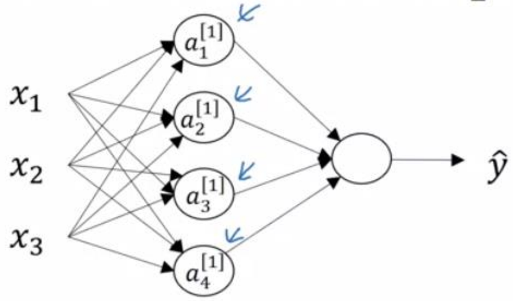
Week 3

$a_i^{[l]}$ ← layer

i ← node in layer.

Neural Network Representation



$(w_1^{[1]})^T x$

$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}, \quad a_1^{[1]} = \sigma(z_1^{[1]})$$

$$z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]}, \quad a_2^{[1]} = \sigma(z_2^{[1]})$$

$$z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]}, \quad a_3^{[1]} = \sigma(z_3^{[1]})$$

$$z_4^{[1]} = w_4^{[1]T} x + b_4^{[1]}, \quad a_4^{[1]} = \sigma(z_4^{[1]})$$

$$z^{[1]} = \underbrace{\begin{bmatrix} — w_1^{[1]T} — \\ — w_2^{[1]T} — \\ — w_3^{[1]T} — \\ — w_4^{[1]T} — \end{bmatrix}}_{(4,3)} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix} = \begin{bmatrix} w_1^{[1]T} x + b_1^{[1]} \\ w_2^{[1]T} x + b_2^{[1]} \\ w_3^{[1]T} x + b_3^{[1]} \\ w_4^{[1]T} x + b_4^{[1]} \end{bmatrix} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix}$$

Vectorizing across multiple examples

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

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

$a^{[1](i)} = \sigma(z^{[1](i)})$

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

$a^{[2](i)} = \sigma(z^{[2](i)})$

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

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

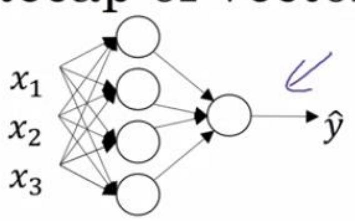
(n_x, m)

$$Z^{[1]} = \begin{bmatrix} | & | & & | \\ z^{[1](1)} & z^{[1](2)} & \cdots & z^{[1](m)} \\ | & | & & | \end{bmatrix}$$

