CSC2515 – Assignment #1

Due: Oct19 2004, 2pm at the **START** of class
Worth: 18%
Late assignments not accepted.

1 Training/Testing Error Curves (1.5%)

This question asks you to show your general understanding of underfitting and overfitting as they relate to model complexity and training set size. Consider a continuous domain and a smooth joint distribution over inputs and outputs, so that no test or training case is ever duplicated exactly. Indicate on your vertical axes where zero error is and draw your graphs with increasing error upwards and increasing complexity/training set size rightwards.

- For a fixed training set size, sketch a graph of the typical behaviour of training error rate versus model complexity in a learning system. Add to this graph a curve showing the typical behaviour of test error rate (for an infinite test set drawn independently from the same input distribution as the training set) versus model complexity, on the same axes. Mark a vertical line showing where you think the most complex model your data supports is; chose your horizontal range so that this line is neither on the extreme left nor on the extreme right. Mark a horizontal line showing the Bayes error, which one of the lines should cross.

- For a fixed model complexity, sketch a graph of the typical behaviour of training error rate versus training set size in a learning system. Add to this graph a curve showing the typical behaviour of test error rate (again on an iid infinite test set) versus training set size, on the same axes. Mark a horizontal line showing the Bayes error.

2 Learning Random Boolean Functions (2.5%)

Consider learning random boolean functions under the following setup.

- Target functions with k binary inputs and 1 binary output are generated by randomly assigning outputs (independently with probability one half) to each possible combination of inputs.
- Noiseless training data is generated by randomly selecting a setting of the inputs (uniformly), reporting its output, and repeating this process N times independently.
- Test data inputs are generated by randomly selecting a setting of the inputs (uniformly) and repeating this process M times independently.

Answer the following questions about this setup:

- Let a be the expected number of *distinct* training cases in a training set [Obviously $1 \leq a \leq \min(N, 2^k)$.] Derive an expression for a in terms of k and N.

- What is the expected number b of cases in the test set which *also* appeared in the training set? (In terms of a, k, M.)

- What is the lowest possible expected test set error rate we can hope to achieve? (In terms of b, M.) Give an example of an algorithm which achieves this error rate and an argument (in <25 words) of why no algorithm can do better.
- Does the expected test set error in the above setup measure generalization? Why or why not?

3 Class-Conditional Gaussians (3%)

In this question, you'll derive for yourself the maximum likelihood estimates for class-conditional Gaussians with independent features (diagonal covariance matrices). Start with the following generative model for a discrete class label $y \in (1, 2, \ldots, K)$ and a real valued vector of D features $\mathbf{x} = (x_1, x_2, \ldots, x_D)$:

$$p(y = k) = \alpha_k$$
$$p(\mathbf{x}|y = k) = \left(\prod_{i=1}^{D} 2\pi\sigma_i^2 \right)^{-1/2} \exp\left\{ -\sum_{i=1}^{D} \frac{1}{2\sigma_i^2}(x_i - \mu_{ki})^2 \right\}$$

where α_k is the prior on class k, σ_i^2 are the shared variances for each feature (in all classes), and μ_{ki} is the mean of the feature i conditioned on class k.

- Use Bayes' rule $p(a|b) = p(b|a)p(a)/p(b)$ to invert the model above and write the expression for $p(y = k|\mathbf{x})$. [Hint: remember that $p(\mathbf{x}) = \sum_{k=1}^{K} p(\mathbf{x}|y = k)\alpha_k$.]

- Write down the expression for the likelihood function $\ell(\theta; \mathcal{D}) = \log p(y^1, x^1, y^2, x^2, \ldots, y^M, x^M|\theta)$ of a particular dataset $\mathcal{D} = \{y^1, x^1, y^2, x^2, \ldots, y^M, x^M\}$ with parameters $\theta = \{\alpha, \mu, \sigma^2\}$. (Assume the data are iid.)

- Take partial derivatives of the likelihood with respect to each of the parameters μ_{ki}, with respect to the shared variances σ_i^2, and with respect to the class priors α_k. Since the variances must be positive you might want to take the derivative with respect to their logarithms.

- Set these partial derivatives to zero and solve for the maximum likelihood parameter values μ_{ki}, σ_i^2 and α_k. When solving for the class priors, remember that α_k, must be between 0 and 1 and sum to unity (across k), so you need to use Lagrange multipliers to enforce this constraint.

4 Handwritten Digit Classification (11%)

For this question you will build three classifiers to label images of handwritten digits collected by the United States Post Office. The images are 8 by 8 in size, which we will represent as a vector $\mathbf{x}$ of dimension 64 by listing all the pixel values in raster scan order. The labels y are $1, 2, \ldots, 9, 10$ corresponding to which character was written in the image. Label 10 is used for the digit "0". There are 700 training cases and 400 test cases for each digit; they can be found in the file `a1digits.mat` or `a1digits.zip`. DO NOT HAND IN ANY CODE.

