Copyright Notice

These slides are distributed under the Creative Commons License.

DeepLearning.AI makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite DeepLearning.AI as the source of the slides.

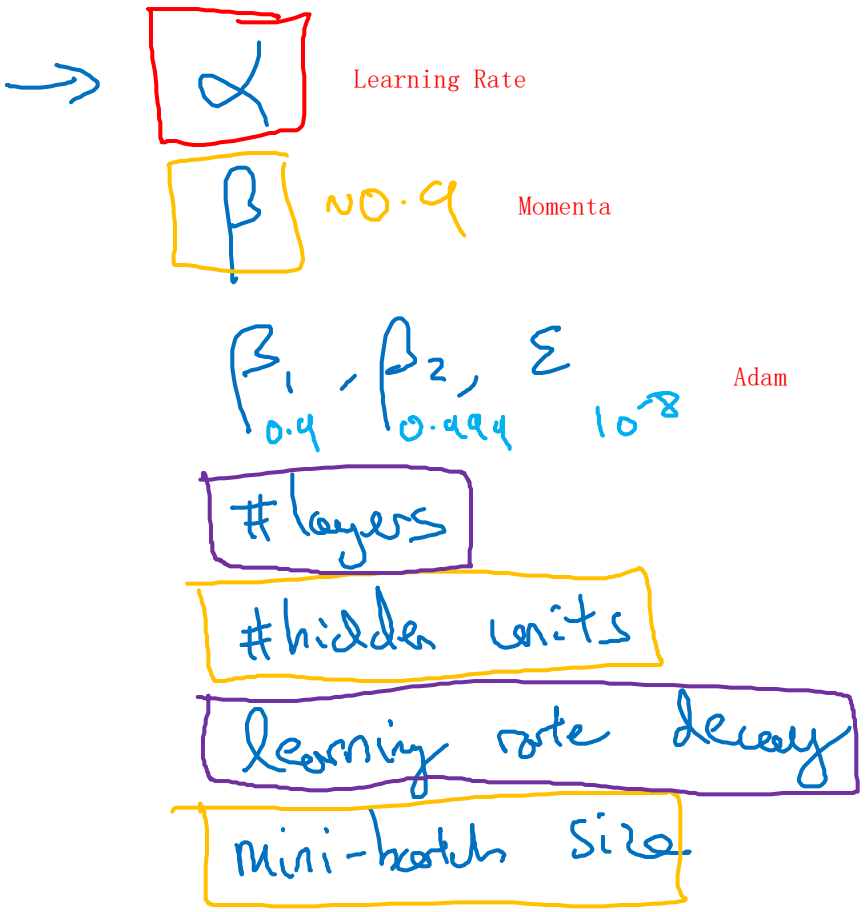
For the rest of the details of the license, see https://creativecommons.org/licenses/by-sa/2.0/legalcode

