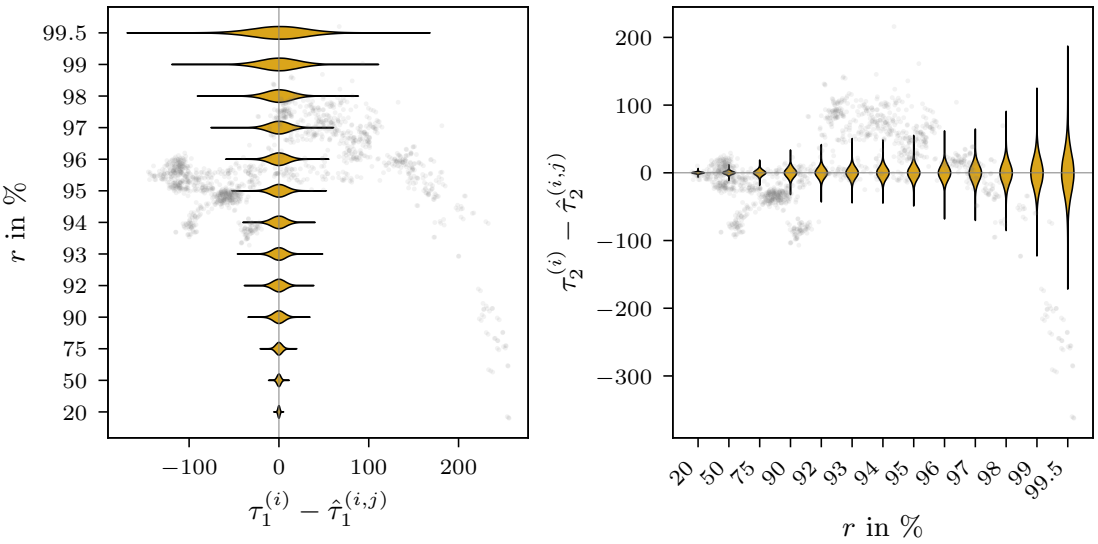


Figure D.2: Violin plots of discrepancy distributions in PC1 and PC2 at varying downsampling rates (r). To assess the impact of missing loci, loci were randomly omitted from 15 high-coverage ancient samples at progressively increasing rates (r). For each sample i, this process was repeated 20,000 times. The discrepancies between the resulting SmartPCA projections $\{\hat{\boldsymbol{\tau}}^{(i,j)}\}_{j=1,\ldots,20000}$ and their respective reference projection $\boldsymbol{\tau}^{(i)}$ were computed. The component-wise distributions of these discrepancies are visualized as violin plots, illustrating how the spread of discrepancies in PC1 and PC2 changes with increasing downsampling rates. To provide a reference scale for the magnitude of $\boldsymbol{\tau}^{(i)} - \hat{\boldsymbol{\tau}}^{(i,j)}$, the map of modern PCA projections is shown in the background. Note that only the corresponding axis (left: PC1, right: PC2) is meaningful for interpreting the spread of discrepancies.

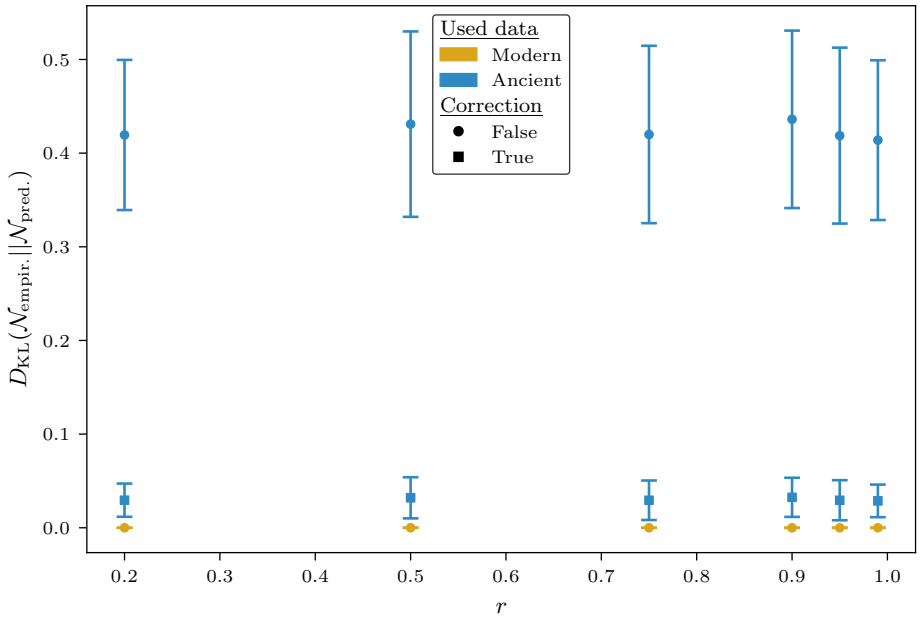


Figure D.3: Kullback-Leibler (KL) divergence between empirical and predicted Gaussian discrepancy distributions. High-coverage modern and ancient samples were analyzed by randomly omitting loci at a specified rate r. For each sample, the discrepancy in the first two principal components between the resulting SmartPCA embeddings and the reference embeddings was computed. From these discrepancies, the empirical Gaussian distribution was estimated. Using the same sets of omitted loci, the Gaussian discrepancy distribution was predicted based on Equation (10.8)-Equation (10.12). The KL divergence between the empirical and predicted distributions was then calculated. This process was repeated for 100 random draws of missing loci sets, with the mean KL divergence and standard deviation plotted to summarize the results. Marker types indicate whether a variance adjustment was applied during the prediction of the distribution.

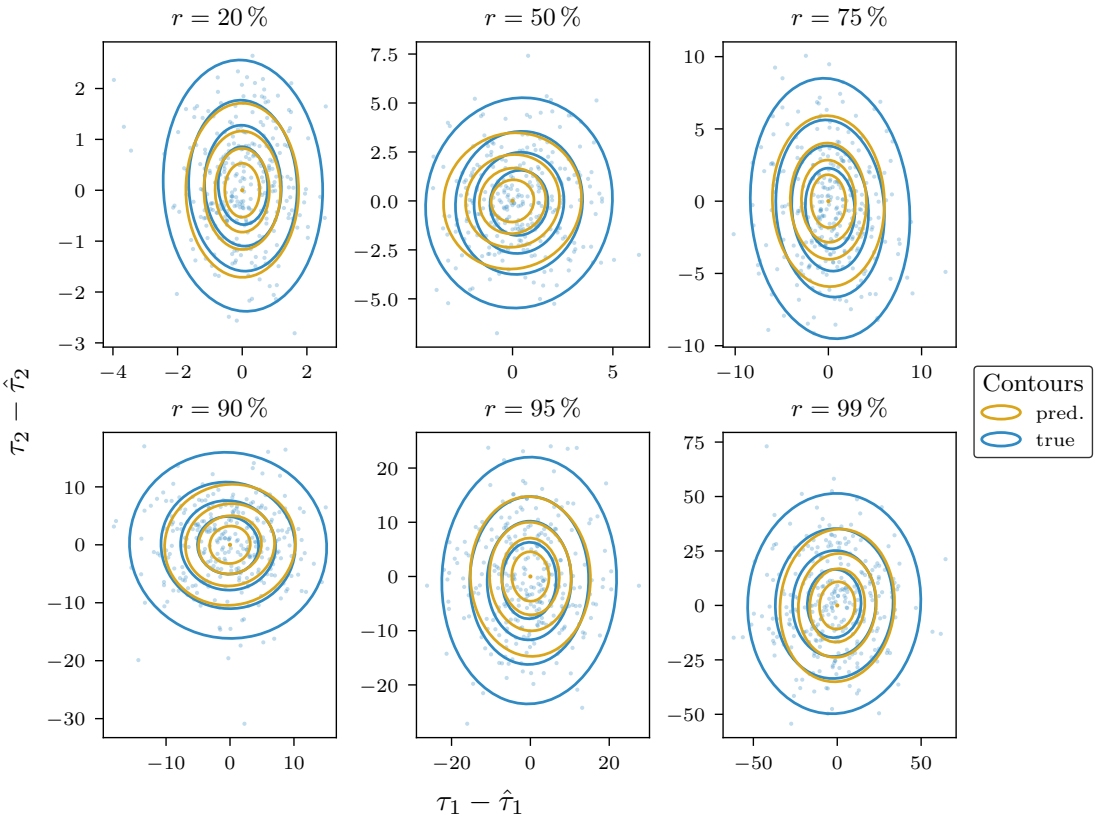


Figure D.4: Comparison of empirically determined and predicted Gaussian discrepancy distributions for ancient genotype subset embeddings. For each of 304 high-coverage ancient West Eurasian genotype samples, a random set of loci was omitted at the specified downsampling rate r. The discrepancy in the first two principal components between the resulting SmartPCA subset embeddings and the reference embeddings was calculated and plotted as scatter points. From this data, the empirical Gaussian distribution was estimated and represented by blue contours at the specified quantiles. Using the corresponding sets of omitted loci, the Gaussian discrepancy distribution was predicted based on Equation (10.8)-Equation (10.12), with the predicted contours shown in gold overlaying the blue contours. These predictions were made without adjustments for the increased variance typically observed in ancient samples compared to modern samples.

