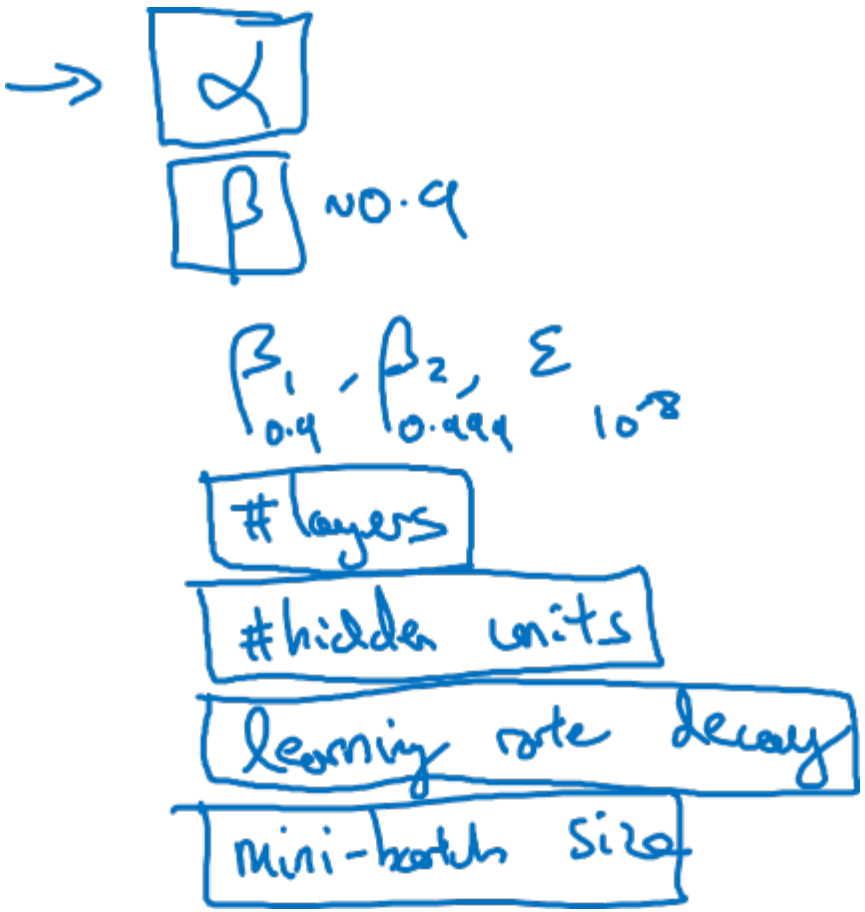
deeplearning.ai

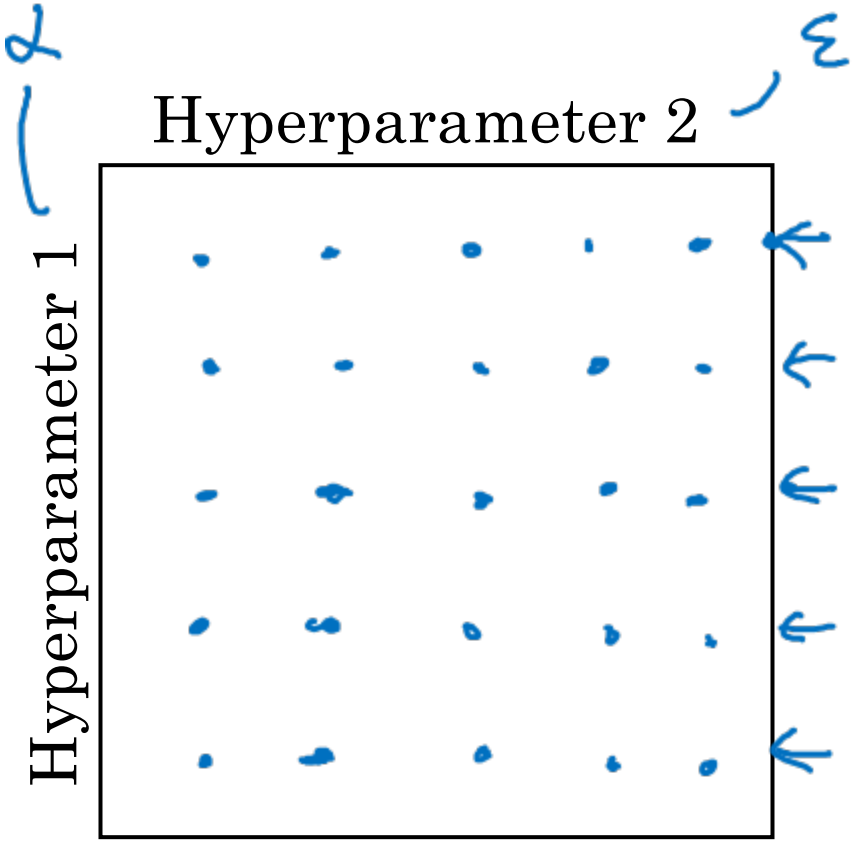
Hyperparameter tuning

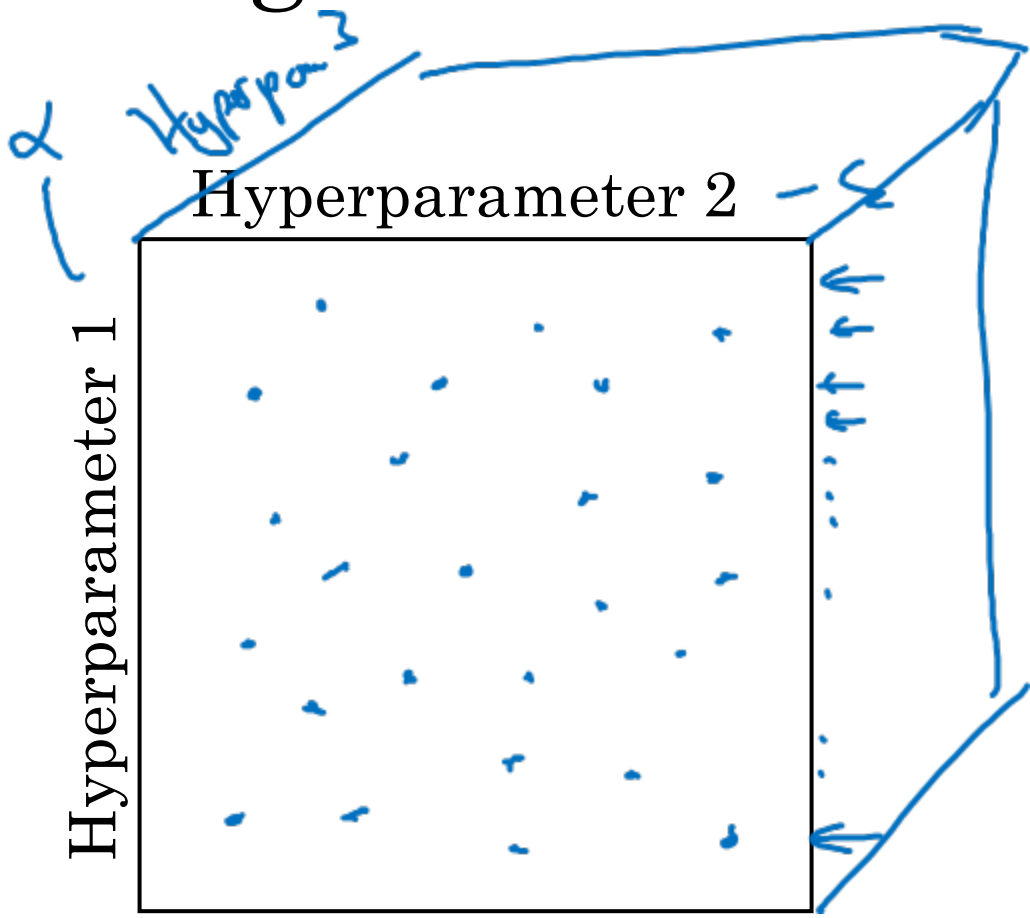
Tuning process

Hyperparameters

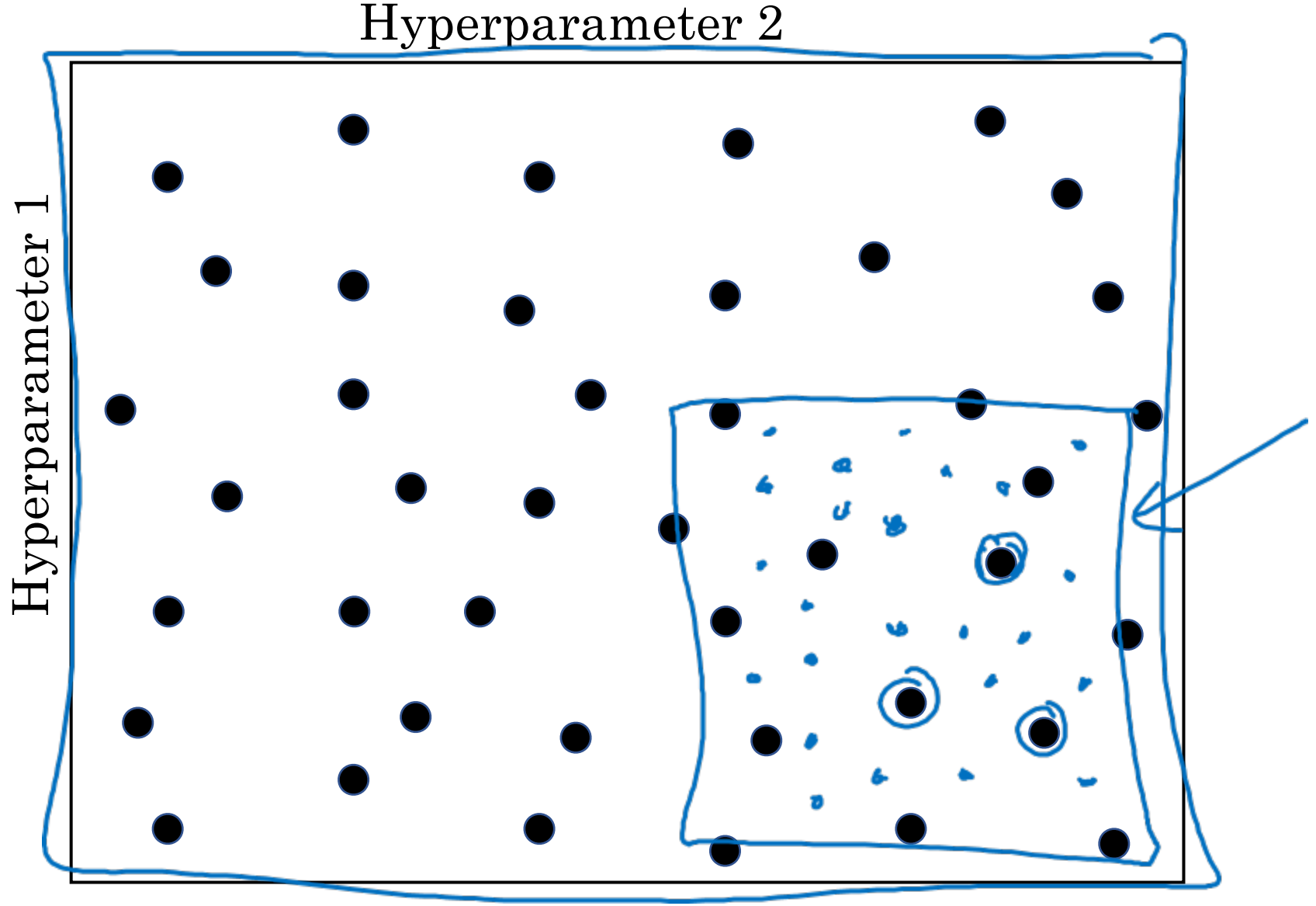


Try random values: Don't use a grid





Coarse to fine



Andrew Ng

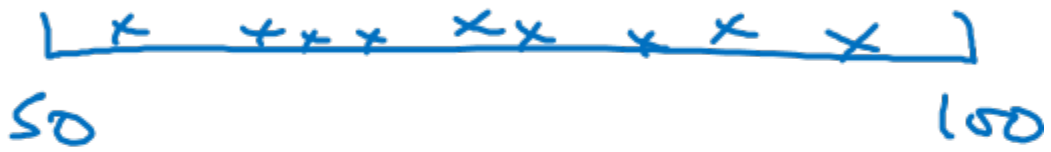
deeplearning.ai

Hyperparameter tuning