deeplearning.ai

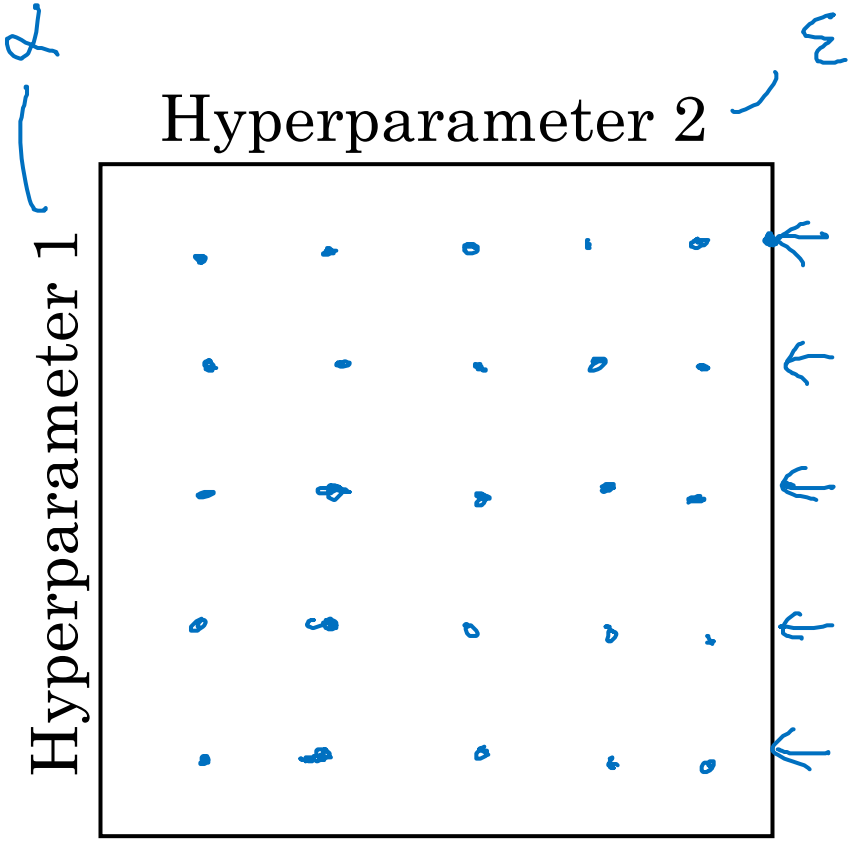
Hyperparameter tuning

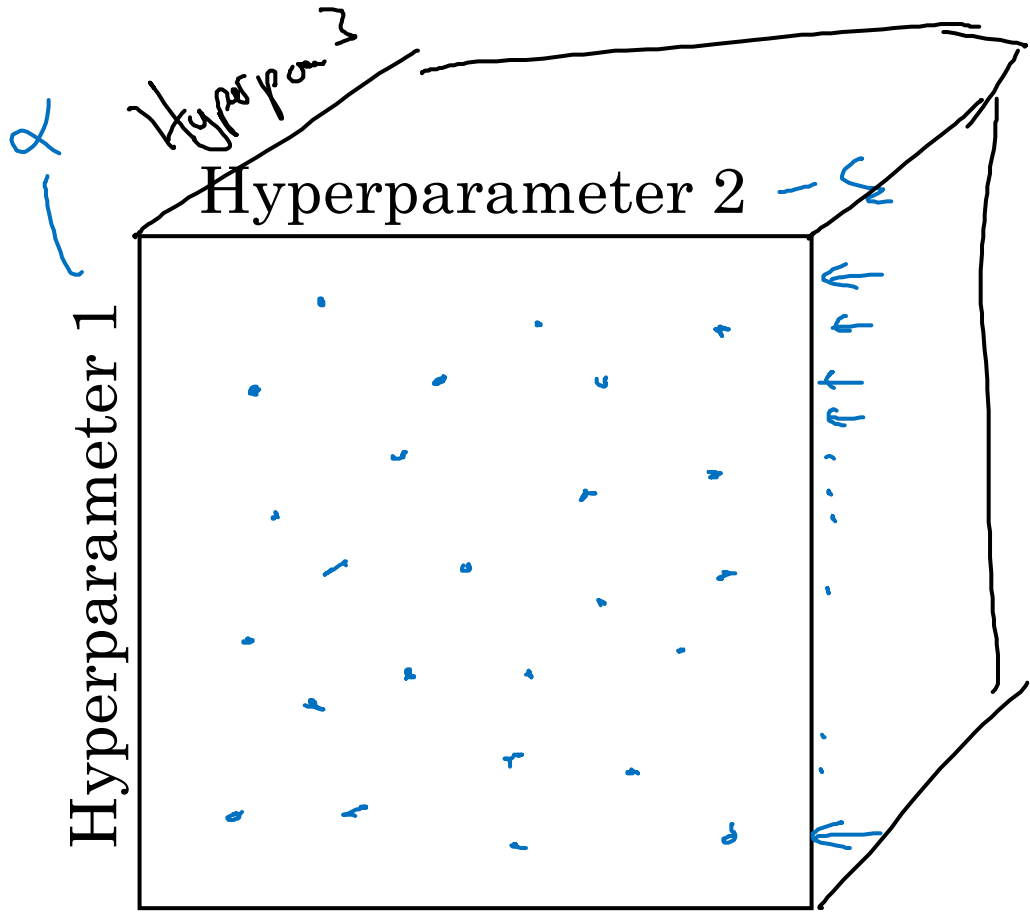
Tuning process

Hyperparameters

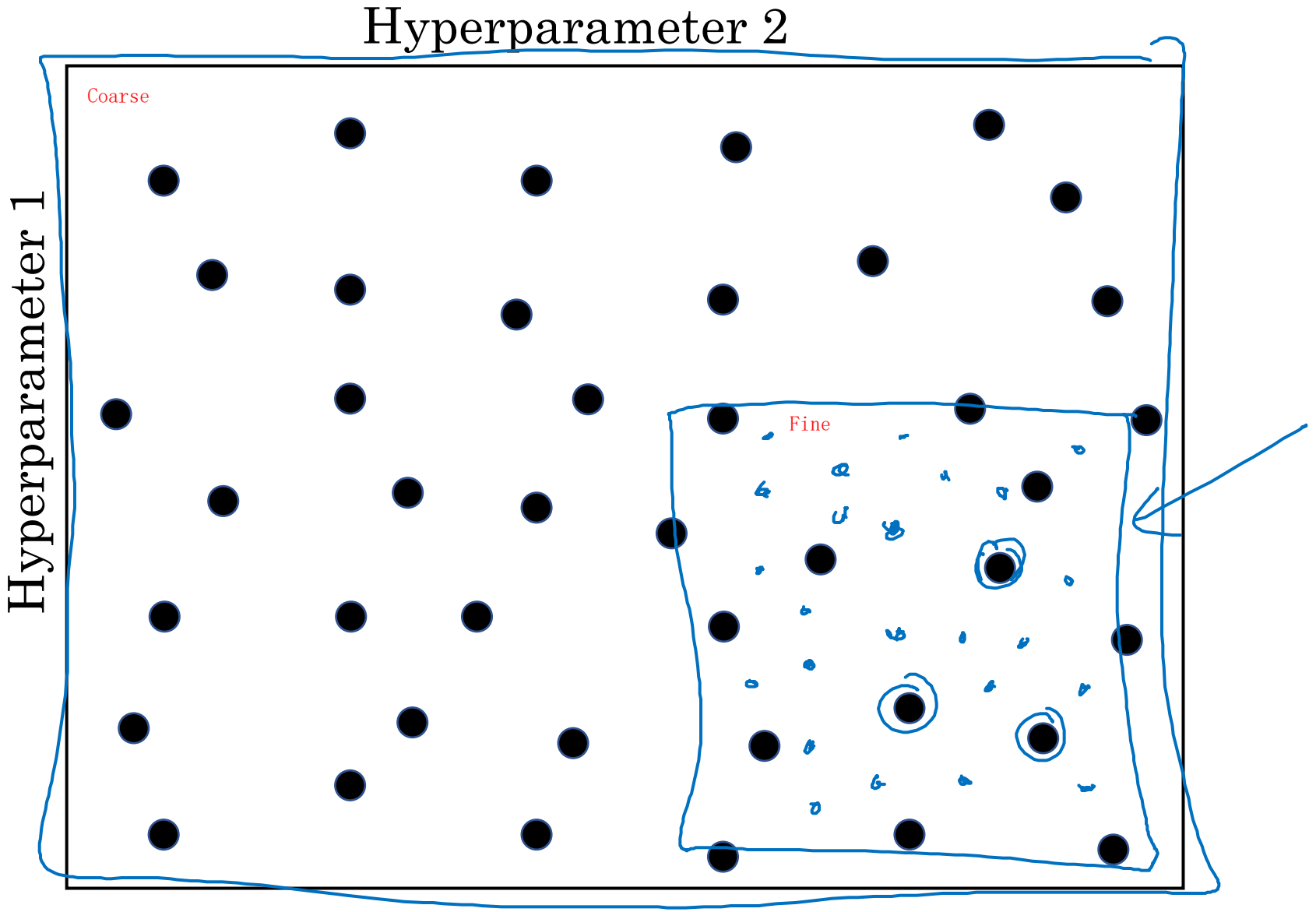


Try random values: Don't use a grid





Coarse to fine



Andrew Ng

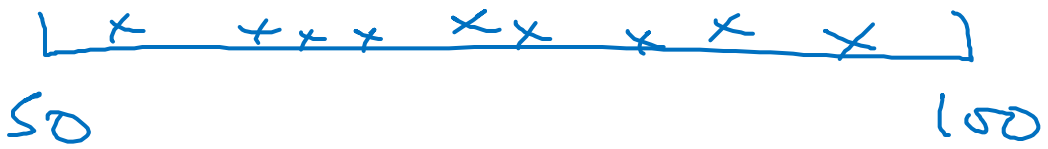
deeplearning.ai

Hyperparameter tuning

Using an appropriate scale to pick hyperparameters

Picking hyperparameters at random

→ $n^{[l]} = 50, \ldots, 100$



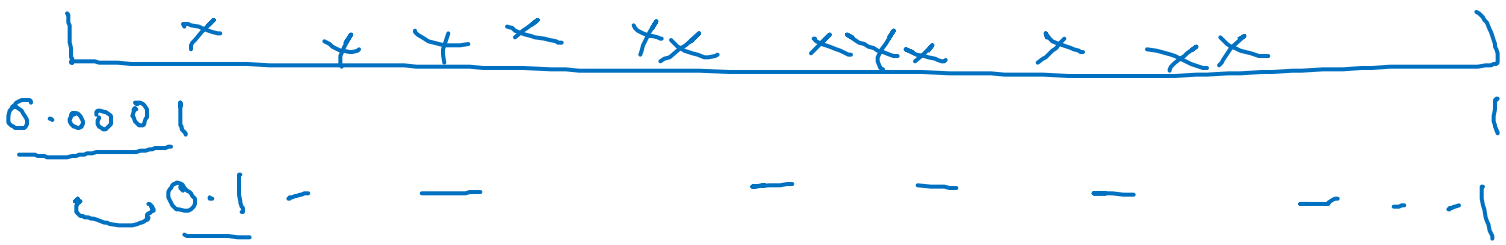
→ #layers L: 2 — 4

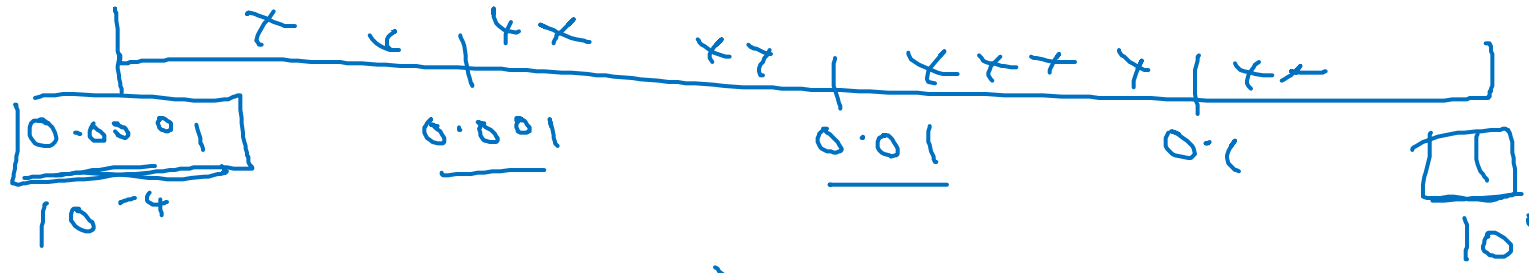
sometimes sampling uniformly at random might be reasonable

