CSC2515 – Assignment #1

Due: Oct18 2005, 2pm at the **START** of class
Worth: 18%
Late assignments not accepted.

1 Generalization and Model Complexity (3%)

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 one-dimensional classification problem in which $p(y = 0) = p(y = 1) = 1/2$ and $p(x|y = k)$ is uniformly distributed between ℓ_k and u_k. Assume $\ell_0 < \ell_1 < u_0 < u_1$.
  - What is the Bayes-optimal classification rule?
  - What error rate does it achieve (ie what is the Bayes error)?
  - What is the probability that for a training set of size N drawn from this model, the Bayes classifier has training error zero?
  - [Harder.] Answer the same questions if $p(x|y = k)$ is a Gaussian with mean μ_k and variance σ_k^2. (You'll need to express your answer in terms of the `erf` function.)
- Consider a continuous domain and a smooth joint distribution over inputs and outputs, so that no test or training case is ever duplicated exactly.
  - 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 the corresponding test error rate (for an infinite test set drawn independently from the same joint 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. Indicate on your vertical axes where zero error is and draw your graphs with increasing error upwards and increasing complexity rightwards.
  - 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. Indicate on your vertical axes where zero error is and draw your graphs with increasing error upwards and increasing training set size rightwards.
  - For a fixed range of model complexity (from very simple to very complex), sketch a graph of training set size versus *the model complexity which achieves the best test performance* (on an iid infinite test set).

2 Naive-Bayes for Continuous Inputs (3%)

In this question, you'll derive for yourself the maximum likelihood estimates for the Naive-Bayes classification model when the inputs are continuous values modeled by Gaussian distributions.

Recall that Naive-Bayes makes the assumption that the features are conditionally independent given the class. Hence, for a discrete class label $y \in (1, 2, \ldots, K)$ and a real valued vector of D features $\mathbf{x} = (x_1, x_2, \ldots, x_D)$ we have the following model:

$$p(y = k) = \alpha_k$$

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

where α_k is the prior on class k, and μ_{ki},σ_{ki}^2 are the means,variances of feature i given that the data is in class 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} and σ_{ki}^2. Since the variances are positive quantities, things will be easier for you if you 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_{ki}^2.
- If you are feeling ambitious, also take partial derivatives with respect to the class priors α_k and find their optimal values. [Remember that α_k, must be between 0 and 1 and sum to unity (across k), so you'll need to use Lagrange multipliers to enforce this constraint. Don't waste too much time on this question, but if you can't do it, you probably need to brush up on your calculus.]
- This classifier is not (in general) linear in the input features. What additional constraint would you have to add to the parameters in order to make it linear? After this constraint has been added, how is the model different from Fisher's Discriminant (in the two-class case)?

3 Spam Email Classification (12%)

For this question you will build three classifiers to label emails (from Prof. Roweis' archives) as either spam or non-spam ("ham"). DO NOT HAND IN ANY CODE. Seriously, we are not joking. We don't want to read your code, we just want it's output. (Perhaps in the same way that you don't want to read the schematics for your MP3 player but still want access to its output.)

Thanks to your hard working TAs, the emails have already been converted into a very convenient representation in which each email is a vector of 185 binary features. Most of these features indicate whether or not particular words occurred in the email or not; a few of them indicate things like capitalization, presence of attachments, etc. The complete list of features and what they represent can be found in the file `a1features.txt`. The label y takes on two values, one corresponding to ham and the other to spam. There are 1000 training cases and 4000 test cases; they can be found on the course website in the file `a1spam.mat` (Matlab V7), `a1spamv6.mat` (Matlab V6) or `a1spam.zip` (zipped ascii).

- To get started, load the data into your favourite computing environment and determine by eye which class label corresponds to spam and which one corresponds to ham. (This should be quite easy if you take a look at specific features from the wordlist, such as "viagra".) Write the answer somewhere on what you hand in.

