Übung zur Vorlesung

Maschinelles Lernen 1

Wintersemester 2009/2010

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 7

Abgabe bis Abgabe bis **Mittoch, den 2. Dezember 2009** bei Mikio Braun (FR 6058, notfalls unter der Türe durchschieben) und per Email an mikio@cs.tu-berlin.de.

Nächste Nachbarregel

In der Vorlesung wurde die k-Nächste-Nachbarregel (kNN) vorgestellt. Sie besteht darin, gegeben einen Trainingssatz $X_1, \ldots, X_n \in \mathbb{R}^d$ und $Y_1, \ldots, Y_n \in \{\pm 1\}$, für einen neuen Punkt die Vorhersage zu treffen, die der Mehrzahl der k-nächsten Nachbarn entspricht.

Auf diesem Aufgabenblatt wird ein Datensatz bestehend aus zwei überlappenden Gaussverteilungen, die jedoch nicht identische Kovarianzmatrizen haben, verwendet. Wir wissen, dass die optimale Entscheidungsgrenz in diesem Fall eine Parabel ist.

Das Skript berechnet die kNN für den Trainingsdatensatz und einen Testdatensatz und plottet die Daten sowie die Entscheidungsgrenze und zeigt den Tranings- und Testfehler.

1. Schreibe die Funktion `knn`, die die kNN-Vorhersagen berechnet. **(10 Punkte)**
2. Schreibe die Funktion `loss`, die den 0-1-Fehler berechnet:

$$\ell(Y, \hat{Y}) = \frac{1}{n} \sum_{i=1}^{n} 1_{\{Y_i \neq \hat{Y}_i\}}$$

   (5 Punkte)
3. Schreibe die Funktion `plotgrid`, die die kNN-Vorhersagen auf einem Gitter berechnet und mit Hilfe von `contour` die Entscheidungsgrenze anzeigt (als Höhenlinie `[0, 0]`, verwende auch noch `meshgrid` um die Punkte auf dem Gitter zu erzeugen). **(10 Punkte)**
4. Führe das Programm für $k = 1, 3, 5, 7, 9, 11, 13, 15, 17, 19$ aus und notiere Trainings- und Testfehler. Ab wann entspricht die Entscheidungsgrenze ungefähr der theoretisch korrekten? **(5 Punkte)**

```
function sheet07

N = 300; % size of training data set
K = 5;   % number of neighbors

% generate a training and test data set
[X, Y] = generate_data(N);
[XE, YE] = generate_data(1000);

% plot training data set
IP = find(Y == 1);
IN = find(Y == -1);
plot(X(IP, 1), X(IP, 2), 'r+', X(IN, 1), X(IN, 2), 'bo');

% predict on training set
Yh = knn(K, X, Y, X);

% predict on test set
YEh = knn(K, X, Y, XE);

% Plot predictions and training and test error in the title.
hold on;
```

```
plotgrid(K, X, Y, -6, 6, -6, 6);
hold off;
colormap([0 0 0])
title(sprintf('training error = %.2f%%, test error = %.2f%%', ...
              loss(Y, Yh), loss(YE, YEh)));

function [X, Y] = generate_data(N)
X = [2*randn(N, 2) + repmat([1, 2], N, 1); ...
     0.5*randn(N, 2) + repmat([-2, 1], N, 1)];
Y = [ones(N, 1); -ones(N, 1)];

% 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';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Your Solutions Below

% 1. knn - implement k-nearest neighbor prediction
%
% inputs:
%   K - number of neighbors
%   X - [n, d]-matrix with n training data points of dimension d (in the rows!)
%   Y - [n, 1]-matrix of +1, -1 training labels
%   XE - [m, d]-matrix with test labels, for which predictions should
%        be computed
%
% output:
%   Yh - [m, 1]-matrix of predicted labels (real number, sign
%        indicates class.
%
% Note: one for-loop is allowed.
function Yh = knn(K, X, Y, XE)
% ...

% 2. loss - compute the 0-1 loss. Sign of entries of Y and Yh indicate
% class.
function E = loss(Y, Yh)
% ...

% 3. plotgrid - plot knn predictions on a grid
%
% inputs:
%   K, X, Y - as inputs to knn
%   XMIN, XMAX - minimal and maximal value of X
%   YMIN, YMAX - minimal and maximal value of Y
%
% Hint: use meshgrid, contour
function plotgrid(K, X, Y, XMIN, XMAX, YMIN, YMAX)
% ...
```

Für Fragen zum Übungsblatte bitte in der Google Group http://groups.google.com/group/mikiobraun-lehre registrieren und die Frage an die Mailingliste stellen.