CS74/174, Winter 2015, Problem Set # 2

February 5, 2015

Due: February 19, 2015 @ 11:59pm

This problem set requires you to provide written answers to a few technical questions. In addition, you will need to implement some learning algorithms in MATLAB and apply them to the enclosed data sets. The questions requiring a written answer are denoted with **W**, while the MATLAB programming questions are marked with **M**.

You must provide your written answers in paper form and leave them in the course mailbox located near the lobby of the Sudikoff building (the mailbox is tagged with the label CS74/174). Since each problem will be graded by a different person, we ask that you provide a separate submission for each individual problem. **Please use the cover pages included at the end of this document. For each problem, fill out the corresponding cover page by writing your name, date and time of submission and use it as front page for your problem submission. A penalty of 5 points will be applied if you fail to follow these instructions.**

Instead, your MATLAB code must be submitted via Canvas. We have provided MATLAB function files with an empty body for all the functions that you need to implement. Each file specifies in detail the input and the output arguments of the function. A few of the functions include some commands to help you get started with the implementation. We have also provided scripts that check the sizes of the inputs and outputs of your functions. Do not modify these scripts. In order to allow us to test your code, it is imperative that you do not rename the files or modify the syntax of the functions. Make sure to include all the files necessary to run your code. **If we are unable to run one of your functions, you will receive 0 points for that question.** Please place all your files (data, .fig files automatically generated by the provided scripts, and function files) in a single folder (**without subfolders**). Zip up the entire folder, and upload the compressed file to Canvas.

1. **[W, 20 points]** Consider a binary classification problem where the input features are *real-valued*, i.e., $x_j \in \mathbb{R}$ for $j = 1, \ldots, n$. We want to learn a Naive Bayes classifier from a training set $\mathcal{D} = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$ via maximum likelihood estimation (as usual the examples are assumed to be I.I.D. within each class, and independent across classes). Recall that Naive Bayes makes the assumption that the features are conditionally independent given the class. We model the class conditional density of each feature x_j given the class as

a separate Gaussian distribution. Thus, our generative model is:

$$p(y=1) = \phi_y \quad (1)$$

$$p(x|y=0) = \prod_{j=1}^{n} \mathcal{N}(x_j; \mu_{j|0}, \sigma^2_{j|0}) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2_{j|0}}} \exp\left\{-\frac{1}{2\sigma^2_{j|0}}(x_j - \mu_{j|0})^2\right\} \quad (2)$$

$$p(x|y=1) = \prod_{j=1}^{n} \mathcal{N}(x_j; \mu_{j|1}, \sigma^2_{j|1}) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2_{j|1}}} \exp\left\{-\frac{1}{2\sigma^2_{j|1}}(x_j - \mu_{j|1})^2\right\} \quad (3)$$

(a) [**W**, 8 points] Write down the expression for the log-likelihood function.

(b) [**W**, 12 points] Compute the maximum likelihood estimates for parameters $\phi_y, \mu_{j|c}, \sigma^2_{j|c}$ where $c \in \{0, 1\}$. Show your derivation but omit redundant calculations (e.g., show the derivations for $\mu_{j|0}$ and $\sigma^2_{j|0}$ but omit those for $\mu_{j|1}$ and $\sigma^2_{j|1}$, since they are analogous).

2. **[25 points]** For this problem you will need to implement logistic regression in MATLAB. You will evaluate your algorithm on the Parkinsons dataset contained in *parkinsons.mat*, which we obtained from the UCI Machine Learning Repository (the data is formatted as in the previous homework). Look at *parkinsons.names* for a description of the features. The objective is to recognize healthy people from those with Parkinson's disease using a series of biomedical voice measurements.

(a) **[M, 16 points]** Implement a gradient ascent algorithm maximizing the logistic regression log-likelihood function (you are not allowed to use MATLAB built-in functionalities for gradient ascent optimization). The method requires selecting a value for the learning rate α: for now, use a fixed small value (e.g., $\alpha = 10^{-6}$). Your task is to fill the body of the following functions that we have provided to you:

