CS 577

HW #2

Khalid Saifullah

ID : A20423546
Semester : Spring 2021
Date : 17th March 2021

Part 1 (theoretical questions)

Artificial neurons

Ans

Answer to the Question No. 1

$$\hat{y} = [0.2 \quad 0.3] \begin{bmatrix} 1 \\ 1 \end{bmatrix} + 0.1 = 0.6.$$

Answer to the Question No. 2

The linear discriminant should have positive values (that is, $g(x) > 0$) on one ~~side~~ side of the decision boundary and negative values (that is, $g(x) < 0$) on another side of the decision boundary. On the decision boundary, the linear discriminant function $g(x)$ should give 0.

Answer to the Question No. 3

In general, the hyperplane H (let's say) divides the feature space into two half spaces: decision region R_1 for Q_1 and region R_2 for Q_2 where θ_1 and θ_2 are the normal to any vector lying on the hyperplane and θ_0 is the negative distance from the origin.

Answer to the Question No. 4

Normal decision boundary $= \sqrt{\theta_1^2 + \theta_2^2} = \sqrt{2^2 + 3^2}$

~~Answer to the Question No.~~

Distance of the decision boundary from the origin $= -\frac{1}{\sqrt{2^2+3^2}} = -\frac{1}{\sqrt{13}}$

Answer to the Question No. 5

$$g(X) = \theta^T X + \theta_0 = \begin{bmatrix} n_1^T \\ \vdots \\ n_k^T \end{bmatrix} \begin{bmatrix} x \end{bmatrix} + \begin{bmatrix} d_1 \\ \vdots \\ d_k \end{bmatrix}$$

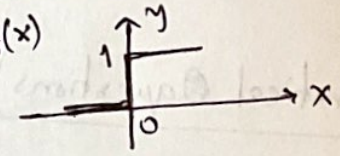
The bias coefficient in θ is denoted by θ_0 which is the first term of the vector θ. In order to write $g(x) = \theta^T x$, we need to put a 1 in the first row of x in order to match the dimensionality of x with θ^T which contains the bias coefficient.

Answer to the Question No. 6

Step activation function:

$f(x) = 1$, if $x >= 0$

$f(x) = 0$, if $x < 0$



Logistic (sigmoid) activation function:

$$\text{Sigmoid}(X) = \frac{1}{1 + \exp(-X)}$$

$$\text{Sigmoid}(\theta^T X) = \frac{1}{1 + \exp(-\theta^T X)}$$

The advantage sigmoid activation has over step activation is that it is non-linear. Beside it is also continuously differentiable.

As θ gets small, the output gets closer to 0.5.

Answer to the Question No. 7

The odds ratio $\frac{p}{1-p}$ describes the ratio between the probability that a certain, positive, event occurs and the probability that it doesn't occur – where positive refers to the event that we want to predict i.e. $P(y=1|X)$.

So, the more likely it is that the positive event occurs, the larger the odds' ratio.

$\frac{P(y=1|X)}{P(y=0|X)}$ If the ratio >1, then it's class 1

If the ratio <1, then it's class 0

Now, taking natural log of the odds' ratio,

$\log\left(\frac{P(y=1|X)}{P(y=0|X)}\right)$ If the ratio >0, then it's class 1

If the ratio <0, then it's class 0

The $\log\left(\frac{P(y=1|X)}{P(y=0|X)}\right)$ is modelled as a linear function by $\theta^T X$ which defines the linear discriminant and gives us the decision boundary.

So, $\log\left(\frac{P(y=1|X)}{P(y=0|X)}\right) = \theta^T X$

$$\frac{P(y=1|X)}{P(y=0|X)} = e^{\theta^T X}$$

$$P(y=1|X) = [1 - P(y=1|X)]\, e^{\theta^T X}$$

$$P(y=1|X)\left(1 + e^{\theta^T X}\right) = e^{\theta^T X}$$

$$P(y=1|X) = \frac{e^{\theta^T X}}{1 + e^{\theta^T X}}$$

$\therefore h_\theta(x) = P(y=1|x) = \frac{1}{1-e^{-\theta^T x}}$ where h is the hypothesis function, θ is the parameter and x is the input.

Thus, we modelled the log of probability as a linear discriminant and that gives us the sigmoid function as a predictor for $P(y=1|x)$.

Answer to the Question No. 8

$$\text{Sigmoid}(x) = \frac{1}{1+e^{-x}}$$

$$\frac{d}{dx}\{\text{Sigmoid}(x)\} = \text{Sigmoid}(x)\{1-\text{Sigmoid}(x)\}$$

$$\frac{d}{dx}\{\log(\text{Sigmoid}(x))\} = \frac{1}{\text{Sigmoid}(x)} \cdot \text{Sigmoid}(x)\{1-\text{Sigmoid}(x)\}$$

$$= 1-\text{Sigmoid}(x)$$

Again,

$$h_\theta(x) = \text{Sigmoid}(\theta^T x) = \frac{1}{1-e^{-\theta^T x}}$$

$$\frac{d}{d\theta} h_\theta(x) = \text{Sigmoid}(\theta^T x)\{1-\text{Sigmoid}(\theta^T x)\} x$$

$$= h_\theta(x)(1-h_\theta(x))x$$

$$\frac{d}{d\theta}\{\log(h_\theta(x))\} = \frac{1}{h_\theta(x)} \cdot h_\theta(x)\{1-h_\theta(x)\} x$$

$$= (1-h_\theta(x))x$$

Answer to the Question No. 9

The direction used for updating parameters in gradient descent is towards the minimum point of the cost function which is basically the direction specified by the negative gradient.

The size of the update is controlled by the learning rate.

Answer to the Question No. 10

Gradient descent stops when

$$J(\theta^{(i+1)}) - J(\theta^{(i)}) < \tau$$

In other words, when the cost function converges to a minimum value.

The condition should not use the parameter change as we are interested in minimizing the loss and don't know how sensitive the loss is to the change of parameter.

Answer to the Question No. 11

With a high learning rate, we can cover more ground each step, but we risk overshooting the lowest point since the slope of the hill is constantly changing. With a very low learning rate, we can confidently move in the direction of the negative gradient since we are recalculating it so frequently. A low learning rate is more precise, but calculating the gradient is time-consuming, so it will take us a very long time to get to the bottom.

Answer to the Question No. 12

The empirical error loss is computed by counting the total no. of misclassified examples.

