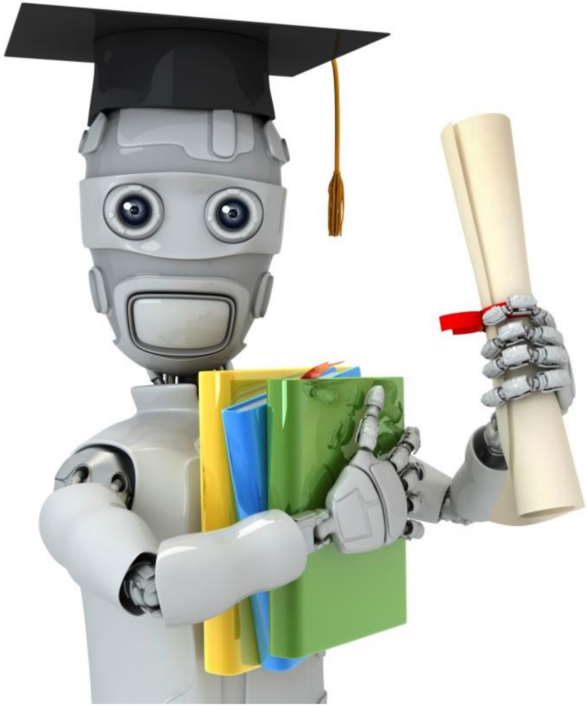
CSE 4621
Machine Learning

Lecture 8

Md. Hasanul Kabir, PhD.
Professor, CSE Department
Islamic University of Technology (IUT)



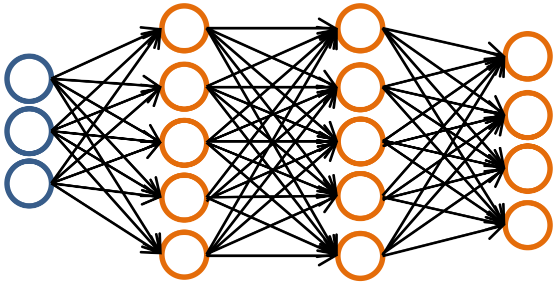
Neural Networks: Learning

Cost function

Machine Learning

Source & Special Thanks to (Coursera) Machine Learning / NN&DL Courses

Neural Network (Classification)



$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$$

$L =$ total no. of layers in network

$n^{[l]} =$ no. of units (not counting bias unit) in layer l

Binary classification

$y = 0 \text{ or } 1$

1 output unit

Multi-class classification (K classes)

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

pedestrian car motorcycle truck

K output units

Cost function

Logistic regression:

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

Neural network:

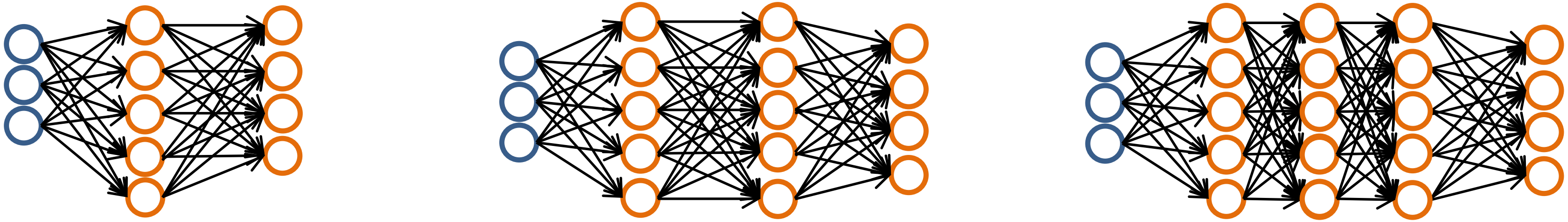
$$h_\Theta(x) \in \mathbb{R}^K \quad (h_\Theta(x))_i = i^{th} \text{ output}$$

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1-y_k^{(i)}) \log(1-(h_\Theta(x^{(i)}))_k)\right]$$

$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{n^{[l]}}\sum_{j=1}^{n^{[l-1]}}\left(\Theta_{ij}^{[l]}\right)^2$$

Training a neural network

Pick a network architecture (connectivity pattern between neurons)



No. of input units: Dimension of features $x^{(i)}$

No. output units: Number of classes

Reasonable default: 1 hidden layer, or if >1 hidden layer, have same no. of hidden units in every layer (usually the more the better)

deeplearning.ai

One hidden layer Neural Network

Gradient descent for neural networks

Gradient Computation

Cost Function: $$J(W,b)=\frac{1}{m}\sum_{i=1}^{m}L(\hat{\mathrm{y}}^{(i)},\mathrm{y}^{(i)})$$ + Regularization Term

