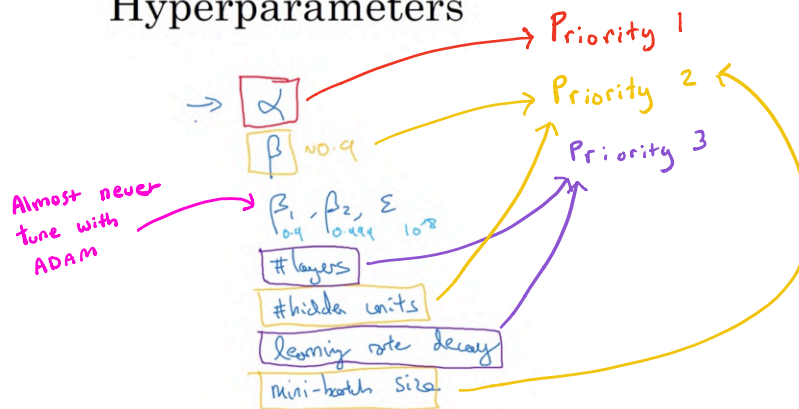
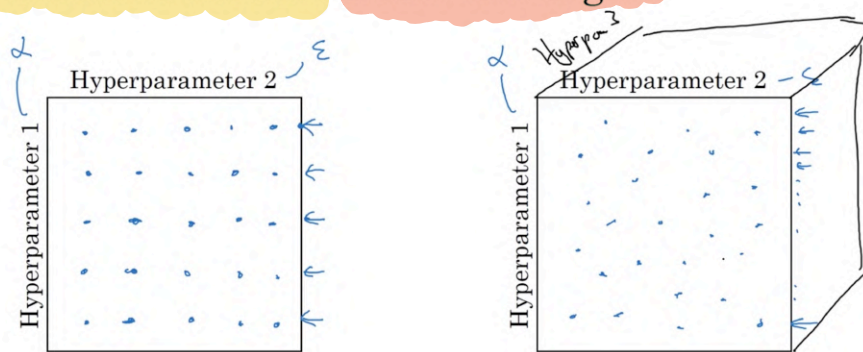