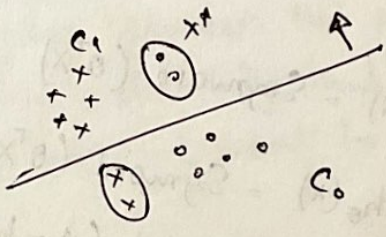
$$X^* = \{ x^{(i)} \mid 1(y^{(i)} = 0) \wedge \theta^T x > 0 \} \rightarrow \text{misclassified as } c_1.$$

$$X^{**} = \{ x^{(i)} \mid 1(y^{(i)} = 1) \wedge \theta^T x < 0 \} \rightarrow \text{misclassified as } c_0.$$

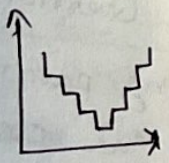
$E(\theta)$ = no. of X^* + no. of X^{**}

(= total no. of errors

↳ Empirical loss function



The empirical loss function, $E(\theta)$ is a piecewise constant function and we cannot apply gradient descent on piecewise constant functions. $E(\theta)$ is a piecewise constant function because it is possible to change θ without affecting $E(\theta)$.



Answer to the Question No. 13

$$l(\theta) = \log \left(\prod_{x^{(i)} \in c_1} P(y=1 \mid x^{(i)}) \prod_{x^{(i)} \in c_0} P(y=0 \mid x^{(i)}) \right)$$

↳ log likelihood of a binary classifier

If the classifier is good, i.e. every example belonging to class 1 is classified as 1 & every example belonging to class 0 is classified as class 0 then the product of the two probabilities will be very high (if perfect, it should be 1).

We can write $l(\theta)$ as

$$l(\theta) = \log \prod_{i=1}^{m} P(y=1 \mid x^{(i)})^{y^{(i)}} P(y=0 \mid x^{(i)})^{(1-y^{(i)})}$$

$$\boxed{l(\theta) = \sum_{i=1}^{m} y^{(i)} \log(\underbrace{P(y=1 \mid x^{(i)})}_{h_\theta(x)}) + (1-y^{(i)}) \log(1 - \underbrace{P(y=1 \mid x^{(i)})}_{h_\theta(x)})}$$

→ Binary Cross Entropy

Answer to the Question No. 14

$$\frac{d\,\ell(\theta)}{d\theta} = \frac{d}{d\theta} \sum_{i=1}^{m} y^{(i)} \log\left(h_\theta(x)\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_\theta(x)\right)$$

$$= \sum_{i=1}^{m} y^{(i)} \left(1 - h_\theta(x^{(i)})\right) x^{(i)} + \sum_{i=1}^{m} \left(1 - y^{(i)}\right) \left(-h_\theta(x^{(i)}) x^{(i)}\right)$$

$$= \sum_{i=1}^{m} \left(y^{(i)} - y^{(i)} h_\theta(x^{(i)})\right) - h_\theta(x^{(i)}) + y^{(i)} h_\theta(x^{(i)}) x^{(i)}$$

$$= \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)})\right) x^{(i)}$$

And $$\frac{d\left(-\ell(\theta)\right)}{d\theta} = \sum_{i=1}^{m} \left(h_\theta(x^{(i)} - y^{(i)}\right) x^{(i)}$$

So, the gradient descent update is

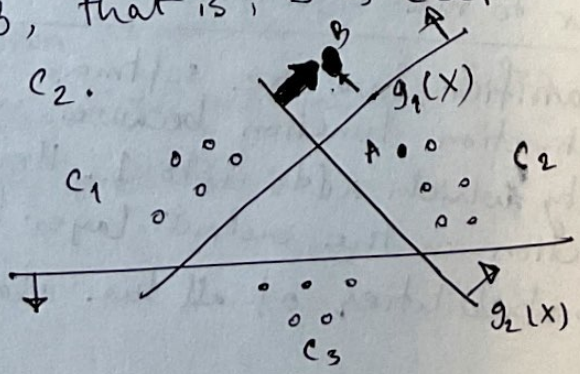
$$\theta \leftarrow \theta - \eta \sum_{i=1}^{m} \left(\underbrace{h_\theta(x^{(i)})}_{\hat{y}^{(i)}} - y^{(i)}\right) x^{(i)}$$

The update of the parameter turns out to be only subtracting the sum of the feature vectors, after multiplying with the learning rate, from the parameter in order to update it.

Answer to the Question No. 15

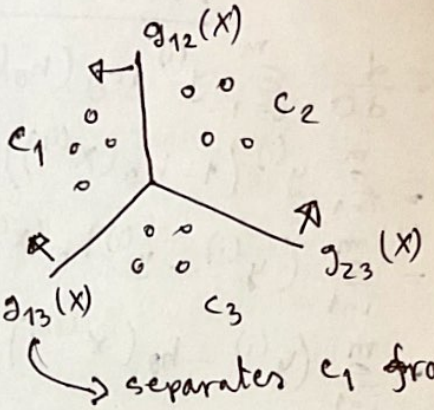
In 'one against all other' strategy for multi-class classification, we have k discriminant functions, $g_1(x) \ldots g_k(x)$. Here $g_1(x)$ is a binary classifier separating c_1 from both c_2 & c_3. A point A, for instance, is on the positive side of $g_2(x)$ but side of both $g_1(x)$ and $g_3(x)$. So, we select $g_2(x)$ as the linear discriminant and classify it as c_2. A point B, for instance, is on the positive side of both $g_1(x)$ and $g_2(x)$ but negative side of $g_3(x)$. However, the normal distance between $g_2(x)$ and B is greater than that of $g_1(x)$ and B, that is, B is deeper into $g_2(x)$ and so we classify it as c_2.

$$\hat{y} = \underset{j}{\text{argmax}}\; g_j(x)$$



In 'one against each other' strategy, we have $\frac{k(k-1)}{2}$ discriminant functions.

$$\hat{y} = \underset{i}{\operatorname{argmax}} \sum_{j \neq i} g_{ij}(x)$$



This strategy is easier than "one against all others" as it is easier to separate one class from another class than it is to separate one class from all other classes. However, we need more discriminants in this approach.

Answer to the Question No. 16

In the template matching interpretation of linear discriminants, the rows of $\Theta^T X$ are considered as templates, that is, with k rows of Θ^T, we have k templates (one template per class). And $\Theta^T X$ measures how well X matches each of the k templates (dot product of x with the rows of Θ^T). In other words, take X and project it on all the vectors (i.e. n_1 to n_k) and choose the one with the highest projection. $\hat{y} = \Theta^T X + \Theta_0$

$$= \begin{bmatrix} n_1^T \\ \vdots \\ n_k^T \end{bmatrix} \begin{bmatrix} x \end{bmatrix} + \begin{bmatrix} d_0 \\ \vdots \\ d_k \end{bmatrix}$$

$$= \begin{bmatrix} n_1 \cdot x \\ \vdots \\ n_k \cdot x \end{bmatrix} + \begin{bmatrix} d_0 \\ \vdots \\ d_k \end{bmatrix}$$

A high similarity to a template of a particular class indicates high membership in this class.

