# cs577 Assignment 2

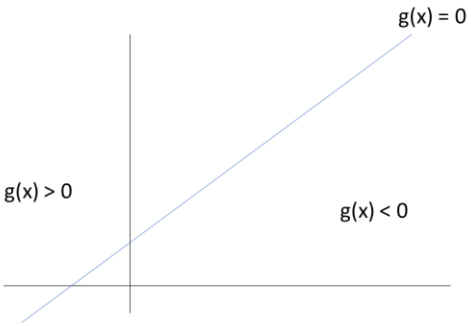
Jorge Gonzalez Lopez
A20474413
Department of Computer Science
Illinois Institute of Technology
February 9, 2021

# Part 1 (theoretical questions):

# Artificial neurons:

1. $w = [0.1, 0.2, 0.3]^T$
   $x = [1, 1]^T$

   $$y = w^T x = [0.1, 0.2, 0.3] \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = 0.6$$

2. 


   On the decision boundary its value would be zero and, on the sides, either greater than or lower than zero (as it can be seen in the Figure)

3. $g(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$

   $(\theta_0, \theta_1, \theta_2)$ are the values of the weights of the neuron. The neuron multiplies these weights to the inputs values to get the linear discriminant output.
   $\theta_0$: It is the distance to the origin (or bias)
   $(\theta_1, \theta_2)$: the normal vector of the decision boundary.

4. $g(x) = 1 + 2x_1 + 3x_2$

   The normal o the decision boundary: n = (2,3)
   The distance of the decision boundary from the origin: d = 1

5. To be able to write the discriminant function as $g(x) = \theta^T x$ :
   - The weights ($\theta$) have to be implemented as a column vector with the bias as the first element.

- The inputs (x) have to be implemented also as a column vector with the constant value of 1 in the first position to match the weights dimensions.

6.

Sigmoid function: $\sigma(z) = \frac{1}{1+e^{-z}}$

Step function: $h(z) = \begin{cases} 1 & if \quad z > 0 \\ 0 & otherwise \end{cases}$

The main advantage of the sigmoid function over the step function is that it can be used with gradient descent as it is differentiable for x = 0 and unlike the step function, its derivative is not zero for rest of the values (so it can progressively learn).
Finally, for very small values of z, the sigmoid function works pretty much as a linear output.

7.

$$\frac{p(y=1|x)}{p(y=0|x)} = \begin{cases} >1 \rightarrow c_1 \\ <1 \rightarrow c_0 \end{cases} \Leftrightarrow \log\left(\frac{p(y=1|x)}{p(y=0|x)}\right) = \begin{cases} >0 \rightarrow c_1 \\ <0 \rightarrow c_0 \end{cases}$$

$$\text{therefore} \rightarrow \log\left(\frac{p(y=1|x)}{p(y=0|x)}\right) = \theta^T x \Leftrightarrow \frac{p(y=1|x)}{p(y=0|x)} = e^{\theta^T x}$$

$$p(y=1|x) = p(y=0|x) \cdot e^{\theta^T x} = (1 - p(y=1|x)) \cdot e^{\theta^T x}$$

$$p(y=1|x) \cdot (1 + e^{\theta^T x}) = e^{\theta^T x} \Leftrightarrow \boxed{p(y=1|x) = \frac{e^{\theta^T x}}{1 + e^{\theta^T x}} = \frac{1}{1 + e^{-\theta^T x}}}$$

8.

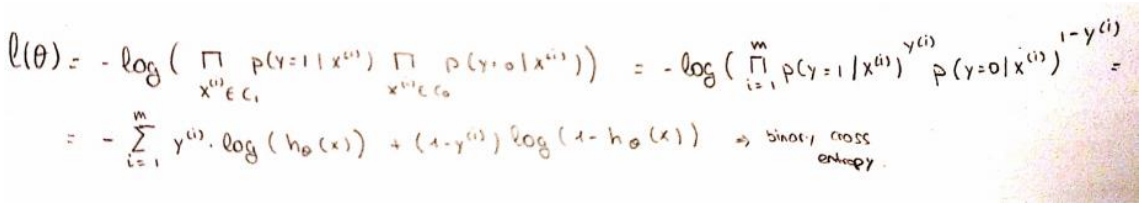
Derivative of a sigmoid: $\frac{\partial\, \sigma(z)}{\partial\, z} = \sigma(z)(1 - \sigma(z))$
Derivative of a logarithm: $\frac{\partial\, log(x)}{\partial\, x} = \frac{1}{x}$
Derivative of the log-sigmoid: $\frac{\partial \log(\sigma(z))}{\partial\, z} = \frac{1}{\sigma(z)}\sigma(z)\big(1 - \sigma(z)\big) = \big(1 - \sigma(z)\big)$

