Cognitive Algorithms
Lecture 5

Unsupervised Learning

Klaus-Robert Müller, Johannes Niediek,
Augustin Krause, Joanina Oltersdorff, Ken Schreiber

Technische Universität Berlin
Machine Learning Group

Announcements

- There will be a practice exam ("Probeklausur"), published around 1st of February
- Relevant dates 2024
  - January 9th, last course lecture
  - February 6th, recap lecture by Johannes Niediek
  - Week of February 12th, discussion of the practice exam with all tutors
  - February 19th, 14.00–16.30 written exam ("first date")
  - April 8th, 08.30–11.00 written exam ("second date")
- All relevant rooms will be announced on ISIS
- Small changes are still possible, check ISIS

Notation

Link to notation document

$$\varphi(X)\varphi(X)^\top = \begin{pmatrix} \varphi_1(\boldsymbol{x}_1) & \dots & \varphi_1(\boldsymbol{x}_n) \\ & \dots & \\ \varphi_{\tilde{d}}(\boldsymbol{x}_1) & \dots & \varphi_{\tilde{d}}(\boldsymbol{x}_n) \end{pmatrix} \begin{pmatrix} \varphi_1(\boldsymbol{x}_1) & \dots & \varphi_{\tilde{d}}(\boldsymbol{x}_1) \\ & \dots & \\ \varphi_1(\boldsymbol{x}_n) & \dots & \varphi_{\tilde{d}}(\boldsymbol{x}_n) \end{pmatrix}$$
$$= \begin{pmatrix} \sum_i \varphi_1(\boldsymbol{x}_i)\varphi_1(\boldsymbol{x}_i) & \dots & \sum_i \varphi_1(\boldsymbol{x}_i)\varphi_{\tilde{d}}(\boldsymbol{x}_i) \\ & \dots & \\ \sum_i \varphi_{\tilde{d}}(\boldsymbol{x}_i)\varphi_1(\boldsymbol{x}_i) & \dots & \sum_i \varphi_{\tilde{d}}(\boldsymbol{x}_i)\varphi_{\tilde{d}}(\boldsymbol{x}_i) \end{pmatrix}$$

This expression occurs in the derivation of ridge regression.

Notation continued

$$
\begin{aligned}
\varphi(X)^\top \varphi(X) &= \begin{pmatrix} \varphi_1(\boldsymbol{x}_1) & \dots & \varphi_{\tilde{d}}(\boldsymbol{x}_1) \\ & \dots & \\ \varphi_1(\boldsymbol{x}_n) & \dots & \varphi_{\tilde{d}}(\boldsymbol{x}_n) \end{pmatrix} \begin{pmatrix} \varphi_1(\boldsymbol{x}_1) & \dots & \varphi_1(\boldsymbol{x}_n) \\ & \dots & \\ \varphi_{\tilde{d}}(\boldsymbol{x}_1) & \dots & \varphi_{\tilde{d}}(\boldsymbol{x}_n) \end{pmatrix} \\
&= \begin{pmatrix} \varphi(\boldsymbol{x}_1)^\top \varphi(\boldsymbol{x}_1) & \dots & \varphi(\boldsymbol{x}_1)^\top \varphi(\boldsymbol{x}_{\tilde{d}}) \\ & \dots & \\ \varphi(\boldsymbol{x}_{\tilde{d}})^\top \varphi(\boldsymbol{x}_1) & \dots & \varphi(\boldsymbol{x}_{\tilde{d}})^\top \varphi(\boldsymbol{x}_{\tilde{d}}) \end{pmatrix} \\
&= \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \dots & k(\boldsymbol{x}_1, \boldsymbol{x}_{\tilde{d}}) \\ & \dots & \\ k(\boldsymbol{x}_{\tilde{d}}, \boldsymbol{x}_1) & \dots & k(\boldsymbol{x}_{\tilde{d}}, \boldsymbol{x}_{\tilde{d}}) \end{pmatrix}
\end{aligned}
$$

This expression occurs in the derivation of kernel ridge regression.
Bias and Variance questions: will be considered later!

Recap: Kernels

- Basic idea: take a known machine learning method, then replace $\boldsymbol{x}_i^\top \boldsymbol{x}_j$ by $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

Recap: Kernels

- Basic idea: take a known machine learning method, then replace $\boldsymbol{x}_i^\top \boldsymbol{x}_j$ by $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$.
  - Mercer's theorem: if k is symmetric positive semi-definite, then there is some $\varphi : \mathcal{X} \to \mathcal{F}$ such that $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i)^\top \varphi(\boldsymbol{x}_j)$.

Recap: Kernels

- Basic idea: take a known machine learning method, then replace $\boldsymbol{x}_i^\top \boldsymbol{x}_j$ by $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$.
  - Mercer's theorem: if k is symmetric positive semi-definite, then there is some $\varphi : \mathcal{X} \to \mathcal{F}$ such that $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i)^\top \varphi(\boldsymbol{x}_j)$.
- Why are kernels useful?

Recap: Kernels

- Basic idea: take a known machine learning method, then replace $\boldsymbol{x}_i^\top \boldsymbol{x}_j$ by $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$.
  - Mercer's theorem: if k is symmetric positive semi-definite, then there is some $\varphi : \mathcal{X} \to \mathcal{F}$ such that $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i)^\top \varphi(\boldsymbol{x}_j)$.
- Why are kernels useful?
  - Implicitly work in high-dimensional space

Recap: Kernels

- Basic idea: take a known machine learning method, then replace $\boldsymbol{x}_i^\top \boldsymbol{x}_j$ by $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$.
  - Mercer's theorem: if k is symmetric positive semi-definite, then there is some $\varphi : \mathcal{X} \to \mathcal{F}$ such that $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i)^\top \varphi(\boldsymbol{x}_j)$.
- Why are kernels useful?
  - Implicitly work in high-dimensional space
  - Representer theorem: for minimizer of *regularized* error function, have $f(\boldsymbol{x}) = \sum_i \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i)$.
    "compare new data to all training data points"

Recap: Kernels

- Basic idea: take a known machine learning method, then replace $\boldsymbol{x}_i^\top \boldsymbol{x}_j$ by $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$.
  - Mercer's theorem: if k is symmetric positive semi-definite, then there is some $\varphi : \mathcal{X} \to \mathcal{F}$ such that $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i)^\top \varphi(\boldsymbol{x}_j)$.
- Why are kernels useful?
  - Implicitly work in high-dimensional space
  - Representer theorem: for minimizer of *regularized* error function, have $f(\boldsymbol{x}) = \sum_i \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i)$.
    "compare new data to all training data points"
- How can you show that a function $k(\cdot, \cdot)$ is a kernel function?
  - Find φ such that $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$ or show that k is symmetric positive semi-definite

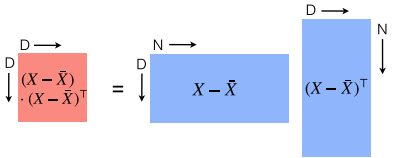
Recap: covariance matrix

Given n data points $\boldsymbol{x}_i \in \mathbb{R}^d$ in a data matrix $X \in \mathbb{R}^{d \times n}$, the empirical estimate of the **covariance matrix** is defined as

$$\bar{\Sigma} = \frac{1}{n} (X - \bar{X})(X - \bar{X})^\top$$

where $\bar{\boldsymbol{x}} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i$

$$\bar{X} = (\bar{\boldsymbol{x}}, \bar{\boldsymbol{x}}, \ldots, \bar{\boldsymbol{x}}) \in \mathbb{R}^{d \times n}$$



$\bar{\Sigma}$ measures how much the data-points co-vary, for all pairs of dimensions.

Unsupervised learning

- **Supervised** algorithms
  - **Classification** and **regression**
  - Use labels in training

Approach	Labels
Binary classification	
Regression	

Unsupervised learning

- **Supervised** algorithms
  - **Classification** and **regression**
  - Use labels in training

Approach	Labels
Binary classification	$\in \{+1, -1\}$
Regression	$\in \mathbb{R}$

Unsupervised learning

- **Supervised** algorithms
  - **Classification** and **regression**
  - Use labels in training
- But labels not always given e.g.
  - Mixtures of different speakers in an audio recording
  - Complex artifacts in experimental recordings

Approach	Labels
Binary classification	$\in \{+1, -1\}$
Regression	$\in \mathbb{R}$
Unsupervised	Do not exist!

Unsupervised learning

- **Supervised** algorithms
  - **Classification** and **regression**
  - Use labels in training
- But labels not always given e.g.
  - Mixtures of different speakers in an audio recording
  - Complex artifacts in experimental recordings
- **Unsupervised** algorithms
  - No labels for training

Approach	Labels
Binary classification	$\in \{+1, -1\}$
Regression	$\in \mathbb{R}$
Unsupervised	Do not exist!

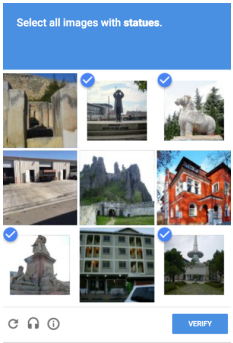
Unsupervised learning

Why unsupervised learning?

Unsupervised learning

Why unsupervised learning?

- No need for human-made labels
- Algorithm finds structure autonomously



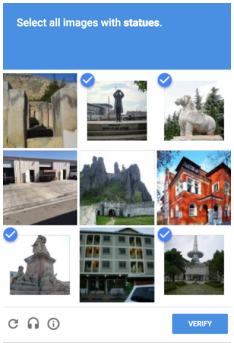
Unsupervised learning

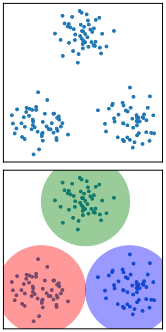
Why unsupervised learning?

- No need for human-made labels
- Algorithm finds structure autonomously

What can it be used for?

- Clustering
- Dimensionality reduction
- ...





Dimensionality reduction

In many applications...

- have high-dimensional data
- believe the data lie close to a low-dimensional subspace

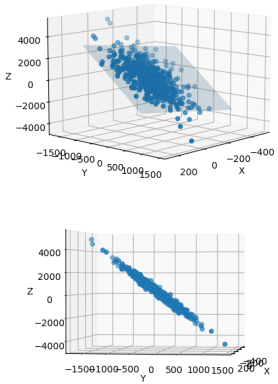
→ Fewer parameters needed to account for the data properties
(the low-dimensional variables are sometimes called *hidden causes* or *latent variables*)

Dimensionality reduction

In many applications...

- have high-dimensional data
- believe the data lie close to a low-dimensional subspace

→ Fewer parameters needed to account for the data properties
(the low-dimensional variables are sometimes called *hidden causes* or *latent variables*)



How to find a transformation $\mathbb{R}^3 \to \mathbb{R}^2$ that preserves the data structure?

