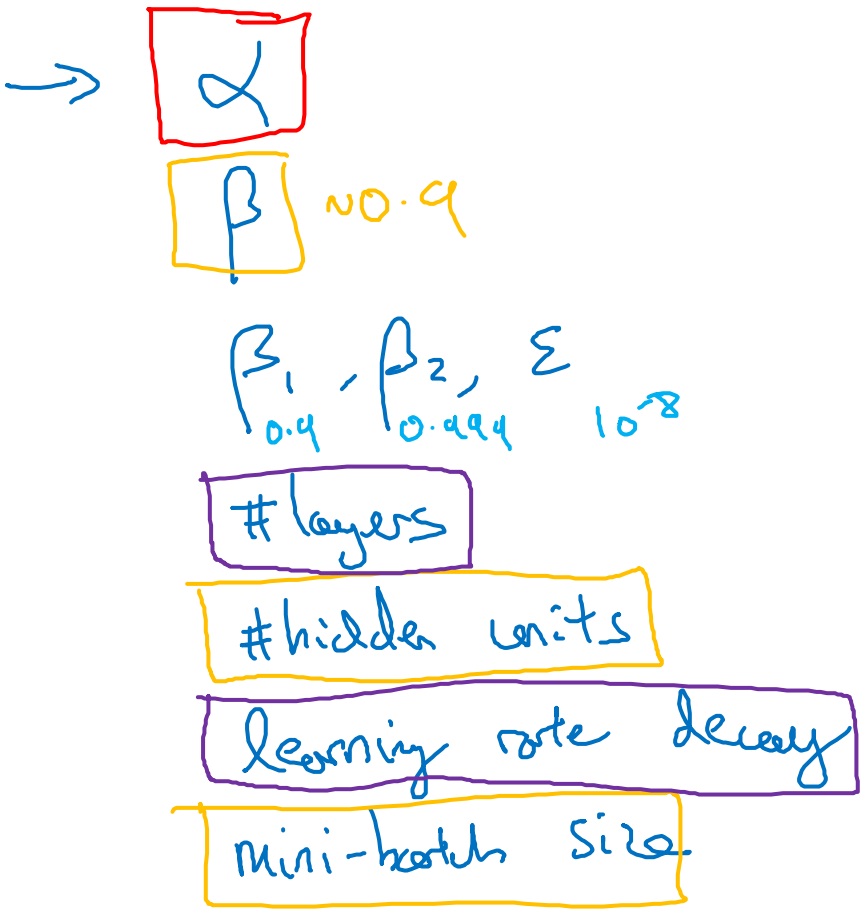
deeplearning.ai

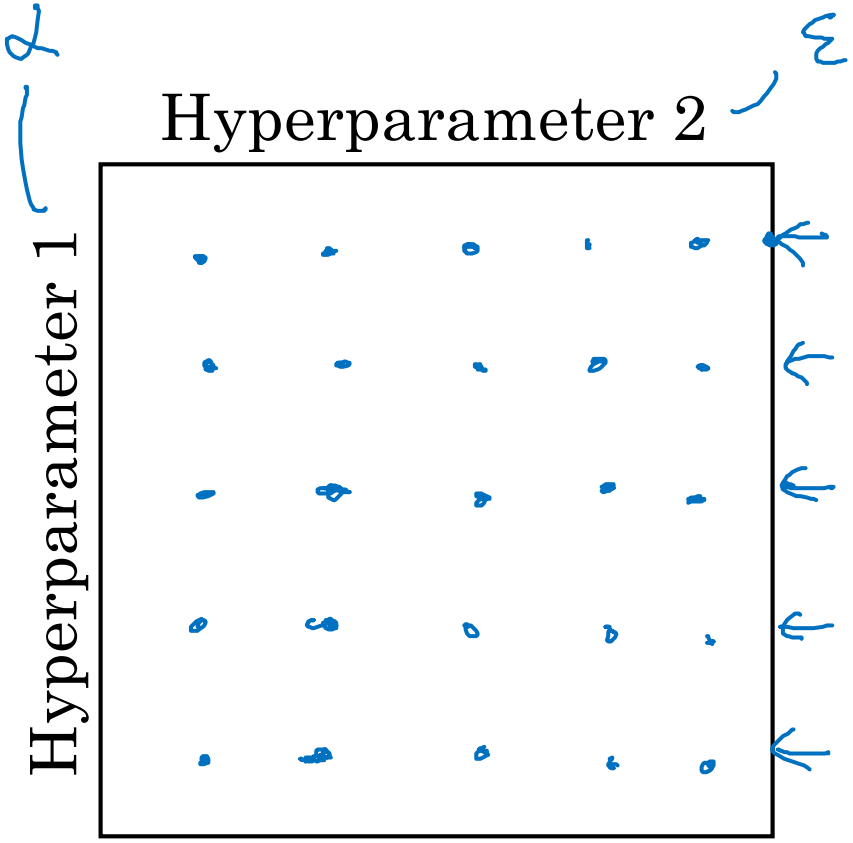
# Hyperparameter tuning

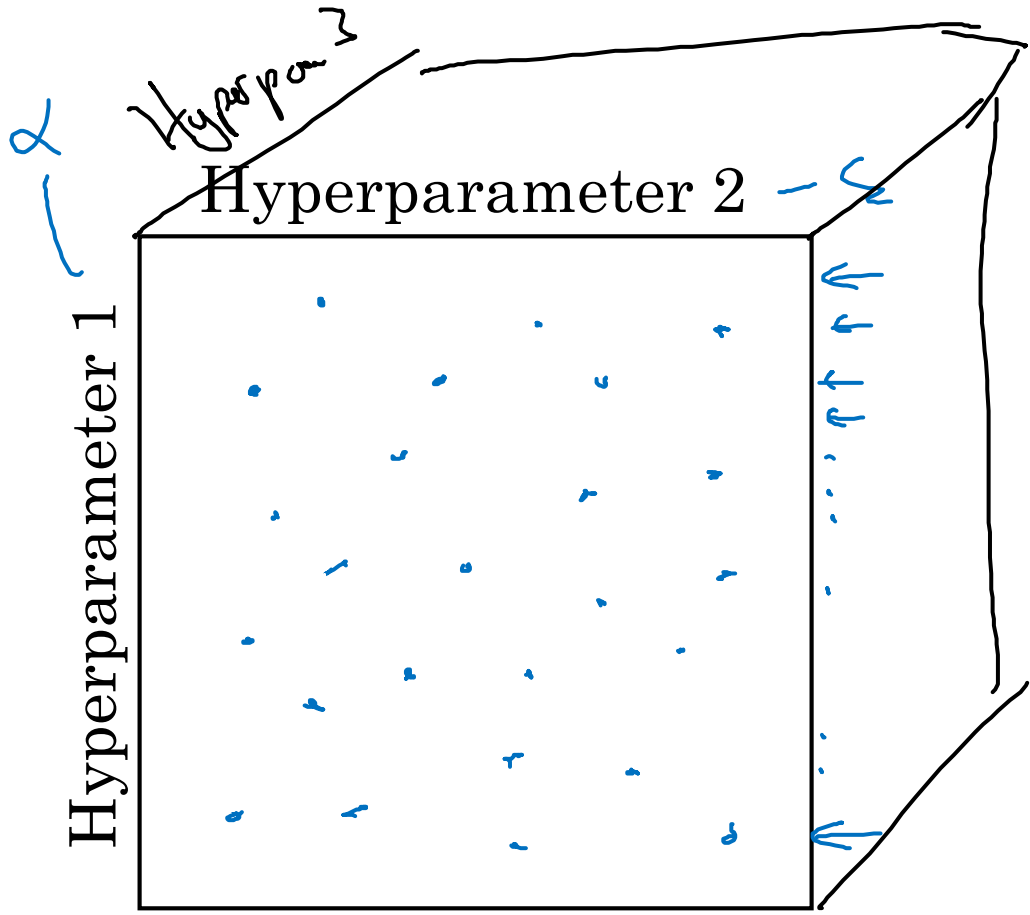
## Tuning process

# Hyperparameters



# Try random values: Don't use a grid



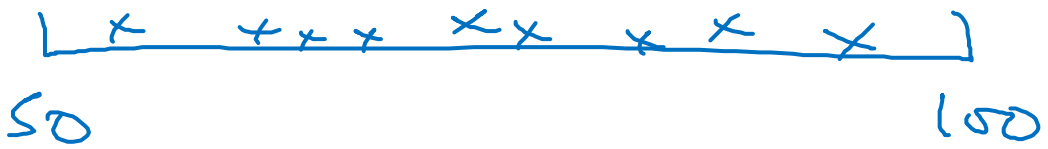
deeplearning.ai

# Hyperparameter tuning

## Using an appropriate scale to pick hyperparameters

# Picking hyperparameters at random

→ $n^{[l]} = 50, \ldots, 100$

