Feed Forward Neural Networks

Input: n dimensional vector (0^{th} layer)

Hidden layers: L-1 hidden layers with 2 neurons each (2 in our case)

Output layers: 1 output layer containing k neurons (L^{th} layer) (for k classes)

Each layer in the hidden & output layer will be split into two parts: pre-activation(a_i) & activation(h_i)

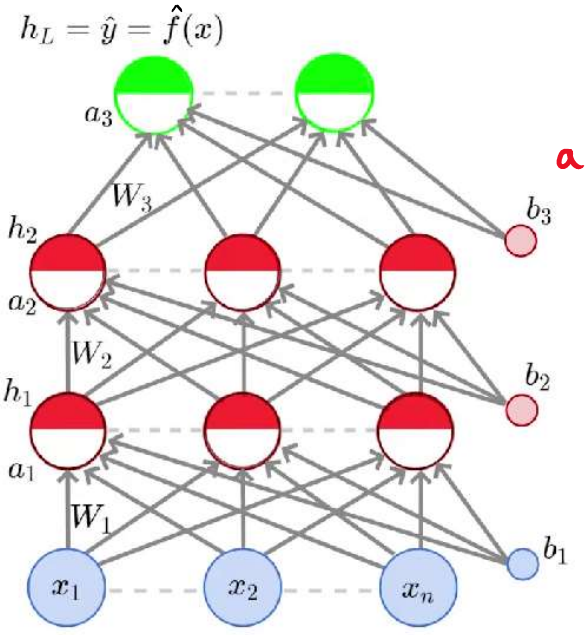
Weights: $W_i \in \mathbb{R}^{n \times n}$ & $W_L \in \mathbb{R}^{k \times n}$ b/w last hidden layer & output (L = 3 in this case)

Bias: $b_i \in \mathbb{R}^n$ & $b_L' \in \mathbb{R}^k$

b/w layers i-1 & i

$(0 < i < L)$

$$a_i(x) = b_i + W_i h_{i-1}(x)$$



$$a_1 = \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \end{bmatrix} = \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix} + \begin{bmatrix} w_{111} & w_{112} & w_{113} \\ \vdots & & \\ w_{131} & w_{132} & w_{133} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$h_i(x) = g(a_i(x))$$

$$h_1 = g\left(\begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \end{bmatrix}\right) = \begin{bmatrix} g(a_{11}) \\ g(a_{12}) \\ g(a_{13}) \end{bmatrix}$$

Taking g as a sigmoid function.

$$= \begin{bmatrix} \frac{1}{1+e^{-a_{11}}} \\ \frac{1}{\cdots} \end{bmatrix}$$

g is called activation function

eg - logistic, tanh, linear etc.

function
eg - logistic, tanh, linear etc. = $\begin{bmatrix} \frac{1}{1+e^{-a_{12}}} \\ \frac{1}{1+e^{-a_{13}}} \end{bmatrix}$

The activation in the output layer is given by,

$$f(x) = h_L(x) = O(a_L(x))$$

O is the output activation function
eg - softmax, linear etc.

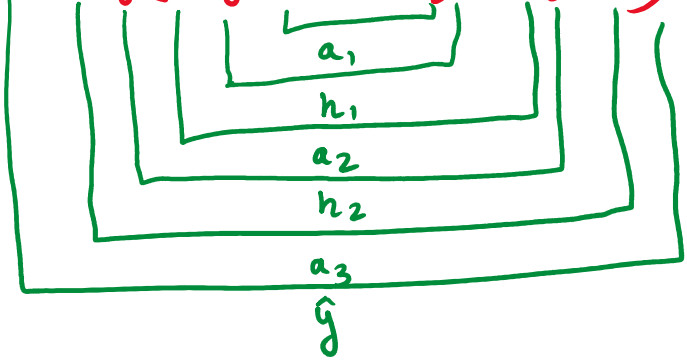
For simplicity, we are going to write $a_i(x)$ as a_i & $h_i(x)$ as h_i

$$\therefore \quad \boxed{\begin{aligned} a_i &= b_i + W_i h_{i-1} \\ h_i &= g(a_i) \\ f(x) &= h_L = O(a_L) \end{aligned}}$$

In a typical machine learning setup,

Data: $\{x_i, y_i\}_{i=1}^{N}$

Model: $\hat{y}_i = f(x_i) = O(W_3 g(W_2 g(W_1 x_i + b_1) + b_2) + b_3)$ (For this case of 3 inputs)



Parameters: $\theta = W_1 \cdots W_L, b_1 \cdots b_L$ ($L = 3$ in this case)

Algorithm: Gradient Descent or Backpropagation

Algorithm: Gradient Descent or Backpropagation

Loss Function: say $\min \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{k} (\hat{y}_{ij} - y_{ij})^2$

