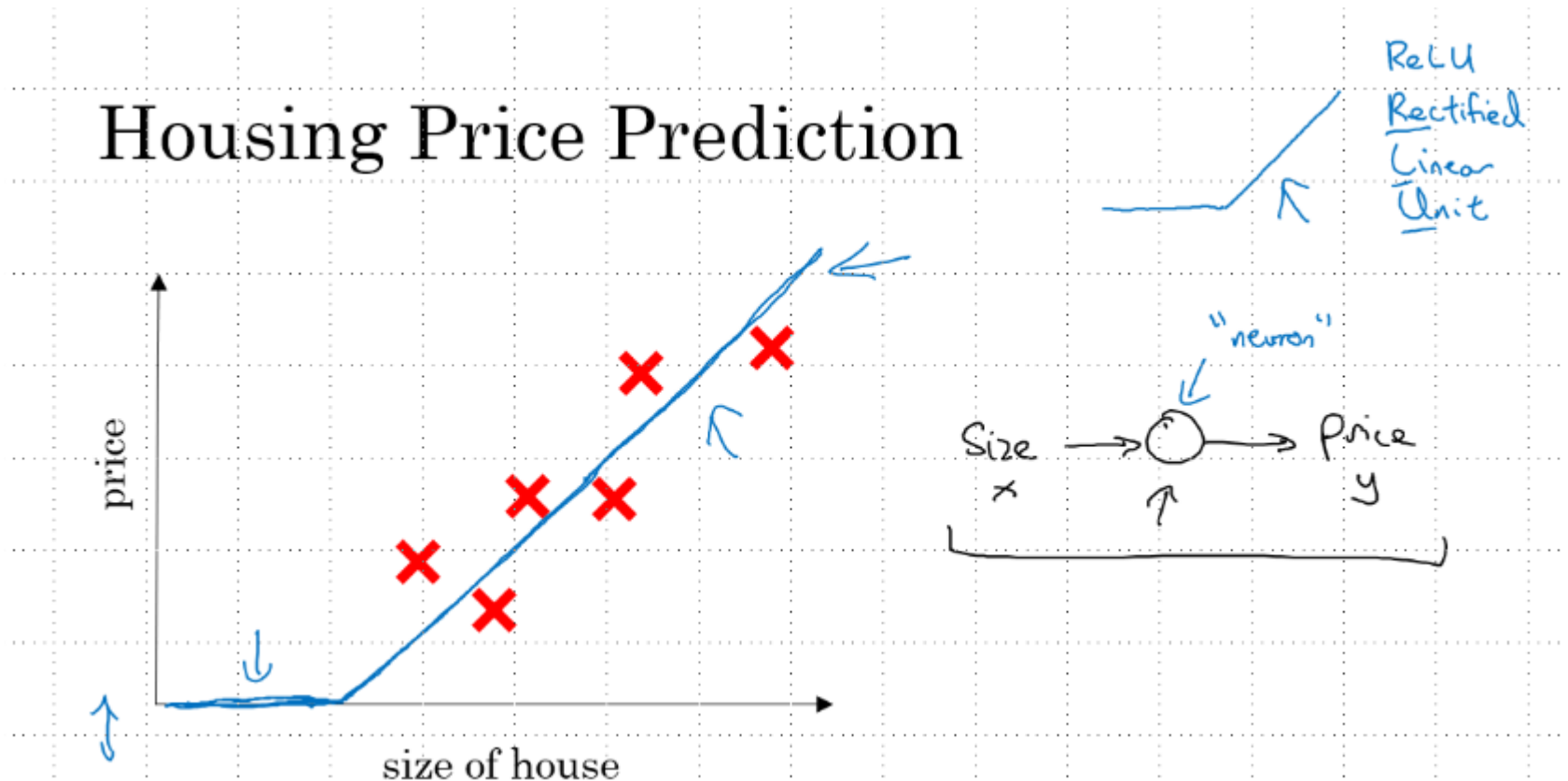


Neural Networks and Deep Learning

Andrew Ng

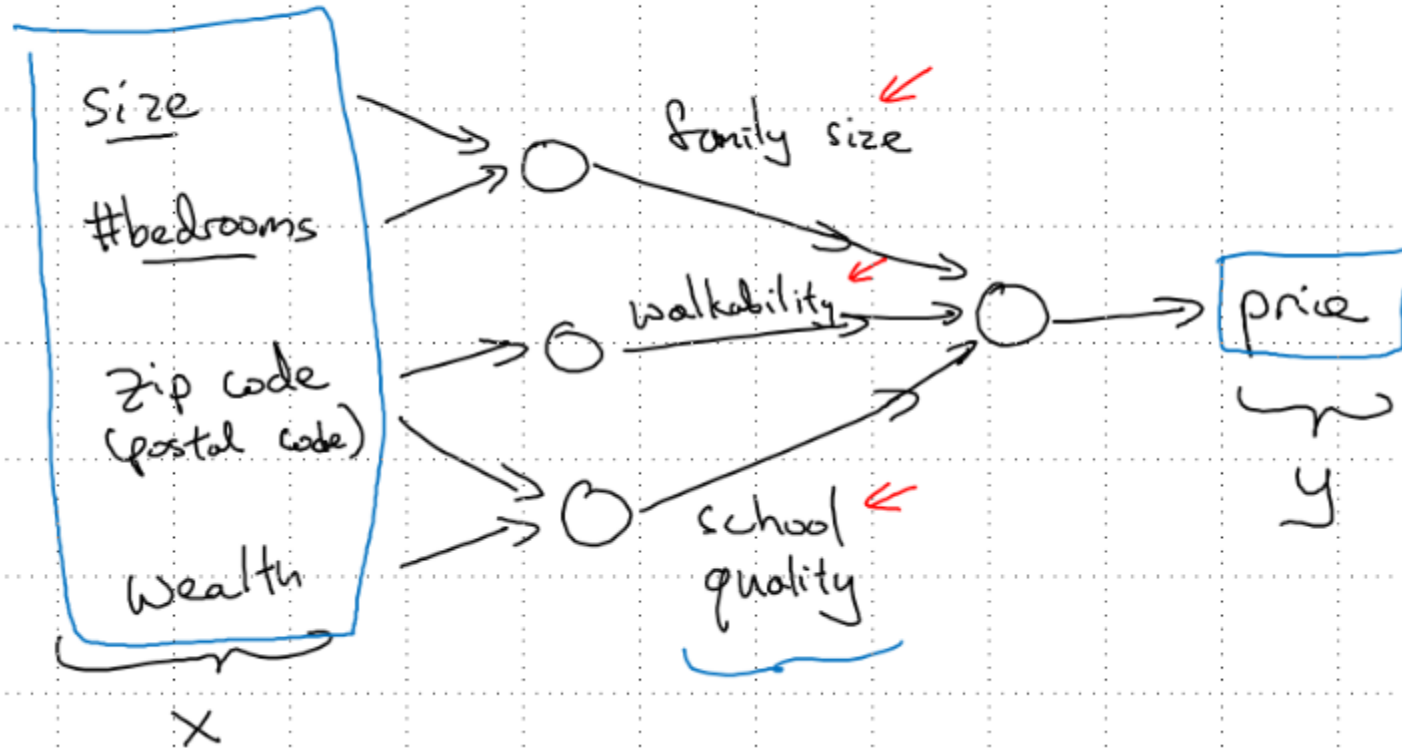
1. What is neural network?



이것도 하나의 뉴런(LeLU)으로 이루어진 신경망

1. What is neural network?

Housing Price Prediction



Input 각각의 특성에 맞게 (크기와 침실의 개수는 가족 규모에, 우편번호와 재산은 학교의 수준에, 우편번호는 도보생활반경에 영향) $y(\text{price})$ 에 영향을 줌

2. Supervised learning for Neural Network

Supervised Learning		
Input(x) ←	Output (y) ←	Application
Home features	Price	Real Estate
Ad, user info ←	Click on ad? (0/1)	Online Advertising
Image	Object (1,...,1000)	Photo tagging
<u>Audio</u>	Text transcript	Speech recognition
<u>English</u>	Chinese	Machine translation
<u>Image, Radar info</u> ↑	Position of other cars ↑	Autonomous driving

Standard NN이 많이 품 (정형데이터)

