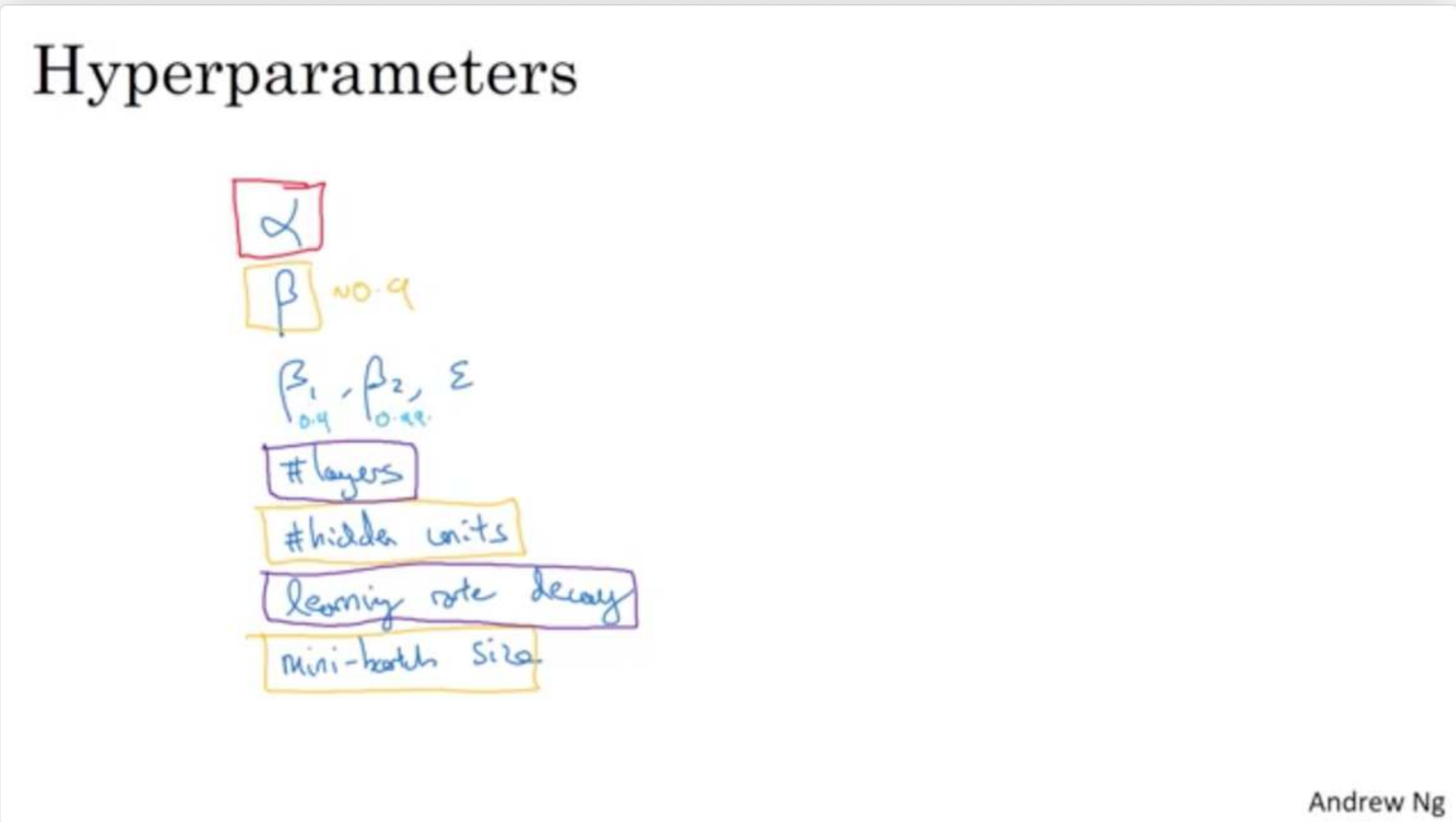
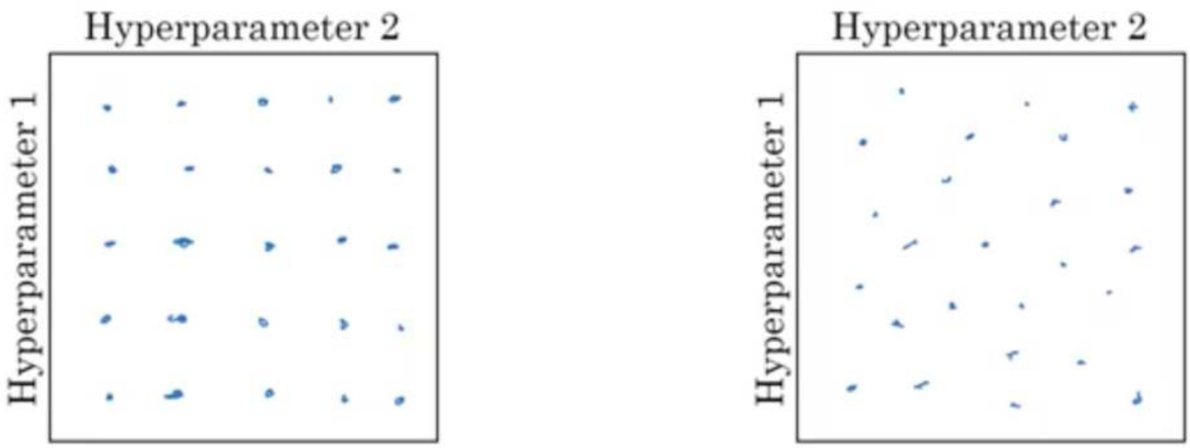
