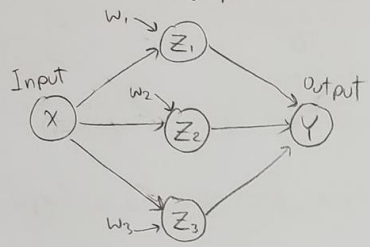
CS577 – Deep Learning - Homework 2

Niti Wattanasirichaigoon
A20406934

PART 1

1. Scalar Computation graph

a)



Z_i is logistic Activation:

$$f(x,y) = \frac{1}{1+\exp(-(w_0+w_1x_1+w_2x_2))}$$

Y is linear activation:

$$f(x,y) = V_0 + x_1V_1 + x_2V_2 + x_3V_3$$

b) Compute forward pass

for $x_1 = (1,2)$:

$$Z_1 = \frac{1}{1+\exp(-(0.01+0.02(1)+0.03(2)))} = \frac{1}{1+\exp(-0.09)} = 0.5225$$

$$Z_2 = \frac{1}{1+\exp(-(0.03+0.01(1)+0.02(2)))} = \frac{1}{1+\exp(-0.08)} = 0.5200$$

$$Z_3 = \frac{1}{1+\exp(-(0.02+0.03(1)+(0.01)(2)))} = \frac{1}{1+\exp(-0.07)} = 0.5175$$

$$Y = 0.01 + 0.02(0.5225) + 0.03(0.52) + (0.04)(0.5175) = 0.05675$$

for $x_2 = (1,3)$:

$$Z_1 = \frac{1}{1+\exp(-(0.01+0.02(1)+0.03(3)))} = \frac{1}{1+\exp(-0.12)} = 0.5300$$

$$Z_2 = \frac{1}{1+\exp(-(0.03+0.01(1)+(0.02)(3)))} = \frac{1}{1+\exp(-0.1)} = 0.5250$$

$$Z_3 = \frac{1}{1+\exp(-(0.02+0.03(1)+0.01(3)))} = \frac{1}{1+\exp(-0.08)} = 0.5200$$

$$Y = 0.01 + 0.02(.5300) + 0.03(0.525) + 0.04(0.52) = 0.05715$$

for $x_3 = (2,2)$:

$$Z_1 = \frac{1}{1+\exp(-(0.01+0.02(2)+0.03(2)))} = \frac{1}{1+\exp(-0.11)} = 0.5275$$

$$Z_2 = \frac{1}{1+\exp(-(0.03+0.01(2)+0.02(2)))} = \frac{1}{1+\exp(-0.09)} = 0.5225$$

$$Z_3 = \frac{1}{1+\exp(-(0.02+0.03(2)+0.01(2)))} = \frac{1}{1+\exp(-0.1)} = 0.5250$$

$$Y = 0.01 + 0.02(0.5275) + 0.03(0.5225) + 0.04(0.525) = 0.057225$$

---

c) Compute gradients of loss function

L2 loss: $L = (\hat{y}-y)^2$ $\quad \frac{\partial L}{\partial \hat{y}} = 2(\hat{y}-y)$

Let $\hat{y} = hv$ $\quad \frac{\partial \hat{y}}{\partial v} = h^T$ $\quad \frac{\partial L}{\partial v} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial v} = 2h^T(\hat{y}-y)$

$h = [1, Z_1, Z_2, Z_3]$

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial Z_i} \cdot \frac{\partial Z_i}{\partial w_i}$$

So for $x_1, y_1 = [(1,2), 8]$:

$$\frac{\partial L}{\partial v} = 2 \begin{bmatrix} 1 \\ 0.5225 \\ 0.52 \\ 0.5175 \end{bmatrix} (0.05675 - 8) = \begin{bmatrix} -15.89 \\ -8.30 \\ -8.26 \\ -8.22 \end{bmatrix}$$

$$\frac{\partial L}{\partial w_1} = 2(\hat{y}_1 - y_1)(0.02)(Z_1)(1-Z_1)\, x_1$$

$$= 2(0.05675 - 8)(0.02)(0.5225)(1-0.5225) \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$= \begin{bmatrix} -0.0793 \\ -0.1585 \end{bmatrix}$$

