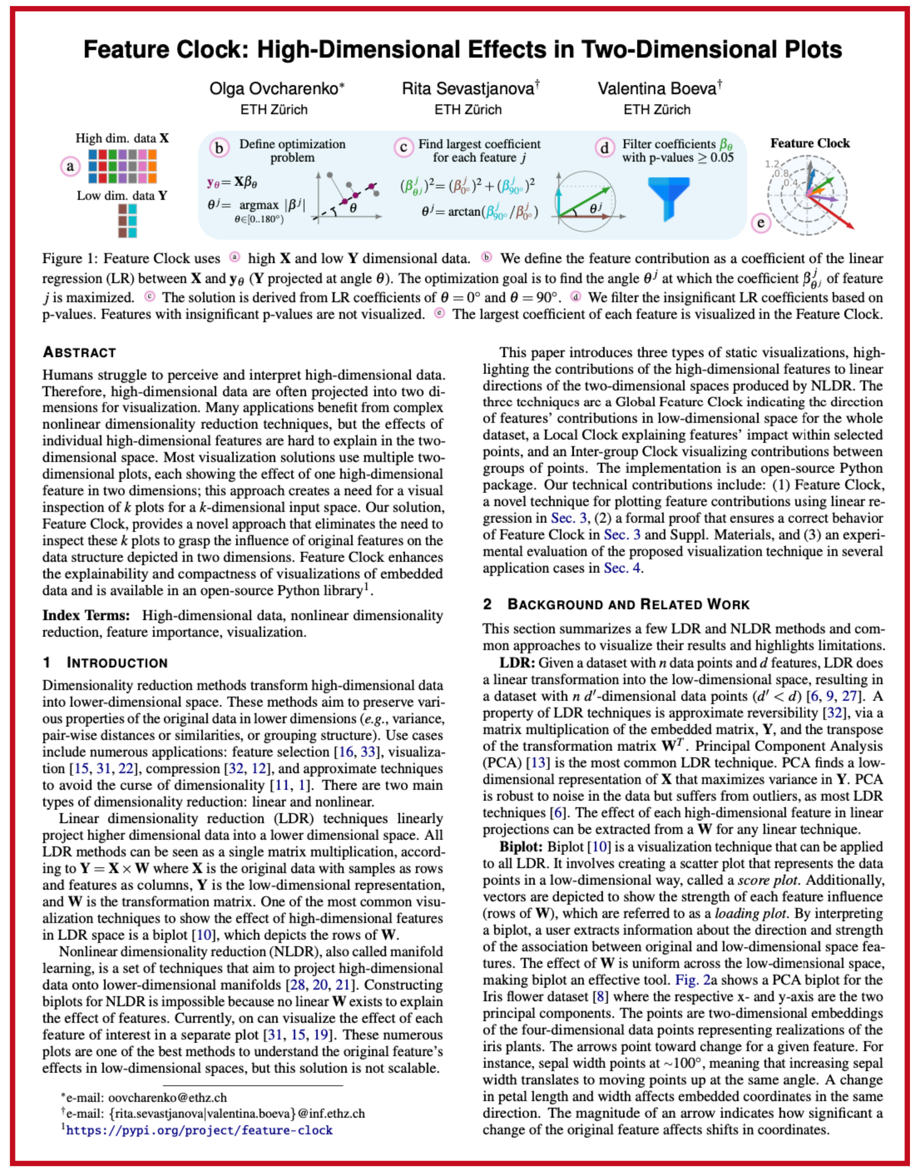
Feature Clock:
High-Dimensional Effects in Two-Dimensional Plots

Olga Ovcharenko Rita Sevastjanova Valentina Boeva

ETH zürich

VIS2024



Feature Clock: High-Dimensional Effects in Two-Dimensional Plots

Olga Ovcharenko* — ETH Zürich
Rita Sevastjanova† — ETH Zürich
Valentina Boeva† — ETH Zürich

Figure 1: Feature Clock uses (a) high **X** and low **Y** dimensional data. (b) We define the feature contribution as a coefficient of the linear regression (LR) between **X** and $\mathbf{y}_\theta$ (**Y** projected at angle θ). The optimization goal is to find the angle θ^j at which the coefficient $\beta^j_{\theta^j}$ of feature j is maximized. (c) The solution is derived from LR coefficients of $\theta = 0°$ and $\theta = 90°$. (d) We filter the insignificant LR coefficients based on p-values. Features with insignificant p-values are not visualized. (e) The largest coefficient of each feature is visualized in the Feature Clock.

Abstract

Humans struggle to perceive and interpret high-dimensional data. Therefore, high-dimensional data are often projected into two dimensions for visualization. Many applications benefit from complex nonlinear dimensionality reduction techniques, but the effects of individual high-dimensional features are hard to explain in the two-dimensional space. Most visualization solutions use multiple two-dimensional plots, each showing the effect of one high-dimensional feature in two dimensions; this approach creates a need for a visual inspection of k plots for a k-dimensional input space. Our solution, Feature Clock, provides a novel approach that eliminates the need to inspect these k plots to grasp the influence of original features on the data structure depicted in two dimensions. Feature Clock enhances the explainability and compactness of visualizations of embedded data and is available in an open-source Python library[1].

