

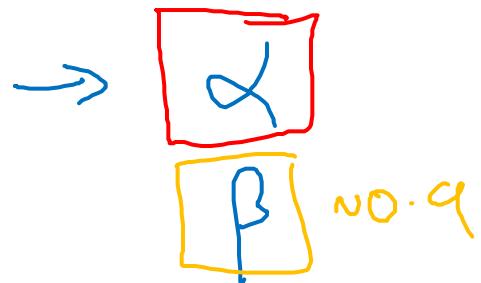


deeplearning.ai

Hyperparameter tuning

Tuning process

Hyperparameters



$\beta_1, \beta_2, \epsilon$
 $0.9, 0.99, 10^{-8}$

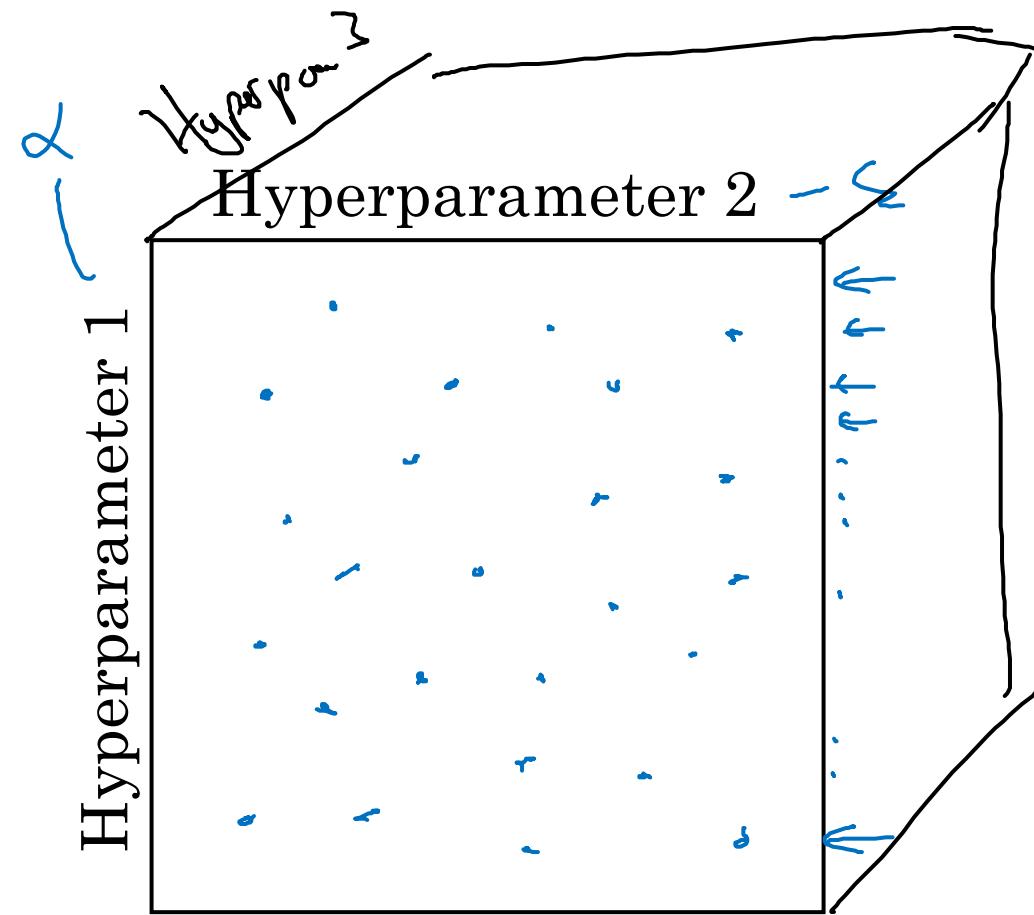
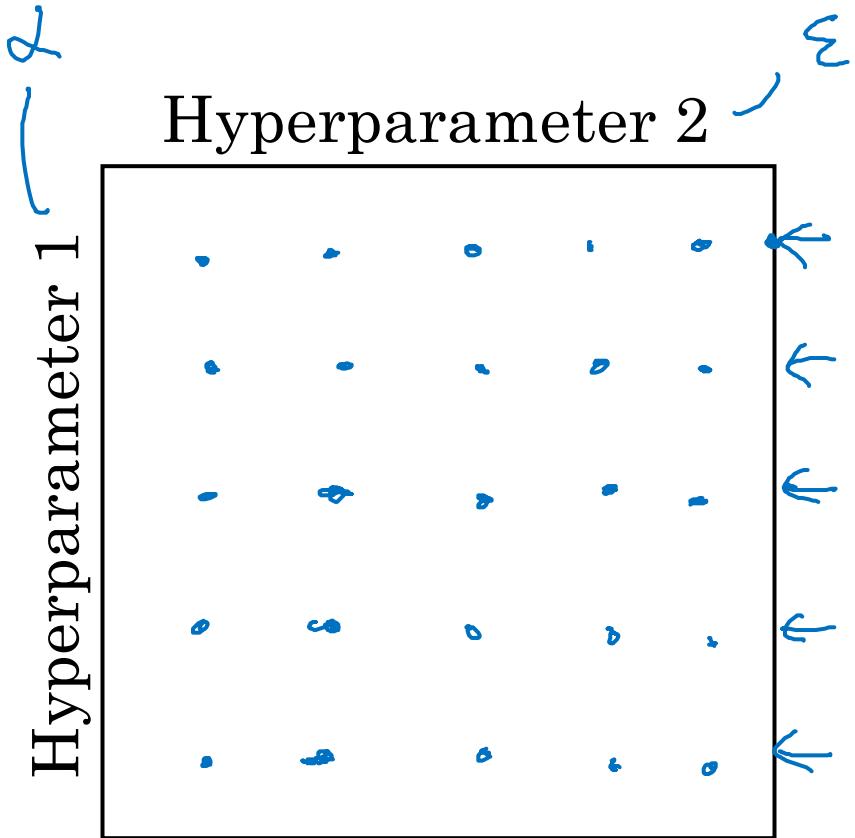
layers

