CS 615 - Deep Learning

Assignment 3 - Backprop and Basic Architectures
Winter 2023

Introduction

In this assignment we will implement backpropagation and train/validate a few simple architectures using real datasets.

Allowable Libraries/Functions

Recall that you **cannot** use any ML functions to do the training or evaluation for you. Using basic statistical and linear algebra function like *mean*, *std*, *cov* etc.. is fine, but using ones like *train* are not. Using any ML-related functions, may result in a **zero** for the programming component. In general, use the "spirit of the assignment" (where we're implementing things from scratch) as your guide, but if you want clarification on if can use a particular function, DM the professor on slack.

Grading

Do not modify the public interfaces of any code skeleton given to you. Class and variable names should be exactly the same as the skeleton code provided, and no default parameters should be added or removed.

Part 1 (Theory)	20pts
Part 2 (Visualizing Gradient Descent)	10pts
Part 5 (Linear Regression)	35pts
Part 6 (Logistic Regression)	35pts
TOTAL	100pts

Table 1: Grading Rubric

Datasets

Medical Cost Personal Dataset For our regression task we'll once again use the medical cost dataset that consists of data for 1338 people in a CSV file. This data for each person includes:

1. age
2. sex
3. bmi
4. children
5. smoker
6. region
7. charges (target value, Y)

This time I preprocessed the data for you to again convert the *sex* and *smoker* features into binary features and the *region* into a *set* of binary features (basically one-hot encoded this). In addition, we now *included* the *charges* information as we will want to predict this.

For more information, see https://www.kaggle.com/mirichoi0218/insurance

Kid Creative We will use this dataset for binary classification. This dataset consists of data for 673 people in a CSV file. This data for each person includes:

1. Observation Number (we'll want to omit this)
2. Buy (binary target value, Y)
3. Income
4. Is Female
5. Is Married
6. Has College
7. Is Professional
8. Is Retired
9. Unemployed
10. Residence Length
11. Dual Income
12. Minors
13. Own
14. House

15. White
16. English
17. Prev Child Mag
18. Prev Parent Mag

We'll omit the first column and use the second column for our binary target Y. The remaining 16 columns provide our feature data for our observation matrix X.

1 Theory

1. For the function $J = (x_1 w_1 - 5x_2 w_2 - 2)^2$, where $w = [w_1, w_2]^T$ are our weights to learn:

   (a) What are the partial gradients, $\frac{\partial J}{\partial w_1}$ and $\frac{\partial J}{\partial w_2}$? Show work to support your answer (6pts).

   Ans . for the following function of J our partial gradients will be $\frac{\partial J}{\partial w_1}$ and $\frac{\partial J}{\partial w_2}$. To compute these values,

   $\frac{\partial J}{\partial w_1} = \frac{\partial}{\partial w_1}(x_1 w_1 - 5x_2 w_2 - 2)^2 = 2(x_1 w_1 - 5x_2 w_2 - 2) \cdot x_1$

   $\frac{\partial J}{\partial w_2} = \frac{\partial}{\partial w_2}(x_1 w_1 - 5x_2 w_2 - 2)^2 = 2(x_1 w_1 - 5x_2 w_2 - 2) \cdot -5x_2$

   (b) What are the value of the partial gradients, given current values of $w = [0, 0]^T, x = [1, 1]$ (4pts)?

   Ans. $\frac{\partial J}{\partial w_1} = 2(x_1 w_1 - 5x_2 w_2 - 2) \cdot x_1 = 2(1 \cdot 0 - 5 \cdot 1 \cdot 0 - 2) \cdot 1 = -2 \cdot 2 = -4$

   $\frac{\partial J}{\partial w_2} = 2(x_1 w_1 - 5x_2 w_2 - 2) \cdot -5x_2 = 2(1 \cdot 0 - 5 \cdot 1 \cdot 0 - 2) \cdot -5 \cdot = 20$

2. Given the objective function $J = \frac{1}{4}(x_1 w_1)^4 - \frac{4}{3}(x_1 w_1)^3 + \frac{3}{2}(x_1 w_1)^2$:

   (a) What is the gradient $\frac{\partial J}{\partial w_1}$ (2pts)?