Index Terms: High-dimensional data, nonlinear dimensionality reduction, feature importance, visualization.

1 Introduction

Dimensionality reduction methods transform high-dimensional data into lower-dimensional space. These methods aim to preserve various properties of the original data in lower dimensions (*e.g.*, variance, pair-wise distances or similarities, or grouping structure). Use cases include numerous applications: feature selection [16, 33], visualization [15, 31, 22], compression [32, 12], and approximate techniques to avoid the curse of dimensionality [11, 1]. There are two main types of dimensionality reduction: linear and nonlinear.

Linear dimensionality reduction (LDR) techniques linearly project higher dimensional data into a lower dimensional space. All LDR methods can be seen as a single matrix multiplication, according to $\mathbf{Y} = \mathbf{X} \times \mathbf{W}$ where $\mathbf{X}$ is the original data with samples as rows and features as columns, $\mathbf{Y}$ is the low-dimensional representation, and $\mathbf{W}$ is the transformation matrix. One of the most common visualization techniques to show the effect of high-dimensional features in LDR space is a biplot [10], which depicts the rows of $\mathbf{W}$.

Nonlinear dimensionality reduction (NLDR), also called manifold learning, is a set of techniques that aim to project high-dimensional data onto lower-dimensional manifolds [28, 20, 21]. Constructing biplots for NLDR is impossible because no linear $\mathbf{W}$ exists to explain the effect of features. Currently, on can visualize the effect of each feature of interest in a separate plot [31, 15, 19]. These numerous plots are one of the best methods to understand the original feature's effects in low-dimensional spaces, but this solution is not scalable.

This paper introduces three types of static visualizations, highlighting the contributions of the high-dimensional features to linear directions of the two-dimensional spaces produced by NLDR. The three techniques are a Global Feature Clock indicating the direction of features' contributions in low-dimensional space for the whole dataset, a Local Clock explaining features' impact within selected points, and an Inter-group Clock visualizing contributions between groups of points. The implementation is an open-source Python package. Our technical contributions include: (1) Feature Clock, a novel technique for plotting feature contributions using linear regression in Sec. 3, (2) a formal proof that ensures a correct behavior of Feature Clock in Sec. 3 and Suppl. Materials, and (3) an experimental evaluation of the proposed visualization technique in several application cases in Sec. 4.

2 Background and Related Work

This section summarizes a few LDR and NLDR methods and common approaches to visualize their results and highlights limitations.

LDR: Given a dataset with n data points and d features, LDR does a linear transformation into the low-dimensional space, resulting in a dataset with n d'-dimensional data points ($d' < d$) [6, 9, 27]. A property of LDR techniques is approximate reversibility [32], via a matrix multiplication of the embedded matrix, $\mathbf{Y}$, and the transpose of the transformation matrix $\mathbf{W}^T$. Principal Component Analysis (PCA) [13] is the most common LDR technique. PCA finds a low-dimensional representation of $\mathbf{X}$ that maximizes variance in $\mathbf{Y}$. PCA is robust to noise in the data but suffers from outliers, as most LDR techniques [6]. The effect of each high-dimensional feature in linear projections can be extracted from a $\mathbf{W}$ for any linear technique.

Biplot: Biplot [10] is a visualization technique that can be applied to all LDR. It involves creating a scatter plot that represents the data points in a low-dimensional way, called a *score plot*. Additionally, vectors are depicted to show the strength of each feature influence (rows of $\mathbf{W}$), which are referred to as a *loading plot*. By interpreting a biplot, a user extracts information about the direction and strength of the association between original and low-dimensional space features. The effect of $\mathbf{W}$ is uniform across the low-dimensional space, making biplot an effective tool. Fig. 2a shows a PCA biplot for the Iris flower dataset [8] where the respective x- and y-axis are the two principal components. The points are two-dimensional embeddings of the four-dimensional data points representing realizations of the iris plants. The arrows point toward change for a given feature. For instance, sepal width points at ~100°, meaning that increasing sepal width translates to moving points up at the same angle. A change in petal length and width affects embedded coordinates in the same direction. The magnitude of an arrow indicates how significant a change of the original feature affects shifts in coordinates.