Objective: $$\min J(W,b)$$

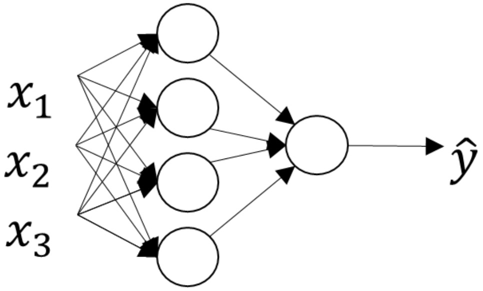
Need to Compute:

- $J(W,b)$
- $\dfrac{\partial J(W,b)}{\partial w_{ij}^{(l)}}$ $\dfrac{\partial J(W,b)}{\partial b_{i}^{(l)}}$

Use Update equation for both W & b

Gradient descent for neural networks

Forward Calculation



$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$$

$$\mathrm{A}^{[1]} = \begin{bmatrix} | & | & & | \\ a^{[1](1)} & a^{[1](2)} & \cdots & a^{[1](m)} \\ | & | & & | \end{bmatrix}$$

```
for i = 1 to m
```

$$z^{[1](i)} = W^{[1]}x^{(i)} + b^{[1]}$$
$$a^{[1](i)} = \sigma(z^{[1](i)})$$
$$z^{[2](i)} = W^{[2]}a^{[1](i)} + b^{[2]}$$
$$a^{[2](i)} = \sigma(z^{[2](i)})$$

$$Z^{[1]} = W^{[1]}X + b^{[1]}$$
$$A^{[1]} = \sigma(Z^{[1]})$$
$$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$
$$A^{[2]} = \sigma(Z^{[2]})$$

Formulas for computing derivatives

deeplearning.ai

One hidden layer Neural Network

Backpropagation intuition (Calculation)

Computing gradients

Logistic regression

$$\begin{array}{l} x \searrow \\ w \rightarrow \boxed{z = w^T x + b} \rightarrow \boxed{a = \sigma(z)} \rightarrow \boxed{\mathcal{L}(a, y)} \\ b \nearrow \end{array}$$

Neural network gradients

$$
\begin{tikzcd}
x \arrow[dr] & & & W^{[2]},\ b^{[2]} \arrow[d] & & \\
W^{[1]} \arrow[r] & \boxed{z^{[1]} = W^{[1]}x + b^{[1]}} \arrow[r] & \boxed{a^{[1]} = \sigma(z^{[1]})} \arrow[r] & \boxed{z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}} \arrow[r] & \boxed{a^{[2]} = \sigma(z^{[2]})} \arrow[r] & \boxed{\mathcal{L}(a^{[2]}, y)} \\
b^{[1]} \arrow[ur] & & & & &
\end{tikzcd}
$$

Neural network gradients

$W^{[2]}$

$b^{[2]}$

x

$W^{[1]}$

$b^{[1]}$

$$z^{[1]} = W^{[1]}x + b^{[1]} \rightarrow a^{[1]} = \sigma(z^{[1]}) \rightarrow z^{[2]} = W^{[2]}a^{[1]} + b^{[2]} \rightarrow a^{[2]} = \sigma(z^{[2]}) \rightarrow \mathcal{L}(a^{[2]}, \mathrm{y})$$

