6.891 Fall 1999: Problem Set #2

Due: Thursday, 9/30/99

Warning: This problem set will require programming in MATLAB. It is also substantially longer than problem set 1. In order to schedule your time, please read the problem set through completely as soon as you can. For those of you that do not know MATLAB, there are a number of resources available to you. See the course web page for some pointers.

In this problem set you are asked to program a series of classifiers based on what you have learned in class. The time that you take to work out the implementation is time well spent. It will reinforce your understanding and it will empower you to sit down and write your own machine learning programs.

Guidelines: Document and hand in all of the code that you write. In addition, there are a number of thinking questions associated with the programming assignment. Write up and hand in answers to these questions.

The Data: The data set was created by sampling from a distribution model that we constructed by hand. There are 3 classes; 100 examples of each class; and each example has 2 feature values. To load the data type: `load artificial.mat` . See the file `demo.mat` for an introduction.

The data set is contained in three matrices of data. The feature values for each example are collected in the `input` matrix. Each example is stored as a row (i.e. the i'th row contains the feature values for the i'th example). The correct class of the i'th example is stored in the i'th row of the `target` matrix. There is one column in the `target` matrix for each class. The column that contains the value 1 is the correct class; the other columns will contain 0. A third matrix called `test` contains a collection of test examples.

Problem 1: Review of Bayesian Decision Theory

A: Solve problem 2.13 from DHS (page 47 of Ch 2).

B: Solve problem 2.18 from DHS (page 49 of Ch 2).

Problem 2: Bayesian Classification

A: Solve problem 3.17 from DHS (page 59 of Ch 3).

Problem 3: Density Estimation

The files ps2a.mat and ps2b.mat (available from the class web site) each contain sampled points for a different distribution. Using MATLAB, for each of the data sets, estimate the underlying distributions, output the parameters and plot the results using each of the following methods:

A: Histograms with cell widths of: 0.5, 1, 5.

B: Parzen windows:

- Gaussians with variances of: 0.5, 1, 5
- Box car functions of widths: 0.5, 1, 5

[hint: the plots for these may take a while to compute; try debugging the code using only the first 50-100 data points]

C: Mixtures of Gaussians (using EM): 2, 3, 10 Gaussians (For this one, please also print out the parameters of the found models.)

Problem 4: Gaussian Classifier

In order to get you started programming in this problem set we have provided all the necessary programs for this first problem. There use is demonstrated in the file `demo.m`.

A: As with all problems, a good first step is to try a Gaussian classifier. Start out by fitting a Gaussian distribution to each class. Note that this data is multidimensional - the mean is now a vector while the covariance is a matrix. Use this information to build a classifier for the data. Be sure to include an estimate for the prior probability.

B: Plot the means for the Gaussians on a graph that contains all of the data points. Also draw in the first and second eigenvectors of the covariance matrix scaled by the square root of the eigenvalues. Note that Matlab has code that can definitely help in these computations.

C: Plot the decision boundaries generated by the classifier.

Problem 5: Spherically-Symmetric Gaussians

Sir William of Ockham appears to you in a dream and warns you of the dire consequences of including too many parameters in your density estimator. You awake in a cold sweat and decide to severely limit your estimates of the covariance matrices, so that they have the form $\sum = \sigma^2 I$ (i.e. a multiple of the identity matrix). Now rather than requiring N^2 parameters to determine each covariance matrix, you require only one.

A: Using maximum likelihood, derive an estimate for σ. Is the maximum likelihood estimate for the mean of a spherically symmetric Gaussian the same as for a full Gaussian? Justify.

B: Write a program that computes σ from the data and build a classifier from the estimated densities.

C: Plot the means and eigenvectors for the Gaussians and the decision boundaries generated by the classifier as in the previous problem.

Problem 6: Parzen Classifier

A: Construct a classifier based on the Parzen density estimator. Use Gaussian kernels with a covariance matrix that is a multiple of the identity matrix. For this problem use the same σ for each class (in general you might want to use different kernels for each class).

The difficulty with Parzen density techniques is that it is not clear how to pick the best kernel function size and shape.

B: Imagine trying to set σ for the Parzen estimator in the same way as we did for the single Gaussian - using maximum likelihood. What is the maximum likelihood estimate for σ? Is this a good estimate?

C: Propose a general technique for choosing σ which could work for all foreseeable problems. Your answer should be a concrete, but you need not prove that the σ selected is optimal (that is very tough).

D: Plot the decision boundaries generated by this classifier.

Problem 7: Nearest Neighbor Classifier

The Parzen classifier is very flexible and effective approach, but determining the width of the component Gaussians can be a pain. A nearest neighbor classifier is similarly flexible, but does not require a width parameter. A nearest neighbor classifier classifies new data by finding the nearest example in the training set. The class of the new example is taken to be the same as its "nearest neighbor".

A: Plot the decision boundaries generated by this classifier.

B: On the surface, it does not appear as though the nearest neighbor classifier is a probabilistic classifier. We did not start out by computing the density of each class. Nevertheless, is it possible to think of the nearest neighbor classifier as an example of a probabilistic classifier? If so, write out a density associated with each class which will lead to the same classification decisions as the nearest neighbor classifier.

Problem 8: Multiple Linear Discriminants

A: Compute the optimal linear discriminants for this dataset (using the squared error cost function).

B: Plot the decision boundaries generated by this classifier.

Problem 9: Perceptrons

A: First write code which implements the Perceptron learning rule for two classes. Construct a Perceptron which separates the first and second classes of the data (don't worry about the other class right now).

B: Plot the decision boundary generated by this classifier.

C: When there are multiple classes, a single perceptron won't do. Like linear discriminants, Perceptrons can be generalized to perform multi-way classification. Note: this can be a bit tricky. One way to go is to train one classifier for each class so that it returns 1 for elements in the class and a 0 for other classes. Such a scheme can definitely lead to ambiguities (see DHS Figure 5.3). Propose a scheme to disambiguate these outputs.

D: Using this scheme, write code that trains a group of Perceptrons to perform multi-way classification.

E: Plot the decision boundary for class 1 vs. class 2, class 1 vs. class 3, and class 2 vs. class 3 (three graphs total). Generate an additional graph that shows the three decision regions for the multi-class perceptron classifier.

Problem 10: Generalized Linear Techniques

The performance of the linear techniques can be improved if additional non-linear features are included in the input set (e.g. the XOR of two variables, x_1 and x_2, cannot be computed by a perceptron, but can be solved with the additional feature $x_1 x_2$).

A: Propose and then add at least two additional non-linear features to the data. Describe these new features and then add them to `input`.

B: Compute the optimal linear discriminants for this data and plot the decision boundaries.

C: Train a multi-class perceptron for this data and plot the decision boundaries.