CSCC11 Assignment 2: Probability, Estimation, and Classification

Written Part (5%): due Friday, October 7th, 11am

In the following questions be sure to show all your work. Answers without derivations will be given no marks.

Probability

1. Consider a test which detects if a person has a disease. Let θ_1 be the probability that it reports that someone has the disease ($R = 1$) when they actually do ($D = 1$). Let θ_0 be the probability that it reports that someone does not have the disease ($R = 0$) when they indeed do not ($D = 0$). That is:

$$\begin{aligned} P(R{=}1 \,|\, D{=}1) &= \theta_1 \\ P(R{=}0 \,|\, D{=}0) &= \theta_0 \end{aligned}$$

Also, suppose that the proportion of people that have this disease is a number $\alpha \in [0, 1]$.

   a) A patient goes to the doctor and has the test has performed and it comes back positive. Derive the posterior probability that the person actually has the disease, and simplify it in terms of θ_1, θ_0 and α.

   b) After the results of the first test come back positive, the doctor runs it a second time. This time it comes back negative. Derive the posterior probability that the person actually has the disease after this second round of testing and simplify in terms of θ_1, θ_0 and α.

   c) Suppose $\theta_1 = .999$, $\theta_0 = .99$ and $\alpha = 0.001$. Suppose 1000 patients get tested, and they're all negative. What is the expected number of false negatives?

2. A bag contains four dice: a 4-sided die, two 8-sided dice and a 12-sided die. A person chooses a die from the bag at random and rolls it. The number of faces that the selected die has is denoted by the random variable S and the number rolled by the random variable X.

   a) Derive the probability distribution of the number rolled with the selected die. Use your expression for $p(X)$ to determine the probability that the first number rolled is 3. That is, what is $p(X = 3)$?

   b) Given that the roll was a 3, what's the posterior probability that the selected die had 12 sides? That is, what is $p(S = 12|X = 3)$?

   c) Suppose the person rolls the same die a second time. Denote the random variable of this second roll by Y. If the second roll comes up a 7, how does this change the posterior probability that the selected die has 12 sides? Specifically, what is $p(S = 12|X = 3, Y = 7)$?

d) Suppose the person rolls the same die a third time after telling you that the first two rolls were 3 and 7. Denote the third roll by Z. What is the predictive probability distribution of seeing 2 on this final roll given the knowledge of the first two rolls? I.E., what is $p(Z = 2|X = 3, Y = 7)$?

Estimation

3. A lot of real world process has a property called *Heteroscedasticity*, which intuitively means that the level of variability is not constant through the input space (unlike the constant noise in LS regression that we have seen so far). One example of such process is in the motor system of our body, where faster motion is noisier and less controllable. If you are not convinced, try to connect two points on a paper with straight line by one fast stroke without overshooting, then try the same but more slowly.

Now suppose that we have some measurements of a motor process, $\{x_i, y_i\}_{i=1}^N$, where x_i, y_i's are scalers, and x_i is some motor command and y_i is the corresponding result. We will use a set of nonlinear basis functions $\{b_j(x)\}_{j=1}^K$, along with weight vector $\mathbf{w} \in \mathbb{R}^K$ to model the nonlinear mapping between x and y. To model the heteroscedasticity in this problem, we will assume a Gaussian noise model whose standard deviation scales with $|x| + 1$, where $|x|$ denotes the absolute value of x. Putting everything together, the model can be expressed as:

$$f(x) = \mathbf{w}^\top \mathbf{b}(x) + (|x| + 1)\epsilon$$

where $\epsilon \sim N(0, \sigma^2)$, and $\mathbf{b}(x)$ is a column vector whose j-th entry is $b_j(x)$.

Furthermore, assume a prior over $\mathbf{w}$ that treats each dimension as independent with a Gaussian distribution, i.e. $p(w_j) = \frac{1}{\sqrt{2\pi\alpha}} \exp(\frac{-w_j^2}{2\alpha})$.

Derive the MAP estimate of $\mathbf{w}$ given training data $\{x_i, y_i\}_{i=1}^N$, express the final result with only matrices and vectors, no summation sign. Define all matrices, vectors and other symbols clearly, showing dimensions whenever necessary.

Submission

Answers to the above questions should be submitted at the beginning of class on the due date.

Programming Part (8%): ~~due Friday October 21st, 11am~~ due Tues Oct 25th, 4pm (IC 4th floor C11 dropbox)

Classification

For this problem, you will implement three basic classification algorithms:

- KNN: k-Nearest Neighbors
- GCC: Gaussian class-conditional models
- LR: Logistic regression (we will provide gradient descent code for the LR estimation)

They will be applied to four data sets. Each data set comprises two parts, the training set and the test set. You should train your algorithms on the training data, and evaluate their performance on the test set. For all experiments, the test performance will be evaluated as average 0-1 loss on the test set. In other words, once you have fit a model $f(x)$ to the training data, evaluate it by counting the number of correctly labelled test points, for which $y_i = f(x_i)$, and then dividing by N.

Data Sets

In the starter zip file available on the course web site there is a `dataset` directory. In that directory there are several mat files for the different datasets, as described below. To plot the data, see the script `plotDataSets.m` in that directory. The individual datasets are described below.

Fruit: The 2D measurements (features) are the heights and widths of oranges and lemons (in cm to the nearest mm). There are two files, `fruit_train.mat` and `fruit_test.mat`. The training file contains two arrays, `inputs_train` and `target_train`. The array `inputs_train` is 2×30, with height and width measurements for 30 training exemplars. The array `target_train` is also 2×30, with one column for each training exemplar, being (1,0) for oranges and (0,1) for lemons. The test data file, `fruit_test.mat`, is structured in a similar fashion, comprising 10 test instances.