Answer to the Question No. 17

In a multi-class classifier, we use softmax activation function instead of sigmoid activation function because we want the output to be a probability which adds upto 1. However, if we use sigmoid activation function in the output layer of a multi-class classifier, the sum of probabilities of all the classes is greater than 1.

Answer to the Question No. 18

$$\text{Softmax}(\theta_j^T x) = \frac{e^{\theta_j^T x}}{\sum_{i=1}^{k} e^{(\theta_j^T x)}}$$

$$\frac{\partial}{\partial \theta_j}\left(\text{Softmax}(\theta_j^T x)\right) = \text{Softmax}(\theta_j^T x)\left(\delta_{ij} - \text{Softmax}(\theta_i^T x)\right) x$$

$$\frac{\partial}{\partial \theta_j} \log\left(\text{Softmax}(\theta_j^T x)\right) = \frac{1}{\text{Softmax}(\theta_j^T x)} \cdot \text{Softmax}(\theta_j^T x)\cdot\left(\delta_{ij} - \text{Softmax}(\theta_i^T x)\right) x$$

$$= \left(\delta_{ij} - \text{Softmax}(\theta_i^T x)\right) x$$

Answer to the Question No. 19

Log likelihood of a multiclass classifier is given by

$$l(\theta) = \log\left(\prod_{i=1}^{m}\prod_{j=1}^{k} P(y^{(i)} = j \mid x^{(i)})^{1(y^{(i)}=j)}\right)$$

$$= \sum_{i=1}^{m}\sum_{j=1}^{k} 1(y^{(i)}=j)\log\left(P(y^{(i)}=j \mid x^{(i)})\right)$$

$$= \sum_{i=1}^{m}\sum_{j=1}^{k} 1(y^{(i)}=j)\log\left(h_{\theta_j}(x^{(i)})\right)$$ ← Categorical Cross Entropy

Answer to the Question No. 20

$$\frac{\partial l(\theta)}{\partial \theta_j} = \sum_{i=1}^{m} \frac{1}{h_{\theta_j}(x^{(i)})} \cdot h_{\theta_j}(x^{(i)})\left(1 - h_{\theta_j}(x^{(i)})\right) \cdot x^{(i)}$$

$$= \sum_{i=1}^{m} 1(y^{(i)}=j)\left(1 - h_{\theta_j}(x^{(i)})\right) \cdot x^{(i)}$$

$$\frac{\partial(-l(\theta))}{\partial \theta_j} = \sum_{i=1}^{m}\left(h_{\theta_j}(x^{(i)}) - 1(y^{(i)}=j)\right) x^{(i)}$$

Gradient descent update :

$$\theta_j \leftarrow \theta_j - \eta \sum_{i=1}^{m}\left(h_{\theta_j}(x^{(i)}) - 1(y^{(i)}=j)\right) x^{(i)}$$

This equation accounts for only the errors by adding up the positives or negatives of the features of our example which were wrongly predicted.

Neural Network

Answer to the Question No. 1

If the no. of hidden units is greater than the no. of inputs, we say this is dimensionality increase. In general, if we are aiming to find more complex ~~dimens~~ dipendencies, we carry out diminensionality increase in order to find some relationships in higher dimensional space that may be easier to find in higher dimensional space.

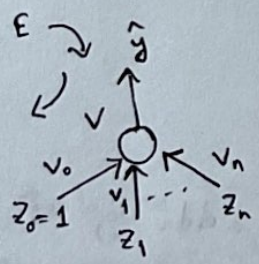
Answer to the Question No. 2

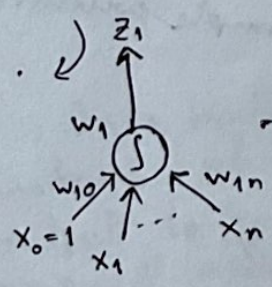
If the no. of hidden units is smaller than the no. of inputs, we call this dimensionality reduction. If our input contains too many features that correlate/depend on each other, we assume that the input is redundant and can be embedded in a subspace with lower dimensionality. By doing this, we sort of like condense the data and extract the important features.

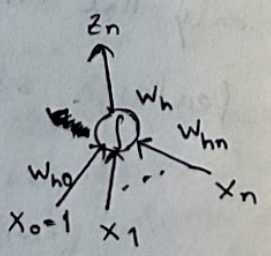
Answer to the Question No. 3

Because the output of the hidden ~~units~~ layer depends on the weights of the input to the network which needs to be considered as well.

Answer to the Question No. 4







Loss: L_2 loss

$$E = \frac{1}{2}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

$$\hat{y} = V^T Z = V_0 + V_1 z_1 + \dots V_n z_n$$

$$z_j = \text{Sigmoid}(w_j^T x)$$

Chain rule:

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

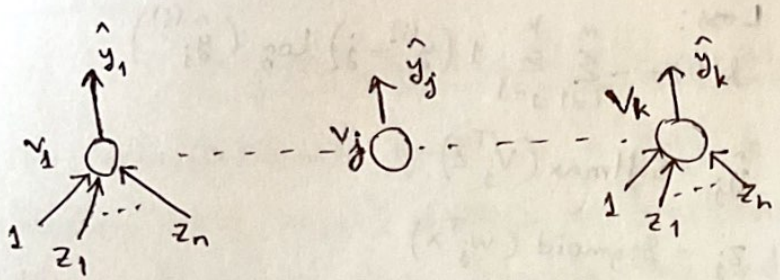
$$= \frac{1}{2} \times 2 \sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right) z^{(i)}$$

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

$$\frac{\partial E}{\partial w_j} = \frac{\partial E}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_j} \frac{\partial z_j}{\partial w_j}$$

$$= \sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right) V_j z_j^{(i)}\left(1 - z_j^{(i)}\right) x^{(i)}$$