Summary of gradient descent

$$dz^{[2]} = a^{[2]} - y$$

$$dW^{[2]} = dz^{[2]} {a^{[1]}}^T$$

$$db^{[2]} = dz^{[2]}$$

$$dz^{[1]} = W^{[2]T} dz^{[2]} * g^{[1]\prime}(\mathrm{z}^{[1]})$$

$$dW^{[1]} = dz^{[1]} x^T$$

$$db^{[1]} = dz^{[1]}$$

Summary of gradient descent

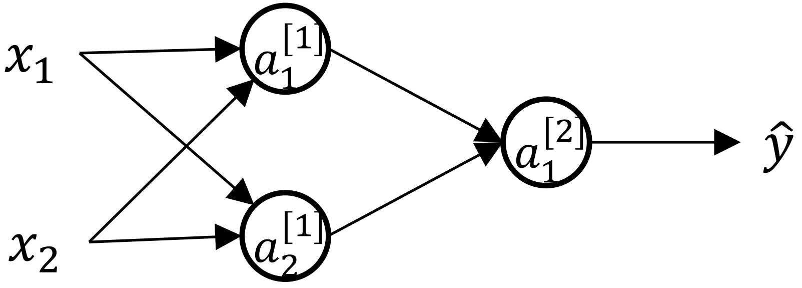
$dz^{[2]} = a^{[2]} - y$	$dZ^{[2]} = A^{[2]} - Y$
$dW^{[2]} = dz^{[2]}{a^{[1]}}^T$	$dW^{[2]} = \frac{1}{m} dZ^{[2]}{A^{[1]}}^T$
$db^{[2]} = dz^{[2]}$	$db^{[2]} = \frac{1}{m} np.sum(dZ^{[2]}, axis = 1, keepdims = True)$
$dz^{[1]} = W^{[2]T} dz^{[2]} * g^{[1]\prime}(\mathrm{z}^{[1]})$	$dZ^{[1]} = W^{[2]T} dZ^{[2]} * g^{[1]\prime}(\mathrm{Z}^{[1]})$
$dW^{[1]} = dz^{[1]} x^T$	$dW^{[1]} = \frac{1}{m} dZ^{[1]} X^T$
$db^{[1]} = dz^{[1]}$	$db^{[1]} = \frac{1}{m} np.sum(dZ^{[1]}, axis = 1, keepdims = True)$

One hidden layer Neural Network

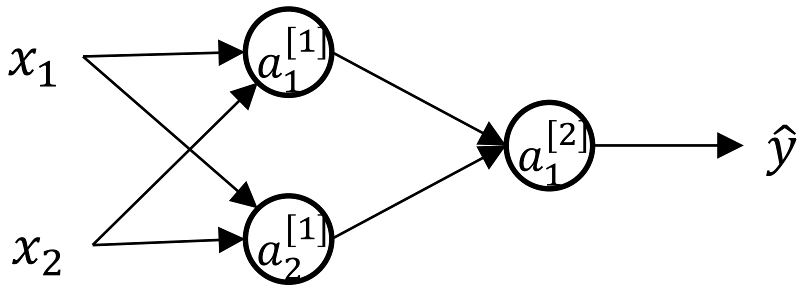
Random Initialization

deeplearning.ai

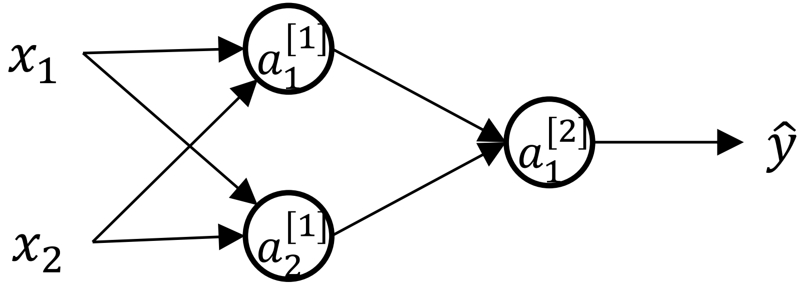
What happens if you initialize weights to zero?

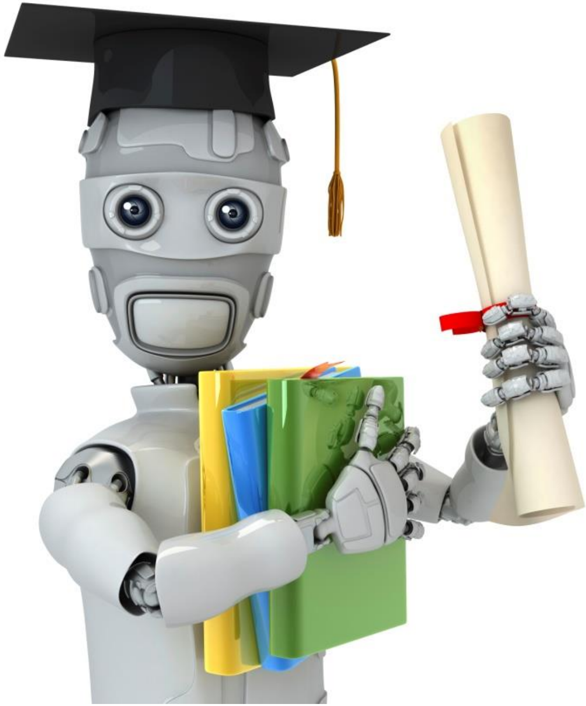


What happens if you initialize weights to zero?



Random initialization





Machine Learning

Neural Networks: Learning

Backpropagation example: Autonomous driving

Direction choosen by human driver

Direction selected by learning algorithm

[Courtesy of Dean Pomerleau]