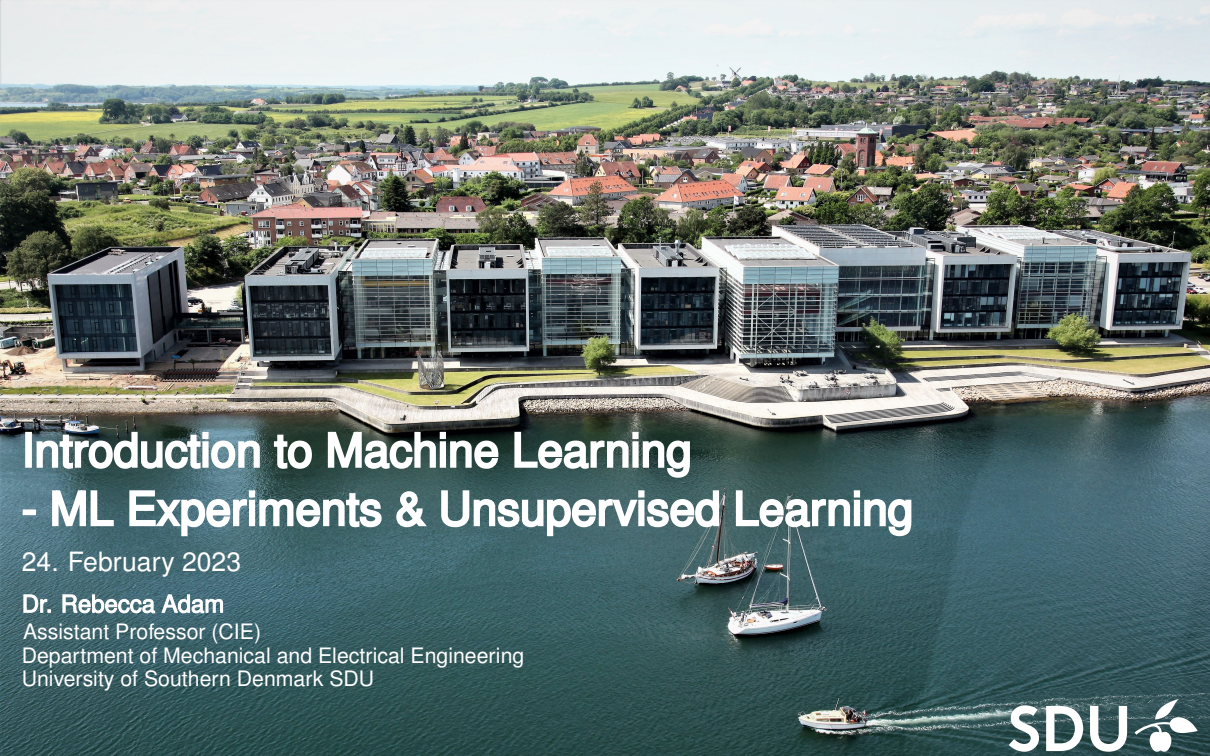
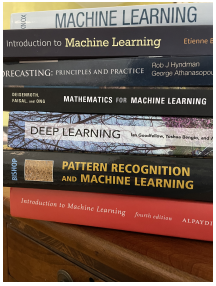


Introduction to Machine Learning
- ML Experiments & Unsupervised Learning

24. February 2023

Dr. Rebecca Adam
Assistant Professor (CIE)
Department of Mechanical and Electrical Engineering
University of Southern Denmark SDU

SDU



Literature Used in This Course

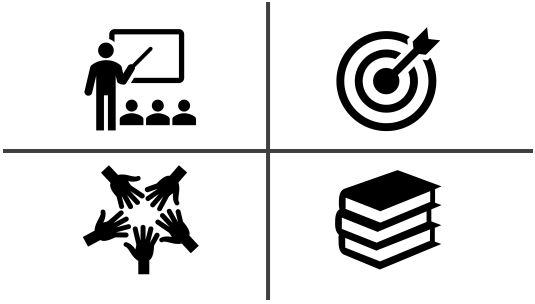
1. Knox, S. W., *"Machine Learning: A concise Introduction"*, Wiley, 2018
2. Bishop, C. M., *"Pattern Recognition and Machine Learning"*, Springer Berlin/Heidelberg, 2006
3. Goodfellow, I. and Bengio, Y. and Corville, A., *"Deep Learning"*, Adaptive computation and machine learning MIT Press, (2016)
4. Bernard, E., *"Introduction to Machine Learning"*, Wolfram Media, 2021
5. Alpaydin, E., *"Introduction to Machine Learning"*, 4th ed., MIT Press, 2020
6. Deisenroth, M. P. and Faisal, A. A., Ong, C. S. *"Mathematics for Machine Learning"*, 1st ed., Cambridge Press, 2020

Useful Internet Sources

1. **Basics:** https://numpy.org/doc/stable/ & https://pandas.pydata.org/docs/index.html
2. **Plotting:** https://matplotlib.org, https://seaborn.pydata.org, https://plotly.com/python/
3. **ML:** https://scikit-learn.org/stable/ & https://www.tensorflow.org/guide
4. **Useful Blog:** https://machinelearningmastery.com
5. **Data:** https://archive.ics.uci.edu/ml/index.php

Learning Goals Today

- Performance Metrics for Supervise Learning
- Designing Machine Learning Experiments
- Theoretical fundamentals for Unsupervised Learning (Eigendecomposition and Singulare Value Decomposition)
- Unsupervised Learning: Clustering
- Unsupervised Learning: Principle Component Analysis



Confusion Matrix

		Prediction	
		1	0
Actual	1	TP	FN
	0	FP	TN

- TP = True Positive count
- FN = False Negative count
- FP = False Positive count
- TN = True Negative count

Confusion Matrix

		Prediction	
		1	0
Actual	1	TP	FN
	0	FP	TN

- TP = True Positive count
- FN = False Negative count
- FP = False Positive count
- TN = True Negative count

Exercise (5min) - Confusion Matrix

Calculate the confusion matrix entries for

$$\hat{\mathbf{y}} = [0, 0, 1, 1, 0, 1, 1, 1, 0, 0]$$
$$\mathbf{y} = [1, 0, 1, 1, 0, 1, 0, 0, 0, 0]$$

		Prediction	
		1	0
Actual	1	TP	FN
	0	FP	TN

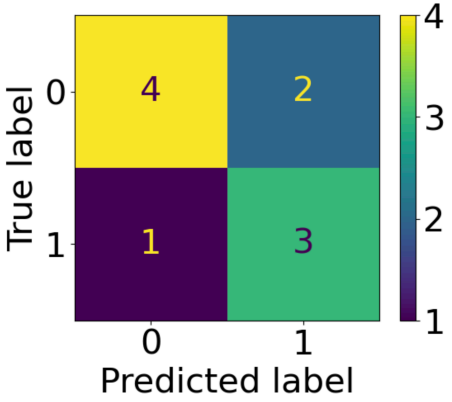
- TP = True Positive count
- FN = False Negative count
- FP = False Positive count
- TN = True Negative count

Exercise (5min) - Confusion Matrix

Calculate the confusion matrix entries for

$$\hat{\mathbf{y}} = [0, 0, 1, 1, 0, 1, 1, 1, 0, 0]$$
$$\mathbf{y} = [1, 0, 1, 1, 0, 1, 0, 0, 0, 0]$$

Confusion Matrix



In sklearn use labels to specify the ”positive” class.

```
cm=confusion_matrix(y_true, y_pred, labels=[1,0])
```

Accuracy:

Fraction of correctly predicted data of total data

$$A(\mathbf{y}, \hat{\mathbf{y}}) = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}.$$

Accuracy:

Fraction of correctly predicted data of total data

$$A(\mathbf{y}, \hat{\mathbf{y}}) = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}.$$

Precision:

Fraction of correctly predicted positive samples out of positive predictions.

$$P(\mathbf{y}, \hat{\mathbf{y}}) = \frac{TP}{TP + FP}.$$

Recall or Sensitivity:

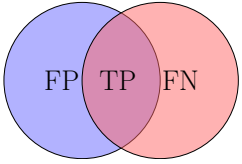
Fraction of correctly predicted positive samples out of the total positive sample number.

$$R(\mathbf{y}, \hat{\mathbf{y}}) = \frac{TP}{TP + FN}.$$

Classified as Spam Email

Precision

$\mathrm{r}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$



Email is Spam Ground Truth

Recall

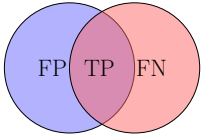
$\mathrm{r}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$

Exercise (5min) - Precision and Recall: Draw and interpret Precision $= 1$ and Recall $= 1$.

SDU

Classified as Spam Email

Precision
$\mathrm{r}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$

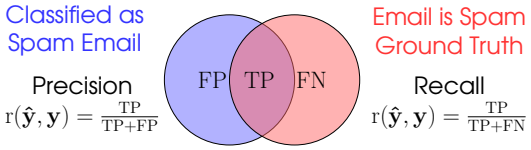


Email is Spam Ground Truth

Recall
$\mathrm{r}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$

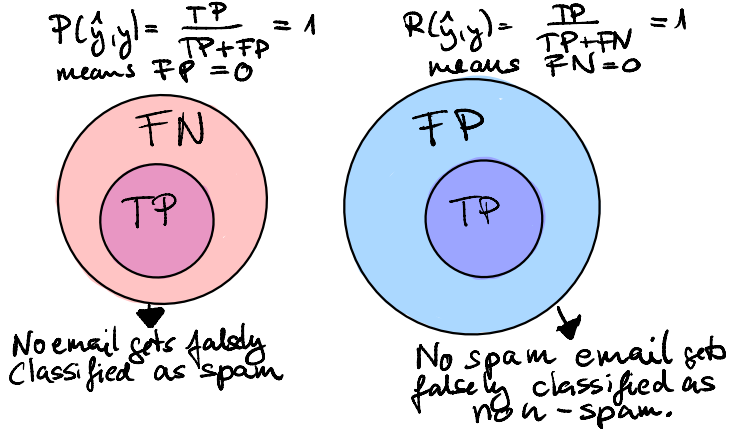
Exercise (5min) - Precision & Recall:

Draw and interpret precision $= 1$ and recall $= 1$ for spam classifier.



Exercise (5min) - Precision & Recall:

Draw and interpret precision $= 1$ and recall $= 1$ for spam classifier.



Classification Metrics

- **Precision:** $P = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$
- **Recall, Sensitivity:** $R = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$
- **False Positive Rate, Fall-Out:** $\frac{\mathrm{FP}}{\mathrm{FP+TN}}$
- **Specificity, True Negative Rate**
  $1 - \mathrm{fp\text{-}rate} = \frac{\mathrm{TN}}{\mathrm{TN+FP}}$

Classification Metrics

- **Precision:** $P = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$
- **Recall, Sensitivity:** $R = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$
- **False Positive Rate, Fall-Out:** $\frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}}$
- **Specificity, True Negative Rate**
  $1 - \mathrm{fp\text{-}rate} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}$

F1-Score

The F1 score can be interpreted as the harmonic mean between precision and recall

$$\mathrm{F1} = 2\frac{P(\mathbf{y}, \hat{\mathbf{y}})R(\mathbf{y}, \hat{\mathbf{y}})}{P(\mathbf{y}, \hat{\mathbf{y}}) + R(\mathbf{y}, \hat{\mathbf{y}})}$$

Classification Metrics

- **Precision:** $P = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$
- **Recall, Sensitivity:** $R = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$
- **False Positive Rate, Fall-Out:** $\frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}}$
- **Specificity, True Negative Rate**
  $1 - \mathrm{fp\text{-}rate} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}$

F1-Score

The F1 score can be interpreted as the harmonic mean between precision and recall

$$\mathrm{F1} = 2\frac{P(\mathbf{y}, \hat{\mathbf{y}})R(\mathbf{y}, \hat{\mathbf{y}})}{P(\mathbf{y}, \hat{\mathbf{y}}) + R(\mathbf{y}, \hat{\mathbf{y}})}$$

Exercise (5min) - Specificity

Assume 100 of 1000 citizens are infected with Corona. Given the two Specificities $S_A = 96.0\%$ and $S_B = 99.8\%$. Determine many people not having Corona will be tested as positive (FP).