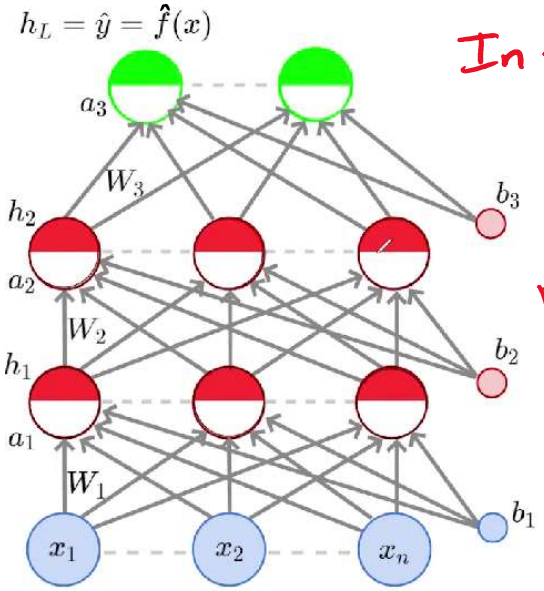
in general $\min L(\theta)$ some loss function

Learning Parameters (intuition)

Writing the gradient descent algorithm more concisely:

```
t ← 0;
max_iterations ← 1000;
Initialize θ_t = [w_0 b_0];
while t++ < max_iterations do
    θ_{t+1} ← θ_t − η∇θ_t
end
```

where $\nabla\theta_t = \left[\frac{\partial L(\theta)}{\partial w_t} \quad \frac{\partial L(\theta)}{\partial b_t}\right]^T$



In this feed forward neural network we have,

$$\theta = [w_1, \cdots, w_L, b_1, \cdots, b_L]$$

Now,

$$\nabla\theta_t = \left[\frac{\partial L(\theta_t)}{\partial w_{1,t}}, \cdots \frac{\partial L(\theta_t)}{\partial w_{L,t}}, \frac{\partial L(\theta_t)}{\partial b_{1,t}}, \cdots \frac{\partial L(\theta_t)}{\partial b_{L,t}}\right]^T$$

$\nabla\theta$ is now composed of

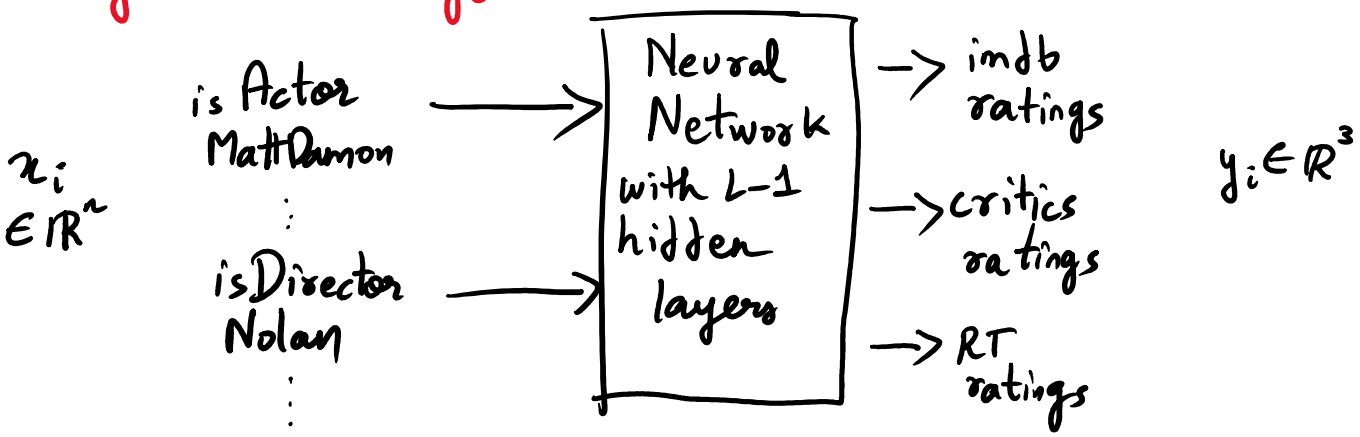
$\nabla w_1, \nabla w_2, \cdots \nabla w_{L-1} \in \mathbb{R}^{n \times n}$, $\nabla w_L \in \mathbb{R}^{k \times n}$,

$\nabla b_1, \nabla b_2, \cdots \nabla b_{L-1} \in \mathbb{R}^n$, $\nabla b_L \in \mathbb{R}^k$

Output Function: & Loss Functions

The choice of loss function depends upon the problem

Consider the movie example but this time we are predicting movie ratings.

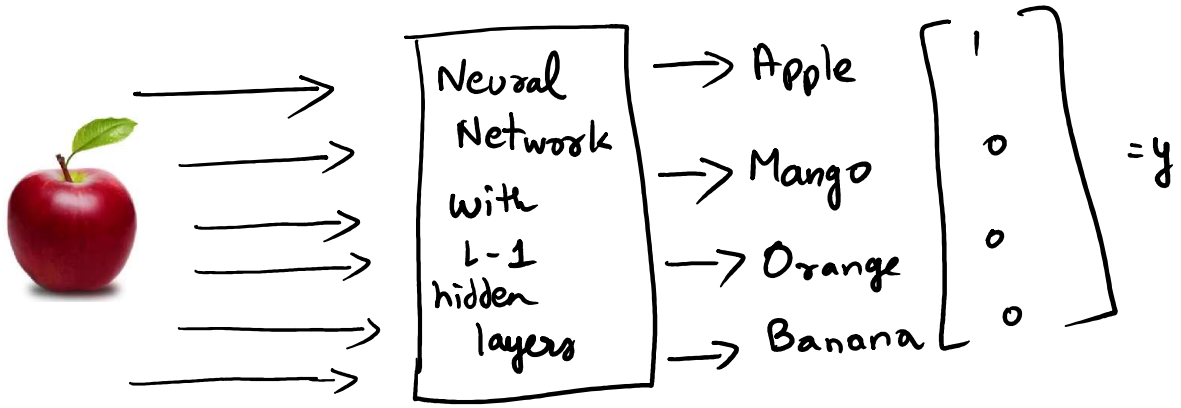


The loss function should capture how much $\hat{y}_i$ deviates from y_i

If $y \in \mathbb{R}^2$ then the squared loss can capture this deviation

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{3} (\hat{y}_{ij} - y_{ij})^2$$

