VIPurPCA: Visualizing and Propagating Uncertainty in Principal Component Analysis

Susanne Zabel, Philipp Hennig, and Kay Nieselt

***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.**

***Index Terms*—Uncertainty, Dimensionality Reduction, Visualization.**

I. 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 [1], in bioinformatics, high-throughput methods provide large-scale omics data [2], in computer vision, high-resolution images of thousands of pixels are processed for image recognition [3], 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. Methods for dimensionality reduction can be divided into linear and nonlinear approaches. The most prominent linear dimensionality reduction is principal component analysis (PCA; [4]), which produces new features as linear combinations of the original variables. 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 (Fig. 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 [5]. 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 Fig. 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 (Fig. 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 (Fig. 2). 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.

Philipp Hennig is with MPI for Intelligent Systems, 72076 Tübingen, Germany.

The authors are with the Faculty of Science, University of Tübingen, 72076 Tübingen, Germany.

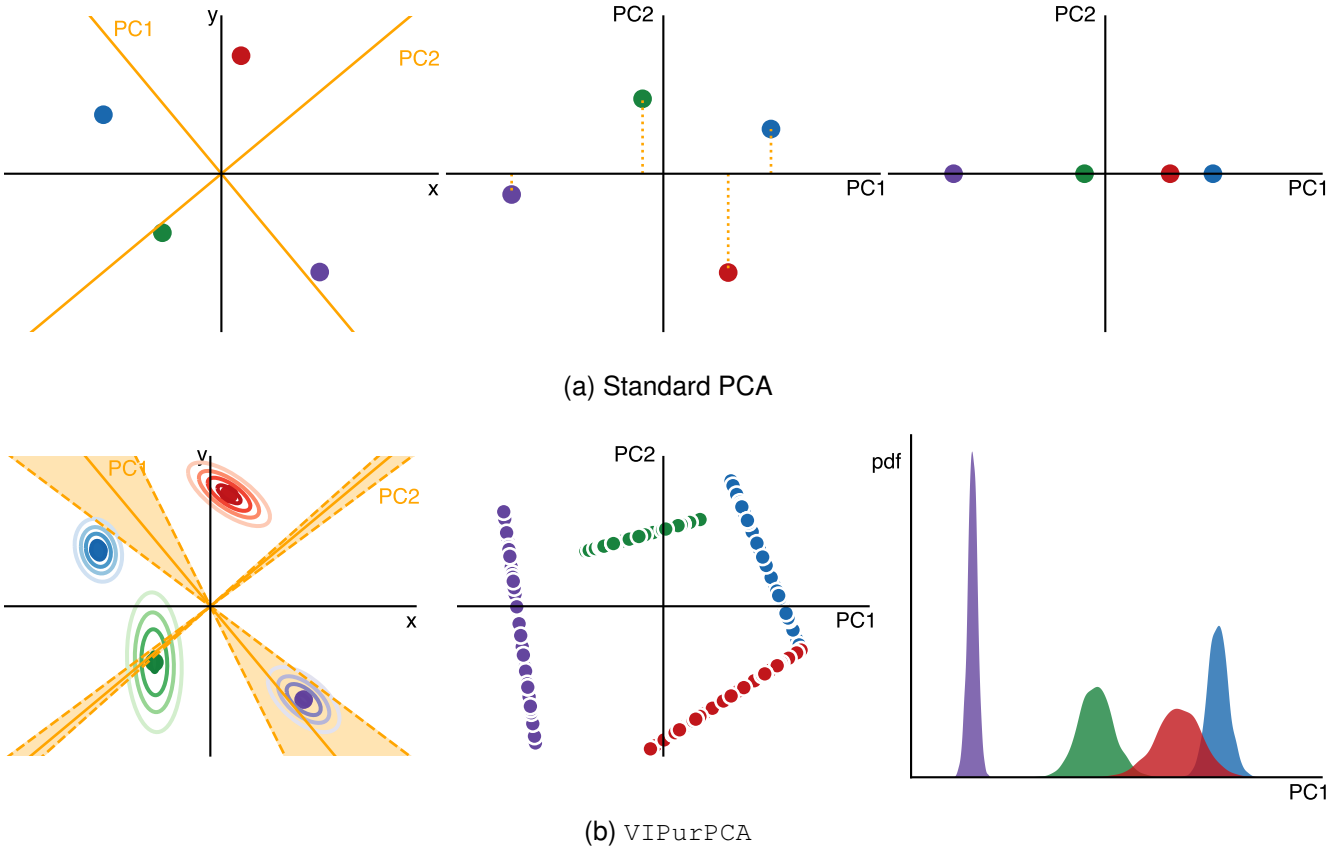


Fig. 1. **Comparison of standard PCA (a) and `VIPurPCA` (b).** (a) Left: PCA is applied to four 2-dimensional samples. The directions of the computed Principal Components (PCs) are indicated in orange. Middle: Dataset in terms of the new coordinate system. Right: Dimensionality reduction by projecting the data to the first PC. (b) Left: Input data including (correlated) Gaussian uncertainties are used as an input for `VIPurPCA`, which propagates the uncertainty to the PCA's output. Mean and 1σ-interval of computed PCs are indicated in orange. 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.

II. Related Work

There exists a large variety of dimensionality reduction techniques, for reviews see for example [6]–[9]. Traditionally, linear techniques like principal component analysis (PCA) [4], factor analysis [10] and classic multidimensional scaling [11] were used, which more recently have been complemented by a large number of nonlinear techniques to handle complex nonlinear data [8]. 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) [12] 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. [13] 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. [13] 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 [14] and Generalized Sensitivity Scatterplots [15] using derivatives to determine the sensitivities. Rather than

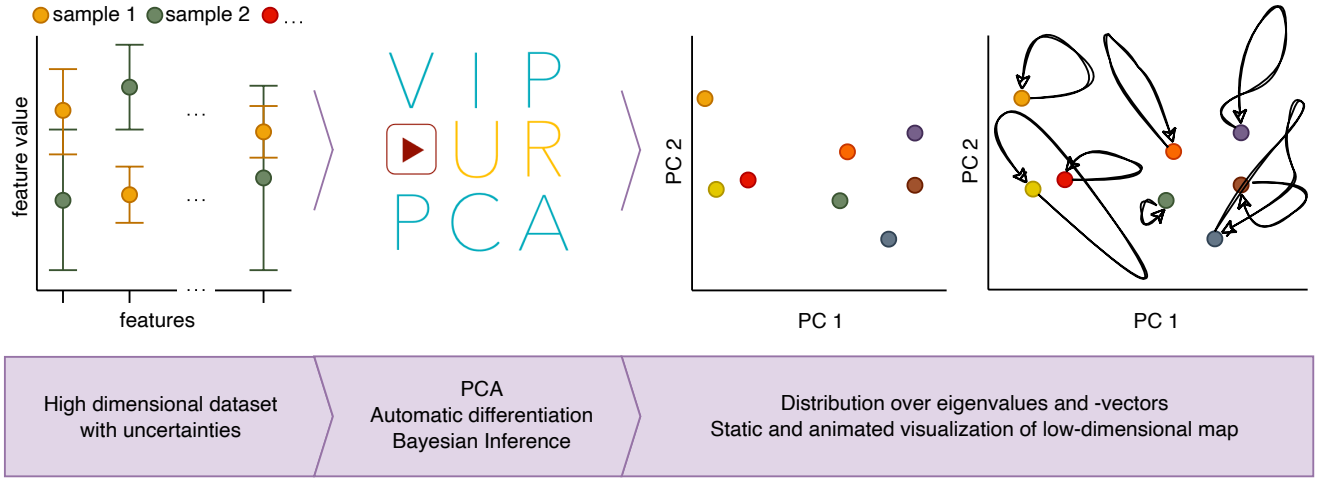


Fig. 2. **Workflow of `VIPurPCA`**. As an input `VIPurPCA` uses a high-dimensional dataset carrying uncertainties. Applying `VIPurPCA` to the inputs' mean provides the standard output of PCA (eigenvectors, eigenvalues, and lower dimensional map). Additionally, uncertainties of the inputs are propagated using automatic differentiation and Bayesian inference. The inferred uncertainty of the eigenvectors is visualized in an animated low-dimensional map.

showing sensitivities of one variable with respect to another variable, Faust et al. [16] 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 [17], [18] or volume rendering [19] is not covered in detail.

Communicating uncertainties visually is a big challenge [20]. Levontin and Walton [21] recently reviewed available approaches to tackle this task and remaining challenges. There are a variety of approaches to classify uncertainty visualization methods [22]–[25]. In this work, uncertainty is modeled using a probabilistic approach which requires the visualization of distributions. 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. [26]. 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 [27], shown in hypothetical outcome plots [28] one after another in an animation [29] 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 [30] 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).

