Hyperparameter tuning

Tuning process

deeplearning.ai

Hyperparameters

$\rightarrow$ $\boxed{\alpha}$

$\boxed{\beta}$ ≈ 0.9

$\beta_1, \beta_2, \varepsilon$
$_{0.9}$ $_{0.999}$ $_{10^{-8}}$

$\boxed{\# \text{layers}}$

$\boxed{\# \text{hidden units}}$

$\boxed{\text{learning rate decay}}$

$\boxed{\text{mini-batch size}}$

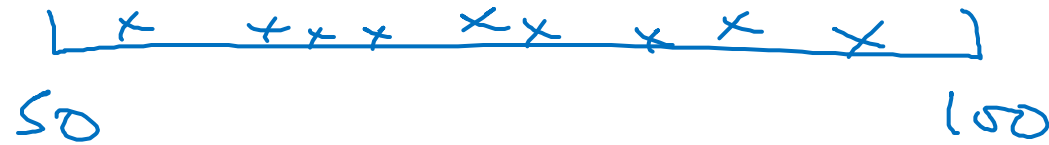# Try random values: Don't use a grid

Andrew Ng

Coarse to fine

Andrew Ng
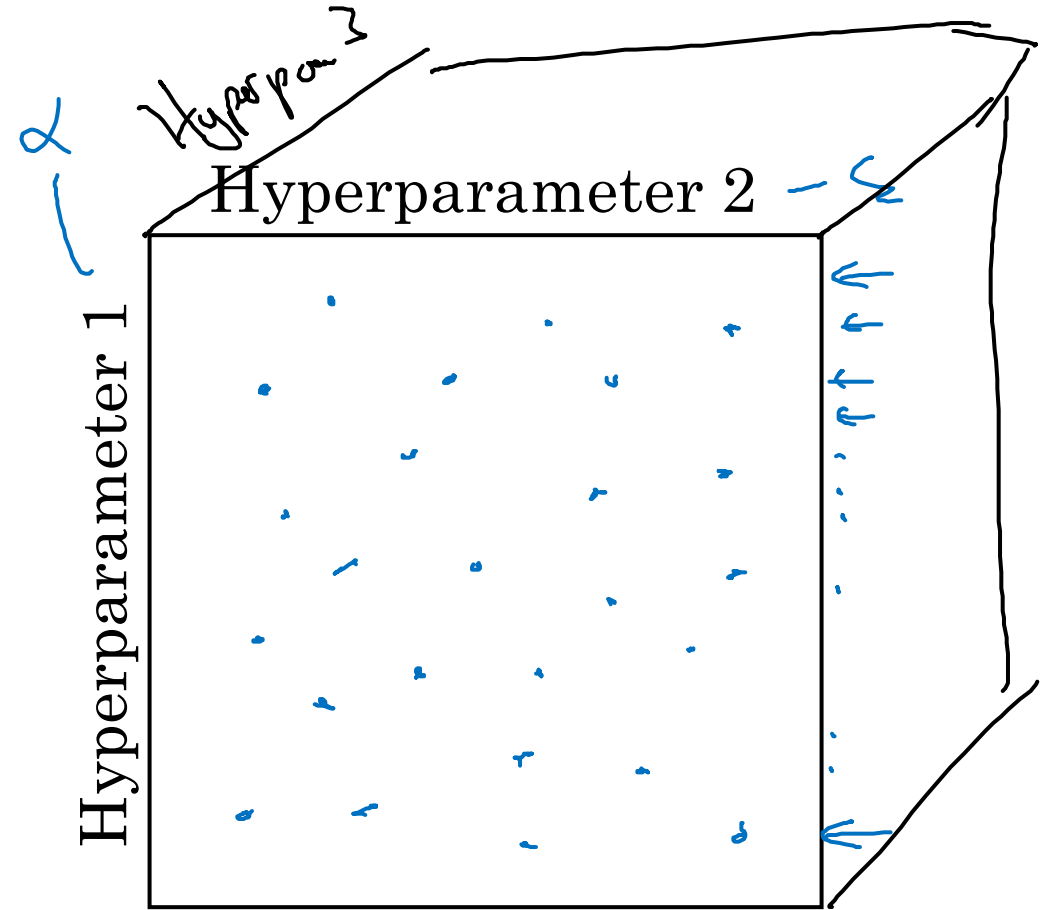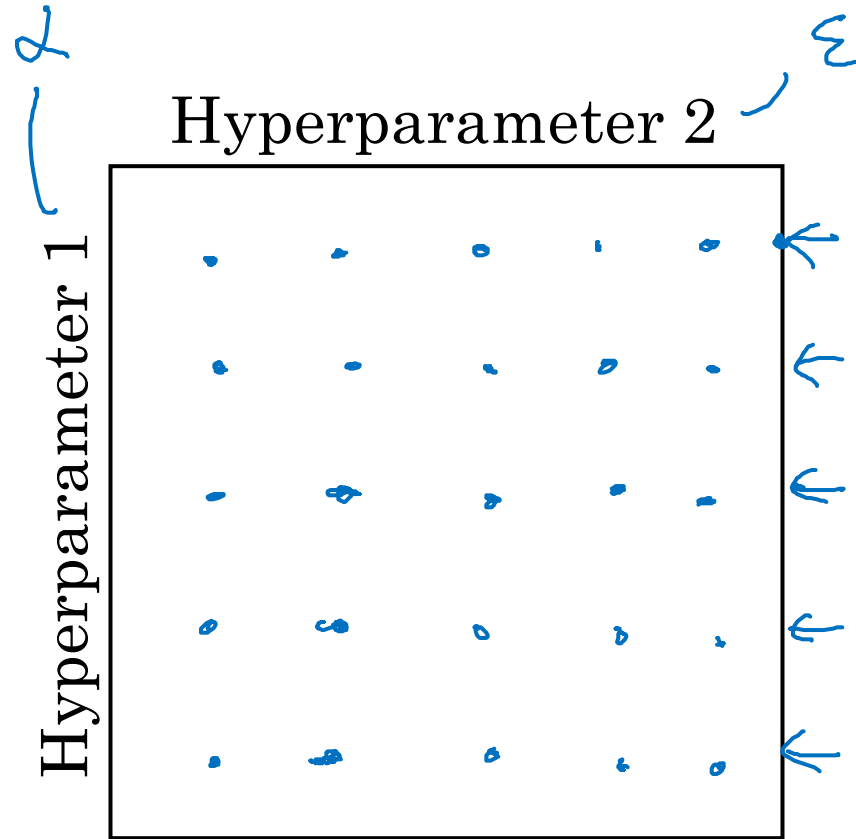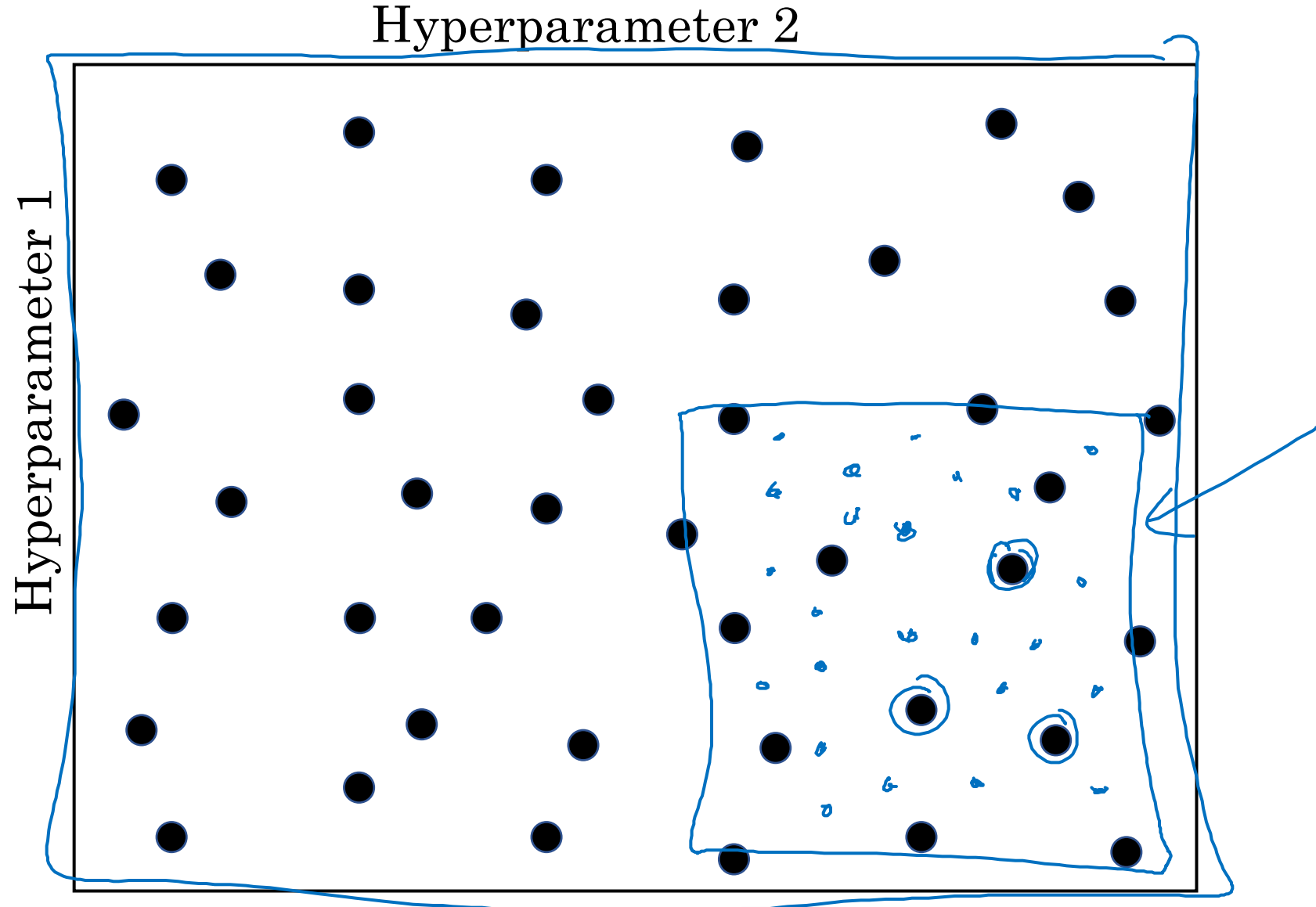
deeplearning.ai