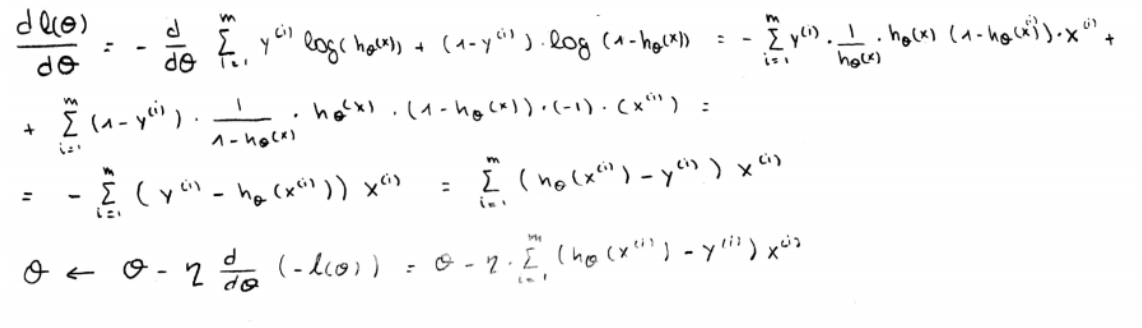
9. The direction used for updating the parameters in gradient descent is computed by doing the derivative of the loss function with respect to the parameters to try to minimize its value.
The size of the update is controlled by a hyperparameter call learning rate. This hyper-parameter scales the derivative when it gets subtracted from the value of the parameters in every update.

10. The stop condition for gradient descent is that if the loss function does no change much. Specifically, if the difference between the computed loss with the previous values of the parameters and the current ones is less than a defined threshold.
The condition uses the loss change as the sensitivity of the loss function with the changes in the parameters is unknown, hence a really small change in the parameters could increase significatively the value of the loss.

11. If the learning rate is too small, the model will converge very slowly and sometimes it may not get to the minimum as it will run out of time (if there is a timeout for training). If the learning rate is too large, the model instead of converging, the model will diverge and never get to the minimum.

12. The empirical error loss is computed by counting all the samples that have been misclassified (samples classified as c0 being c1 and vice versa), that is to say, the total number of errors.
    The problem of using this loss for gradient descent is that changes in theta may return the same exact value of loss.

13. 


14. 


15. One against all others: Each linear discriminant separates one class from the rest of them (binary classification). Hence, this approach works as a combination of n binary classifiers.
    One against each other: Each linear discriminant separates two different classes, ignoring the rest of them.

16. The rows of $\theta^T$ (columns of $\theta$) are templates, so there is one template per class. $\theta^T x$ measures how well x matches each of the templates and high similarity to a template of a particular class indicates high membership in that class.
    Using multiple linear discriminant functions just implies that there are many templates to be compared with the input values to determine under which category they are placed.

17. In a multi-class classifier, there are many possible output values. If a logistic function is used for these outputs, the sum of the probabilities of all the outputs would be greater than one. However, as all the possible outputs belong to the same class, the sum of their probabilities has to be equal to one. That is the reason that the softmax activation is used in the output layer of a multi-class classifier.

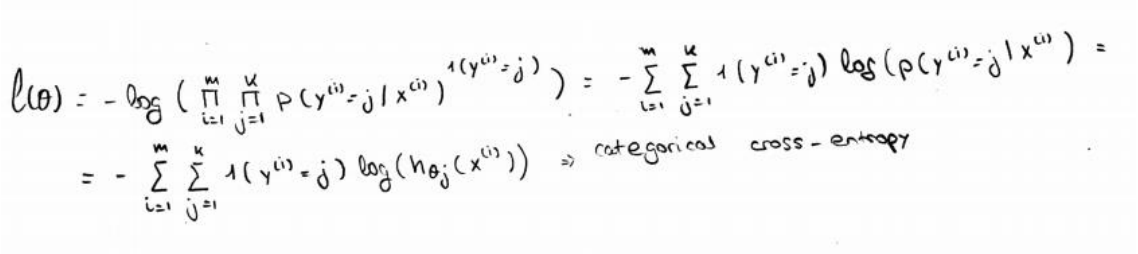
18.