Hyperparameters



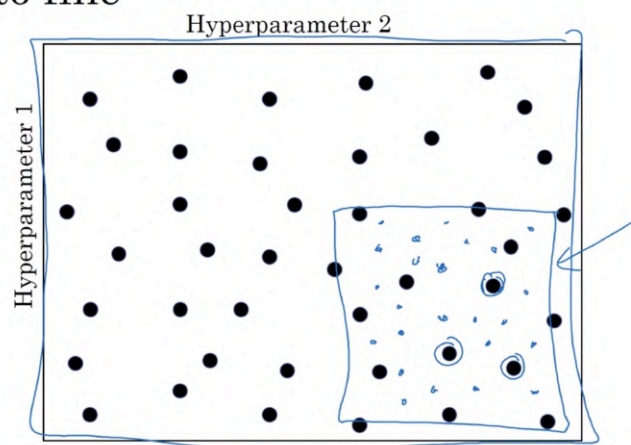
Andrew Ng

Try random values: Don't use a grid



Andrew Ng

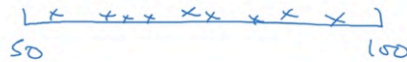
Coarse to fine



Andrew Ng

Picking hyperparameters at random

$$\rightarrow n^{test} = 50, \dots, 100$$



$$\rightarrow \# \text{layers } L: 2 - 4$$

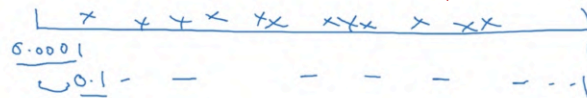
$$2, 3, 4$$

OR

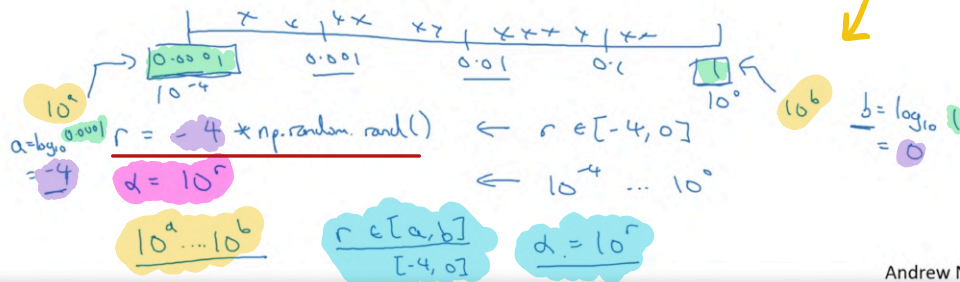
Andrew Ng

Appropriate scale for hyperparameters

$$\alpha = 0.0001, \dots, 1$$



Uniform
log scale



Andrew Ng

so $\alpha = 10^r$ where $r \in [-4, 0]$

Hyperparameters for exponentially weighted averages

$$\beta = 0.9 \dots 0.999$$

$$\downarrow \quad \quad \downarrow$$

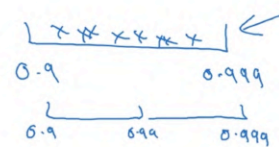
$$10 \quad \quad 1000$$

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

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

$$\beta: 0.999 \rightarrow 0.9995 \} \sim 1000$$

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



$$0.9 \quad \quad 0.999$$

$$0.9 \quad 0.99 \quad 0.999$$

$$0.1 \quad 0.01 \quad 0.001$$

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

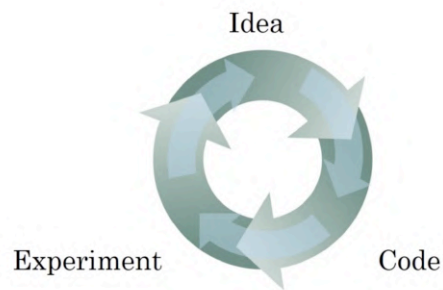
$$r \in [-3, -1]$$

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

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

Andrew Ng

Re-test hyperparameters occasionally

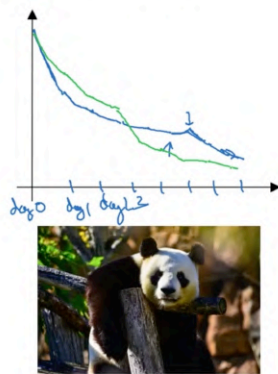


- NLP, Vision, Speech,
Ads, logistics,

- Intuitions do get stale.
Re-evaluate occasionally.

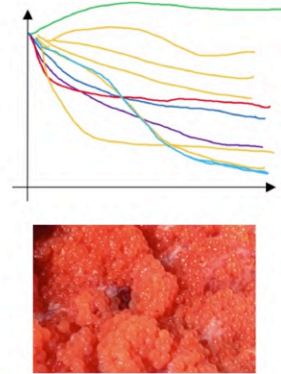
Andrew Ng

Babysitting one model



Panda

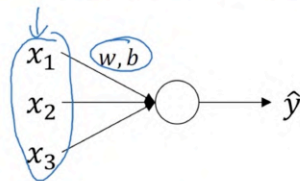
Training many models in parallel



Caviar

Andrew Ng

Normalizing inputs to speed up learning



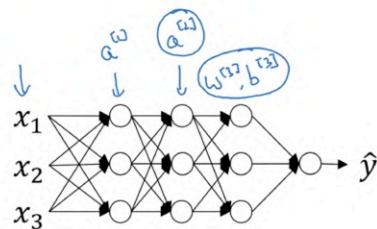
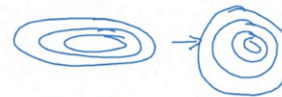
$$\mu = \frac{1}{n} \sum_i x^{(i)}$$

$$X = X - \mu$$

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

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

← element-wise



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

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

Andrew Ng

Implementing Batch Norm

Given some intermediate values in NN

$$\begin{aligned} \mu &= \frac{1}{m} \sum_i z^{(i)} \\ \sigma^2 &= \frac{1}{m} \sum_i (z_i - \mu)^2 \\ z_{\text{norm}}^{(i)} &= \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \epsilon}} \\ \tilde{z}^{(i)} &= z_{\text{norm}}^{(i)} + \gamma \end{aligned}$$

If $\sigma = \sqrt{\sigma^2 + \epsilon}$
 $\beta = \mu \leftarrow$
 then $\hat{z}^{(i)} = z^{(i)}$

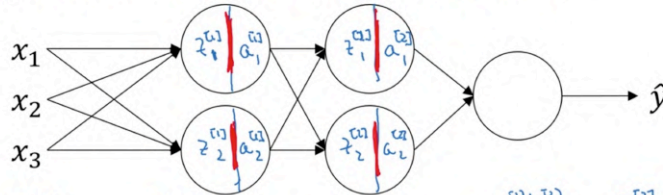
learnable parameters
of model.