Let's take another example:



We want to classify the image into 1 of k classes

We want to classify the image into 1 of k classes
We could use squared error loss to capture deviation
Notice that y is a probability distribution
$\therefore \hat{y}$ should be one as well

For this, let O be a softmax function

$$\hat{y}_j = O(a_L)_j = \frac{e^{a_{L,j}}}{\sum_{i=1}^{k} e^{a_{L,i}}}$$

$O(a_L)_j$ is the j^{th} element of $\hat{y}$ & $a_{L,j}$ is the j^{th} element of vector a_L

Cross-entropy : $L(\theta) = \sum_{i=1}^{k} y_c \log \hat{y}_c$

$y_c = 1$ if $c = l$ (true class label)
$= 0$ otherwise

$L(\theta) = -\log \hat{y}_l$

For classification, we use the following objective function

$$\underset{\theta}{\text{minimize}} \quad L(\theta) = -\log \hat{y}_l$$

or

$$\underset{\theta}{\text{maximize}} \quad -L(\theta) = \log \hat{y}_l$$

$\hat{y}_l$ is a function of all parameters $\theta = [w_1, \cdots w_L, b_1, \cdots b_L]$

$$\hat{y}_l = \left[O(w_3 g(w_2 g(w_1 x + b_1) + b_2)\, b_3)\right]_l$$

It is the probability that x belongs to l^{th} class
We want it to be as close to 1

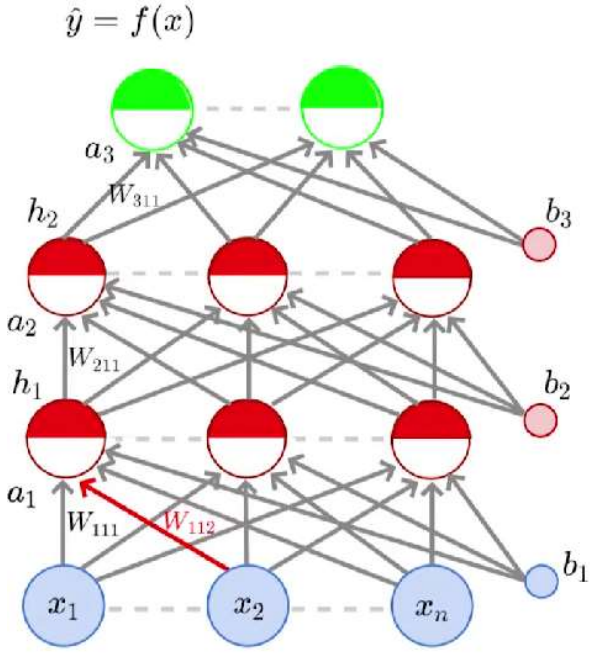
It is the probability that x belongs to x class

We want it to be as close to 1

	Outputs	
	Real Values	Probabilities
Output Function	Linear	Softmax
Loss Function	Squared Error	Cross Entropy

Backpropagation (Intuition)

In the feed foward neural network, lets focus on one weight W_{112}.

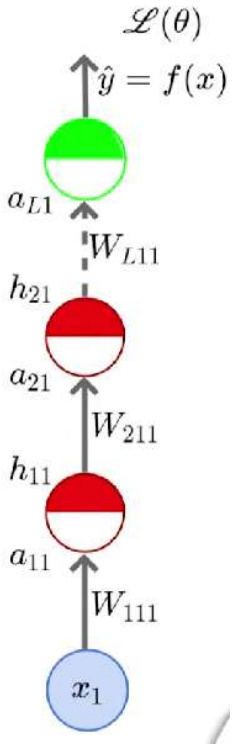


To learn this weight using SDG, we need a formula for $\frac{\partial L(\theta)}{\partial W_{112}}$

We want to calculate all partial derivatives of all weights in that layer in one go & other layers as well.

Let us consider the example of a deep but thin network

It is easy to find derivative by chain rule:



$$\frac{\partial L(\theta)}{\partial W_{111}} = \frac{\partial L(\theta)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{L11}} \frac{\partial a_{L11}}{\partial h_{21}} \frac{\partial h_{21}}{\partial a_{21}} \frac{\partial a_{21}}{\partial h_{11}} \frac{\partial h_{11}}{\partial a_{11}} \frac{\partial a_{11}}{\partial W_{111}}$$

$$\frac{\partial L(\theta)}{\partial W_{111}} = \frac{\partial L(\theta)}{\partial h_{11}} \frac{\partial h_{11}}{\partial W_{111}}$$ (just compressing chain rule)

$$\frac{\partial L(\theta)}{\partial W_{211}} = \frac{\partial L(\theta)}{\partial h_{21}} \frac{\partial h_{21}}{\partial W_{211}}$$

$$\frac{\partial L(\theta)}{\partial} \quad \frac{\partial L(\theta)}{\partial h_{11}}$$

$$\frac{\partial L(\theta)}{\partial w_{L11}} = \frac{\partial L(\theta)}{\partial h_{L1}} \frac{\partial h_{L1}}{\partial w_{L11}}$$

