Python für Machine Learning

9 Python für Machine Learning
- Profiling
- Numpy und SciPy
- scikit-learn

Python für Machine Learning

9 Python für Machine Learning
- Profiling
- Numpy und SciPy
- scikit-learn
  - Überblick
  - Maschinelles Lernen
  - scikit-learn: Aufbau und Beispiele

Was?

- *SciKits* („SciPy Toolkits") sind Add-ons für SciPy
- Für uns interessant: `scikit-learn` - Toolkit für maschinelles Lernen, basierend auf SciPy
- Ausführliche Dokumentation: `http://scikit-learn.org/stable/user_guide.html`

Warum?

- Klare, gut dokumentierte API
- Schnell (verwendet NumPy Strukturen, kompilierten Code, Schnittstellen zu C-Bibliotheken für bestimmte Lernverfahren)
- Robust
- Relativ einfach zu bedienen

Python für Machine Learning

9 Python für Machine Learning
- Profiling
- Numpy und SciPy
- scikit-learn
  - Überblick
  - Maschinelles Lernen
  - scikit-learn: Aufbau und Beispiele

Einige Aufgabentypen

- **Klassifikation**
- Clustering

Einige Aufgabentypen

- **Klassifikation**
  Elemente (*items*) sollen anhand ihrer Merkmale (*features*) bestimmten Klassen (*classes*, *labels*) zugeordnet werden. Es ist im Voraus bekannt, welche (und wie viele) Klassen es gibt.

- **Clustering**

Einige Aufgabentypen

- **Klassifikation**
  Elemente (*items*) sollen anhand ihrer Merkmale (*features*) bestimmten Klassen (*classes*, *labels*) zugeordnet werden. Es ist im Voraus bekannt, welche (und wie viele) Klassen es gibt.

- **Clustering**

Einige Aufgabentypen

- **Klassifikation**
  Elemente (*items*) sollen anhand ihrer Merkmale (*features*) bestimmten Klassen (*classes*, *labels*) zugeordnet werden. Es ist im Voraus bekannt, welche (und wie viele) Klassen es gibt.

- **Clustering**
  Elemente sollen anhand ihrer Merkmale gruppiert werden, so dass ähnliche Dinge in derselben Gruppe sind.

- **Structured Prediction**

- Regression

- **Structured Prediction**
  Komplexen Strukturen (z.B. Sätzen) sollen andere komplexe Strukturen (z.B. Parse-Bäume) zugeordnet werden.

- **Regression**

- **Structured Prediction**
  Komplexen Strukturen (z.B. Sätzen) sollen andere komplexe Strukturen (z.B. Parse-Bäume) zugeordnet werden.

- **Regression**

- **Structured Prediction**
  Komplexen Strukturen (z.B. Sätzen) sollen andere komplexe Strukturen (z.B. Parse-Bäume) zugeordnet werden.

- **Regression**
  Für reellwertige Eingaben (z.B. Größe einer Wohnung in Quadratmetern) sollen reellwertige Vorhersagen gemacht werden (z.B. potentieller Kaufpreis).

Was sind die Elemente?

- Elemente sind die zu klassifierenden Objekte.
- Im Prinzip alles möglich.
- Wichtig: Nützliche und generalisierbare Merkmale für die Elemente extrahieren!

Was sind die Elemente?

- Elemente sind die zu klassifierenden Objekte.
- Im Prinzip alles möglich.
- Wichtig: Nützliche und generalisierbare Merkmale für die Elemente extrahieren!

Was sind die Elemente?

- Elemente sind die zu klassifierenden Objekte.
- Im Prinzip alles möglich.
- Wichtig: Nützliche und generalisierbare Merkmale für die Elemente extrahieren!

Was sind die Klassen?

- Elemente werden Klassen zugeordnet.
- Am Ende soll jedes Element zu genau einer Klasse gehören[4]
- Einfachste Klassen: “ja” und “nein” (binäre Klassifikation)

[4] Ausnahme: *multi-label classification* or *multi-class logistic regression*

Was sind die Klassen?

- Elemente werden Klassen zugeordnet.
- Am Ende soll jedes Element zu genau einer Klasse gehören[4]
- Einfachste Klassen: “ja” und “nein” (binäre Klassifikation)

[4] Ausnahme: *multi-label classification* or *multi-class logistic regression*

Was sind die Klassen?