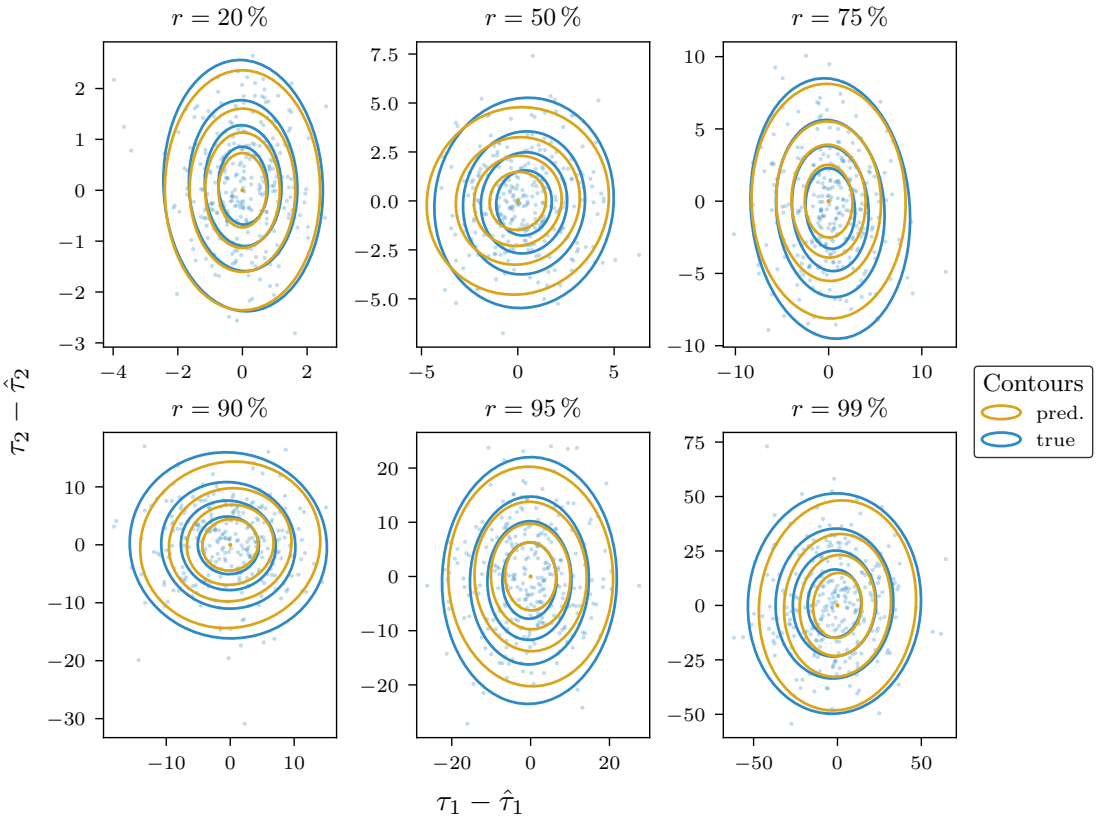


Figure D.5: Comparison of empirically determined and predicted Gaussian discrepancy distributions for ancient genotype subset embeddings. For each of 304 high-coverage ancient West Eurasian genotype samples, a random set of loci was omitted at the specified downsampling rate r. The discrepancy in the first two principal components between the resulting SmartPCA subset embeddings and the reference embeddings was calculated and plotted as scatter points. From this data, the empirical Gaussian distribution was estimated and represented by blue contours at the specified quantiles. Using the corresponding sets of omitted loci, the Gaussian discrepancy distribution was predicted based on Equation (10.8)-Equation (10.12), with the predicted contours shown in gold overlaying the blue contours. Variance adjustments were incorporated into the prediction framework to account for the increased variance typically observed in ancient samples compared to modern ones.

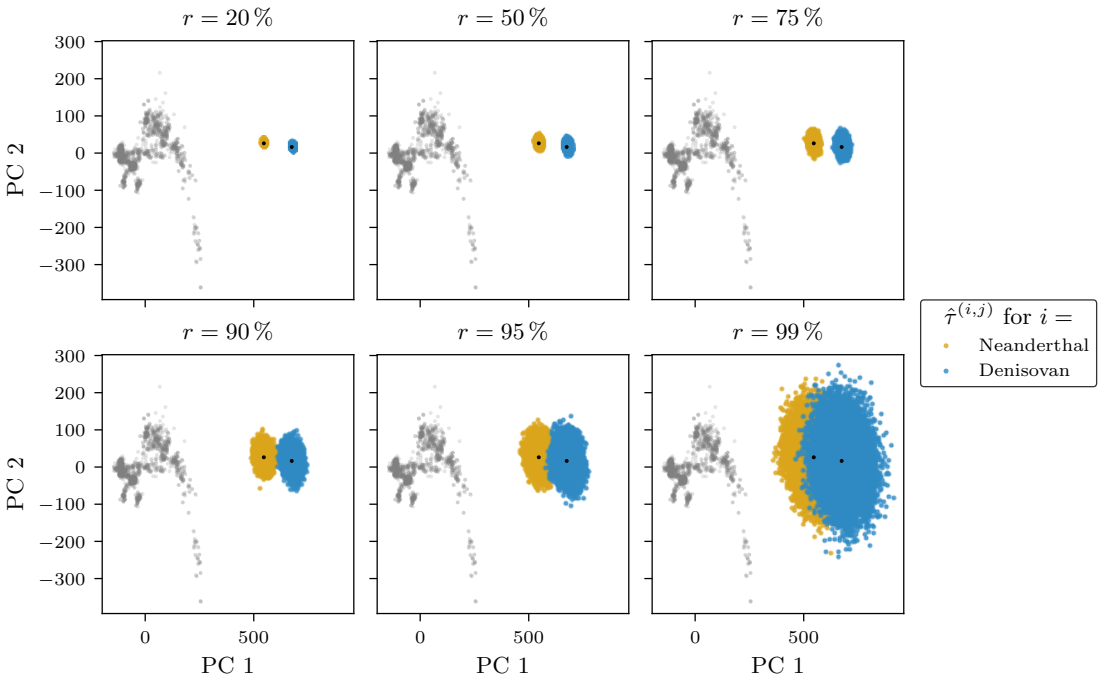


Figure D.6: Impact of missing loci on SmartPCA projections of out-of-distribution ancient genotype samples. The projections of the modern West Eurasian samples, used to compute the principal component (PC) subspace, are shown in gray. The black point represents the projection of the high-coverage ancient individuals (Neanderthal & Denisovan), considered as the *true* projection $\{\boldsymbol{\tau}^{(i)}\}$. To evaluate the impact of missing loci, loci were randomly omitted from the ancient sample at an increasing rate r. This process was repeated 20,000 times. The resulting SmartPCA projections, $\{\hat{\boldsymbol{\tau}}^{(i,j)}\}_{j=1,\ldots,20000}$, are shown in orange and blue for the respective individuals.

Additional Material for Chapter 11 | E.

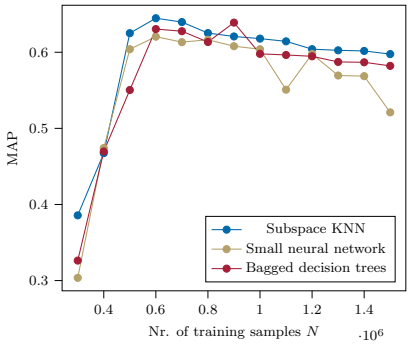


Figure E.1: Performance comparison in terms of MAP for different training set sizes and different classifiers. Performance values for phylum level classification of the distantly related dataset are shown. A grid size G of 10, and k-mer size of 3 was used. The parameter N describes the number of training samples included by the balancing algorithm.

Table E.1: Performance evaluation of a classifier (ensemble of bagged decision trees) trained on unbalanced or balanced training data of the distantly related dataset. Relative k-mer frequencies were used as features. The second column shows the respective choice of k. Classification was performed at the superkingdom and phylum level for different grid sizes G, and performance values are given as MAP. The best results appear in bold.

Classification on	k	No balancing	Balancing with				
			$G = 10$	$G = 20$	$G = 30$	$G = 50$	$G = 100$
superkingdom level	1	0.3718	0.7658	0.7661	0.7671	0.7668	0.7667
	2	0.6774	0.7916	0.7909	0.7925	0.7914	0.7902
	3	0.7433	0.8164	**0.8171**	0.8153	0.8149	0.8160
	4	0.7252	0.7911	0.7881	0.7896	0.7912	0.7877
	5	0.6496	0.7693	0.7069	0.7439	0.7478	0.7693
phylum level	1	0.1267	0.2834	0.2840	0.2843	0.2849	0.2835
	2	0.4362	0.5829	0.5783	0.5816	0.5835	0.5828
	3	0.5397	**0.6971**	0.6910	0.6928	0.6964	0.6936
	4	0.4982	0.6562	0.6282	0.6582	0.6605	0.6452
	5	0.3242	0.5728	0.6036	0.6134	0.5708	0.5372

Table E.2: Performance comparison in terms of MAP of several pipelines implementing our approach to state-of-the-art methods. Note that performance values for state-of-the-art methods were taken from [181]. Different combinations of the proposed approach are shown. The best results appear in bold.

		Distantly related		Final model		
	Methods	Superk.	Phylum	Superk.	Phylum	Genus
State-of-the-art (numbers taken from [181])	MMseqs2	0.6276	0.4136	0.9694	0.9290	0.7476
	MMseqs2 tax.	0.6747	0.4344	0.9811	0.9347	0.7509
	DeepMicrobes	0.6725	0.3661	0.9813	0.9211	0.6643
	BERTax	**0.9006**	0.5410	0.9862	0.9510	0.6692
Proposed	Subspace KNN, 3-mer	0.8823	0.6756	0.9828	0.9316	0.8108
	Subspace KNN, 4-mer	0.8803	0.6577	**0.9907**	**0.9553**	**0.8643**
	Bagged decision trees, 3-mer	0.8164	**0.6971**	0.9251	0.8517	0.7610
	Bagged decision trees, 4-mer	0.7911	0.6562	0.9198	0.8513	0.6911
	Chance level	0.25	0.0333	0.25	0.0227	0.0064

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems". 2015. URL: https://www.tensorflow.org/.

[2] J. Abramson, J. Adler, J. Dunger, R. Evans, T. Green, A. Pritzel, O. Ronneberger, L. Willmore, A. J. Ballard, J. Bambrick, et al. "Accurate structure prediction of biomolecular interactions with AlphaFold 3". *Nature* 630.8016 (2024), pages 493–500.

[3] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. "On the surprising behavior of distance metrics in high dimensional space". *Database Theory—ICDT 2001: 8th International Conference London, UK, January 4–6, 2001 Proceedings 8*. Springer. 2001, pages 420–434.