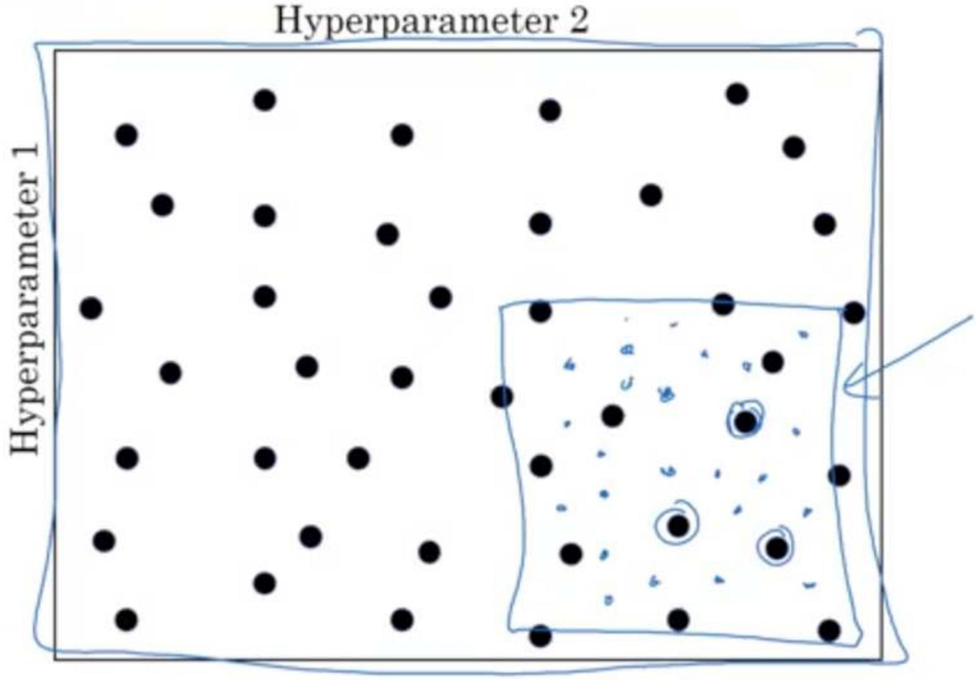

