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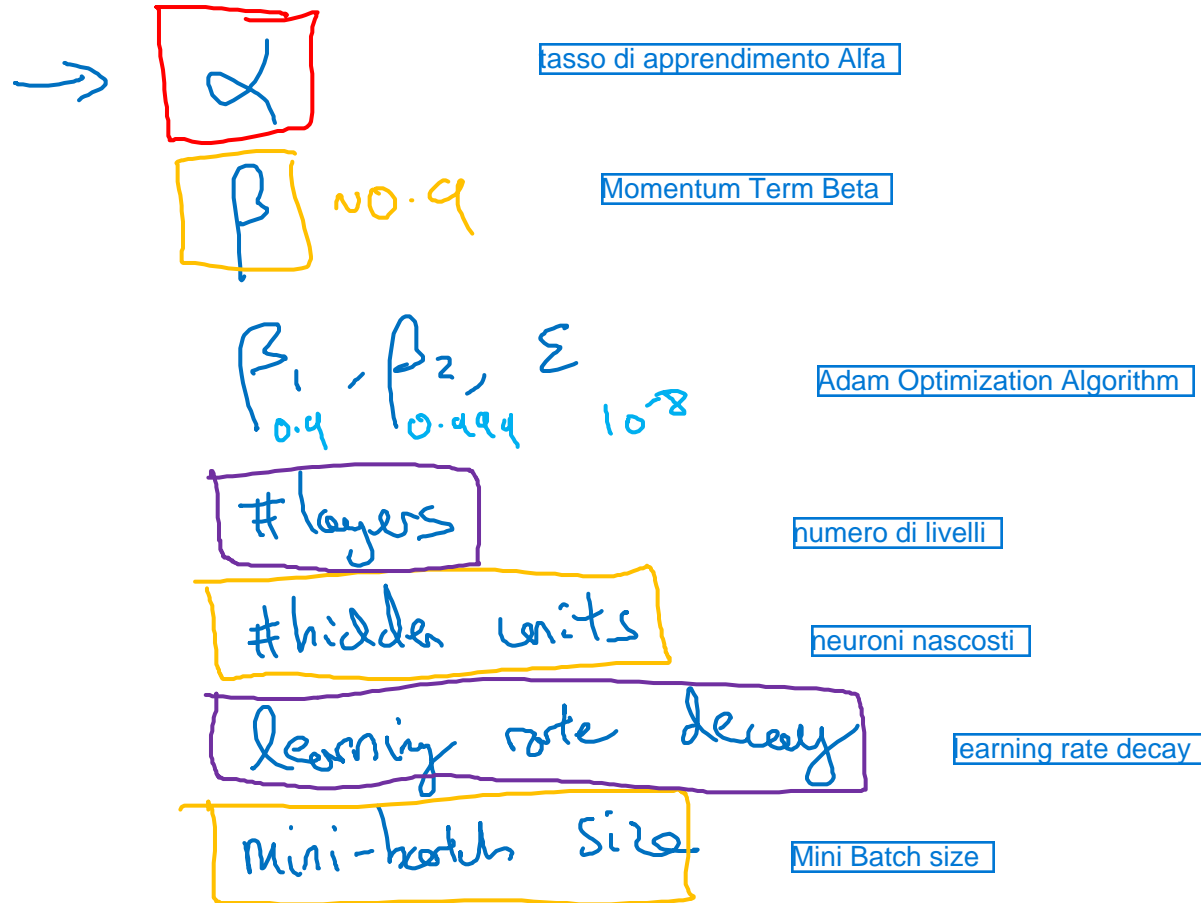
For the rest of the details of the license, see https://creativecommons.org/licenses/by-sa/2.0/legalcode

Hyperparameter tuning

Tuning process

deeplearning.ai

Hyperparameters

$\rightarrow$ α tasso di apprendimento Alfa

β NO.9   Momentum Term Beta

                                                 Quelle in giallo sono gli iperparametri meno importanti per il tasso di apprendimento Alfa

$\beta_1 , \beta_2 , \varepsilon$
$_{0.9} \quad _{0.999} \quad 10^{-8}$ Adam Optimization Algorithm