hidden units

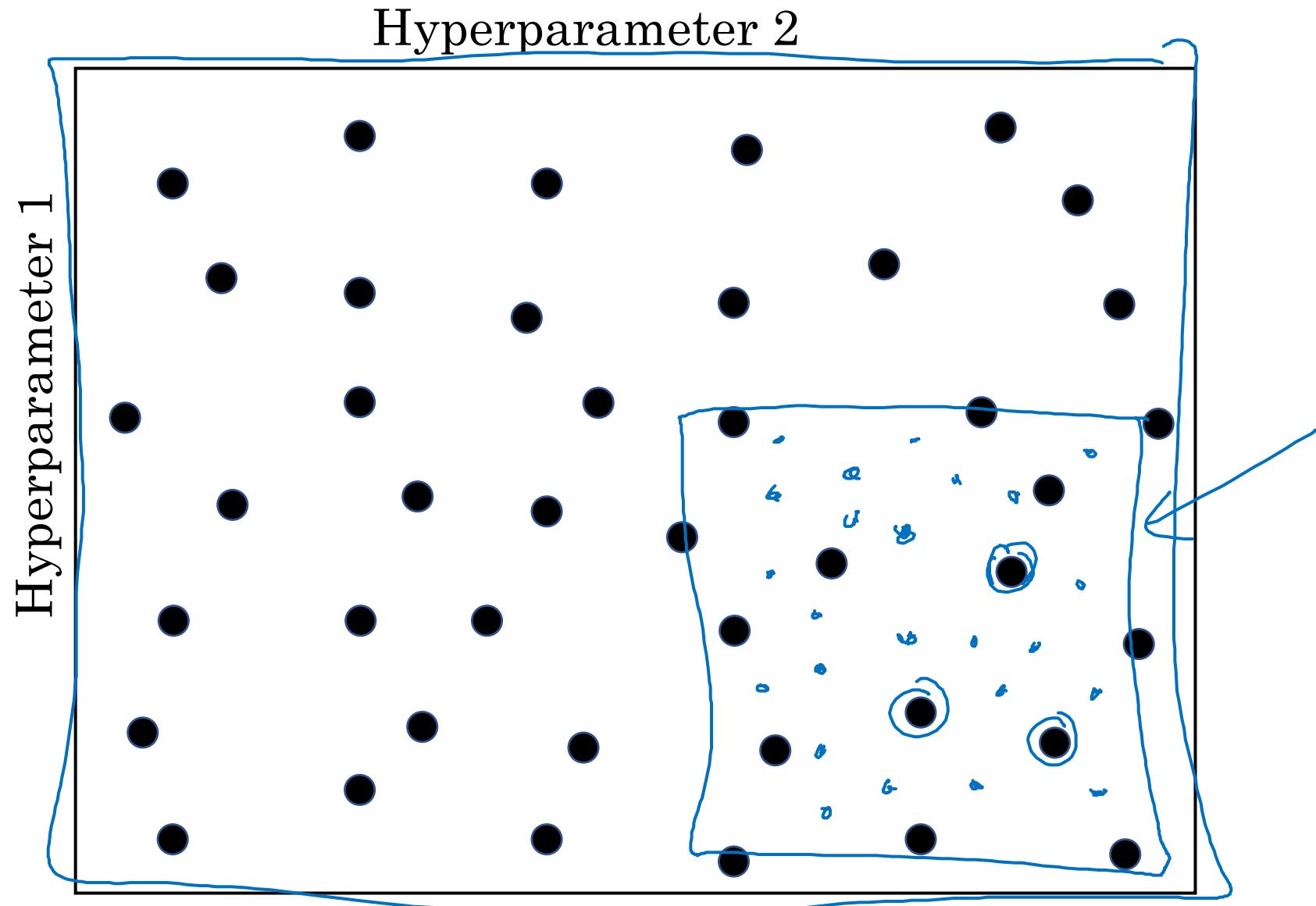
learning rate decay

mini-batch size

Try random values: Don't use a grid



Coarse to fine





deeplearning.ai

Hyperparameter tuning

Using an appropriate
scale to pick
hyperparameters

Picking hyperparameters at random

→ $n^{[l]} = 50, \dots, 100$

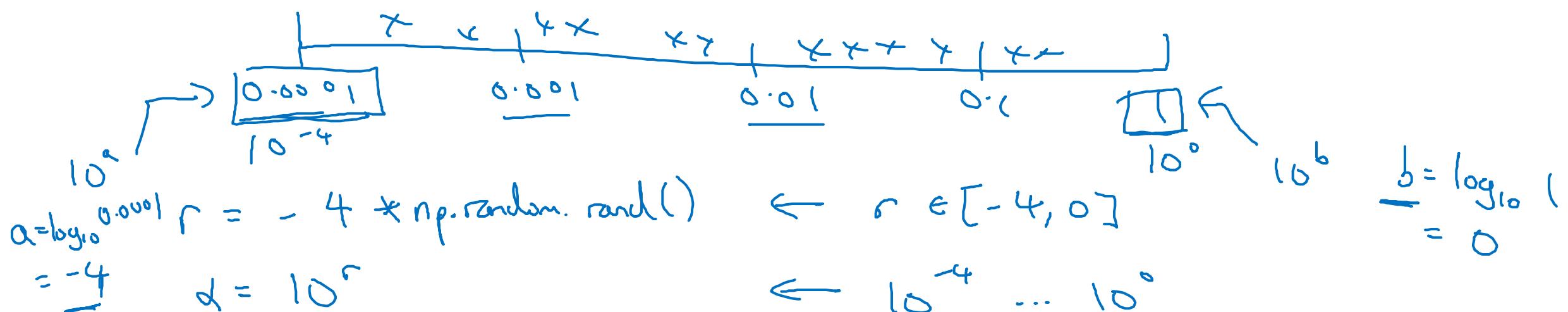
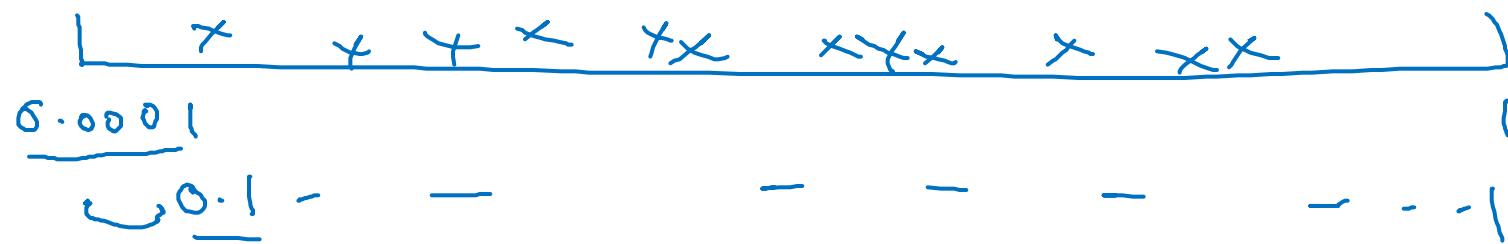


→ #layers L : 2 - 4

2, 3, 4

Appropriate scale for hyperparameters

$$\lambda = 0.0001, \dots, 1$$



$$10^a \dots 10^b$$

$$\frac{r \in [a, b]}{[-4, 0]}$$

$$\lambda = 10^r$$

Hyperparameters for exponentially weighted averages

$$\beta = 0.9 \dots 0.999$$

\downarrow \downarrow
 10 1000

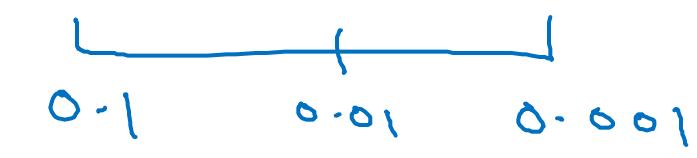
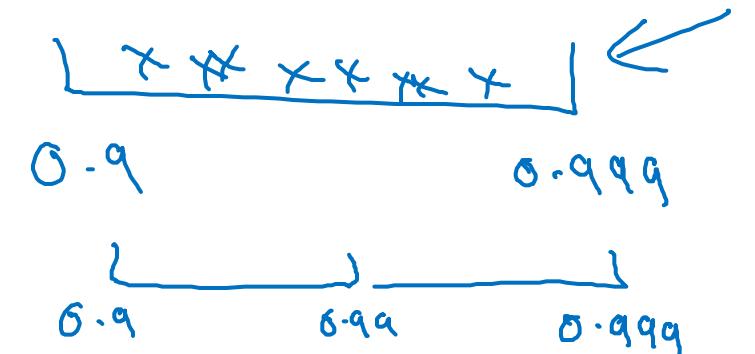
$$1-\beta = 0.1 \dots 0.001$$

$$\beta: 0.900 \rightarrow 0.9005 \quad \} \sim 10$$

$$\beta: 0.999 \rightarrow 0.9995$$

~ 1000 ~ 2000

$$\frac{1}{1-\beta}$$



$$\frac{10^{-1}}{10^{-3}}$$

$r \in [-3, -1]$

$$1-\beta = 10^r$$

$$\beta = 1 - 10^r$$

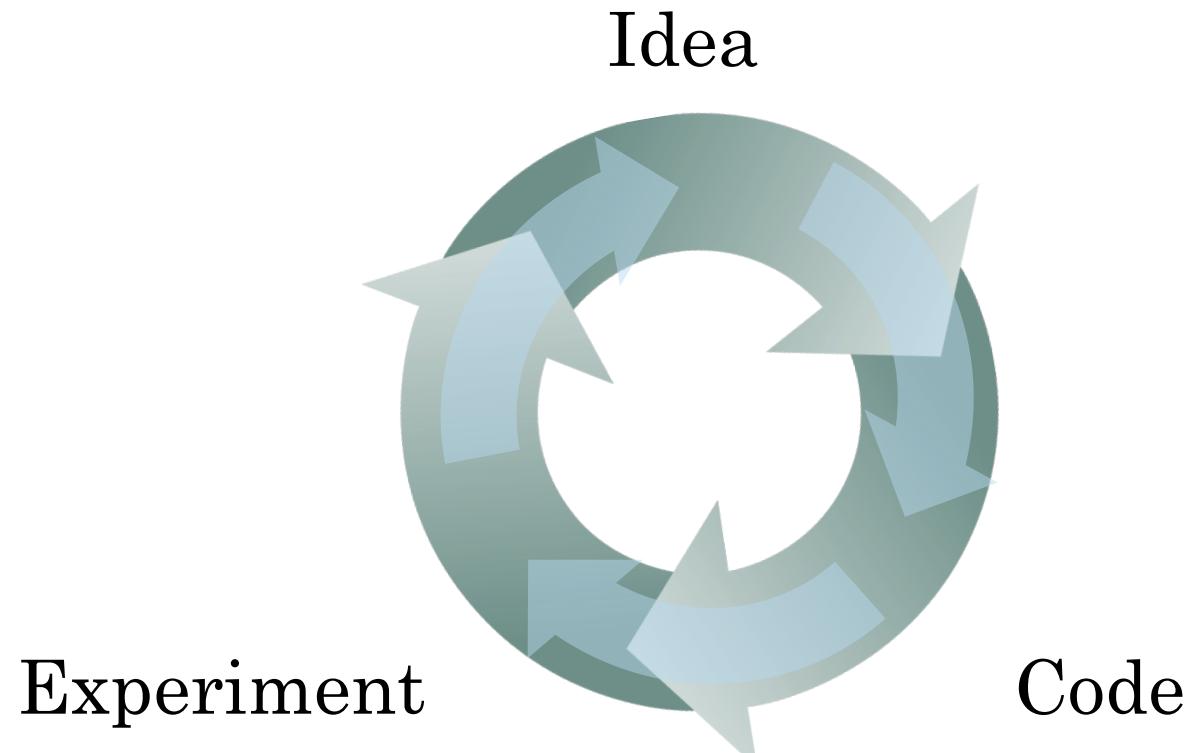


deeplearning.ai

Hyperparameters tuning

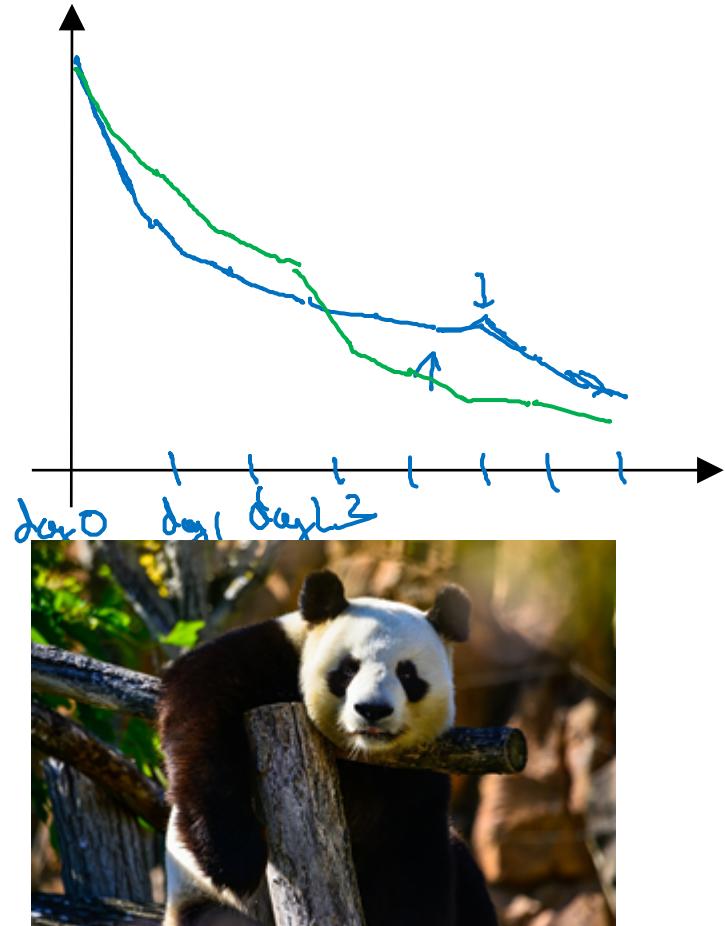
Hyperparameters tuning in practice: Pandas vs. Caviar