Use $\sum_{i=1}^n u(i)$ instl of $\sum_{i=1}^n u(i)$

learnable

Andrew Ng

Adding Batch Norm to a network



$$X \xrightarrow{\omega^{(1)}, b^{(1)}} z^{(1)} \xrightarrow{\beta^{(1)}, \gamma^{(1)}} \tilde{z}^{(1)} \xrightarrow{a^{(1)} = g(\tilde{z}^{(1)})} z^{(1)} \xrightarrow{\omega^{(2)}, b^{(2)}} z^{(2)} \xrightarrow{\beta^{(2)}, \gamma^{(2)}} \tilde{z}^{(2)} \xrightarrow{a^{(2)}} z^{(2)}$$

Parameters: $\left\{ \begin{array}{l} w^{[1]}, \underline{b}^{[1]}, w^{[2]}, b^{[2]}, \dots, w^{[L]}, b^{[L]} \\ \rightarrow \underline{\beta}^{[1]}, \gamma^{[1]}, \underline{\beta}^{[2]}, \gamma^{[2]}, \dots, \underline{\beta}^{[L]}, \gamma^{[L]} \\ \rightarrow \beta \end{array} \right\}$

$$d\beta^{(L)} \quad \beta = \beta^{(L)} - \alpha d\beta^{(L)}$$

tf.nn.batch-normalization ←

Andrew Ng

Working with mini-batches

Working with mini batches

Per minibatch

$$\left\{ \begin{array}{l} \tilde{X}^{\{1\}} \xrightarrow{W^{\{1\}}, b^{\{1\}}} \underline{z^{\{1\}}} \xrightarrow[\text{BN}]{\beta^{\{1\}}, \gamma^{\{1\}}} \underline{\tilde{z}^{\{1\}}} \rightarrow g^{\{1\}}(\tilde{z}^{\{1\}}) \xrightarrow{W^{\{2\}}, b^{\{2\}}} \underline{z^{\{2\}}} \rightarrow \dots \\ \boxed{X^{\{2\}}} \rightarrow \underline{z^{\{2\}}} \xrightarrow[\text{BN}]{\beta^{\{2\}}, \gamma^{\{2\}}} \underline{\tilde{z}^{\{2\}}} \rightarrow \dots \\ X^{\{2\}} \rightarrow \dots \end{array} \right.$$

Parameters: $\omega^{T(2)}$, $\beta^{T(2)}$, $\sigma^{T(2)}$.

$$\begin{matrix} \uparrow \\ (n^{122}, 1) \end{matrix} \quad (n^{122}, 1) \quad (n^{122}, 1)$$

$$\rightarrow \underline{z^{[2]}} = W^{[2]} a^{[1-1]} + \cancel{b^{[2]}}$$

$$z^{[2]} = W^{[2]} a^{[1-1]}$$

$$\rightarrow \frac{N(z)}{Z} = \frac{1}{Z} \int \mathcal{D}z \, e^{-\beta H(z)} \quad \leftarrow$$

