Neural Network Learning

KAUSHIK ROY

Neural Networks

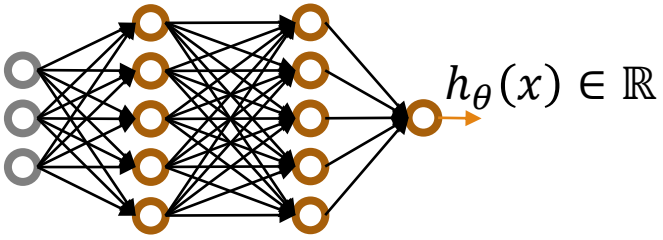
Classification

Training set consists m training samples: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$

L = total no. of layers in the network

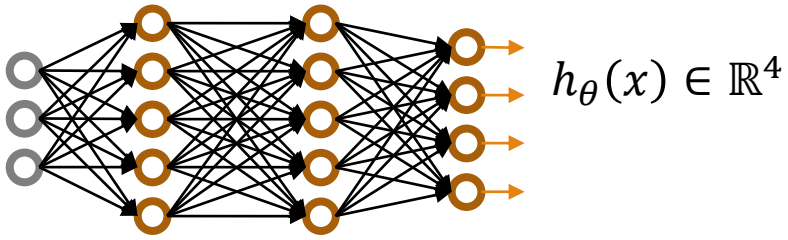
s_l = no. of units (not counting bias unit) in layer l

Binary classification



$y = 0$ or 1

Multi-class classification



$y \in \mathbb{R}^4$ E.g. $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

Each output neuron performs a binary classification task.

The number of output neurons = number of classes.

C-BRIC

Neural Networks

Cost function

Recall the logistic regression cost function:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[log\left(h_\theta\left(x^{(i)}\right)\right) * y^{(i)} + log\left(1 - h_\theta\left(x^{(i)}\right)\right) * \left(1 - y^{(i)}\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

Neural network cost function:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[log\left(h_\theta\left(x^{(i)}\right)\right)_k * y_k^{(i)} + log\left(1 - \left(h_\theta\left(x^{(i)}\right)\right)_k\right) * \left(1 - y_k^{(i)}\right)\right]$$
$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(\theta_{ji}^{(l)}\right)^2$$

In which, $h_\theta(x) \in \mathbb{R}^K$, and $\left(h_\theta(x)\right)_k = k^{th}$ output, and $1 \leq k \leq K$ (outputs)

C-BRIC

Neural Networks

- Gradient computation

Recall Neuron network cost function:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[log\left(h_\theta(x^{(i)})\right)_k * y_k^{(i)} + log\left(1-\left(h_\theta(x^{(i)})\right)_k\right) * \left(1-y_k^{(i)}\right)\right]$$
$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(\theta_{ji}^{(l)}\right)^2$$

In which, $h_\theta(x) \in \mathbb{R}^K$, and $\left(h_\theta(x)\right)_k = k^{th}$ output, and $1 \le k \le K$

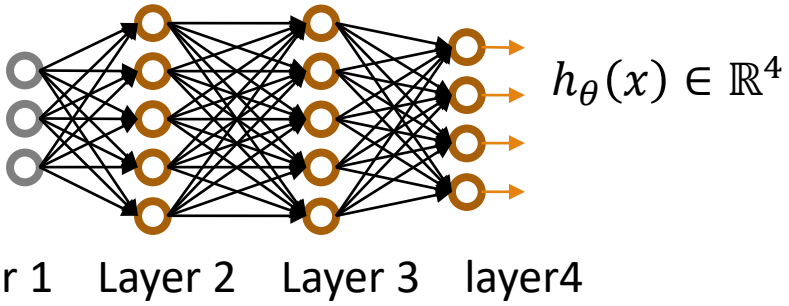
To minimize $J(\theta)$, we need to compute the gradient: $\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta)$

C-BRIC

Neural Networks

Gradient computation

E.g. Assume a 4 layer network and one training example (x, y):



In the forward path:
Recall forward propagation:

$$a^{(1)} = x$$
$$z^{(2)} = \theta^{(1)} a^{(1)}$$
$$a^{(2)} = g\left(z^{(2)}\right) \quad \left(\text{add } a_0^{(2)}\right)$$
$$z^{(3)} = \theta^{(2)} a^{(2)}$$
$$a^{(3)} = g\left(z^{(3)}\right) \quad \left(\text{add } a_0^{(3)}\right)$$
$$z^{(4)} = \theta^{(3)} a^{(3)}$$
$$a^{(4)} = h_\theta(x) = g\left(z^{(4)}\right)$$