Derivative of a softmax: $\frac{\partial\, softmax(z_j)}{\partial\, z_i} = softmax(z_j)(\delta_{i,j} - softmax(z_i))$

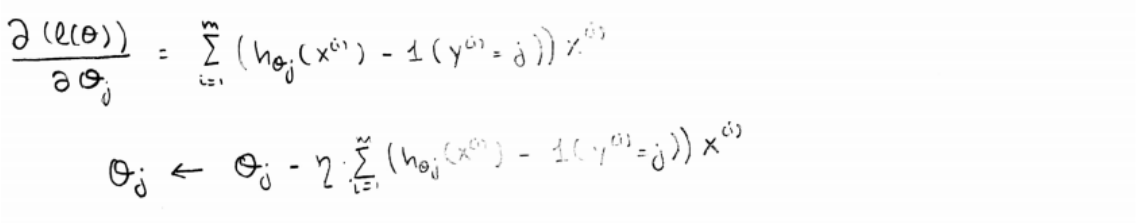
Derivative of a logarithm: $\frac{\partial\, log(x)}{\partial\, x} = \frac{1}{x}$

Derivative of the log-softmax: $\frac{\partial \log(softmax(z_j))}{\partial\, z_i} = \frac{1}{softmax(z_j)} softmax(z_j)\Big(\delta_{i,j} - softmax(z_i)\Big) = \delta_{i,j} - softmax(z_i)$

19.




20.



# Neural networks:

1. The dimensionality increase of the hidden layer can be interpreted as a non-linear map to higher dimensional spaces. Therefore, it is possible to benefit of this increase as it allows to find certain relations in high dimensional spaces which are harder to find with fewer dimensions.

2. The dimensionality decrease of the hidden layer can be interpreted as a condensation of the data as there are too many features that are correlated and depend on each other. Therefore, it is possible to benefit of this increase by subtracting all the features that do not provide useful information.

3. The update equations depend on the output of the layers. Hence, for the output layer, the update equations will depend on the actual output predictions of the model and for the hidden layer, the update equations will depend on its outputs (the inputs of the output layer).

4. Output layer: $\frac{\partial E}{\partial v} = \frac{\partial E}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial v} = \sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})\, z^{(i)}$
   Input layer: $\frac{\partial E}{\partial w_j} = \frac{\partial E}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z_j}\frac{\partial z_j}{\partial w_j} = \sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})v_j\, z_j^{(i)}(1 - z_j^{(i)})x^{(i)}$

5. Output layer: $\frac{\partial E}{\partial v_j} = \frac{\partial E}{\partial \hat{y}_j}\frac{\partial \hat{y}_j}{\partial v_j} = \sum_{i=1}^{m}(\hat{y}_j^{(i)} - y_j^{(i)})\, z^{(i)}$
   Input layer: $\frac{\partial E}{\partial w_j} = \sum_{l=1}^{k}\frac{\partial E}{\partial \hat{y}_l}\frac{\partial \hat{y}_l}{\partial z_j}\frac{\partial z_j}{\partial w_j} = \sum_{i=1}^{m}\sum_{l=1}^{k}\left(\hat{y}_l^{(i)} - y_l^{(i)}\right)v_{lj}\, z_j^{(i)}(1 - z_j^{(i)})x^{(i)}$

6. Output layer: $\frac{\partial E}{\partial v} = \frac{\partial E}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial v} = \cdots = \sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})\, z^{(i)}$
   Input layer: $\frac{\partial E}{\partial w_j} = \frac{\partial E}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z_j}\frac{\partial z_j}{\partial w_j} = \cdots = \sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})v_j\, z_j^{(i)}(1 - z_j^{(i)})x^{(i)}$

7. Output layer: $\frac{\partial E}{\partial v_j} = \frac{\partial E}{\partial \hat{y}_j}\frac{\partial \hat{y}_j}{\partial v_j} = \cdots = \sum_{i=1}^{m}(\hat{y}_j^{(i)} - y_j^{(i)})\, z^{(i)}$
   Input layer: $\frac{\partial E}{\partial w_j} = \sum_{i=1}^{k}\frac{\partial E}{\partial \hat{y}_i}\frac{\partial \hat{y}_i}{\partial z_j}\frac{\partial z_j}{\partial w_j} = \cdots = \sum_{i=1}^{m}\sum_{l=1}^{k}\left(\hat{y}_l^{(i)} - y_l^{(i)}\right)v_{lj}\, z_j^{(i)}(1 - z_j^{(i)})x^{(i)}$

8. The weights in a neural network must be initialized randomly and with low values, so that the symmetry is broken, and the not all neurons converge to the same solution.

