Übung zur Vorlesung

Maschinelles Lernen 1

Wintersemester 2010/2011

Abteilung Maschinelles Lernen
Institut für Softwaretechnik und theoretische Informatik
Fakultät IV, Technische Universität Berlin
Prof. Dr. Klaus-Robert Müller
Email: klaus-robert.mueller@tu-berlin.de

Blatt 4

Abgabe bis Mittwoch, den 24. November 2010 bis 13 Uhr bei Konrad Rieck (FR 6061) oder in sein Fach in FR6052. Praktische Aufgaben zusätzlich an irene.winkler@tu-berlin.de mailen.

Aufgaben (K-Means Clustering)

1. Der k-means Algorithmus findet lokale Minima der Kostenfunktion

$$\mathcal{C}(C_1, \ldots, C_k) = \sum_{i=1}^{k} \frac{1}{\#C_i} \sum_{j \in C_i} \|x_j - m_i\|^2,$$

wobei C_i die Indizes von Cluster i sind, $\#C_i$ die Anzahl der Punkte in C_i ist, und $m_i = \frac{1}{\#C_i} \sum_{j \in C_i} x_j$. Mit anderen Worten: k-means Clustering findet k Clusterzentren m_i, so dass der quadratische Fehler zu diesen minimiert wird.

Leite eine alternative Form der Kostenfunktion her, in der die Mittelpunkte nicht mehr explizit vorkommen:

$$\mathcal{C}(C_1, \ldots, C_k) = \sum_{i=1}^{k} \frac{1}{2(\#C_i)^2} \sum_{j,j' \in C_i} \|x_j - x_{j'}\|^2.$$

Hinweise:

- Verwende, dass $\|x - y\|^2 = \langle x, x \rangle - 2\langle x, y \rangle + \langle y, y \rangle$.
- Betrachte die jeweiligen Summen für festes i. Schreibe $n = \#C_i$ und $m = m_i$ und lasse das $\in C_i$ in den Summen weg, um die Notation zu vereinfachen. Zeigen dann, dass

$$\frac{1}{n} \sum_j \|x_j - m\|^2 = \frac{1}{n} \sum_j \langle x_j, x_j \rangle - \frac{1}{n^2} \sum_{j,j'} \langle x_j, x_{j'} \rangle = \frac{1}{2n^2} \sum_{j,j'} \|x_j - x_{j'}\|^2,$$

  wobei Du jeweils an den Endpunkten anfängst.

Der Vorteil der alternativen Variante ist, dass man den quadratischen Abstand $\|x_j - x_{j'}\|^2$ auch durch ein allgemeineres Unähnlichkeitsmaß $s(x_j, x_{j'})$ ersetzen kann. Dieser Fall tritt zum Beispiel ein, wenn die Punkt x_j nicht bekannt sind, jedoch die Unähnlichkeiten s experimentell ermittelt wurden. Diese Methode wird auch "pair-wise clustering" genannt.

2. Implementiere k-means clustering. Ergänze dafür die folgenden beiden Funktionen im Programmskelett `sheet04.m`:

- `generate_data`: Erzeuge normalverteilte Cluster mit den gegebenen Zentren. Übergeben wird eine $k \times 2$ Matrix, die die Clusterzentren in jeder Zeile enthält. Erzeuge von jedem Cluster n Punkte.
- `k_means_clustering`: Implementere den k-means-clustering-Algorithmus.
  - Zur Initialisierung der Clusterzentren wähle fünf zufällige Datenpunkte (siehe `randperm`).
  - Iteriere solange der Unterschied zwischen alten und neuen Clusterzentren größer als 10^{-10} ist. Das heisst, wenn M alle Clusterzentren enthält, dann soll `norm(..., 'fro')` größer als 10^{-10} sein.
  - Verwende die Funktion `pwdist` um alle Distanzen zwischen den Clusterzentren und den Datenpunkten schnell auszurechnen.

Siehe unten für Pseudocode. Auch hier gilt wieder, dass möglichst die Verwendung von for-Schleifen zu vermeiden ist.

Algorithm 1 K-means clustering

Require: data points $x_1, \ldots, x_n \in \boldsymbol{R}^d$, number of clusters k.
Ensure: cluster centres $\mu_1, \ldots, \mu_k \in \boldsymbol{R}^d$, assignment vector $r \in \boldsymbol{R}^n$

Choose random data points as initial cluster centres $\mu_1 \leftarrow x_{i_1}, \ldots, \mu_k \leftarrow x_{i_k}$ where $i_j \neq i_l$ for all $j \neq l$.
$c \leftarrow 1$
$i \leftarrow 0$
while $c > 10^{-10}$ **do**
  $d_{ij} \leftarrow \|x_i - \mu_j\|^2$ for all $i = 1, \ldots, n$, $j = 1, \ldots, k$.
  $c_i \leftarrow$ closest cluster center (based on the minimum value in each row of d)
  $\mu'_j \leftarrow \mu_j$ for all $j = 1, \ldots, k$.
  for $j \leftarrow 1$ to k **do**
    Compute new cluster center $\mu_j \leftarrow \frac{1}{|\{l : r'_l = j\}|} \sum_{l : r'_l = j} x_l$
  end for
  $c \leftarrow \sqrt{\sum_{j=1}^{k} \|\mu'_j - \mu_j\|^2}$ (Frobenius-norm)
  $i \leftarrow i + 1$
end while

```
function sheet04
% Generate data
centers = [0, 0; 7, 3; -2, 4; 0, 10; -5, -5];

[X, Y] = generate_data(centers, 50);

% plot the result
figure(1)
gscatter(X(:,1), X(:,2), Y);

% cluster the data set with different choices of K
figure(2)
[Y2, M2] = k_means_clustering(2, X);
plot_clustering(X, Y2, M2)

figure(3)
[Y5, M5] = k_means_clustering(5, X);
plot_clustering(X, Y5, M5)

figure(4)
[Y8, M8] = k_means_clustering(8, X);
plot_clustering(X, Y8, M8)


% 1. Generate data with normally distributed clusters
% with centers given by the rows of C. Generate N
% points from each cluster, and also return a vector K
% which contains the cluster indices for each point.
function [X, K] = generate_data(C, N)
% ...

% 2. Compute K-means clustering. Randomly select K points
% as initial means. Iterate while the difference
% between the old means and the new means matrix in the
% Frobenius norm (norm(..., 'fro')) is larger than 1e-10.
%
% Return the cluster indices and the matrix of means (means
% are rows).
function [Y, MEANS] = k_means_clustering(K, X)
% ...

% Plot the clustering and the centers.
```

```
function plot_clustering(X, Y, MEANS)
gscatter(X(:, 1), X(:, 2), Y);
hold on;
plot(MEANS(:, 1), MEANS(:, 2), '+', 'MarkerSize', 10, 'LineWidth', 3);
hold off;

% Compute all pairwise distances quickly.
function D = pwdist(X, Y)
D = size(X, 2);
N = size(X, 1);
M = size(Y, 1);

XX = sum(X.*X, 2);
YY = sum(Y.*Y, 2);
D = repmat(XX, 1, M) + repmat(YY', N, 1) - 2*X*Y';
```