Hyperparameter tuning

Using an appropriate scale to pick hyperparameters

Picking hyperparameters at random

$\rightarrow n^{[\ell]} = 50, \ldots, 100$

$$\underbrace{[\times \quad \times\times\times \quad \times\times \quad \times \quad \times \quad \times]}_{50 \qquad\qquad\qquad\qquad\qquad 100}$$
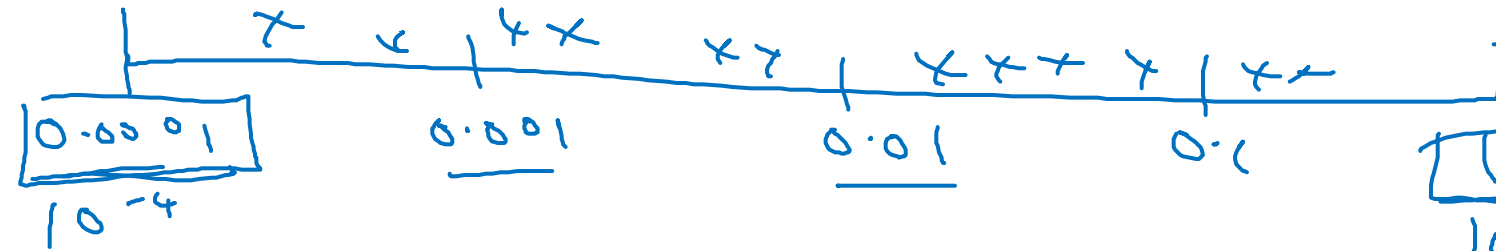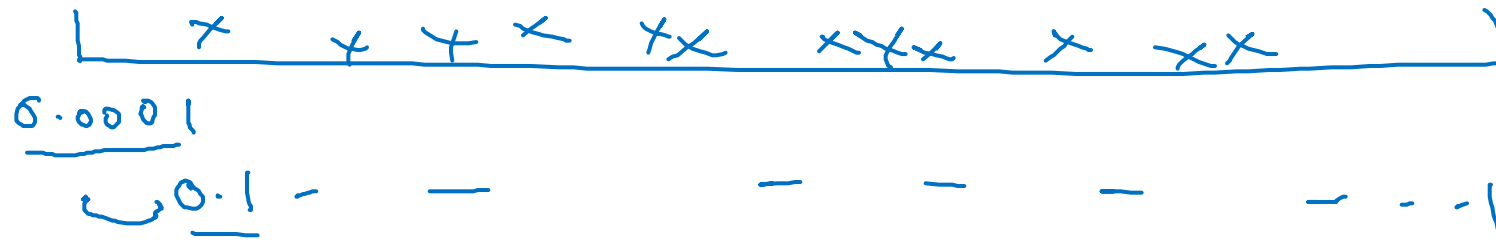
$\rightarrow$ #layers $\quad L: \quad 2 - 4$