Why dimensionality reduction?

Why dimensionality reduction?

- **Visualization**
  Insights into high-dimensional structures in the data

[Xiao et al., 2017]

Why dimensionality reduction?

- **Visualization**
  Insights into high-dimensional structures in the data
- **Better Generalization**
  Fewer dimensions $\rightarrow$ more robust parameter estimation

[Xiao et al., 2017]

Why dimensionality reduction?

- **Visualization**
  Insights into high-dimensional structures in the data
- **Better Generalization**
  Fewer dimensions $\rightarrow$ more robust parameter estimation
- **Speeding up learning algorithms**
  Most algorithms scale badly with increasing data dimensionality

[Xiao et al., 2017]

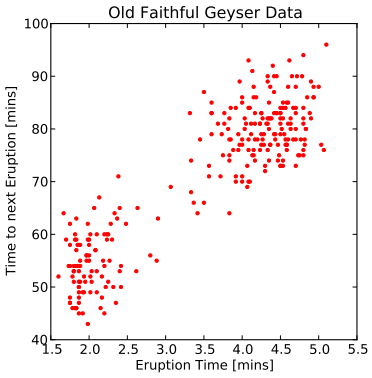
Why dimensionality reduction?

- **Visualization**
  Insights into high-dimensional structures in the data
- **Better Generalization**
  Fewer dimensions $\rightarrow$ more robust parameter estimation
- **Speeding up learning algorithms**
  Most algorithms scale badly with increasing data dimensionality
- **Data compression**
  Smaller storage requirements

[Xiao et al., 2017]

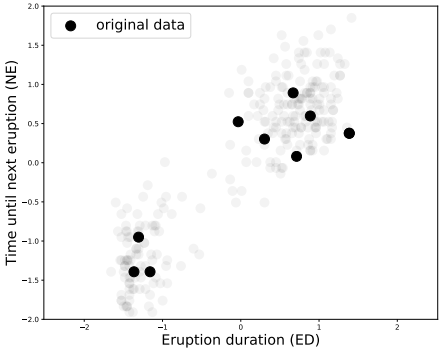
Example: Old Faithful geyser dataset

Old Faithful geyser eruption



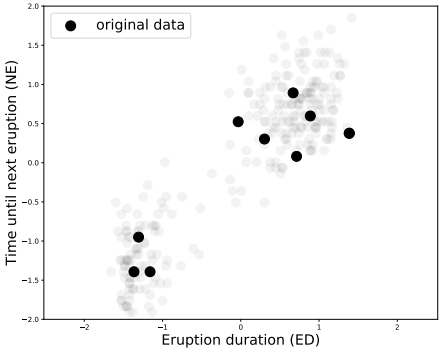
Informal example dimensionality reduction

- High correlation between *NE* and *ED*



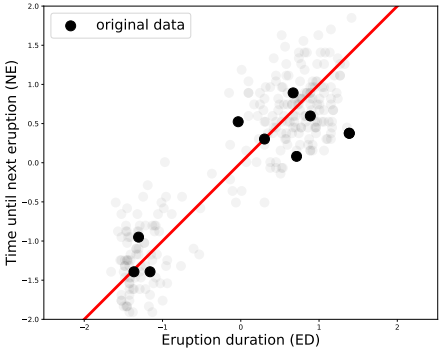
Informal example dimensionality reduction

- High correlation between *NE* and *ED*
- Let's try to project data on $1D$ subspace



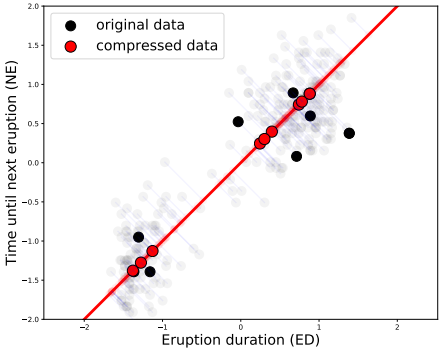
Informal example dimensionality reduction

- High correlation between *NE* and *ED*
- Let's try to project data on $1D$ subspace



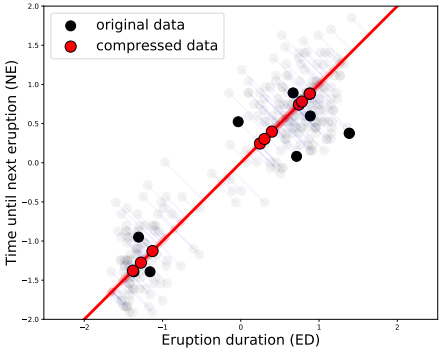
Informal example dimensionality reduction

- High correlation between *NE* and *ED*
- Let's try to project data on $1D$ subspace



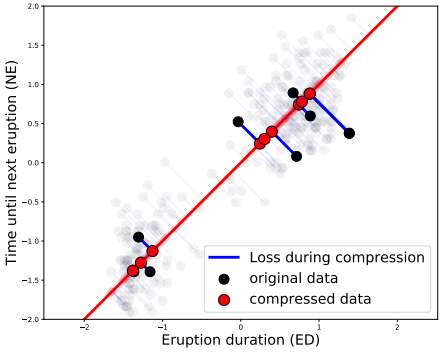
Informal example dimensionality reduction

- High correlation between *NE* and *ED*
- Let's try to project data on $1D$ subspace
- Relatively good representation



Informal example dimensionality reduction

- High correlation between *NE* and *ED*
- Let's try to project data on $1D$ subspace
- Relatively good representation



The mathematical model for linear dimensionality reduction

We have

- high-dimensional data $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$
- reason to believe they lie close to a lower-dimensional subspace

$\rightarrow$ $m < d$ parameters needed to account for the data properties
hidden causes or *latent variables* $H = [\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_n] \in \mathbb{R}^{m \times n}$

The mathematical model for linear dimensionality reduction

We have

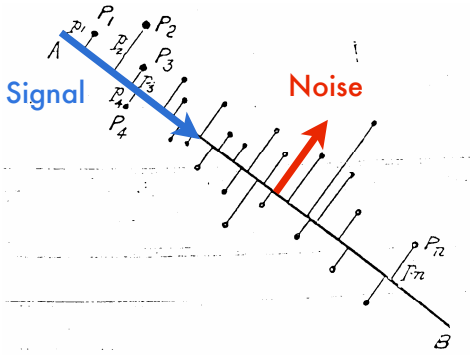
- high-dimensional data $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$
- reason to believe they lie close to a lower-dimensional subspace

$\rightarrow$ $m < d$ parameters needed to account for the data properties
hidden causes or *latent variables* $H = [\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_n] \in \mathbb{R}^{m \times n}$

Goal: Find $m < d$ hidden causes $H \in \mathbb{R}^{m \times n}$, that explain the observed data via a (linear) *mixing* $W \in \mathbb{R}^{d \times m}$:

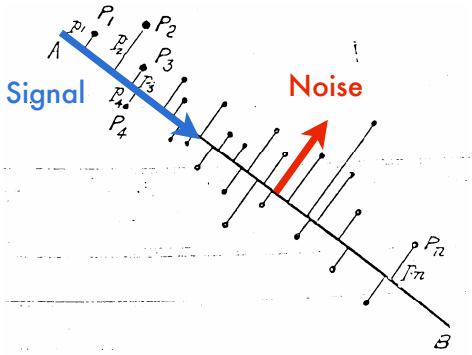
$$X \approx WH$$

Principal component analysis (PCA)



Adapted from Pearson [1901]

Principal component analysis (PCA)



Adapted from Pearson [1901]

Objective of PCA

Find a 1-dimensional subspace $\boldsymbol{w}$ that maximizes the variance of the projected data.

Equivalently: construct an orthonormal subspace such that the projected data and the original data have minimal Euclidean distance. See Bishop [2007, Section 12.1.2]

Maximizing variance

We obtained some data $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$

PCA finds a direction $\boldsymbol{w} \in \mathbb{R}^d$ such that the sample variance of the projected data $\boldsymbol{w}^\top X$ is maximal. Write $\overline{\text{Var}}$ to indicate the variance of a sample.

$$\overline{\text{Var}}(\boldsymbol{w}^T X) = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{w}^\top \boldsymbol{x}_i - \frac{1}{n}\sum_{j=1}^{n}\boldsymbol{w}^\top \boldsymbol{x}_j)^2$$

Maximizing variance

We obtained some data $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$

PCA finds a direction $\boldsymbol{w} \in \mathbb{R}^d$ such that the sample variance of the projected data $\boldsymbol{w}^\top X$ is maximal. Write $\overline{\text{Var}}$ to indicate the variance of a sample.

$$\begin{aligned}\overline{\text{Var}}(\boldsymbol{w}^T X) &= \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{w}^\top \boldsymbol{x}_i - \frac{1}{n}\sum_{j=1}^{n}\boldsymbol{w}^\top \boldsymbol{x}_j)^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{w}^\top \boldsymbol{x}_i - \boldsymbol{w}^\top \frac{1}{n}\sum_{j=1}^{n}\boldsymbol{x}_j)^2 = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{w}^\top (\boldsymbol{x}_i - \bar{\boldsymbol{x}}))^2\end{aligned}$$

Maximizing variance

We obtained some data $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$

PCA finds a direction $\boldsymbol{w} \in \mathbb{R}^d$ such that the sample variance of the projected data $\boldsymbol{w}^\top X$ is maximal. Write $\overline{\text{Var}}$ to indicate the variance of a sample.

$$\begin{aligned}
\overline{\text{Var}}(\boldsymbol{w}^T X) &= \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{w}^\top \boldsymbol{x}_i - \frac{1}{n}\sum_{j=1}^{n}\boldsymbol{w}^\top \boldsymbol{x}_j)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{w}^\top \boldsymbol{x}_i - \boldsymbol{w}^\top \frac{1}{n}\sum_{j=1}^{n}\boldsymbol{x}_j)^2 = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{w}^\top (\boldsymbol{x}_i - \bar{\boldsymbol{x}}))^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{w}^\top (\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^\top \boldsymbol{w}
\end{aligned}$$

Maximizing variance

We obtained some data $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$

PCA finds a direction $\boldsymbol{w} \in \mathbb{R}^d$ such that the sample variance of the projected data $\boldsymbol{w}^\top X$ is maximal. Write $\overline{\text{Var}}$ to indicate the variance of a sample.

$$
\begin{aligned}
\overline{\text{Var}}(\boldsymbol{w}^T X) &= \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{w}^\top \boldsymbol{x}_i - \frac{1}{n}\sum_{j=1}^{n}\boldsymbol{w}^\top \boldsymbol{x}_j)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{w}^\top \boldsymbol{x}_i - \boldsymbol{w}^\top \frac{1}{n}\sum_{j=1}^{n}\boldsymbol{x}_j)^2 = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{w}^\top (\boldsymbol{x}_i - \bar{\boldsymbol{x}}))^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{w}^\top (\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^\top \boldsymbol{w} \\
&= \boldsymbol{w}^\top \underbrace{\left(\frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^\top\right)}_{\text{Empirical covariance matrix } \bar{\Sigma}} \boldsymbol{w}
\end{aligned}
$$