Andrew Ng

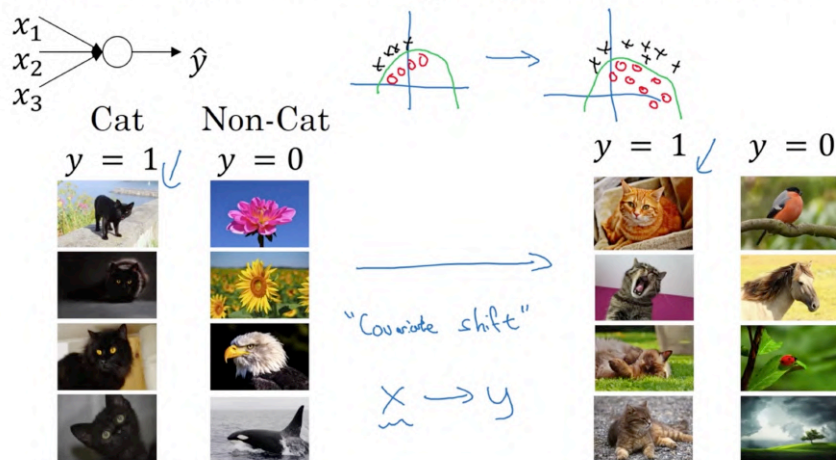
Implementing gradient descent

for $t = 1 \dots \text{num MiniBatches}$
 Compute forward pass on X^{batch} .
 In each hidden layer, use BN to replace z^{hid} with \hat{z}^{hid} .
 Use backprop & compute $\frac{dL}{dW^{\text{hid}}}$, ~~$\frac{dL}{dW^{\text{hid}}}$~~ , $\frac{dL}{d\beta^{\text{hid}}}$, $\frac{dL}{d\gamma^{\text{hid}}}$.
 Update params $\left. \begin{aligned} W^{\text{hid}} &:= W^{\text{hid}} - \alpha \frac{dL}{dW^{\text{hid}}} \\ \beta^{\text{hid}} &:= \beta^{\text{hid}} - \alpha \frac{dL}{d\beta^{\text{hid}}} \\ \gamma^{\text{hid}} &:= \dots \end{aligned} \right\} \leftarrow$

Works w/ momentum, RMSprop, Adam.

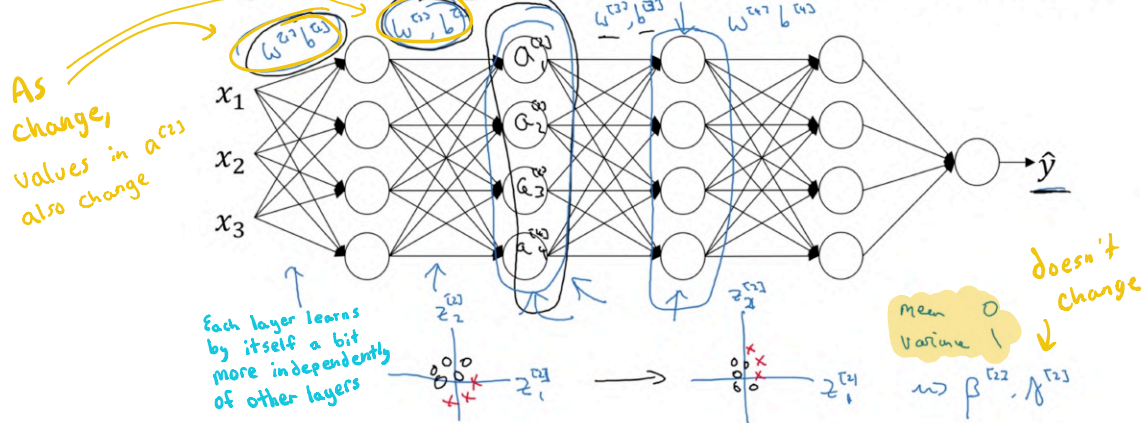
Andrew Ng

Learning on shifting input distribution



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 with dropout for greater regularization effect of batch norm

I wouldn't really use batch norm as a regularizer, that's really not the intent of batch norm, but sometimes it has this extra intended or unintended effect on your learning algorithm. But, really, don't turn to batch norm as a regularization. Use it as a way to normalize your hidden units activations and therefore speed up learning. And I think the regularization is an almost unintended side effect

Andrew Ng

Batch Norm at test time

$$\begin{aligned} \mu &= \frac{1}{m} \sum_i z^{(i)} \\ \sigma^2 &= \frac{1}{m} \sum_i (z^{(i)} - \mu)^2 \\ z_{\text{norm}}^{(i)} &= \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \epsilon}} \\ \tilde{z}^{(i)} &= \gamma z_{\text{norm}}^{(i)} + \beta \end{aligned}$$

