Lecture 14

Dimensionality Reduction II: Feature Extraction

short version

STAT 479: Machine Learning, Fall 2019

Sebastian Raschka

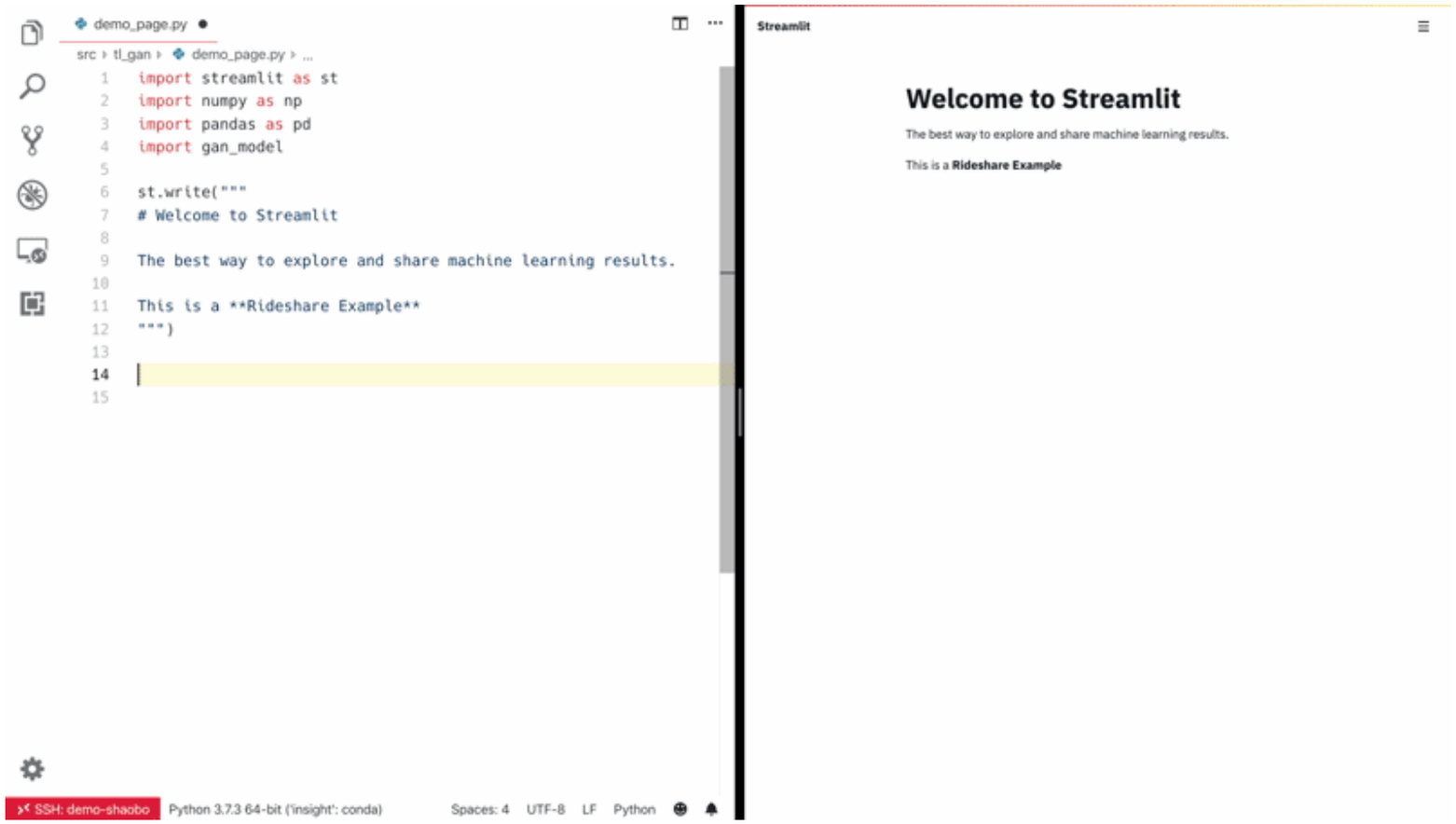
http://stat.wisc.edu/~sraschka/teaching/stat479-fs2019/

Tools for ML Tool Development

Welcome to Streamlit 👋

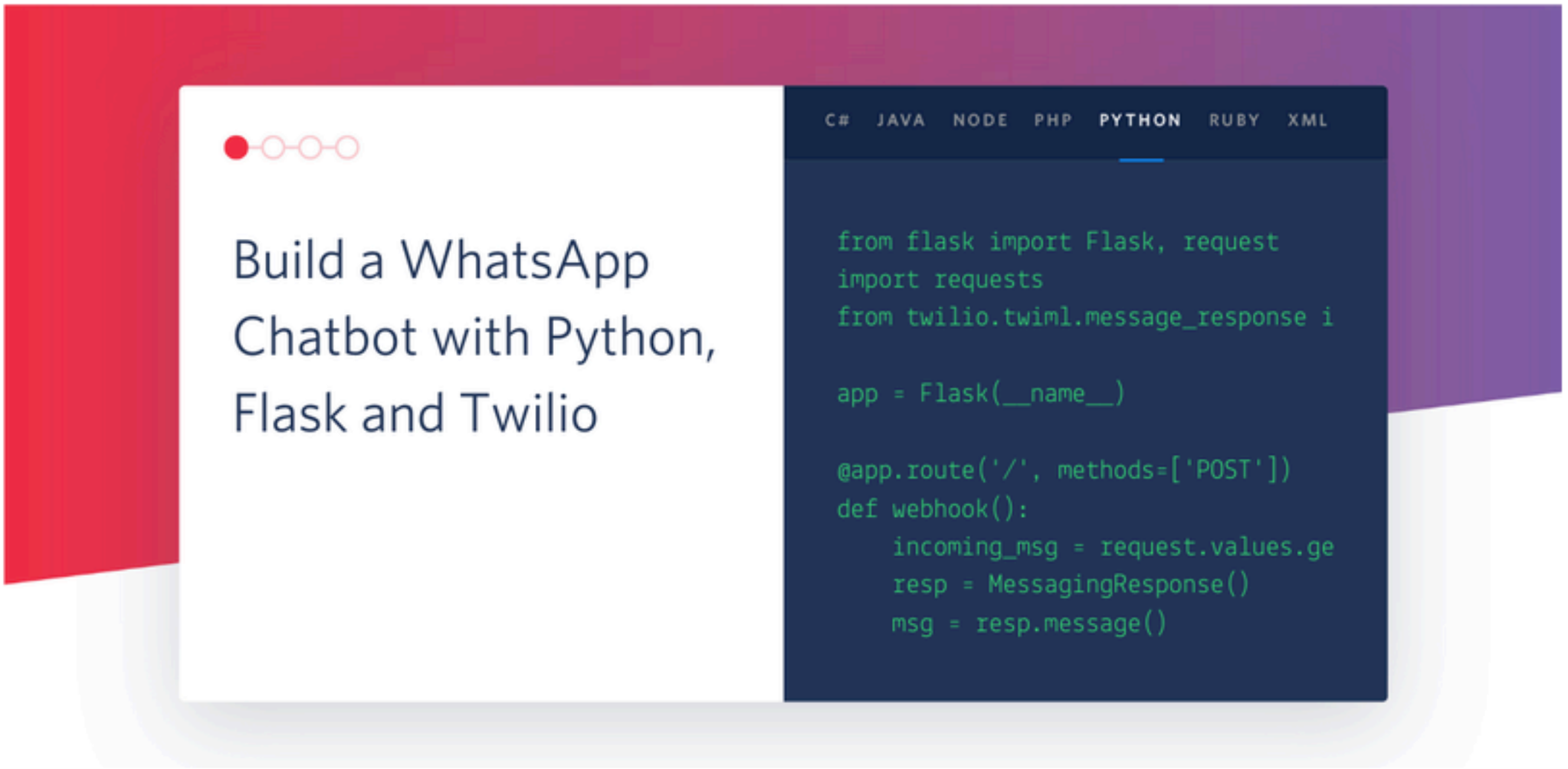
The fastest way to build custom ML tools.

Streamlit lets you create apps for your machine learning projects with deceptively simple Python scripts. It supports hot-reloading, so your app updates live as you edit and save your file. No need to mess with HTTP requests, HTML, JavaScript, etc. All you need is your favorite editor and a browser. Take a look at Streamlit in action:



https://github.com/streamlit/streamlit

Build a WhatsApp Chatbot With Python, Flask and Twilio



https://www.twilio.com/blog/build-a-whatsapp-chatbot-with-python-flask-and-twilio

Lecture 14

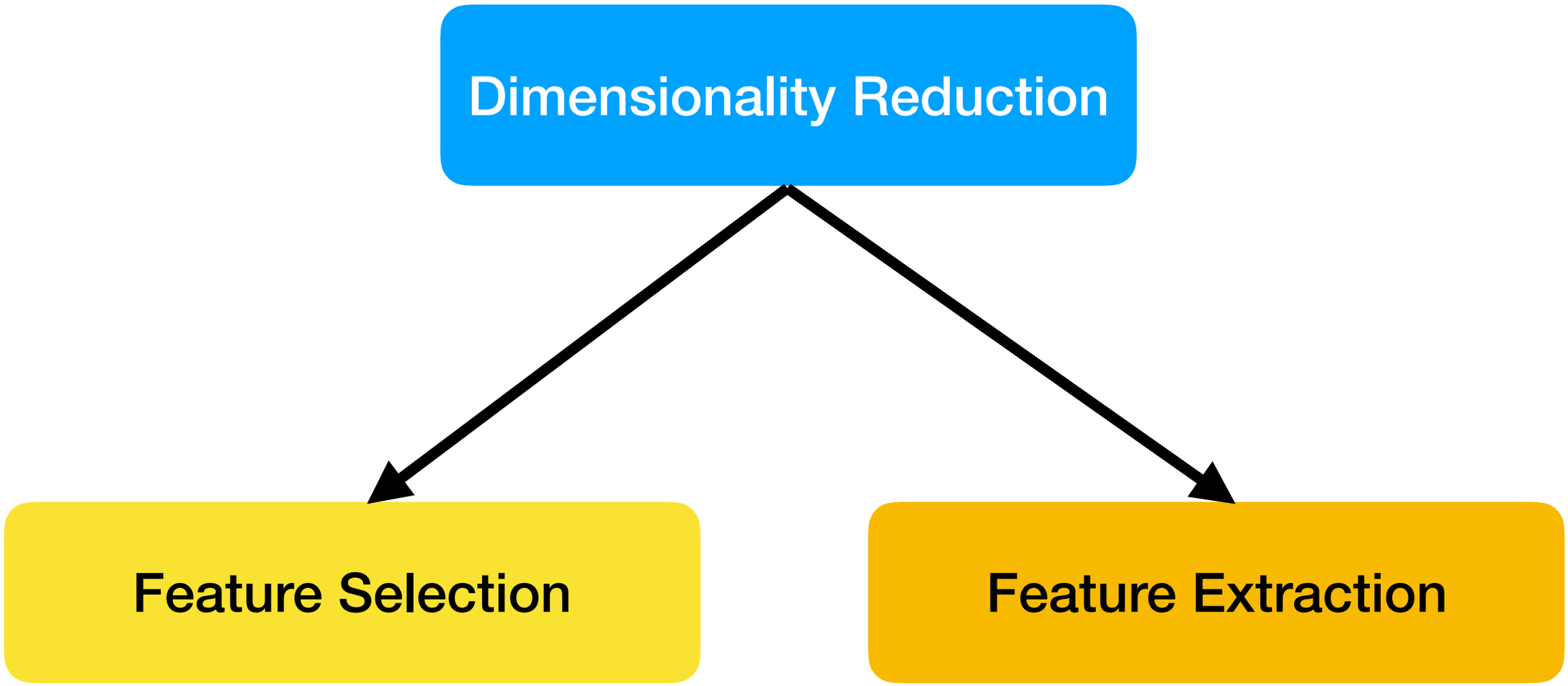
Dimensionality Reduction II: Feature Extraction