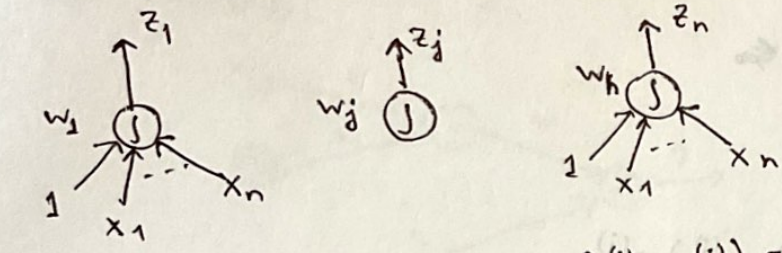
Answer to the Question No. 5



Loss:
$$E = \frac{1}{2} \sum_{i=1}^{m} \sum_{l=1}^{k} \left(\hat{y}_l^{(i)} - y_l^{(i)}\right)^2$$

$$\hat{y}_j = V_j^T Z = V_{j0} + V_{j1} + V_{j1} z_1 + \cdots + V_{jn} z_n$$

$$z_j = \text{Sigmoid}(w_j^T x)$$

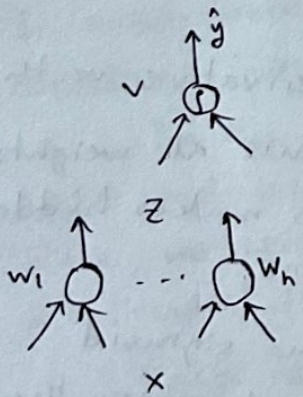


So,
$$\frac{\partial E}{\partial V_j} = \frac{\partial E}{\partial \hat{y}_j} \frac{\partial \hat{y}_j}{\partial V_j} = \frac{1}{2} \times 2 \sum_{i=1}^{m} \left(\hat{y}_j^{(i)} - y_j^{(i)}\right) z^{(i)}$$

$$\frac{\partial E}{\partial w_j} = \sum_{l=1}^{k} \frac{\partial E}{\partial \hat{y}_l} \frac{\partial \hat{y}_l}{\partial z_j} \frac{\partial z_j}{\partial w_j}$$

$$= \sum_{i=1}^{m} \sum_{l=1}^{k} \left(\hat{y}_l^{(i)} - y_l^{(i)}\right) V_{lj} z_j^{(i)} \left(1 - z_j^{(i)}\right) x^{(i)}$$

Answer to the Question No. 6



Loss:
$$l(\theta) = -\sum_{i=1}^{m} y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

$$\hat{y} = \text{Sigmoid}(V^T Z)$$

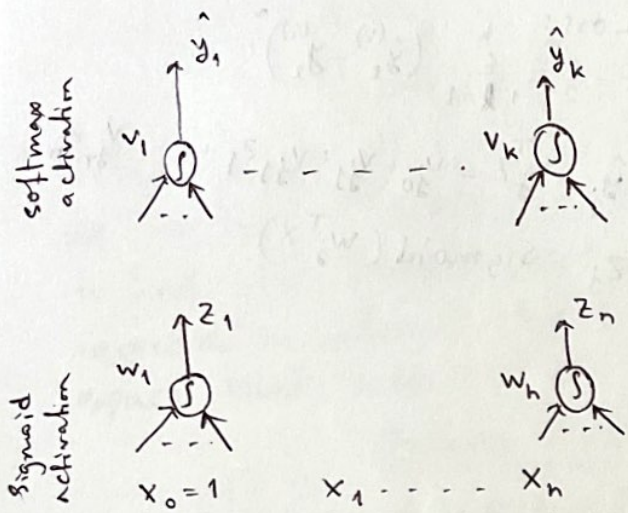
$$z_j = \text{Sigmoid}(w_j^T x)$$

So,
$$\frac{\partial l}{\partial V} = \frac{\partial l}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial V} = -\sum_{i=1}^{m} \left(y^{(i)} \frac{1}{\hat{y}^{(i)}} - (1 - y^{(i)}) \frac{1}{1 - \hat{y}^{(i)}}\right) \hat{y}^{(i)} \left(1 - \hat{y}^{(i)}\right) Z^{(i)}$$

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

$$\frac{\partial l}{\partial w_j} = \frac{\partial l}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_j} \frac{\partial z_j}{\partial w_j} = \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)}\right) V_j z_j^{(i)} \left(1 - z_j^{(i)}\right) x^{(i)}$$

Answer to the Question No. 7



Loss:

$$l(\theta) = -\sum_{i=1}^{m}\sum_{j=1}^{k} 1(y^{(i)} = j) \log\left(\hat{y}_j^{(i)}\right)$$

$$\hat{y}_j = \text{Softmax}(v_j^T z)$$

$$z_j = \text{Sigmoid}(w_j^T x)$$

So,

$$\frac{\partial E}{\partial v_j} = \frac{\partial E}{\partial \hat{y}_j}\frac{\partial \hat{y}_j}{\partial v_j} = \frac{1}{2}\times 2 \sum_{i=1}^{m}\left(\hat{y}_j^{(i)} - y_j^{(i)}\right) z^{(i)}$$

$$\frac{\partial E}{\partial w_j} = \sum_{l=1}^{k}\frac{\partial E}{\partial \hat{y}_l}\frac{\partial \hat{y}_l}{\partial z_j}\frac{\partial z_j}{\partial w_j}$$

$$= \sum_{i=1}^{m}\sum_{l=1}^{k}\left(\hat{y}_l^{(i)} - y_l^{(i)}\right) v_{lj}\, z_j^{(i)}\left(1 - z_j^{(i)}\right) x^{(i)}$$

Answer to the Question No. 8

In general, biases are initiated with 0 and weights are initiated with random numbers close to 0.

If all the weights are initialised with 0, the derivative with respect to loss function is the same for every w, thus all weights have the same value in subsequent iterations. This makes hidden units symmetric and continues for all the n iterations.

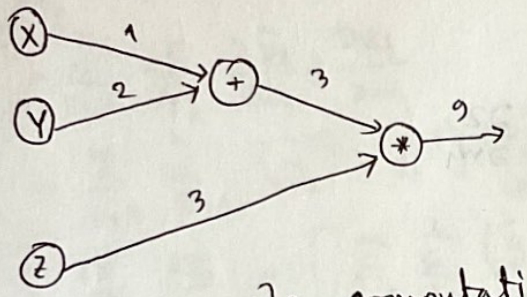
If weights are initialised with very high values, the sigmoid activation function, for instance, maps its value near to 1 where the slope of gradient changes slowly and learning takes a lot of time.

And if the weights are initiated with low values, it gets mapped to 0, where the case is the same as it is for initialising weights with very high values.

Computation Graphs

Answer to the Question No. 1

A computational graph is a way to represent a mathematical function. Nodes are input values or functions for combining them; as data flows through this graph, the edges receive their weights.



This is a simple computational graph with 5 nodes and 5 edges. But even simpler deep neural networks observe hundreds of thousands of nodes. In such a case, it would be practically impossible to calculate a function expression for it. Then, computational graphs come in handy. Such graphs also help us describe backpropagation more precisely.

