deeplearning.ai

Hyperparameter tuning

Tuning process

Hyperparameters

first importance one circled in red
then second the one in orange
third in importance is the one in blue
He never tunes the parameters for adam algo optim, just use those as defoult.

this is the most important hyperparameter to tune. Next most imp is the momentum term. → α

β ~ 0.9 momentum term beta if you are using momentum

$\beta_1, \beta_2, \varepsilon$ for adam optimization algo

$0.9 \quad 0.999 \quad 10^{-8}$

layers maybe you have to pick the number of layer

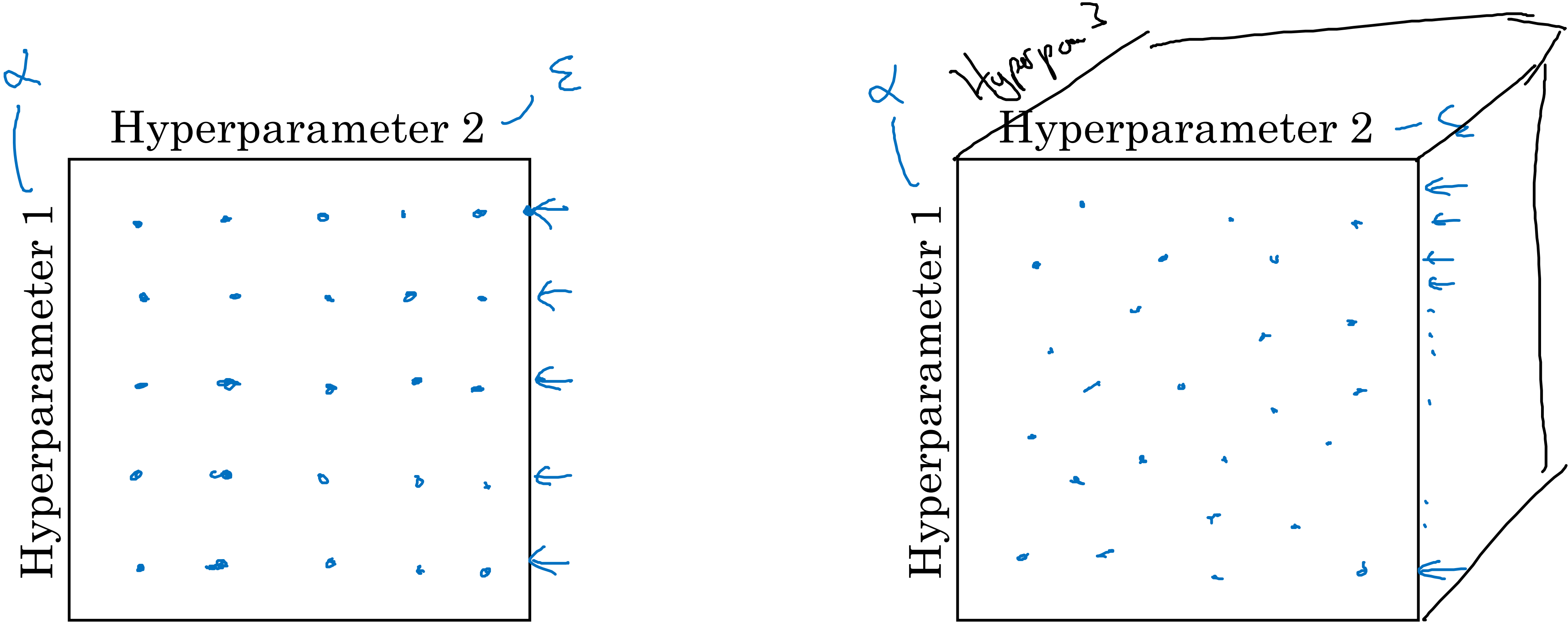
hidden units maybe you have to pick the number of units

learning rate decay

mini-batch size you might need to choose the mini-batch size

HOW DO YOU SELECT A SET OF VALUES TO EXPLORE???????
IN EARLY GENERATIONS IF YOU HAD TWO HYPERPARAMETERS IT WAS COMMON PRACTICE TO SAMPLE THE POINTS IN A GRID LIKE BELOW AND SYSTEMATICALLY EXPLORE THESE VALUES, YOU TRY OUT ALL 25 VALUES(CAN BE MORE) AND PICK WHICHEVER HYPERPARAMETER WORKS BEST. THIS WORKED WHEN NR OF HYPERPARAMETERS WAS SMALL.