layers   numero di livelli

#hidden units   neuroni nascosti

learning rate decay   learning rate decay

Mini-batch size   Mini Batch size

Andrew Ng

Try random values: Don't use a grid
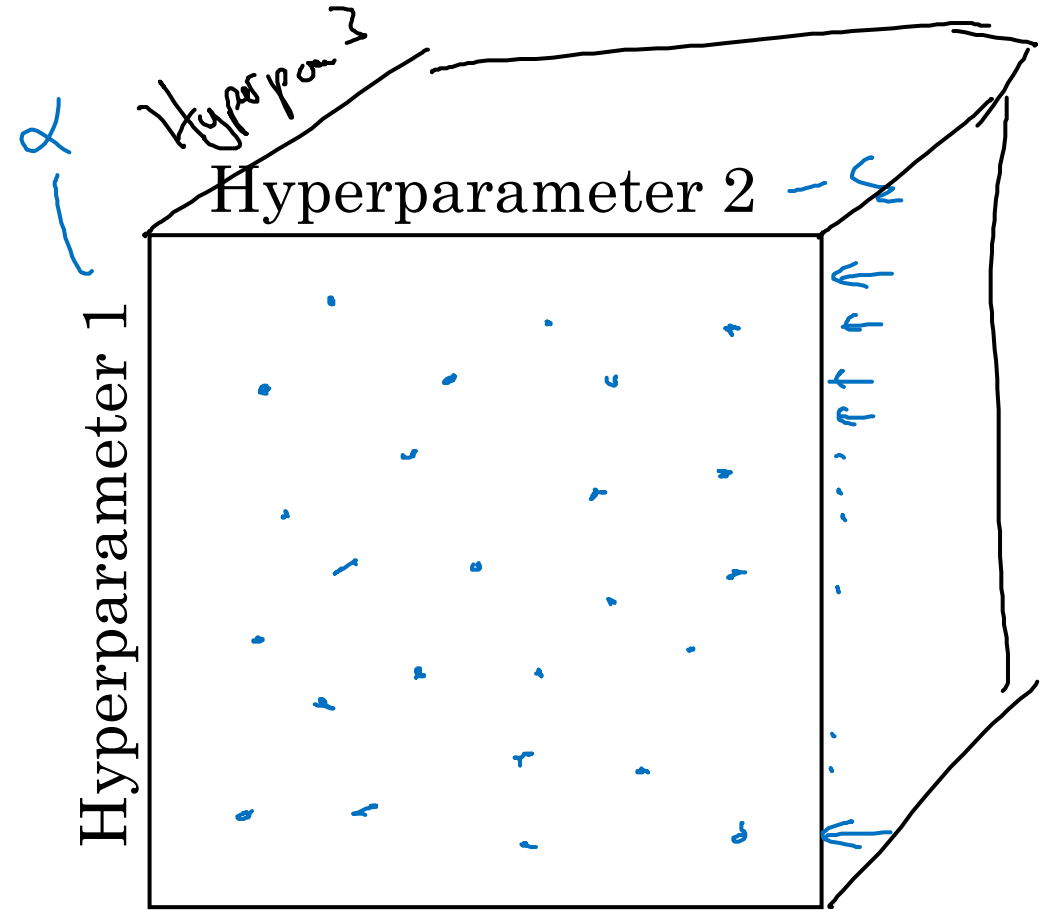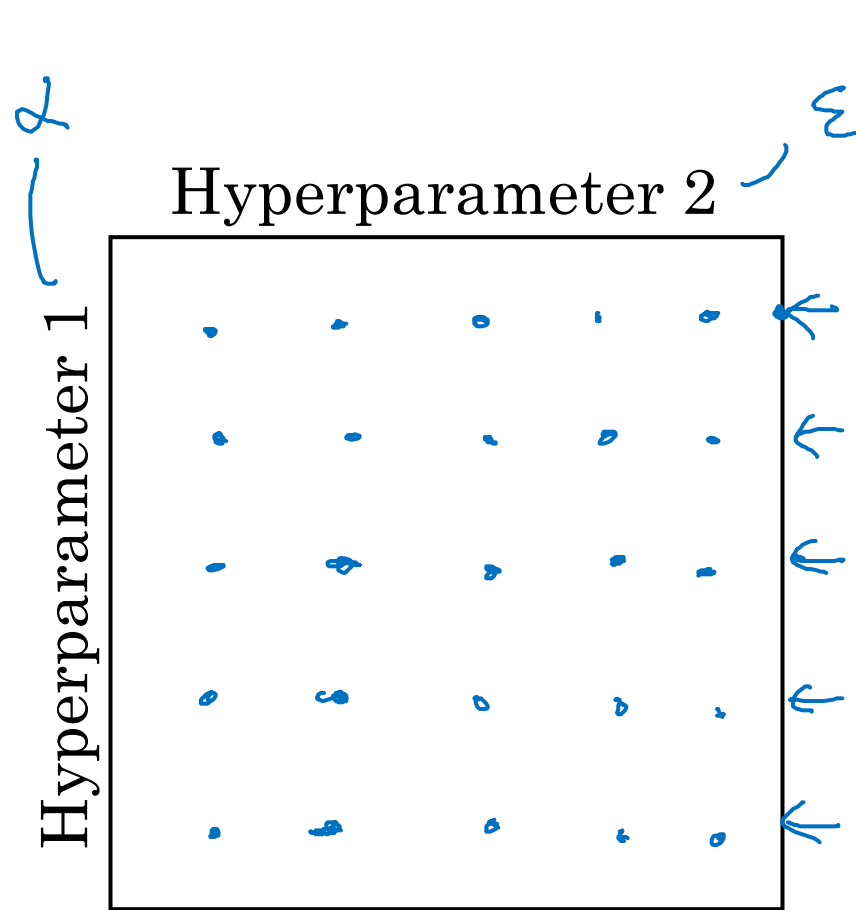
Hyperparameter 2

Hyperparameter 1

α

ε

Hyperparameter 2

Hyperparameter 1

α

Hyperpar 3

ε

Andrew Ng

Coarse to fine

Hyperparameter 2

Hyperparameter 1

Andrew Ng
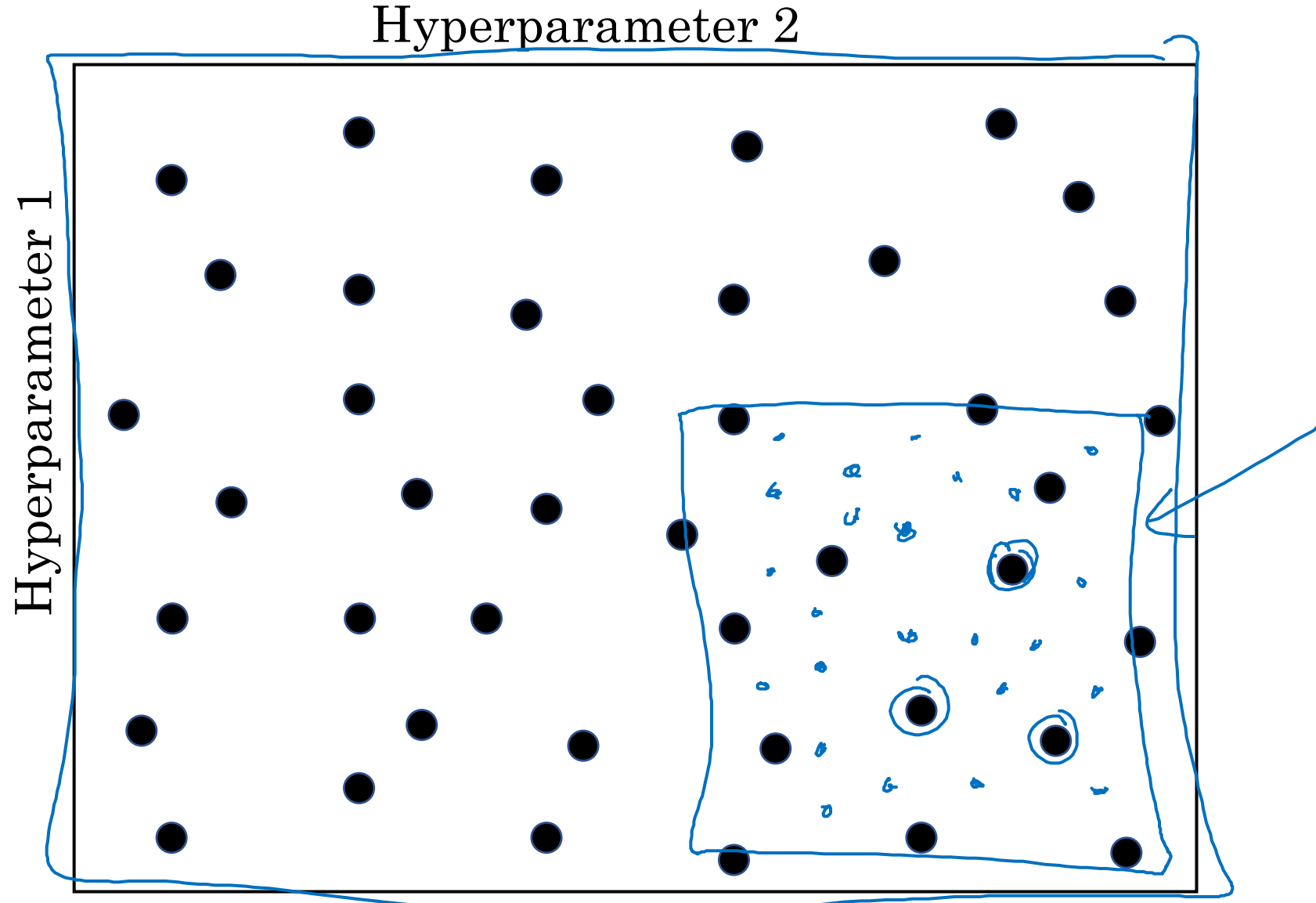
deeplearning.ai

Hyperparameter tuning

Using an appropriate scale to pick hyperparameters

Picking hyperparameters at random

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

$$\underbrace{[\overset{\times \quad \times \times \times \quad \times \times \quad \times \quad \times \quad \times}{\qquad}]}$$