2, 3, 4

Appropriate scale for hyperparameters

$\alpha = 0.0001, \ldots, 1$





10^{a} 10^{b}

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

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

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

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

$10^{a} \ldots 10^{b}$

$r \in [a, b]$

$[-4, 0]$

$\alpha = 10^{r}$

Andrew Ng

Hyperparameters for exponentially weighted averages

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

average over last $10 \quad 1000$

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

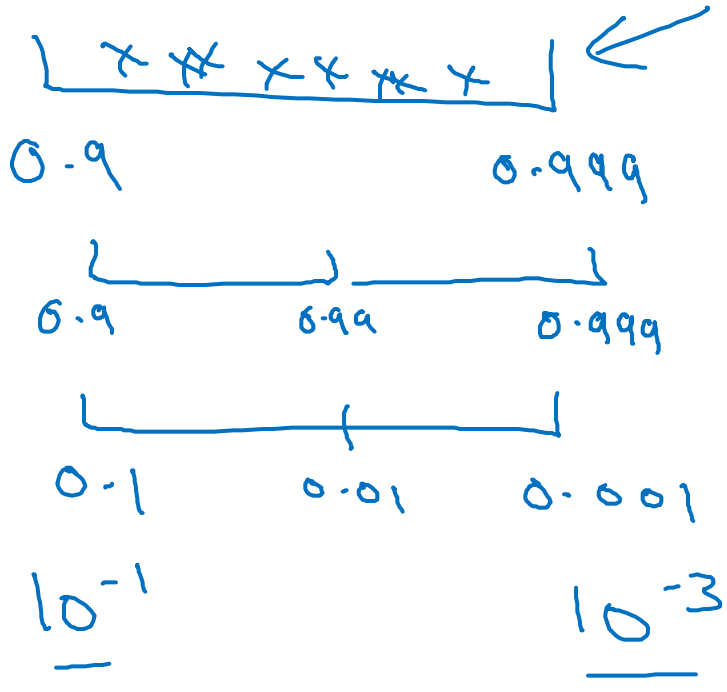
$\beta: \ 0.900 \rightarrow 0.9005$ } ~ 10

$\beta: \ 0.999 \rightarrow 0.9995$

$\sim 1000 \qquad \sim 2000$

become away
more sensitive

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



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

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

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

Andrew Ng

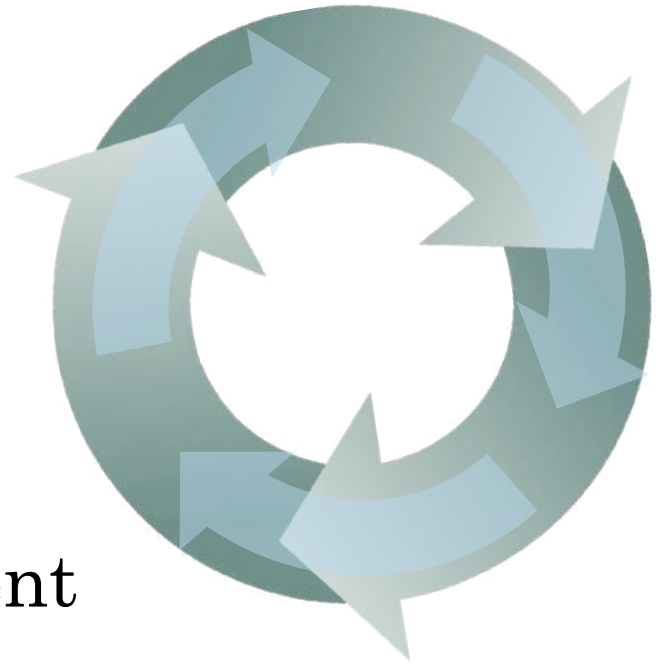
deeplearning.ai

Hyperparameters tuning

Hyperparameters tuning in practice: Pandas vs. Caviar

Re-test hyperparameters occasionally

Idea



Experiment

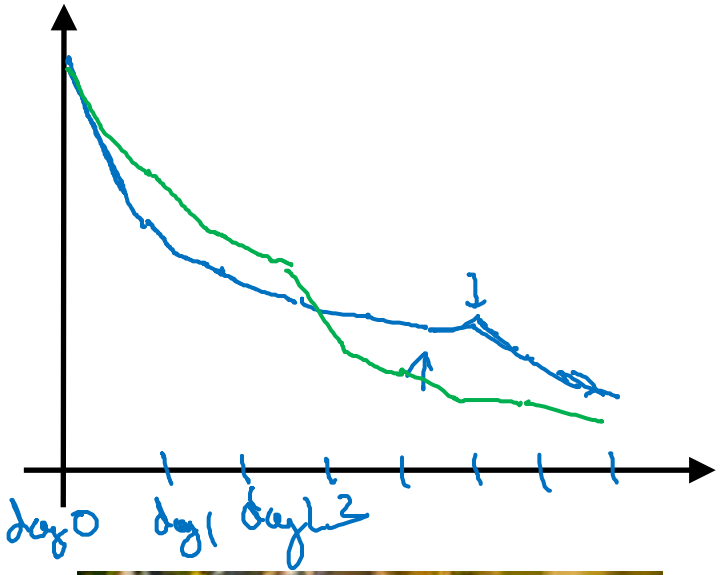
Code

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

- Intuitions do get stale. Re-evaluate occasionally.

