ECE194N: Homework 1 (due April 27)

Part I: Written Part

Turn in this part during the discussion session on April 27th.

1. Given
$$\boldsymbol{x} = [1, x_1, x_2, ..., x_n]^T \quad and \quad \boldsymbol{w} = [w_0, w_1, w_2, ..., w_n]^T \tag{1}$$
Derive the gradients of following expressions with respect to $\boldsymbol{x}$, $\boldsymbol{w}$.
   (a) $\boldsymbol{w}^T\boldsymbol{x}$
   (b) $e^{\boldsymbol{w}^T\boldsymbol{x}}$
   (c) $\frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}}$
   (d) $(1 - \boldsymbol{w}^T\boldsymbol{x})^2$
   (e) $max(0, 1 - (\boldsymbol{w}^T\boldsymbol{x}))$

2. Given x_i is the input vector and y_i is the corresponding class label for i = 1,2,3,4 and w is the weight vector
$$\boldsymbol{w} = [w_0, w_1, w_2]^T \tag{2}$$
Where,
$$\begin{aligned} \boldsymbol{x_1} &= [1, 2, 0]^T, \quad y_1 = -1 \\ \boldsymbol{x_2} &= [1, 0, 2]^T, \quad y_2 = -1 \\ \boldsymbol{x_3} &= [1, 4, 4]^T, \quad y_3 = 1 \\ \boldsymbol{x_4} &= [1, 3, 4]^T, \quad y_4 = 1 \end{aligned}$$
We want to discriminate class 1 from class 0 using a simple perceptron with hinge loss. Use stochastic gradient descent (one sample at a time) to update your weights $\hat{\boldsymbol{w}}$ and show weights at each step for 4 steps. Initialize weight vector $\boldsymbol{w} = [w_0, w_1, w_2]^T = [0, 0, 0]$ and the learning rate as 1. Hint: Your prediction will be $\hat{y} = \boldsymbol{w}^T\boldsymbol{x}$ and hinge loss is defined as $L = max(0, 1 - y\hat{y})$

Part II: Programming Part

(upload the zip file by 12 noon on Sunday, April 29, on Gauchospace).

You will write a report on each of the questions below. Upload your report together with your code as a zip file to Gauchospace. Organize the main folder into three sub-folders and name them as "Regression", "classification", "KNN". Name the main folder as "yourlastname_firstname". Include the individual reports for each of the sub-problems in the respective sub-folders. Organize the corresponding code into a subfolder named "code" inside the resepctive sub-folders.

1. **Linear Regression:** Python implementation (without Tensorflow),
   Implement linear regression on the housing dataset (house price prediction with 2-dim features(square feet and number of bedrooms)). Dataset is located in the file

   housing_prices.txt

   Each row contains the square footage, number of bedrooms and selling price separated by commas.

   Choose last 10 rows as your testing set and do NOT train on these samples.

   (a) Visualize your data.

   (b) Choose a loss function and derive the update rule for weights. Include the loss function and the update rule in your report.

   (c) Using the update rule, implement and train the linear regression model.

   (d) **Data Normalization:** When features differ by orders of magnitude, first performing feature scaling will give better results. Lets define the data normalization as follows

      i. Subtract the mean value of each feature from the dataset

      ii. After subtracting the mean, additionally scale the feature values by their respective standard deviations

      What are the results using data normalization?

   (e) Compare the results of with-normalization against without-normalization and Comment on them.

   (f) Train the model using different learning rate values. How does the learning rate affect the results?

2. **Classification:** Following the tutorial from below, implement a MNIST classification model that will classify odd and even numbers.
   `https://www.tensorflow.org/versions/r1.1/get_started/mnist/beginners`
   You are expected to use same number of layers as the original network, which means you need to change the number of neurons.

   Explain all your code changes and report your results clearly.

(a) For each of the output classes, identify 10 misclassified test samples and visualize them in your report. And why do you think they got misclassified?

(b) Instead of modifying the model add a new layer at the end of the network from tutorial. Explain the code changes required and report the results.

(c) Explain the conceptual difference between the two methods i.e., modifying the last layer and adding a new layer.

3. **K-Nearest Neighbors:** In this part, we will be using K-Nearest Neighbor to classify image from the CIFAR-10 dataset. Basic idea behind K-Nearest Neighbor algorithm is to use a distance measure in order to find the K-nearest neighbors of a test sample. The test sample is then assigned the label based on the labels of these neighbors.
Dataset: We are going to use CIFAR-10 dataset in order to implement K-Nearest neighbor algorithm. Dataset can be found at `https://www.cs.toronto.edu/~kriz/cifar.html` Training and testing splits are also mentioned on the webpage.
Distance Measure: An example distance between 2 images can be the sum of squared difference between pixel intensities.
Note: If you try to find difference between two 32x32 RGB (Red, Green, Blue) images, you will need to compute 32x32 differences for each of the three, R G B channels.
Label assignment strategy: Once the K-Nearest Neighbors to a test sample are found, assign it the to the label which majority of its neighbors have. In cases of a tie, Consider the distance information in order to break the tie.

(a) Apply K-Nearest Neighbor algorithm with k = 1 on the test samples. We define the classification error rate as

$$P_e = \frac{\text{Number of Wrongly Classified Test Samples}}{\text{Total Number of Test Samples}} \tag{3}$$

What is the error rate of your classification?

(b) Repeat last step for k = 2, 5, 10, 20 and plot the error rate P_e against k. Is the error rate decreases with k? Should the error rate always decrease with k?

(c) For each of the ten classes, pick a random image from test data and report its 10 nearest neighbors.