CS6923 Machine Learning, Spring 2018
Prof. Hellerstein, NYU School of Engineering

Homework 3

Submit on NYU Classes by Thurs. April 5 at 6:00 p.m. You may work together with one other person on this homework. If you do that, hand in JUST ONE homework for the two of you, with both of your names on it. You may *discuss* this homework with other students but YOU MAY NOT SHARE WRITTEN ANSWERS OR CODE WITH ANYONE BUT YOUR PARTNER.

IMPORTANT SUBMISSION INSTRUCTIONS:
(1) Don't submit a single zip/tar, etc. Submit 3 separate files: one file for your written answers to Part I, one file for your written answers/output for the questions in Part II, and one file with your code (a zip file if your code requires more than one file).
(2) If you submit a jupyter notebook, make sure to also submit a separate pdf with the answers to the questions in Part II.
(3) Just one submission per team.
(4) Write your team member names/ids on all submitted files.

Part I: Written Exercises

1. Suppose we use logistic regression to solve a binary classification problem. Assume that we train the classifier using gradient descent, as described in the lecture and in the textbook.

   At each stage of the gradient descent, values are assigned to the weights of the classifier. The classifier corresponds to a function of the form $h(x) = \frac{1}{1+e^{-(w^T x+w_0)}}$ whose output is an estimate of $P(C_1|x)$. It classifies an example as belonging to C_1 if $h(x) > 1/2$, and to C_2 otherwise.

   Assume that in the classification problem, there are two real-valued attributes, x_1 and x_2, and the training set contains only two examples, $x^{(1)} = (x_1 = -5, x_2 = 3)$ with label $r = 1$, and $x^{(2)} = (x_1 = 2, x_2 = 3)$ with label $r = 0$. (Here $r = 1$ denotes that $x \in C_1$, and $r = 0$ denotes that $x \in C_2$). At the start of the gradient descent procedure, let the initial weights be $w_0 = w_1 = w_2 = 0.01$. Assume we used a fixed learning rate of $\eta = .005$.

   For this problem, do not write a program to calculate the answers. You should plug the values into the appropriate expressions by hand, to make sure you understand them. You can then evaluate the expressions using a calculator. Show your work.

   Helpful resource: The file logisticexercises.pdf, which also posted on NYU Classes with this homework, contains answers to problems similar to some of the ones here.

(a) What is the predicted (estimated) value of $P(C_1|x^{(1)})$, using the initial weights?

(b) What is the predicted *label* of $x^{(1)}$, using the initial weights. (Recall that we label an example as belonging to C_1 if $P(C_1|x) > 1/2$.) Is the prediction consistent with the label of this example in the training set?

(c) Repeat the above two questions for $x^{(2)}$.

(d) Again using the initial weights, what is the training error of the classifier? That is, what percentage of the labels in the training set does it predict incorrectly?

(e) In logistic regression, we are trying to minimizing the error function $Err(\mathbf{w}, w_0|\mathcal{X}) = -\sum_t (r^t \log y^t + (1 - r^t) \log(1 - y^t))$.

This quantity is sometimes called the *cross-entropy* error. Recall that for logistic regression, $y = \frac{1}{1+e^{-(\mathbf{w}^T x + w_0)}}$.

Using the initial weights, what is the cross-entropy error on the training set, obtained using the initial weights?

(f) Suppose we do one iteration of the gradient descent procedure (i.e., one iteration of the main loop, so we update each weight once) on this training set, beginning with the initial weights.

Recall that the update rule for the weights are as follows:

$$w_j \leftarrow w_j + \eta \sum_t (r^t - y^t) x_j \quad \text{for } j = 1, \ldots, d$$

and

$$w_j \leftarrow w_j + \eta \sum_t (r^t - y^t) \quad \text{for } j = 0$$

Give the values of the updated weights.

(g) Using the updated weights, what is the cross-entropy (error) of the classifier on the training set?

(h) Did the cross-entropy (error) go up or down after one iteration of the gradient descent? Is this what you expected? Why or why not?

(i) What is the training error of the classifier after one iteration of gradient descent?

2. When using gradient descent to minimize an error function, there are common problems that people encounter. For each of the following problems, explain why it might be happening, and suggest a way to fix the problem.

You may want to do the first problem in Part II before answering this question.

(Note: Gradient descent is a useful generic technique but it is relatively slow. Other optimization methods are often used in practice to speed up computation.)

(a) The error doesn't decrease steadily with the number of iterations; it sometimes goes up, and it sometimes goes down. Also, the weights don't converge.

(b) The error decreases steadily, but very slowly. Even after 1,000,000 iterations, the error is still not much smaller than it was at the beginning.

(c) The error decreases and it converges to a single value, but that value is large.

