

## Problem assignment 4

*Due: Wednesday, February 19, 2003*

### Problem 1. Data analysis

The dataset we use in this problem set is very simple and consists of a two-dimensional input and a class label. The data are available on the course web page and are divided into two files - one used for training (*classification\_train.txt*), the other for testing (*classification\_test.txt*). The data in files are in rows such that first two columns represent inputs and the last (third) column the class label (0 or 1).

Since inputs are only two dimensional we can easily visualize the data for the two classes in a 2D plot.

- Write a program that plots the input data points in *classification\_train.txt* such that the plot distinguishes between data points with different class labels (use different color and symbol for a point, e.g. 'x' or 'o').
- Include the plot in your report. Is it possible to separate the two classes perfectly with a linear decision boundary?

### Problem 2. Logistic regression

We are interested in building a classifier based on the logistic regression model and gradient optimization methods.

- (a) During the class you were given the expression for the gradient of the logistic regression model. Use the loglikelihood setup from the lecture to derive the expression. Show clearly the steps of the derivation.
- (b) Write and submit a gradient procedure *GRR.m* for updating the parameters of the logistic regression model. Your gradient procedure should:
  - start from unit weights (all weights set to 1 at the beginning);
  - use the annealed learning rate  $2/k$ ;

- executes for  $K$  steps where  $K$  is the parameter of the procedures.
- (c) Write and submit a program *main2.m* that runs the GLR function for 500 steps and after the training computes the *confusion matrix* and mean misclassification errors for both the training and test set. In your report include, the resulting weights, confusion matrices and misclassification errors.
- (d) Update the *main2.m* with plot functions that let you observe the progress of the errors after every 50 update steps. Use functions defined in PS-3 for this purpose. Include the resulting graph in your report.
- (e) Experiment with the GLR function by: (I) changing the number of steps  $K$  and (II) trying different learning rates. In particular, try some constant learning rates and  $1/\sqrt{k}$  learning rate schedule. Report the results and graph from your experiments and explanations of behaviors you have observed.

### Problem 3. Online gradient descent

- (a) Write an on-line gradient procedure for learning the weight parameters. The procedure should select sequentially examples from the training set, repeating the examples whenever necessary.
- (b) Write a program *main3.m* that performs the on-line gradient method for 500 steps, uses progress plots and at the end computes the confusion matrices and mean misclassifications errors for both datasets.
- (c) Repeat the analysis in problem 2. Compare the results of Problem 2 and Problem 3. What are the differences in the behavior of the two gradient methods.

### Problem 4. Generative classification model

An alternative way to learn a classifier is to learn a generative model with class-conditional densities and use the parameters of such a model to do the prediction.

Assume that an input  $\mathbf{x}$  for each class ( $c = 0$  or  $1$ ) follows a multivariate normal distribution. That is,

$$p(\mathbf{x}|c = 0) \sim N(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$$

$$p(\mathbf{x}|c = 1) \sim N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1).$$

Further assume that the prior probability of a class is represented by a Bernoulli distribution.

Parameters of the generative model can be computed from the training data using density estimation techniques, such as maximum likelihood estimation. Once this is accomplished, we can use the estimates to make class predictions for new inputs. Let  $\tilde{\Theta} =$

$\{\tilde{\mu}_0, \tilde{\Sigma}_0, \tilde{\mu}_1, \tilde{\Sigma}_1, \tilde{\theta}_{c=1}\}$  represent parameter estimates. To predict the class we use discriminant functions based on the posterior probability of a class given the input. and model parameters via Bayes rule, for example:

$$g_1(\mathbf{x}) = p(c = 1 | \mathbf{x}, \tilde{\Theta}) = \frac{p(\mathbf{x} | \tilde{\mu}_1, \tilde{\Sigma}_1) \tilde{\theta}_{c=1}}{p(\mathbf{x} | \tilde{\mu}_0, \tilde{\Sigma}_0) (1 - \tilde{\theta}_{c=1}) + p(\mathbf{x} | \tilde{\mu}_1, \tilde{\Sigma}_1) \tilde{\theta}_{c=1}}.$$

Assume we want to use a generative model in which the two class conditional densities share the same covariance matrix  $\Sigma$ . That is:

$$p(\mathbf{x} | c = 0) \sim N(\mu_0, \Sigma)$$

$$p(\mathbf{x} | c = 1) \sim N(\mu_1, \Sigma).$$

Provide the following answers:

- (a) Give the formula for computing ML estimates of means of class conditional densities?
- (b) How would you go about computing the estimate of the covariance matrix  $\Sigma$ ? Note that the estimate of  $\Sigma$  must combine both class 0 and class 1 examples.
- (c) How would you estimate the prior of class 1  $\tilde{\theta}_{c=1}$ ?
- (d) Implement function *MaxLikelihood* that computes the estimates of model parameters using the training set.
- (e) Implement the function *PredictClass* that chooses the class using the discriminant functions based on class posteriors.
- (f) Write and submit a program *main4.m* that learns the generative model and then uses it to compute the predictions. The program should compute the confusion matrices and mean misclassification errors for both training and testing datasets.
- (g) Report the results (parameters of the generative model), confusion matrices and errors. Compare them to the results obtained in problem 2.