Using same methods... $\frac{\partial L}{\partial w_2} = \begin{bmatrix} -0.1190 \\ -0.2379 \end{bmatrix}$

$$\frac{\partial L}{\partial w_3} = \begin{bmatrix} -0.1587 \\ -0.3173 \end{bmatrix}$$

for $x_2, y_2 = [(1,3), 11]$:

$$\frac{\partial L}{\partial V} = 2\begin{bmatrix}1\\0.53\\0.525\\0.52\end{bmatrix}(0.05715 - 11) = \begin{bmatrix}-21.89\\-11.60\\-11.49\\-11.38\end{bmatrix}$$

$$\frac{\partial L}{\partial w_1} = 2(\hat{y}-y)\frac{\partial \hat{y}}{\partial z_1}\cdot\frac{\partial z_1}{\partial w_1} = \begin{bmatrix}-0.1090\\-0.3271\end{bmatrix}$$

$$\frac{\partial L}{\partial w_2} = \begin{bmatrix}-0.1637\\-0.4912\end{bmatrix}$$

$$\frac{\partial L}{\partial w_3} = \begin{bmatrix}-0.1639\\-0.4916\end{bmatrix}$$

for $x_3, y_3 = [(2,2), 10]$:

$$\frac{\partial L}{\partial V} = 2\begin{bmatrix}1\\0.5275\\0.5225\\0.5250\end{bmatrix}(0.057225 - 10) = \begin{bmatrix}-19.89\\-10.49\\-10.39\\-10.44\end{bmatrix}$$

$$\frac{\partial L}{\partial w_1} = \begin{bmatrix}-0.1949\\-0.1949\end{bmatrix}$$

$$\frac{\partial L}{\partial w_2} = \begin{bmatrix}-0.2977\\-0.2977\end{bmatrix} \qquad \frac{\partial L}{\partial w_3} = \begin{bmatrix}-0.3967\\-0.3967\end{bmatrix}$$

---

d) Compute gradients using formula (single output)

$$\frac{\partial L}{\partial V} = \sum_{i=1}^{m}(\hat{y}_i - y_i)z^{(i)} = \left(z^{(1)}(0.05675-8) + z^{(2)}(0.05715-11) + z^{(3)}(0.057225-10)\right)$$

$$= \begin{bmatrix}-7.943 & -4.150 & -4.130 & -4.110\end{bmatrix}$$
$$+ \begin{bmatrix}-10.943 & -5.800 & -5.745 & -5.690\end{bmatrix}$$
$$+ \begin{bmatrix}-9.943 & -5.245 & -5.195 & -5.220\end{bmatrix}$$
$$= \begin{bmatrix}-28.829 & -15.195 & -15.07 & -15.02\end{bmatrix}$$

$$\frac{\partial L}{\partial w_1} = \sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})V_1 z_1^{(i)}(1 - z_1^{(i)})X^{(i)}$$

$$\frac{\partial L}{\partial w_1} = \begin{bmatrix}-0.1933\\-0.3420\end{bmatrix} \qquad \frac{\partial L}{\partial w_2} = \begin{bmatrix}-0.2902\\-0.5134\end{bmatrix} \qquad \frac{\partial L}{\partial w_3} = \begin{bmatrix}-0.3869\\-0.6848\end{bmatrix}$$

The values correspond to the half of the sum of individual gradients of the points. This is because the loss function here is L2 loss.

2. Vector computation graph.

a) $f(x,y) = (2x+3y)^2$ $\quad \nabla f(x,y) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} 4(2x+3y) \\ 6(2x+3y) \end{bmatrix}$

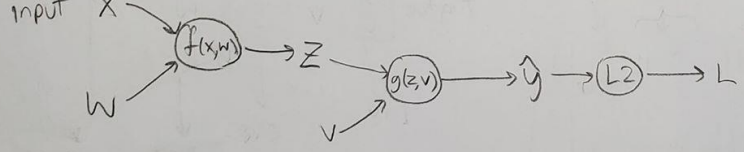
b) $F(x,y) = \begin{bmatrix} x^2+2y \\ 3x+4y^2 \end{bmatrix}$ $\quad DF(1,2) = \begin{bmatrix} \frac{\partial F_1}{\partial x} & \frac{\partial F_1}{\partial y} \\ \frac{\partial F_2}{\partial x} & \frac{\partial F_2}{\partial y} \end{bmatrix} = \begin{bmatrix} 2x & 2 \\ 3 & 8y \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 3 & 16 \end{bmatrix}$

c) $G(x) = \begin{bmatrix} x \\ x^2 \end{bmatrix}$ $\quad F \circ G(x) = \begin{bmatrix} x^2+2x^2 \\ 3x+4x^4 \end{bmatrix}$ $\quad D\,F \circ G(2) = \begin{bmatrix} 6x \\ 3+16x^3 \end{bmatrix} = \begin{bmatrix} 12 \\ 131 \end{bmatrix}$ without chain rule