Maximizing variance

We obtained some data $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$

PCA finds a direction $\boldsymbol{w} \in \mathbb{R}^d$ such that the variance of the projected data $\boldsymbol{w}^\top X$ is maximal

[1] i.e. we assume $\bar{\boldsymbol{x}} = 0$

Maximizing variance

We obtained some data $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$

PCA finds a direction $\boldsymbol{w} \in \mathbb{R}^d$ such that the variance of the projected data $\boldsymbol{w}^\top X$ is maximal
Let's assume centered data for easier notation [1]

[1] i.e. we assume $\bar{\boldsymbol{x}} = 0$

Maximizing variance

We obtained some data $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$

PCA finds a direction $\boldsymbol{w} \in \mathbb{R}^d$ such that the variance of the projected data $\boldsymbol{w}^\top X$ is maximal
Let's assume centered data for easier notation [1]

$$\overline{\mathrm{Var}}(\boldsymbol{w}^T X) = \boldsymbol{w}^\top \underbrace{\left(\frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^\top \right)}_{\text{Empirical Covariance matrix } \bar{\Sigma}} \boldsymbol{w}$$

[1] i.e. we assume $\bar{\boldsymbol{x}} = 0$

Maximizing variance

We obtained some data $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$

PCA finds a direction $\boldsymbol{w} \in \mathbb{R}^d$ such that the variance of the projected data $\boldsymbol{w}^\top X$ is maximal
Let's assume centered data for easier notation [1]

$$\begin{aligned} \overline{\mathrm{Var}}(\boldsymbol{w}^T X) &= \boldsymbol{w}^\top \underbrace{\left(\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^\top\right)}_{\text{Empirical Covariance matrix } \bar{\Sigma}} \boldsymbol{w} \\ &\propto \boldsymbol{w}^\top \underbrace{XX^\top}_{\text{Scatter matrix } S} \boldsymbol{w} \end{aligned}$$

[1] i.e. we assume $\bar{\boldsymbol{x}} = 0$

Maximizing variance

We obtained some data $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$

PCA finds a direction $\boldsymbol{w} \in \mathbb{R}^d$ such that the variance of the projected data $\boldsymbol{w}^\top X$ is maximal
Let's assume centered data for easier notation [1]

$$\begin{aligned} \overline{\text{Var}}(\boldsymbol{w}^T X) &= \boldsymbol{w}^\top \underbrace{\left(\frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^\top \right)}_{\text{Empirical Covariance matrix } \bar{\Sigma}} \boldsymbol{w} \\ &\propto \boldsymbol{w}^\top \underbrace{XX^\top}_{\text{Scatter matrix } S} \boldsymbol{w} \end{aligned}$$

We need to constrain $\boldsymbol{w}$ (because we can always make the variance larger by making $\|\boldsymbol{w}\|$ larger).

[1] i.e. we assume $\bar{\boldsymbol{x}} = 0$

Maximizing variance

PCA finds a direction $\boldsymbol{w} \in \mathbb{R}^d$ such that the variance of the projected data $\boldsymbol{w}^T X$ is maximal

$$\underset{\boldsymbol{w}}{\operatorname{argmax}} \frac{\boldsymbol{w}^\top S \boldsymbol{w}}{\boldsymbol{w}^\top \boldsymbol{w}}$$

This objective function is independent of the scaling of $\boldsymbol{w}$.

Maximizing variance

PCA finds a direction $\boldsymbol{w} \in \mathbb{R}^d$ such that the variance of the projected data $\boldsymbol{w}^T X$ is maximal

$$\underset{\boldsymbol{w}}{\operatorname{argmax}} \frac{\boldsymbol{w}^\top S \boldsymbol{w}}{\boldsymbol{w}^\top \boldsymbol{w}}$$

This objective function is independent of the scaling of $\boldsymbol{w}$.

Note the similarity to the objective of Linear Discriminant Analysis!

$\rightarrow$ Different covariance matrices, different problem, but: same maths solve it.

$$\boldsymbol{w}_{\text{LDA}} = \underset{\boldsymbol{w}'}{\operatorname{argmax}}\, J(\boldsymbol{w}') = \underset{\boldsymbol{w}'}{\operatorname{argmax}} \frac{\boldsymbol{w}'^\top S_B \boldsymbol{w}'}{\boldsymbol{w}'^\top S_W \boldsymbol{w}'}$$

Maximizing variance

$$\underset{\boldsymbol{w}}{\text{argmax}} \frac{\boldsymbol{w}^\top S \boldsymbol{w}}{\boldsymbol{w}^\top \boldsymbol{w}} \tag{1}$$

Maximizing variance

$$\underset{\boldsymbol{w}}{\text{argmax}} \frac{\boldsymbol{w}^\top S \boldsymbol{w}}{\boldsymbol{w}^\top \boldsymbol{w}} \tag{1}$$

Set the derivative with respect to $\boldsymbol{w}$ to zero:

Maximizing variance

$$\underset{\boldsymbol{w}}{\operatorname{argmax}} \frac{\boldsymbol{w}^\top S \boldsymbol{w}}{\boldsymbol{w}^\top \boldsymbol{w}} \tag{1}$$

Set the derivative with respect to $\boldsymbol{w}$ to zero:

$$\frac{\partial}{\partial \boldsymbol{w}} \frac{\boldsymbol{w}^\top S \boldsymbol{w}}{\boldsymbol{w}^\top \boldsymbol{w}} = \frac{(\boldsymbol{w}^\top \boldsymbol{w}) 2S\boldsymbol{w} - (\boldsymbol{w}^\top S \boldsymbol{w}) 2\boldsymbol{w}}{(\boldsymbol{w}^\top \boldsymbol{w})^2} \stackrel{!}{=} 0$$

Maximizing variance

$$\underset{\boldsymbol{w}}{\text{argmax}} \frac{\boldsymbol{w}^\top S \boldsymbol{w}}{\boldsymbol{w}^\top \boldsymbol{w}} \tag{1}$$

Set the derivative with respect to $\boldsymbol{w}$ to zero:

$$\frac{\partial}{\partial \boldsymbol{w}} \frac{\boldsymbol{w}^\top S \boldsymbol{w}}{\boldsymbol{w}^\top \boldsymbol{w}} = \frac{(\boldsymbol{w}^\top \boldsymbol{w}) 2S\boldsymbol{w} - (\boldsymbol{w}^\top S \boldsymbol{w}) 2\boldsymbol{w}}{(\boldsymbol{w}^\top \boldsymbol{w})^2} \stackrel{!}{=} 0$$

$$\Rightarrow \underbrace{(\boldsymbol{w}^\top S \boldsymbol{w})}_{\text{scalar}} \boldsymbol{w} = \underbrace{(\boldsymbol{w}^\top \boldsymbol{w})}_{\text{scalar}} S\boldsymbol{w}$$

Maximizing variance

$$\underset{\boldsymbol{w}}{\text{argmax}} \frac{\boldsymbol{w}^\top S \boldsymbol{w}}{\boldsymbol{w}^\top \boldsymbol{w}} \tag{1}$$

Set the derivative with respect to $\boldsymbol{w}$ to zero:

$$\frac{\partial}{\partial \boldsymbol{w}} \frac{\boldsymbol{w}^\top S \boldsymbol{w}}{\boldsymbol{w}^\top \boldsymbol{w}} = \frac{(\boldsymbol{w}^\top \boldsymbol{w}) 2S\boldsymbol{w} - (\boldsymbol{w}^\top S \boldsymbol{w}) 2\boldsymbol{w}}{(\boldsymbol{w}^\top \boldsymbol{w})^2} \stackrel{!}{=} 0$$

$$\Rightarrow \underbrace{(\boldsymbol{w}^\top S \boldsymbol{w})}_{\textit{scalar}} \boldsymbol{w} = \underbrace{(\boldsymbol{w}^\top \boldsymbol{w})}_{\textit{scalar}} S\boldsymbol{w}$$

Maximizing variance

$$\underset{\boldsymbol{w}}{\text{argmax}} \frac{\boldsymbol{w}^\top S \boldsymbol{w}}{\boldsymbol{w}^\top \boldsymbol{w}} \tag{1}$$

Set the derivative with respect to $\boldsymbol{w}$ to zero:

$$\frac{\partial}{\partial \boldsymbol{w}} \frac{\boldsymbol{w}^\top S \boldsymbol{w}}{\boldsymbol{w}^\top \boldsymbol{w}} = \frac{(\boldsymbol{w}^\top \boldsymbol{w}) 2S\boldsymbol{w} - (\boldsymbol{w}^\top S \boldsymbol{w}) 2\boldsymbol{w}}{(\boldsymbol{w}^\top \boldsymbol{w})^2} \stackrel{!}{=} 0$$

$$\Rightarrow \underbrace{(\boldsymbol{w}^\top S \boldsymbol{w})}_{\textit{scalar}} \boldsymbol{w} = \underbrace{(\boldsymbol{w}^\top \boldsymbol{w})}_{\textit{scalar}} S\boldsymbol{w}$$

$$\Rightarrow \underbrace{\frac{\boldsymbol{w}^\top S \boldsymbol{w}}{\boldsymbol{w}^\top \boldsymbol{w}}}_{\equiv \lambda} \boldsymbol{w} = S\boldsymbol{w}$$

Maximizing variance

$$\underset{\boldsymbol{w}}{\operatorname{argmax}} \frac{\boldsymbol{w}^\top S \boldsymbol{w}}{\boldsymbol{w}^\top \boldsymbol{w}} \tag{1}$$

Set the derivative with respect to $\boldsymbol{w}$ to zero:

$$\frac{\partial}{\partial \boldsymbol{w}} \frac{\boldsymbol{w}^\top S \boldsymbol{w}}{\boldsymbol{w}^\top \boldsymbol{w}} = \frac{(\boldsymbol{w}^\top \boldsymbol{w}) 2S\boldsymbol{w} - (\boldsymbol{w}^\top S \boldsymbol{w}) 2\boldsymbol{w}}{(\boldsymbol{w}^\top \boldsymbol{w})^2} \stackrel{!}{=} 0$$

$$\Rightarrow \underbrace{(\boldsymbol{w}^\top S \boldsymbol{w})}_{\text{scalar}} \boldsymbol{w} = \underbrace{(\boldsymbol{w}^\top \boldsymbol{w})}_{\text{scalar}} S\boldsymbol{w}$$

$$\Rightarrow \underbrace{\frac{\boldsymbol{w}^\top S \boldsymbol{w}}{\boldsymbol{w}^\top \boldsymbol{w}}}_{\equiv \lambda} \boldsymbol{w} = S\boldsymbol{w}$$