3. Consider a classification problem with real valued attributes, and two classes, C_1 and C_2.

   If we apply logistic regression to this problem, we learn an expression for $P(C_1|x)$ of the form $h(x) = \frac{1}{1+e^{-(\mathbf{w}^T x + w_0)}}$. Usually, we predict class C_1 for x if $f(x) > 0.5$. This is true iff $w^T x + w_0 > 0$. This is a linear function of x. Therefore, our prediction uses the linear discriminant function $g(x) = \mathbf{w}^T x + w_0$. It predicts positive if $g(x) > 0$ and negative otherwise.

   If we want to avoid false negatives, we might instead predict class C_1 if $h(x) > .30$. Which linear discriminant function $g(x)$ would be used for prediction in this case?

4. To prevent overfitting in logistic regression, we sometimes want to minimize a regularized version of the cross-entropy error function:

$$Err_{reg}(\mathbf{w}, w_0) = \left[-\sum_t (r^t \log y^t + (1 - r^t) \log(1 - y^t) \right] + \frac{\lambda}{2} \sum_{j=1}^{d} w_j^2$$

   where λ is a tunable parameter.

   (Note that w_0 is not included in the penalty term.)

   This is a type of regularization known as L_2 regularization, because it includes a term that penalizes weight vectors $\mathbf{w}$ with high L_2 norm. (The L_2 norm of a vector $z = (z_1, \ldots, z_n)$ is $\sqrt{z_1^2 + z_2^2 + \ldots + z_n^2}$.)

   Prove that the update rules in this case are as follows:

$$w_j \leftarrow w_j + \eta(\left[\sum_t (r^t - y^t) x_j \right] - \lambda w_j) \quad \text{for } j = 1, \ldots, d$$

   and

$$w_j \leftarrow w_j + \eta(\sum_t (r^t - y^t)) \quad \text{for } j = 0$$

   In your proof, you may use the fact that for the unregularized cross-entropy error function Err, because $y = \frac{1}{1+e^{-(\mathbf{w}^T x + w_0)}}$, the following holds:

$$\frac{\partial Err}{\partial w_j} = -\sum_t (r^t - y^t) x_j \quad \text{for } j = 1, \ldots d$$

and

$$\frac{\partial Err}{\partial w_j} = -\sum_t (r^t - y^t) \quad \text{for } j = 0$$

You do not have to re-prove these results (so your proof should be short and easy).

Part II: Programming Exercises

1. Gradient descent is a general technique for finding the minimum point of a function. In this exercise you will get practice with performing gradient descent.

   Consider the function

   $$f(x) = 16x^4 - 32x^3 - 8x^2 + 10x + 9$$

   Graph this function for x in the interval [-2,3]. You will see that it has two minima in that range, a local minimum and a global minimum. These are the only minima of the function.

   (a) What is the value of x at the local minimum and at the global minimum? Find the answer to this question like either by hand with calculus or using software.

   (b) Suppose we apply gradient descent to this function, starting with $x = -2$. To do this, we will need to update x using the update rule

   $$x = x - \eta(f'(x))$$

   where f' is the derivative of f, and eta is the "step size".

   Setting x=-1 and $\eta = 0.001$, run gradient descent for 5 iterations (that is, do the update 5 times). Give the values of x and $f(x)$ at the start and after each of the first 5 iterations. If you have implemented this correctly, the value of x after 1 iteration should be -0.866 and during the 5 iterations, the value of $f(x)$ should be decreasing steadily.

   Run the gradient descent again, starting with $x = -1$, for 1000 iterations. Report the last 5 values of x and $f(x)$. Has the value of x converged? Has the gradient descent found a minimum? Is it the global or the local minimum?

(c) Repeat the previous exercise, but this time, start with x=2.

(d) Setting $x = -1$ and $\eta = 0.01$, run gradient descent for 1000 iterations. As in the previous two exercises, report the initial values of x and $f(x)$, the next 5 values of x and $f(x)$, and the last 5 values of x and $f(x)$. Compare the results obtained this time to the results obtained in the first experiment (with $x = -1$ and $\eta = 0.001$). What happened?

(e) Setting $x = -1$ and $\eta = 0.05$, run gradient descent for 100 iterations. What happened? Explain.

2. In this problem, you will experiment with a version of a sonar dataset from the UCI Machine Learning Repository. (The original dataset is http://archive.ics.uci.edu/ml/datasets/connectionist+bench+(sonar,+mines+vs.+rocks)).

   The task here is to learn to distinguish between two types of objects, using sonar data: a metal cylinder (representing a mine) and a roughly cynlindrical rock. The dataset is in the file sonar.csv, on NYU Classes with this homework. Each line corresponds to a single object, and is labeled either *mine* or *rock*. The transmitted sonar signal is a frequency-modulated chirp, rising in frequency. There are 60 attributes for each object, corresponding to signals obtained from a variety of different aspect angles, spanning 90 degrees for the cylinder and 180 degrees for the rock.