# Computational Graphs:

1. The main advantage of computation graphs is that they show in a simple and straightforward way all the structure of the models, easing the task of computing the forward and backward propagation. Specifically, it helps carrying out the gradient descent in big neural networks as the nodes carry out simple operations with simple derivatives. Each node is in charge of performing a mathematical operation that can be a sum, a multiplication, a specific function (sigmoid, L2…), etc.
   During the forward pass, each node computes and returns the value of its function given some inputs and in the backward pass the node computes its local gradients and outputs the product of the incoming gradient with the local gradients.

2.

$$z = f_1(w_1, x)$$
$$\hat{y} = f_2(w_2, z) \quad \Big\} \quad \hat{y} = f_2(w_2, f_1(w_1, x))$$

$$\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_2} = \frac{\partial E_*}{\partial \hat{y}} \cdot \frac{\partial f_2(w_2, f_1(w_1, x))}{\partial w_2}$$

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_1} = \frac{\partial E_*}{\partial \hat{y}} \cdot \frac{\partial f_2(w_2, f_1(w_1, x))}{\partial w_1} = \frac{\partial E_*}{\partial \hat{y}} \; f_2(w_2, f_1(w_1, x)) \cdot \frac{\partial f_1(w_1, x)}{\partial w_1}$$

* $\frac{\partial E_{L_2}}{\partial \hat{y}} = \frac{\partial}{\partial \hat{y}} \left( \frac{1}{2} \sum_{i=1}^{m} (\hat{y} - y)^2 \right) = \frac{2}{2} \cdot \sum_{i=1}^{m} (\hat{y} - y) = \sum_{i=1}^{m} (\hat{y} - y)$

* $\frac{\partial E_{cross\ entropy}}{\partial \hat{y}} = \frac{\partial}{\partial \hat{y}} \left( -\sum_{i=1}^{m} y \log(\hat{y}) + (1-y) \log(1-\hat{y}) \right) = -\sum_{i=1}^{m} \left[ y \cdot \frac{1}{\hat{y}} + (1-y) \cdot \frac{-1}{1-\hat{y}} \right] =$

$$= -\sum_{i=1}^{m} \left[ \frac{y(1-\hat{y}) - \hat{y}(1-y)}{(1-\hat{y})\hat{y}} \right] = -\sum_{i=1}^{m} \frac{y - \hat{y}}{(1-\hat{y})\hat{y}} = \sum_{i=1}^{m} \frac{\hat{y} - y}{(1-\hat{y})\hat{y}}$$

3.

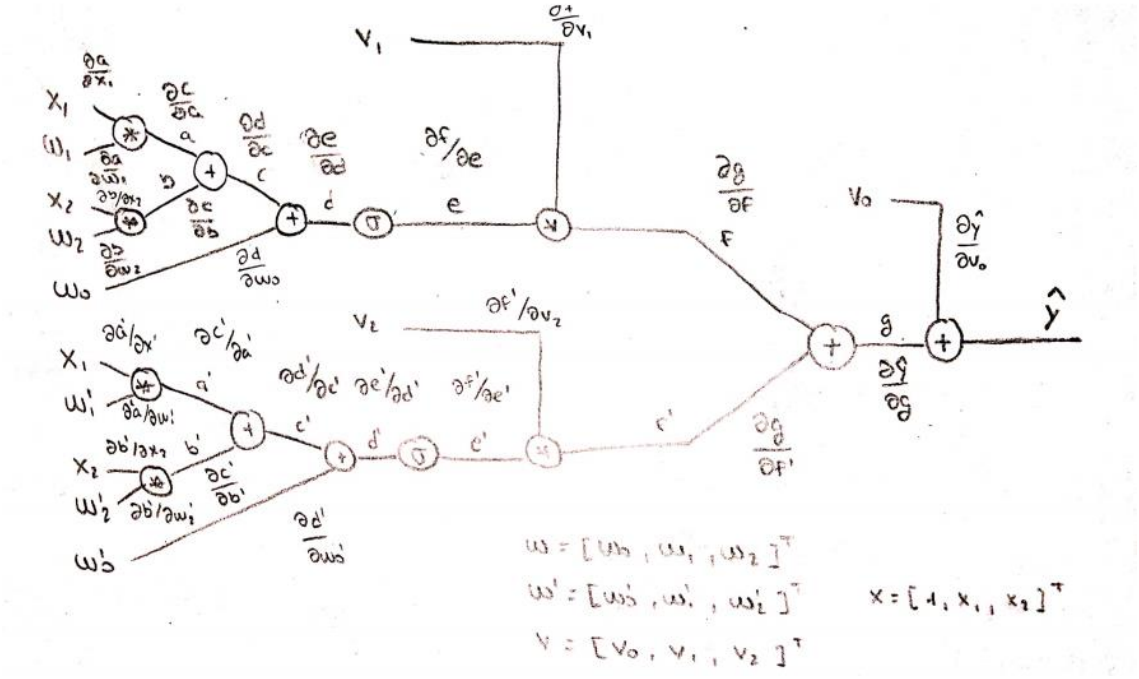
The same; but taken into consideration all outputs.