[4] Z. Allen-Zhu and Y. Li. "LazySVD: Even faster SVD decomposition yet without agonizing pain". *Advances in neural information processing systems* 29 (2016).

[5] M. E. Allentoft, M. Sikora, A. Refoyo-Martínez, E. K. Irving-Pease, A. Fischer, W. Barrie, A. Ingason, J. Stenderup, K.-G. Sjögren, A. Pearson, et al. "Population genomics of post-glacial western Eurasia". *Nature* 625.7994 (2024), pages 301–311.

[6] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. "Basic local alignment search tool". *Journal of molecular biology* 215.3 (1990), pages 403–410.

[7] I. Araf, A. Idri, and I. Chairi. "Cost-sensitive learning for imbalanced medical data: a review". *Artificial Intelligence Review* 57.4 (2024), page 80.

[8] F. Asnicar, G. Weingart, T. L. Tickle, C. Huttenhower, and N. Segata. "Compact graphical representation of phylogenetic data and metadata with GraPhlAn". *PeerJ* 3 (2015), e1029.

[9] A. G. Atanasov, S. B. Zotchev, V. M. Dirsch, and C. T. Supuran. "Natural products in drug discovery: advances and opportunities". *Nature Reviews Drug Discovery* 20.3 (2021), pages 200–216.

[10] T. Athawale and A. Entezari. "Uncertainty quantification in linear interpolation for isosurface extraction". *IEEE Transactions on Visualization and Computer Graphics* 19.12 (2013), pages 2723–2732.

[11] T. M. Athawale, B. Ma, E. Sakhaee, C. R. Johnson, and A. Entezari. "Direct volume rendering with nonparametric models of uncertainty". *IEEE Transactions on Visualization and Computer Graphics* 27.2 (2020), pages 1797–1807.

[12] S. W. Attwood, S. C. Hill, D. M. Aanensen, T. R. Connor, and O. G. Pybus. "Phylogenetic and phylodynamic approaches to understanding and combating the early SARS-CoV-2 pandemic". *Nature Reviews Genetics* 23.9 (2022), pages 547–562.

[13] K. Ausmees, F. Sanchez-Quinto, M. Jakobsson, and C. Nettelblad. "An empirical evaluation of genotype imputation of ancient DNA". *G3 Genes | Genomes | Genetics* 12.6 (2022), jkac089.

[14] A. Barlow, S. Hartmann, J. Gonzalez, M. Hofreiter, and J. L. A. Paijmans. "Consensify: A Method for Generating Pseudohaploid Genome Sequences from Palaeogenomic Datasets with Reduced Error Rates". *Genes (Basel)* 11.1 (2020), page 50.

[15] J. Barnes and P. Hut. "A hierarchical $\mathcal{O}(N \log N)$ force-calculation algorithm". *Nature* 324.6096 (1986), pages 446–449.

[16] F. L. Bauer. "Computational graphs and rounding error". *SIAM Journal on Numerical Analysis* 11.1 (1974), pages 87–96.

[17] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. "Automatic differentiation in machine learning: a survey". *Journal of Marchine Learning Research* 18 (2018), pages 1–43.

[18] T. Bayes. "LII. An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, FRS communicated by Mr. Price, in a letter to John Canton, AMFR S". *Philosophical transactions of the Royal Society of London* 53 (1763), pages 370–418.

[19] D. Becker, D. Champredon, C. Chato, G. Gugan, and A. Poon. "SUP: a probabilistic framework to propagate genome sequence uncertainty, with applications". *NAR Genomics and Bioinformatics* 5.2 (2023), lqad038.

[20] A. C. Belkina, C. O. Ciccolella, R. Anno, R. Halpert, J. Spidlen, and J. E. Snyder-Cappione. "Automated optimized parameters for t-distributed stochastic neighbor embedding improve visualization and analysis of large datasets". *Nature communications* 10.1 (2019), page 5415.

[21] R. Bellman. "Adaptive Control Processes: A Guided Tour". *Princeton University Press*, 1961.

[22] D. A. Benson, M. Cavanaugh, K. Clark, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and E. W. Sayers. "GenBank". *Nucleic acids research* 41.D1 (2012), pages D36–D42.

[23] M. Betancourt. "A geometric theory of higher-order automatic differentiation". *arXiv preprint arXiv:1812.11592* (2018).

[24] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. "When is "nearest neighbor" meaningful?" *Database Theory—ICDT'99: 7th International Conference Jerusalem, Israel, January 10–12, 1999 Proceedings 7*. Springer. 1999, pages 217–235.

[25] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. "Variational inference: A review for statisticians". *Journal of the American Statistical Association* 112.518 (2017), pages 859–877.

[26] V. Bolón-Canedo, B. Remeseiro, K. Sechidis, D. Martinez-Rego, and A. Alonso-Betanzos. "Algorithmic challenges in big data analytics." *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*. 2017.

[27] G.-P. Bonneau, H.-C. Hege, C. R. Johnson, M. M. Oliveira, K. Potter, P. Rheingans, and T. Schultz. "Overview and state-of-the-art of uncertainty visualization". *Scientific Visualization: Uncertainty, Multifield, Biomedical, and Scalable Visualization* (2014), pages 3–27.

[28] B. E. Boser, I. M. Guyon, and V. N. Vapnik. "A training algorithm for optimal margin classifiers". *Proceedings of the fifth annual workshop on Computational learning theory*. 1992, pages 144–152.

[29] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. 2018.

[30] A. M. Breipohl. "Probabilistic Systems Analysis: An Introduction to Probabilistic Models, Decisions, and Applications of Random Processes". *John Wiley & Sons*, 1990.

[31] L. Brieman, J. Friedman, C. J. Stone, and R. Olshen. "Classification and regression tree analysis". *CRC Press: Boca Raton, FL, USA*, 1984.

[32] F. S. Brinkman, I. Wan, R. E. Hancock, A. M. Rose, and S. J. Jones. "PhyloBLAST: facilitating phylogenetic analysis of BLAST results". *Bioinformatics* 17.4 (2001), pages 385–387.

[33] B. L. Browning, Y. Zhou, and S. R. Browning. "A one-penny imputed genome from next-generation reference panels". *The American Journal of Human Genetics* 103.3 (2018), pages 338–348.

[34] B. Buchfink, K. Reuter, and H.-G. Drost. "Sensitive protein alignments at tree-of-life scale using DIAMOND". *Nature methods* 18.4 (2021), pages 366–368.

[35] B. Buchfink, C. Xie, and D. H. Huson. "Fast and sensitive protein alignment using DIAMOND". *Nature methods* 12.1 (2015), pages 59–60.

[36] D. Bzdok, N. Altman, and M. Krzywinski. "Statistics versus machine learning". *Nature methods* 15.4 (2018), page 233.

[37] A.-L. Cauchy. "Résumé d'un mémoir sur la mécanique céleste et sur un nouveau calcul appelé calcul des limites". *Oeuvres Complétes d'Augustun Cauchy* 12.48-112 (1831), page 3.

[38] A.-L. Cauchy. "Méthode générale pour la résolution des systemes d'équations simultanées". *Comptes rendus de l'Académie des sciences Paris* 25.1847 (1847), pages 536–538.

[39] L. L. Cavalli-Sforza and A. Piazza. "Analysis of evolution: evolutionary rates, independence and treeness". *Theoretical Population Biology* 8.2 (1975), pages 127–165.

[40] Y.-H. Chan, C. D. Correa, and K.-L. Ma. "Flow-based scatterplots for sensitivity analysis". *2010 IEEE Symposium on Visual Analytics Science and Technology*. IEEE. 2010, pages 43–50.

[41] Y.-H. Chan, C. D. Correa, and K.-L. Ma. "The generalized sensitivity scatterplot". *IEEE transactions on visualization and computer graphics* 19.10 (2013), pages 1768–1781.

[42] J. Chung and S. Wark. *Visualising uncertainty for decision support*. Technical report. Defence Science and Technology Group Fishermans Bend Victoria Australia, 2016.

[43] J. E. Clarridge III. "Impact of 16S rRNA gene sequence analysis for identification of bacteria on clinical microbiology and infectious diseases". *Clinical microbiology reviews* 17.4 (2004), pages 840–862.

[44] I. H. G. S. Consortium. "Initial sequencing and analysis of the human genome". *Nature* 409.6822 (2001), pages 860–921.

[45] C. Cortes and V. Vapnik. "Support-vector networks". *Machine learning* 20 (1995), pages 273–297.

[46] T. Cover and P. Hart. "Nearest neighbor pattern classification". *IEEE transactions on information theory* 13.1 (1967), pages 21–27.

[47] M. Dagréou, P. Ablin, S. Vaiter, and T. Moreau. "How to compute Hessian-vector products?" *The Third Blogpost Track at ICLR 2024* (2024).

[48] F. J. Dangel. "Backpropagation beyond the gradient". PhD thesis. University of Tübingen, 2023.

[49] C. Darwin. "Origin of the Species". *British Politics and the environment in the long nineteenth century*. *Routledge*, 2023, pages 47–55.

[50] "Database Resources of the National Genomics Data Center, China National Center for Bioinformation in 2022". *Nucleic Acids Research* 50.D1 (2022), pages D27–D38.

[51] P. De Laplace. "Mémoire sur les suites récurro-récurrentes et sur leurs usages dans la théorie des hasards". *Mém. Acad. Roy. Sci. Paris* 6 (1774), pages 353–371.

[52] M. P. Deisenroth, A. A. Faisal, and C. S. Ong. "Mathematics for machine learning". *Cambridge University Press*, 2020.

[53] S. Deitrick. "Evaluating implicit visualization of uncertainty for public policy decision support". *Proc. AutoCarto*. 2012.

[54] T. Denoeux and M.-H. Masson. "Principal component analysis of fuzzy data using autoassociative neural networks". *IEEE Transactions on Fuzzy Systems* 12.3 (2004), pages 336–349.

[55] A. T. Dilthey, C. Jain, S. Koren, and A. M. Phillippy. "Strain-level metagenomic assignment and compositional estimation for long reads with MetaMaps". *Nature communications* 10.1 (2019), page 3066.