Try random values: Don’t use a grid

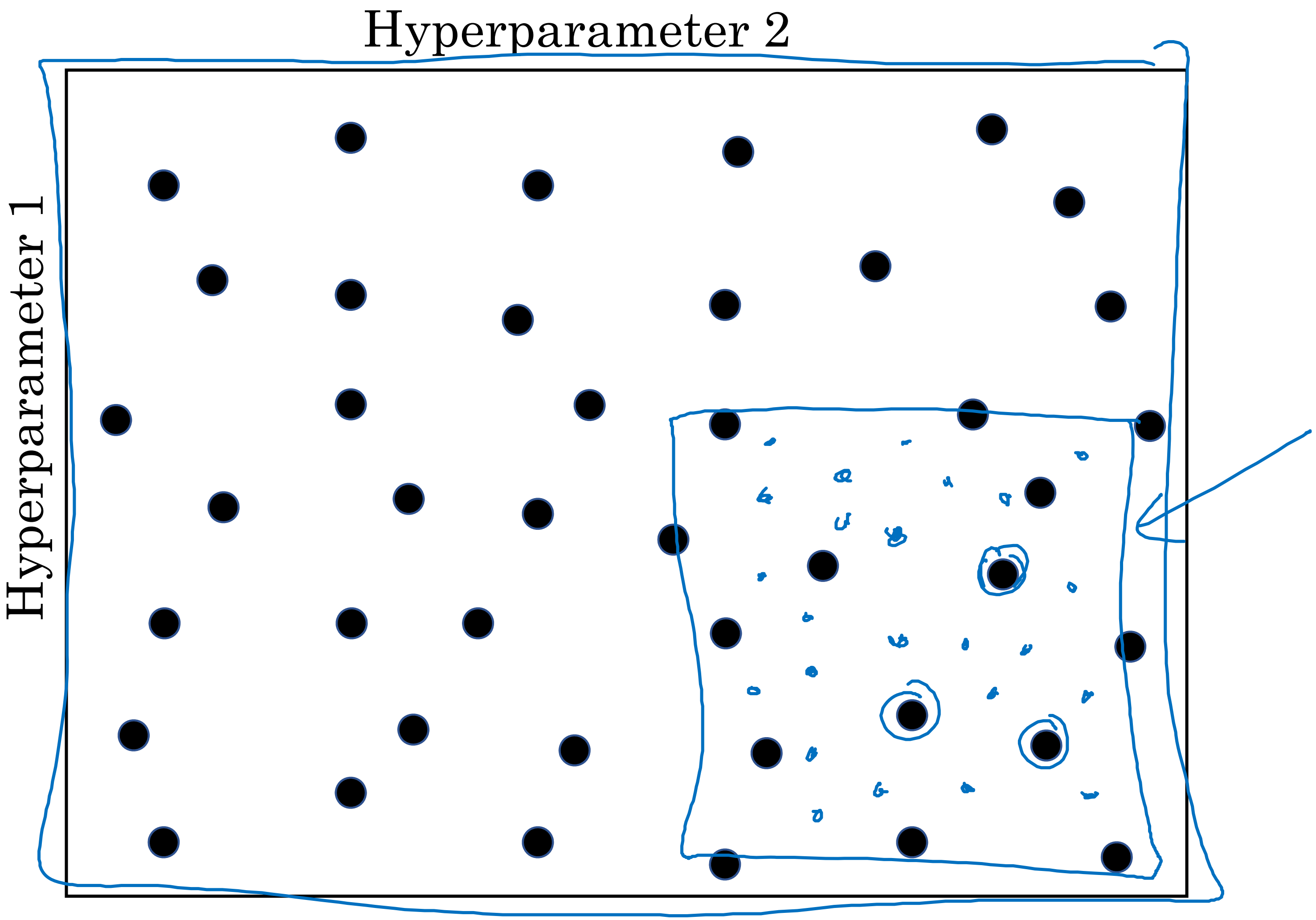


IN DEEP LEARNING WHAT WE TEND TO DO IS TO CHOOSE POINTS AT RANDOM. SO U CHOOSE 25 POITS AT RANDOM AND THEN TRY OUT HYPERPARAMETER ON THIS RANDOMLY CHOSEN SET OF POINTS. ITS DIFFICULT TO KNOW IN ADVANCE WHICH HYPERPARAMETER ARE GOING TO BE THE MOST IMPORTANT FOR YOUR PROBLEM, AND AS WE SAW PREVIEWSLY SOME HYPERPARAMETERS ARE MORE IMPORTNT THEN OTHERS
THATS SAY THE FIRST HYPERPARAMETER IS ALFA AND THE SECOND IS EPSILON THAT IS IN DENOMINATOR OF ADAM ALGO
SO YOUR CHOISE OF ALFA METTERS A LOT WHLE CHOISE OF EPSILON HARDLY. SO WHEN YOU SAMPLE IN THE GRID YOU TRY OUT FIVE VALUES OF ALFA AND YOU MIGHT FIND THAT ALL OF THE DIFFERENT VALUES OF EPSILON GIVE YOU ESSENTIALLLY THE SAME ANSWER.
SO NOW YOU HAVE TRAINED 25 MODELS AND ONLY GOT INTO TRIAL FIVE VALUES FOR THE LEARNING RATE ALFA, WHICH I THINK IS REALLY IMPORTANT.
WHEREAS IN CONTRAST, IF YOU WERE TO SAMPLE AT RANDOM YOU WILL HAVE TRIED OUT 25 DISTINCT VALUES AND THEREFORE YOU WILL BE MORE LIKELY TO FIND A VALUE THAT WORKS REALLY WELL. WE EXPLAINED WITH JUST TWO HYPERPARAMETERS WHILE YOU CAN FACE THIS WITH MANY PARAMETERS.
ITS HARD TO KNOW IN ADVANCE WHICH HYPERPARAMETER ARE REALLY IMPORTANT FOR OUR APPLICATION.

WHEN YOU SAMPLE HYPERPARAMETERS ANOTHER COMMON PRACTICE IS TO USE A COARSE TO FINE SAMPLING SCHEME. SO THATS SAY IN THIS TWO DIMENTIONAL EXAMPLE THAT YOU SAMPLE THESE POINTS AND YOU FIND THAT THIS POINT WORKS BEST AND A FEW AROUND ALSO WORK OK.

IN COARSE OF THE FINAL SCHEME WHAT YOU MIGHT DO IS ZOOM IN TO A SMALLER REGION OF HYPERPARAMETERS AND SAMPLE MORE DENSLY.

Coarse to fine



Andrew Ng

WE SAW HOW SAMPLING OVER A RANGE OF HYPERPARAMETERS CAN ALLOW YOU TO SEARCH OVER THE SPACE OF HYPERPARAMETERS MORE EFFICIENTLY. IT TURNS OUT THAT SAMPLING AT RANDOM DOES NOT MEAN SAMPLING UNIFORMLY AT RANDOM OVER THE RANGE OF VALUES, INSTEAD IS IMPORTATNT TO PICK THE APPROPRIATE SCALE ON WHICH TO EXPLORE THE HYPERPARAMETERS, IN THIS VIDEO I WANT TO SHOW YOU HOW TO DO THAT.

deeplearning.ai

Hyperparameter tuning

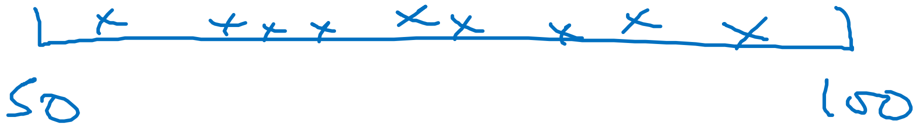
Using an appropriate scale to pick hyperparameters

LETS SAY YOU ARE TRYING TO CHOOSE THE NUMBER OF HIDDEN UNITS n[l] FOR A GIVEN LAYER L

Picking hyperparameters at random

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

SAY YOU THINK A GOOD RANGE OF VALUES IS 50 - 100



PICKING HERE UNIF RAND MIGHT BE OK

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

YOU THINK NUMBER OF LAYER IS BETWEEN 2 TO 4

$2, 3, 4$

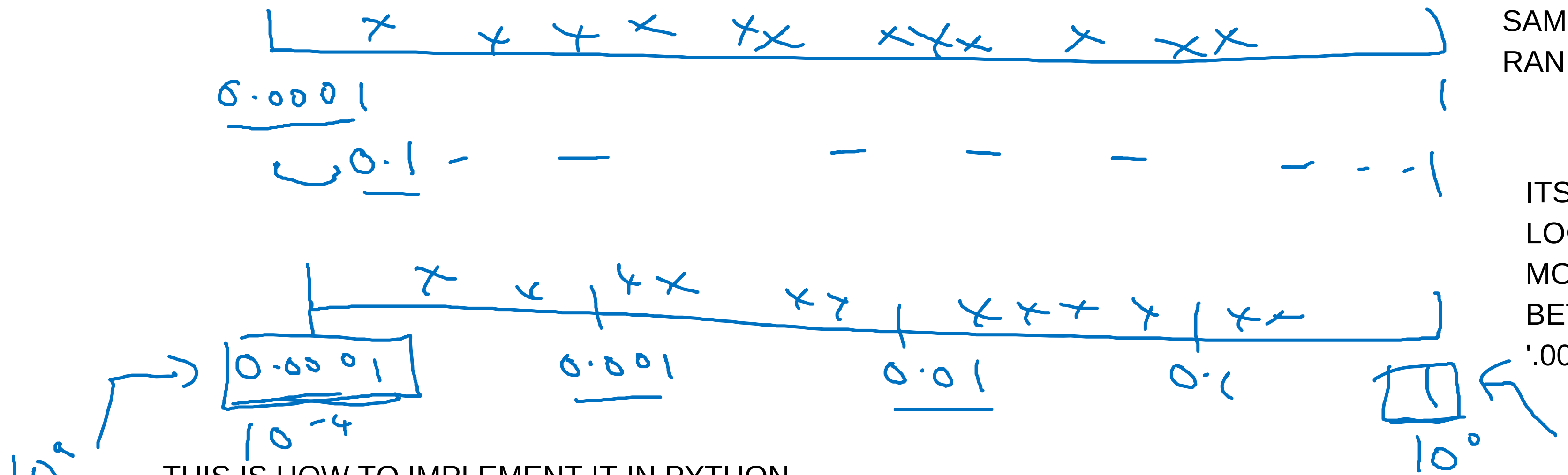
ALSO HERE PICKING AT RANDOM
ITS OK, ALSO BUILDING A GRID SEARCH
WHERE EXPLICITLY EVALUATE THE VALUES
2 3 4 ARE MIGHT BE REASONABLE
THIS IS NOT TRUE FOR ALL HYPERPARAMETERS

LETS LOOK AT ANOTHER EXAMPLE. SAY YOU ARE SEARCHING FOR LEARNING RATE

Appropriate scale for hyperparameters

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

THIS IS RANGE YOU SUSPECT



SAMPLING UNIFORMLY AT RANDOM

ITS BETTER TO SEARCH ON A LOG SCALE AND YOU HAVE MORE RESOURCES SEARCHING BETWEEN 0.001 ND 0.001 AND '.001

THIS IS HOW TO IMPLEMENT IT IN PYTHON

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

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

r will be rand nr between -4 and 0

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

alfa will ne rand nr between above values

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

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

Andrew Ng

Hyperparameters for exponentially weighted averages

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

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

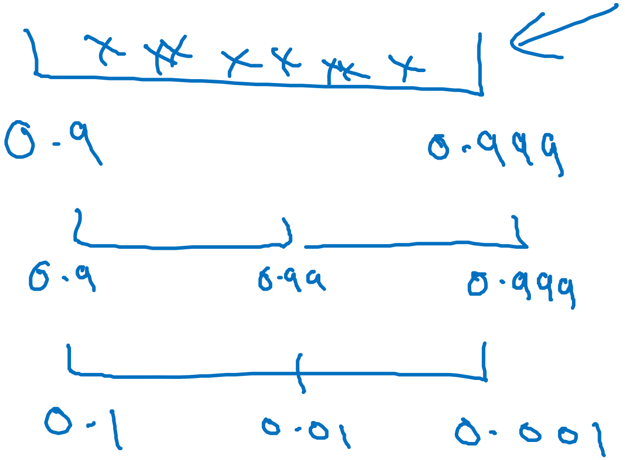
$$10 \qquad\qquad\qquad 1000$$

0.9 is like averaging over 10 last days

while 0.999 is like averaging over 1000 days.

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

you suspect beta has this range



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

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

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

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

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

$$\beta: \underset{\sim 1000}{0.999} \rightarrow \underset{\sim 2000}{0.9995}$$

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

we need to give more weight to changes near 0.999 since it has higher implications in the number of averaging days.