50 $\qquad$ 100

$\rightarrow \#layers \qquad L: \quad 2 - 4$

$$2, 3, 4$$

Andrew Ng
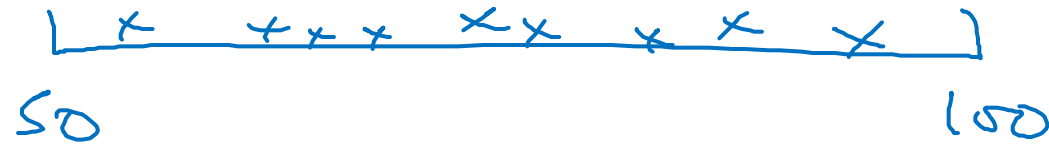
Appropriate scale for hyperparameters

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

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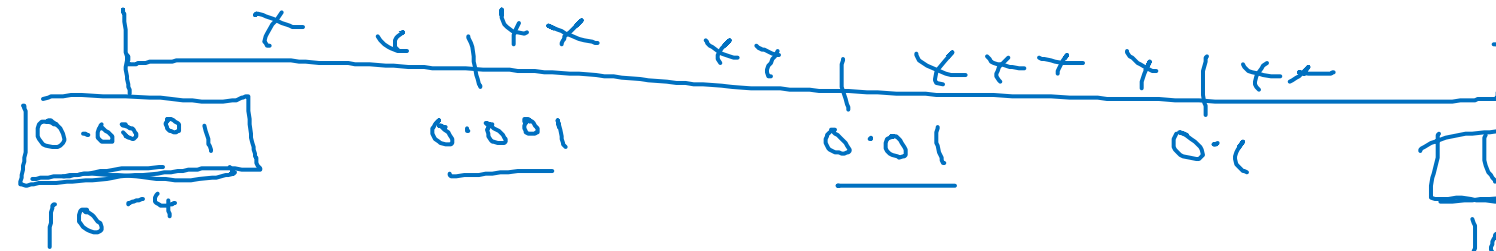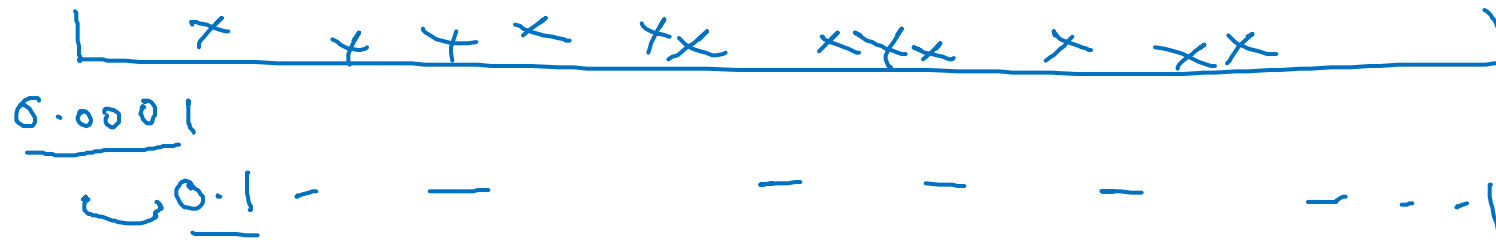
$a = \log_{10} 0.0001$
$= -4$

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

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

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

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

Andrew Ng

Hyperparameters for exponentially weighted averages

$\beta = 0.9 \quad \dots \quad 0.999$

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

$10 \qquad\qquad\qquad 1000$

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

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

$\beta: 0.999 \rightarrow 0.9995$

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

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

$\boxed{\; \times \;\times\; \times\times\; \times\times\; \times \;} \Leftarrow$