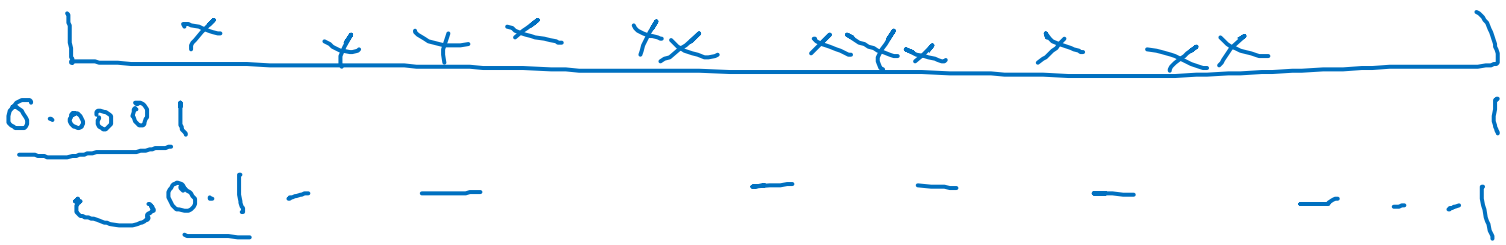


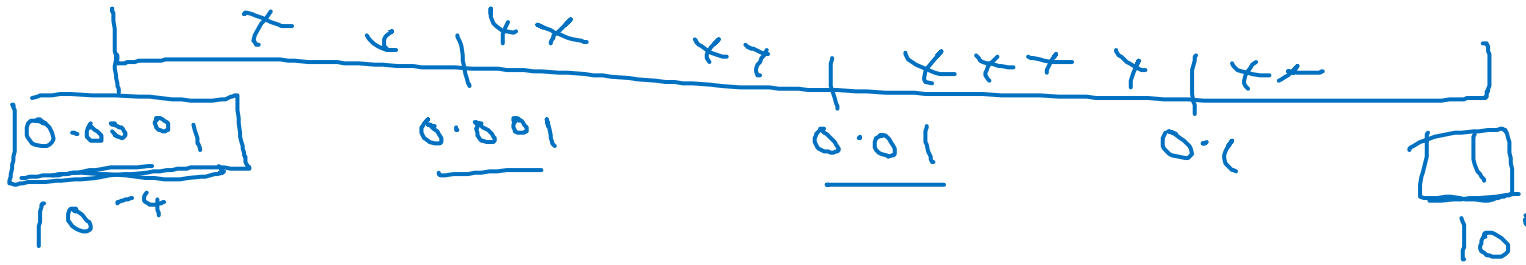

→ #layers $L$: 2 – 4

2, 3, 4

# Appropriate scale for hyperparameters

$\alpha = 0.0001, \ldots, 1$




$10^{a}$ $10^{b}$

$a = \log_{10} 0.0001 = -4$

$r = -4 * np.random.rand()$ $\leftarrow$ $r \in [-4, 0]$

$\alpha = 10^{r}$ $\leftarrow$ $10^{-4} \ldots 10^{0}$

$b = \log_{10} 1 = 0$

$10^{a} \ldots 10^{b}$ $\quad r \in [a, b]$ $\quad \alpha = 10^{r}$

