IDS575: Machine Learning & Statistical Methods *Updated: 2/11/2020*

Homework #2

Instructor: Moontae Lee *Total Points: 150+10*

Policy

1. HW2 is due by 03/01/2020 11:59PM in Central Time. One submission per each group.
2. You are allowed to work individually or as a group of up to three students.
3. Having wider discussions is not prohibited. Put all the names of students beyond your group members. However individual students/groups must write their own solutions.
4. Put your write-up and results from the coding questions in a single pdf file. Compress R source codes into one zip file. Each student/group must submit only two files. (You will lose the points if the answers for coding questions are not included the pdf report)
5. If you would include some graphs, be sure to include the source codes together that were used to generate those figures. Every result must be reproducible!
6. Maximally leverage Piazza to benefit other students by your questions and answers. Try to be updated by checking notifications in both Piazza and the class webpage.
7. Late submissions will be penalized 20% per each late day. No assignment will be accepted more than 3 days after its due date. For HW2, it will be 03/04/2020 11:59pm

Problem 1: Linear Regression and Gradient Learning [30 points]

In class we derived linear regression and various learning algorithms based on gradient descent. In addition to the least square objective, we also learned its probabilistic perspective where each observation is assumed to have a Gaussian noise. (i.e., Noise of each example is an *independent and identically distributed sample from a normal distribution*) In this problem, you are supposed to deal with the following regression model that includes two linear features and one quadratic feature.

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \epsilon \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

Your goal is to develop a gradient descent learning algorithm that will estimate the best parameters $\boldsymbol{\theta} = \{\theta_0, \theta_1, \theta_2, \theta_3\}$.

(a) Given the definition of noise, derive the corresponding mean and variance parameters of the normal distribution for $y|x_1, x_2; \boldsymbol{\theta}$. Write also down its probability density function.

(b) You are provided with a training observations $D = \{(x_1^{(i)}, x_2^{(i)}, y^{(i)}) | 1 \leq i \leq m\}$. Derive the conditional log-likelihood that will be later maximized to make D most likley.

(c) If you omit all the constant that does not relate to our parameters $\boldsymbol{\theta}$, what will be the objective function $J(\theta_0, \theta_1, \theta_2, \theta_3)$ that you are going to perform Maximum Likelihood Estimation? Does J look similar to the Least Square objective for this problem?

(d) Compute the gradient of $J(\boldsymbol{\theta})$ with respect to each parameter. (Hint: You should evaluate the partial derivatives of $J(\boldsymbol{\theta})$ with respect to each θ_j for $0 \leq j \leq 3$)

(e) [Coding] Develop two learning algorithms: batch and stochastic gradient descent for this problem on the **Auto** dataset given in the Problem 4 in Homework 1. Try to find the two best input features for predicting the output **mpg**. Compare and report the difference between your full coding and R's built-in function call: **lm**. (Hint: At least your stochastic gradient algorithm must learn parameters $\boldsymbol{\theta}$ comparable to the result from calling the R's built-in function. Otherwise try to tune the learning rate $\alpha < 1.0$)

Problem 2: Logistic Regression and Evaluations [30 points]

Recall the grading problem to predict pass/fail in the class. Suppose you collect data for a group of students in the class that consist of two input features $X_1 =$ *hours studied* and $X_2 =$ *undergrad GPA*. Your goal is to predict the output $Y \in \{$*pass*, *fail*$\}$. Suppose that you fit a logistic regression, learning its parameter $(\theta_0, \theta_1, \theta_2) = (-6, 0.05, 1)$.

(a) What will be the probability for a student who studies for 40 hours and has a GPA of 3.5 to pass the class?

(b) How many hours would the student in part (a) needs to study in order to have at least 50% chance of passing the class?

The following questions must be answered using **Weekly** dataset in ISLR package. It contains 1,089 weekly returns for 21 years from the beginning of 1990 to the end of 2010. You will use its 1990-2008 as a training data and 2009-2010 as a test data.

(c) [Coding] Given the training data, you are to perform a logistic regression where the input features are five of **Lag** variables and **Volume**, and the binary output is **Direction**. Use the summary function to print our the results. Report the confusion matrix and the accuracy on both training and test data given the learned model. (Hint: As nothing is specified, your default threshold to decide **Up/Down** must be 0.5)

(d) [Coding] Now you will run logistic regression five times with only one input features $\boldsymbol{Lag_j} (1 \leq j \leq 5)$ for each time. Compute the confusion matrix and the accuracy on both training and test data given each of the learned models. Which are the best models among the five models here and the earlier model in part (c) in terms of the accuracy and F-score, respectively? Does the best model also achieve the best accuracy or F-score

on the training data? (Hint: The best model must be chosen based on the test data, not the training data!)