- As a warm up question, load the data and plot a few examples. Decide if the pixels were scanned out in row-major or column-major order, and write that somewhere on your answer to this question.

4.1 K-NN Classifier

- Build a simple K nearest neighbour classifier using dumb Euclidean distance on the raw pixel data.
- Set K to the value which gives the best performance on the *training data*. Report this value of K.
  Don't cheat and look at the test data to set K; that isn't fair and will give you a different value of K.
- Clearly state your policy for breaking ties.

4.2 Conditional Gaussian Classifier Training

- Using maximum likelihood, fit a set of 10 class-conditional Gaussians with a separate, full covariance matrix for each class. In particular, fit the model below to maximize the average of $\log p(\mathbf{x}, y)$ on the training set (where D is the dimension of $\mathbf{x}$ (64 in our case) and $|M|$ is the determinant of the matrix M).

$$p(y = k) = \alpha_k$$
$$p(\mathbf{x}|y = k) = (2\pi)^{-D/2}|\Sigma_k|^{-1/2} \exp\left\{-\frac{1}{2}(\mathbf{x} - \mu_k)^\top \Sigma_k^{-1}(\mathbf{x} - \mu_k)\right\}$$

- You should get parameters μ_{kn} and Σ_k for $k \in (0 \ldots 9), n \in (1 \ldots 64)$.
  (The ML estimates for α_k=1/10 since all classes have the same number of observations in this training set.)
- After fitting the Gaussians, regularize each class' covariance matrix by adding a small amount of the identity matrix. (For this assignment, add $0.01I$ to the sample covariance matrix.)
- Hand in plot showing an 8 by 8 image of each mean μ_k, and below the mean another image showing the log of the diagonal elements of the covariance matrix Σ_k. Plot all ten classes side by side, and the same grayscale for all 10 means. Use another grayscale for the log variances, again keeping it the same for all 10. Indicate in some way the grayscales you used, and indicate what white and black mean.

4.3 Naive Bayes Classifier Training

- Convert the real-valued features $\mathbf{x}$ into binary features $\mathbf{b}$ by thresholding: b_n=1 if $x_n > 0.5$ otherwise $b_n = 0$.
- For smoothing (regularization) purposes, add two training cases to each class, one which has every pixel off and one which has every pixel on.
- Using these new binary features $\mathbf{b}$ and the class labels, train a Naive Bayes classifier. In particular, fit the model below to maximize the average of $\log p(\mathbf{b}, y)$ on the training set (including the extra smoothing examples).

$$p(y = k) = \alpha_k$$
$$p(b_n = 1|y = k) = \eta_{kn}$$
$$p(\mathbf{b}|y = k, \eta) = \prod_n (\eta_{nk})^{b_n}(1 - \eta_{nk})^{(1-b_n)}$$

- You should get parameters $\eta_{kn} \equiv p(b_n = 1|y = k)$ for $k \in (0 \ldots 9), n \in (1 \ldots 64)$.
  (Again, all class priors α_k are equal since all classes have the same number of observations.)
- Hand in plot showing an 8 by 8 image of each vector $\log \eta_k$, all ten side by side sharing a single grayscale. Indicate in some way the grayscale you used, and indicate what white and black mean.

4.4 Performance Evaluation

- Using the K you found, for each training and each test case, run the K-NN classifier and (using your rules for tie breaking if necessary) label each case according the the majority class of its K neighbours. If this matches the true label, the classifier is correct. If not, the classifier has made an error.
- Using the parameters you fit on the training set compute $\log p(y|\mathbf{x})$ for each of the training and test cases under both the Gaussian-conditional and Naive Bayes models.
- What is the average conditional log likelihood achieved by each of these two classifiers on the training set? On the test set? Average both over data cases and digit classes.
- Select the most likely posterior class for each training and test case under each of these two classifiers. If this matches the label, the classifier is correct. If not, the classifier has made an error.
- Hand in a 3 column by 11 row table showing how many errors (out of 400) each classifier makes on each of the 10 test sets and what the overall training and testing error rate (in %) is.
- Hand in an identical table showing the performance on the training set.

4.5 Tips

If you are using MATLAB , here are some tips:

- The `imagesc` function can be used to display vectors as images. In particular, try the line:
  `imagesc(reshape(xx,8,8)'); axis equal; axis off; colormap gray;`
  to display the vector `xx`.
- The `subplot` command is useful for displaying many small images beside each other. The `caxis` and `colorbar` commands can set the grayscale and display it as a shaded bar.
- The `repmat` command in conjunction with `sum` and the operators .* and ./ are helpful in renormalizing arrays so that the rows or columns sum to one.
- The expression `(M > a)` for a matrix `M` and a scalar `a` performs the comparison at every element and evaluates to a binary matrix the same size as `M`.