Problem 1: Part A

F(x) = ||X||22

F(x) = ((i2)1/2)2

F(x) = i2

Can ignore the summation as n is 1.

= 2x

Problem 1: Part B

i – ||22

= 2(x-Got the derivative, could just put it in the answer for Part A but accounting for This works because this is the derivative at each point and we have the summation of all of the points already in the expression.

Now we set the derivative with summation = 0

(xi –

Do some algebra and we end up with:

i)/n

Problem 2: Part A

L(w) = ||x||1

L(w) = ||Xw – y||1­

Sizes: X(data) -> n\*c, w(weights) -> c\*1, y(labels) -> n\*1

Problem 2: Part B

No, because L1 norm isn’t differentiable at zero therefore gradient optimization cannot be used. The explanation is partly explained in Part C as well.

Problem 2: Part C

Part A was straightforward since we just need the difference from the estimate and actual value. Since L1 is the summation of ||x||, we can just replace Xw-y for x. For part B, it wouldn’t have a value at zero if we took the derivative as it’s non-convex there, there would be no unique global minimum so we can’t minimize the loss function. This would mostly be concerned with the loss function part of the 3-step recipe as we are calculating the losses from the prediction regarding the label.

Problem 3: Part A

I do not have an answer for this question:

I first tried to do the math by having it all listed out in vectors and matrices but there’s just a lot of numbers I would have to brute force. I did:

[X][WHI]+[BHI] = ZI where X is the input data and WHI and BHI is the weights and biases for each neuron in the hidden layer and ZI­ is the output the ReLu activation function will take in. I then wrapped the ReLu function around that so that it would either be zero or ZI­, I then did the following:

[BO]+ [WOI][ZI] = y, where BO­ is the bias for the output layer, WOI is the weights of the output layer,

and ZI is either o or the ZI from before and y is the predicted output.

The problem I keep running into is that this just leads to a method where I have to brute force numbers

for a solution, and this is a lot of trial and error.

I also tried just calculating it out straightforward for each x value listed on the datasets. This doesn’t

work out because that is even more brute forcing then the previous method.

I found it easier to setup an optimization using excel solver to find a solution than to do the math or code

It and my computer actually ran out of RAM for both datasets before finding a solution and I have 128

GB of RAM.

You can see how I had it setup in excel below:

A screenshot of a computer

Description automatically generated

Solver:

A screenshot of a computer

Description automatically generated

Examples of formulas used:

Hidden Layer:

A screenshot of a computer

Description automatically generated

ReLu activation function:A screenshot of a spreadsheet

Description automatically generated

Output Layer:

A screenshot of a computer

Description automatically generated

Boolean value for if the prediction and label match:

A screenshot of a graph

Description automatically generated

Sum of labels that match:

A screenshot of a graph

Description automatically generated

Problem 3: Part B­

L() = (yi – f(xi,))2

▽L() = ((yi – f(xi,)) \* ▽f(xi, ))

Problem 3: Part C

First layer values after plugging in the input(x) and parameters

Z1 = -1

Z2 = 3

Z3 = 1

Z4 = -1

After ReLu

Z1 = 0

Z2 = 3

Z3 = 1

Z4 = 0

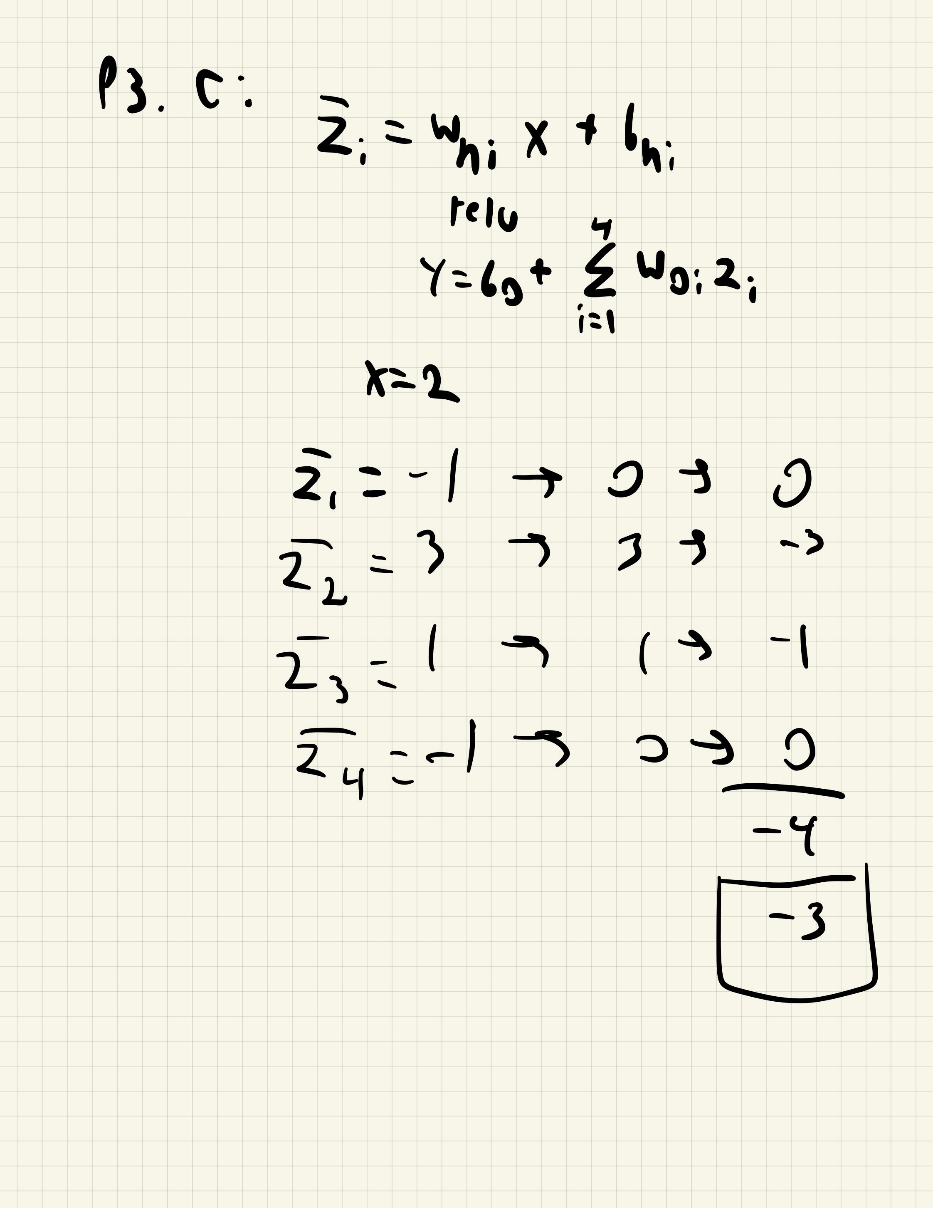
After summation in output layer

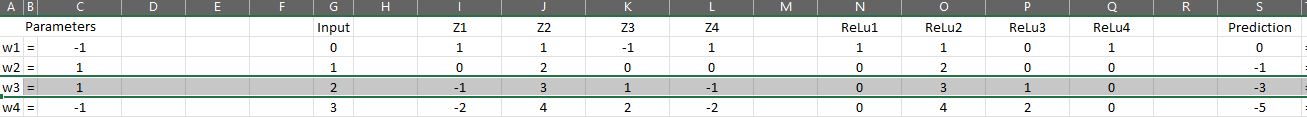
-4

After adding final bias

-3

Did this both manually and on the excel calculator I made:





Problem 3: Part D