Copyright Notice

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}

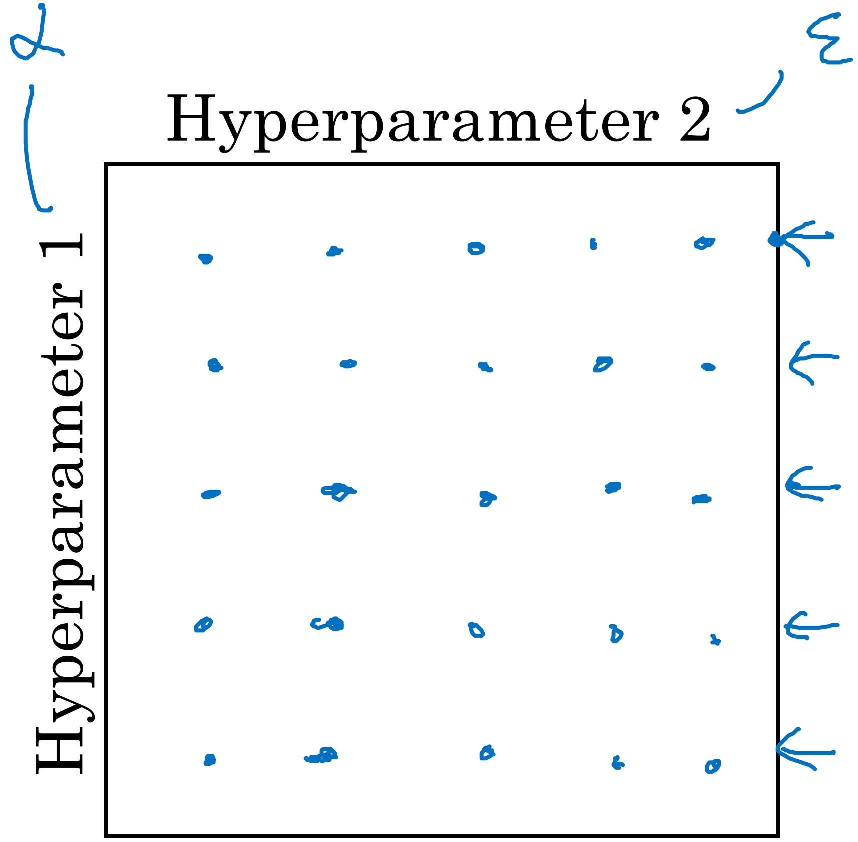
layers

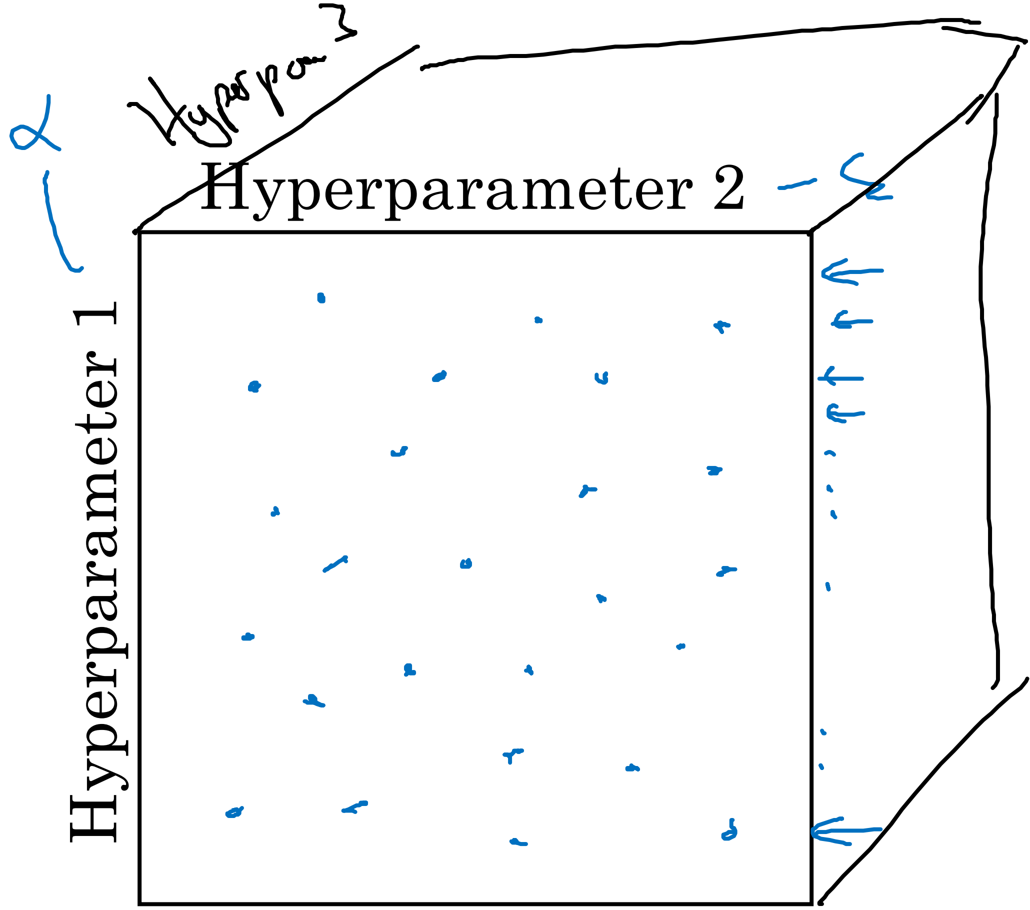
hidden units

learning rate decay

mini-batch size

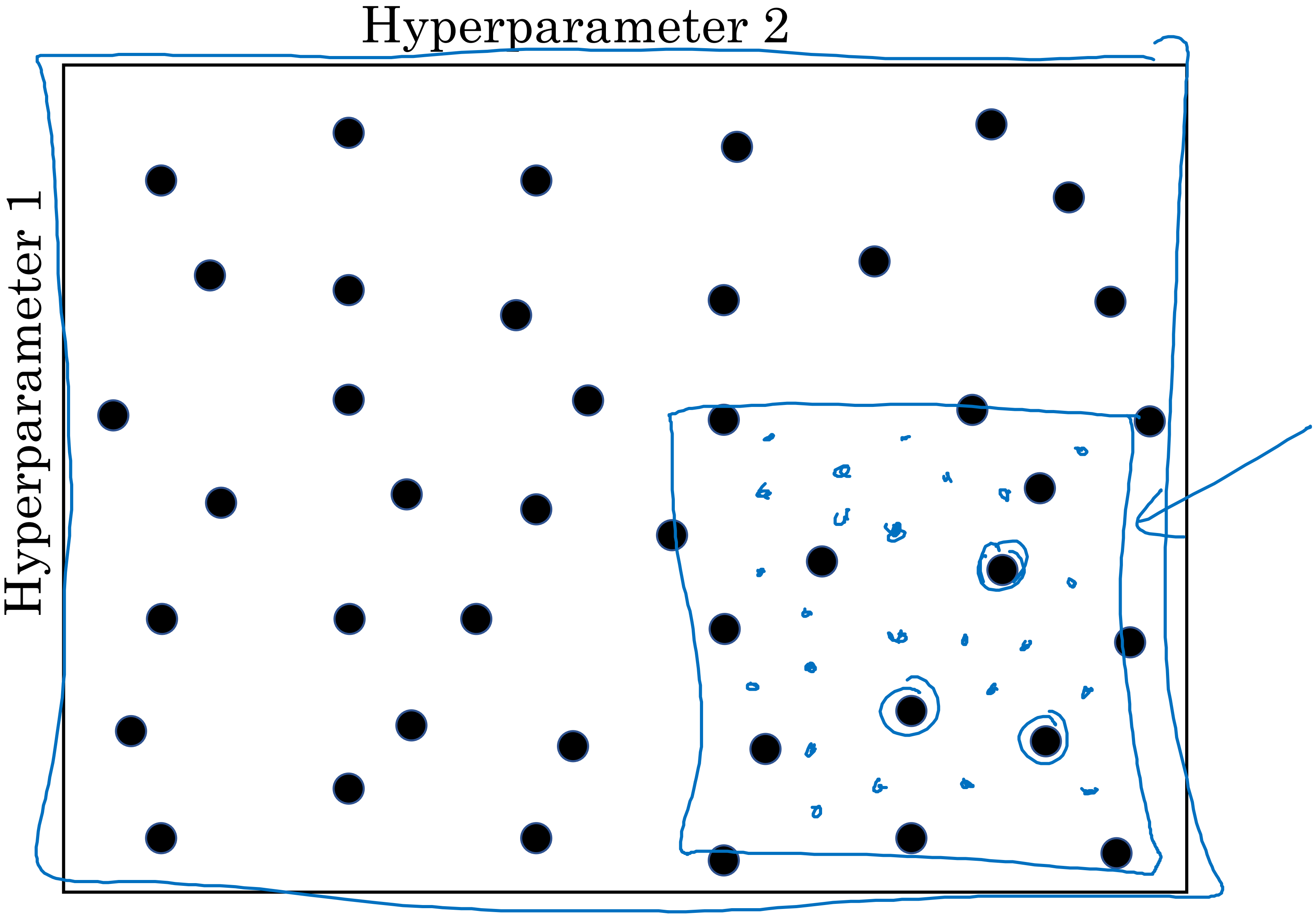
Try random values: Don't use a grid





Andrew Ng

Coarse to fine



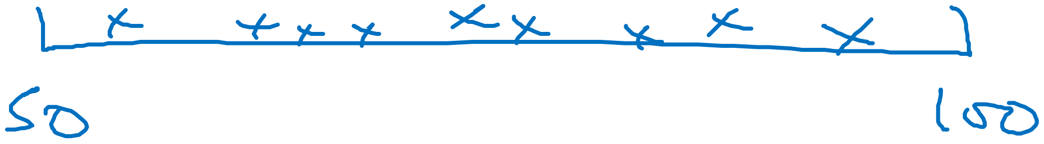
deeplearning.ai

Hyperparameter tuning