$0.9 \qquad\qquad\qquad 0.999$

$0.9 \qquad 0.99 \qquad 0.999$

$0.1 \qquad 0.01 \qquad 0.001$

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

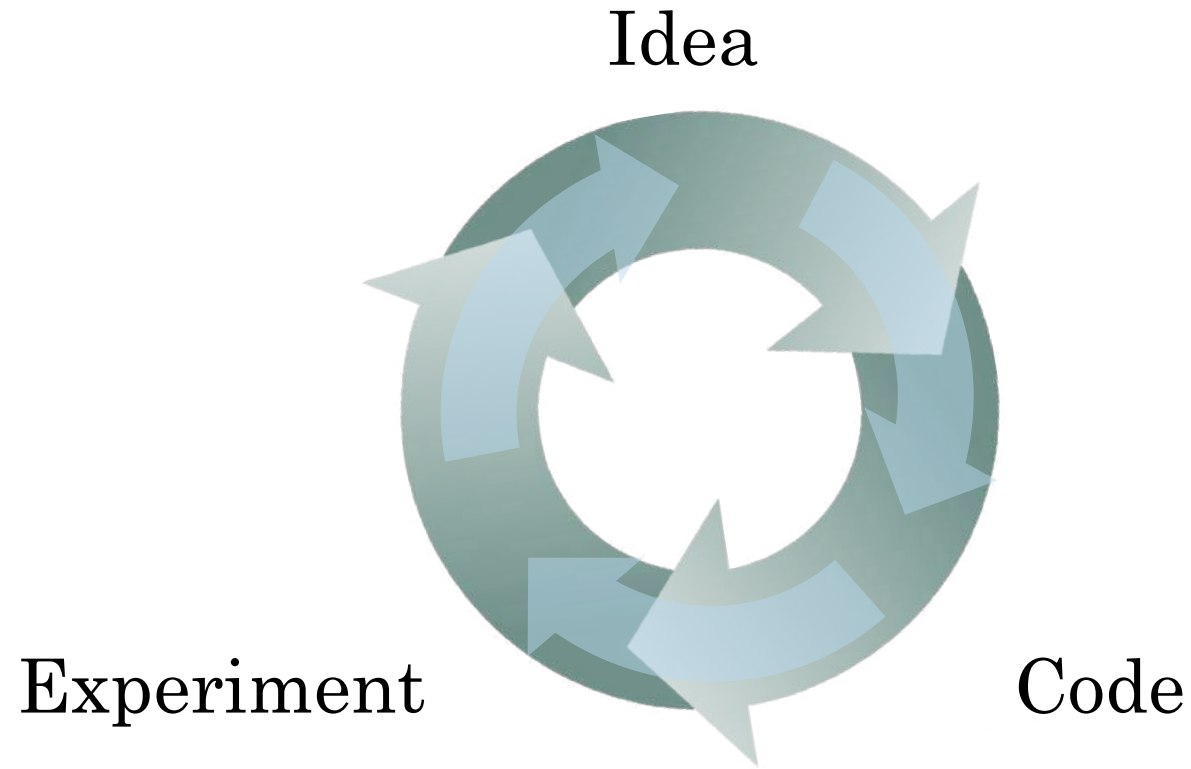$r \in [-3, -1]$

$1-\beta = 10^r$

$\beta = 1 - 10^r$

Andrew Ng
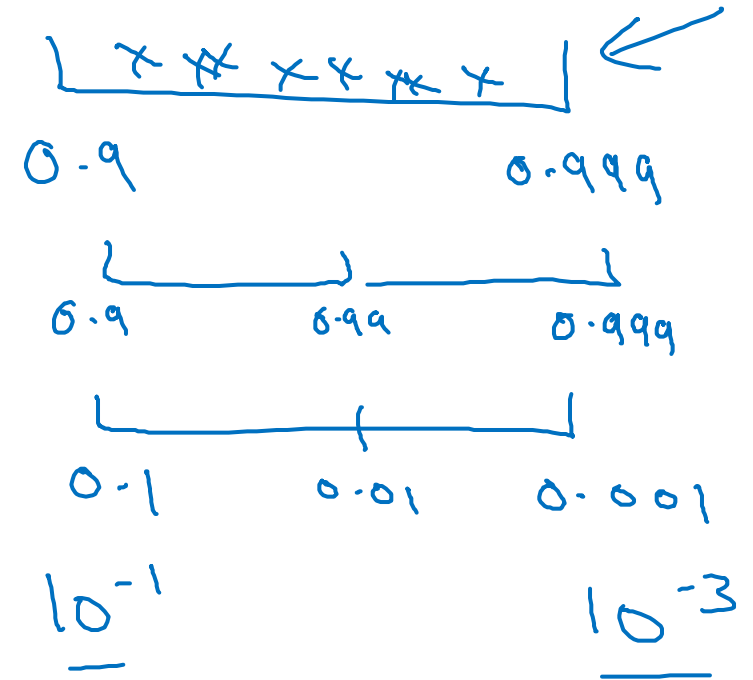
deeplearning.ai
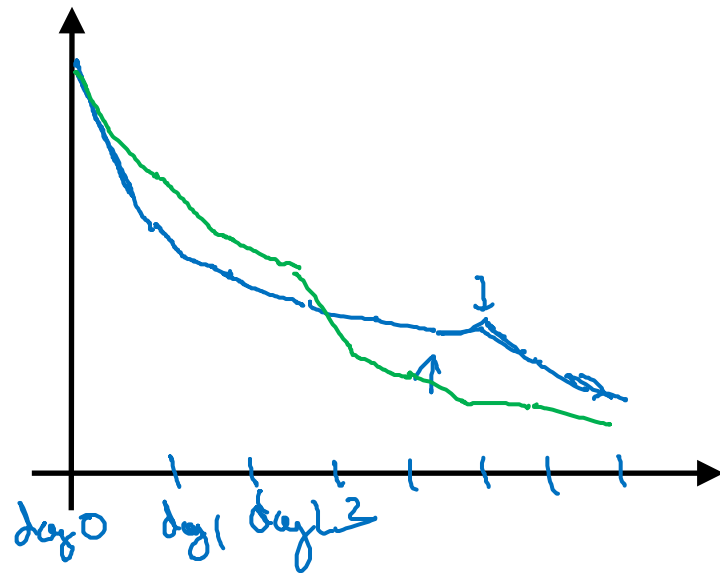
Hyperparameters tuning

Hyperparameters tuning in practice: Pandas vs. Caviar

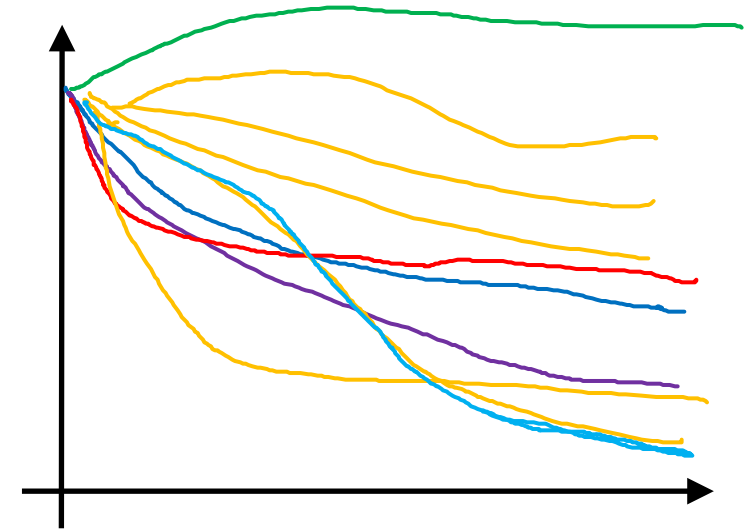# Re-test hyperparameters occasionally

Idea

Experiment   Code

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

- Intuitions do get stale. Re-evaluate occasionally.

