CS 443 – Introduction to Data Science
Spring 2015 – Homework 2 – 100 points
Due 11:59pm (Eastern) March 23, 2013

This homework consists of problems covering data streams, decision trees, naïve Bayes, and evaluation techniques.

A few instructions to make life easier for all of us:

- Turn in your answers in a single file, called <Last-Name><First-Name>HW2.pdf, by uploading it into Sakai.
- Question 8 requires you to also upload the following 6 Matlab files into the Sakai:
  1. **train_lr.m**
  2. **test_lr.m**
  3. **train_gnb.m**
  4. **test_gnb.m**
  5. **transform_std.m**
  6. **transform_log.m**
- Please write concisely and clearly. There are points for intermediate steps, but not in "talking problems to death."
- Have the following at the top of every page: <Last-name>, <First-name> [<page-number>/<number-of-pages>] Example: Smith, John [1/10]
- Your solution should have a cover-page that provides the following information:

<First-name> <Last-name>

Problem Number	Solution Page
1a	
1b	
…	

  If you did not answer a problem, then enter "not done" instead of the solution page.
- It is highly recommended that you turn-in a typed copy of your homework (as opposed to scanned hand-written copy). This is especially true for equations and plots.
- As always, our TA is here to help you with any doubt or confusion that you may have.

Q1. [5 points] Suppose we hash the elements of a set S having 20 members, to a bit array of length 99. The array is initially all-0's, and we set a bit to 1 whenever a member of S hashes to it. The hash function is random and uniform in its distribution. You may assume that 99 is large.

1a. [3 points] What is the expected fraction of 1's in the array after hashing? Give your answer to three decimal places.

1b. [2 points] What is the expected fraction of 0's in the array after hashing? Give your answer to three decimal places.

Q2. [5 points] A variant of a Bloom filter uses a single hash function that generates several distinct bit numbers. We shall consider this variant in a tiny example. It uses an array of 11 bits and a hash function h that generates two distinct bit numbers. We want to test membership in a set S of three elements, so we hash each of the three elements using h, and we set to 1 the two bits that any of the three elements is hashed to by h.

When a new element x arrives, we compute $h(x)$, and we say x is in the set if both these bits are 1. Assume x is not in the set S. What is the probability of a false positive (i.e., the probability of saying that x is in S)? Give your answer to two decimal places.

Q3. [9 points] Suppose we wish to simulate the "popular elements" algorithm of Section 4.7.3 of http://infolab.stanford.edu/~ullman/mmds/ch4.pdf, which uses an exponentially decaying window on a stream. Recall that for a given decay parameter c and a threshold t, the algorithm is as follows:

1. For each element whose score we are currently maintaining, multiply its score by $(1–c)$.
2. If there exists a score for the newly arrived element, then add 1 to its score. Otherwise, create a score and initialize it to 1.
3. If any score is below the threshold t, drop that score.

The sequence of elements arriving on our stream is:

a b c a d e a c b b
^ end of the stream

Suppose we use an unrealistically large decay parameter $c = 0.1$ and a threshold $t = 0.6$.

3a. [3 points] Is there an element at the end of the stream whose popularity is below the threshold of 0.6 (and is therefore dropped)? If so, which element is it **and** what is its popularity score at the time it is dropped? When computing scores you should round to 4 decimal places after each step.

3b. [6 points] At the end of the stream, what is the most popular element **and** what is its popularity score? When computing scores you should round to 4 decimal places after each step.

Q4. [16 points] We wish to use the Flajolet-Martin Algorithm (described in Section 4.4.2 of http://infolab.stanford.edu/~ullman/mmds/ch4.pdf) to count the number of distinct elements in a stream. Suppose that there are ten possible elements, 1, 2, …, 10, that could appear in the stream, but only four of them have actually appeared. To make our estimate of the count of distinct elements, we hash each element to a 4-bit binary value. Specifically, the element x is hashed to $h(x) = (3x + 7) \pmod{11}$ and

converted into a 4-bit binary value. For example, element 8 hashes to (3×8 + 7)(mod 11) = 31 (mod 11) = 9. Thus, the 4-bit hashed value for element 8 is 1001.

A set of four of the elements 1 through 10 could give an estimate that is exact (i.e., 4 distinct elements have appeared), or too high, or too low. List a set of four elements that gives the exact estimate. Briefly describe why this set gives the exact estimate of having seen 4 distinct elements.

Q5. [3 points] Give a decision tree that represents the following Boolean function:

$$A \vee (B \wedge C)$$

Q6. [18 points] Consider the following set of training examples for the unknown target *Y*.

Y	X_1	X_2	Count
+	T	T	3
+	T	F	4
+	F	T	4
+	F	F	1
–	T	T	0
–	T	F	1
–	F	T	3
–	F	F	5

Each row indicates the values observed, and how many times that set of values was observed. For example, (+, *T*, *T*) was observed 3 times, while (–, *T*, *T*) was never observed.

6a. [3 points] Compute the sample entropy $H(Y)$ for this training data (with logarithms base 2)? Give your answer to four decimal places.