Standard NN

CNN

RNN

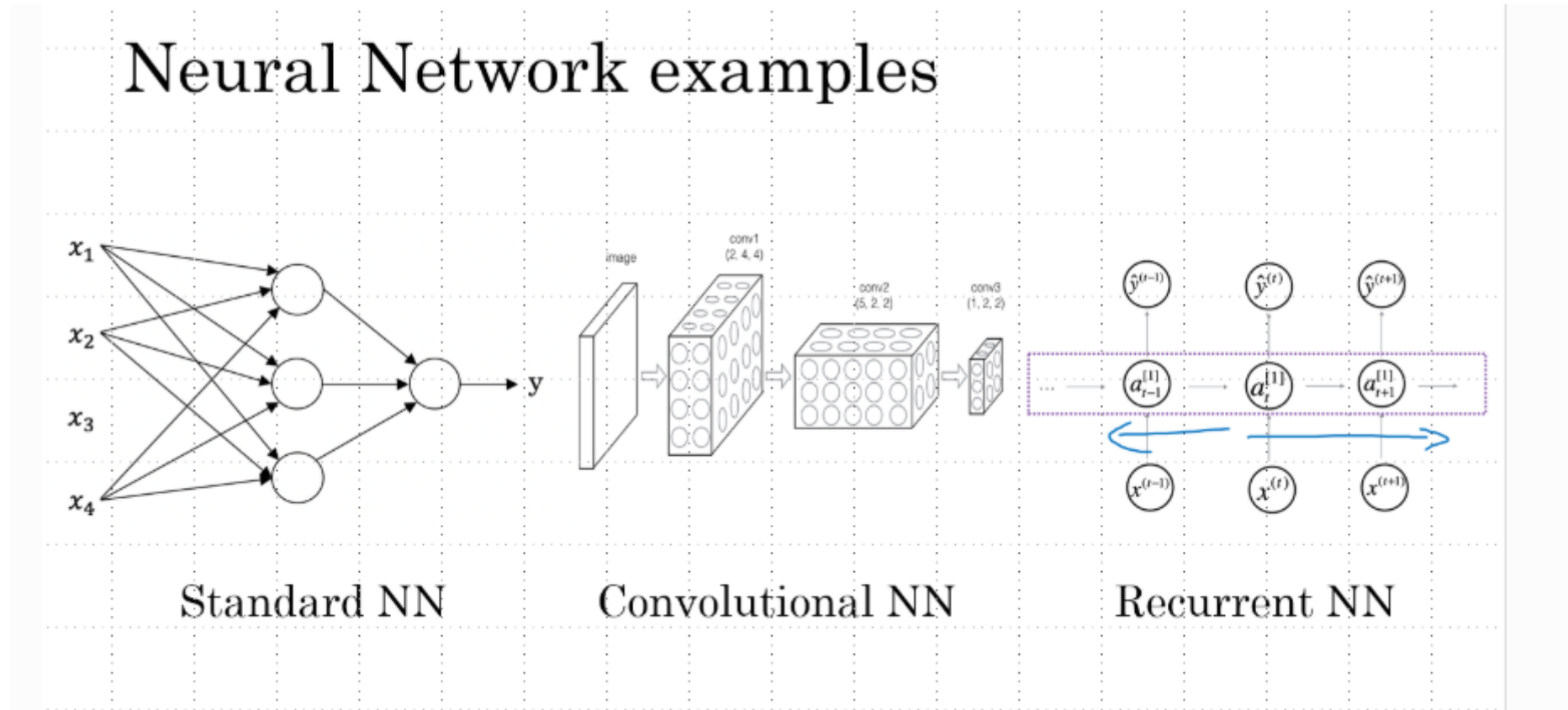
Custom Hybrid

대부분 CNN 적용 (이미지)

RNN (음성, 번역)

custom./hybride (자율주행)

2. Supervised learning for Neural Network



대충 차이는 이렇다고 함.

CNN은 지역 특징 잡는데 유리하고 RNN은 시퀀스 데이터에 유리하겠지?

