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 7

Abgabe bis Mittwoch, den 15. Dezember 2010 um 12:00

Die Lösungen bitte per Email an **irene.winkler@tu-berlin.de** senden
und im **Postfach** von Dr. Konrad Rieck abgeben.

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.

In dieser Aufgabe 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. **(5 Punkte)**
2. Schreibe die Funktion `loss`, die den 0-1-Fehler berechnet:

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

   (2 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). **(5 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? **(3 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)
% ...
```

Lineare Regression

1. Let $\{(x_i, y_i)\}_{i=1}^n \subset \mathbb{R}^2$ be a sample where the mean of the *target* value y_i is a linear function of the *data* x_i, i.e.

$$y_i = \alpha x_i + \beta + \epsilon_i,$$

where the noise term ϵ_i follows a Gaussian distribution, $\epsilon_i \sim \mathcal{N}(0, \sigma)$. Show that the Maximum Likelihood estimator for the bias term β is

$$\hat{\beta}_{\mathrm{ML}} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \alpha x_i).$$

(5 Punkte)

2. Implement multiviariate Ordinary Least Squares (OLS) without constant term in a Matlab function

```
alpha = ols(X,y)
```

where `X` is a $d \times n$ matrix of data in column vectors and `y` is a $1 \times n$ vector of target values. The result `alpha` is a $1 \times d$ vector of coefficients for the regression model. Thus, for a data point `x` the prediction is given by `alpha*x`.

In this problem, you should investigate the influence of correlated data on the variance of the parameter estimates. The toy data $\{(x_i, y_i)\}_{i=1}^n \subset \mathbb{R}^2 \times \mathbb{R}$ is generated according to the following model,

$$y_i = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix} x_i + \epsilon_i,$$

where $\epsilon_i \sim \mathcal{N}(0, 0.1)$ and the size of the dataset is fixed to $n = 50$. The data x_i follows a Gaussian distribution

$$x_i \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} \right),$$

where we will vary the correlation ρ. Let $P \subset [-0.99, 0.99]$ be a set of 50 evenly spaced values (see Matlab function `linspace`) for ρ. For each of these values repeat the following steps 100 times,

- randomly sample a data set $\{(x_i, y_i)\}_{i=1}^n$ using `mvnrnd`,
- estimate the parameters `alpha` using your function `ols` and
- store the obtained estimates `alpha`.

Create a plot that shows the variance of the estimated first parameter (vertical axis) against the correlation ρ (horizontal axis). Explain why the curve has this shape. What does this mean in practice, when we are interested in the true value of a parameter? **(10 Punkte)**

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