- [**M**, 2 points] `q2_initialize.m` initializes the weights for maximum log likelihood training. The function provides two options: *random* and *heuristic* initialization. We have implemented for you the *random* option. Your task is to code the *heuristic* option that we have discussed in class, i.e., the one involving the solution of the least squares system originating by setting $g(\theta^\top x^{(i)}) = w^{(i)}$ for $i = 1, ..., m$, where g is the logistic function, and $w^{(i)} = 0.95$ if $y^{(i)} = 1, w^{(i)} = 0.05$ otherwise.
- [**M**, 1 point] `q2_predict.m` computes the label predictions and the class posterior probabilities for the input examples in matrix X using the parameter vector θ.
- [**M**, 4 points] `q2_loglik.m` computes the log likelihood of the training data given the model θ. Add the Matlab constant `eps` (i.e., a tiny positive number) to the probabilities before taking the log in order to avoid numerical issues. [Hint: you can call `q2_predict.m` inside this function].
- [**M**, 3 points] `q2_gradient.m` computes the gradient of the log likelihood at the current θ. [Hint: use `q2_predict.m` inside this function].
- [**M**, 5 points] `q2_train.m` trains logistic regression with gradient ascent using a fixed step size α. [Hint: you should call `q2_loglik.m` and `q2_gradient.m` inside this function].

- [**M**, 1 point] `q2_error.m` calculate the misclassification rate by comparing the predicted labels to the true labels Y. The misclassification rate is given by the number of misclassified examples over the total number of examples.

After implementing the above functions, you should be able to run `q2a.m`. This script will train logistic regression via gradient ascent using a small fixed step size $\alpha = 10^{-6}$. It will report both the training and the test error, the number of iterations needed to reach convergence, and will plot a curve of the log likelihood as a function of the number of iterations. [Hint: if you have implemented the functions correctly, the log likelihood curve should be monotonically increasing].

(b) **[M, 5 points]** Implement the function `q2_train_line_search.m` which trains logistic regression using line search to refine the step size α adaptively at each iteration (see lecture slides for the pseudocode). The line search method requires an initial value α_0. As discussed in class this should be chosen fairly large, e.g., $\alpha_0 = 10^{-4}$. Again, you are not allowed to use MATLAB built-in functionalities for gradient ascent optimization. After finishing this function, you can run the `q2b.m` to test your gradient ascent algorithm with line search. The script will report, for both the version using the fixed α (as coded for the previous question) and the one using line search, the following information: the training and the test errors, the number of iterations needed to reach convergence, and the log likelihood as a function of the number of iterations. Based on these results, answer the questions below:

(c) **[W, 2 points]** Compare the training and test errors of the two variants of gradient ascent. Are the errors different in the two cases? Explain why or why not.

(d) **[W, 2 points]** Now compare the two log likelihood curves. Does the method using line search converge faster or slower than the version using a fixed step size? Explain the result.

3. *Problem for graduate credit only: if you are enrolled in COSC74 (instead of COSC174) you do not need to solve this problem.*
[15 points] This problem requires you to implement the k-Nearest Neighbors classifier (kNN) based on Euclidean distance and evaluate it on the Parkinsons dataset *parkinsons.mat* of the previous problem.

(a) **[M, 7 points]** Implement the following functions:

- [**M**, 5 points] `q3_predict.m` predicts the class label of an input test example using kNN.
- [**M**, 2 point] `q3_test_error.m` computes the misclassification rates of kNN on a test set for different choices of hyperparameters k. [Hint: reuse the function `q2_error.m` that you wrote for the previous question].

You now should be able to run the script `q3a.m` to plot the test errors for different parameters $k \in \{1, 3, 5, 7, 9, 11, 13\}$.

(b) **[M, 2 points]** Implement `q3_cross_validation_error.m` to evaluate the cross validation errors for different parameters k. Note that the cross validation error for a classification problem is defined as the average misclassification rate over the N folds. After you

complete this function, run the script `q3b.m` to plot the 5-fold cross validation errors for different parameters $k \in \{1, 3, 5, 7, 9, 11, 13\}$.

(c) **[M, 2 points]** Implement `q3_leave_one_out_error.m` to evaluate the validation errors of *leave-one-out* for different parameters k. After you complete this function, run the script `q3c.m` to plot the leave-one-out errors for different parameters $k \in \{1, 3, 5, 7, 9, 11, 13\}$.

Now, looking at the curves of the cross-validation error, the leave-one-out error and the test error, answer the following questions:

(d) **[W, 2 points]** Is the shape of the leave-one-out validation error different from the curve generated using 5-fold cross-validation? Why?

(e) **[W, 2 points]** Which of the two validation strategies (5-fold cross-validation or leave-one-out) produces the error curve most similar to the test error curve? Why?

