Homework 3.

Due on Gradescope by 5 PM, Friday, April 23

Part I: Written Part

Turn in this part during the lecture on February 6th.

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 $\boldsymbol{x}$, $\boldsymbol{w}$. Use the numerator layout for gradients, so all final answers should evaluate to a row vector.
   (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,
$$\begin{aligned} \boldsymbol{x_1} &= [1, 2, 0]^T, \quad y_1 = -1 \\ \boldsymbol{x_2} &= [1, 4, 4]^T, \quad y_3 = 1 \\ \boldsymbol{x_3} &= [1, 0, 2]^T, \quad y_2 = -1 \\ \boldsymbol{x_4} &= [1, 3, 4]^T, \quad y_4 = 1 \end{aligned}$$
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. You should update your weights 4 times, once for each sample above. Initialize weight vector $\boldsymbol{w} = [w_0, w_1, w_2]^T = [0, 0, 0]$ and the learning rate as 1.

   For a desired target output $y \in \pm 1$, your prediction will be $\hat{y} = \boldsymbol{w}^T\boldsymbol{x}$ and hinge loss is defined as $L = max(0, 1 - y\hat{y})$

   After each of the 4 training steps, provide the predictions $\hat{\boldsymbol{y}}$, loss value, and updated weights in $\boldsymbol{w}$.

Homework 3.

Due on Gradescope by 5 PM, Friday, April 23

Part II: Programming Part

1. **Linear Regression using Gradient Descent:**
   In this programming exercise, you will implement linear regression using gradient descent (without deep learning libraries such as Tensorflow or Pytorch). You will submit this part of HW3 through a Google Colab notebook. The skeleton for the notebook and questions can be found here. Report all your answers to the questions below in Google Colab and submit the Jupyter notebook as a separate pdf.

   You will be using the housing dataset for this task. The input data is a 2-dimensional feature vector containing (square feet and number of bedrooms) and the expected output is the house price. This housing dataset is located in the file: housing_prices.txt in the HW3 folder in Gauchospace or it can also be found here. Each row in housing_prices.txt contains the square footage, number of bedrooms and selling price separated by commas.

   The linear regression model we want to train is: $y = m1 \cdot x1 + m2 \cdot x2 + m0$. Here, y is the housing price, $x1$ is the square footage, $x2$ is the number of bedrooms. The parameters of the model are $m1, m2, m0$.

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

   (a) Visualize your data.

   (b) Using mean-squared error as the loss function, derive the update rule for parameters. Mention the update rule in your report.

   (c) Using the update rule, implement and train the linear regression model. You can train the model for 10 epochs, with a learning rate of 10^{-7}. Show the plot of the average train and test loss as a function of the number of epochs (you can use one plot for both train and test, use a different line style and specify a legend).

   (d) **Data Normalization:** When input data or 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. Divide the features obtained from (i) by their respective standard deviations.

   What are the results of training with normalized data and a learning rate of 0.1. Show the plot of the average train and test loss as a function of the number of epochs (plot train and test on a single plot as done in (c)).

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

(f) Train the model using different learning rate values: $0.01, 0.05, 0.1, 5, 10$ **with** data normalization. For each learning rate, show the plot of the average train and test loss as a function of the number of epochs (plot train and test on a single plot as done in (c)).