*e-mail: oovcharenko@ethz.ch
†e-mail: {rita.sevastjanova|valentina.boeva}@inf.ethz.ch
[1] https://pypi.org/project/feature-clock

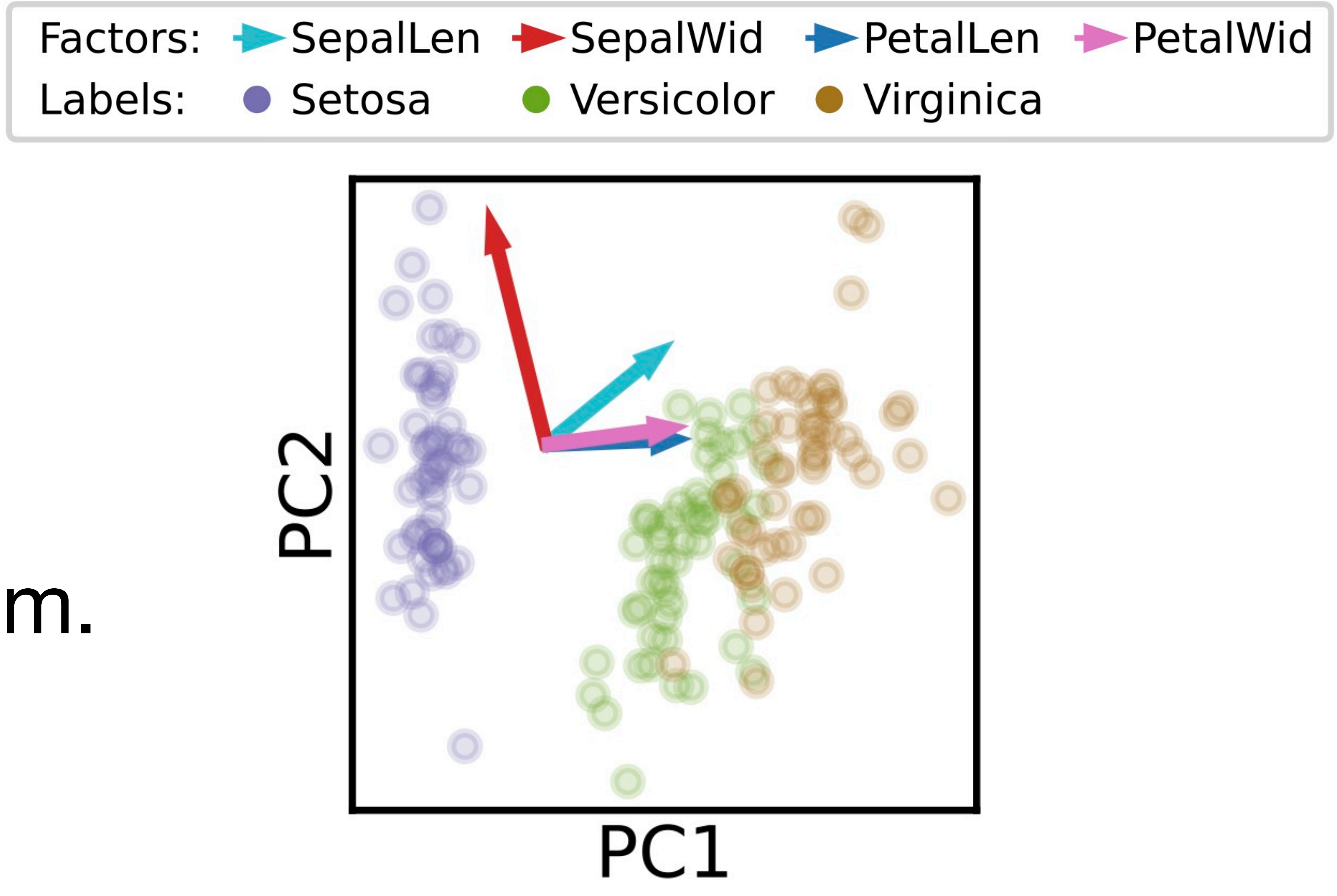
Motivation

Motivation

- Difficult to perceive high-dimensional data
- Use dimensionality reduction and visualize in 2D

Motivation

- Difficult to perceive high-dimensional data
- Use dimensionality reduction and visualize in 2D



Biplot for the Iris dataset

- **Linear dimensionality reduction**
  - Depicts a transformation matrix W
    - $Y = X \times W$ with X high-dim. and Y low-dim.
  - **Easy** to analyze
    - **Biplot** with feature loadings W