Classification Metrics

- **Precision:** $P = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$
- **Recall, Sensitivity:** $R = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$
- **False Positive Rate, Fall-Out:** $\frac{\mathrm{FP}}{\mathrm{FP+TN}}$
- **Specificity, True Negative Rate**
  $1 - \text{fp-rate} = \frac{\mathrm{TN}}{\mathrm{TN+FP}}$

F1-Score

The F1 score can be interpreted as the harmonic mean between precision and recall

$$\mathrm{F1} = 2\frac{P(\mathbf{y}, \hat{\mathbf{y}})R(\mathbf{y}, \hat{\mathbf{y}})}{P(\mathbf{y}, \hat{\mathbf{y}}) + R(\mathbf{y}, \hat{\mathbf{y}})}$$

Exercise (5min) - Specificity

Assume 100 of 1000 citizens are infected with Corona. Given the two Specificities $S_A = 96.0\%$ and $S_B = 99.8\%$. Determine many people not having Corona will be tested as positive (FP).

CASE 1

$N = TN + FP = 900$ (1)

$0{,}04 = \frac{FP}{FP+TN}$ (2)

From (1) we get $TN = 900 - FP$ (3)

Substituting (3) in (2):

$0{,}04 = \frac{FP}{900}$

$\Leftrightarrow FP = \frac{4 \cdot 900}{100} = 36$

CASE 2

$0{,}002 = \frac{FP}{900}$

$FP = 0{,}2 \cdot 9 = 1{,}8$

SDU

Classification Metrics

- **Precision:** $P = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$
- **Recall, Sensitivity:** $R = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$
- **False Positive Rate, Fall-Out:** $\frac{\mathrm{FP}}{\mathrm{FP+TN}}$
- **Specificity, True Negative Rate**
  $1 - \text{fp-rate} = \frac{\mathrm{TN}}{\mathrm{TN+FP}}$

F1-Score

The F1 score can be interpreted as the harmonic mean between precision and recall

$$\mathrm{F1} = 2\frac{P(\mathbf{y}, \hat{\mathbf{y}})R(\mathbf{y}, \hat{\mathbf{y}})}{P(\mathbf{y}, \hat{\mathbf{y}}) + R(\mathbf{y}, \hat{\mathbf{y}})}$$

Exercise (5min) - Specificity

Assume 100 of 1000 citizens are infected with Corona. Given the two Specificities $S_A = 96.0\%$ and $S_B = 99.8\%$. Determine many people not having Corona will be tested as positive (FP).

CASE 1

$N = TN + FP = 900$ (1)

$0{,}04 = \frac{FP}{FP+TN}$ (2)

From (1) we get $TN = 900 - FP$ (3)

Substituting (3) in (2):

$0{,}04 = \frac{FP}{900}$

$\Leftrightarrow FP = \frac{4 \cdot 900}{100} = 36$

CASE 2

$0{,}002 = \frac{FP}{900}$

$FP = 0{,}2 \cdot 9 = 1{,}8$

Note that

slight changes in the specificity and sensitivity can have a large impact in real life.

SDU

Let y_n denote the ground truth and let $\hat{y}_n \ \ \forall n \in \{1, \ldots, N_{\text{test}}\}$ denote the predicted values.

The Root Mean Squared Error (MSE)

$$\text{RMSE}(\mathbf{y}, \hat{\mathbf{y}}) = \sqrt{\frac{1}{N_{\text{test}}} \sum_{n=1}^{N_{\text{test}}} (y_n - \hat{y}_n)^2}.$$

Let y_n denote the ground truth and let $\hat{y}_n \quad \forall n \in \{1, \ldots, N_{\text{test}}\}$ denote the predicted values.

The Root Mean Squared Error (MSE)

$$\text{RMSE}(\mathbf{y}, \hat{\mathbf{y}}) = \sqrt{\frac{1}{N_{\text{test}}} \sum_{n=1}^{N_{\text{test}}} (y_n - \hat{y}_n)^2}.$$

Mean Absolute Error

$$\text{MAE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N_{\text{test}}} \sum_{n=1}^{N_{\text{test}}} |y_n - \hat{y}_n|.$$

Let y_n denote the ground truth and let $\hat{y}_n \quad \forall n \in \{1, \ldots, N_{\text{test}}\}$ denote the predicted values.

The Root Mean Squared Error (MSE)

$$\text{RMSE}(\mathbf{y}, \hat{\mathbf{y}}) = \sqrt{\frac{1}{N_{\text{test}}} \sum_{n=1}^{N_{\text{test}}} (y_n - \hat{y}_n)^2}.$$

Mean Absolute Error

$$\text{MAE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N_{\text{test}}} \sum_{n=1}^{N_{\text{test}}} |y_n - \hat{y}_n|.$$

Mean Absolute Percentage Error

$$\text{MAPE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N_{\text{test}}} \sum_{n=1}^{N_{\text{test}}} \frac{|y_n - \hat{y}_n|}{\max(\varepsilon, |y_i|)} \quad \rightarrow \text{sensitive to relative errors.}$$

Let y_n denote the ground truth and let $\hat{y}_n \quad \forall n \in \{1, \ldots, N_{\text{test}}\}$ denote the predicted values.

The Root Mean Squared Error (MSE)

$$\text{RMSE}(\mathbf{y}, \hat{\mathbf{y}}) = \sqrt{\frac{1}{N_{\text{test}}} \sum_{n=1}^{N_{\text{test}}} (y_n - \hat{y}_n)^2}.$$

Mean Absolute Error

$$\text{MAE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N_{\text{test}}} \sum_{n=1}^{N_{\text{test}}} |y_n - \hat{y}_n|.$$

Mean Absolute Percentage Error

$$\text{MAPE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N_{\text{test}}} \sum_{n=1}^{N_{\text{test}}} \frac{|y_n - \hat{y}_n|}{\max(\varepsilon, |y_i|)} \quad \rightarrow \text{sensitive to relative errors.}$$

Median Absolute Error

$$\text{MedAE}(\mathbf{y}, \hat{\mathbf{y}}) = \text{median}(|y_1 - \hat{y}|, \ldots, |y_{N_{\text{test}}} - \hat{y}_{N_{\text{test}}}|) \quad \rightarrow \text{robust wrt. outliers.}$$

Let $\mathbf{x}$ and $\mathbf{y}$ be two random variables with the observations $\mathbf{x}_n, \mathbf{y}_n \in \mathbb{R}^{M\times 1}$ with $n \in \{1\dots, N\}$

- The empirical **mean** $\hat{\mathrm{E}}[\mathbf{x}] = \overline{\mathbf{x}} \approx \mathrm{E}[\mathbf{x}]$ determines the estimated mean and is the observation average:

$$\mathrm{E}[\mathbf{x}] \approx \hat{\mathrm{E}}[\mathbf{x}] = \overline{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n.$$

- The **covariance matrix** $\mathrm{Cov}[\mathbf{x}, \mathbf{y}]$ reads:

$$\begin{aligned} M = 1: \ \mathrm{Cov}[\mathbf{x}, \mathbf{y}] &= \mathrm{E}\left[(\mathbf{x} - \mathrm{E}[\mathbf{x}])(\mathbf{y} - \mathrm{E}[\mathbf{y}])^T\right] \\ &= \mathrm{E}[\mathbf{x}\mathbf{y}^{\mathbf{T}}] - \mathrm{E}[\mathbf{x}]\mathrm{E}[\mathbf{y}^T] \\ M > 1: \ \mathbf{C}_{\mathbf{x},\mathbf{y}} &\approx \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \overline{\mathbf{x}})(\mathbf{y}_n - \overline{\mathbf{y}})^T. \end{aligned}$$

- The **variance** $\mathrm{Var}[\mathbf{x} = \mathrm{Cov}[\mathbf{x}, \mathbf{x}^T]$ for $M = 1$ reads:

$$\begin{aligned} \mathrm{Var}[\mathbf{x}] &= \mathrm{Cov}[\mathbf{x}, \mathbf{x}] = \mathrm{E}[\mathbf{x}\mathbf{x}^T] - \mathrm{E}[\mathbf{x}]\mathrm{E}[\mathbf{x}^T] \\ M > 1: \mathbf{C}_{\mathbf{x},\mathbf{x}} &\approx \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \overline{\mathbf{x}})(\mathbf{x}_n - \overline{\mathbf{x}})^T \end{aligned}$$

Let $\mathbf{x}$ and $\mathbf{y}$ be two random variables with the observations $\mathbf{x}_n, \mathbf{y}_n \in \mathbb{R}^{M\times 1}$ with $n \in \{1\dots, N\}$

- The empirical **mean** $\hat{\mathrm{E}}[\mathbf{x}] = \overline{\mathbf{x}} \approx \mathrm{E}[\mathbf{x}]$ determines the estimated mean and is the observation average:

$$\mathrm{E}[\mathbf{x}] \approx \hat{\mathrm{E}}[\mathbf{x}] = \overline{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n.$$

- The **covariance matrix** $\mathrm{Cov}[\mathbf{x},\mathbf{y}]$ reads:

$$\begin{aligned} M=1:\ \mathrm{Cov}[\mathbf{x},\mathbf{y}] &= \mathrm{E}\left[(\mathbf{x}-\mathrm{E}[\mathbf{x}])(\mathbf{y}-\mathrm{E}[\mathbf{y}])^T\right] \\ &= \mathrm{E}[\mathbf{x}\mathbf{y}^{\mathbf{T}}] - \mathrm{E}[\mathbf{x}]\mathrm{E}[\mathbf{y}^T] \\ M>1:\ \mathbf{C}_{\mathbf{x},\mathbf{y}} &\approx \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n-\overline{\mathbf{x}})(\mathbf{y}_n-\overline{\mathbf{y}})^T. \end{aligned}$$

- The **variance** $\mathrm{Var}[\mathbf{x} = \mathrm{Cov}[\mathbf{x},\mathbf{x}^T]$ for $M=1$ reads:

$$\begin{aligned} \mathrm{Var}[\mathbf{x}] &= \mathrm{Cov}[\mathbf{x},\mathbf{x}] = \mathrm{E}[\mathbf{x}\mathbf{x}^T] - \mathrm{E}[\mathbf{x}]\mathrm{E}[\mathbf{x}^T] \\ M>1: \mathbf{C}_{\mathbf{x},\mathbf{x}} &\approx \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n-\overline{\mathbf{x}})(\mathbf{x}_n-\overline{\mathbf{x}})^T \end{aligned}$$

Let $\mathbf{x}$ and $\mathbf{y}$ be two random variables with the observations $\mathbf{x}_n, \mathbf{y}_n \in \mathbb{R}^{M\times 1}$ with $n \in \{1\ldots, N\}$

- The empirical **mean** $\hat{\mathrm{E}}[\mathbf{x}] = \overline{\mathbf{x}} \approx \mathrm{E}[\mathbf{x}]$ determines the estimated mean and is the observation average:

$$\mathrm{E}[\mathbf{x}] \approx \hat{\mathrm{E}}[\mathbf{x}] = \overline{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n.$$

- The **covariance matrix** $\mathrm{Cov}[\mathbf{x}, \mathbf{y}]$ reads:

$$\begin{aligned} M = 1: \ \mathrm{Cov}[\mathbf{x}, \mathbf{y}] &= \mathrm{E}\left[(\mathbf{x} - \mathrm{E}[\mathbf{x}])(\mathbf{y} - \mathrm{E}[\mathbf{y}])^T\right] \\ &= \mathrm{E}[\mathbf{x}\mathbf{y}^{\mathbf{T}}] - \mathrm{E}[\mathbf{x}]\mathrm{E}[\mathbf{y}^T] \\ M > 1: \ \mathbf{C}_{\mathbf{x},\mathbf{y}} &\approx \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \overline{\mathbf{x}})(\mathbf{y}_n - \overline{\mathbf{y}})^T. \end{aligned}$$

- The **variance** $\mathrm{Var}[\mathbf{x} = \mathrm{Cov}[\mathbf{x}, \mathbf{x}^T]$ for $M = 1$ reads:

$$\mathrm{Var}[\mathbf{x}] = \mathrm{Cov}[\mathbf{x}, \mathbf{x}] = \mathrm{E}[\mathbf{x}\mathbf{x}^T] - \mathrm{E}[\mathbf{x}]\mathrm{E}[\mathbf{x}^T]$$

$$M > 1 : \mathbf{C}_{\mathbf{x},\mathbf{x}} \approx \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \overline{\mathbf{x}})(\mathbf{x}_n - \overline{\mathbf{x}})^T$$

