

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>

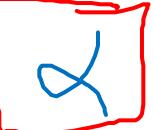
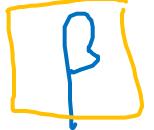


deeplearning.ai

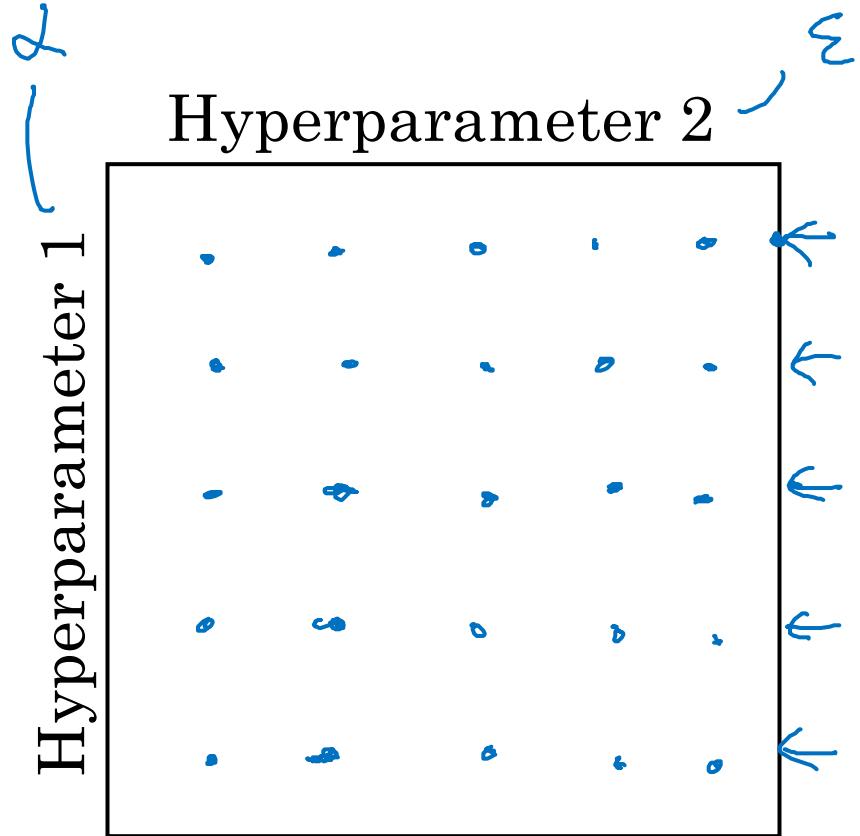
Hyperparameter tuning

Tuning process

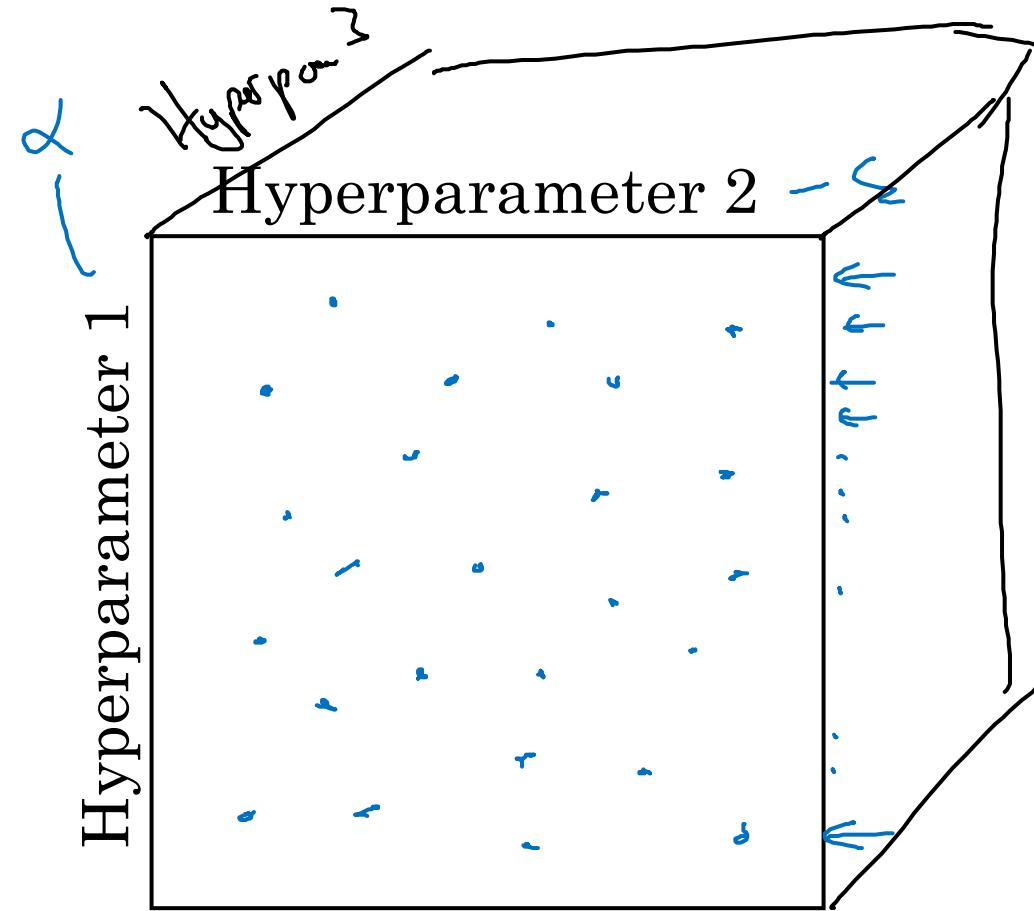
Hyperparameters

-  : 7125 28 (1st)
-  NO. q : 200012 28
- $\beta_1, \beta_2, \epsilon$
 $0.9, 0.999, 10^{-8}$: Adam Optimization parameters
(η , β_1 , β_2 , ϵ)
-  : 300012 28
-  : 2^n
-  : ~3^n
-  : 2^n

Try random values: Don't use a grid



예를 들어, 학습률(Hyperparameter 1)은 learning rate인 α 이고, 손실함수(Hyperparameter 2)는 Adam 알고리즘이다.
이로 정의하는 것은 매우 흔한데 이를 그려주면 다음과 같다:
그림과 같이, 학습률(Hyperparameter 1)은 0에서 25까지, 손실함수(Hyperparameter 2)는 0에서 25까지를 사용해 된다.
이제 차트를 살펴보면, 25x 25인 대로 시각화된다.