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

Andr

Recap of vectorizing across multiple examples




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

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

$a^{[1](i)} = \sigma(z^{[1](i)})$

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

$a^{[2](i)} = \sigma(z^{[2](i)})$

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

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

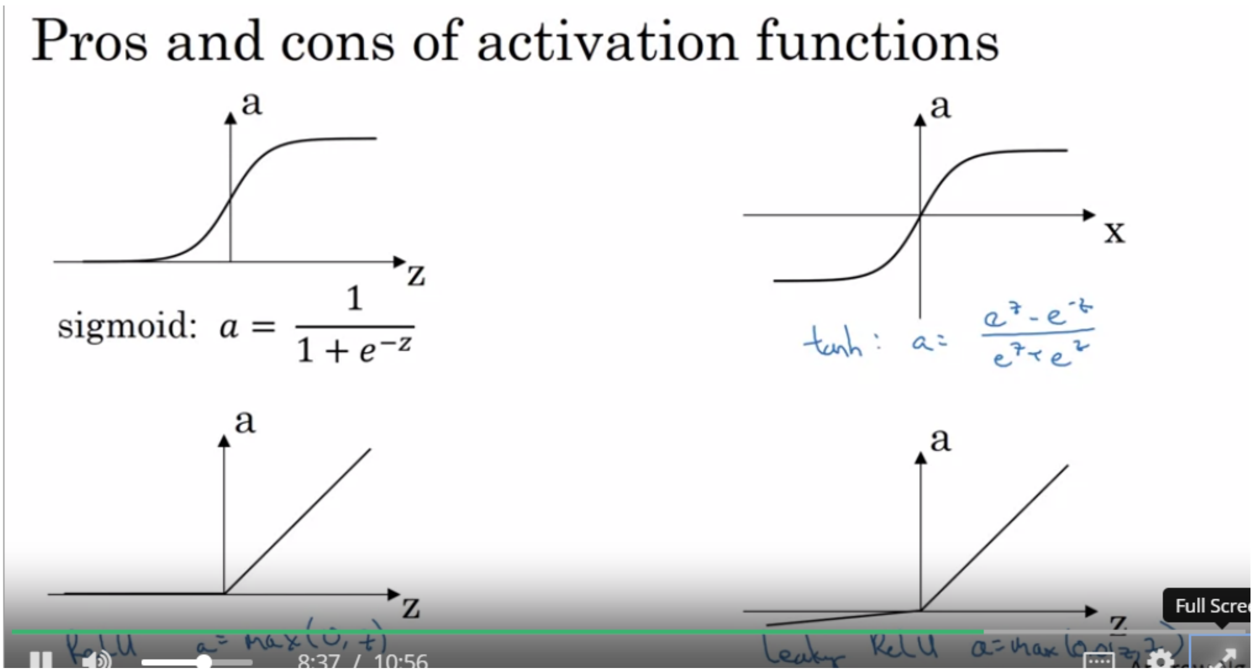
$Z^{[1]} = W^{[1]}X + b^{[1]}$

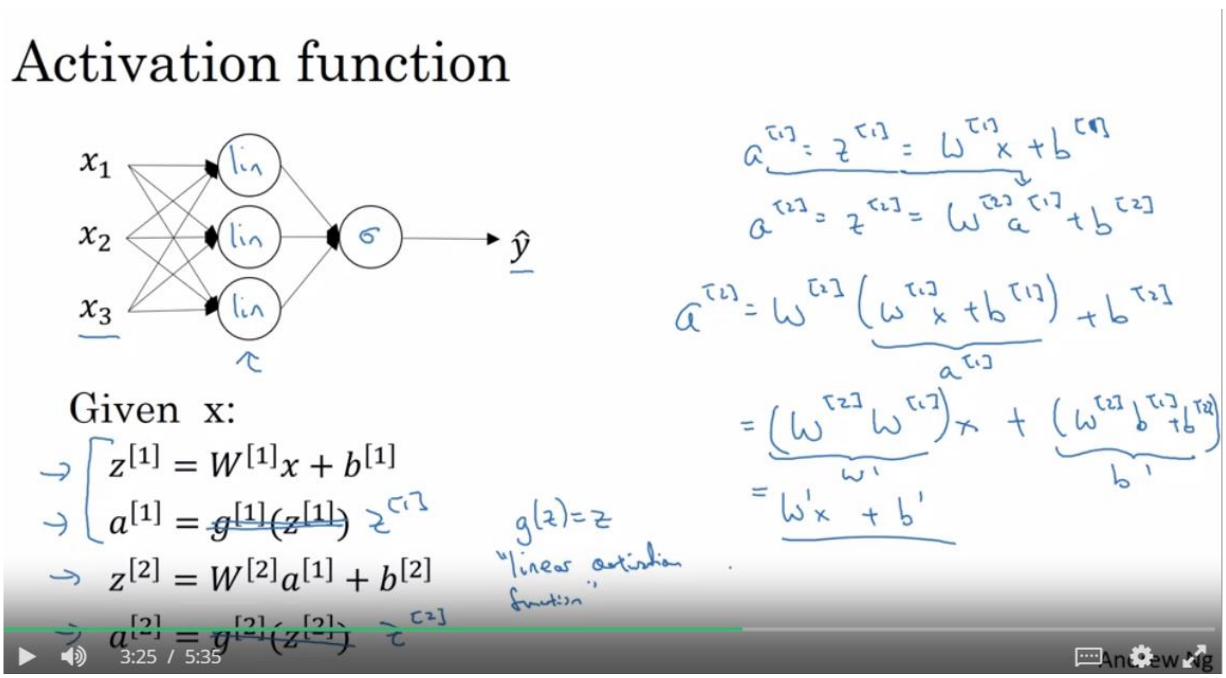
$A^{[1]} = \sigma(Z^{[1]})$

$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$

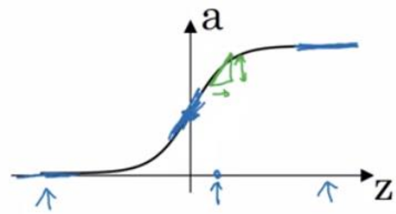
$A^{[2]} = \sigma(Z^{[2]})$

Andrew Ng


Pros and cons of activation functions
a
z
sigmoid: $a = \frac{1}{1+e^{-z}}$
a
x
tanh: $a = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
a
z
a
z
ReLU $a = \max(0, z)$
Leaky ReLU
Full Scre
8:37 / 10:56


Activation function
x_1
x_2
x_3
lin
lin
lin
σ
$\hat{y}$
Given x:
$z^{[1]} = W^{[1]}x + b^{[1]}$
$a^{[1]} = \cancel{g^{[1]}(z^{[1]})}\ z^{[1]}$