[56] J. A. Eisen. "Phylogenomics: improving functional predictions for uncharacterized genes by evolutionary analysis". *Genome research* 8.3 (1998), pages 163–167.

[57] J. A. Eisen. "Horizontal gene transfer among microbial genomes: new insights from complete genome analysis". *Current opinion in genetics & development* 10.6 (2000), pages 606–611.

[58] E. Elhaik. "Principal Component Analyses (PCA)-based findings in population genetic studies are highly biased and must be reevaluated". *Scientific Reports* 12 (2022), page 14683.

[59] G. Eraslan, Ž. Avsec, J. Gagneur, and F. J. Theis. "Deep learning: new computational modelling techniques for genomics". *Nature Reviews Genetics* 20.7 (2019), pages 389–403.

[60] A. Esteva, A. Robicquet, B. Ramsundar, V. Kuleshov, M. DePristo, K. Chou, C. Cui, G. Corrado, S. Thrun, and J. Dean. "A guide to deep learning in healthcare". *Nature medicine* 25.1 (2019), pages 24–29.

[61] J. Fan, S. Huang, and S. D. Chorlton. "BugSeq: a highly accurate cloud platform for long-read metagenomic analyses". *BMC bioinformatics* 22 (2021), pages 1–12.

[62] S.-H. Fan, P. Ebner, S. Reichert, T. Hertlein, S. Zabel, A. K. Lankapalli, K. Nieselt, K. Ohlsen, and F. Götz. "MpsAB is important for *Staphylococcus aureus* virulence and growth at atmospheric CO_2 levels". *Nature communications* 10.1 (2019), pages 1–12.

[63] S.-H. Fan, M. Matsuo, L. Huang, P. M. Tribelli, and F. Götz. "The MpsAB Bicarbonate Transporter Is Superior to Carbonic Anhydrase in Biofilm-Forming Bacteria with Limited CO2 Diffusion". *Microbiology Spectrum* 9.1 (2021), e00305–21.

[64] R. Faust, D. Glickenstein, and C. Scheidegger. "DimReader: Axis lines that explain non-linear projections". *IEEE transactions on visualization and computer graphics* 25.1 (2018), pages 481–490.

[65] J. Felsenstein. "The number of evolutionary trees". *Systematic zoology* 27.1 (1978), pages 27–33.

[66] J. Felsenstein. "Inferring phylogenies". *Sinauer Associates*, 2004.

[67] E. Fix and J. L. Hodges Jr. *Discriminatory analysis-nonparametric discrimination: Small sample performance*. Technical report. California Univ Berkeley, 1952.

[68] W. Fuhl, S. Zabel, and K. Nieselt. "Improving taxonomic classification with feature space balancing". *Bioinformatics Advances* 3.1 (2023), vbad092.

[69] W. Fuhl, S. Zabel, and K. Nieselt. "Scanpath Classification with an n-mer Deep Neural Network Architecture". *Proceedings of the 2025 Symposium on Eye Tracking Research and Applications*. ETRA '25. New York, NY, USA: *Association for Computing Machinery*, 2025.

[70] T. Gabaldón and M. A. Huynen. "Prediction of protein function and pathways in the genome era". *Cellular and Molecular Life Sciences CMLS* 61.7 (2004), pages 930–944.

[71] K. R. Gabriel. "The biplot graphic display of matrices with application to principal component analysis". *Biometrika* 58.3 (1971), pages 453–467.

[72] O. Ganea, G. Bécigneul, and T. Hofmann. "Hyperbolic neural networks". *Advances in neural information processing systems* 31 (2018).

[73] J. Gauthier, A. T. Vincent, S. J. Charette, and N. Derome. "A brief history of bioinformatics". *Briefings in bioinformatics* 20.6 (2019), pages 1981–1996.

[74] A. H. Gebremedhin and A. Walther. "An introduction to algorithmic differentiation". *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 10.1 (2020), e1334.

[75] A. Ghalichi, S. Reinhold, A. B. Rohrlach, A. A. Kalmykov, A. Childebayeva, H. Yu, F. Aron, L. Semerau, K. Bastert-Lamprichs, A. B. Belinskiy, et al. "The rise and transformation of Bronze Age pastoralists in the Caucasus". *Nature* (2024), pages 1–9.

[76] J. R. van Ginkel. "Handling Missing Data in Principal Component Analysis Using Multiple Imputation". *Essays on Contemporary Psychometrics*. Cham: *Springer International Publishing*, 2023, pages 141–161.

[77] G. P. Gladstone, P. Fildes, and G. Richardson. "Carbon dioxide as an essential factor in the growth of bacteria". *British Journal of Experimental Pathology* 16.3 (1935), page 335.

[78] I. Goodfellow, Y. Bengio, and A. Courville. "Deep learning". *MIT press*, 2016.

[79] S. Goodwin, J. D. McPherson, and W. R. McCombie. "Coming of age: ten years of next-generation sequencing technologies". *Nature Reviews Genetics* 17.6 (2016), pages 333–351.

[80] J. Görtler, T. Spinner, D. Streeb, D. Weiskopf, and O. Deussen. "Uncertainty-Aware Principal Component Analysis". *IEEE Transactions on Visualization and Computer Graphics* 26.1 (2019), pages 822–831.

[81] J. G. Greener, S. M. Kandathil, L. Moffat, and D. T. Jones. "A guide to machine learning for biologists". *Nature Reviews Molecular Cell Biology* 23.1 (2022), pages 40–55.

[82] A. Griewank and A. Walther. "Evaluating derivatives: principles and techniques of algorithmic differentiation". *SIAM*, 2008.

[83] M. Grinberg. "Flask web development: developing web applications with python". *O'Reilly Media, Inc.*, 2018.

[84] A. K. Gupta and D. K. Nagar. "Matrix variate distributions". *Chapman* and *Hall/CRC*, 2018.

[85] Y. Hasin, M. Seldin, and A. Lusis. "Multi-omics approaches to disease". *Genome biology* 18.1 (2017), pages 1–15.

[86] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman. "The elements of statistical learning: data mining, inference, and prediction". Volume 2. *Springer*, 2009.

[87] P. Hennig. *Animating Samples from Gaussian Distributions*. Technical Report 8. Max Planck Institute for Intelligent Systems, 2013.

[88] P. Hennig, M. A. Osborne, and H. P. Kersting. "Probabilistic Numerics". *Cambridge University Press*, 2022.

[89] W. Hennig. "Phylogenetic systematics". *University of Illinois Press*, 1999.

[90] C. Higuera, K. J. Gardiner, and K. J. Cios. "Self-organizing feature maps identify proteins critical to learning in a mouse model of down syndrome". *PLoS one* 10.6 (2015), e0129126.

[91] G. E. Hinton and S. Roweis. "Stochastic neighbor embedding". *Advances in neural information processing systems* 15 (2002).

[92] M. Hofreiter, D. Serre, H. N. Poinar, M. Kuch, and S. Pääbo. "Ancient DNA". *Nature Reviews Genetics* 2.5 (2001), pages 353–359.

[93] C. Holzhüter, A. Lex, D. Schmalstieg, H.-J. Schulz, H. Schumann, and M. Streit. "Visualizing uncertainty in biological expression data". *Visualization and Data Analysis 2012*. Volume 8294. SPIE. 2012, pages 251–261.

[94] T. Hoshino, H. Doi, G.-I. Uramoto, L. Wörmer, R. R. Adhikari, N. Xiao, Y. Morono, S. D'Hondt, K.-U. Hinrichs, and F. Inagaki. "Global diversity of microbial communities in marine sediment". *Proceedings of the national academy of sciences* 117.44 (2020), pages 27587–27597.

[95] H. Hotelling. "Analysis of a complex of statistical variables into principal components." *Journal of Educational Psychology* 24.6 (1933), pages 417–441.

[96] A. C. Howe, J. K. Jansson, S. A. Malfatti, S. G. Tringe, J. M. Tiedje, and C. T. Brown. "Tackling soil diversity with the assembly of large, complex metagenomes". *Proceedings of the National Academy of Sciences* 111.13 (2014), pages 4904–4909.

[97] B. N. Howie, P. Donnelly, and J. Marchini. "A flexible and accurate genotype imputation method for the next generation of genome-wide association studies". *PLoS genetics* 5.6 (2009), e1000529.

[98] Y. Huang, C. Yang, X.-f. Xu, W. Xu, and S.-w. Liu. "Structural and functional properties of SARS-CoV-2 spike protein: potential antivirus drug development for COVID-19". *Acta Pharmacologica Sinica* 41.9 (2020), pages 1141–1149.

[99] P. Hugenholtz, M. Chuvochina, A. Oren, D. H. Parks, and R. M. Soo. "Prokaryotic taxonomy and nomenclature in the age of big sequence data". *the ISME Journal* 15.7 (2021), pages 1879–1892.

[100] D. Hughes. "Exploiting genomics, genetics and chemistry to combat antibiotic resistance". *Nature Reviews Genetics* 4.6 (2003), pages 432–441.

[101] R. Hui, E. D'Atanasio, L. M. Cassidy, C. L. Scheib, and T. Kivisild. "Evaluating genotype imputation pipeline for ultra-low coverage ancient genomes". *Scientific Reports* 10.1 (2020), page 18542.

[102] J. Hullman, X. Qiao, M. Correll, A. Kale, and M. Kay. "In pursuit of error: A survey of uncertainty visualization evaluation". *IEEE transactions on visualization and computer graphics* 25.1 (2018), pages 903–913.

[103] J. Hullman, P. Resnick, and E. Adar. "Hypothetical Outcome Plots Outperform Error Bars and Violin Plots for Inferences About Reliability of Variable Ordering". *PLoS one* 10.11 (2015).

[104] D. H. Huson, B. Albrecht, C. Bağcı, I. Bessarab, A. Gorska, D. Jolic, and R. B. Williams. "MEGAN-LR: new algorithms allow accurate binning and easy interactive exploration of metagenomic long reads and contigs". *Biology direct* 13 (2018), pages 1–17.