Quantities of interest (roadmap for the remaining part):

Gradient w.r.t. output units

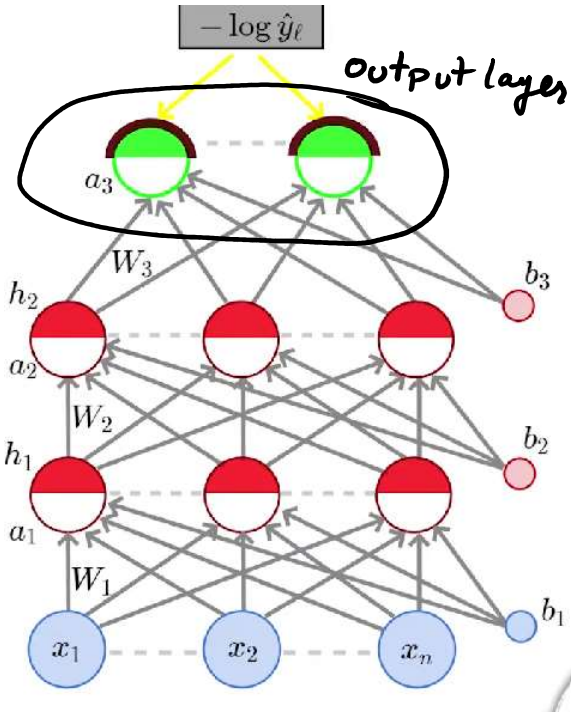
Gradient w.r.t. hidden units

Gradient w.r.t. weights and biases

$$\underbrace{\frac{\partial \mathcal{L}(\theta)}{\partial W_{111}}}_{\text{Talk to the weight directly}} = \underbrace{\frac{\partial \mathcal{L}(\theta)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_3}}_{\text{Talk to the output layer}} \underbrace{\frac{\partial a_3}{\partial h_2} \frac{\partial h_2}{\partial a_2}}_{\text{Talk to the previous hidden layer}} \underbrace{\frac{\partial a_2}{\partial h_1} \frac{\partial h_1}{\partial a_1}}_{\text{Talk to the previous hidden layer}} \underbrace{\frac{\partial a_1}{\partial W_{111}}}_{\text{and now talk to the weights}}$$

Gradient wrt Output Units

Let us first consider the partial derivative wrt i^{th} output : $\nabla_{a_i} L(\theta)$



$$L(\theta) = -\log \hat{y}_\ell \quad (\ell = \text{true class label})$$

$$\frac{\partial}{\partial \hat{y}_i}(L(\theta)) = \frac{\partial}{\partial \hat{y}_i}(-\log \hat{y}_\ell)$$

$$= -\frac{1}{\hat{y}_\ell} \quad \text{if } i = \ell$$

$$= 0 \quad \text{otherwise}$$

$$\boxed{\frac{\partial}{\partial \hat{y}_i} L(\theta) = -\frac{\mathbb{1}_{i=\ell}}{\hat{y}_\ell}}$$

We can now talk about the gradient wrt vector $\hat{y}$

$$\nabla_{\hat{y}} L(\theta) = \begin{bmatrix} \frac{\partial L(\theta)}{\partial \hat{y}_1} \\ \vdots \\ \frac{\partial L(\theta)}{\partial \hat{y}_K} \end{bmatrix} = -\frac{1}{\hat{y}_\ell} \begin{bmatrix} \mathbb{1}_{\ell=1} \\ \mathbb{1}_{\ell=2} \\ \vdots \\ \mathbb{1}_{\ell=K} \end{bmatrix}$$

this vector will have 1 in only place for that value of ℓ & 0 everywhere else

eg: if $\ell = 3$ & $k = 5$ then:

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

These kinds of vectors are called one-hot vectors denoted by e_ℓ

denoted by e_l

$$\nabla_{y_l} L(\theta) = -\frac{1}{\hat{y}_l} e_l$$

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

e_l is a k^{th} dimension vector, l^{th} element is 1 & every other element is 0

Now,

$$\frac{\partial L(\theta)}{\partial a_{L_i}} = \frac{\partial (-\log \hat{y}_l)}{\partial a_{L_i}}$$

$$= \frac{\partial (-\log \hat{y}_l)}{\partial \hat{y}_l} \frac{\partial \hat{y}_l}{\partial a_{L_i}}$$

Does $\hat{y}_l$ depend on a_{L_i}? Yes.

