Due Date: Friday, November 18, 2022, 11:59 PM

Abstract

In this assignment, you will learn about linear regression and classification. The questions below can be answered with the help of the book. Show all calculations done step-by-step. In problem 3, you need not worry about being "mathematically" correct. Please follow the same notation as in the book. You are allowed to work on teams of up to three people. If working as a team, every member of the team should submit the (same) report. All reports and supplemental material must be zipped as `team17_Assignment3.zip` and uploaded on Blackboard.

Problem 1. *Assume your input data consists of only numerical values. You have an arbitrary number of features where values have different ranges. For example, X1 is a person's weight, X2 is the price of a house, etc... You are building a neural network to perform classification/regression using such data. Explain in detail how would normalizing or not normalizing your input affects the learning process of a neural network. Explain your answer and provide examples for both scenarios.*

Solution: Input data should be small values, probably in the range of 0 to 1 or standardized with a zero mean and unit variance. One of the best practices for training neural networks is to normalize the data to obtain a mean close to zero. Normalizing the input speeds up the learning and also leads to faster convergence (merging). Also, normalization lets the features to assume that their magnitude is more or less the same.

If we don't normalize the data, the network does not work well. It is like teaching the network with on values from lower part of range, while the actual inputs are from the higher part of the range.

In case of the example mentioned in the question: Weight of a person will be of two or three digits while price of the house can be seven or eight digits (1 million, etc..). If we don't normalize these features, then price of the house will dominate the prediction of the neural network. But if we normalize those features, values of the both features will lie in the range of 0 to 1.

Normalizing the input data for the neural network:
1. Find the mean of the feature.
2. Find the standard deviation of the feature.
3. Normalize the feature by first, subtracting the mean from the feature and then dividing it by the standard deviation. ((X - X.mean) /X.std)

Problem 2. *Explain how the following scenarios affect the learning process and what they represent:*
Scenario 1: A neural network's weights are all initialized to 0's.
Scenario 2: A neural network's weights are all initialized to 1's.

Solution: Scenario 1: A neural network's weights are all initialized to 0's.

If we initialize all the weights to be zero, then all the neurons of all the layers perform the same calculation (0 * input) and gives zero as output. Whole neural network will be same as single neuron.

Scenario 2: A neural network's weights are all initialized to 1's.

If we initialize all the weights to be ones, then all the neurons of all the layers perform the same calculation (sum of inputs) and gives sum of inputs as output. Whole neural network will be same as single neuron.

In both the above cases, every neuron in all the layers performs the same calculation. If given 0, output is 0. If given 1, output is sum of all inputs.

It is good idea to give, random weights or different weights to improve the neural network to arrive at better answer.

Three things to consider while choosing weight initialization:
1. Weights should not be same.
2. Weights should not be too small or too large.
3. There should be good variance between the weights.

Problem 3. *There are plenty of activation functions used to allow for non-linear transformations of data in neural networks. In the early days, the sigmoid activation function was a popular choice. It has one aw that was resolved by introducing the ReLU activation function. Explain and discuss what are the following terms and how they are resolved: 1) vanishing gradients and 2) exploding gradients.*

Solution: Vanishing Gradient Problem:
When the backpropagation algorithm advances from output to input layer, where gradients shrink to zero, leaving the weights intact. This makes the gradient descent never reach the optimum. Sigmoid activation function transforms the larger input space to smaller output, there by creating a huge difference. Hence during the backpropagation there are gradients that keep on reducing from higher layers and remains nothing for lower layers.

Solution:
Rectified Linear Units(ReLUs) activation function is mainly used to replace the sigmoid function, because f(x) = max(0, x) this saturates on one direction and is more robust in hidden layers of complex networks. ReLU derivate is '1' for positive values and for negative values it's '0' which helps in maintaining the value intact there by preventing the vanishing gradient problem.

