Homework 2

Xuhang He

November 11, 2023

1 Question 0:

If you had a classification problem where the training data had 10^3 features and 10^9 samples, which of the following methods would you consider using first: nearest neighbors, or logistic regression? Under what situation would your answer change?

Answer: Logistic Regression is a better solution.
For KNN, the cost would be $O(10^{3+9}) = O(10^{12})$ because the runtime for KNN is O(nd) where n is the total sample size and d is the number of features each data point has.
For Logistic Regression, the cost is $O(qdT) = O(10^3dT)$, where q is the sample size of each mini-batch of stochastic gradient descent and T is number of epoch. Both q and T are hyper-parameters, which means they are decided before the training. In this case, with Logistic Regression, a relatively accurate classifier can be get with lower cost than KNN, so Logistic Regression is better.
However, when the total sample size and the number of features is smaller, KNN would be a better choice than logistic regression, because for a smaller data set, training cost for KNN is smaller than performing gradient descent on the entire set.

2 Question 1

The decision boundary in any binary classification problem can be intuitively viewed as a line/surface that partitions the space of all test data points into two parts, one for each class. In this problem, we explore decision boundaries for k-nearest neighbors, measured on the 2D plane in terms of the L2 (or straight-line) distance.

a. Let us suppose that k = 1. Suppose we have the following very simple training dataset with d = 2 features, where the data points

$$\{(0, 1),(2, 3),(4, 4)\}$$

are labeled class *Blue* and the data points

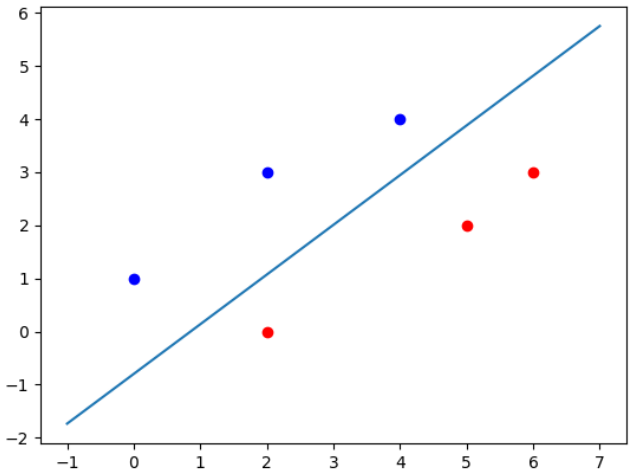

Figure 1: Decision Boundary

$$\{(2,\ 0),(5,\ 2),(6,\ 3)\}$$

are labeled class *Red*. Draw the decision boundary between the two classes.

b. Let us now suppose that k = 3. Argue that (somewhat counter-intuitively) there might exist a test data point that exactly coincides with one of the training data points, but that will still be misclassified. Which point is this, and what is its label?

Answer: The point is (4, 4). The nearest three neighbors are two red class points and one blue class point. According to KNN algorithm with k = 3, (4, 4) should be classified as a red class. However, its label is blue.

c. What does this example teach us about k-nearest neighbors?

Answer: KNN is brittle to noise.

3 Question 2

The use of l_2 regularization for training simple models such as logistic regression or more complicated neural networks has a special name: weight decay. Assume any arbitrary dataset $\{(x_i, y_i)\}_{i=1}^{n}$ and a arbitrary (differentiable) loss function *L(w)*. Here, w represents the vector of all trainable weights (and biases) of the model.

a.Write down the l_2 regularized training loss, using a weighting parameter λ for the regularizer.

Answer: $l_2 = \lambda\|w\|_2^2$ So the updated loss function is $L^{'}(w) = L(w) + \lambda\|w\|_2^2$.

b. Derive the gradient descent update rules for this loss.

Answer: $w_{k+1} = w_k - \eta(\nabla L(w) + 2\lambda w)$ where η is the learning rate. Therefore, weight is updated by $\eta(\nabla L(w) + 2\lambda w)$ for each batch.

c. Conclude that the weights are shrunk or decayed by a multiplicative factor in each step before applying the descent update.

Answer: When updating the weight, it is to decrease the parameter vector by the gradient of the loss function. Therefore, the use of l_2 regularization is to decrease w every time by $\nabla L(w) + 2\lambda w$ instead of only decreasing w by $\nabla L(w)$.

4 Question 3

The Fashion MNIST dataset is a database of (low-resolution) clothing images. There are ten classes You can load it in Google Colab using the Python code below.

a. Load the dataset and display 10 representative images from each of the 10 classes.

see `https://github.com/micheeeal/CSUY4563_Fall2023/blob/main/HW2/HW2_show_images.py`

b. Implement the following classification methods: k-NN and logistic regression. You don't have to implement these from scratch; just use the sklearn package.

see `https://github.com/micheeeal/CSUY4563_Fall2023/blob/main/HW2/HW2_Logistic.py` and `https://github.com/micheeeal/CSUY4563_Fall2023/blob/main/HW2/HW2_KNN.py`

c. Report best possible test-error performance by tuning hyperparameters (number of nearest neighbors for k-NN, learning rate and regularization for logistic regression) in each of your methods.

Answer: For KNN, I tested for K from 1 to 20, and the highest testing score on the dataset is when k = 4, and the score on the test data set is 0.8577. The lowest test score is 0.8415 when k = 20. There is a trend of decreasing test score from k = 11 to k = 20.

For Logistic Regression, the method I used to test regulation and learning rate is to set the power of the learning rate to be the regulation parameter, ranged from 2 to 9. So the test score is tested from $\lambda = 2$ to 9 and learning rate = 10^{-2} to 10^{-9}. The best set of hyper-parameters is when $\lambda = 8$, $lr = 10^{-8}$, test score is 0.8432, and the lowest test score comes from the set of hyper-parameters of $\lambda = 3$ and $lr = 10^{-3}$, which is 0.8416.

d. Report train- and test-running time of each of your methods. Comment on the relative tradeoffs across the different methods.

Answer: The tradeoff exists between the training time and prediction time. For KNN, the prediction time for the highest test score is only 0.00253 seconds, but the prediction time is 11.404 seconds. For Logistic Regression, the prediction time for the highest test score is only 0.0130 seconds, but the training time is 8.0745 seconds.