Hyperparameters
α
β ~0.9
β1 , β2 , ε
0.9 0.999
#layers
#hidden units
learning rate decay
mini-batch size
Andrew Ng

Try random values: Don't use a grid



Andrew Ng

Coarse to fine



Andrew Ng

Appropriate scale for hyperparameters

$\alpha = 0.0001, \ldots, 1$

0.0001 ... 1

0.1 ... 1

0.0001   0.001   0.01   0.1   1

10^{-4} 10^{0}

10^{a} 10^{b}

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

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

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

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

Andrew Ng

Hyperparameters for exponentially weighted averages

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

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

$$10 \qquad\qquad 1000$$

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

0.9 $\qquad$ 0.999

0.9 $\qquad$ 0.99

0.1 $\qquad$ 0.01 $\qquad$ 0.001

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

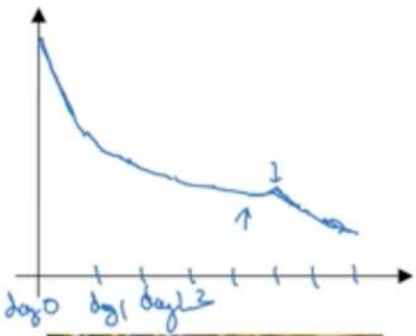
$$r \in [-3, -1]$$

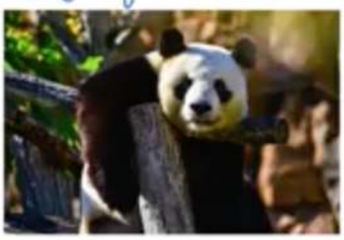
$$1-\beta = 10^{r}$$

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

Andrew Ng

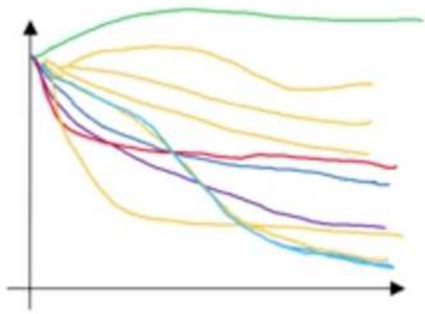
Babysitting one model





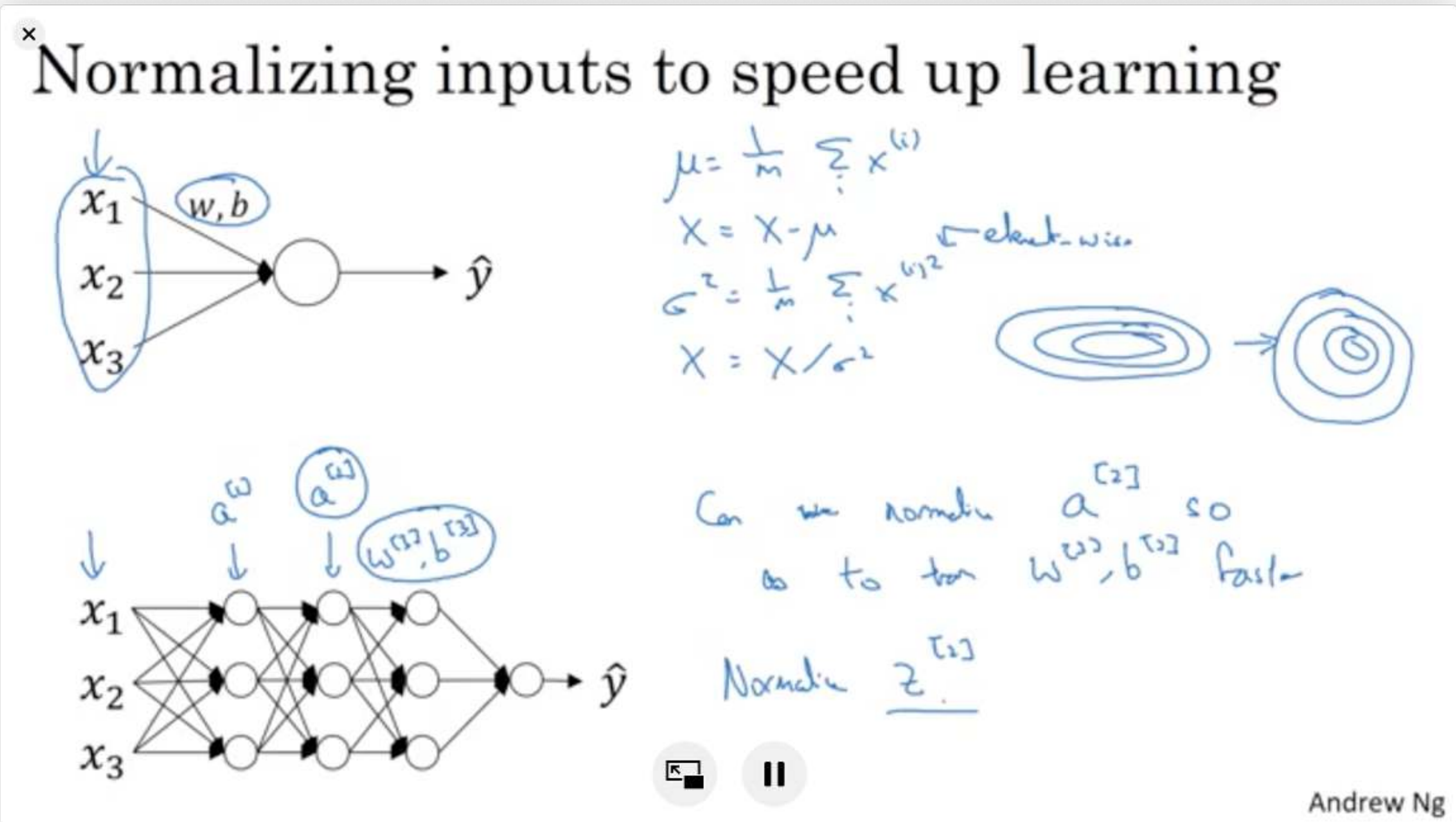
Panda

Training many models in parallel



Caviar

Andrew Ng


Normalizing inputs to speed up learning
x_1
x_2
x_3
w, b
$\hat{y}$
$\mu = \frac{1}{m} \sum_i x^{(i)}$
$X = X - \mu$
element-wise
$\sigma^2 = \frac{1}{m} \sum_i x^{(i)2}$
$X = X / \sigma^2$
$a^{[1]}$
$a^{[2]}$
$w^{[3]}, b^{[3]}$
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 $\underbrace{z^{(1)}, \ldots, z^{(m)}}_{z^{[l](i)}}$

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

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

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

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

learnable parameters of model.

If

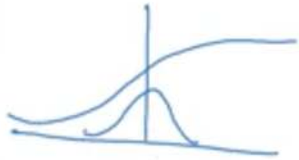
$$\gamma = \boxed{\sqrt{\sigma^2 + \varepsilon}}$$