Re-test hyperparameters occasionally



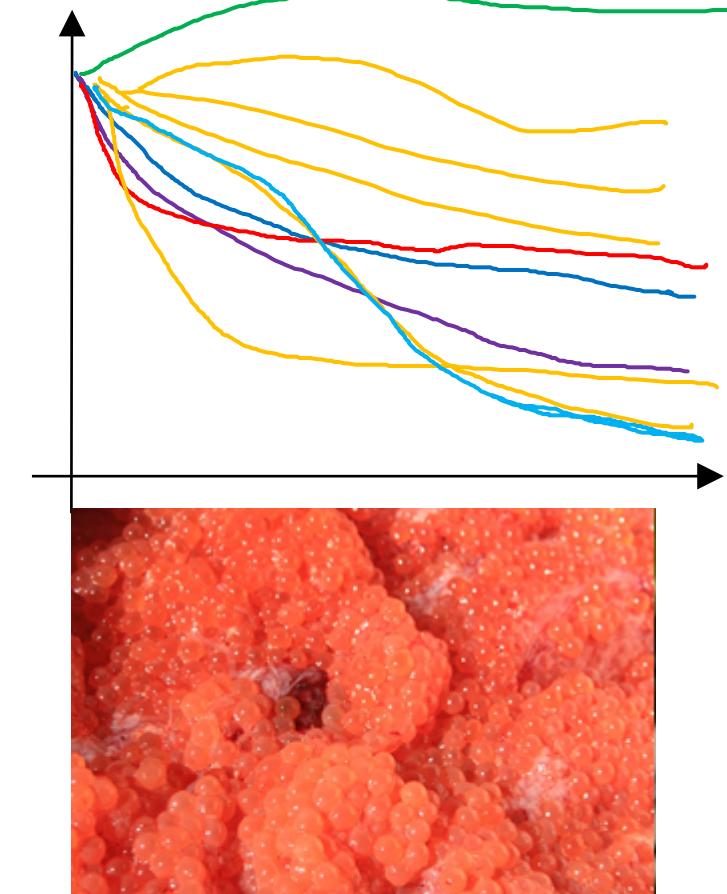
- NLP, Vision, Speech,
Ads, logistics,
- Intuitions do get stale.
Re-evaluate occasionally.

Babysitting one model



Panda

Training many models in parallel



Caviar

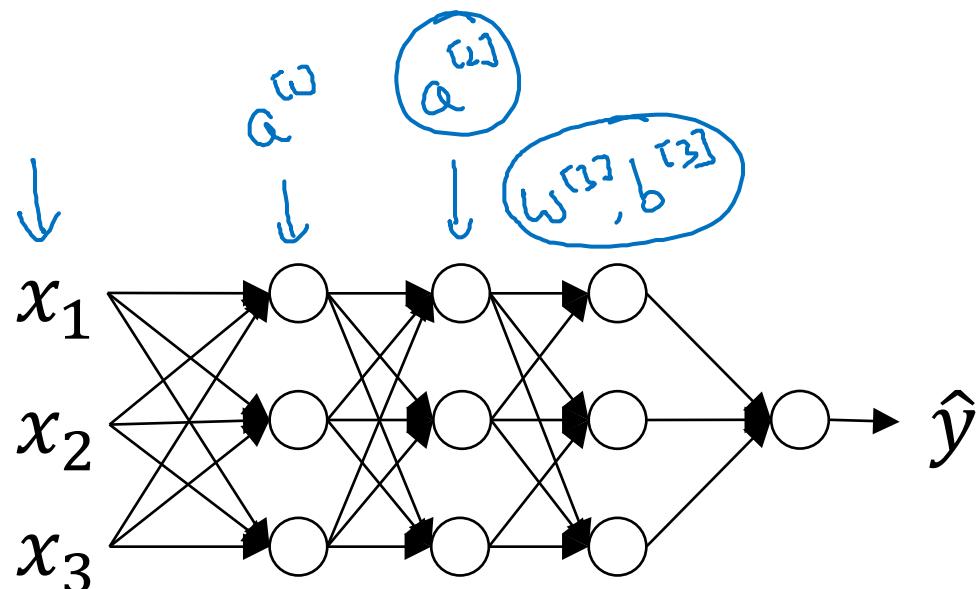
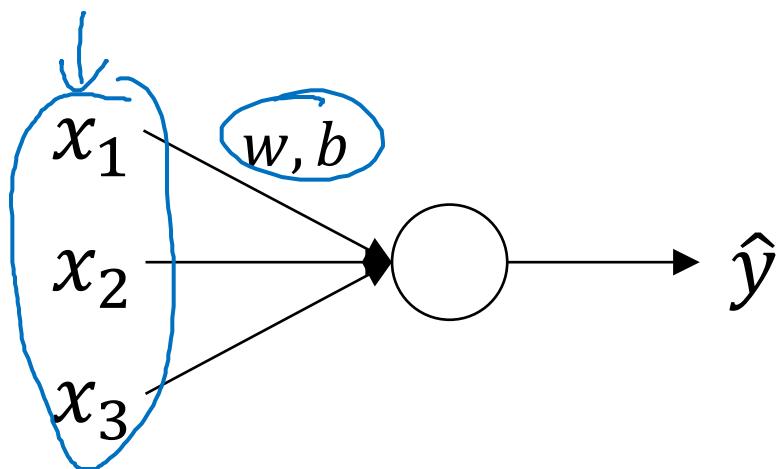


deeplearning.ai

Batch Normalization

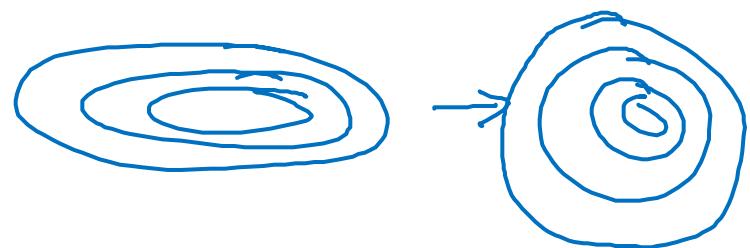
Normalizing activations in a network

Normalizing inputs to speed up learning



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

electr-wiss



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

Normalize $\frac{z^{[2]}}{\uparrow}$

Implementing Batch Norm

Given some intermediate values in NN

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

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

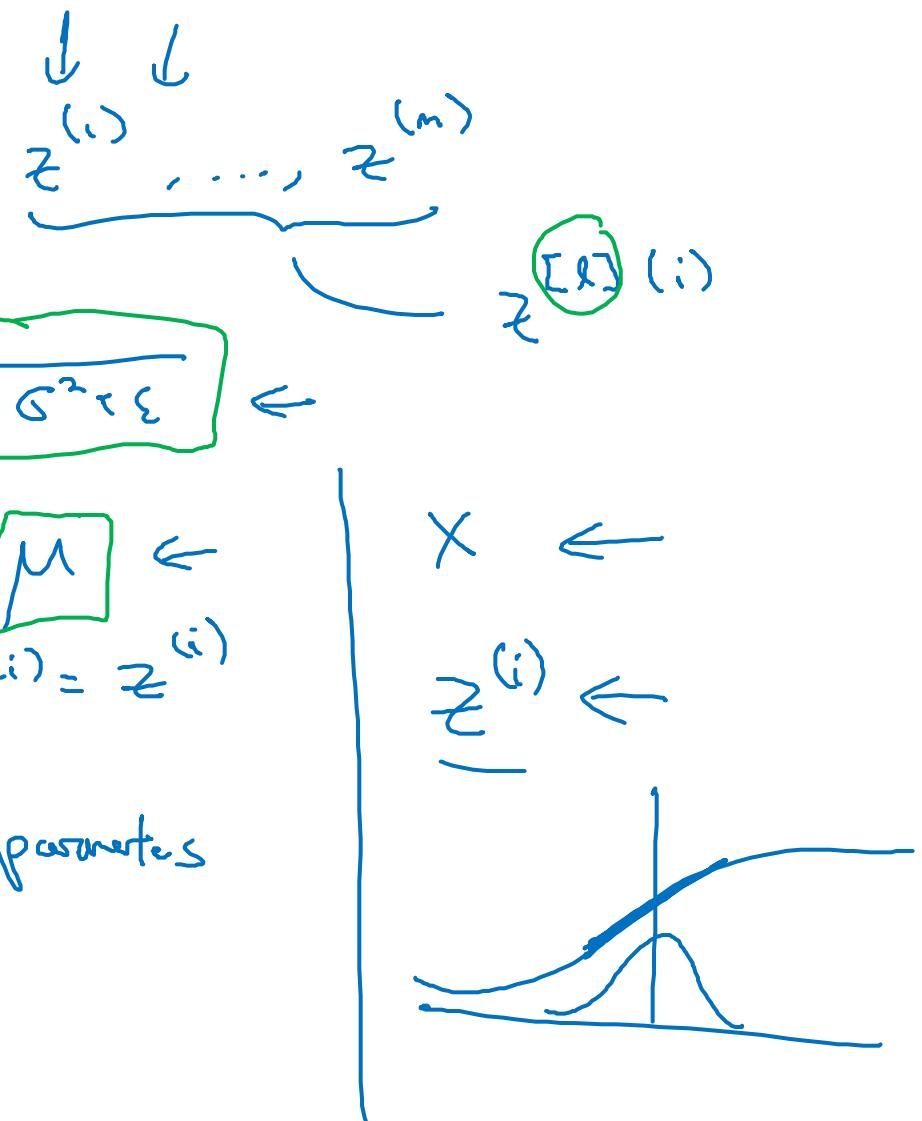
$$z_{\text{norm}}^{(i)} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