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

$x^{[2]}, x^{[3]}, \dots$

$\mu^{[2]}, \mu^{[3]}, \dots$

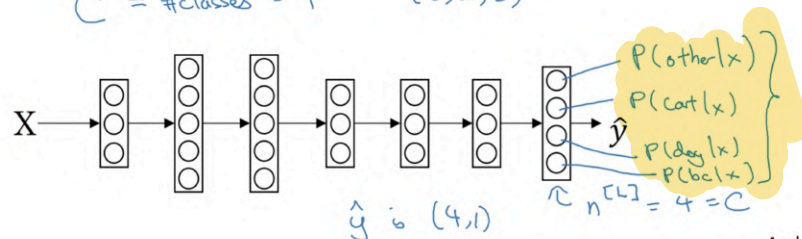
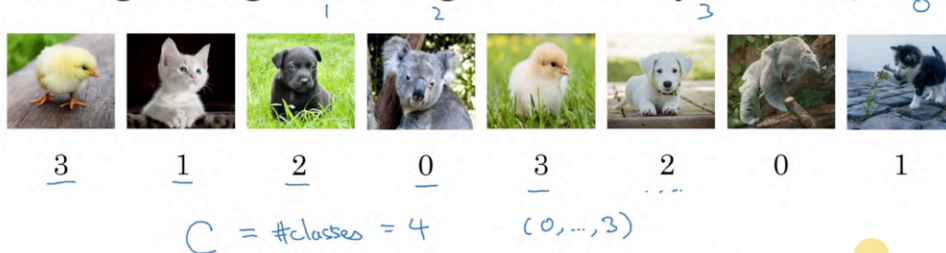
$\sigma_1^2, \sigma_2^2, \sigma_3^2, \dots$