III. Background

A. Uncertainty and Sensitivity Analysis

Uncertainty analysis or uncertainty quantification determines the imprecision of a model outcome $y = f(x_1, \ldots, x_n)$ given the variation of all model variables $x_1, \ldots, x_n$. Uncertainties associated with y can be expressed in many ways, for example, lower-order moments such as mean and (co-)variance, or a complete probability distribution over the output(s) [31]. There exist different probabilistic approaches for uncertainty propagation including simulation-based methods like Monte Carlo simulations and local expansion-based methods like the Taylor series method [32]. Basically, in Monte-Carlo-based uncertainty analysis, a desired function is executed many times on randomly drawn samples from the distributions over the function inputs to obtain the probability distribution over the targeted outcomes. Though highly general, Monte Carlo methods can be computationally expensive [33]. As the Monte Carlo approach outperforms the linearization approach in terms of accuracy it is often used for validation [34].

As analytical solutions for uncertainty propagation through (non)linear functions are only available for a limited number of cases (e.g., in very low dimensions), approximations have to be applied. Here, the Taylor series expansion is used to approximate the function of interest by a surrogate function facilitating the computation of the closed-form solution. Using the first-order Taylor series approximation, the mean and the covariance of the output variables can be estimated, which are sufficient to describe the output distribution, if the output is Gaussian distributed [35].

Let $\boldsymbol{x} \in \mathbb{R}^p$ be a random normal distributed vector with mean $\boldsymbol{\mu_x}$ and covariance matrix $\boldsymbol{\Sigma_x}$ ($\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu_x}, \boldsymbol{\Sigma_x})$), and let $f : \mathbb{R}^p \to \mathbb{R}^q$ be a nonlinear function such that $\boldsymbol{y} = f(\boldsymbol{x})$. The first-order Taylor approximation expanded at the point $\boldsymbol{\mu_x}$ is given by

$$\boldsymbol{y} \approx \hat{\boldsymbol{y}} = f(\boldsymbol{\mu_x}) + \boldsymbol{J}(\boldsymbol{x} - \boldsymbol{\mu_x}) \tag{1}$$

where $\boldsymbol{J} \in \mathbb{R}^{q \times p}$ is the Jacobian $\frac{\partial f}{\partial \boldsymbol{x}}\big|_{\boldsymbol{x}=\boldsymbol{\mu_x}}$ evaluated at $\boldsymbol{\mu_x}$. The resulting mean $\boldsymbol{\mu_{\hat{y}}}$ and covariance $\boldsymbol{\Sigma_{\hat{y}}}$ of $\hat{\boldsymbol{y}}$ are expressed by

$$\boldsymbol{\mu_{\hat{y}}} = f(\boldsymbol{\mu_x}) \tag{2}$$

$$\boldsymbol{\Sigma_{\hat{y}}} = \boldsymbol{J}\boldsymbol{\Sigma_x}\boldsymbol{J}^T. \tag{3}$$

As $\boldsymbol{y}$ is approximated by a linearization, the Taylor series expansion is biased when applied to highly nonlinear functions [35], [36].

Sensitivity analysis addresses the question of how much each input contributes to the output uncertainty. The Jacobian provides sensitivity coefficients of all outputs with respect to each input value and therefore provides information on the importance of individual inputs on the function's outputs. Using linearization for uncertainty propagation as described before includes the computation of the Jacobian and thereby automatically facilitates sensitivity analysis [34]. A computationally efficient way of differentiating (non)linear functions is described in section III-D.

B. Principal Component Analysis

Consider a data matrix $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ where rows represent observations and columns represent features, such that each of the n observations is defined by a p-dimensional feature vector $\{\boldsymbol{x}_i\}_{i=1,\dots,n}$ with $\boldsymbol{x}_i \in \mathbb{R}^p$. Furthermore, column-wise zero empirical mean is assumed. Starting with p-dimensional feature vectors PCA projects them into a q-dimensional subspace ($q \leq p$) which is spanned by q orthonormal directions, called principal components (PCs). The projections $\{\boldsymbol{t}_i\}_{i=1,\dots,n}$ with $\boldsymbol{t}_i \in \mathbb{R}^q$ are obtained by an orthogonal linear transformation of the original observations $\boldsymbol{x}_i$ and the weight vectors (PCs) $\{\boldsymbol{w}_k\}_{k=1,\dots,q}$ with $\boldsymbol{w}_k \in \mathbb{R}^q$:

$$\boldsymbol{t}_{ik} = \boldsymbol{x}_i \cdot \boldsymbol{w}_k \tag{4}$$

Mathematically, the weight vectors can be either computed by maximizing the sum of variances of the projections [37] or by minimizing the mean-squared error between the original vectors and their projections [4], which can be shown to be equivalent. Given a principal component vector $\boldsymbol{w}$, the projected data onto the PC is computed by $\boldsymbol{X}\boldsymbol{w}$ and its sample variance $\sigma^2_{\boldsymbol{X}\boldsymbol{w}}$ is given by:

$$\sigma^2_{\boldsymbol{X}\boldsymbol{w}} = \frac{1}{n-1}(\boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{X}\boldsymbol{w}) \tag{5}$$

$$= \boldsymbol{w}^T \frac{\boldsymbol{X}^T\boldsymbol{X}}{n-1}\boldsymbol{w} \tag{6}$$

$$= \boldsymbol{w}^T\boldsymbol{\Sigma_x}\boldsymbol{w} \tag{7}$$

where $\boldsymbol{\Sigma_x}$ denotes the covariance matrix of the original features in $\boldsymbol{X}$. The vector $\boldsymbol{w}$ is chosen in a way such that $\boldsymbol{\sigma}^2_{\boldsymbol{X}\boldsymbol{w}}$ is maximized under the constraint that $\|\boldsymbol{w}\| = 1$ [37]:

$$\boldsymbol{w} = \arg\max_{\|\boldsymbol{w}\|=1} \boldsymbol{w}^T\boldsymbol{\Sigma_x}\boldsymbol{w} \tag{8}$$

Using the Lagrange multiplier λ the objective function can be rewritten as:

$$L(\boldsymbol{w}) = \boldsymbol{w}^T\boldsymbol{\Sigma_x}\boldsymbol{w} - \lambda(\boldsymbol{w}^T\boldsymbol{w} - 1) \tag{9}$$

$$\frac{\partial L}{\partial \boldsymbol{w}} = 2\boldsymbol{\Sigma_x}\boldsymbol{w} - 2\lambda\boldsymbol{w} = 0 \tag{10}$$

$$\boldsymbol{\Sigma_x}\boldsymbol{w} = \lambda\boldsymbol{w} \tag{11}$$

Thus, the desired vector $\boldsymbol{w}$ is an eigenvector of the covariance matrix $\boldsymbol{\Sigma_x}$, corresponding to the largest eigenvalue λ. It can be shown that $\boldsymbol{\Sigma_x}$ has a set of p real eigenvalues λ_k, and their corresponding eigenvectors $\boldsymbol{w}_k$ are the weight vectors used for the linear transformation:

$$\boldsymbol{T} = \boldsymbol{X}\boldsymbol{W} \tag{12}$$

where $\boldsymbol{W}$ denotes the weight matrix of eigenvectors.

C. Gaussian Distributions on Matrices

In this section, Gaussian probability distributions over matrices are shortly introduced [38]. Given a matrix $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ describing observations of p variables for n samples. Typically, row vectors $\boldsymbol{x}_i$ are considered as samples drawn from a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with $\boldsymbol{\mu} \in \mathbb{R}^p$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{p \times p}$. In contrast, matrix variate normal distributions consider each observation to be distributed around its own mean, and to potentially co-vary with all other observations. One way to represent this distribution is to distribute $\mathrm{vec}(\boldsymbol{X}) \in \mathbb{R}^{np}$ as a multivariate Gaussian distribution. $\mathrm{vec}(\boldsymbol{X})$ describes the vectorization of the matrix, which is equal to concatenating the columns of $\boldsymbol{X}$ into one vector. Mean $\boldsymbol{M} \in \mathbb{R}^{np}$ and covariance $\boldsymbol{C} \in \mathbb{R}^{np \times np}$ are defined as:

$$\boldsymbol{M} = \mathbb{E}[\mathrm{vec}(\boldsymbol{X})] \tag{13}$$

$$\boldsymbol{C} = \mathbb{E}[(\mathrm{vec}(\boldsymbol{X}) - \boldsymbol{M})(\mathrm{vec}(\boldsymbol{X}) - \boldsymbol{M})^T] \tag{14}$$

The matrix normal distribution can be considered a multivariate normal distribution by rearranging matrices into a vector [39]. The covariance matrix is structured by a Kronecker product of two smaller matrices $\boldsymbol{V} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{W} \in \mathbb{R}^{p \times p}$ representing the sample's and feature's covariance matrices, respectively, such that

$$\mathrm{vec}(\boldsymbol{X}) \sim \mathcal{N}(\mathrm{vec}(\boldsymbol{M}), \boldsymbol{W} \otimes \boldsymbol{V})$$

D. Auto-differentiating Linear Algebra

Many machine learning algorithms require the evaluation of derivatives when optimizing an objective function. The backpropagation algorithm has been widely used in training neural networks to compute gradients of the loss function with respect to the weights of the network given a single input-output training sample. Backpropagation belongs to a bigger family of techniques for the computation of derivatives, called automatic (or algorithmic) differentiation (AD). Instead of calculating the derivative analytically, this set of techniques evaluates the derivative of a function at a given point. AD can be applied to any regular code containing not only elementary arithmetic operations and functions, but also statements like branchings, loops, and recursions. As the derivatives of these individual operations are known, AD repeatedly applies the chain rule to compute the overall derivative for any arbitrary combination of operations at machine precision [40].