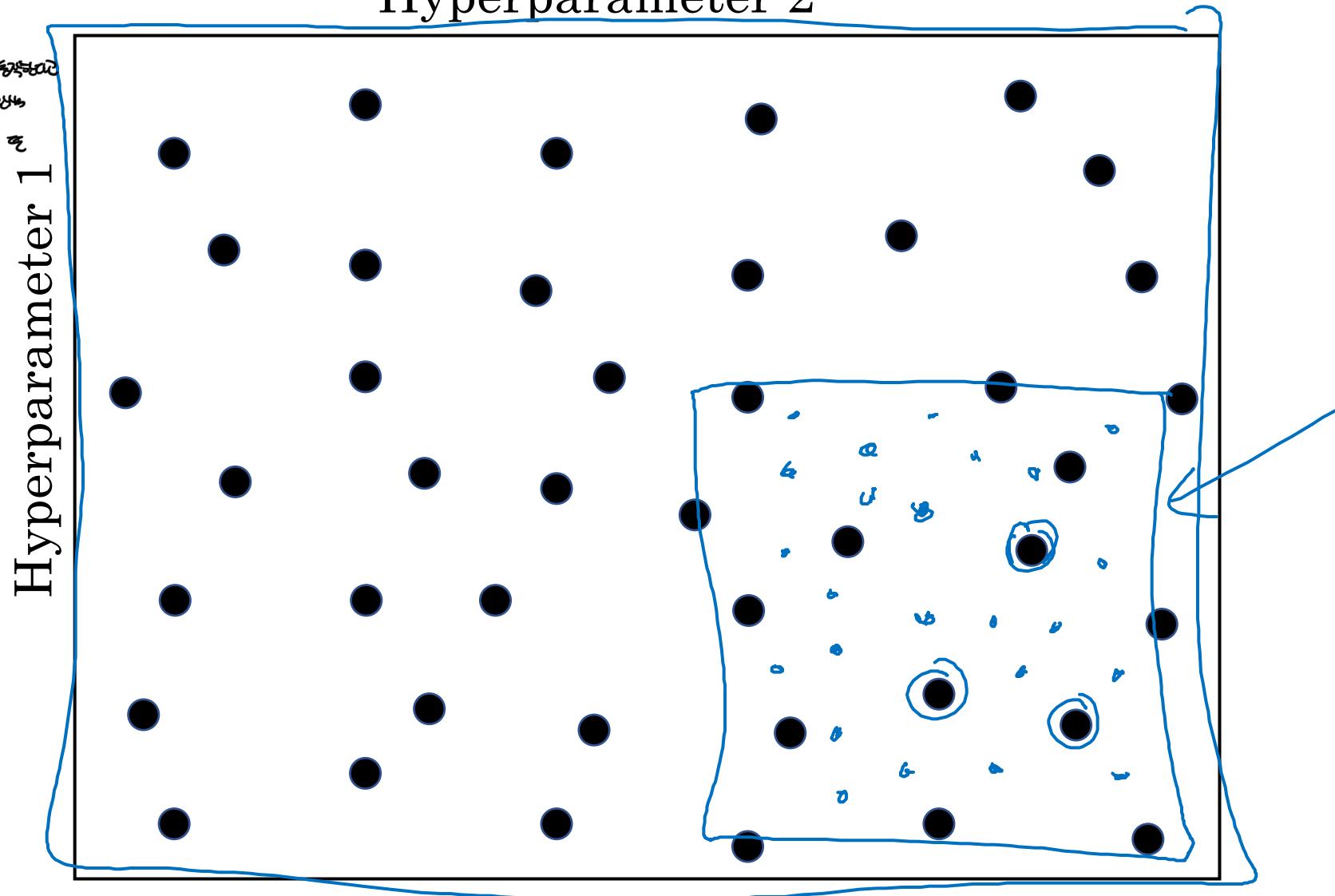


이제 차트를 살펴보면, 25x 25인 대로 시각화된다.
이제 차트를 살펴보면, 25x 25인 대로 시각화된다.
이제 차트를 살펴보면, 25x 25인 대로 시각화된다.

Coarse to fine

coarse to find는 열거의 끝에 위치한
point는 그 이후로 모든 가능한 점을 포함하는
알려진다.
시작하는 점은 다음 점을 찾을 때 사용되는
다시 사용되는 방식이다.

Hyperparameter 2





deeplearning.ai

〈 ပြည်သူများ နေပါတီမှုပညာ နှင့် ဒါန ၏
ပညာနှင့် ပြည်သူများ နေပါတီမှုပညာ နှင့် ဒါန ၏ ပညာ 〉

Hyperparameter tuning

Using an appropriate
scale to pick
hyperparameters

Picking hyperparameters at random

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



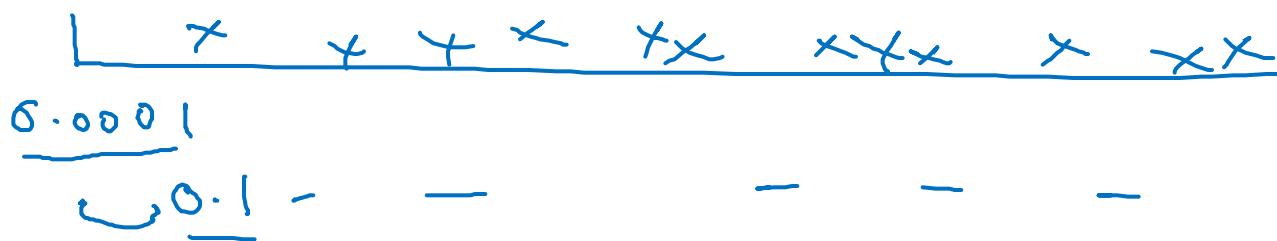
$$\rightarrow \# \text{ layers } L : 2 - 4$$

2, 3, 4

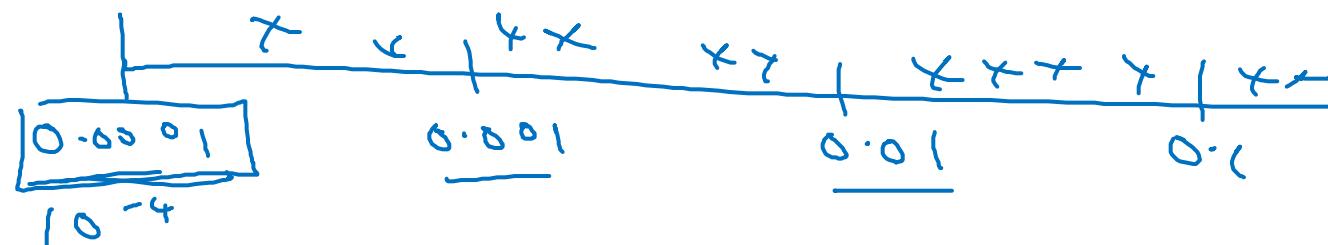
Deep hidden layers may need to be randomly initialized
and also randomly initialized from 0 or 100 or more.
or randomly initialized from 50 or 100 or more.
Memory usage is very large for large networks.
due to size, NN layer. Matrix multiplication is slow.
layer size 2-4 times less computation required.
2-3-4 times less memory required to store.

Appropriate scale for hyperparameters

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



이전에 learning rate alpha를 터무니없이 했었다.
이 경우에는 0.0001이나 1 사이의 값을 임의로 선택하는데
성능을 저해하면, 예 90%의 값인 0.1과 1 사이에 값을 찾고
즉, 성률은 90%. 즉 0.1이나 1 사이의 값이 성률에
맞는 경우, 10% 확률 만큼 0.0001과 0.1 사이의 값이
정답을 찾는다.



$$a = \log_{10} 0.0001 \quad r = -4 * np.random.rand() \quad \leftarrow r \in [-4, 0] \quad \leftarrow 10^{-4} \dots 10^0$$

$$= -4 \quad \alpha = 10^r$$

이전 방법은 성률에 맞지 않았다. 위 방법처럼 터무니
없는 값을 linear scale or also log scale을 사용해
성률을 맞는 경우를 찾는다.

따라서 $-4 + np.random.rand() * -\log(0.0001)$
같은 식으로 선택해 주면, 10% 확률 만큼 0.0001보다
작은 값을 찾는 경우에 성률을 A로 한다.

$$10^a \dots 10^b$$

$$\frac{r \in [a, b]}{[-4, 0]}$$

$$\alpha = 10^r$$

Hyperparameters for exponentially weighted averages

$$\beta = 0.9 \dots 0.999$$

\downarrow \downarrow

1.0 1.000

< Exponentially Weighted Averages 의 β 선택 범위 >
선택 β 값은 0.9 미만 0.999 사이로 선택하는 편이.
이때 선택한 범위는 0.9 ~ 0.99999
여기서 선택한 범위는 대체로 Linear Scale of
이면 log scale 을 선택하는 편이:
가장 좋은 범위는 β 선택 범위
 $\approx 0.001 \sim 0.1$ 사이로 선택하는 편이다.

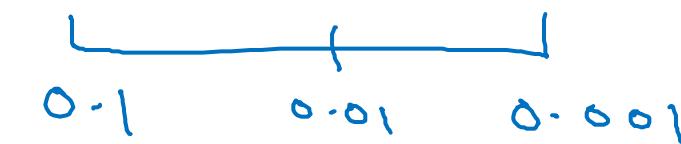
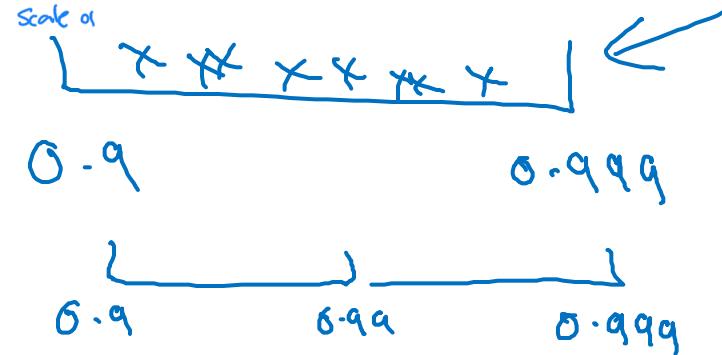
$$1-\beta = 0.1 \dots 0.001$$