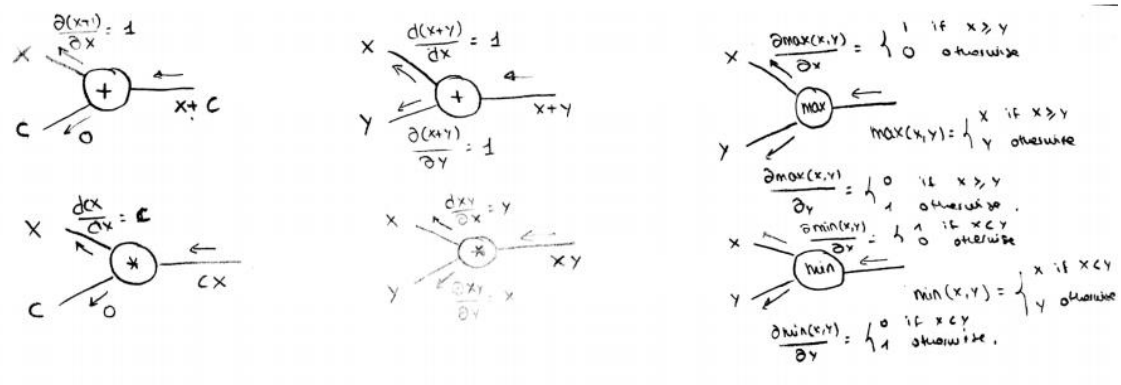
* $\frac{\partial E_{L_2}}{\partial \hat{y}} = \frac{\partial}{\partial \hat{y}} \left( \frac{1}{2} \sum_{i=1}^{m} \sum_{e=1}^{K} (\hat{y}_e - y_e)^2 \right) = \sum_{i=1}^{m} \sum_{e=1}^{K} (\hat{y}_e - y_e)$

* $\frac{\partial E_{cross}}{\partial \hat{y}} = \dots = \sum_{i=1}^{m} \sum_{e=1}^{K} \frac{\hat{y}_e - y_e}{(1-\hat{y}_e)\hat{y}_e}$

4.

5.





6. When the input to a node is a vector and the output is a scalar, the derivative expected is also a vector (rank 1 tensor) with the following components:

$$\frac{\partial \hat{y}}{\partial w} = [\frac{\partial \hat{y}}{\partial w_0}, \frac{\partial \hat{y}}{\partial w_1}, \dots, \frac{\partial \hat{y}}{\partial w_n}]^T$$

7. When the input to a node is a vector and the output is a vector, the derivative expected is a matrix (rank 2 tensor) with the following components:

$$\frac{\partial \hat{y}}{\partial w} = \begin{bmatrix} \frac{\partial \hat{y}_0}{\partial w_0} & \dots & \frac{\partial \hat{y}_n}{\partial w_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \hat{y}_0}{\partial w_n} & \dots & \frac{\partial \hat{y}_n}{\partial w_n} \end{bmatrix}$$