short version

STAT 479: Machine Learning, Fall 2019

Sebastian Raschka

http://stat.wisc.edu/~sraschka/teaching/stat479-fs2019/


Dimensionality Reduction
Feature Selection
Feature Extraction

Today

Dimensionality Reduction

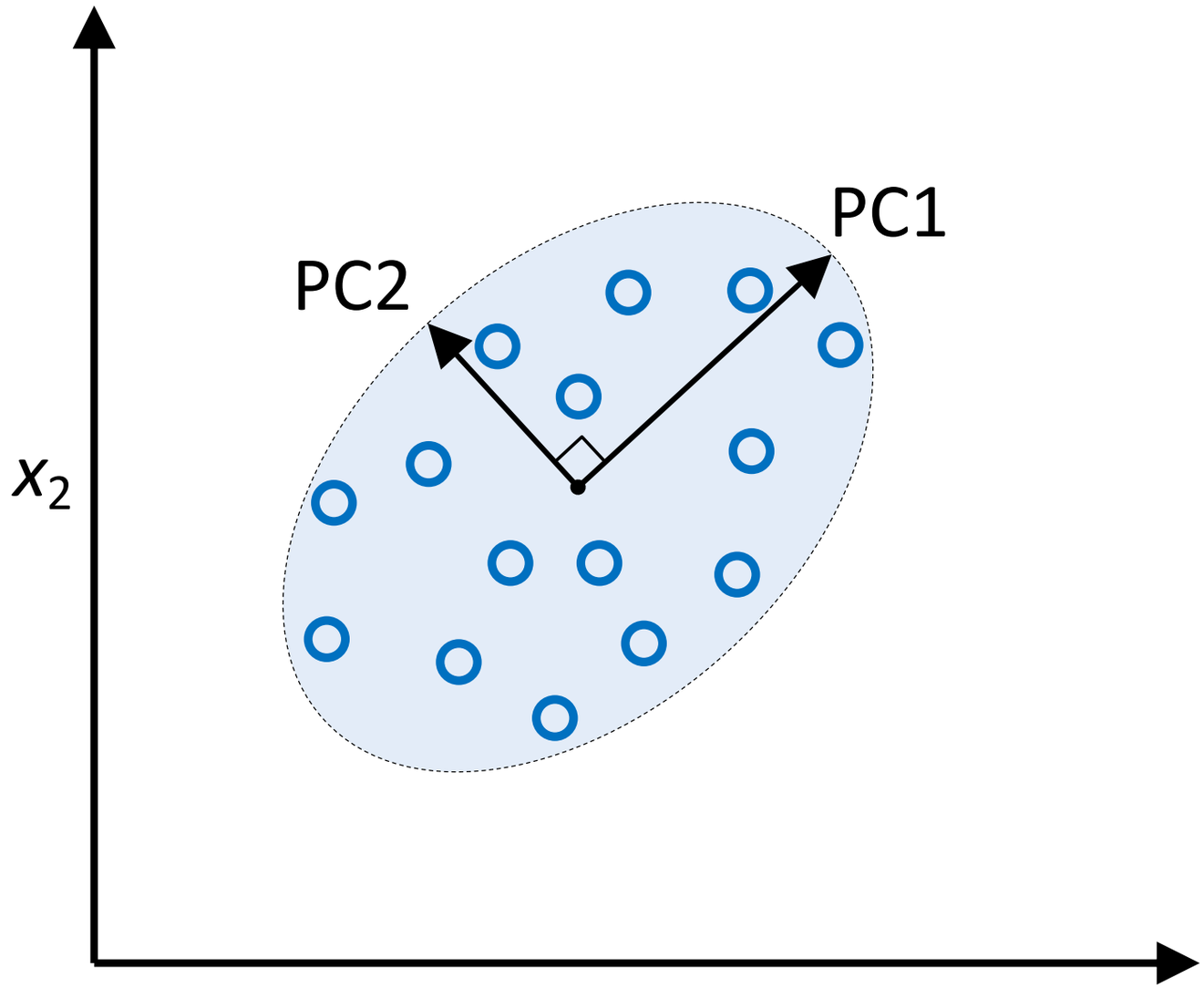
- Feature Selection
- Feature Extraction
  - Linear Methods
    - Principal Component Analysis (PCA)
    - Independent Component Analysis (ICA)
    - Autoencoders (linear act. func.)
    - Singular Vector Decomposition (SVD)
    - Linear Discriminant Analysis (LDA) (Supervised)
    - ...
  - Nonlinear Methods
    - t-Distr. Stochastic Neigh. Emb. (t-SNE)
    - Uniform Manifold Approx. & Proj. (UMAP)
    - Kernel PCA
    - Spectral Clustering
    - Autoencoders (non-linear act. func.)
    - ...

Goals of Dimensionality Reduction

- Reduce *Curse of Dimensionality* problems
- Increase storage and computational efficiency
- Visualize data in 2D or 3D

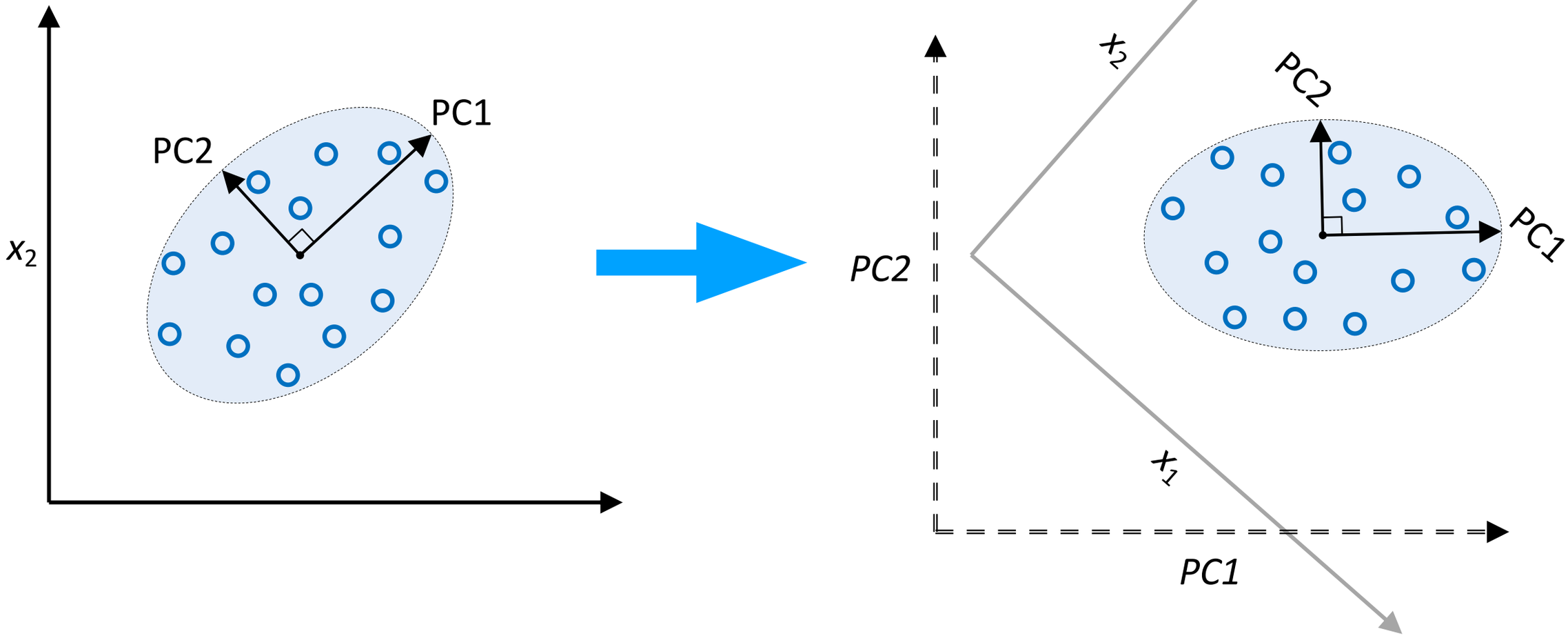
Principal Component Analysis (PCA)

1) Find directions of maximum variance



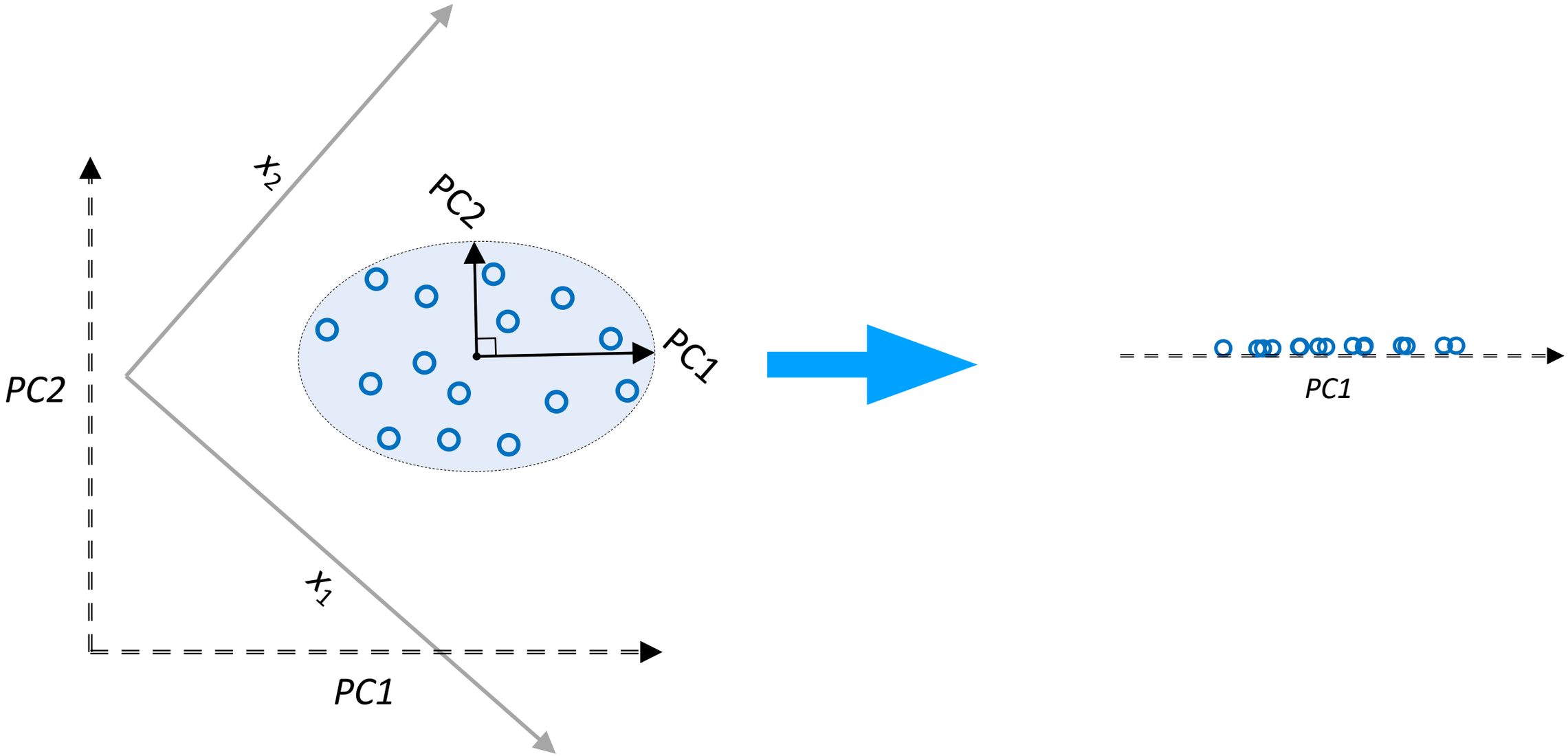
Principal Component Analysis (PCA)