[105] D. H. Huson, A. F. Auch, J. Qi, and S. C. Schuster. "MEGAN analysis of metagenomic data". *Genome research* 17.3 (2007), pages 377–386.

[106] D. H. Huson, R. Rupp, and C. Scornavacca. "Phylogenetic networks: concepts, algorithms and applications". *Cambridge University Press*, 2010.

[107] H. Hwang, H. Jeon, N. Yeo, and D. Baek. "Big data and deep learning for RNA biology". *Experimental & Molecular Medicine* (2024), pages 1–29.

[108] A. Inselberg and B. Dimsdale. "Parallel coordinates". *Human-Machine Interactive Systems* (2009), pages 199–233.

[109] Y. Ji, Z. Zhou, H. Liu, and R. V. Davuluri. "DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome". *Bioinformatics* 37.15 (2021), pages 2112–2120.

[110] M. I. Jordan and T. M. Mitchell. "Machine learning: Trends, perspectives, and prospects". *Science* 349.6245 (2015), pages 255–260.

[111] D. W. Joyce, A. Kormilitzin, K. A. Smith, and A. Cipriani. "Explainable artificial intelligence for mental health through transparency and interpretability for understandability". *npj Digital Medicine* 6.1 (2023), page 6.

[112] A. Kale, F. Nguyen, M. Kay, and J. Hullman. "Hypothetical outcome plots help untrained observers judge trends in ambiguous data". *IEEE transactions on visualization and computer graphics* 25.1 (2018), pages 892–902.

[113] A. Kamal, P. Dhakal, A. Y. Javaid, V. K. Devabhaktuni, D. Kaur, J. Zaientz, and R. Marinier. "Recent advances and challenges in uncertainty visualization: a survey". *Journal of Visualization* 24.5 (2021), pages 861–890.

[114] M. Kanehisa, M. Furumichi, M. Tanabe, Y. Sato, and K. Morishima. "KEGG: new perspectives on genomes, pathways, diseases and drugs". *Nucleic acids research* 45.D1 (2017), pages D353–D361.

[115] P. Kapli, Z. Yang, and M. J. Telford. "Phylogenetic tree building in the genomic age". *Nature Reviews Genetics* 21.7 (2020), pages 428–444.

[116] W. Karush. "Minima of functions of several variables with inequalities as side constraints". *M. Sc. Dissertation. Dept. of Mathematics, Univ. of Chicago* (1939).

[117] K. Katoh and H. Toh. "Recent developments in the MAFFT multiple sequence alignment program". *Briefings in Bioinformatics* 9.4 (2008), pages 286–298.

[118] A. J. Kavran and A. Clauset. "Denoising large-scale biological data using network filters". *BMC bioinformatics* 22 (2021), pages 1–21.

[119] P. J. Keeling and J. D. Palmer. "Horizontal gene transfer in eukaryotic evolution". *Nature Reviews Genetics* 9.8 (2008), pages 605–618.

[120] D. A. Keim, F. Mansmann, J. Schneidewind, J. Thomas, and H. Ziegler. "Visual analytics: Scope and challenges". *Visual data mining: Theory, techniques and tools for visual analytics*. *Springer*, 2008, pages 76–90.

[121] H. J. Kelley. "Gradient theory of optimal flight paths". *Ars Journal* 30.10 (1960), pages 947–954.

[122] M. C. Kim, R. Gate, D. S. Lee, A. Tolopko, A. Lu, E. Gordon, E. Shifrut, P. E. Garcia-Nieto, A. Marson, V. Ntranos, et al. "Method of moments framework for differential expression analysis of single-cell RNA sequencing data". *Cell* 187.22 (2024), pages 6393–6410.

[123] N. Kim, J. Ma, W. Kim, J. Kim, P. Belenky, and I. Lee. "Genome-resolved metagenomics: a game changer for microbiome medicine". *Experimental & Molecular Medicine* (2024), pages 1–12.

[124] C. Kinkeldey, A. M. MacEachren, and J. Schiewe. "How to assess visual communication of uncertainty? A systematic review of geospatial uncertainty visualisation user studies". *The Cartographic Journal* 51.4 (2014), pages 372–386.

[125] A. Klimovskaia, D. Lopez-Paz, L. Bottou, and M. Nickel. "Poincaré maps for analyzing complex hierarchies in single-cell data". *Nature communications* 11.1 (2020), page 2966.

[126] D. Kobak and P. Berens. "The art of using t-SNE for single-cell transcriptomics". *Nature communications* 10.1 (2019), page 5416.

[127] E. V. Koonin, K. S. Makarova, and L Aravind. "Horizontal gene transfer in prokaryotes: quantification and classification". *Annual Reviews in Microbiology* 55.1 (2001), pages 709–742.

[128] T. S. Korneliussen, A. Albrechtsen, and R. Nielsen. "ANGSD: analysis of next generation sequencing data". *BMC bioinformatics* 15 (2014), pages 1–13.

[129] S. G. Krantz and H. R. Parks. "The implicit function theorem: history, theory, and applications". *Springer Science & Business Media*, 2002.

[130] M. E. Kreye, Y. M. Goh, and L. B. Newnes. "Manifestation of uncertainty-A classification". *DS 68-6: Proceedings of the 18th International Conference on Engineering Design (ICED 11), Impacting Society through Engineering Design, Vol. 6: Design Information and Knowledge, Lyngby/Copenhagen, Denmark, 15.-19.08. 2011*. 2011.

[131] D. P. Kroese, T. Taimre, and Z. I. Botev. "Handbook of monte carlo methods". *John Wiley & Sons*, 2013.

[132] M. Krzywinski and N. Altman. "Points of significance: Importance of being uncertain." *Nature methods* 10.9 (2013).

[133] H. W. Kuhn and A. W. Tucker. "Nonlinear programming". *Berkeley: University of California Press.* (1951), pages 481–492.

[134] P. Kumar, R. K. Paul, H. S. Roy, M. Yeasin, Ajit, and A. K. Paul. "Big Data Analysis in Computational Biology and Bioinformatics". *Reverse Engineering of Regulatory Networks*. *Springer*, 2023, pages 181–197.

[135] S. Kumar, G. Stecher, M. Li, C. Knyaz, and K. Tamura. "MEGA X: molecular evolutionary genetics analysis across computing platforms". *Molecular biology and evolution* 35.6 (2018), pages 1547–1549.

[136] D. Lähnemann, J. Köster, E. Szczurek, D. J. McCarthy, S. C. Hicks, M. D. Robinson, C. A. Vallejos, K. R. Campbell, N. Beerenwinkel, A. Mahfouz, et al. "Eleven grand challenges in single-cell data science". *Genome biology* 21 (2020), pages 1–35.

[137] Y. Lan, G. Rosen, and R. Hershberg. "Marker genes that are less conserved in their sequences are useful for predicting genome-wide similarity levels between closely related prokaryotic strains". *Microbiome* 4 (2016), pages 1–13.

[138] E. S. Lander. "Initial impact of the sequencing of the human genome". *Nature* 470.7333 (2011), pages 187–197.

[139] D. J. Lane, B. Pace, G. J. Olsen, D. A. Stahl, M. L. Sogin, and N. R. Pace. "Rapid determination of 16S ribosomal RNA sequences for phylogenetic analyses." *Proceedings of the National Academy of Sciences* 82.20 (1985), pages 6955–6959.

[140] I. Lazaridis, N. Patterson, A. Mittnik, et al. "Ancient human genomes suggest three ancestral populations for present-day Europeans". *Nature* 513.7518 (2014), pages 409–413.

[141] J. A. Lee and M. Verleysen. "Nonlinear dimensionality reduction". *Springer Science & Business Media*, 2007.

[142] A. Lehrmann, M. Huber, A. C. Polatkan, A. Pritzkau, and K. Nieselt. "Visualizing dimensionality reduction of systems biology data". *Data Mining and Knowledge Discovery* 27 (2013), pages 146–165.

[143] I. Letunic and P. Bork. "Interactive tree of life (iTOL) v3: an online tool for the display and annotation of phylogenetic and other trees". *Nucleic acids research* 44.W1 (2016), W242–W245.

[144] P. Levontin and J. L. Walton. "Visualising uncertainty a short introduction". *Sad Press* and *Friends*, 2020.

[145] X. Li, Z. Li, H. Zhou, S. M. Gaynor, Y. Liu, H. Chen, R. Sun, R. Dey, D. K. Arnett, S. Aslibekyan, et al. "Dynamic incorporation of multiple in silico functional annotations empowers rare variant association analysis of large whole-genome sequencing studies at scale". *Nature genetics* 52.9 (2020), pages 969–983.

[146] Q. Liang, P. W. Bible, Y. Liu, B. Zou, and L. Wei. "DeepMicrobes: taxonomic classification for metagenomics with deep learning". *NAR Genomics and Bioinformatics* 2.1 (2020), lqaa009.

[147] W.-J. Lin and J. J. Chen. "Class-imbalanced classifiers for high-dimensional data". *Briefings in bioinformatics* 14.1 (2013), pages 13–26.

[148] G. C. Linderman, M. Rachh, J. G. Hoskins, S. Steinerberger, and Y. Kluger. "Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data". *Nature methods* 16.3 (2019), pages 243–245.

[149] C. Linnaeus. "Systema naturae". Volume 1. part 1. *Stockholm Laurentii Salvii*, 1758.

[150] S. Linnainmaa. "The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors". PhD thesis. Master's Thesis (in Finnish), Univ. Helsinki, 1970.

[151] S. Linnainmaa. "Taylor expansion of the accumulated rounding error". *BIT Numerical Mathematics* 16.2 (1976), pages 146–160.

[152] M. Lipson, A. Szécsényi-Nagy, S. Mallick, A. Pósa, B. Stégmár, V. Keerl, N. Rohland, K. Stewardson, M. Ferry, M. Michel, et al. "Parallel palaeogenomic transects reveal complex genetic history of early European farmers". *Nature* 551.7680 (2017), pages 368–372.

[153] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. Van Der Laak, B. Van Ginneken, and C. I. Sánchez. "A survey on deep learning in medical image analysis". *Medical image analysis* 42 (2017), pages 60–88.

