

# Machine learning – Selbsthilfe-Gruppe

$\alpha$ Kevin ft. features

Kevin Lamkiewicz

Friedrich Schiller University Jena

20.07.2017

# Einführung

# NO FREE LUNCH

# NO FREE LUNCH

Theorem 1:

$$\sum_f \mathcal{P}(h_m^y | f, m, a_1) = \sum_f \mathcal{P}(h_m^y | f, m, a_2)$$

# NO FREE LUNCH

## Theorem 1:

$$\sum_f \mathcal{P}(h_m^y | f, m, a_1) = \sum_f \mathcal{P}(h_m^y | f, m, a_2)$$

In Worten:

Angenommen das Auftreten aller Funktionen  $f$  ist gleichverteilt, dann ist die Wahrscheinlichkeit  $\mathcal{P}$  unabhängig vom gewählten Algorithmus.

# FEATURES

- ▶ idealerweise quantifizierend

# FEATURES

- ▶ idealerweise quantifizierend
- ▶ alles, was euch so einfällt...
  - ⇒ sinnvolle Trennung von Klassen

## BEISPIEL: FEATURES

### RNAs

Wir sollen pflanzliche mRNA von humaner mRNA unterscheiden.  
Welche features können wir dazu nehmen?



# BEISPIEL: FEATURES

## RNAs

Wir sollen pflanzliche mRNA von humaner mRNA unterscheiden.  
Welche features können wir dazu nehmen?

- ▶ mRNA Länge
- ▶ durchschnittliche Exon/Intron Länge
- ▶ Anzahl Exons/Introns
- ▶ GC-Gehalt (%)
- ▶ Codon Frequenz
- ▶ ...

# Feature Design in Python

# IMPLEMENTIERUNG IN PYTHON

```
1  #!/usr/bin/env python3
2  ...
3  X = [[0,1.5,2.13,0,0], [2,0.04,1.3,0.3,3]]
4  y = ["class1", "class2"]
5  ...
```

# IMPLEMENTIERUNG IN PYTHON

```
1  #!/usr/bin/env python3
2  ...
3  X = [[0,1.5,2.13,0,0], [2,0.04,1.3,0.3,3]]
4  y = ["class1", "class2"]
5  ...
```

oder allgemein:

$X$  ist eine Matrix der Größe  $[n\_samples, n\_features]$

$y$  ist ein Vektor der Größe  $[n\_samples]$

# Feature Selection

# FEATURE SELECTION I

Woher weiß ich, welche features wichtig für meine Klassifikation ist?

# FEATURE SELECTION I

Woher weiß ich, welche features wichtig für meine Klassifikation ist?

scikit-learn

Einige Classifier besitzen vorimplementierte Methoden, die wichtige features identifizieren:

```
1  from sklearn.ensemble import RandomForestClassifier
2  X = [[0, 1.5, 2.13, 0, 0], [2, 0.04, 1.3, 0.3, 3]]
3  y = ["class1", "class2"]
4  clf = RandomForestClassifier()
5  clf.fit(X,y)
6  print(clf.feature_importances_)
```

## FEATURE SELECTION II

Woher weiß ich, welche features wichtig für meine Klassifikation ist?

### F-Score

$$F(i) = \frac{(\bar{x}_i^{(+)} - \bar{x}_i^2) + (\bar{x}_i^{(-)} - \bar{x}_i^2)}{\frac{1}{n_+ - 1} \sum_{k=1}^{n_+} (x_{k,i}^{(+)} - \bar{x}_i^{(+)})^2 + \frac{1}{n_- - 1} \sum_{k=1}^{n_-} (x_{k,i}^{(-)} - \bar{x}_i^{(-)})^2}$$

In Worten:

Je höher der F-Score  $F(i)$  für feature  $i$ , desto höher ist die Diskrepanz dieses features in den unterschiedlichen Klassen.



## FEATURE SELECTION III

Woher weiß ich, welche features wichtig für meine Klassifikation ist?

### Loss of accuracy

Ähnlich wie der F-Score ein Maß zur Bestimmung der *feature importance*:

1. Berechnung von Accuracy mit allen features  $N(f)$
2. Iterative Neuberechnung der Accuracy mit  $N(f) - 1$

⇒ auch bekannt als *leave-one-out* Methode

## FEATURE SELECTION IV

Woher weiß ich, welche features wichtig für meine Klassifikation ist?

Mean decrease accuracy: MDA

Idee basiert auf der leave-one-out bootstrap Methode:

1. Berechnung von Accuracy mit allen features
2. Iterative Permutation jedes features
  - ▶ wird wichtiges feature permutiert, sollte die Accuracy sinken
  - ▶ wird unwichtiges feature permutiert, sollte sich wenig ändern
3. Neuberechnung der Accuracy mit permutiertem feature

# Hands On

# EINFACHE KLASSIFIKATION

scikit-learn bringt einige Standardbeispiele mit sich:

## Iris Datensatz

DAS Paradebeispiel für Machine learning.

- ▶ Klassifikation verschiedener Schwertlilien
- ▶ {Kelchblatt, Kronblatt}{länge, breite}
- ▶ 150 Instanzen

```
1 from sklearn import datasets
2 iris = datasets.load_iris()
3 X_iris, y_iris = iris.data, iris.target
4 print(X_iris, y_iris)
5 print(X_iris[0], y_iris[0])
```

# FEATURE DESIGN

## Minimalbeispiel für DNA/RNA Sequenzen:

```
1 import random
2 from collections import Counter
3 # random sequence creation
4 seqs = [''.join([random.choice("ACGT") for l in range(50)]) \
5           for x in range(10)]
6
7 # important features design code
8 dataframe = []
9 for s in seqs:
10     vector = [len(s), (Counter(s)["C"]+Counter(s)["G"]) / len(s) * 100]
11     dataframe.append(vector)
12 for data_sample in zip(seqs, dataframe):
13     print(data_sample)
```

## F-SCORE AUF MEHR KLASSEN

Die Berechnung des F-Scores ist auch möglich, wenn mehr als zwei Klassen vorhanden sind.

Obacht: Die Aussagekraft wird jedoch schwächer.

### F-Score

$$F(i) = \sum_{c \in \text{class}} \frac{(\bar{x}_i^{(c)} - \bar{x}_i^2)}{\frac{1}{n_c - 1} \sum_{k=1}^{n_c} (x_{k,i}^{(c)} - \bar{x}_i^{(c)})^2}$$