$$2, 3, 4$$

Appropriate scale for hyperparameters

$\alpha = 0.0001, \ldots, 1$

10^a

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

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

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

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

$10^a \ldots 10^b$ $\dfrac{r \in [a, b]}{[-4, 0]}$ $\alpha = 10^r$

Hyperparameters for exponentially weighted averages

$\beta = 0.9 \quad .... \quad 0.999$

$\downarrow \qquad\qquad\qquad \downarrow$

$10 \qquad\qquad\qquad 1000$

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

$\beta : 0.9000 \rightarrow 0.9005 \quad\} \sim 10$

$\beta : 0.999 \rightarrow 0.9995$

$\sim 1000 \qquad\qquad \sim 2000$

$\dfrac{1}{1-\beta}$

0.9 $\qquad\qquad\qquad$ 0.999

0.9 $\qquad$ 0.99 $\qquad$ 0.999

0.1 $\qquad$ 0.01 $\qquad$ 0.001

$10^{-1} \qquad\qquad\qquad 10^{-3}$
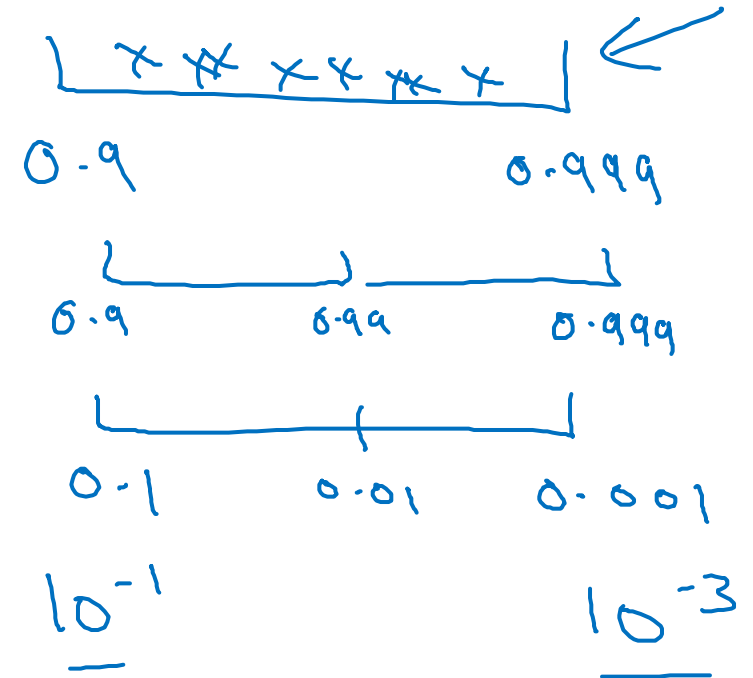
$r \in [-3, -1]$