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

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

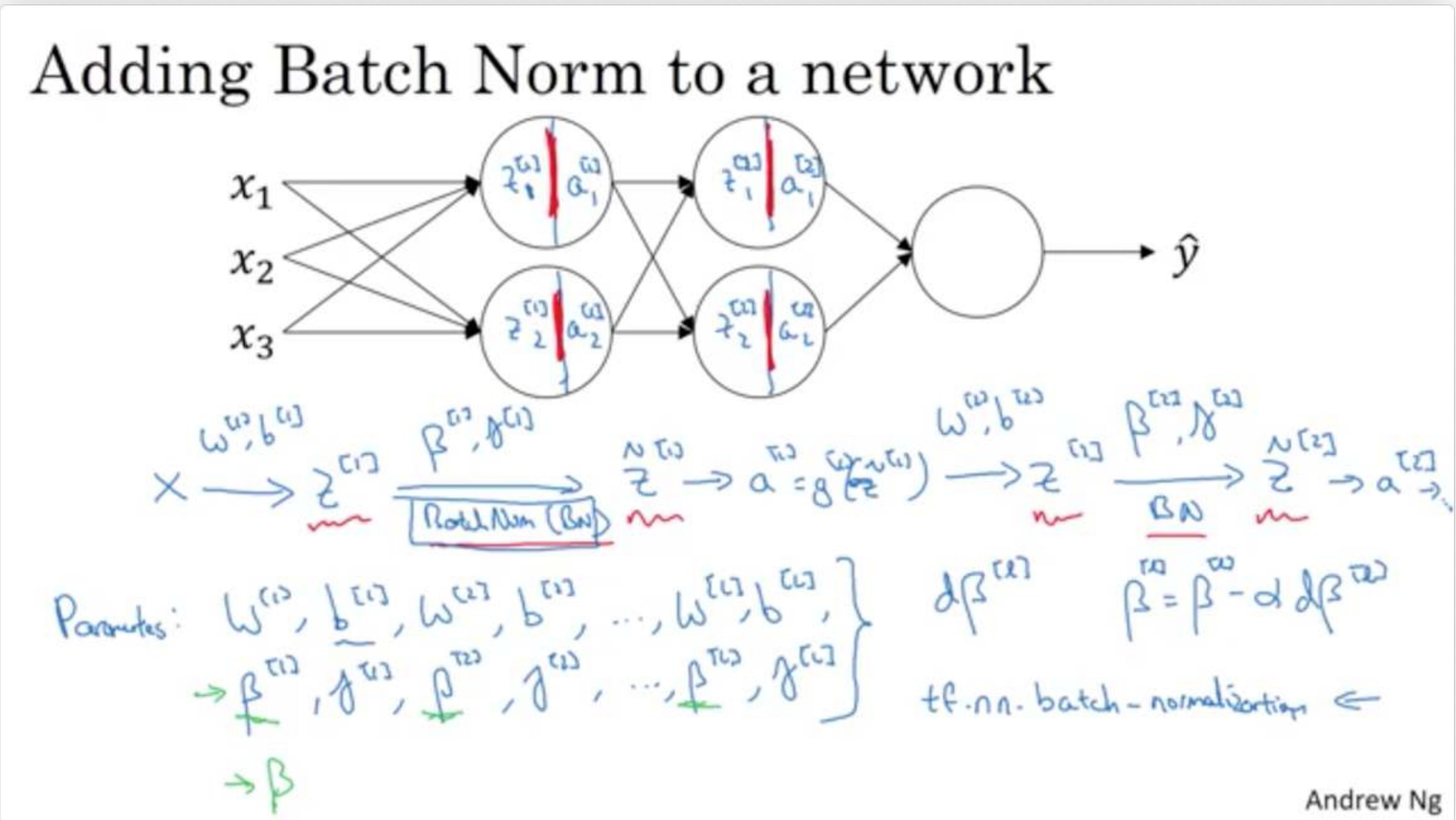
X

$z^{(i)}$



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

Andrew Ng


Adding Batch Norm to a network
x_1
x_2
x_3
$\hat{y}$
$X \xrightarrow{w^{[1]}, b^{[1]}} z^{[1]} \xrightarrow[\text{Batch Norm (BN)}]{\beta^{[1]}, \gamma^{[1]}} \tilde{z}^{[1]} \rightarrow a^{[1]} = g^{[1]}(\tilde{z}^{[1]}) \xrightarrow{w^{[2]}, b^{[2]}} z^{[2]} \xrightarrow[\text{BN}]{\beta^{[2]}, \gamma^{[2]}} \tilde{z}^{[2]} \rightarrow a^{[2]} \rightarrow \cdots$
Parameters: $w^{[1]}, b^{[1]}, w^{[2]}, b^{[2]}, \ldots, w^{[L]}, b^{[L]}$
$\beta^{[1]}, \gamma^{[1]}, \beta^{[2]}, \gamma^{[2]}, \ldots, \beta^{[L]}, \gamma^{[L]}$
$\rightarrow \beta$
$d\beta^{[l]}$
$\beta^{[l]} = \beta^{[l]} - \alpha \, d\beta^{[l]}$
tf.nn.batch_normalization
Andrew Ng

Working with mini-batches

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

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

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

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

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

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

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

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

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

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

Andrew Ng

Implementing gradient descent

for $t = 1 \dots$ 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 $dW^{[l]}$, $\cancel{db^{[l]}}$, $d\beta^{[l]}$, $d\gamma^{[l]}$