Equation 1 can be reduced to the standard eigenvalue problem:

$$S\boldsymbol{w} = \lambda \boldsymbol{w}$$

Maximizing variance

Setting $S\boldsymbol{w} = \lambda \boldsymbol{w}$ in Eq. 1, we see that the variance in direction $\boldsymbol{w}$ is given by:

$$\underset{\boldsymbol{w}}{\text{argmax}} \frac{\boldsymbol{w}^\top S \boldsymbol{w}}{\boldsymbol{w}^\top \boldsymbol{w}} = \frac{\boldsymbol{w}^\top \lambda \boldsymbol{w}}{\boldsymbol{w}^\top \boldsymbol{w}} = \lambda$$

Conclusion

- The variance of the projected data in the maximizing direction $\boldsymbol{w}$ is given by the corresponding eigenvalue

Maximizing variance

Setting $S\boldsymbol{w} = \lambda \boldsymbol{w}$ in Eq. 1, we see that the variance in direction $\boldsymbol{w}$ is given by:

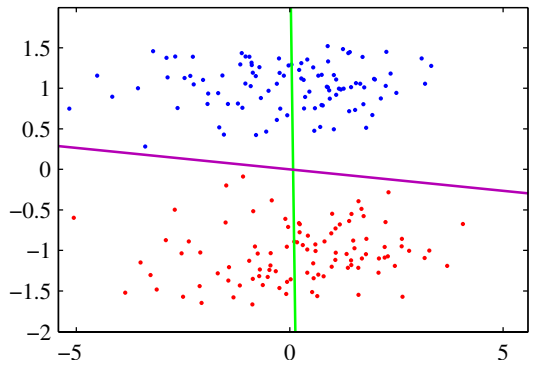
$$\underset{\boldsymbol{w}}{\text{argmax}} \frac{\boldsymbol{w}^\top S \boldsymbol{w}}{\boldsymbol{w}^\top \boldsymbol{w}} = \frac{\boldsymbol{w}^\top \lambda \boldsymbol{w}}{\boldsymbol{w}^\top \boldsymbol{w}} = \lambda$$

Conclusion

- The variance of the projected data in the maximizing direction $\boldsymbol{w}$ is given by the corresponding eigenvalue
- The direction of maximal variance in the data is equal to the eigenvector having the largest eigenvalue.

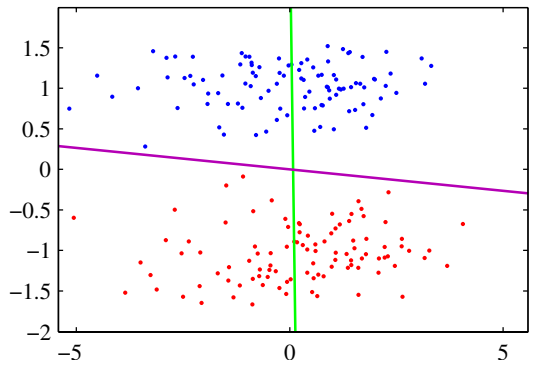
PCA vs. LDA

Which is which and why?



PCA vs. LDA

Which is which and why?



Directions found by PCA (magenta) and LDA(green) Bishop [2007]

Finding more principal components

Incremental PCA finds additional principal components, by looking at directions **orthogonal** to previous ones, that maximize variance.

Finding more principal components

Incremental PCA finds additional principal components, by looking at directions **orthogonal** to previous ones, that maximize variance.

Characterization of PCA

The k first PCA basis vectors are the eigenvectors corresponding to the largest k eigenvalues

$$SW = W\Lambda,$$

where $W = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_k]$ contains the eigenvectors sorted according to their eigenvalues and Λ is a diagonal matrix containing the corresponding eigenvalues.

Recall (from linear algebra): since S is symmetric, there exist d orthogonal eigenvectors ($\boldsymbol{w}_i \perp \boldsymbol{w}_j$) and the eigenvalues are real numbers.

Encoding (from real data to latent representation)

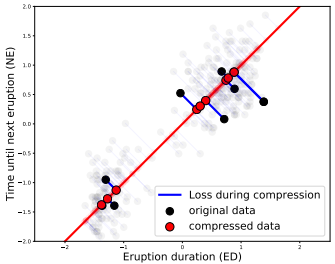
Now that we have $W = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_k] \in \mathbb{R}^{d \times k}$, we project each data point $\boldsymbol{x}$ onto W

$$h_i = \begin{bmatrix} \boldsymbol{w}_1^T \boldsymbol{x}_i \\ \vdots \\ \boldsymbol{w}_k^T \boldsymbol{x}_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{w}_1^T \\ \vdots \\ \boldsymbol{w}_k^T \end{bmatrix} \boldsymbol{x}_i = W^T \cdot \boldsymbol{x}_i$$

Encoding (from real data to latent representation)

Now that we have $W = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_k] \in \mathbb{R}^{d \times k}$, we project each data point $\boldsymbol{x}$ onto W

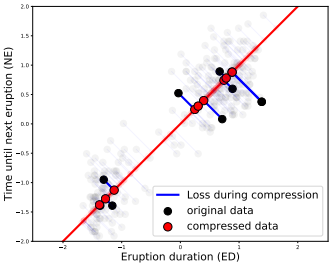
$$h_i = \begin{bmatrix} \boldsymbol{w}_1^T \boldsymbol{x}_i \\ \vdots \\ \boldsymbol{w}_k^T \boldsymbol{x}_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{w}_1^T \\ \vdots \\ \boldsymbol{w}_k^T \end{bmatrix} \boldsymbol{x}_i = W^T \cdot \boldsymbol{x}_i$$

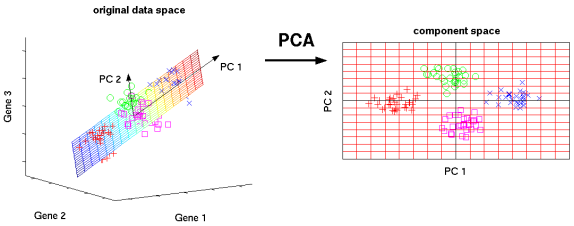


Encoding (from real data to latent representation)

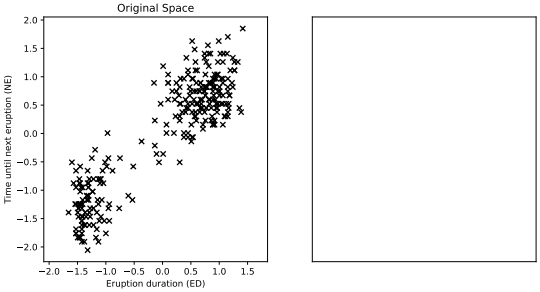
Now that we have $W = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_k] \in \mathbb{R}^{d \times k}$, we project each data point $\boldsymbol{x}$ onto W

$$h_i = \begin{bmatrix} \boldsymbol{w}_1^T \boldsymbol{x}_i \\ \vdots \\ \boldsymbol{w}_k^T \boldsymbol{x}_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{w}_1^T \\ \vdots \\ \boldsymbol{w}_k^T \end{bmatrix} \boldsymbol{x}_i = W^T \cdot \boldsymbol{x}_i$$



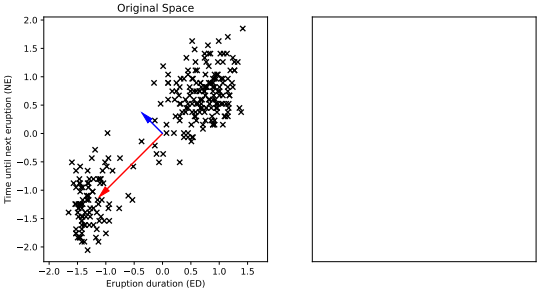


Principal Component Analysis



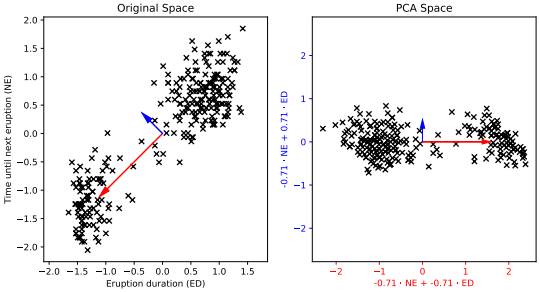
- PCA basis has maximum variance directions on basis vectors
  - Principal components $\boldsymbol{w}_j$ and latent representation h_{ij} do not change with increasing k

Principal Component Analysis



- PCA basis has maximum variance directions on basis vectors
  - Principal components $\boldsymbol{w}_j$ and latent representation h_{ij} do not change with increasing k

Principal Component Analysis



- PCA basis has maximum variance directions on basis vectors
  - Principal components $\boldsymbol{w}_j$ and latent representation h_{ij} do not change with increasing k

Summary: Principal Component Analysis

1. Estimate the covariance matrix S of the data $X \in \mathbb{R}^{d \times n}$
2. Compute the eigenvectors of S
3. $W = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_k] \in \mathbb{R}^{d \times k}$ where $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k \in \mathbb{R}^d$ are the eigenvectors corresponding to the k largest eigenvalues
4. Project the data onto W: $H = W^\top \cdot X$
5. If needed: reconstruct data by $X \approx \tilde{X} = WH$.
   This holds for all linear matrix factorization methods.

Summary: Principal Component Analysis

Algorithm 1: Principal Component Analysis

Require: data $x_1, \ldots, x_n \in \mathbb{R}^d$, number of principal components k

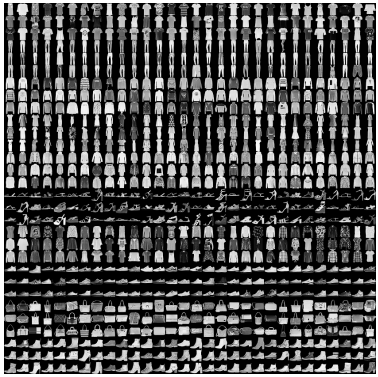
1: # Compute Sample Covariance Matrix
2: $C = 1/n\ (X - 1/n \sum_i x_i)(X - 1/n \sum_i x_i)^\top$
3: # Compute eigenvectors corresponding to the k largest eigenvalues
4: $W = \text{eig}(C)$
5: # Project data onto W
6: $H = W^\top X$
7: **return** W, H

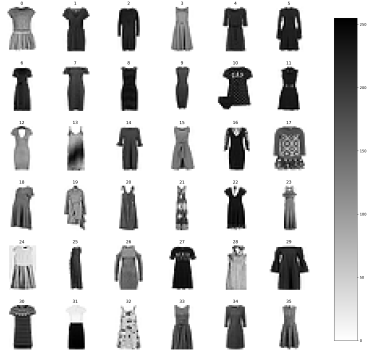
PCA on Fashion MNIST

Fashion-MNIST is a dataset of Zalando's article images consisting of 70,000 examples. Each example is a 28×28 grayscale image, associated with a label from 10 classes.