[154] X. Liu, M. Milo, N. D. Lawrence, and M. Rattray. "A tractable probabilistic model for Affymetrix probe-level analysis across multiple chips". *Bioinformatics* 21.18 (2005), pages 3637–3644.

[155] K. J. Locey and J. T. Lennon. "Scaling laws predict global microbial diversity". *Proceedings of the National Academy of Sciences* 113.21 (2016), pages 5970–5975.

[156] M. I. Love, W. Huber, and S. Anders. "Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2". *Genome biology* 15 (2014), pages 1–21.

[157] Y. Lu, A. Yang, C. Quan, Y. Pan, H. Zhang, Y. Li, C. Gao, H. Lu, X. Wang, P. Cao, et al. "A single-cell atlas of the multicellular ecosystem of primary and metastatic hepatocellular carcinoma". *Nature communications* 13.1 (2022), page 4594.

[158] S. M. Lundberg and S.-I. Lee. "A unified approach to interpreting model predictions". *Advances in neural information processing systems* 30 (2017).

[159] A. Luqman, M. Z. Muttaqin, S. Yulaipi, P. Ebner, M. Matsuo, S. Zabel, P. M. Tribelli, K. Nieselt, D. Hidayati, and F. Götz. "Trace amines produced by skin bacteria accelerate wound healing in mice". *Communications Biology* 3.1 (2020), page 277.

[160] A. Luqman, S. Zabel, S. Rahmdel, B. Merz, N. Gruenheit, J. Harter, K. Nieselt, and F. Götz. "The neuromodulator-encoding sadA gene is widely distributed in the human skin microbiome". *Frontiers in Microbiology* 11 (2020), page 3045.

[161] L. Van der Maaten and G. Hinton. "Visualizing data using t-SNE." *Journal of Machine Learning Research* 9.11 (2008).

[162] A. M. MacEachren, R. E. Roth, J. O'Brien, B. Li, D. Swingley, and M. Gahegan. "Visual semiotics & uncertainty visualization: An empirical study". *IEEE transactions on visualization and computer graphics* 18.12 (2012), pages 2496–2505.

[163] D. J. MacKay. "Bayesian interpolation". *Neural computation* 4.3 (1992), pages 415–447.

[164] D. J. MacKay. "Information theory, inference and learning algorithms". *Cambridge university press*, 2003.

[165] J. R. Magnus and H. Neudecker. "Matrix differential calculus with applications in statistics and econometrics". *John Wiley & Sons*, 2019.

[166] T. Malepathirana, D. Senanayake, R. Vidanaarachchi, V. Gautam, and S. Halgamuge. "Dimensionality reduction for visualizing high-dimensional biological data". *Biosystems* 220 (2022), page 104749.

[167] S. Mallick and D. Reich. "The Allen Ancient DNA Resource (AADR): A curated compendium of ancient human genomes". Version V9. 2023.

[168] A. Margaryan, D. J. Lawson, M. Sikora, F. Racimo, S. Rasmussen, I. Moltke, L. M. Cassidy, E. Jørsboe, A. Ingason, M. W. Pedersen, et al. "Population genomics of the Viking world". *Nature* 585.7825 (2020), pages 390–396.

[169] C. C. Margossian. "A review of automatic differentiation and its efficient implementation". *Wiley interdisciplinary reviews: data mining and knowledge discovery* 9.4 (2019), e1305.

[170] R. L. Markelle Kelly and K. Nottingham. "The UCI Machine Learning Repository". URL: http://archive.ics.uci.edu/ml (visited on 06/27/2025).

[171] R. Martín, S. Miquel, P. Langella, and L. G. Bermúdez-Humarán. "The role of metagenomics in understanding the human microbiome in health and disease". *Virulence* 5.3 (2014), pages 413–423.

[172] M Matthews, L Rehak, J Famewo, T Taylor, J Robson, and H. I. Guelph. "Evaluation of new visualization approaches for representing uncertainty in the recognized maritime picture". *DRDC Atlantic CR* 177 (2008).

[173] L. McInnes, J. Healy, and J. Melville. "Umap: Uniform manifold approximation and projection for dimension reduction". *arXiv preprint arXiv:1802.03426* (2018).

[174] A. B. McIntyre, R. Ounit, E. Afshinnekoo, R. J. Prill, E. Hénaff, N. Alexander, S. S. Minot, D. Danko, J. Foox, S. Ahsanuddin, et al. "Comprehensive benchmarking and ensemble approaches for metagenomic classifiers". *Genome biology* 18.1 (2017), pages 1–19.

[175] G. McVean. "A genealogical interpretation of Principal Components Analysis". *PLoS Genet* 5 (10 2009), e1000686.

[176] K. Mendler, H. Chen, D. H. Parks, B. Lobb, L. A. Hug, and A. C. Doxey. "AnnoTree: visualization and exploration of a functionally annotated microbial tree of life". *Nucleic acids research* 47.9 (2019), pages 4442–4448.

[177] P. Menzel, K. L. Ng, and A. Krogh. "Fast and sensitive taxonomic classification for metagenomics with Kaiju". *Nature communications* 7.1 (2016), pages 1–9.

[178] J. Mercer. "Xvi. functions of positive and negative type, and their connection the theory of integral equations". *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character* 209.441-458 (1909), pages 415–446.

[179] R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley. "Deep learning for healthcare: review, opportunities and challenges". *Briefings in bioinformatics* 19.6 (2018), pages 1236–1246.

[180] J. Mistry, S. Chuguransky, L. Williams, M. Qureshi, G. A. Salazar, E. L. Sonnhammer, S. C. Tosatto, L. Paladin, S. Raj, L. J. Richardson, et al. "Pfam: The protein families database in 2021". *Nucleic acids research* 49.D1 (2021), pages D412–D419.

[181] F. Mock, F. Kretschmer, A. Kriese, S. Böcker, and M. Marz. "Taxonomic classification of DNA sequences beyond sequence similarity using deep neural networks". *Proceedings of the National Academy of Sciences* 119.35 (2022), e2122636119.

[182] A. M. Mood, F. A. Graybill, and D. C. Boes. "Introduction to the Theory of Statistics." *McGraw-Hill New York, NY*, 1974.

[183] B. Sousa da Mota, S. Rubinacci, D. I. Cruz Dávalos, C. E. G. Amorim, M. Sikora, N. N. Johannsen, M. H. Szmyt, P. Włodarczak, A. Szczepanek, M. M. Przybyła, et al. "Imputation of ancient human genomes". *Nature communications* 14.1 (2023), page 3660.

[184] K. P. Murphy. "Machine learning: a probabilistic perspective". *MIT press*, 2012.

[185] V. M. Narasimhan, N. Patterson, P. Moorjani, N. Rohland, R. Bernardos, S. Mallick, I. Lazaridis, N. Nakatsuka, I. Olalde, M. Lipson, et al. "The formation of human populations in South and Central Asia". *Science* 365.6457 (2019), eaat7487.

[186] National Center for Biotechnology Information. "Protein BLAST". 2019. URL: https://blast.ncbi.nlm.nih.gov/Blast.cgi?PROGRAM=blastp&PAGE_TYPE=BlastSearch&LINK_LOC=blasthome (visited on 06/27/2025).

[187] P. R. Nelson, P. Taylor, and J. MacGregor. "Missing data methods in PCA and PLS: Score calculations with incomplete observations". *Chemometrics and Intelligent Laboratory Systems* 35 (1996), pages 45–65.

[188] M. Nickel and D. Kiela. "Poincaré embeddings for learning hierarchical representations". *Advances in neural information processing systems* 30 (2017).

[189] R. Nielsen, T. Korneliussen, A. Albrechtsen, Y. Li, and J. Wang. "SNP calling, genotype calling, and sample allele frequency estimation from new-generation sequencing data". *PLoS one* (2012).

[190] R. Nielsen, J. S. Paul, A. Albrechtsen, and Y. S. Song. "Genotype and SNP calling from next-generation sequencing data". *Nature Reviews Genetics* 12.6 (2011), pages 443–451.

[191] R. Nielsen, A. H. Vaughn, and Y. Deng. "Inference and applications of ancestral recombination graphs". *Nature Reviews Genetics* 26.1 (2025), pages 47–58.

[192] K. Nieselt, F. Battke, A. Herbig, P. Bruheim, A. Wentzel, Ø. M. Jakobsen, H. Sletta, M. T. Alam, M. E. Merlo, J. Moore, et al. "The dynamic architecture of the metabolic switch in Streptomyces coelicolor". *BMC genomics* 11 (2010), pages 1–9.

[193] J. F. Nolan. "Analytical differentiation on a digital computer". PhD thesis. Massachusetts Institute of Technology, 1953.

[194] L. G. Nonato and M. Aupetit. "Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and layout enrichment". *IEEE Transactions on Visualization and Computer Graphics* 25.8 (2018), pages 2650–2673.

[195] D. Oktay, N. McGreivy, J. Aduol, A. Beatson, and R. P. Adams. "Randomized automatic differentiation". *International Conference on Learning Representations (ICLR)* (2021).

[196] I. Olalde, S. Brace, M. E. Allentoft, I. Armit, K. Kristiansen, T. Booth, N. Rohland, S. Mallick, A. Szécsényi-Nagy, A. Mittnik, et al. "The Beaker phenomenon and the genomic transformation of northwest Europe". *Nature* 555.7695 (2018), pages 190–196.

[197] G. J. Olsen and C. R. Woese. "Ribosomal RNA: a key to phylogeny". *The FASEB journal* 7.1 (1993), pages 113–123.

[198] L. Orlando, R. Allaby, P. Skoglund, C. D. Sarkissian, P. Stockhammer, M. C Ávila-Arcos, Q. Fu, J. Krause, E. Willerslev, A. C. Stone, and C. Warinner. "Ancient DNA analysis". *Nature Reviews Methods Primers* 1.1 (2021).

[199] J. A. O'Rawe, S. Ferson, and G. J. Lyon. "Accounting for uncertainty in DNA sequencing data". *Trends in Genetics* 31.2 (2015), pages 61–66.