Using an appropriate scale to pick hyperparameters

Picking hyperparameters at random

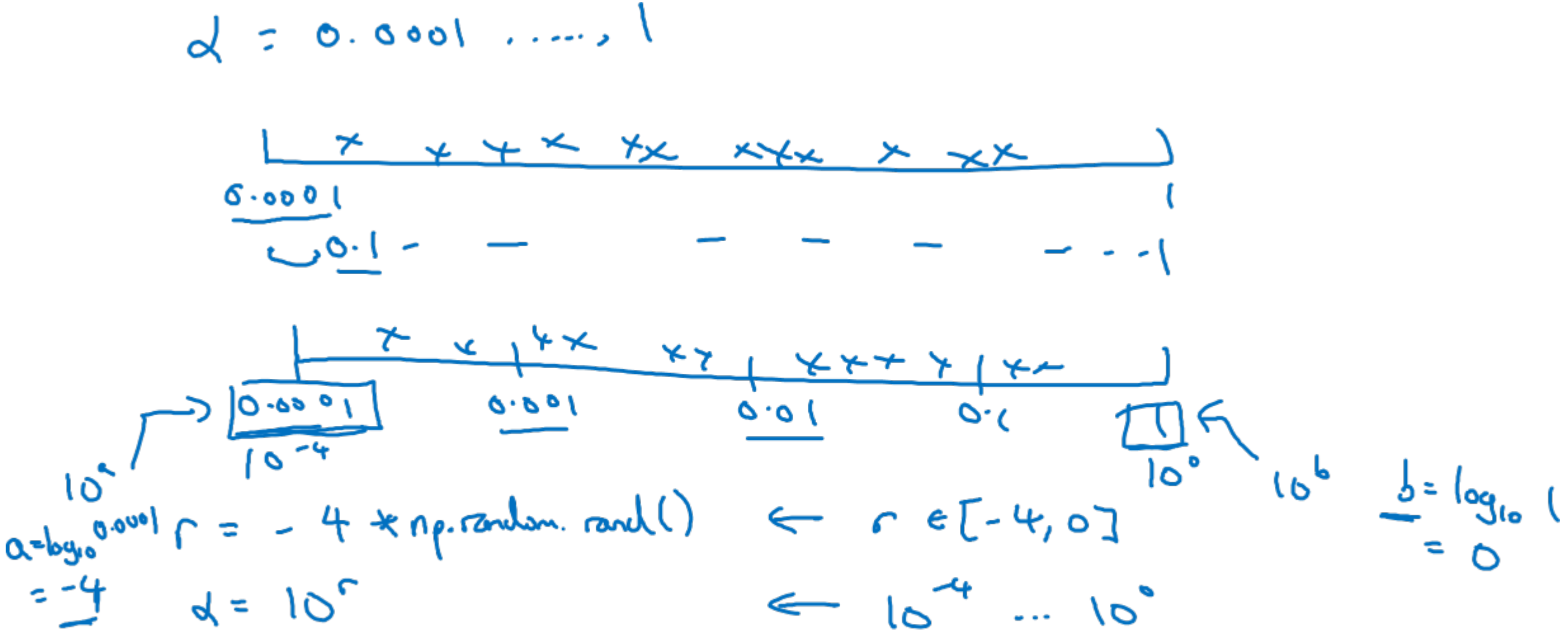
→ $n^{[l]} = 50, \ldots, 100$



→ #layers L: 2 – 4

2, 3, 4

Appropriate scale for hyperparameters



Andrew Ng

Hyperparameters for exponentially weighted averages

$\beta = 0.9 \quad \ldots \quad 0.999$

$10 \qquad\qquad 1000$

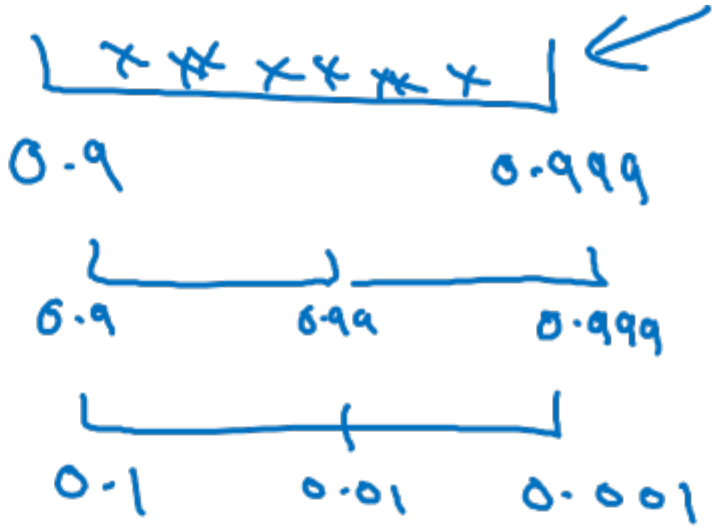
$1-\beta = 0.1 \quad \ldots \quad 0.001$