- Elemente werden Klassen zugeordnet.
- Am Ende soll jedes Element zu genau einer Klasse gehören[4]
- Einfachste Klassen: “ja” und “nein” (binäre Klassifikation)

[4] Ausnahme: *multi-label classification* or *multi-class logistic regression*

Elemente und Klassen

Beispiele

Problem	Elemente	Klassen
POS-Tagging	Wörter	POS-Tags
Disambiguierung	Wörter im Kontext	Synsets
Anaphernresolution	Anapher/Antezedenten-Paare	Ja/Nein

Workflow

Gegeben *Trainingsdaten*, also mit Klassen annotierten Elementen, führen wir folgende Arbeitsschritte durch:

- *Feature-Extraktion*: Für jedes Element extrahieren wir Merkmalsausprägungen (*feature values*) aus den Daten
- *Training*: Ein Algorithmus "lernt" anhand der extrahierten Merkmale der annotierten Daten und der annotierten Klassen ein Modell.
- *Testing/Validierung*: Das trainierte Modell wird auf ungesehene, annotierte Beispiele angewendet, um die Performance des Modells zu testen.
- Dann kann das Modell oder die Feature-Extraktion eventuell verbessert werden.

Workflow

Gegeben *Trainingsdaten*, also mit Klassen annotierten Elementen, führen wir folgende Arbeitsschritte durch:

- *Feature-Extraktion*: Für jedes Element extrahieren wir Merkmalsausprägungen (*feature values*) aus den Daten
- *Training*: Ein Algorithmus "lernt" anhand der extrahierten Merkmale der annotierten Daten und der annotierten Klassen ein Modell.
- *Testing/Validierung*: Das trainierte Modell wird auf ungesehene, annotierte Beispiele angewendet, um die Performance des Modells zu testen.
- Dann kann das Modell oder die Feature-Extraktion eventuell verbessert werden.

Workflow

Gegeben *Trainingsdaten*, also mit Klassen annotierten Elementen, führen wir folgende Arbeitsschritte durch:

- *Feature-Extraktion*: Für jedes Element extrahieren wir Merkmalsausprägungen (*feature values*) aus den Daten
- *Training*: Ein Algorithmus "lernt" anhand der extrahierten Merkmale der annotierten Daten und der annotierten Klassen ein Modell.
- *Testing/Validierung*: Das trainierte Modell wird auf ungesehene, annotierte Beispiele angewendet, um die Performance des Modells zu testen.
- Dann kann das Modell oder die Feature-Extraktion eventuell verbessert werden.

Workflow

Gegeben *Trainingsdaten*, also mit Klassen annotierten Elementen, führen wir folgende Arbeitsschritte durch:

- *Feature-Extraktion*: Für jedes Element extrahieren wir Merkmalsausprägungen (*feature values*) aus den Daten
- *Training*: Ein Algorithmus "lernt" anhand der extrahierten Merkmale der annotierten Daten und der annotierten Klassen ein Modell.
- *Testing/Validierung*: Das trainierte Modell wird auf ungesehene, annotierte Beispiele angewendet, um die Performance des Modells zu testen.
- Dann kann das Modell oder die Feature-Extraktion eventuell verbessert werden.

Workflow

Gegeben *Trainingsdaten*, also mit Klassen annotierten Elementen, führen wir folgende Arbeitsschritte durch:

- *Feature-Extraktion*: Für jedes Element extrahieren wir Merkmalsausprägungen (*feature values*) aus den Daten
- *Training*: Ein Algorithmus "lernt" anhand der extrahierten Merkmale der annotierten Daten und der annotierten Klassen ein Modell.
- *Testing/Validierung*: Das trainierte Modell wird auf ungesehene, annotierte Beispiele angewendet, um die Performance des Modells zu testen.
- Dann kann das Modell oder die Feature-Extraktion eventuell verbessert werden.

Feature-Extraktion

Wichtiger Teil der Arbeit:

- Entscheiden, welche Merkmale verwendet werden sollen
- Code schreiben, der die Merkmale aus den Daten extrahiert

Trainings- und Testdaten

- Für faire und nützliche Ergebnisse darf nicht auf denselben Daten trainiert und getestet werden!
- Prozentuale Aufteilung: 30% der Daten werden zufällig ausgewählt und als Testdaten beiseite gelassen (während des Trainings)
- Cross Validation: Mehrere Trainingsläufe, wobei jeweils ein anderer Teil der Daten als Testset ausgelassen wird. Besonders sinnvoll bei wenig Daten.