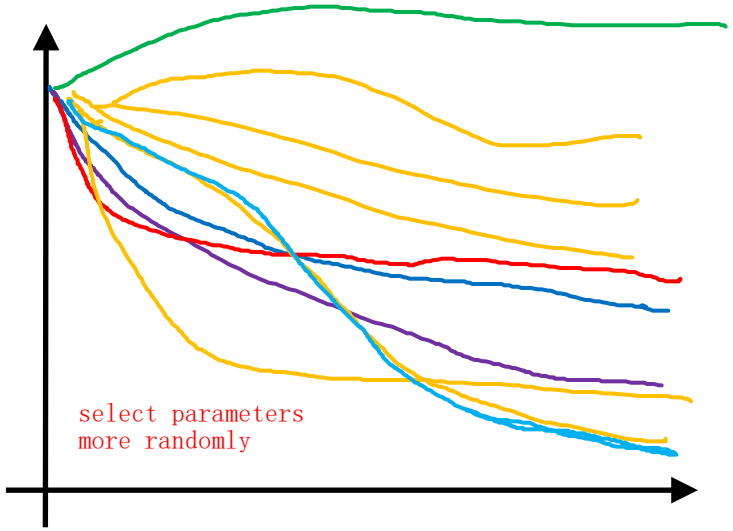
Andrew Ng

Babysitting one model



Panda

Training many models in parallel



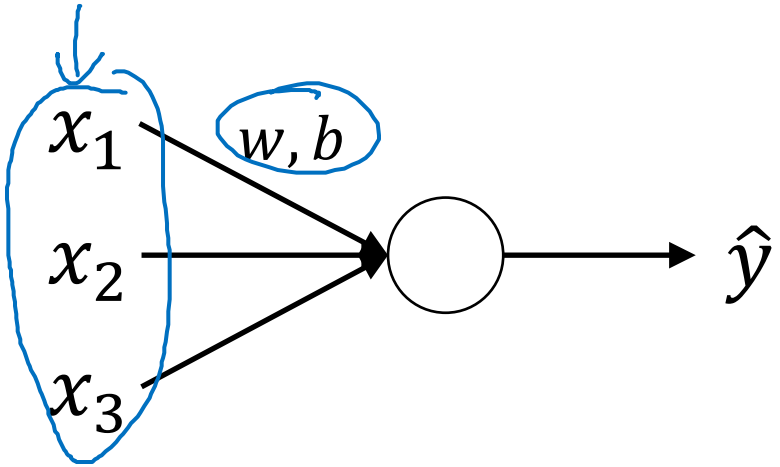
Caviar

deeplearning.ai

Batch Normalization

Normalizing activations in a network

Normalizing inputs to speed up learning

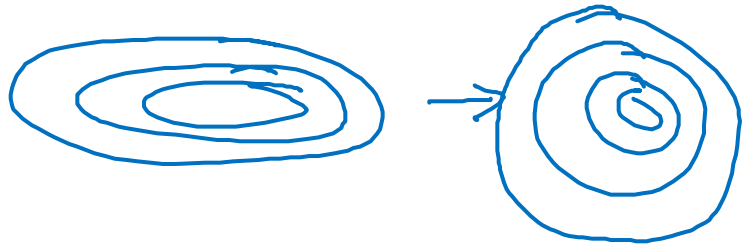


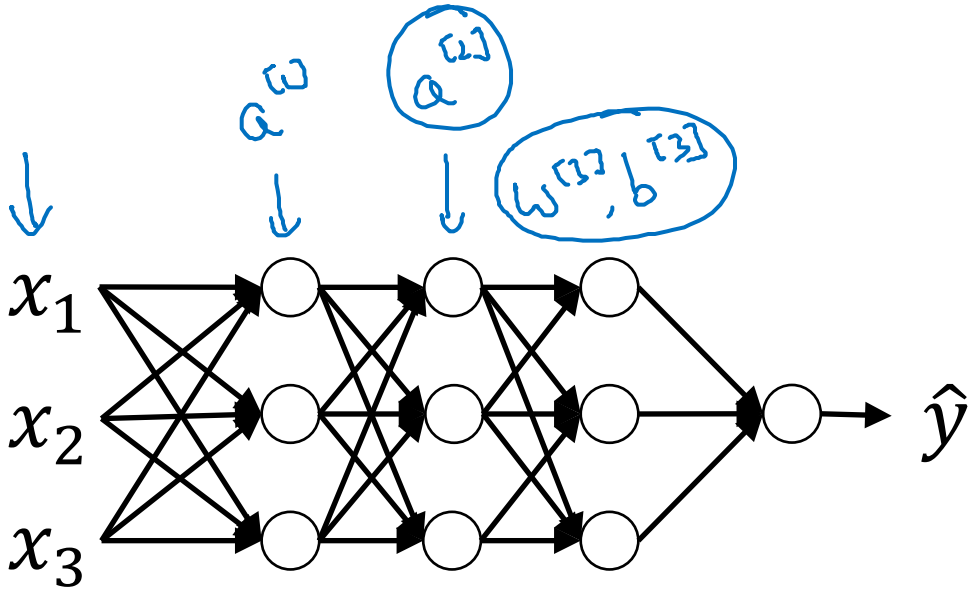
$$\mu = \frac{1}{m}\sum_i x^{(i)}$$

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

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

$$X = X / \sigma$$





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 $z^{(1)}, \ldots, z^{(m)}$ of $z^{[l](i)}$

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

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

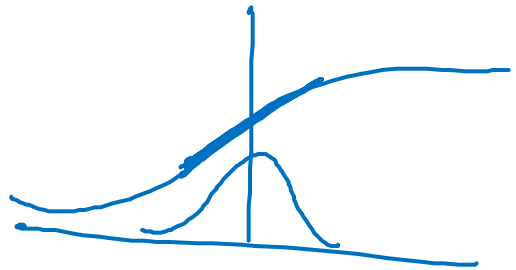
$$z^{(i)}_{norm} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

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

learnable parameters of model.

If $\gamma = \sqrt{\sigma^2 + \varepsilon}$, $\beta = \mu$, then $\tilde{z}^{(i)} = z^{(i)}$