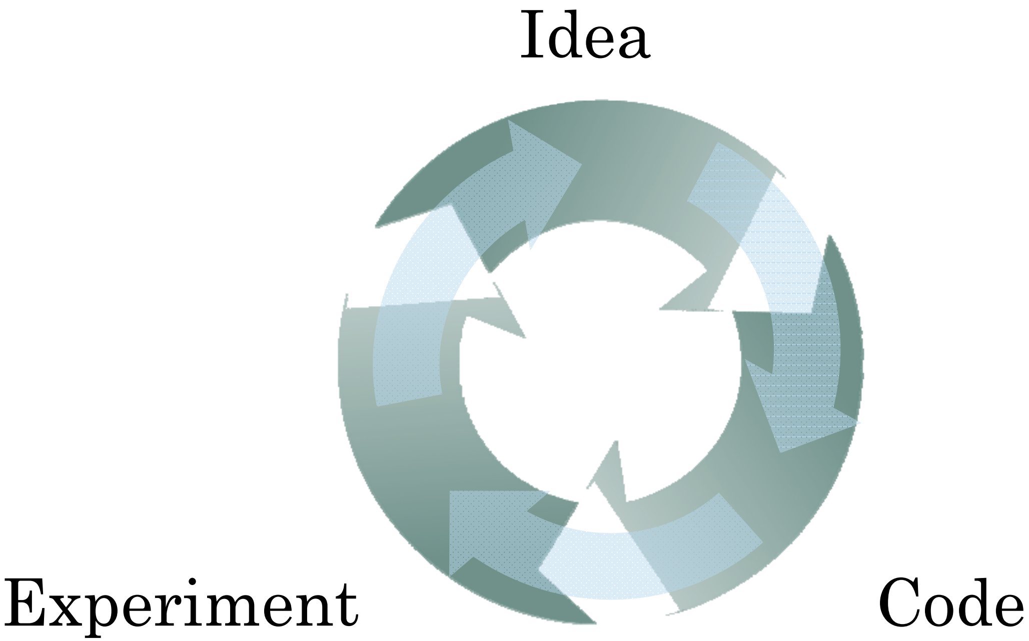
deeplearning.ai

Hyperparameters tuning

Hyperparameters tuning in practice: Pandas vs. Caviar

final tips and trick on how to search hyperparameters

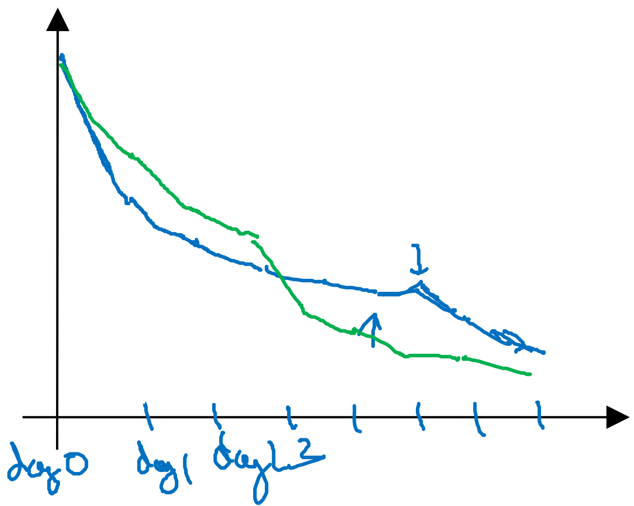
Re-test hyperparameters occasionally



- NLP, Vision, Speech, Ads, logistics, ….
- Intuitions do get stale. Re-evaluate occasionally.

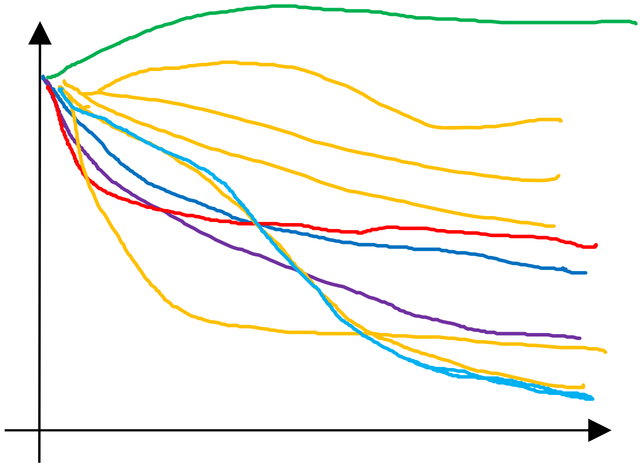
Andrew Ng

Babysitting one model



Panda

Training many models in parallel



Caviar

Andrew Ng

Pand approach if u dont have much CPU otherwise go for caviar approach.

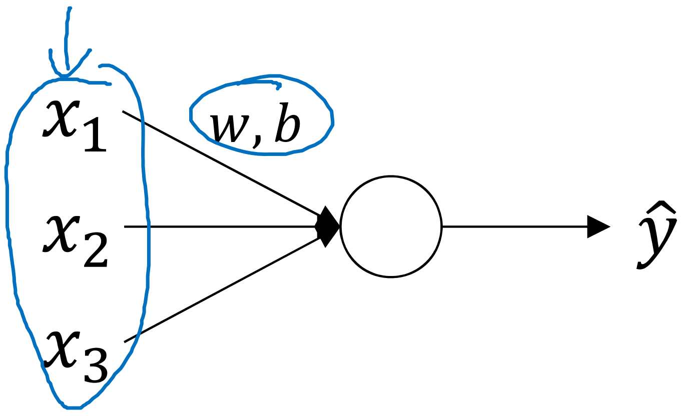
This makes the search for the hyperparameter much easy and makes the NN much more rubust. The choise of hyperparameters is a much bigger rnge of hyperparameters that work and also allows to train very deeo NN.
Does not work allways for all NN but when it works it can make training very fast.

deeplearning.ai

Batch Normalization

Normalizing activations in a network

Normalizing inputs to speed up learning



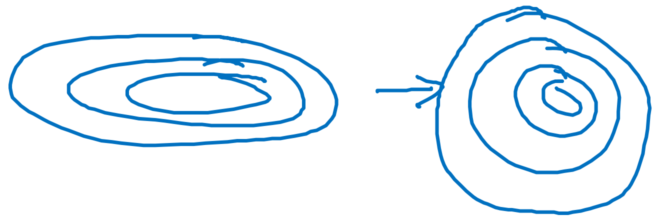
$$\mu = \frac{1}{m} \sum_i x^{(i)}$$

$$X = X - \mu$$

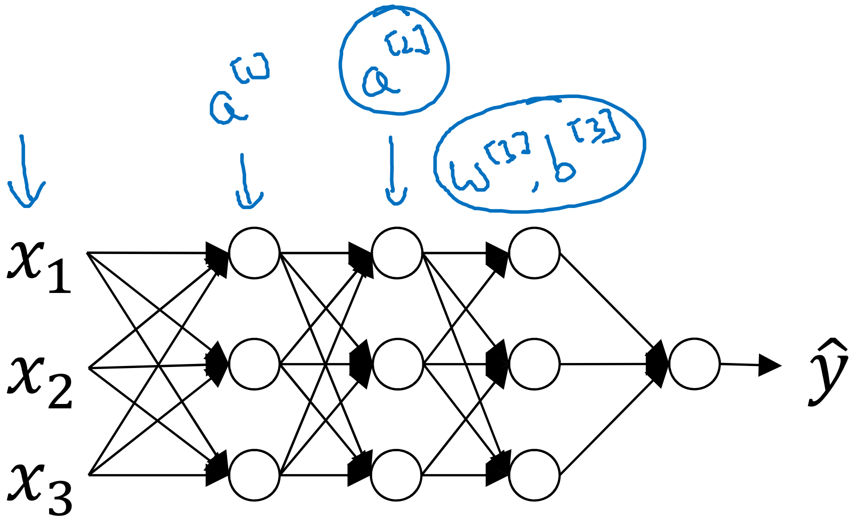
$$\sigma^2 = \frac{1}{m} \sum_i x^{(i)2} \leftarrow \text{element-wise}$$

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

Normalizing imput features can speed up learning



For any hidden layer can we normalize the values of any hidden layer to train faster



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

Normalize $\underline{z^{[2]}}$

↑

should we normalize the value before activation function, so z2 or whether after applying activation function a2.

In practice normalizing z2 is done much often, so thats the version I'll present.

So here is how to implement batch norm to normalize the values in the hidden layer.

Implementing Batch Norm

given some intermediate values in NN say those z1 z2 ...

Given some intermediate values in NN $\underbrace{\overset{\downarrow}{z^{(1)}}, \ldots, \overset{\downarrow}{z^{(m)}}}_{z^{\boxed{[l]}(i)}}$

we gone ommit this l for semplification

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

compute mean var

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

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

← normalize

epsilon for numerical stability

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

learnable parameters of model.

