Name: Connor Raymond Stewart
ID: 101041125

COMP 4107 Assignment Two:

① Find $\frac{\partial f}{\partial x}$ if:

a) $f = x_1 + x_2 + x_3,\ x = (x_1, x_2, x_3)$:

Since f is a scaler & x is a vector, we use Scalar-by-vector Conversion:

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \frac{\partial f}{\partial x_3} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x_1}(x_1 + \cancel{x_2} + \cancel{x_3}) & \frac{\partial}{\partial x_2}(\cancel{x_1} + x_2 + \cancel{x_3}) & \frac{\partial}{\partial x_3}(\cancel{x_1} + \cancel{x_2} + x_3) \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

b) $f = (x_1 + x_2,\ x_3 x_2,\ x_1 - x_3^2,\ x_2 - 8),\ x = (x_1, x_2, x_3)$:

Since f is a vector & x is a vector, use vector-by-vector conversion:

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} \\ \frac{\partial f_4}{\partial x_1} & \frac{\partial f_4}{\partial x_2} & \frac{\partial f_4}{\partial x_3} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x_1}(x_1 + \cancel{x_2}) & \frac{\partial}{\partial x_2}(\cancel{x_1} + x_2) & \frac{\partial}{\partial x_3}(\cancel{x_1 + x_2}) \\ \frac{\partial}{\partial x_1}(\cancel{x_3 x_2}) & \frac{\partial}{\partial x_2}(x_3 x_2) & \frac{\partial}{\partial x_3}(x_3 x_2) \\ \frac{\partial}{\partial x_1}(x_1 - \cancel{x_3^2}) & \frac{\partial}{\partial x_2}(\cancel{x_1 - x_3^2}) & \frac{\partial}{\partial x_3}(\cancel{x_1} - x_3^2) \\ \frac{\partial}{\partial x_1}(\cancel{x_2 - 8}) & \frac{\partial}{\partial x_2}(x_2 - \cancel{8}) & \frac{\partial}{\partial x_3}(\cancel{x_2 - 8}) \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & 0 \\ 0 & x_3 & x_2 \\ 1 & 0 & 2x_3 \\ 0 & 1 & 0 \end{bmatrix}$$

c) $f = (x, 2x^3, -\cos 2x + 14)$:

Since f is a vector & x is a scaler, use vector-by-scalar:

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x} \\ \frac{\partial f_2}{\partial x} \\ \frac{\partial f_3}{\partial x} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x}(x) \\ \frac{\partial}{\partial x}(2x^3) \\ \frac{\partial}{\partial x}(-\cos 2x + \cancel{14}) \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \cdot 3 \cdot x^2 \\ -(-2\sin 2x) \end{bmatrix} = \boxed{\begin{bmatrix} 1 \\ 6x^2 \\ 2\sin(2x) \end{bmatrix}}$$

d) $f = \begin{bmatrix} 9x & 3y \\ 2 & \sin(xy) \end{bmatrix},\ \frac{\partial y}{\partial x} = c$, Hint: $\frac{\partial xy}{\partial x} = 2cx + b$:

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial}{\partial x}(9x) & \frac{\partial}{\partial x}(3y) \\ \frac{\partial}{\partial x}(2) & \frac{\partial}{\partial x}(\sin(xy)) \end{bmatrix} = \begin{bmatrix} 9 & \frac{\partial y}{\partial x}(3) \\ 0 & \cos(xy) \cdot \frac{\partial(xy)}{\partial x} \end{bmatrix}$$

↳ As $\frac{\partial y}{\partial x} = c$ & $\frac{\partial xy}{\partial x} = 2cx + b$ we Note:

$$= \begin{bmatrix} 9 & 3c \\ 0 & \cos(xy)(2cx + b) \end{bmatrix} = \begin{bmatrix} 9 & 3c \\ 0 & (2cx + b)\cos(xy) \end{bmatrix}$$

thus:

$$\frac{\partial f}{\partial x} = \begin{bmatrix} 9 & 3c \\ 0 & (2cx + b)\cos(xy) \end{bmatrix}$$

e) $f = \begin{bmatrix} 3p & 2q \\ p-q & 7 \end{bmatrix}, p = x_1 - 2x_2, q = x_2 x_3, x = (x_1, x_2, x_3)$

$$f = \begin{bmatrix} 3(x_1 - 2x_2) & 2(x_2 x_3) \\ (x_1 - 2x_2) - (x_2 x_3) & 7 \end{bmatrix} = \begin{bmatrix} 3x_1 - 6x_2 & 2x_2 x_3 \\ x_1 - 2x_2 - x_2 x_3 & 7 \end{bmatrix}$$