$$\hat{y}_l = \frac{e^{a_{L_l}}}{\sum_{i=1}^{k} e^{a_{L_i}}}$$

Now we can derive the full expression

$$\frac{\partial}{\partial a_{L_i}} -\log \hat{y}_l = \frac{-1}{\hat{y}_l} \frac{\partial}{\partial a_{L_i}} \hat{y}_l$$

$$= \frac{-1}{\hat{y}_l} \frac{\partial}{\partial a_{L_i}} \text{softmax}(a_L)_l$$

$$= \frac{-1}{\hat{y}_l} \frac{\partial}{\partial a_{L_i}} \frac{e^{a_L}}{\Sigma e^{a_{L_i}}}$$

$$= -\frac{1}{\hat{y}_l} \left[\frac{\partial}{\partial a_{L_i}} e^{a_{L_l}} \cdots - \frac{e^{a_{L_l}} \left(\frac{\partial}{\partial a_{L_i}} \sum_{i'} e^{a_{L_{i'}}} \right)}{\cdots} \right]$$

$$a_L = [a_{L_1}\ a_{L_2} \cdots a_{L_k}]$$

$$\hat{y} = \text{softmax}(a_L)$$

$$\hat{y} = \left[\frac{e^{L_1}}{\Sigma e^{L_i}} \frac{e^{L_2}}{\Sigma e^{L_i}} \cdots \frac{e^{L_k}}{\Sigma e^{L_i}} \right]$$

$$= -\frac{1}{\hat{y}_l}\left[\frac{\partial a_{L_l}}{\sum_{i'} e^{a_{L i'}}} - \frac{e^{a_{L_l}}\left(\partial a_{L_i} \sum_{i'}\right)}{\left(\sum_{i'} \exp(a_L)_{i'}\right)^2}\right]$$

$$= \frac{-1}{\hat{y}_l}\left[\frac{\mathbb{1}_{l=i}\, e^{a_{L_l}}}{\sum_{i'} e^{a_{L i'}}} - \frac{e^{a_{L_l}}}{\sum_{i'} e^{a_{L i'}}} \cdot \frac{e^{a_{L i}}}{\sum_{i'} e^{a_{L i'}}}\right]$$

$$= -\frac{1}{\hat{y}_l}\left[\mathbb{1}_{l=i}\, \text{softmax}(a_L)_l - \text{softmax}(a_L)_l\, \text{softmax}(a_L)_i\right]$$

$$= \frac{-1}{\hat{y}_l}\left[\mathbb{1}_{l=i}\, \hat{y}_l - \hat{y}_l\, \hat{y}_i\right]$$

$$= \frac{-1}{\hat{y}_l}\, \hat{y}_l \left[\mathbb{1}_{l=i} - \hat{y}_i\right]$$

$$\nabla_{a_{L_i}} L(\theta) = \frac{\partial L(\theta)}{\partial a_{L_i}} = -\left[\mathbb{1}_{l=i} - \hat{y}_i\right]$$

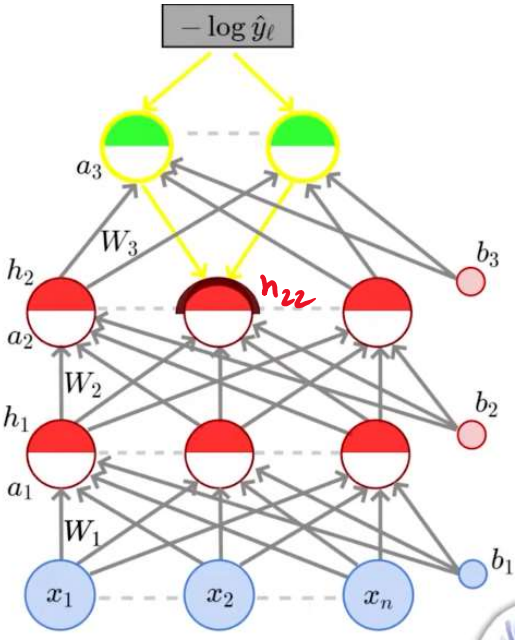
Partial derivative of loss function wrt one of the elements of output vector

Now for the entire output vector a_L:

$$\nabla_{a_L} L(\theta) = \begin{bmatrix} \frac{\partial L(\theta)}{\partial a_{L_1}} \\ \frac{\partial L(\theta)}{\partial a_{L_2}} \\ \vdots \\ \frac{\partial L(\theta)}{\partial a_{L_K}} \end{bmatrix} = \begin{bmatrix} -\left(\mathbb{1}_{l=1} - \hat{y}_1\right) \\ -\left(\mathbb{1}_{l=2} - \hat{y}_2\right) \\ \vdots \\ -\left(\mathbb{1}_{l=k} - \hat{y}_k\right) \end{bmatrix}$$

$$\nabla_{a_L} L(\theta) = \frac{\partial L(\theta)}{\partial a_L} = -(e_k - \hat{y})$$

Gradient wrt Hidden Units



$a_{i+1} = W_{i+1} h_{ij} + b_{i+1}$

The derivative of the loss $-\log \hat{y}_\ell$ depends on h_{22}. There are multiple paths (say k paths) from h_{22} to loss

Let $p(z)$ be loss function of $L(\theta)$

$z = h_{ij}$

$q_m(z) = a_{Lm}$

If a function $p(z)$ can be written as function of intermediate results $q_i(z)$ then we have:

$$\frac{\partial p(z)}{\partial z} = \sum_m \frac{\partial p(z)}{\partial q_m} \frac{\partial q_m}{\partial z}$$