In general, for any vector valued function $\boldsymbol{y} = f(\boldsymbol{x})$, with $f : \mathbb{R}^p \rightarrow \mathbb{R}^q$, the Jacobian $\boldsymbol{J}_f$ can be computed at a specific input point $\boldsymbol{a}$:

$$\boldsymbol{J}_f = \begin{bmatrix} \frac{\partial \boldsymbol{y}_1}{\partial \boldsymbol{x}_1} & \cdots & \frac{\partial \boldsymbol{y}_1}{\partial \boldsymbol{x}_p} \\ \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{y}_q}{\partial \boldsymbol{x}_1} & \cdots & \frac{\partial \boldsymbol{y}_q}{\partial \boldsymbol{x}_p} \end{bmatrix} \Bigg|_{\boldsymbol{x}=\boldsymbol{a}} \tag{15}$$

AD computes the gradient of each element of the output with respect to each element of the input and it can be extended to matrix-valued functions mapping from $\mathbb{R}^{k\times l} \rightarrow \mathbb{R}^{m\times n}$ resulting in a Jacobian of size $m \times n \times k \times l$, but also to higher ranking tensors.

There exists a large number of AD implementations for a large variety of programming languages. For example, TensorFlow [41], PyTorch [42], autograd [43], mxnet [44], and JAX [45] provide libraries for AD in Python. However, gradient computation of linear algebra primitives like QR decomposition, Cholesky decomposition, singular value decomposition, or symmetric eigendecomposition is not available in all frameworks. Furthermore, automatic full Jacobian computation instead of summarized gradients is only provided for some platforms. JAX provides both functionalities and can differentiate native Python and Numpy code in a highly efficient manner by using XLA (Accelerated Linear Algebra) to compile and run the code on accelerators like GPUs and TPUs. Therefore, using JAX it becomes possible to compute the Jacobian of an eigen decomposition with respect to the input data which is required for the computation of the covariance matrix of the conditional distribution $p(\boldsymbol{U}, \boldsymbol{\Lambda}|\boldsymbol{Y})$ (eq. 22).

IV. Methods

A. Bayesian Inference

Let $p(\boldsymbol{X}) = \sim \mathcal{N}(\boldsymbol{X}; \boldsymbol{\mu}, \boldsymbol{\Xi})$, where $\boldsymbol{X} \in \mathbb{R}^{n\times p}$, be a matrix Gaussian distribution describing the prior knowledge about a high-dimensional data set. Let the observations $\boldsymbol{Y} \in \mathbb{R}^{n\times p}$, given $\boldsymbol{X}$, be distributed according to the matrix Gaussian distribution $p(\boldsymbol{Y}|\boldsymbol{X}) = \mathcal{N}(\boldsymbol{Y}; \boldsymbol{X}, \boldsymbol{W} \otimes \boldsymbol{V})$, where $\boldsymbol{V} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{W} \in \mathbb{R}^{p\times p}$. The covariance matrix $\boldsymbol{W} \otimes \boldsymbol{V}$ describes the (structured) noise introduced by observing the data. Note here that the covariance matrix of $\boldsymbol{Y}$ can be any positive semidefinite matrix.

Let $\boldsymbol{F} : \mathbb{R}^{n\times p} \rightarrow \mathbb{R}^{p\times q}$, $\mathbb{R}^q$ be the function $\boldsymbol{U}, \boldsymbol{\Lambda} = \boldsymbol{F}(\boldsymbol{X})$ to compute the PCA ($\boldsymbol{U}$: eigenvectors, $\boldsymbol{\Lambda}$: eigenvalues) of the high-dimensional data $\boldsymbol{X}$ as an eigendecomposition of the covariance matrix (see Section III-B). Note that the function $\boldsymbol{F}(\boldsymbol{X})$, i.e. the computation of the covariance matrix and its eigendecomposition, is not linear in its input $\boldsymbol{X}$. In the following, the functions $\boldsymbol{F}_{\boldsymbol{U}}$ and $\boldsymbol{F}_{\boldsymbol{\Lambda}}$ consider the outputs $\boldsymbol{U}$ and $\boldsymbol{\Lambda}$, respectively.

Since we do not have access to $\boldsymbol{X}$, but to $\boldsymbol{Y}$, we are interested in the distributions $p(\boldsymbol{U}|\boldsymbol{Y})$ and $p(\boldsymbol{\Lambda}|\boldsymbol{Y})$. Below, results are derived for $p(\boldsymbol{U}|\boldsymbol{Y})$, which can be used analogously to derive $p(\boldsymbol{\Lambda}|\boldsymbol{Y})$. $p(\boldsymbol{U}|\boldsymbol{Y})$ can be computed by marginalizing over $\boldsymbol{X}$ assuming that $\boldsymbol{U}$ is independent of $\boldsymbol{Y}$ given $\boldsymbol{X}$:

$$p(\boldsymbol{U}|\boldsymbol{Y}) = \int p(\boldsymbol{U}|\boldsymbol{X}, \boldsymbol{Y}) \cdot p(\boldsymbol{X}|\boldsymbol{Y}) d\boldsymbol{X} \tag{16}$$

$$= \int p(\boldsymbol{U}|\boldsymbol{X}) \cdot p(\boldsymbol{X}|\boldsymbol{Y}) d\boldsymbol{X} \tag{17}$$

Mean and variance of the linear conditional Gaussian distribution $p(\boldsymbol{X}|\boldsymbol{Y})$ can be easily inferred given $p(\boldsymbol{Y}|\boldsymbol{X})$ and $p(\boldsymbol{X})$ [46]:

$$p(\boldsymbol{X}|\boldsymbol{Y}) \sim \mathcal{N}(\boldsymbol{X}; \boldsymbol{M}, \boldsymbol{P}) \tag{18}$$

$$\boldsymbol{M} = \boldsymbol{\mu} + \boldsymbol{\Xi} \cdot (\boldsymbol{\Xi} + \boldsymbol{W} \otimes \boldsymbol{V})^{-1} \cdot (\boldsymbol{Y} - \boldsymbol{\mu}) \tag{19}$$

$$\boldsymbol{P} = (\boldsymbol{X} i^{-1} + (\boldsymbol{W} \otimes \boldsymbol{V})^{-1})^{-1} \tag{20}$$

The other term $p(\boldsymbol{U}|\boldsymbol{X})$ can be written as a Dirac likelihood $p(\boldsymbol{U}|\boldsymbol{X}) = \delta(\boldsymbol{U} - \boldsymbol{F}_{\boldsymbol{U}}(\boldsymbol{X}))$. Due to the non-linearity of $\boldsymbol{F}_{\boldsymbol{U}}$ in its input $\boldsymbol{X}$, the first and second moment of $p(\boldsymbol{U}|\boldsymbol{X})$ can not be inferred directly. Therefore, $\boldsymbol{F}_{\boldsymbol{U}}$ is linearized around $\boldsymbol{M}$ (eq. 19) by a first-order Taylor approximation (eq. 1). By only considering the first two Taylor polynomials and by assuming that $\boldsymbol{F}_{\boldsymbol{U}}(\boldsymbol{X})$ is differentiable, the linear approximation is given as

$$\boldsymbol{F}_{\boldsymbol{U}}(\boldsymbol{X}) \approx \boldsymbol{F}_{\boldsymbol{U}}(\boldsymbol{M}) + \boldsymbol{\nabla}_{\boldsymbol{X}} \boldsymbol{F}_{\boldsymbol{U}}(\boldsymbol{M})(\boldsymbol{X} - \boldsymbol{M}), \tag{21}$$

where $\boldsymbol{\nabla}_{\boldsymbol{X}} \boldsymbol{F}_{\boldsymbol{U}}(\boldsymbol{M}) := \boldsymbol{J}_{\boldsymbol{M}} \in \mathbb{R}^{pq\times np}$ is the Jacobian of the function $\boldsymbol{F}_{\boldsymbol{U}}(\boldsymbol{X})$ evaluated at $\boldsymbol{M}$. Since Gaussians reproduce under linear operations [46], a closed-form solution (eq. 2, eq. 3) for $p(\boldsymbol{U}|\boldsymbol{Y})$ can be approximated by

$$p(\boldsymbol{U}|\boldsymbol{Y}) \approx \mathcal{N}(\boldsymbol{U}; \boldsymbol{F}_{\boldsymbol{U}}(\boldsymbol{M}), \boldsymbol{J}_{\boldsymbol{M}} \boldsymbol{P} \boldsymbol{J}_{\boldsymbol{M}}^T). \tag{22}$$