Using an appropriate scale to pick hyperparameters

Picking hyperparameters at random

→ $n^{[l]} = 50, \ldots, 100$

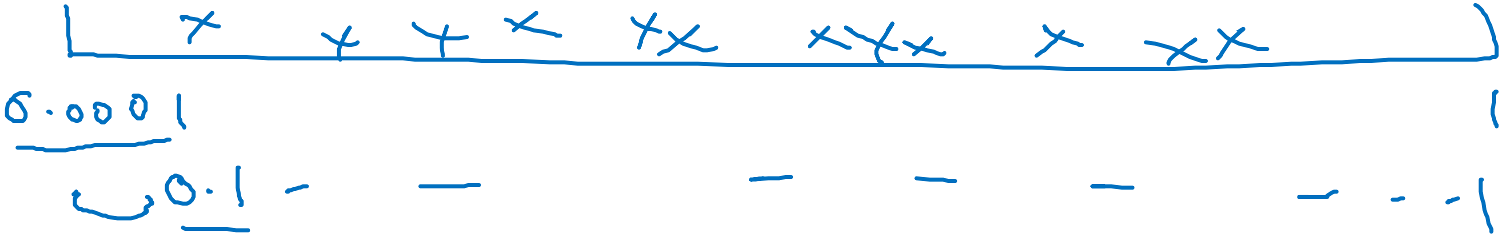


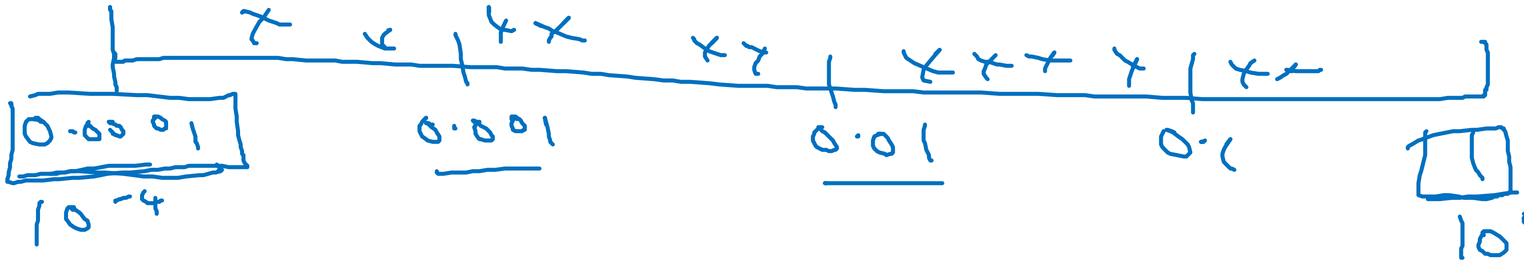
→ #layers $L: 2 - 4$

$2, 3, 4$

Appropriate scale for hyperparameters

$\alpha = 0.0001, \ldots\ldots, 1$





10^a $\qquad$ 10^b

$a = \log_{10} 0.0001 = -4$ $\qquad$ $b = \log_{10} 1 = 0$

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

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

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

Hyperparameters for exponentially weighted averages

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

$\quad\;\; 10 \qquad\quad 1000$

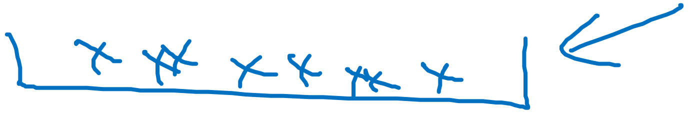
$1-\beta = 0.1 \quad \ldots \quad 0.001$