$$\beta: 0.900 \rightarrow 0.9005 \quad \} \sim 10$$

$$\beta: 0.999 \rightarrow 0.9995$$

$\sim 1000 \quad \sim 2000$

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



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

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

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

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

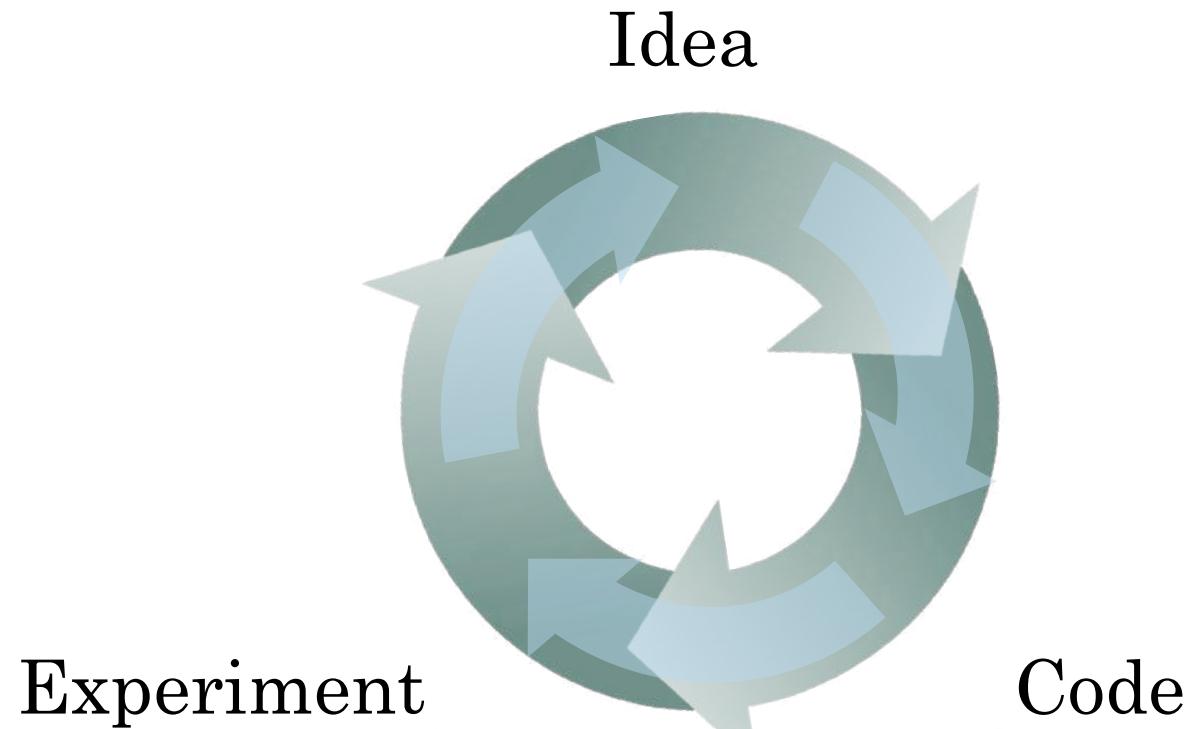


deeplearning.ai

Hyperparameters tuning

Hyperparameters tuning in practice: Pandas vs. Caviar

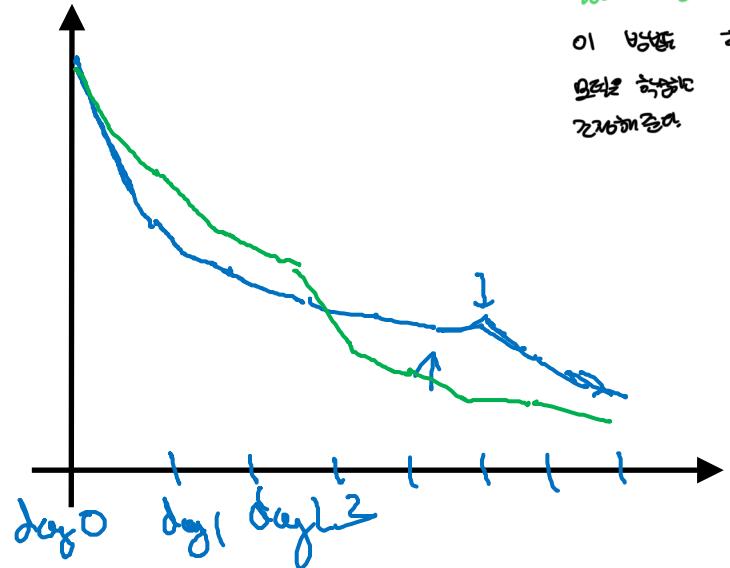
Re-test hyperparameters occasionally



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

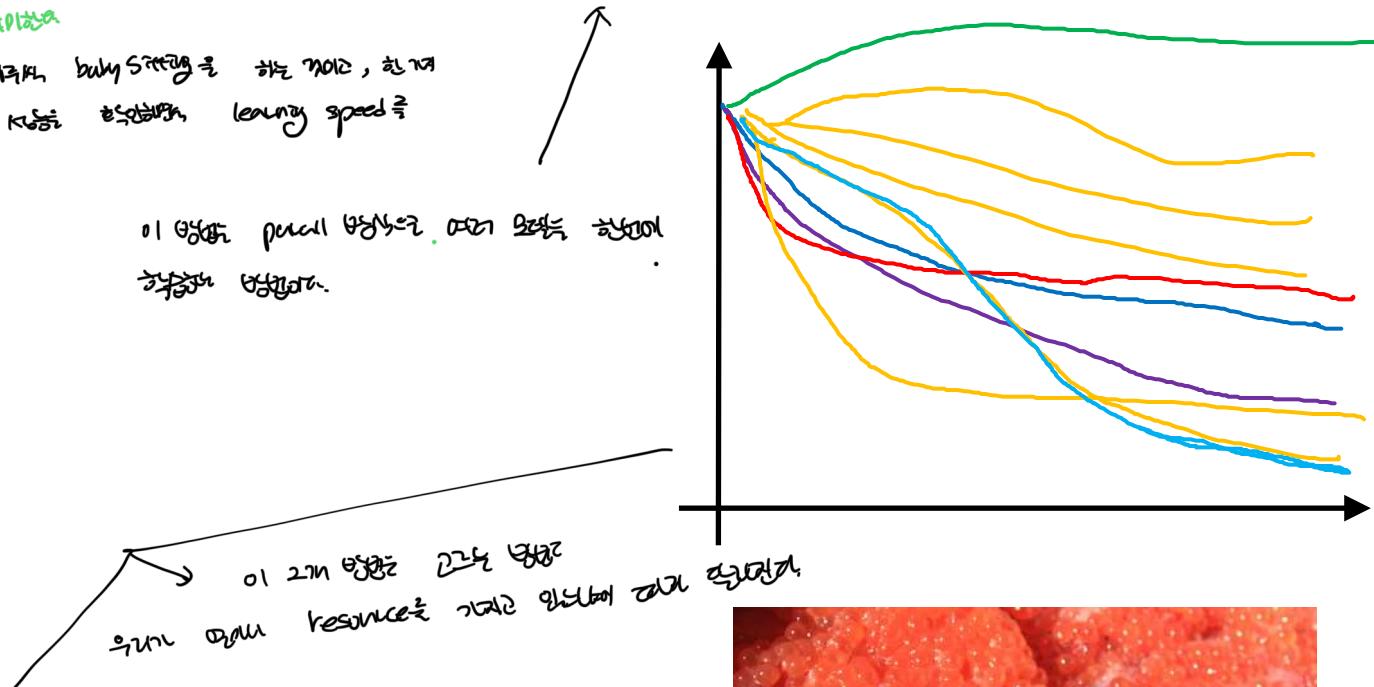
모델을 훈련한 후에는 향후 프로그램의 같은 조건에서도, 특히
구현부의 디테일한 부분이나, 다른 학습률 설정을 고려하는 등에,
이미는 변화를 대비해 올바른 것을 향후 프로그램에 있어 더
이상 필요로 하거나, 혹은 그 외의 이유로 필요로 한다.
그럼 어떤 retesting 시나 향후 프로그램을 re-evaluate 하는지
하면 (예전에 했던)

Babysitting one model



Panda ←

Training many models in parallel

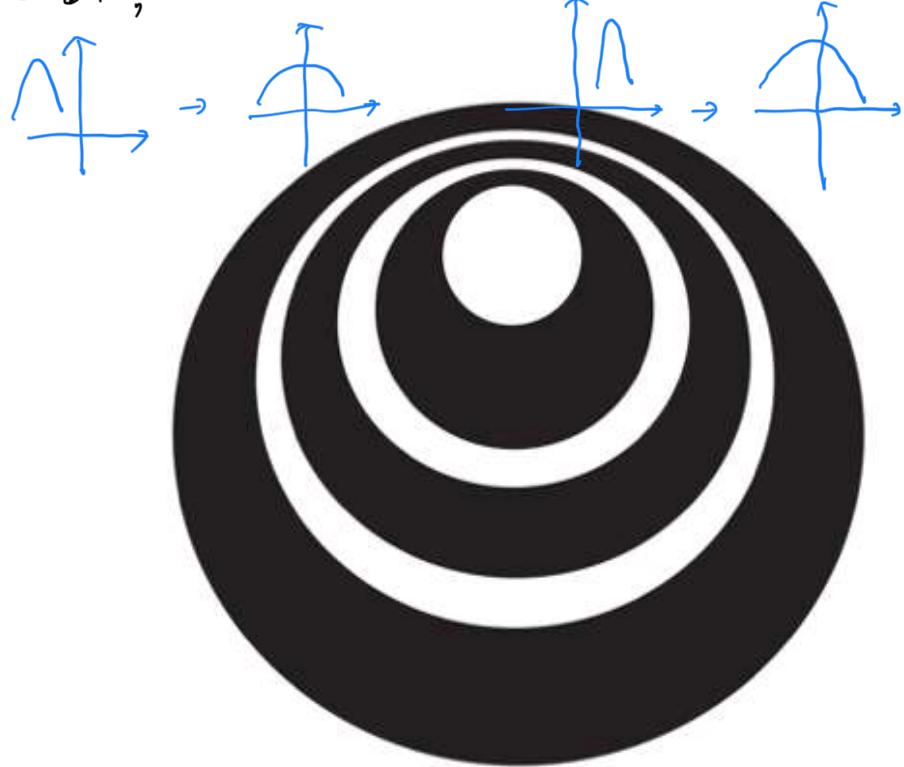


Caviar ←

Andrew Ng

Batch Normalization

: 학습 과정에서 Batch size가 크거나 뉴런 수가 많을 경우에 Gradient의 흐름을
제어하는 데 있어 차이가 생기고 이를 해결하기 위해 Batch Normalization
이라는 방법이 있다. 단계별 차이가 생기면 학습하는 데에 어려움이 생기기
때문에, Batch size가 많거나 뉴런 수가 많을 때에는 사용하는 방법이다.



deeplearning.ai

1) Layer Scale 통일

각각의 레이어의 유통과 양쪽으로 흐르는 모든 차이를 제거해
내려온 모든 Layer의 Feature + 동일한 Scale = 1
이도록 만든다.