$[-4, 0]$

Andrew Ng

deeplearning.ai

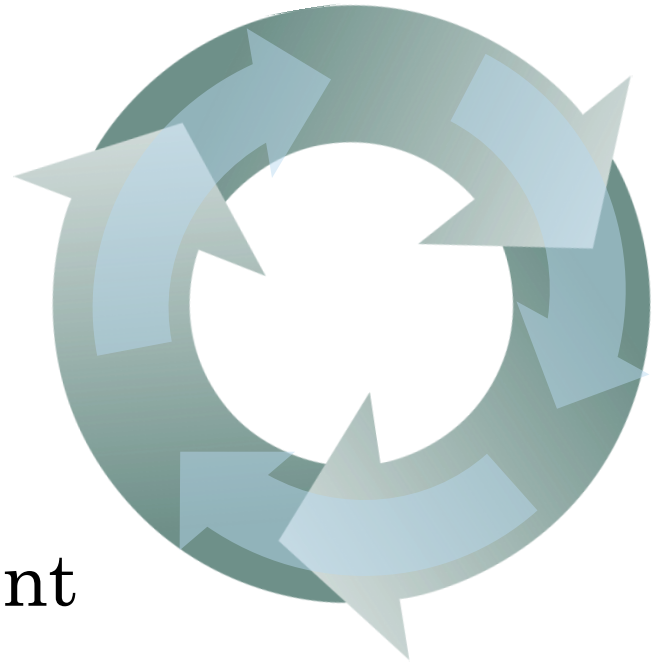
# Hyperparameters tuning

---

# Hyperparameters tuning in practice: Pandas vs. Caviar

# Re-test hyperparameters occasionally

Idea



Experiment

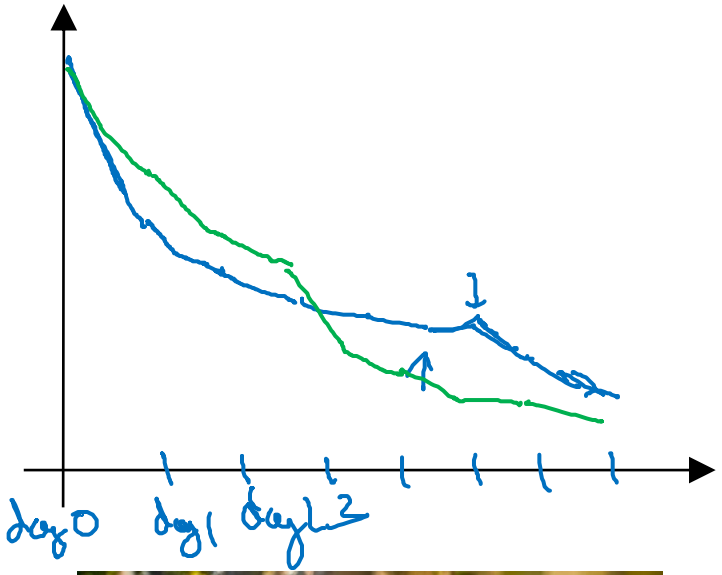
Code

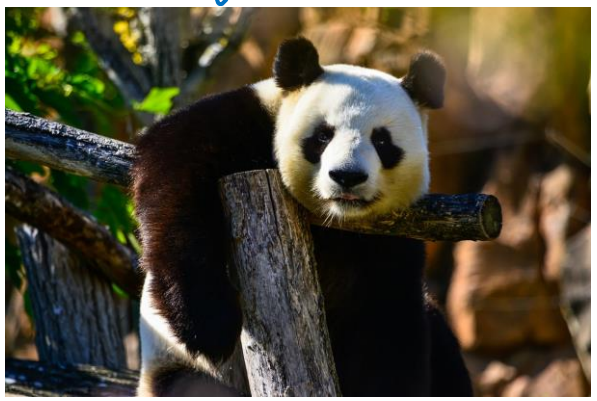
- NLP, Vision, Speech, Ads, logistics, ….

- Intuitions do get stale. Re-evaluate occasionally.

