Project 1 Naive Bayes and Logistic Regression

In this project you will code up two of the classification algorithms covered in class: Naive Bayes and Logistic Regression. The framework code for this question can be downloaded from CANVAS.

- **Programming Language:** You must write your code in R.
- **Submission Instructions:** For each sub-question you will be given a single function signature. You will be asked to write a single R function which satisfies the signature. In the framework code, we have provided you with a R script for the functions you need to complete. *Do not change the structure of the file.* Complete each of these functions, and compress the code and the results files, evaluation.t xt *as a* .tar *file* and submit it to Canvas. You may submit it multiple times. Each submission will overwrite the previous submission. Only the last submission before the deadline will be graded.
- **Presentation slides:** Make slides to summarize your results. You do not need submit the slides, but I will randomly draw a couple of groups to present their slides in class.
- **SUBMISSION CHECKLIST**
  - Submission executes in less than 20 minutes.
  - Submission is smaller than 100K.
  - Submission is a .tar file.
  - Submission returns matrices of the *exact* dimension specified.
- **Data:** All questions will use the following datastructures:
  - $xTrain \in \mathbb{R}^{n \times f}$ is a matrix of training data, where each row is a training point, and each column is a feature.
  - $xTest \in \mathbb{R}^{m \times f}$ is a matrix of test data, where each row is a test point, and each column is a feature.
  - $yTrain \in \{1, ..., c\}^{n \times 1}$ is a vector of training labels
  - $yTest \in \{1, ..., c\}^{m \times 1}$ is a (hidden) vector of test labels.

1 Logspace Arithmetic [10 pts]

When working with very small and very large numbers (such as probabilities), it is useful to work in *logspace* to avoid numerical precision issues. In logspace, we keep track of the logs of numbers, instead of the numbers themselves. (We generally use natural logs for this). For example, if $p(x)$ and $p(y)$ are probability values, instead of storing $p(x)$ and $p(y)$ and computing $p(x) * p(y)$, we work in log space by storing $\log p(x), \log p(y), \log[p(x) * p(y)]$, where $\log[p(x) * p(y)]$ is computed as $\log p(x) + \log p(y)$.

The challenge is to add and multiply these numbers *while remaining in logspace*, without exponentiating. Note that if we exponentiate our numbers at any point in the calculation it completely defeats the purpose of working in log space.

1. **Logspace Multiplication [5 pts]**
   Complete logProd=function(x) which takes as input a vector of numbers in logspace (i.e., $x_i = \log p_i$), and returns the product of these numbers in logspace – i.e., $\mathsf{logProd}(x) = \log \prod_i p_i$.
2. **Logspace Addition [5 pts]**
   Complete logSum=function(x) which takes as input a vector of numbers in logspace (i.e., $x_i = \log p_i$), and returns the sum of these numbers in logspace – i.e., $\mathsf{logSum}(x) = \log \sum_i p_i$.

2 Gaussian Naive Bayes [25 pts]

We will download the *Ecoli* dataset from CANVAS and load it using R command in Project1.R file. You will implement the Gaussian Naive Bayes Classification algorithm. As a reminder, in the Naive Bayes algorithm we calculate $p(c|f) \propto p(f|c)p(c) = p(c)\prod_i p(f_i|c)$. In Gaussian Naive Bayes, we learn a one-dimensional Gaussian for each feature in each class, i .e. $p(f_i|c) = \mathcal{N}(f_i; \mu_{i,c}, \sigma^2_{i,c})$, where $\mu_{i,c}$ is the mean of feature f_i for those instances in class c, and $\sigma^2_{i,c}$ is the variance of feature f_i for instances in class c. You can (and should) test your implementation locally using the $xTrain$ and $yTrain$ data provided.

1. **Training Model - Learning Class Priors** [5 pts]
   Complete the function prior=function(yTrain). It returns a $c \times 1$ vector p, where p_i is the prior probability of class i.

2. **Training Model - Learning Class-Conditional Feature Probabilities** [8 pts]
   Complete the function likelihood=function(xTrain, yTrain). It returns two matrices, M and V. M is an $m \times c$ matrix where $M_{i,j}$ is the conditional mean of feature i given class j. V is an $m \times c$ matrix where $V_{i,j}$ is the conditional variance of feature i given class j.

3. **Naive Bayes** Classifier [8 **pts**]
   Complete the function naiveBayesClassify=function(xTest, M, V, p). It returns a vector t, which is a $m \times 1$ vector of predicted class values, where t_i is the predicted class for the ith row of $xTest$.

4. **Evaluation** [4 pts]
   Let's analyze the accuracy of the classifier on the test data. Create a text file **evaluation.txt**. Each on a separate line, report the evaluation metric in decimal format, to 3 decimal places.

   - Fraction of test samples classified correctly
   - Precision for class 1
   - Recall for class 1
   - Precision for class 5
   - Recall for class 5

3 Logistic Regression [25 pts]

In this question you will implement the Logistic Regression algorithm. You will learn the weights using Gradient Descent. Once again you can test your implementation locally using the $xTrain$ and $yTrain$ data provided.

1. **Sigmoid Probability** [7 pts]
   Complete the function sigmoidProb = function(y, x, w), where $y \in 0, 1$ is a single class, x is a single training example, w is a weights vector. The function returns a value $p = p(y|x)$.

2. **Training Logistic Regression** [7pts]
   Complete the function logisticRegressionWeights=function(xTrain, yTrain, w0, nIter), where $w0$ is the initial weight value and $nIter$ is the number of times to pass through the dataset. It outputs a f × 1 weights vector w. You can use step-size=0.1 in this question.

3. **Logistic** Regression Classifier [7**pts**]
   Complete the function logisticRegressionClassify=function(xTest, w), where w is a $f \times 1$ weights vector. The output should be a single binary value indicating which class you predict.

4. Evaluation [4 pts]
   Evaluate the accuracy of the classifier on the Ecoli dataset as in Question 2. Report your results in the file evaluation.txt, compare with the results from Question 2, and comment on the comparison.