2) Transform features onto directions of maximum variance



Principal Component Analysis (PCA)

3) Usually consider a subset of vectors of most variance (dimensionality reduction)



Principal Component Analysis (PCA) (in a nutshell)

Given design matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$

1) find vector α_i with maximum variance

2) repeat step 1: find α_{i+1}
with maximum variance uncorrelated with α_i

(repeat *k* times, where *k* is the desired number of dimensions; $k \leq m$)

Principal Component Analysis (PCA) (in a nutshell)

Approaches to solve PCA (on standardized data):

1. Constrained maximization (e.g., Lagrange multipliers)

2. Singular Vector Decomposition
   (right singular vectors are the principal components)

3. Eigen-decomposition of covariance matrix directly

Principal Component Analysis (PCA)

Eigen-decomposition of covariance matrix

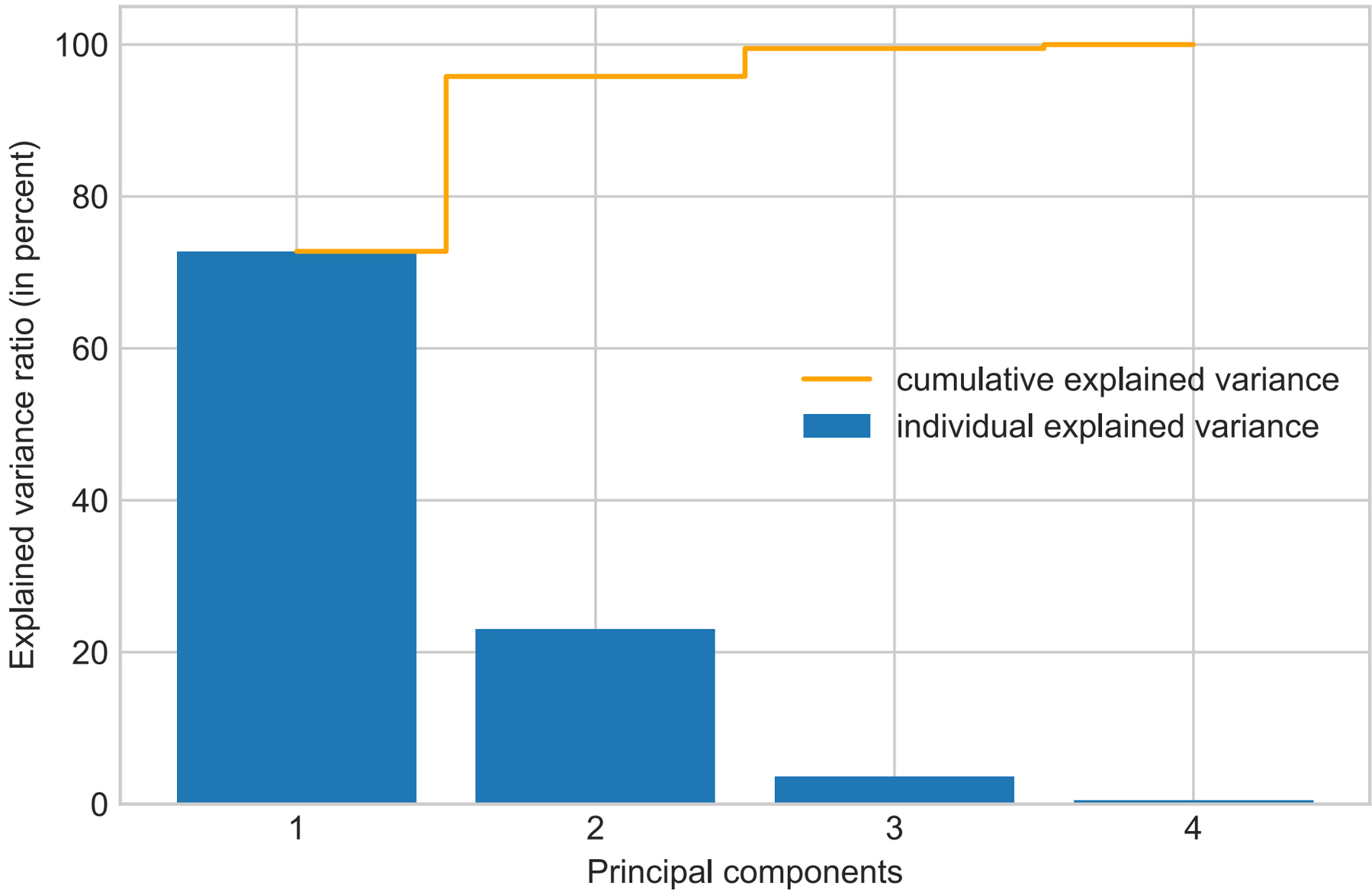
Collect eigenvectors α_i in a projection matrix $\mathbf{A} \in \mathbb{R}^{m \times k}$

(Sorted from highest to lowest associated eigenvalue)

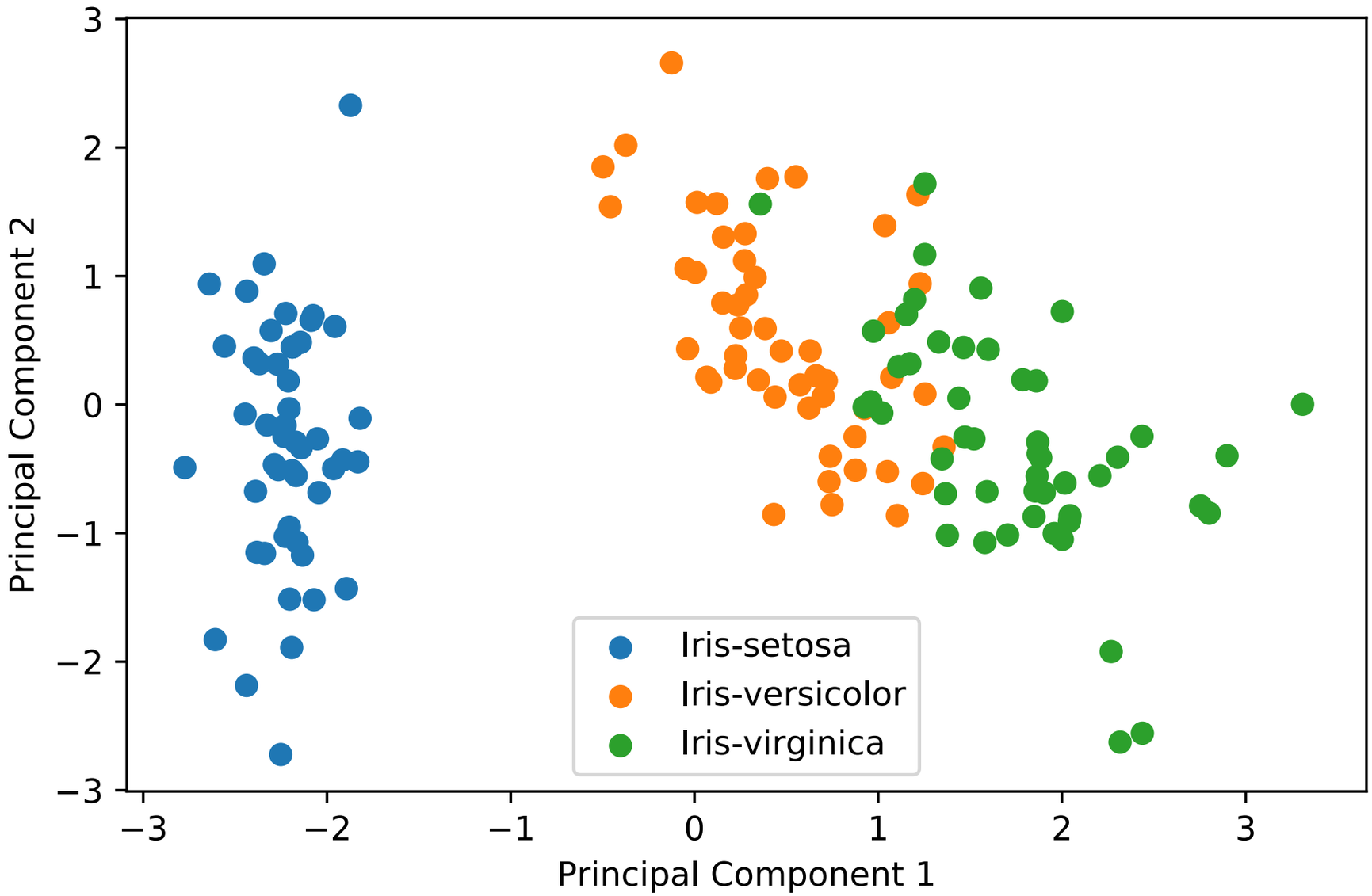
Compute projected data points: $\mathbf{Z} = \mathbf{X}\mathbf{A}$

Principal Component Analysis (PCA)

Usually useful to plot the explained variance (normalized eigenvalues)



Principal Component Analysis (PCA)

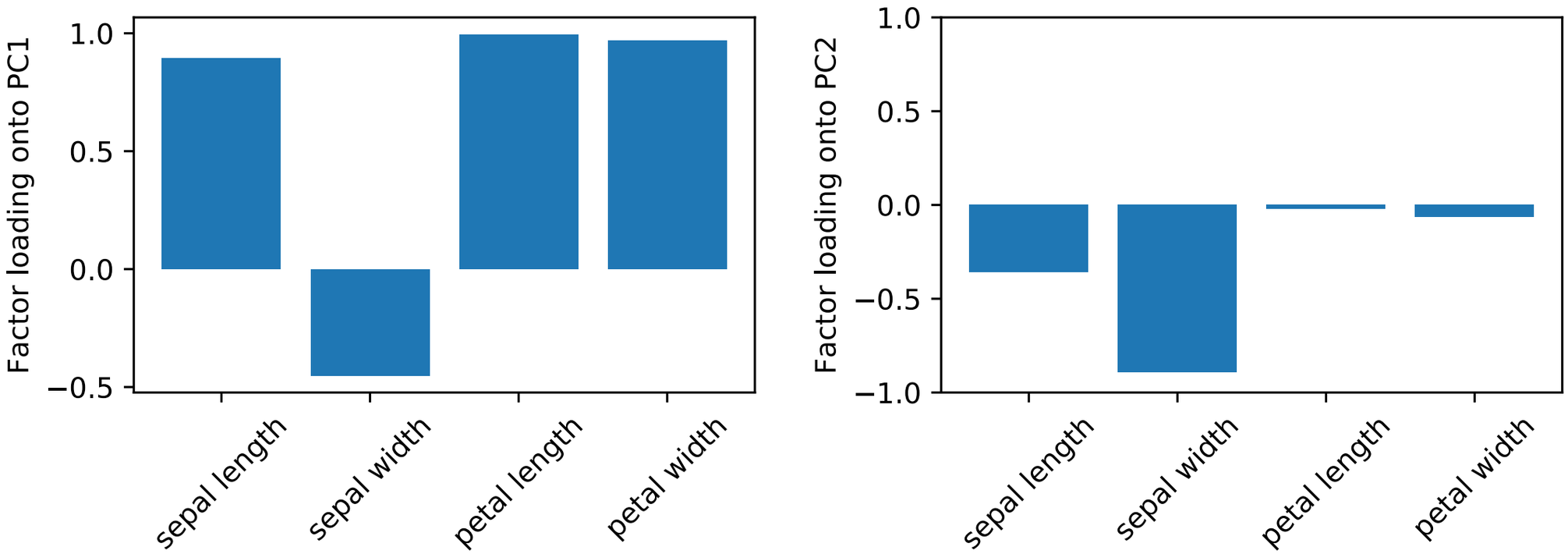


Keep in mind that PCA is unsupervised!

