Prof. Dr. Agnès Voisard
Nicolas Lehmann

Datenbanksysteme, SoSe 18

Übung 04

TutorIn: Toni Draßdo
Tutorium 014

Eduard Beiline, Mark Niehues, Antoen Oehler

21. Mai 2018

Task 1: ER-Modellierung

Task 2: Relationales Modell

Task 3: Reverse Engineering

Task 4: Data Mining

1 - K-Means

Der Log des K-Mean Aufrufs ist in Listing 1 angegeben. Der dazugehörige entwickelte Code in Listing 2.

Listing 1: Log File des K-Means Algorithmus

```
:::::Centers and their Clusters at Step 1
Center for this Cluster: [10 10  3]
Contains:
[10 10  3]
[7 9 1]

Center for this Cluster: [9 2 3]
Contains:
[9 2 3]
[9 4 1]
[6 6 8]
[4 3 5]

Center for this Cluster: [ 3 10  1]
Contains:
[ 3 10  1]
[4 5 1]
```

```
:::::Centers and their Clusters at Step 2
Center for this Cluster: [ 8.5  9.5  2. ]
Contains:
[10 10  3]
[7 9 1]

Center for this Cluster: [ 7.    3.75  4.25]
Contains:
[9 2 3]
[9 4 1]
[6 6 8]
[4 3 5]

Center for this Cluster: [ 3.5  7.5  1. ]
Contains:
[ 3 10  1]
[4 5 1]


:::::Final Centers and their Clusters:
Center for this Cluster: [ 8.5  9.5  2. ]
Contains:
[10 10  3]
[7 9 1]

Center for this Cluster: [ 7.    3.75  4.25]
Contains:
[9 2 3]
[9 4 1]
[6 6 8]
[4 3 5]

Center for this Cluster: [ 3.5  7.5  1. ]
Contains:
[ 3 10  1]
[4 5 1]
```

Listing 2: K-Means Implementierung

```python
#!/usr/bin/env python3
# coding: utf-8
import numpy as np
import sys

DATA = np.array([
    [3,9,9,10,6,7,4,4],
    [10,2,4,10,6,9,5,3],
    [1,3,1,3,8,1,1,5]
]).T


def expectation(data, centers):
    """
    Assigns datapoints to centers
    """
    clusters = [[] for _ in range(centers.shape[0])]

    for point in data:
            # Calculate distances from centers
            d = np.linalg.norm(centers - point, axis=1)

            # Find the index according to the lowest distance
            id_min = np.argmin(d)

            # Assing point to minimum distance
            clusters[id_min].append(point)

    for clt in clusters:
```

```python
        if len(clt) == 0:
            print("Error: Empty cluster occured, please re-run the program.")
            sys.exit(1)
    return clusters


def minimization(clusters):
    """
    Computes new cluster means
    """
    centers = [np.mean(cls, axis=0) for cls in clusters]

    return np.vstack(centers)


def k_means(data, k, sigma):
    # Initialize with k random points
    centers = data[np.random.randint(data.shape[0], size=k)]

    dist = sigma + 1
    i = 0
    while dist > sigma:
        i+=1
        # Assign data points to centers
        clusters = expectation(data, centers)
        print(":::::Centers and their Clusters at Step " + str(i))
        print_clusters(centers, clusters)
        print()
        # Calculate new centers
        new_centers = minimization(clusters)

        # Calc maximum center movement
        dist = max(np.linalg.norm(centers - new_centers, axis=1))
        centers = new_centers

    return centers, clusters


def print_clusters(centers, clusters):
    for center, clst in zip(centers, clusters):
        print("Center for this Cluster: {}".format(center))
        print("Contains:")
        for _ in clst:
            print(_)
        print()


centers, clusters = k_means(DATA, k=3, sigma=3/4)
print(":::::Final Centers and their Clusters:")
print_clusters(centers, clusters)
```

2 - Naive Bayes

1 - Wahrscheinlichkeit einer Grippe bei laufender Nase

Naive Bayes Formula:

$$P(C|x) = \frac{P(C)\,P(x|C)}{P(x)} \tag{1}$$

Aus Formel 1 folgt für die Wahrscheinlichkeit an einer Grippe zu leiden, bei laufender Nase:

$$P(Grippe|Nase) = \frac{P(Gripp)\,P(Nase|Grippe)}{P(Nase)} \tag{2}$$

wobei:

$$\begin{aligned}
&P(Nase) = 4/8 = 1/2\\
&P(Grippe) = 1/2\\
&P(Nase|Grippe) = 3/4
\end{aligned}$$

Durch einsetzen in Formel 2 erhält man:

$$P(Grippe|Nase) = 3/4$$

2 - Grippe, wenn X

Um die Frage zu beantwortet, ob jemand eher Grippe oder keine Grippe besitzt wird, um die Rechnung zu Vereinfachen der Quotient aus $P(Grippe|x)$ und $P(\neg Grippe|x)$ gebildet, dadurch kürzt sich die aufwändig zu berechnende Evidenz $P(x)$ heraus:

$$Q = \frac{P(Grippe|x)}{P(\neg Grippe|x)} = \frac{P(x|Grippe)\,P(Grippe)}{P(x|\neg Grippe)\,P(\neg Grippe)} \tag{3}$$

wobei:

$$\begin{aligned}
&x = \{Schttelfrost, schwacheKopfschmerzen, Fieber\}\\
&P(Schttelfrost|Grippe) = 3/4\\
&P(Schttelfrost|\neg Grippe) = 1/2\\
&P(schwacheKopfschmerzen|Grippe) = 1/4\\
&P(schwacheKopfschmerzen|\neg Grippe) = 1/4\\
&P(Fieber|Grippe) = 1/2\\
&P(Fieber|\neg Grippe) = 1/2\\
&P(Grippe) = P(\neg Grippe) = 1/2
\end{aligned}$$

Unter Annahme der (hinreichenden) Unabhängigkeit der Variablen, gilt $P(x_1, x_2|C) = P(x_1|C)\,P(x_2|C)$.