6b. [6 points] What are the information gains IG(X_1) and IG(X_2) for this sample of training data? Give your answers to four decimal places.

6c. [8 points] Draw the decision tree that would be learned by ID3 (the method described in class; without post-pruning) from this sample of training data. Make sure to label the leaf nodes with the target value.

6d. [1 points] Given the decision tree from 6c, how probable is a $Y = +$ for $X_1 = \text{F}$ and $X_2 = \text{T}$? Give your answer to four decimal places.

Q7. [14 points] Suppose we play a game where I present to you three doors, one of which has a prize behind it. The doors are closed and so you choose a door which you think conceals the prize. After you make your choice, I open one of the two doors you didn't pick, and reveal that the prize wasn't there. Then I give you the choice whether to stick with your current door, or switch to the remaining un-opened door. What should you do to have the highest probability of winning the prize? (Hint, consider the event that your initial door conceals the prize, then consider the probability that you win given that you decide to switch or not).

Q8. [30 points] Programming assignment. Matlab is a high-level programming language and is very popular for quickly coding/prototyping machine-learning algorithms. For this question, you will use the iLab machines, which have Matlab installed on them. You can find lots of good quick tutorials on Matlab on the Web via a Google search. Examples include these:

- http://web.eecs.umich.edu/~aey/eecs451/matlab.pdf
- http://www.cs.cmu.edu/~guestrin/Class/10701/recitations/matlab/pretty_matlab_pres.pdf
- http://www.math.utah.edu/~eyre/computing/matlab-intro/
- http://www.mathworks.com/help/pdf_doc/matlab/getstart.pdf

To make the problem more tractable, we are providing a file called **`run_experiment.m`**, which is the matlab script that preprocesses the data, trains the classifier, tests the classifier, and reports the results.

8a. [10 points] In Matlab, implement a logistic regression with l_2 regularization for binary classification. The file **`run_experiment.m`** has code that uses 5-fold cross-validation to choose the strength of the logistic regression l_2 regularizer (denoted by λ). Turn in your code (by uploading it into Sakai) in two files called **`train_lr.m`** and **`test_lr.m`**. Hint: The headers for your functions should be as follows:

- **`function [w,loglike] = train_lr(X,y,w0,eps,lambda,eta,maxitr)`**
  - **`train_lr`** trains the logistic regression classifier. It takes as input a train data matrix **`X`**, a train target vector **`y`**, the initial weight vector **`w0`**, the threshold **`eps`**,[1] the regularizer **`lambda`**, the learning rate **`eta`**, and the maximum number of gradient ascent iterations **`maxitr`**. The function outputs the learnt weight vector **`w`** and the log likelihood value **`loglike`**.
- **`function [ypred,tp,tn,fp,fn] = test_lr(Xtest,ytest,w)`**
  - **`test_lr`** tests the logistic regression classifier. It takes as input a test data matrix **`Xtest`**, a test target vector **`ytest`**, and the learnt weight vector **`w`**. The function outputs the vector of predictions **`ypred`**, and the values for true positive (**`tp`**), true negative (**`tn`**), false positive (**`fp`**), and false negative (**`fn`**).

8b. [10 points] In Matlab, implement a Gaussian Naïve Bayes for binary classification. Use MLE to estimate the parameters. Turn in your code (by uploading it into Sakai) in two files called **`train_gnb.m`** and **`test_gnb.m`**. Hint: The headers for your functions should be as follows:

- **`function [mu,stdev,pr1] = train_gnb(X,y)`**
  - **`train_gnb`** trains the Gaussian Naïve Bayes classifier. It takes as input a train data matrix **`X`** and a train target vector **`y`**. The function outputs two matrices **`mu`** and **`stdev`**. The matrix **`mu`** holds the mean values for the features per class, so its dimensions are (# of classes) × (# of features). The matrix **`stdev`** holds the standard deviation values for the features per class, so its dimensions are also (# of classes) × (# of features). The function also outputs the prior probability of the positive class (i.e., prior probability of target = 1) in **`pr1`**.
- **`function [ypred,tp,tn,fp,fn] = test_gnb(Xtest,ytest,mu,stdev,pr1)`**
  - **`test_gnb`** tests the Gaussian Naïve Bayes classifier. It takes as input a test data matrix **`Xtest`**, a test target vector **`ytest`**, the matrices **`mu`** and **`stdev`**, and the prior probability of

[1] Recall that if the weight vector between gradient ascent iterations does not change by more than epsilon, then we declare that the gradient ascent has converged.

the positive class **pr1**. As in **train_gnb**, the matrix **mu** holds the mean values for the features per class, so its dimensions are (# of classes) × (# of features). The matrix **stdev** holds the standard deviation values for the features per class, so its dimensions are also (# of classes) × (# of features). The function outputs the vector of predictions **ypred**, and the values for true positive (**tp**), true negative (**tn**), false positive (**fp**), and false negative (**fn**).