Assuming that $p(\boldsymbol{X})$ is centered around zero ($\boldsymbol{\mu} \rightarrow 0$) and that we are totally uncertain about this distribution ($\boldsymbol{\Xi} \rightarrow \infty$), $\boldsymbol{M}$ (eq. 19) and $\boldsymbol{P}$ (eq. 20) have the following limits:

$$\boldsymbol{M} \xrightarrow{\boldsymbol{\mu}\rightarrow 0, \boldsymbol{\Xi}\rightarrow\infty} \boldsymbol{Y} \tag{23}$$

$$\boldsymbol{P} \xrightarrow{\boldsymbol{\mu}\rightarrow 0, \boldsymbol{\Xi}\rightarrow\infty} \boldsymbol{W} \otimes \boldsymbol{V} \tag{24}$$

Therefore,

$$p(\boldsymbol{U}|\boldsymbol{Y}) \xrightarrow{\boldsymbol{M}\rightarrow\boldsymbol{Y}, \boldsymbol{P}\rightarrow\boldsymbol{W}\otimes\boldsymbol{V}} \mathcal{N}(\boldsymbol{U}; \boldsymbol{F}_{\boldsymbol{U}}(\boldsymbol{Y}), \boldsymbol{J}_{\boldsymbol{Y}}(\boldsymbol{W} \otimes \boldsymbol{V}) \boldsymbol{J}_{\boldsymbol{Y}}^T). \tag{25}$$

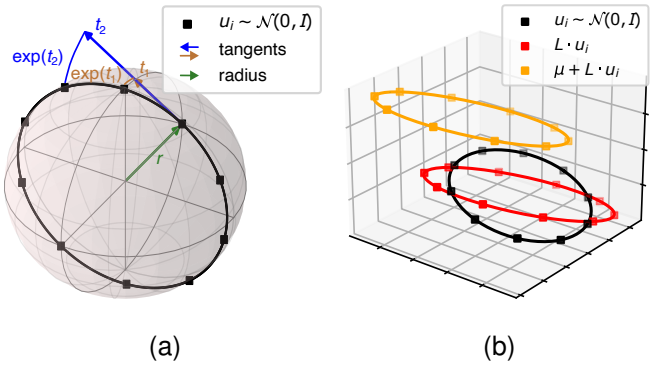


Fig. 3. Schematic view of drawing equipotential samples from a Gaussian distribution. (a) The grey sphere (2-manifold) describes equipotential samples of a three-dimensional standard Gaussian distribution defined by $\boldsymbol{r}$ (green vector). The randomly chosen 1-dimensional black circle lies on the manifold and is used to draw samples by looping through the orbit in equidistant intervals. The exponential map is used to map from the tangent bundle (Lie algebra) to the manifold (Lie group). (b) Standard normal distributed samples lying on the orbit (black) are scaled, rotated (red), and shifted (yellow) corresponding to the mean and covariance of the target distribution.

B. Animating Samples from Gaussian Distributions

To get an idea about the structure of samples from Gaussian distributions, an animation is used to display animated samples following a trajectory of equipotential lines of the probability distribution. This approach was introduced by [30] on animating samples from correlated Gaussian beliefs and adopted in this work for our purpose. Assume we want to display samples from a Gaussian distribution $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}), \boldsymbol{x} \in \mathbb{R}^m$. It is important to mention that this also includes matrices of size $n \times p$ vectorized to a vector of size $m = np$. Samples $\boldsymbol{s} \sim \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ can be drawn in the following way:

1) Construct the Cholesky decomposition $\boldsymbol{L}$ of $\boldsymbol{\Sigma}$, such that $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^T$ and $\boldsymbol{L}$ is a lower triangular matrix with real and positive diagonal entries
2) Draw a sample $\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{u}, \boldsymbol{0}, \boldsymbol{I}), \boldsymbol{u} \in \mathbb{R}^m$ from the standard Gaussian distribution
3) Compute a sample $\boldsymbol{s} = \boldsymbol{\mu} + \boldsymbol{L}\boldsymbol{u}$ distributed according to the target distribution $\boldsymbol{s} \sim \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$

The key step of the animation approach is step 2, while steps 1 and 3 make it applicable to any Gaussian distribution. $\mathcal{N}(\boldsymbol{u}, \boldsymbol{0}, \boldsymbol{I})$ is a spherically symmetric distribution. To draw samples one could also draw a random sample for the radius $\boldsymbol{r}$ (Fig. 3a, length of green vector) from a standard Gaussian and draw $\boldsymbol{u}$ uniformly at random on the $(n-1)$-manifold (Fig. 3a, grey sphere) defined by $\boldsymbol{r}$. All points lying on this $(m-1)$-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 $(m-1)$-dimensional space, time could be used as an extra dimension to traverse through the space. Therefore, the animation loops through a one-dimensional orbit of the $(m-1)$-manifold (Fig 3a, black circle), which is drawn uniformly at random and defined by a tangent $\boldsymbol{t}$ at point $\boldsymbol{u}_0$. The tangent is part of the Lie algebra of the Lie group's manifold. Tangents of length 0 to 2π (the number depends on the total number of frames f) are mapped through the exponential map to the orbit which build the samples for the animation. The set of standard normal samples $\boldsymbol{u}_1, ..., \boldsymbol{u}_f$ is subsequently scaled, rotated, and shifted according to the mean and covariance of the target distribution by $\boldsymbol{s}_i = \boldsymbol{\mu} + \boldsymbol{L}\boldsymbol{u}_i$ (Fig. 3b).

For the animation, samples are drawn from the distribution over the eigenvectors $p(\boldsymbol{U}|\boldsymbol{Y})$ (eq. 22) as described above and used to transform the input data accordingly:

$$\boldsymbol{S}_1 \sim p(\boldsymbol{U}|\boldsymbol{Y}) \rightarrow \boldsymbol{T}_1 = \boldsymbol{Y}\boldsymbol{S}_1 \rightarrow \text{frame } 1 \tag{26}$$

$$\boldsymbol{S}_2 \sim p(\boldsymbol{U}|\boldsymbol{Y}) \rightarrow \boldsymbol{T}_2 = \boldsymbol{Y}\boldsymbol{S}_2 \rightarrow \text{frame } 2 \tag{27}$$

$$\ldots$$

$$\boldsymbol{S}_f \sim p(\boldsymbol{U}|\boldsymbol{Y}) \rightarrow \boldsymbol{T}_f = \boldsymbol{Y}\boldsymbol{S}_f \rightarrow \text{frame } f \tag{28}$$

To visualize the uncertainty of the two-dimensional map, these samples build frames in an animated visualization. Depending on the uncertainty (covariance) of the eigenvectors the individual frames will differ more or less from each other. The animation smoothly connects the individual frames.

C. Evaluation of `VIPurPCA`

To evaluate the performance of `VIPurPCA` its accuracy and runtime were assessed in relation to Monte Carlo sampling. In contrast to `VIPurPCA` Monte Carlo sampling is an iterative algorithm, whose accuracy and runtime depend on the number of iterations. Therefore, comparative analyses were performed relative to the number of Monte Carlo iterations. To this end, an input distribution was simulated following a matrix normal distribution with a Kronecker product-factored covariance matrix. To ensure clear directions of eigenvectors, features $\boldsymbol{p}_i$ were scaled by i. In each Monte Carlo iteration, a matrix variate sample was drawn from this distribution and PCA is applied to compute a two-dimensional embedding of the data. The directions of the individual eigenvectors across all Monte Carlo iterations were aligned to estimate the first (mean) and second (covariance) order moment of the distribution over eigenvectors. Those estimates converged for an increasing number of iterations and were compared to those computed by `VIPurPCA` using the relative Frobenius norm E_{rel}, where θ represents either the first or second order moment:

$$E_{\text{rel}} = \frac{||\theta_{\text{MC}} - \theta_{\texttt{VIPurPCA}}||_{\text{F}}}{||\theta_{\texttt{VIPurPCA}}||_{\text{F}}}$$

D. Data and Software Availability

`VIPurPCA` (pronounced 'vip your PCA') is provided as a python package. Code and data are available at https://github.com/Integrative-Transcriptomics/VIPurPCA. Example animations can be viewed at https://integrative-transcriptomics.github.io/VIPurPCA/examples/.

V. Experiments and Results

We demonstrate `VIPurPCA` on three different example data sets. The first dataset, the students grades dataset by Denoueux and Masson [47], is an example of non-correlated uncertainty,

TABLE I
TEST RESULTS OF SIX STUDENTS [47] PROVIDED AS REAL NUMBERS, INTERVALS, OR QUALITATIVE STATEMENT.

	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]

arising from an imprecise grading scheme. The second dataset by Higuera et al. [48] contains protein expression levels of the mouse cortex for different experimental treatments. The uncertainty results from technical replicates of the experiment and is assumed to be potentially correlated. The third dataset [49] contains human gene expression levels and their uncertainties for different experimental conditions that were inferred from microarray Affymetrix chips using Bayesian methods. The results of `VIPurPCA` and Monte-Carlo sampling were compared on a simulated multivariate Gaussian dataset to estimate the accuracy of `VIPurPCA`. These simulations were also used to measure the performance of `VIPurPCA` concerning runtime and space complexity.