Since f is a matrix & x is a vector, use a matrix-by-vector conversion:

thus, derive f by each x_i to get three matracies for a tensor

$$A_1 = \frac{\partial}{\partial x_1} \begin{bmatrix} 3x_1 - \cancel{6x_2} & \cancel{2x_2 x_3} \\ x_1 - \cancel{2x_2} - \cancel{x_2 x_3} & \cancel{7} \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 1 & 0 \end{bmatrix}$$

$$A_2 = \frac{\partial}{\partial x_2} \begin{bmatrix} \cancel{3x_1} - 6x_2 & 2x_2 x_3 \\ \cancel{x_1} - 2x_2 - x_2 x_3 & \cancel{7} \end{bmatrix} = \begin{bmatrix} -6 & 2x_3 \\ -2 - x_3 & 0 \end{bmatrix}$$

$$A_3 = \frac{\partial}{\partial x_3} \begin{bmatrix} \cancel{3x_1} - \cancel{6x_2} & 2x_2 x_3 \\ \cancel{x_1} - \cancel{2x_2} - x_2 x_3 & \cancel{7} \end{bmatrix} = \begin{bmatrix} 0 & 2x_2 \\ -x_2 & 0 \end{bmatrix}$$

So, the resulting tensor is:

$$\frac{\partial f}{\partial x} = \{A_1, A_2, A_3\}$$

② For a Perception model with the current weight vector $W=(-0.3, 2.1, 1)$ & the Current bias term $W_0=-0.2$, the input/output pair $(x,y)=((1,1,0),-1)$ is given. If our learning Constant $a=0.2$ & $C=0$, find the updated weights & the bias of the neuron after training for 3 iterations on this data point:

We Know the training Rule:

$$W_i \leftarrow W_i + \Delta W_i \quad \text{where} \quad \Delta w_i = n(t-o)X_i$$

Since $C=0$: (use bipolar mode)

$$O = \text{output} = \begin{cases} 1 & \text{if Sum } W_i * X_i > 0 \\ -1 & \text{otherwise} \end{cases}$$

As $a=0.2$ & $n=a$, $n=0.2$

We are targeting a value of -1, thus $t=-1$

thus, we have:

$t=-1$, $O=\text{output}$, $n=0.2$ & $W_i * X_i = W_1X_1 + W_2X_2 + W_3X_3 + W_0$

iteration One:

$$W_1 = (-0.3)(1) + (2.1)(1) + (1)(0) - 0.2 = 1.6 > 0$$

$\Rightarrow O = 1$

$\therefore$ Our output is 1, yet $y=-1$, So reweight

$$\Delta W_i = 0.2(-1-1)(1,1,0) = -0.4(1,1,0) = (-0.4, -0.4, 0)$$

$$W_i \leftarrow (-0.3, 2.1, 1) + (-0.4, -0.4, 0) = (-0.7, 1.7, 1)$$

iteration Two:

$t=-1$, $O=\text{output}$, $n=0.2$, $W_1=1.6$

$$W_2 = (-0.7)(1) + (1.7)(1) + (1)(0) + 1.6 = 2.6 > 0$$

$\Rightarrow O=1$ yet $y=-1$, So reweight

$$\Delta W_i = 0.2(-1-1)(1,1,0) = 0.2(-2)(1,1,0) = (-0.4, -0.4, 0)$$

$$W_i \leftarrow (-0.7, 1.7, 1) + (-0.4, -0.4, 0) = (-1.1, 1.3, 1)$$

iteration Three:

$t=-1$, $O=\text{Output}$, $n=0.2$, $W_2=2.6$

$$W_3 = (-1.1)(1) + (1.3)(1) + (1)(0) + 2.6 = 2.8 > 0$$

$\Rightarrow O=1$ yet $y=-1$, So rewrite

$$\Delta W_i = 0.2(-1-1)(1,1,0) = (-0.4, -0.4, 0)$$

$$W_i \leftarrow (-1.1, 1.3, 1) + (-0.4, -0.4, 0) = (-1.5, 0.9, 1)$$

thus, after three iterations we see that $W=(-1.5, 0.9, 1)$ & $W_3 = 2.8$

③ For each image, Pick the best number of layers for a hypothetical neural network Classifier. Cite the reason:

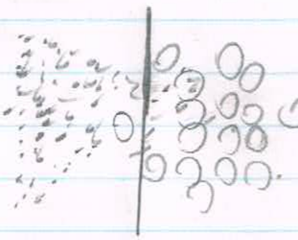
a) A Single layer neural network would work best.

