CPSC-340 Machine Learning and Data Mining 2005

Homework # 5
Due Friday, Nov 4 at 4pm.

NAME:_____

Signature:_____

STD. NUM: _____

Email: _____

General guidelines for homeworks:

You are encouraged to discuss the problems with others in the class, but all write-ups are to be done on your own.
Homework grades will be based not only on getting the "correct answer," but also on good writing style and clear presentation of your solution. It is your responsibility to make sure that the graders can easily follow your line of reasoning.
Try every problem. Even if you can't solve the problem, you will receive partial credit for explaining why you got stuck on a promising line of attack. More importantly, you will get valuable feedback that will help you learn the material.
Please acknowledge the people with whom you discussed the problems and what sources you used to help you solve the problem (e.g. websites, books from the library). This won't affect your grade but is important as academic honesty.
When dealing with Matlab exercises, please attach a printout with all your code and show your results clearly.

1. **PCA and Kernel Machines for Motion Capture Data**: In this exercise the data consists of a matrix $\mathbf{X} \in \mathbb{R}^{\mathbf{993}\times\mathbf{69}}$ of 993 captured human poses of 69 points. Each point corresponds to a position of a body part, e.g. neck, left arm, right foot and so on. The goal of the exercise is to project the 993 poses to a 2D space using PCA. Next, we construct a nonlinear regression function from the 2D PCA space to the 69D body part space. That is, by specifying a point in the 2D space, the nonlinear regression model should predict a pose in the 69D space. This enables us to control a character with many degrees of freedom using a 2-dimensional interface.

   The website provides three files. The file `loaddata.m` simply loads the data. The file `defineLines.m` describes how each body part is connected to the other body parts using lines. Finally, the file `mocaph.m` is the most important one. You should edit this file and enter the right commands (where question marks appear) in order to do PCA and nonlinear regression. Note that the nonlinear regression machine is tested both on training and test data and compared to the true data.

   You should hand in all your code and a picture of the last frame of animation.

2. **Support Vector Machines for Medical Diagnosis**: Here, we consider an interesting nonlinear classification data set collected as part of a study to identify patients with muscle tremor. The data was gathered from a group of patients (9 with, primarily, Parkinson's disease or multiple sclerosis) and from a control group (not exhibiting the disease). Arm muscle tremor was measured with a 3-D mouse and a movement tracker in three linear and three angular directions. The time series of the measurements were parameterised using a set of autoregressive models. The number of features was then reduced to two with PCA. Our goal is, therefore, to use these 2D features to classify patients.

   (i) Download the necessary function files and data from the homework website. Complete the quadprog function in the following m-file:

```
clear; echo off;

% PARAMETERS:
% ==========

kernelFunction = 'expK';    % Kernel function: either polyK or expK
kernelPar = 1e0;            % Kernel parameter: either polynomial degree
                            % (if polyK) or Gaussian variance (if expK).


% LOAD TREMOR DATA:
% ================
load diagnosisData;
[r,c] = size(data);
N = 60;                     % Number of data used for training.
Ntest = r - N;              % remaining points are used for testing
trainData = data(1:N,:);
testData = data(N+1:r,:);
x = trainData(1:N,1:2);             % Train set input data.
y = trainData(1:N,3);               % Train set target data.
y = 2*y-1;                          % Convert (0,1) labels to (-1,1).
xt = data(N+1:r,1:2);               % Test set input data.
```

```
yt = data(N+1:r,3);                % Test set target data.
yt = 2*yt-1;


% TRAIN SVM:
% ==========

C = 1e+10; % Upper bound on alpha.
K = feval(kernelFunction,x,x,kernelPar);
[alpha, fval, exitFlag] = quadprog(???,???,[],[],y',0,zeros(N,1),C*ones(N,1));
alphaY = alpha .* y;  % The alpha are the lagrange multipliers

% Account for numerical innacuracies by ignoring alphas very close to 0 or
% C (infinity ideally).
eps = 0.001*max(alpha);
supVecs = find(alpha>=eps)
% These following are better for dealing with numerical accuracies:
margVecs =  find(alpha >= eps & alpha <= C-eps)
% Compute the bias parameter.
b = median(y(margVecs)-K(margVecs,:)*alphaY);
% Calculate the margin:
margin = 1/sqrt(alphaY'*K*alphaY);


% PLOT RESULTS:
% =============
% We compute the SVM prediction over a grid of data. This allows
% us to plot the separating hyperplane.
% First we generate the 2D grid:
mm = [min(x) ; max(x)] ;
mm = mm + [-0.1 ; 0.1] * (mm(2,:)-mm(1,:)) ;
x1range = mm(:,1);
x2range = mm(:,2);
x1rangeFilled = x1range(1):((x1range(2)-range(1))/100):x1range(2);
[x1,x2] = meshgrid(x1rangeFilled,x1rangeFilled);
[s1 s2] = size(x1);
% Generate predictions over the 2D grid:
[yPred,z] = svmPredict([reshape(x1,[],1) reshape(x2,[],1)],x,y,alpha,b,kernelFunction,kernelPar);
yPred = reshape(yPred,s1,s2);
z = reshape(z,s1,s2);

figure(1)
clf;
hold on;
% Plot the decision boundary
contour(x1,x2,z,[0 0],'b');
contour(x1,x2,z,[-1 1],'b:');
% Circle the support vectors
%scatter(x(supVecs,1),x(supVecs,2),70,'k');
% Double circle the margin vectors
scatter(x(margVecs,1),x(margVecs,2),150,'k');
% Plot the training set
plot(x(y>0,1),x(y>0,2),'r+',x(y<0,1),x(y<0,2),'gx','linewidth',2);
hold off;
```

(ii) Plot the decision boundary again, but this time plot the test data and not the training data.

(iii) Change the Kernel parameter (Gaussian variance) from 1 to 100. Plot the boundary with the training data and then with the test data (i.e., 2 plots). What can you conclude?

In total, you should hand in 4 plots (2 train, 2 test).