$\beta: 0.9000 \to 0.9005 \quad \Big\} \sim 10$

$\beta: 0.999 \to 0.9995$

$\quad\;\; \sim 1000 \qquad \sim 2000$

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



$0.9 \qquad\qquad 0.999$

$0.9 \qquad 0.99 \qquad 0.999$

$0.1 \qquad 0.01 \qquad 0.001$

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

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

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

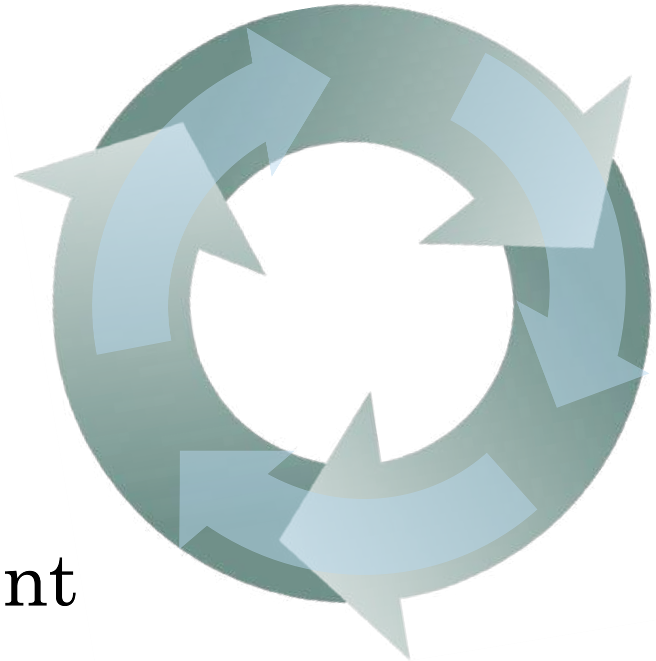
$\beta = 1 - 10^{r}$

deeplearning.ai

Hyperparameters tuning

Hyperparameters tuning in practice: Pandas vs. Caviar

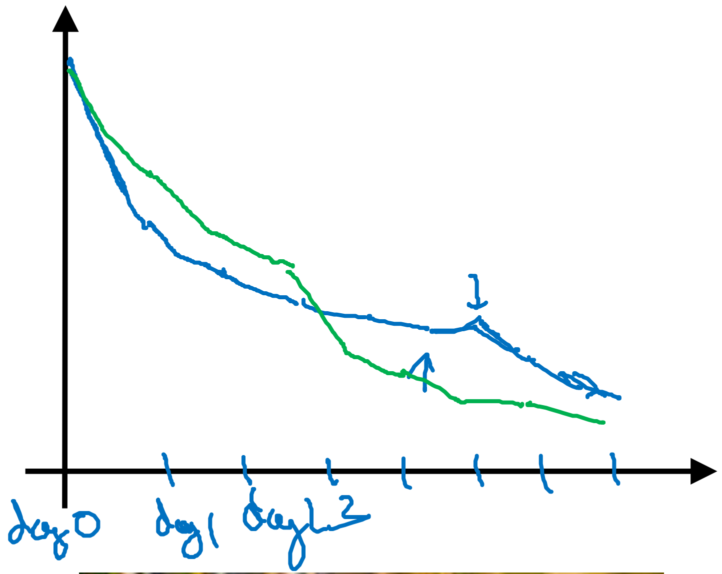
Re-test hyperparameters occasionally



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

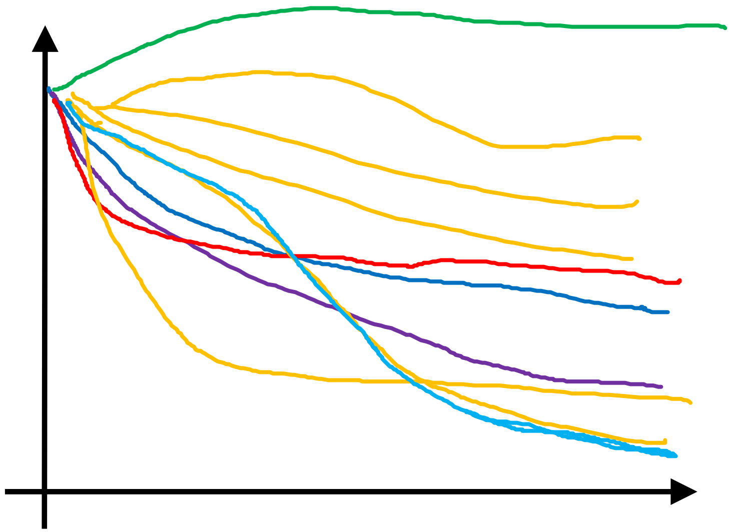
- Intuitions do get stale. Re-evaluate occasionally.