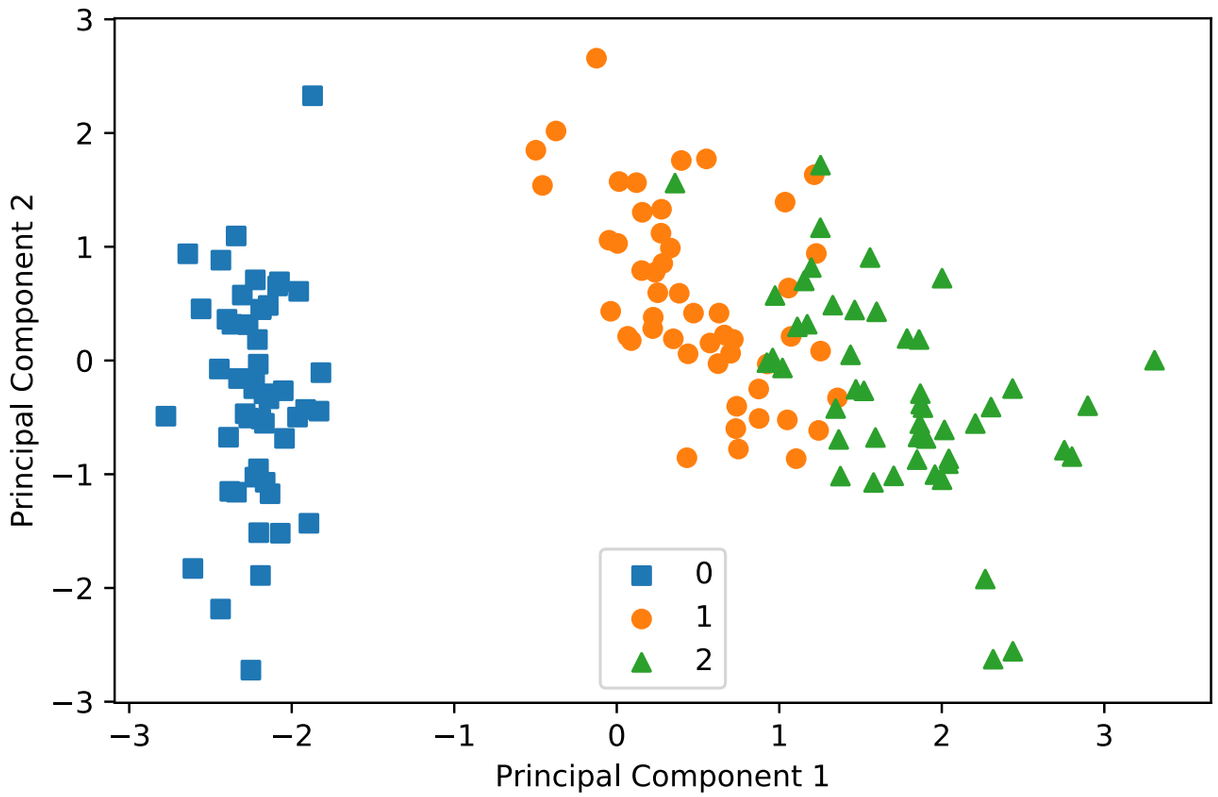
PCA Factor Loadings

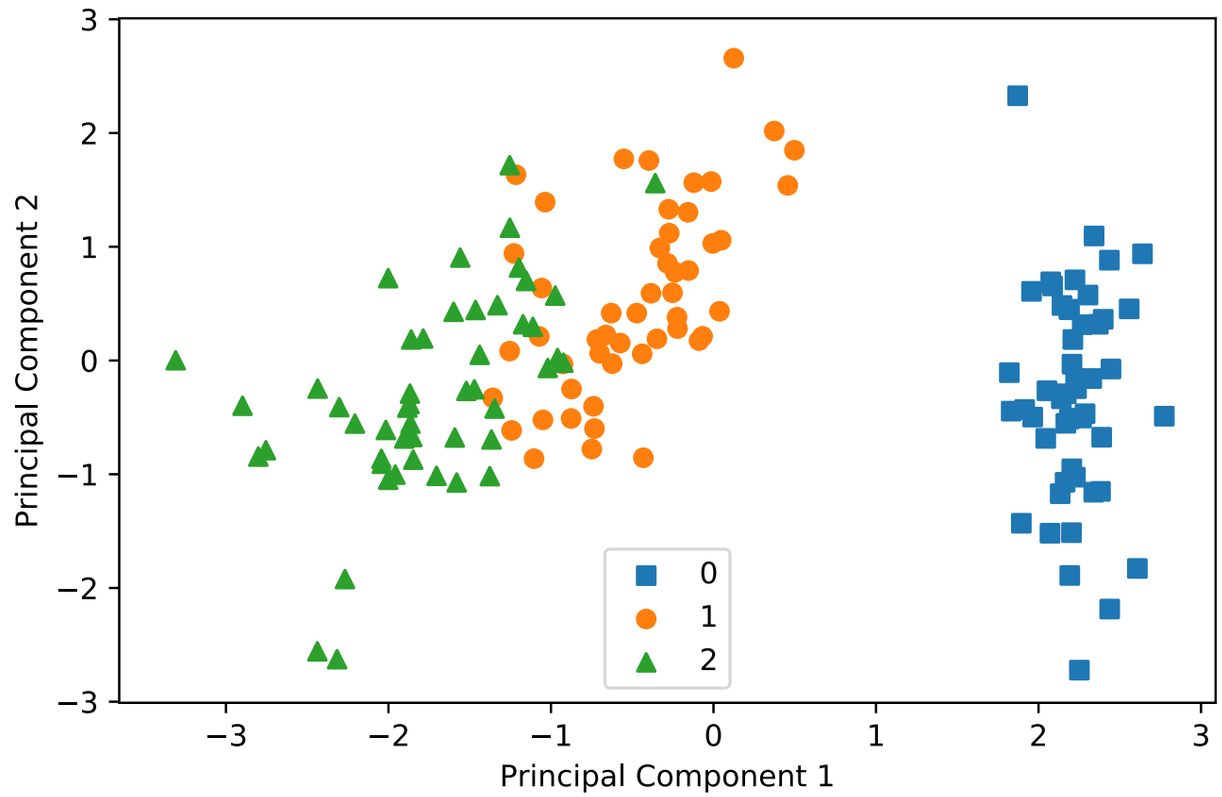
- The loadings are the unstandardized values of the eigenvectors

- We can interpret the loadings as the covariances (or correlation in case we standardized the input features) between the input features and the and the principal components (or eigenvectors), which have been scaled to unit length

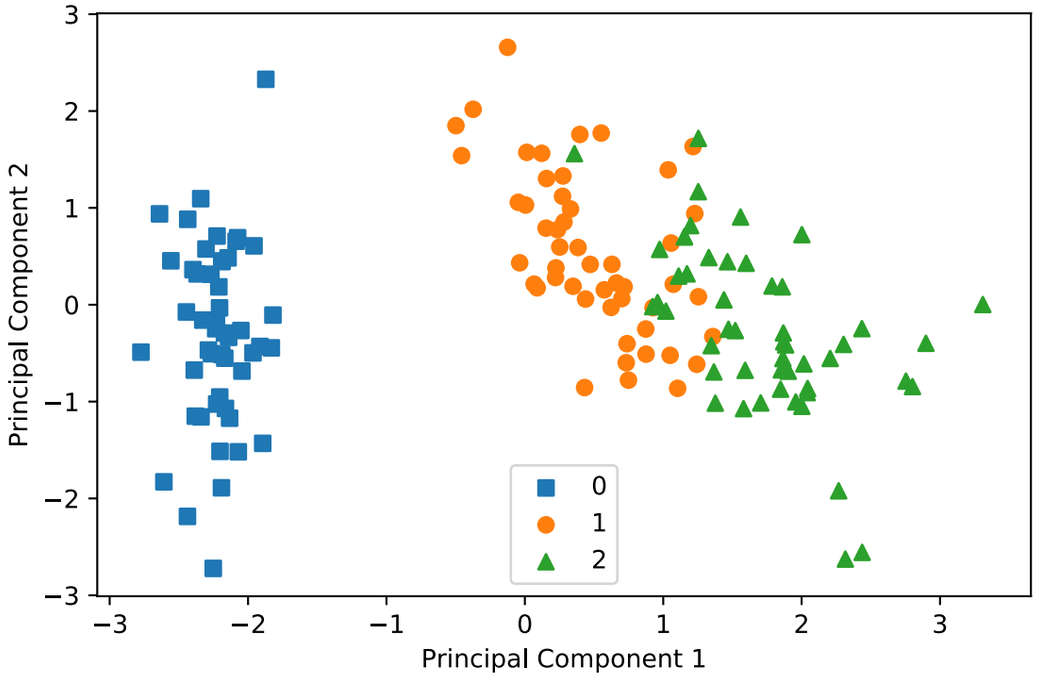


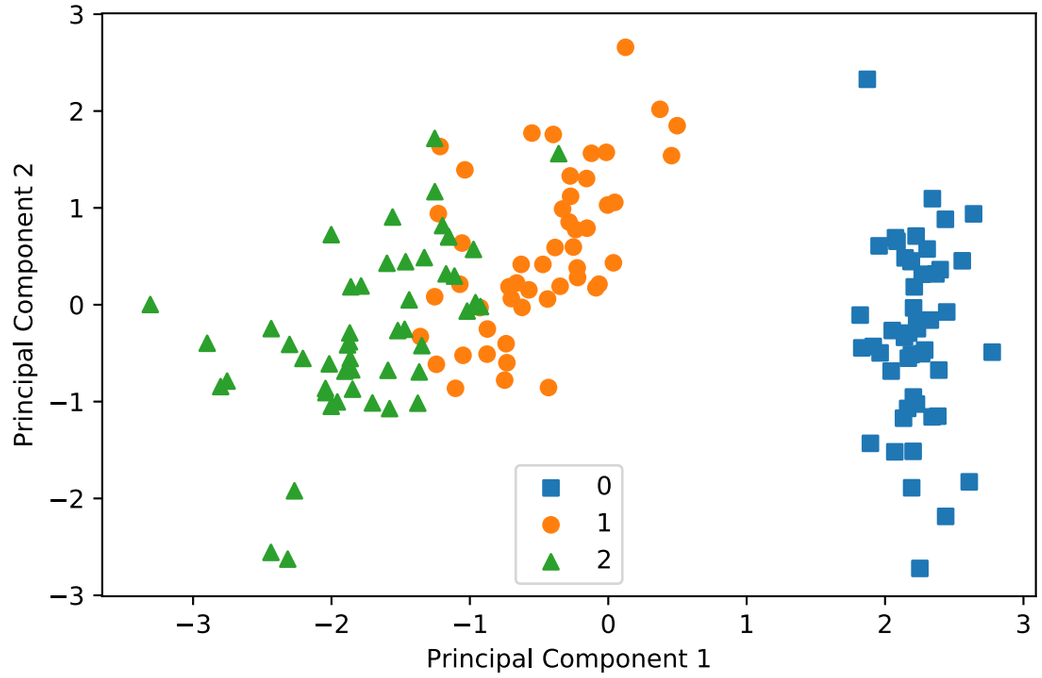
Mirrored Results in PCA





Mirrored Results in PCA





- Not due to an error; reason for this difference is that, depending on the eigensolver, eigenvectors can have either negative or positive signs