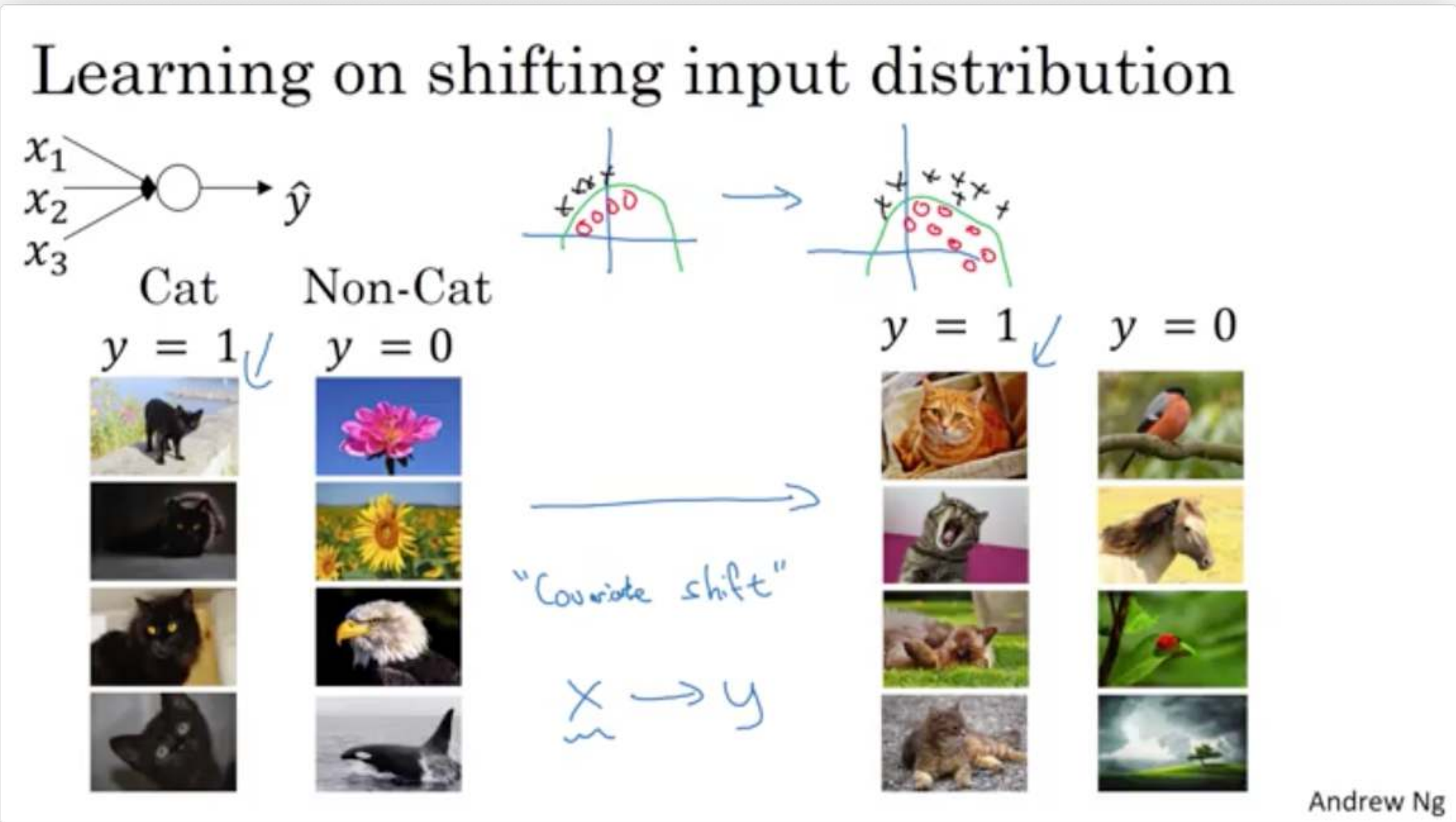
Update params

$$\left.\begin{aligned} W^{[l]} &:= W^{[l]} - \alpha\, dW^{[l]} \\ \beta^{[l]} &:= \beta^{[l]} - \alpha\, d\beta^{[l]} \\ \gamma^{[l]} &:= \dots \end{aligned}\right\} \leftarrow$$

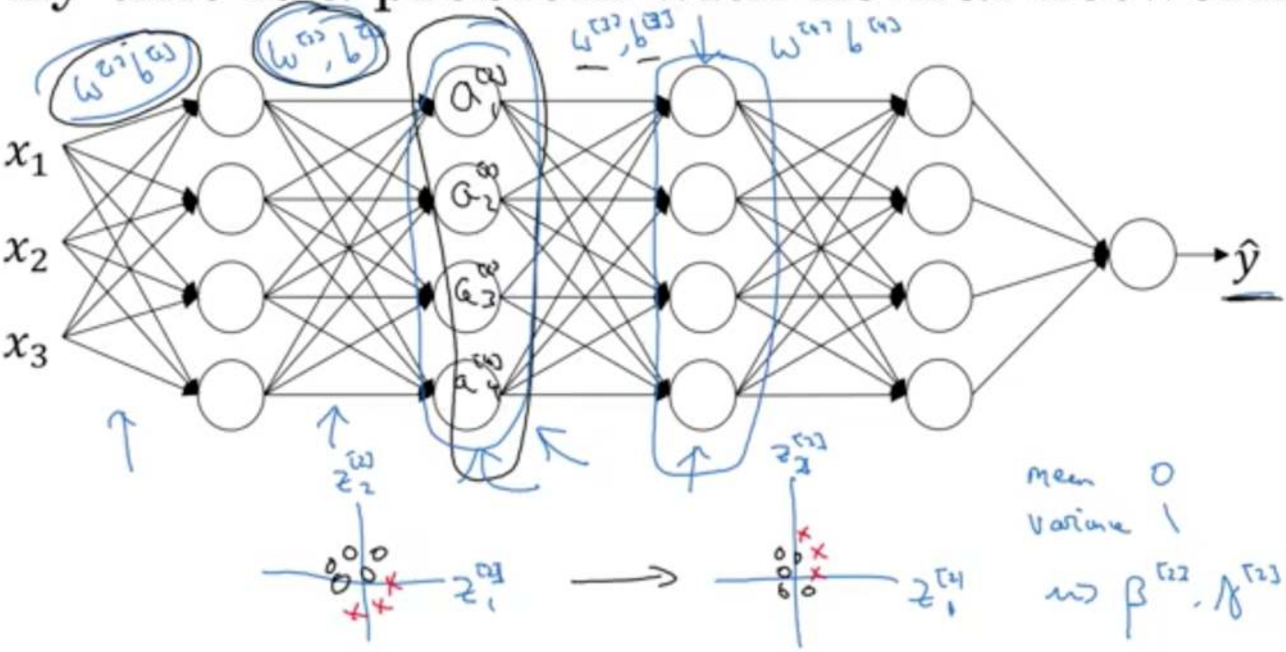
Works w/ momentum, RMSprop, Adam.

the parameters Beta and Gamma that Batch Norm added to algorithm.

Andrew Ng


Learning on shifting input distribution
x_1
x_2
x_3
$\hat{y}$
Cat
Non-Cat
$y = 1$
$y = 0$
"Covariate shift"
$X \rightarrow y$
$y = 1$
$y = 0$
Andrew Ng

Why this is a problem with neural networks?



Andrew Ng

Batch Norm as regularization

X

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

$X^{\{t\}}$

$\tilde{z}^{[l]}$ 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.
- This has a slight regularization effect.