A. Student Grades Dataset

This simulated dataset was introduced by Denoueux and Masson [47] and consists of four marks from 0 to 20 obtained by six students in mathematics (M1 and M2) and physics (P1 and P2) (Tab. I). Some of the marks are not precise and are given as intervals represented by uniform distributions or linguistic labels like "fairly good" or "very bad" which are represented by trapezoidal distributions which are defined by

$$f(x|a,b,c,d) = \begin{cases} u(\frac{x-a}{b-a}) & a \le x < b \\ u & b \le x < c \\ u(\frac{d-x}{d-c}) & c \le x < d \\ 0 & \text{else} \end{cases} \tag{29}$$

where $a \le b \le c \le d$ and $u = 2(d + c - b - a)^{-1}$ [50]. The test result P1 of the student Tom is unknown and modeled by the normal distribution $\mathcal{N}(14, 5.7^2)$ as suggested by [13]. Means and variances were computed for individual test results depending on the given distribution (for details, see Table S1). Real valued marks are given a very small variance of 0.1. This example addresses uncorrelated uncertainties meaning that data points are spherically uncertain around the mean. After applying `VIPurPCA` several output features are obtained (Fig. 4). The mean (Fig. 4a) and variance (Fig. 4c) of the first eigenvalue have relatively high absolute values compared to the other three. However, it is striking that the (co-)variances (Fig. 4d) associated with the first eigenvector (Fig. 4b) are relatively low. This can be explained by the fact that the first principal component comprises a high proportion of the total variance of the data (74 %). The variance covered by the first principal component is so high that the small uncertainties of the data points don't have a big effect on the direction of the component. Directions of low variance are much more sensitive to uncertainties in the data. An interesting insight is also provided by the Jacobian of the PCA applied to the data mean (Fig. 4e). Individual derivatives of the Jacobian matrix represent the influence of the respective sample feature on the respective output feature. The absolute value of the derivatives depends on the locations of the samples in space but also on the orientation of the samples' features w.r.t. the orientation of the eigenvectors. In this example, the Jacobian (upper 4 rows) shows that perturbing any of the inputs will not change the direction of the first eigenvector that much. On the other hand, the direction of the second eigenvector (rows 5-8) is less stable to minor perturbations of the inputs.

Samples drawn from the computed distribution over the eigenvectors can be used to transform the mean input accordingly and project the data into lower dimensional space. Fig. 4f shows this distribution of projections as kernel density plots in comparison to standard PCA for two different combinations of principal components. The shape of the individual densities shows that samples are more uncertain in the direction of PC 2 compared to PC 1. This matches the finding of the low (co-)variances observed for the first eigenvector (Fig. 4d). The small size and closeness of the second and third eigenvalue result in a much more uncertain low-dimensional map (Fig. 4f). For this example, it becomes apparent that the visualization of the first and second PC is fairly reliable while looking at the plot for the second and third PC is not trustworthy at all. This shows the importance of considering the uncertainties in dimensionality reductions, as by only looking at the standard PCA result it is not clear how trustworthy the result is and that the reliability changes when looking at different principal components.

B. Mice Protein Expression Dataset

This real-world dataset is available at the UCI Machine Learning Repository [51] and was initially analyzed by Higuera et al. [48]. It contains expression levels of 77 proteins (features) from the nuclear fraction of the mouse cortex for a total of 72 mice (samples). The experiment was repeated 15 times. The mice have been assigned to 8 classes depending on their genetics and experimental treatment. For this work, we chose only one experimental treatment indicating if the mice were stimulated to learn (context shock (CS), no context shock (SC)) as a label for visualization purposes. After preprocessing the data as suggested by Higuera et al. [48] 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 as columns in a matrix such that rows are the variables and columns the replicates used to compute a covariance matrix. An important property of the resulting covariance matrix is that it has covariance entries different from zero. Using kernel density plots to show the uncertainty of the lower dimensional map is inappropriate for this example due to visual clutter. It becomes clear that this visualization technique is limited to examples with a low number of observations.

To solve the problem of visual clutter individual samples are traversed using time as an additional dimension.

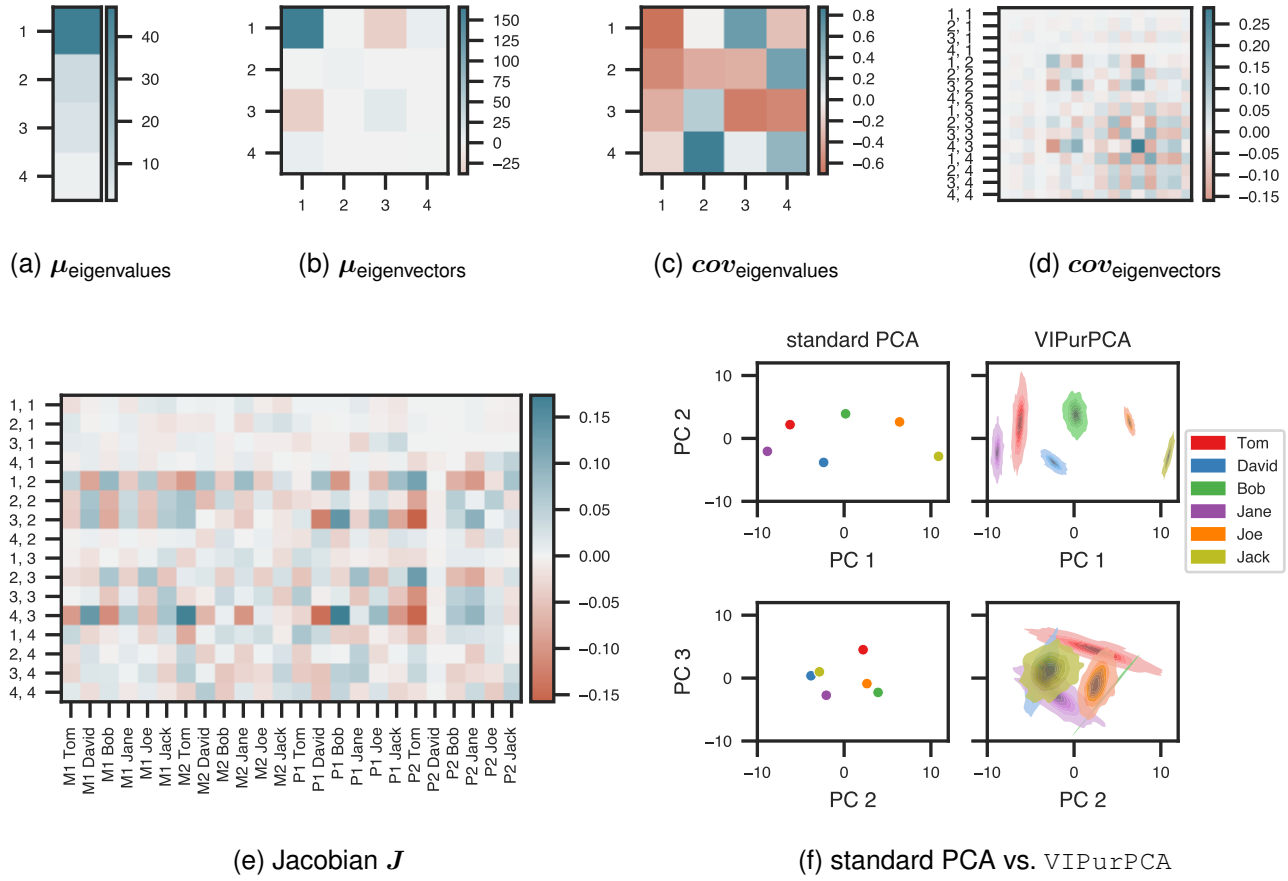


Fig. 4. Features obtained by VIPurPCA. While classical PCA only provides the mean of the eigenvalues (a) and eigenvectors (b), VIPurPCA provides uncertainty information of the mean eigenvalues and eigenvectors in terms of covariance matrices (c, d) according to Equation 25. (e) Applying VIPurPCA includes the computation of the full Jacobian of all output variables w.r.t. all inputs giving information about the influence of individual data points onto the result of the method. (d, e) The labels (y-axis) correspond to the index notation indicating the matrix elements of the mean eigenvector matrix (Fig. 4b). (f) Given the distribution over the eigenvectors (see Fig. 4b, 4d) samples of eigenvectors are randomly drawn and the mean input data is projected accordingly to get an intuition of the distribution. In comparison to conventional PCA, the outcome of VIPurPCA is shown as a kernel density estimate plot of the lower dimensional map. PC 1 vs. PC 2 as well as PC 2 vs. PC 3 are shown exemplarily.

The animation for the mice protein expression dataset can be found here (https://integrative-transcriptomics.github.io/VIPurPCA/examples/mice/). To get an intuition of the animation ten frames are shown as small multiples in Fig. 5. We highlighted two data points in green and pink which can be followed through all ten frames. The sample labeled in green moves the most in the shown dimensions but always stays within the blue class. The pink labeled sample on the other hand also greatly switches its position and thereby its possible class label (frame 4-6 in Fig. 5).