Let $\mathbf{x}$ and $\mathbf{y}$ be two random variables with the observations $\mathbf{x}_n, \mathbf{y}_n \in \mathbb{R}^{M\times 1}$ with $n \in \{1\ldots, N\}$

- The empirical **mean** $\hat{\mathrm{E}}[\mathbf{x}] = \overline{\mathbf{x}} \approx \mathrm{E}[\mathbf{x}]$ determines the estimated mean and is the observation average:

$$\mathrm{E}[\mathbf{x}] \approx \hat{\mathrm{E}}[\mathbf{x}] = \overline{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n.$$

- The **covariance matrix** $\mathrm{Cov}[\mathbf{x}, \mathbf{y}]$ reads:

$$\begin{aligned} M = 1: \ \mathrm{Cov}[\mathbf{x}, \mathbf{y}] &= \mathrm{E}\left[(\mathbf{x} - \mathrm{E}[\mathbf{x}])(\mathbf{y} - \mathrm{E}[\mathbf{y}])^T\right] \\ &= \mathrm{E}[\mathbf{x}\mathbf{y}^{\mathbf{T}}] - \mathrm{E}[\mathbf{x}]\mathrm{E}[\mathbf{y}^T] \\ M > 1: \ \mathbf{C}_{\mathbf{X},\mathbf{Y}} &\approx \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \overline{\mathbf{x}})(\mathbf{y}_n - \overline{\mathbf{y}})^T. \end{aligned}$$

- The **variance** $\mathrm{Var}[\mathbf{x} = \mathrm{Cov}[\mathbf{x}, \mathbf{x}^T]$ for $M = 1$ reads:

$$\mathrm{Var}[\mathbf{x}] = \mathrm{Cov}[\mathbf{x}, \mathbf{x}] = \mathrm{E}[\mathbf{x}\mathbf{x}^T] - \mathrm{E}[\mathbf{x}]\mathrm{E}[\mathbf{x}^T]$$

$$M > 1 : \mathbf{C}_{\mathbf{X},\mathbf{X}} \approx \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \overline{\mathbf{x}})(\mathbf{x}_n - \overline{\mathbf{x}})^T$$

Let $\mathbf{x}$ and $\mathbf{y}$ be two random variables with the observations $\mathbf{x}_n, \mathbf{y}_n \in \mathbb{R}^{M\times 1}$ with $n \in \{1\dots,N\}$

- The empirical **mean** $\hat{\mathrm{E}}[\mathbf{x}] = \overline{\mathbf{x}} \approx \mathrm{E}[\mathbf{x}]$ determines the estimated mean and is the observation average:

$$\mathrm{E}[\mathbf{x}] \approx \hat{\mathrm{E}}[\mathbf{x}] = \overline{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n.$$

- The **covariance matrix** $\mathrm{Cov}[\mathbf{x},\mathbf{y}]$ reads:

$$M=1:\ \mathrm{Cov}[\mathbf{x},\mathbf{y}] = \mathrm{E}\left[(\mathbf{x}-\mathrm{E}[\mathbf{x}])(\mathbf{y}-\mathrm{E}[\mathbf{y}])^T\right] = \mathrm{E}[\mathbf{x}\mathbf{y}^{\mathbf{T}}] - \mathrm{E}[\mathbf{x}]\mathrm{E}[\mathbf{y}^T]$$

$$M>1:\ \mathbf{C}_{\mathbf{X},\mathbf{Y}} \approx \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n-\overline{\mathbf{x}})(\mathbf{y}_n-\overline{\mathbf{y}})^T.$$

- The **variance** $\mathrm{Var}[\mathbf{x} = \mathrm{Cov}[\mathbf{x},\mathbf{x}^T]$ for $M=1$ reads:

$$\mathrm{Var}[\mathbf{x}] = \mathrm{Cov}[\mathbf{x},\mathbf{x}] = \mathrm{E}[\mathbf{x}\mathbf{x}^T] - \mathrm{E}[\mathbf{x}]\mathrm{E}[\mathbf{x}^T]$$

$$M>1: \mathbf{C}_{\mathbf{X},\mathbf{X}} \approx \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n-\overline{\mathbf{x}})(\mathbf{x}_n-\overline{\mathbf{x}})^T$$

Correlation r for M=1 Reads:

$$r[x,y] = \frac{\mathrm{Cov}[x,y]}{\sqrt{\mathrm{Var}[x]\mathrm{Var}[y]}} \in [-1,1]\,.$$

$$= \frac{\frac{1}{N}\sum_{n=1}^{N}(x_n-\overline{x})(y_n-\overline{y})}{\sqrt{\sigma_x^2\sigma_y^2}} \in [-1,1]\,.$$

Let $\mathbf{x}$ and $\mathbf{y}$ be two random variables with the observations $\mathbf{x}_n, \mathbf{y}_n \in \mathbb{R}^{M\times 1}$ with $n \in \{1\dots,N\}$

- The empirical **mean** $\hat{\mathrm{E}}[\mathbf{x}] = \overline{\mathbf{x}} \approx \mathrm{E}[\mathbf{x}]$ determines the estimated mean and is the observation average:

$$\mathrm{E}[\mathbf{x}] \approx \hat{\mathrm{E}}[\mathbf{x}] = \overline{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n.$$

- The **covariance matrix** $\mathrm{Cov}[\mathbf{x},\mathbf{y}]$ reads:

$$M=1:\ \mathrm{Cov}[\mathbf{x},\mathbf{y}] = \mathrm{E}\left[(\mathbf{x}-\mathrm{E}[\mathbf{x}])(\mathbf{y}-\mathrm{E}[\mathbf{y}])^T\right] = \mathrm{E}[\mathbf{x}\mathbf{y}^{\mathbf{T}}] - \mathrm{E}[\mathbf{x}]\mathrm{E}[\mathbf{y}^T]$$

$$M>1:\ \mathbf{C}_{\mathbf{x},\mathbf{y}} \approx \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n-\overline{\mathbf{x}})(\mathbf{y}_n-\overline{\mathbf{y}})^T.$$

- The **variance** $\mathrm{Var}[\mathbf{x} = \mathrm{Cov}[\mathbf{x},\mathbf{x}^T]$ for $M=1$ reads:

$$\mathrm{Var}[\mathbf{x}] = \mathrm{Cov}[\mathbf{x},\mathbf{x}] = \mathrm{E}[\mathbf{x}\mathbf{x}^T] - \mathrm{E}[\mathbf{x}]\mathrm{E}[\mathbf{x}^T]$$

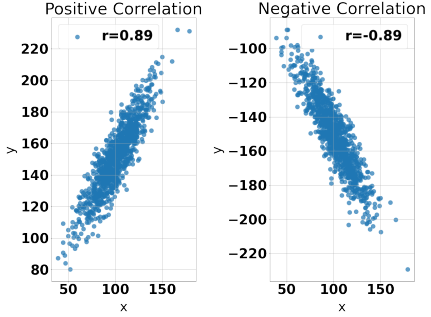
$$M>1: \mathbf{C}_{\mathbf{x},\mathbf{x}} \approx \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n-\overline{\mathbf{x}})(\mathbf{x}_n-\overline{\mathbf{x}})^T$$

Correlation r for M=1 Reads:

$$r[x,y] = \frac{\mathrm{Cov}[x,y]}{\sqrt{\mathrm{Var}[x]\mathrm{Var}[y]}} \in [-1,1]\,.$$

$$= \frac{\frac{1}{N}\sum_{n=1}^{N}(x_n-\overline{x})(y_n-\overline{y})}{\sqrt{\sigma_x^2\sigma_y^2}} \in [-1,1]\,.$$

Visually → more intuitive



Note $\rightarrow$ Correlation is not Causation!

- $\mathbf{x} \in \mathbb{R}^{N \times 1}$ can be useful to predict $\hat{y} \in \mathbb{R}^{N \times 1}$,
- although it does not cause $\hat{\mathbf{y}}$.

Note → Correlation is not Causation!

- $\mathbf{x} \in \mathbb{R}^{N \times 1}$ can be useful to predict $\hat{y} \in \mathbb{R}^{N \times 1}$,
- although it does not cause $\hat{\mathbf{y}}$.

Example: Ice-cream Sales & Beach-resort Drownings

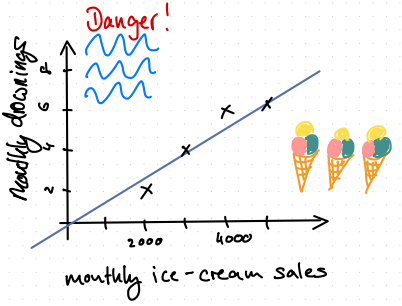
- The number of sold ice-creams per month can be used to model the monthly drownings.
- The model can perform well, not because ice-creams causes drownings, but because Ice-cream sales depend like drownings/swimming on the temperature.

Note → Correlation is not Causation!

- $\mathbf{x} \in \mathbb{R}^{N \times 1}$ can be useful to predict $\hat{y} \in \mathbb{R}^{N \times 1}$,
- although it does not cause $\hat{\mathbf{y}}$.

Example: Ice-cream Sales & Beach-resort Drownings

- The number of sold ice-creams per month can be used to model the monthly drownings.
- The model can perform well, not because ice-creams causes drownings, but because Ice-cream sales depend like drownings/swimming on the temperature.

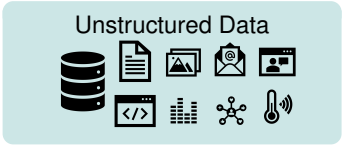


Important

- Correlations are useful for predictions.
- Knowing the causal mechanism (like temperature) can provide better predictors and models.

Feature Extraction:

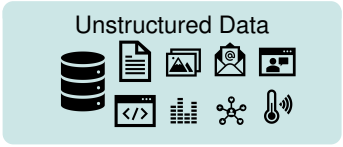
- Selecting structured data from unstructured datasets.
- Features created from existing measurements (Most common: statistical features like mean, max/min, and standard deviation).



Feature 1	...	Feature M

Feature Extraction:

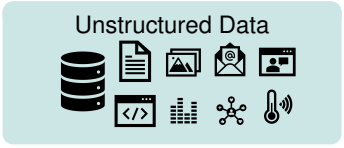
- Selecting structured data from unstructured datasets.
- Features created from existing measurements (Most common: statistical features like mean, max/min, and standard deviation).



Feature 1	...	Feature M

Feature Extraction:

- Selecting structured data from unstructured datasets.
- Features created from existing measurements (Most common: statistical features like mean, max/min, and standard deviation).



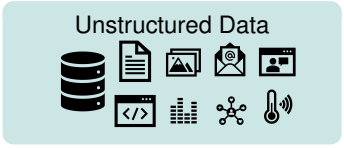
Feature 1	...	Feature M

Why Do We Need Feature Extraction?

- Learning from data without extracting features may require a high number of variables.
- A high number of variables
  - means a high amount of memory and computation power.
  - may cause a phenomenon called overfitting: model has so many degrees of freedom that it will perfectly match the training data, but perform poorly on the test data.

Feature Extraction:

- Selecting structured data from unstructured datasets.
- Features created from existing measurements (Most common: statistical features like mean, max/min, and standard deviation).



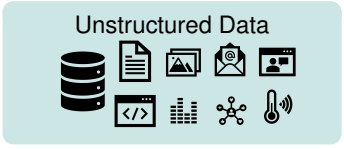
Feature 1	...	Feature M

Why Do We Need Feature Extraction?

- Learning from data without extracting features may require a high number of variables.
- A high number of variables
  - means a high amount of memory and computation power.
  - may cause a phenomenon called overfitting: model has so many degrees of freedom that it will perfectly match the training data, but perform poorly on the test data.

Feature Extraction:

- Selecting structured data from unstructured datasets.
- Features created from existing measurements (Most common: statistical features like mean, max/min, and standard deviation).



Feature 1	...	Feature M

Why Do We Need Feature Extraction?

- Learning from data without extracting features may require a high number of variables.
- A high number of variables
  - means a high amount of memory and computation power.
  - may cause a phenomenon called overfitting: model has so many degrees of freedom that it will perfectly match the training data, but perform poorly on the test data.