Andrew Ng

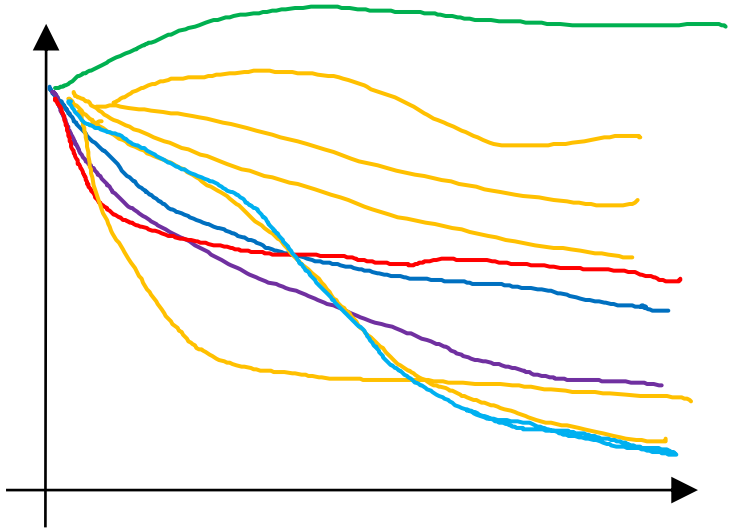
# Babysitting one model



Panda

# Training many models in parallel



Caviar

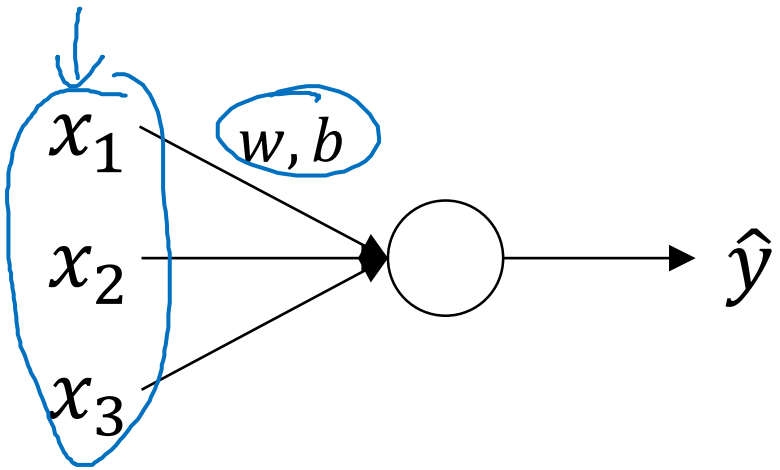
Andrew Ng

deeplearning.ai

# Batch Normalization

## Normalizing activations in a network

# Normalizing inputs to speed up learning

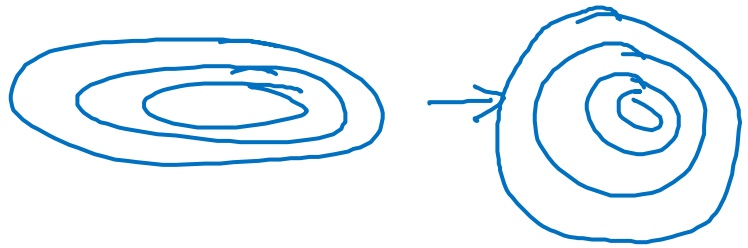


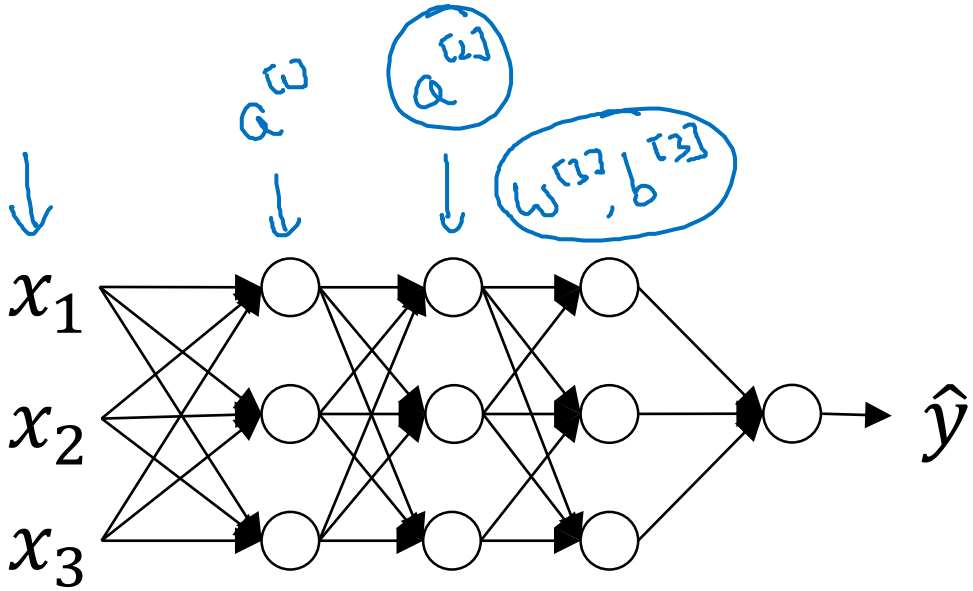

$$\mu = \frac{1}{m} \sum_i x^{(i)}$$

$$X = X - \mu \quad \text{← element-wise}$$

$$\sigma^2 = \frac{1}{m} \sum_i x^{(i)2}$$

$$X = X / \sigma^2$$






Can we normalize $a^{[2]}$ so as to train $w^{[3]}, b^{[3]}$ faster?

Normalize $z^{[2]}$

# Implementing Batch Norm

Given some intermediate values in NN $z^{(1)}, \ldots, z^{(m)}$ — $z^{[l](i)}$

$\mu = \frac{1}{m} \sum_i z^{(i)}$

$\sigma^2 = \frac{1}{m} \sum_i (z_i - \mu)^2$

$z^{(i)}_{norm} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$ ←

$\tilde{z}^{(i)} = \gamma z^{(i)}_{norm} + \beta$ ← learnable parameters of model.