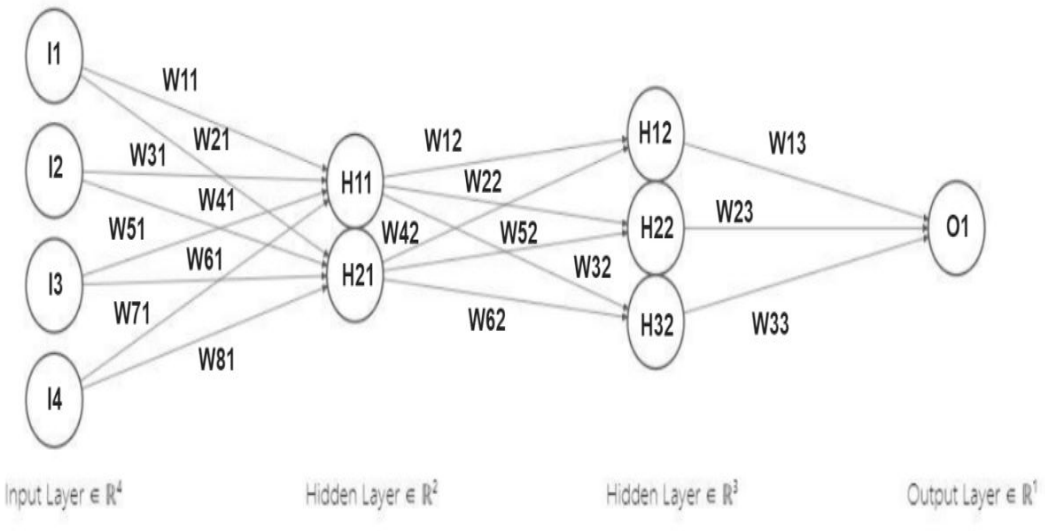
Exploding Gradient Problem:
In few cases opposite to vanishing gradient problem, the gradients keep on increasing as the

backpropagation algorithm advances from output to input layer, which eventually increase the weights drastically and making the gradient descent to diverge from the optimum. Due to the large losses generated by initial weights and in the next layer this weights gets accumulated and results in network instability and incapable of learning from the training data.

Solution:
ReLU causes exploding gradient problem when the weights are large. Bias plays a key role in the activation function, though it's defaulted to '1', if we set the bias value to '0.1' in ReLU, so when weights are assigned with smaller values, they are centered to zero and most of the entries will result '0'. But in some cases if the output values are zeroes, then there's a scope of growing in size of weights, there by bringing instability in the model. Clipnorm is an approach which is used to overcome this, deploying regularization L1 and L2 norms helps in reducing the generalization errors.

Problem 4. *Consider a neural network with two hidden layers: p = 4 input units, 2 units in the first hidden layer, 3 units in the second hidden layer, and a single output.*
a) Draw a picture of the network.
b) Write out an expression for f(X) (assume ReLUs). Be as explicit as you can.
c) Plug in some values for the coefficients and write out the value of f(X).
d) How many parameters are there?

Solution:
4.a)



4.b)

$$x = b + \sum_{i=1}^{n} I_i w_i$$

b - bias

$w_i - weights$

$I_i - Inputvalues$

$$ReLU Activation, f(x) = max(0, x)$$

$$ReLU Activation = f(x) = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases}$$

4.c)

$$I1 = 2, I2 = 4, I3 = 6, I4 = 8, b = 0$$

W11 = 1, W21 = 2, W31 = 3, W41 = 4, W51 = 5

W61 = 6, W71 = 7, W81 = 8

H11 = I1W11 + I2W31 + I3W51 + I4W71 + b = 2 + 12 + 30 + 56 + 0 = 100
f(x) = max(0, 100) = 100

H21 = 4 + 16 + 36 + 64 = 120
f(x) = max(0, 120) = 120

W12 = 1, W22 = 2, W32 = 3, W42 = 4, W52 = 5, W62 = 6
H12 = 100 + 480 = 580
H22 = 200 + 600 = 800
H32 = 300 + 720 = 1020

W13 = 1, W23 = 0, W33 = 0
O1 = 580 + 0 + 0 = 580

4.d)

Parameters between Input Layer and H1 = 2 + (4 x 2) = 2 + 8 = 10

Parameters between H1 and H2 = 3 + (2 x 3) = 3 + 6 = 9

Parameters between H2 and Output = 1 + (3 x 1) = 4

Total Parameters = 10 + 9 + 4 = 23

Problem 5. *What is a saddle point? Which is more likely, getting stuck in a local minima or in a saddle point? Explain why and show proof. (Hint: critical points, number of dimensions)*

Solution: Saddle points are the critical points in a multivariable function where the function attains neither a local maxima or minima.