$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$
$a^{[2]} = \cancel{g^{[2]}(z^{[2]})}\ z^{[2]}$
$g(z) = z$
"linear activation function"
$a^{[1]} = z^{[1]} = W^{[1]}x + b^{[1]}$
$a^{[2]} = z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$
$a^{[2]} = W^{[2]}(\underbrace{W^{[1]}x + b^{[1]}}_{a^{[1]}}) + b^{[2]}$
$= \underbrace{(W^{[2]}W^{[1]})}_{W'}x + \underbrace{(W^{[2]}b^{[1]} + b^{[2]})}_{b'}$
$= W'x + b'$
3:25 / 5:35
Andrew Ng

Sigmoid activation function



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

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

$$\boxed{g'(z)} = \boxed{\frac{d}{dz}g(z)} = \text{slope of } g(x) \text{ at } z$$

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

$$= g(z)\left(1-g(z)\right) \leftarrow$$

$$= \boxed{a\,(1-a)}$$

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

$z = 10. \quad g(z) \approx 1$

$\frac{d}{dz}g(z) \approx 1\,(1-1) \approx 0$

$z = -10 \quad g(z) \approx 0$

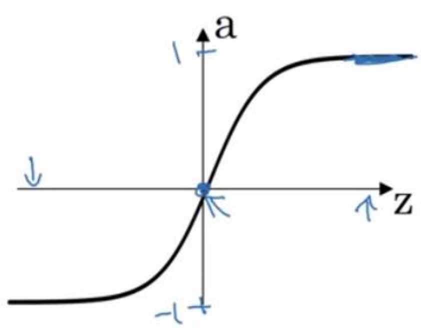
$\frac{d}{dz}g(z) \approx 0 \cdot (1-0) \approx 0$