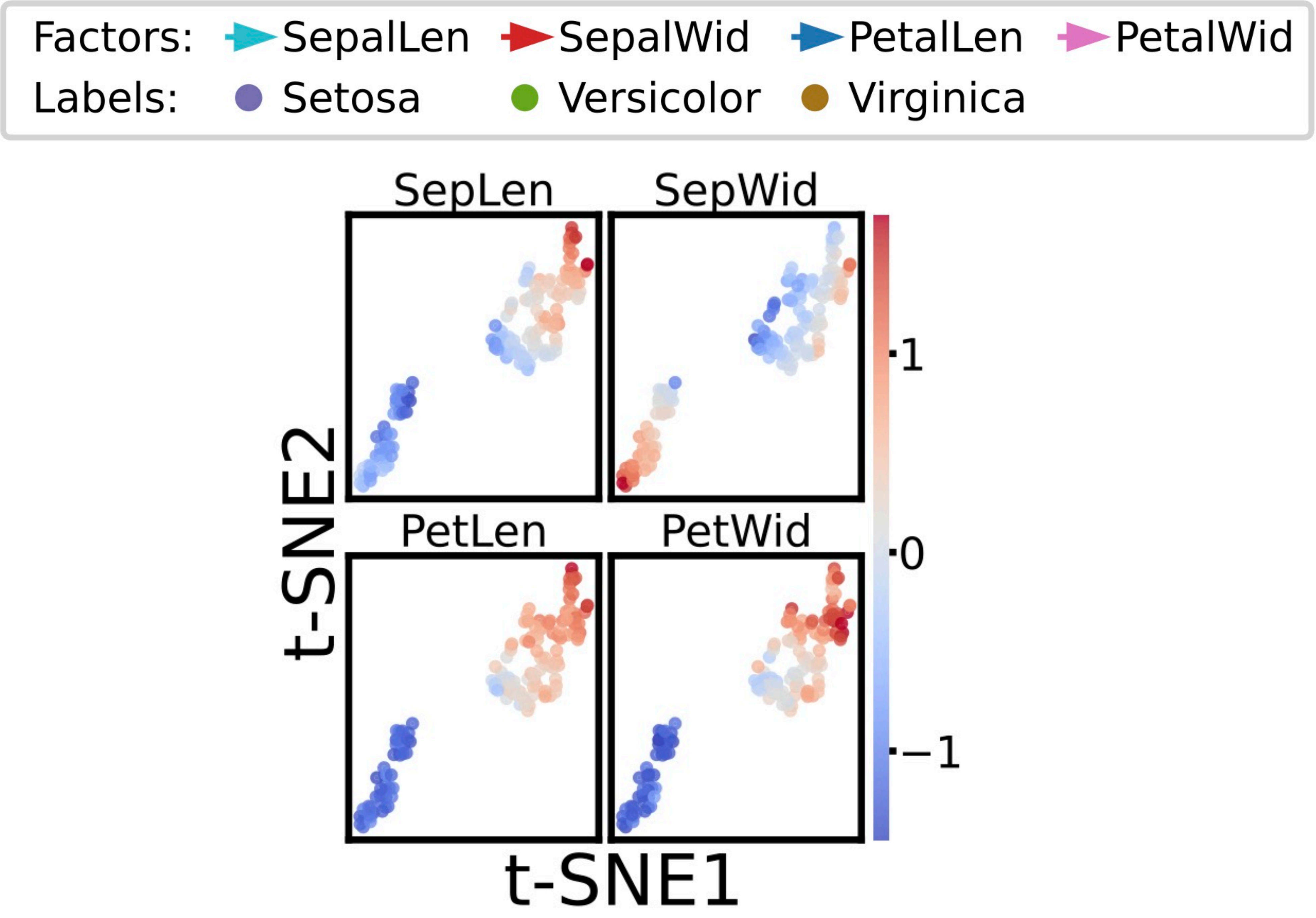
Motivation

- Linear projections can't capture complex structures

Motivation

- Linear projections can't capture complex structures

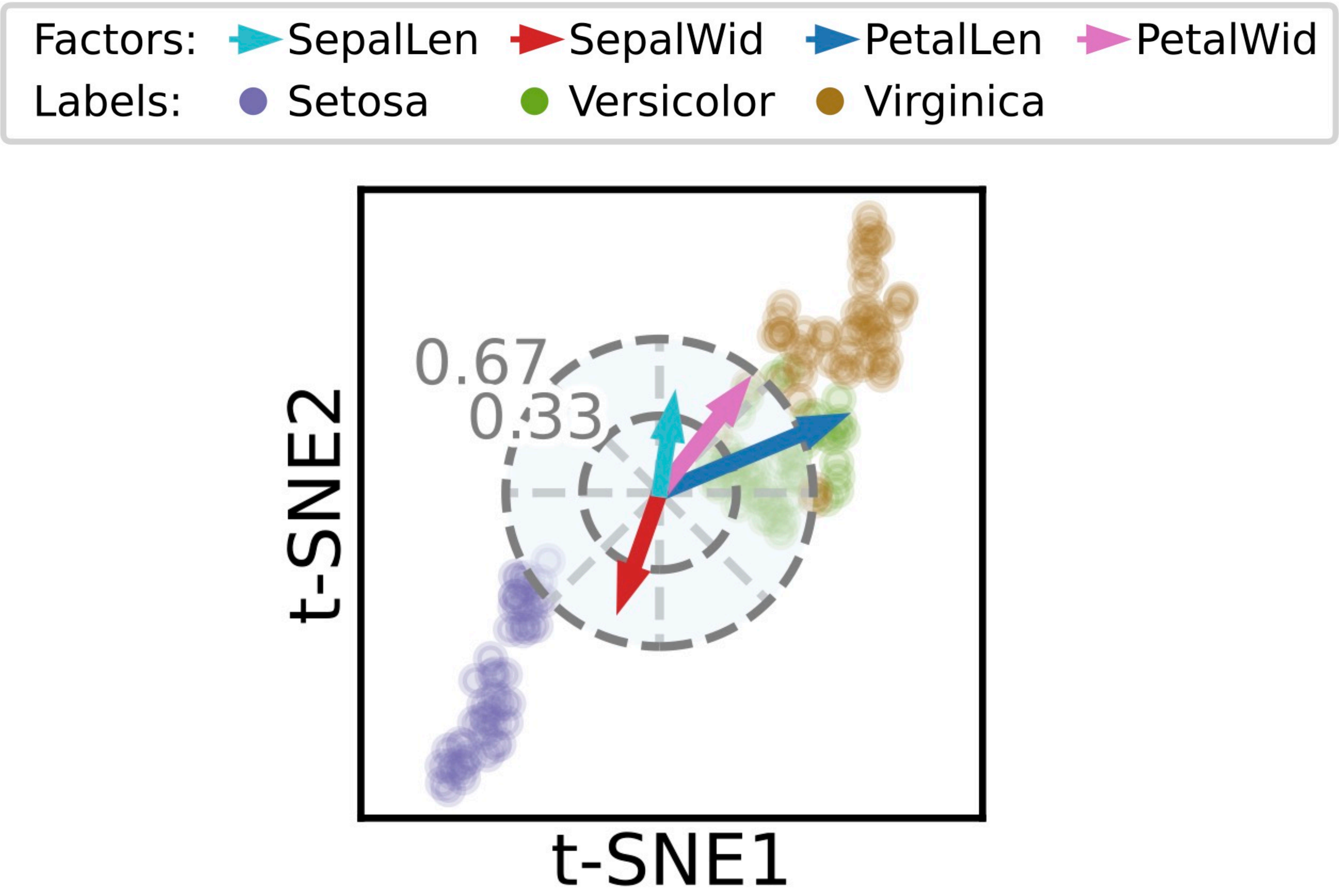
- **Nonlinear dimensionality reduction**
  - **Difficult** to analyze (**not scalable**)
    - There exists no W and **no biplot**
    - Visualize each feature in low-dim. space
  - Examples: t-SNE, UMAP, and PHATE