Digits: The Digits datasets are for training classifiers to recognize handwritten digits, 4s and 9s, from small images. Each image is a 28×28 array of real-valued intensities (greylevels). In the dataset files, each image is represented as a vector of length 784 (the images are converted from arrays to vectors by stacking the columns into one column vector).

There are two files, `mnist_train.mat` and `mnist_test.mat`. The traning file contains two arrays, `inputs_train` and `target_train`. The array `inputs_train` is 784×60, representing 60 training exemplars. The array `target_train` is 2×60, with one column for each training exemplar, being (1,0) for 4s and (0,1) for 9s. The test data file is structured in a similar fashion, comprising 200 test instances.

Generic1/2: There are two other generic datasets, `generic1.mat` and `generic2.mat`, both of which have the same basic format. The measurements are 2D and real-valued. Each matlab file comprises four arrays, namely, `c1_train` and `c2_train` comprising 21 training exemplars for class 1 and class 2, and `c1_test` and `c2_test` comprising 69 instances of class 1 and class 2 for testing.

Implementing Classifiers

As described above there are three classifiers to be implemented. Some initial code to get you started is available in the directory named `starterCode` in the starter file from the course web site.

k-Nearest Neighbours: There is an m-file named `knnClassify.m` which provides the basic template for the knnClassification function you need to implement. In addition, there is a script named `knnStarter.m` in which you can do much of the performance testing.

Gaussian Class-Conditional Models: For your implementation of the GCC model, we have provided two template m-files, one for learning the GCC model, `learnGCCmodel.m`, and one for classification of test instances, `gccClassify.m`. Your implementation will require writing code to complete these two functions. In addition, there is a script named `gccStarter.m` in which you can do much of the performance testing.

Logistic Regression: You should implement the regularized logistic regression model explained in Chapter 8 of the online class notes. To do so, you need derive the gradient of weight and bias with respective to the regularized loss of logistic regression:

$$E(\mathbf{w}, b) \quad = \quad -log \left[p(\mathbf{w}, b) \prod_{i=1}^{N} p(y = y_i \,|\, \mathbf{w}, b, \mathbf{x}_i) \right]$$

where

$$p(\mathbf{w}, b) \quad \propto \quad (2\pi\alpha)^{-\frac{D}{2}} \exp\left(-\frac{\mathbf{w}^T \mathbf{w}}{2\alpha} \right)$$

Hints given in lecture should help you deriving the correct expressions for gradients. To get you started on programming, there are several m-files in the starter zip file and a script named `logisticStarter.m` in which you can do much of the performance testing.

- `logistic.m` evaluates the logistic regression model given an array of exemplars and the weight parameters. It computes the posterior class probability given the data and model parameters. This function is used mainly for classification and has already been implemented for you.
- `logisticNLP.m` computed the negative log likelihood of the input data, and the gradient of the negative log likelihood. It is essential for learning the logistic regression model parameters. Based on hints given in class and Chapter 8 of the notes you should derive the gradients and implement this function.
- `learnLogReg.m` is a template for a gradient descent method, with line search, for minimizing the negative log likelihood of the data. It uses `logisticNLP.m` extensively, and it provides a facility to check your gradients numerically. But most of the code for this function is already written. You only have to modify a small amount of code here.

Performance Testing

To evaluate these three different methods, you will learn the models with different parameter settings, and compute the test errors for them on different datasets. Make use of MATLAB scripts and the functions above to help automate these processes.

Generic1/2 Begin your testing on the generic 2D datasets. In doing so, address the following questions and produce the plots requested (all plots should be clearly labeled with the MATLAB commands legend, xlabel, ylabel, title):

1. For KNN, plot the test error as a function of k. Use $k = 3, 5, 7, ..., 21$.
2. For LR, plot of the test error as a function of the regularization weight. Use $\alpha = 0.25, 0.5, 1, 2$, and 5.
3. For one instance of KNN (your choice of k), one instance of LR (your choice of α), and the GCC model, plot the model fit to the data. To this end you could plot the decision boundaries, and/or you could plot which training and test points are correctly or incorrectly labelled.
4. Provide answers, with explanations, for the following questions: Which of these methods worked best for this data set? Which seemed easiest to use? Which were easiest to understand?

Fruit: Apply all three algorithms to the fruit data. Which of these methods appeared to work best for this data set? Why?

Digits: Apply kNN and LR to this dataset. Which method appears to work best for this data set? Why? What problems can you foresee if you were to try to apply GCC to this data?

What to submit

Submit an electronic version of your solution by creating a tar file comprising the scripts and Matlab function files. This tar file should be named `A2.tar` and submitted it according to instructions on the course web site by the due date.

You also need to submit a printed version of this part of your assignment. It should include:

- a printed version of your Matlab scripts and function files,
- printouts of the plots described above with their axes clearly labeled and a title indicating which plot it is, and
- answers to the questions described above.

Your paper copy should include a coversheet with the course number, assignment number, your name, student number and mathlab username. Hand in any written or formatted work for the assignment at the beginning of class on the due date. For the electronic version of your solution, create a tar file comprising all Matlab scripts and function files (but not the dataset directory). Call the tar file `A2.tar`, and submit it according to instructions on the course web site (again, by the due date).