For instance, if v is an eigenvector of a matrix Σ , we have $\Sigma v = \lambda v$,

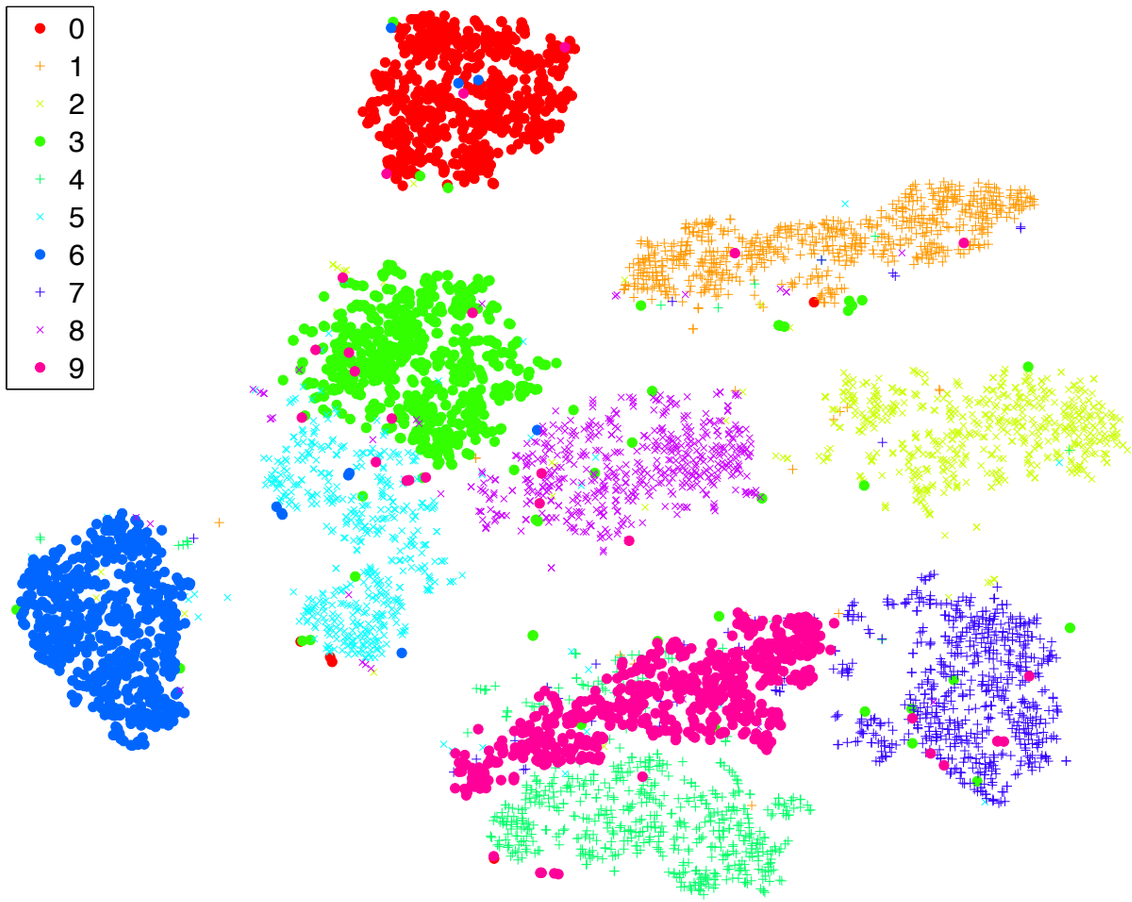
where λ is the eigenvalue then $-\lambda$ is also an eigenvalue of the same value

since $\Sigma(-v) = -\Sigma v = -\lambda v = \lambda(-v)$

t-Distributed Stochastic Neighbor Embedding (t-SNE)

(t-SNE is only meant for visualization not for preparing datasets!)

Note that MNIST has
28 x 28 = 784 dimensions



(a) Visualization by t-SNE.

Shown are 6000 images from MNIST projected in 2D

Maaten, L. V. D., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of machine learning research*, *9*(Nov), 2579-2605.

Stochastic Nearest Neighbor Embeddings (SNE)

Given high-dimensional datapoints, $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^m$

represent intrinsic structure of the data in 1D, 2D, or 3D (for visualization)

How?

1) Model neighboring datapoint pairs based on the distance of those points in the high-dimensional space
2) Find a probability distribution of the pairwise distances in the low dimensional space that is as close as possible as the original probability distribution

Main Idea: Map points near on a manifold to a near position in low-dimensional space

Stochastic Nearest Neighbor Embeddings (SNE)

Based on probability of selecting neighboring points $p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}$

(this is a modification to make the entropy [later slides] symmetric)

where the conditional probability, $p_{j|i}$, that x_i would pick x_j as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian centered at x_i

$$p_{j|i} = \frac{\exp(-||x_i - x_j||^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-||x_i - x_k||^2 / 2\sigma_i^2)}$$

neighborhood size is controlled by σ_i
(in turn controlled by perplexity parameter)

Denominator makes sure that similarity is independent of the point's density

For reference, note that the normal distribution is defined as

$$p(x \,|\, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} \right)$$

t-Distributed Stochastic Neighbor Embedding (t-SNE)

$$q_{ij} = \frac{(1 + ||z_i - z_j||^2)^{-1}}{\sum_{k \neq i} (1 + ||z_i - z_k||^2)^{-1}}$$

in **t**-SNE, modeled with Student's t-distribution to prevent crowding problem

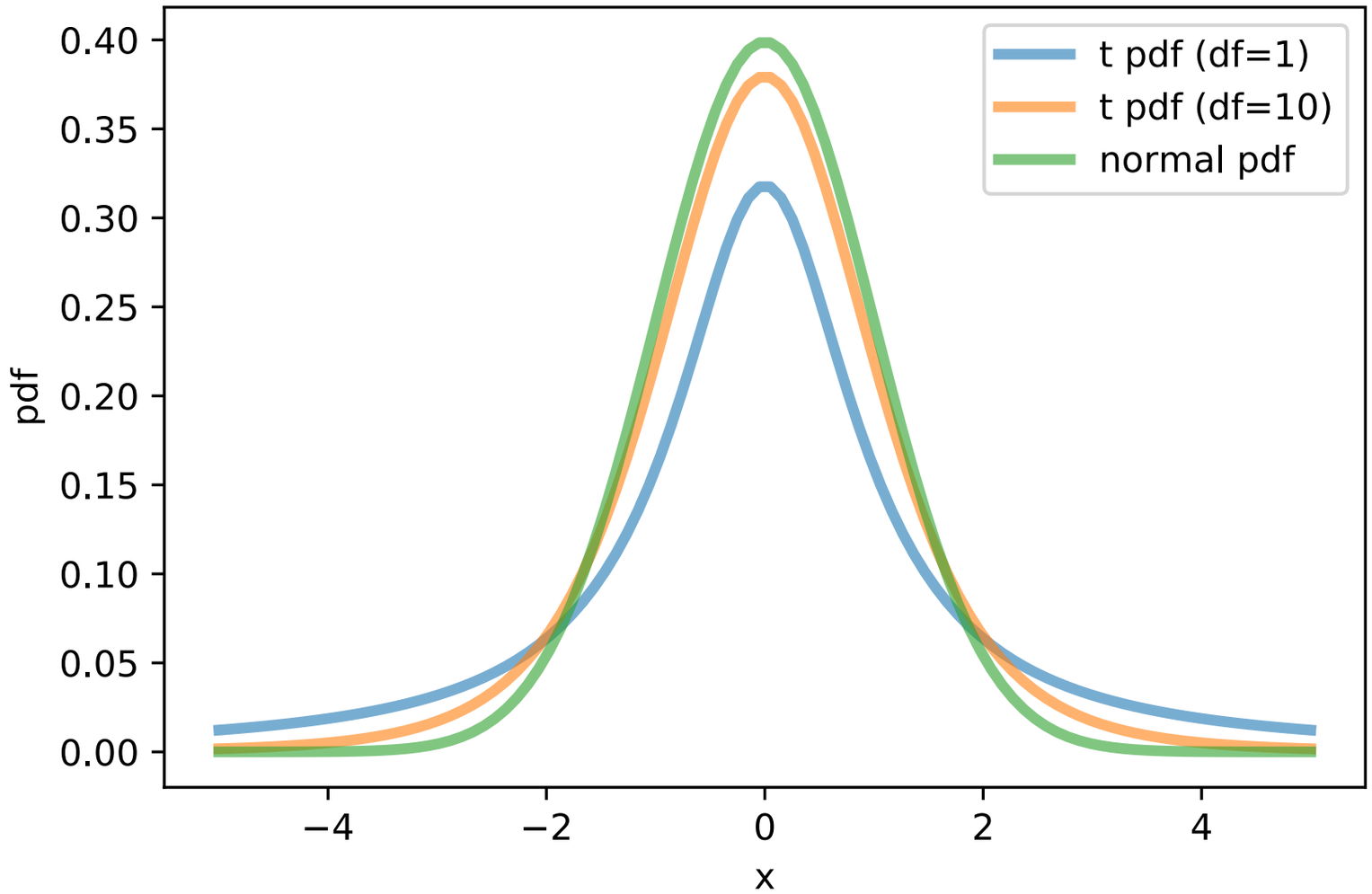
where z_i and z_j

are the points in the low-dimensional space

- t-distribution has "fatter" tails (more scale invariant to points far away)
- t-distribution avoids crowding problem
- (minor point: is faster for computing the density; no exponential)

t-Distribution

With 1 degree of freedom same as Cauchy distribution



t-Distributed Stochastic Neighbor Embedding (t-SNE)

Idea: Map points near on a manifold to a near position in low-dimensional space

1. Measure euclidean distance in **high** dim & convert to probability of picking a point as a neighbor (similarity is proportional to probability); use **Gaussian distribution** for density of each point
2. Same as 1. in **low** dimensionality but with **t distribution** (has heavier tails)
3. Minimize the difference of the conditional probabilities (KL-divergence)

Kullback Leibler divergence

Measures difference between 2 distributions; asymmetric

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx$$

$$= \underbrace{\int_{-\infty}^{\infty} p(x) \log p(x) dx}_{\text{Entropy}} - \underbrace{\int_{-\infty}^{\infty} p(x) \log q(x) dx}_{\text{Cross-Entropy}}$$

Remember Entropy from the Decision Tree lecture for discrete distributions?