Batch Norm as regularization

X

- Each mini-batch is scaled by the mean/variance computed on just that mini-batch. $X^{\{t\}}$

$\tilde{z}^{[l]}$ 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 : $64 \longrightarrow 512$

Andrew Ng

Batch Norm at test time

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

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

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

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

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

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

$$\begin{array}{ccccccc} X^{\{1\}} & & X^{\{2\}} & & X^{\{3\}} & & \\ \downarrow & & \downarrow & & \downarrow & & \\ \mu^{\{1\}[l]} & & \mu^{\{2\}[l]} & & \mu^{\{3\}[l]} & \longrightarrow & \mu \\ \theta_1 & & \theta_2 & & \theta_3 & & \sigma^2 \end{array}$$

Andrew Ng

Batch Norm at test time

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

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

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

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

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

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

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

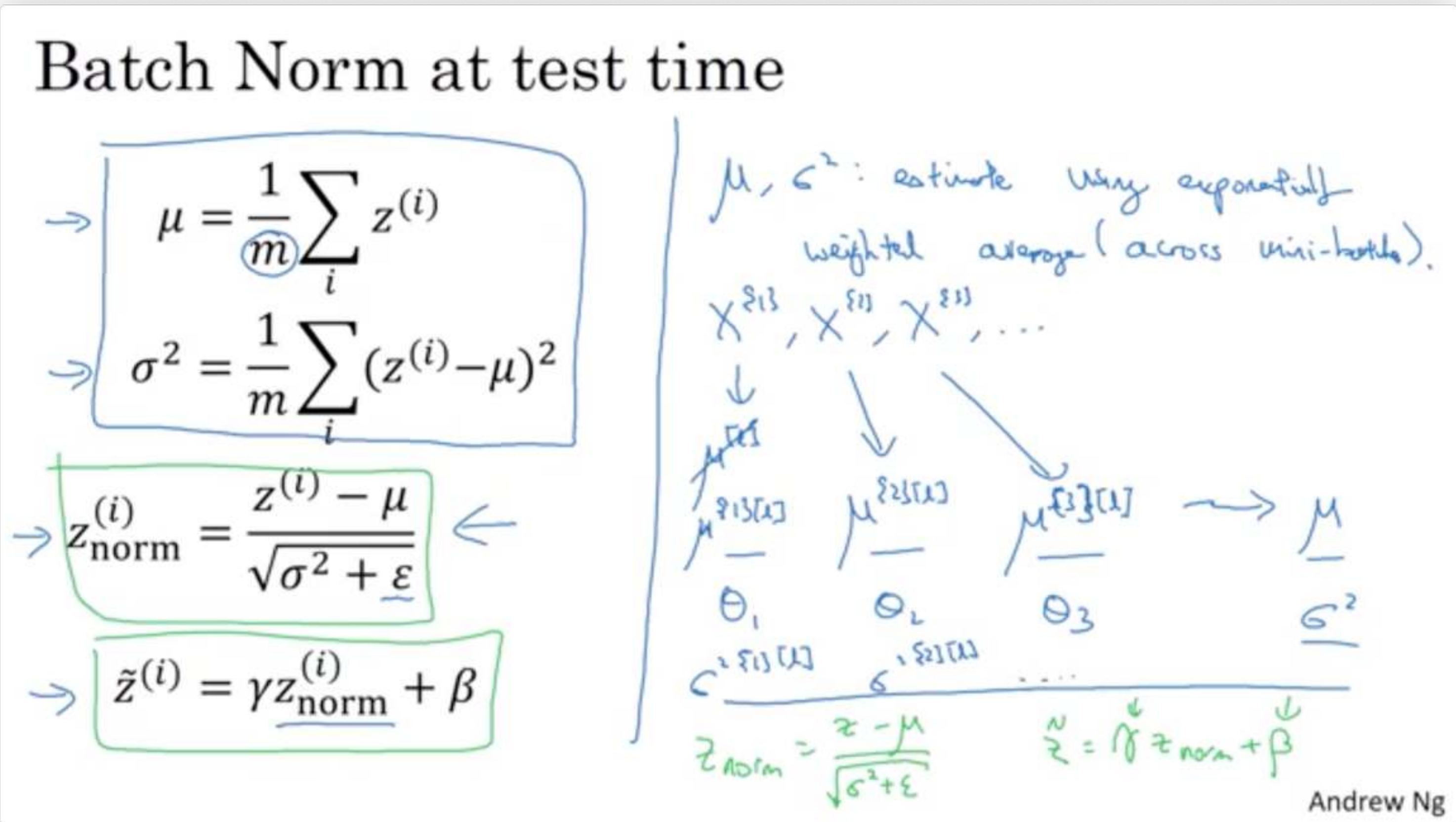
$\theta_1 \quad \theta_2 \quad \theta_3 \qquad \sigma^2$