If

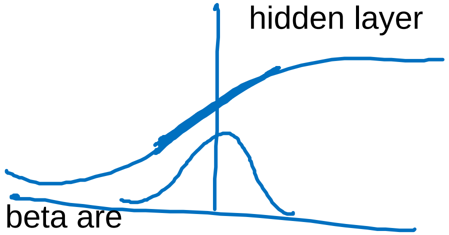
$$\gamma = \boxed{\sqrt{\sigma^2+\varepsilon}} \leftarrow$$

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

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

X ← we saw how normalizing features x helps learning algo

$\underline{z^{(i)}}$ ← What BatchNorm does it applys that to values in hidden layer



we dont want hidden layers to have mean 0 and var 1 allways, so we compute z tilda. gamma and beta are learnable parameters of your model.

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

The effect of gamma and beta lets you set the mean and va fo z i whatever you want.

Here gammma and beta are learnable parameters of your model so if you are using grad descent or rms prop or adam you would update the weights of your nn just like for the weights of the NN.
The effect of gamma end beta is that it allows you to set of z tilde to be whatever you want it to be.
What it does its ensures that the hidden units have standardized mean and variance where mean and var are controlled by gamma and beta parameters that can make mean 0 and var 1 or other values as well, so it normalize hidden values to fixed mean and var, can be 0 and 1 or some other value

We see how to fit Batch norm ito a deep NN and how to make it work for the many different layers of NN
We will also give some intuition why Batch norm to train the NN.

We have seen how to fit batch norm for a single hidden layer thats see how it fits in the training of a deep network

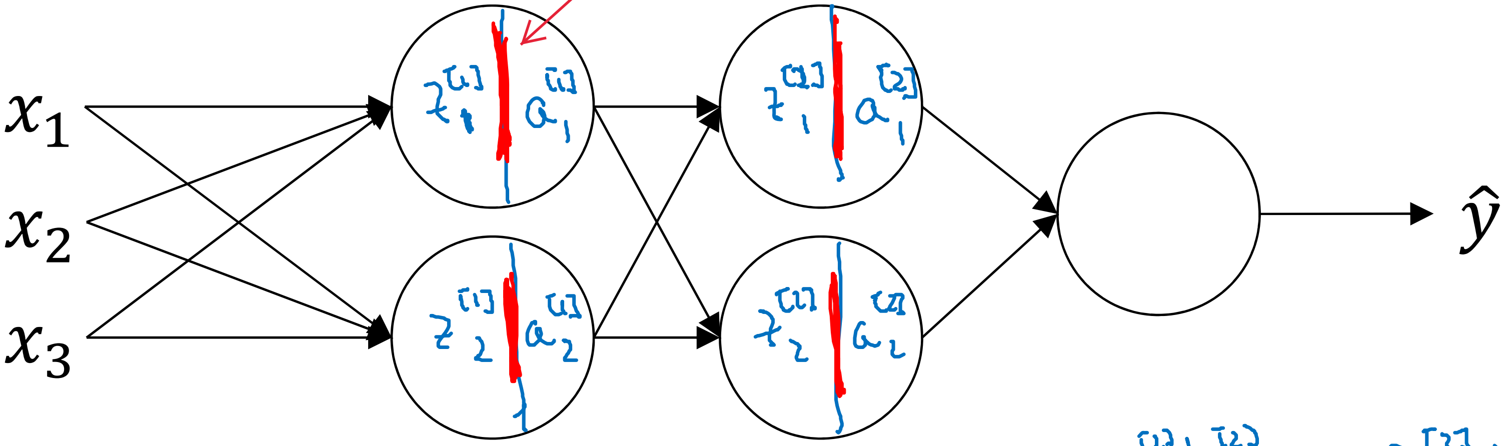
deeplearning.ai

Batch Normalization

Fitting Batch Norm into a neural network

So first calcs the z and then applies activation function to compute a

Adding Batch Norm to a network



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

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

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

$\rightarrow \beta$

$d\beta^{[l]}$

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

using tensorflow to implement batch norm:

tf.nn.batch-normalization ←

these betas have nothing to do with the betas that we computed for momentum or expon weighted avg.
You can use grad descent or rms prop or adam to update the betas.

If you are using a Deep Learning programming framework usually you wont have to implement the Batch Norm step on Batch Norm layer yourself.
In tensorflow you can implement Batch Normalization with the above function.

Now so far, we've talked about Batch Norm as if you were training on your entire training site at the time as if you are using Batch gradient descent. In practice Batch Norm is usually applied with mini-batches of your training set.

Working with mini-batches

So the way u apply Batch Norm is you take your first mini batch and compute z1.

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

first min batch

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

second mini batch

you update the parameters beta gamma on every mini batch

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

third mini batch

but this z l is gone be normalized fist to mean 0 and variance 1 so adding that constant b is irrelevant

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

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

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

n l number of hidden units in layer l

There is no point in having this point b l, is sort replaced by beta l

this is way z l is computed

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

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

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

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

Andrew Ng

So thats put all together and describe how to implement grad descent usikng Batch Norm.
Assume we are using mini- batch grad descent.

Implementing gradient descent

(number of mini batches)

```
for t = 1 .... 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  $W^{[l]} := W^{[l]} - \alpha dW^{[l]}$
                   $\beta^{[l]} := \beta^{[l]} - \alpha d\beta^{[l]}$   $\leftarrow$
                   $\gamma^{[l]} := \dots$
```

Works w/ momentum, RMSprop, Adam.

Works with momentum, RMSprop, Adam where instead of taking this gradient descent update you can use the updates given by these other algorithms as we discussed in the previews week.
So these other optimization algo can be used to update the parameters beta and gamma that Batch Norm added to algorithm.

So we have seen how normalizing imputs can speed up learning.
The intuition is that this is doing a similr think but for values in your hidden units and not just in the inputs.

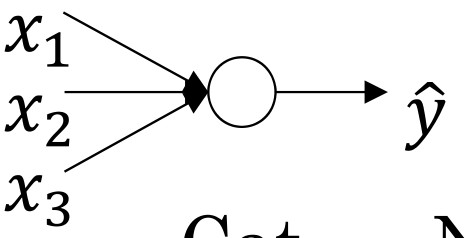
deeplearning.ai

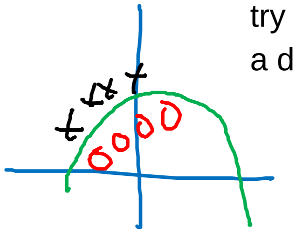
Batch Normalization

Why does Batch Norm work?

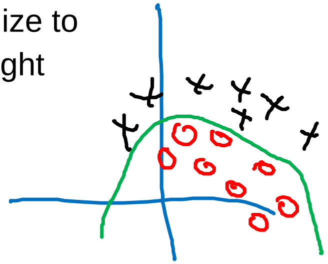
A second reason why batch norm works is it makes weights later or deeper in your network more rubust to changes to weights in earlier layers of the NN, say, in layer one.
To motivate the above that look at this exmple.

Learning on shifting input distribution





try to generalize to a datast on right

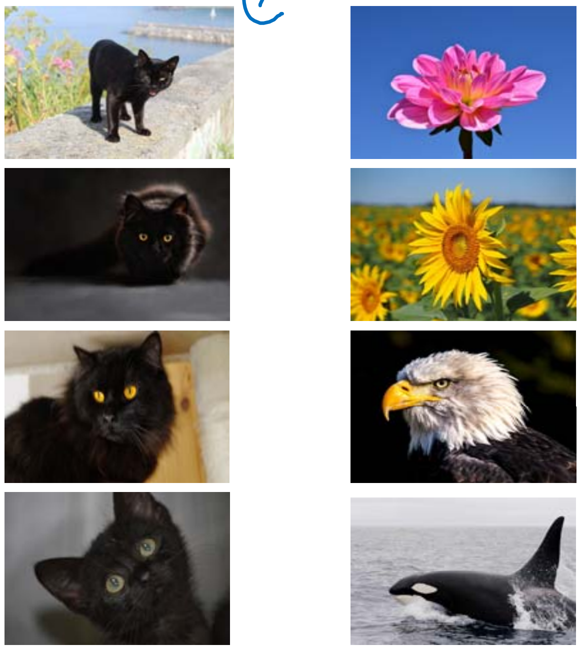


You might not expect a model trained on data on left to do well on data on right. They might be the same function that actually works, but you wouldnt expect your learning algo to discover that green decision boundary just looking to data on left.