Babysitting one model
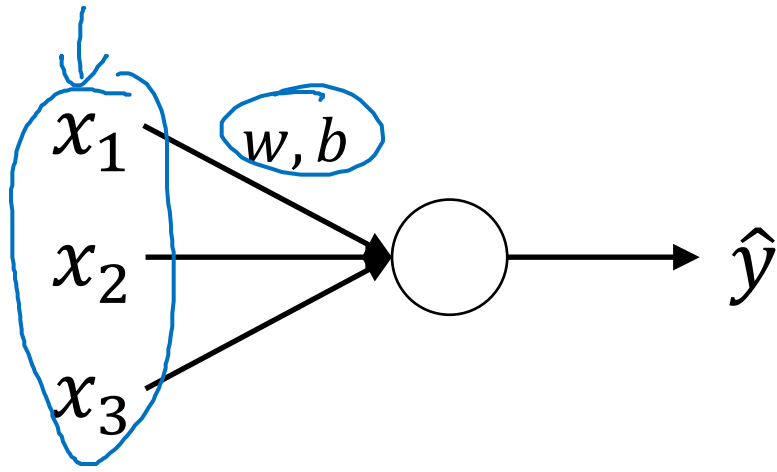

Panda

Training many models in parallel

Caviar

Andrew Ng

Batch
Normalization

Normalizing activations
in a network

deeplearning.ai

Normalizing inputs to speed up learning

x_1
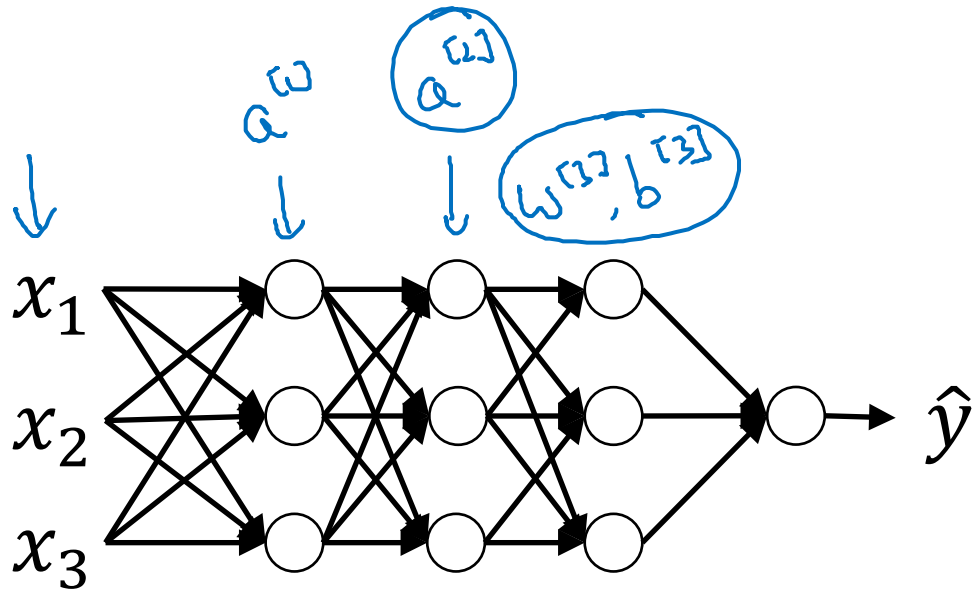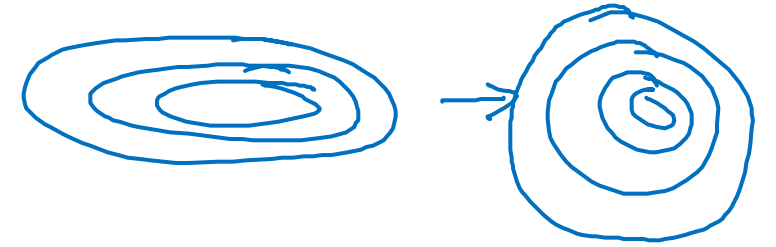w, b
x_2
x_3
$\hat{y}$

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

$$X = X - \mu$$

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

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

$a^{[1]}$
$a^{[2]}$
$w^{[3]}, b^{[3]}$

x_1
x_2
x_3
$\hat{y}$

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

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

Andrew Ng

Implementing Batch Norm

Given some intermediate values in NN $\quad z^{(i)}, \ldots, z^{(m)}$

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

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

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

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

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

learnable parameters of model.

If

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

$\beta = \mu \quad \Leftarrow$

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

$X \Leftarrow$

$z^{(i)} \Leftarrow$

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_1

x_2

x_3

$z_1^{[1]}$ $a_1^{[1]}$

$z_2^{[1]}$ $a_2^{[1]}$

$z_1^{[2]}$ $a_1^{[2]}$

$z_2^{[2]}$ $a_2^{[2]}$

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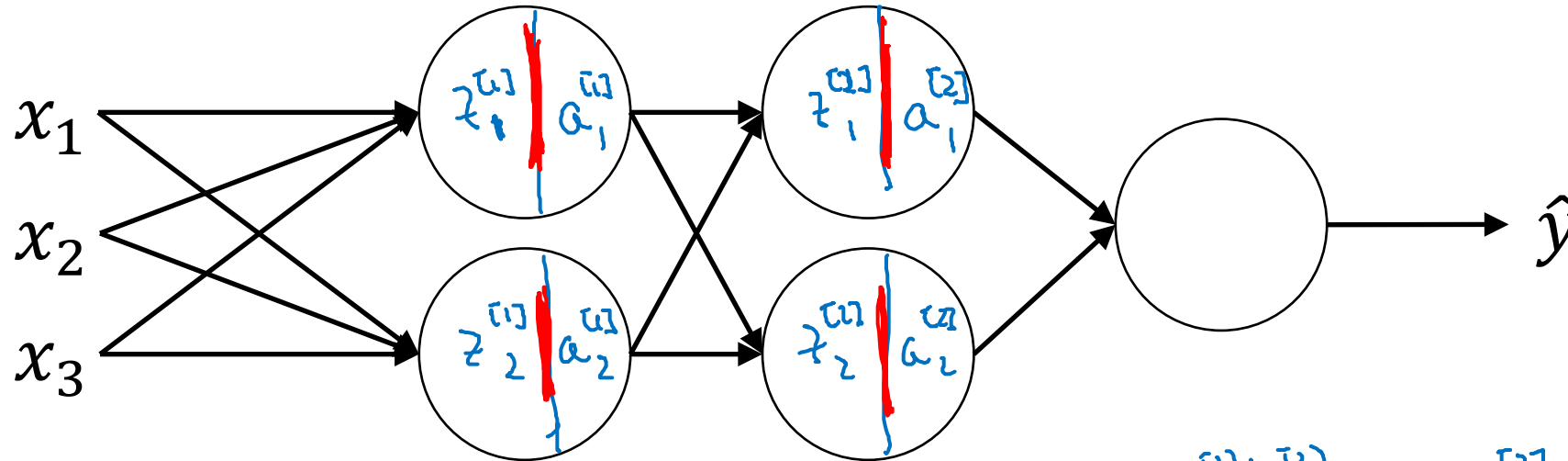
$$X \xrightarrow{W^{[1]}, b^{[1]}} z^{[1]} \xrightarrow[\boxed{\text{Batch Norm (BN)}}]{\beta^{[1]}, \gamma^{[1]}} \tilde{z}^{[1]} \rightarrow a^{[1]} = g^{[1]}(\tilde{z}^{[1]}) \xrightarrow{W^{[2]}, b^{[2]}} z^{[2]} \xrightarrow[BN]{\beta^{[2]}, \gamma^{[2]}} \tilde{z}^{[2]} \rightarrow a^{[2]} \rightarrow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