Critical points of a function are the points of the function where either the first derivative is '0' or doesn't exist.

In other words, if c = f(a, b), the point (a, b, c) is said to be saddle point if both partial derivatives become '0' or doesn't exist, but the function 'f' attains neither maximum or minimum values at (a, b).

Second Order Partial Derivative Tests are used to find the saddle points.

Step 1: f_{xx}(a, b), f_{yy}(a, b), f_{xy}(a, b) partial derivatives are calculated.

Step 2: Discriminant D = f_{xx}(a, b) * f_{yy}(a, b) - $[f_{xy}(a, b)]^2$.

If D ¡ 0, then f has a saddle point at (a, b).

Example:

$$f(x, y) = x^2–y^2$$

First Order
f_x(x, y) = 2x
f_y(x, y) = -2y

Second Order
f_{xx}(x, y) = 2
f_{yy}(x, y) = -2
f_{xy}(x, y) = 0

D = 2 * -2 - $0^2 = -4 < 0, soat(0, 0)ithasasaddlepoint.$

In the optimization problem, when the model is DNN trained local minima is not a problem because it'll vanish as the number of dimensions increase, but the saddle points persist and loss functions stuck there.

We all know that the saddle, minima and maxima are calculated by computing the second order partial derivative and followed by discriminant(D) of Hessian matrix of loss

function(Discriminant D = f_{xx}(a, b) * f_{yy}(a, b) - [f_{xy}(a, b)]2), based on the D value we determine the point existance. The observation here is saddle points doesn't follow the phenomenon of sharing the same negative directions across all dimensions of the function unlike local minima, through which we can infer that as the number of dimensions increase the existence of saddle point increases rather than the minima.

Problem 6. *The goal is to implement an end-to-end neural network with gradient descent from scratch using only Python and NumPy. You are provided starter code in Collab. You must comment each step in detail and explain your implementation to receive full points.*

Solution: Collab notebook link: Question 6

References

[1] Reference to normalizing inputs - used to answer Q1 - shorturl.at/cqsx0

[2] Reference to normalizing inputs - used to answer Q1 https://stackoverflow.com/questions/4674623/why-do-we-have-to-normalize-the-input-for-an-artificial-neural-network

[3] Reference to normalizing inputs - used to answer Q1 https://towardsdatascience.com/why-data-should-be-normalized-before-training-a-neural-network-c626b7f66c7d

[4] Reference to initializing weights - used to answer Q2 https://stackoverflow.com/questions/20027598/why-should-weights-of-neural-networks-be-initialized-to-random-numbers

[5] Reference to initializing weights - used to answer Q2 https://towardsdatascience.com/weight-initialization-techniques-in-neural-networks-26c649eb3b78

[6] Reference to initializing weights - used to answer Q2 https://machinelearningmastery.com/why-initialize-a-neural-network-with-random-weights/

[7] Reference to vanishing and exploding gradients - used to answer Q3 - shorturl.at/eltFL

[8] Reference to Q3 https://deepai.org/machine-learning-glossary-and-terms/relu

[9] Reference to Q4 - https://www.youtube.com/watch?v=68BZ5f7P94E

[10] Reference to Q4 - https://www.youtube.com/watch?v=6MmGNZsA5nI

[11] Reference to saddle points - used to answer Q5 - https://byjus.com/maths/saddle-points/

[12] Reference to saddle points - used to answer Q5 - https://datascience.stackexchange.com/questions/22853/local-minima-vs-saddle-points-in-deep-learning