Structured vs Unstructured data

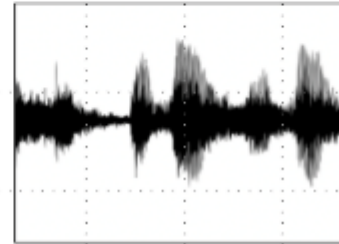
Supervised Learning

Structured Data

Size	#bedrooms	...	Price (1000\$)
2104	3		400
1600	3		330
2400	3		369
⋮	⋮		⋮
3000	4		540

User Age	Ad Id	...	Click
41	93242		1
80	93287		0
18	87312		1
⋮	⋮		⋮

Unstructured Data



Audio



Image

Four scores and seven
years ago...

Text

엑셀로 예쁘게 정리된 데이터를 정형 데이터라고 함

엑셀로 예쁘게 정리하려면 노가다가 필요한 데이터를
비정형 데이터라고 함

딥러닝 덕분에 비정형데이터 분석이 급격하게 발전됐는데 정형데이터 부분도 무시할 수 없으니 같이 공부하기로 함!

3. Why is deep learning taking off?

Deep Learning이 최근에 급부상한 이유?

Scale drives deep learning progress

데이터가 적으면 파라미터 조절로도

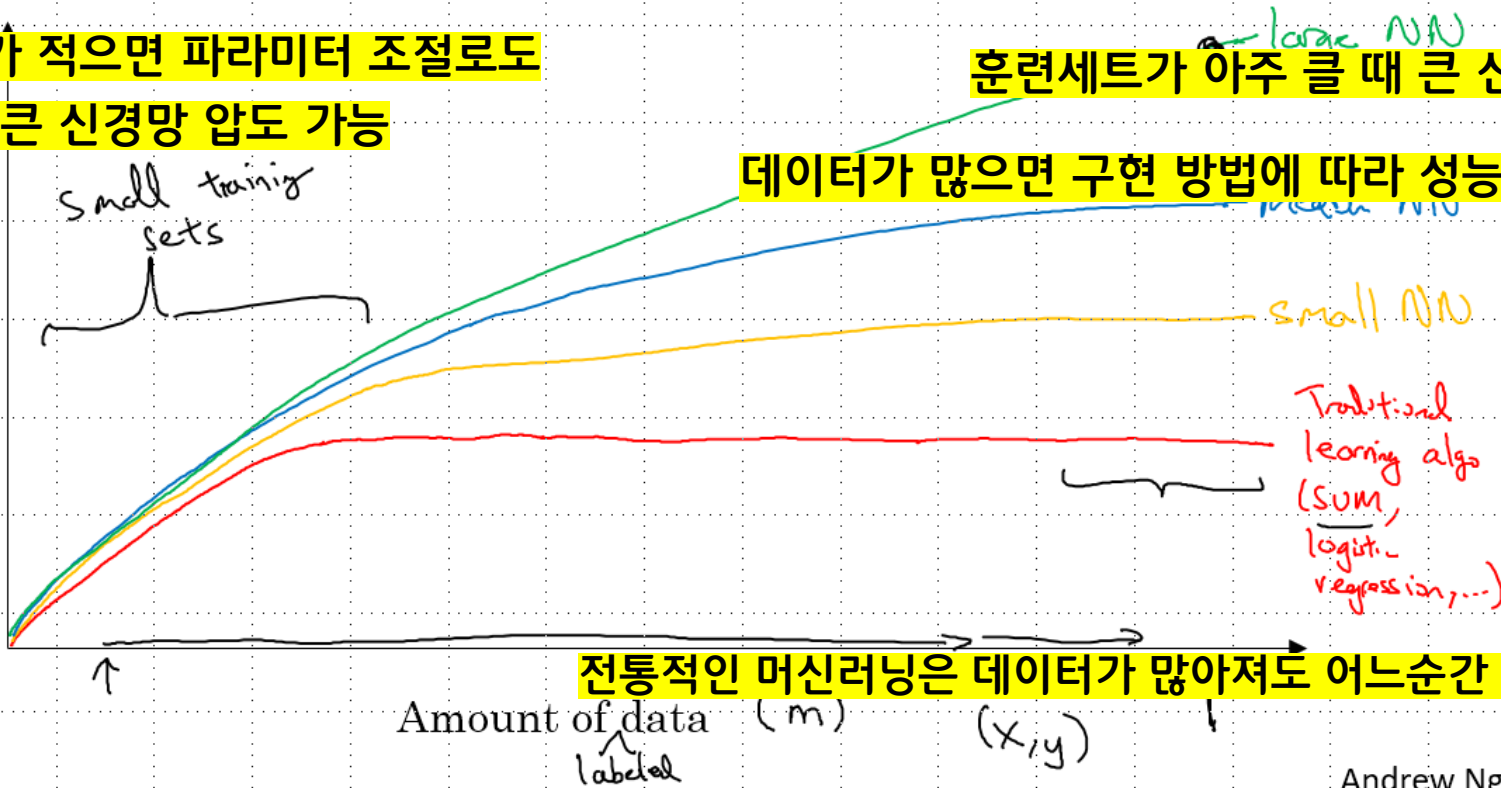
큰 신경망 압도 가능

Small training sets

훈련세트가 아주 클 때 큰 신경망이 압도

데이터가 많으면 구현 방법에 따라 성능이 결정되는 경우가 많음

Performance



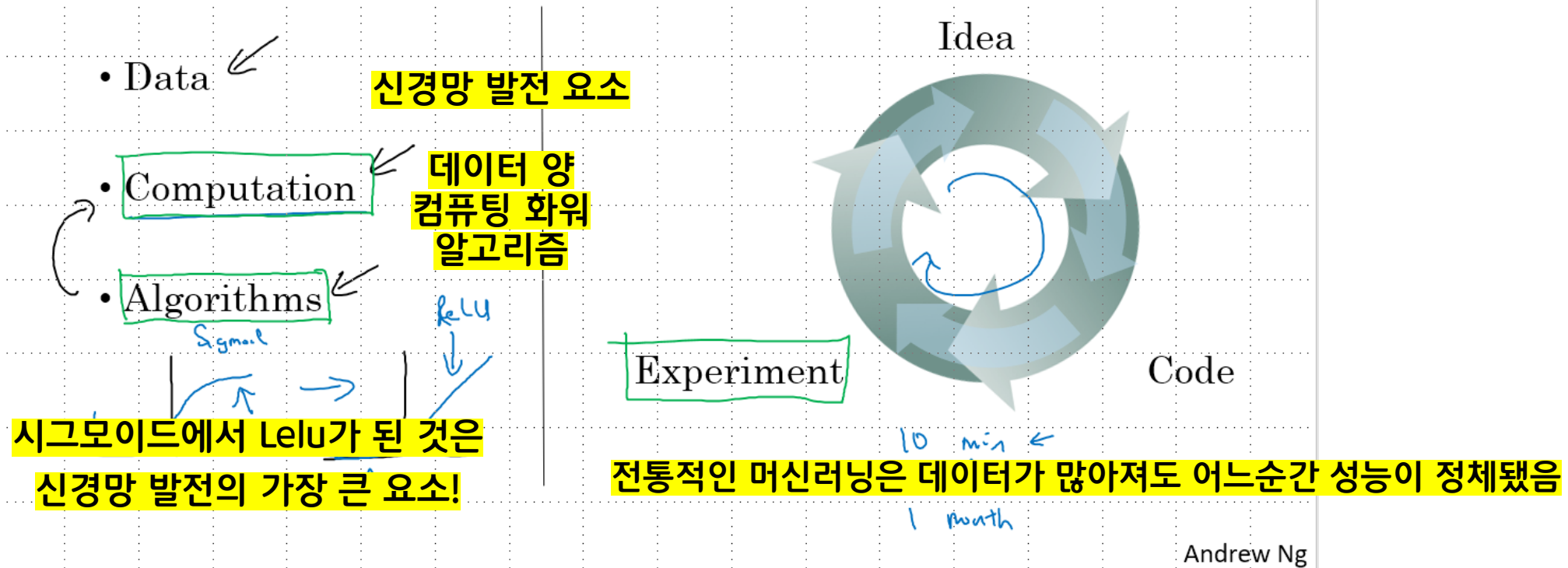
전통적인 머신러닝은 데이터가 많아져도 어느순간 성능이 정체됐음

중요한 것! NN은 데이터 양 많아지면 성능도 올라감!

3. Why is deep learning taking off?

Deep Learning이 최근에 급부상한 이유?

Scale drives deep learning progress



경사가 급격하게 느려지지 않음!