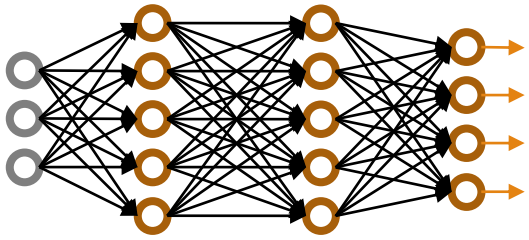
Dimensions equal number of neurons in the layer
E.g.

$$a^{(1)} \in \mathbb{R}^3$$
$$a^{(2)} \in \mathbb{R}^5$$
$$a^{(3)} \in \mathbb{R}^5$$
$$a^{(4)} \in \mathbb{R}^4$$

C-BRIC

Neural Networks

- Gradient computation



In the backward path:

Intuition:

$\delta_j^{(l)}$ = "error" of node j in layer l

For each output unit (layer L=4)

$$\delta^{(4)} = a^{(4)} - y = \left(h_\theta(x)\right) - y$$
$$\delta^{(3)} = \left(\theta^{(3)}\right)^T \delta^{(4)} .* g'\left(z^{(3)}\right)$$
$$\delta^{(2)} = \left(\theta^{(2)}\right)^T \delta^{(3)} .* g'\left(z^{(2)}\right)$$

No need to compute $\delta^{(1)}$ for input layer

Dimensions equal number of neurons in the layer
E.g.

$\delta^{(4)} \in \mathbb{R}^4$
$\delta^{(3)} \in \mathbb{R}^5$
$\delta^{(2)} \in \mathbb{R}^5$

$g'\left(z^{(3)}\right) = a^{(3)} .* (1- a^{(3)})$

$g'\left(z^{(2)}\right) = a^{(2)} .* (1- a^{(2)})$

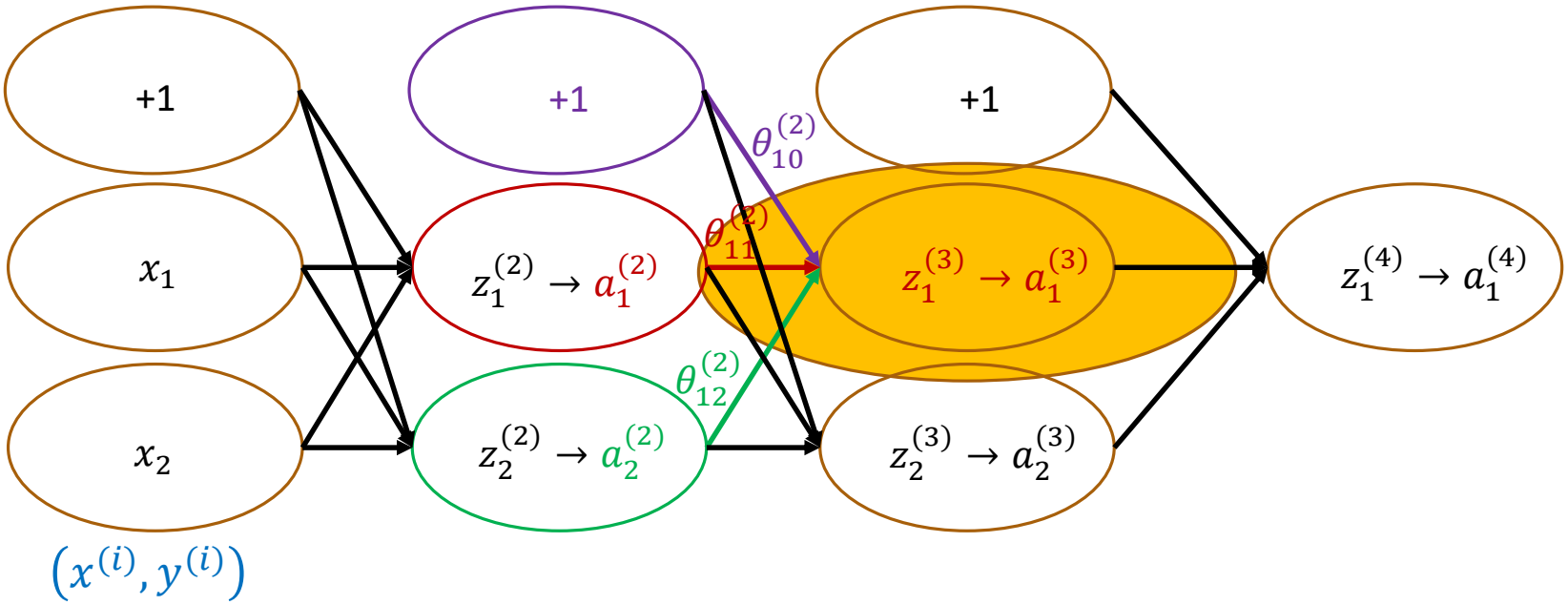
One can show: $\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta) = a_j^{(l)} \delta_i^{(l+1)}$