Reasoning:

The region in diagram A is a half-Plane bounded by a Hyperplane, Since it is a region Seperable by a Single line.

For example:

Single layer



Note:

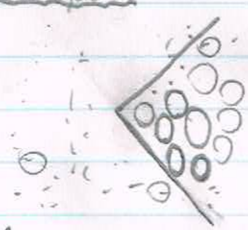
the o's & .'s are Seperable by a Single line. Also, outliers exist, but can't easily be included using more layers as this would disrupt the main relationship.

b) A double layer neural network would work best:

The region in diagram B is a Convex region since it is a region Seperable by a double line — V:

For example:

Two layer



Note:

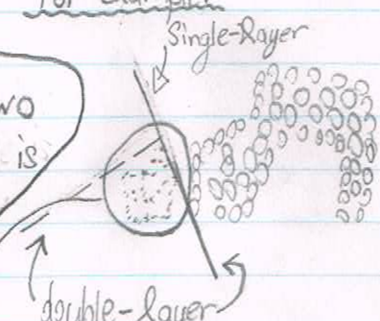
the V Seperates the .'s & o's. Outliers are mostly scattered & can't be included by a higher-order model.

c) A Single layer neural network would work, & a double can to:

The region in diagram C is a half-Plane bounded by a Hyperplane, Since it is a region Seperated by a Single line.

For example:

Single or Two layer (two is more accurate)



thus:

Single-layer's faster

double-layer's more accurate

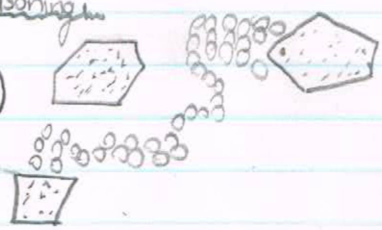
Note:

- the dashed line represents the Second layer which is optional. The Second layer can be used to Segment out the extra-empty white Space to make higher resolution results.
- The Single layer model excludes the dashed line, whereas the double layer model forms a V-Shape at the vertex & excludes the top half of the Solid line (it forms a Convex region)

d) A Three layer neural network would work since their are multiple disconnected clusters of points. This indicates a high-complexity model should be used:

Reasoning:

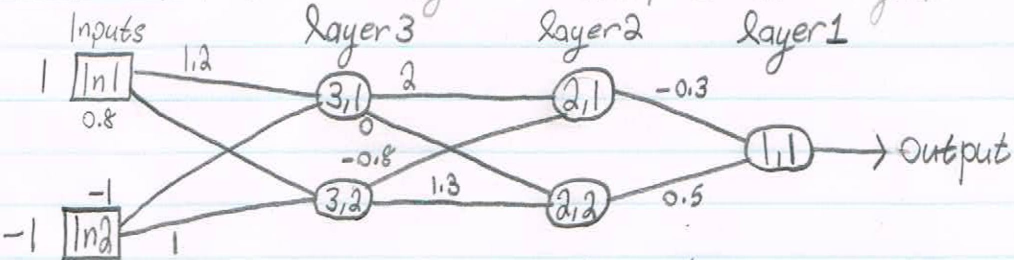
Three layer



Note:

this lets us separate the .'s from the O's, since we can skirt around the meshed regions between the .'s & O's.