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

$\beta = 1 - 10^{r}$
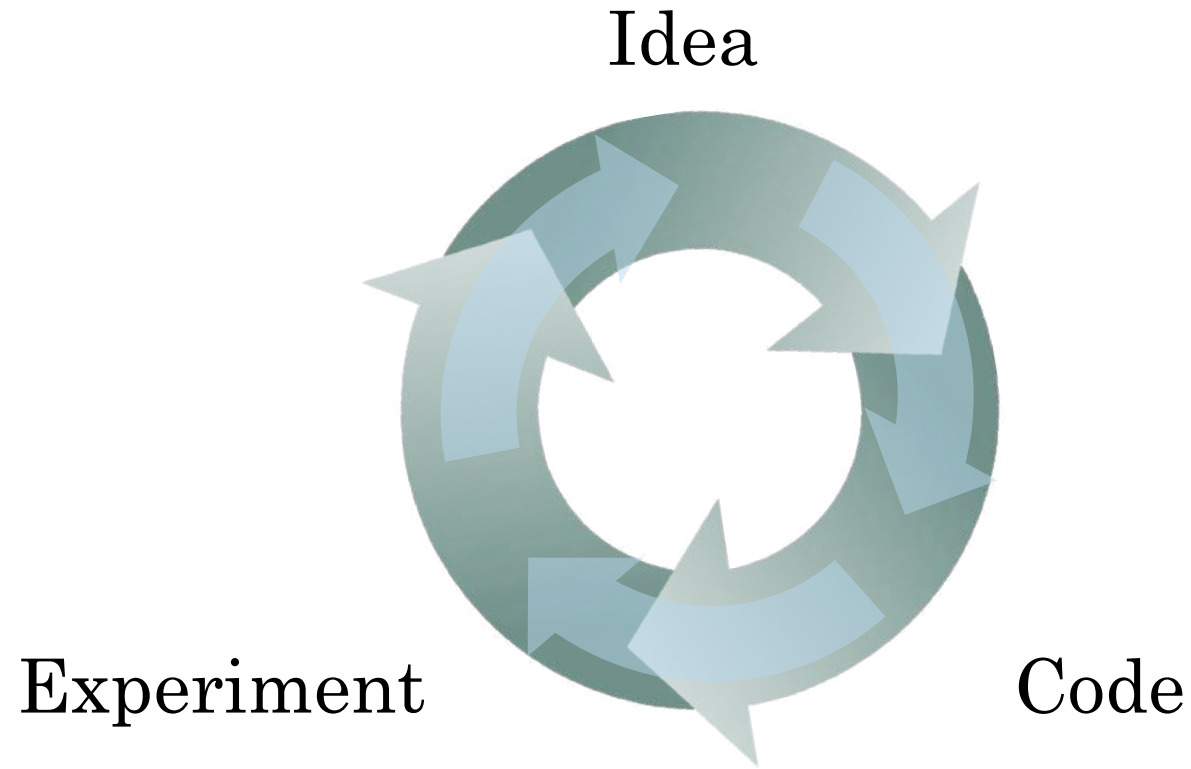
deeplearning.ai

Hyperparameters tuning

Hyperparameters tuning in practice: Pandas vs. Caviar
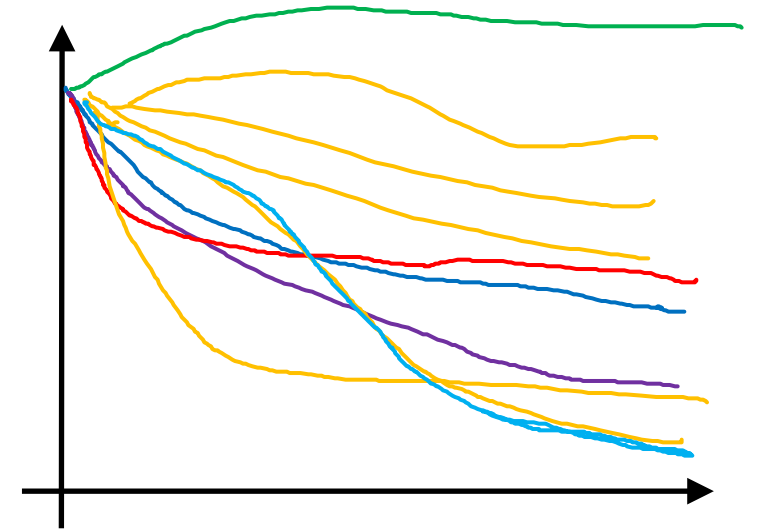
Re-test hyperparameters occasionally

Idea

Code

Experiment

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

- Intuitions do get stale. Re-evaluate occasionally.

Andrew Ng

Babysitting one model

day 0 day 1 day 2 3
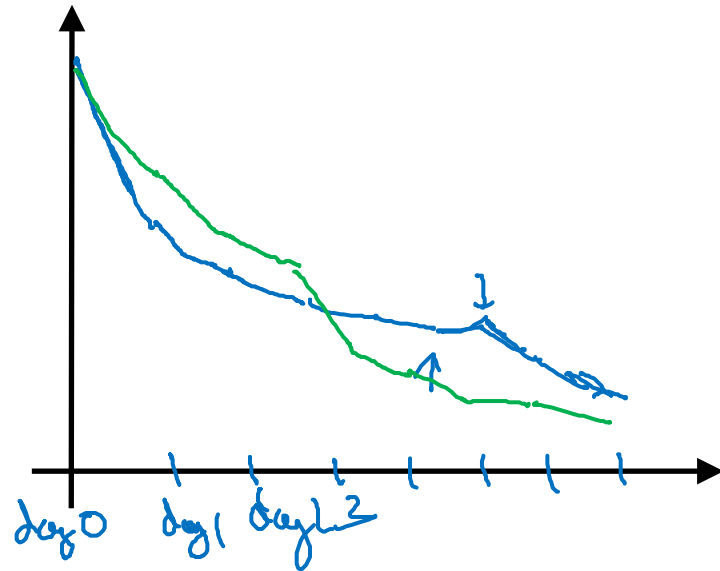
Panda ←

Training many models in parallel

Caviar ←

Andrew Ng

Batch
Normalization

Normalizing activations
in a network

deeplearning.ai
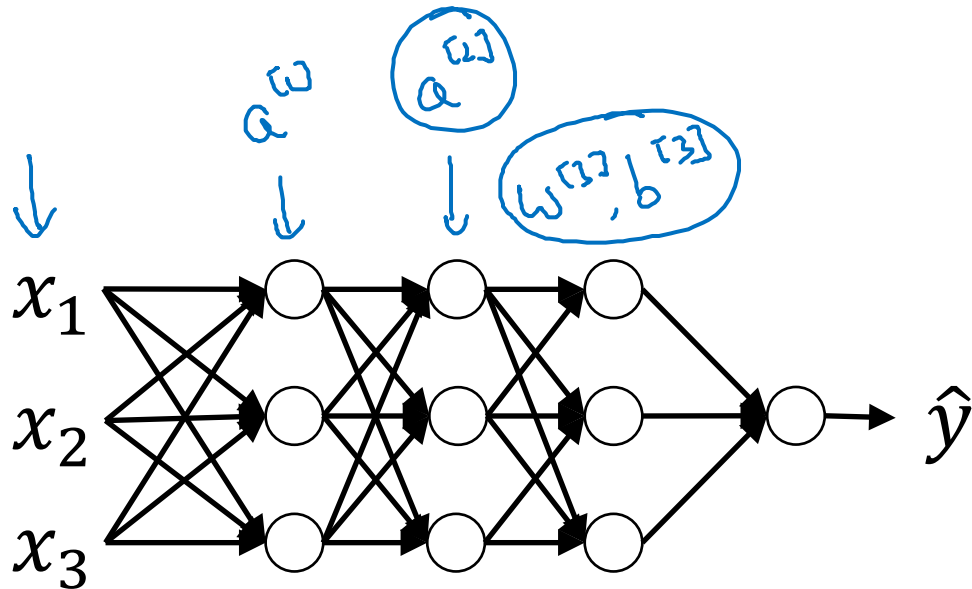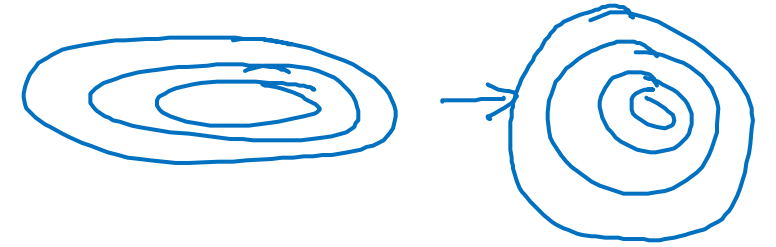
Normalizing inputs to speed up learning