C-BRIC

Backpropagation

➢ Forward Propagation

Forward path:
In the forward path of backpropagation, input data propagates in the forward direction layer by layer until it reaches the output layer. The input to one layer is the output of the previous layer.



E.g. Outputs from layer 2 are used to compute $z_1^{(3)}$ in layer 3:

$$z_1^{(3)} = \theta_{10}^{(2)} \times 1 + \theta_{11}^{(2)} \times a_1^{(2)} + \theta_{12}^{(2)} \times a_2^{(2)}$$

C-BRIC

Backpropagation

- Backpropagation

What is backpropagation doing?

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[log\left(h_\theta\left(x^{(i)}\right)\right)_k * y_k^{(i)} + log\left(1 - \left(h_\theta\left(x^{(i)}\right)\right)_k\right) * \left(1 - y_k^{(i)}\right)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(\theta_{ji}^{(l)}\right)^2$$

Focusing on a single example $x^{(i)}, y^{(i)}$, the case of 1 output unit, and ignoring regularization ($\lambda = 0$),

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

(Think of $cost(i) \approx \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$)

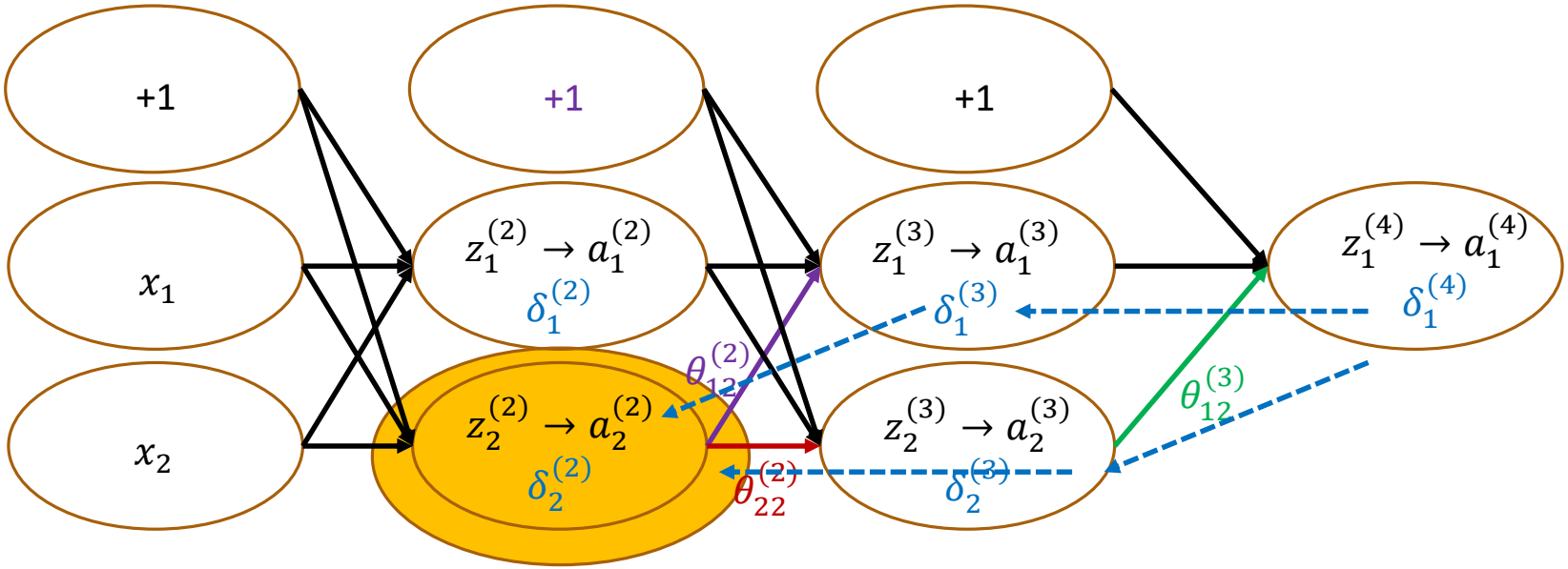
How well is the network doing on example i?