8c. [4 points] In Matlab, implement a function that transforms a matrix by standardizing each of its columns so they have mean 0 and unit variance. Turn in your code (by uploading it into Sakai) in one file called **transform_std.m**. Hint: The header for your function should be as follows, where X is the data matrix: **function [X] = transform_std(X)**

8d. [2 points] In Matlab, implement a function that transforms a matrix by doing a log conversion. Specifically, the function takes each element x_{ij} of a matrix X and replaces it with $\log(x_{ij} + 0.1)$, where log is the natural log. Turn in your code (by uploading it into Sakai) in one file called **transform_log.m**. Hint: The header for your function should be as follows, where X is the data matrix: **function X = transform_log(X)**

8e. [4 points] Run experiments on your implementations of logistic regression with l_2 regularization and Gaussian Naïve Bayes with the standard and log data transformations.

Data Description. We will use the email spam data set discussed on p300 of the HTF book.[2] It consists of 4601 email messages, from which 57 features have been extracted. The features are as follows:

- 48 features giving the percentage (0 to 100) of words in a given message that match a given word on the list. The list contains words such as "business", "free", etc.
- 6 features giving the percentage (0 to 100) of characters in the email that match a given character on the list. The characters are ; ([! $ #
- Feature 55 is the average length of an uninterrupted sequence of capital letters (max is 40.3, mean is 4.9).
- Feature 56 is the length of the longest uninterrupted sequence of capital letters (max is 45.0, mean is 52.6).
- Feature 57 is the sum of the lengths of uninterrupted sequence of capital letters (max is 25.6, mean is 282.2).

The class label (Y) is binary: spam (=1) or not (=0).

Conduct Experiments. Run the file **run_experiments.m**, which performs the following steps:

1. Loads the data from **spamdata.mat** (available on the Sakai site) into Matlab. The data file contains a training set (of size 3065) and a test set (of size 1536).
2. Pre-processes the data by using your functions in your **transform_std.m** and **transform_log.m**.
3. Creates balanced cross-validation folds
4. Does cross-validation to select the best regularizer λ for standard data transformation and logistic regression with l_2 regularization

[2] http://www-stat.stanford.edu/~tibs/ElemStatLearn/

5. Does cross-validation to select the best regularizer λ for log data transformation and logistic regression with l_2 regularization.
6. Trains and tests these four models (using your **`train_lr.m`**,, **`test_lr.m`**, **`train_gnb.m`**, **`test_gnb.m`**); and reports the classifcation results.
    a. Regularized logistic regression with standard data transformation
    b. Regularized logistic regression with log data transformation
    c. Gaussian Naïve Bayes with standard data transformation
    d. Gaussian Naïve Bayes with log data transformation

What to report? Report the confusion matrices on the test set only, for each model and each data-transformation method. That is, for this question in your <Last-Name><First-Name>HW2.pdf, add Tables 1 through 4 (see below) with your numerical results filled in. Provide brief answers (~3 sentences) to these questions:

- Which model do you recommend for spam classification?
- Why?

Regularized LR with Standard Data Transformation	Predicted class = spam	Predicted class = not spam
Actual class = spam	# True Positives (TP)	# False Negatives (FN)
Actual class = no spam	# False Positives (FP)	# True Negatives (TN)

Table 1. Regularized Logistic Regression with Standard Data Transformation

Regularized LR with Log Data Transformation	Predicted class = spam	Predicted class = not spam
Actual class = spam	# True Positives (TP)	# False Negatives (FN)
Actual class = no spam	# False Positives (FP)	# True Negatives (TN)

Table 2. Regularized Logistic Regression with Log Data Transformation

GNB with Standard Data Transformation	Predicted class = spam	Predicted class = not spam
Actual class = spam	# True Positives (TP)	# False Negatives (FN)
Actual class = no spam	# False Positives (FP)	# True Negatives (TN)

Table 3. Gaussian Naïve Bayes with Standard Data Transformation

GNB with Log Data Transformation	Predicted class = spam	Predicted class = not spam
Actual class = spam	# True Positives (TP)	# False Negatives (FN)
Actual class = no spam	# False Positives (FP)	# True Negatives (TN)

Table 4. Gaussian Naïve Bayes with Log Data Transformation

Additional Instructions for Q8.

- Make sure that it is easy to change the location of the data sets in your code.
- Do not submit any other files besides the ones mentioned above.
- Add comments to your code, especially on lines that could be confusing to understand.
- Use standard MATLAB functions if needed (version 7.10 and above).
- Caution: Take care when handling small probabilities ($<10^{-16}$), which might give you numerical

errors in your implementation. As described in lecture, you should use log probabilities instead, which convert multiplication/division to addition/subtraction and then exponentiate to get the probabilities back. More info about this can be found at these site:

- "Computing Log-Sum-Exp"
  - https://hips.seas.harvard.edu/blog/2013/01/09/computing-log-sum-exp/
- "How to Prevent Overflow and Underflow in Logistic Regression"
  - http://lingpipe-blog.com/2012/02/16/howprevent-overflow-underflow-logistic-regression/