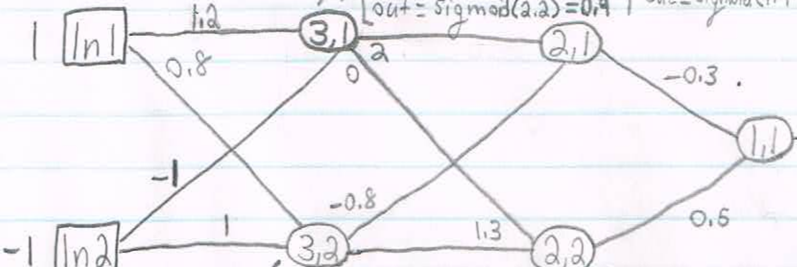
④ A multi-layered feed-forward network for classification is given. Suppose that no bias term exists in this particular network & all neurons have a sigmoid activation function. A data points (1,−1) with ~~label (1,−1) with~~ label 1 is given to this network to train. Calculate the weights of the neurons after 1 iteration of training for this data point with learning rate $\alpha = 1$:



Find Sigma outputs: $Sum_i^K = \sum w_{i,j} \cdot out_j^{K-1}$, $out_i^K = Sigmoid(Sum_i^K)$

Sum = 1(1.2) − 1(−1) = 2.2
out = Sigmoid(2.2) ≐ 0.9

Sum = 0.9(2) + 0.45(−0.8) = 1.44
out = Sigmoid(1.44) ≐ 0.808



Sum = 0.808(−0.3) + 0.64(0.5) = 0.0776
out = Sigmoid(0.0776) = 0.52

error = 1 − out = 1 − 0.52 = 0.48
Correct = error · out · (1 − out)
= 0.48(0.52)(1 − 0.52) = 0.119808
≐ 0.12

Sum = 1(0.8) − 1(1) = −0.2
out = Sigmoid(−0.2) ≐ 0.45

Sum = ~~0.9(2)~~ + 0.45(1.3) = 0.585
out = Sigmoid(0.585) = 0.64

Where LastNeuronError = desiredOutput − Output, Assume desiredOutput = 1

LastNeuronCorrect = LastNeuronError · Output · (1 − Output)

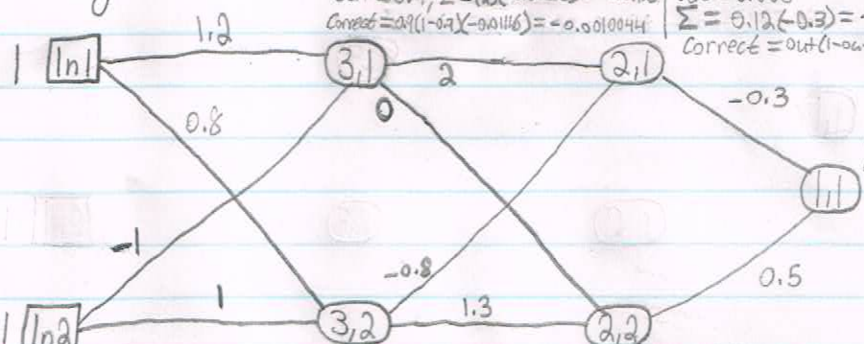
Now, we correct the neurons based on the data from above:

using $Correct_j^K = out_j^K (1 - out_j^K) \cdot \sum w_{i,j} \cdot Correct_i^{K+1}$

out = 0.9, Σ = (2)(−0.00558) = −0.01116
Correct = 0.9(1 − 0.9)(−0.01116) = −0.0010044

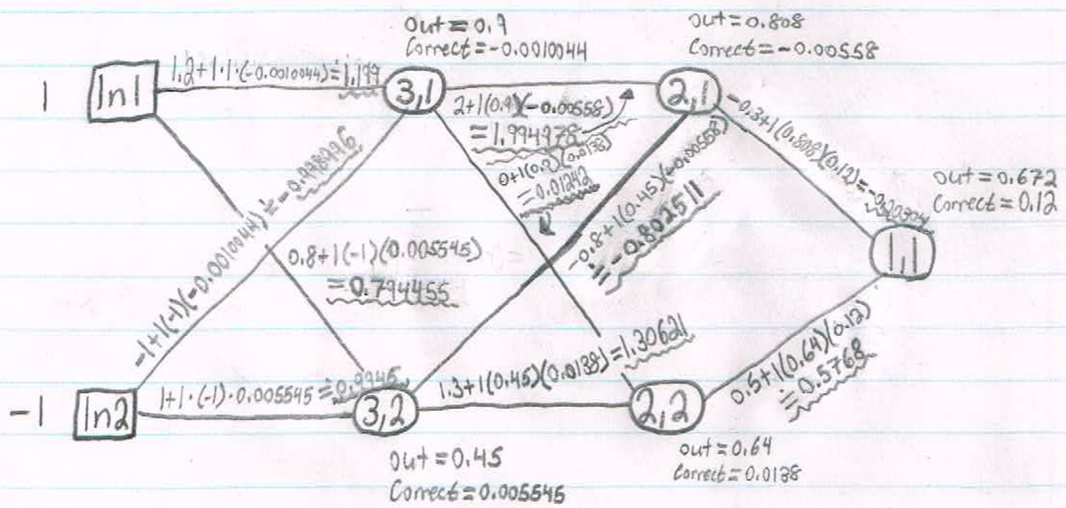
out = 0.808
Σ = 0.12(−0.3) = −0.036
Correct = out(1 − out)(Σ) = 0.808(1 − 0.808)(−0.036)
≐ −0.00558