C-BRIC

Backpropagation

- Backpropagation

Backward path:



$\delta_j^{(l)}$ = "error" of cost for $a_j^{(l)}$ (unit j in layer l)

Formally, $\delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} cost(i)$ (for $j \geq 0$), where

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

E.g. $\delta_1^{(4)} = y^{(i)} - a_1^{(4)}$

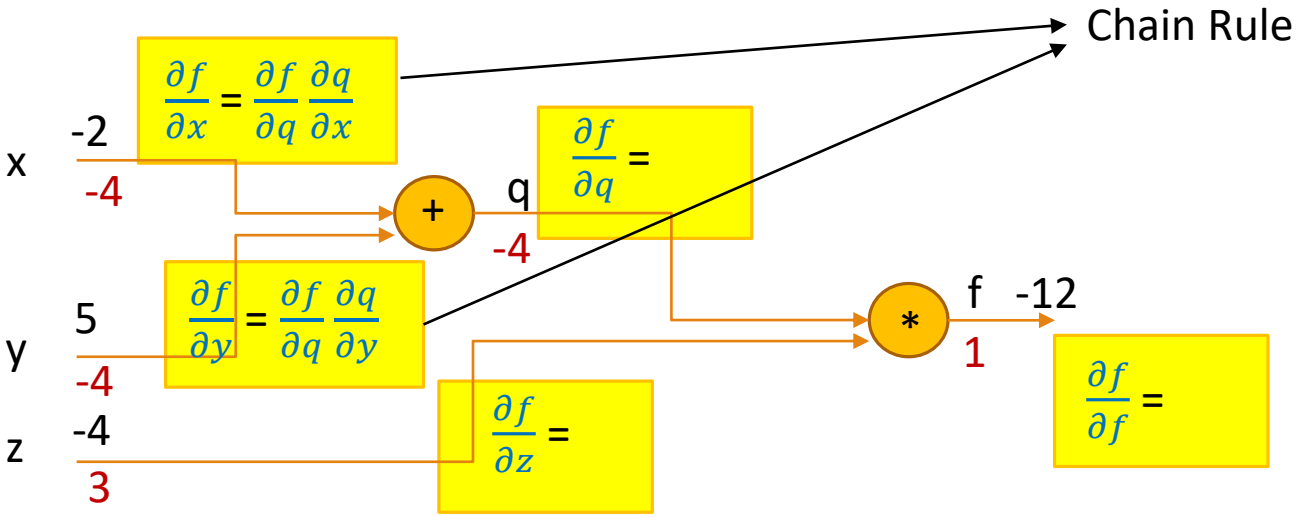
$\delta_2^{(3)} = \theta_{12}^{(3)} \delta_1^{(4)}$

$\delta_2^{(2)} = \theta_{12}^{(2)} \delta_1^{(3)} + \theta_{22}^{(2)} \delta_2^{(3)}$

C-BRIC

Example Backpropagation

- f (x,y,z) = (x + y)z



$q = (x + y)$

$\frac{\partial q}{\partial x} = 1$

$\frac{\partial q}{\partial y} = 1$

$f = qz$

$\frac{\partial f}{\partial q} = z$

$\frac{\partial f}{\partial z} = q$

Example:

- f(w,x) = sigmoid

Backpropagation

- Backpropagation algorithm

Training set $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$

Set $\Delta_{ij}^{(l)} = 0$ (for all l, i, j) (used to compute $\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta)$)

For $i = 1$ *to* m $(x^{(m)}, y^{(m)})$

Set $a^{(1)} = x^{(i)}$

Perform forward propagation to compute $a^{(l)}$ for $l = 2,3, \ldots, L$

Using $y^{(i)}$, compute $\delta^{(L)} = a^{(L)} - y^{(i)}$ (for output layer)

Compute $\delta^{(L-1)}, \delta^{(L-2)}, \ldots, \delta^{(2)}$ (using BP; no need to compute $\delta^{(1)}$)

$\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$ (possible to vectorize: $\Delta^{(l)} := \Delta^{(l)} + \delta^{(l+1)} (a^{(l)})^T$)

$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} + \lambda \theta_{ij}^{(l)}$, if $j \neq 0$

$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)}$, if $j = 0$; bias term

If can be shown that $\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta) = D_{ij}^{(l)}$

C-BRIC