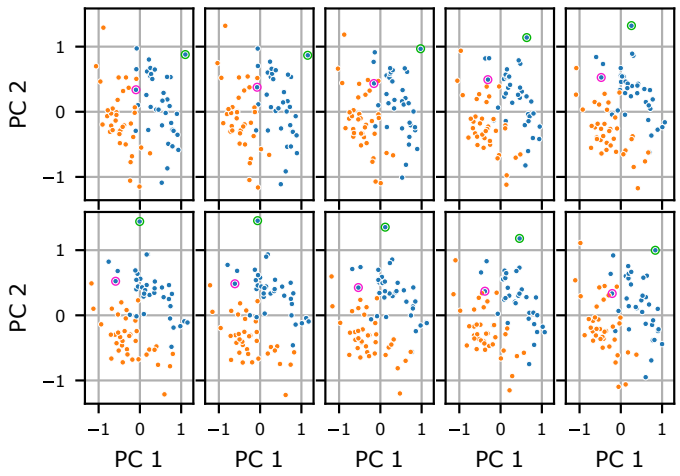


Fig. 5. Low dimensional maps (PC 1 vs. PC 2) of the mice dataset. The animation's frames (upper row: 1-5, lower row: 6-10) are shown as small multiples. A data point that is highly uncertain w.r.t. its position is labeled in green. The pink-tagged sample is uncertain w.r.t. its class membership and changes location from a blue to an orange neighborhood and back. The full animation is available online https://integrative-transcriptomics.github.io/VIPurPCA/examples/mice/.

C. Human Gene Expression Dataset

This microarray gene expression dataset [49] comes along with the Bioconductor package puma (propagating uncertainty in microarray analysis) [52]. It was obtained from eight Human Genome U95Av2 Arrays (HG-U95Av2) from Affymetrix. Estrogen (absent, present) and time (10h, 48h) are the two factors of interest in the 2×2 factorial-designed experiment, with two replicates for each combination of factors. Bayesian methods [53] applied to the Affymetrix GeneChips data provide gene expression levels along with uncertainty estimates. Out of

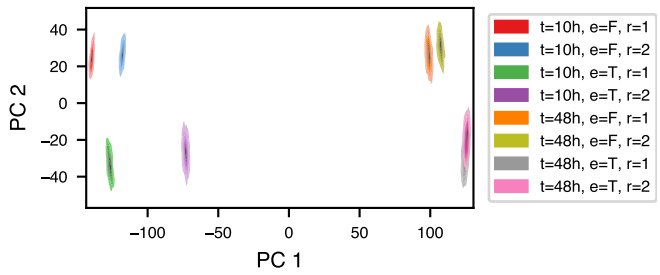


Fig. 6. Kernel density estimation plot showing the result of `VIPurCPA` applied to the human gene expression dataset. Time (10h, 48h) and estrogen (present=T, absent=F) were the experimental factors of interest. Two replicates were measured for each combination of factors.

12,625 genes, the 10% most variant genes were selected and subjected to `VIPurPCA`. As seen in Fig. 6, PC 1 separates time points and PC 2 separates estrogen availability. The extension of the individual densities mainly in the direction of PC2 indicates that the lower dimensional representation is mainly uncertain in this direction.

D. Evaluation of the Performance

The accuracy and computational costs of `VIPurPCA` were assessed in comparison to Monte Carlo sampling. Due to the iterative nature of Monte Carlo sampling comparative analyses were jointly done in relation to the number of Monte Carlo iterations. `VIPurPCA` and Monte Carlo sampling were applied to simulated matrix-variate distributions of different sizes and the resulting distributions of the two leading eigenvectors analyzed w.r.t. the relative Frobenius norm of the respective means and covariance matrices (Fig. 7). For each dataset, there exists a critical number of iterations at which Monte Carlo sampling becomes more time intensive than `VIPurPCA` when adding more iterations (see Fig. 7, intersection of orange lines). At these points, the estimation of the mean eigenvectors (Fig. 7, blue dashed line) by Monte Carlo sampling has converged for some datasets. However, their estimated uncertainty in terms of the covariance matrix (Fig. 7, blue line) has not converged yet for almost all datasets. A qualitative comparison of the visualizations of uncertain low-dimensional maps computed by Monte Carlo sampling and `VIPurPCA` shows that the estimated uncertainties of individual low-dimensional data points are very similar (Figure S1). An extensive analysis of the runtime and memory consumption of `VIPurPCA` is available in Figures S2 and S3, respectively.

E. Evaluation of the Visualization

In this work, uncertainty is modeled probabilistically resulting in a distribution over directions of high variance in the data. Drawing samples from this distribution and projecting the data accordingly results in a distribution over embeddings that indicates the uncertainty of the low-dimensional representation of the data and that provides the same kind of visualization users usually encounter when inspecting their PCA results. Using a simulated high-dimensional uncertain dataset of four clusters several visualization possibilities of hypothetical embedding outcomes were compared: showing the union of all samples as a point cloud in one scatterplot (Fig. 8a), using a density plot to aggregate uncertainty information of individual data points (Fig. 8b), or displaying hypothetical outcomes as small multiples (Fig. 8c) or as an animation (https://github.com/Integrative-Transcriptomics/VIPurPCA/blob/gh-pages/_includes/animation.gif). All views show that all four clusters are uncertain. However, the point cloud and density view do not show whether the uncertainties of samples from the blue and orange cluster correlate and therefore do not indicate whether the blue and orange clusters are separable given the uncertainties. The small multiple view and the animation show that the uncertainties of samples from the blue and orange clusters are anti-correlated in the direction of PC 2 and the clusters are in fact not separable given the uncertainties.

VI. Discussion and Conclusion

Dimensionality reduction techniques enable the classification, visualization, and compression of high-dimensional data. Including uncertainty quantification and visualization of the lower dimensional map enhances its interpretability and trustworthiness. In this work, we introduce `VIPurPCA`, which propagates uncertainties through PCA, the most commonly used dimensionality reduction technique. Assuming Gaussian distributed input and output distributions the computation of the outputs' covariance matrix boils down to linear algebra routines in the setting of Bayesian inference. Compared to methods like probabilistic principal component analysis or factor analysis also correlated uncertainties of the inputs can be handled. When the input uncertainties become zero, `VIPurPCA` returns identical results as common PCA applied to the input means. It is a common approach for uncertainty propagation to linearly approximate a function's outcome using a Taylor series expansion. It provides valid results if the function of interest is not too far from linear within one standard deviation from the mean, and if the input variabilities are relatively small [35]. Therefore, if the input uncertainties become too large, the computed output distribution might be inaccurate as a matter of principle. Using higher-order terms could improve the approximation, but includes the computation of a Hessian matrix of a vector-valued function, which in the case of PCA is unfeasible. Directions corresponding to eigenvalues close to zero are extremely sensitive to input variabilities and their order might also be precarious. Thus, their distribution is more likely to be incorrectly approximated by `VIPurPCA`. However, those low-variant dimensions are usually not kept, as already a few dimensions explain most of the variance due to an underlying principle explaining the input data.

`VIPurPCA` provides an approximate closed-form solution to the final distribution, which can be advantageous over Monte Carlo sampling for performance reasons. In particular, when Monte Carlo methods are applied to complex datasets, many samples may be required for convergence. Computing a PCA for all of these samples can be time consuming. Although `VIPurPCA` computes an approximation to the final distribution, we have shown that the relative error is small compared