4. **[M, 20 points]** In this problem you will have a chance to develop your own spam filter using a Naïve Bayes classifier. The data set that you will be using ("spamdata.mat") was derived from spam and non-spam emails collected a few years ago at Hewlett Packard Research Labs to test different spam detection algorithms. Each email is represented as a vector of 48 binary features indicating whether or not particular words occur in the email (the complete list of words can be found in the file "spambase_names.txt"). The label y takes on two values, one corresponding to "ham" (i.e., valid email) and the other to "spam". It should be easy for you to figure out whether "spam" is denoted by 0 or 1 (hint: look at the frequency of occurrence of certain cue words in examples of class 0 and class 1). There are 1500 training examples and 3101 test examples.

(a) **[M, 15 points]** Implement the Maximum Likelihood Naïve Bayes classifier for *binary* features with Laplacian smoothing illustrated in class and train it on the training set. Report the misclassification rate (i.e., the fraction of examples misclassified) obtained on the training set as well as on the test set. In order to do this, you need to implement the following functions:

- [**M**, 9 points] `q4_nb_train.m` trains Naïve Bayes given a training set. The function should learn the class prior and the class conditional probabilities for every feature.
- [**M**, 6 points] `q4_nb_predict.m` predicts the class labels for a given set of examples using a trained model. **Note:** Consider these tips to implement this function: first, because $p(x)$ is a constant for a given x, instead of computing exactly $p(y = 0|x)$ and $p(y = 1|x)$, you just need to find $\arg\max_y p(x|y)p(y)$; second, to avoid the numerical issues arising from the product of many small numbers in $p(x|y)$, use the logarithm function, which being a monotonically increasing function, will not change the result of the arg max operator.

After you are done coding the two functions above, run the script `q4a.m` to compute the training and the test errors of your Naïve Bayes classifier.

(b) **[M, 5 points]** Using the learned Naive Bayes model, compute $p(y = 0|x_j = 1)$ for all features $j = 1, \ldots, 48$. Based on these probabilities, list the 6 words that are most indicative of a message being "spam", and the 6 words most indicative of a message being "ham" (the words associated to the binary features can be found in file "spambase_names.txt").

To accomplish this task, you must simply implement the function `q4_top_words.m`. Then run the script `q4b.m` to output the top 6 most indicative words for both classes.

5. **[W, 20 points]** Consider a problem in which the class label $y \in \{0, 1\}$ and each training example x has two binary attributes, i.e., $x = [x_1\ x_2]^\top$ where $x_1, x_2 \in \{0, 1\}$. Let the class prior be $p(y = 1) = 0.5$. Furthermore, let us assume that:

$$\begin{aligned} p(x_1 = 1|y = 1) &= 0.8 \\ p(x_2 = 1|y = 1) &= 0.5 \\ p(x_1 = 0|y = 0) &= 0.7 \\ p(x_2 = 0|y = 0) &= 0.9. \end{aligned}$$

   (a) [**W**, 3 points] Assume that x_1 and x_2 are truly independent given y. Write down the decision rule of Naive Bayes for this particular case, i.e., list the predicted label y^{pred} for the four possible values of vector x.

   (b) [**W**, 7 points] Show that if Naive Bayes uses both attributes, x_1 and x_2, the error rate is 0.235. The error rate is defined as the probability that the predicted label is incorrect, i.e., $p(y \neq y^{pred})$. To receive credit you must explain your derivation.

   (c) [**W**, 2 points] Calculate the error rate when using only the single attribute x_1 for prediction. What is the error rate when using only x_2? Are these errors higher or lower than when using both attributes? Can you explain why?

   (d) [**W**, 5 points] Now, suppose that we create a new attribute x_3, which is an exact copy of x_2. So, for every training example, attributes x_2 and x_3 have the same value, $x_2 = x_3$. What is the error rate of Naive Bayes now? (Hint: The true distribution has not changed.)

   (e) [**W**, 2 points] Explain what is happening with Naive Bayes in this last case.

   (f) [**W**, 1 points] Would Logistic Regression suffer from the same problem? Explain your answer.

HW2 | CS74/174, Winter 2015

Problem 1

First name _____ **Last name** _____

Date and time of submission _____

HW2 | CS74/174, Winter 2015

Problem 2

First name _____ **Last name** _____

Date and time of submission _____

HW2 | CS74/174, Winter 2015

Problem 3

First name _____ **Last name** _____

Date and time of submission _____

HW2 | CS74/174, Winter 2015

Problem 5

First name _____ **Last name** _____

Date and time of submission _____