Parameters: $\left. W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, \cdots, W^{[L]}, b^{[L]}, \right.$

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

$d\beta^{[l]}$

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

tf.nn.batch-normalization $\Leftarrow$

$\rightarrow \beta$

Andrew Ng

Working with mini-batches

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

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

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

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

$$\qquad\qquad\qquad \big|\qquad\quad \big|\qquad\quad \big|$$
$$\qquad\qquad (n^{[l]}, 1)\quad (n^{[l]}, 1)\quad (n^{[l]}, 1)$$
$$\qquad\qquad\qquad \uparrow$$

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

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

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

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

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

Andrew Ng

Implementing gradient descent

for $t = 1 \dots$ num MiniBatches

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

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

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

    Update params
$$W^{[l]} := W^{[l]} - \alpha\, dW^{[l]}$$
$$\beta^{[l]} := \beta^{[l]} - \alpha\, d\beta^{[l]} \Bigg\} \leftarrow$$
$$\gamma^{[l]} := \dots$$

Works w/ momentum, RMSprop, Adam.

deeplearning.ai

Batch
Normalization

Why does
Batch Norm work?

Learning on shifting input distribution

x_1
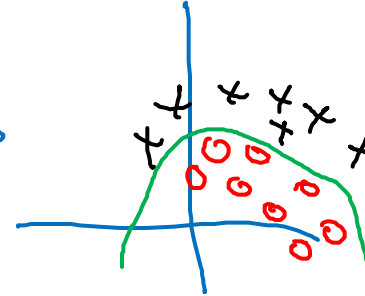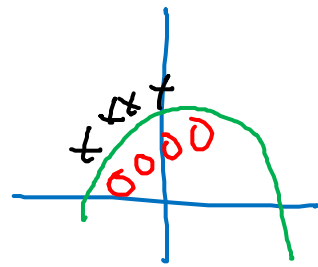x_2
x_3
$\hat{y}$
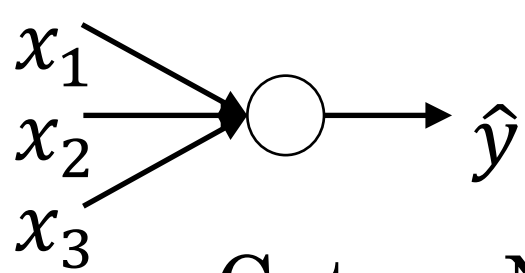
Cat   Non-Cat

$y = 1$ $y = 0$

$y = 1$ $y = 0$

"Covariate shift"

$x \longrightarrow y$

Andrew Ng

Why this is a problem with neural networks?

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

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

$a_1^{[2]}$

$a_2^{[2]}$

$a_3^{[2]}$

$a_4^{[2]}$

$z_2^{[2]}$

$z_2^{[2]}$

$z_1^{[2]}$

$\hat{y}$

$z_1^{[2]}$

$z_1^{[2]}$

Mean 0
Variance 1

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

Andrew Ng
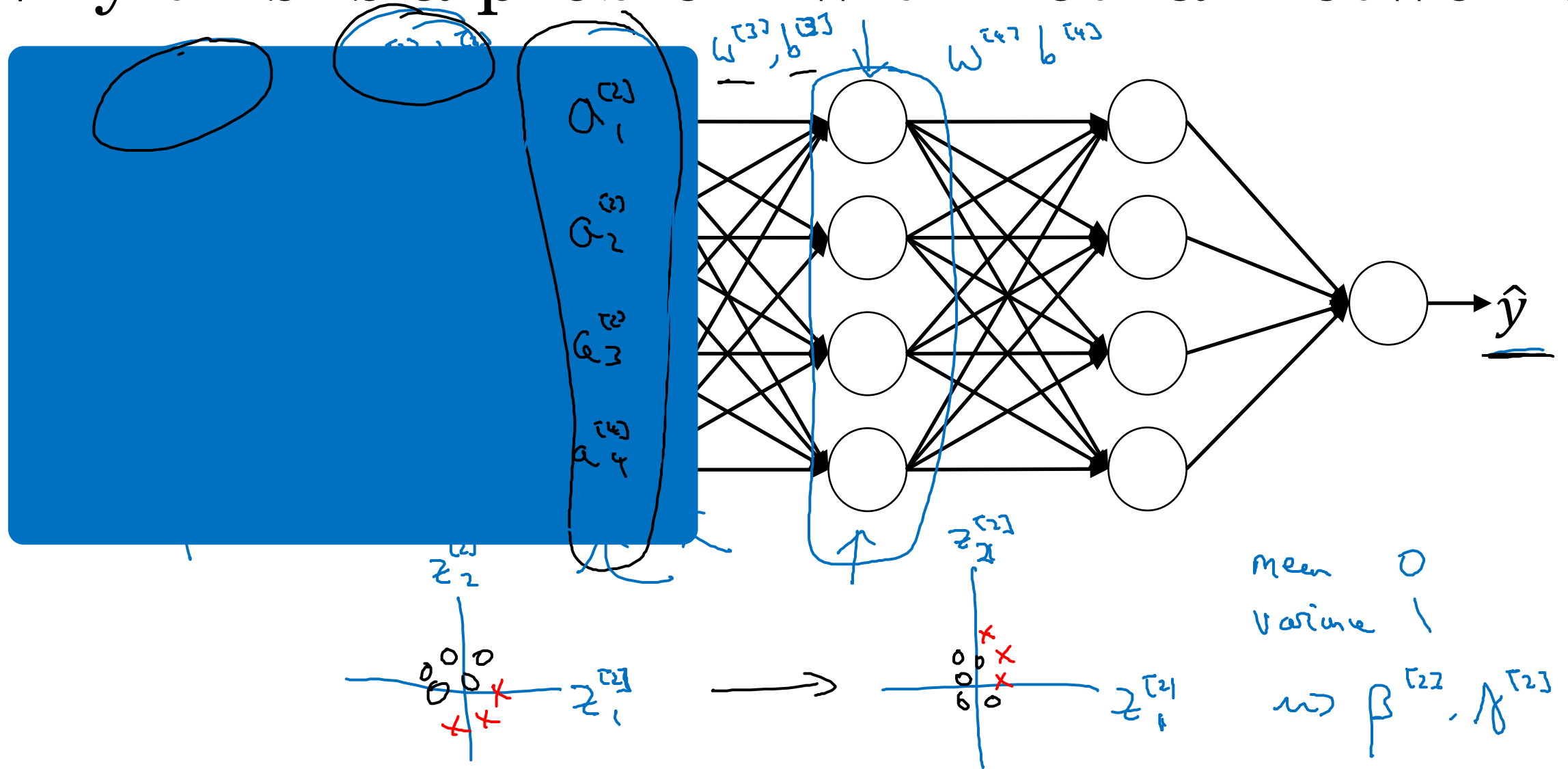
Batch Norm as regularization

- 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.