3. Why is deep learning taking off?

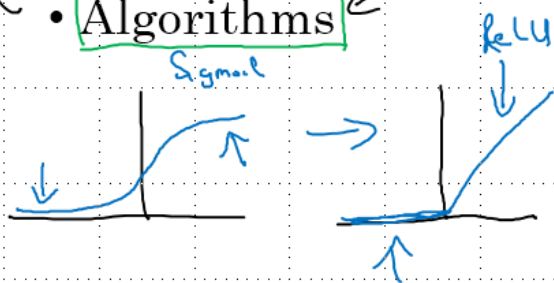
Deep Learning이 최근에 급부상한 이유?

Scale drives deep learning progress

• Data

• Computation

• Algorithms



Experiment

Idea

Code

신경망의 빠른 계산이 중요한 이유는
많은 경우, 신경망을 학습시키는
과정이 매우 반복적이기 때문

10 min
1 day
1 month

Andrew Ng

1. Logistic Regression as a Neural Network

Notation

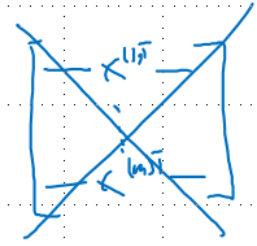
$$(x, y) \quad x \in \mathbb{R}^{n_x}, y \in \{0, 1\}$$

$$m \text{ training examples: } \{(\underline{x}^{(1)}, \underline{y}^{(1)}), (\underline{x}^{(2)}, \underline{y}^{(2)}), \dots, (\underline{x}^{(m)}, \underline{y}^{(m)})\}$$

M개의 데이터

$$M = M_{\text{train}}$$

$$M_{\text{test}} = \# \text{test examples.}$$

$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ | & | & & | \end{bmatrix}$$


$$X \in \mathbb{R}^{n_x \times m}$$

$$X.\text{shape} = (n_x, m)$$

X의 shape,
(피쳐갯수, 데이터 개수)

$$Y = [y^{(1)} \ y^{(2)} \ \dots \ y^{(m)}]$$

$$Y \in \mathbb{R}^{1 \times m}$$

$$Y.\text{shape} = (1, m)$$

1. Logistic Regression as a Neural Network

LR에는 두 개의 Parameter가 있습니다. w 와 b 입니다.

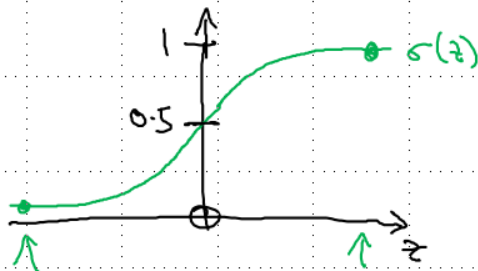
w 와 b 를 update할 것 입니다. w 는 x 와 같은 차원의 열벡터입니다. b 는 상수입니다.

Logistic Regression

Given x , want $\hat{y} = P(y=1|x)$
 $x \in \mathbb{R}^{n_x}$ $0 \leq \hat{y} \leq 1$

Parameters: $\underline{w} \in \mathbb{R}^{n_x}$, $\underline{b} \in \mathbb{R}$.

Output $\hat{y} = \sigma(\underbrace{w^T x + b}_z)$



$$x_0 = 1, \quad x \in \mathbb{R}^{n_x+1}$$
$$\hat{y} = \sigma(\theta^T x)$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{n_x} \end{bmatrix} \quad \left. \begin{matrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{matrix} \right\} b \leftarrow$$
$$\left. \begin{matrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{n_x} \end{matrix} \right\} w \leftarrow$$

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

z 가 커지면 시그모이드 함수는 1에 가까워짐

$$\text{If } z \text{ large } \sigma(z) \approx \frac{1}{1+0} = 1$$

If z large negative number

$$\sigma(z) = \frac{1}{1+e^{-z}} \approx \frac{1}{1+\text{Big num}} \approx 0$$

z 가 음수로 커지면 시그모이드 함수는 0에 가까워짐

목적은 y 가 1일 확률을 잘 예측하도록 파라미터 w 와 b 를 잘 학습

1. Logistic Regression as a Neural Network

Logistic Regression cost function

→ $\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b)$, where $\sigma(z^{(i)}) = \frac{1}{1+e^{-z^{(i)}}}$ $z^{(i)} = w^T x^{(i)} + b$

Given $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$, want $\hat{y}^{(i)} \approx y^{(i)}$.