[200] L. Padilla, M. Kay, and J. Hullman. "Uncertainty Visualization". *Computational Statistics in Data Science*. *Wiley*, 2022. Chapter 22, pages 405–421.

[201] R. D. Page and E. C. Holmes. "Molecular evolution: a phylogenetic approach". *John Wiley & Sons*, 2009.

[202] S. Pal, S. Mondal, G. Das, S. Khatua, and Z. Ghosh. "Big data in biology: the hope and present-day challenges in it". *Gene Reports* 21 (2020), page 100869.

[203] T. Palys, L. Nakamura, and F. M. Cohan. "Discovery and classification of ecological diversity in the bacterial world: the role of DNA sequence data". *International Journal of Systematic and Evolutionary Microbiology* 47.4 (1997), pages 1145–1156.

[204] D. H. Parks, M. Chuvochina, D. W. Waite, C. Rinke, A. Skarshewski, P.-A. Chaumeil, and P. Hugenholtz. "A standardized bacterial taxonomy based on genome phylogeny substantially revises the tree of life". *Nature biotechnology* 36.10 (2018), pages 996–1004.

[205] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. "Automatic differentiation in pytorch" (2017).

[206] N. Patterson, M. Isakov, T. Booth, L. Büster, C.-E. Fischer, I. Olalde, H. Ringbauer, A. Akbari, O. Cheronet, M. Bleasdale, et al. "Large-scale migration into Britain during the Middle to Late Bronze Age". *Nature* 601.7894 (2022), pages 588–594.

[207] N. Patterson, P. Moorjani, Y. Luo, S. Mallick, N. Rohland, Y. Zhan, T. Genschoreck, T. Webster, and D. Reich. "Ancient admixture in human history". *Genetics* 192.3 (2012), pages 1065–1093.

[208] N. Patterson, A. L. Price, and D. Reich. "Population structure and eigenanalysis". *PLoS genetics* 2.12 (2006), e190.

[209] K. Pearson. "LIII. On lines and planes of closest fit to systems of points in space". *The London, Edinburgh, and Dublin philosophical magazine and journal of science* 2.11 (1901), pages 559–572.

[210] R. Pearson. *pumadata: Various data sets for use with the puma package*. R package version 2.26.0. 2020.

[211] R. D. Pearson, X. Liu, G. Sanguinetti, M. Milo, N. D. Lawrence, and M. Rattray. "puma: a Bioconductor package for propagating uncertainty in microarray analysis". *BMC bioinformatics* 10.1 (2009), pages 1–10.

[212] W. R. Pearson. "An introduction to sequence similarity ("homology") searching". *Current protocols in bioinformatics* 42.1 (2013), pages 3–1.

[213] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. "Scikit-learn: Machine Learning in Python". *Journal of Machine Learning Research* 12 (2011), pages 2825–2830.

[214] W. Peng, T. Varanka, A. Mostafa, H. Shi, and G. Zhao. "Hyperbolic deep neural networks: A survey". *IEEE Transactions on pattern analysis and machine intelligence* 44.12 (2021), pages 10023–10044.

[215] R. Perez-Lopez, N. Ghaffari Laleh, F. Mahmood, and J. N. Kather. "A guide to artificial intelligence for cancer researchers". *Nature Reviews Cancer* (2024), pages 1–15.

[216] B. M. Peter. "A geometric relationship of F_2, F_3 and F_4-statistics with principal component analysis". *Philosophical Transactions of the Royal Society B* 377.1852 (2022), page 20200413.

[217] C. Plaisant, J. Grosjean, and B. B. Bederson. "SpaceTree: supporting exploration in large node link tree, design evolution and empirical evaluation". *IEEE Symposium on Information Visualization, 2002. INFOVIS 2002.* 2002, pages 57–64.

[218] M. C. Politi, P. K. Han, and N. F. Col. "Communicating the uncertainty of harms and benefits of medical interventions". *Medical Decision Making* 27.5 (2007), pages 681–695.

[219] D. M. Portik, C. T. Brown, and N. T. Pierce-Ward. "Evaluation of taxonomic classification and profiling methods for long-read shotgun metagenomic sequencing datasets". *BMC bioinformatics* 23.1 (2022), page 541.

[220] C. Posth, H. Yu, A. Ghalichi, H. Rougier, I. Crevecoeur, Y. Huang, H. Ringbauer, A. B. Rohrlach, K. Nägele, V. Villalba-Mouco, et al. "Palaeogenomics of upper palaeolithic to neolithic European hunter-gatherers". *Nature* 615.7950 (2023), pages 117–126.

[221] C. Posth, V. Zaro, M. A. Spyrou, S. Vai, G. A. Gnecchi-Ruscone, A. Modi, A. Peltzer, A. Mötsch, K. Nägele, Å. J. Vågene, et al. "The origin and legacy of the Etruscans through a 2000-year archeogenomic time transect". *Science Advances* 7.39 (2021), eabi7673.

[222] K. Pöthkow, B. Weber, and H.-C. Hege. "Probabilistic marching cubes". *Computer Graphics Forum*. Volume 30. 3. Wiley Online Library. 2011, pages 931–940.

[223] A. L. Price, N. J. Patterson, R. M. Plenge, M. E. Weinblatt, N. A. Shadick, and D. Reich. "Principal components analysis corrects for stratification in genome-wide association studies". *Nature genetics* 38.8 (2006), pages 904–909.

[224] M. N. Price, P. S. Dehal, and A. P. Arkin. "FastTree: computing large minimum evolution trees with profiles instead of a distance matrix". *Molecular biology and evolution* 26.7 (2009), pages 1641–1650.

[225] J. L. Puga, M. Krzywinski, and N. Altman. "Points of significance: Bayes' theorem." *Nature methods* 12.4 (2015).

[226] S. Purcell, B. Neale, K. Todd-Brown, L. Thomas, M. A. R. Ferreira, D. Bender, J. Maller, P. I. W. de Bakker, M. J. Daly, and P. C. Sham. "PLINK: a toolset for whole-genome association and population-based linkage analyses". *American Journal of Human Genetics* 81 (2007), pages 559–575.

[227] J. R. Quinlan. "Induction of decision trees". *Machine learning* 1 (1986), pages 81–106.

[228] A. Ralston. "Operons and prokaryotic gene regulation". *Nature Education* 1 (1 2008), page 216.

[229] S. Raue, S.-H. Fan, R. Rosenstein, S. Zabel, A. Luqman, K. Nieselt, and F. Götz. "The genome of *Staphylococcus epidermidis* O47". *Frontiers in Microbiology* 11 (2020), page 2061.

[230] J. A. Raven, J. Beardall, and P. Sánchez-Baracaldo. "The possible evolution and future of CO2-concentrating mechanisms". *Journal of Experimental Botany* 68.14 (2017), pages 3701–3716.

[231] A. Regev, S. A. Teichmann, E. S. Lander, I. Amit, C. Benoist, E. Birney, B. Bodenmiller, P. Campbell, P. Carninci, M. Clatworthy, et al. "The human cell atlas". *elife* 6 (2017), e27041.

[232] M. D. Robinson, D. J. McCarthy, and G. K. Smyth. "edgeR: a Bioconductor package for differential expression analysis of digital gene expression data". *bioinformatics* 26.1 (2010), pages 139–140.

[233] M. G. Rooks and W. S. Garrett. "Gut microbiota, metabolites and host immunity". *Nature Reviews Immunology* 16.6 (2016), pages 341–352.

[234] G. Rosen, E. Garbarine, D. Caseiro, R. Polikar, and B. Sokhansanj. "Metagenome Fragment Classification Using N-Mer Frequency Profiles". *Advances in bioinformatics* 2008 (2008).

[235] G. L. Rosen, E. R. Reichenberger, and A. M. Rosenfeld. "NBC: the Naive Bayes Classification tool webserver for taxonomic classification of metagenomic reads". *Bioinformatics* 27.1 (2011), pages 127–129.

[236] F. Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review* 65.6 (1958), page 386.

[237] F. Rosenblatt. "Principles of neurodynamics: Perceptrons and the theory of brain mechanisms". Volume 55. *Spartan books Washington, DC*, 1962.

[238] O. Rozenblatt-Rosen, M. J. Stubbington, A. Regev, and S. A. Teichmann. "The Human Cell Atlas: from vision to reality". *Nature* 550.7677 (2017), pages 451–453.

[239] S. Rubinacci, D. M. Ribeiro, R. J. Hofmeister, and O. Delaneau. "Efficient phasing and imputation of low-coverage sequencing data using large reference panels". *Nature genetics* 53.1 (2021), pages 120–126.

[240] C. Rudin. "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead". *Nature machine intelligence* 1.5 (2019), pages 206–215.

[241] N. Saitou and M. Nei. "The neighbor-joining method: a new method for reconstructing phylogenetic trees." *Molecular biology and evolution* 4.4 (1987), pages 406–425.

[242] F. Sanger, S. Nicklen, and A. R. Coulson. "DNA sequencing with chain-terminating inhibitors". *Proceedings of the national academy of sciences* 74.12 (1977), pages 5463–5467.

[243] L. K. Saul, K. Q. Weinberger, F. Sha, J. Ham, and D. D. Lee. "Spectral methods for dimensionality reduction." *Semi-supervised learning* 3 (2006).

[244] R. Schenkendorf. "A general framework for uncertainty propagation based on point estimate methods". PhD thesis. 2014.

[245] J. P. Schessner, E. Voytik, and I. Bludau. "A practical guide to interpreting and generating bottom-up proteomics data visualizations". *Proteomics* 22.8 (2022), page 2100103.

[246] C. Schulz, A. Nocaj, J. Goertler, O. Deussen, U. Brandes, and D. Weiskopf. "Probabilistic graph layout for uncertain network visualization". *IEEE Transactions on Visualization and Computer Graphics* 23.1 (2016), pages 531–540.

[247] J. J. Shynk. "Probability, random variables, and random processes: theory and signal processing applications". *John Wiley & Sons*, 2012, pages 533–535.