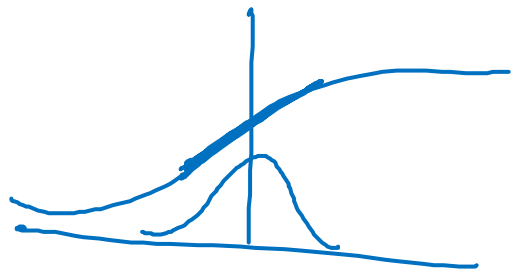
If $\gamma = \sqrt{\sigma^2 + \varepsilon}$ ←

$\beta = \mu$ ←

then $\tilde{z}^{(i)} = z^{(i)}$

$X$ ←

$z^{(i)}$ ←



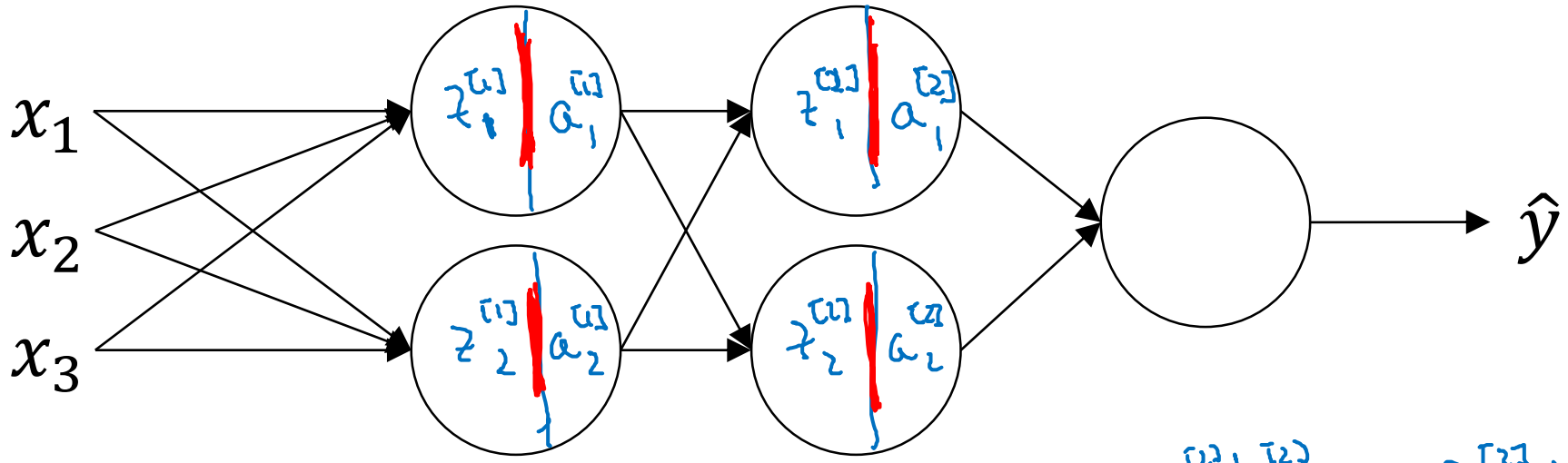
Use $\tilde{z}^{[l](i)}$ instd of $z^{[l](i)}$.

deeplearning.ai

# Batch Normalization

## Fitting Batch Norm into a neural network

# Adding Batch Norm to a network




$X \xrightarrow{W^{[1]}, b^{[1]}} z^{[1]} \xrightarrow[\text{Batch Norm (BN)}]{\beta^{[1]}, \gamma^{[1]}} \tilde{z}^{[1]} \rightarrow a^{[1]} = g^{[1]}(\tilde{z}^{[1]}) \xrightarrow{W^{[2]}, b^{[2]}} z^{[2]} \xrightarrow[BN]{\beta^{[2]}, \gamma^{[2]}} \tilde{z}^{[2]} \rightarrow a^{[2]} \rightarrow \dots$

Parameters: $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, \dots, W^{[L]}, b^{[L]}, \beta^{[1]}, \gamma^{[1]}, \beta^{[2]}, \gamma^{[2]}, \dots, \beta^{[L]}, \gamma^{[L]}$