X

$X^{\{t\}}$

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

$64, 128$

$z^{[l]}$

μ, σ^2

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

Multi-class classification

Softmax regression

deeplearning.ai

Recognizing cats, dogs, and baby chicks
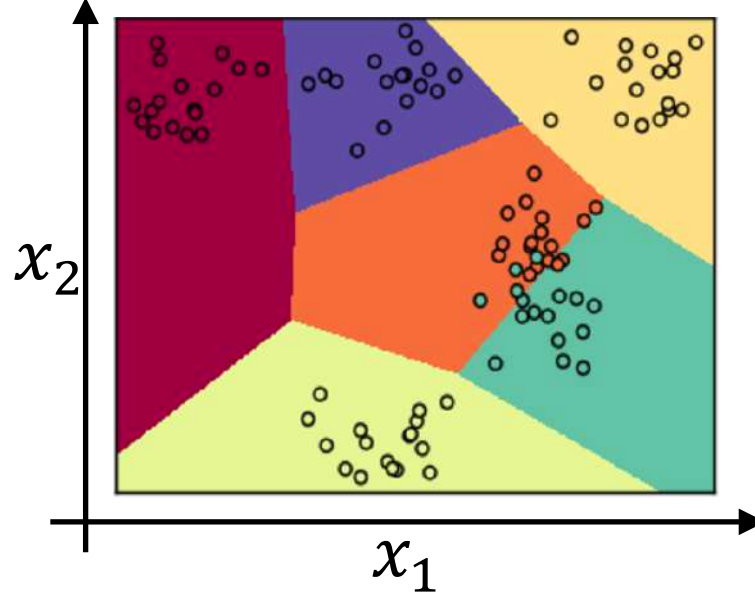

3   1   2   0   3   2   0   1

$$X \rightarrow \hat{y}$$

Softmax layer

$$X \rightarrow \hat{y}$$
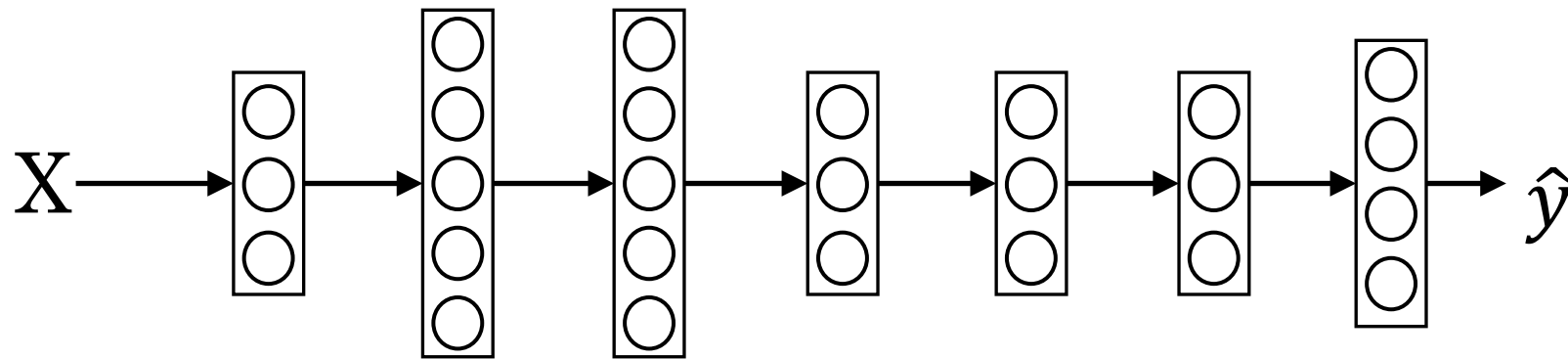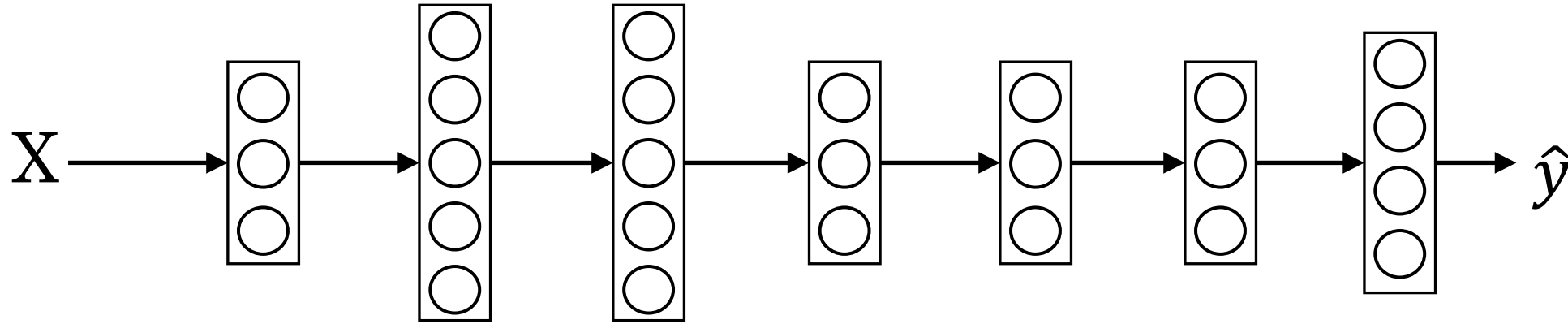