   Ans. $\frac{\partial J}{\partial w_1} = \frac{\partial}{\partial w_1}\left(\frac{1}{4}(x_1 w_1)^4 - \frac{4}{3}(x_1 w_1)^3 + \frac{3}{2}(x_1 w_1)^2\right)$

   Using the power rule, we get $\frac{\partial J}{\partial w_1} = (x_1^4 w_1^3 - 4x_1^3 w_1^2 + 3x_1^2\ w_1)$

   (b) What are the locations of the extrema points for this objective function J if $x_1 = 1$? Recall that to find these you take the derivative of the objective function with respect to the unknown, set that equal to zero and solve for said unknown (in this case, w_1). (5pts)

   Ans. if $x_1 = 1$, the derivative of J with respect to w_1 will be computed against 0,

   $=> w_1(w_1^2 - 4.w_1 + 3) = 0$

   $=> w_1(w_1^2 - 3.w_1 - w_1 + 3) = 0$

   $w_1.(w_1 - 3).(w_1 - 1) = 0$

   So, the value of w_1 which are the point of extrema is

   $w_1 = 0, w_1 = 3, w_1 = 1$

   (c) What does J evaluate to at each of your extrema points, again when $x_1 = 1$ (3pts)?

   Ans. For the value of $w_1 = 0$, $J = 0$

   For the value of $w_1 = 1$ $J = 5/12$

   For the value of $w_1 = 3$ $J = 63/4$

2 Visualizing Gradient Descent

In this section we want to visualize the gradient descent process for the following function (which was part of one of the theory questions):

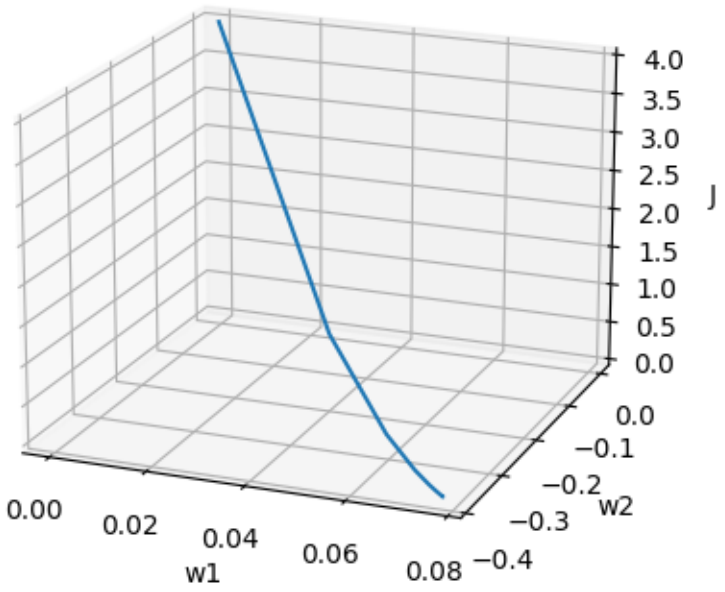
$$J = (x_1 w_1 - 5 x_2 w_2 - 2)^2$$

Note that this is more of a *toy problem* to explore the idea of gradient-based learning than it is a deep learning architecture.

Hyperparameter choices will be as follows:

- Initialize your weights to zero.
- Set the learning rate to $\eta = 0.01$.
- Terminate after 100 epochs.

Using the partial gradients you computed in the theory question, perform gradient descent, using $x = [1, 1]$. After each training epoch, evaluate J so that you can plot w_1 vs w_2, vs J as a 3D line plot. Put this figure in your report.