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

$$X = X - \mu$$
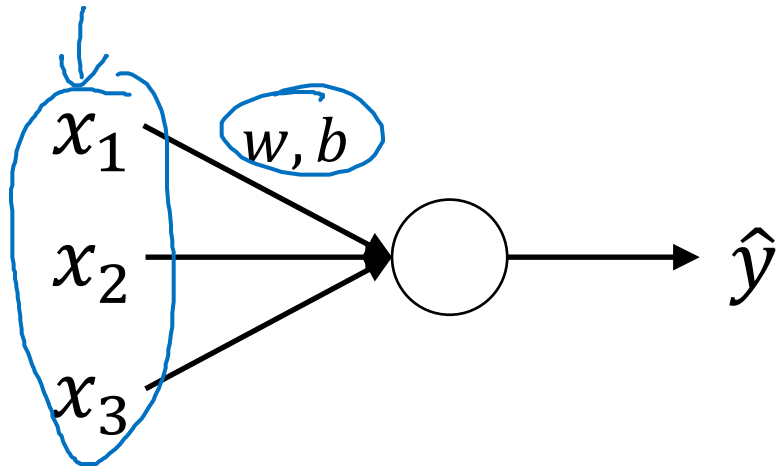
$$\sigma^2 = \frac{1}{m} \sum_i x^{(i)2} \quad \leftarrow \text{element-wise}$$

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

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

$$z^{(i)}, \ldots, z^{(m)}$$

$$z^{[l](i)}$$

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

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

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

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

learnable parameters of model.

If

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

$$\beta = \mu$$

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

X

$z^{(i)}$

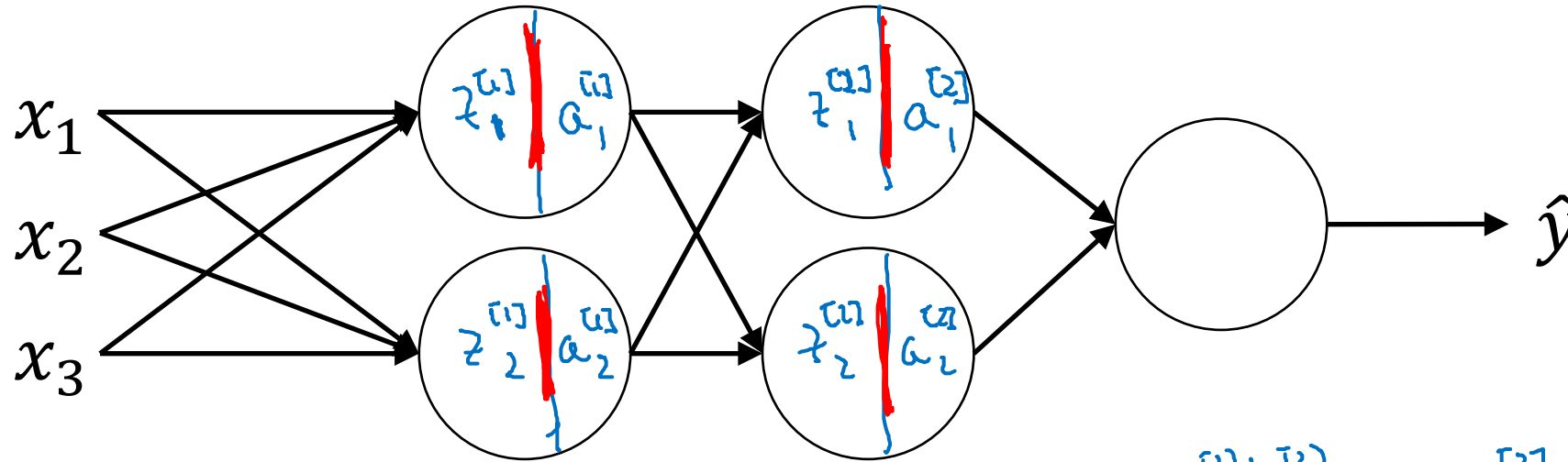Use $\tilde{z}^{[l](i)}$ instead of $z^{[l](i)}$.

Andrew Ng

Batch
Normalization

Fitting Batch Norm
into a neural network

deeplearning.ai

Adding Batch Norm to a network

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

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

$d\beta^{[l]}$

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

$\to \beta$

tf.nn.batch-normalization

Andrew Ng

Working with mini-batches

$$X^{\{1\}} \xrightarrow{W^{[1]}, b^{[1]}} Z^{[1]} \xrightarrow[BN]{\beta^{[1]}, \gamma^{[1]}} \tilde{Z}^{[1]} \longrightarrow g^{[1]}(\tilde{Z}^{[1]}) = a^{[1]} \xrightarrow{W^{[2]}, b^{[2]}} Z^{[2]} \rightarrow \ldots$$

$$\boxed{X^{\{2\}}} \longrightarrow Z^{[1]} \xrightarrow[\boxed{BN}]{\beta^{[1]}, \gamma^{[1]}} \tilde{Z}^{[1]} \longrightarrow \ldots$$

$$X^{\{3\}} \longrightarrow \ldots$$

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

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

$Z^{[l]}$

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

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

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

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

$\rightarrow \tilde{Z}^{[l]} = \gamma^{[l]} Z^{[l]}_{norm} + \boxed{\beta^{[l]}} \leftarrow$

Andrew Ng

Implementing gradient descent

for $t = 1 \ldots$ num MiniBatches

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

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

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

   Update paramters $\left. \begin{array}{l} W^{[l]} := W^{[l]} - \alpha \, dW^{[l]} \\ \beta^{[l]} := \beta^{[l]} - \alpha \, d\beta^{[l]} \\ \gamma^{[l]} := \ldots \end{array} \right\} \leftarrow$
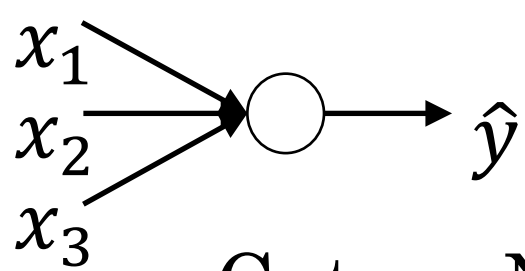
Works w/ momentum, RMSprop, Adam.