$z = 0 \quad g(z) = \frac{1}{2}$

$\frac{d}{dz}g(z) = \frac{1}{2}\left(1-\frac{1}{2}\right) = \frac{1}{4}$

3:47 / 7:57

Tanh activation function



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

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

$$g'(z) = \frac{d}{dz}g(z) = \text{slope of } g(z) \text{ at } z$$

$$= 1 - (\tanh(z))^2 \leftarrow$$

$$a = g(z), \quad g'(z) = 1 - a^2$$

$z = 10 \quad \tanh(z) \approx 1$

$g'(z) \approx 0$

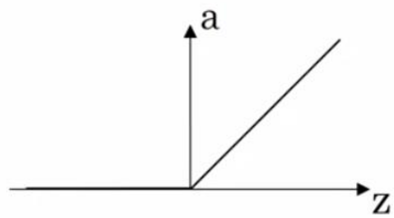
$z = -10 \quad \tanh(z) \approx -1$

$g'(z) \approx 0$

$z = 0 \quad \tanh(z) = 0$

$g'(z) = 1$

ReLU and Leaky ReLU

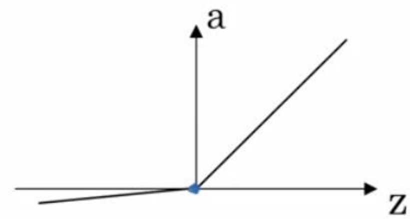


ReLU

$g(z) = \max(0, z)$

$$\rightarrow g'(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \\ \cancel{\text{undef}} & \cancel{\text{if } z = 0} \end{cases}$$

$z = 0.0000\ldots0$



Leaky ReLU

$g(z) = \max(0.01z, z)$

$$g'(z) = \begin{cases} 0.01 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

Andrew N

Formulas for computing derivatives

Forward propagation:

$Z^{[1]} = W^{[1]}X + b^{[1]}$

$A^{[1]} = g^{[1]}(Z^{[1]}) \leftarrow$

$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$

$A^{[2]} = g^{[2]}(Z^{[2]}) = \underline{\sigma}(Z^{[2]})$

Back propagation:

$dZ^{[2]} = A^{[2]} - Y \swarrow$ $\qquad Y = [y^{(1)}\ y^{(2)} \ldots, y^{(m)}]$

$dW^{[2]} = \frac{1}{m} dZ^{[2]} A^{[1]T}$ $\qquad (n^{[1]},) \leftarrow$

$db^{[2]} = \frac{1}{m} \underline{\text{np.sum}}(dZ^{[2]}, \underline{\text{axis}=1}, \underline{\text{keepdims}=\text{True}})$ $\qquad \swarrow (n^{[2]}, 1) \leftarrow$

$dZ^{[1]} = \underbrace{W^{[2]T} dZ^{[2]}}_{(n^{[1]}, m)} * \underbrace{g^{[1]\prime}(Z^{[1]})}_{(n^{[1]}, m)}$

$\uparrow$ element-wise product

$dW^{[1]} = \frac{1}{m} dZ^{[1]} X^T$

$db^{[1]} = \frac{1}{m} \text{np.sum}(dZ^{[1]}, \text{axis}=1, \underline{\text{keepdims}=\text{True}})$

$(n^{[1]}, 1) \qquad (n^{[1]},)$

Computing gradients

Logistic regression

$x_1, x_2, x_3 \rightarrow \sigma \rightarrow \hat{y} = a$

$x, w, b \rightarrow \boxed{z = w^T x + b} \rightarrow \boxed{a = \sigma(z)} \rightarrow \boxed{\mathcal{L}(a, y)}$

$dz = a - y$

$da = \frac{d}{da}\mathcal{L}(a,y) = \underline{-y\log a - (1-y)\log(1-a)}$

$= -\frac{y}{a} + \frac{1-y}{1-a}$

$dw =$

db

$\boxed{dz = da \cdot g'(z)}$

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

$\boxed{\underbrace{\frac{\partial \mathcal{L}}{\partial z}}_{\text{"}dz\text{"}} = \underbrace{\frac{\partial \mathcal{L}}{\partial a}}_{\text{"}da\text{"}} \cdot \boxed{\frac{da}{dz}}}$