Babysitting one model



Panda

Training many models in parallel



Caviar

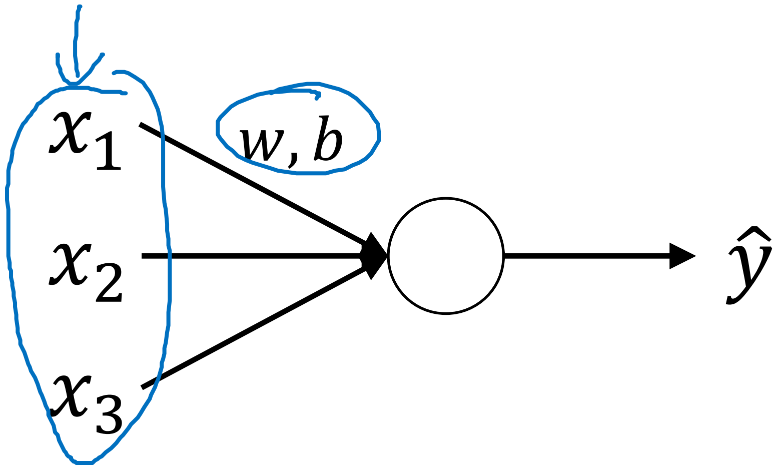
Andrew Ng

deeplearning.ai

Batch Normalization

Normalizing activations in a network

Normalizing inputs to speed up learning

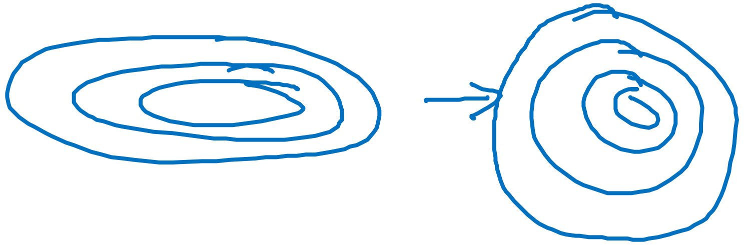


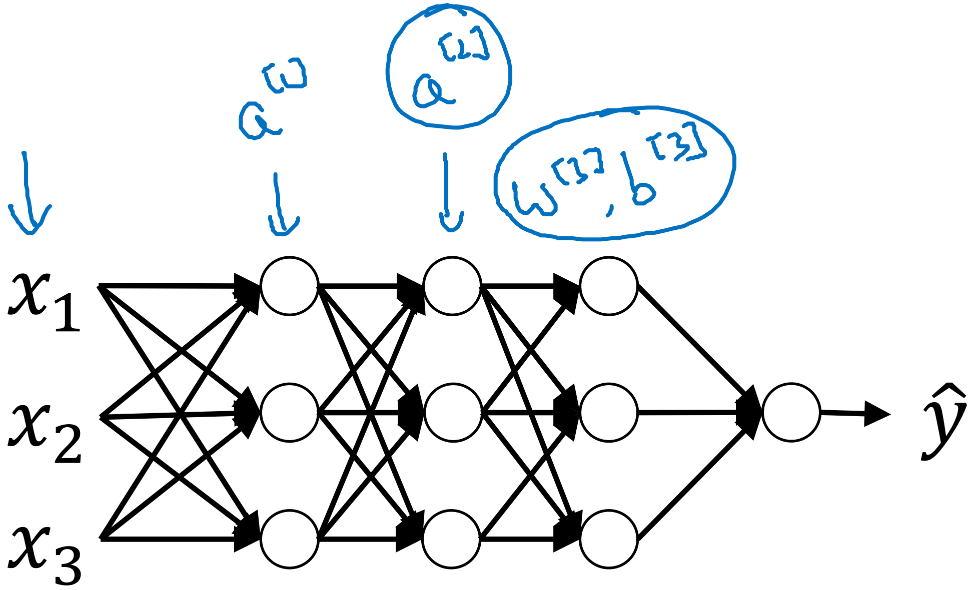
$\mu = \frac{1}{m} \sum_i x^{(i)}$

$X = X - \mu$ ← 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 $\underbrace{z^{(1)}, \ldots, z^{(m)}}$ $z^{[l](i)}$

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

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

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

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

learnable parameters of model.

If

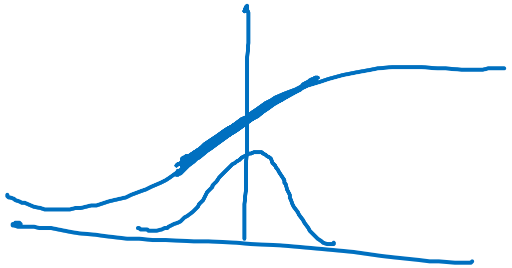
$$\gamma = \boxed{\sqrt{\sigma^2 + \varepsilon}}$$

$$\beta = \boxed{\mu}$$

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

X

$z^{(i)}$



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

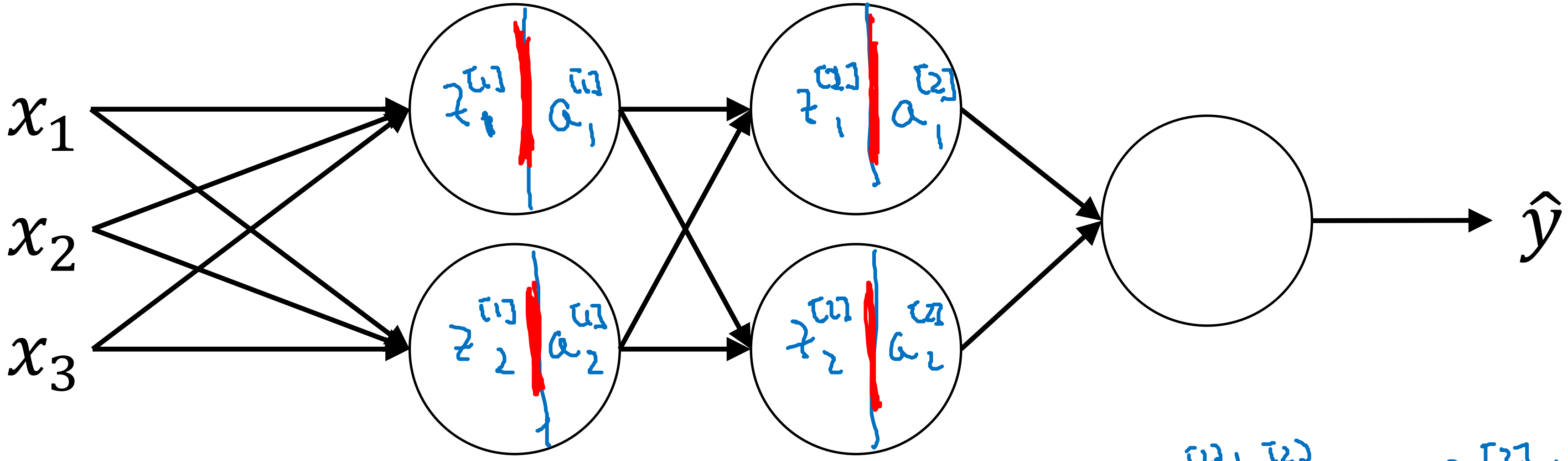
Andrew Ng

deeplearning.ai

Batch Normalization

Fitting Batch Norm into a neural network

Adding Batch Norm to a network



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

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

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

$d\beta^{[l]}$

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

tf.nn.batch_normalization $\Leftarrow$

$\rightarrow \beta$

Andrew Ng

Working with mini-batches

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

$$X^{\{2\}} \rightarrow z^{[1]} \xrightarrow[BN]{\beta^{[1]},\, \gamma^{[1]}} \tilde{z}^{[1]} \rightarrow \cdots$$