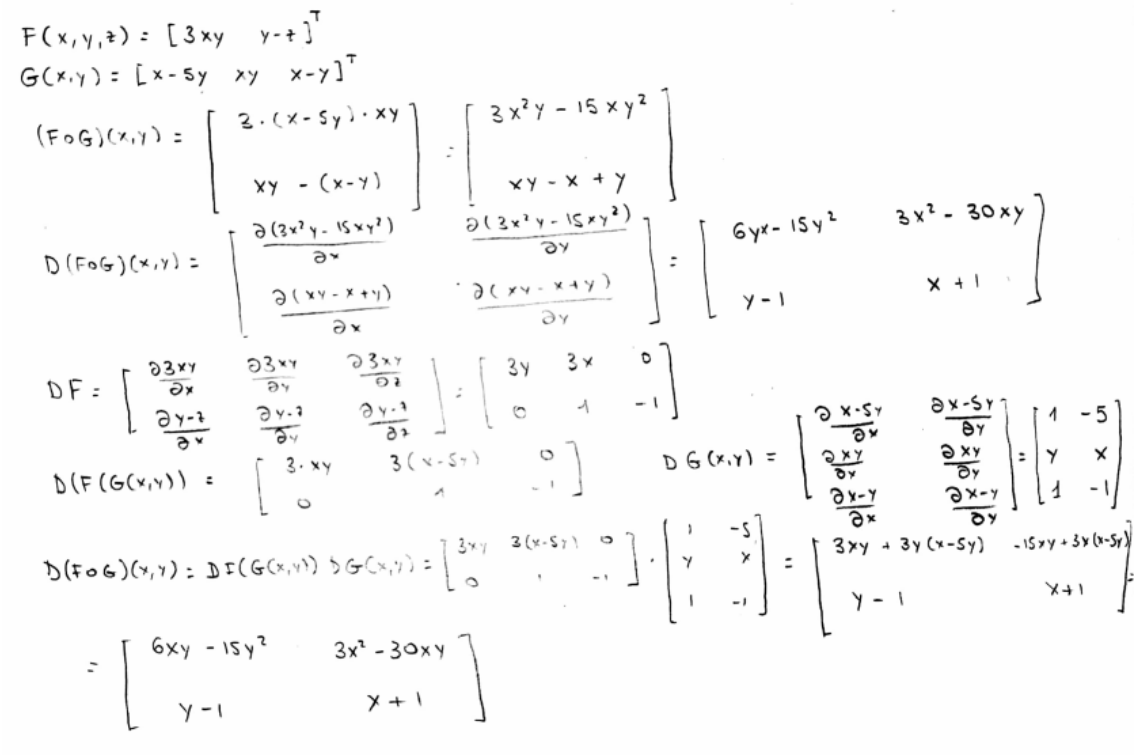
8. When the input to a node is a matrix and the output is a scalar, the derivative expected is also a matrix (rank 2 tensor) with the following components:

$$\frac{\partial \hat{y}}{\partial W} = \begin{bmatrix} \frac{\partial \hat{y}}{\partial w_{00}} & \cdots & \frac{\partial \hat{y}}{\partial w_{0n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \hat{y}}{\partial w_{n0}} & \cdots & \frac{\partial \hat{y}}{\partial w_{nn}} \end{bmatrix}$$

9.

$$D\big((F \circ G)(x,y)\big) = D\,F\,\big(\,G\,(x,y)\big)\,D\,G(x,y)$$

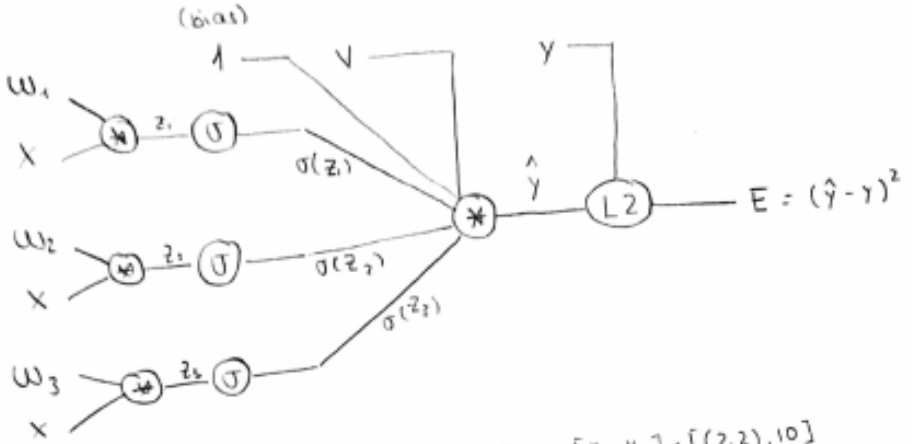
10.




11. The computational graph represents a network by breaking it down into small mathematical operations to ease the computation of the forward and backward propagation. Therefore, the nodes must be ordered correctly as their outputs (both in forward and backwards propagation) and the gradients they compute depend on the values of their inputs.