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

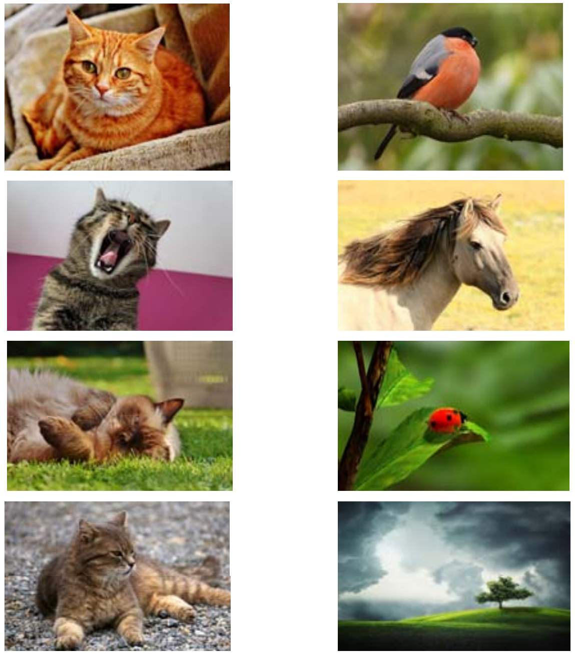
$y = 1$ $y = 0$

So this idea of data distribution changing goes by the name ^covariance shift^



"Covariate shift"

$X \rightarrow y$



Andrew Ng

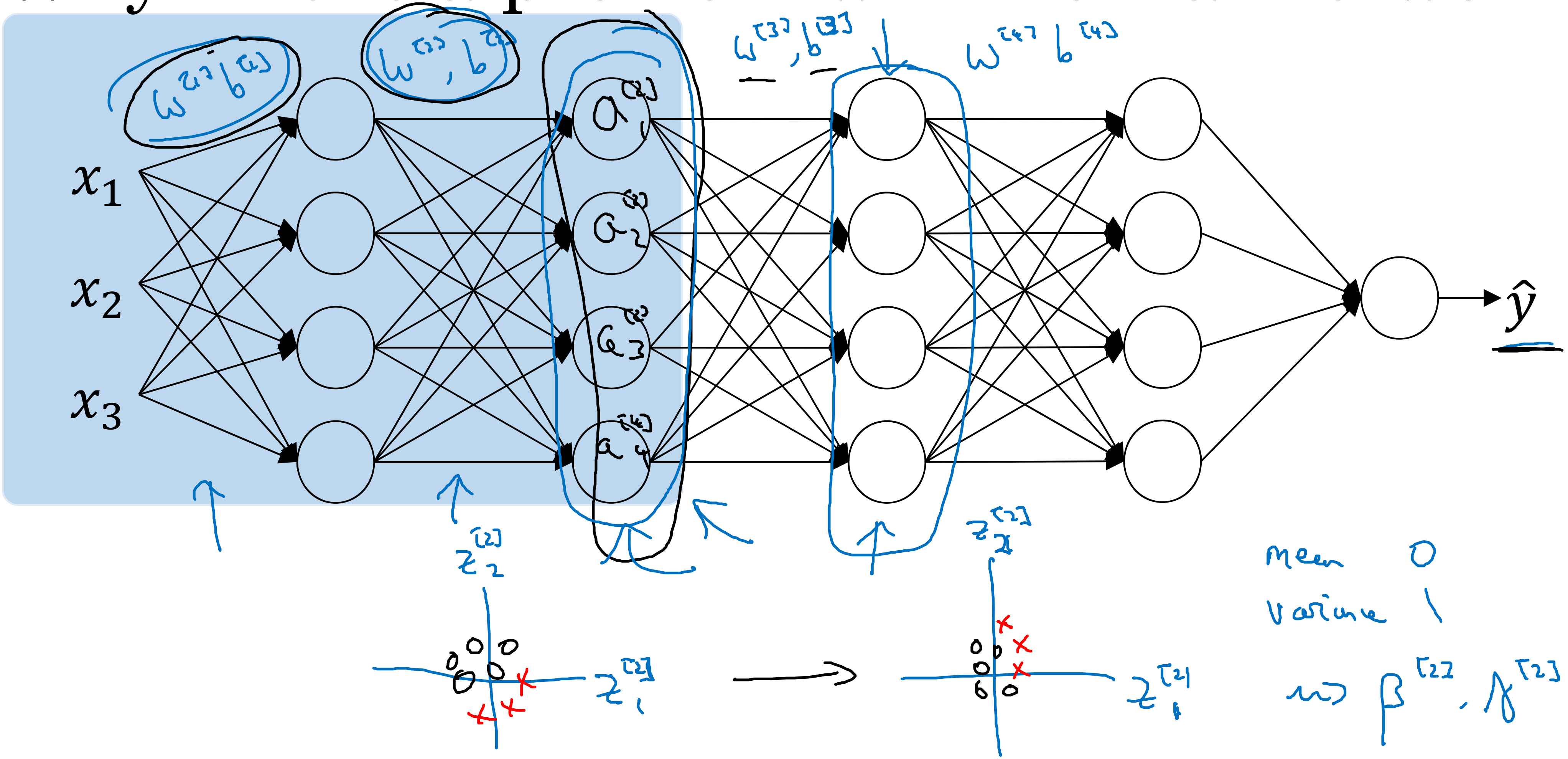
you have trained NN on black cats

If you have learned some X->Y mapping if the distribution of X changes then you might need to retrain your learning algo. This is true even if the function mapping from x to y reamins unchanged, which it is in this example as the function is telling is cat or not.

apply it to colored cats, then classifier might not do well

How does this problem of covariance shif applies to a NN.
Consider a deep network like this, and thats look at the procss from the perspective of this certain layer, the third hidden layer.
So this network has learned the parameters W3 and b3. From the perspective of the third hidden layer it gets some set of values from the earlier layers, and then has to do some stuff to hopefully make the output y close to groud true value.

Why this is a problem with neural networks?



if we conver up part on left of third layer and see from perspective of this third layer.

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 : $\underline{64} \longrightarrow \underline{512}$

Batch norm handles data one mini batch at a time, it computes mean and variance on mini batches. So at test time you try to make predictions, try and evaluate the NN, you might not have a mini batch example, you might be processing one single example at a time, so you might need to do something different to make sure your predictions make sense.

deeplearning.ai

Batch Normalization

Batch Norm at test time

m number of example in the mini batch

Batch Norm at test time

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

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

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

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

At test time you have to compute mean and variance and if you have just one example computing the mean and variance ot that example doesnt make sence.
So we come up with some separate estimate of my and sigma

What u do to estimate those is to comp expon weighted avg the mini batches

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

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

$\cancel{\mu^{[l]}}$

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

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

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

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

So here at test time
we use the mean and var
of the exponentially weighted

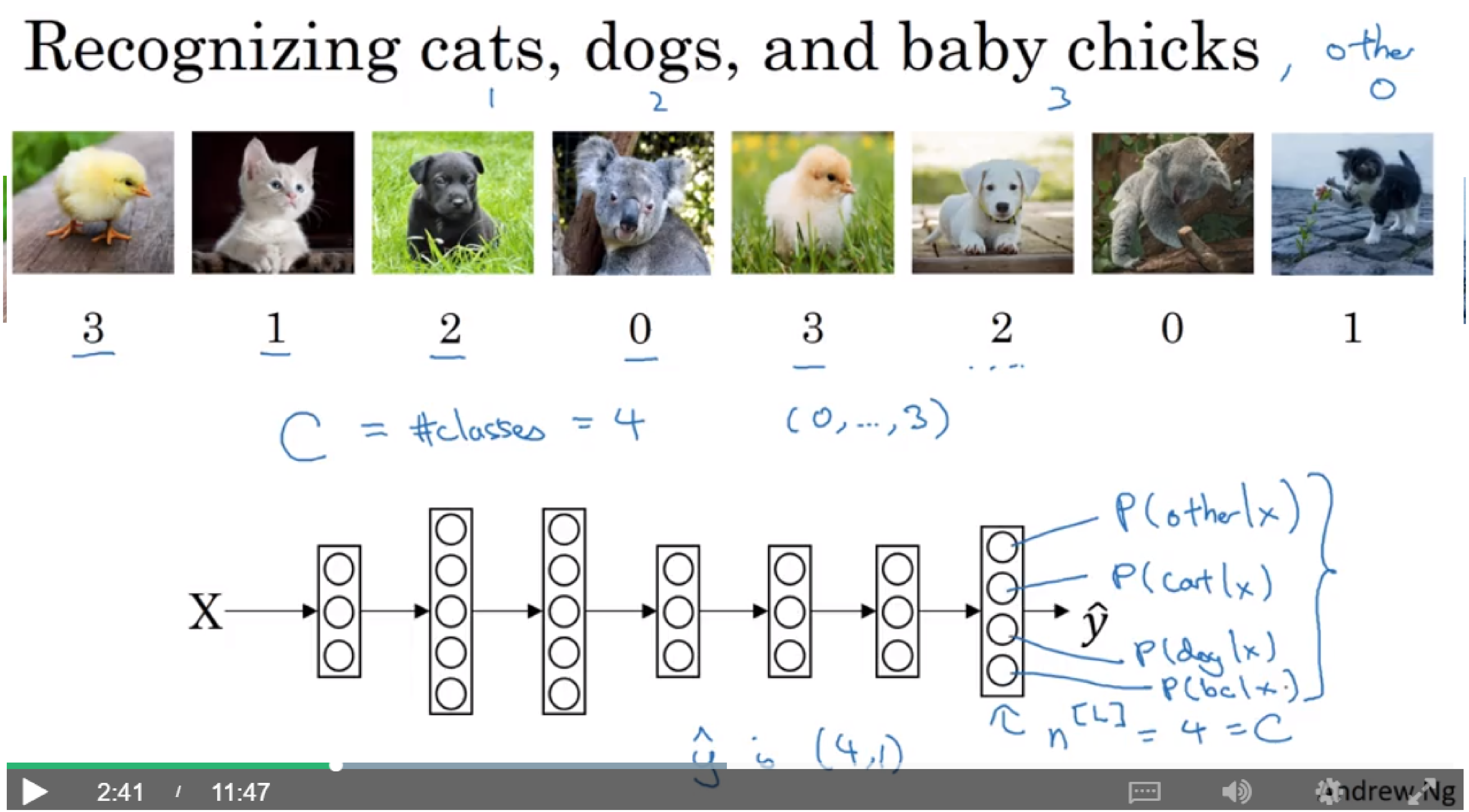
If we use a framework to compute NN we dont need to worry much about this as its easily done there.