Recap: Feature Selection:

- Keep features that improve model in training process.
- Discard Features that do not considerably improve model in the training process.

Eg. 100 Features

Feature 1	...	Feature 100

Feature Selection

Eg. 10 Features

Feature 1	...	Feature 10

Note...

- Besides feature extraction and feature selection the third option is feature reduction!
- We will discuss feature reduction later in more detail.

Normalization and Scaling

- Different features can have different numerical ranges.
- To assure that we train our algorithm with input scaled to the same range we can perform normalization.

Exercise (5-10min) - Normalization

Go to https://scikit-learn.org/stable/modules/preprocessing.html#preprocessing and get familiar with different normalization and scaling methods.





No Free Lunch Theorem

- The ”No Free Lunch” Theorem states that there is no such thing as the best learning algorithm.
- For any learning algorithm there is a dataset, where the algorithm performs poorly.





No Free Lunch Theorem

- The ”No Free Lunch” Theorem states that there is no such thing as the best learning algorithm.
- For any learning algorithm there is a dataset, where the algorithm performs poorly.





No Free Lunch Theorem

- The ”No Free Lunch” Theorem states that there is no such thing as the best learning algorithm.
- For any learning algorithm there is a dataset, where the algorithm performs poorly.

- The application is represented in the data we have
- Any result is true for a special dataset and application.





No Free Lunch Theorem

- The ”No Free Lunch” Theorem states that there is no such thing as the best learning algorithm.
- For any learning algorithm there is a dataset, where the algorithm performs poorly.

- The application is represented in the data we have
- Any result is true for a special dataset and application.

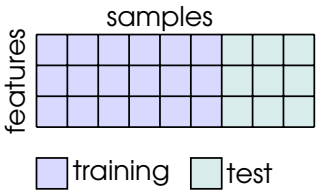




No Free Lunch Theorem

- The ”No Free Lunch” Theorem states that there is no such thing as the best learning algorithm.
- For any learning algorithm there is a dataset, where the algorithm performs poorly.

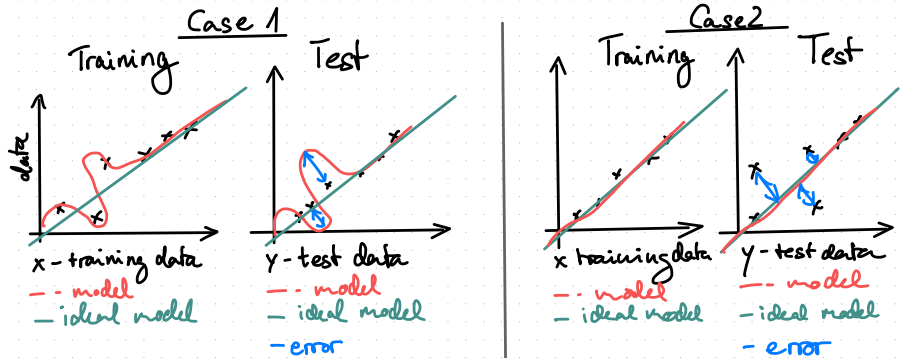
- The application is represented in the data we have
- Any result is true for a special dataset and application.



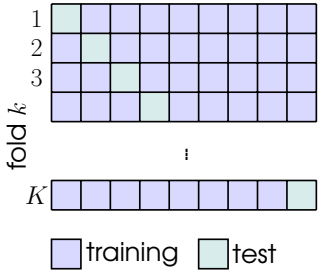
Small test sizes: Accuracy depends on chosen test set

Performance (or error) depends on the choice of the test set.

- Extraordinary training, representative test → large error
- Extraordinary test, representative training → large error

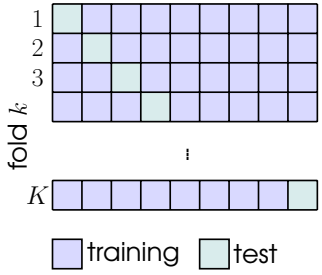


Solution → Cross Validation



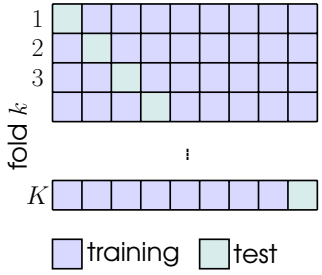
K-fold Cross Validation

- → Assumption: uncorrelated samples! (atypical for time series)
- → Partition the data set into k splits.
- → Randomly choose (k-1)-train sets and one test. Repeat training k times on leaving a test set out (Monte Carlo).
- → Calculate mean error of all training errors.



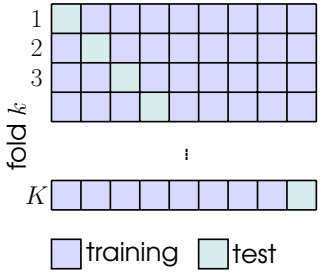
K-fold Cross Validation

- → Assumption: uncorrelated samples! (atypical for time series)
- → Partition the data set into k splits.
- → Randomly choose (k-1)-train sets and one test. Repeat training k times on leaving a test set out (Monte Carlo).
- → Calculate mean error of all training errors.



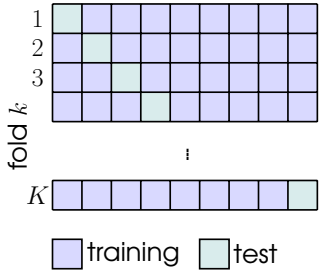
K-fold Cross Validation

- → Assumption: uncorrelated samples! (atypical for time series)
- → Partition the data set into k splits.
- → Randomly choose (k-1)-train sets and one test. Repeat training k times on leaving a test set out (Monte Carlo).
- → Calculate mean error of all training errors.



K-fold Cross Validation

- → Assumption: uncorrelated samples! (atypical for time series)
- → Partition the data set into k splits.
- → Randomly choose (k-1)-train sets and one test. Repeat training k times on leaving a test set out (Monte Carlo).
- → Calculate mean error of all training errors.

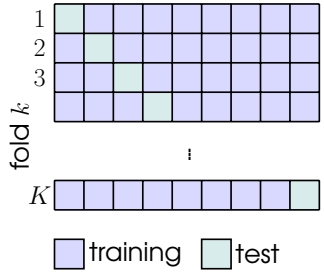


K-fold Cross Validation

- → Assumption: uncorrelated samples! (atypical for time series)
- → Partition the data set into k splits.
- → Randomly choose (k-1)-train sets and one test. Repeat training k times on leaving a test set out (Monte Carlo).
- → Calculate mean error of all training errors.

Cross validation purpose

- → Reliable, independent error estimation.
- → Model selection (Hyperparameter Tuning).
- → Generalizing the chosen model (preventing overfitting, underfitting).

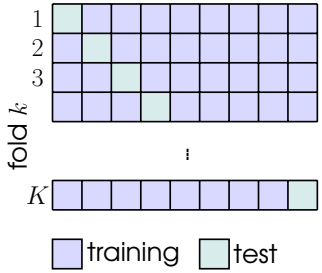


K-fold Cross Validation

- → Assumption: uncorrelated samples! (atypical for time series)
- → Partition the data set into k splits.
- → Randomly choose (k-1)-train sets and one test. Repeat training k times on leaving a test set out (Monte Carlo).
- → Calculate mean error of all training errors.

Cross validation purpose

- → Reliable, independent error estimation.
- → Model selection (Hyperparameter Tuning).
- → Generalizing the chosen model (preventing overfitting, underfitting).

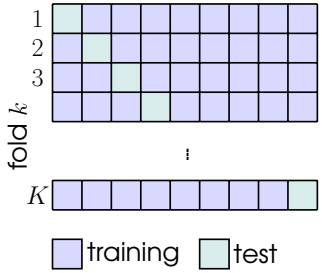


K-fold Cross Validation

- → Assumption: uncorrelated samples! (atypical for time series)
- → Partition the data set into k splits.
- → Randomly choose (k-1)-train sets and one test. Repeat training k times on leaving a test set out (Monte Carlo).
- → Calculate mean error of all training errors.

Cross validation purpose

- → Reliable, independent error estimation.
- → Model selection (Hyperparameter Tuning).
- → Generalizing the chosen model (preventing overfitting, underfitting).



K-fold Cross Validation

- → Assumption: uncorrelated samples! (atypical for time series)
- → Partition the data set into k splits.
- → Randomly choose (k-1)-train sets and one test. Repeat training k times on leaving a test set out (Monte Carlo).
- → Calculate mean error of all training errors.

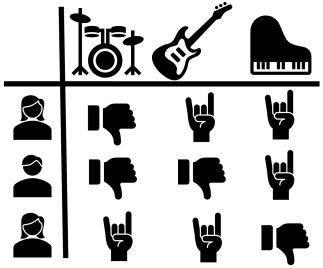
Cross validation purpose

- → Reliable, independent error estimation.
- → Model selection (Hyperparameter Tuning).
- → Generalizing the chosen model (preventing overfitting, underfitting).

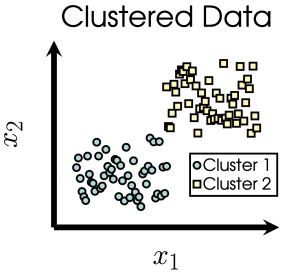
Practitioners View:

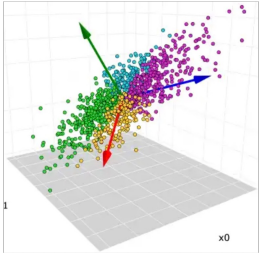
Partition the data block into three parts → train set, validation set, test set.
For us here → train test split is okay.