Shannon Entropy:
average amount of information produced by a stochastic source of data

$$\mathrm{H}(i; x_j) = -\sum_{i=1}^{n} \mathrm{p}(i \mid x_j) \log_2 \mathrm{p}(i \mid x_j)$$

for feature x_j and class label *i*

t-Distributed Stochastic Neighbor Embedding (t-SNE)

conditional similarity between points in original space:

$$p_{j|i} = \frac{\exp(-|x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} -\exp(||x_i - x_k||^2/2\sigma_i^2)}$$

replace with p_{ij}

in symmetric SNE and t-SNE

where $p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}$

The remaining parameter to be selected is the variance σ_i of the Gaussian that is centered over each high-dimensional datapoint, x_i. It is not likely that there is a single value of σ_i that is optimal for all datapoints in the data set because the density of the data is likely to vary. In dense regions, a smaller value of σ_i is usually more appropriate than in sparser regions. Any particular value of σ_i induces a probability distribution, P_i, over all of the other datapoints. This distribution has an entropy which increases as σ_i increases. SNE performs a binary search for the value of σ_i that produces a P_i with a fixed perplexity that is specified by the user.[3] The perplexity is defined as

$$Perp(P_i) = 2^{H(P_i)},$$

where $H(P_i)$ is the Shannon entropy of P_i measured in bits

$$H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}.$$

The perplexity can be interpreted as a smooth measure of the effective number of neighbors. The performance of SNE is fairly robust to changes in the perplexity, and typical values are between 5 and 50.

Maaten, L. V. D., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of machine learning research*, *9*(Nov), 2579-2605.

t-Distributed Stochastic Neighbor Embedding (t-SNE)

Gradient Descent Optimization

Cost function C:

$$C = \sum_i KL(P_i || Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

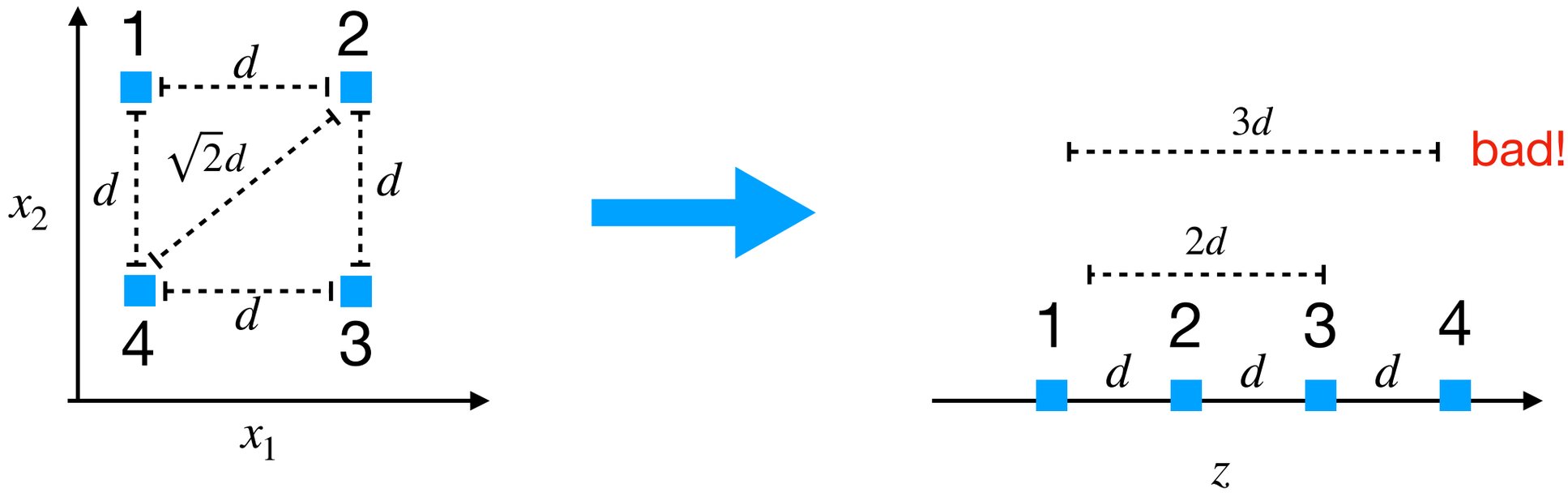
Regular SNE Gradient w.r.t. *z:*

$$\frac{\partial C}{\partial z_i} = 2 \sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(z_i - z_j)$$

replace $p_{j|i}$ with p_{ij} in symmetric SNE

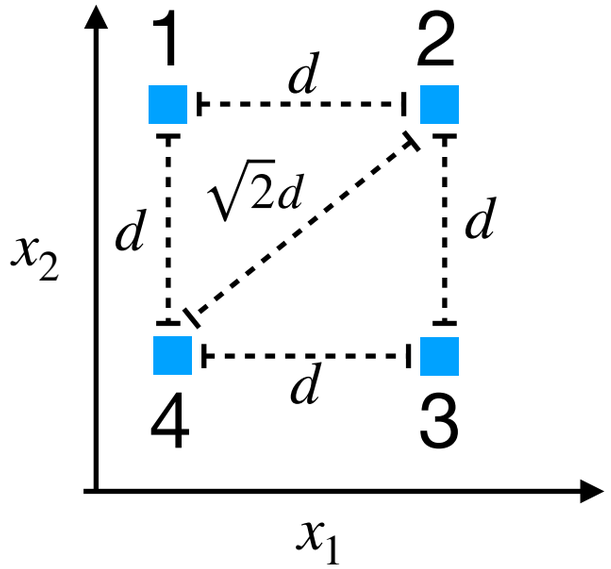
t-Distributed Stochastic Neighbor Embedding (t-SNE)

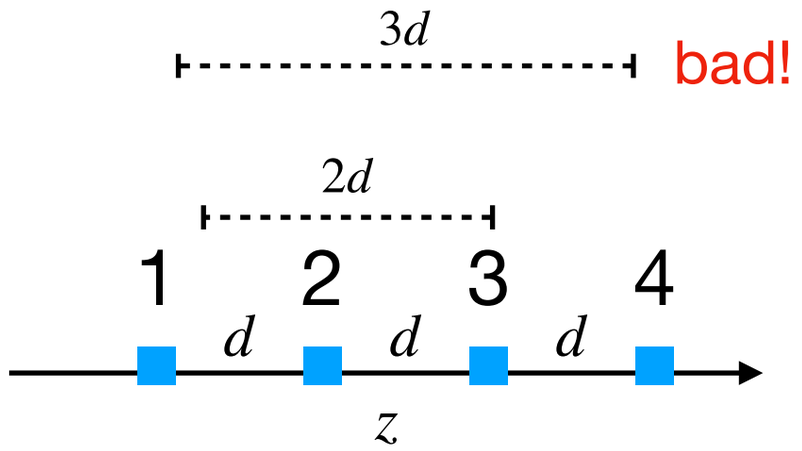
Crowding problem

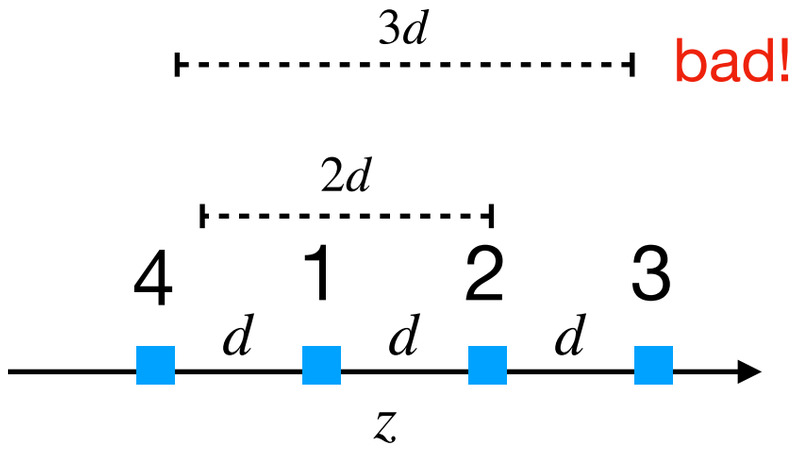


suppose you want to maintain the neighbor-ship of the 2D space in 1D

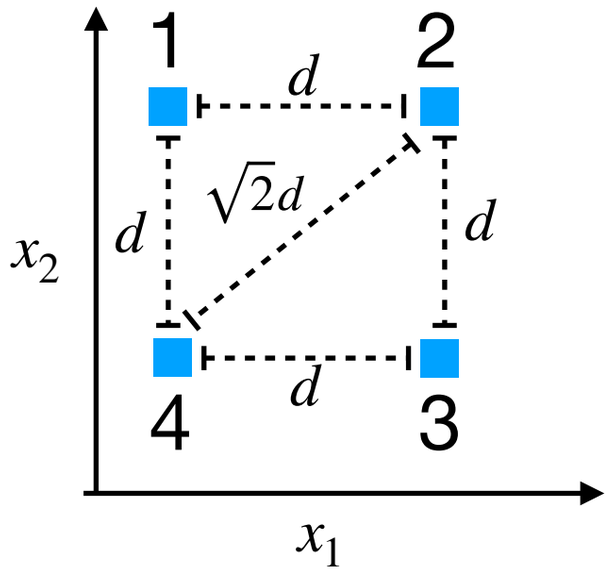
t-Distributed Stochastic Neighbor Embedding (t-SNE)

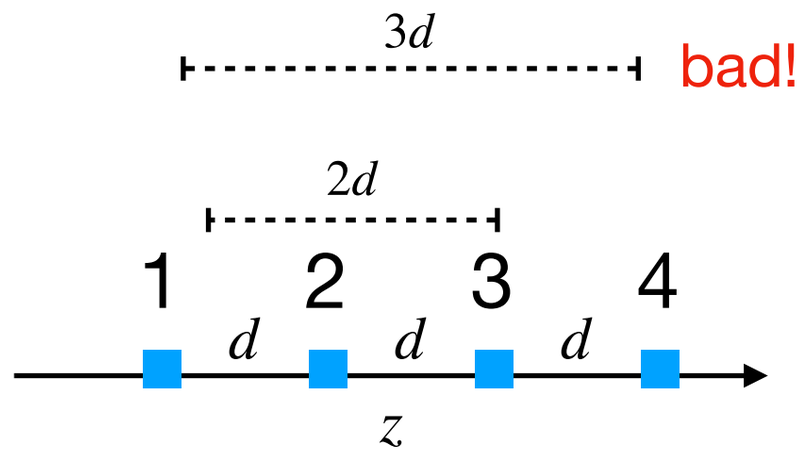


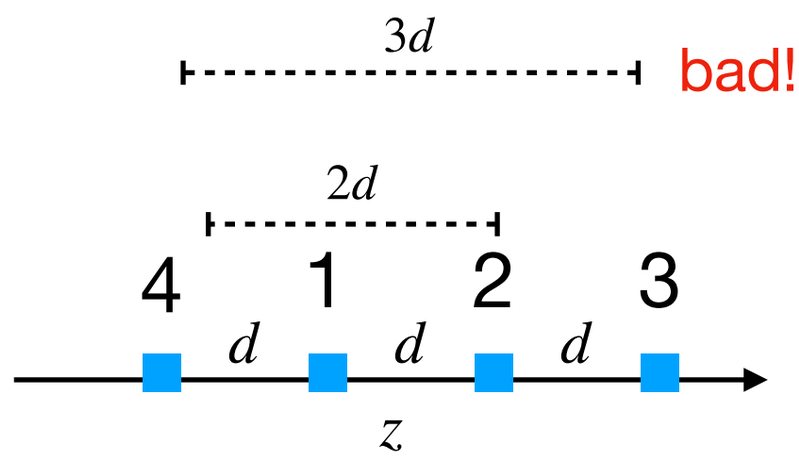


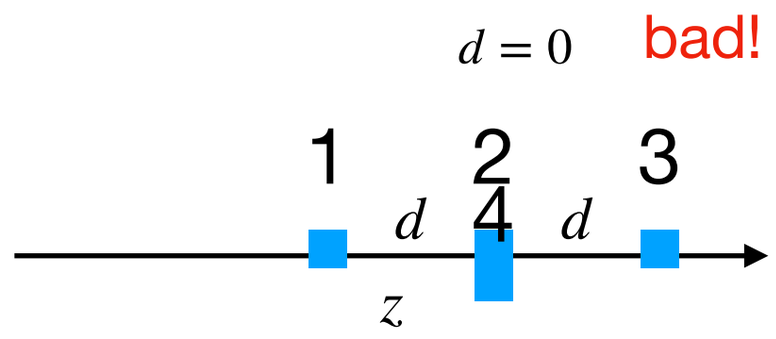


t-Distributed Stochastic Neighbor Embedding (t-SNE)