3 Backpropagate

Let's add the *backward* method to our activate and fully-connected layers! You might want to consider having a default version in the abstract class *Layer*, although we'll leave those design decisions to

you. In general, the *backward* methods should takes as inputs the backcoming gradient, and returns the updated gradient to be backpropagated. The methods' prototype should look like:

```
def backward(self,gradIn):
  #TODO
```

4 Updating Fully Connected Layer's Weights and Biases

We also need to add an *updateWeights* method for our Fully Connected layer. This method takes a backcoming gradient and a learning rate as parameters, and updates its weights and biases according to the formulas in lecture. The method's prototype should look like:

```
def updateWeights(self,gradIn, eta = 0.0001):
  #TODO
```

5 Linear Regression

In this section you'll use your modules to assemble a linear regression model and train and validate it using the *medical cost dataset*. The architecture of your linear regression should be as follows:

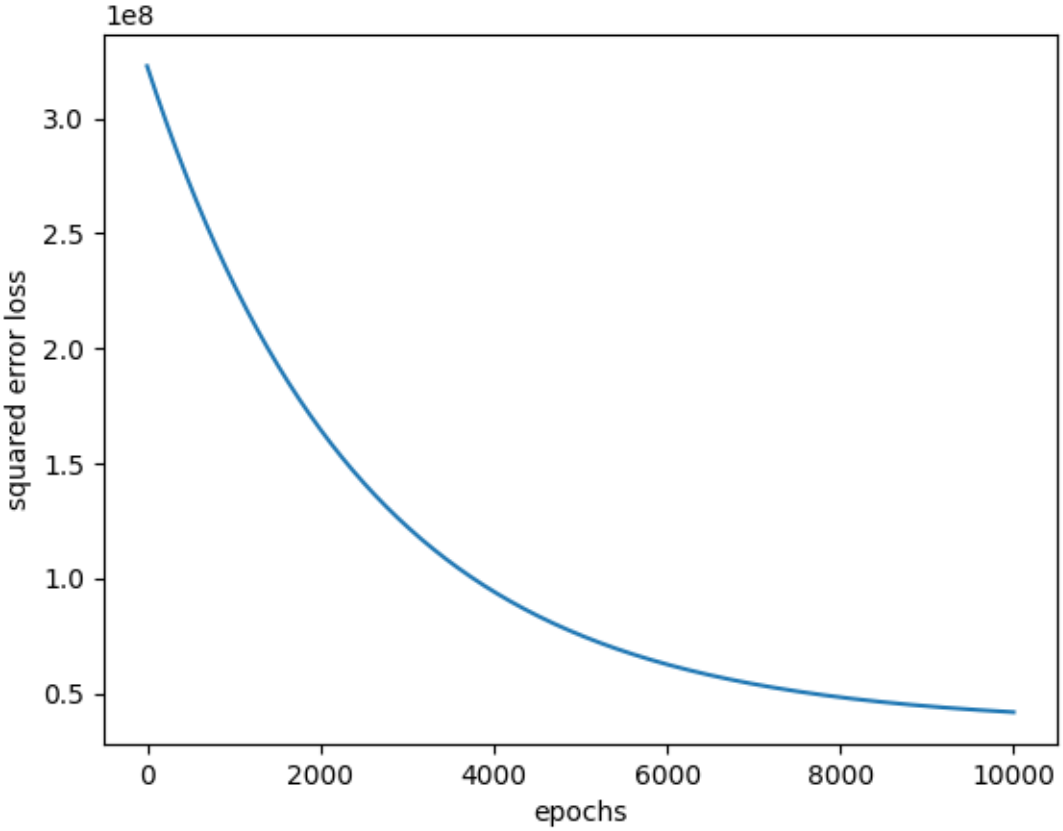
$$Input \rightarrow \text{Fully-Connected} \rightarrow \text{Squared-Error-Objective}$$

Your code should do the following:

1. Read in the dataset to assemble X and Y (recall that our target Y is the *charges* column for this dataset).
2. Train, via gradient learning, your linear regression system using the training data. Refer to the pseudocode in the lecture slides on how this training loop should look. Initialize your weights to be random values in the range of $\pm 10^{-4}$ and your learning rate to be $\eta = 10^{-4}$. Terminate the learning process when the absolute change in the mean squared error on the training data is less than 10^{-10} or you pass $10,000$ epochs. During training, keep track of the **mean squared error** (MSE) for the training set so that we can plot this as a function of the epoch.