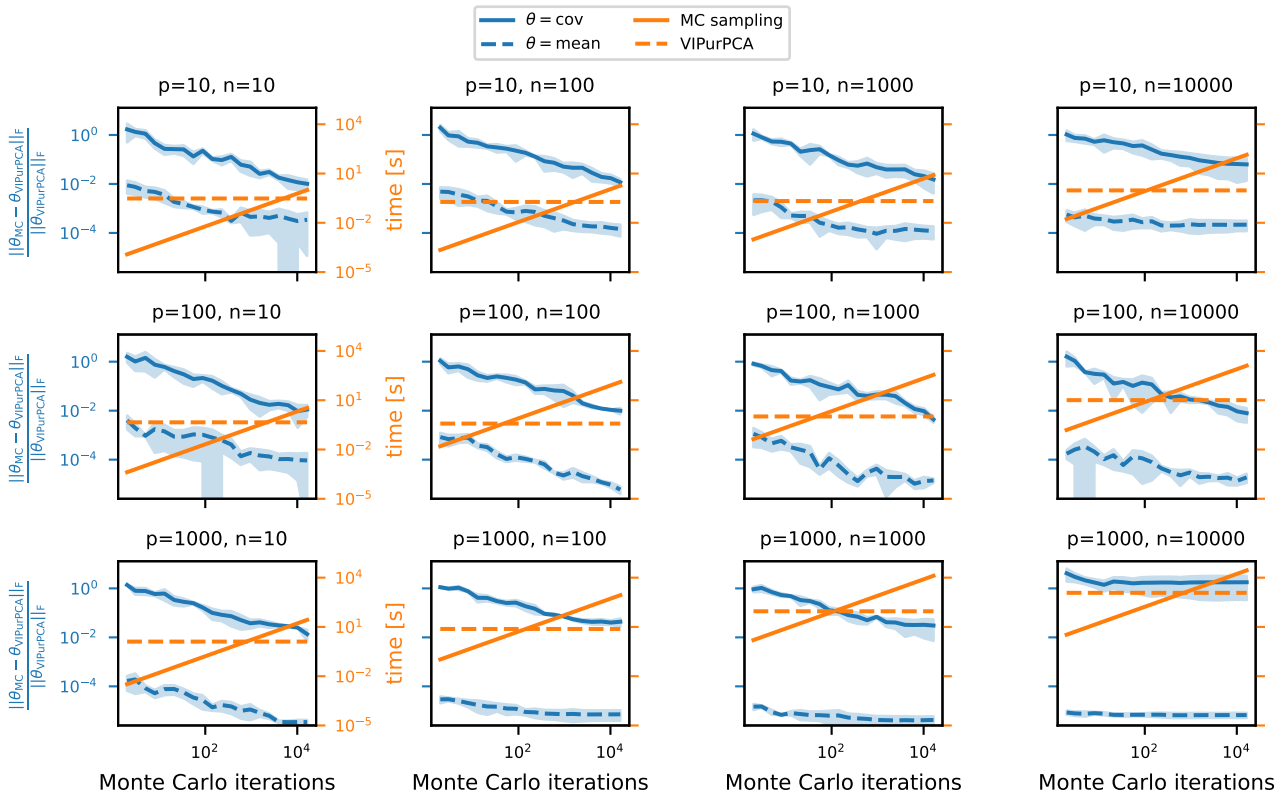


Fig. 7. Performance evaluation of VIPurPCA compared to Monte Carlo sampling. For simulated datasets of an increasing number of samples (columns) and dimensions (rows), the runtime (orange) of VIPurPCA and Monte Carlo sampling is evaluated for an increasing number of Monte Carlo iterations. The runtime of VIPurPCA is not dependent on the Monte Carlo iterations and indicated as an orange dashed line as a reference. In addition, the relative Frobenius norms (blue) 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.

to Monte Carlo sampling. For the simulated data, when comparing the first and second moments of the final distribution calculated by Monte Carlo sampling and VIPurPCA, the relative error decreases with an increasing number of Monte Carlo iterations (Fig. 7). 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 seen in Figure 7 (solid blue line), VIPurPCA approximates the uncertainties to around 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. A visual comparison of resulting uncertain maps for both methods is given in Figure S1. Since it depends on the number of iterations, the runtime of Monte Carlo sampling is difficult to compare with the runtime of VIPurPCA. Depending on the number of dimensions p and samples n, for a certain number of iterations, Monte Carlo sampling becomes computationally more expensive than VIPurPCA (Fig. 7, intersection of orange lines). In general, runtime and memory consumption are highly dependent on the implementation and are mutually dependent. For VIPurPCA, the time (Fig. S2) and space complexity (Fig. S3) 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 only on the order of minutes.

Another focus of this work was to visualize the uncertainty of the lower dimensional map properly and intuitively. It is possible to visualize the covariance matrix of the eigenvectors directly for example as a heatmap [54]. However, the effect of individual entries of the covariance matrix on the stability of the lower dimensional map is not apparent. We, therefore, decided to rather show the structure of samples drawn from the distribution over eigenvectors by using those samples to transform the input data accordingly. Overall, showing the uncertainty of the lower dimensional map as an animation intends 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. being not clearly arranged, the animation can not fix this. When looking at the different visualization options (Fig. 8 and https://github.com/Integrative-Transcriptomics/VIPurPCA/blob/gh-pages/_includes/animation.gif), various individual advantages and disadvantages become apparent. The point cloud (Fig. 8a) and the density plot (Fig. 8b) are capable of displaying many samples such that the entire distribution of potential embedding locations for each subject

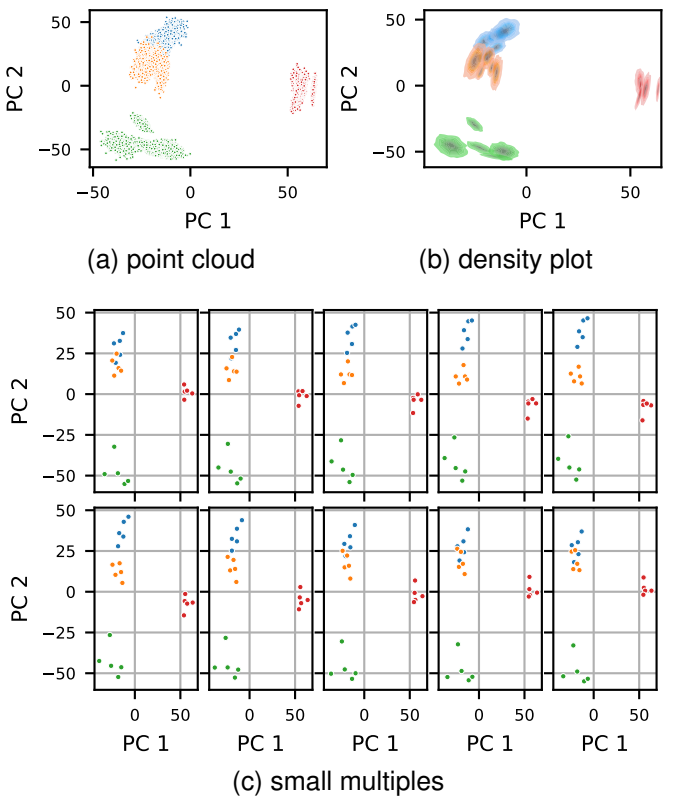


Fig. 8. Visualization possibilities using samples from the distribution which models the uncertainty of the embedding. (a) Individual samples are overlaid as points in a combined scatterplot. (b) Points belonging to the same subject are summarized as a density. (c) Individual samples are shown sequentially as small multiples.

is visible. While the point cloud can represent outliers, the density plot reduces visual clutter, though not completely. Nevertheless, both visualization forms show the union of all samples, and the structure of individual samples is lost as a result. However, this is especially important with overlapping point clouds/density curves to determine if the subjects are actually overlapping or oscillating together. The small multiples visualization (Fig. 8c) or the animation (https://github.com/Integrative-Transcriptomics/VIPurPCA/blob/gh-pages/_includes/animation.gif) 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 [30]), a smooth transition between frames is possible and individual data points can be tracked, especially in the animation. While the small multiples visualization needs more space, the animation has the disadvantage of not being able to be printed. How well the uncertainty of the low-dimensional map is perceived by the user in any of the mentioned visualization techniques depends on several properties of the data including the number of samples, the amount of uncertainty that is 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 the perception of uncertainty for differently structured datasets in detail. `VIPurPCA` also visualizes the Jacobian matrix, which indicates the influence of individual features and samples on the outcome of the PCA, as a heatmap. Input features or samples with a large impact on the outcome can thereby be easily identified.

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 not only depends 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 might be questionable (see for example Fig. 5, pink sample). For the human gene expression dataset `VIPurPCA` helped to identify that the direction in the data splitting samples by time (PC1) is stable, while the direction that splits samples by estrogen availability is more uncertain.

To simplify the use of `VIPurPCA` for a broad range of scientists we plan to release `VIPurPCA` as a web tool. By that, additional possibilities to interact with the data and visualizations could further improve the interpretability of the data. Despite that PCA is effectively one of the most commonly used dimensionality reduction techniques, nonlinear methods such as t-SNE [55] and UMAP [56] are favored when the data is distributed near a low-dimensional nonlinear manifold. Therefore, future work could include applying our approach of uncertainty propagation and visualization to those methods.

Acknowledgments

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany´s Excellence Strategy – EXC-Nummer 2064/1 – Projektnummer 390727645.

References

[1] R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley, "Deep learning for healthcare: review, opportunities and challenges," *Briefings in bioinformatics*, vol. 19, no. 6, p. 1236—1246, November 2018. [Online]. Available: https://europepmc.org/articles/PMC6455466

[2] Y. Hasin, M. Seldin, and A. Lusis, "Multi-omics approaches to disease," *Genome biology*, vol. 18, no. 1, pp. 1–15, 2017.

[3] 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*, vol. 42, pp. 60–88, 2017.

[4] 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*, vol. 2, no. 11, pp. 559–572, 1901.

[5] 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*, vol. 7, no. 39, p. eabi7673, 2021.

[6] J. A. Lee and M. Verleysen, *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007.

[7] L. K. Saul, K. Q. Weinberger, F. Sha, J. Ham, and D. D. Lee, "Spectral methods for dimensionality reduction." *Semi-supervised learning*, vol. 3, 2006.

[8] L. Van Der Maaten, E. Postma, and J. Van den Herik, "Dimensionality reduction: a comparative review," *J Mach Learn Res*, vol. 10, no. 66-71, p. 13, 2009.

[9] 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*, vol. 25, no. 8, pp. 2650–2673, 2018.

[10] C. Spearman, "General intelligence objectively determined and measured," *American Journal of Psychology*, vol. 15, pp. 107–197, 1904.

[11] W. S. Torgerson, "Multidimensional scaling: I. theory and method," *Psychometrika*, vol. 17, no. 4, pp. 401–419, 1952.