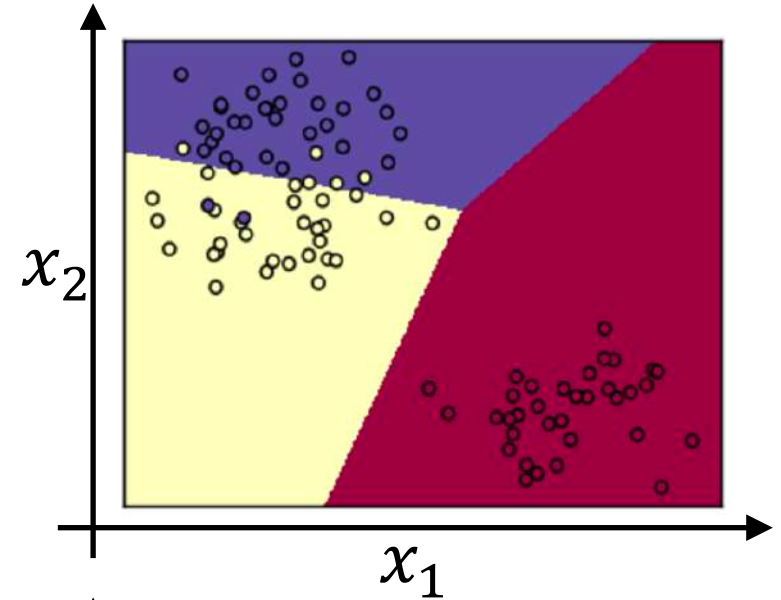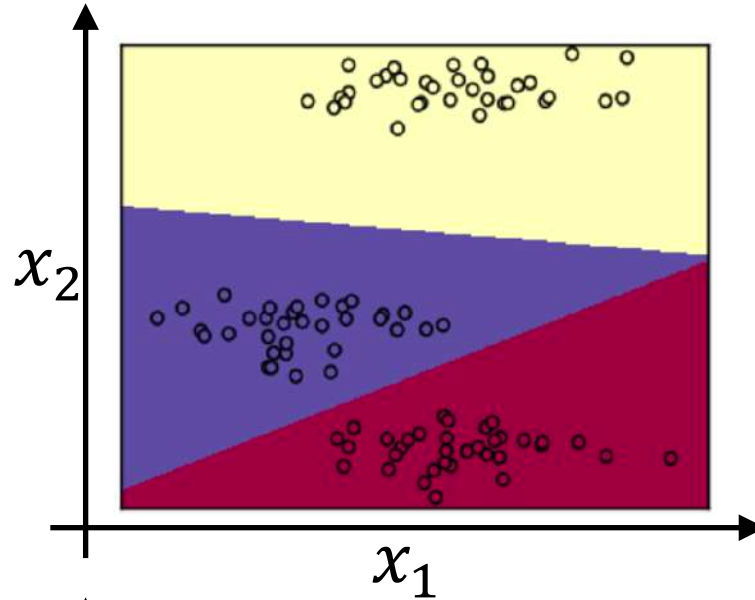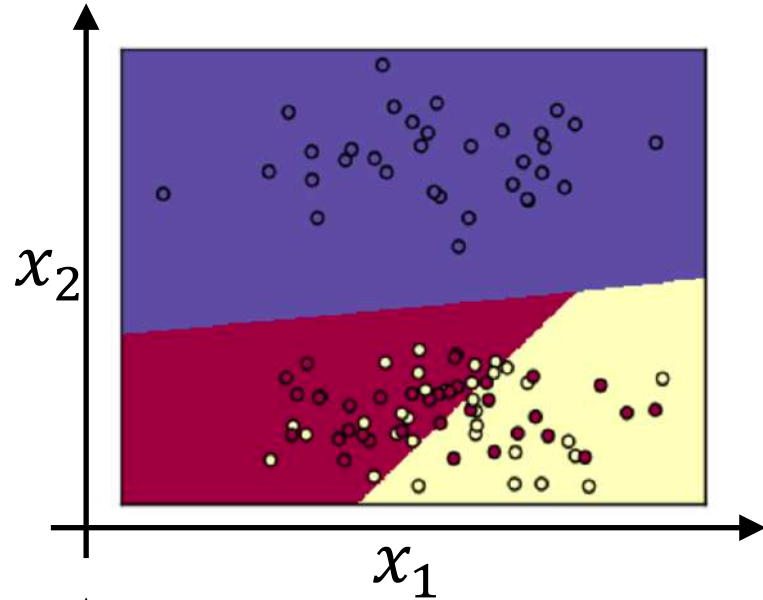
Softmax examples
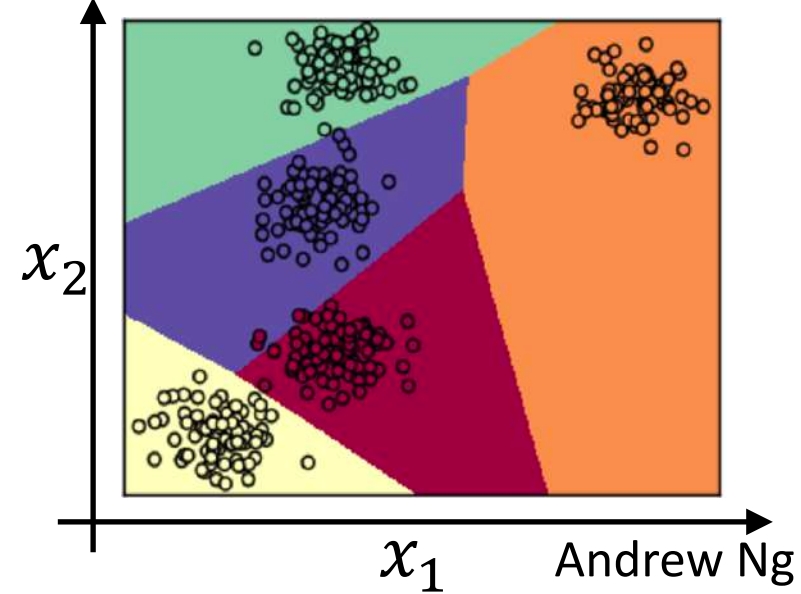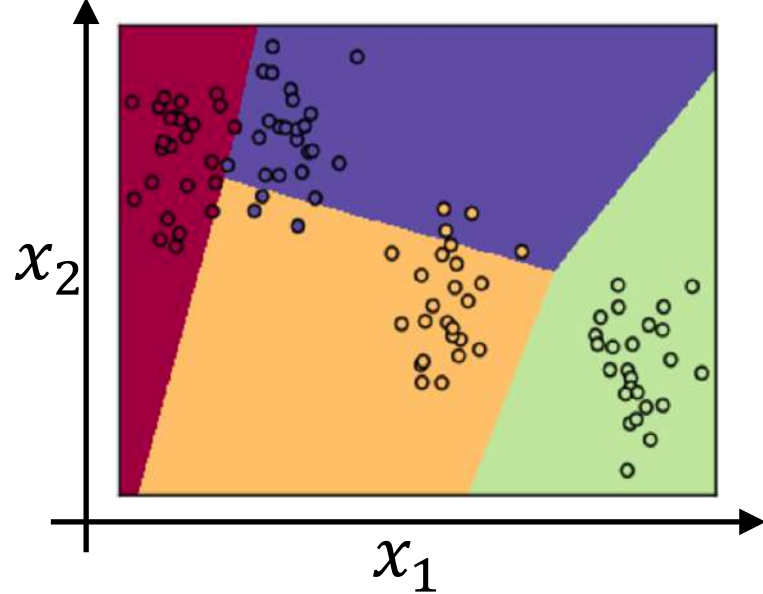
deeplearning.ai

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)

Programming Frameworks

TensorFlow

deeplearning.ai

Motivating problem

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

(cost)

$$(\omega - 5)^2$$

$$\omega = 5$$

$$J(W, b)$$
$$\uparrow \quad \uparrow$$