$$X^{\{3\}} \rightarrow \cdots$$

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

$b^{[l]}$: $(n^{[l]}, 1)$ $\qquad \beta^{[l]}$: $(n^{[l]}, 1)$ $\qquad \gamma^{[l]}$: $(n^{[l]}, 1)$

$z^{[l]}$

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

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

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

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

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

Implementing gradient descent

for $t = 1 \ldots$ numMiniBatches

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

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

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

Update params $\left.\begin{array}{l} W^{[l]} := W^{[l]} - \alpha\, dW^{[l]} \\ \beta^{[l]} := \beta^{[l]} - \alpha\, d\beta^{[l]} \\ \gamma^{[l]} := \ldots \end{array}\right\} \leftarrow$

Works w/ momentum, RMSprop, Adam.

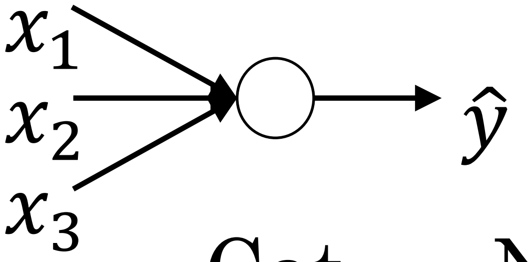
Andrew Ng

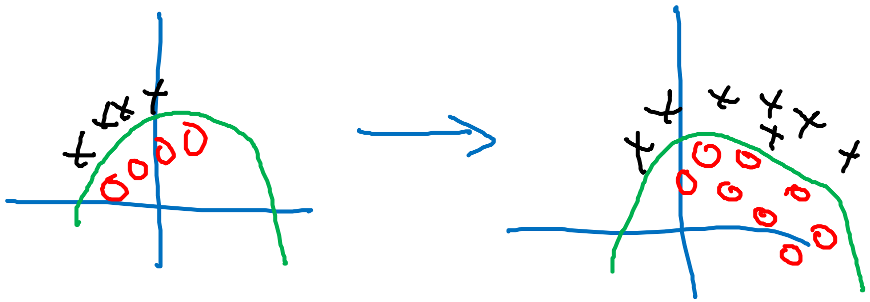
deeplearning.ai

Batch Normalization

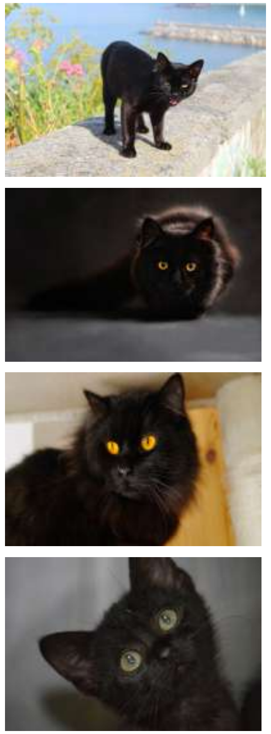
Why does Batch Norm work?