(e) [Coding] Try to draw six ROC curves for 6 models from the part (c) and (d) with varying thresholds. Determine the best model in terms of the Area Under Curve (AUC). (Note: 6 curves must be plotted at the same time as a single graph. You can use R's built-in function to compute the AUC)

(f) [Coding] Try to draw six Precision-Recall curves for 6 models from the part (c) and (d) with varying thresholds. Determine the best model in terms of the Area Under Curve (AUC). (Note: 6 curves must be plotted at the same time as a single graph. You can use R's builtin function to compute the AUC)

(g) Report your observation about different evaluation measures, and pick the one that you think the most appropriate for this problem. Explain why.

Problem 3: Generalized Linear Model [20 points]

Generalized Linear Models provide a standardized recipe for developing a new learning model and algorithm. In our lecture, you have seen three examples: Linear regression, Logistic regression, and Multi-class classification where outputs given an input are modeled by Normal, Bernoulli, and Categorical distributions, respectively. In this problem, your output will be a different type of random variable, which models the number of times that an event occurs in an interval of time or space. The distribution you are to use has the following form:

$$p(y;\lambda) = \frac{e^{-\lambda}\lambda^y}{y!}$$

, where the only parameter λ indicates the rate of the event which occurs independently. (If you receive in average 4 postal mails every day, then receiving any particular mail will not affect the arrival times of future mails. For this example, $\lambda = 4$)

(a) Prove or disprove whether this distribution is in the exponential family. If it is an exponential family, write down clearly what are the natural parameter η, the natural sufficient statistics $T(\eta)$, the log partition function $a(\eta)$, and the base measure $b(y)$.

(b) Recall that the natural parameter of Logistic Regression was the log odd ratio of Bernoulli parameter. (e.g., $\eta = \log \frac{\phi}{1-\phi}$) When we express ϕ in terms of η, it corresponded to the logistic function, explaining why the logistic function is the natural choice for the binary classification. What will be such a natural function for this distribution? Try to express η in terms of λ. (Hints: Mean of this distribution is λ)

(c) Suppose you are given with a training data $D = \{(x^{(i)}, y^{(i)}) | 1 \leq i \leq m\}$. Evaluate the partial derivative of one single example $p(y^{(i)}|x^{(i)};\theta)$ with respect to θ_j. Derive a stochastic gradient ascent algorithm for learning the model parameter θ.

(d) (Optional +5pts) In class we have seen both Linear regression and Logistic regression have exactly same learning algorithms, and it was explained because they are all GLMs. For majority members of the exponential family that satisfy $T(y) = y$, prove the stochastic gradient algorithm on its log-likelihood shares the same update $\theta_j := \theta_j - \alpha(h_\theta(x) - y)x_j$.

Problem 4: Support Vector Machine [20 points]

Given the following toy dataset consisting of seven exmaples just two features, you are supposed to learn maximal margin classifier.

i	$x_1^{(i)}$	$x_2^{(i)}$	Y
1	3	4	Yes
2	2	2	Yes
3	4	4	Yes
4	1	4	Yes
5	2	1	No
6	4	1	No
7	4	3	No

(a) Visualize all seven training examples and sketch the optimal separating hyperplane. Write down the equation for this hyperplane.

(b) State the rule of classification between **Yes** and **No**. It should be of the form "Yes" if $b + w_1x_1 + w_2x_2 > 0$, and "No" otherwise.

(c) Is the data linearly separable? If so, on top of your sketch, indicate the geometric margin and the support vectors.

(d) Would a slight perturbation of the 6-th example affect the optimal margin hyperplane? Justify your answer.

Problem 5: Multiclass Support Vector Machine [50 points]

Text classification is an important problem in information retrieval and natural language processing. The task is to classify each text article into one of the predefined classes/categories. Here you have four sets of articles about: 1) operating systems, 2) vehicles, 3) sports, and 4) politics. Each set consists of various articles collected from a corresponding newsgroup where each article is represented by Bag-of-Words (BoW) features. That is, after extracting all the relevant words from the entire dataset, feature vectors are defined as the frequency of words occurring in each article.