In your report provide:

1. Your plots of training MSE vs epoch.



2. Your final RMSE for the training data.: 0.183
3. Your final SMAPE for the training data.: 6474.27

6 Logistic Regression

Next we'll use a logistic regression model on the *kid creative* dataset to predict if a user will purchase a product. The architecture of this model should be:

$$Input \rightarrow \text{Fully-Connected} \rightarrow \text{Sigmoid-Activation} \rightarrow \text{Log-Loss-Objective}$$

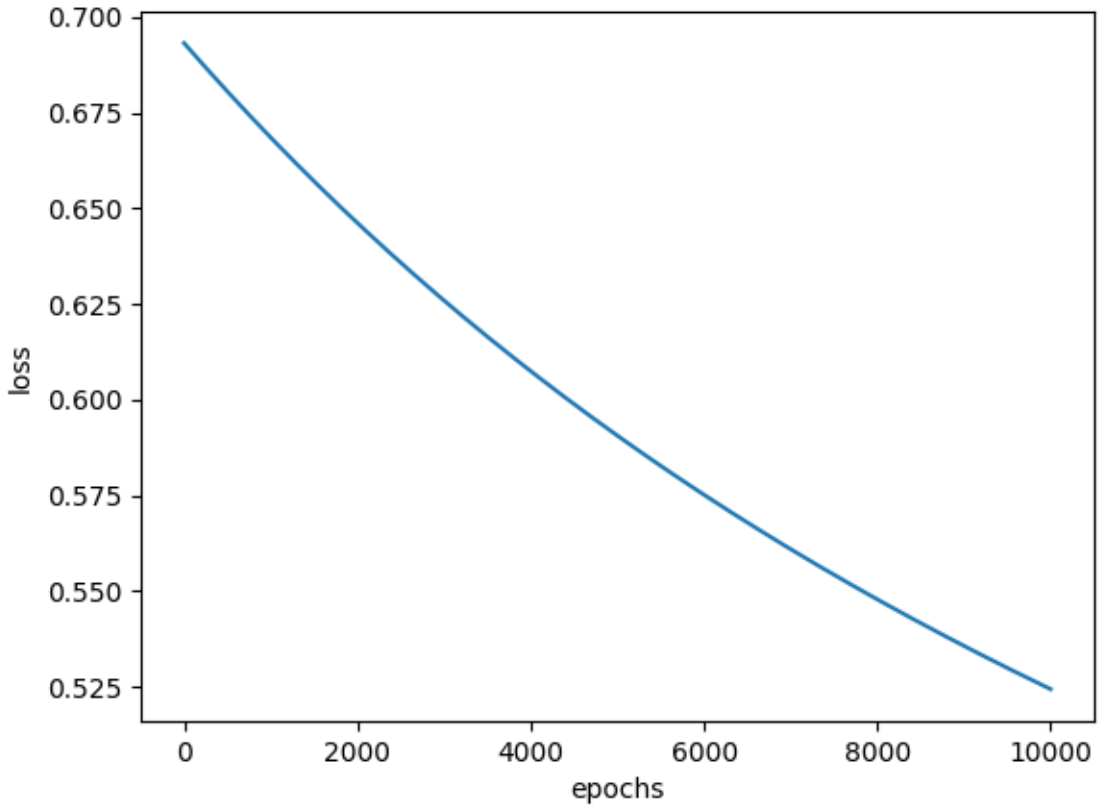
Your code should do the following:

1. Read in the dataset to assemble X and Y (rcall that our target Y is the *Buy* column for this dataset).

2. Train, via gradient learning, your logistic regression system using the training data. Initialize your weights to be random values in the range of $\pm 10^{-4}$ and your learning rate to be $\eta = 10^{-4}$. Terminate the learning process when the absolute change in the log loss is less than 10^{-10} or you pass $10,000$ epochs. During training, keep track of the log loss for the training dataset so that we can plot this as a function of the epoch.

In your report provide:

1. Your plots of training log loss vs epoch.



2. Assigning an observation to class 1 if the model outputs a value greater than 0.5, report the training accuracy.: 0.842

Submission

For your submission, upload to Blackboard a single zip file containing:

1. PDF Writeup
2. Source Code
3. readme.txt file

The readme.txt file should contain information on how to run your code to reproduce results for each part of the assignment.

The PDF document should contain the following:

1. Part 1: Your solutions to the theory question
2. Part 2: Your plot.
3. Parts 3-4: Nothing.
4. Part 5: Your plot and requested statistics.
5. Part 6: Your plot and requested accuracies.