In forward pass, we loop over nodes in a topological order. In order words, we pass the values of the variables in the forward direction. Given a node's inputs, we compute its values.

In the backward propagation, we start at a final goal node and loop over the nodes in a reverse topological order. Here, we compute the derivatives of the final goal node ~~wrt~~ value with respect to each edge's tail node.

Each node should be able to compute its output given the inputs, which will be required during forward pass, and also compute its derivative with respect to the tail of the node, which will be required during backward pass.

Answer to the Question No. 2

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

L2 loss, $L = \sum_{i=1}^{m} (y^{(i)} - \hat{y}^{(i)})^2$

$$= \sum_{i=1}^{m} (y^{(i)} - f_2(w_2, f_1(w_1, x)))^2$$

$$\frac{\partial L}{\partial \hat{y}} = \sum_{i=1}^{m} 2(y^{(i)} - \hat{y}^{(i)})(-1)$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial w_1}$$

$$= \sum_{i=1}^{m} 2(y^{(i)} - \hat{y}^{(i)})(-1) \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial w_1}$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w_2}$$

$$= \sum_{i=1}^{m} 2(y^{(i)} - \hat{y}^{(i)})(-1) \cdot \frac{\partial \hat{y}}{\partial w_2}$$

Using binary cross entropy loss,

$$l(\theta) = -\sum_{i=1}^{m} y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

$$\frac{\partial l(\theta)}{\partial \hat{y}} = -\sum_{i=1}^{m} \left(y^{(i)} \cdot \frac{1}{\hat{y}^{(i)}} - (1 - y^{(i)} \cdot \frac{1}{1 - \hat{y}^{(i)}} \right)$$

So, $\frac{\partial l(\theta)}{\partial w_1} = \frac{\partial l(\theta)}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial w_1}$

$$= \left[-\sum_{i=1}^{m} \left(y^{(i)} \frac{1}{\hat{y}^{(i)}} - (1 - y^{(i)}) \cdot \frac{1}{1 - \hat{y}^{(i)}} \right) \right] \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial w_1}$$

$$\frac{\partial l(\theta)}{\partial w_2} = \frac{\partial l(\theta)}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_2}$$

$$= \left[-\sum_{i=1}^{m} \left(y^{(i)} \cdot \frac{1}{\hat{y}^{(i)}} - (1 - y^{(i)}) \cdot \frac{1}{1 - \hat{y}^{(i)}} \right) \right] \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial w_2}$$

Answer to the Question No. 3

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

L_2 loss, $E = \frac{1}{2} \sum_{i=1}^{m} \sum_{l=1}^{k} (\hat{y}_l^{(i)} - y_l^{(i)})^2$

$$\frac{\partial E}{\partial \hat{y}_j} = \sum_{i=1}^{m} \sum_{l=1}^{k} (\hat{y}_l^{(i)} - y_l^{(i)})$$

So, $$\frac{\partial E}{\partial w_1} = \sum_{l=1}^{k} \frac{\partial E}{\partial \hat{y}_l} \frac{\partial \hat{y}_l}{\partial z_j} \frac{\partial z_j}{\partial w_1}$$
$$= \sum_{i=1}^{m} \sum_{l=1}^{k} (\hat{y}_l^{(i)} - y_l^{(i)}) \frac{\partial \hat{y}_l}{\partial z_j} \frac{\partial z_j}{\partial w_1}$$

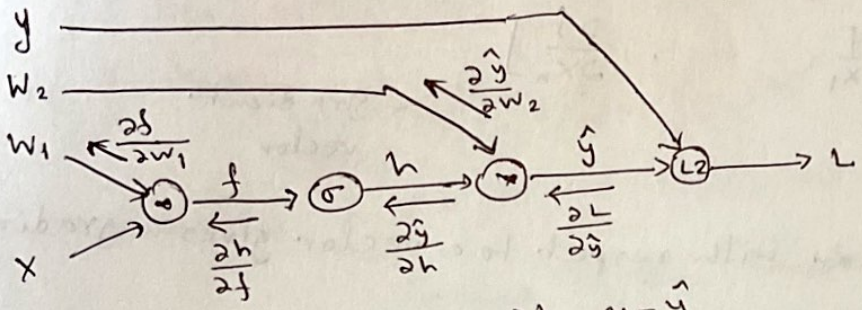
$$\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial \hat{y}_j} \frac{\partial \hat{y}_j}{\partial w_2} = \left[\sum_{i=1}^{m} \sum_{l=1}^{k} (\hat{y}_l^{(i)} - y_l^{(i)}) \right] \frac{\partial \hat{y}_j}{\partial w_2}$$

Categorical Cross Entropy, $l(\theta) = -\sum_{i=1}^{m} \sum_{j=1}^{k} 1(y^{(i)} - j) \log(\hat{y}_j^{(i)})$

$$\frac{\partial l(\theta)}{\partial w_1} = \sum_{l=1}^{k} \frac{\partial l}{\partial \hat{y}_l} \frac{\partial \hat{y}_l}{\partial z_j} \frac{\partial z_j}{\partial w_1}$$
$$= \sum_{i=1}^{m} \sum_{l=1}^{k} (\hat{y}_l^{(i)} - y_l^{(i)}) \frac{\partial \hat{y}_l}{\partial z_j} \frac{\partial z_j}{\partial w_1}$$

$$\frac{\partial l(\theta)}{\partial w_2} = \frac{\partial E}{\partial \hat{y}_j} \frac{\partial \hat{y}_j}{\partial w_2} = \left[\sum_{i=1}^{m} \sum_{l=1}^{k} (\hat{y}_l^{(i)} - y_l^{(i)}) \right] \frac{\partial \hat{y}_j}{\partial w_2}$$

Answer to Question No. 4



$f = W_1^T X$

$h = \text{Sigmoid}(f)$

$\hat{y} = W_2^T h$

$L = \frac{1}{2}(y - \hat{y})^2$

$\frac{\partial L}{\partial \hat{y}} = y - \hat{y}$

$\frac{\partial \hat{y}}{\partial h} = W_2$

$\frac{\partial h}{\partial f} = h(1-h)$

$\frac{\partial f}{\partial W_1} = X$

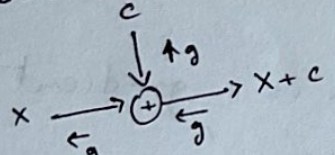
$\frac{\partial \hat{y}}{\partial W_2} = h$

Answer to the Question No. 5

Backflow patterns:

$\frac{\partial (x+c)}{\partial x} = 1$

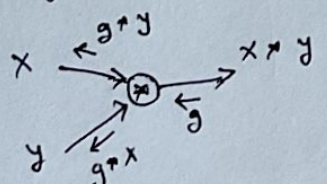
If the incoming gradient is g, it is multiplied by 1 in case of addition node.