t-SNE projections for the Iris dataset

Motivation

Feature Clock shows all feature loadings in one plot without W for **linear** and **nonlinear** dimensionality reductions

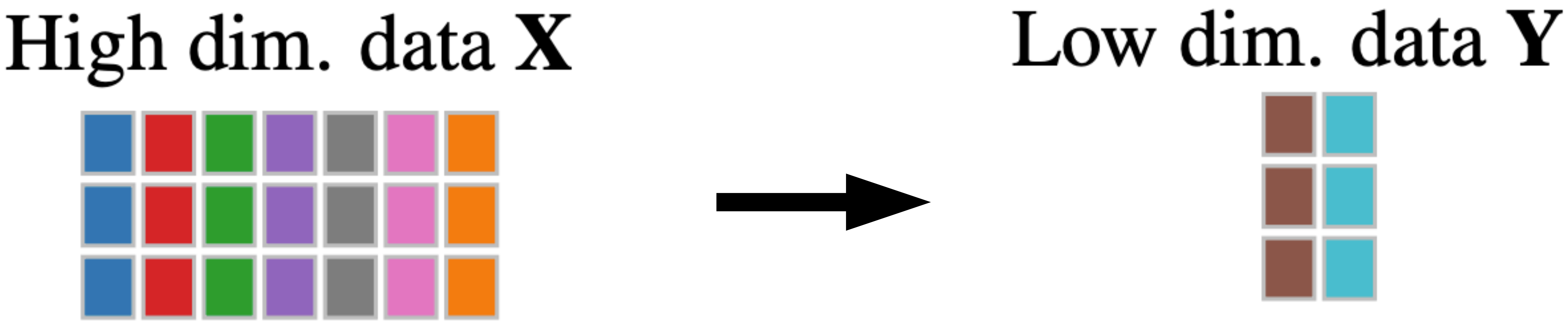


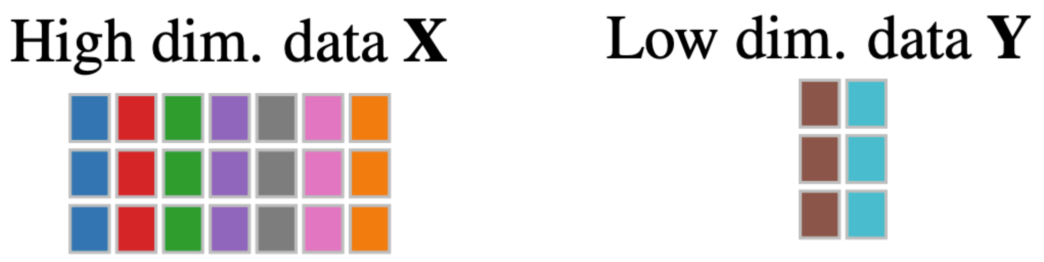
Feature Clock for the Iris dataset

Method

Method. Step 1

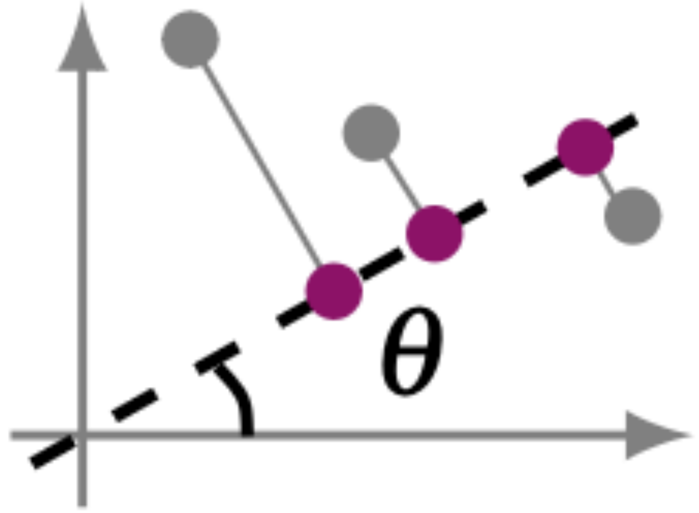
- Apply any dimensionality reduction (e.g., PCA, t-SNE, UMAP, autoencoder)



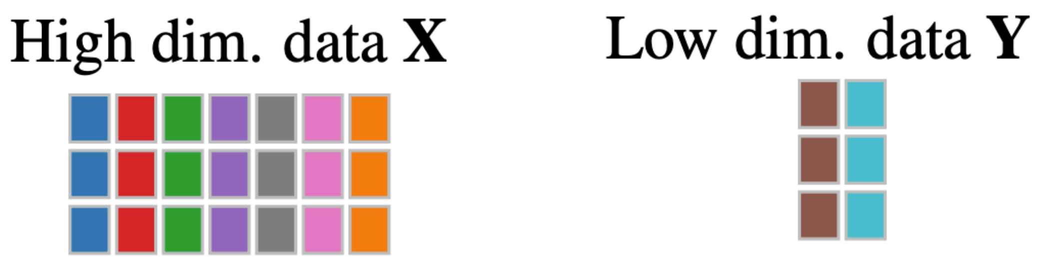


Method. Step 2

- **Coefficients** β_θ of the linear regression (LR) between X and y_θ measure feature effects in **direction** θ
  - y_θ = Y projected at angle θ

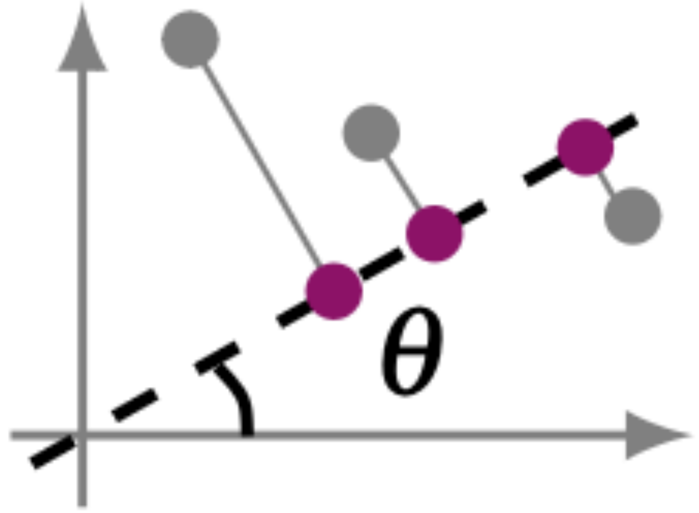


$$y_\theta = X\beta_\theta$$



Method. Step 2

- **Coefficients** β_θ of the linear regression (LR) between X and y_θ measure feature effects in **direction** θ
  - y_θ = Y projected at angle θ



$$y_\theta = X\beta_\theta$$

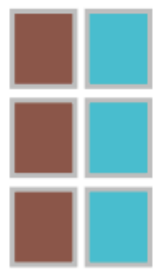
- Optimization problem
  - For each **feature j**, find angle θ^j at which coefficient β^j_θ is maximized

$$\boldsymbol{\theta}^j = \underset{\boldsymbol{\theta} \in [0..180^\circ)}{\operatorname{argmax}} |\boldsymbol{\beta}^j|$$