We are interested in $\nabla_{h_2} L(\theta)$ but since h_2 (hidden layer) is a vector, we'll focus on one element of h_2 i.e. h_{22} we'll write it h_{ij} where i is the layer number & j is the neuron

$$\frac{\partial L(\theta)}{\partial h_{ij}} = \sum_{m=1}^{k} \frac{\partial L(\theta)}{\partial a_{i+1,m}} \cdot \frac{\partial a_{i+1,m}}{\partial h_{ij}}$$

$$= \sum_{m=1}^{k} \frac{\partial L(\theta)}{\partial a_{i+1,m}} W_{i+1,m,j}$$

We know $a_3 = W_3 h_2 + b_3$

$$\begin{bmatrix} a_{31} \\ a_{32} \end{bmatrix} = \begin{bmatrix} W_{311} & W_{312} & W_{313} \\ W_{321} & W_{322} & W_{323} \end{bmatrix} \begin{bmatrix} h_{21} \\ h_{22} \\ h_{23} \end{bmatrix} + \begin{bmatrix} b_{31} \\ b_{32} \end{bmatrix}$$

$$\begin{bmatrix} a_{31} \\ a_{32} \end{bmatrix} = \begin{bmatrix} w_{311} & w_{312} & w_{313} \\ w_{321} & w_{322} & w_{323} \end{bmatrix} \begin{bmatrix} h_{21} \\ h_{22} \\ h_{23} \end{bmatrix} + \begin{bmatrix} b_{31} \\ b_{32} \end{bmatrix}$$

only two neurons in current layer

(2×3)

(3×1)

3 neurons in previous layer

2 biases

$$a_{31}^{i+1} = w_{311} h_{21} + w_{312} h_{22} + w_{313} h_{23} + b_{31}$$

(m)

$$\frac{\partial a_{31}}{\partial h_{22}} = 0 + w_{312} + 0 + 0 = w_{312}^{m}$$

(i, j on the left; $i+1$, j on the right)

We know, $\nabla_{a_{i+1}} L(\theta) = \begin{bmatrix} \frac{\partial L(\theta)}{\partial a_{i+1,1}} \\ \frac{\partial L(\theta)}{\partial a_{i+1,2}} \\ \vdots \\ \frac{\partial L(\theta)}{\partial a_{i+1,k}} \end{bmatrix}$

$w_{i+1,:,j} = \begin{bmatrix} w_{i+1,1,j} \\ w_{i+1,2,j} \\ \vdots \\ w_{i+1,k,j} \end{bmatrix}$; j^{th} column of w_{i+1}

$$(w_{i+1,:,j})^T \nabla_{a_{i+1}} L(\theta) = \sum_{m=1}^{k} \frac{\partial L(\theta)}{\partial a_{i+1,m}} w_{i+1,m,j}$$

$\therefore$ We have $\frac{\partial L(\theta)}{\partial h_{ij}} = (w_{i+1,:,j})^T \nabla_{a_{i+1}} L(\theta)$

Now we can compute gradient wrt any hidden layer h_i

$$\begin{bmatrix} \frac{\partial L(\theta)}{\partial h_{i1}} \\ \vdots \end{bmatrix} \quad \begin{bmatrix} (w_{i+1,:,1})^T \nabla_{a_{i+1}} L(\theta) \\ \vdots \end{bmatrix}$$

$$\nabla_{h_i} L(\theta) = \begin{bmatrix} \frac{\partial L(\theta)}{\partial h_{i1}} \\ \frac{\partial L(\theta)}{\partial h_{i2}} \\ \vdots \\ \frac{\partial L(\theta)}{\partial h_{in}} \end{bmatrix} = \begin{bmatrix} (W_{i+1,\cdot,1})^T \nabla_{a_{i+1}} L(\theta) \\ (W_{i+1,\cdot,2})^T \nabla_{a_{i+1}} L(\theta) \\ \vdots \\ (W_{i+1,\cdot,n})^T \nabla_{a_{i+1}} L(\theta) \end{bmatrix}$$

$$\boxed{\nabla_{h_i} L(\theta) = (W_{i+1})^T \left(\nabla_{a_{i+1}} L(\theta) \right)}$$

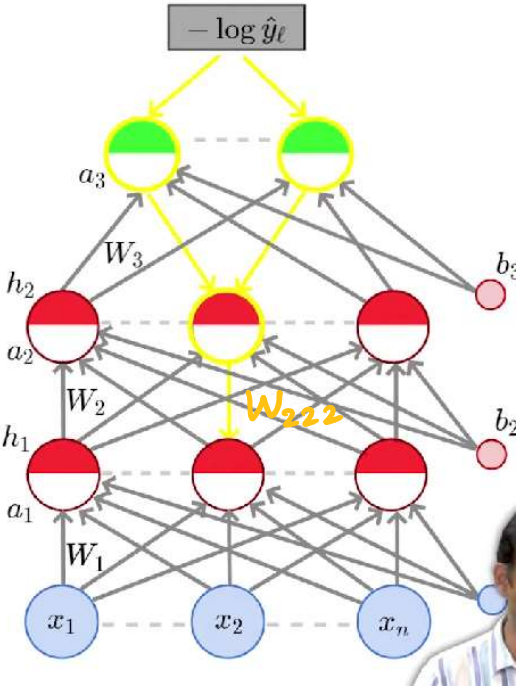
Now we need to know how to calculate $\nabla_{a_{i+1}} L(\theta)$ for $i < L-1$

$$\frac{\partial L(\theta)}{\partial a_{ij}} = \frac{\partial L(\theta)}{\partial h_{ij}} \cdot \frac{\partial h_{ij}}{\partial a_{ij}}$$

$$= \frac{\partial L(\theta)}{\partial h_{ij}} g'(a_{ij}) \qquad \left[\because h_{ij} = g(a_{ij}) \right]$$

$$\nabla_{a_i} L(\theta) = \begin{bmatrix} \frac{\partial L(\theta)}{\partial h_{i1}} g'(a_{i1}) \\ \frac{\partial L(\theta)}{\partial h_{i2}} g'(a_{i2}) \\ \vdots \\ \frac{\partial L(\theta)}{\partial h_{in}} g'(a_{in}) \end{bmatrix}$$