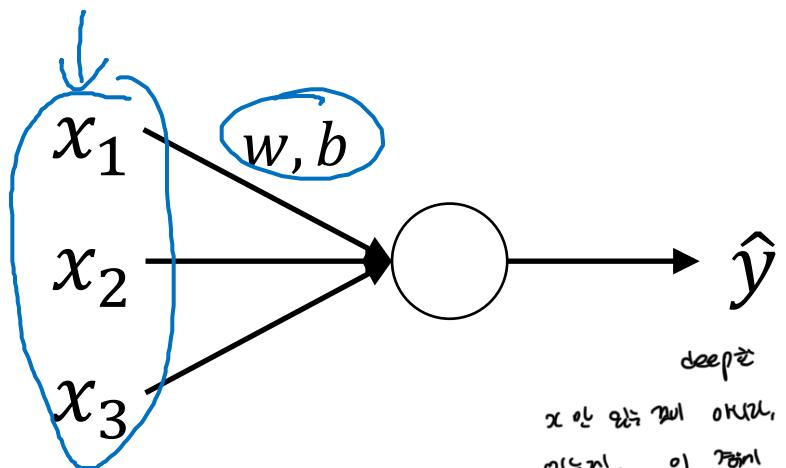
2) 편향과 편차를 제거

Batch Normalization 시 초기값 설정(Initial Value)과 편향(bias)을
활성화 함수(Activation Function) 초기값 및 편차
부분으로 분리해 두는 것이다.

Batch Normalization

Normalizing activations
in a network

Normalizing inputs to speed up learning



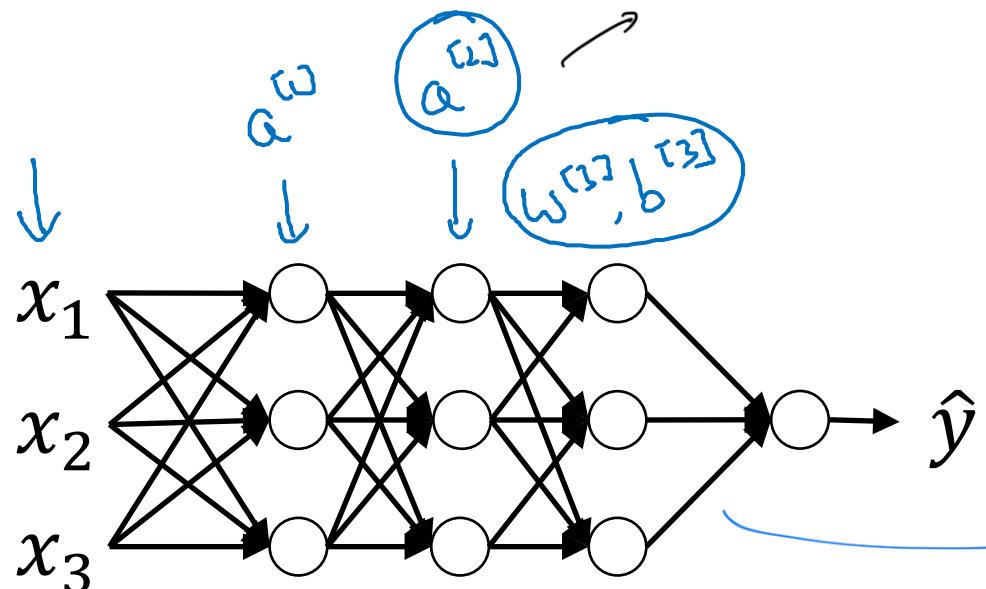
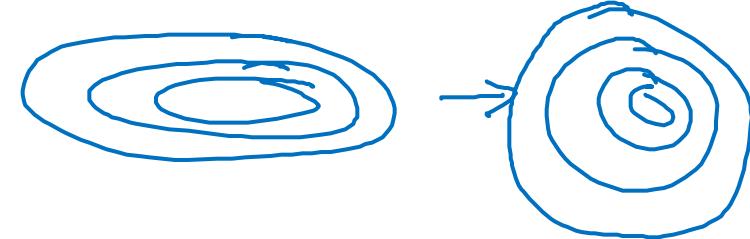
deep model forward pass
x는 각각 다른 차원, $a^{(1)}, a^{(2)}, \dots$, activation
단위, 이 단위 batch Normalization
단위.

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

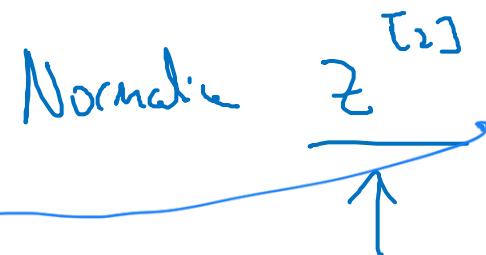
$$X = X - \mu$$

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

~~$X = X / \sigma$~~



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



예 NN의 출력 층의 이전 층의 $a^{[2]}$ 를
이제 Normalization 해 주면 좋겠다.
그리고 이전의 $w^{[2]}, b^{[2]}$ 를 더 쉽게 학습할 수 있다.
 $a^{[2]} = w^{[2]}, b^{[2]}$ 대신에 $z^{[2]}$ 를 사용하는 것이다.
이제 Batch Normalization을 $z^{[2]}$ 로 Batch Norm
하겠다.

Implementing Batch Norm

Given some intermediate values in NN

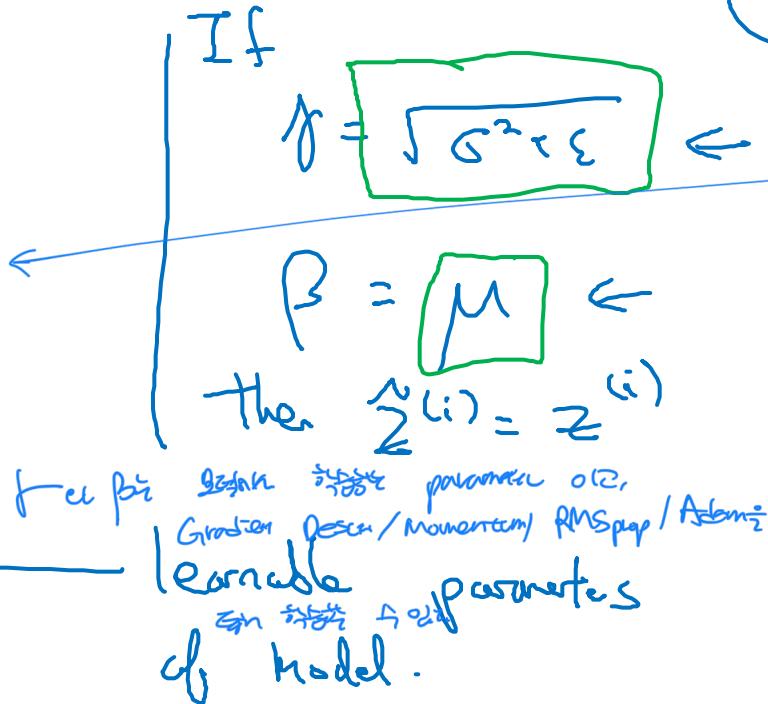
$$\mu = \frac{1}{m} \sum_i z^{(i)}$$

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

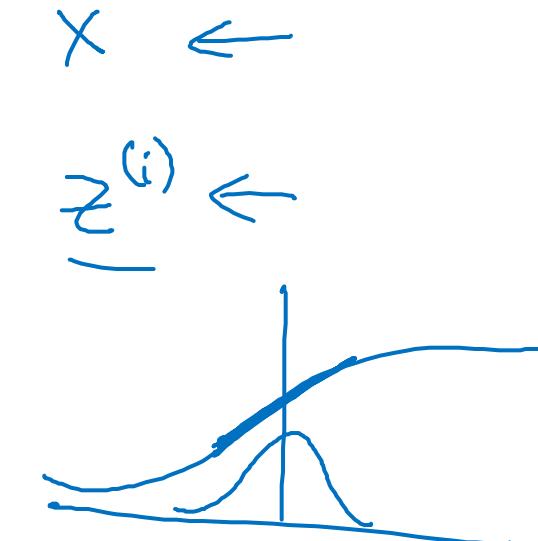
$$z_{\text{norm}}^{(i)} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

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

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



For β 를 사용해 정규화된 $\hat{z}^{[l]}$ 는 hidden layer는 흡수 $\beta=0$, $\gamma=1$ 이면 $\hat{z}^{[l]}=z^{[l]}$ 가 됨. β 는 $\hat{z}^{[l]}$ 의 표준偏差를 줄여 줄 때 사용.