Method. Step 3

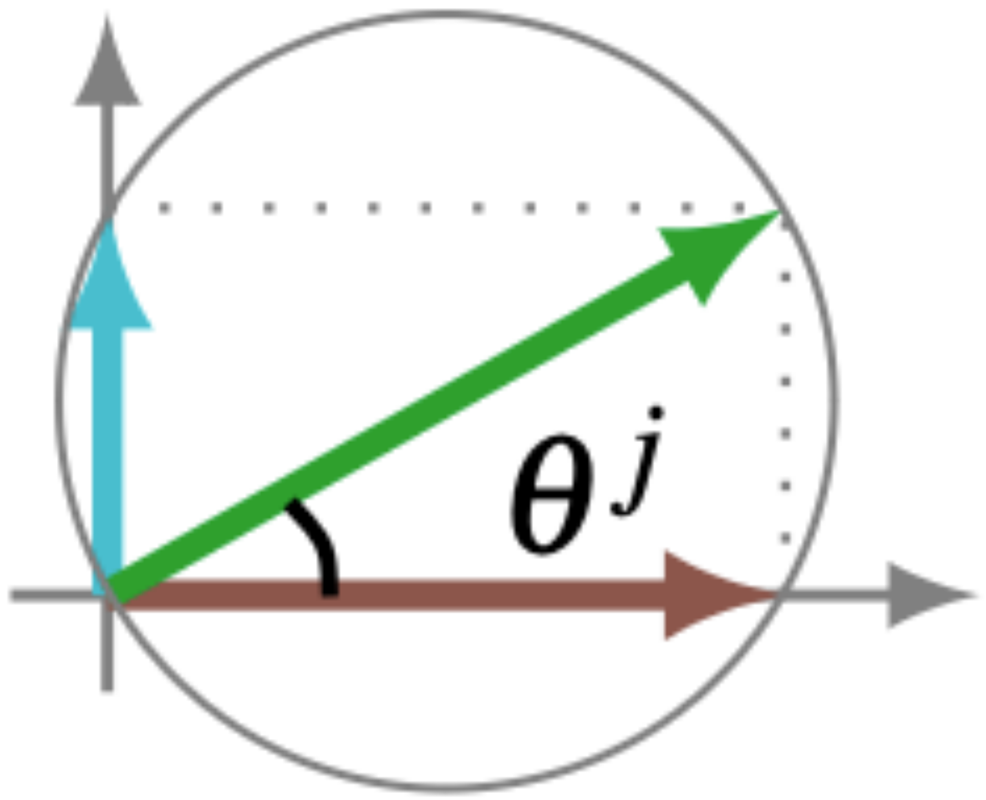
- Find largest coefficient for each **feature j**
  - Solution from Pythagoras theorem and two linear regression coefficients
    - β^j_θ biggest coefficient, β^j_0 at angle 0, β^j_{90} at angle 90

Low dim. data $\mathbf{Y}$



$$(\beta^j_{\theta^j})^2 = (\beta^j_{0^\circ})^2 + (\beta^j_{90^\circ})^2$$

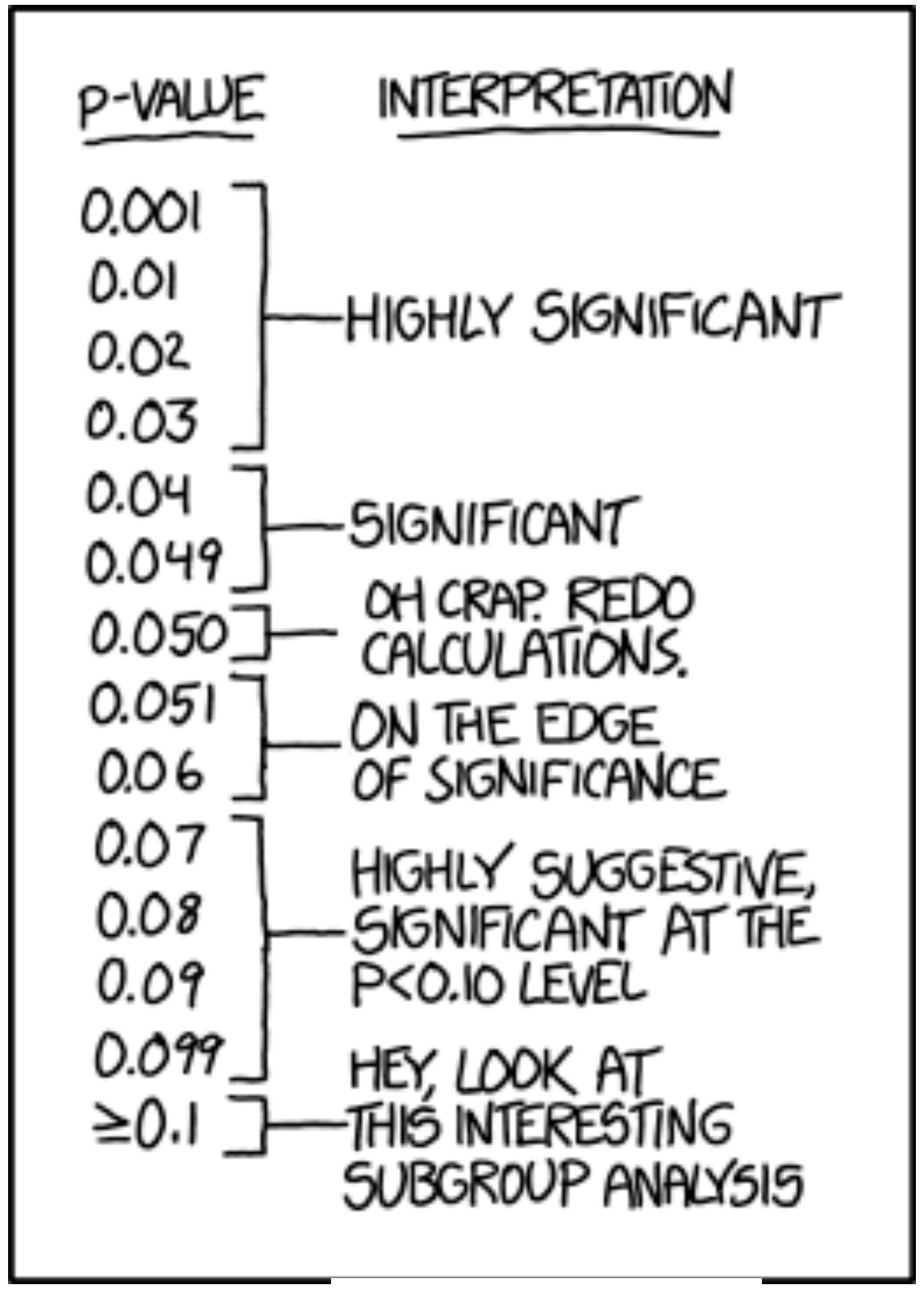
$$\theta^j = \arctan(\beta^j_{90^\circ} / \beta^j_{0^\circ})$$