12.

a)




b) $[x_1, y_1] = [(1,2), 8]$ , $[x_2, y_2] = [(1,3), 11]$ , $[x_3, y_3] = [(2,2), 10]$

$w_1 = [0.01, 0.02, 0.03]^T$, $w_2 = [0.03, 0.01, 0.02]^T$, $w_3 = [0.07, 0.03, 0.01]^T$

$v = [0.01, 0.02, 0.03, 0.04]^T$

① $z_1 = [0.01, 0.02, 0.03] \cdot \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix} = 0.09 \rightarrow \sigma(z_1) = \frac{1}{1+e^{-z_1}} = 0.5225$

$z_2 = [0.03, 0.01, 0.02] \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix} = 0.08 \rightarrow \sigma(z_2) = \ldots = 0.52$

$z_3 = w_3^T \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix} = 0.07 \rightarrow \sigma(z_3) = \ldots = 0.5175$

$\hat{y} = [0.01, 0.02, 0.03, 0.04] \cdot \begin{bmatrix} 1 \\ 0.5225 \\ 0.52 \\ 0.5175 \end{bmatrix} = 0.05675 \Rightarrow E = (\hat{y} - y)^2 = (0.05675 - 8)^2 = 63,095$

② $z_1 = w_1^T \cdot \begin{bmatrix} 1 \\ 1 \\ 3 \end{bmatrix} = 0.12$ , $z_2 = w_2^T \cdot \begin{bmatrix} 1 \\ 3 \end{bmatrix} = 0.1$ , $z_3 = w_3^T \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix} = 0.08$

$\sigma(z_1) = 0.53$ $\sigma(z_2) = 0.525$ $\sigma(z_3) = 0.52$

$\hat{y} = v^T \cdot \begin{bmatrix} 1 \\ 0.53 \\ 0.525 \\ 0.52 \end{bmatrix} = 0.05715 \Rightarrow E = (\hat{y} - y)^2 = (0.05715 - 11)^2 = 119,746$

③ $z_1 = w_1^T \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix} = 0.11$ , $z_2 = w_2^T \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix} = 0.09$ , $z_3 = w_3^T \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix} = 0.1$

$\sigma(z_1) = 0.5275$ $\sigma(z_2) = 0.5225$ $\sigma(z_3) = 0.525$

$\hat{y} = v^T \cdot \begin{bmatrix} 1 \\ 0.5275 \\ 0.5275 \\ 0.525 \end{bmatrix} = 0.05723 \Rightarrow E = (\hat{y} - y)^2 = (0.05723 - 10)^2 = 98.859$

c) $\frac{\partial E}{\partial w_j} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{d\hat{y}}{d\sigma(z_j)} \cdot \frac{d\sigma(z_j)}{dz_j} \cdot \frac{dz_j}{dw_j} = 2 \cdot (\hat{y} - y) \cdot v_j \cdot \sigma(z_j) \cdot (1 - \sigma(z_j)) \cdot x$

$\frac{\partial E}{\partial v_j} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial v_j}$

* $\frac{\partial E}{\partial \hat{y}} = 2(\hat{y} - y)$ * $\frac{\partial \hat{y}}{\partial \sigma(z_j)} = v_j$ * $\frac{\partial \sigma(z_j)}{\partial z_j} = \sigma(z_j)(1 - \sigma(z_j))$ * $\frac{\partial z_j}{\partial w_j} = x$

** $\frac{\partial \hat{y}}{\partial v_j} = \sigma(z_j)$

d) ① $\frac{\partial E}{\partial w_1} = 2 \cdot (0.05675 - 8) \cdot 0.02 \cdot 0.5225 \cdot (1 - 0.5225) \cdot \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} -0.079 \\ -0.079 \\ -0.159 \end{bmatrix}$

$\frac{\partial E}{\partial w_2} = 2 \cdot (0.05675 - 8) \cdot 0.03 \cdot 0.52 \cdot (1 - 0.52) \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} -0.119 \\ -0.119 \\ -0.238 \end{bmatrix}$

$\frac{\partial E}{\partial w_3} = 2 \cdot (0.05675 - 8) \cdot 0.04 \cdot 0.5175 (1 - 0.5175) \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} -0.1584 \\ -0.1584 \\ -0.3173 \end{bmatrix}$

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

② $\frac{\partial E}{\partial w_1} = \ldots = \begin{bmatrix} -0.109 \\ -0.109 \\ -0.327 \end{bmatrix}$ , $\frac{\partial E}{\partial w_2} = \ldots = \begin{bmatrix} -0.164 \\ -0.164 \\ -0.492 \end{bmatrix}$ , $\frac{\partial E}{\partial w_3} = \ldots = \begin{bmatrix} -0.219 \\ -0.219 \\ -0.656 \end{bmatrix}$ , $\frac{\partial E}{\partial v} = \ldots = \begin{bmatrix} -21.89 \\ -11.60 \\ -11.49 \\ -11.38 \end{bmatrix}$

