Machine Learning
Assignment 2
Due to **24/12/2017.**

General Instructions:

The solution should be formatted as a report and running code should be included in a digital form. The solution can be done in pairs independently to other groups. Identical (or very similar solutions) are not allowed!

1. C_1 and C_2 are two classes with one-dimensional normal densities $N_1(-2,1)$ and $N_2(1,1.5)$ and the same priors, $P(C_1)=P(C_2)$.

i. Create 300 normally distributed samples of each class (you may use the function randn.m)

ii. Estimate parametrically the mean and standard deviation of each class and compare to the true values. Discuss the differences.

iii. Given that the errors from both classes have the same cost, what is the best decision rule and what is the classification error for this rule. Add this rule to your previous plot.

v. Given that misclassifying a sample from C_1 costs four times as misclassifying a sample from C_2, plot on the same graph the new decision boundary.

2. Implement in Matlab text classification Naïve Bayes Algorithm:

a. Write a function `[Pw, P]=learn_NB_text` that computes probabilities (`Pw` –a matrix of class-conditional probabilities, `P-` a vector of class priors).

b. Write a function **`suc=ClassifyNB_text(Pw, P)`** that classifies all documents from the test set and computes the success rate (**`suc`**) as a number of correctly classified documents divided by the number of all documents in the test set.

c. Report the classification rate.

Note: Multiplying lots of probabilities, which are between 0 and 1 by definition, can result in floating-point underflow. Since $\log(xy) = \log(x) + \log(y)$, it is better to perform all computations by summing logs of probabilities rather than multiplying probabilities. Class with highest final un-normalized log probability score is still the most probable.

Data and Supplied Code: Download `textClassif.zip` from moodle. The archive contains the following files.

`r8-test-stemmed.txt` -- test set for 8 categories.

`r8-train-stemmed.txt` -- train set for 8 categories.

`readTrainData.m` -- a matlab function that reads all documents from the train set creates the following data structures and saves them in a mat file "corpus_train.mat":

`texAll` – cell array of documents; each entry corresponds to a document which is a cell array of words.
`lbAll` – cell array of documents' labels.
`Voc` – cell array of all distinct words in the train set.
`cat` – cell array of document categories.

`readText.m` a matlab function used in `readTrainData`.m

Instructions for using the code:

1. Unzip the data and the code in the same directory.
2. Run `readTrainData('r8-train-stemmed.txt')`. It should create `corpus_train.mat` file in the same directory.
3. Write the required functions and run them. You can use the code in `readTrainData`.m as an example for reading the test data.
4. Report the success rate on the test set.

3. Parzen Window:

Bayesian decision rule under zero-one loss function using Parzen window can be reduced to the following rule: "assign category of majority vote of neighbors", which for the cubic window function can be written as

$$\sum_{l=1}^{n_j} \varphi\left(\frac{x_l^j - x}{h}\right) \geq \sum_{l=1}^{n_i} \varphi\left(\frac{x_l^i - x}{h}\right) \quad \text{for } \forall i \neq j \qquad [1]$$

where x is a test sample, c is the number of classes and n_i is the number of samples in class $i = 1,...,c$.

Using the rule in eq. 1 for a Gaussian window results in:

$$\frac{1}{n_j}\sum_{l=1}^{n_j} \exp\left(-\frac{\left\|x_l^j - x\right\|^2}{2\sigma^2}\right) \geq \frac{1}{n_i}\sum_{l=1}^{n_i} \exp\left(-\frac{\left\|x_l^i - x\right\|^2}{2\sigma^2}\right) \quad \text{for } \forall i \neq j \quad [2]$$

Probabilistic neural network (PNN) is closely related to Parzen window pdf estimator. A PNN consists of several sub-networks, each of which is a Parzen window pdf estimator for each of the classes.

Algorithm 1 (PNN)

The input nodes are the set of measurements**.**
The second layer consists of the Gaussian functions formed using the given set of data points as centers.
The third layer performs an average operation of the outputs from the second layer for each class.
The fourth layer performs a vote, selecting the largest value. The associated class label is then determined.

Algorithm1 essentially implements the rule in Eq. 2.

Note: if you are not familiar with neural networks, consider layer as a step in the algorithm. No knowledge of neural networks is required for this problem set.

Example: Suppose that for class 1, there are five data points [2, 2.5, 3, 1, 6]. For class 2 there are three data points [6, 6.5, 7]. Using the Gaussian window function with $\sigma = 1$, the Parzen pdf for class 1 and class 2 at *x* are

$$y_1(x) = \frac{1}{5}\sum_{i=1}^{5} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x_{1,i}-x)^2}{2}\right)$$

and

$$y_2(x) = \frac{1}{3}\sum_{i=1}^{3} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x_{2,i}-x)^2}{2}\right)$$

respectively. The PNN classifies a new *x* by comparing the values of $y_1(x)$ and $y_2(x)$. If $y_1(x) \geq y_2(x)$, then *x* is assigned to class 1; otherwise class 2. For this example $y_1(x) =$ 0.2103.

$$\begin{aligned} y_2(3) = \frac{1}{3\sqrt{2\pi}} \Big\{ &\exp\left(-\frac{(6-3)^2}{2}\right) \\ &+ \exp\left(-\frac{(6.5-3)^2}{2}\right) \\ &+ \exp\left(-\frac{(7-3)^2}{2}\right)\Big\} \\ &= 0.0011 < 0.2103 = y_1(x) \end{aligned}$$

so the sample *x* = 3 will be classified as class 1 using PNN.

Instructions:

Implement PNN as described in Algorithm 1 for 2-class problem of classifying an input sample as letter 'A' or 'B'. Report the classification rate as the number of correctly classified test samples divided by the number of all tested samples.

You should write your own implementation of PNN. The use of NN Matlab toolbox or any other NN toolbox is not allowed (and is unrelated to the problem).

Data: Download `dataAB.mat` from moodle. The file contains the following data:

train_dataA , train_dataB are matrices of size 500x16, containing 500 16-dimensional training samples (the letters are represented by 16 attributes).

valid_dataA, valid_dataB are matrices of size 100x16, containing 100 16-dimensional validation samples (use these for choosing the window size).

test_dataA, test_dataB are matrices containing the remaining samples for test. Note that the number of test samples is different for 'A' and 'B'.

4. Use KNN to classify between 4 letters from the test set. Use the validation set for choosing the best K. Report the classification rate on the test set.
You can use the MATLAB function **knnsearch** in your implementation.

Data: Download `ABCD_data.mat` from moodle. The file contains the following data:

train_data is a matrix of size 2000x16, containing 500 16-dimensional training samples for letters A,B,C,D in that order (in contains 500 samples of A, then 500 samples of B, and so on).

valid_data is a matrix of size 400x16, containing 100 16-dimensional validation samples for the four letters (100 samples for A, then 100 samples for B and so on)

test_data is a matrix containing 16-dimensional test samples for the four letters (100 samples for A, then 100 samples for B and so on).

GOOD LUCK!