Gradient wrt Parameters



Now we want to calculate derivative of loss function wrt one of the weight matrices.

$$\nabla_{W_k} L(\theta)$$

Let's start by computing derivative wrt one element.

$$\frac{\partial L(\theta)}{\partial W_{kij}}$$

We know, $a_k = b_k + W_k h_{k-1}$

$$\frac{\partial L(\theta)}{\partial W_{kij}} = \frac{\partial L(\theta)}{\partial a_{ki}} \frac{\partial a_{ki}}{\partial W_{kij}}$$

$$a_k = \begin{bmatrix} a_{k1} \\ a_{k2} \\ a_{k3} \end{bmatrix} = \begin{bmatrix} W_{k11} & W_{k12} & W_{k13} \\ W_{k21} & W_{k22} & W_{k23} \\ W_{k31} & W_{k32} & W_{k33} \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \end{bmatrix}$$

(ignoring bias)

$$\therefore a_{k2} = W_{k21} \cdot h_{11} + W_{k22} \cdot h_{12} + W_{k23} \cdot h_{13}$$

$$\frac{\partial a_{k2}}{\partial W_{k22}} = h_{12}^{k-1}$$

$$\frac{\partial L(\theta)}{\partial W_{kij}} = \frac{\partial L(\theta)}{\partial a_{ki}} h_{k-1,j}$$

$$\frac{\partial W_{kij}}{\partial a_{ki}}$$

$\therefore$ for entire weight W_k the gradient will be:

$$\nabla_{W_k} L(\theta) = \begin{bmatrix} \frac{\partial L(\theta)}{\partial W_{k11}} & \frac{\partial L(\theta)}{\partial W_{k12}} & \cdots & \frac{\partial L(\theta)}{\partial W_{k1n}} \\ \vdots & \cdots & & \vdots \\ \cdots & \cdots & & \frac{\partial L(\theta)}{W_{knn}} \end{bmatrix}$$

For example let's take $W \in \mathbb{R}^{3\times 3}$ then,

$$\nabla_{W_k} L(\theta) = \begin{bmatrix} \frac{\partial L(\theta)}{\partial W_{k11}} & \frac{\partial L(\theta)}{\partial W_{k12}} & \frac{\partial L(\theta)}{\partial W_{k13}} \\ \frac{\partial L(\theta)}{\partial W_{k21}} & \frac{\partial L(\theta)}{\partial W_{k22}} & \frac{\partial L(\theta)}{\partial W_{k23}} \\ \frac{\partial L(\theta)}{\partial W_{k31}} & \frac{\partial L(\theta)}{\partial W_{k32}} & \frac{\partial L(\theta)}{\partial W_{k33}} \end{bmatrix}$$

$$\frac{\partial L(\theta)}{\partial W_{kij}} = \frac{\partial L(\theta)}{\partial a_{ki}} \frac{\partial a_{ki}}{\partial W_{kij}}$$

$$\nabla_{W_k} L(\theta) = \begin{bmatrix} \frac{\partial L(\theta)}{\partial a_{k1}} h_{k-1,1} & \frac{\partial L(\theta)}{\partial a_{k1}} h_{k-1,2} & \frac{\partial L(\theta)}{\partial a_{k1}} h_{k-1,3} \\ \frac{\partial L(\theta)}{\partial a_{k2}} h_{k-1,1} & \frac{\partial L(\theta)}{\partial a_{k2}} h_{k-1,2} & \frac{\partial L(\theta)}{\partial a_{k2}} h_{k-1,3} \\ \frac{\partial L(\theta)}{} h_{\cdot\cdot} & \frac{\partial L(\theta)}{} h_{\cdot\cdot 2} & \frac{\partial L(\theta)}{} h_{\cdot\cdot 3} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial L(\theta)}{\partial a_{k3}} h_{k-1,1} & \frac{\partial L(\theta)}{\partial a_{k3}} h_{k-1,2} & \frac{\partial L(\theta)}{\partial a_{k3}} h_{k-1,3} \end{bmatrix}$$

$$\nabla_{W_k} L(\theta) = \begin{bmatrix} \frac{\partial L(\theta)}{\partial a_{k1}} \\ \frac{\partial L(\theta)}{\partial a_{k2}} \\ \frac{\partial L(\theta)}{\partial a_{k3}} \end{bmatrix} \cdot \begin{bmatrix} h_{k-1,1} & h_{k-1,2} & h_{k-1,3} \end{bmatrix}$$