$X \rightarrow z^{(i)}$



not necessarly mean 0, variance 1

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

apply normalization not only in the input layer but also in hidden layers

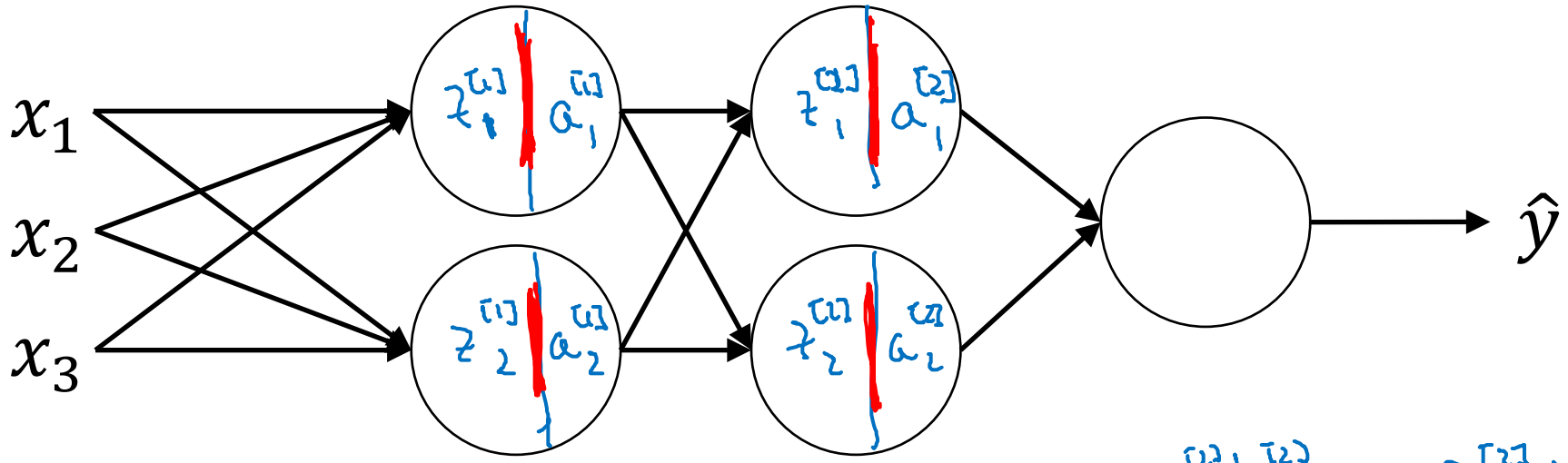
Andrew Ng

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

$\rightarrow \beta$

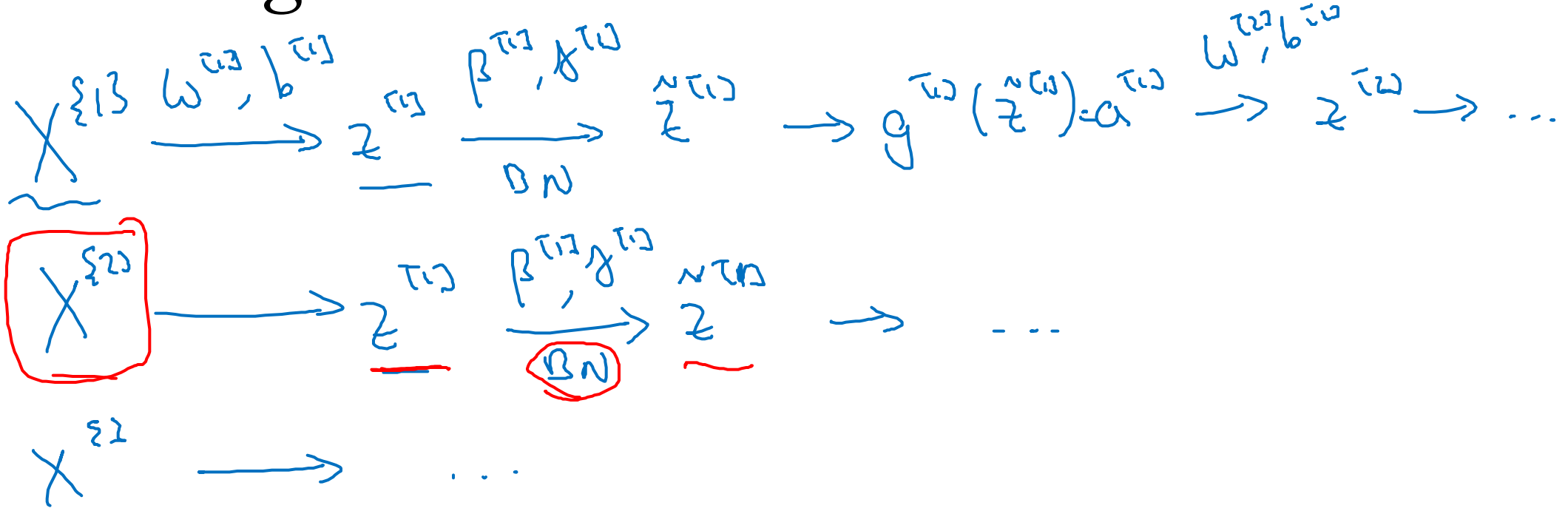
$d\beta^{[l]}$

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

tf.nn.batch_normalization

Andrew Ng

Working with mini-batches



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

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

$Z^{[l]}$

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

$$Z^{[l]} = W^{[l]} a^{[l-1]} + \cancel{b^{[l]}}$$

subtracted out

$$Z^{[l]} = W^{[l]} a^{[l-1]}$$

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

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

Andrew Ng

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

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

$\gamma^{[l]} := \ldots$ ←

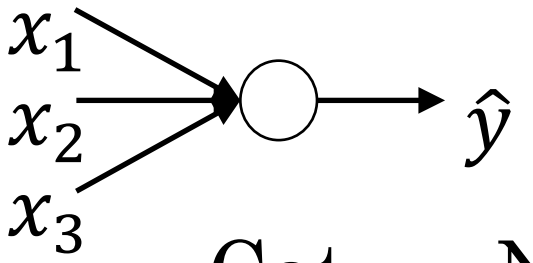
Works w/ momentum, RMSprop, Adam.

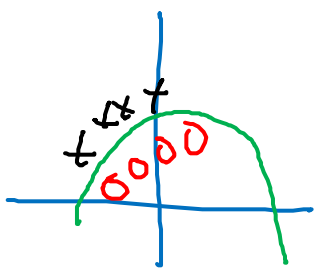
deeplearning.ai

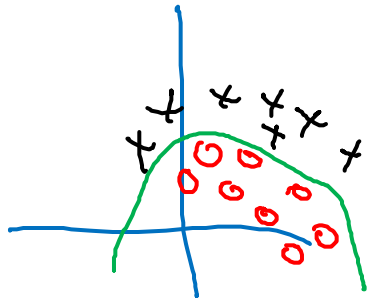
Batch Normalization

Why does Batch Norm work?

Learning on shifting input distribution







Cat $y = 1$




Non-Cat $y = 0$




"Covariate shift"

if the distribution of x changes, then we need to retrain the model

$X \rightarrow y$

$y = 1$

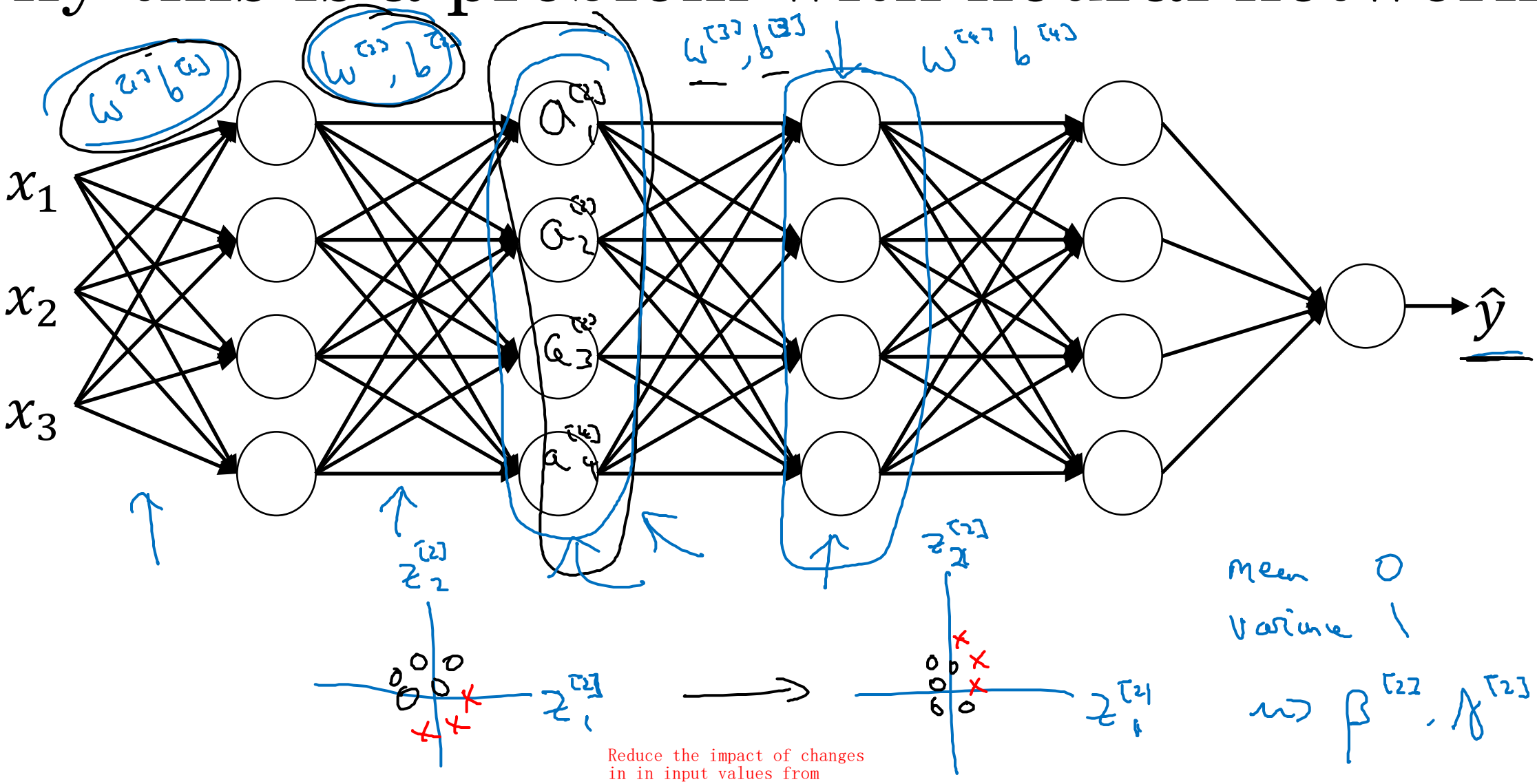



$y = 0$




Andrew Ng

Why this is a problem with neural networks?



Andrew Ng

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 $z^{[l]}$ within that minibatch. So similar to dropout, it adds some noise to each hidden layer's activations.
- This has a slight regularization effect.

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).