$$\hat{z}^{(i)} = \gamma z_{\text{norm}}^{(i)} + \beta$$

Use $\hat{z}^{(i)}$ instead of $z^{(i)}$.

If $\gamma = \sqrt{\sigma^2 + \epsilon}$ ←
then $\beta = \mu$ ←
 $\hat{z}^{(i)} = z^{(i)}$

learnable parameters
of model.



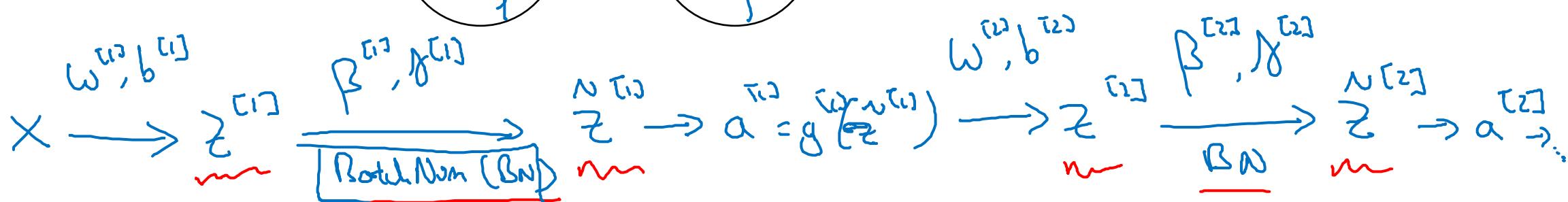
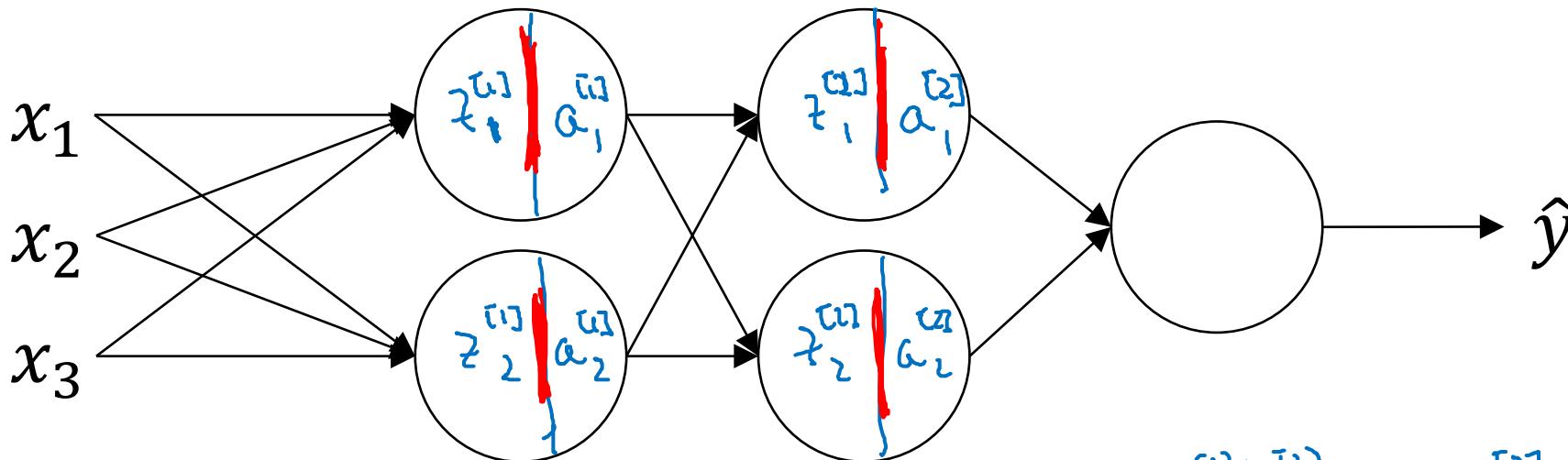


deeplearning.ai

Batch Normalization

Fitting Batch Norm
into a neural network

Adding Batch Norm to a network



Parameters:

$$\left\{ w^{[1]}, b^{[1]}, w^{[2]}, b^{[2]}, \dots, w^{[L]}, b^{[L]}, \right.$$

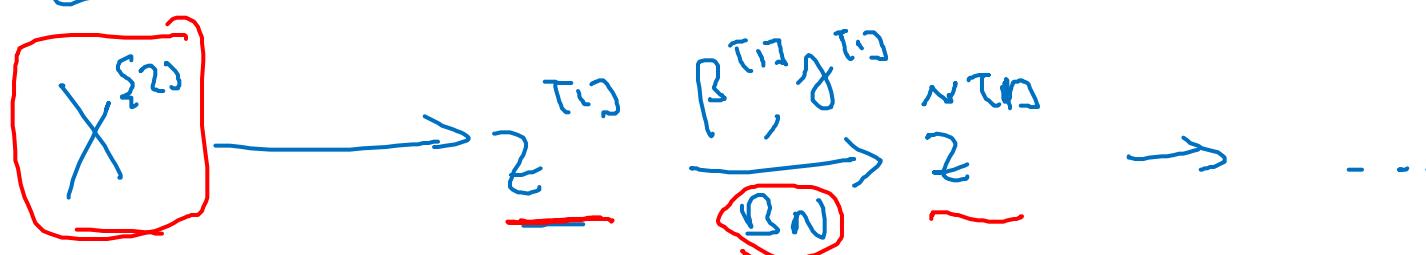
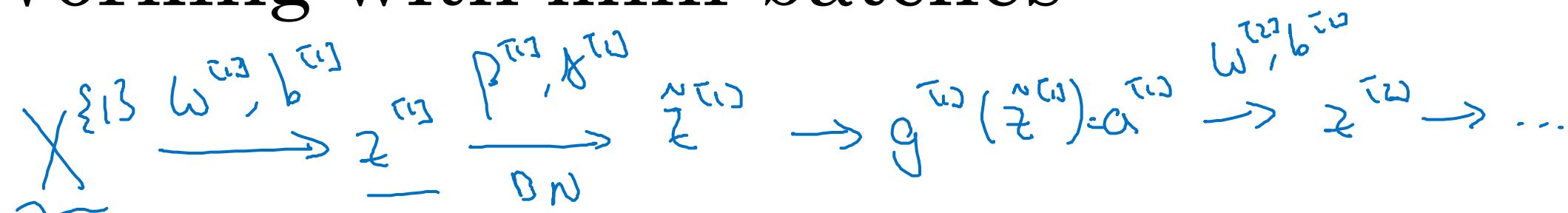
$$\left. \rightarrow \beta^{[1]}, \gamma^{[1]}, \beta^{[2]}, \gamma^{[2]}, \dots, \beta^{[L]}, \gamma^{[L]} \right\}$$

$$\rightarrow \beta$$

$$\beta = \bar{\beta} - d \delta \beta^{[L]}$$

`tf.nn.batch_normalization` ←

Working with mini-batches



X^{S2}

...
...

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

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

$$\begin{matrix} | & | & | \\ (n^{[l]}, 1) & (n^{[l]}, 1) & (n^{[l]}, 1) \end{matrix}$$

$$\rightarrow \underline{z}^{[l]} = w^{[l]} \underline{a}^{[l-1]} + \cancel{b}^{[l]}$$
$$\underline{z}^{[l]} = w^{[l]} \underline{a}^{[l-1]}$$
$$\underline{z}^{[l]}_{norm}$$
$$\rightarrow \underline{\tilde{z}}^{[l]} = \gamma^{[l]} \underline{z}^{[l]}_{norm} + \beta^{[l]}$$

Andrew Ng

Implementing gradient descent

for $t = 1 \dots \text{num MiniBatches}$
Compute forward prop on $X^{[t]}$.

In each hidden layer, use BN to replace $\underline{z}^{[l]}$ with $\hat{\underline{z}}^{[l]}$.