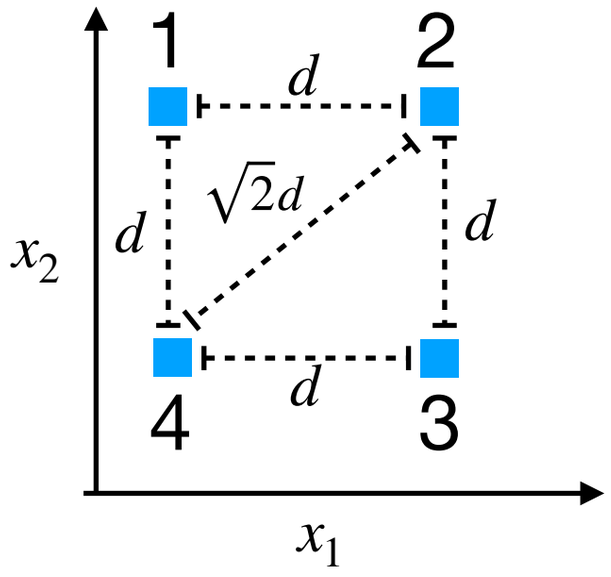


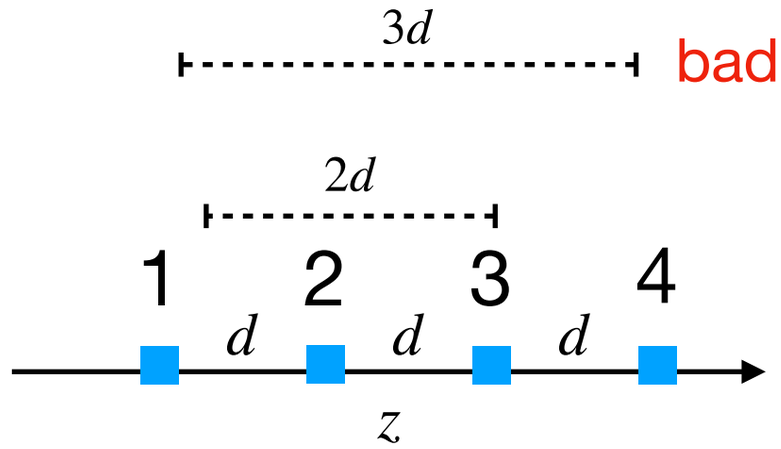


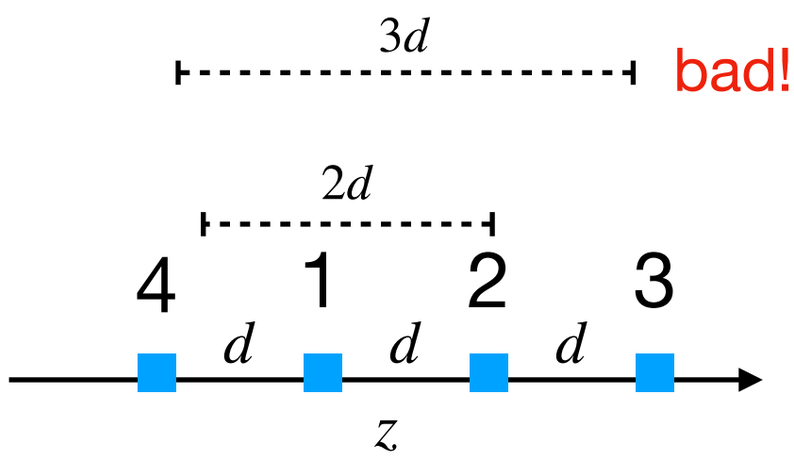




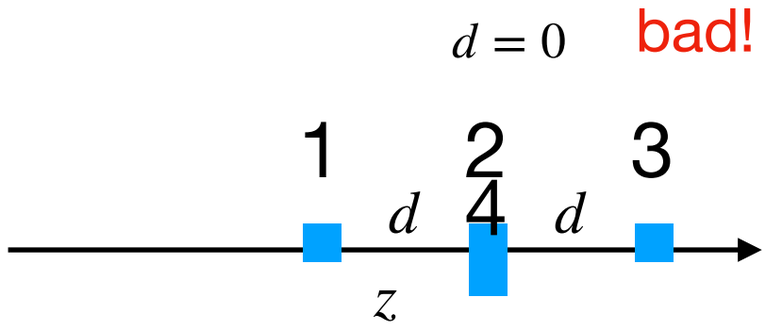
t-Distributed Stochastic Neighbor Embedding (t-SNE)





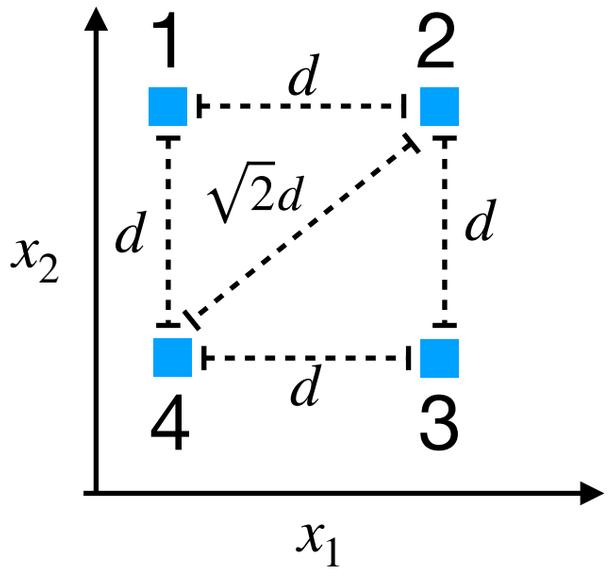


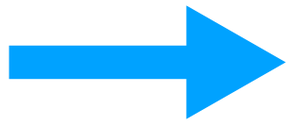
case where distance representation in low dimension is impossible :(

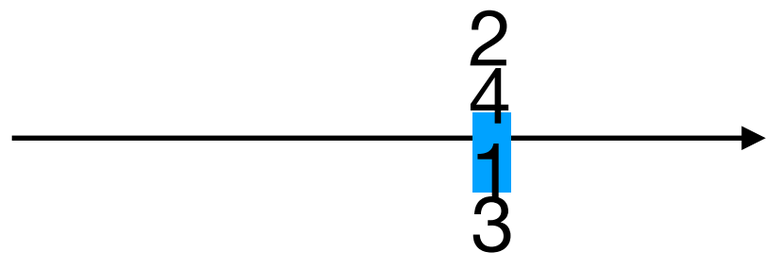


t-Distributed Stochastic Neighbor Embedding (t-SNE)

What would regular SNE do?





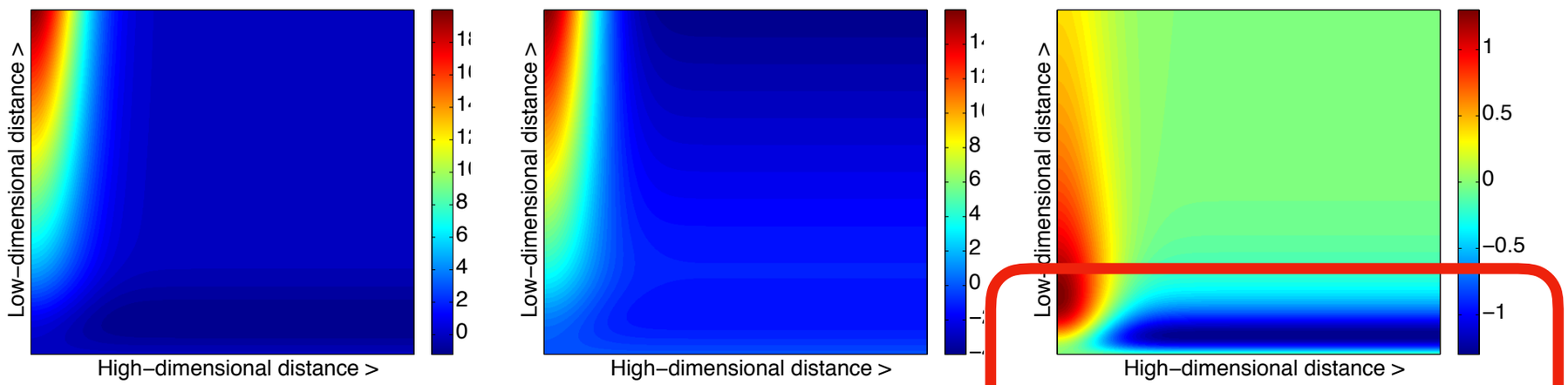


Crowding problem!

Squashes all points!

t-Distributed Stochastic Neighbor Embedding (t-SNE)

Maaten, L. V. D., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of machine learning research*, *9*(Nov), 2579-2605.



(a) Gradient of SNE. (b) Gradient of UNI-SNE. (c) Gradient of t-SNE.