Andrew Ng

deeplearning.ai

Batch Normalization

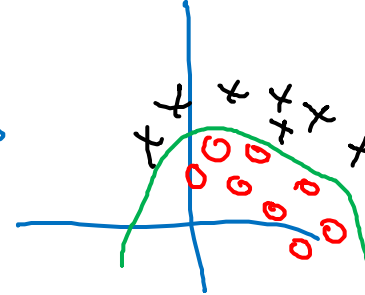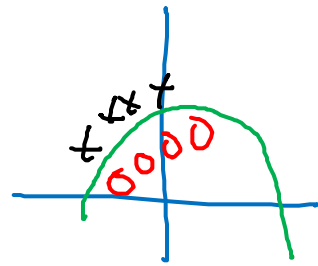Why does
Batch Norm work?

Learning on shifting input distribution

x_1
x_2
x_3

$\hat{y}$

Cat
$y = 1$

Non-Cat
$y = 0$

$y = 1$

$y = 0$

"Covariate shift"

$x \longrightarrow y$

Andrew Ng

Why this is a problem with neural networks?

$W^{[3]}, b^{[3]}$ $W^{[4]}, b^{[4]}$

$a^{[2]}_1$

$a^{[2]}_2$

$a^{[2]}_3$

$a^{[2]}_4$

$\hat{y}$

$z^{[2]}_2$

$z^{[2]}_1$

$z^{[2]}_1$

$z^{[2]}_1$

Mean 0

Variance 1

$\Rightarrow \beta^{[2]}, \gamma^{[2]}$

Andrew Ng
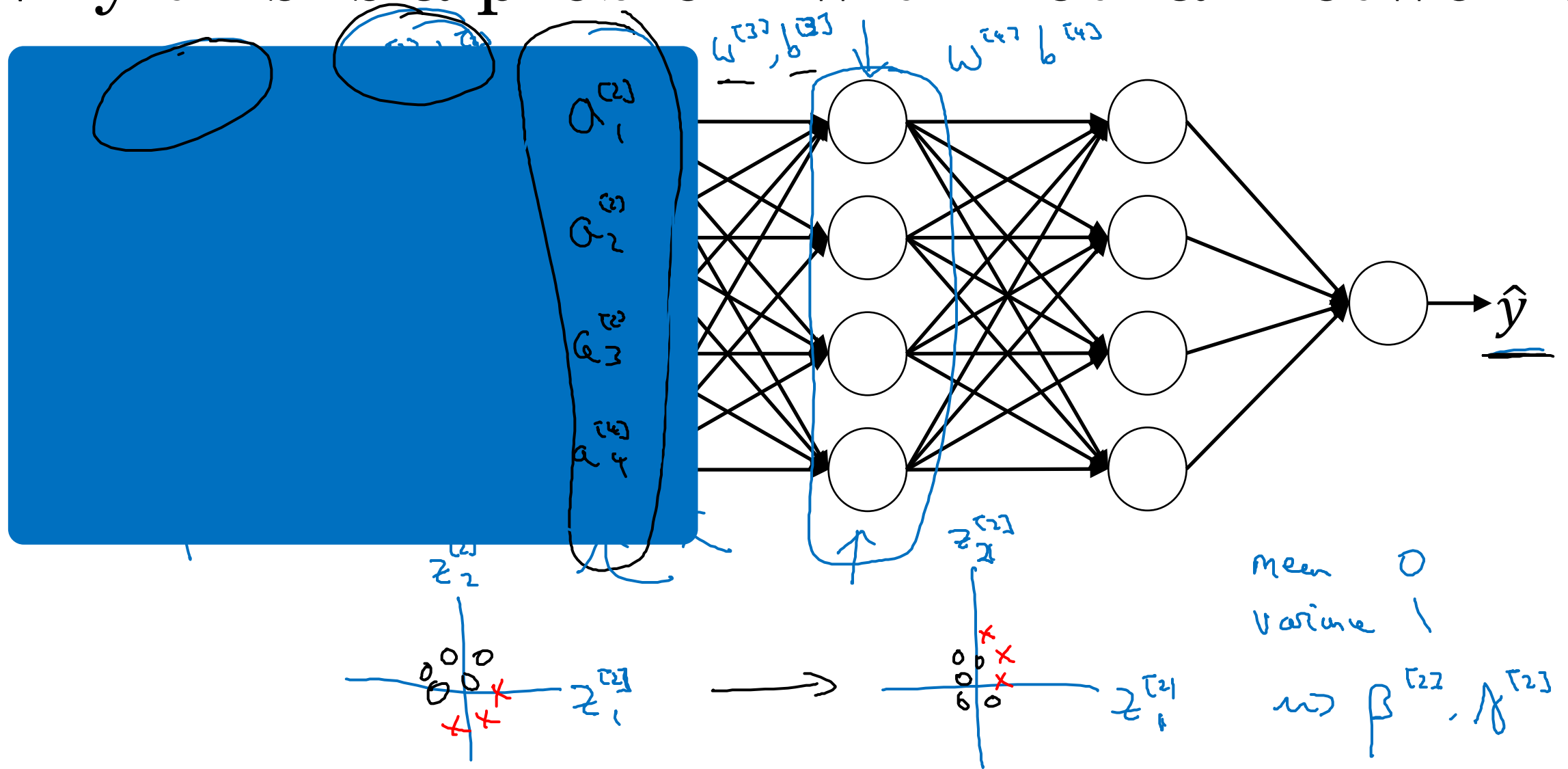
Batch Norm as regularization

- 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.