Use backprop to compute $\underline{dw}^{[l]}$, ~~$\underline{db}^{[l]}$~~ , $\underline{d\beta}^{[l]}$, $\underline{dg}^{[l]}$

Update parameters $\left. \begin{array}{l} w^{[l]} := w^{[l]} - \alpha \underline{dw}^{[l]} \\ \beta^{[l]} := \beta^{[l]} - \alpha \underline{d\beta}^{[l]} \\ g^{[l]} := \dots \end{array} \right\} \leftarrow$

Works w/ momentum, RMSprop, Adam.

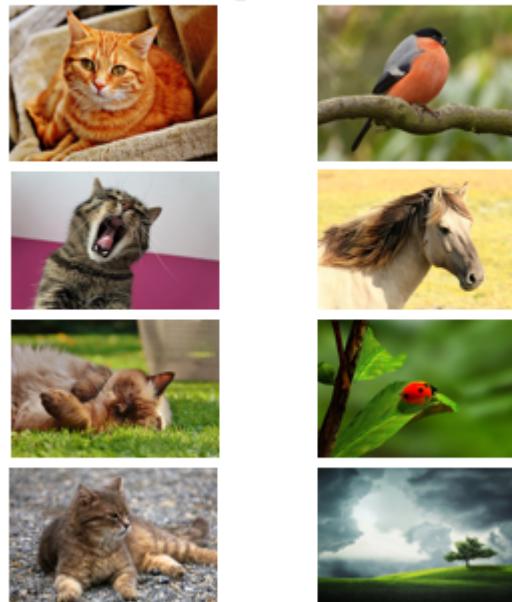
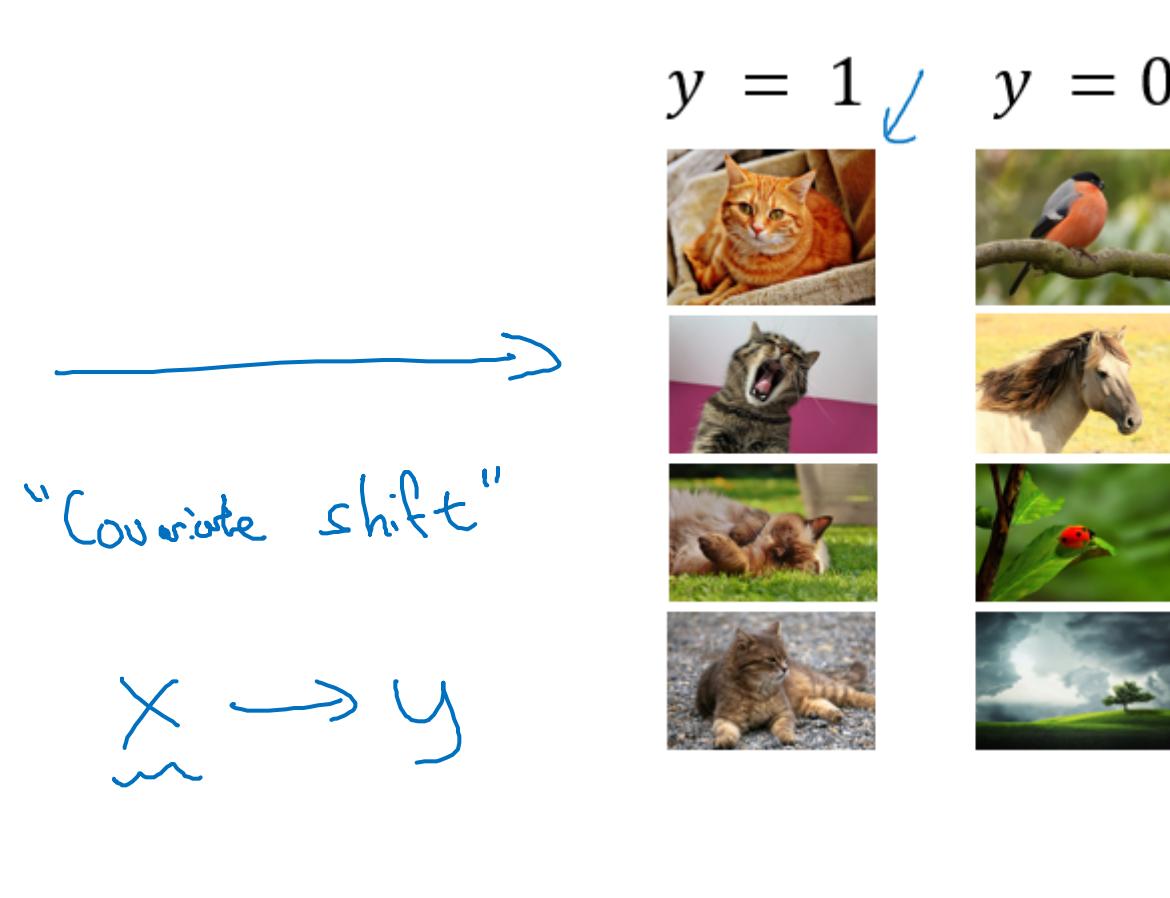
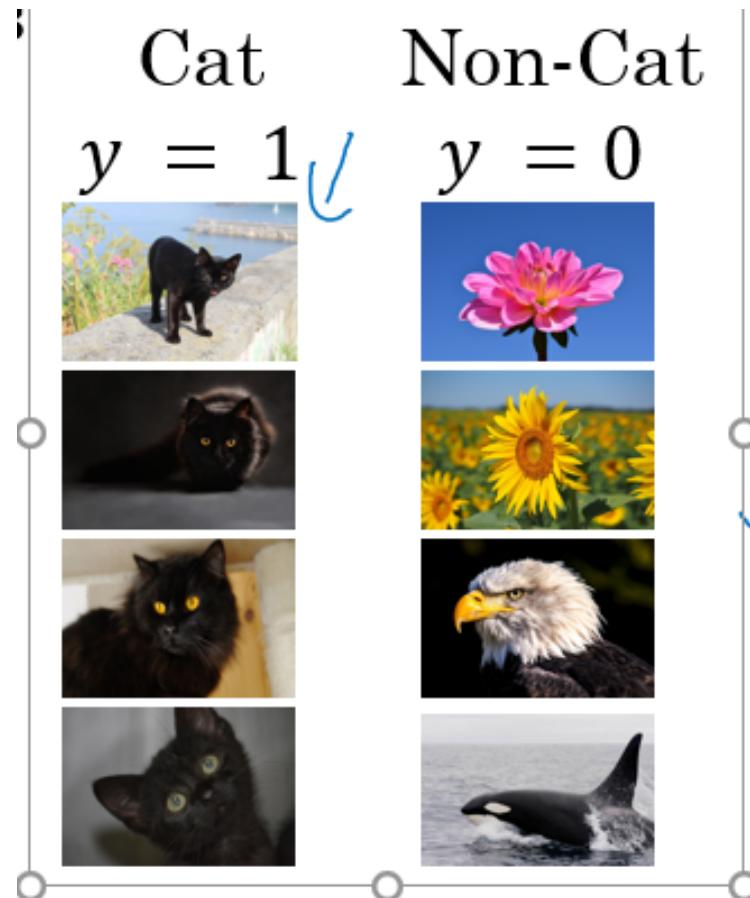
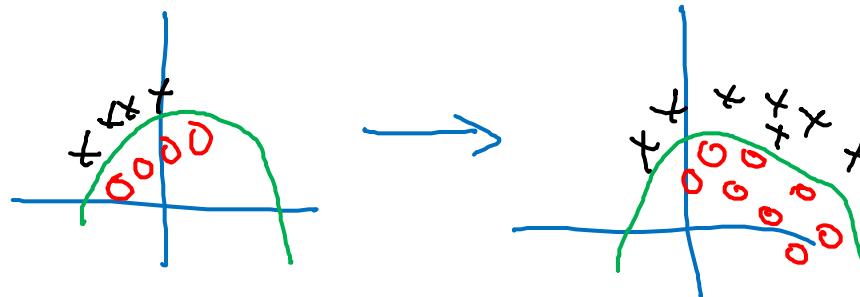
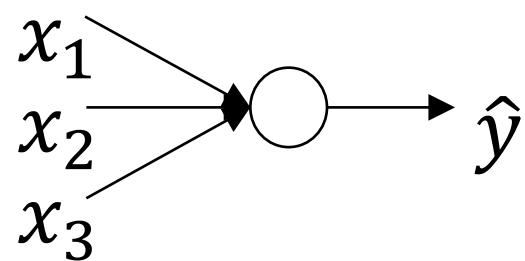


