CPSC-340 Machine Learning and Data Mining 2004

Homework # 5

Due Friday, Nov 19 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. **Question 1: K-means and EM**: The file `clusterData.dat` on the homework website contains 200 rows of (x_1, x_2) pairs.

   (i) Plot this data. The plots in the following parts should be plotted on top of this plot.

   (ii) Implement the K-means algorithm and apply it to the data, using 3 clusters and iterating until the algorithm converges. For initialisation, use the cluster centroids $\mu_1 = (0.5, 2.3)$, $\mu_2 = (0.5, 0.5)$ and $\mu_3 = (5.2, -0.1)$. Plot the evolution of the cluster centroids as the algorithm runs (that is, plot $\mu_c^{(t)}$ for each c and iteration t).

   (iii) Implement the EM algorithm for fitting a mixture of Gaussians,

$$p(x|\theta) = \sum_{c=1}^{3} p(c)\mathcal{N}(x|\mu_c, \Sigma_c),$$

   on the same dataset. For initialisation, use the same means as in (ii) and set $p(c) = (1/3, 1/3, 1/3)$ and $\Sigma_1 = \Sigma_2 = \Sigma_3 = I$. Plot the evolution of the means and (optional) superimpose the elipses for the variances around the respective means.

   (iv) Comment on the relative rates at which the two algorithms converge.

2. **Question 2: Vector quantisation and image compression**: In this problem, we will apply the K-means algorithm to lossy image compression, by reducing the number of colors used in an image.

   The data website contains a 512x512 image of a mandrill represented in 24-bit colour. This means that, for each 262144 pixels in the image, there are three 8-bit numbers (each ranging from 0 to 255) that represent the green, red and blue intensity values for that pixel. The straightforward representation of this image therefore takes about 262144x3 = 786432 bytes (a byte being 8 bits). To compress the image, we will use K-means to reduce the image to K=16 colours. More specifically, each pixel in the image is considered a point in the three-dimensional (r,g,b)-space. To compress the image, we will cluster these points in colour-space into 16 clusters, and replace each pixel with the closest cluster centroid.

   (i) Load and plot the image `mandrill-large.tiff` using the following Matlab commands:

   ```
   A = double(imread('mandrill-large.tiff'));
   imshow(uint8(round(A)));
   ```

   The first line creates a three dimensional matrix, such that `A(:,:,1)`,`A(:,:,2)` and `A(:,:,3)` are 512x512 arrays that respectively contain the red, green and blue values for each pixel. Since this image is large, K-means would take too long. Instead, you should load and cluster the image `mandrill-small.tiff`. In particular, treat each pixel's (r,g,b) values as an element of $\mathbb{R}^3$ and run K-means with 16 clusters on this image. Iterate to (preferably) convergence, but in any case for less than 30 iterations. For initialisation, set each cluster centroid to the (r,g,b) values of a randomly chosen pixel in the image.

   (ii) Take the matrix A from `mandril-large.tiff`, and replace each pixel's (r,g,b) values with the value of the closest cluster centroid. Display the new image, and compare it visually to the original image. Hand in all your code and a printout of your compressed image next to the original image (printing on a black-and-white printer is fine).

3. **5. Support Vector Machines**: 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).