$X^{\{1\}}, X^{\{2\}}, X^{\{3\}}, \ldots$

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

θ_1 θ_2 θ_3 σ^2

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

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

Andrew Ng

deeplearning.ai

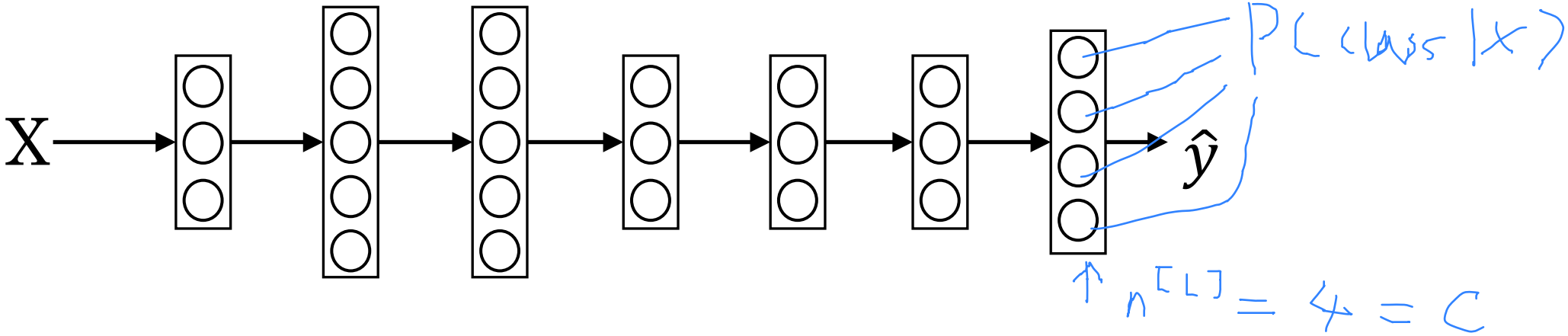
Multi-class classification

Softmax regression

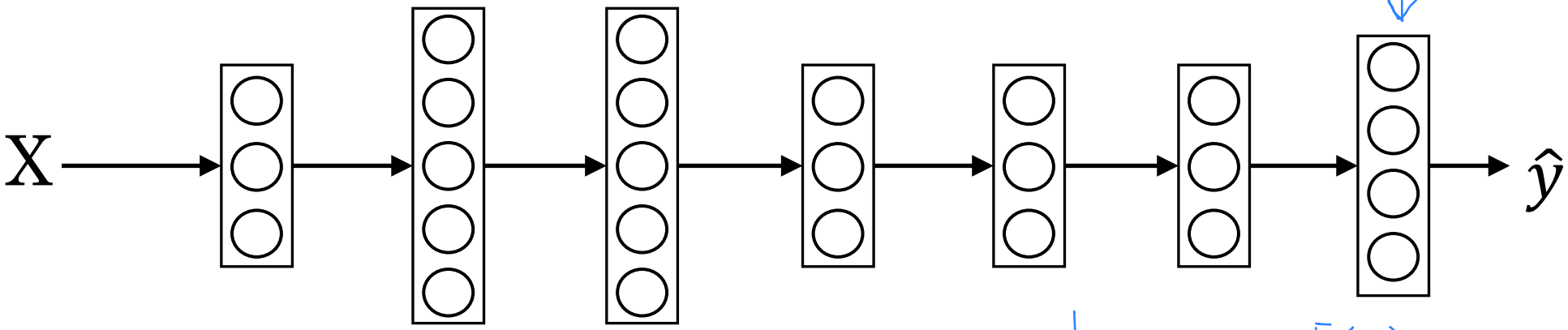
Recognizing cats, dogs, and baby chicks

3 1 2 0 3 2 0 1

$C = \#\text{classes} = 4 \quad (0, \ldots, 3)$



Softmax layer



$a^{[L]}$

$z^{[L]} = W^{[L]} a^{[L-1]} + b^{[L]}$

$(4,1)$

Activation Function:

$t = e^{(z^{[L]})}$ $(4,1)$

$a^{[L]} = \frac{e^{z^{[L]}}}{\sum_{j=1}^{4} t_j}$, $a_i^{[L]} = \frac{t_i}{\sum_{j=1}^{4} t_i}$

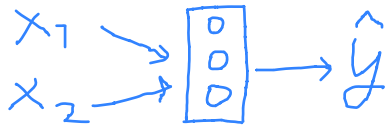
$(4,1)$

$z^{[L]} = \begin{bmatrix} 5 \\ 2 \\ -1 \\ 3 \end{bmatrix}$

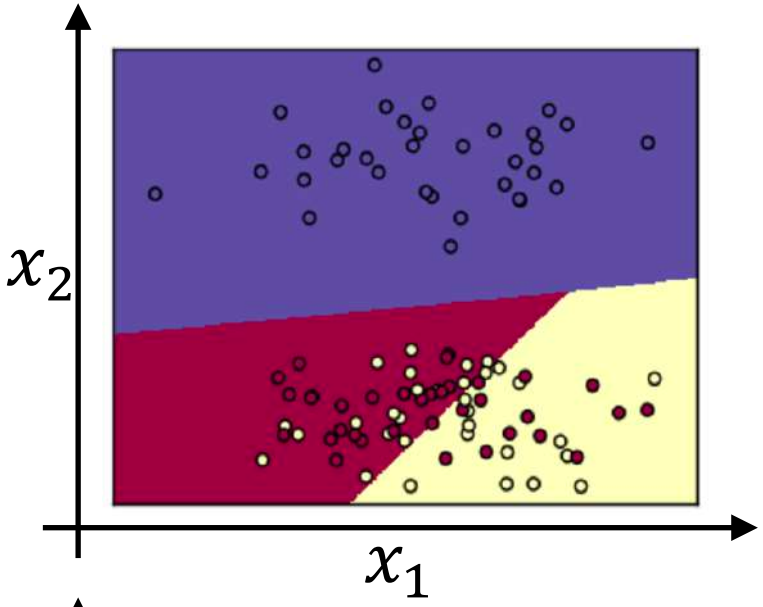
$t = \begin{bmatrix} e^5 \\ e^2 \\ e^{-1} \\ e^3 \end{bmatrix} = \begin{bmatrix} 148.4 \\ 7.4 \\ 0.4 \\ 20.1 \end{bmatrix}$ $\sum_{j=1}^{4} t_j = 176.3$

