ISE - 364/464: Introduction to Machine Learning
Homework Assignment 4

This assignment requires the accompanying Python file "HW_4_Code.py" to complete problems 4 and 5 (found in the .zip file).

The goal of this assignment is to provide a series of problems that strengthen knowledge in the mathematics, geometry, and intuition behind logistic regression and the K-nearest neighbors algorithm.

Grading: This assignment is due on Coursesite by E.O.D. 11/01/2024. All problems are worth the same number of points. If a problem has multiple parts, each of those parts will be worth equal amounts and will sum to the total number of points of the original problem (Example: If each problem is worth a single point, and problem 1 has 4 parts, each part will be worth 1/4th of a point). ISE - 364 students are only required to answer problems 1 through 4; however, you are allowed to answer the 5th graduate-level question (if done so correctly, you will receive extra credit in the amount that the 5th problem will be worth for the ISE - 464 students). ISE - 464 students are required to answer all 5 problems.

Submitting: Only electronic submissions on Coursesite are accepted. Students should submit a .zip file for this assignment, with two files inside: one file for the homework write-up and another for the code the students will write.

1 Problems

1. (**Hessian of the Cross-Entropy Loss for Logistic Regression**) Recall that the cross-entropy loss function for a logistic regression model is defined as the negative log-likelihood, given by

$$-\ell(\theta) := -\sum_{i=1}^{m} \left[y^{(i)} \log\left(\sigma\left(\theta^\top x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log\left(\sigma\left(-\theta^\top x^{(i)}\right)\right)\right],$$

where $x^{(i)} \in \mathbb{R}^n$ is a single datapoint, $y^{(i)} \in \{0, 1\}$ is the corresponding target class, and $\theta \in \mathbb{R}^n$ is the vector of parameters. Prove that the Hessian matrix of the negative log-likelihood function (the cross-entropy loss function) for logistic regression can be written as $-\nabla^2 \ell(\theta) = X^\top D X$, where $X \in \mathbb{R}^{m \times n}$ is the design matrix and $D \in \mathbb{R}^{m \times m}$ is a diagonal matrix (a matrix with all elements equal to 0 except for the elements on the diagonal). Further, what are the elements along the diagonal of D?
(Hint 1: I would start from the gradient derived on slide 10 of the logistic regression lecture, but notice that was for the log-likelihood and not the ***negative*** *log-likelihood.)*

(Hint 2: I would also first begin by deriving the expression for the entry of the Hessian corresponding to the j-th row and k-th column and then write it in matrix-vector notation afterward (this may make the derivation more intuitive).)

2. (**Properties of the Cross-Entropy Loss for Logistic Regression**) Consider the fact that a quadratic form $z^\top \Lambda z$, where $\Lambda \in \mathbb{R}^{m \times m}$ is a **diagonal** matrix and $z \in \mathbb{R}^m$, can be written as

$$z^\top \Lambda z = \sum_{i=1}^{m} \lambda_i z_i^2,$$

where $\lambda_i \in \mathbb{R}$ are the diagonal elements of the matrix Λ. The Hessian matrix that you derived in question (1) can be written as $-\nabla^2 \ell(\theta) = X^\top D X$, where $D \in \mathbb{R}^{m \times m}$ is a diagonal matrix with positive terms along the diagonal (i.e., $d_{ii} > 0$ for all $i \in \{1, 2, ..., m\}$) under the assumption that $h_\theta(x) \in (0, 1)$ (i.e., the logistic model does not take on values of exactly 0 or 1) and assuming that X has full-column rank (its columns are linearly independent, i.e., for all nonzero $z \in \mathbb{R}^n$, it follows that $Xz \neq 0$). Using this information, prove that the Hessian matrix of the cross-entropy loss function is positive definite.

3. (**Logistic Regression & KNN By Hand**) Suppose that you are given two datapoints defined by the feature vectors $x^{(1)} = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$, $x^{(2)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, and $x^{(3)} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$ along with the corresponding target values of $y^{(1)} = 0$, $y^{(2)} = 1$, and $y^{(3)} = 1$.

a) Given a logistic regression model $h_\theta(x) = \sigma\left(\theta^\top x\right)$ with the parameter vector $\theta = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$, compute the cross-entropy loss for this dataset. You can leave the expression written in terms of logs and sigmoid functions, just simplify the expression using the values of $y^{(i)}$ and compute the dot products $\theta^\top x^{(i)}$, then simply use a calculator (or Python) for the numerical output.

b) Using the K-NN algorithm with $K = 1$ neighbors, determine the predicted class $\hat{y}$ for the new datapoint $\hat{x} = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$ according to the given dataset.

4. (**Training a Logistic Regression Model via Gradient Descent**) For this problem, you will use Python to train a logistic regression model via gradient descent. You will do this by writing a series of functions that will be called in your implementation of the gradient descent algorithm to minimize the cross-entropy loss function for a logistic regression model. Use the "HW_4_Code.py" file accompanying this document in the .zip file. This is the Python file that you will use to write your code. The functions that you are required to write have been pre-defined for you as well as all of the code needed for initialization (generating the dataset and visualization functions). Follow the comments in the .py file; they should guide you where to write your code.