$x^{(i)}$
 $y^{(i)}$ i -th example.

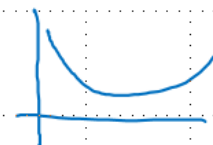
Loss (error) function:

$\mathcal{L}(\hat{y}, y) = \frac{1}{2} (\hat{y} - y)^2$
손실함수

예측값이 실제값과 같아져야함

손실함수는 구불구불하면 안되고

$\mathcal{L}(\hat{y}, y) = - (y \log \hat{y} + (1-y) \log (1-\hat{y}))$ ←



이래야 최적화 됨

If $y=1$: $\mathcal{L}(\hat{y}, y) = -\log \hat{y}$ ← want $\log \hat{y}$ large, want \hat{y} large.

If $y=0$: $\mathcal{L}(\hat{y}, y) = -\log (1-\hat{y})$ ← want $\log (1-\hat{y})$ large ... want \hat{y} small

Cost function: $J(w, b) = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{m} \sum_{i=1}^m [y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log (1-\hat{y}^{(i)})]$

Andrew Ng

손실함수는 훈련 샘플 하나에 관하여 정의해서 그 하나가 얼마나 잘 예측되었는지 측정해 줌

1. Logistic Regression as a Neural Network

LR에서 cost function과 loss function에 대해 알아보겠습니다. x 값을 입력해 \hat{y} 를 구하면 실제 y 값과의 차이 (error)가 생길것 입니다. 이 error를 나타내는 것이 cost function과 loss function입니다. 따라서 이 함수의 값이 작 으면 작을수록 LR의 성능이 좋다고 할 수 있겠죠. 그래서 이 함수들은 후에 LR에서는 cost function을 최소화하도록 두개의 parameter인 w, b 를 update합니다. loss function은 하나의 데이터에 대한 error를 나타내는 것이고, cost function은 전체(m 개)의 training data에 대한 error입니다. 즉 모든 데이터의 loss function의 평균이 cost function입니다. 고전적인 loss function은 $(y - \hat{y})^2$ 인데 LR에선 모양이 많이 다른 것을 알수 있습니다. LR의 cost function같은 형태를 negative log function이라 합니다. 왜 이러한 함수를 쓰냐면 이 함수는 convex모양(민무 닉 토기)이기 때문입니다. 여러 극값(local minimum)을 가지면 최적의 w, b 로 update하기가 어렵기 때문입니다

1. Logistic Regression as a Neural Network

로지스틱 함수를 학습한다는 것

: 손실 함수 J 를 최소화해주는 매개변수 w, b 를 찾는 것

1. Logistic Regression as a Neural Network

Gradient Descent

Gradient Descent
매개변수 w 와 b 학습

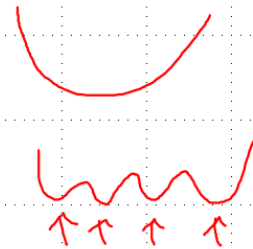
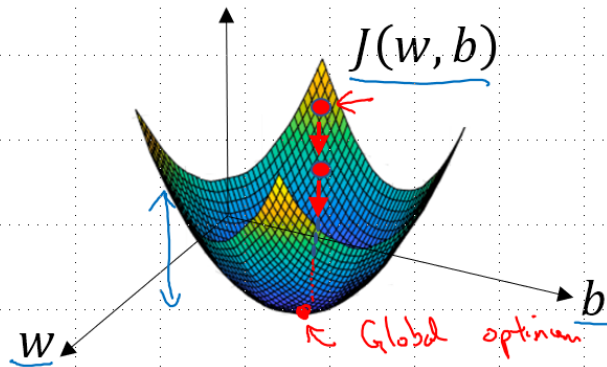
Recap: $\hat{y} = \sigma(w^T x + b)$, $\sigma(z) = \frac{1}{1+e^{-z}}$ ←

$$J(w, b) = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m} \sum_{i=1}^m y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