Plot activation function sigmoid $\sigma(z) = \frac{1}{1+e^{-z}}$.
 Instead of 0 , we have β normalization.
 Same as α β linear transformation
 Different approach to regularize model.

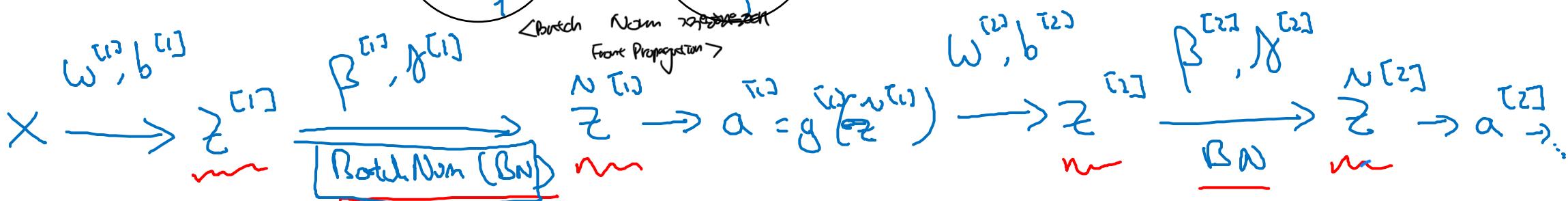
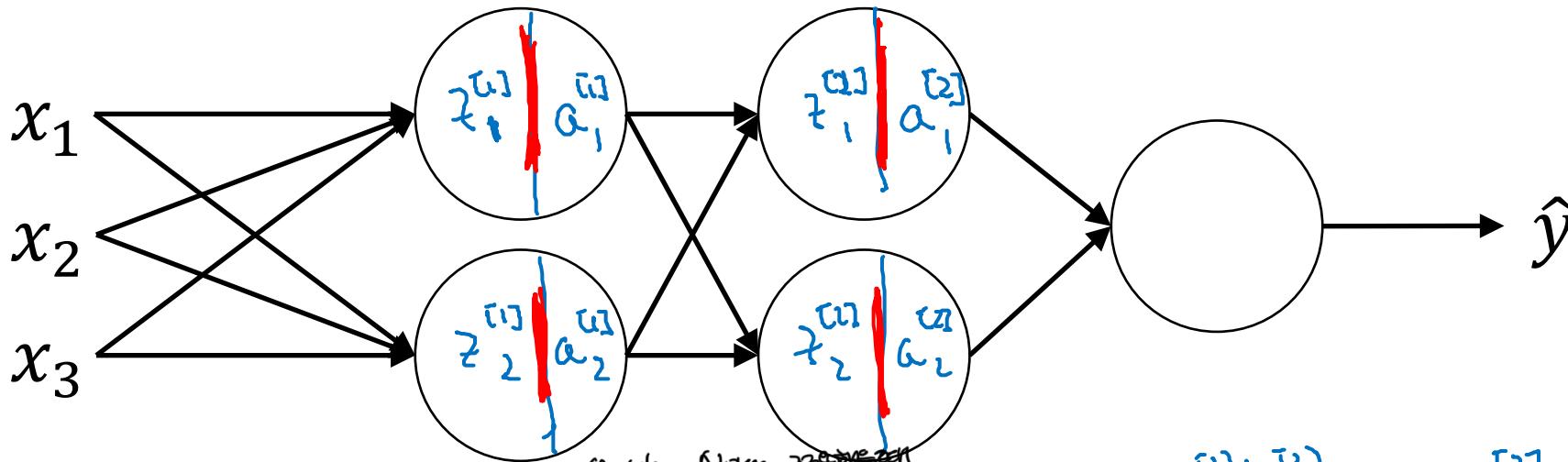


deeplearning.ai

Batch Normalization

Fitting Batch Norm into a neural network

Adding Batch Norm to a network



Parameters: $w^{(1)}, b^{(1)}, w^{(2)}, b^{(2)}, \dots, w^{(L)}, b^{(L)}$,
 $\beta^{(1)}, \gamma^{(1)}, \beta^{(2)}, \gamma^{(2)}, \dots, \beta^{(L)}, \gamma^{(L)}$

→ β → β
 → β

설명: β 는 Adam의 momentum인 beta의 학습률.
 그리고 β, γ 는 w, b 의 학습률이 아니라, 그들의 학습률이다.
 Adam은 β 와 γ 를 학습하는 대신 그들의 학습률을 학습한다.

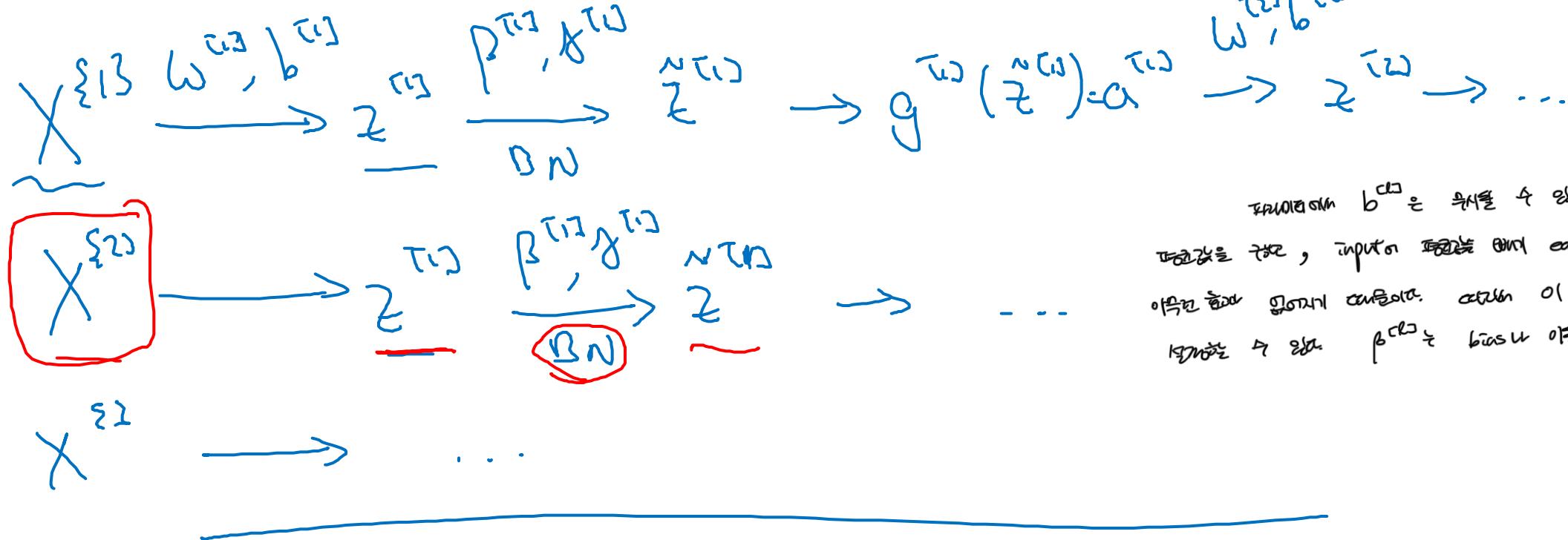
$$\begin{aligned} \beta &= \beta - \alpha d\beta^{(l)} \\ \frac{d\beta}{dt} &= \beta^{(l)} - \alpha d\beta^{(l)} \end{aligned}$$

tf.nn.batch_normalization ←
 tf Deep learning Framework 노드 텐서
 텐서를 평균화.

Working with mini-batches

<mini-batch with batch norm 2018>

각 mini-batch의 global 평균
normalization은 각각 다른 것.



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

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

$$\begin{array}{c} | \\ (n^{[l]}, 1) \end{array} \quad \begin{array}{c} | \\ (n^{[l]}, 1) \end{array} \quad \begin{array}{c} | \\ (n^{[l]}, 1) \end{array}$$

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

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

$$\underline{z}^{[l]}$$

$$\rightarrow \underline{z}^{[l]} = \gamma^{[l]} \underline{z}^{[l]} + \beta^{[l]}$$

Andrew Ng

Implementing gradient descent

for $t = 1 \dots \text{num MiniBatches}$

Compute forward prop on $X^{[t]}$.

In each hidden layer, use BN (batch norm) to replace $\underline{z}^{[l]}$ with $\hat{z}^{[l]}$.

Use backprop to compute $\underline{dw}^{[l]}$, ~~$\underline{db}^{[l]}$~~ , $\underline{d\beta}^{[l]}$, $\underline{dg}^{[l]}$

Update parameters

$$\left. \begin{aligned} w^{[l]} &:= w^{[l]} - \alpha \underline{dw}^{[l]} \\ \beta^{[l]} &:= \beta^{[l]} - \alpha \underline{d\beta}^{[l]} \\ g^{[l]} &:= f^{[l]} - \alpha \underline{dg}^{[l]} \end{aligned} \right\} \quad (* \text{parameters by weight})$$

Works w/ momentum, RMSprop, Adam.