Fashion-MNIST on GitHub

PCA on Fashion MNIST





PCA on Fashion MNIST: Shirts

Figure: Eigenvectors (W)

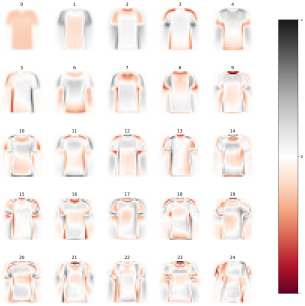
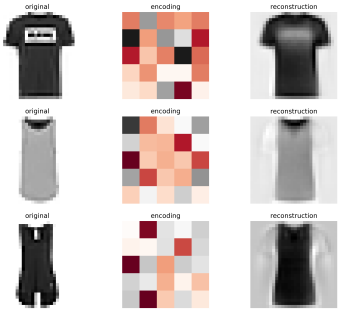


Figure: 3 examples (X, H, WH)



PCA on Fashion MNIST: Shirts

Figure: Eigenvectors (W)

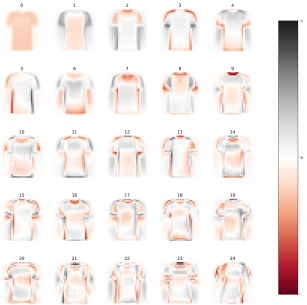


Figure: 3 examples (X, H, WH)

- Here we compress images to 25 dimensions instead of 784 (28×28), the compression ratio is 31.36.

PCA for high-dimensional data

Input: centered data $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$ with $n \ll d$

- Covariance matrix $XX^\top$ will be very large (d-by-d)
- Too few samples for a robust covariance matrix estimate

PCA for high-dimensional data

Input: centered data $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$ with $n \ll d$

- Covariance matrix $XX^\top$ will be very large (d-by-d)
- Too few samples for a robust covariance matrix estimate

We know the direction of maximal variance $\boldsymbol{w}$ must lie in the span of the data (for $\lambda \neq 0$):

PCA for high-dimensional data

Input: centered data $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$ with $n \ll d$

- Covariance matrix $XX^\top$ will be very large (d-by-d)
- Too few samples for a robust covariance matrix estimate

We know the direction of maximal variance $\boldsymbol{w}$ must lie in the span of the data (for $\lambda \neq 0$):

$$\lambda \boldsymbol{w} = XX^\top \boldsymbol{w} \;\rightarrow\; \boldsymbol{w} = X\boldsymbol{\alpha},$$

PCA for high-dimensional data

Input: centered data $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$ with $n \ll d$

- Covariance matrix $XX^\top$ will be very large (d-by-d)
- Too few samples for a robust covariance matrix estimate

We know the direction of maximal variance $\boldsymbol{w}$ must lie in the span of the data (for $\lambda \neq 0$):

$$\lambda \boldsymbol{w} = XX^\top \boldsymbol{w} \;\rightarrow\; \boldsymbol{w} = X\boldsymbol{\alpha},$$

where $\boldsymbol{\alpha} = 1/\lambda \cdot X^\top \boldsymbol{w}$ is a weighting of each data point

PCA for high-dimensional data

We can plug $\boldsymbol{w} = X\boldsymbol{\alpha}$ in and obtain

$$XX^\top \boldsymbol{w} = \lambda \boldsymbol{w}$$

PCA for high-dimensional data

We can plug $\boldsymbol{w} = X\boldsymbol{\alpha}$ in and obtain

$$\begin{aligned} XX^\top \boldsymbol{w} &= \lambda \boldsymbol{w} \\ XX^\top X\boldsymbol{\alpha} &= \lambda X\boldsymbol{\alpha} \end{aligned}$$

PCA for high-dimensional data

We can plug $\boldsymbol{w} = X\boldsymbol{\alpha}$ in and obtain

$$\begin{aligned} XX^\top \boldsymbol{w} &= \lambda \boldsymbol{w} \\ XX^\top X\boldsymbol{\alpha} &= \lambda X\boldsymbol{\alpha} \\ \underbrace{X^\top X}_{K}\underbrace{X^\top X}_{K}\boldsymbol{\alpha} &= \lambda \underbrace{X^\top X}_{K}\boldsymbol{\alpha} \end{aligned}$$

PCA for high-dimensional data

We can plug $\boldsymbol{w} = X\boldsymbol{\alpha}$ in and obtain

$$\begin{aligned} XX^\top \boldsymbol{w} &= \lambda \boldsymbol{w} \\ XX^\top X\boldsymbol{\alpha} &= \lambda X\boldsymbol{\alpha} \\ \underbrace{X^\top X}_{K}\underbrace{X^\top X}_{K}\boldsymbol{\alpha} &= \lambda \underbrace{X^\top X}_{K}\boldsymbol{\alpha} \\ KK\boldsymbol{\alpha} &= \lambda K\boldsymbol{\alpha} \end{aligned}$$

which can be solved by [Schölkopf et al., 1998]

$$K\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}.$$

PCA for high-dimensional data

We can plug $\boldsymbol{w} = X\boldsymbol{\alpha}$ in and obtain

$$\begin{aligned} XX^\top \boldsymbol{w} &= \lambda \boldsymbol{w} \\ XX^\top X\boldsymbol{\alpha} &= \lambda X\boldsymbol{\alpha} \\ \underbrace{X^\top X}_{K}\underbrace{X^\top X}_{K}\boldsymbol{\alpha} &= \lambda \underbrace{X^\top X}_{K}\boldsymbol{\alpha} \\ KK\boldsymbol{\alpha} &= \lambda K\boldsymbol{\alpha} \end{aligned}$$

which can be solved by [Schölkopf et al., 1998]

$$K\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}.$$

Solving PCA via $X^\top X$ instead of $XX^\top$ is called **linear kernel PCA**.

PCA for high-dimensional data

We can plug $\boldsymbol{w} = X\boldsymbol{\alpha}$ in and obtain

$$\begin{aligned} XX^\top \boldsymbol{w} &= \lambda \boldsymbol{w} \\ XX^\top X\boldsymbol{\alpha} &= \lambda X\boldsymbol{\alpha} \\ \underbrace{X^\top X}_{K}\underbrace{X^\top X}_{K}\boldsymbol{\alpha} &= \lambda \underbrace{X^\top X}_{K}\boldsymbol{\alpha} \\ KK\boldsymbol{\alpha} &= \lambda K\boldsymbol{\alpha} \end{aligned}$$

which can be solved by [Schölkopf et al., 1998]

$$K\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}.$$

Solving PCA via $X^\top X$ instead of $XX^\top$ is called **linear kernel PCA**.

Note: if we want to use other kernels, we have to take care that data is centered in feature space, see Schölkopf et al. [1998, Appendix B] or Bishop [2007, Section 12.3].

When to use high-dimensional PCA

- If there are more dimensions than samples ($n \ll d$)
  Compute PCA on linear kernel matrix $X^\top X \in \mathbb{R}^{n \times n}$
- If there are more samples than dimensions ($d \ll n$)
  Compute PCA on covariance matrix $XX^\top \in \mathbb{R}^{d \times d}$

Wrap-up: Linear Kernel PCA

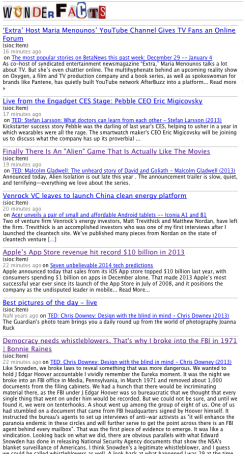
Algorithm 2: Linear Kernel PCA

Require: data $x_1, \ldots, x_n \in \mathbb{R}^d$, $n \ll d$, number of principal components k

1: # Compute Linear Kernel
2: $K = (X - 1/n \sum_i x_i)^\top (X - 1/n \sum_i x_i)$
3: # Compute eigenvectors corresponding to the k largest eigenvalues
4: $\alpha = \text{eig}(K)$
5: $W = X\alpha$
6: # Project data onto W
7: $H = W^\top X$
8: **return** W, H

Trends in text data

Let's look at some more applications of PCA!



Trends in text data

We are looking at bag-of-words data from a news web page

- We store the data in a matrix
  $X \in \mathbb{R}^{d \times t}$
  The entry $X_{i,j} = 10$ means: word i
  was counted 10 times in time bin j

Trends in text data

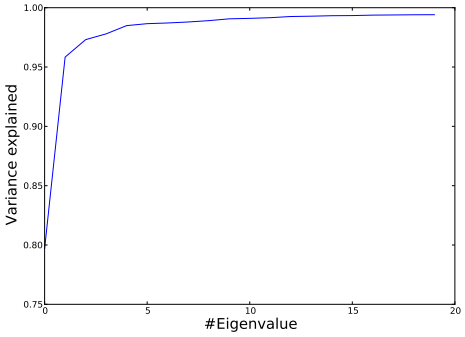
We are looking at bag-of-words data from a news web page

- We store the data in a matrix $X \in \mathbb{R}^{d \times t}$
  The entry $X_{i,j} = 10$ means: word i was counted 10 times in time bin j
- Let's apply PCA and look at the *variance explained* ($\text{EV}_m = \frac{\sum_{i=1}^{m} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$)

Trends in text data

We are looking at bag-of-words data from a news web page

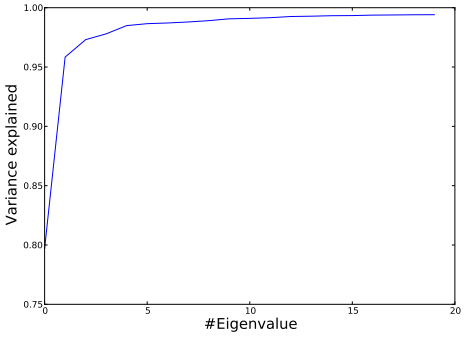
- We store the data in a matrix $X \in \mathbb{R}^{d \times t}$
  The entry $X_{i,j} = 10$ means: word i was counted 10 times in time bin j
- Let's apply PCA and look at the *variance explained* ($\mathrm{EV}_m = \frac{\sum_{i=1}^{m} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$)



Trends in text data

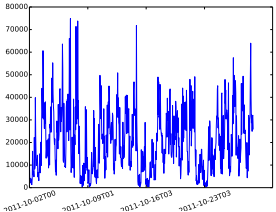
We are looking at bag-of-words data from a news web page

- We store the data in a matrix $X \in \mathbb{R}^{d \times t}$
  The entry $X_{i,j} = 10$ means: word i was counted 10 times in time bin j
- Let's apply PCA and look at the *variance explained* ($\mathrm{EV}_m = \frac{\sum_{i=1}^{m} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$)
- We only need 15 principal directions to "explain" >99% of the data



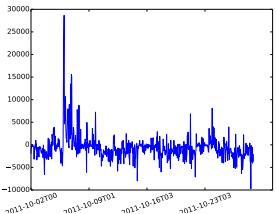
Trends in text data

First principal component



Main variance due to weekly/daily publishing activity

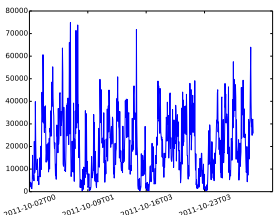
Second principal component



Spike on day of Steve Job's death

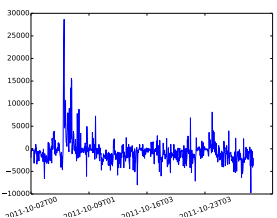
Trends in text data

First principal component



Main variance due to weekly/daily publishing activity

Second principal component



Spike on day of Steve Job's death

We can use PCA as a tool for analyzing big unlabeled data.