$\rightarrow \beta$

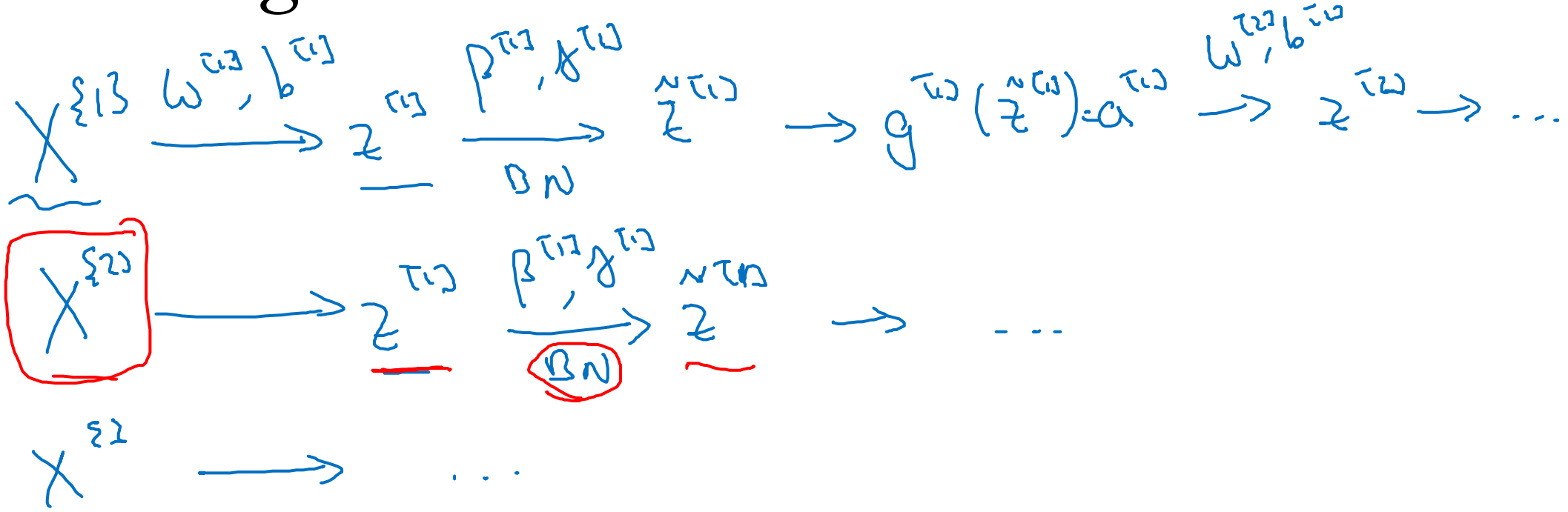
$d\beta^{[l]}$

$\beta^{[l]} = \beta^{[l]} - \alpha\, d\beta^{[l]}$

tf.nn.batch_normalization

Andrew Ng

# Working with mini-batches



Parameters: $w^{[l]}$, ~~$b^{[l]}$~~, $\beta^{[l]}$, $\gamma^{[l]}$.

$(n^{[l]}, 1)$

$z^{[l]}$

$(n^{[l]}, 1)$

$z = w^{[l]} a^{[l-1]} + b^{[l]}$

$z = w^{[l]} a^{[l-1]}$

$z^{[l]}_{norm}$

$z^{[l]} = \gamma^{[l]} z^{[l]}_{norm} + \beta^{[l]}$

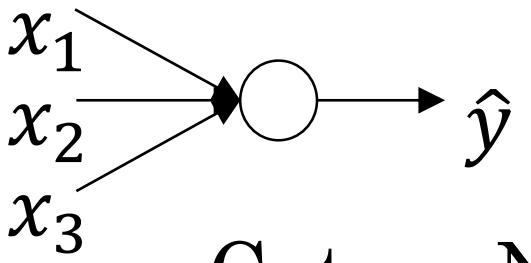
Andrew Ng

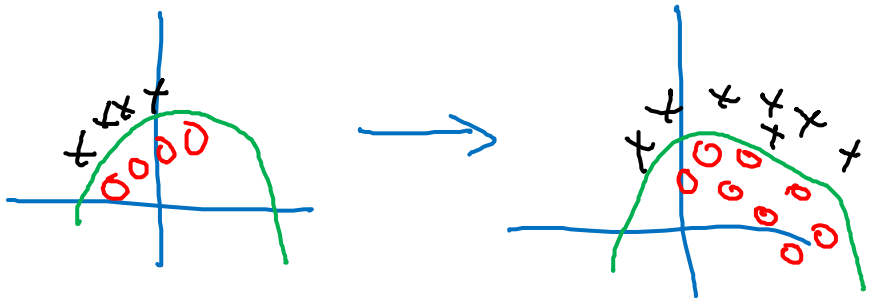
deeplearning.ai

# Batch Normalization

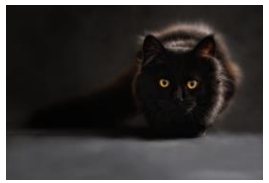
## Why does Batch Norm work?

# Learning on shifting input distribution



Cat $y = 1$

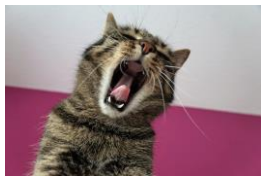
Non-Cat $y = 0$




"Covariate shift"

