6.891 Fall 1999: Problem Set #3

Due: Thursday, 10/14/99

Problem 1: Questions from the DHS

A: Problem 5.1 on page 58

B: Problem 5.29 parts A and B on page 63

C: Problem 6.1 on page 58

D: Problem 6.6 on page 59

Continuing along the same lines as problem set 2, we will start out by implementing a number of discrimination based classifiers. Before we dive right in however, let's discuss the new types of data available for this problem set.

The Data

- **Data Set 1: Artificial Data**

  This is the data set from problem set 2.

- **Data Set 2: Pen Input Characters** This data has been sampled from a pen computer while a user was entering characters. Each character is made up of a series of $\{X, Y\}$ pairs which tell you where the tip of the pen was at different times along the trajectory. The data has been carefully post-processed in order to make the problem a bit simpler. We will not discuss this processing here – though it will be the subject of another problem set. After pre-processing, each character is compressed into 15 numbers and these make up the rows of the input matrix. In order to view the characters, you can use programs that we have provided. For examples of how to use these programs see the file `demo_pen.m`.

  We have included this dataset so that you can build your intuition on harder learning problems (it is also kind of neat).

  In order to load this data type `load pen.mat`.

- **Data Set 3: Mystery Data** We aren't going to tell you what this data is... we will the day after the pset is due. Each input has 21 dimensions and there are 2 classes.

  At the end of this problem set you will be asked to pick a classification technique (or create one) and use it to classify examples from this class.

  In order to load this data type `load mystery.mat`.

For each of the classifiers on this problem set you should try out each of the data sets.

Problem 2: Getting an Accurate and Reliable Classifier

Our goal in this problem set is to find the best possible classifier for a particular problem. The best classifier is the one that has the lowest error rate on future data – data that we have not yet seen. The error that a classifier attains on the training set is one possible estimate for its error on unseen data. As we saw with polynomials it is possible to "overfit" the data (e.g. find a polynomial that fits the training data perfectly, but does a poor job on new data).

In general, one must be very careful to choose a classifier that has the lowest possible bias and the lowest possible variance. One estimate for the bias is the error on the training set. We can estimate the variance by measuring the amount of variation in the classifier observed when it is trained on different independent samples of data. In each of the problems above you are given a single training set, and no means to generate more data. How then can you possibly estimate variance?

One scheme is to "hold-out" or remove a subset of the training data and call it the validation set. One then trains the classifier only on the remaining the data in the training set. One estimate for the variance of the classifier is the difference between the training error on the training set and the prediction error on the validation set (the difference in the error rate).

> **A:** Comment on the validity of the "hold-out" approach for estimating variance. What are the major potential pitfalls?

Another scheme is to N-way cross-validate the classifier. The data set, S, is split into N disjoint subsets of equal size s_i. The classifier is then trained on N different training sets $t_i = S - s_i$ (i.e. the entire set minus the subset). The classifier is then validated on the hold out set: s_i. The bias of the classifier is estimated as the average error across the different hold out sets. The variance of the classifier is estimated as the variance in the errors over the hold out sets.

> **B:** What pitfalls does cross-validation address that were unaddressed by hold-out validation?

Problem 3: Return to the Gaussian Classifiers

Recall the Gaussian classifier that you wrote for the last problem set.

A: This classifier will not work for one of these datasets. Which one? What part of the computation fails? What is the problem?

B: This problem does not arise for the spherically symmetric Gaussian classifier. Why? What is the key difference?

Problem 4: 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 on the artificial data set.

Problem 5: 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 boundaries generated by this classifier on the artificial data set.

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 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 6: 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.

Problem 7: Multi-Layer Perceptron

In general it can be very difficult to find new features which may be useful for classification. The idea behind the Multi-layer Perceptron (MLP) is that it can learn useful non-linear features of the data for you.

A: Implement a code which can evaluate a MLP given a set of weights. Assume that the internal and hidden units of this network are sigmoids.

B: Implement code which trains a MLP to minimizes the squared error of the network. (Hint: Getting the matrix representations for the weights right will make the code pretty easy to write.)

C: Plot the decision boundaries generated by this classifier on the artificial data set.

D: Ungraded but useful exercise: It is possible to plot the output of a hidden unit as if it were a single layer network. Plot each of the output units and conjecture on its function.

Problem 8: Multi-Layer Perceptron Optimization

For each of the datasets experiment with 3 different network topologies. Evaluate each using 5 way cross-validation.

A: Describe each topology, its average error, and the variance of the error.

B: Plot the decision boundary generated by the best cross validated classifier.

Problem 9: Classifier Shootout

Which of the above classifiers do you trust the most? If there was money at stake which set of predictions would you choose? Well there is no money at stake, but there is some ice cream. For each of the data sets we would like you to take your *best* guest for the test data classes. For this task you can use any of the classifiers from this problem set, or you can design your own. Perhaps it might make sense to combine them? If you design your own classifier please describe it concisely but accurately.

As a reward for your efforts, the entry with the lowest error will win a gift certificate for Toscanini's ice cream.

Please submit these predictions in a standard format by email to Kinh. Like the target matrix, the prediction matrix must have one row per example and one column per class. The predicted class has the value 1.0 and the other columns have the value 0.0. In order to help Kinh evaluate your responses, you should write out this matrix using the `save` command in Matlab with the `-ASCII` flag. This will insure that the file can be emailed.

The prediction matrices must be named in a regular fashion. The prediction matrix for the artificial dataset should be called `pred_art`. The prediction matrix for the pen dataset should be called `pred_pen`. The prediction matrix for the mystery dataset should be called `pred_myst`.

When you are done with the pset take the resulting file and include it in an email to Kinh. The subject line of the message should be "PSET 3 predictions". Make sure you don't add any formatting or line breaking, etc. It is important that you use the "-ascii" flag. This will insure that the file can be emailed.