$$\rightarrow \tilde{z}^{[l]}$$

$$64, 128 \qquad z^{[l]}$$

$$X^{\{t\}}$$

$$\mu, \sigma^2$$

mini-batch : $\underline{64} \longrightarrow \underline{512}$

Multi-class classification

Softmax regression

deeplearning.ai

Recognizing cats, dogs, and baby chicks

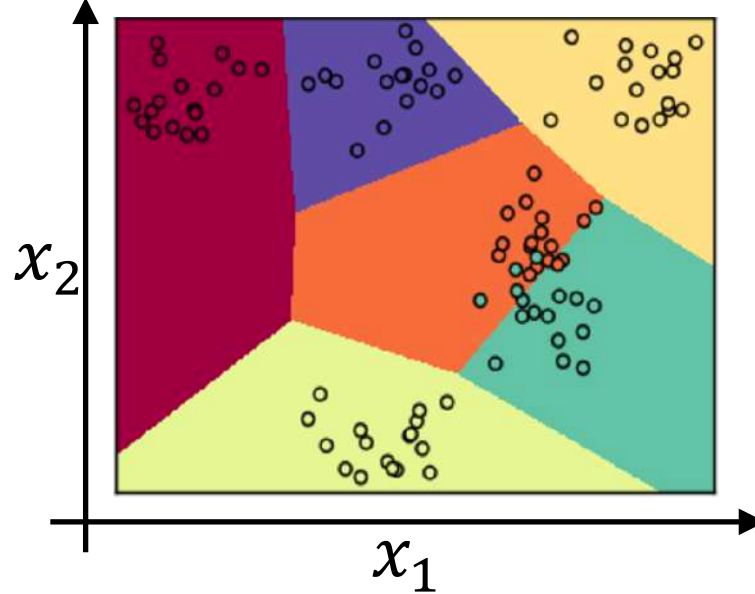3   1   2   0   3   2   0   1

$$X \rightarrow \hat{y}$$
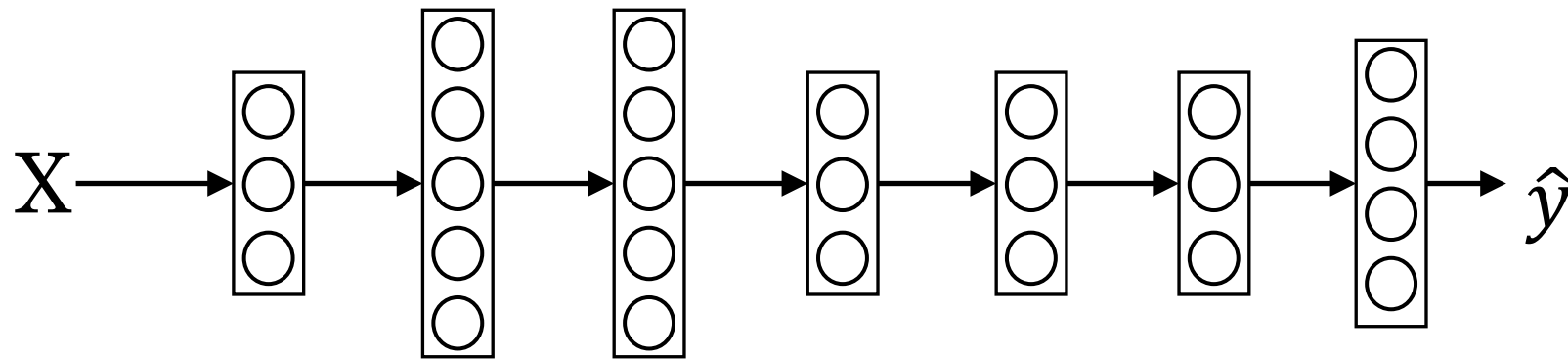
Softmax layer
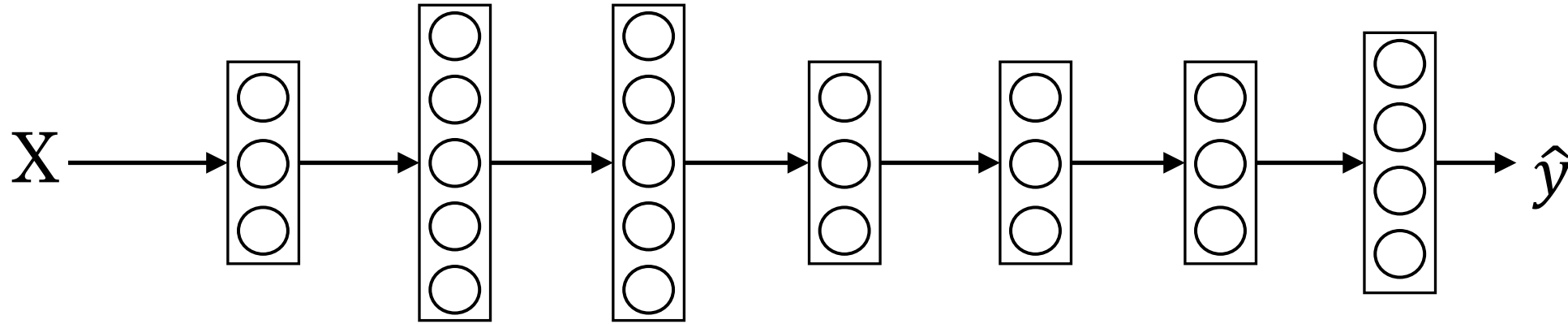
Softmax examples
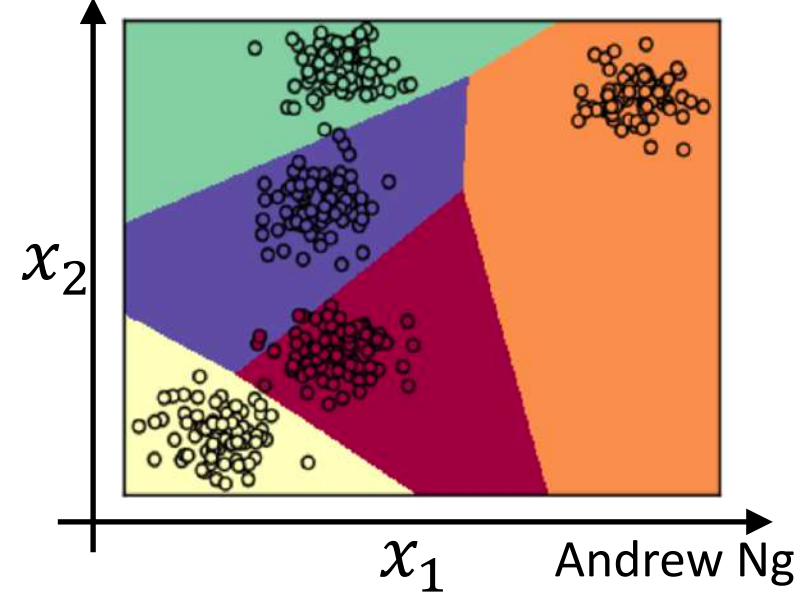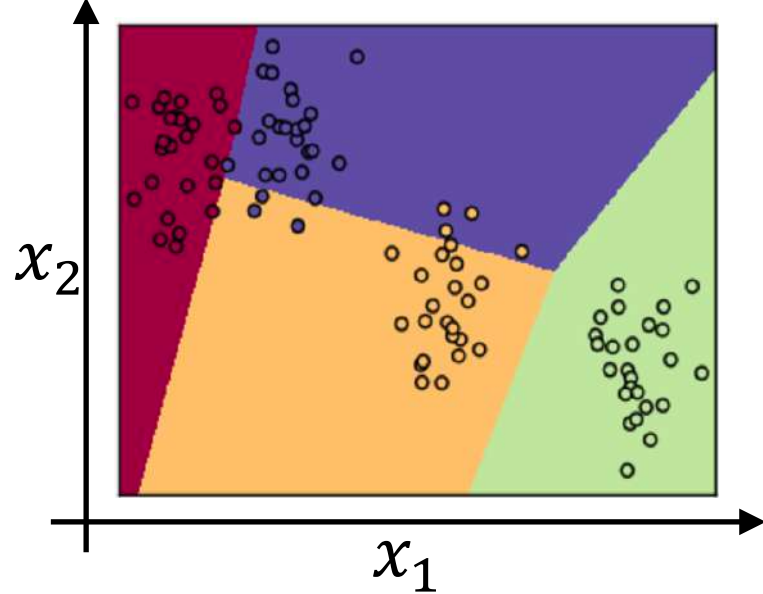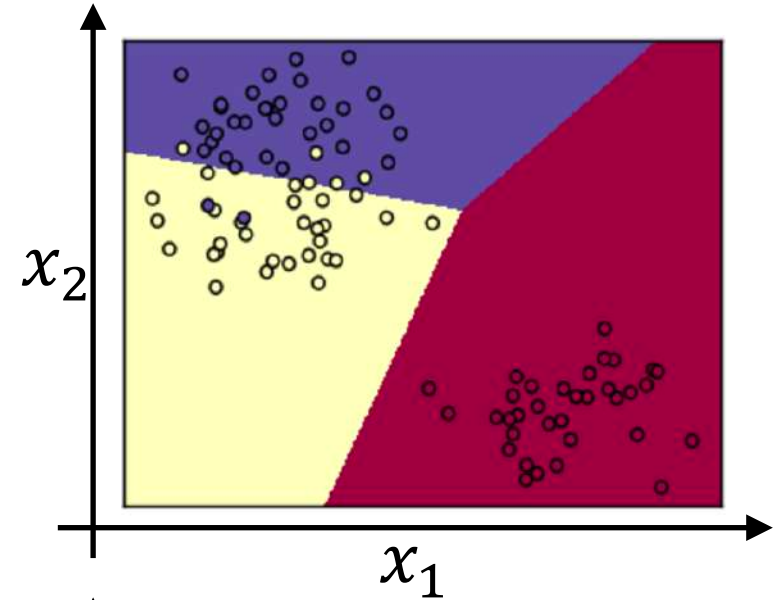