$\beta: 0.900 \rightarrow 0.9005$ } ~ 10

$\beta: 0.999 \rightarrow 0.9995$

$\sim 1000 \qquad \sim 2000$

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



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

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

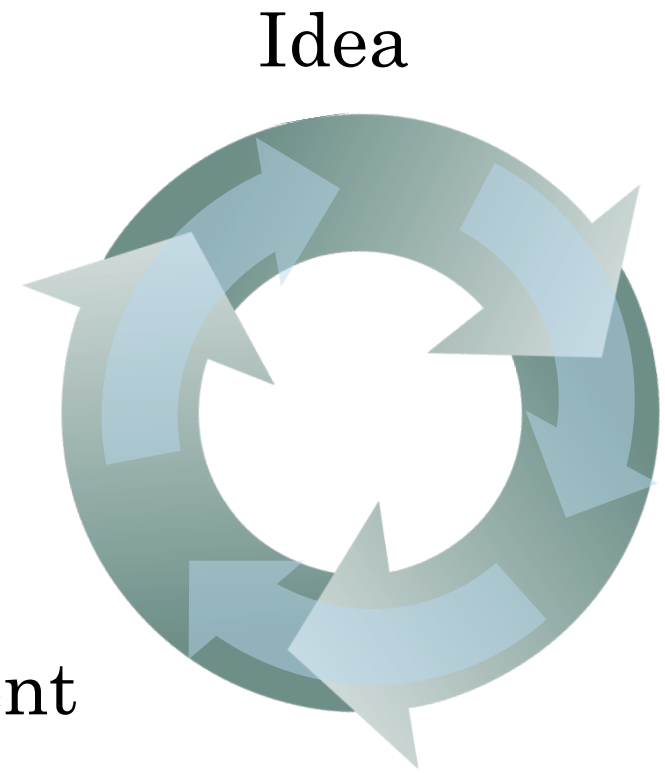
$1-\beta = 10^{r}$

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

deeplearning.ai

Hyperparameters tuning in practice: Pandas vs. Caviar

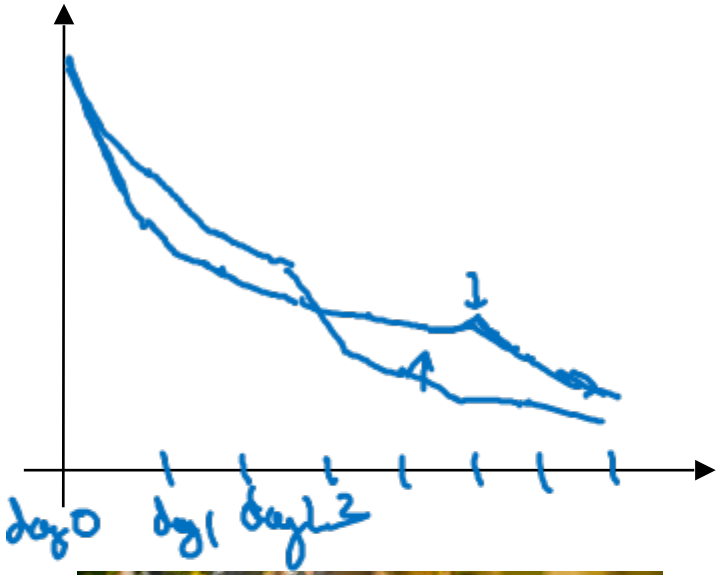
Re-test hyperparameters occasionally

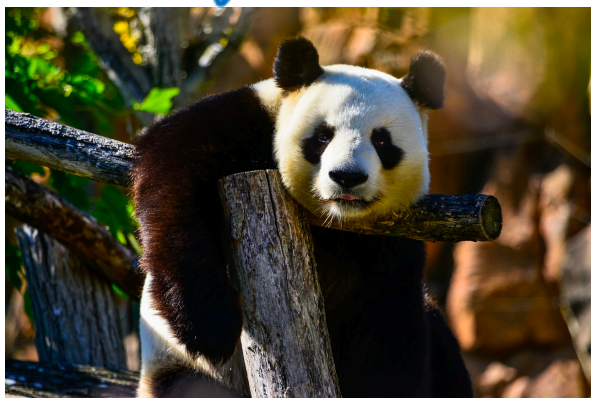


- NLP, Vision, Speech, Ads, logistics, ….

- Intuitions do get stale. Re-evaluate occasionally.

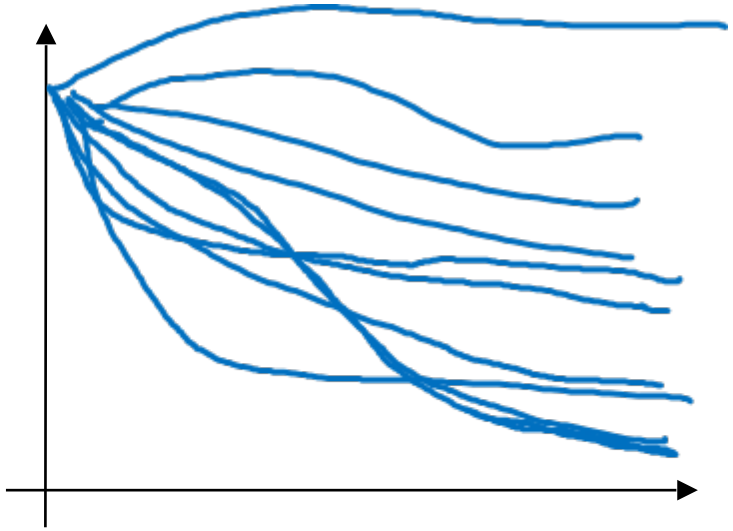
Babysitting one model





Panda

Training many models in parallel



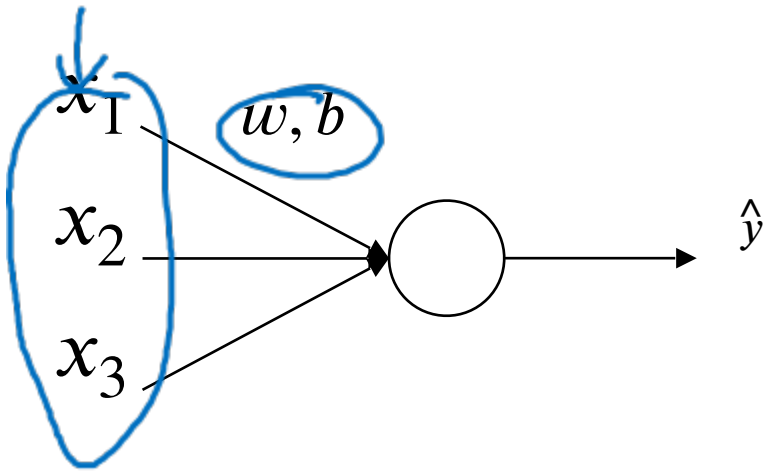
Caviar

deeplearning.ai

Batch Normalization

Normalizing activations in a network

Normalizing inputs to speed up learning

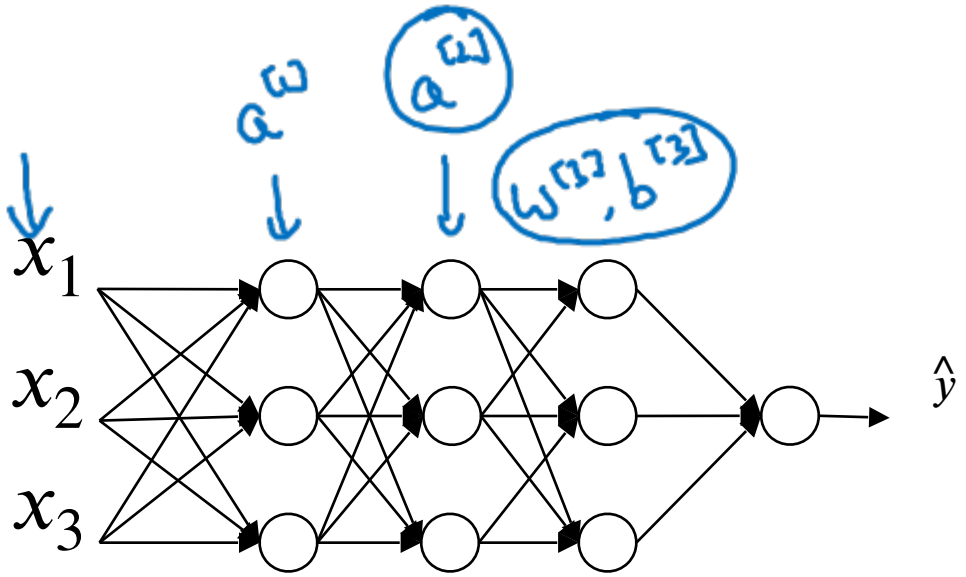


$$\mu = \frac{1}{m}\sum_i x^{(i)}$$

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

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

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



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

Normalize $z^{[2]}$

Andrew Ng

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.