Andrew Ng

deeplearning.ai

Multi-class classification

Softmax regression



Recognizing cats, dogs, and baby chicks, other

1 2 3 0

3 1 2 0 3 2 0 1

C = #classes = 4 (0, ..., 3)

$X \rightarrow \hat{y}$

P(other|x)

P(cat|x)

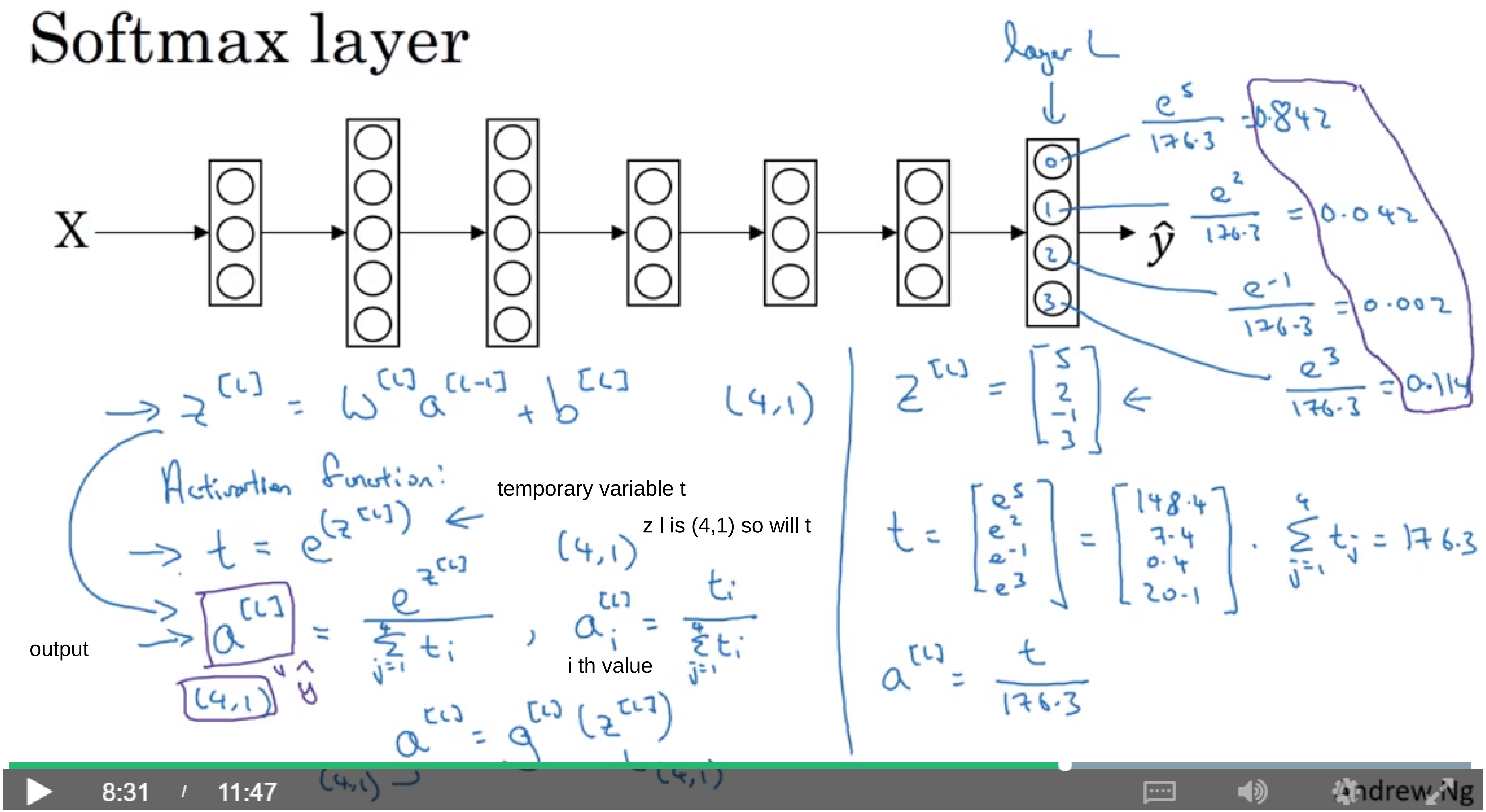
P(dog|x)

P(bc|x)

$n^{[L]} = 4 = C$

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

Andrew Ng



Softmax layer

$$\frac{e^5}{176.3} = 0.842 \qquad \frac{e^2}{176.3} = 0.042 \qquad \frac{e^{-1}}{176.3} = 0.002 \qquad \frac{e^3}{176.3} = 0.114$$

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

Activation function:

temporary variable t

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

z l is (4,1) so will t

output

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

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

i th value

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

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

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

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

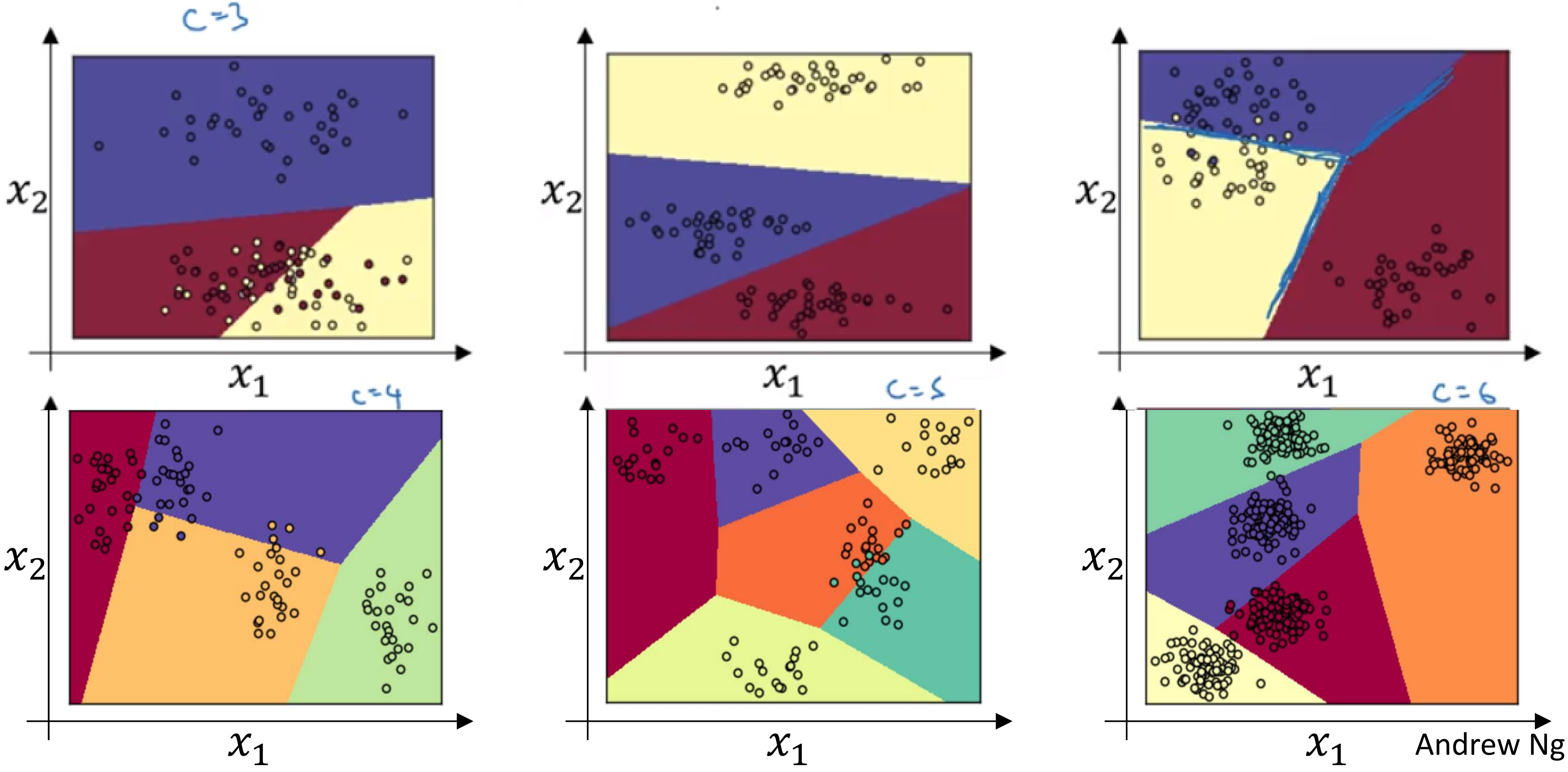
Andrew Ng

So it just described the activation function.
The only strange think is takes a vector and outputs a vector