Figure 1: Gradients of three types of SNE as a function of the pairwise Euclidean distance between two points in the high-dimensional and the pairwise distance between the points in the low-dimensional data representation.

negative gradient if points are too close in low-dim space to provide some repulsion against crowding

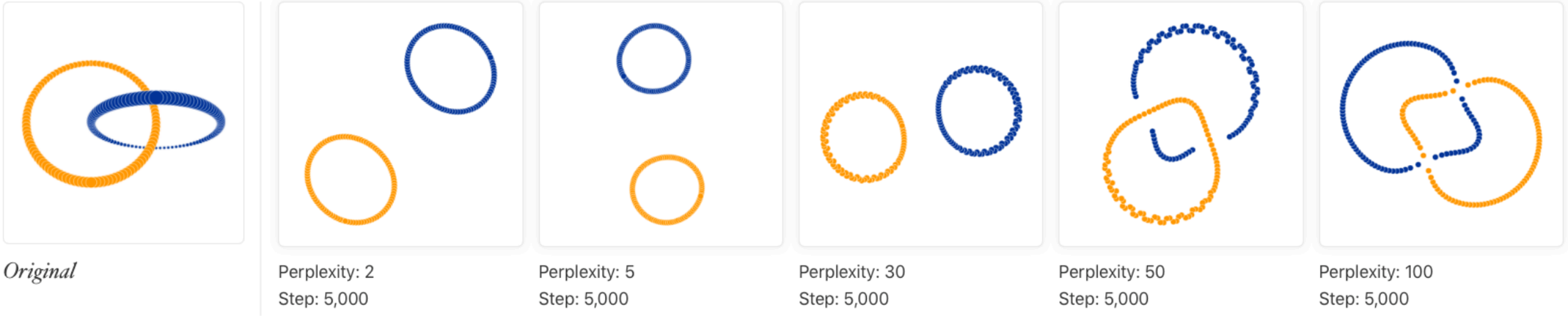
t-SNE Gradient w.r.t. *z:*

$$\frac{\partial C}{\partial z_i} = 4 \sum_j (p_{ij} - q_{ij})(z_i - z_j)\left(1 + ||z_i - z_j||^2\right)^{-1} \qquad \text{where } p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}$$

t-Distributed Stochastic Neighbor Embedding (t-SNE)

- Great for visualizing datasets in 2D
- Need to analyze multiple perplexity values (tuning parameter related to standard deviation of the Gaussian, to balance local and global attention)
- Not deterministic, the cost function for t-SNE is not convex
- More hyperparameters (learning rate epsilon)

t-Distributed Stochastic Neighbor Embedding (t-SNE)



Original

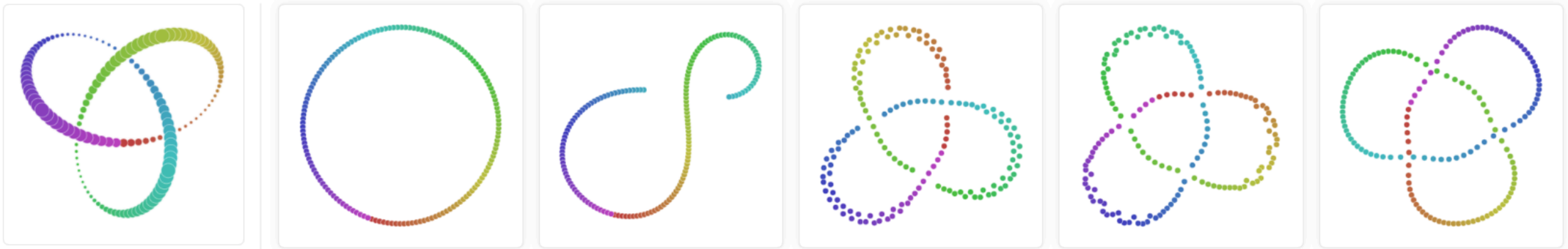
Perplexity: 2
Step: 5,000

Perplexity: 5
Step: 5,000

Perplexity: 30
Step: 5,000

Perplexity: 50
Step: 5,000

Perplexity: 100
Step: 5,000



Original

Perplexity: 2
Step: 5,000

Perplexity: 5
Step: 5,000

Perplexity: 30
Step: 5,000

Perplexity: 50
Step: 5,000

Perplexity: 100
Step: 5,000

Source: https://distill.pub/2016/misread-tsne/

f-Divergences

In probability theory f-divergence is a function $D_f(P||Q)$

for measuring the difference between 2 probability distributions

P and Q

Csiszár, I. (1963). "Eine informationstheoretische Ungleichung und ihre Anwendung auf den Beweis der Ergodizitat von Markoffschen Ketten". *Magyar. Tud. Akad. Mat. Kutato Int. Kozl*. **8**: 85–108.

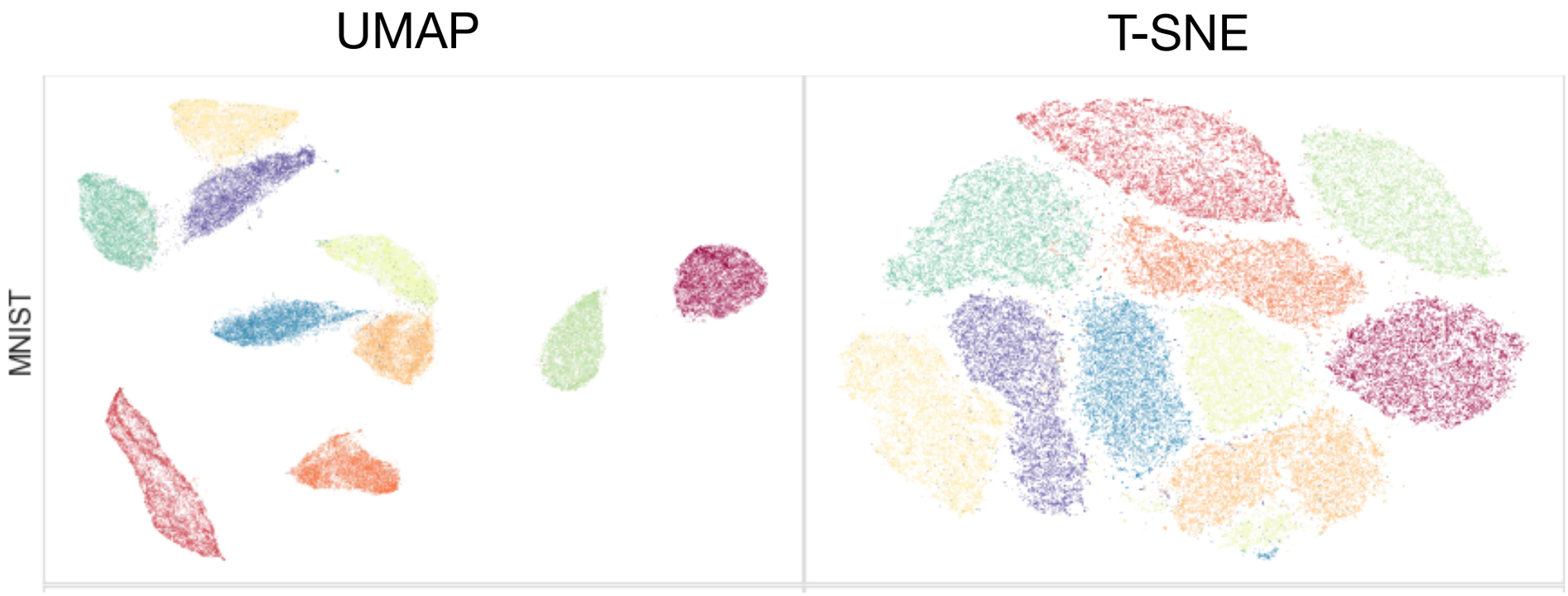
Morimoto, T. (1963). "Markov processes and the H-theorem". *J. Phys. Soc. Jpn*. **18** (3): 328–331

t-SNE embeddings based on five different f-divergences

$D_f(P\|Q)$	$f(t)$	ft-SNE objective	Emphasis
Kullback-Leibler (KL)	$t \log t$	$\sum p_{ij} \left(\log \frac{p_{ij}}{q_{ij}} \right)$	Local
Chi-square ($\mathcal{X}^2$ or CH)	$(t-1)^2$	$\sum \frac{(p_{ij}-q_{ij})^2}{q_{ij}}$	Local
Reverse-KL (RKL)	$-\log t$	$\sum q_{ij} \left(\log \frac{q_{ij}}{p_{ij}} \right)$	Global
Jensen-Shannon (JS)	$(t+1) \log \frac{2}{(t+1)} + t \log t$	$\frac{1}{2}(\mathrm{KL}(p_{ij} \| \frac{p_{ij}+q_{ij}}{2}) + \mathrm{KL}(q_{ij} \| \frac{p_{ij}+q_{ij}}{2}))$	Both
Hellinger distance (HL)	$(\sqrt{t}-1)^2$	$\sum (\sqrt{p_{ij}} - \sqrt{q_{ij}})^2$	Both

Im DJ, Verma N, Branson K. Stochastic Neighbor Embedding under f-divergences. arXiv preprint arXiv:1811.01247. 2018 Nov 3.

Uniform Manifold Approximation and Projection (UMAP)



McInnes, L., & Healy, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426.

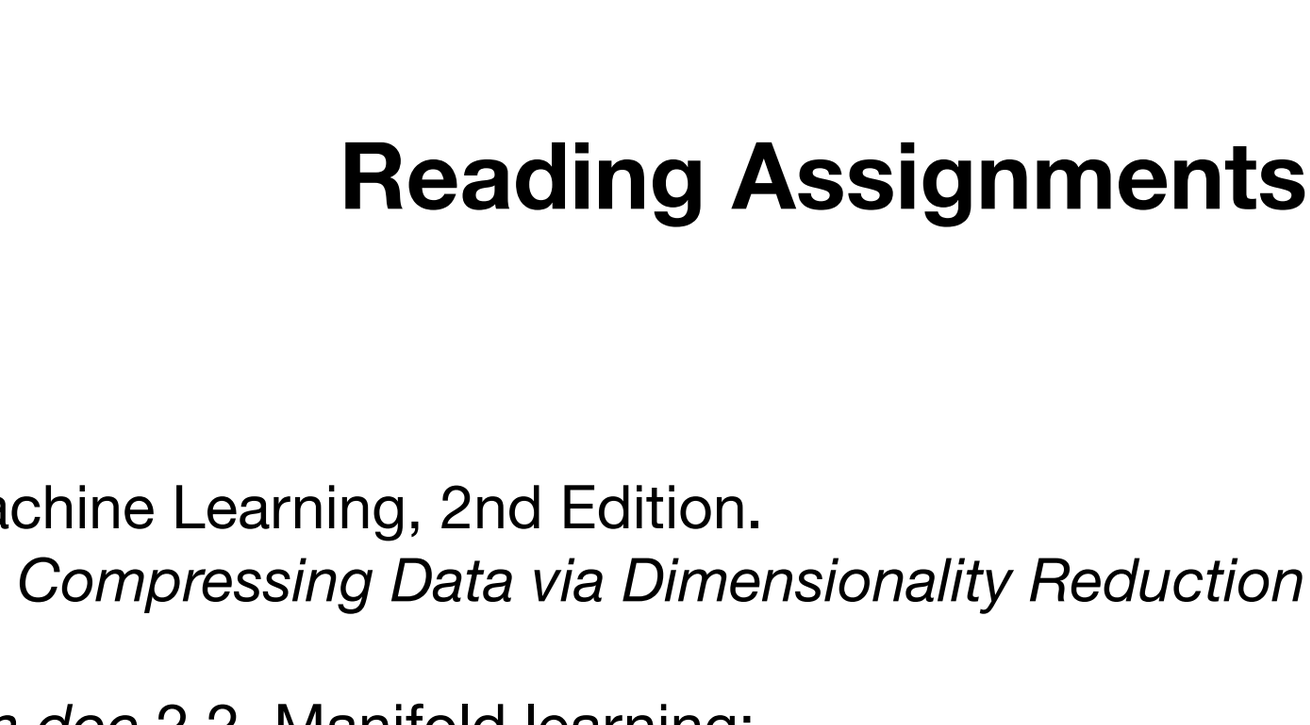
Compared to t-SNE, UMAP seems to be

- faster
- deterministic
- better at preserving clusters

Reading Assignments

- Python Machine Learning, 2nd Edition.
  Chapter 5: Compressing Data via Dimensionality Reduction

- *Scikit-learn doc* 2.2. Manifold learning:
  https://scikit-learn.org/stable/modules/manifold.html



Code Examples

https://github.com/rasbt/stat479-machine-learning-fs19/blob/master/14_feat-extract/14_feat-extract_code.ipynb