③ $\frac{\partial E}{\partial w_1} = \ldots = \begin{bmatrix} -0.099 \\ -0.198 \\ -0.198 \end{bmatrix}$ , $\frac{\partial E}{\partial w_2} = \ldots = \begin{bmatrix} -0.149 \\ -0.298 \\ -0.298 \end{bmatrix}$ , $\frac{\partial E}{\partial w_3} = \ldots = \begin{bmatrix} -0.195 \\ -0.390 \\ -0.390 \end{bmatrix}$ , $\frac{\partial E}{\partial v} = \ldots = \begin{bmatrix} -19.89 \\ -10.49 \\ -10.39 \\ -10.44 \end{bmatrix}$

13.

a) $f(x,y) = (2x+3y)^2$ $\quad \nabla f(x,y) = \left[\frac{\partial f(x,y)}{\partial x} \quad \frac{\partial f(x,y)}{\partial y}\right]^T$

$$\nabla F(x,y) = \begin{bmatrix} 2\cdot(2x+3y)\cdot 2 \\ 2(2x+3y)\cdot 3 \end{bmatrix} = \begin{bmatrix} 8x+12y \\ 12x+18y \end{bmatrix}$$

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

$$DF(1,2) = \begin{bmatrix} 2 & 2 \\ 3 & 16 \end{bmatrix}$$

c) $G(x) = \begin{bmatrix} x \\ x^2 \end{bmatrix} \rightarrow (F\circ G)(x,y) = \begin{bmatrix} x^2+2x^2 \\ 3x+4x^4 \end{bmatrix}$

$$D(F\circ G)(x) = \begin{bmatrix} 6x \\ 3+16x^3 \end{bmatrix}$$

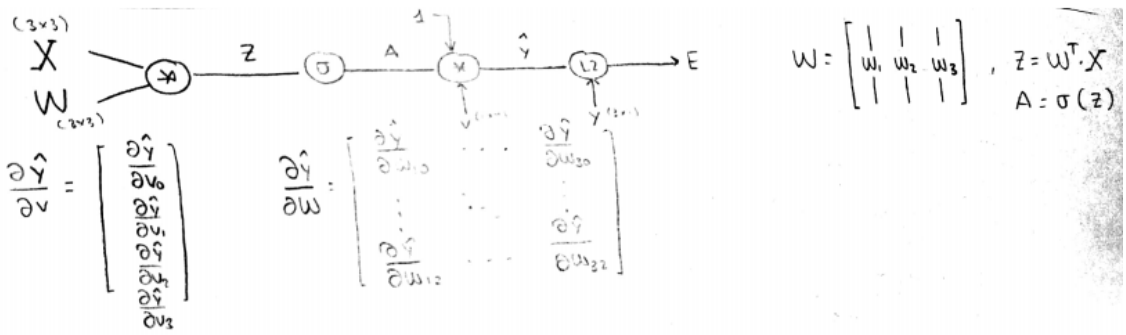
$$D(F\circ G)(2) = \begin{bmatrix} 12 \\ 131 \end{bmatrix}$$

→ Chain rule: $DF(G(x)) = \begin{bmatrix} 2x & 2 \\ 3 & 8x^2 \end{bmatrix}$ $\quad DG(x) = \begin{bmatrix} 1 \\ 2x \end{bmatrix}$

$$D(F\circ G)(x) = \begin{bmatrix} 2x & 2 \\ 3 & 8x^2 \end{bmatrix}\cdot\begin{bmatrix} 1 \\ 2x \end{bmatrix} = \begin{bmatrix} 2x+4x \\ 3x+16x^3 \end{bmatrix} = \begin{bmatrix} 6x \\ 3x+16x^3 \end{bmatrix}$$

$$D(F\circ G)(2) = \begin{bmatrix} 12 \\ 131 \end{bmatrix}$$

d) Graph from last question with a vector notation




14. Assuming a L2 loss function and a sigmoid activation, the derivative of the output weights is the following:

$$\frac{\partial E}{\partial v} = \frac{\partial E}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial v} = \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)}\right)\hat{y}^{(i)}\left(1-\hat{y}^{(i)}\right)z^{(i)}$$

And the update equation for the output weights is:

$$v \leftarrow v - \eta \frac{\partial E}{\partial v}$$