等於前向計算的梯度。

< batch Normal 은 잘 동작하지 >

작성한 글: input feature 를 normalization 한 값은 0, 평균 1 을
반드시 지킬 것, 이를 통해 학습 속도를 증가시킨다. **DE** **이** **는** **normalization** **한**
때문에, input feature x 가 평균 0 이거나 1 이 아니면 학습
속도가 감소하는 **Norm**. 그리고, 이 normalization 은 input layer 뿐만 아니라
hidden layer 와 output layer 를 포함한다.



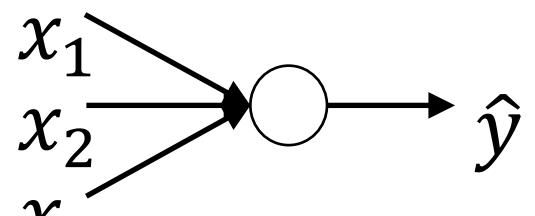
deeplearning.ai

설명: batch normalization 은 **batch**
weight 의 **부호** 를 **obtain** 한다.

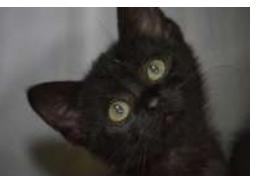
Batch Normalization

Why does
Batch Norm work?

Learning on shifting input distribution



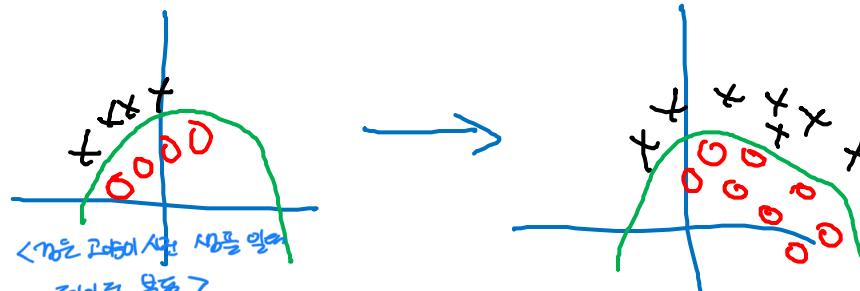
Cat



Non-Cat



$$y = 1$$



$$y = 1$$



$$y = 0$$



"Covariate shift"

$$\underline{x} \rightarrow y$$

이유는 디플레인 학습한 모델의 경계는 원래는 시가지와 고속화
로운 일반화 가능한 분류 결정면을 가질까
있지만, 데이터 분포가 변하는 경우 Covariate shift를
통해, 만약, 영역 x 의 원로로 예측되는 확률은 높아지면
다시 학습 시켜야 할 수도 있다.

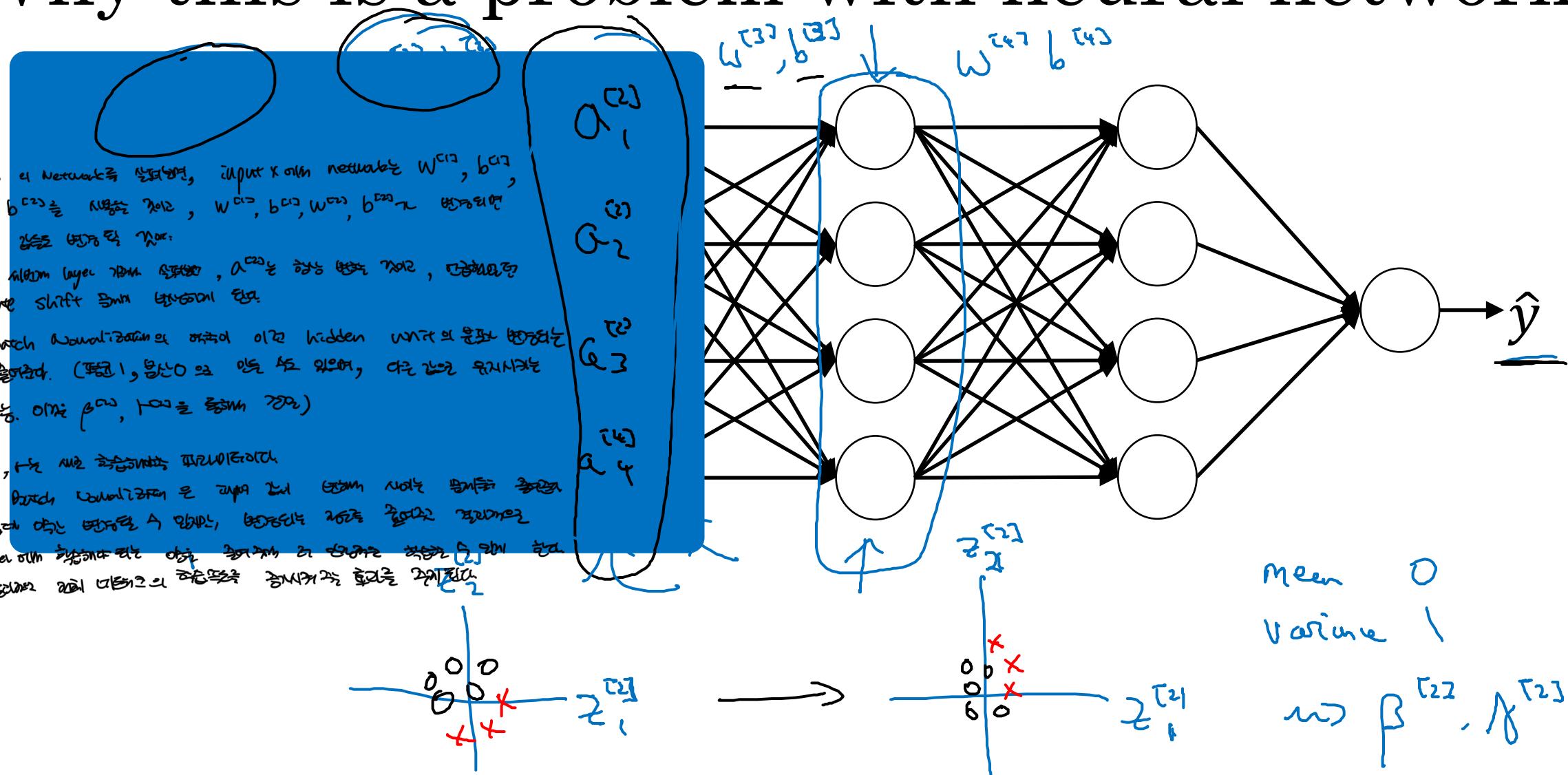
Why this is a problem with neural networks?

< Colatitude shift 등장 Neural Network etc
• 아름다움 판별기 >

Layer 3의 Network를 살펴보면, input x와는 netwoks는 W^{C_1}, b^{C_1} , W^{C_2}, b^{C_2} 를 사용하는 것으로, $W^{C_1}, b^{C_1}, W^{C_2}, b^{C_2}$ 가 연결되어 있다.

2nd wave silicon layer 7nm 厚さで, λ^{CVD} を 80nm とした際, 逆転する
shift 量は 60nm 程度である.

마지막 batch Normalization의 출력이 바로 hidden layer의 활성화를 하는
계수를 출력한다. (표현, 분산으로 만들 수도 있으며, 다른 값으로 유지시키는
것도 가능. 이를 β , γ 로 표기해 보자)



Mean 0
Variance 1

$\rightsquigarrow \beta^{[22]}, \alpha^{[23]}$

Batch Norm as regularization

X

- Each mini-batch is scaled by the mean/variance computed on just that mini-batch.
 $\xrightarrow{\hat{z}^{[l]}}$ μ, σ^2 $\{z^{[l]}\}$
- 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.

각 mini-batch는 각각의 mini-batch에 대한 mean/variance를 계산해 이를 각각의 hidden layer activation에 noisy하게 만들 때, 이를 통해 regularization의 역할을 한다. 즉 noisy한 것은 좋지 않다. (Dropout은 그게 좋은 것 같다.)
그러나 반복되는 batch size가 작을 때 (64 → 256), regularization 역할을 하는 batch size가 작아 regularization 효과가 떨어진다.
batch Norm은 regularization 역할을 하는 것으로 보인다.



deeplearning.ai

Multi-class classification

Softmax regression

Softmax Regression은 Logistic Regression의 확장된 버전으로,
이진분류가 아닌 다중 분류를 다룰 때까지 예측이 가능하는 두 단계로 하는 모델이다.
예전에는 그나마 예상 가능하고 그리고 예상값을 인식하고 싶은 사람이
가장 많았던 것이다.