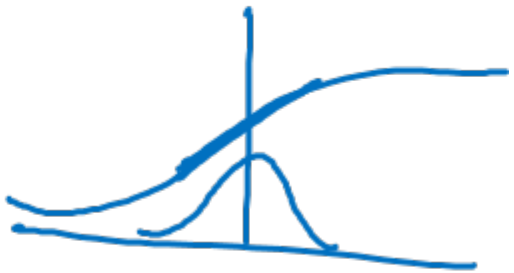
If $\gamma = \sqrt{\sigma^2 + \varepsilon}$ ←

$\beta = \mu$ ←

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

X ←

$z^{(i)}$ ←



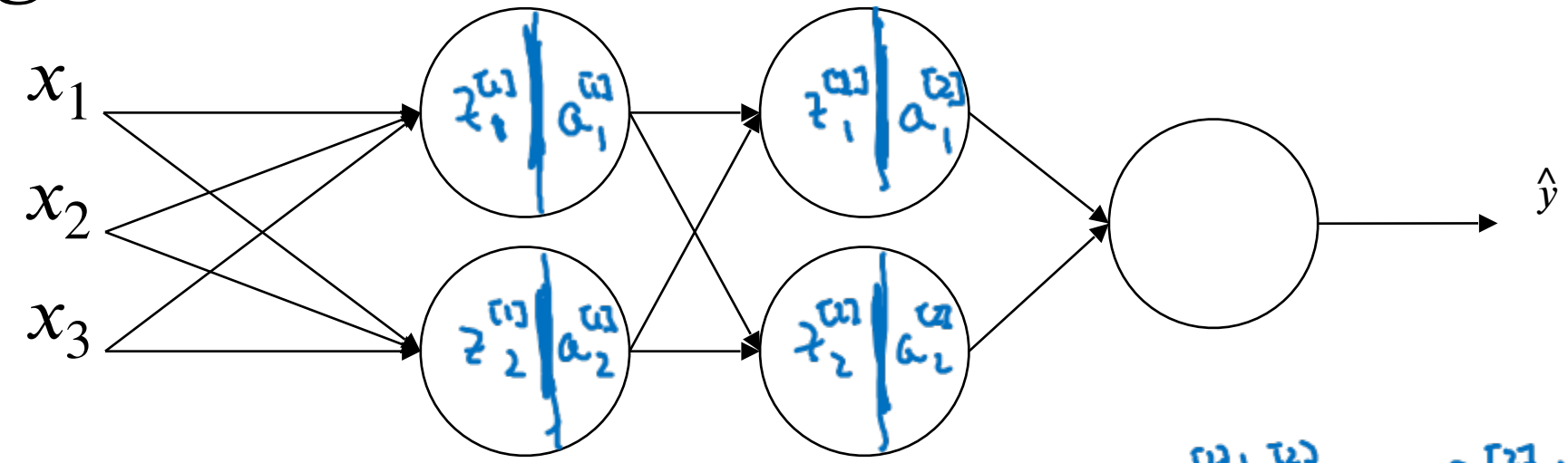
Use $\tilde{z}^{[l](i)}$ instd of $z^{[l](i)}$.

deeplearning.ai

Batch Normalization

Fitting Batch Norm into a neural network

Adding Batch Norm to a network



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

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

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

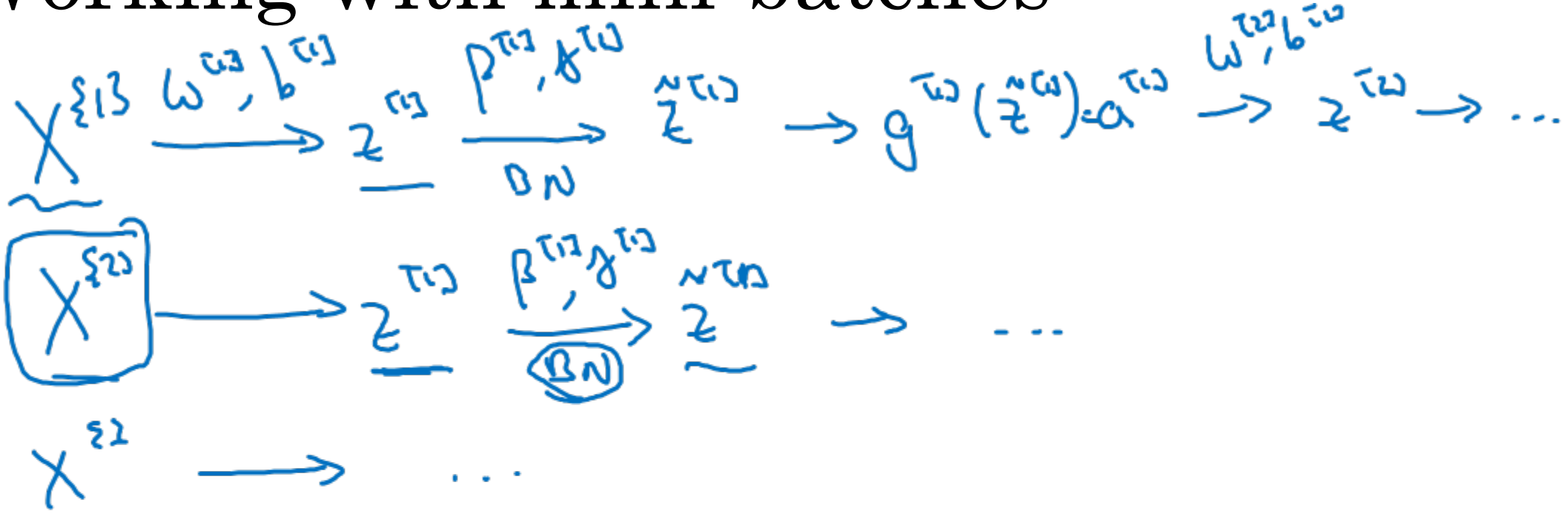
$\rightarrow \beta$

$d\beta^{[2]}$

$\beta^{[2]} = \beta^{[2]} - \alpha\, d\beta^{[2]}$

tf.nn.batch-normalization ⇐

Working with mini-batches



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

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

$z^{[l]}$

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

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

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

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

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

Implementing gradient descent

for $t = 1 \ldots$ numMiniBatches

Compute forward prop on $X^{\{t\}}$.