Since you have more than two classes, using a single binary SVM classifier is not sufficient. Instead, you are going to combine several binary SVM classifiers to predict multiple classes. Thus, the goal is to train multiple linear SVM classifiers that predict binary classes based on BoW features, then combining them to properly predict one of the four classes. You can download the following data files in the class webpage.

- *articles.train* Training data consisting of 4,000 articles (1,000 articles per class)
- *articles.test* Test data consisting of 2,400 articles (600 articles per class)
- *words.map* A word table mapping every relevant word into its line number.

A single line in the training and test data has the following format:

- *Format: (Class #) (Features)*
- *Class #:* One of 1, 2, 3, 4 (1=operating systems, 2=vehicles, 3=sports, 4=politics)
- *Features:* A space-separated sequence of *word #:frequency*

If a line is *1 11:2 13:1 23:12 25:1 27:2 28:1*, for example, this is an article about *operating system* which contains word 11 (*addresses*) twice, word 13 (*organizations*) once, and so forth. Word-integer mapping information is given in the *words.map* where each word is mapped into its line number.[1] The followings are all **coding** questions!

(a) Try to load the training and test data into data frames in R.

(b) First, train four different (hard-margin) linear classifiers. As SVM classifies only binary labels, you have to **replace** the target class number to **1** and all others to **-1** before calling the library function. For instance, if you try to classify whether or not *politics*, you are to use 1,000 articles about *politics* as positive samples and 3,000 others as negative samples for training. Once you learn the four classifiers, the output label of each test example x is determined by the following formula:

$$h_{\mathbf{w},b}(x) = argmax_{k\in\{1,2,3,4\}}(\mathbf{w}^{(k)^T}x + b^{(k)})$$

where $\mathbf{w}^{(k)}$ is the learned weight vector, and $b^{(k)}$ is the biased term for the class k. If the prediction $h_{\mathbf{w},b}(x)$ is different from the ground-truth class, it yields an error. Report the training and test errors of four classifiers, respectively.

(c) Now you are to train soft-margin linear classifiers with different C values from {0.125, 0.25, 0.5, 1, 2, 4, 8, ... , 256, 512}. In order to pick the best C value, you are required to perform a hold-out validation:

[1] *The words.map* is a simple text file having one word per each line. Every word corresponds to its unique line number starting from 1

1. Split the entire training data randomly into 75% for training and 25% for validation.
2. For each C value, learn four binary classifiers similar to part (a) but only on the training data.
3. Measure the overall classification error on the validation data.
4. Pick the C with the lowest validation error.

Plot a graph showing **both training and validation errors together** with varying C in log-scale. (i.e., x-axis: $\log_2 C$, y-axis: error rate) What are the best C value for multiclass classification? (Note: You should apply one uniform C value identically to all four soft-margin classifiers)

(d) With the best C value chosen from part (b), learn four soft-margin classifiers again on the entire training set. (Note that your classifiers from part (b) used only 75% of the training set for learning, holding out 25% for validation) Test your newly learned best classifiers on the test set similar to part (a) where the output label is determined by the *argmax* class given in the formula in part (a). Compare the test error rates to hard-margin classifiers on part (a). Which classifier works better? Justify your observation.

(e) For this problem, you will normalize feature vectors so that the feature vectors of each example have unit length. For each example $x = (x_1, x_2, ..., x_n)$, divide every component into $\|x\|_2$ so that $\|x\|_2 = 1$. Repeat the part (b) with normalized features and measure the test error rates again with newly picked C value. Compare the new test error to previous test error from soft-margin classifier without normalization, and explain why normalization makes a difference.

(f) (Optional +5pts) What you have done so far is one way to extend binary SVM to a multiclass classifier, which is called *1-vs-all*. There is another way of called *1-vs-1*, where you are supposed to train all $\binom{4}{2}$ binary classifiers distinguishing between every pair of classes. Then for each test example, you will pick the (not necessarily unique) class that achieves the highest votes.

Formally speaking, suppose that the binary classifier h_{ij} is trained by taking the examples from class i as positives and the examples from class j as negatives. [2] Then for each test example x, add one vote to class i if h_{ij} says x is in class i. Otherwise, add one vote to class j. Finally *1-vs-1* assigns x to the class with the maximum votes. Since vote is added by one each time, the maximum voted class is not necessarily unique.

Compare the **accuracy** of *1-vs-1* to *1-vs-all*. Accuracy should be measured fairly between two methods via using the normalized features as done the in part (e) and the best C parameters as done in the part (b).

[2] Now you use 1,000 positive samples and only 1,000 negative samples for learning each classifier.