- Fit **2** linear regressions for **exact** solution

Method. Step 4

- Filter insignificant coefficients using t-test and p-values
  - t-test checks the probability of coefficient equaling 0
  - Do not visualize β_{θ}^{j} with p-value $\geq$ 0.05



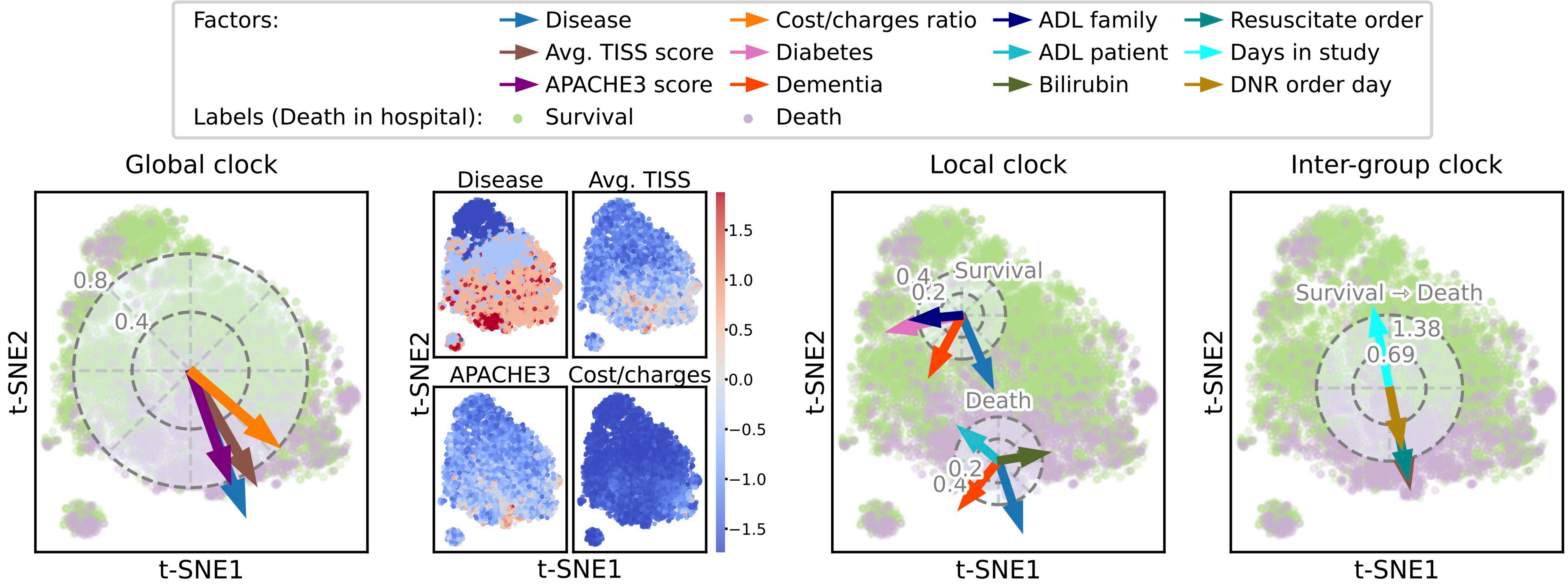
Design Choices

Design Choices

- To explore low-dimensional space at a finer granularity:
  - Global Feature Clock for all points
  - Local Clock for each class / selected points
  - Inter-group Clock to see changes between classes / selected points

Use Cases

Use Cases. Data Analysis



Inspect what differentiates surviving and dying patients in Support data

Summary

- Scalable, compact, intuitive technique that enhances explainability
- Mathematically proven
- Analysis of any nonlinear space
- Open-source installable PyPi package `pip install feature-clock`

Summary

- Scalable, compact, intuitive technique that enhances explainability
- Mathematically proven
- Analysis of any nonlinear space
- Open-source installable PyPi package `pip install feature-clock`

- Future work
  - Estimate the non-linear trajectory of gradients instead of linear regression
  - Dynamic visualization via GPU accelerator

Questions