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

Use backprop to compute $dW^{[l]}$, ~~$db^{[l]}$~~, $d\beta^{[l]}$, $d\gamma^{[l]}$

Update parameters $W^{[l]} := W^{[l]} - \alpha\, dW^{[l]}$

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

$\gamma^{[l]} := \ldots$ ←

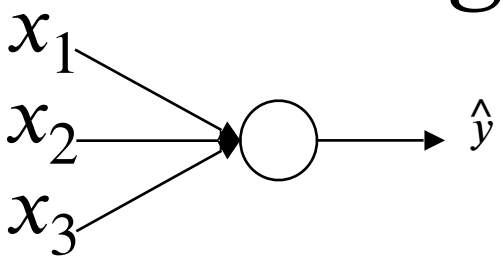
Works w/ momentum, RMSprop, Adam.

deeplearning.ai

Batch Normalization

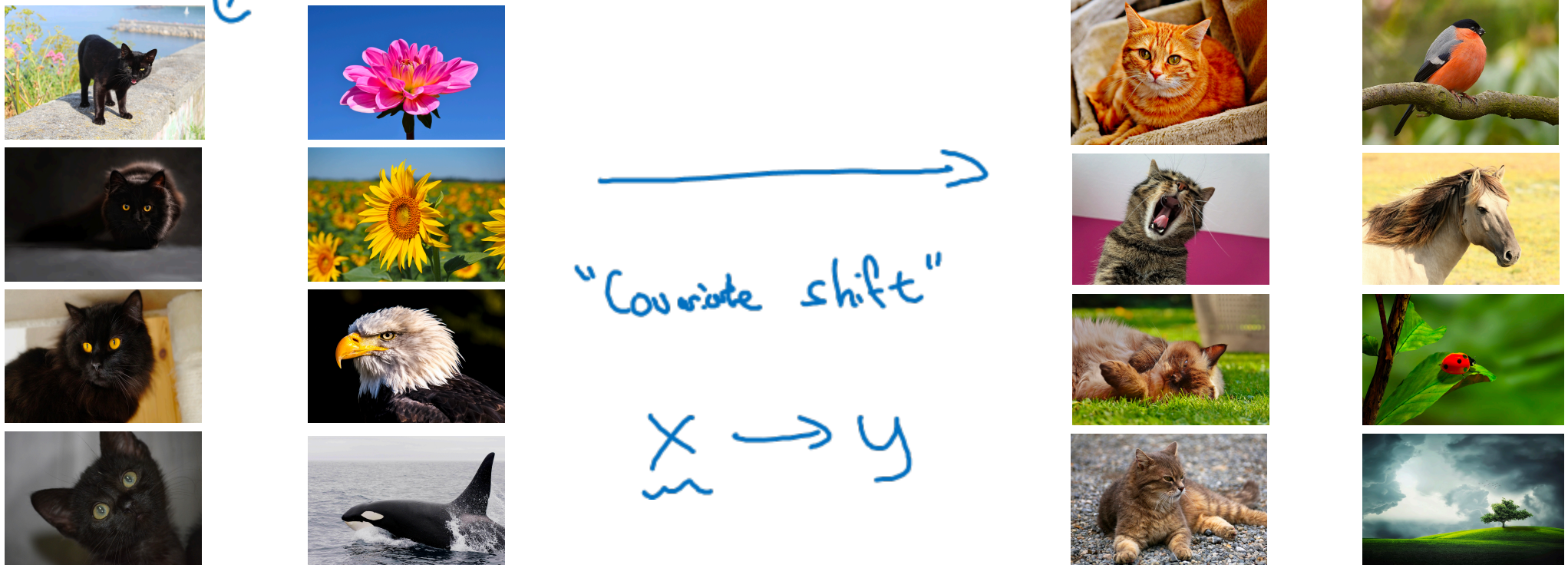
Why does Batch Norm work?