$X \rightarrow y$

$y = 1$

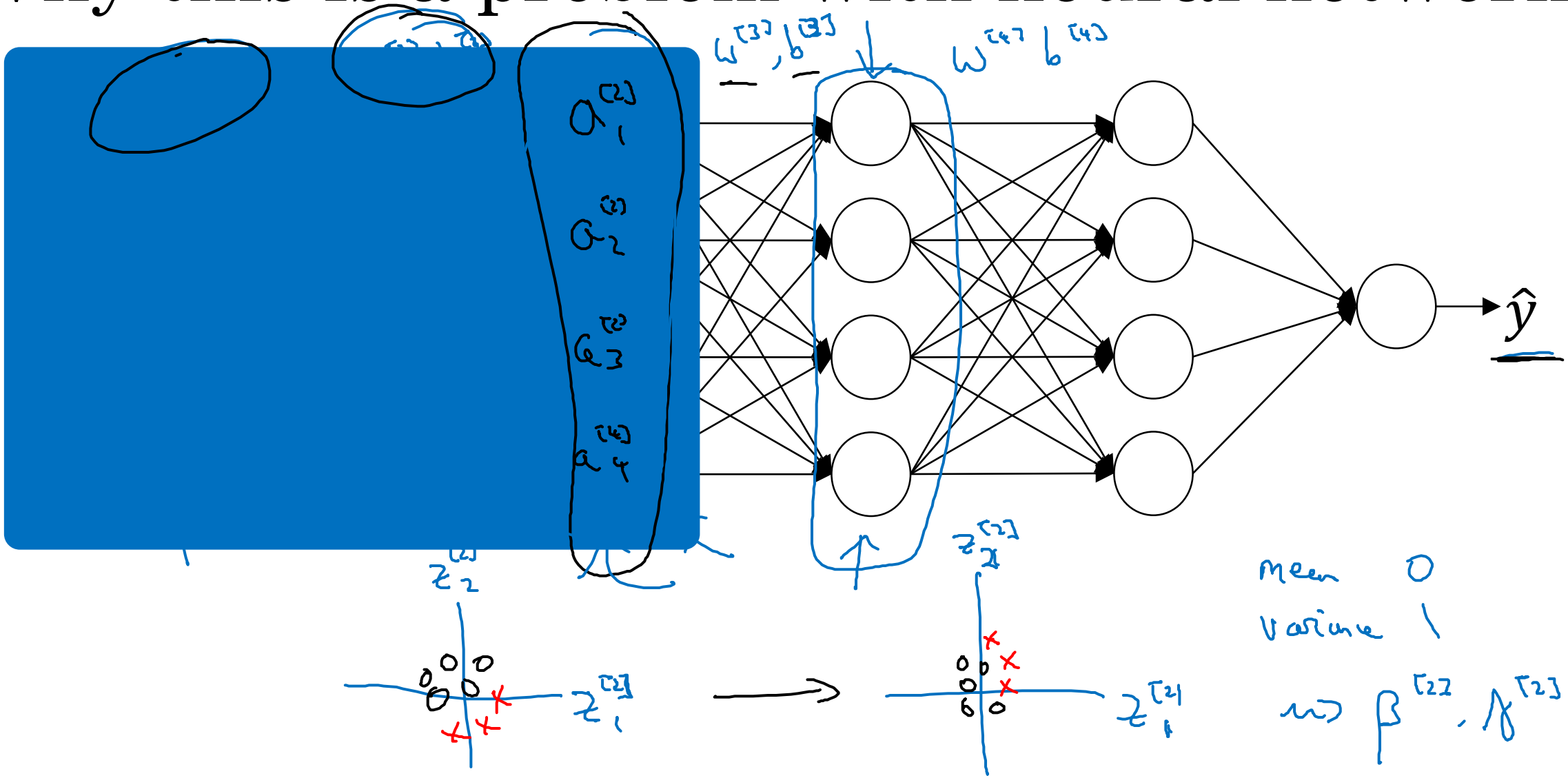



$y = 0$




Andrew Ng

# Why this is a problem with neural networks?



deeplearning.ai

# Batch Normalization

# Batch Norm at test time

# Batch Norm at test time

$$\mu = \frac{1}{m}\sum_i z^{(i)}$$

$$\sigma^2 = \frac{1}{m}\sum_i (z^{(i)} - \mu)^2$$

$$z^{(i)}_{\text{norm}} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

$$\tilde{z}^{(i)} = \gamma z^{(i)}_{\text{norm}} + \beta$$

$\mu$, $\sigma^2$: estimate using exponentially weighted average (across mini-batches).

$X^{\{1\}}, X^{\{1\}}, X^{\{3\}}, \ldots$

$\mu^{\{1\}[l]}$ $\mu^{\{2\}[l]}$ $\mu^{\{3\}[l]}$ $\rightarrow$ $\mu$

$\theta_1$ $\theta_2$ $\theta_3$ $\sigma^2$

$\sigma^{2\{1\}[l]}$ $\sigma^{2\{2\}[l]}$ ....

$z_{\text{norm}} = \frac{z - \mu}{\sqrt{\sigma^2 + \varepsilon}}$ $\quad \tilde{z} = \gamma z_{\text{norm}} + \beta$

Andrew Ng

deeplearning.ai

# Multi-class classification

# Softmax regression

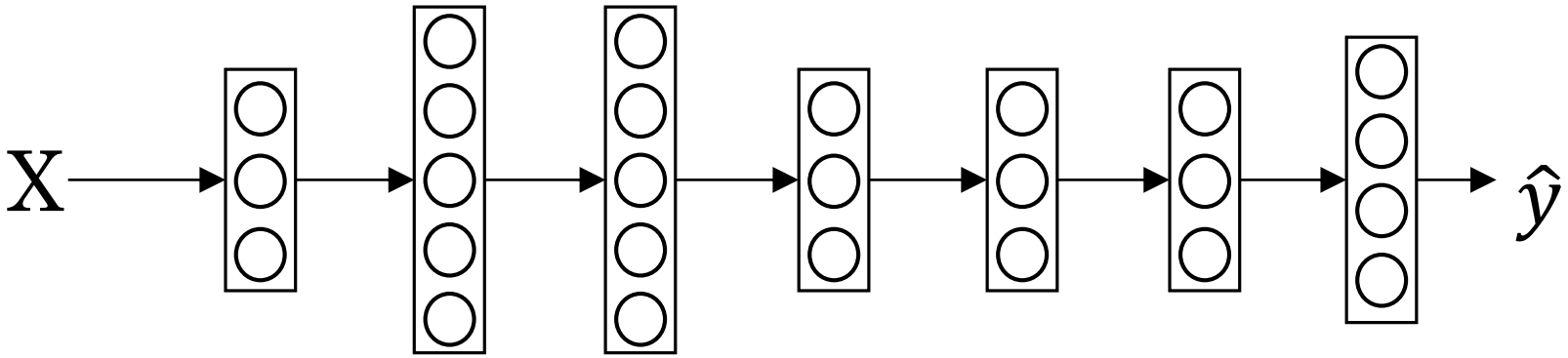
# Recognizing cats, dogs, and baby chicks