deeplearning.ai

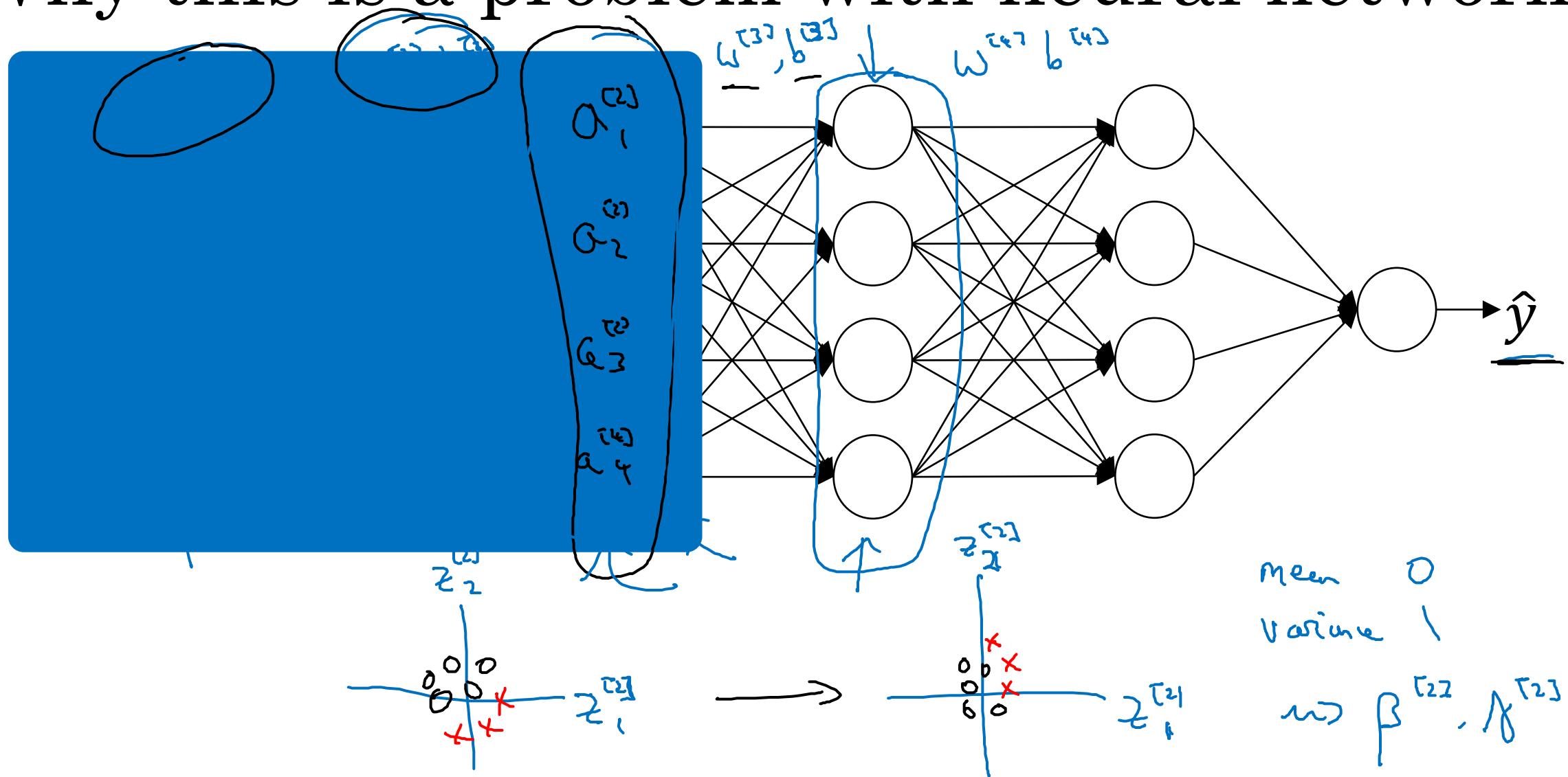
Batch Normalization

Why does
Batch Norm work?

Learning on shifting input distribution



Why this is a problem with neural networks?



Batch Norm as regularization

X

- Each mini-batch is scaled by the mean/variance computed on just that mini-batch.
 $\xrightarrow{\hat{z}^{[l]}}$ μ, σ^2 $\{z^{[l]}\}$
 $\hat{z}^{[l]}$ $\underline{64}, \underline{128}$
- This adds some noise to the values $z^{[l]}$ within that minibatch. So similar to dropout, it adds some noise to each hidden layer's activations.
 μ, σ^2
- This has a slight regularization effect.