Learning on shifting input distribution

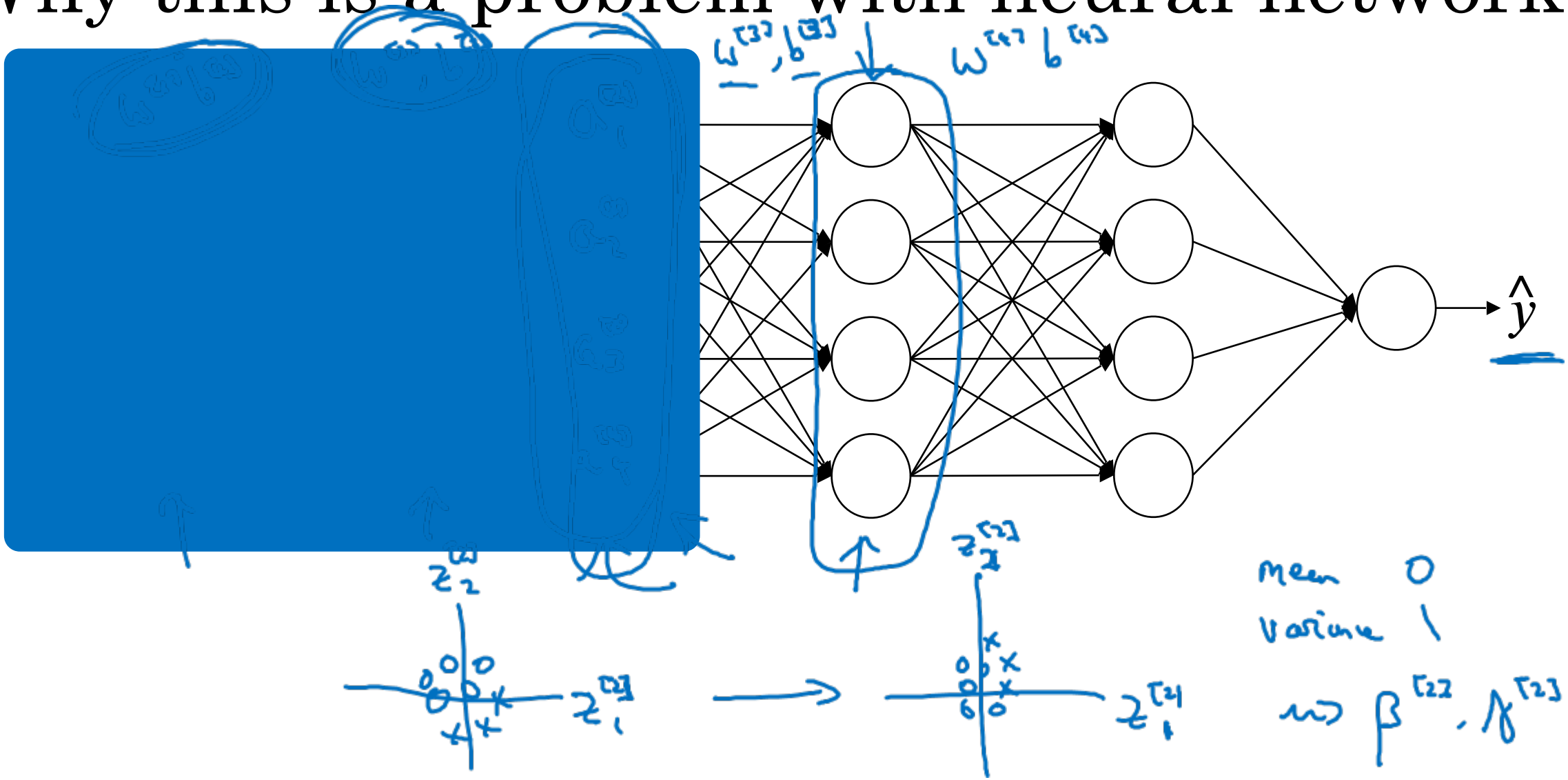


Cat Non-Cat

$y = 1$ $y = 0$



Why this is a problem with neural networks?



Batch Norm as regularization

- Each mini-batch is scaled by the mean/variance computed on just that mini-batch.
- This adds some noise to the values within that minibatch. So similar to dropout, it adds some noise to each hidden layer's activations.
- This has a slight regularization effect.

deeplearning.ai

Batch Normalization

Batch Norm at test time

Batch Norm at test time

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

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

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

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

μ, σ^2: estimate using exponentially weighted average (across mini-batches).