$\sigma^{2\{1\}[l]} \quad \sigma^{2\{2\}[l]} \quad \ldots$

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

$\tilde{z} = \gamma z_{\text{norm}} + \beta$

Andrew Ng

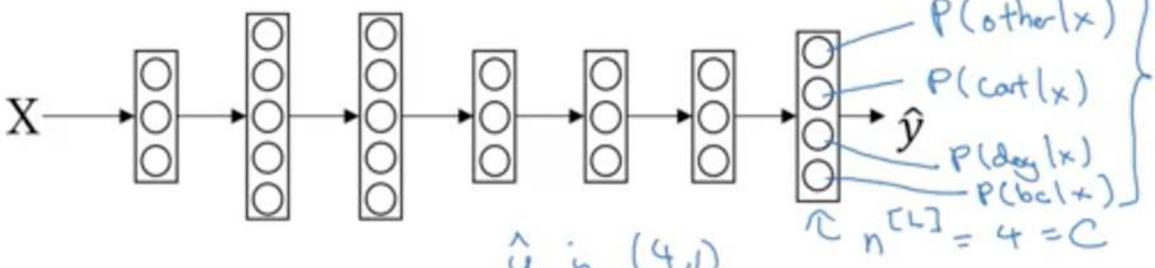

Batch Norm at test time
$\mu = \frac{1}{m}\sum_i z^{(i)}$
$\sigma^2 = \frac{1}{m}\sum_i (z^{(i)} - \mu)^2$
$z^{(i)}_{\text{norm}} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$
$\tilde{z}^{(i)} = \gamma z^{(i)}_{\text{norm}} + \beta$
μ, σ^2: estimate using exponentially weighted average (across mini-batches).
$X^{\{1\}}, X^{\{2\}}, X^{\{3\}}, \ldots$
$\mu^{\{1\}[l]}$ $\mu^{\{2\}[l]}$ $\mu^{\{3\}[l]}$ $\rightarrow \mu$
θ_1 θ_2 θ_3 σ^2
$\sigma^{2\{1\}[l]}$ $\sigma^{2\{2\}[l]}$ ...
$z_{\text{norm}} = \frac{z - \mu}{\sqrt{\sigma^2 + \varepsilon}}$
$\tilde{z} = \gamma z_{\text{norm}} + \beta$
Andrew Ng

Recognizing cats, dogs, and baby chicks , other

1 2 3 0

3 1 2 0 3 2 0 1

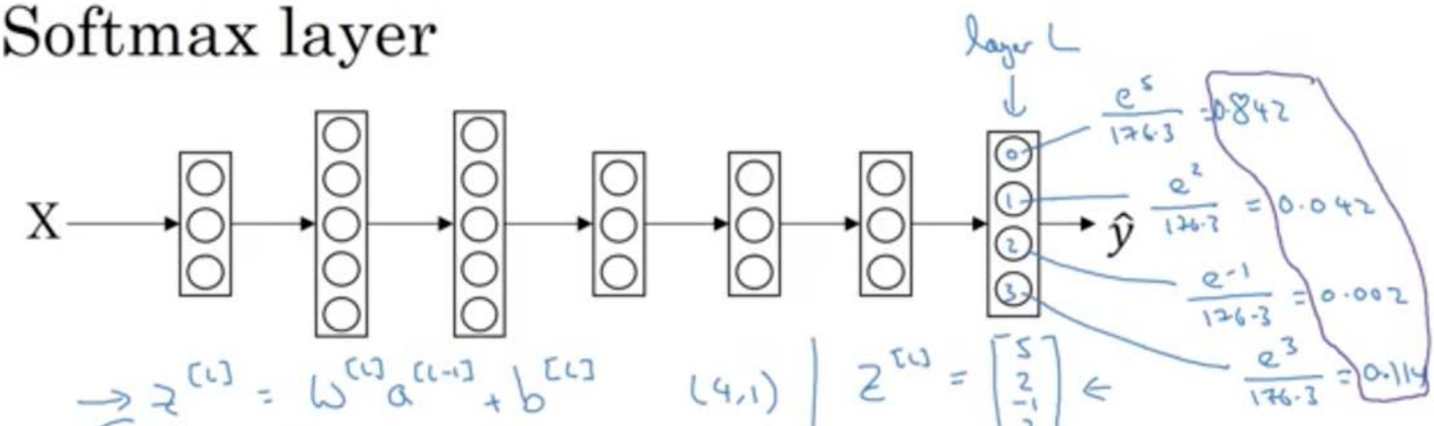
C = #classes = 4 (0, …, 3)



$\hat{y}$ is $(4,1)$

Andrew Ng

Softmax layer



$$z^{[L]} = W^{[L]} a^{[L-1]} + b^{[L]} \qquad (4,1)$$

Activation function:

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

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

$\boxed{(4,1)}$ $\hat{y}$

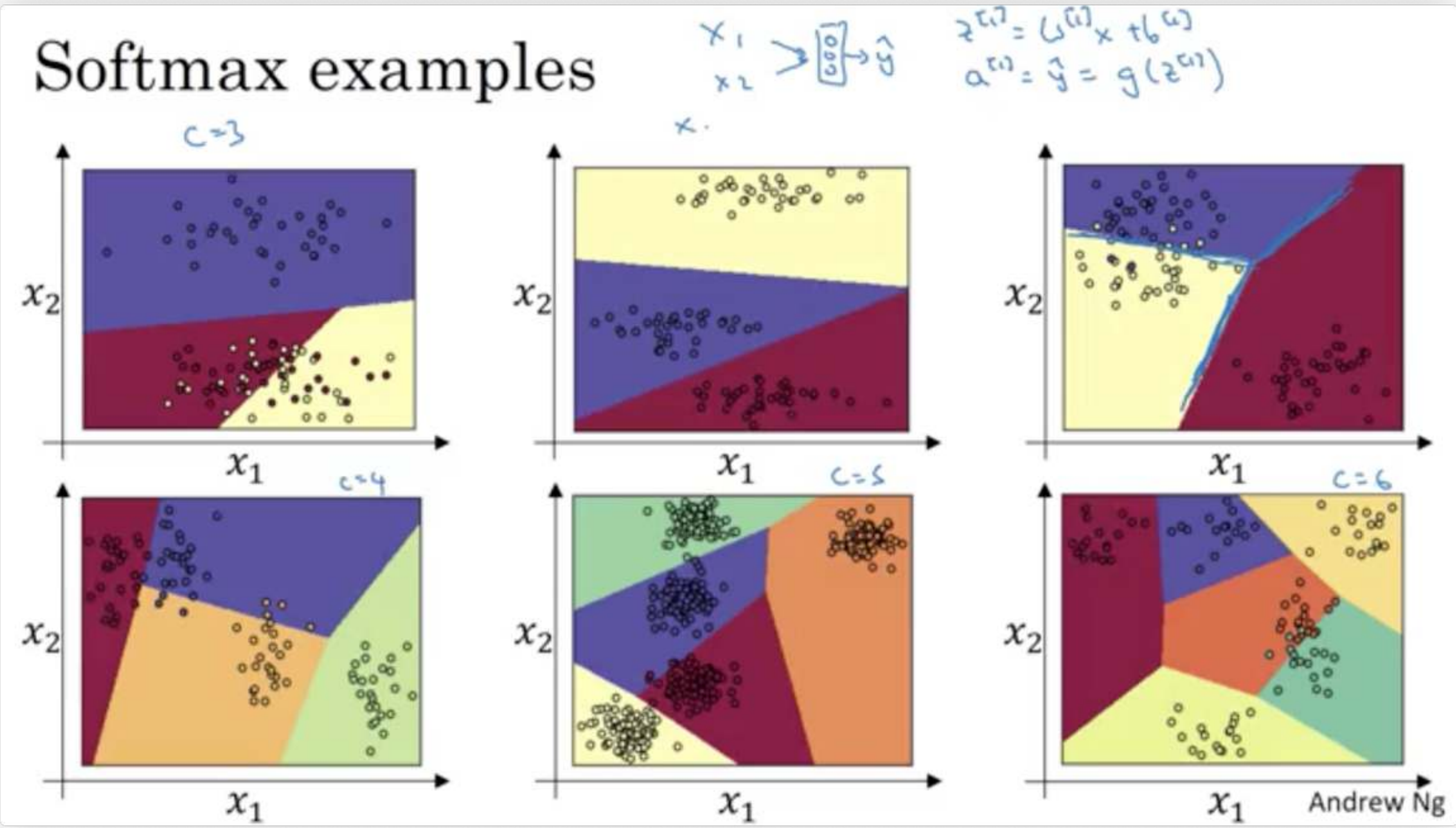
$$\underset{(4,1)}{a^{[L]}} = g^{[L]}(\underset{(4,1)}{z^{[L]}})$$

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

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

$$a^{[L]} = \frac{t}{176.3}$$

Andrew Ng


Softmax examples
$z^{[l]} = W^{[l]}x + b^{[l]}$
$a^{[l]} = \hat{y} = g(z^{[l]})$
C=3
C=4
C=5
C=6
x_1
x_2
Andrew Ng

Understanding softmax

$(4,1)$

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

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

"soft max"

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

"hard max"

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

Softmax regression generalizes logistic regression to C classes.

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

Andrew Ng

Loss function

$(4,1)$

$$y^{(i)} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \text{ — cat} \qquad y_2 = 1 \qquad y_1 = y_3 = y_4 = 0$$

$(4,1)$

$$a^{[L](i)} = \hat{y}^{(i)} = \begin{bmatrix} 0.3 \\ 0.2 \\ 0.1 \\ 0.4 \end{bmatrix} \qquad C = 4$$

$$\underbrace{\mathcal{L}(\hat{y}, y)}_{\text{small}} = -\sum_{j=1}^{4} y_j \log \hat{y}_j$$

$$J(w^{[1]}, b^{[1]}, \dots) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

$$-y_2 \log \hat{y}_2 = -\log \hat{y}_2 .$$

Make $\hat{y}_2$ big.

$$Y = [y^{(1)} \; y^{(2)} \dots, y^{(m)}] = \begin{bmatrix} 0 & 0 & 1 & \dots \\ 1 & 0 & 0 & \\ 0 & 1 & 0 & \\ 0 & 0 & 0 & \end{bmatrix}$$

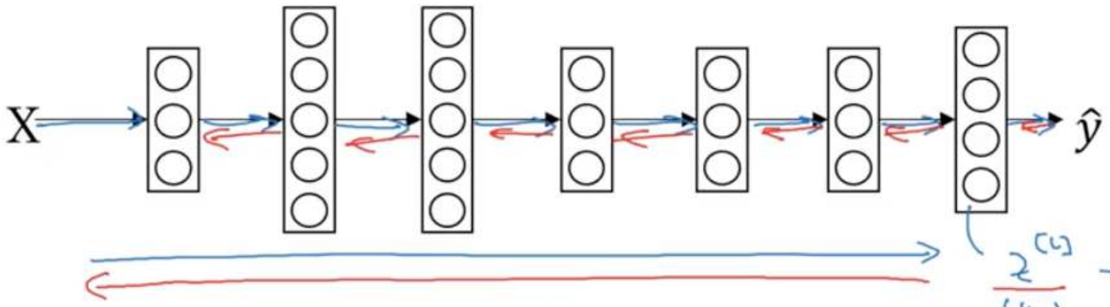
$(4, m)$

$$\hat{Y} = [\hat{y}^{(1)}, \dots, \hat{y}^{(m)}] = \begin{bmatrix} 0.3 & \dots \\ 0.2 & \\ 0.1 & \\ 0.4 & \end{bmatrix}$$

$(4, m)$

Andrew Ng

Gradient descent with softmax



$\underset{(4,1)}{z^{[L]}} \rightarrow a^{[L]} = \hat{y} \rightarrow \mathcal{L}(\hat{y}, y)$

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

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

Andrew Ng

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)

Andrew Ng