Daraus ergibt sich schließlich:

$$Q = \frac{\frac{3}{4}\frac{1}{4}\frac{1}{2}}{\frac{1}{2}\frac{1}{4}\frac{1}{2}} = 3/2$$

Aus der $Q > 1$ folgt, dass der Patient wahrscheinlicher Grippe hat als keine.

3 - Apriori

Die Supports werden anschließend wiederum kombiniert um Beziehungen der Form $\{A, B\} \rightarrow \{C\}$ zu bewerten. Die sogenannte *confidence* berechnet sich aus den Support als:

$$conf(X \rightarrow Y) = \frac{sup(X \cup Y)}{sup(X)} \tag{4}$$

Das Ergebnis der Berechnungen sei an dieser Stelle der Einfachheit halber als Textdatei angegeben:

Tabelle 1: Ausschnitt aus der Berechnung der Supports durch Kombinatorik und Abzählen

C_0	L_0	C_1	L_1	C_2
$\sup(\{A\}) = \frac{1}{2}$	{A}	$\sup(\{A,B\}) = \frac{1}{2}$	{A,B}	...
$\sup(\{B\}) = \frac{5}{6}$	{B}	$\sup(\{A,C\}) = \frac{1}{3}$	{A,E}	...
$\sup(\{C\}) = \frac{1}{2}$	{C}	$\sup(\{A,D\}) = \frac{1}{3}$	{B,C}	...
$\sup(\{D\}) = \frac{5}{6}$	{D}	$\sup(\{A,E\}) = \frac{2}{3}$	{B,D}	...
$\sup(\{E\}) = \frac{5}{6}$	{E}	$\sup(\{A,F\}) = \frac{1}{6}$	{B,E}	...
$\sup(\{F\}) = \frac{1}{2}$	{F}	$\sup(\{B,C\}) = \frac{1}{2}$	{C,D}	...
		$\sup(\{B,D\}) = \frac{2}{3}$	{D,E}	...
		$\sup(\{B,E\}) = \frac{2}{3}$	{D,F}	...
		$\sup(\{B,F\}) = \frac{1}{3}$	{E,F}	...
		$\sup(\{C,D\}) = \frac{1}{2}$		...
		$\sup(\{C,E\}) = \frac{1}{3}$		...
		$\sup(\{C,F\}) = \frac{1}{6}$		...
		$\sup(\{D,E\}) = \frac{2}{3}$		...
		$\sup(\{D,F\}) = \frac{1}{2}$		...
		$\sup(\{E,F\}) = \frac{1}{2}$		...

Listing 3: Ergebnis des Apriori Algorithmus.

```
Supports:
---------
('F') , 0.500
('A') , 0.500
('C') , 0.500
('D') , 0.833
('B') , 0.833
('E') , 0.833
('D', 'F') , 0.500
('A', 'E') , 0.500
('C', 'B') , 0.500
('C', 'D') , 0.500
('A', 'B') , 0.500
('E', 'F') , 0.500
('B', 'D') , 0.667
('B', 'E') , 0.667
('E', 'D') , 0.667
('C', 'B', 'D') , 0.500
('E', 'D', 'F') , 0.500
('A', 'B', 'E') , 0.500
('B', 'E', 'D') , 0.500



Regeln:
------
('B', 'D') ==> ('C') , 0.750
('E', 'D') ==> ('F') , 0.750
('B', 'E') ==> ('A') , 0.750
('B', 'E') ==> ('D') , 0.750
('B', 'D') ==> ('E') , 0.750
('E', 'D') ==> ('B') , 0.750
```

```
('B') ==> ('D') , 0.800
('D') ==> ('B') , 0.800
('B') ==> ('E') , 0.800
('E') ==> ('B') , 0.800
('E') ==> ('D') , 0.800
('D') ==> ('E') , 0.800
('F') ==> ('D') , 1.000
('A') ==> ('E') , 1.000
('C') ==> ('B') , 1.000
('C') ==> ('D') , 1.000
('A') ==> ('B') , 1.000
('F') ==> ('E') , 1.000
('C') ==> ('B', 'D') , 1.000
('C', 'B') ==> ('D') , 1.000
('C', 'D') ==> ('B') , 1.000
('F') ==> ('E', 'D') , 1.000
('E', 'F') ==> ('D') , 1.000
('D', 'F') ==> ('E') , 1.000
('A') ==> ('B', 'E') , 1.000
('A', 'B') ==> ('E') , 1.000
('A', 'E') ==> ('B') , 1.000
```

4 - Lineare Regression

Leider ist bei uns erst eine Korrektur eingetragen, daher nehmen wir für diese Aufgabe folgende Noten an:

Tabelle 2: Bisherige Notenverteilung

x	0	1	2
y	0,89	0,92	0,93

Aus Tabelle 2 folgt: $\bar{x} = 1$ und $\bar{y} = 0.91\bar{3}$.

Die lineare Regression beschreibt die Daten mit einer Funktion der Art:

$$f(x) = \beta_0 + \beta_1 \, x + \epsilon \tag{5}$$

mit

$$\beta_1 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})\,(y - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2} \tag{6}$$

$$\beta_0 = \bar{y} - \beta_1 \, \bar{x} \tag{7}$$

$$\tag{8}$$

Aus Gleichung 6 folgt $\beta_1 = 0,02$ und damit $\beta_0 = 0.893$. Aus der linearen Gleichung 5 ergibt sich also für den nächsten Zettel $f(3) = 0,953$.