[12] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 61, no. 3, pp. 611–622, 1999.

[13] J. Görtler, T. Spinner, D. Streeb, D. Weiskopf, and O. Deussen, "Uncertainty-aware principal component analysis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 822–831, 2019.

[14] Y.-H. Chan, C. D. Correa, and K.-L. Ma, "Flow-based scatterplots for sensitivity analysis," in *2010 IEEE Symposium on Visual Analytics Science and Technology*. IEEE, 2010, pp. 43–50.

[15] ——, "The generalized sensitivity scatterplot," *IEEE transactions on visualization and computer graphics*, vol. 19, no. 10, pp. 1768–1781, 2013.

[16] R. Faust, D. Glickenstein, and C. Scheidegger, "Dimreader: Axis lines that explain non-linear projections," *IEEE transactions on visualization and computer graphics*, vol. 25, no. 1, pp. 481–490, 2018.

[17] K. Pöthkow, B. Weber, and H.-C. Hege, "Probabilistic marching cubes," in *Computer Graphics Forum*, vol. 30, no. 3. Wiley Online Library, 2011, pp. 931–940.

[18] T. Athawale and A. Entezari, "Uncertainty quantification in linear interpolation for isosurface extraction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2723–2732, 2013.

[19] 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*, vol. 27, no. 2, pp. 1797–1807, 2020.

[20] D. Spiegelhalter, M. Pearson, and I. Short, "Visualizing uncertainty about the future," *science*, vol. 333, no. 6048, pp. 1393–1400, 2011.

[21] P. Levontin and J. L. Walton, *Visualising uncertainty a short introduction*. Sad Press and Friends, 2020.

[22] S. Deitrick, "Evaluating implicit visualization of uncertainty for public policy decision support," in *Proc. AutoCarto*, 2012.

[23] 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*, vol. 51, no. 4, pp. 372–386, 2014.

[24] M. Matthews, L. Rehak, J. Famewo, T. Taylor, and J. Robson, "Evaluation of new visualization approaches for representing uncertainty in the recognized maritime picture," HUMANSYSTEMS INC GUELPH (ONTARIO), Tech. Rep., 2008.

[25] M. Skeels, B. Lee, G. Smith, and G. G. Robertson, "Revealing uncertainty for information visualization," *Information Visualization*, vol. 9, no. 1, pp. 70–81, 2010.

[26] D. Weiskopf, "Uncertainty visualization: Concepts, methods, and applications in biological data visualization," *Frontiers in Bioinformatics*, p. 10, 2022.

[27] 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*, vol. 23, no. 1, pp. 531–540, 2016.

[28] 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*, vol. 10, no. 11, 2015. [Online]. Available: http://idl.cs.washington.edu/papers/hops

[29] 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*, vol. 28, no. 1, pp. 443–453, 2021.

[30] P. Hennig, "Animating samples from Gaussian distributions," Max Planck Institute for Intelligent Systems, Spemannstraße, 72076 Tübingen, Germany, Technical Report 8, September 2013.

[31] R. L. Iman and J. C. Helton, "An investigation of uncertainty and sensitivity analysis techniques for computer models," *Risk analysis*, vol. 8, no. 1, pp. 71–90, 1988.

[32] S. H. Lee and W. Chen, "A comparative study of uncertainty propagation methods for black-box-type problems," *Structural and Multidisciplinary Optimization*, vol. 37, no. 3, p. 239, 2009.

[33] H. Janssen, "Monte-carlo based uncertainty analysis: Sampling efficiency and sampling convergence," *Reliability Engineering & System Safety*, vol. 109, pp. 123–132, 2013.

[34] D. A. Pérez, H. Gietler, and H. Zangl, "Automatic uncertainty propagation based on the unscented transform," in *2020 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*. IEEE, 2020, pp. 1–6.

[35] R. Schenkendorf, "A general framework for uncertainty propagation based on point estimate methods," Ph.D. dissertation, 2014.

[36] B. Ochoa and S. Belongie, "Covariance propagation for guided matching," in *Proceedings of the Workshop on Statistical Methods in Multi-Image and Video Processing (SMVP)*, vol. 83, 2006.

[37] H. Hotelling, "Analysis of a complex of statistical variables into principal components." *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[38] S. Barratt, "A matrix Gaussian distribution," *arXiv preprint arXiv:1804.11010*, 2018.

[39] A. K. Gupta and D. K. Nagar, *Matrix variate distributions*. Chapman and Hall/CRC, 2018.

[40] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: a survey," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 5595–5637, 2017.

[41] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[42] 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," in *NIPS-W*, 2017.

[43] D. Maclaurin, "Modeling, inference and optimization with composable differentiable procedures," dissertation, Harvard University, 2016.

[44] M. Seeger, A. Hetzel, Z. Dai, E. Meissner, and N. D. Lawrence, "Auto-differentiating linear algebra," *arXiv preprint arXiv:1710.08717*, 2017.

[45] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, and S. Wanderman-Milne, "JAX: composable transformations of Python+NumPy programs," 2018. [Online]. Available: http://github.com/google/jax

[46] P. Hennig, M. A. Osborne, and H. P. Kersting, *Probabilistic Numerics: Computation as Machine Learning*. Cambridge University Press, 2022, p. 24.

[47] T. Denoeux and M.-H. Masson, "Principal component analysis of fuzzy data using autoassociative neural networks," *IEEE Transactions on Fuzzy Systems*, vol. 12, no. 3, pp. 336–349, 2004.

[48] 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*, vol. 10, no. 6, p. e0129126, 2015.

[49] R. Pearson, *pumadata: Various data sets for use with the puma package*, 2020, r package version 2.26.0. [Online]. Available: http://umber.sbs.man.ac.uk/resources/puma

[50] J. R. van Dorp and S. Kotz, "Generalized trapezoidal distributions," *Metrika*, vol. 58, no. 1, pp. 85–97, 2003.

[51] D. Dua and C. Graff, "UCI machine learning repository," http://archive.ics.uci.edu/ml, 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[52] 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*, vol. 10, no. 1, pp. 1–10, 2009.

[53] X. Liu, M. Milo, N. D. Lawrence, and M. Rattray, "A tractable probabilistic model for affymetrix probe-level analysis across multiple chips," *Bioinformatics*, vol. 21, no. 18, pp. 3637–3644, 2005.

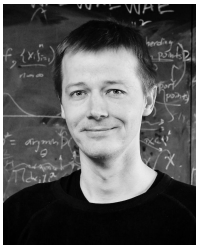
[54] X. Li, Z. Li, H. Zhou, S. M. Gaynor, Y. Liu, H. Chen, R. Sun, R. Dey, D. K. Arnett, S. Aslibekyan *et al.*, "Dynamic incorporation of multiple in silico functional annotations empowers rare variant association analysis of large whole-genome sequencing studies at scale," *Nature genetics*, vol. 52, no. 9, pp. 969–983, 2020.

[55] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[56] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

Susanne Zabel received both, the B.S. degree in biochemistry in 2015 and the M.S. degree in bioinformatics in 2019 from the University of Tübingen, Germany.

From 2018 to 2019, she was with the Department of Empirical Inference at the Max Planck Institute for Intelligent Systems, Tübingen, Germany. She is currently working at the Interfaculty Institute for Biomedical Informatics, University of Tübingen, Germany, as a Ph.D. candidate.



Philipp Hennig received the Diplom degree in physics from Heidelberg University, Germany, in 2007, and the Ph.D. degree from the University of Cambridge, UK, in 2011.

He was an external consultant to Microsoft Research Ltd., Cambridge, UK, from 2008 to 2010. From 2011 to 2015, he was a Research Scientist the Department of Empirical Inference at the Max Planck Institute for Intelligent Systems, Tübingen, Germany, where he became an Emmy Noether Group Leader in 2015, and an independent Max-Planck-Group Leader in 2016. In 2018, he was appointed to the Chair for the Methods of Machine Learning at the University of Tübingen. He remains an adjunct scientist at the Max Planck Institute for Intelligent Systems.

Prof. Dr. Philipp Hennig received an ERC Starting Grant in 2018.

Kay Nieselt received the diploma degree in mathematics from the University of Göttingen, Germany, and the doctoral degree in mathematics from the University of Bielefeld, Germany.

She was a Postdoctoral researcher as a Feodor-Lynen Scholar at the University of Auckland, New Zealand, at the University of Munich, Germany, and at the Max Planck Institute for Biophysical Chemistry, Göttingen, Germany. In 2002, she became a Group Leader for Bioinformatics at the Center for Bioinformatics Tübingen (now Institute for Bioinformatics and Medical Informatics), Germany, and is currently an Associate Professor for Bioinformatics at the University of Tübingen, Germany. Since 2021, she is Vice Dean of Studies of the Faculty of Science at the University of Tübingen, Germany. Her research interests include visualization of biological data and the analysis of multi-omics data.

Prof. Dr. Kay Nieselt is a member of the ISCB, GI, and the Steering Committee of BioVis, and she has been a member of the program committee for BioVis, EuroVis, ISMB and other conferences for many years.