Non-negative matrix factorization (NMF)

- For some data PCA is not intuitive

Non-negative matrix factorization (NMF)

- For some data PCA is not intuitive
- Example: Non-negative data

Non-negative matrix factorization (NMF)

- For some data PCA is not intuitive
- Example: Non-negative data
  - Principal directions will have negative entries

Non-negative matrix factorization (NMF)

- For some data PCA is not intuitive
- Example: Non-negative data
  - Principal directions will have negative entries
  - This can be hard to interpret

Non-negative matrix factorization (NMF)

- For some data PCA is not intuitive
- Example: Non-negative data
  - Principal directions will have negative entries
  - This can be hard to interpret
- Many data sets are strictly non-negative

Non-negative matrix factorization (NMF)

- For some data PCA is not intuitive
- Example: Non-negative data
  - Principal directions will have negative entries
  - This can be hard to interpret
- Many data sets are strictly non-negative
  - Text data

Non-negative matrix factorization (NMF)

- For some data PCA is not intuitive
- Example: Non-negative data
  - Principal directions will have negative entries
  - This can be hard to interpret
- Many data sets are strictly non-negative
  - Text data
  - Image data

Non-negative matrix factorization (NMF)

- For some data PCA is not intuitive
- Example: Non-negative data
  - Principal directions will have negative entries
  - This can be hard to interpret
- Many data sets are strictly non-negative
  - Text data
  - Image data
  - Probabilistic data

Non-negative matrix factorization (NMF)

- For some data PCA is not intuitive
- Example: Non-negative data
  - Principal directions will have negative entries
  - This can be hard to interpret
- Many data sets are strictly non-negative
  - Text data
  - Image data
  - Probabilistic data
- NMF is straightforward to implement

Non-negative matrix factorization (NMF)

- For some data PCA is not intuitive
- Example: Non-negative data
  - Principal directions will have negative entries
  - This can be hard to interpret
- Many data sets are strictly non-negative
  - Text data
  - Image data
  - Probabilistic data
- NMF is straightforward to implement
- Matrix factorization is relevant in recommender systems ("you might also like...") (more info here)

Non-negative matrix factorization

Notation: $\mathbb{R}_+ := \{x \in \mathbb{R} \,|\, x \geq 0\}$.
Given non-negative data $X \in \mathbb{R}_+^{d \times n}$ we want to find $W \in \mathbb{R}_+^{d \times m}$, $H \in \mathbb{R}_+^{m \times n}$ such that the distance between X and WH is minimal, where distance is measured as the Frobenius norm.
W and H are given by

$$\operatorname*{argmin}_{W,H} \|X - WH\|_{\text{Fro}}^2,$$

which is by definition of $\|\cdot\|_{\text{Fro}}$

$$= \operatorname*{argmin}_{W,H} \sum_{i=1}^{d} \sum_{j=1}^{n} (X_{ij} - (WH)_{ij})^2 .$$

Non-negative matrix factorization

$$\underset{W,H}{\operatorname{argmin}} \|X - WH\|_{\text{Fro}}^2 \qquad \text{such that } W \geq 0 \text{ and } H \geq 0 \text{ entry-wise}$$

Note that the constraints make the problem NP-hard and ill-posed [Gillis, 2014]. In particular, there are in general infinitely many solutions.

Non-negative matrix factorization

$$\underset{W,H}{\operatorname{argmin}} \|X - WH\|_{\text{Fro}}^2 \qquad \text{such that } W \geq 0 \text{ and } H \geq 0 \text{ entry-wise}$$

Note that the constraints make the problem NP-hard and ill-posed [Gillis, 2014]. In particular, there are in general infinitely many solutions.

Note

Whenever the problem is too hard, just follow the gradients!

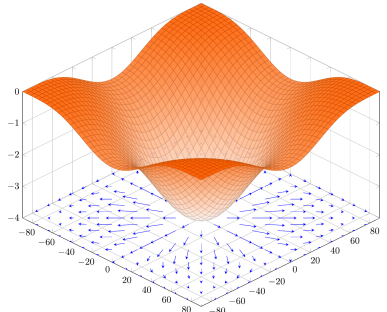
Non-negative matrix factorization

$$\underset{W,H}{\operatorname{argmin}} \|X - WH\|_{\text{Fro}}^2 \qquad \text{such that } W \geq 0 \text{ and } H \geq 0 \text{ entry-wise}$$

Note that the constraints make the problem NP-hard and ill-posed [Gillis, 2014]. In particular, there are in general infinitely many solutions.

Note

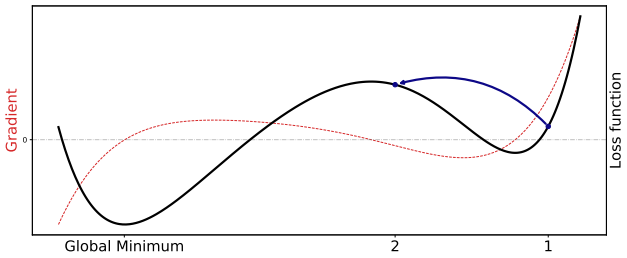
Whenever the problem is too hard, just follow the gradients!



Gradient descent

Gradient descent finds **an** optimum of the loss function $\mathcal{L}(\alpha)$ by iterating

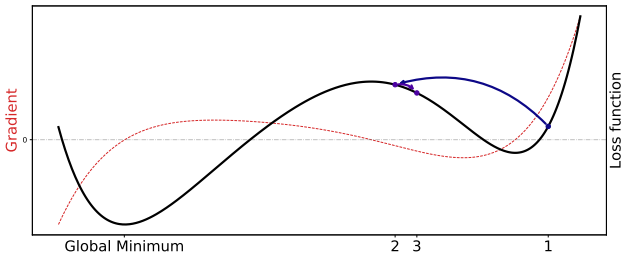
$$\alpha \leftarrow \alpha - \eta \frac{\partial \mathcal{L}(\alpha)}{\partial \alpha}$$



Gradient descent

Gradient descent finds **an** optimum of the loss function $\mathcal{L}(\alpha)$ by iterating

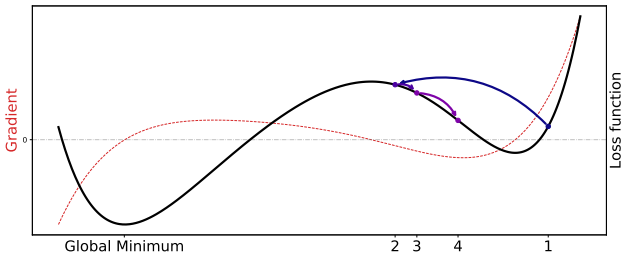
$$\alpha \leftarrow \alpha - \eta \frac{\partial \mathcal{L}(\alpha)}{\partial \alpha}$$



Gradient descent

Gradient descent finds **an** optimum of the loss function $\mathcal{L}(\alpha)$ by iterating

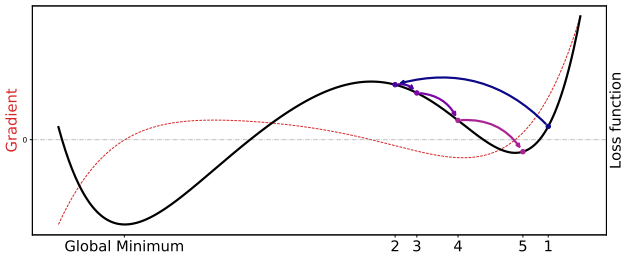
$$\alpha \leftarrow \alpha - \eta \frac{\partial \mathcal{L}(\alpha)}{\partial \alpha}$$



Gradient descent

Gradient descent finds **an** optimum of the loss function $\mathcal{L}(\alpha)$ by iterating

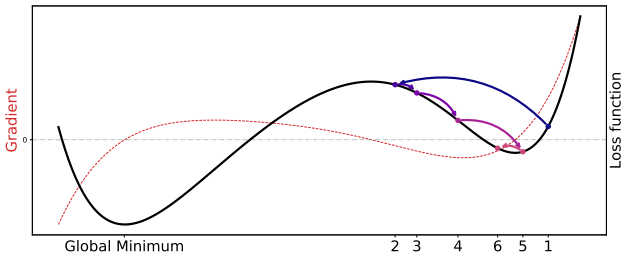
$$\alpha \leftarrow \alpha - \eta \frac{\partial \mathcal{L}(\alpha)}{\partial \alpha}$$



Gradient descent

Gradient descent finds **an** optimum of the loss function $\mathcal{L}(\alpha)$ by iterating

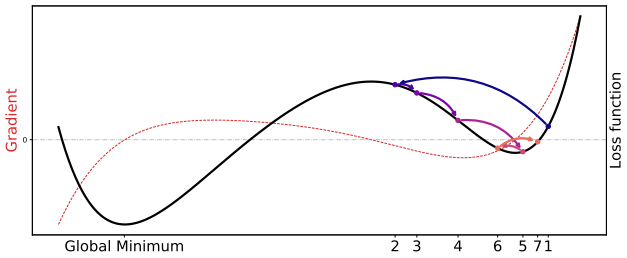
$$\alpha \leftarrow \alpha - \eta \frac{\partial \mathcal{L}(\alpha)}{\partial \alpha}$$



Gradient descent

Gradient descent finds **an** optimum of the loss function $\mathcal{L}(\alpha)$ by iterating

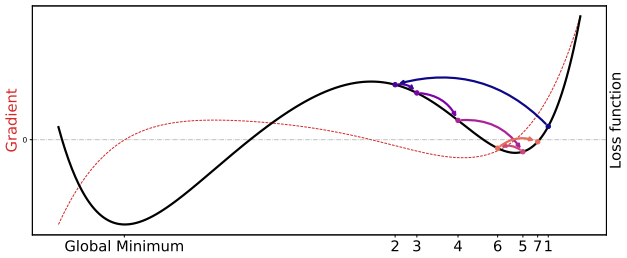
$$\alpha \leftarrow \alpha - \eta \frac{\partial \mathcal{L}(\alpha)}{\partial \alpha}$$