Learning on shifting input distribution





Cat	Non-Cat
$y = 1$	$y = 0$

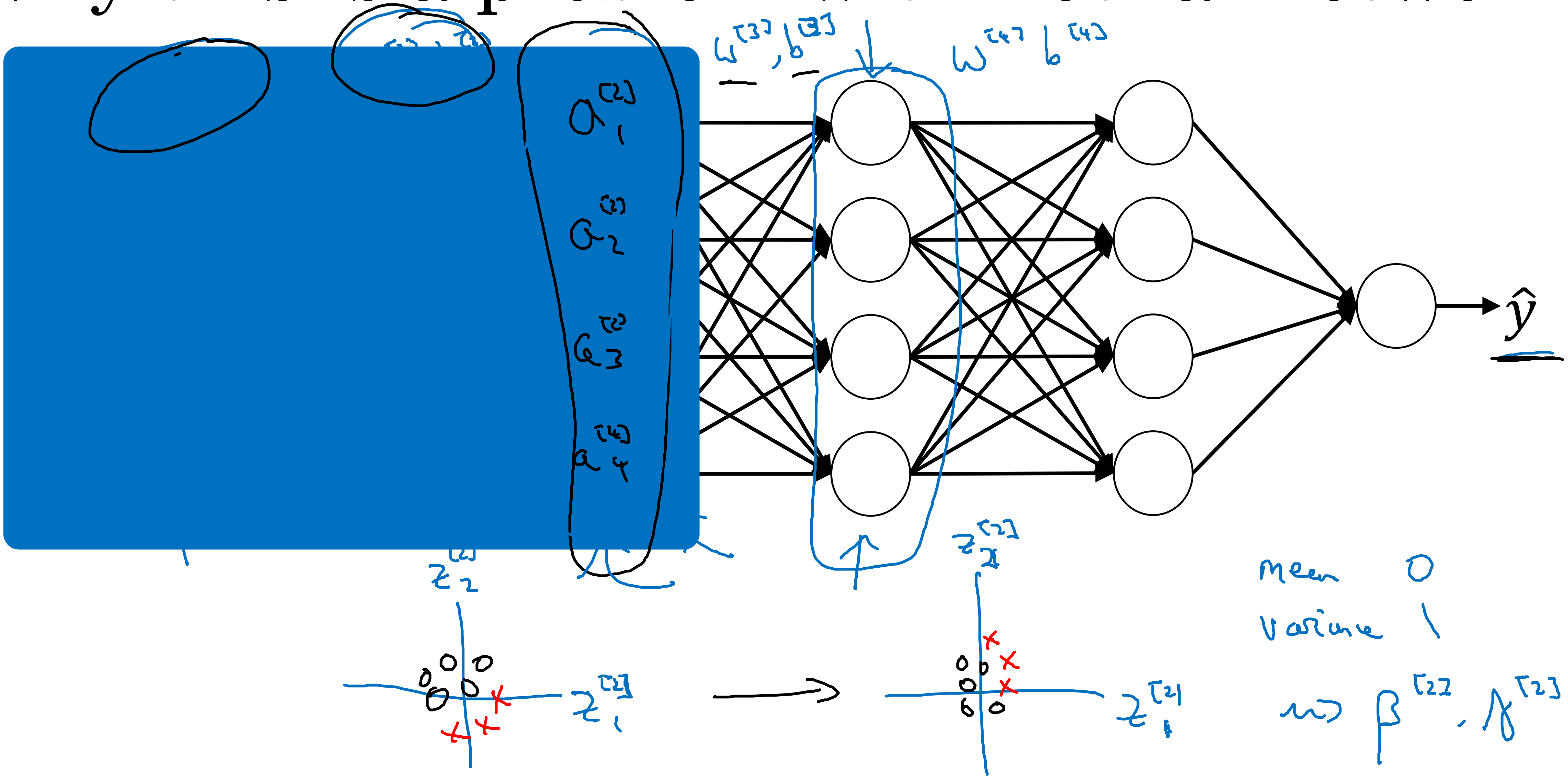


"Covariate shift"

$x \rightarrow y$

$y = 1$	$y = 0$

Why this is a problem with neural networks?



Andrew Ng

Batch Norm as regularization

X

$X^{\{t\}}$

- Each mini-batch is scaled by the mean/variance computed on just that mini-batch.