Once you are finished, you should plot your own Logistic Regression training plot (as shown in the top left plot of Figure 1) as well as a 2D plot of your learned decision boundary (as shown in the bottom plot of Figure 1). Notice that you should "hyperparmeter tune" your learning rate and maximum number of iterations to achieve a "decent" convergence to the optimal solution (I have purposely not displayed plots that have fully converged). Some things to look at:

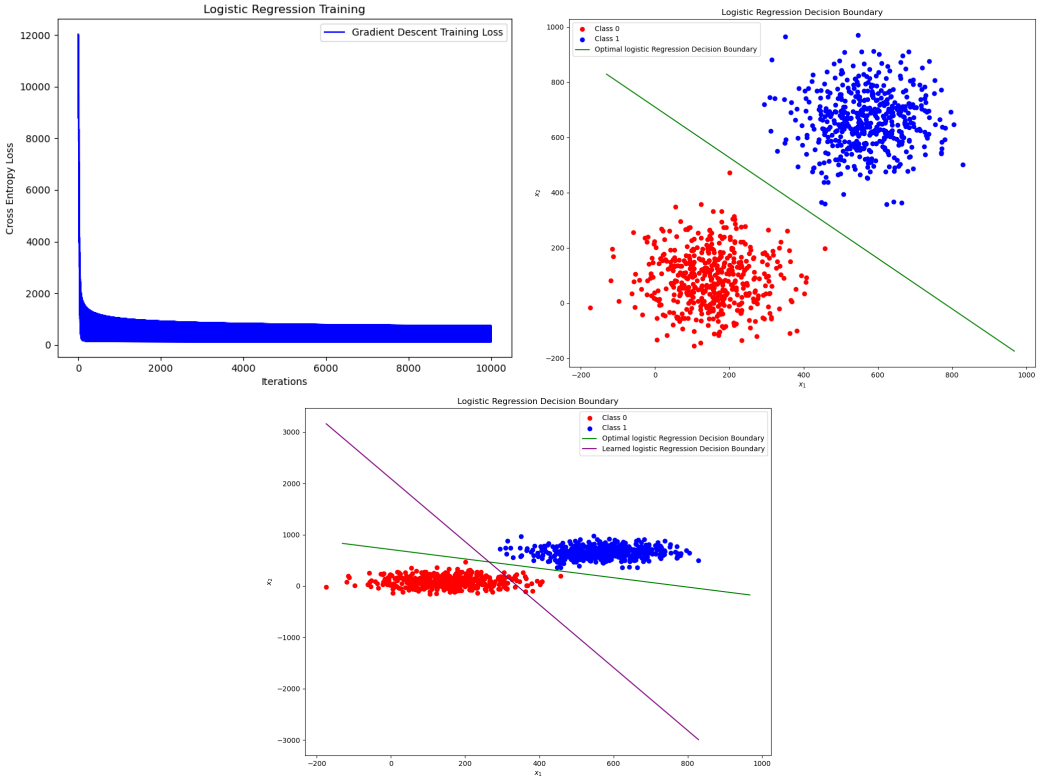

Figure 1: Two methods of illustrating training a logistic regression model as well as visualizing the learned hyperplane (decision boundary) on the dataset that it was trained on. (Top Left) A plot displaying the training progress in the loss function over the number of gradient descent iterations. (Top Right) A 2D scatter plot of the 2 feature dimensions (colored by target class) along with the optimal decision boundary (obtained by Scikit-Learn's "LogisticRegression" model). (Bottom) A 2D scatter plot of the 2 feature dimensions (colored by target class) along with the optimal decision boundary (obtained by Scikit-Learn's LogisticRegression model) along with the trained (via gradient descent) logistic regression model to minimize the cross-entropy loss.

- The optimal cross-entropy loss function value for the parameters you obtained from your trained model.
- What is the gradient evaluated at your final learned parameter value? Is it close to 0?
- Try two different learning rate schemes: a fixed step size (it stays a constant value for every iteration, i.e., $\alpha_k = c \in (0, 1)$ for all iterations k) and a decaying step size (you can try using $\alpha_k = \frac{1}{k}$ or $\alpha_k = \frac{1}{\log(k)}$). You should notice that certain learning rate schemes are more effective than others to converge to a solution.
- How difficult is it to learn the optimal decision boundary when using gradient descent?

Write a brief summary of your findings and submit your .py file along with your written (or typed) answers in a .zip file.

5. **(ISE-464 Graduate Students)** Using the same Python code file that you wrote for

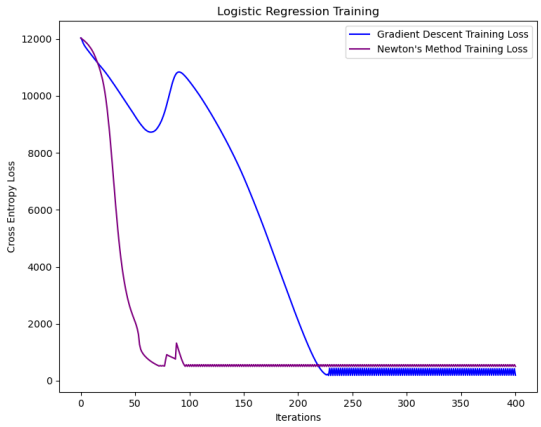

Figure 2: Optimization performance of both gradient descent and Newton's Method when training a logistic regression model.

problem 4, add another algorithm to implement Newton's Method to minimize the cross-entropy loss. This will require you to define another function that computes the Hessian Matrix as well as a function to compute the Newton Direction. You should obtain a performance plot similar to the one displayed in Figure 2. Display your own performance plot that you obtain and write a brief summary of your observations comparing the two algorithms. Simply include your added code in the same file you submit for problem 4.
(Hint 1: You should have derived the expression for the Hessian of the cross-entropy Loss in problem 1.)