Python für Machine Learning

9 Python für Machine Learning
- Profiling
- Numpy und SciPy
- scikit-learn
  - Überblick
  - Maschinelles Lernen
  - scikit-learn: Aufbau und Beispiele

scikit-learn

scikit-learn bietet...

- Überwachte Lernalgorithmen
- Unüberwachte Lernalgorithmen
- Module zur Vektorisierung und Filterung von Daten
- Module zur Validierung und Evaluierung

Verwendung

```
>>> import sklearn
>>> from sklearn import ...
>>> help(sklearn) # Überblick über Pakete
```

Wichtigste Funktionen von Lernalgorithmen

`fit(X[,y])` Lerne von Daten

`predict(X)` Wende Modell auf (Test-)daten an

`transform(X)` Transformiere Daten (z.B. Feature Selection)

Die Funktionen können auch kombiniert werden (z.B. `fit_predict()` oder `fit_transform()`).

Überwachte Methoden I

- Gegeben Beobachtungen X und Klassen Y
- Lerne $f : X \to Y$ so dass eine Zielfunktion minimiert (oder maximiert) wird (`fit()`-Methode)
- Bestimme $y = f(x)$ für ungesehene x (`predict()`-Methode)

`http://scikit-learn.org/stable/supervised_learning.html`

Überwachte Methoden II

Zwei Kategorien

Klassifikation Y besteht aus endlich vielen Elementen (z.B. $Y = \{0, 1\}$)
`sklearn.linear_model.LogisticRegression`,
`sklearn.tree.DecisionTreeClassifier`, ...

Regression Y ist stetig (z.B. $Y = \mathbb{R}$ oder $Y = [0, 1]$)
`sklearn.linear_model.LinearRegression`,
`sklearn.svm.SVR`, ...

Überwachte Methoden: Beispiel

```
>>> from sklearn import linear_model
>>> import numpy
>>> model = linear_model.LogisticRegression()
# erstelle Beispieldaten
>>> X = numpy.array([[0, 1, 1],
...      [1, 0, 1]])
>>> y = [1, 0]
>>> model.fit(X, y)
>>> model.predict(X)
[1, 0]
```