Recognizing cats, dogs, and baby chicks



3

1

2

0

3

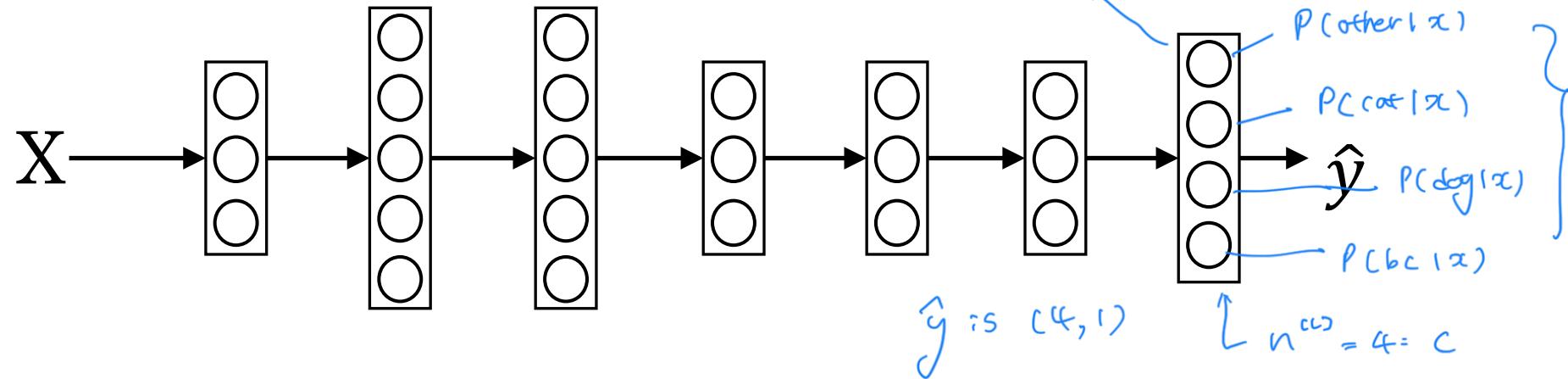
2

0

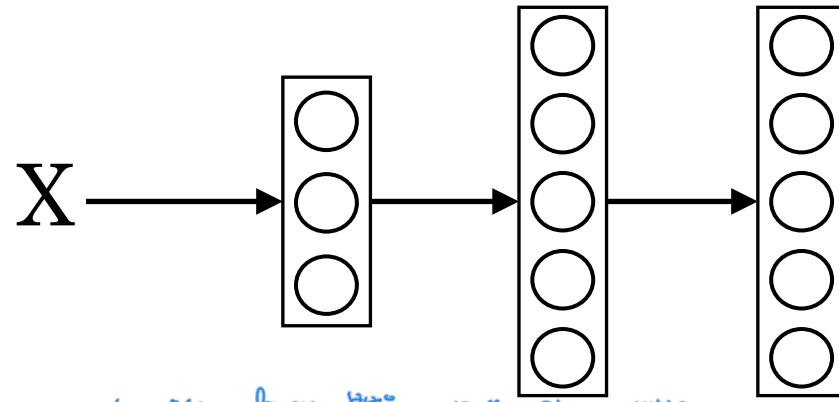
1

$$C = \# \text{classes} = 4 \quad (0, \dots, 3)$$

이 output layer는 4개의 class를 학습하는 경우, 첫 번째 노드는 other class의 학습을
담당하고, 두 번째 노드는 고양이의 학습, 세 번째 노드는 강아지의 학습, 네 번째 노드는
어느 다른 것을 예측하는 경우. Output layer의 결과는 4x1의 dimension을 가짐. 그리고 학습할 때 (이전 단계의 예상,
정답)를 통해 loss function을 계산합니다.



Softmax layer

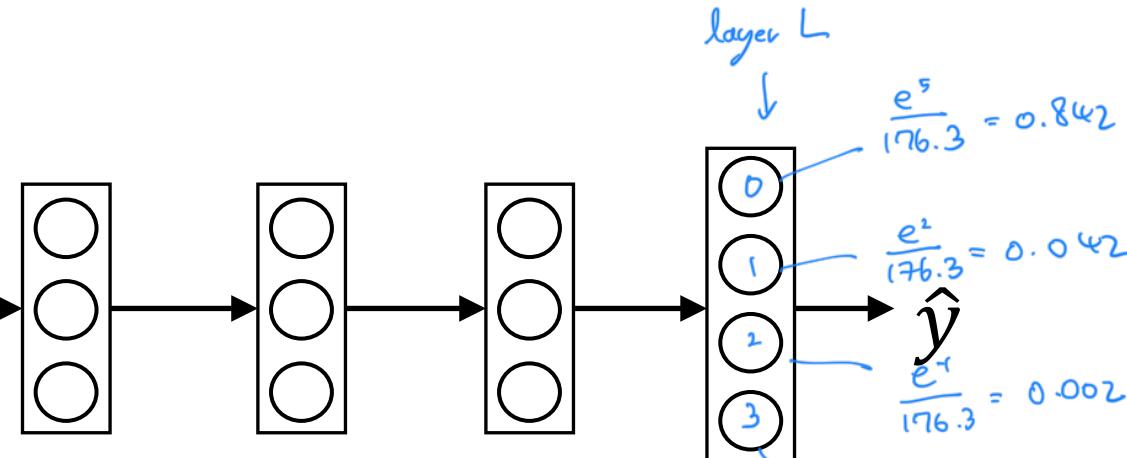
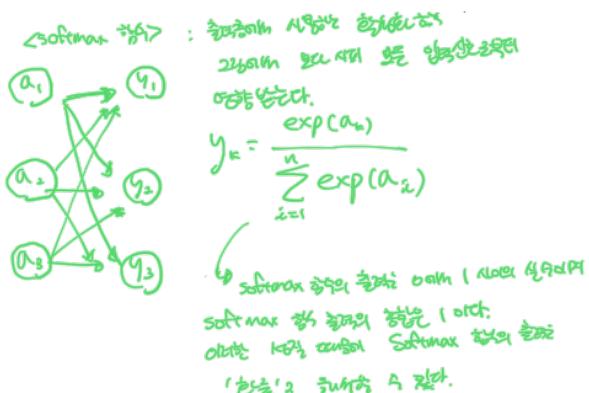


이전 layer에서 힘 있는 feature를 다른 끝에 전달해.

$$z^{[L]} = W^{[L]} a^{[L-1]} + b^{[L]}$$

이전 힘 있는 feature를 softmax activation 함수를 적용해.
이 activation 힘 있는 feature를 softmax activation에 적용.
다른 힘 있는 feature는 softmax로 적용.
Activation function (for softmax):

$$t = e^{(z^{[L]})}, \quad a^{[L]} = \frac{e^{z^{[L]}}}{\sum_{i=1}^n t_i}, \quad a_i^{[L]} = \frac{t_i}{\sum_{i=1}^n t_i}$$



$a^{[L]}$ 의 Dimension은 정확히 $W \times 1$ Vector이다.
그리고 이 Vector가 다른 끝에 전달될 때, 그때에 확률화
여기서

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

이전 Vector t 를 확률화 시키기 때문에 다음과 같다.

$$t = \begin{bmatrix} e^5 \\ e^2 \\ e^{-1} \\ e^3 \end{bmatrix} = \begin{bmatrix} 148.4 \\ 7.4 \\ 0.4 \\ 20.1 \end{bmatrix}$$

$$\sum_{i=1}^4 t_i = 176.3$$

그리고 $a^{[L]}$ 를 확률화 시킬 때 확률화 하는 것이다.

$$a^{[L]} = \frac{t}{176.3} = \begin{bmatrix} \frac{e^5}{176.3} = 0.842 \\ \frac{e^2}{176.3} = 0.042 \\ \frac{e^{-1}}{176.3} = 0.002 \\ \frac{e^3}{176.3} = 0.114 \end{bmatrix}$$

이제 확률화 하는 것은 (이제는 각 행렬을 수 있다).
이전 이전에 확률화 activation이거나 single row의 것
이면 확률화, softmax regression의 경우는
그럼, 확률화 하는 것이다.