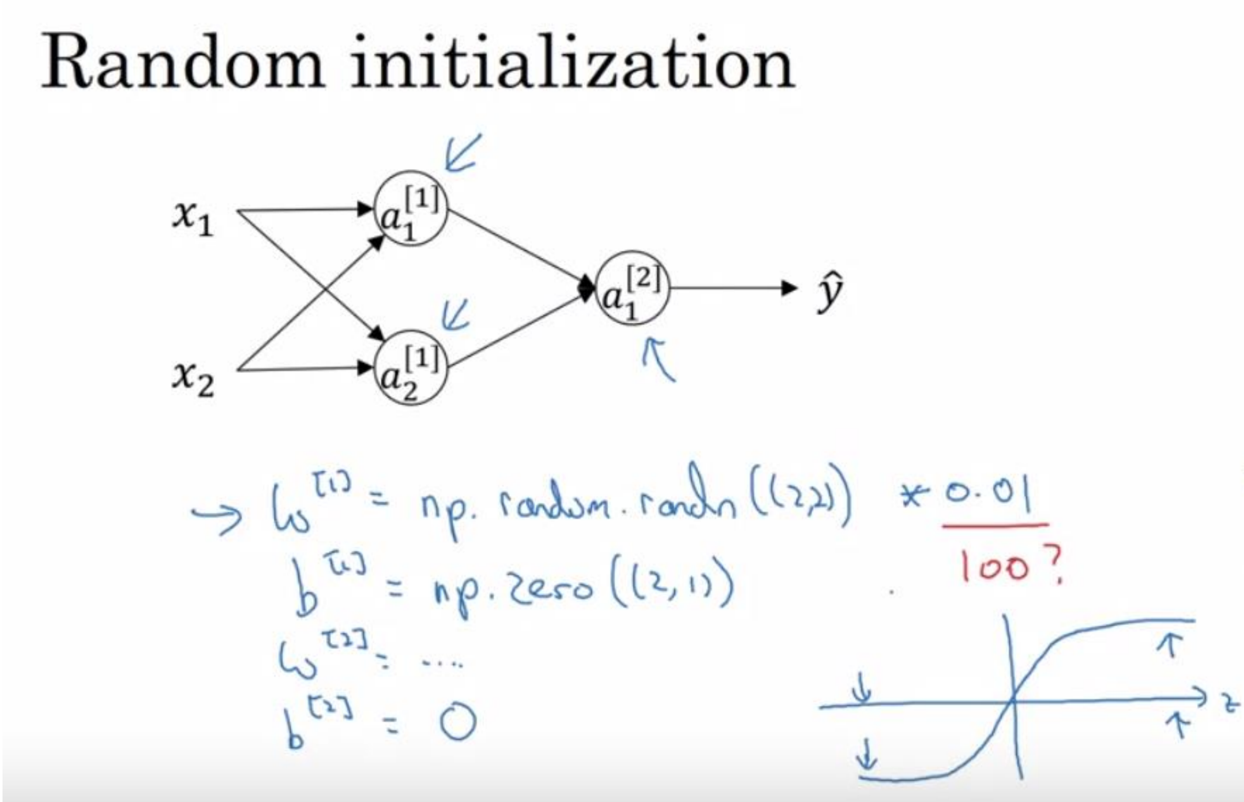
$\frac{d}{dz} g(z) = g'(z)$

Andrew Ng

Summary of gradient descent

$J(\ldots) = \frac{1}{m}\sum_{i=1}^{n} \mathcal{L}(\hat{y}, y)$

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




```
In [6]: # Train the logistic regression classifier
        clf = sklearn.linear_model.LogisticRegressionCV();
        clf.fit(X.T, Y.T);
```

You can now plot the decision boundary of these models. Run the code below.

```
In [7]: # Plot the decision boundary for logistic regression
        plot_decision_boundary(lambda x: clf.predict(x), X, Y)
        plt.title("Logistic Regression")

        # Print accuracy
        LR_predictions = clf.predict(X.T)
        print ('Accuracy of logistic regression: %d ' % float((np.dot(Y,LR_predictions) + np.dot(1-Y,1-LR_predictions))/float(Y.size)*100
               '% ' + "(percentage of correctly labelled datapoints)")
```

```
Accuracy of logistic regression: 47 % (percentage of correctly labelled datapoints)
```