CPSC-540 Machine Learning 2005

Homework # 5

Due Wednesday April 13 in class.

NAME:_____

Signature:_____

STD. NUM: _____

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. 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. **Kernel Ridge Regression**:

   Repeat the experiment of question 5 in Homework 2 using kernel ridge regression. To guide you, you should download the files kerneldemo.m and the files for the three types of kernel discussed in class. You're welcome to use any of the three. You should hand in the code and the plots of the train and test set errors.

2. **Semi-Supervised Learning**

   In this question, you need to fill in the missing code (where it says ???) in order to implement the kernel semi-supervised learning algorithm. To test your algorithm you need to download the data file `ringData.mat`. Only two points are labelled in this data set. The task is to compute the labels of the other points. Repeat the experiment for $\sigma = 0.2$ and $\sigma = 1.75$. Comment on how the choice of σ affects the weight matrix $\mathbf{W}$ and the results. Hand in all the figures and the missing code.

   The partial code follows. When typing it in, please make sure you understand each command and line of code. This is a good Matlab learning exercise. Use the Matlab command `help` when in doubt.

```
clear;
echo off;

% Load the data
load ringData; N = size(Data,1);

%Parameters
sigma = 0.2;  % Kernel width parameter.
kernelFunction = 'expK';   % Kernel function.

% ASSUME THERE ARE ONLY TWO LABELS: Data(L:N,3)
% All the other labels are assumed to be unknown
U = N-2;   % Index of the last unlabeled node
L = N-1;   % Index of the first labeled node

% Plot the input data
figure(1)
clf;
plot(Data(1:N,1), Data(1:N,2),'.k');
axis([min([min(Data(:,1)), -10]), max([max(Data(:,1)), 10]),...
      min([min(Data(:,2)), -10]), max([max(Data(:,2)), 10])]);
title('Input data');

% Compute Gram Matrix W
W = feval(kernelFunction,Data(:,1:2),???,???);

% Compute diagonal matrix D
```

```
D = diag(sum(???,2));

% Plot the matrix W as an image
figure(2) clf; colormap('gray') image(100*W) title('Weight
matrix')


% COMPUTE LABELS:
% ===============
Y = zeros(N,1);
% Labeled data are known
Y(L:N) = Data(L:N,3);
% Unlabeled data must me computed:
Y(1:U) = ???;

% Set classification treshold
Tresh = 0.5;


% Plot the compute unlabeled points Y(1:U) and the threshold.
figure(3)
clf;
subplot(1,2,1);

A_lin = linspace(min(Data(1:U,1)),max(Data(1:U,1)), 50);

B_lin = linspace(min(Data(1:U,2)), max(Data(1:U,2)), 50);

[A, B] = meshgrid(A_lin, B_lin);

C =griddata(Data(1:U,1), Data(1:U,2), Y(1:U), A, B, 'cubic');
mesh(A,B,C);
view(-123, 16.00);
hold on;

plot3(Data(:,1), Data(:,2), Y, 'k.');
[A, B] = meshgrid(-10:2:10,
-10:2:10);
C = zeros(11,11) + Tresh;

mesh(A,B,C, 'EdgeColor', 'r'); hold off;


% Plot the classification results
subplot(1,2,2); hold on; for i=1:N
```

```
    if Y(i) >= Tresh
        plot(Data(i,1), Data(i,2), '+b');
    else
        plot(Data(i,1), Data(i,2), '*r');
    end;
end;
axis([min([min(Data(:,1)), -10]), max([max(Data(:,1)),
10]),...
      min([min(Data(:,2)), -10]), max([max(Data(:,2)), 10])]);
hold off; title('results');
```

3. **Kernel PCA**

   Repeat the PCA exercise of homework 1, but this time use kernel PCA with a kernel of your choice.

4. **Clustering 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).