3.1 K-NN Classifier

- Build a simple K nearest neighbour classifier using dumb Euclidean (Hamming) distance on the binary features. To classify a test point (or a point being held out), label the point according the the majority class of its K nearest neighbours. If this matches the true label, the classifier is correct. If not, the classifier has made an error. If there is a tie, chose the class which was most common in the training data.
- Since the inputs are binary, distances are quantized and so when you find the K nearest neighbours there might actually be several neighbours at exactly the same distance as the K^{th} neighbour. In such cases, you can just select between these equidistant neighbours randomly.
- Try all values of K from 1 to 20. Hand in a plot showing the *training error* (as measured by leave-one-out performance) and the *test error* as a function of K.
- Report the optimal value of both the training and the testing error and the K at which each of these occur.

3.2 Naive Bayes Classifier

- For smoothing (regularization) purposes, add two extra training cases to each class, one which has every feature off and one which has every feature on.
- Using this extended training set, train a Naive Bayes classifier on the features $\mathbf{x}$. In particular, fit the model below to maximize the average of $\log p(\mathbf{x}, y)$ on the training set (including the extra smoothing examples, which will cause you to add one to all your counts).

$$\begin{aligned} p(y = k) &= \alpha_k \\ p(x_i = 1 | y = k) &= \eta_{ki} \\ p(\mathbf{x} | y = k, \eta) &= \prod_i (\eta_{ki})^{x_i} (1 - \eta_{ki})^{(1-x_i)} \end{aligned}$$

- You should get parameters $\eta_{ki} \equiv p(x_i = 1 | y = k)$ for $k \in \{0, 1\}$ and $i \in \{1, \ldots, 185\}$ and parameters α_k for $k \in \{0, 1\}$.
- Sort the features by the value of $\log(\eta_{0i}/\eta_{1i})$ and produce two lists of the 5 features with the largest (most positive) and the 5 features with the smallest (most negative) log ratios.
- Using the parameters you fit on the training set compute $\log p(y|\mathbf{x})$ for each of the training and test cases. Select the most likely posterior class for each training and test case; if this class matches the label, the classifier is correct. If not, the classifier has made an error. Report the average training and average test error rates.

- What is the average conditional log likelihood $\log p(y|\mathbf{x})$ achieved on the spam training/testing data? On the ham training/testing data?

3.3 Logistic Regression Classifier

- Using maximum conditional likelihood, fit a logistic regression classifier to the features, after augmenting the vector $\mathbf{x}$ with a constant (bias) term equal to $+1$. The logistic regression model is:

$$p(y = k|\mathbf{x}, \theta) = \frac{e^{\theta_k^\top \mathbf{x}}}{\sum_j e^{\theta_j^\top \mathbf{x}}}$$

  Fit the model above to maximize the average of $\log p(y|\mathbf{x})$ on the training set, minus 0.5 times the sum of the squares of the weights ($\sum_{k,i} \theta_{ki}^2$).

- You should get parameters θ_{ki} for for $k \in \{0, 1\}$ and $i \in \{0, \ldots, 185\}$ (i=0 is the constant or bias term).

- Sort the features by the value of $(\theta_{0i} - \theta_{1i})$ and produce two lists of the 5 features with the largest (most positive) and the 5 features with the smallest (most negative) differences.

- Remember that you cannot find the model parameters explicitly in closed form; instead you need to compute the gradient of the average conditional log likelihood with respect to θ and use that gradient to numerically maximize the objective. You are free to do this maximization in any way you choose, but the following adaptive gradient procedure, starting with random parameters, should work pretty nicely:

  1. Measure the current objective and gradient.
  2. Adjust the parameters by subtracting Δ times the gradient.
  3. Measure the new objective and new gradient.
  4. If the new objective is better, set $\Delta = 1.1\Delta$
     else if the new objective is worse, go back to the old parameters, objective and gradient, set $\Delta = .5\Delta$.
  5. If $\Delta > \Delta_{min}$, go to (2).