$$\boxed{\nabla_{W_k} L(\theta) = \nabla_{a_k} L(\theta) \cdot h_{k-1}^T}$$

Coming to the biases,

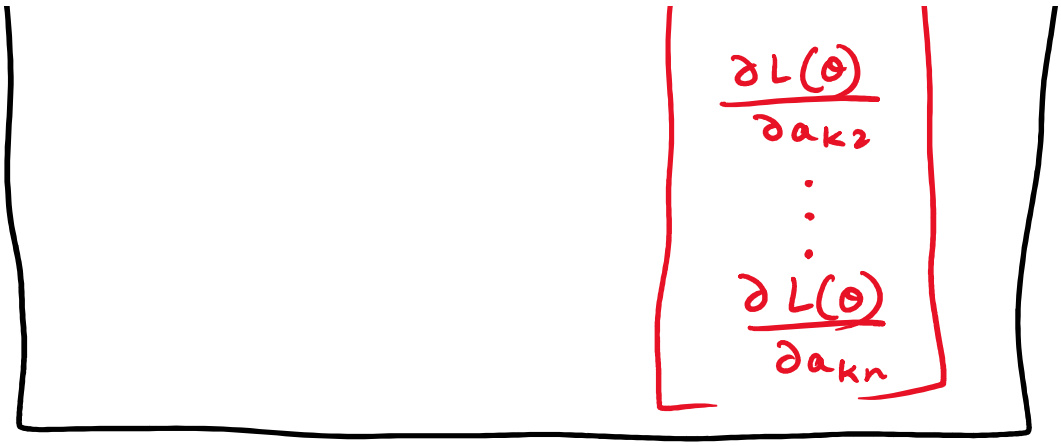
$$a_{ki} = b_{ki} + W_{kij} h_{k-1j}$$

$$\frac{\partial L(\theta)}{\partial b_{ki}} = \frac{\partial L(\theta)}{\partial a_{ki}} \cdot \frac{\partial a_{ki}}{\partial b_{ki}}$$

$$\boxed{\frac{\partial a_{ki}}{\partial b_{ki}} = 1} \quad \therefore \frac{\partial L(\theta)}{\partial b_{ki}} = \frac{\partial L(\theta)}{\partial a_{ki}}$$

Now we can write the gradient wrt b_k

$$\nabla_{b_k} L(\theta) = \nabla_{a_k} L(\theta) = \begin{bmatrix} \frac{\partial L(\theta)}{\partial a_{k1}} \\ \frac{\partial L(\theta)}{\partial} \end{bmatrix}$$


$\frac{\partial L(\Theta)}{\partial a_{k2}}$
$\frac{\partial L(\Theta)}{\partial a_{kn}}$

Backpropagation: Pseudocode

We have the gradients wrt:

Output layer: $\nabla_{a_L} L(\theta)$

Hidden layers: $\nabla_{h_k} L(\theta)$, $\nabla_{a_k} L(\theta)$ $[1 \le k < L]$

Weights & Biases: $\nabla_{W_k} L(\theta)$, $\nabla_{b_k} L(\theta)$ $[1 \le k < L]$

Algorithm

$t \leftarrow 0$

max_iterations $\leftarrow 1000$

Initialize $\theta_0 = [W_1^0, W_2^0, \cdots W_L^0, b_1^0, b_2^0, \cdots b_L^0]$

while $t{+}{+} <$ max_iterations do

  calculate $h_1, \ldots, h_{L-1}, a_1, \cdots a_L, \hat{y}$ = forward_propagation(θ_t)

  $\nabla\theta_t$ = back_propagation$(h_1, \ldots, h_{L-1}, a_1, \cdots a_L, y, \hat{y})$

  $\nabla\theta_{t+1} \leftarrow \theta_t - \eta \nabla\theta_t$

end

Forward Propagation: Algorithm

For $k = 1$ to $L-1$ do

  $a_k = b_k + W_k h_{k-1}$

  $h_k = g(a_k)$

end

end

$a_L = b_L + W_L h_{L-1}$

$\hat{y} = o(a_L)$

Back Propagation

// Compute output gradient

$\nabla_{a_L} L(\theta) = -(e(y) - \hat{y})$

For $k = L-1$ to 1 do

// compute gradient wrt parameters

$\nabla_{W_k} L(\theta) = \nabla_{a_k} L(\theta) h_{k-1}^T$

$\nabla_{b_k} L(\theta) = \nabla_{a_k} L(\theta)$

// compute gradient wrt layer below

$\nabla_{h_{k-1}} L(\theta) = W_k^T \nabla_{a_k} L(\theta)$

// compute gradient wrt layer below (pre-activation)

$\nabla_{a_{k-1}} L(\theta) = \nabla_{h_{k-1}} L(\theta) \odot [\cdots, g'(a_{k-1,j}) \cdots]$

end

This is how we can compute g'

Logistic Function

$g(z) = \sigma(z)$

$= \frac{1}{1+e^{-z}}$

Tanh Function

$g(z) = \tanh(z)$

$= \frac{e^z - e^{-z}}{e^z + e^{-z}}$

$= \frac{1}{1+e^{-z}}$

$g'(z) = (-1)\frac{1}{(1+e^{-z})^2}\frac{d}{dz}(1+e^{-z})$

$= \frac{(-1)}{(1+e^{-z})^2}(-e^{-z})$

$= \frac{1}{(1+e^{-z})}\frac{1+e^{-z}-1}{1+e^{-z}}$

$g'(z) = g(z)(1-g(z))$

$= \frac{}{e^{z}+e^{-z}}$

$g(z) = \frac{d}{dz}\left(\frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}\right)$

$g(z) = 1-(g(z))^2$