$\frac{\partial (x*y)}{\partial x} = y$

$\frac{\partial (x*y)}{\partial y} = x$

If the incoming gradient is g, it is multiplied by the opposite input at the node, in case of multiplication node.

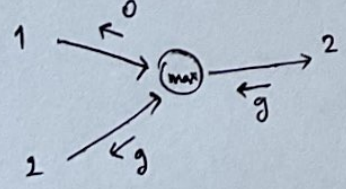


$\max(X, y) = \begin{cases} x & \text{if } x \geq y \\ y & \text{otherwise} \end{cases}$

$\frac{\partial \max(x, y)}{\partial x} = \begin{cases} 1 & \text{if } x \geq y \\ 0 & \text{otherwise} \end{cases}$

$\frac{\partial \max(x, y)}{\partial y} = \begin{cases} 0 & \text{if } x > y \\ 1 & \text{otherwise} \end{cases}$

It basically chooses the maximum of the two inputs. Note, if the gradients at some point is 0, then we can ignore all the sub nodes because all will be multiplied by 0.



If we replace "max" by "min" in the node above,

$\min(X, y) = \begin{cases} x & \text{if } x \leq y \\ y & \text{otherwise} \end{cases}$

$\frac{\partial \min(X, y)}{\partial x} = \begin{cases} 1 & \text{if } x \leq y \\ 0 & \text{otherwise} \end{cases}$

$\frac{\partial \max(X, y)}{\partial y} = \begin{cases} 0 & \text{if } x \leq y \\ 1 & \text{otherwise} \end{cases}$

Answer to the Question No. 6

Scalar valued function

$$f(X) = f(X_1, \ldots, X_n)$$

$$\frac{\partial f}{\partial X} = \nabla_x f = \left[\frac{\partial f}{\partial X_1}, \ldots, \frac{\partial f}{\partial X_n}\right]$$

$\frac{\partial f}{\partial X}$ ← derivative of a scalar wrt a vector

$\left[\frac{\partial f}{\partial X_1}, \ldots, \frac{\partial f}{\partial X_n}\right]$ ← gradient vector

So, derivative of a scalar with respect to a vector gives a gradient vector.

Answer to the Question No. 7

~~Gradient of a~~ Derivative of a vector with respect to a vector is a rank 2 tensor.

$$\begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_m \end{bmatrix}_{m\times 1} \xrightarrow{\text{derivative}} \begin{bmatrix} \frac{\partial \hat{y}_1}{\partial w_1} & \cdots & \frac{\partial \hat{y}_1}{\partial w_n} \\ \vdots & & \\ \frac{\partial \hat{y}_m}{\partial w_1} & \cdots & \frac{\partial \hat{y}_m}{\partial w_n} \end{bmatrix}_{m\times n}$$

Answer to the Question No. 8

Derivative of a scalar with respect to a matrix is a gradient matrix.

$$\frac{\partial f}{\partial w} = \begin{bmatrix} \frac{\partial f}{\partial w_{11}} & \cdots & \frac{\partial f}{\partial w_{1m}} \\ \vdots & & \\ \frac{\partial f}{\partial w_{m1}} & \cdots & \frac{\partial f}{\partial w_{mn}} \end{bmatrix}$$

Answer to the Question No. 9

If $H(x) = G(F(x))$ and if x_i is one of the input variables for F, then

$$\frac{\partial H}{\partial x_i} = \nabla G(F(x)) \cdot \frac{\partial F}{\partial x_i}(x)$$

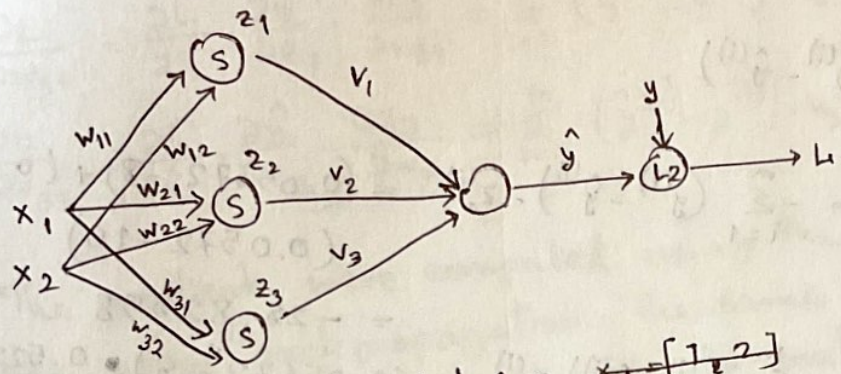
provided G and F are differentiable.

Answer to the Question No. 11

Otherwise the matrix dimensions won't match and also the calculations will show inaccurate results as the inputs may get multiplied by weights of another layer or so.

Answer to the Question No. 12

(a)



(b) For the first data point, ~~i.e.~~ ~~$x_1 = [1 \; 2]$~~

$$\hat{y} = 0.01 + v_1 z_1^{(1)} + v_2 z_2^{(2)} + v_3 z_3^{(3)} = 0.01 + 0.02 \cdot z_1^{(1)} + 0.03 \cdot z_2^{(1)} + 0.04 \cdot z_3^{(1)}$$

$$= 0.05672$$

where

$z_1^{(1)} = \sigma(w_1 X) = 0.5225$

$z_2^{(1)} = \sigma(w_2 X) = 0.52$

$z_3^{(1)} = \sigma(w_3 X) = 0.5175$

For the second data point,

$$\hat{y} = 0.01 + v_1 z_1^{(2)} + v_2 z_2^{(2)} + v_3 z_3^{(3)} = \text{~~0.01 + 0.02 · 2~~} + 0.01 + 0.02 \cdot z_1^{(2)} + 0.03 \cdot z_2^{(2)} + 0.04 \cdot z_3^{(3)}$$

$$= 0.0571$$

where

$z_1^{(2)} = \sigma(w_1 X) = 0.53$

$z_2^{(2)} = \sigma(w_2 X) = 0.525$

$z_3^{(2)} = \sigma(w_3 X) = 0.52$

For the third data point,

$$\hat{y} = 0.01 + v_1 z_1^{(3)} + v_2 z_2^{(3)} + v_3 z_3^{(3)} = 0.01 + 0.02 \cdot z_1^{(3)} + 0.03 \cdot z_2^{(3)} + 0.04 \cdot z_3^{(3)}$$

$$= 0.0572$$

where

$z_1^{(3)} = \sigma(w_1 X) = \text{~~0.0~~}\ 0.5275$

$z_2^{(3)} = \sigma(w_2 X) = 0.5225$

$z_3^{(3)} = \sigma(w_3 X) = 0.5250$

(c)

