

kMeans Clustering

Rohdaten von Sensoren können in den seltensten Fällen ohne Vorverarbeitung für maschinelles Lernen verwendet werden. Zunächst werden die Rohdaten der Sensoren eingelesen und vorverarbeitet. Mithilfe der so generierten Arbeitsdaten wird das Clustering mit dem *k*Means Algorithmus in einem Batch-Lerning Verfahren durchgeführt.

Aufgabe 1: Daten laden - Koordinaten

Zur späteren Visualisierung wird die Datei `coordinates.csv` mit den Positionsdaten mithilfe der Funktion `load` geladen in die Matrix `Coord` geladen. Zur vereinfachten nachfolgenden Bearbeitung werden die Daten transponiert.

a) Löschen Sie zunächst alle bestehenden Variablen aus dem Workspace

```
% lösche workspace  
clear all;
```

a) Laden Sie die Datei `coordinates.csv` mit den Positionsdaten mithilfe der Funktion `load` in die Matrix `Coord`.

```
% lese Koordinaten aus Datei
```

b) Transponieren Sie `Coord`.

```
% transponiere
```

Aufgabe 2: Daten laden - Rohdaten Schallmessung

Im nächsten Schritt werden alle Dateien aus dem Ordner `data` in die Matrix `RawData` geladen. Bei der Visualisierung der Daten fällt auf, dass der Beginn des auszuwertenden Signales teilweise stark variiert oder Störungen aufgezeichnet werden (Beginn des 3. Plots).

a) Setzen Sie eine Variable `pathToData` auf den Unterordner `data\`.

```
% Pfad zu den Rohdaten
```

b) Lesen Sie die Dateinamen mithilfe der Funktion `dir` in die Variable `filenames` ein.

```
% alle Dateien im Verzeichnis erfassen  
  
% Dateinamen speichern
```

c) Initialisieren Sie die Matrix RawData mit der Größe 10000 (Anzahl Messungen pro Datei) x Anzahl Dateien.

```
% Anzahl Dateien  
  
% Länge der Rohdaten  
  
% Initialisierung der Matrix RawData
```

d) Laden Sie die Inhalte der Dateien mithilfe einer Schleife in die Matrix RawData, so dass eine Spalte der Matrix die Messungen einer Datei enthält.

```
% Schiebe Rohdaten in Matrix  
% für jede Datei  
{  
  for i =  
    % benutze Pfad der aktuell bearbeiteten Datei  
    fullpath = strcat(?,?);  
    % lade Rohdaten dieser Datei  
    RawData(?:?) = load(?);  
  end  
}
```

e) Plotten Sie die Datensätze 1, 5 und 26 mit der Funktion plot in separaten Plots und vergeben Sie jeder Grafik die Nummer als Titel.

```
% plote verschiedene Schallsignale  
%plot(RawData(:,:))  
%title(['Rohdatensatz ?'])
```

Aufgabe 3: Datenvorverarbeitung

Die erfassten Rohdaten werden nun vorverarbeitet, um die nachfolgende Auswertung mithilfe maschineller Lernalgorithmen zu ermöglichen. Dazu wird von den Rohdaten zunächst der Mittelwert subtrahiert, um die Auswirkungen des Rauschens zu minimieren. Dann werden fehlerbehaftete Datensätze bereinigt und es wird eine Synchronisierung der Messungen durchgeführt. Abschließend werden die Arbeitsdaten in einer Datei gespeichert.

a) Speichern Sie den Mittelwert aller Spalten von RawData mithilfe der Funktion mean in einem Array.

```
% Mittelwerte berechnen
```

b) Verringern Sie alle Messwerte um die jeweiligen Mittelwerte

```
% Rohdaten um Mittelwerte anpassen
```

c) Setzen Sie bei den Datensätzen 5, 26, 34, 182 und 224 den Wert der ersten 400 Stellen auf 0.

```
% Störungen beseitigen
```

d) Initialisieren Sie die Matrix Data mit 2000 Zeilen.

```
% Initialisierung Matrix Data
```

e) Kopieren Sie die 2000 Messwerte nach dem ersten Überschreiten des Schwellwerts 50 aus RawData nach Data.

```
% Festlegung Schwellwert

% Initialisierung Index

% Für jede Datei
%{
for j = 1:?
    % prüfe wo der Schwellwert zuerst überschritten wird (Betrag)
    for i = 1:?
        % wenn 50 überschritten wird speichere den index

        end
        % Synchronisiere Daten in Matrix Data
        for h = 1:?
            Data(?:?) = RawData((?+?),?);
        end
        ? = ?;
    end
%}
```

f) Speichern Sie die Matrix Data mithilfe der Funktion save in der Datei *Data.mat*

```
% Speichern der Arbeitsdaten
```

g) Visualisieren und betiteln Sie die Arbeitsdatensätze 1, 5 und 26, beschriften Sie die Achsen mit *Messwert* und *Zeit*.

```
% plotte Arbeitsdaten
```

Aufgabe 5: Clustering der Daten und Visualisierung

Aus den so vorbereiteten Daten können nun mithilfe der Matlab-Toolbox Funktion kmeans Gruppen gebildet werden.

a) Rufen Sie kmeans mit Clusterzahl 3, 4, 5 und 6 auf die Arbeitsdaten auf. Nutzen Sie die Cosinus-Distanz.

b) Lassen Sie sich das Ergebnis mithilfe der Funktion plotPoint ausgeben.

```
% Gruppenbildung und Anzeige der Ergebnisse
%{
for cluster = ? : ? : ?
    % K-Means Clustering
    [idx,C] =

    % Ergebnis plotten
    figure;
    title(['kMeans mit ' ? ' Clustern'])
    hold on
    for i = 1:?
        plotPoint(?,?,?)
    end
end
%}
```

c) Finden Sie den Index des fehlerhaften Datensatz und beheben Sie den Fehler!

Hilfsfunktionen zur Visualisierung

```
function plotPoint(y, s, Coord)
    % input:
    % y = number of point
    % s = predicted class
    % Coord = x and y coordinate for point to plot
    % output:
    % point in plot
    color = getColor(s);
    txt1 = [num2str(y)];
    text((Coord(1,1)+1),(Coord(2,1)+1),txt1)
    plot(Coord(1,1),Coord(2,1),'*', 'Color', color);
    plot(Coord(1,1),Coord(2,1),'ko', 'LineWidth', 4, 'MarkerSize', 12, 'Color', color);
end

% return color code for integer
% input:
% colorCode
% output:
% matlab color shortcut
function [color] = getColor(colorCode)
    colorCode = mod(colorCode,10);
    switch colorCode
    case 1
        color = 'r';
    case 2
        color = 'b';
```

```
case 3
    color = [1 0.4 0.6];
case 4
    color = 'y';
case 5
    color = 'm';
case 6
    color = 'c';
case 7
    color = 'k';
case 8
    color = 'g';
otherwise
    color = 'm';
end
end
```