[248] H. Y. Simon, K. J. Siddle, D. J. Park, and P. C. Sabeti. "Benchmarking metagenomics tools for taxonomic classification". *Cell* 178.4 (2019), pages 779–794.

[249] M. G. Simpson. "Plant systematics". *Academic press*, 2019.

[250] M. Skeels, B. Lee, G. Smith, and G. G. Robertson. "Revealing uncertainty for information visualization". *Information Visualization* 9.1 (2010), pages 70–81.

[251] R. R. Sokal and C. D. Michener. "A statistical method for evaluating systematic relationships" (1958).

[252] C. Spearman. "General intelligence objectively determined and measured". *American Journal of Psychology* 15 (1904), pages 107–197.

[253] T. Sperlea. "Multiple Sequence Alignments". *Springer*, 2022.

[254] D. Spiegelhalter, M. Pearson, and I. Short. "Visualizing uncertainty about the future". *Science* 333.6048 (2011), pages 1393–1400.

[255] A. Srivastava, L. Malik, T. Smith, I. Sudbery, and R. Patro. "Alevin efficiently estimates accurate gene abundances from dscRNA-seq data". *Genome biology* 20 (2019), pages 1–16.

[256] M. Steinegger and J. Söding. "MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets". *Nature biotechnology* 35.11 (2017), pages 1026–1028.

[257] Streamlit. "Streamlit: Open-source app framework for Machine Learning and Data Science". https://streamlit.io/. 2024. (Visited on 06/27/2025).

[258] S. Sulheim, T. Kumelj, D. van Dissel, A. Salehzadeh-Yazdi, C. Du, G. P. Van Wezel, K. Nieselt, E. Almaas, A. Wentzel, and E. J. Kerkhoven. "Enzyme-constrained models and omics analysis of Streptomyces coelicolor reveal metabolic changes that enhance heterologous production". *iScience* 23.9 (2020).

[259] T. J. Sullivan. "Introduction to uncertainty quantification". Volume 63. *Springer*, 2015.

[260] M. Sundararajan, A. Taly, and Q. Yan. "Axiomatic attribution for deep networks". *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia, 3319–3328.

[261] Y. Suzuki. "Advent of a new sequencing era: Long-read and on-site sequencing". *Journal of Human Genetics* 65.1 (2020), page 1.

[262] J. Tamames, M. Cobo-Simón, and F. Puente-Sánchez. "Assessing the performance of different approaches for functional and taxonomic annotation of metagenomes". *BMC genomics* 20 (2019), pages 1–16.

[263] B. Taylor. "Methodus incrementorum directa et inversa [direct and reverse methods of incrementation](in latin). 1715.(Translated into English in Struik, DJ (1969). A Source Book in Mathematics 1200–1800. Cambridge, Massachusetts". *A Source Book in Mathematics* (1969), pages 1200–1800.

[264] J. R. Taylor. "An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements". *University Science Books*, 1997.

[265] S.-H. Teng. "Scalable algorithms in the age of big data and network sciences: Characterization, primitives, and techniques". *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 2018, pages 6–7.

[266] C. Tian, R. Kosoy, R. Nassir, A. Lee, P. Villoslada, L. Klareskog, L. Hammarström, H.-J. Garchon, A. E. Pulver, M. Ransom, et al. "European population genetic substructure: further definition of ancestry informative markers for distinguishing among diverse European ethnic groups". *Molecular Medicine* 15 (2009), pages 371–383.

[267] C. Tian, R. M. Plenge, M. Ransom, A. Lee, P. Villoslada, C. Selmi, L. Klareskog, A. E. Pulver, L. Qi, P. K. Gregersen, et al. "Analysis and application of European genetic substructure using 300 K SNP information". *PLoS genetics* 4.1 (2008), e4.

[268] Q. Tian, P. Zhang, Y. Zhai, Y. Wang, and Q. Zou. "Application and comparison of machine learning and database-based methods in taxonomic classification of high-throughput sequencing data". *Genome Biology and Evolution* 16.5 (2024), evae102.

[269] M. E. Tipping and C. M. Bishop. "Probabilistic principal component analysis". *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 61.3 (1999), pages 611–622.

[270] W. S. Torgerson. "Multidimensional scaling: I. Theory and method". *Psychometrika* 17.4 (1952), pages 401–419.

[271] E. R. Tufte. "Envisioning Information." *Cheshire, Connecticut, Graphics Press* 6410 (1990).

[272] N. A. Valous, F. Popp, I. Zörnig, D. Jäger, and P. Charoentong. "Graph machine learning for integrated multi-omics analysis". *British Journal of Cancer* (2024), pages 1–7.

[273] L. Van Der Maaten. "Accelerating t-SNE using tree-based algorithms". *Journal of Machine Learning Research* 15.1 (2014), pages 3221–3245.

[274] L. Van Der Maaten, E. Postma, and J. Van den Herik. "Dimensionality Reduction: a Comparative Review". *Journal of Machine Learning Research* 10.66-71 (2009), page 13.

[275] M. Varadi, D. Bertoni, P. Magana, U. Paramval, I. Pidruchna, M. Radhakrishnan, M. Tsenkov, S. Nair, M. Mirdita, J. Yeo, et al. "AlphaFold Protein Structure Database in 2024: providing structure coverage for over 214 million protein sequences". *Nucleic acids research* 52.D1 (2024), pages D368–D375.

[276] K. R. Veeramah, A. Rott, M. Groß, L. van Dorp, S. López, K. Kirsanow, C. Sell, J. Blöcher, D. Wegmann, V. Link, et al. "Population genomic analysis of elongated skulls reveals extensive female-biased immigration in Early Medieval Bavaria". *Proceedings of the National Academy of Sciences* 115.13 (2018), pages 3494–3499.

[277] A. Vellido. "The importance of interpretability and visualization in machine learning for applications in medicine and health care". *Neural computing and applications* 32.24 (2020), pages 18069–18083.

[278] J. C. Venter, M. D. Adams, E. W. Myers, P. W. Li, R. J. Mural, G. G. Sutton, H. O. Smith, M. Yandell, C. A. Evans, R. A. Holt, et al. "The sequence of the human genome". *Science* 291.5507 (2001), pages 1304–1351.

[279] K. Vervier, P. Mahé, M. Tournoud, J.-B. Veyrieras, and J.-P. Vert. "Large-scale machine learning for metagenomics sequence classification". *Bioinformatics* 32.7 (2016), pages 1023–1032.

[280] K. N. Vokinger, S. Feuerriegel, and A. S. Kesselheim. "Mitigating bias in machine learning for medicine". *Communications medicine* 1.1 (2021), page 25.

[281] J. van Waaij, S. Li, G. Garcia-Erill, A. Albrechtsen, and C. Wiuf. "Evaluation of population structure inferred by principal component analysis or the admixture model". *Genetics* 225.2 (2023), iyad157.

[282] W. G. Wade. "The oral microbiome in health and disease". *Pharmacological research* 69.1 (2013), pages 137–143.

[283] C.-C. Wang, S. Reinhold, A. Kalmykov, A. Wissgott, G. Brandt, C. Jeong, O. Cheronet, M. Ferry, E. Harney, D. Keating, et al. "Ancient human genome-wide data from a 3000-year interval in the Caucasus corresponds with eco-geographic regions". *Nature communications* 10.1 (2019), page 590.

[284] M. Weber, M. Zaheer, A. S. Rawat, A. K. Menon, and S. Kumar. "Robust large-margin learning in hyperbolic space". *Advances in Neural Information Processing Systems* 33 (2020), pages 17863–17873.

[285] D. Weiskopf. "Uncertainty visualization: Concepts, methods, and applications in biological data visualization". *Frontiers in Bioinformatics* 2 (2022).

[286] D. E. Wood, J. Lu, and B. Langmead. "Improved metagenomic analysis with Kraken 2". *Genome biology* 20.1 (2019), pages 1–13.

[287] W. Xie, V. Khosasih, A. Suwanto, and H.-K. Kim. "Characterization of lipases from *Staphylococcus aureus* and *Staphylococcus epidermidis* isolated from human facial sebaceous skin". *Journal of microbiology and biotechnology* 22.1 (2012), pages 84–91.

[288] R. Yaka, I. Mapelli, D. Kaptan, A. Doğu, M. Chyleński, Ö. D. Erdal, D. Koptekin, K. B. Vural, A. Bayliss, C. Mazzucato, et al. "Variable kinship patterns in Neolithic Anatolia revealed by ancient genomes". *Current Biology* 31.11 (2021), pages 2455–2468.

[289] Z. Yu, T. H. Coorens, M. M. Uddin, K. G. Ardlie, N. Lennon, and P. Natarajan. "Genetic variation across and within individuals". *Nature Reviews Genetics* 25.8 (2024), pages 548–562.

[290] S. Zabel, S. Breitling, C. Posth, and K. Nieselt. "A Probabilistic Approach to Visualize the Effect of Missing Data on PCA in Ancient Human Genomics". *BMC genomics* 26.1 (2025), page 537.

[291] S. Zabel, P. Hennig, and K. Nieselt. "VIPurPCA: Visualizing and Propagating Uncertainty in Principal Component Analysis". *IEEE Transactions on Visualization and Computer Graphics* (2023).

[292] S. Zabel, J. Müller, F. Götz, and K. Nieselt. "BLASTphylo: An Interactive Web Tool for Taxonomic and Phylogenetic Analysis of Prokaryotic Genes". *Journal of Computational Biology* (2023).

[293] D. Zhang, E. Adar, and J. Hullman. "Visualizing Uncertainty in Probabilistic Graphs with Network Hypothetical Outcome Plots (NetHOPs)". *IEEE Transactions on Visualization and Computer Graphics* 28.1 (2021), pages 443–453.

[294] Z. Zhao, A. Cristian, and G. Rosen. “Keeping up with the genomes: efficient learning of our increasing knowledge of the tree of life”. *BMC bioinformatics* 21 (2020), pages 1–23.

[295] T. Zuk and S. Carpendale. “Theoretical analysis of uncertainty visualizations”. *Visualization and data analysis 2006*. Volume 6060. SPIE. 2006, pages 66–79.