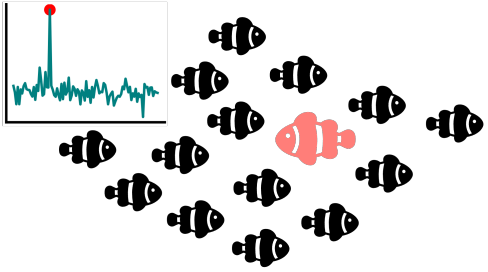
Unsupervised Learning Methods

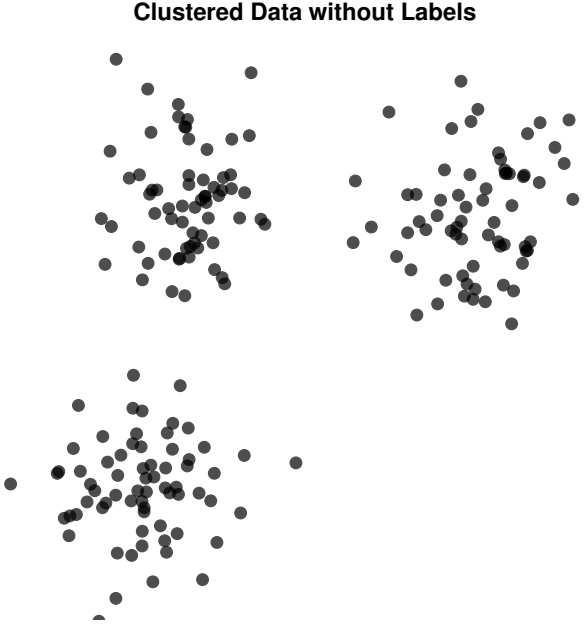


Clustering
Dimensionality Reduction
Recommender Systems
Anomaly Detection

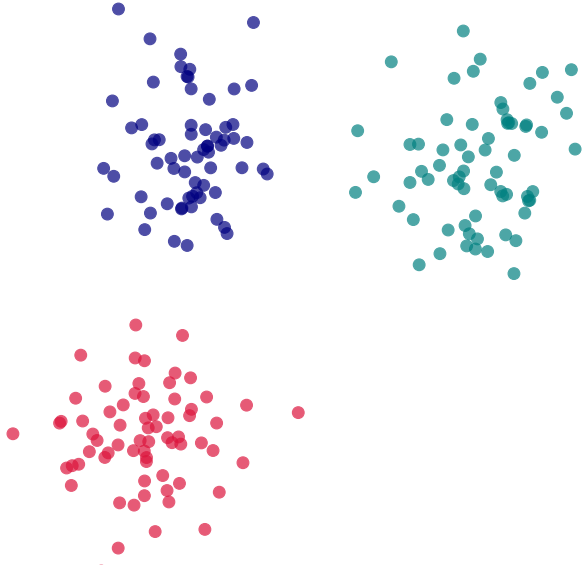


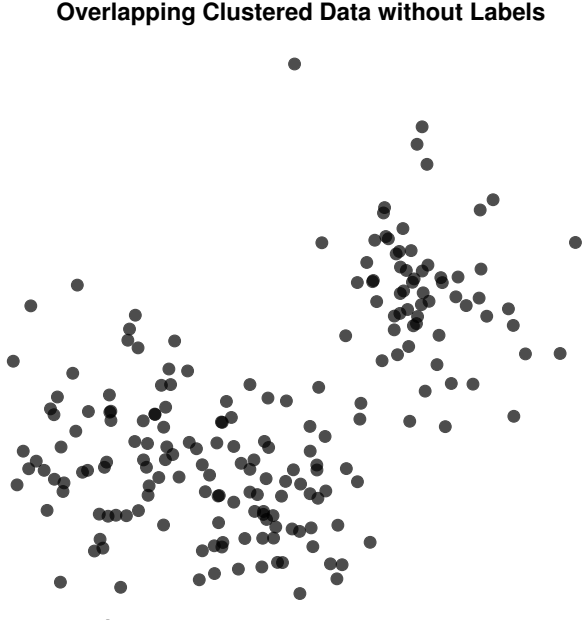




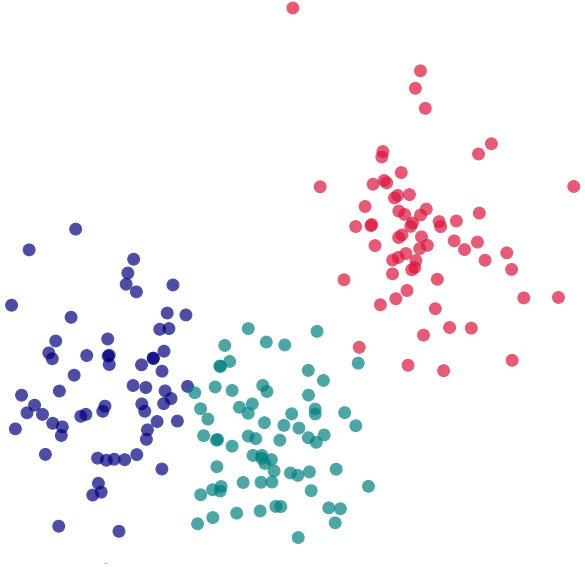

Clustered Data without Labels

Intuitive Visual Labeling

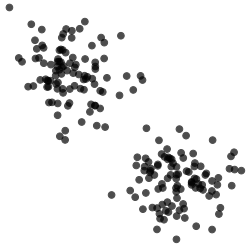


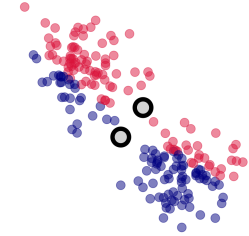

Overlapping Clustered Data without Labels

Intuitive Labeling Difficult



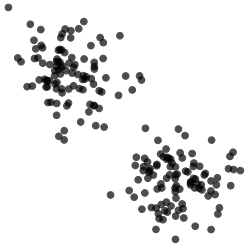
SDU

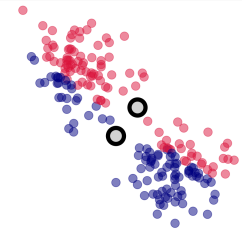


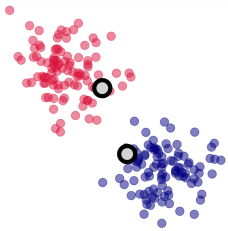


kMeans

- Assign each sample to closes centroid & determine new centroid.
- Repeat until distance between samples & centroids is minimized.
- Stop, when methods converges.

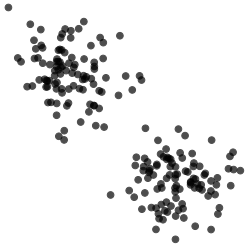






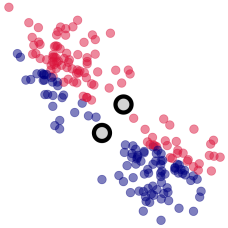
kMeans

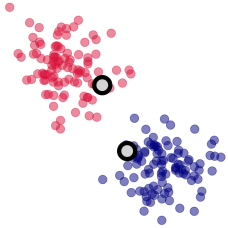
- Assign each sample to closes centroid & determine new centroid.
- Repeat until distance between samples & centroids is minimized.
- Stop, when methods converges.

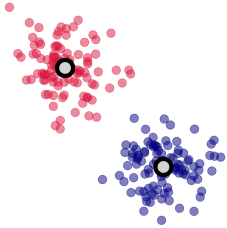


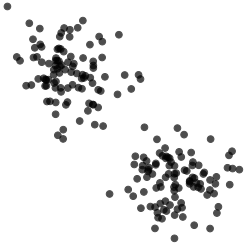
kMeans

- Assign each sample to closes centroid & determine new centroid.
- Repeat until distance between samples & centroids is minimized.
- Stop, when methods converges.



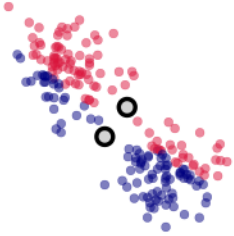


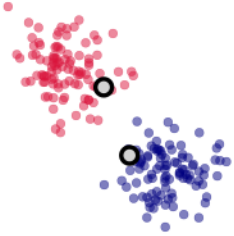


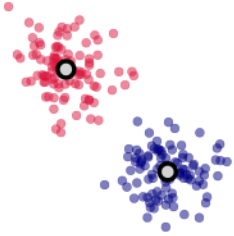


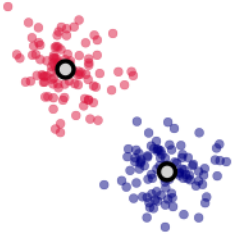
kMeans

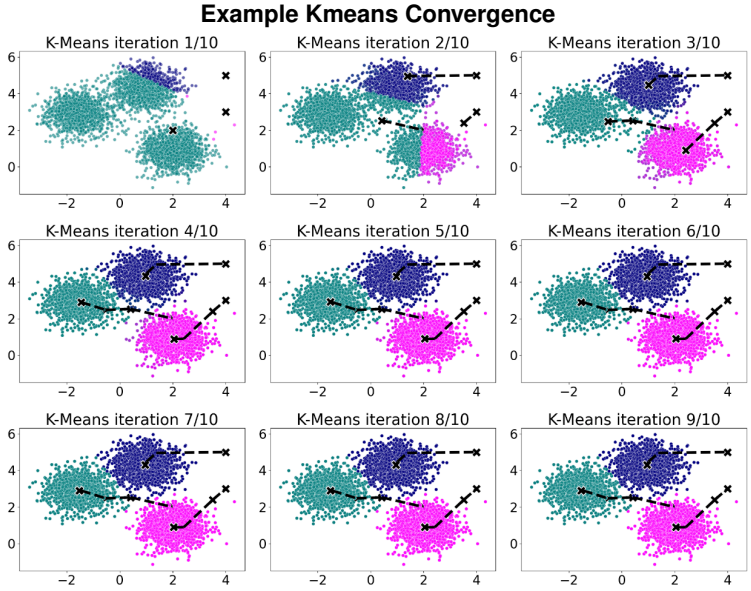
- Assign each sample to closes centroid & determine new centroid.
- Repeat until distance between samples & centroids is minimized.
- Stop, when methods converges.










Example Kmeans Convergence
K-Means iteration 1/10
K-Means iteration 2/10
K-Means iteration 3/10
K-Means iteration 4/10
K-Means iteration 5/10
K-Means iteration 6/10
K-Means iteration 7/10
K-Means iteration 8/10
K-Means iteration 9/10

kMeans Special Case of Expectation-Maximization Algorithm

Input: Dataset $\mathbf{X}$, and K the number of clusters

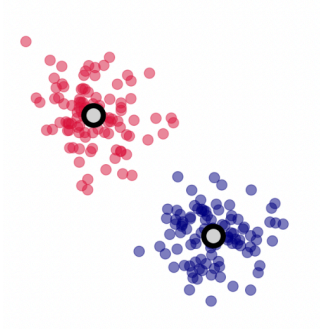
- Define K random centroids randomly located $[\mathbf{c}_1, \ldots, \mathbf{c}_K]$.
- Repeat until converged (centroid positions don't change)
  - **E-step:** Assign row $[\mathbf{X}]_{i,:}$ to closest centroid.

$$\hat{k} = \arg\min_k d([\mathbf{X}]_{i,:}, \mathbf{c}_k)$$

$$\hat{y}_i = \{\hat{k} | \hat{k} \in \{1, \ldots, K\}\}$$

  - **M-step:** Compute new centroid c_k mean of points assigned to cluster $\hat{k}$:

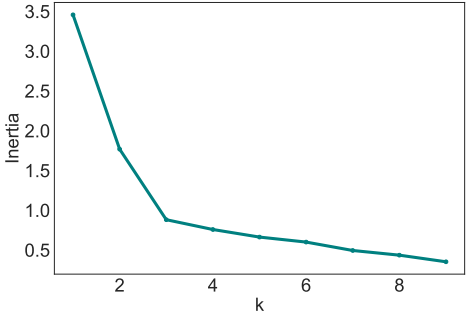
$$\mathbf{c}_k = \frac{1}{m_k} \sum_{j=1}^{m_k} [\mathbf{X}]_{j,:}, \quad j \in \{1, \ldots, m_k\}, \hat{y}_j = \hat{k}$$



The Elbow Method

- Execute K-means clustering on dataset for each $k \in \{1, \ldots, K\}$.
- Determine mean squared distances (inertias) between instances and centroids.
- Plot inertias over number of clusters k.
- The value of k for the inertia that forms the elbow (starts bending) is a reasonable tradeoff.

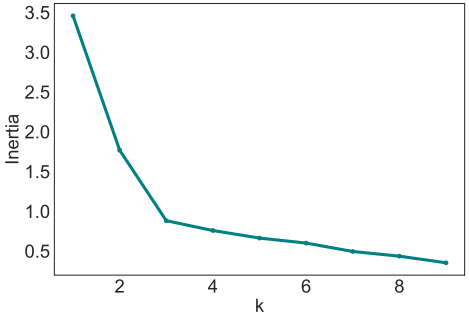
Elbow at $k = 3$



The Elbow Method

- Execute K-means clustering on dataset for each $k \in \{1, \ldots, K\}$.
- Determine mean squared distances (inertias) between instances and centroids.
- Plot inertias over number of clusters k.
- The value of k for the inertia that forms the elbow (starts bending) is a reasonable tradeoff.

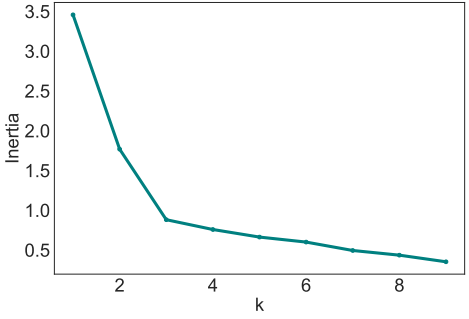
Elbow at $k = 3$



The Elbow Method

- Execute K-means clustering on dataset for each $k \in \{1, \ldots, K\}$.
- Determine mean squared distances (inertias) between instances and centroids.
- Plot inertias over number of clusters k.
- The value of k for the inertia that forms the elbow (starts bending) is a reasonable tradeoff.