we gone show an example where you have two inputs x1 and x2 and they serve to an output y

Softmax examples

$z^{[l]} = W^{[l]}x + b^{[l]}$
$a^{[l]} = \hat{y} = g(z^{[l]})$



Andrew Ng

To be noticed that the decision boudary is linear.

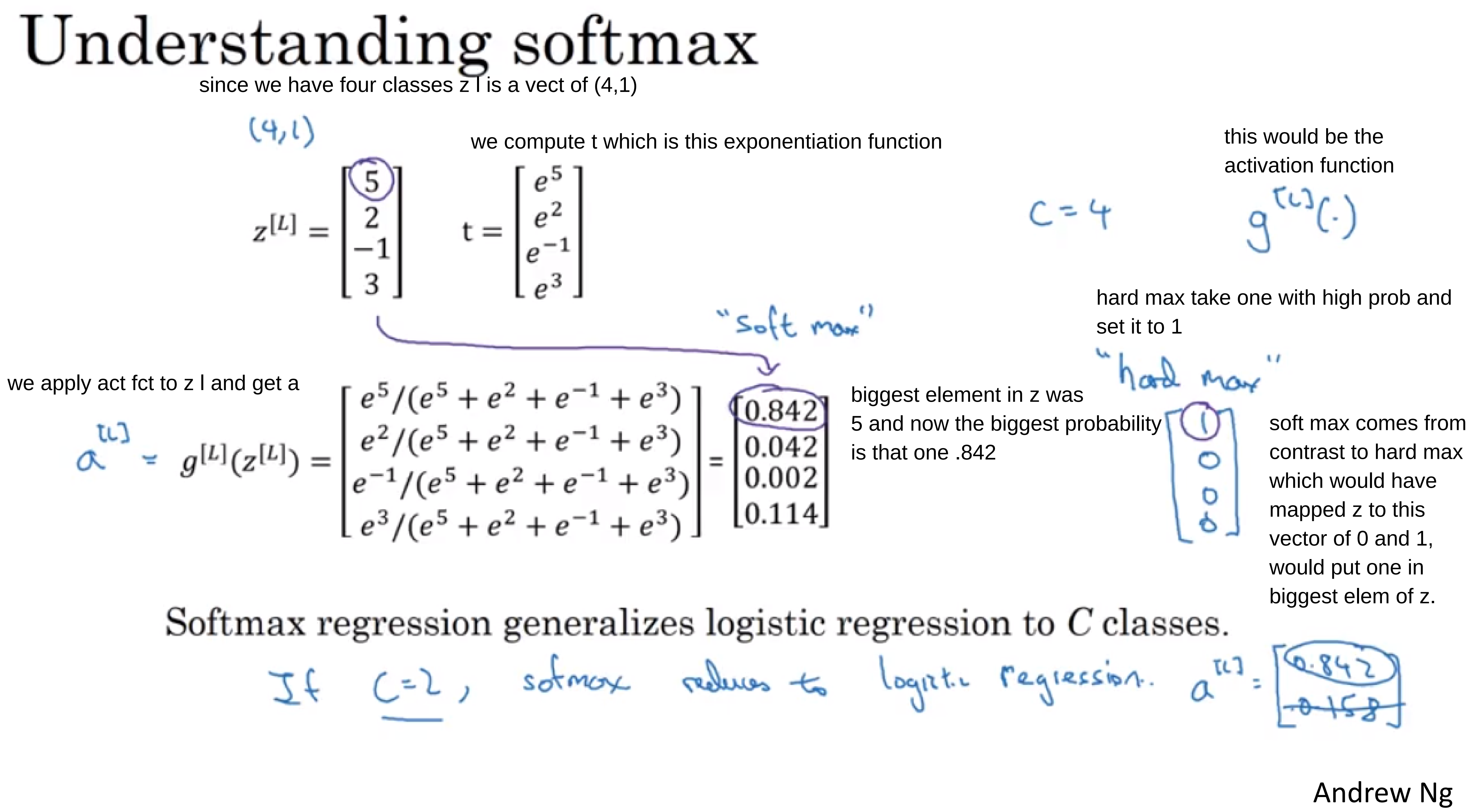
We saw ssome example of soft max and the activation function of the softmax classifier.

We will see how to train a NN that uses a Softmax layer

deeplearning.ai

Multi-class classification

Trying a softmax classifier



Understanding softmax

since we have four classes z l is a vect of (4,1)

we compute t which is this exponentiation function

this would be the activation function

$(4,1)$

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

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

hard max take one with high prob and set it to 1

"soft max"

"hard max"

we apply act fct to z l and get a

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

biggest element in z was 5 and now the biggest probability is that one .842

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

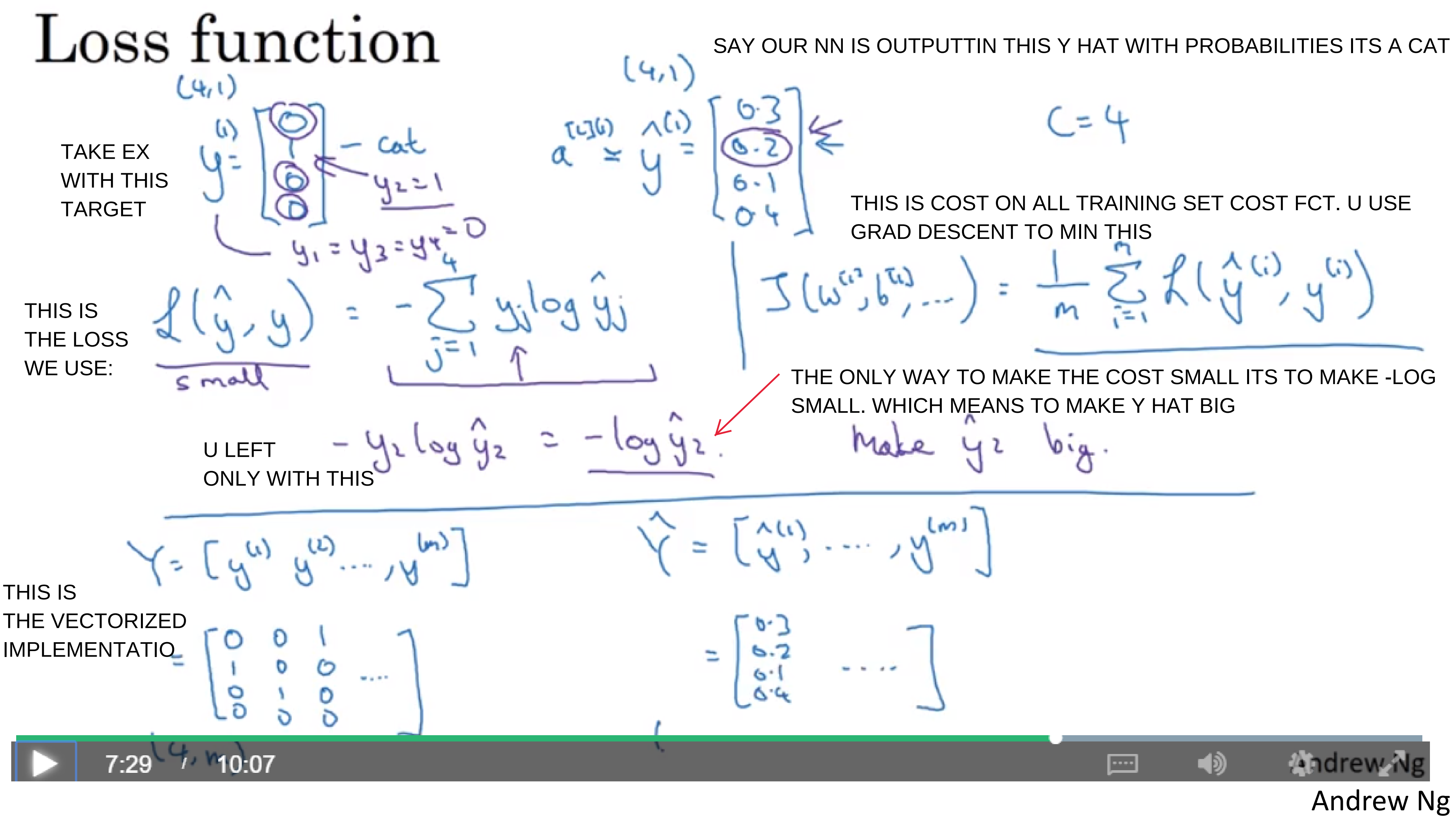
soft max comes from contrast to hard max which would have mapped z to this vector of 0 and 1, would put one in biggest elem of z.