$DF \circ G(2) = \begin{bmatrix} 2x & 2 \\ 3 & 8x^2 \end{bmatrix} \begin{bmatrix} 1 \\ 2x \end{bmatrix} = \begin{bmatrix} 2x+4x \\ 3+16x^3 \end{bmatrix} = \begin{bmatrix} 2(2)+4(2) \\ 3+16(2)^3 \end{bmatrix} = \begin{bmatrix} 12 \\ 131 \end{bmatrix}$ with chain rule

$DF(x,y)$ $\quad DG(x)$

d) $f(x,w) = \dfrac{1}{1+\exp(-(w_0+w_1x_1+w_2x_2))}$ $\quad g(z,v) = v_0+z_1v_1+z_2v_2+z_3v_3$ $\quad L2 = \sum(\hat{y}-y)^2$



$X = \begin{bmatrix} 1 & 2 \\ 1 & 3 \\ 2 & 2 \end{bmatrix}$ $\quad W = \begin{bmatrix} 0.01 & 0.02 & 0.03 \\ 0.03 & 0.01 & 0.02 \\ 0.02 & 0.03 & 0.01 \end{bmatrix}$ $\quad Z = \begin{bmatrix} 0.5225 & 0.52 & 0.575 \\ 0.53 & 0.525 & 0.52 \\ 0.5275 & 0.5225 & 0.525 \end{bmatrix}$

$V^T = \begin{bmatrix} 0.01 \\ 0.02 \\ 0.03 \\ 0.04 \end{bmatrix}$ $\quad \hat{y} = \begin{bmatrix} 0.05675 \\ 0.05715 \\ 0.057225 \end{bmatrix}$ $\quad L = 281.7$

backpropagation

$$\frac{\partial L}{\partial \hat{y}} = 2(\hat{y} - y)$$

$$\frac{\partial L}{\partial w_{ij}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_{ij}} = \sum_{k=1}^{m} \frac{\partial F}{\partial \hat{y}_k} \frac{\partial \hat{y}_k}{\partial w_{ij}}$$

$$\frac{\partial \hat{y}_k}{\partial w_{ij}} = \frac{\partial \hat{y}_k}{\partial z_i} \cdot \frac{\partial z_i}{\partial w_{ij}}$$

(v and x_j)

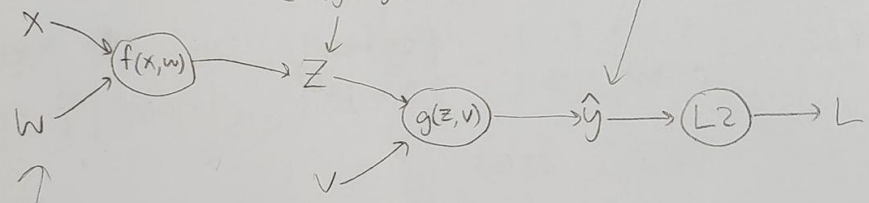
$$\frac{\partial L}{\partial w_{ij}} = \sum_{i=1}^{m} 2(\hat{y}_i - y_i) v_j x_i$$

$$\frac{\partial L}{\partial z_i} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_i}$$

($\frac{\partial \hat{y}}{\partial z_i} \to v$)

$$\frac{\partial L}{\partial z} = \begin{bmatrix} 2(\hat{y}_1 - y_1) v_1 \\ 2(\hat{y}_2 - y_2) v_2 \\ 2(\hat{y}_3 - y_3) v_3 \end{bmatrix}$$

