CSCE 970: Pattern Recognition	**Spring Semester, 1999**
Homework Assignment 1	
February 8, 1999	*Due Date: March 1*

1. (70 pts) Implement the following simple classifiers:

   (a) A Bayesian classifier that assumes the pdf for each class is a Gaussian and uses maximum likelihood estimation to estimate both the mean vector $\boldsymbol{\mu}$ and the covariance matrix Σ for each class. For this part you should report what your estimates were for each class for each data set. *Hint: see Exercise 2.19.*

   (b) A Bayesian classifier that uses Parzen windows to estimate the pdf for each class. Use $\mathcal{N}(\mathbf{0}, I)$ as your kernel function:

   $$\phi(\mathbf{x}) = \left(1/(2\pi)^{\ell/2}\right) \exp\left(-\mathbf{x}^T \cdot \mathbf{x}/2\right).$$

   (c) k-nearest neighbor, using at least three different values of k.

   (d) The Perceptron algrorithm, using at least three different learning rates and at least three different numbers of passes through the training data.

   (e) Winnow, using at least three different learning rates and at least three different numbers of passes through the training data.

The number of classes is $M = 2$, the number of dimensions is $\ell = 2$. The data sets were generated according to different probability distributions and some are linearly separable while others are not (each testing set is generated in the same fashion as its corresponding training set). *Do not forget to estimate $P(\omega_i)$ in your Bayesian classifiers, since not all classes are equally probable!*

When you choose values for your different parameters, choose widely varying numbers (e.g. differing by factors of 2 or more), so you can see how the parameter values impact training speed and classification error.

Report on the performance of these classifiers on the different data sets with the different parameter values (one way is to hold all parameter values fixed while varying one, measuring its effect). (You might also try varying the number of training examples to see which classifiers perform better with less data.) Include comments on time (in number of iterations and in real time) to train, time to test, and error rates during training and testing. Contrast the performances (error rates and times) of these classifiers on the different sets. From these results, what can you infer about the characteristics of each data set in terms of linear separability, probability distribution, etc.? Based on the characteristics of each data set, which classifier do you feel is most appropriate?

Do you think better results could be obtained by using other methods (including those from Chapter 4)? Which methods do you think would improve performance and why?

When you hand in your results, submit source code electronically as well as a *well-written*, *detailed* report (much of your grade will depend on your presentation).

The data sets are in /home/fac/sscott/970-data on cse. Each directory has four files: class1.train, class2.train, class1.test, class2.test. The *.train files are for training the classifier, the *.test files are for testing it (if you wish to experiment with various amounts of training data, you can prune some data out of the training sets). The files class1.* (resp., class2.*) contain one ω_1 (resp., ω_2) feature vector per line, real numbers separated by spaces. You may reformat the data however you wish, so long as you keep the training and testing sets separate.

2. (10 pts) Describe at least two things covered in lecture and/or in the textbook that you do not fully understand.

3. (15 pts) List at least three topics that you feel would be good subjects for a class project. These topics may be on any material related to pattern recognition, i.e. they need not be related to topics we have covered in class so far.

4. (5 pts) State the approximate amount of time you spent on each problem, especially each subproblem of number 1.

Extra Credit (15 pts) Find a good neural network toolset that is freely available (or write your own). Download and compile it (if necessary) and run it on different training and testing sets (including the ones I generated, if you wish). Briefly report on its performance with different network architectures and parameters. Also report on its ease of use.