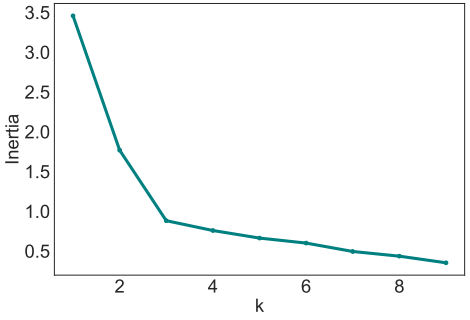
Elbow at $k = 3$

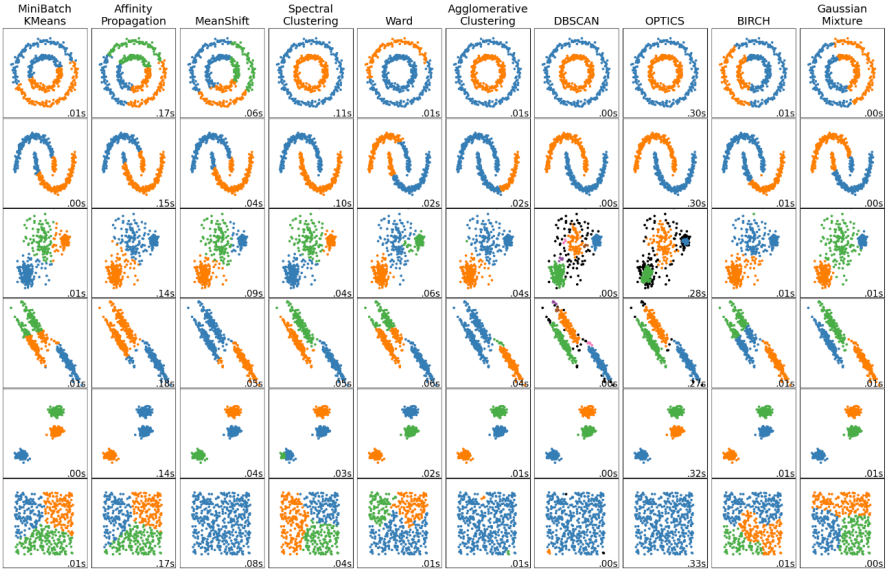


The Elbow Method

- Execute K-means clustering on dataset for each $k \in \{1, \ldots, K\}$.
- Determine mean squared distances (inertias) between instances and centroids.
- Plot inertias over number of clusters k.
- The value of k for the inertia that forms the elbow (starts bending) is a reasonable tradeoff.

Elbow at $k = 3$



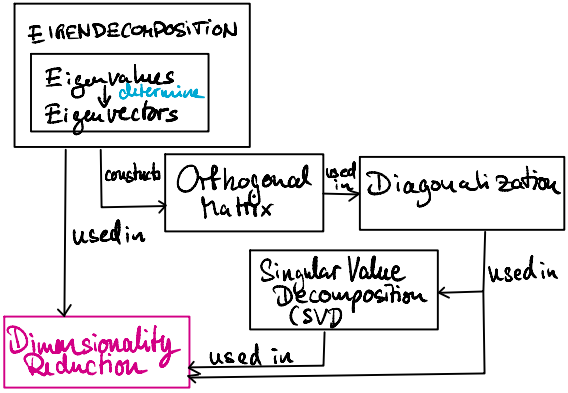


A comparison of the clustering algorithms in scikit-learn

Source:
https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html

What we Theoretically Need to Understand Dimensionality Reduction

- Standard Dimensionality Reduction needs Singular Value Decomposition (SVD).
- SVD requires Eigendecompositions.



Definition (Eigenvalue and Eigenvector)

Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be a square matrix.

Equivalent Properties for Eigenvector and -values.

Note that the following properties are equivalent

- λ is an eigenvalue of $\mathbf{A} \in \mathbb{R}^{n \times n}$.
- There exists $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ with $\mathbf{A}\mathbf{x} = \lambda \mathbf{x}$,
- or equivalently $(\mathbf{A} - \lambda \mathbf{I}_n)\mathbf{x} = 0$ can be solved non-trivially, i.e. $\mathbf{x} \neq \mathbf{0}$
- $\det(\mathbf{A} - \lambda \mathbf{I}_n) = 0$.

Definition (Eigenvalue and Eigenvector)

Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be a square matrix.
Then $\lambda \in \mathbb{R}$ is an eigenvalue of $\mathbf{A}$ and $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ is the corresponding eigenvector of $\mathbf{A}$ if

$$\mathbf{A}\mathbf{x} = \lambda \mathbf{x}$$

Equivalent Properties for Eigenvector and -values.

Note that the following properties are equivalent

- λ is an eigenvalue of $\mathbf{A} \in \mathbb{R}^{n \times n}$.
- There exists $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ with $\mathbf{A}\mathbf{x} = \lambda \mathbf{x}$,
- or equivalently $(\mathbf{A} - \lambda \mathbf{I}_n)\mathbf{x} = 0$ can be solved non-trivially, i.e. $\mathbf{x} \neq \mathbf{0}$
- $\det(\mathbf{A} - \lambda \mathbf{I}_n) = 0$.

Definition (Eigenvalue and Eigenvector)

Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be a square matrix.
Then $\lambda \in \mathbb{R}$ is an eigenvalue of $\mathbf{A}$ and $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ is the corresponding eigenvector of $\mathbf{A}$ if

$$\mathbf{A}\mathbf{x} = \lambda \mathbf{x}$$

Equivalent Properties for Eigenvector and -values.

Note that the following properties are equivalent

- λ is an eigenvalue of $\mathbf{A} \in \mathbb{R}^{n \times n}$.
- There exists $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ with $\mathbf{A}\mathbf{x} = \lambda \mathbf{x}$,
- or equivalently $(\mathbf{A} - \lambda \mathbf{I}_n)\mathbf{x} = 0$ can be solved non-trivially, i.e. $\mathbf{x} \neq \mathbf{0}$
- $\det(\mathbf{A} - \lambda \mathbf{I}_n) = 0$.

Definition (Eigenvalue and Eigenvector)

Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be a square matrix.
Then $\lambda \in \mathbb{R}$ is an eigenvalue of $\mathbf{A}$ and $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ is the corresponding eigenvector of $\mathbf{A}$ if

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$$

Equivalent Properties for Eigenvector and -values.

Note that the following properties are equivalent

- λ is an eigenvalue of $\mathbf{A} \in \mathbb{R}^{n \times n}$.
- There exists $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ with $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$,
- or equivalently $(\mathbf{A} - \lambda\mathbf{I}_n)\mathbf{x} = 0$ can be solved non-trivially, i.e. $\mathbf{x} \neq \mathbf{0}$
- $\det(\mathbf{A} - \lambda\mathbf{I}_n) = 0$.

Definition (Eigenvalue and Eigenvector)

Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be a square matrix.
Then $\lambda \in \mathbb{R}$ is an eigenvalue of $\mathbf{A}$ and $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ is the corresponding eigenvector of $\mathbf{A}$ if

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$$

Equivalent Properties for Eigenvector and -values.

Note that the following properties are equivalent

- λ is an eigenvalue of $\mathbf{A} \in \mathbb{R}^{n \times n}$.
- There exists $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ with $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$,
- or equivalently $(\mathbf{A} - \lambda\mathbf{I}_n)\mathbf{x} = 0$ can be solved non-trivially, i.e. $\mathbf{x} \neq \mathbf{0}$
- $\det(\mathbf{A} - \lambda\mathbf{I}_n) = 0$.

Definition (Eigenvalue and Eigenvector)

Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be a square matrix.
Then $\lambda \in \mathbb{R}$ is an eigenvalue of $\mathbf{A}$ and $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ is the corresponding eigenvector of $\mathbf{A}$ if

$$\mathbf{A}\mathbf{x} = \lambda \mathbf{x}$$

Equivalent Properties for Eigenvector and -values.

Note that the following properties are equivalent

- λ is an eigenvalue of $\mathbf{A} \in \mathbb{R}^{n \times n}$.
- There exists $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ with $\mathbf{A}\mathbf{x} = \lambda \mathbf{x}$,
- or equivalently $(\mathbf{A} - \lambda \mathbf{I}_n)\mathbf{x} = 0$ can be solved non-trivially, i.e. $\mathbf{x} \neq \mathbf{0}$
- $\det(\mathbf{A} - \lambda \mathbf{I}_n) = 0$.

Definition (Eigenvalue and Eigenvector)

Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be a square matrix.
Then $\lambda \in \mathbb{R}$ is an eigenvalue of $\mathbf{A}$ and $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ is the corresponding eigenvector of $\mathbf{A}$ if

$$\mathbf{A}\mathbf{x} = \lambda \mathbf{x}$$

Equivalent Properties for Eigenvector and -values.

Note that the following properties are equivalent

- λ is an eigenvalue of $\mathbf{A} \in \mathbb{R}^{n \times n}$.
- There exists $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ with $\mathbf{A}\mathbf{x} = \lambda \mathbf{x}$,
- or equivalently $(\mathbf{A} - \lambda \mathbf{I}_n)\mathbf{x} = 0$ can be solved non-trivially, i.e. $\mathbf{x} \neq \mathbf{0}$
- $\det(\mathbf{A} - \lambda \mathbf{I}_n) = 0$.

Theorem

$\lambda \in \mathbb{R}$ is an eigenvalue of $\mathbf{A} \in \mathbb{R}^{n \times n}$ if and only if λ is a root of the characteristic polynomial $p_{\mathbf{A}}(\lambda) = \det(\mathbf{A} - \lambda \mathbf{I})$ of $\mathbf{A}$.

Theorem (Eigendecomposition)

A square matrix $\mathbf{A} \in \mathbf{A} \in \mathbb{R}^{N \times N}$ can be factored into

$$\mathbf{A} = \mathbf{P}\mathbf{D}\mathbf{P}^{-1}, \quad \mathbf{D} = \begin{bmatrix} \lambda_1 & \dots 0 & \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_N \end{bmatrix}$$

where $\mathbf{P} \in \mathbb{R}^{N \times N}$ and $\mathbf{D}$ is a diagonal matrix whose diagonal entries are the eigenvalues of $\mathbf{A}$.

Theorem (Eigendecomposition)

A square matrix $\mathbf{A} \in \mathbf{A} \in \mathbb{R}^{N \times N}$ can be factored into

$$\mathbf{A} = \mathbf{P}\mathbf{D}\mathbf{P}^{-1}, \quad \mathbf{D} = \begin{bmatrix} \lambda_1 & \dots 0 & \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_N \end{bmatrix}$$

where $\mathbf{P} \in \mathbb{R}^{N \times N}$ and $\mathbf{D}$ is a diagonal matrix whose diagonal entries are the eigenvalues of $\mathbf{A}$.

How To Determine Eigendecomposition

- **Small matrix $\mathbf{A}$:** Eigenvalues $\lambda_1, \dots \lambda_N$ can be calculated as roots of characteristic polynomial
- **Small matrix $\mathbf{A}$:** Eigenvectors $\mathbf{p}$ can be found solving $\mathbf{A}\mathbf{p}_n = \lambda_n \mathbf{p}_n$
- **Example small matrix $\mathbf{A}$:**

$$\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \rightarrow \det(\mathbf{A} - \lambda \mathbf{I}) = \det\left(\begin{bmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{bmatrix} \right)$$
$$= (2-\lambda)^2 - 1 = \lambda^2 - 4\lambda + 3 = (\lambda - 3)(\lambda - 1) \rightarrow \lambda_1 = 1, \quad \lambda_2 = 3$$

- **Large matrices $\mathbf{A}$:** Numerical methods are used to solve the Eigendecomposition.

Theorem (Eigendecomposition)

A square matrix $\mathbf{A} \in \mathbf{A} \in \mathbb{R}^{N \times N}$ can be factored into

$$\mathbf{A} = \mathbf{P}\mathbf{D}\mathbf{P}^{-1}, \quad \mathbf{D} = \begin{bmatrix} \lambda_1 & \dots 0 & \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_N \end{bmatrix}$$

where $\mathbf{P} \in \mathbb{R}^{N \times N}$ and $\mathbf{D}$ is a diagonal matrix whose diagonal entries are the eigenvalues of $\mathbf{A}$.

How To Determine Eigendecomposition

- **Small matrix $\mathbf{A}$:** Eigenvalues $\lambda_1, \dots \lambda_N$ can be calculated as roots of characteristic polynomial
- **Small matrix $\mathbf{A}$:** Eigenvectors $\mathbf{p}$ can be found solving $\mathbf{A}\mathbf{p}_n = \lambda_n \mathbf{p}_n$
- **Example small matrix $\mathbf{A}$:**

$$\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \rightarrow \det(\mathbf{A} - \lambda \mathbf{I}) = \det\left(\begin{bmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{bmatrix}\right)$$
$$= (2-\lambda)^2 - 1 = \lambda^2 - 4\lambda + 3 = (\lambda - 3)(\lambda - 1) \rightarrow \lambda_1 = 1, \quad \lambda_2 = 3$$

- **Large matrices $\mathbf{A}$:** Numerical methods are used to solve the Eigendecomposition.