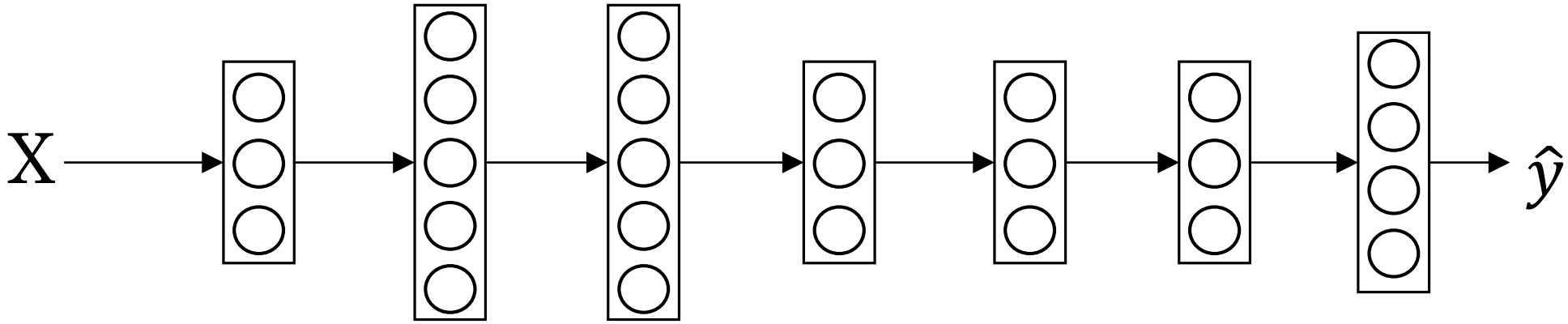



3 1 2 0 3 2 0 1




Andrew Ng

# Softmax layer



deeplearning.ai

# Programming Frameworks

---

# Deep Learning frameworks

# Deep learning frameworks

- Caffe/Caffe2
- CNTK
- DL4J
- Keras
- Lasagne
- mxnet
- PaddlePaddle
- TensorFlow
- Theano
- Torch

Choosing deep learning frameworks

- Ease of programming (development and deployment)
- Running speed
- Truly open (open source with good governance)

deeplearning.ai

# Programming Frameworks

---

# TensorFlow

# Motivating problem

$J(w) = w^2 - 10w + 25$

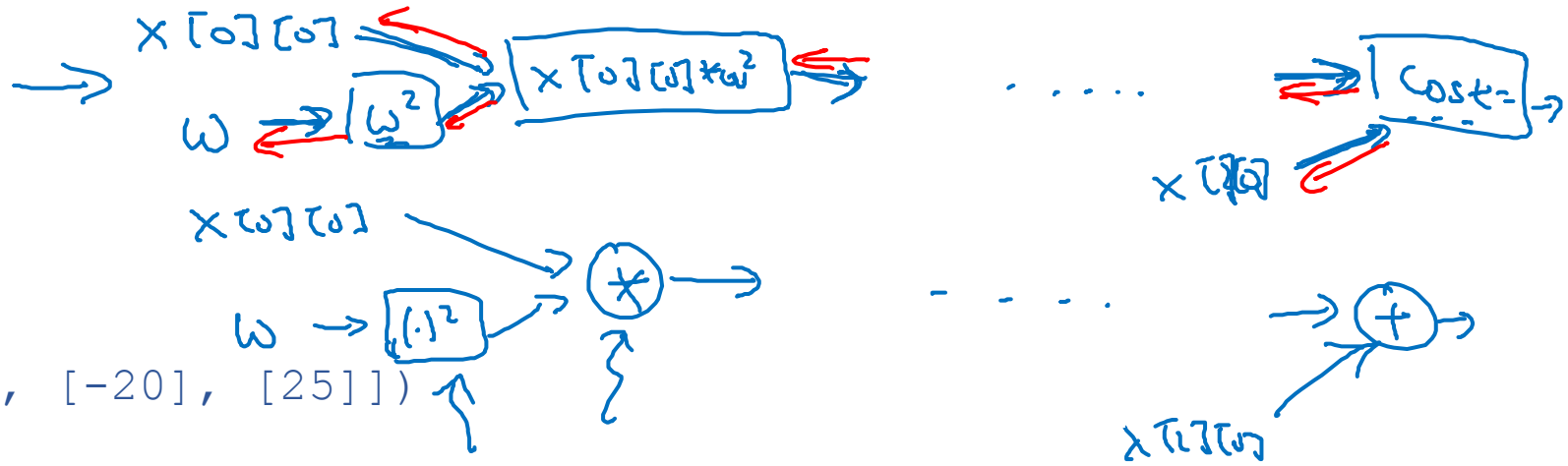
(cost)

$(w-5)^2$

$w = 5$

$J(w, b)$

# Code example



```
import numpy as np
import tensorflow as tf

coefficients = np.array([[1], [-20], [25]])

w = tf.Variable([0],dtype=tf.float32)
x = tf.placeholder(tf.float32, [3,1])
cost = x[0][0]*w**2 + x[1][0]*w + x[2][0]     # (w-5)**2
train = tf.train.GradientDescentOptimizer(0.01).minimize(cost)
init = tf.global_variables_initializer()
session = tf.Session()
session.run(init)
print(session.run(w))

for i in range(1000):
    session.run(train, feed_dict={x:coefficients})
print(session.run(w))
```

```
with tf.Session() as session:
    session.run(init)
    print(session.run(w))
```

Andrew Ng