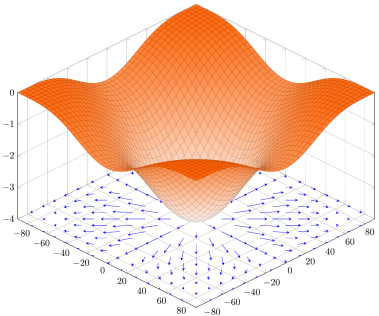


Gradient descent

Gradient descent finds **an** optimum of the loss function $\mathcal{L}(\alpha)$ by iterating

$$\alpha \leftarrow \alpha - \eta \frac{\partial \mathcal{L}(\alpha)}{\partial \alpha}$$



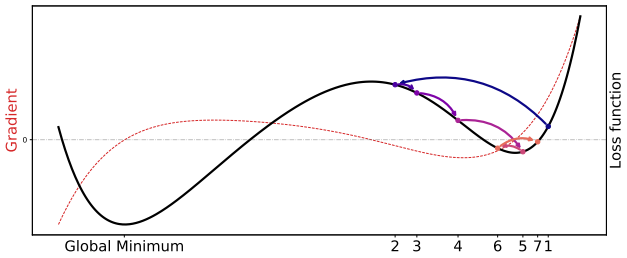


from Wikipedia

Gradient descent

Gradient descent finds **an** optimum of the loss function $\mathcal{L}(\alpha)$ by iterating

$$\alpha \leftarrow \alpha - \eta \frac{\partial \mathcal{L}(\alpha)}{\partial \alpha}$$



- **Not guaranteed to find global optimum**
- Adaptive η can yield improvements
- Stochastic gradient descent is basis for state-of-the-art machine learning (deep learning)

Let's do the calculations

Useful formulas (to be understood entry-wise)

$$\frac{\partial}{\partial X}\|X\|_{\text{Fro}}^2 = \frac{\partial}{\partial X}\operatorname{Tr}(XX^T)$$

$$\operatorname{Tr}(A) = \sum_i A_{ii}$$

$$\frac{\partial \operatorname{Tr}(AXB)}{\partial X} = \frac{\partial \operatorname{Tr}(B^T X^T A^T)}{\partial X} = A^T B^T$$

$$\frac{\partial \operatorname{Tr}(AXX^T A^T)}{\partial X} = 2A^T AX$$

$$\frac{\partial \operatorname{Tr}(XAX^T)}{\partial X} = XA^T + XA$$

$$\frac{\partial \|X - WH\|_{\text{Fro}}^2}{\partial H} =$$

$$\frac{\partial \|X - WH\|_{\text{Fro}}^2}{\partial W} =$$

Let's do the calculations

Useful formulas (to be understood entry-wise)

$$\frac{\partial}{\partial X}\|X\|_{\text{Fro}}^2 = \frac{\partial}{\partial X}\operatorname{Tr}(XX^T)$$

$$\operatorname{Tr}(A) = \sum_i A_{ii}$$

$$\frac{\partial \operatorname{Tr}(AXB)}{\partial X} = \frac{\partial \operatorname{Tr}(B^T X^T A^T)}{\partial X} = A^T B^T$$

$$\frac{\partial \operatorname{Tr}(AXX^T A^T)}{\partial X} = 2A^T AX$$

$$\frac{\partial \operatorname{Tr}(XAX^T)}{\partial X} = XA^T + XA$$

$$\frac{\partial \|X - WH\|_{\text{Fro}}^2}{\partial H} = 2(W^T WH - W^T X)$$

$$\frac{\partial \|X - WH\|_{\text{Fro}}^2}{\partial W} =$$

Let's do the calculations

Useful formulas (to be understood entry-wise)

$$\frac{\partial}{\partial X}\|X\|_{\text{Fro}}^2 = \frac{\partial}{\partial X}\operatorname{Tr}(XX^T)$$

$$\operatorname{Tr}(A) = \sum_i A_{ii}$$

$$\frac{\partial \operatorname{Tr}(AXB)}{\partial X} = \frac{\partial \operatorname{Tr}(B^T X^T A^T)}{\partial X} = A^T B^T$$

$$\frac{\partial \operatorname{Tr}(AXX^T A^T)}{\partial X} = 2A^T AX$$

$$\frac{\partial \operatorname{Tr}(XAX^T)}{\partial X} = XA^T + XA$$

$$\frac{\partial \|X - WH\|_{\text{Fro}}^2}{\partial H} = 2(W^T WH - W^T X)$$

$$\frac{\partial \|X - WH\|_{\text{Fro}}^2}{\partial W} = 2(WHH^T - XH^T)$$

Non-negative matrix factorization

Gradient descent finds a locally optimal solution by iterating

$$H \leftarrow H - \eta \left(W^\top WH - W^\top X \right)$$

$$W \leftarrow W - \eta \left(WHH^\top - XH^\top \right)$$

[2] Note that convergence has to be shown for these η.

Non-negative matrix factorization

Gradient descent finds a locally optimal solution by iterating

$$H \leftarrow H - \eta \left(W^\top WH - W^\top X \right)$$
$$W \leftarrow W - \eta \left(WHH^\top - XH^\top \right)$$

- By choosing $\eta_{ij}^H := \frac{H_{ij}}{(W^\top WH)_{ij}}$, $\eta_{ij}^W := \frac{W_{ij}}{(WHH^\top)_{ij}}$ one can transform the additive update into a multiplicative update[2] [Lee and Seung, 1999, 2000]:

$$H_{ij} \leftarrow H_{ij} \frac{(W^\top X)_{ij}}{(W^\top WH)_{ij}}$$
$$W_{ij} \leftarrow W_{ij} \frac{(XH^\top)_{ij}}{(WHH^\top)_{ij}}.$$

[2] Note that convergence has to be shown for these η.

NMF algorithm

Algorithm 3: Non-negative Matrix Factorization

Require: data $X = [x_1, \ldots, x_n] \in \mathbb{R}_+^{d \times n}$, number of factors k

1: # Initialize $W \in \mathbb{R}_+^{d \times k}$, $H \in \mathbb{R}_+^{k \times n}$ randomly
2: # Add a small constant $\epsilon = 10^{-19}$ to X to avoid zero-divisions
3: **for** it $\leq$ Iterations **do**
4: $\quad H = H \odot W^\top X \oslash W^\top WH$
5: $\quad W = W \odot XH^\top \oslash WHH^\top$
6: **end for**
7: **return** W, H

where

- $\odot$ is *element-wise* multiplication ($*$ in numpy)
- $\oslash$ is *element-wise* division ($/$ in numpy)

NMF on fashion MNIST: Shirts

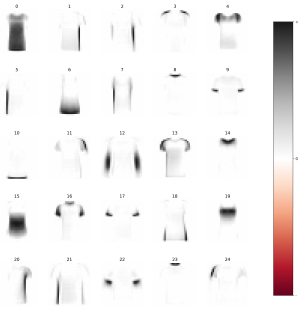


Figure: 25 main components (W)

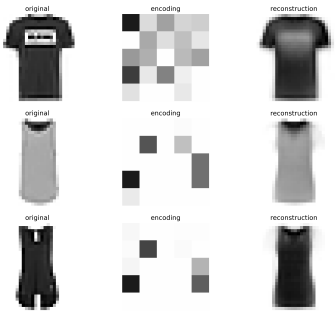


Figure: 3 examples (X, H, WH)

PCA vs. NMF - Fashion MNIST

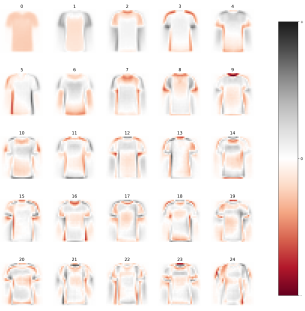


Figure: W_{PCA}

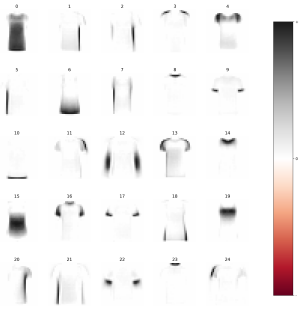
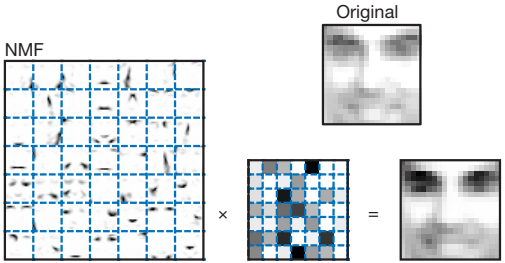
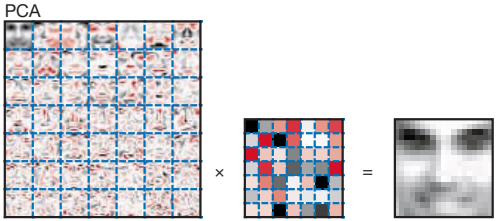


Figure: W_{NMF}

PCA vs. NMF - Face Parts

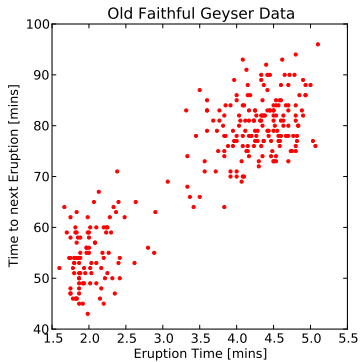




Taken from Lee and Seung [1999]

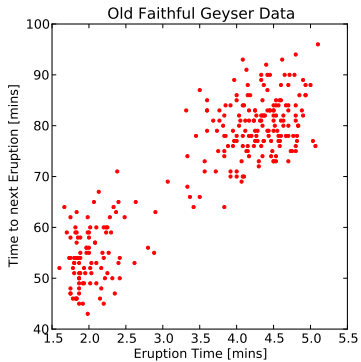
Clustering

- PCA/NMF are applied to reduce dimensionality



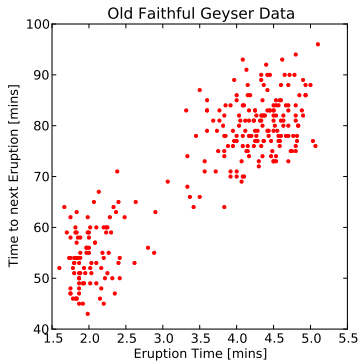
Clustering

- PCA/NMF are applied to reduce dimensionality
- Often the problem setting is different:



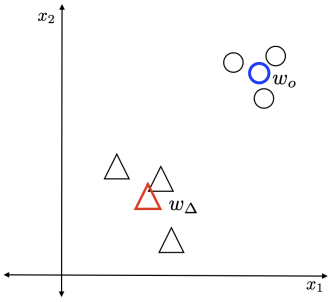
Clustering

- PCA/NMF are applied to reduce dimensionality
- Often the problem setting is different:
  - You want to categorize eruptions without labels



Clustering

Remember Lecture 2:
Psychological Models of Categorization: Prototypes



Prototypes $\boldsymbol{\mu}_\Delta$ and $\boldsymbol{\mu}_o$:

$$\boldsymbol{\mu}_\Delta = 1/n_\Delta \sum_{n}^{n_\Delta} \boldsymbol{x}_{\Delta,n}$$

$$\boldsymbol{\mu}_o = 1/n_o \sum_{n}^{n_o} \boldsymbol{x}_{o,n}$$

New data points $\boldsymbol{x}$ are assigned to their closest cluster center $\boldsymbol{\mu}^*$

$$\boldsymbol{\mu}^* = \underset{i}{\text{argmin}}(\|\boldsymbol{\mu}_i - \boldsymbol{x}\|_2)$$

K-means clustering

Objective for k-means

Find cluster centers $\boldsymbol{\mu}_1, \dots \boldsymbol{\mu}_k$ such that the sum of distances of data points to their respective cluster centers ("WCSS", "WSS"[3]) is minimized

$$L(\{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_k\}, r) = \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{\mu}_{r_i}\|^2$$

where r_i : cluster index of data point i

[3] Within cluster sum of squares

K-means clustering

Objective for k-means

Find cluster centers $\boldsymbol{\mu}_1, \dots \boldsymbol{\mu}_k$ such that the sum of distances of data points to their respective cluster centers ("WCSS", "WSS"[3]) is minimized

$$L(\{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_k\}, r) = \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{\mu}_{r_i}\|^2$$

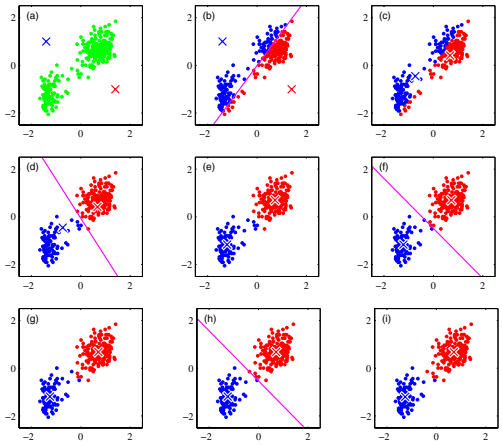
where r_i : cluster index of data point i

We minimize L by re-iterating two steps

1. Assign each data point $\boldsymbol{x}_i$ to their closest cluster $\boldsymbol{\mu}_{r_i}$
2. Update $\boldsymbol{\mu}_r$ to the mean of the members in that cluster r

[3] Within cluster sum of squares

K-means clustering step-by-step



Re-iterate two steps:

1. Assign each $\boldsymbol{x}_i$ to closest cluster $\boldsymbol{\mu}_{r_i}$
2. Update $\boldsymbol{\mu}_r$ to mean of members in cluster r

K-means clustering algorithm

Require: data $x_1, \ldots, x_n \in \mathbb{R}^d$, number of clusters k, iterations m.

Choose random data points as initial cluster centers $\boldsymbol{\mu}_1 \leftarrow x_{i_1}, \ldots, \boldsymbol{\mu}_k \leftarrow x_{i_k}$ where $i_j \neq i_l$ for all $j \neq l$.
$r \leftarrow \mathbf{0}_n$
$r' \leftarrow \mathbf{0}_n$
$i \leftarrow 0$
while $i < m$ **do**
    for $j \leftarrow 1$ to n **do**
        Find nearest cluster center $r'_j \leftarrow \text{argmin}_{1 \leq l \leq k} \|\boldsymbol{x}_j - \boldsymbol{\mu}_l\|_2$
    end for
    for $j \leftarrow 1$ to k **do**
        Compute new cluster center $\boldsymbol{\mu}_j \leftarrow \frac{1}{|\{l : r'_l = j\}|} \sum_{l : r'_l = j} x_l$
    end for
    if $r = r'$ **then**
        break
    end if
    $r \leftarrow r'$
    $i \leftarrow i + 1$
end while
return cluster centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k \in \mathbb{R}^d$, assignment vector $r \in \mathbb{R}^n$

Application example: image compression

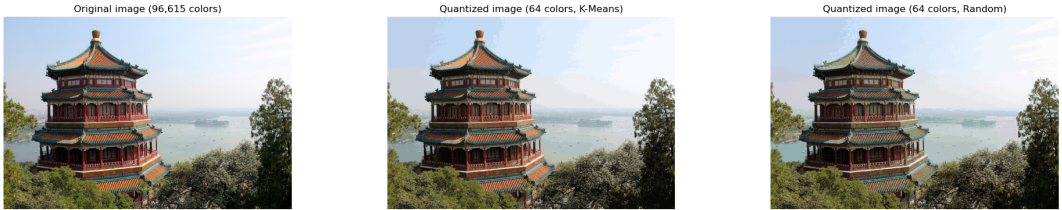
Original image (96,615 colors)

Quantized image (64 colors, K-Means)

Quantized image (64 colors, Random)

Adapted from sklearn, more information here.

Application example: image compression

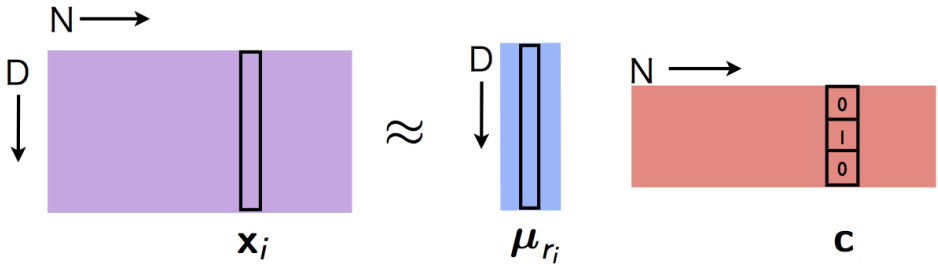


Adapted from sklearn, more information here.

- Encode color with 6 bit instead of 24 bit
- Only need one quarter of the bandwidth (and dictionary of colors)

Clustering can be seen as matrix factorization

Clustering finds an **optimal partitioning** of a data set[4]



For a clustering with k clusters, we have $\boldsymbol{\mu} \in \mathbb{R}^{D \times k}$ and $\mathbf{c} \in \{0,1\}^{k \times N}$.

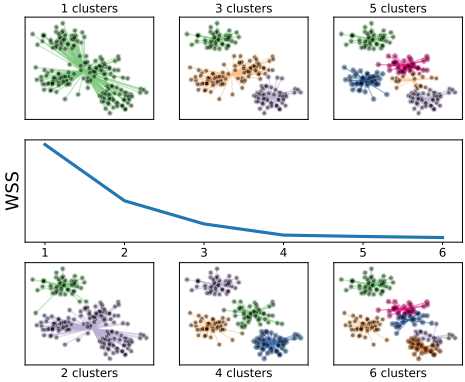
[4] In the previous example, we actually used this by approximating the original colors $\boldsymbol{x}_i$ with the corresponding $\boldsymbol{\mu}_{r_i}$.

How to choose k

- Number of clusters k is critical hyper-parameter
- In supervised settings we use model selection (grid search) to optimize hyper-parameters for accuracy on test data
- How can we optimize the number of clusters?

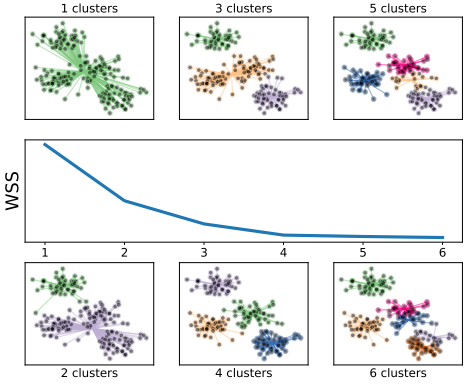
How to choose k

- Number of clusters k is critical hyper-parameter
- In supervised settings we use model selection (grid search) to optimize hyper-parameters for accuracy on test data
- How can we optimize the number of clusters?



How to choose k

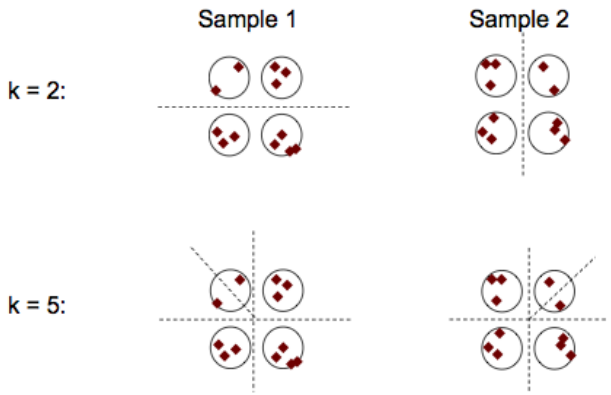
- Number of clusters k is critical hyper-parameter
- In supervised settings we use model selection (grid search) to optimize hyper-parameters for accuracy on test data
- How can we optimize the number of clusters?



→ One approach: find "elbow"; lowest k after which no real change

Clustering Instability

Number of Clusters is a critical parameter



Clusterings are unstable (i.e., converge to different results) if number of clusters is too small or too large

Summary I

- Matrix Factorization Methods
  - PCA and NMF belong to this class
  - Linearly approximate original data X with WH
- Principal Component Analysis
  - is a popular dimensionality reduction tool
  - aligns directions of maximal variance with standard basis
  - finds orthogonal directions
  - finds optimal matrix factorization
    (smallest Euclidean distance to subspace that the data is projected on)

Summary II

- Non-negative Matrix Factorization
  - works for non-negative data (count data, probabilistic data)
  - does not find orthogonal directions/uncorrelated factors
  - NMF encoding typically more sparse than PCA encoding
- Gradient Descent
  - useful for non-convex optimization
  - work-horse of Machine Learning
- K-Means Clustering
  - finds an optimal partitioning of a data set
  - K-Means requires
    - Good initialization
    - Knowledge of optimal k

References

C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 2007.

N. Gillis. The why and how of nonnegative matrix factorization. *Regularization, Optimization, Kernels, and Support Vector Machines*, 12(257):257–291, 2014.

D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–91, 1999. doi: 10.1038/44565.

D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *In NIPS*, pages 556–562. MIT Press, 2000.

K. Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

B. Schölkopf, A. J. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(6):1299–1319, 1998.

H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. 08 2017.