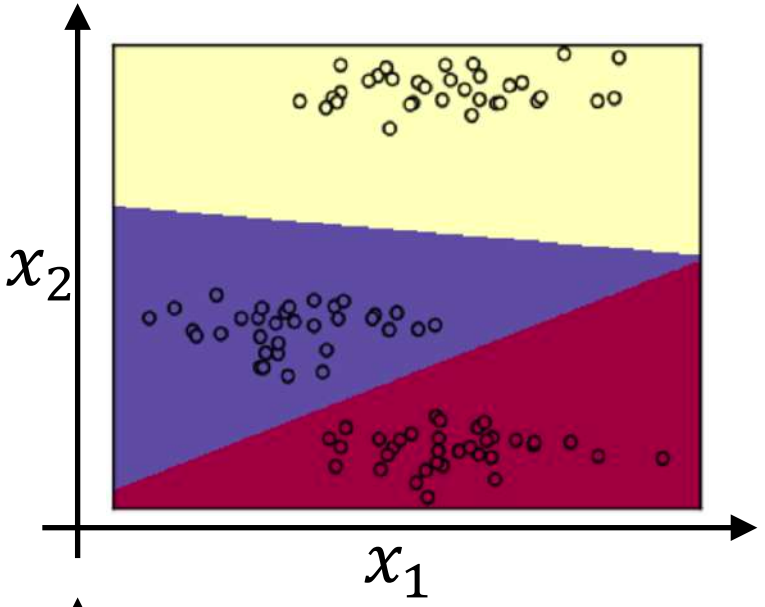
$a^{[L]} = t / 176.3$

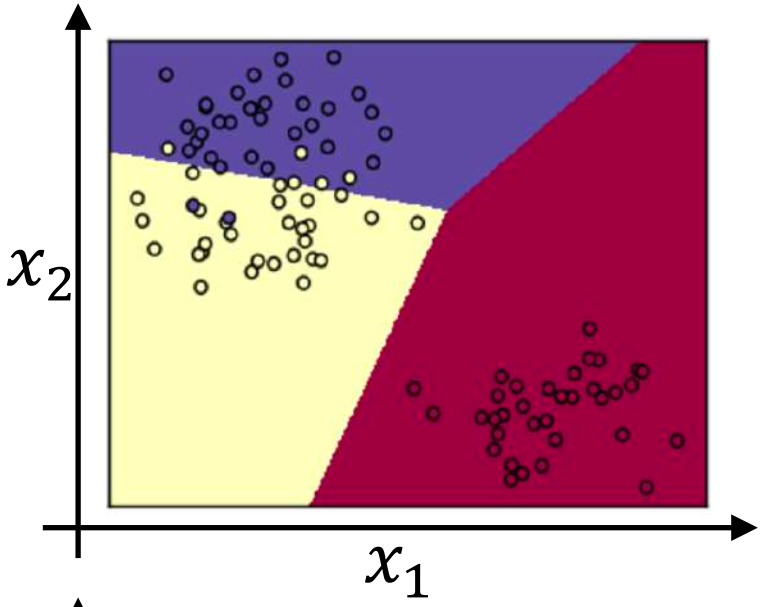
Softmax examples

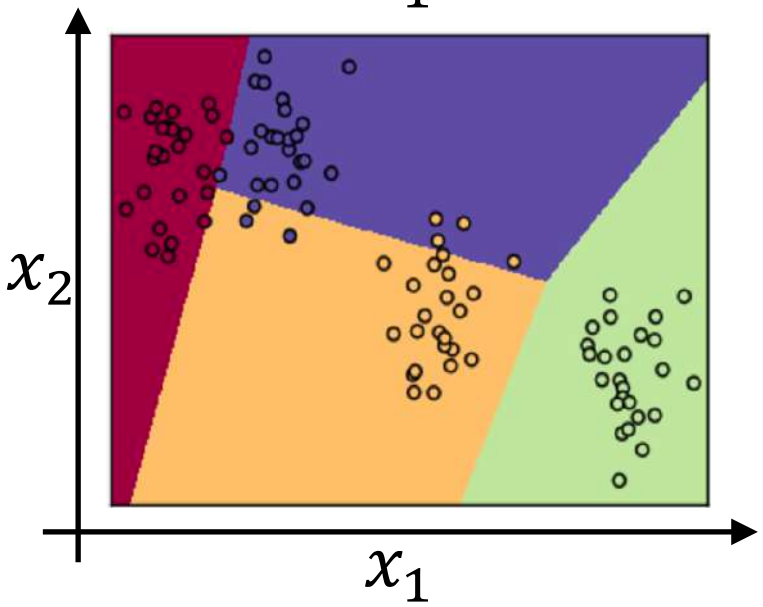


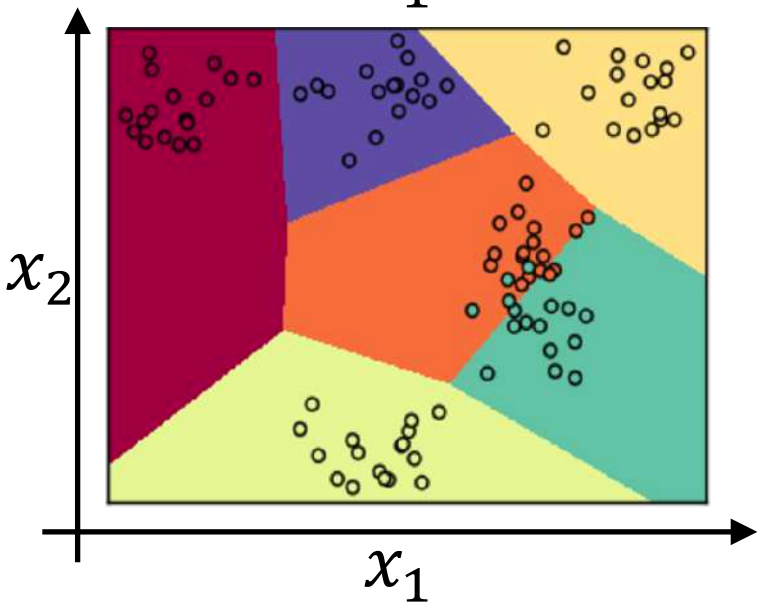
$$z^{[l]} = w^{[l]}x + b^{[l]}$$
$$a^{[l]} = \hat{y} = g(z^{[l]})$$

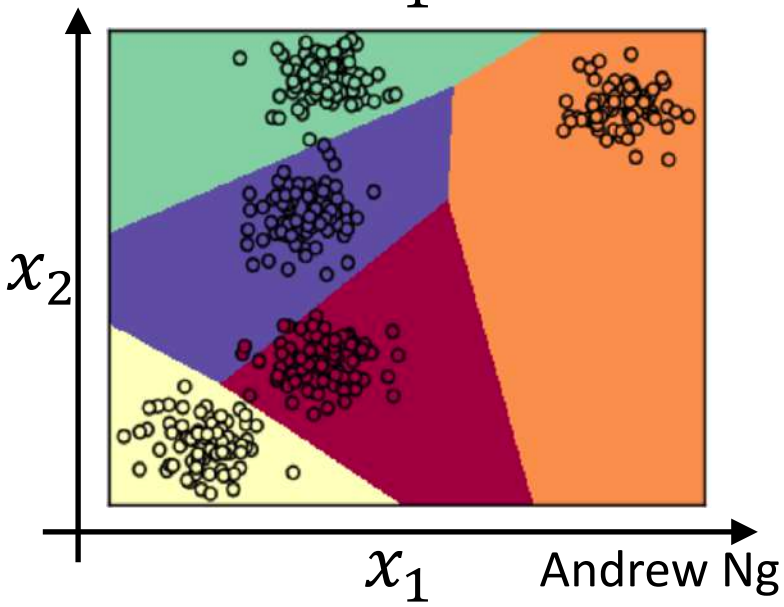












Understanding softmax

(4,1)

$$z^{[L]} = \begin{bmatrix} 5 \\ 2 \\ -1 \\ 3 \end{bmatrix} \qquad t = \begin{bmatrix} e^5 \\ e^2 \\ e^{-1} \\ e^3 \end{bmatrix}$$

"Soft max"

$$a^{[L]} = g^{[L]}(z^{[L]}) = \begin{bmatrix} e^5/(e^5+e^2+e^{-1}+e^3) \\ e^2/(e^5+e^2+e^{-1}+e^3) \\ e^{-1}/(e^5+e^2+e^{-1}+e^3) \\ e^3/(e^5+e^2+e^{-1}+e^3) \end{bmatrix} = \begin{bmatrix} 0.842 \\ 0.042 \\ 0.002 \\ 0.114 \end{bmatrix}$$