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

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

θ_1 θ_2 θ_3 σ^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

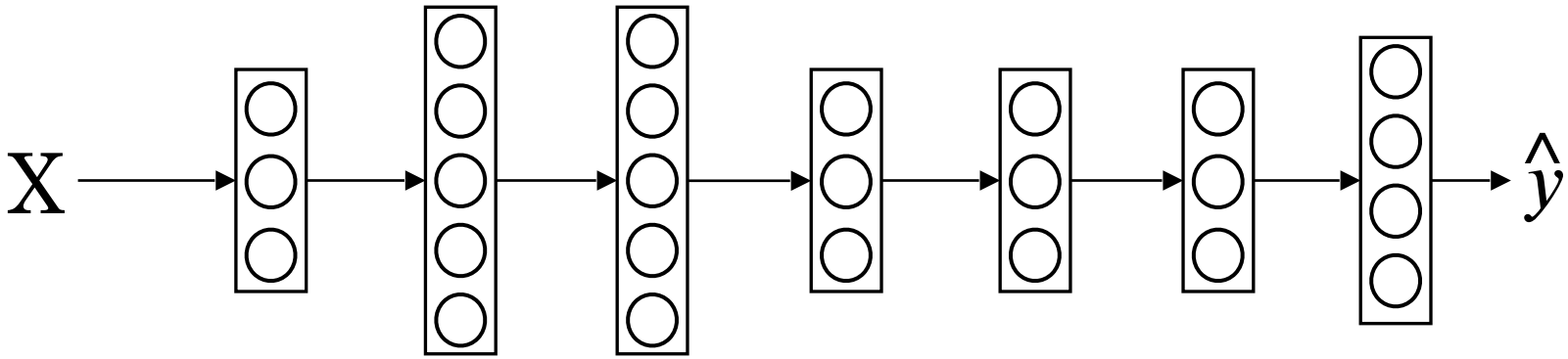
deeplearning.ai

Multi-class classification

Softmax regression

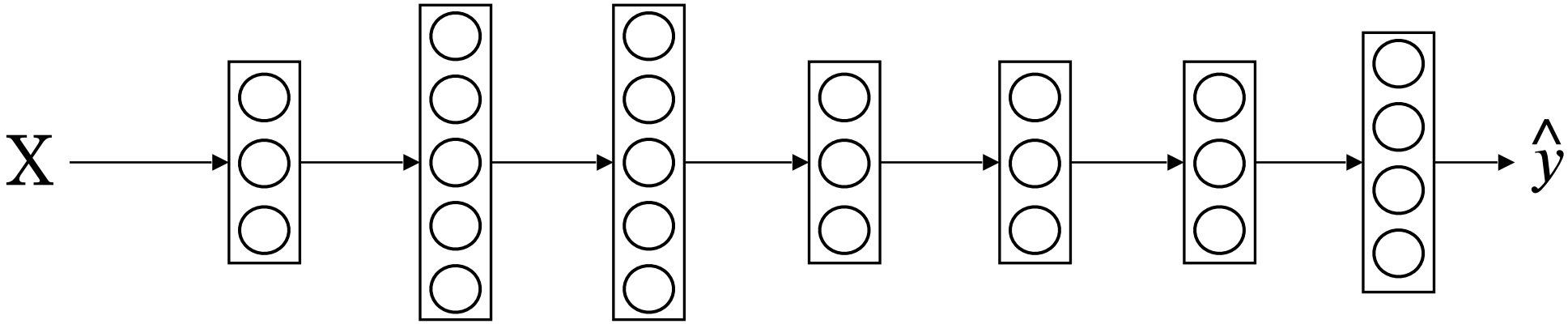
Recognizing cats, dogs, and baby chicks

3 1 2 0 3 2 0 1