~~$L = (\hat{y} - y)$~~

$$L = \frac{1}{2} \sum_{i=1}^{m} \left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

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

So, $\frac{\partial L}{\partial v_0} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial v_0} = -\sum_{i=1}^{m} \left(y^{(i)} - \hat{y}^{(i)}\right) \cdot z_0^{(i)} = (0.05672 - 8) + (0.0571 - 11) + (0.0572 - 10) = -28.82898$

$\frac{\partial L}{\partial v_1} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial v_1} = -\sum_{i=1}^{m} \left(y^{(i)} - \hat{y}^{(i)}\right) \cdot z_1^{(i)} = (0.05672 - 8) \cdot 0.5225 + (0.0571 - 11) \cdot 0.53 + (0.0572 - 10) \cdot 0.5275 = -15.195$

$\frac{\partial L}{\partial v_2} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial v_2} = -\sum_{i=1}^{m} \left(y^{(i)} - \hat{y}^{(i)}\right) \cdot z_2^{(i)} = -15.066$

$\frac{\partial L}{\partial v_3} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial v_3} = -\sum_{i=1}^{m} \left(y^{(i)} - \hat{y}^{(i)}\right) \cdot z_3^{(i)} = -15.017$

again,

$\frac{\partial L}{\partial w_{10}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_{10}} = \sum_{i=1}^{m} \left(y^{(i)} - \hat{y}^{(i)}\right) \cdot v_1 \cdot z_1^{(i)} \left(1 - z_1^{(i)}\right) \cdot 1 = -0.1437$

$\frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_{11}} = \sum_{i=1}^{m} \left(y^{(i)} - \hat{y}^{(i)}\right) \cdot v_1 \cdot z_1^{(i)} \left(1 - z_1^{(i)}\right) \cdot x_1^{(i)} = -0.1933$

$\frac{\partial L}{\partial w_{12}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_{12}} = \sum_{i=1}^{m} \left(y^{(i)} - \hat{y}^{(i)}\right) \cdot v_1 \cdot z_1^{(i)} \left(1 - z_1^{(i)}\right) \cdot x_2^{(i)} = -0.34219$

again,

$\frac{\partial L}{\partial w_{20}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial w_{20}} = \sum_{i=1}^{m} \left(y^{(i)} - \hat{y}^{(i)}\right) \cdot v_2 \cdot z_2^{(i)} \cdot \left(1 - z_1^{(i)}\right) \cdot 1 = -0.2157$

$\frac{\partial L}{\partial w_{21}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial w_{21}} = \sum_{i=1}^{m} \left(y^{(i)} - \hat{y}^{(i)}\right) \cdot v_2 \cdot z_2^{(i)} \cdot \left(1 - z_2^{(i)}\right) \cdot x_1^{(i)} = -0.2897$

$\frac{\partial L}{\partial w_{22}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial w_{22}} = \sum_{i=1}^{m} \left(y^{(i)} - \hat{y}^{(i)}\right) \cdot v_2 \cdot z_2^{(i)} \cdot \left(1 - z_2^{(i)}\right) \cdot x_2^{(i)} = -0.5133$

again,

$$\frac{\partial L}{\partial w_{30}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_3} \cdot \frac{\partial z_3}{\partial w_{30}} = \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)}\right) \cdot v_3 \cdot z_3^{(i)} \left(1 - z_3^{(i)}\right) \cdot 1 = -0.2877$$

$$\frac{\partial L}{\partial w_{31}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_3} \cdot \frac{\partial z_3}{\partial w_{31}} = \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)}\right) \cdot v_3 \cdot z_3^{(i)} \left(1 - z_3^{(i)}\right) \cdot x_1^{(i)} = -0.3866$$

$$\frac{\partial L}{\partial w_{32}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_3} \cdot \frac{\partial z_3}{\partial w_{32}} = \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)}\right) \cdot v_3 \cdot z_3^{(i)} \left(1 - z_3^{(i)}\right) \cdot x_2^{(i)} = -0.6847$$

(d) The gradients were computed using the chain rule in the class when using backpropagation. The same thing is used in part (c). Hence, the results should match had we used any other method.

Answer to the Question No. 13

(a) $f(x, y) = (2x + 3y)^2$

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

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

$$DF(x, y) = \begin{bmatrix} 2x & 2 \\ 3 & 8y \end{bmatrix}$$

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

(c) Without chain rule

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

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

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

With chain rule

$$DF(G(x)) = \begin{bmatrix} 2x & 2 \\ 3 & 8x^2 \end{bmatrix}$$

$$DG(x) = \begin{bmatrix} 1 \\ 2x \end{bmatrix}$$

$$D(F \circ G)(x) = DF(G(x)) \; DG(x)$$

$$= \begin{bmatrix} 2x & 2 \\ 3 & 8x^2 \end{bmatrix} \begin{bmatrix} 1 \\ 2x \end{bmatrix}$$

$$= \begin{bmatrix} 6x \\ 3 + 16x^3 \end{bmatrix}$$

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

(d)

$$\frac{\partial L}{\partial v^{(i)}} = D(L \circ Y)(v)$$

$$= DL(Y(v)) \cdot DY(v)$$

$$= -\left[y^{(i)} - \hat{y}^{(i)}\right] \cdot \begin{bmatrix} z_0^{(i)} & z_1^{(i)} & z_2^{(i)} & z_3^{(i)} \end{bmatrix}$$

So,

$$\frac{\partial L}{\partial v} = \begin{bmatrix} -28.82898 & -15.195 & -15.066 & -15.017 \end{bmatrix}$$

