Assignment 3

For this assignment submit implementation of ONE of the following 4 problems.

1. Perceptron and Neural Networks

In this homework, you will implement the Perceptron algorithm in python and compare it with WEKA implementation of Neural networks. You will also compare it with either your own or WEKA implementations of Logistic Regression and Naive Bayes.

TASK 1: Download the datasets available on the class webpage or elearning. As in homework 2, the classification task is spam/ham.

80 Points: Implement the perceptron algorithm in Python (use the perceptron training rule and not the gradient descent rule). Your task here is to experiment with different values of number of iterations and the learning rate. Report the accuracy for 20 suitable combinations of number of iterations and the learning rate. Repeat your experiment by filtering out the stop words. Compare the accuracy of your perceptron implementation with that of Naive Bayes and Logistic Regression (implemented in Homework 2). If you did not implement Naive Bayes (NB) and/or Logistic regression (LR) in homework 2 or you are not sure that your implementation is correct, please use the LR and NB implementations available in WEKA.

20 points Neural networks in WEKA.
Download WEKA http://www.cs.waikato.ac.nz/ml/weka/.
Convert the spam/ham dataset into the ARFF format used by WEKA.

Using the Neural networks implementation in WEKA (called MultiLayered Perceptron), report the accuracy on the test set. Experiment with different number of hidden layers and units. Report on how the number of hidden layers and units as well as other options such as momentum, number of iterations, and learning rate affect the accuracy.

2. K-means clustering on images

In this problem, you will use K-means clustering for image compression. We have provided you with two images.
• Display the images after data compression using K-means clustering for different values of K (2, 5, 10, 15, 20).
• Is there a tradeoff between image quality and degree of compression? What would be a good value of K for each of the two images?
You have to implement the function k-means using python. Note that your program must compile and we should be able to replicate your results. Otherwise no credit will be given.
What to turn in:
1. Your code in Python and datasets

2. A README for your compiling/using your code
3. A report (pdf or doc file) containing answers to the questions posed.

3. Support vector machines

• Download LIBSVM, currently the most widely used SVM implementation. Read the documentations to understand how to use it.
• Download the new promoter's dataset in the LIBSVM format. (Available on the elearning).
• Run LIBSVM to classify promoters. Try different kernels (0-3) and use default parameters for everything else. How does it vary with different choice of kernel?

4. EM algorithm

• Download the data from the class web page.
• Implement the EM algorithm in Python for general Gaussian mixture models (assume that the data is an array of long doubles). Use the algorithm to cluster the given data Remember that the data is 1-D. The recommendation is that you run the algorithm multiple times from a number of different initialization points (different θ^0 values) and pick the one that results in the highest log-likelihood (since EM in general only finds local maxima). One heuristic is to select r different randomly-chosen initialization conditions. For example, for each start, select the initial K Gaussian means by randomly selecting K initial data points, and select the initial K variances as all being some multiple of the overall data variance – the selection of initial covariance is not as critical as the initial means. Another option for initialization is to randomly assign class labels to the training data points and then calculate θ^0 based on this initial random assignment. Another option is to begin the iterations by executing a single M-step.

Report the parameters you get for different initializations. What initialization strategy did you use? How sensitive was the performance to the initial settings of parameters.

Now assume that variance equals 1.0 for all the three clusters and you only have to estimate the means of the three clusters using EM. Report the parameters you get for different initializations. Which approach worked better in terms of convergence and final probability function, this one or the previous one, Why? Explain your answer.

What to turn in for this part:
• Your code. EM for general GMMs and EM for GMMs with known variance.
• A report containing answers to the questions above.