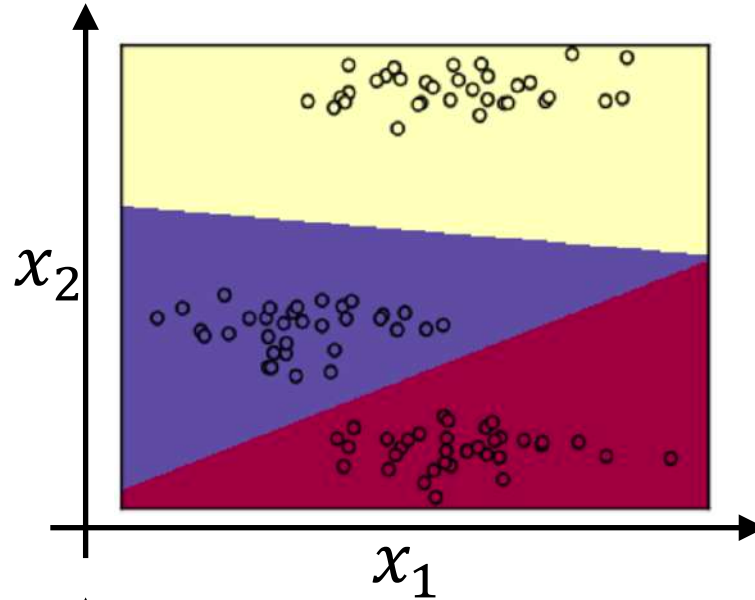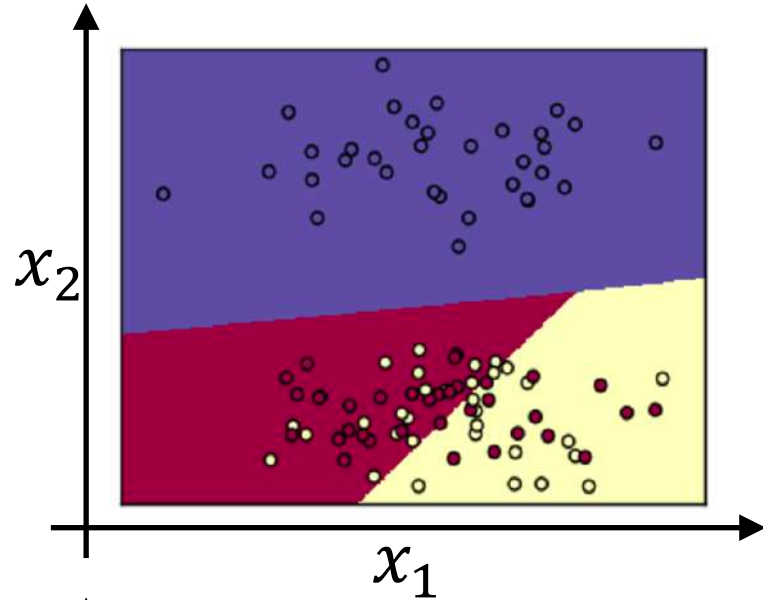
Programming
Frameworks

Deep Learning
frameworks

deeplearning.ai

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)

Programming
Frameworks
_____

TensorFlow

deeplearning.ai

Motivating problem

$$J(w) = \boxed{w^2 - 10w + 25}$$

(cost)

$$(w-5)^2$$

$$w = 5$$

$$J(W, b)$$

Code example

```python
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))
```

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

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

Andrew Ng