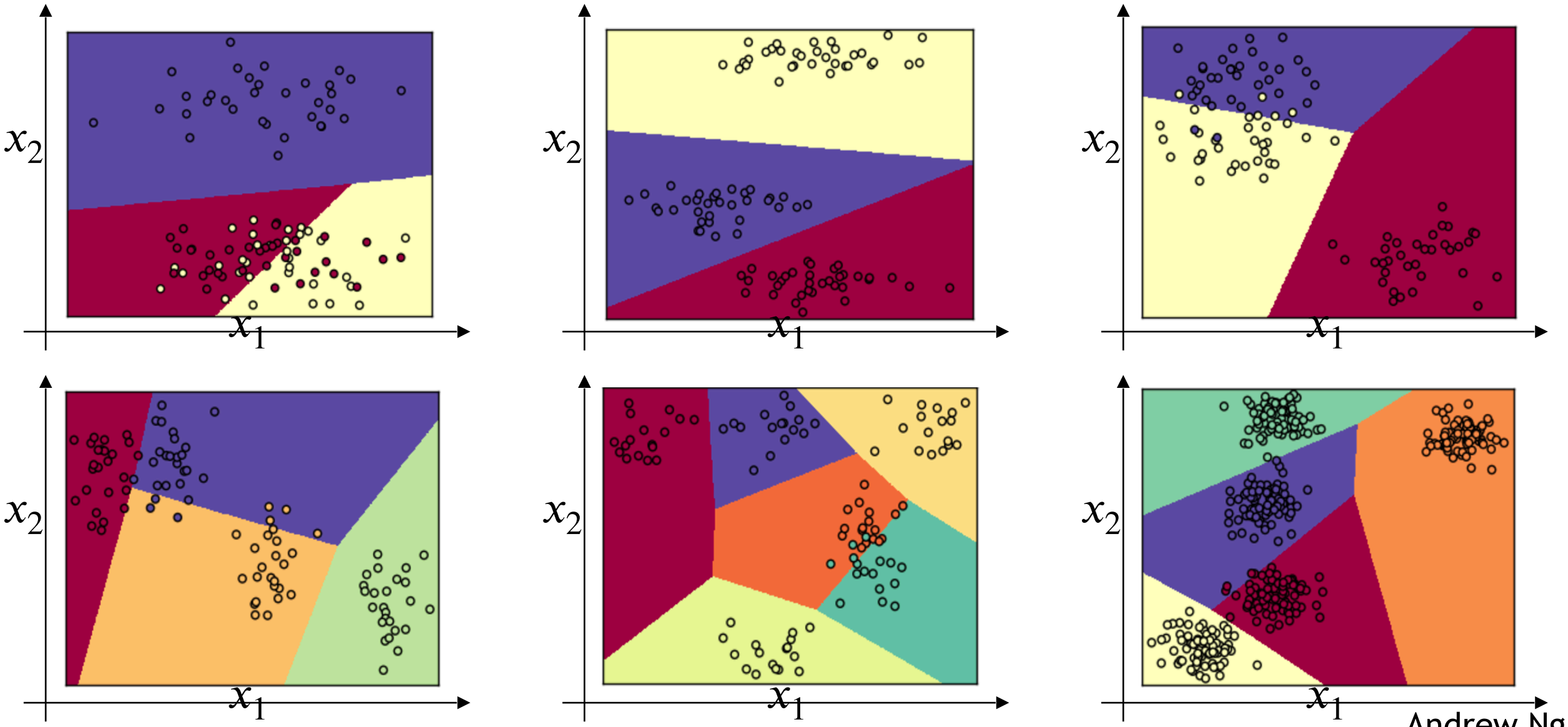


Andrew Ng

Softmax layer



Softmax examples



deeplearning.ai

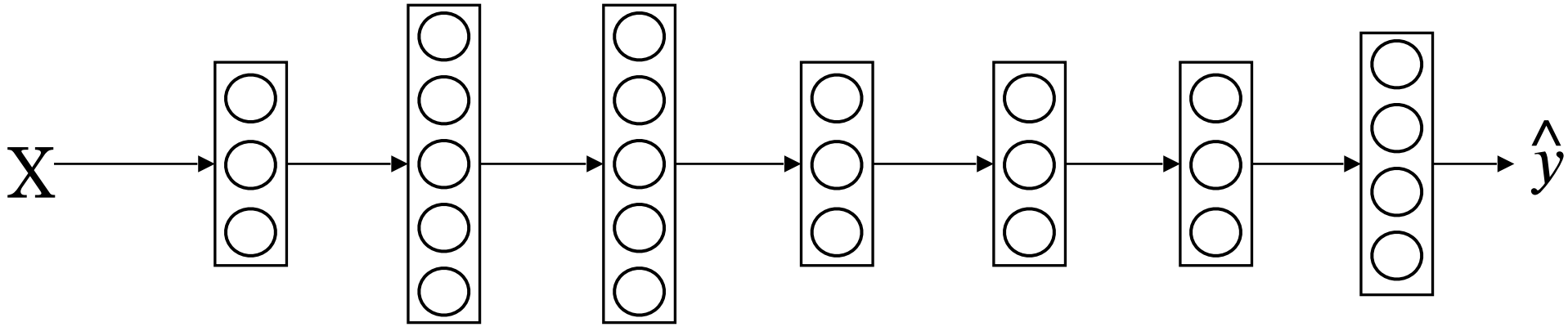
Multi-class classification

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

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

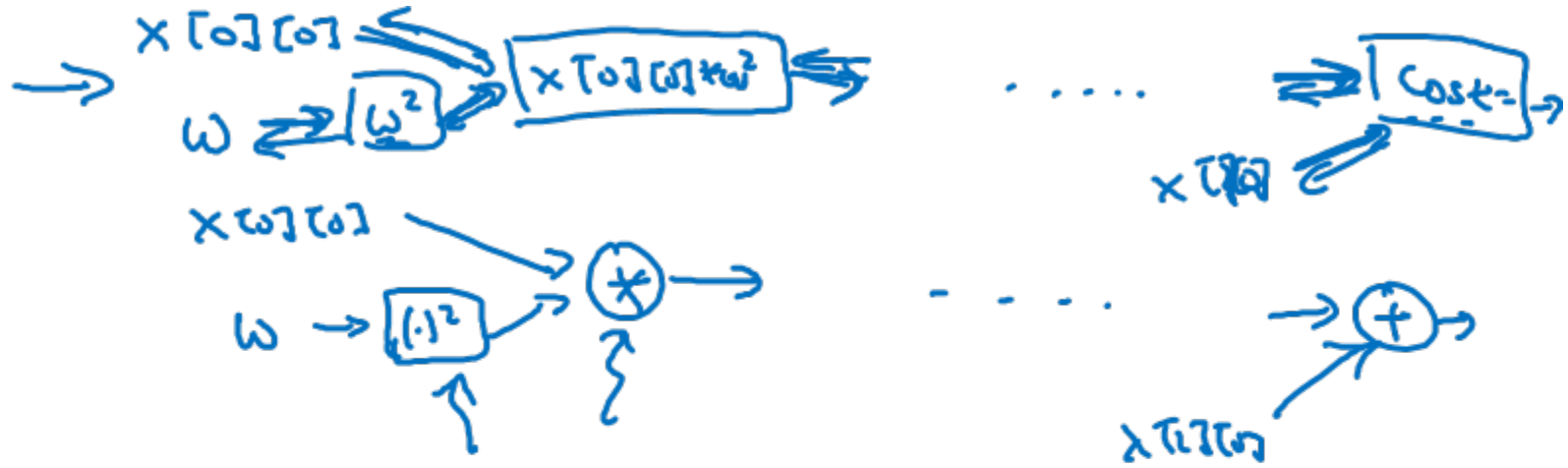
$(w-5)^2$

$w = 5$

(cost)

$J(w, b)$

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