Theorem (Eigendecomposition)

A square matrix $\mathbf{A} \in \mathbf{A} \in \mathbb{R}^{N \times N}$ can be factored into

$$\mathbf{A} = \mathbf{P}\mathbf{D}\mathbf{P}^{-1}, \quad \mathbf{D} = \begin{bmatrix} \lambda_1 & \dots 0 & \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_N \end{bmatrix}$$

where $\mathbf{P} \in \mathbb{R}^{N \times N}$ and $\mathbf{D}$ is a diagonal matrix whose diagonal entries are the eigenvalues of $\mathbf{A}$.

How To Determine Eigendecomposition

- **Small matrix $\mathbf{A}$:** Eigenvalues $\lambda_1, \dots \lambda_N$ can be calculated as roots of characteristic polynomial
- **Small matrix $\mathbf{A}$:** Eigenvectors $\mathbf{p}$ can be found solving $\mathbf{A}\mathbf{p}_n = \lambda_n \mathbf{p}_n$
- **Example small matrix $\mathbf{A}$:**

$$\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \rightarrow \det(\mathbf{A} - \lambda \mathbf{I}) = \det\left(\begin{bmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{bmatrix}\right)$$
$$= (2-\lambda)^2 - 1 = \lambda^2 - 4\lambda + 3 = (\lambda - 3)(\lambda - 1) \rightarrow \lambda_1 = 1, \quad \lambda_2 = 3$$

- **Large matrices $\mathbf{A}$:** Numerical methods are used to solve the Eigendecomposition.

Theorem (Eigendecomposition)

A square matrix $\mathbf{A} \in \mathbf{A} \in \mathbb{R}^{N \times N}$ can be factored into

$$\mathbf{A} = \mathbf{P}\mathbf{D}\mathbf{P}^{-1}, \quad \mathbf{D} = \begin{bmatrix} \lambda_1 & \dots 0 & \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_N \end{bmatrix}$$

where $\mathbf{P} \in \mathbb{R}^{N \times N}$ and $\mathbf{D}$ is a diagonal matrix whose diagonal entries are the eigenvalues of $\mathbf{A}$.

How To Determine Eigendecomposition

- **Small matrix $\mathbf{A}$:** Eigenvalues $\lambda_1, \dots \lambda_N$ can be calculated as roots of characteristic polynomial
- **Small matrix $\mathbf{A}$:** Eigenvectors $\mathbf{p}$ can be found solving $\mathbf{A}\mathbf{p}_n = \lambda_n \mathbf{p}_n$
- **Example small matrix $\mathbf{A}$**:

$$\begin{aligned} \mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \rightarrow \det(\mathbf{A} - \lambda \mathbf{I}) &= \det\left(\begin{bmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{bmatrix} \right) \\ &= (2-\lambda)^2 - 1 = \lambda^2 - 4\lambda + 3 = (\lambda - 3)(\lambda - 1) \rightarrow \lambda_1 = 1, \ \ \lambda_2 = 3 \end{aligned}$$

- **Large matrices $\mathbf{A}$**: Numerical methods are used to solve the Eigendecomposition.

Theorem - Singular Value Decomposition (SVD)

Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ let be a matrix of rank $r \in [0, \min(m, n)]$.
The SVD of $\mathbf{A}$ is a decomposition of the form

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T,$$

- with an othogonal matrix $\mathbf{U} \in \mathbb{R}^{m \times m}$ with column vectors $\mathbf{u}_i, \quad i \in \{1, \ldots, m\}$.
- with an othogonal matrix $\mathbf{V} \in \mathbb{R}^{n \times n}$ with column vectors $\mathbf{v}_j, \quad j \in \{1, \ldots, n\}$.
- with a matrix

$$\mathbf{\Sigma} \in \mathbb{R}^{m \times n} \text{ with } [\mathbf{\Sigma}]_{i,j} = \begin{cases} \sigma_i \geq 0 & i = j \\ 0 & \text{else.} \end{cases}$$

Theorem - Singular Value Decomposition (SVD)

Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ let be a matrix of rank $r \in [0, \min(m, n)]$.
The SVD of $\mathbf{A}$ is a decomposition of the form

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T,$$

- with an othogonal matrix $\mathbf{U} \in \mathbb{R}^{m \times m}$ with column vectors $\mathbf{u}_i, \quad i \in \{1, \ldots, m\}$.
- with an othogonal matrix $\mathbf{V} \in \mathbb{R}^{n \times n}$ with column vectors $\mathbf{v}_j, \quad j \in \{1, \ldots, n\}$.
- with a matrix $\mathbf{\Sigma} \in \mathbb{R}^{m \times n}$ with $[\mathbf{\Sigma}]_{i,j} = \begin{cases} \sigma_i \geq 0 & i = j \\ 0 & \text{else.} \end{cases}$

Theorem - Singular Value Decomposition (SVD)

Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ let be a matrix of rank $r \in [0, \min(m, n)]$.
The SVD of $\mathbf{A}$ is a decomposition of the form

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T,$$

- with an othogonal matrix $\mathbf{U} \in \mathbb{R}^{m \times m}$ with column vectors $\mathbf{u}_i, \quad i \in \{1, \ldots, m\}$.
- with an othogonal matrix $\mathbf{V} \in \mathbb{R}^{n \times n}$ with column vectors $\mathbf{v}_j, \quad j \in \{1, \ldots, n\}$.
- with a matrix

$$\mathbf{\Sigma} \in \mathbb{R}^{m \times n} \text{ with } [\mathbf{\Sigma}]_{i,j} = \begin{cases} \sigma_i \geq 0 & i = j \\ 0 & \text{else.} \end{cases}$$

Theorem - Singular Value Decomposition (SVD)

Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ let be a matrix of rank $r \in [0, \min(m, n)]$.
The SVD of $\mathbf{A}$ is a decomposition of the form

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T,$$

- with an othogonal matrix $\mathbf{U} \in \mathbb{R}^{m \times m}$ with column vectors $\mathbf{u}_i, \quad i \in \{1, \ldots, m\}$.
- with an othogonal matrix $\mathbf{V} \in \mathbb{R}^{n \times n}$ with column vectors $\mathbf{v}_j, \quad j \in \{1, \ldots, n\}$.
- with a matrix

$$\mathbf{\Sigma} \in \mathbb{R}^{m \times n} \text{ with } [\mathbf{\Sigma}]_{i,j} = \begin{cases} \sigma_i \geq 0 & i = j \\ 0 & \text{else.} \end{cases}$$

Theorem - Singular Value Decomposition (SVD)

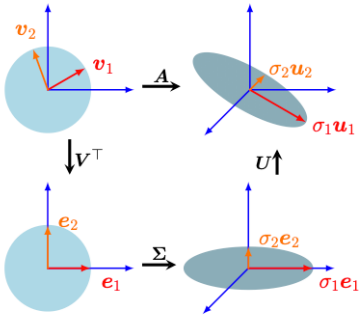
Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ let be a matrix of rank $r \in [0, \min(m, n)]$.
The SVD of $\mathbf{A}$ is a decomposition of the form

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T,$$

- with an othogonal matrix $\mathbf{U} \in \mathbb{R}^{m \times m}$ with column vectors $\mathbf{u}_i, \quad i \in \{1, \ldots, m\}$.
- with an othogonal matrix $\mathbf{V} \in \mathbb{R}^{n \times n}$ with column vectors $\mathbf{v}_j, \quad j \in \{1, \ldots, n\}$.
- with a matrix

$$\mathbf{\Sigma} \in \mathbb{R}^{m \times n} \text{ with } [\mathbf{\Sigma}]_{i,j} = \begin{cases} \sigma_i \geq 0 & i = j \\ 0 & \text{else.} \end{cases}$$

Geometric Interpretation



Source: Deisenroth et al., *Mathematics for Machine Learning*

Geometric Interpretation

- $\mathbf{V}^T$ performs a basis change into standard basis.
- $\mathbf{\Sigma}$ scales the basis vectors and maps to different domain.
- $\mathbf{U}$ performs a basis change represented by rotation.

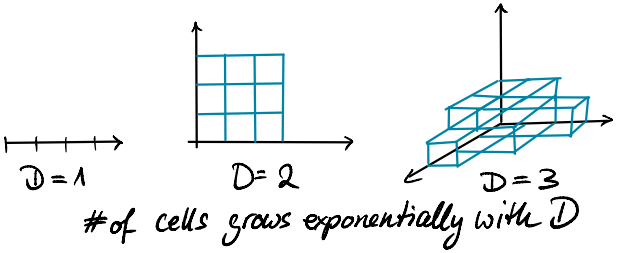
How are The Eigendecomposition and the SVD related?

In the SVD construction for $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$

- the eigenvalues of $\mathbf{A}^T\mathbf{A}$ are the squared singular values of $\mathbf{\Sigma}$.
- For square symmetric matrices the SVD coincides with the eigendecomposition.

Information on SVD and Eigendecomposition Fundamentals

- is only relevant to understand the principle of PCA.
- Will not be part of the exam.
- The interested reader finds the full derivation and relationship in Deisenroth et al., *Mathematics for Machine Learning*, p. (88-115)



The Curse of Dimensionality

Many ML problems involve thousands or even millions of features implying:

- Slow or computationally expensive training processes.
- Bad convergence.

Dimensionality Reduction

Finding a smaller feature set representing (compressing) the original feature vector information.

Intution - Principal Component Analysis (PCA)

- **Principle:** Project data on to closest hyperplane to data.
- **Or:** Finds best subspace that data approximately lies in.

Intution - Principal Component Analysis (PCA)

- **Principle:** Project data on to closest hyperplane to data.
- **Or:** Finds best subspace that data approximately lies in.

Intution - Principal Component Analysis (PCA)

- **Principle:** Project data on to closest hyperplane to data.
- **Or:** Finds best subspace that data approximately lies in.

Nomenclature

- Let $\mathbf{X} \in \mathbb{R}^{N \times 2}$ define the input dataset
- Let $\mathbf{U}$ denote a matrix filled with column vectors defining the hyperplane.
- Let $\mathbf{Z} \in \mathbb{R}^{N \times d}$ define a reduced transformation of $\mathbf{X}$ with dimensions $d < N$.

Intution - Principal Component Analysis (PCA)

- **Principle:** Project data on to closest hyperplane to data.
- **Or:** Finds best subspace that data approximately lies in.

Nomenclature

- Let $\mathbf{X} \in \mathbb{R}^{N \times 2}$ define the input dataset
- Let $\mathbf{U}$ denote a matrix filled with column vectors defining the hyperplane.
- Let $\mathbf{Z} \in \mathbb{R}^{N \times d}$ define a reduced transformation of $\mathbf{X}$ with dimensions $d < N$.

Intution - Principal Component Analysis (PCA)

- **Principle:** Project data on to closest hyperplane to data.
- **Or:** Finds best subspace that data approximately lies in.

Nomenclature

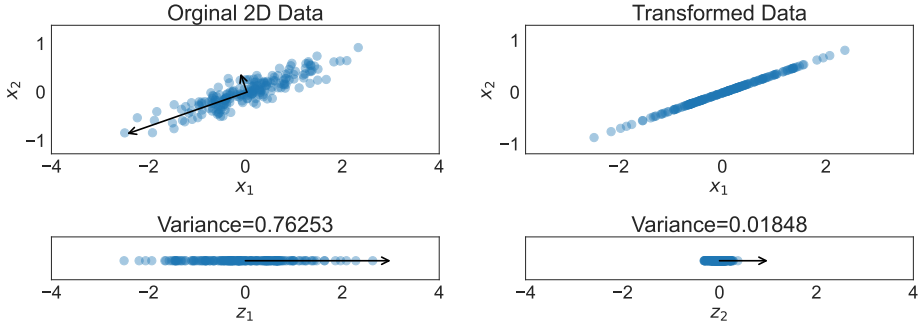
- Let $\mathbf{X} \in \mathbb{R}^{N \times 2}$ define the input dataset
- Let $\mathbf{U}$ denote a matrix filled with column vectors defining the hyperplane.
- Let $\mathbf{Z} \in \mathbb{R}^{N \times d}$ define a reduced transformation of $\mathbf{X}$ with dimensions $d < N$.