Softmax examples

각각의 hidden layer는 softmax regression

$x_1 \rightarrow \begin{bmatrix} \cdot \\ \cdot \\ \cdot \end{bmatrix} \rightarrow \hat{y}$

$x_2 \rightarrow \begin{bmatrix} \cdot \\ \cdot \\ \cdot \end{bmatrix} \rightarrow \hat{y}$

$x \rightarrow \begin{bmatrix} \cdot \\ \cdot \\ \cdot \end{bmatrix} \rightarrow \dots \rightarrow \begin{bmatrix} \cdot \\ \cdot \\ \cdot \end{bmatrix} \rightarrow \hat{y}$

$$z^{(c=3)} = w^{(c=3)}x + b^{(c=3)}$$

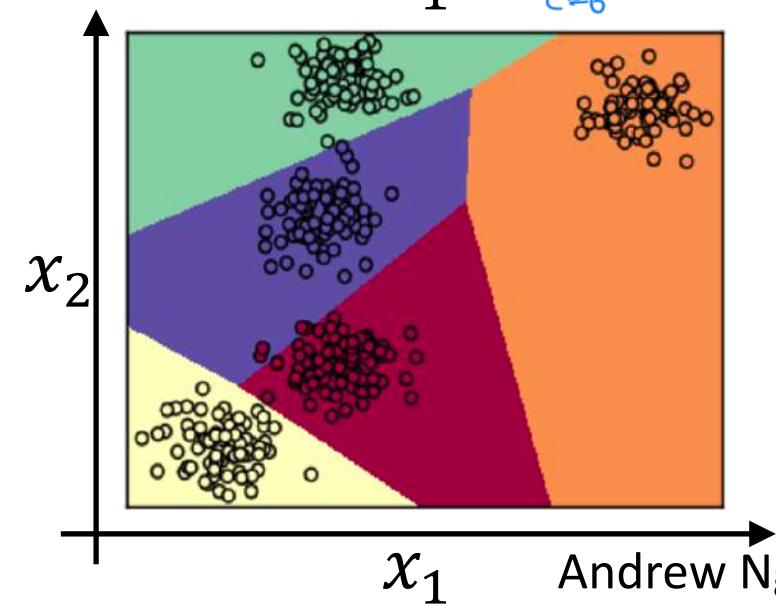
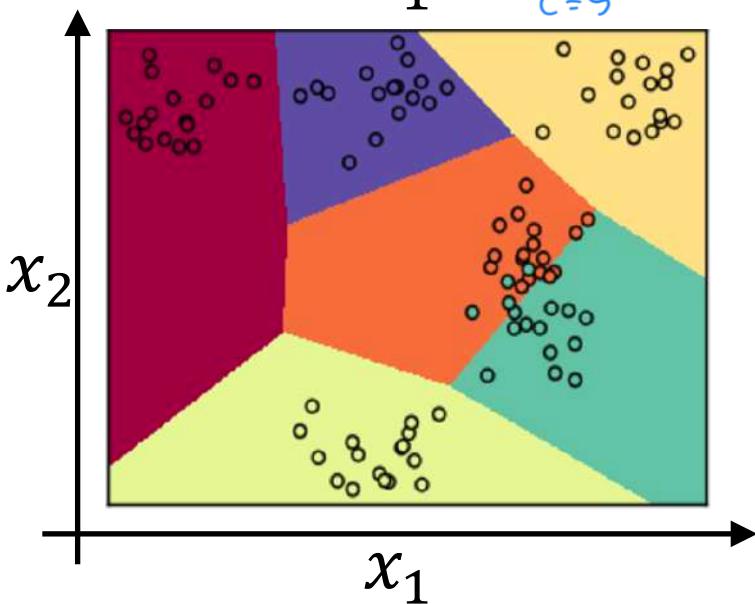
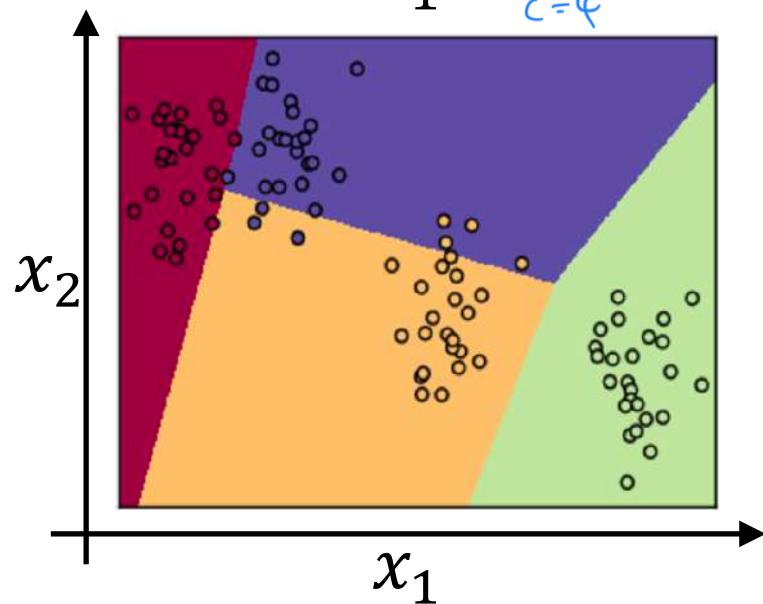
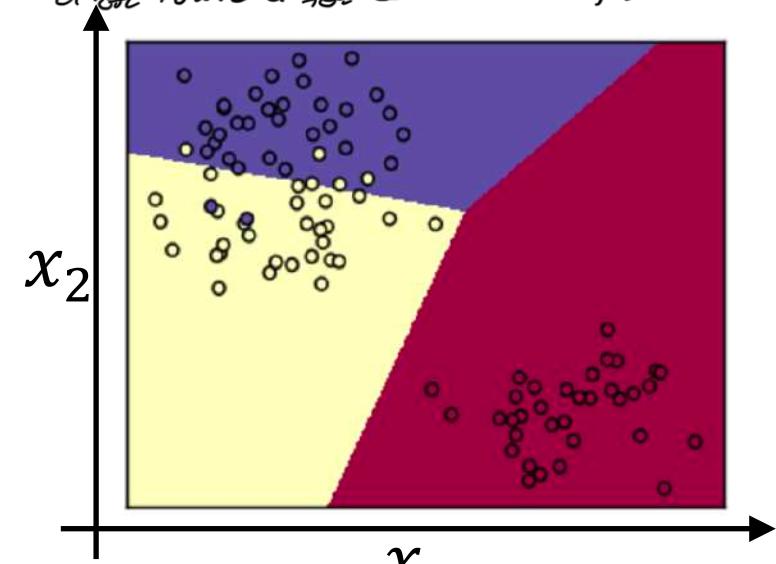
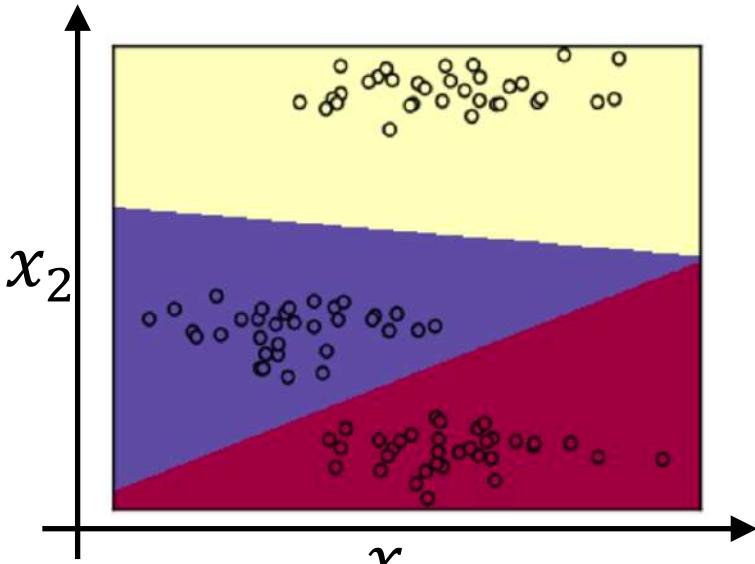
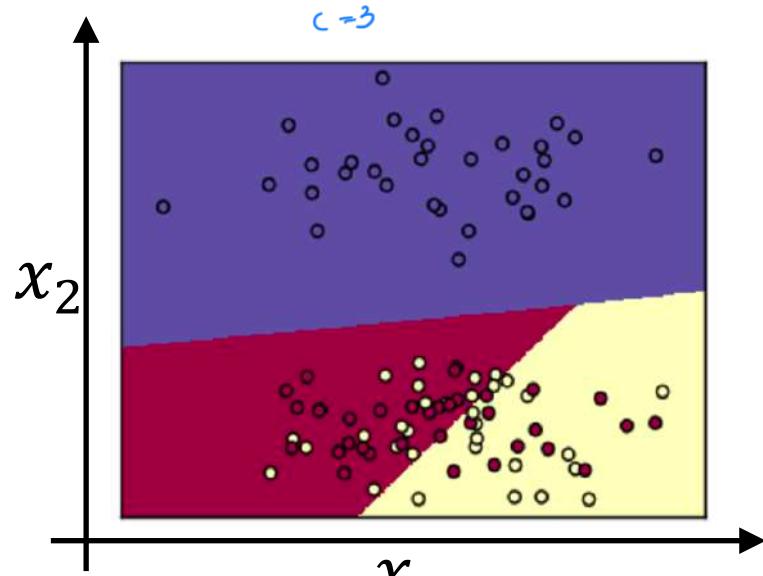
$$\alpha^{(c=3)} = \hat{y} = g(z^{(c=3)})$$

$c=3$ 일 때 단위 \hat{y}

Decision Boundary $\hat{y} = 0.5$ 일 때

own decision boundary는 logistic regression을 확장한
즉,深层次의 Decision Boundary가 되어버려 유효하지 않다.
단위 \hat{y} 에 대한 decision boundary가 되어야 한다.

deep neural network의 hidden layer, hidden unit은
G가 되어야 하는 G에 대한 decision boundary는 되어야 한다.



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

$$J(\omega) = \frac{(\omega^2 - 10\omega + 25)}{(\omega - 5)^2}$$

$\omega = 5$

$J(\omega, b)$

↑ ↑