C = 4 $\quad g^{[L]}(\cdot)$

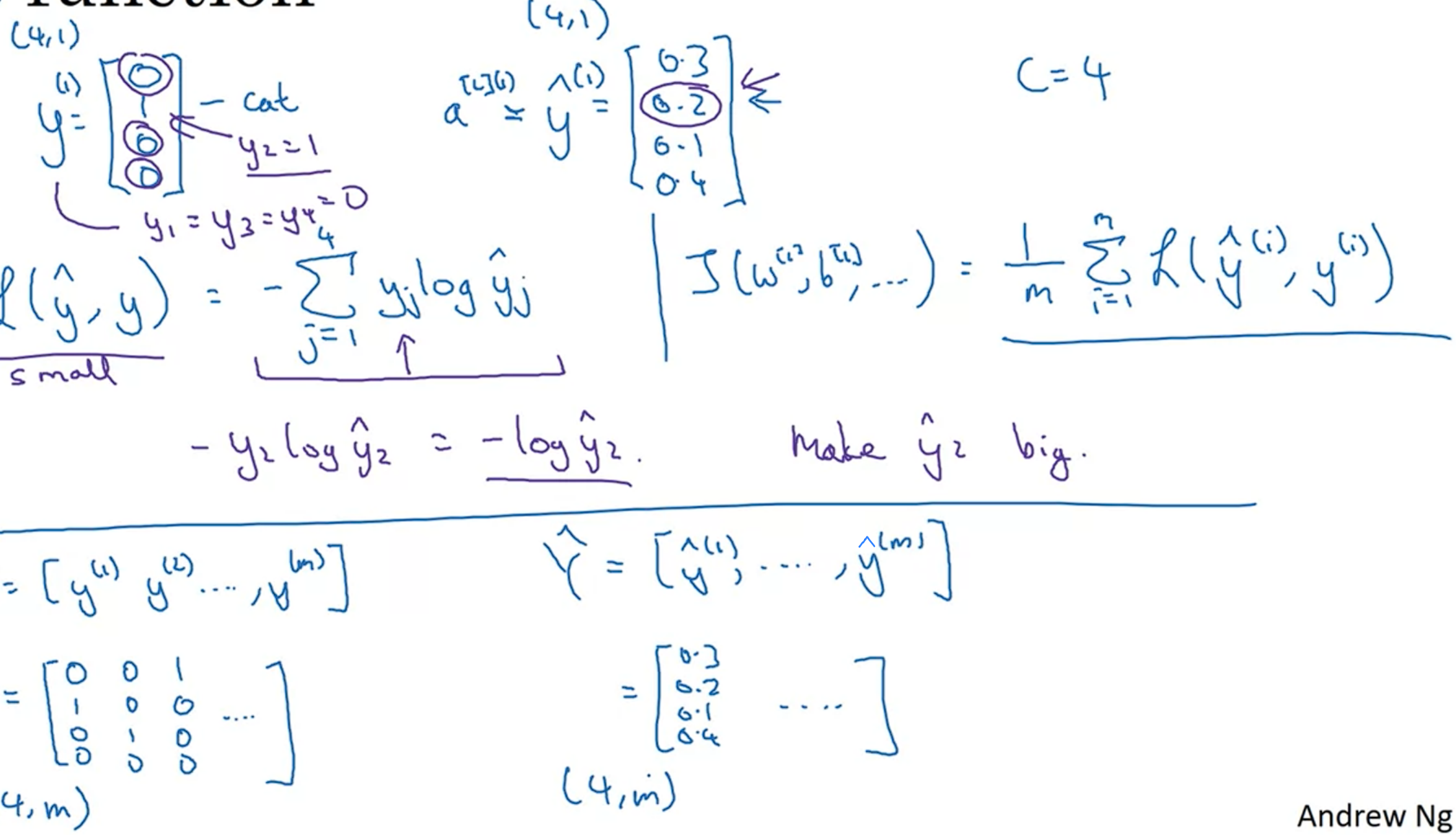
"hard max"

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Softmax regression generalizes logistic regression to C classes.

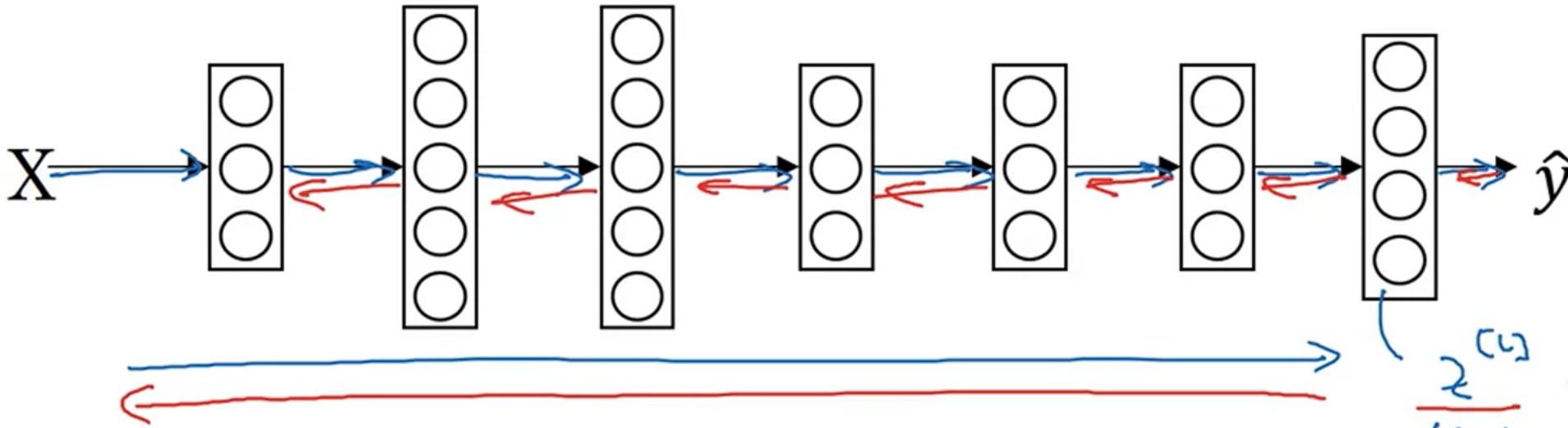
If C=2, softmax reduces to logistic regression. $a^{[L]} = \begin{bmatrix} 0.842 \\ 0.158 \end{bmatrix}$

Loss function



Andrew Ng

Gradient descent with softmax



$z^{[L]}$
$(4,1)$

$a^{[L]} = \hat{y} \rightarrow \mathcal{L}(\hat{y}, y)$

Backprop:

$dz^{[L]} = \hat{y} - y$
$(4,1)$

$\frac{\partial J}{\partial z^{[L]}}$

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

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

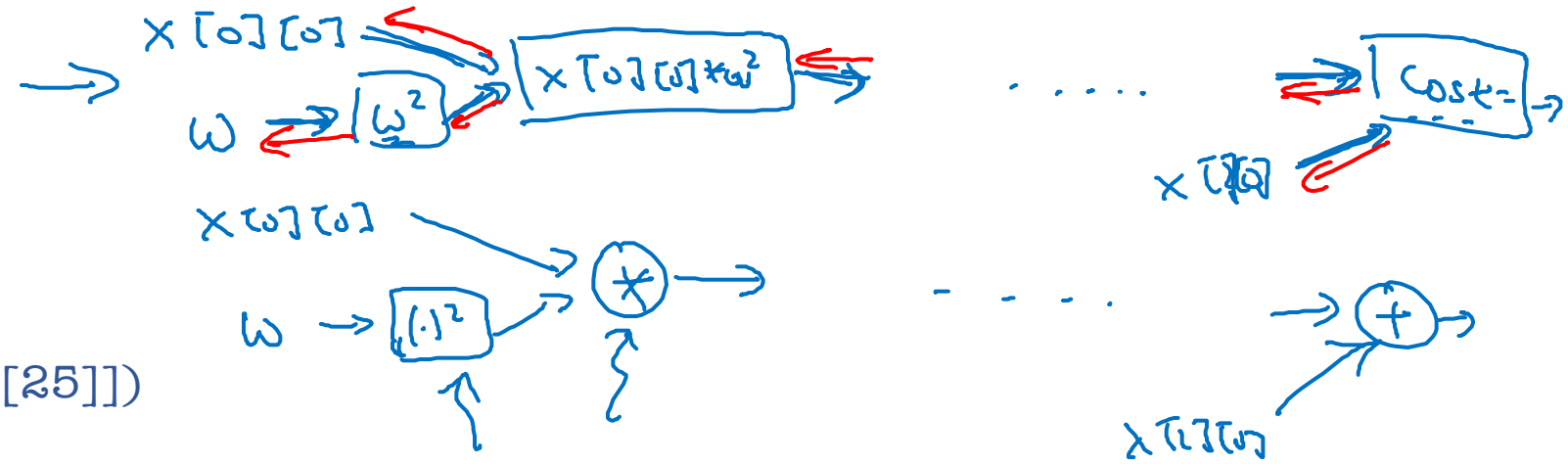
(cost)

$(w-5)^2$

$w = 5$

$J(w, b)$

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