mini-batch : 64 \longrightarrow 512



deeplearning.ai

Batch Normalization

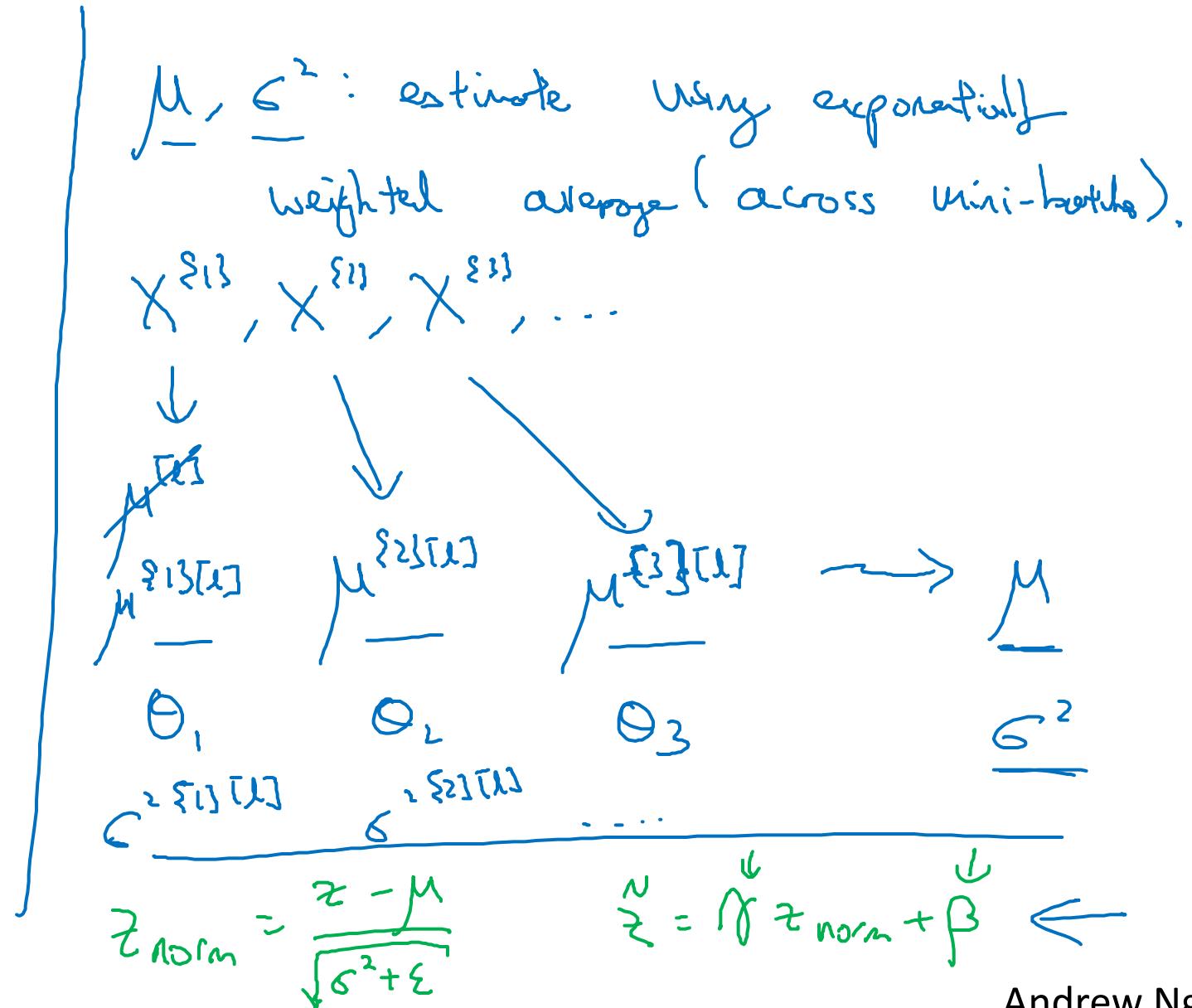
Batch Norm at test time

Batch Norm at test time

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

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

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





deeplearning.ai

Batch Normalization

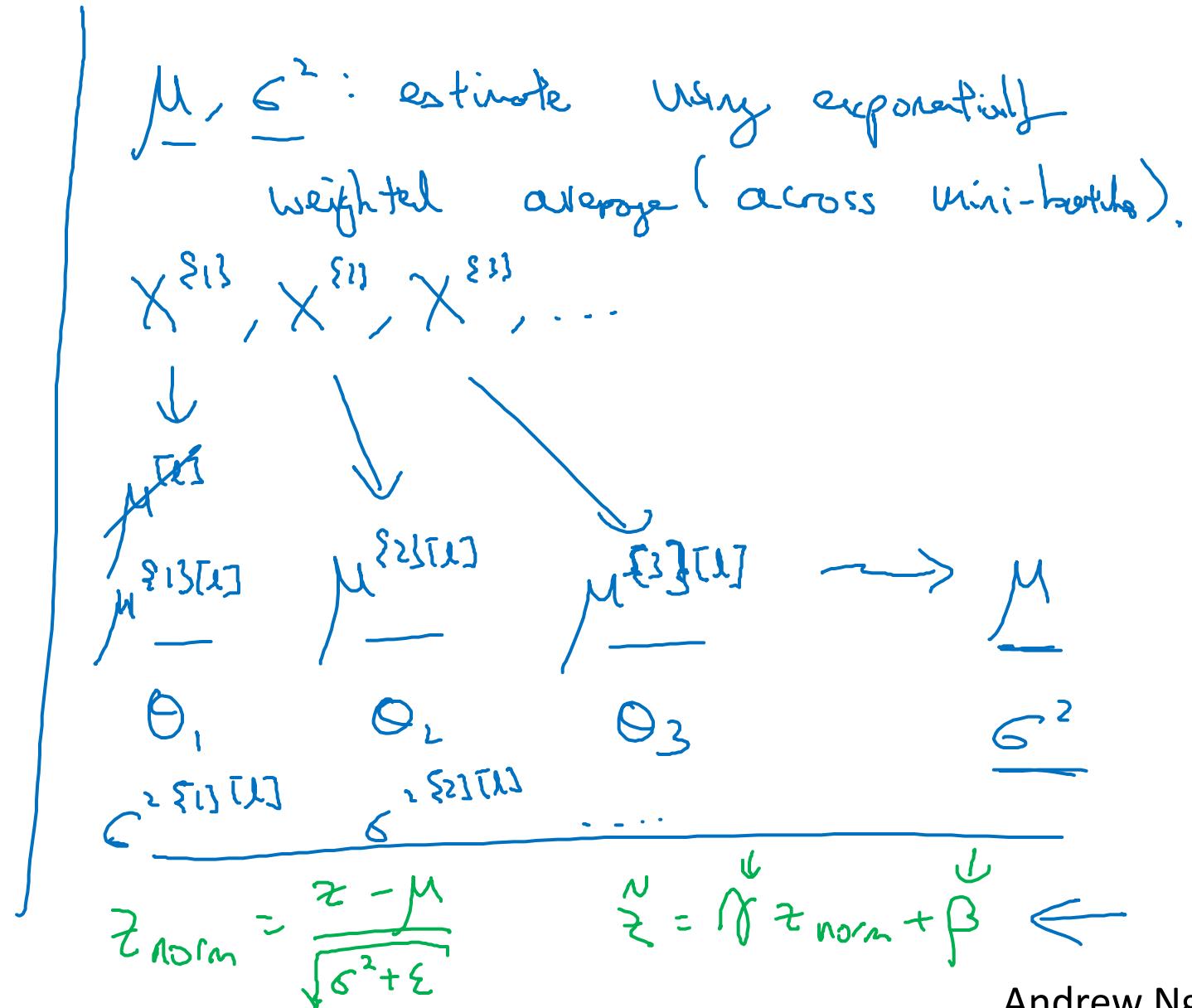
Batch Norm at test time

Batch Norm at test time

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

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

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





deeplearning.ai

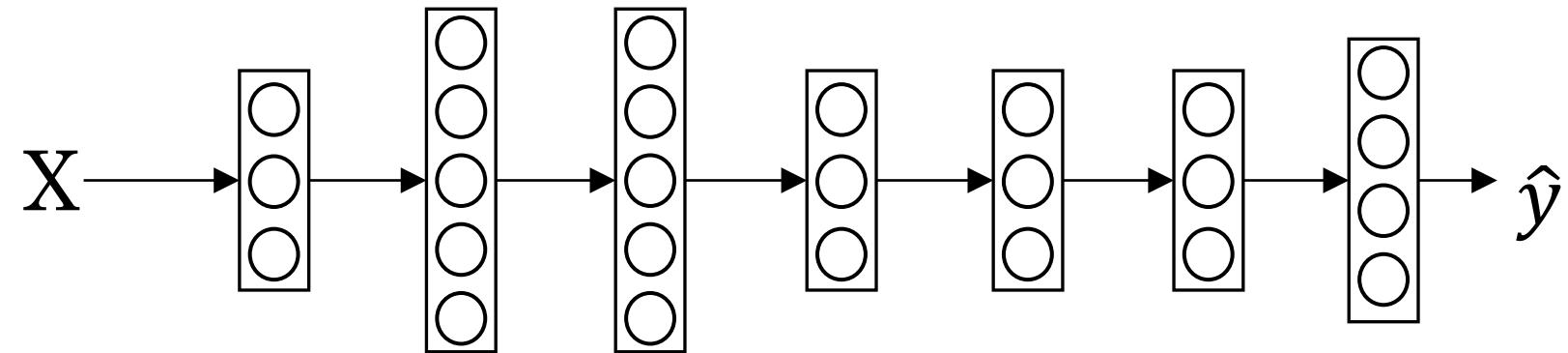
Multi-class
classification

Softmax regression

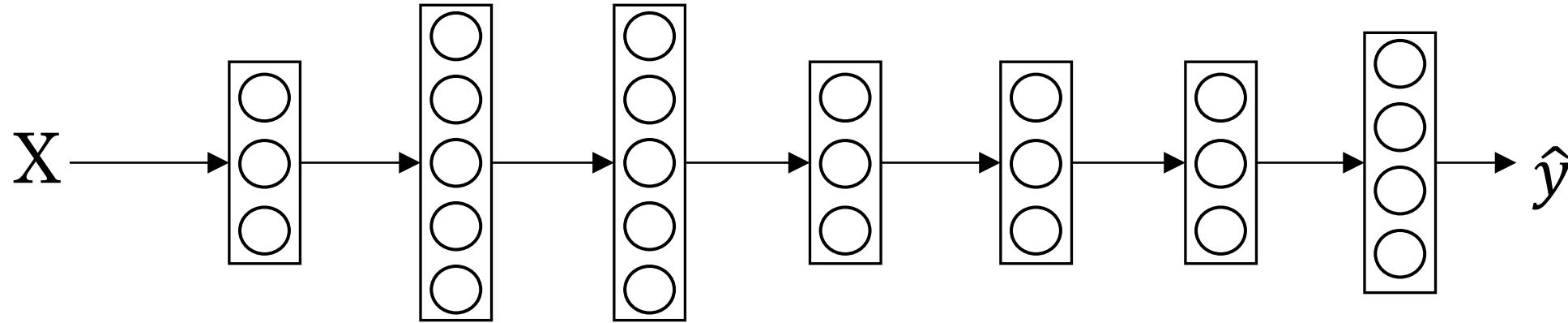
Recognizing cats, dogs, and baby chicks