Again,

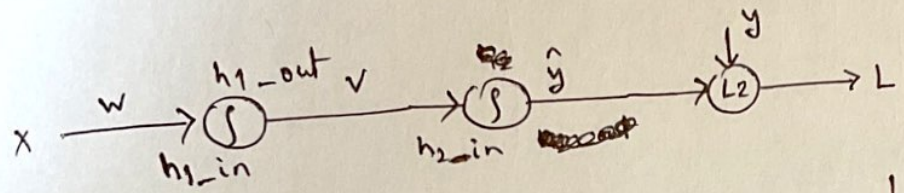
$$\frac{\partial L}{\partial w^{(i)}} = D(L \circ Y)(w) = DL(Y(w)) \cdot DY(w)$$

$$= -\left[y^{(i)} - \hat{y}^{(i)}\right] \cdot \begin{bmatrix} v_1 \cdot z_1^{(i)}(1 - z_1^{(i)}) \cdot x_0^{(i)} & v_1 \cdot z_1^{(i)}(1 - z_1^{(i)}) \cdot x_1^{(i)} & v_1 \cdot z_1^{(i)}(1 - z_1^{(i)}) \cdot x_2^{(i)} \\ v_2 \cdot z_2^{(i)}(1 - z_2^{(i)}) \cdot x_0^{(i)} & v_2 \cdot z_2^{(i)}(1 - z_2^{(i)}) \cdot x_1^{(i)} & v_2 \cdot z_2^{(i)}(1 - z_2^{(i)}) \cdot x_2^{(i)} \\ v_3 \cdot z_3^{(i)}(1 - z_3^{(i)}) \cdot x_0^{(i)} & v_3 \cdot z_3^{(i)}(1 - z_3^{(i)}) \cdot x_1^{(i)} & v_3 \cdot z_3^{(i)}(1 - z_3^{(i)}) \cdot x_2^{(i)} \end{bmatrix}$$

So,

$$\frac{\partial L}{\partial w} = \begin{bmatrix} -0.1437 & -0.1933 & -0.3419 \\ -0.2157 & -0.2897 & -0.5133 \\ -0.2874 & -0.3866 & -0.6847 \end{bmatrix}$$

Answer to the Question No. 14



$$L = \frac{1}{2}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

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

$$\frac{\partial L}{\partial v} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h_2\_in} \cdot \frac{\partial h_2\_in}{\partial v}$$

$$= -\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right) \cdot \hat{y}^{(i)}\left(1 - \hat{y}^{(i)}\right) \cdot h_1\_out^{(i)}$$

Again,

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h_2\_in} \cdot \frac{\partial h_2\_in}{\partial h_1\_out} \cdot \frac{\partial h_1\_out}{\partial h_1\_in} \cdot \frac{\partial h_1\_in}{\partial w}$$

$$= -\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right) \cdot \hat{y}^{(i)}\left(1 - \hat{y}^{(i)}\right) \cdot v \cdot h_1\_out^{(i)}\left(1 - h_1\_out^{(i)}\right) \cdot x^{(i)}$$

Part 2 (Implementation Questions)

Abstract
In implementing a fully connected neural network from scratch with 2 hidden layers, 2 datasets were used as per instruction.

The Iris Data Set [1] is perhaps the best known database found in the pattern recognition literature. The dataset is balanced and contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other two; the latter two, however, are not linearly separable from each other.

The other dataset used is the Wine Data Set [2]. These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents (i.e. features) found in each of the three types of wines. There are 178 examples in this dataset.

Here, a three-layer neural network (i.e. 2 hidden layer) has been implemented to classify correctly the type of flower and wine using forward propagation, backward propagation, categorical cross entropy loss and stochastic gradient descent with a learning rate.

Methodology

The neural network has been implemented using only the numpy and scipy libraries. The activation functions (sigmoid and softmax) have been implemented from the scipy library and the input output data, the weights, and the gradients are implemented using numpy arrays. The implemented neural network is discussed below:

1. Defining the model
   The datasets were loaded and splitted for training, validating and testing purposes by isolating a fraction (which is, 0.2) from the total examples. All the functions were defined along the model without using any built-in model functions and derivatives were also computed where updates of weights were required. The activation function selected for the hidden layer is sigmoid, and that for the output layer is softmax (in order to have probabilities as an output vector). Categorical cross entropy is used to compute the loss function, and stochastic gradient descent is used as the optimizer to minimize the loss function and update the weights accordingly. The total number of classes is 3 for both the datasets.

2. Preparing the data

The initial weights were assigned randomly from the standard normal distribution between -1 and 1, but the performance of the model improved if the range of normal distribution is kept between -0.5 and 0.5. Hence, the latter was kept as the weights are now closer to zero than they were before which converges better to the solution. Both the iris and wine datasets were checked if they included any missing values, which they did not. Then the datasets were shuffled to remove any biases due to ordering of the examples.

3. Testing the data

In running the model for the wine dataset, the performance of the neural network was around 40% and it was not showing any significant improvement by changing the hyperparameters. Then, a careful look into the wine dataset showed that the last feature was in the order of thousands whereas all other features were in the order of ones or tens. Hence, “min max normalization” was performed to help the model converge. Although no significant improvement was noticed in the performance of the iris dataset due to the implementation of “min max normalization”. This makes sense because the iris dataset has features in the similar scale. Different values of learning rate and epochs have been used, yielding different accuracy results. The results have been summarized under the following section. The model accuracy was computed on the test dataset through addition of 1 to the score value for correct guesses and ‘0’ otherwise. The average of the score values is the accuracy of the neural network.

For the iris dataset, the number of hidden nodes is selected to be 4 at first to match the number of input features. It was then changed to higher and lower values to see any improvement in performance, which it did not. Then the number of hidden units was also set to 4 at first and checked to see any changes in performance by varying it. The performance of the model kept increasing until the number of hidden units for the second hidden layer was incremented to 8. For the wine dataset, the number of hidden nodes is selected to be 13 at first to match the number of input features. It was then changed to higher and lower values to see any improvement in performance. Hence, setting it to 8 instead of 13 resulted in the best performance. Then the number of hidden units was also set to 13 at first and checked to see any changes in performance by varying it. The optimum performance for the second hidden layer was found if the number of hidden units for the second layer is set to 8.

Results

A summary of the hyperparameter tuning for this neural network is shown below.

Results of Tuning Hyperparameters for Iris Dataset		
Epochs	**Learning Rate**	**Accuracy**
50	0.1	0.60000000
50	0.01	0.40000000
90	0.1	0.93333333
90	0.01	0.80000000
90	0.001	0.26666667

The best accuracy result of ~93% is obtained when a learning rate of 0.1 is used with 90 epochs. Hidden layer 1 had 4 units and hidden layer 2 had 8 units in this case.

Results of Tuning Hyperparameters for Wine Dataset		
Epochs	**Learning Rate**	**Accuracy**
50	0.1	0.94444444
50	0.01	0.27777778
30	0.1	0.94444444
30	0.01	0.611111111
30	0.001	0.38888889

The best accuracy result of ~94% is obtained when a learning rate of 0.1 is used with 30 epochs. Hidden layer 1 had 8 units and hidden layer 2 had 8 units in this case.

Reference:

1. https://archive.ics.uci.edu/ml/datasets/Iris
2. https://archive.ics.uci.edu/ml/datasets/wine