Übung zur Vorlesung

Maschinelles Lernen 1

Wintersemester 2008/2009

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

Blatt 4

Abgabe bis Mittwoch, 12. November 2008, 14:00 Uhr
unter https://ml01.service.tu-berlin.de/~mikio/pass.pl?conf=blatt4.conf
Schreibt eine Email an mikio@cs.tu-berlin.de für Eure Zugangsdaten.

Aufgaben

Auf diesem Übungsblatt sollen die Bayessche Entscheidungstheorie zusammen mit Maximum-Likelihood-Schätzern an einem einfachen Beispiel praktisch umgesetzt werden. Hierzu kann auf der Vorlesungsseite [1] ein Programmskelett heruntergeladen werden, in welches Ihr die Lösungen einfügt. Die vollständige Datei wird dann über ein Webinterface (siehe oben) abgegeben. Zugriff auf Matlab habt ihr auf den Solarisrechnern des IRBs unter /home/ml/ml/bin/matlab.

Es gibt pro Aufgabenteil jeweils 5 Punkte. Besonders wichtig ist beim Programmieren in Matlab, dass man, wenn irgendwie möglich, for-Schleifen vermeidet und statt dessen mit den entsprechenden Matrix- und Vektorausdrücken hantiert. Zur Lösung dieser Aufgaben ist keine einzige for-Schleife nötig! Um einen zusätzlichen Anreiz zu schaffen wird jede for-Schleife mit einem Punkt Abzug gewertet.

1. `plot_data`: Plotte den Datensatz. Die Klasse $Y = 1$ soll mit roten Kreuzen und die Klasse $Y = -1$ mit blauen Kreisen dargestellt werden. Stelle sicher, dass die Achsen gleich skaliert sind und ein Gitter gezeichnet wird. (siehe `plot`, `axis`, `grid`)

2. `ml_estimate`: Schätze den Mittelwert und die Kovarianzmatrix mittels des Maximum-Likelihood-Schätzers für die multivariate Gaußverteilung:

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i, \qquad \hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu})(x_i - \hat{\mu})^T.$$

3. `prior_estimate`: Schätze die Klassenpriors für die jeweilige Klasse durch den tatsächlichen Anteil an den Daten.

4. `gaussian`: Berechne die Dichte der multivariaten Gaußverteilung

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right),$$

   wobei d die Dimension von x ist. Schreibe die Funktion so, dass man direkt mehrere Punkte als Matrix übergeben kann (mit den Punkten als Zeilen). Vermeide die Verwendung von for-Schleifen!

5. `predict`: Sage die Bayes-optimale Klasse gegeben die geschätzten Gaußverteilungen voraus. Auch diese Funktion sollte so geschrieben sein, dass sie direkt für mehrere Punkte in einer Matrix funktioniert.

6. `posterior_grid`: Berechne die Posterior-Verteilung für die gegebene Klasse auf dem als zwei Matrizen übergebenen Gitter. (siehe `meshgrid`, `reshape`)

[1] https://ml01.zrz.tu-berlin.de/cgi-bin/twiki/view/Main/MaschinellesLernenW08

Die folgende Datei kann auf der Vorlesungsseite heruntergeladen werden!

```
function sheet04
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DO NOT EDIT STARTING HERE, EDIT ONLY BELOW

% Generate some data.
[X, Y] = generate_data(1000, 500);

% and test data
[XE, YE] = generate_data(2000, 1000);

plot_data(X, Y);

% estimate mean and covariance function for the two classes
[C.MP, C.CP] = ml_estimate(X, Y, 1);
[C.MN, C.CN] = ml_estimate(X, Y, -1);

% estimate the class priors
C.PP = prior_estimate(X, Y, 1);
C.PN = prior_estimate(X, Y, -1);

C

% predict class labels for the training and test data
YH = predict(C, X);
YEH = predict(C, XE);

% display errors
fprintf('training error: %.2f%%\n', mean(YH ~= Y)*100);
fprintf('test error: %.2f%%\n', mean(YEH ~= YE)*100);

% plot decision function
[MX, MY] = meshgrid(linspace(-20, 20, 30), linspace(-10, 20, 30));
Z = posterior_grid(C, MX, MY, 1);

hold on;
contour(MX, MY, Z, [0.5]);
colormap([0 0 0]);
hold off;

function [X, Y] = generate_data(n1, n2)
X = [randn(n1, 2)*diag([1, 2])*rotmat(pi/4) + repmat([-7, 0], n1, 1);...
     randn(n2, 2)*diag([3,4])*rotmat(3*pi/4) + repmat([4, 4], n2, 1)];
Y = [ones(n1, 1); -ones(n2, 1)];

function R = rotmat(phi)
R = [cos(phi), sin(phi); -sin(phi), cos(phi)];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% EDIT BELOW

% 1. Plot the data set by plotting the class for Y = 1 with red crosses and
% the class with Y == -1 with blue circles. Make sure axes are equally
% scaled and add a grid (5 Punkte).
function plot_data(X, Y)
% ...

% 2. Estimate the mean and covariance using the ml-estimator for
% multivariate Gaussians for the given class (5 Punkte).
function [M, C] = ml_estimate(X, Y, class)
```

```
% ...

% 3. Estimate the class priors for the given class (5 Punkte)
function P = prior_estimate(X, Y, class)
% ...

% 4. Compute the multivariate density (5 Punkte)
function P = gaussian(M, C, X)
% ...

% 5. Predict the more likely class based on the class posterior. (5 Punkte)
function YH = predict(C, X)
% ...

% 6. Compute class posteriors on a grid. (5 Punkte)
function P = posterior_grid(C, MX, MY, class)
X = [reshape(MX, prod(size(MX)), 1), ...
     reshape(MY, prod(size(MY)), 1)];
% ...
```