out = 0.672
Correct = 0.12



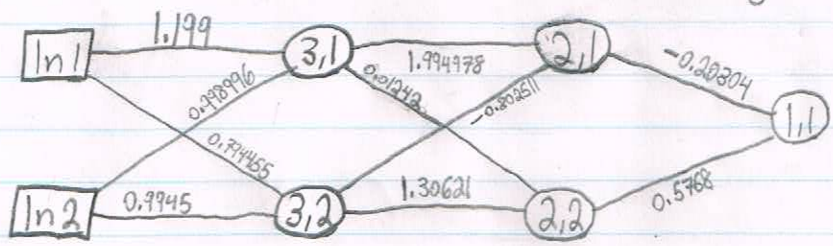
out = 0.45
Σ = −0.8(−0.00558)
+ 1.3(0.0138) = 0.022404
Correct = 0.45(1 − 0.45)(0.022404)
= 0.00554499 ≐ 0.005545

out = 0.64
Σ = 0.5(0.12) = 0.06
Correct = 0.64(1 − 0.64)(0.06) = 0.013824 ≐ 0.0138



Find the new weights using: $W_{i,j} = W_{i,j} + h \cdot out_j^K \cdot Correct_i^{K+1}$ given $h=\alpha=1$

So, we have the following network after 1 iteration of training:



⑤ We have trained a neural network on a dataset that Consists of a train set & a test set. We decide to Split the train set into 10 folds & do 10-fold Cross-Validation on the training before finding the final test accuracy by retraining the best network we get (in terms of Cross Validation accuracy) on the whole training set & then find the testing accuracy. After doing so, we have got these results:

Training accuracy: 81%

10-fold cross-Validation accuracy: 78%

Test accuracy: 59%

Explain why the test accuracy is lower than training or validation accuracy. What do you think happened here? Which of these results is the Correct result? What do you think Should be done in this Situation (or maybe this is not a problem in the first Place)?

Why the test accuracy is lower:

the testing accuracy is lower due to overfitting. When the training accuracy is greater then the testing, the model was overfitted. This is akin to having memorized an exact Pattern in the dataset without retaining the underlieing trend in the dataset. Thus, we end up making an overly elaborate model that fits the training set Perfectly but doesn't retain the trend (or Pattern) behind the data. Therefore, when testing the data on the testing set, we find the accuracy is low since we havn't acually learned the underlying rules behind the data. This is an error due to variance.

The Correct result is ultimatly the testing accuracy. Although the training data is Closely modeled by the program, it didn't Pickup on the true trend, meaning it "memorized" the dataset but didn't "learn" anything, which defeats the point of the task.

I think that the overfitting can be resolved by reducing the variance in the Sampling data, resulting in higher precision estimators in the dataset. This would reduce the rate of false treatments in the data & also prevent false variables from being in the model. We can also change the regularization parameters to increase weight decay (λ), which will Smooth out the models learning rate to form a Solid (rather then Squiggly) Curve to model the datas underlying trend with. Lastly, we can validate the dataset better. We currently are using a Simple 10-fold cross-validation model (1 of 10 datasets used validate), but we can ramp up the validation by using a 3-fold or leave-one-out model (uses 1/3 & 1/2 to validate). This would slow the Program since we would be wasting $\frac{1}{3}$ or $\frac{1}{2}$ data for validation, but it would greatly help to Prevent overfitting like we have in the Current model.