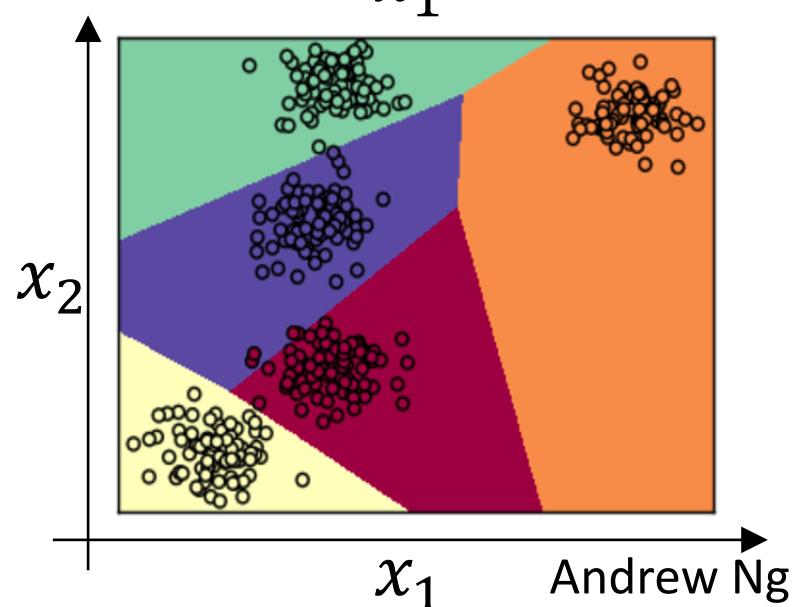
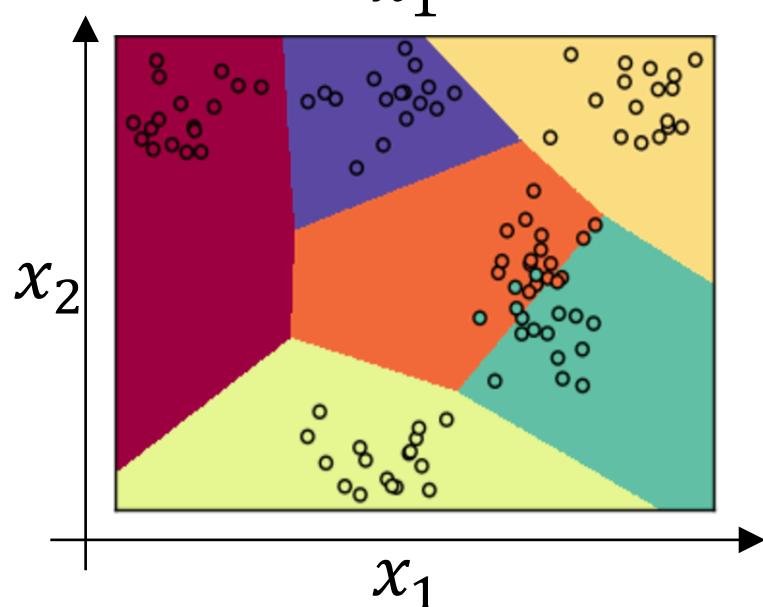
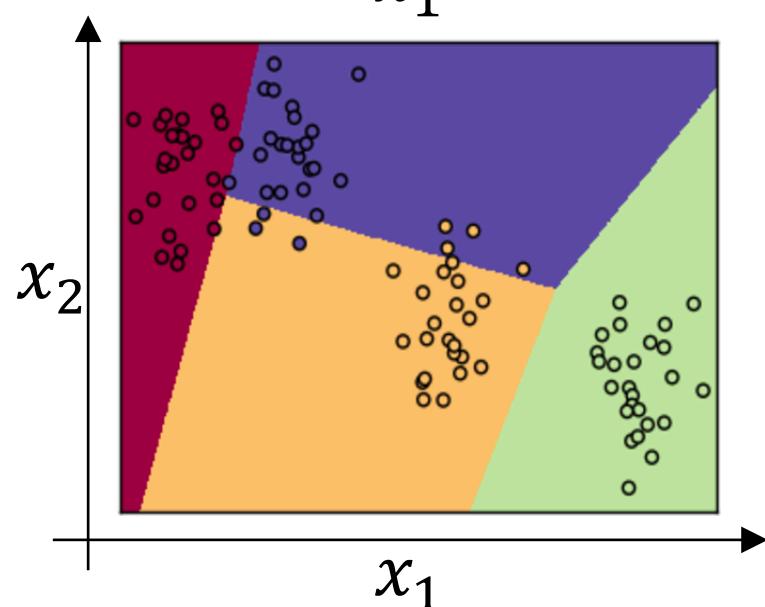
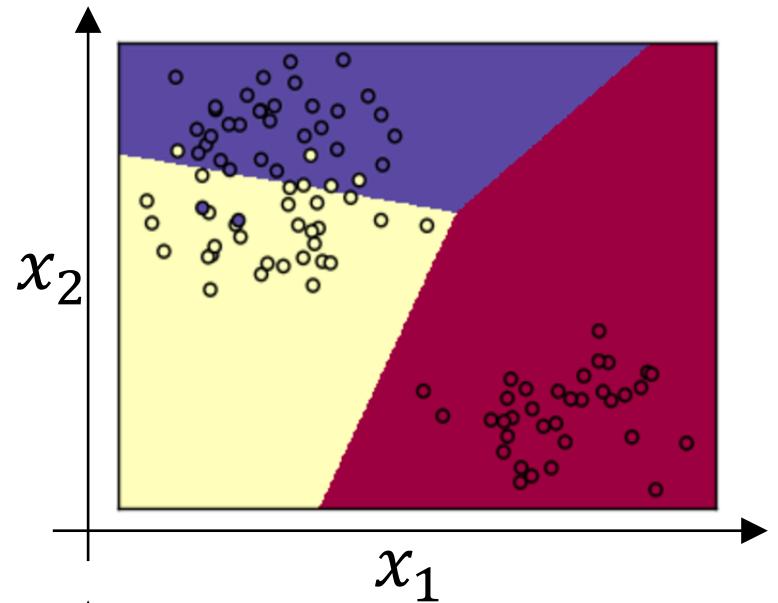
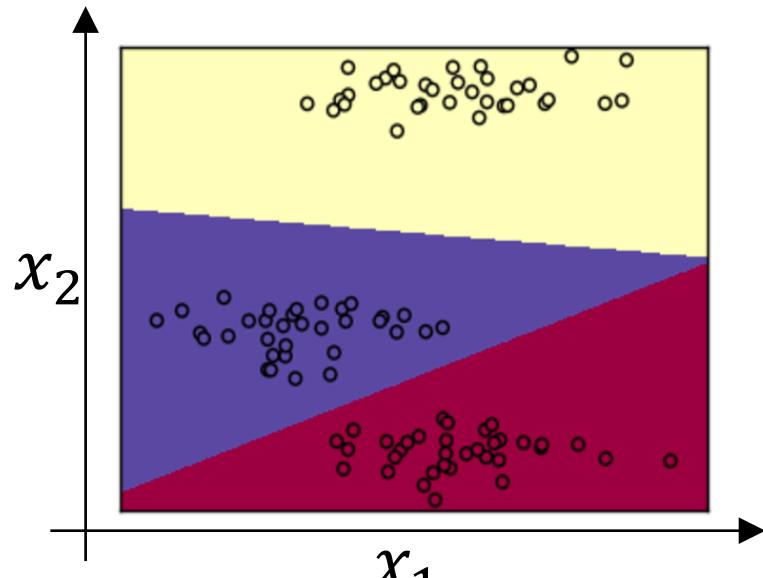
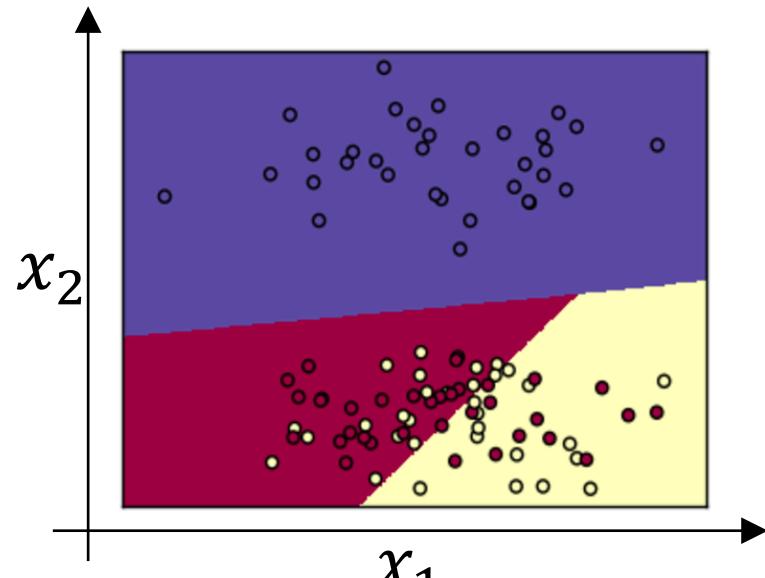
3 1 2 0 3 2 0 1



Softmax layer



Softmax examples





deeplearning.ai

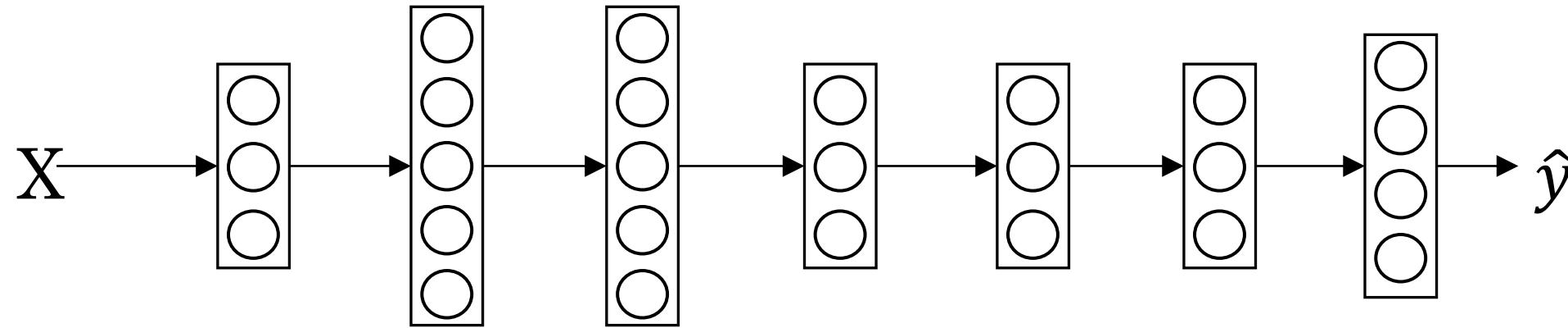
Multi-class
classification

Trying a softmax
classifier

Understanding softmax

Loss function

Summary of softmax classifier





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

$$J(\omega) = \frac{(\omega^2 - 10\omega + 25)}{(\omega - 5)^2}$$

$\omega = 5$

$J(\omega, b)$

↑ ↑

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()
```

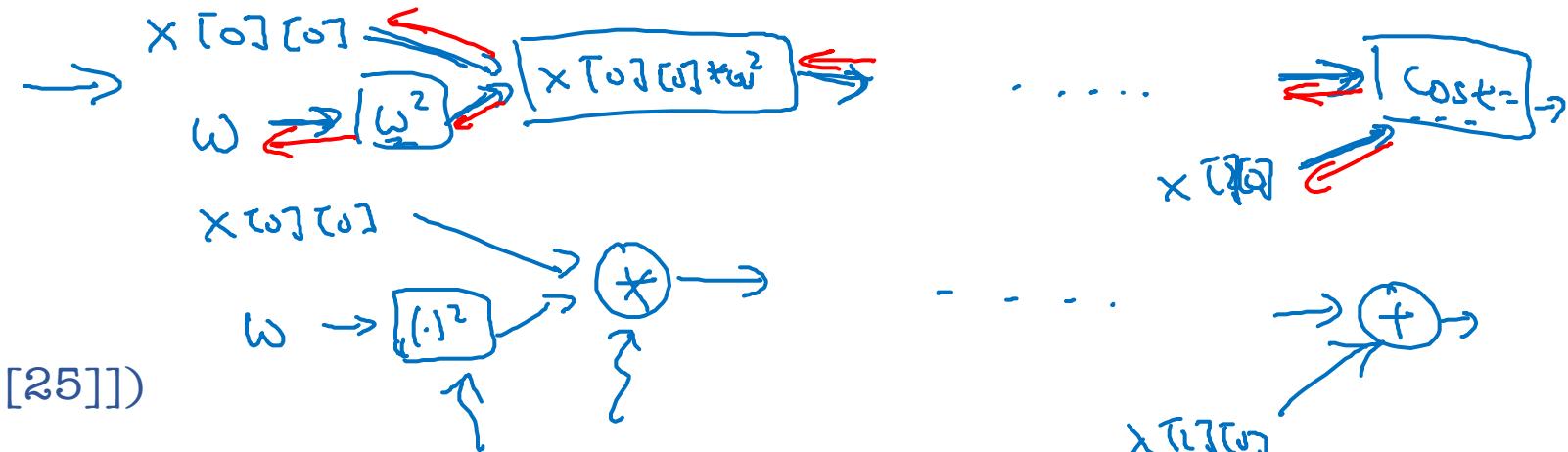
```
session.run(init)
```

```
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))
```