Problem Set 6

MAS 622J/1.126J: Pattern Recognition and Analysis

Due Monday, 20 November 2006

[Note: All instructions to plot data or write a program should be carried out using either Python accompanied by the `matplotlib` package or Matlab. Feel free to use either or both, but in order to maintain a reasonable level of consistency and simplicity we ask that you do not use other software tools.]

Problem 1: Mixture of Gaussians

Implement the EM algorithm for estimating the parameters of a mixture of Gaussians with isotropic covariances $\mathbf{\Sigma_j} = \sigma_j \mathbf{I}$. Note: The data sets are two-dimensional.

To solve this problem you can write your own code or use any MATLAB/Python toolboxes available for the purpose. In particular there is a MATLAB mixture of Gaussians algorithm available for download here that would be good to explore:

`http://dataclustering.cse.msu.edu/`

- Experiment with the number of mixtures and comment on the tradeoff between the number of mixtures and goodness of fit (i.e. loglikelihood) of the data. Suggestion: Plot the loglikelihood as a function of the number of components of a mixture of Gaussians to support your argument.
- Find a fixed number of Gaussians that works well for each data set.
- Plot the estimated Gaussians as one-sigma countours of each mixing component on top of the training data.
- List mean, covariance and mixing weights of each mixture component.
- Include your source code.

Problem 2:

This problem consists of two subproblems that use the same data set. The training and testing data can be downloaded from the class web site. There are two classes with equal prior probabilities and scalar-valued features. Each subproblem focuses on a different classification technique. Each classification technique has free parameters you must estimate using *leave-one-out validation*.

2.1 k-Nearest-Neighbor

a. Use the k-nearest-neighbor method to classify the data (DHS second edition section 4.5.4). As before, the parameter k is undetermined and must be estimated using an evidence curve $v(k)$. For a particular value of k, compute $v(k)$ as follows:

- Break the training data set into two parts, A and B, where B contains only a single sample.
- Determine if B is correctly classified by its k-nearest-neighbors estimate based on A.
- Repeat this process for every possible choice of B. Define $v(h)$ as the number of times the sample left out was correctly classified.

Plot the evidence curve for odd values of k in the range 1 to 19. Pick three values of k from different parts of the curve and plot the resulting discriminant function over the scalar feature space; 1 when choosing the first class, -1 otherwise. Intuitively speaking, what is the best value of k to use?

b. Design a minimum error rate classifier using the k that maximizes $v(k)$. What is its performance on the test data?

2.2 Generalized Linear Discriminant

a. Use the generalized linear discriminant (DHS second edition section 5.3) in conjunction with the pseudoinverse technique (DHS second edition section 5.8.1) to classify the data. That is, for a polynomial of degree k, let

$$\mathbf{y} = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^k \end{bmatrix}$$

be the augmented data vector. Let the margin vector be $\mathbf{b} = \mathbf{1}$, so the weight vector $\mathbf{a}$ can be computed from the pseudoinverse of the data matrix. The degree k is undetermined and is to be estimated using leave-one-out validation.

To estimate k, compute the evidence curve $v(k)$ and then choose the k that maximizes $v(k)$. For a particular value of k, $v(k)$ should be computed as follows:

- Break the training data set into two parts, A and B, where B contains only a single sample.
- Compute the generalized linear discriminant from A only.

- Determine if B (the sample left out) is classified correctly.
- Repeat this process for every possible choice of B. Define $v(h)$ as the number of times the sample left out was correctly classified.

In this process, it is helpful to multiply the augmented data from the second class by -1 (DHS second edition section 5.4).

Plot the evidence curve for $k = \{1, 2, \ldots, 20\}$. Pick three values of k from different parts of the curve and plot the sign of the discriminant function over the scalar feature space; 1 when choosing the first class, -1 otherwise.

b. Design a minimum error rate classifier using the k that maximizes $v(k)$. What is this classifier's performance on the test data set?

Problem 3: Neural Networks

In this problem, we are going to build a classifier to recognize handwritten digits. The datasets can be downloaded from the course webpage, they are from the UCI Machine Learning Repository:

`http://www.ics.uci.edu/~mlearn/MLSummary.html`

The 32 x 32 bitmaps of handwritten digits are preprocessed and are divided into non-overlapping blocks of 4 x 4 and the number of on-pixels are counted in each block. This generates an input matrix of 8 x 8 where each element is an integer in the range [0..16]. This reduces the dimensionality and gives invariance to small distortions.

The dataset was further processed, to rescale the input elements to a number between 0-1, and to transform the output values into 10 binary outputs. The dataset is available on the course webpage. You get two arrays, *in* and *out*. *in* contains the classification for the examples in *in*, as binary flags with zeros everywhere except for a 1 in the correct position (i.e. 0 0 0 0 0 1 0 0 0 0 is a 6).

For this problem you are free to write your own code or use any MATLAB/Python toolboxes available for the purpose. If you are using MATLAB, you can use the default Neural Network toolbox available. Try *help nnet, help nntool* to help you get started. If you type demo on the MATLAB prompt, under the option Toolboxes, you can select *Neural Networks* to view some examples.

- Train a two-layer neural network with sigmoidal hidden units (i.e. 1-hidden layer). Train the network using the back-propagation algorithm with the provided training set. Test your network and report recognition results.
- Experiment with different numbers of hidden units to optimize recognition accuracy.
- Find a fixed number of hidden units, n that works well for the dataset and report the recognition results.

- Comment on the effects of varying the number of hidden units on recognition accuracy. Suggestion: Plot the average percentage of recognition accuracy as a function of the number of neurons in the hidden layer to support your argument.
- Train a 2-hidden layer model using $n/2$ hidden units in each layer, where n corresponds to the value you found in part (c). Test your network and report recognition results.
- Compare and comment on the results of the 1-hidden layer model (part c) and the 2-hidden layer model (part e)