   Convert *mine* into label $r = 1$ (for Class 1) and *rock* into label $r = 0$ (for Class 2).

   Implement gradient descent for logistic regression to minimize the cross-entropy error, using the update rules from class that were reviewed in Part I, Problem 1. Do not use regularization.

   Initialize the weights to all be equal to 0.5. (Note: It would be better to set them to be random values between 0 and 1, but we want everyone to use the same initialization.)

   If y^t is very close to 0, the computer may calculate the log of the quantity to be negative infinity. To avoid problems in the computation, you should check whether y^t is less than e^{-16} before taking the log. If it is, you should change it to e^{-16} and then take the log.

   Your code should take two values as input: the number of iterations and the learning rate η.

   (Note: In an iteration, each of the weights is updated using the given update rule. Updating a weight involves calculating a summation over all training examples.)

   (a) For $\eta = \{0.001, 0.01, 0.05, 0.1, 0.5, 1.0, 1.5\}$, do the following:

      i. With your code, train a logistic classifier using the entire dataset as your training set. Run the gradient descent for 50 iterations. Make a plot with the iteration number on the horizontal axis, and the cross-entropy error on the vertical axis.

ii. Report the cross-entropy error and the classification error of the learned logistic hypothesis, on the training set. (For classification error, report the percentage of training examples that are misclassified.) Use the final values of $\mathbf{w}$,w_0 after the 50 iterations. In making the predictions, predict that an example is a mine if the predicted probability that it is a mine is greater than 0.5.

iii. Give the value of the L_2 norm of w, $||w||_2 = \sqrt{\sum_{i=1}^{d} w_i^2}$ using the final weights.

Present your results for (ii) and (iii) above in the following table:

η	0.001	0.01	0.05	0.1	0.5	1.0	1.5
Cross-entropy error (training)	**102.836**	**161.046**	**622.443**	**845.287**	**1048.828**	**1063.026**	**1064.488**
Classification error (training)	**28.33%**	**36.11%**	**36.11%**	**36.66%**	**36.66%**	**36.66%**	**36.66%**
$\|\|\mathbf{w}\|\|_2$	**2.615**	**6.596**	**32.759**	**65.007**	**324.261**	**649.271**	**973.892**

(b) Make a copy of your code and modify it to use regularization, using the update rule from Part I, Problem 4. Your modified code should take an additional input, which is the value of the penalty parameter λ. For $\eta = 0.1$ do the following for each $\lambda \in \{0, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$ (doing 50 iterations in the gradient descent, as before):

i. Report the cross-entropy error and the classification error of the learned logistic hypothesis, when trained on the entire dataset set. (For classification error, report the percentage of training examples that are misclassified.) Use the final values of $\mathbf{w}$,w_0 after the 50 iterations. In making the predictions, predict that an example is a mine if the predicted probability that it is a mine is greater than 0.5. Also report the value of the L_2 norm of w, $||w||_2 = \sqrt{\sum_{i=1}^{d} w_i^2}$ using the final weights.

ii. Compute the 5-fold cross-validation classification error, and report it. To do the cross-validation, divide the dataset up into examples 1 through 36, 37 through 72, etc, in the order they appear in the csv file. Report the cross-validation error as the percentage of the 180 examples that were misclassified. (You can reuse your code from HW1 to implement the cross-validation.)

Present your results for (i) and (ii) above in the following table:

λ	0	0.05	0.1	0.2	0.3	0.4	0.5
Cross-entropy error (training)	**102.83**	**103.025**	**103.212**	**103.583**	**103.947**	**104.306**	**104.658**
Classification error (training)	**28.33**	**28.33%**	**28.33%**	**28.33%**	**27.77%**	**27.77%**	**27.77%**
Classification error (cross-validation)							
$\|\|\mathbf{w}\|\|_2$	**2.615**	**2.6097**	**2.6038**	**2.5921**	**2.5804**	**2.5688**	**2.557**

(c) What happened as you increased the value of λ in the previous experiments? How did this affect training error (cross-entropy and

cross-validation) and the value of $||\mathbf{w}||_2$? Is this what you expected to happen? Explain.

3. When using regularized logistic regression in practice, we want to output just one final hypothesis at the end, which will be used to classify future examples. If we knew which value of λ to use, we would just output the classifier obtained by training on the entire dataset, using that value of λ.

   Since we don't know which value of λ to use, one possible approach is to perform cross-validation on the whole training set of for different values of λ (as we did above), and then to choose the λ that yielded the smallest classification error. What is a possible problem with this approach?