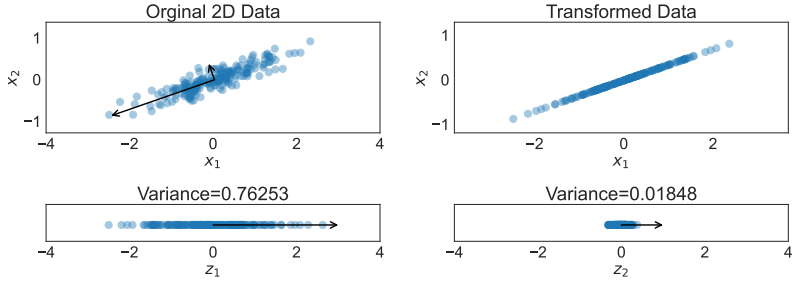
Intution - Principal Component Analysis (PCA)

- **Principle:** Project data on to closest hyperplane to data.
- **Or:** Finds best subspace that data approximately lies in.

Nomenclature

- Let $\mathbf{X} \in \mathbb{R}^{N \times 2}$ define the input dataset
- Let $\mathbf{U}$ denote a matrix filled with column vectors defining the hyperplane.
- Let $\mathbf{Z} \in \mathbb{R}^{N \times d}$ define a reduced transformation of $\mathbf{X}$ with dimensions $d < N$.



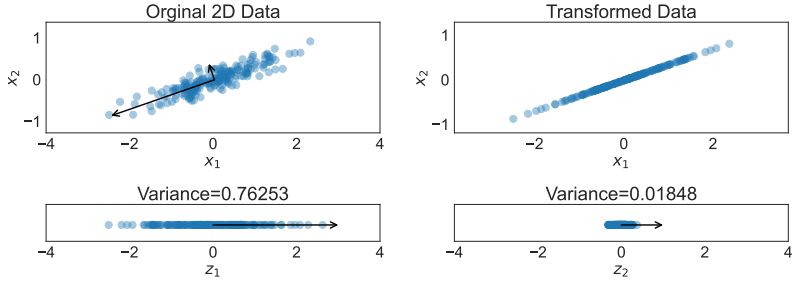


Principle Component Analysis (PCA)

- Find vector u_1 maximizing data variance in projected space best.
- Find vector u_2 maximizing data variance in projected space second best.
- Find vector u_3 maximizing data variance in projected space third best.
- Proceed as above until $d < N$ is reached.

Which Components do we Keep?

The procedure retains the highest variability principal components.

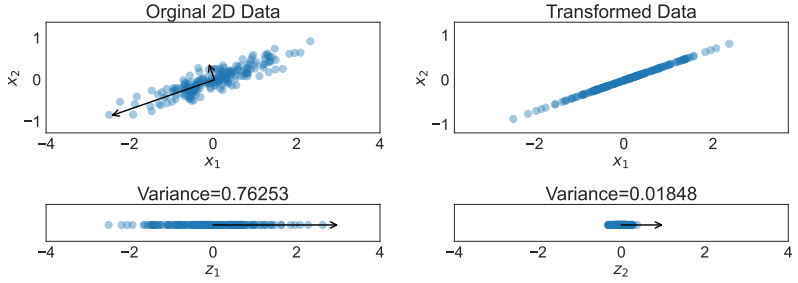


Principle Component Analysis (PCA)

- Find vector u_1 maximizing data variance in projected space best.
- Find vector u_2 maximizing data variance in projected space second best.
- Find vector u_3 maximizing data variance in projected space third best.
- Proceed as above until $d < N$ is reached.

Which Components do we Keep?

The procedure retains the highest variability principal components.

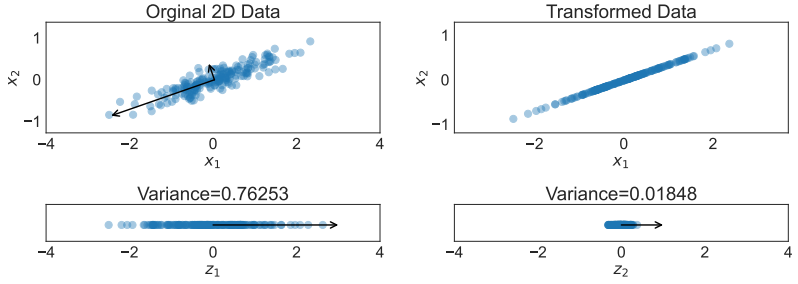


Principle Component Analysis (PCA)

- Find vector u_1 maximizing data variance in projected space best.
- Find vector u_2 maximizing data variance in projected space second best.
- Find vector u_3 maximizing data variance in projected space third best.
- Proceed as above until $d < N$ is reached.

Which Components do we Keep?

The procedure retains the highest variability principal components.



Principle Component Analysis (PCA)

- Find vector u_1 maximizing data variance in projected space best.
- Find vector u_2 maximizing data variance in projected space second best.
- Find vector u_3 maximizing data variance in projected space third best.
- Proceed as above until $d < N$ is reached.

Which Components do we Keep?

The procedure retains the highest variability principal components.

SVD and PCA

SVD decomposes matrix $\mathbf{A}$ into $\mathbf{A} = \frac{1}{N}\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. The covariance matrix of $\mathbf{X}$ links the SVD and the Eigendecomposition

$$\mathbf{C}_\mathbf{X} = \frac{1}{n}\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}\mathbf{V}^T$$

- $\mathbf{\Sigma}$ contains the singular values (square roots of non-zero eigenvalues).
- The columns of $\mathbf{U}$ (or the eigenvectors) of $\mathbf{C}_\mathbf{X}$ are the principle components of $\mathbf{X}$.
- To reduce dimensionality $\rightarrow$ discard all columns $> d$.

SVD and PCA

SVD decomposes matrix $\mathbf{A}$ into $\mathbf{A} = \frac{1}{N}\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. The covariance matrix of $\mathbf{X}$ links the SVD and the Eigendecomposition

$$\mathbf{C}_\mathbf{X} = \frac{1}{n}\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}\mathbf{V}^T$$

- $\mathbf{\Sigma}$ contains the singular values (square roots of non-zero eigenvalues).
- The columns of $\mathbf{U}$ (or the eigenvectors) of $\mathbf{C}_\mathbf{X}$ are the principle components of $\mathbf{X}$.
- To reduce dimensionality $\rightarrow$ discard all columns $> d$.

PCA - Algorithm

Let the input dataset be $\mathbf{X} \in \mathbb{R}^{N \times M}$.

- Normalize $\mathbf{X}$ (mean normalization).
- Compute covariance matrix $\mathbf{C} - \mathbf{X} = \frac{1}{N}\mathbf{X}^T\mathbf{X}$.
- Determine SVD or eigenvectors of $\mathbf{C}_\mathbf{X}$.
- Reduce dimensionality of $\mathbf{U}$ to $d \leq M$ by $\mathbf{U}_r = [\mathbf{U}]_{(:,1:d)}$.
- Transform data into

$$\mathbf{Z} = \mathbf{U}_r^T\mathbf{X}.$$

SVD and PCA

SVD decomposes matrix $\mathbf{A}$ into $\mathbf{A} = \frac{1}{N}\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. The covariance matrix of $\mathbf{X}$ links the SVD and the Eigendecomposition

$$\mathbf{C}_\mathbf{X} = \frac{1}{n}\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}\mathbf{V}^T$$

- $\mathbf{\Sigma}$ contains the singular values (square roots of non-zero eigenvalues).
- The columns of $\mathbf{U}$ (or the eigenvectors) of $\mathbf{C}_\mathbf{X}$ are the principle components of $\mathbf{X}$.
- To reduce dimensionality $\rightarrow$ discard all columns $> d$.

PCA - Algorithm

Let the input dataset be $\mathbf{X} \in \mathbb{R}^{N \times M}$.

- Normalize $\mathbf{X}$ (mean normalization).
- Compute covariance matrix $\mathbf{C} - \mathbf{X} = \frac{1}{N}\mathbf{X}^T\mathbf{X}$.
- Determine SVD or eigenvectors of $\mathbf{C}_\mathbf{X}$.
- Reduce dimensionality of $\mathbf{U}$ to $d \leq M$ by $\mathbf{U}_r = [\mathbf{U}]_{(:,1:d)}$.
- Transform data into

$$\mathbf{Z} = \mathbf{U}_r^T\mathbf{X}.$$

SVD and PCA

SVD decomposes matrix $\mathbf{A}$ into $\mathbf{A} = \frac{1}{N}\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. The covariance matrix of $\mathbf{X}$ links the SVD and the Eigendecomposition

$$\mathbf{C}_\mathbf{X} = \frac{1}{n}\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}\mathbf{V}^T$$

- $\mathbf{\Sigma}$ contains the singular values (square roots of non-zero eigenvalues).
- The columns of $\mathbf{U}$ (or the eigenvectors) of $\mathbf{C}_\mathbf{X}$ are the principle components of $\mathbf{X}$.
- To reduce dimensionality $\rightarrow$ discard all columns $> d$.

PCA - Algorithm

Let the input dataset be $\mathbf{X} \in \mathbb{R}^{N \times M}$.

- Normalize $\mathbf{X}$ (mean normalization).
- Compute covariance matrix $\mathbf{C} - \mathbf{X} = \frac{1}{N}\mathbf{X}^T\mathbf{X}$.
- Determine SVD or eigenvectors of $\mathbf{C}_\mathbf{X}$.
- Reduce dimensionality of $\mathbf{U}$ to $d \leq M$ by $\mathbf{U}_r = [\mathbf{U}]_{(:,1:d)}$.
- Transform data into

$$\mathbf{Z} = \mathbf{U}_r^T\mathbf{X}.$$

SVD and PCA

SVD decomposes matrix $\mathbf{A}$ into $\mathbf{A} = \frac{1}{N}\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. The covariance matrix of $\mathbf{X}$ links the SVD and the Eigendecomposition

$$\mathbf{C}_\mathbf{X} = \frac{1}{n}\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}\mathbf{V}^T$$

- $\mathbf{\Sigma}$ contains the singular values (square roots of non-zero eigenvalues).
- The columns of $\mathbf{U}$ (or the eigenvectors) of $\mathbf{C}_\mathbf{X}$ are the principle components of $\mathbf{X}$.
- To reduce dimensionality $\rightarrow$ discard all columns $> d$.

PCA - Algorithm

Let the input dataset be $\mathbf{X} \in \mathbb{R}^{N \times M}$.

- Normalize $\mathbf{X}$ (mean normalization).
- Compute covariance matrix $\mathbf{C} - \mathbf{X} = \frac{1}{N}\mathbf{X}^T\mathbf{X}$.
- Determine SVD or eigenvectors of $\mathbf{C}_\mathbf{X}$.
- Reduce dimensionality of $\mathbf{U}$ to $d \leq M$ by $\mathbf{U}_r = [\mathbf{U}]_{(:,1:d)}$.
- Transform data into

$$\mathbf{Z} = \mathbf{U}_r^T\mathbf{X}.$$

SVD and PCA

SVD decomposes matrix $\mathbf{A}$ into $\mathbf{A} = \frac{1}{N}\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. The covariance matrix of $\mathbf{X}$ links the SVD and the Eigendecomposition

$$\mathbf{C}_\mathbf{X} = \frac{1}{n}\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}\mathbf{V}^T$$

- $\mathbf{\Sigma}$ contains the singular values (square roots of non-zero eigenvalues).
- The columns of $\mathbf{U}$ (or the eigenvectors) of $\mathbf{C}_\mathbf{X}$ are the principle components of $\mathbf{X}$.
- To reduce dimensionality $\rightarrow$ discard all columns $> d$.

PCA - Algorithm

Let the input dataset be $\mathbf{X} \in \mathbb{R}^{N \times M}$.

- Normalize $\mathbf{X}$ (mean normalization).
- Compute covariance matrix $\mathbf{C} - \mathbf{X} = \frac{1}{N}\mathbf{X}^T\mathbf{X}$.
- Determine SVD or eigenvectors of $\mathbf{C}_\mathbf{X}$.
- Reduce dimensionality of $\mathbf{U}$ to $d \leq M$ by $\mathbf{U}_r = [\mathbf{U}]_{(:,1:d)}$.
- Transform data into

$$\mathbf{Z} = \mathbf{U}_r^T\mathbf{X}.$$

Explained Variance Ratio

In PCA we can compute the variance fraction captured by d selected components.

Criteria for Choosing Dimensionality d

Keep as many principal components as needed to retain 99% of the explained variance ratio.