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

- 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

Multi-class classification

Softmax regression

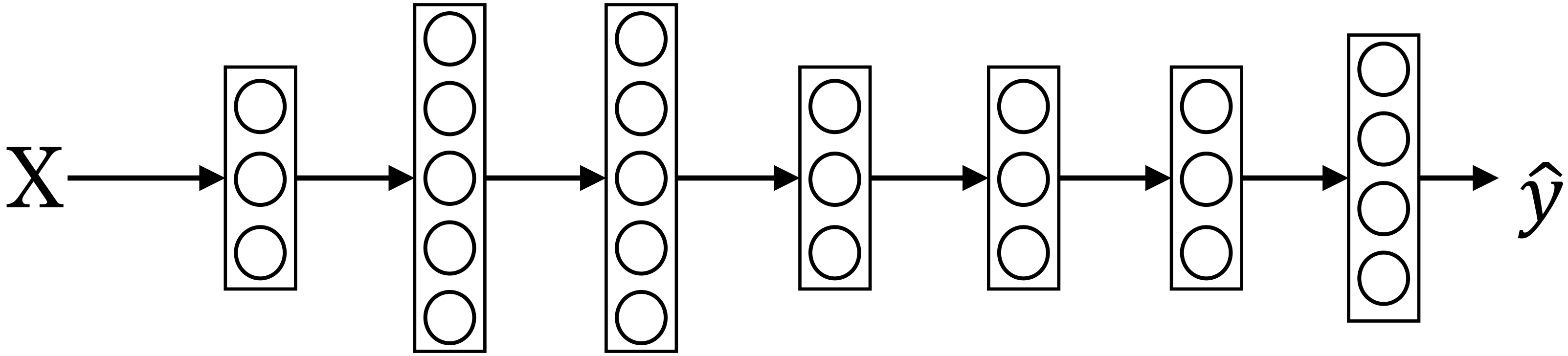
Recognizing cats, dogs, and baby chicks



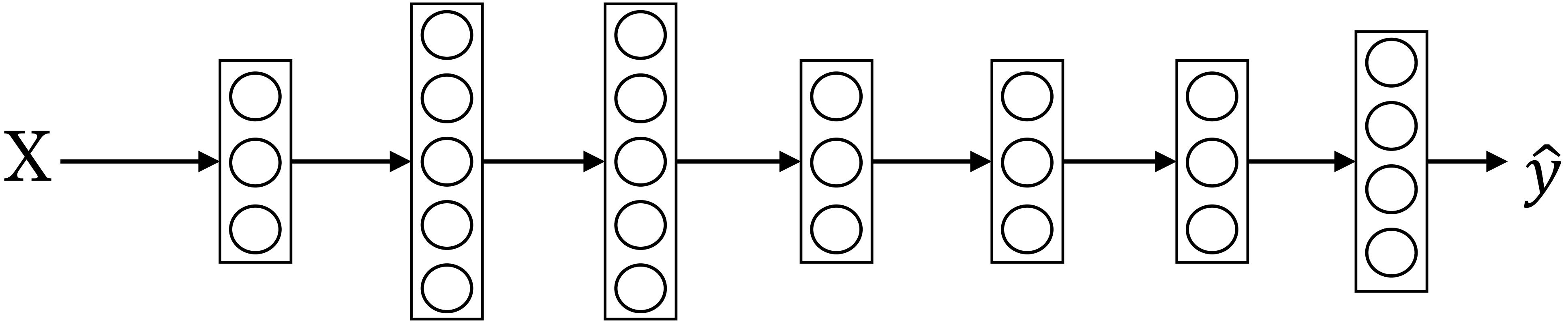




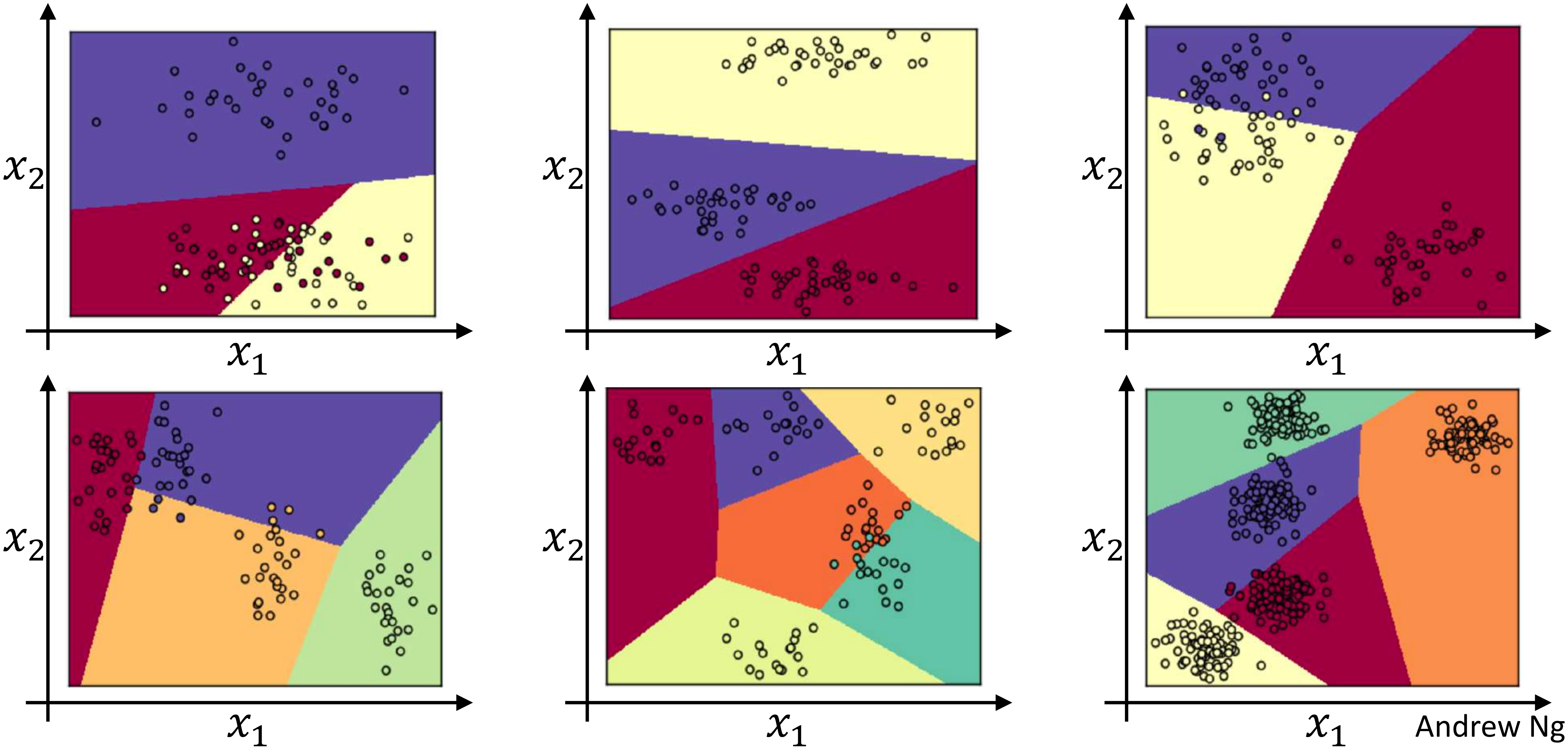

3 1 2 0 3 2 0 1



Softmax layer



Softmax examples



Andrew Ng

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

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

(cost)

$(w-5)^2$

$w = 5$

$J(w, b)$

$\uparrow \quad \uparrow$