$\tilde{z}_{\text{norm}} = \frac{z - \mu}{\sqrt{\sigma^2 + \epsilon}}$

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

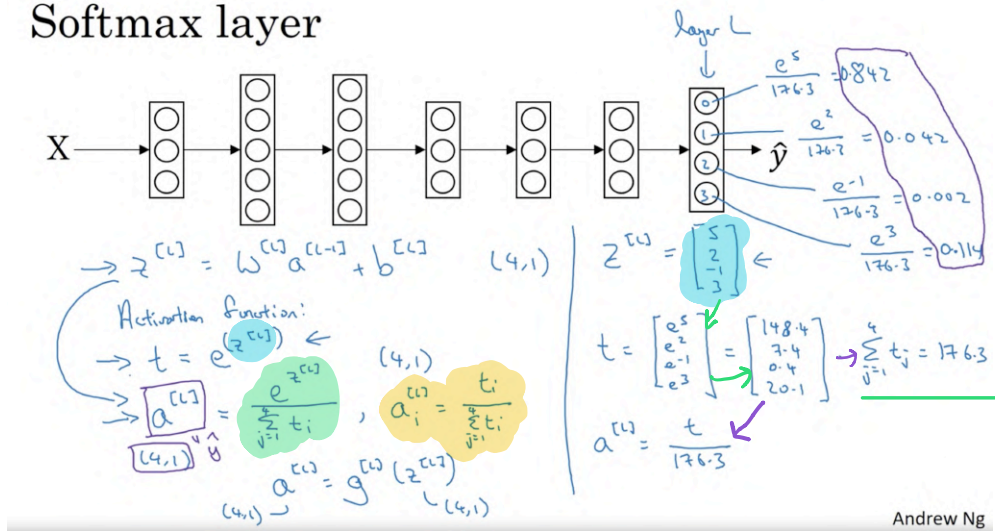
Andrew Ng

Recognizing cats, dogs, and baby chicks, other

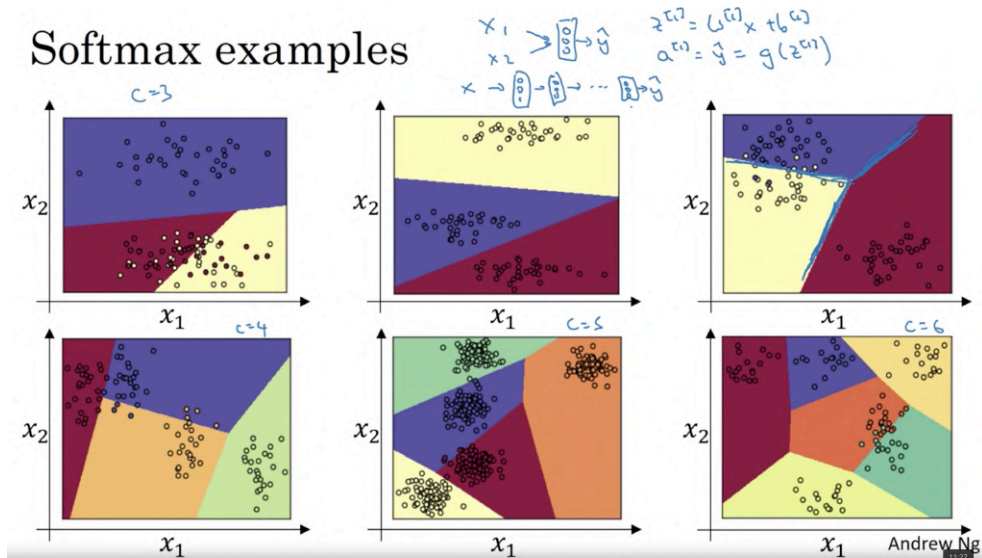


Andrew Ng

Softmax layer



Softmax examples



Understanding softmax

$$z^{(L)} = \begin{bmatrix} 5 \\ 2 \\ -1 \\ 3 \end{bmatrix} \quad t = \begin{bmatrix} e^5 \\ e^2 \\ e^{-1} \\ e^3 \end{bmatrix}$$

Handwritten notes: $C=4$, $g^{(L)}(\cdot)$

Soft max:

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

Hard max:

Softmax regression generalizes logistic regression to C classes.

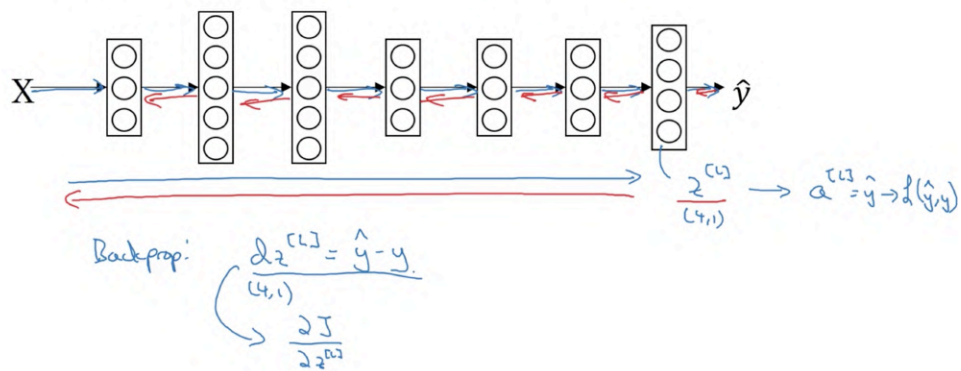
If $C=2$, softmax reduces to logistic regression. $a^{(L)} = \begin{bmatrix} 0.842 \\ 0.158 \end{bmatrix}$

Loss function

$(4,1)$
 $y = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ - cat $y_2 = 1$
 $y_1 = y_2 = y_3 = y_4 = 0$
 $(4,1)$
 $\hat{y} = \begin{bmatrix} 0.3 \\ 0.2 \\ 0.1 \\ 0.4 \end{bmatrix}$
 $C = 4$
 $\mathcal{L}(\hat{y}, y) = - \sum_{j=1}^C y_j \log \hat{y}_j$
 $\underbrace{\mathcal{L}(\hat{y}, y)}_{\text{small}} = - \sum_{j=1}^C y_j \log \hat{y}_j$
 $- y_2 \log \hat{y}_2 = - \log \hat{y}_2 \rightarrow \text{make } \hat{y}_2 \text{ big.}$
 $\mathcal{J}(w^{(1)}, b^{(1)}, \dots) = \frac{1}{n} \sum_{i=1}^n \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$
 $Y = [y^{(1)} \ y^{(2)} \ \dots \ y^{(n)}]$
 $\hat{Y} = [\hat{y}^{(1)} \ \dots \ \hat{y}^{(n)}]$
 $= \begin{bmatrix} 0 & 0 & 1 & 0 & \dots \\ 1 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \end{bmatrix}$
 $(4, m)$
 $= \begin{bmatrix} 0.3 & \dots & \dots \\ 0.2 & \dots & \dots \\ 0.1 & \dots & \dots \\ 0.4 & \dots & \dots \end{bmatrix}$
 $(4, m)$

Andrew Ng

Gradient descent with softmax



Andrew Ng