Softmax regression generalizes logistic regression to C classes.

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

Andrew Ng

Thats see how you would train NN with softmax.
Thats define the loss function u use to train the nn



Loss function

SAY OUR NN IS OUTPUTTIN THIS Y HAT WITH PROBABILITIES ITS A CAT

TAKE EX WITH THIS TARGET

$$\underset{(4,1)}{y^{(i)}} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad - \text{cat} \quad \underline{y_2 = 1} \qquad y_1 = y_3 = y_4 = 0$$

$$a^{[L](i)} = \hat{y}^{(i)} = \underset{(4,1)}{\begin{bmatrix} 0.3 \\ 0.2 \\ 0.1 \\ 0.4 \end{bmatrix}} \qquad C = 4$$

THIS IS COST ON ALL TRAINING SET COST FCT. U USE GRAD DESCENT TO MIN THIS

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

THIS IS THE LOSS WE USE:

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

THE ONLY WAY TO MAKE THE COST SMALL ITS TO MAKE -LOG SMALL. WHICH MEANS TO MAKE Y HAT BIG

U LEFT ONLY WITH THIS

$$- y_2 \log \hat{y}_2 = \underline{- \log \hat{y}_2}. \qquad \text{Make } \hat{y}_2 \text{ big.}$$

THIS IS THE VECTORIZED IMPLEMENTATIO

$$Y = [y^{(1)}\ y^{(2)} \dots, y^{(m)}] = \underset{(4,m)}{\begin{bmatrix} 0 & 0 & 1 & \dots \\ 1 & 0 & 0 & \\ 0 & 1 & 0 & \\ 0 & 0 & 0 & \end{bmatrix}}$$

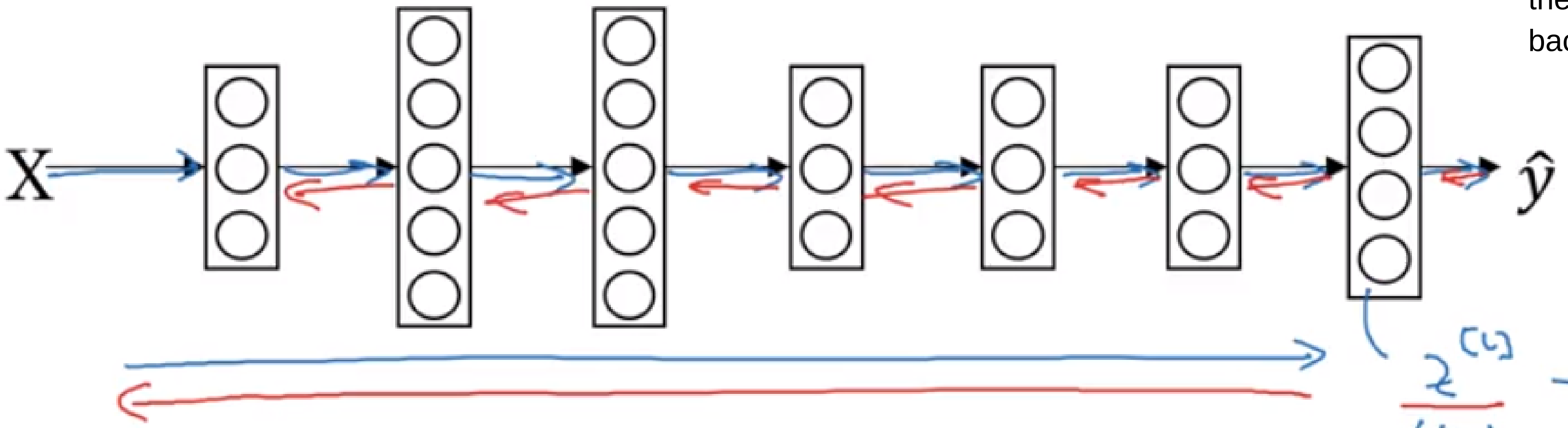
$$\hat{Y} = [\hat{y}^{(1)}, \dots, \hat{y}^{(m)}] = \begin{bmatrix} 0.3 & \dots \\ 0.2 & \\ 0.1 & \\ 0.4 & \end{bmatrix}$$

Andrew Ng

tHATS SEE HOW YOU WOULD IMPLEMENT GRAD DESCENT WHEN YOU HAVE A SOFTMAX OUTPUT LAYER

Gradient descent with softmax

We have talked how to implement the forward pro, what about the backpropagation.



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

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

The key eq you need to initialize backprop is this expression. That the derivative wrt z at the loss layer, this turns out, you can compute as that on left.

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

that dz is our
prtial deriv of the cost fct wrt z

In this week prog exercise we will start using a deep learning prog framework and in this contest you just need to focus on getting the forward pro right. So we dont need bother about deriv calc as the framework takes care of that.

Thats it for soft max classifications where u characterize inputs into not just one of two classes but one of C different classes

Andrew Ng

As you implement more complex models such as CNN and RNN, or as you start to implement very large models that is not practical to implement everythink yourself.

Its good to know how to do matrix multiplicatio but as u go into more complex model you probably dont want to do that, instead u call a numerical linear algebra library that could do it more efficiently for you.

Programming Frameworks

deeplearning.ai

Deep Learning frameworks

Deep learning frameworks

There are many deep learning frameworks, here are some of the leading ones

- 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)

each of these frameworks has a community of users and developers
They are evolving and you can see the pros and cons of each of these ones

deeplearning.ai

Programming Frameworks

TensorFlow

Motivating problem

say we have this simple cost fct to min

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

(cost)

$$(w-5)^2$$

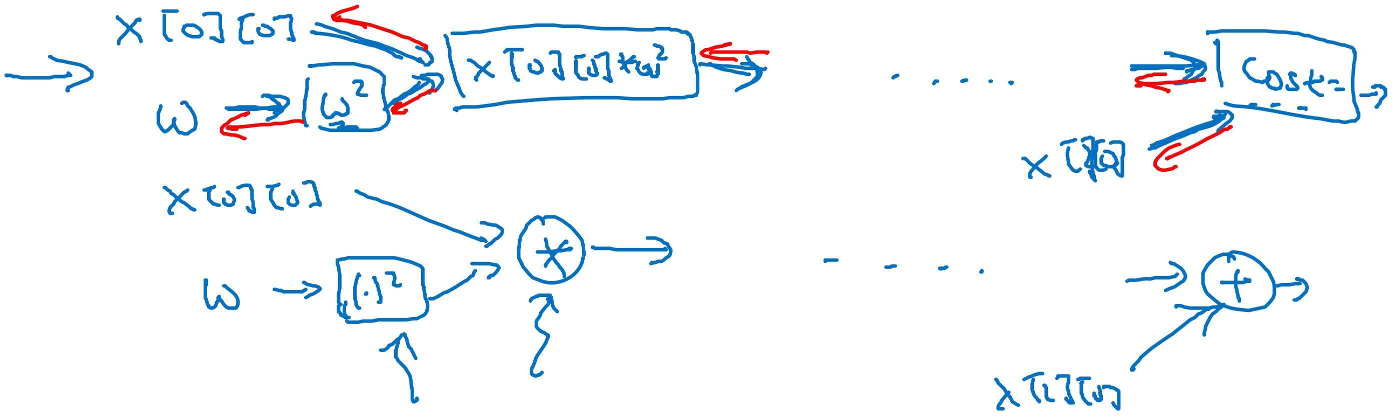
$$w = 5$$

this is the solution but thats
say we dont know that
and we try solve with tesnsorflow

$$J(w, b)$$

↑ ↑

Code example



```
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))

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

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

this is an alternative form of those three lines which are quite idiomatic. does same think.

Andrew Ng