Unüberwachte Methoden I

- Hier sind nur Beobachtungen X gegeben
- Diese sollen z.B. gruppiert werden

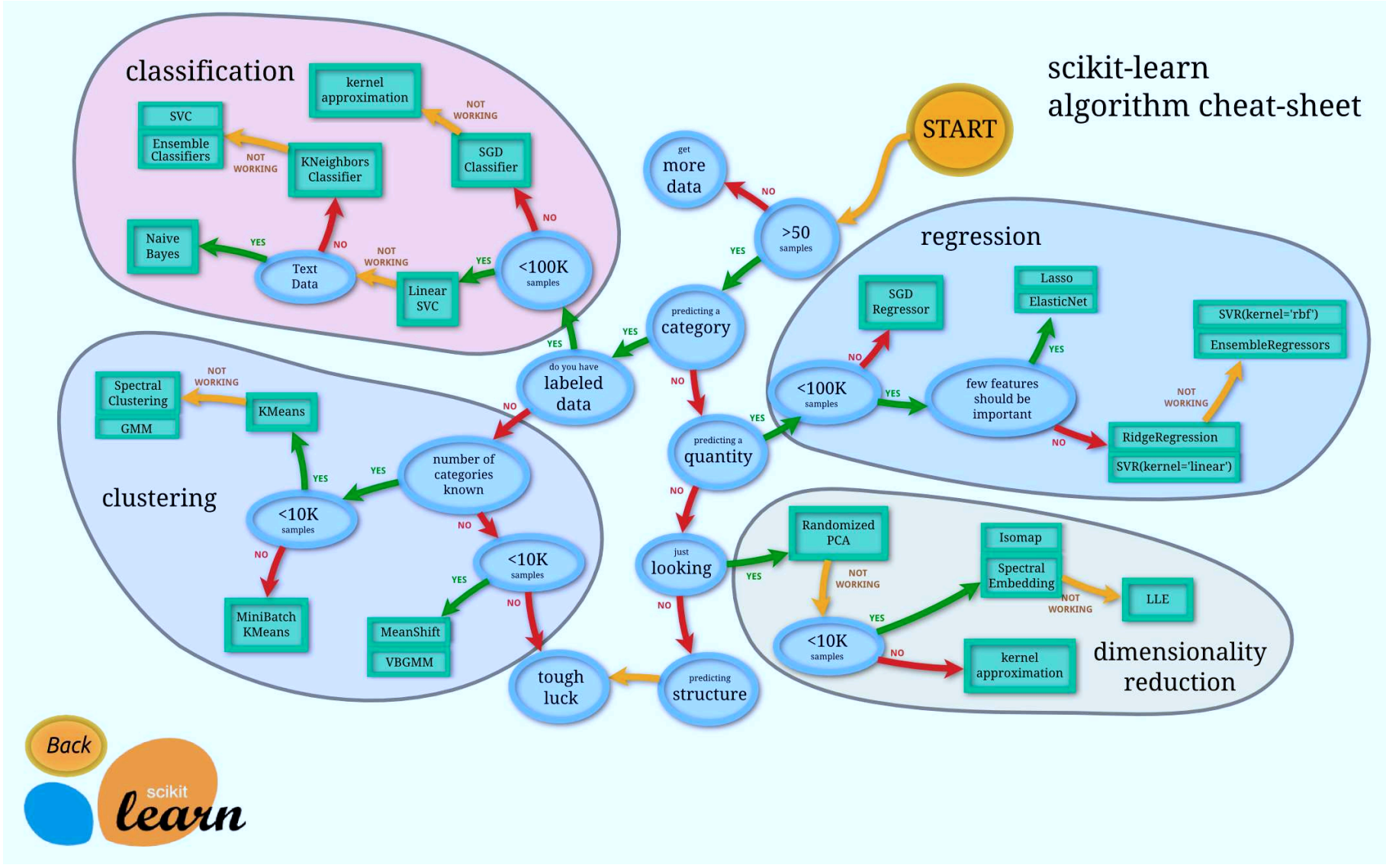
Unüberwachte Methoden II

Einige unüberwachte Lernverfahren

Clustering Daten werden in *Cluster* gruppiert
`sklearn.cluster`

Matrixzerlegung Oft: „Wichtigste" Dimensionen einer Beobachtungsmatrix finden
`sklearn.decomposition`

Welches Lernverfahren wann?



Datenvorbereitung

Preprocessing Normalisierung, Skalierung, kategoriale Features
`sklearn.preprocessing`

Feature-Extraktion Features in korrektes Format für Learner bringen, einfache Text- und Bildfeatures extrahieren
`sklearn.feature_extraction`

`http://scikit-learn.org/stable/modules/preprocessing.html`
`http://scikit-learn.org/stable/modules/feature_extraction.html`

Datenvorbereitung: Beispiel

- `feature_extraction.text` enthält Tools für Umgang mit Textdaten

```
>>> from sklearn.feature_extraction  import text
>>> countvec = text.CountVectorizer()
>>> text = ['Erster Satz', 'Zweiter Satz']
# 'lernt' das Vokabular
>>> countvec.fit(text)
>>> countvec.get_feature_names()
[u'erster', u'satz', u'zweiter']
# 'lernt' das Vokabular und
# generiert Dokument-Term-Matrix
>>> vectors = countvec.fit_transform(text)
# (sparse) Dokument-Term-Matrix
>>> print vectors
  (0, 0)    1
  (0, 1)    1
  (1, 1)    1
  (1, 2)    1
```

Evaluierung

Cross Validation Train/Test Split, Cross Validation (`sklearn.cross_validation`)

Grid Search Hyperparameter einstellen (`sklearn.grid_search`)

Evaluationsmetriken Precision, Recall, F1-Score (`sklearn.metrics`)

`http://scikit-learn.org/stable/model_selection.html`

Evaluierung: Beispiel

```
>>> predicted =  model.predict(X)
>>> sklearn.metrics.accuracy_score(y, predicted)
1.0
```

Modell speichern

- trainiertes Modell (z.B. SVM-Objekt) für spätere Anwendung speichern
- Lösung: Serialisierung mit `pickle` (Python-Modul zur Objekt-Serialisierung)

Übung 15