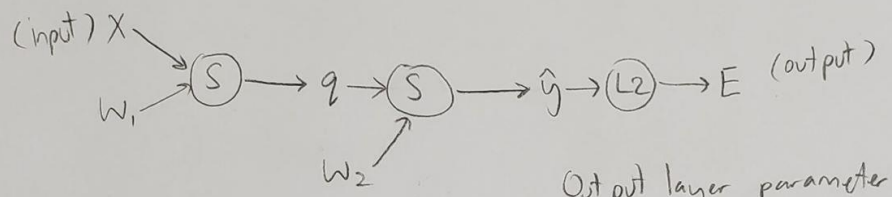
$$\frac{\partial L}{\partial \hat{y}} = \begin{bmatrix} 2(\hat{y}_1 - y_1) \\ 2(\hat{y}_2 - y_2) \\ 2(\hat{y}_3 - y_3) \end{bmatrix}$$



$$\frac{\partial L}{\partial w} = \begin{bmatrix} 2(\hat{y}_1 - y_1) v_1 x_1 & \cdots & 2(\hat{y}_1 - y_1) v_3 x_1 \\ \vdots & & \\ 2(\hat{y}_3 - y_3) v_1 x_3 & \cdots & 2(\hat{y}_3 - y_3) v_3 x_3 \end{bmatrix}$$

3. Assume 2-layer classification, single output

$\{x^{(i)}, y^{(i)}\}_{i=1}^{m}$ $\quad x^{(i)} \in \mathbb{R}^n$ $\quad y^{(i)} = \{0, 1\}$ $\quad$ L2 loss $= \frac{1}{2}\sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})^2$ (E)



Output layer parameter update:

$$\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_2} = \sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})\, q^{(i)}$$

$$w_2 = w_2 - \eta \sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})\, q^{(i)}$$

η: learning rate

Part 2

Problem Statement

Implement and test a three layer neural network for a three class classification using python and without using a GPU framework. Use categorical entropy for loss, sigmoid activation functions for hidden layers and softmax for the output layer. The computation tree and its forward and backward traversal should be hard coded in the program without having to support dynamic configurations. Evaluate the network you implemented using similar steps to that of the previous assignment.

Proposed Solution

1. **Network Structure Design:**
   This problem can be approached by many methods. I chose to code the network and gates as python classes. The network class (CGraph) will have a list containing layers and methods for executing forward, backward pass, parameter update, as well as methods for computing the loss and accuracies. The gate classes (sigmoid and softmax) will have methods for forwarding values and back-propagating derivatives. The gate classes will also store the weights for that layer which can be updated during back-propagation. The data that runs through and back the network is a numpy matrix and the output is a 3-categorical vector.

2. **Coding:**
   The neural network will be coded in python and then tested with a mock sample data set. We can test this small dataset with a small network (size 2-2-3) just to see if the data properly runs through and back the network. Epochs can be easily coded with a 'for loop', and we can observe the decrease in loss after the first few epochs.

3. **Model Evaluation:**
   The next step is to load sample datasets from the UCI ML repository and prepare data for a three classification problem. We then set the number of input layers to correspond with the number of features, select a number of nodes for the hidden layer, and test the model over 25 epochs. We adjust the learning rate if it does not converge. We then evaluate their performances compared to a random classifier, and if it performs better, our model is working.

Implementation Details

Coding the neural network was very challenging. Since I plan to use 2-dimensional tensors and inputs, I need to make sure what matrix multiplications are being carried at each step. One note worth mentioning is how the weights at each layer will have a bias term (w_0) that causes the matrices not able to multiply immediately. Matrix multiplication is done only up to the last line where the bias term is added later. While during back-propagation, an extra row of 1's are added for calculating the error that will be used to update the weight of the bias term.

I used the wine dataset and the seeds dataset from the UCI ML repository because they are already datasets for classification with 3 classes[1][2]. Although I still need to vectorize the responses into categorical form before using it with the neural network.

During training I realized that my model was always predicting only one result. I later discovered that it is because I fed the entire dataset into the model before updating an epoch. The update would result in favoring the label with the highest frequency. I fixed this problem by coding in the batch size to let the model update more frequently.

