Übung zur Vorlesung

Maschinelles Lernen 2

Sommersemester 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 Montag, 18. Mai 2009, 13:00 Uhr
per Email an mikio@cs.tu-berlin.de, Ausarbeitung im Sekretariat FR6052, oder bei Mikio Braun, FR6058.

Aufgaben

Auf diesem Aufgabenblatt soll Semi-Supervised-Learning mit Graph-Laplacians untersucht werden. Hierzu seien $X_1, \ldots, X_n$ die bekannten Eingaben und $Y_1, \ldots, Y_m$ die bekannten Ausgaben (mit $m < n$!).

Gelernt wird eine Kernfunktion

$$f(x) = \sum_{i=1}^{n} \alpha_i k(X_i, x),$$

das heisst mit einem Kern für jeden Eingabepunkt. Minimiert werden soll der quadratische Fehler auf den bekannten Ausgaben, wobei das Problem noch durch zwei weitere Terme regularisiert wird, einmal das bekannte $\alpha^t K \alpha$, das auch bei Kernel-Ridge-Regression Verwendung findet, und dann noch $\alpha^t K L K \alpha$, wobei L der *Graph-Laplacian* ist, der wie folgt definiert ist:

$$L = D - A, \text{ mit } A_{ij} = \exp(-\|X_i - X_j\|^2/2g),$$
$$D_{ii} = \sum_{s=1}^{n} A_{sj}.$$

Das Optimierungsproblem lautet also

$$\min_{\alpha} \|K'\alpha - Y\|^2 + \lambda \alpha^t K \alpha + \tau \alpha^t K L K \alpha,$$

wobei

K' eine $m \times n$ Matrix mit Einträgen $k(X_i, X_j)$ für $1 \leq i \leq m$, $1 \leq j \leq n$ ist, und
K eine $n \times n$ Matrix mit Einträgen $k(X_i, X_j)$ für $1 \leq i, j \leq n$

ist.

1. **(10 Punkte)** Zeige, dass die Lösung

   $$\hat{\alpha} = (K'^t K' + \lambda K + \tau K L K)^{-1} K'^t Y$$

   ist.

2. **(5 Punkte)** Implementiere `twomoon`, das den Two-Moon Datensatz erzeugt (siehe Beispiel unten). Stelle sicher, dass die Klassen nicht einfach linear zu trennen sind.

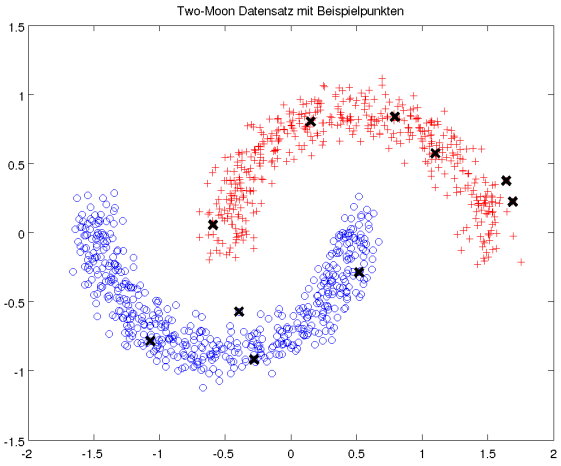
3. **(3 Punkte)** Implementiere `subset`, das aus einem Datensatz ein zufällig Teilmenge auswählt.

4. **(5 Punkte)** Implementiere `graphLaplacian`, das den Graph-Laplacian berechnet.

5. **(7 Punkte)** Implementiere `trainGLKRR`, das obiges Gleichssystem löst. (Hinweis: Es kann möglich sein, dass die Matrix noch ein wenig regularisiert werden muß. Dies erreicht man z.B. durch Addieren einer Einheitsmatrix, die mit einer kleinen Zahl wie 10^{-6} multipliziert ist. Probiere verschiedene Parameter für die Kernbreiten und Regularisierungen aus.

```
function sheet04

% get the data
[X, Y] = twomoon(1000);

% get a subset of the data
figure(1)
NS = 10;
[XS, YS] = subset(NS, X, Y);
plot2ddata(X, Y);
hold on
plot(XS(:, 1), XS(:, 2), 'kx', 'MarkerSize', 10, 'LineWidth', 3);
hold off
title('data and example points')

% train KRR
figure(2);
K = rbfkern(1, XS, XS);
alpha = inv(K + 1e-6*eye(NS))*YS;

YH = rbfkern(1, X, XS) * alpha;

plot2ddata(X, YH);
hold on;
plot(XS(:,1), XS(:, 2), 'kx', 'MarkerSize', 10, 'LineWidth', 3);
hold off

title('learned only using sample points')

% train with KRR + Graph Laplacian regularization
figure(3)
alpha = trainGLKRR(1, 0.001, 0.1, 1, XS, YS, X);
YH = rbfkern(1, X, X) * alpha;
plot2ddata(X, YH);
hold on;
plot(XS(:,1), XS(:, 2), 'kx', 'MarkerSize', 10, 'LineWidth', 3);
hold off

title('learned with graph Laplacian regularizer')

function K = rbfkern(w, X, Y)
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';
K = exp(-D/(2*w));

function plot2ddata(X, Y)
P = (sign(Y) == 1);
N = (sign(Y) == -1);
plot(X(P, 1), X(P, 2), 'r+', X(N, 1), X(N, 2), 'bo');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% your solution below
%

% 2. Generate a two-moon data set with N points and some noise. Make sure
% that classes are not linearly seprable.
function [X, Y] = twomoon(N)
```

```
% ...

% 3. Choose random subset of size N from a data set X, Y.
function [X, Y] = subset(N, X, Y)
% ...

% 4. Compute the graph-Laplacian for Gaussian weights with width w
function L = graphLaplacian(w, X)
% ...

% 5. Train KRR with Graph-Laplacian regularization on the data
%
% Parameters:
%    w - width for the rbf kernel
%    g - width for the graph-Laplacian
%    lambda - regularization constant for general smoothness
%    tau - regularization constant for GL-regularizer
%    XS, YS - data set with labels
%    X - unlabeled data
function alpha = trainGLKRR(w, g, lambda, tau, XS, YS, X)
% ...
```



Beispieldatensatz “Two-Moons”