Results and Discussion

Wine Dataset

I trained the neural network model using 13 input nodes (corresponding to the 13 predictors), 8 nodes in the hidden layer, and 3 for the output. I also used a learning rate of 0.001, a regularization factor of 1e-1, trained for 20 epochs, and a batch size of 30 (since we only have 178 data points[1]).

```
wineNN = None
wineNN = CGraph(13,8,3)
wineNN.trainNetwork(xTrain,yTrain,20,30)
```

```
Epoch: 1 , Loss: 1.9099414531885541 , Accuracy: 0.2833333333333333
Epoch: 2 , Loss: 1.90894655233974 , Accuracy: 0.35
Epoch: 3 , Loss: 1.9079861125033224 , Accuracy: 0.3833333333333333
Epoch: 4 , Loss: 1.9070472897769775 , Accuracy: 0.3833333333333333
Epoch: 5 , Loss: 1.9061194733656417 , Accuracy: 0.3833333333333333
Epoch: 6 , Loss: 1.9051943756198582 , Accuracy: 0.3833333333333333
Epoch: 7 , Loss: 1.9042663974776464 , Accuracy: 0.3833333333333333
Epoch: 8 , Loss: 1.90333329743938 , Accuracy: 0.3833333333333333
Epoch: 9 , Loss: 1.9023971668001434 , Accuracy: 0.3833333333333333
Epoch: 10 , Loss: 1.9014656496688775 , Accuracy: 0.3833333333333333
Epoch: 11 , Loss: 1.9005532294256953 , Accuracy: 0.3833333333333333
Epoch: 12 , Loss: 1.8996822366025263 , Accuracy: 0.3833333333333333
Epoch: 13 , Loss: 1.8988830612929126 , Accuracy: 0.3833333333333333
Epoch: 14 , Loss: 1.8981929841693315 , Accuracy: 0.3833333333333333
Epoch: 15 , Loss: 1.8976532287078223 , Accuracy: 0.3833333333333333
Epoch: 16 , Loss: 1.8973043861553238 , Accuracy: 0.3833333333333333
Epoch: 17 , Loss: 1.897181159225 , Accuracy: 0.3833333333333333
Epoch: 18 , Loss: 1.8973079893943794 , Accuracy: 0.3833333333333333
Epoch: 19 , Loss: 1.8976970481183997 , Accuracy: 0.3833333333333333
Epoch: 20 , Loss: 1.898349123603391 , Accuracy: 0.3833333333333333
```

The model's accuracy stopped increasing after the 3rd epoch but the loss reached its minimum around 17 epochs. I retrained the model to 17 epochs and used it on the test data to validate its performance.

```
wineNN.testNetwork(xTest,yTest)
```

```
Loss: 1.85302980789532 , Accuracy: 0.4444444444444444
```

The model achieved an accuracy of **0.4444**, which is better than a random classifier (0.3333). The performance is considered good for a simple coded neural network.

Seeds Dataset

For this dataset we have 7 nodes in the input layer (7 features), 6 nodes in hidden layer and 3 in the output layer. I trained this over 25 epochs with a batch size of 10. We want to update the network more frequently because there is only a small number of features. After training, the optimal number of epochs is found to be 13. I then evaluated the model.

```
seedNN = CGraph(7,6,3)
seedNN.trainNetwork(xTrain,yTrain,13,10)
```

```
Epoch: 1 , Loss: 1.910005249072427 , Accuracy: 0.29375
Epoch: 2 , Loss: 1.9099125787244013 , Accuracy: 0.325
Epoch: 3 , Loss: 1.9098270530590333 , Accuracy: 0.275
Epoch: 4 , Loss: 1.9097475810088083 , Accuracy: 0.3375
Epoch: 5 , Loss: 1.909673165989819 , Accuracy: 0.35
Epoch: 6 , Loss: 1.9096027820845052 , Accuracy: 0.35
Epoch: 7 , Loss: 1.9095353139232754 , Accuracy: 0.35
Epoch: 8 , Loss: 1.9094696588149243 , Accuracy: 0.35
Epoch: 9 , Loss: 1.909405121011484 , Accuracy: 0.35
Epoch: 10 , Loss: 1.9093421360074712 , Accuracy: 0.35
Epoch: 11 , Loss: 1.9092830293229535 , Accuracy: 0.35
Epoch: 12 , Loss: 1.9092320582497808 , Accuracy: 0.35
Epoch: 13 , Loss: 1.9091940488436756 , Accuracy: 0.35
```

```
seedNN.testNetwork(xTest,yTest)
```

```
Loss: 1.910160844655512 , Accuracy: 0.30952380952380953
```

The accuracy on the test set happened to be only **0.3095**, which is lower than a random classifier. We may conclude that the model is still lacking much improvement or that the features of our data may be misleading towards the prediction.

References

[1] Wine Dataset – UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/wine

[2] Seeds Dataset – UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/seeds