Want to find w, b that minimize $J(w, b)$

비용함수는 매개변수 w 와 b 가 훈련세트를 잘 예측하는지 측정

$J(w, b)$ 를 가장 작게 만드는 w 와 b 를 찾음



Andrew Ng

손실함수는 알고리즘이 각 훈련샘플의 y 의 예측값이 얼마나 좋은지 각 훈련 샘플에 대한 참값 $y^{(i)}$ 와 비교해 체크

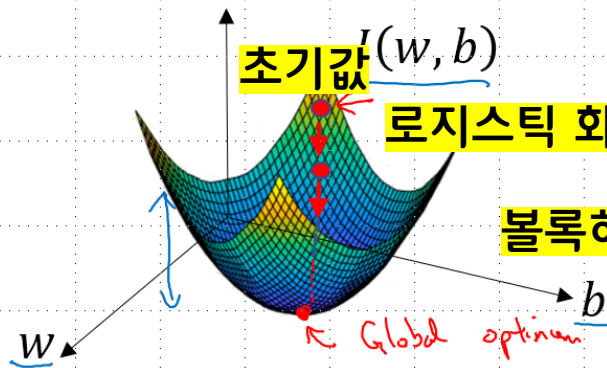
1. Logistic Regression as a Neural Network

Gradient Descent

Recap: $\hat{y} = \sigma(w^T x + b)$, $\sigma(z) = \frac{1}{1+e^{-z}}$ ←

$$J(w, b) = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m} \sum_{i=1}^m y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

Want to find w, b that minimize $J(w, b)$



로지스틱 회귀에 비용 함수 J를 사용한 큰 이유 중 하나 :

볼록, 볼록하지 않은 함수는 지역 최적값이 여러개

초기값은 보통 랜덤
초기점에서 시작해 가장 가파른 내리막으로 한단계 내려감

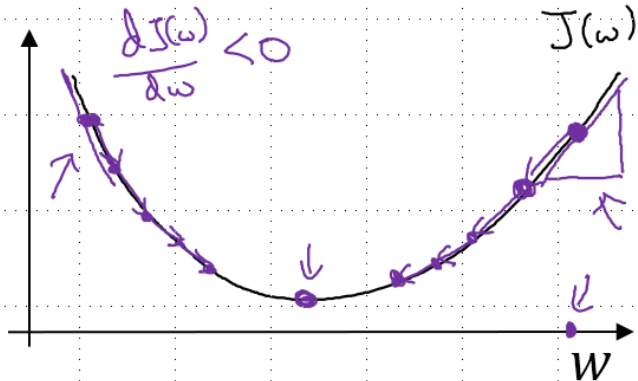
Andrew Ng

손실함수는 알고리즘이 각 훈련샘플의 y의 예측값이 얼마나 좋은지 각 훈련 샘플에 대한 참값 y^(i)와 비교해 체크

1. Logistic Regression as a Neural Network

cost function을 2차원

Gradient Descent



Repeat {
 $w := w - \alpha \frac{dJ(w)}{dw}$
}

learning rate α

$\frac{dJ(w)}{dw}$ "dw"

$w := w - \alpha dw$

$\frac{dJ(w)}{dw} = ?$

$J(w, b)$

$$w := w - \alpha \frac{\partial J(w, b)}{\partial w}$$

$$b := b - \alpha \frac{\partial J(w, b)}{\partial b}$$

$$\frac{\partial J(w, b)}{\partial w}$$

$$\frac{\partial J(w, b)}{\partial b}$$

"partial derivative"

dw

db

특정 parameter에서 cost function의 minimum으로 가는 가장 가파른 경사를 갖는 길을 찾고, 그 길을 따라 (Learning rate - α)만큼 내려가는(w, b 를 update) : 방법은 편미분

특정 w 일 때의 $J(w, b)$ 의 경사(기울기)를 구하고 그 경사를 따라 내려오도록 w 를 update

Differential

$f(a)$ 의 미분값 $2a$

A가 0.001만큼 멀어졌을 때 $f(a)$ 의 값은 $2a$ 즉, 0.002만큼 변한다

이것이 미분!

1. 도함수 : 기울기, 함수에 위치에 따라 다른 값
2. 함수의 도함수를 찾아야 할 때 공식을 찾아야 함

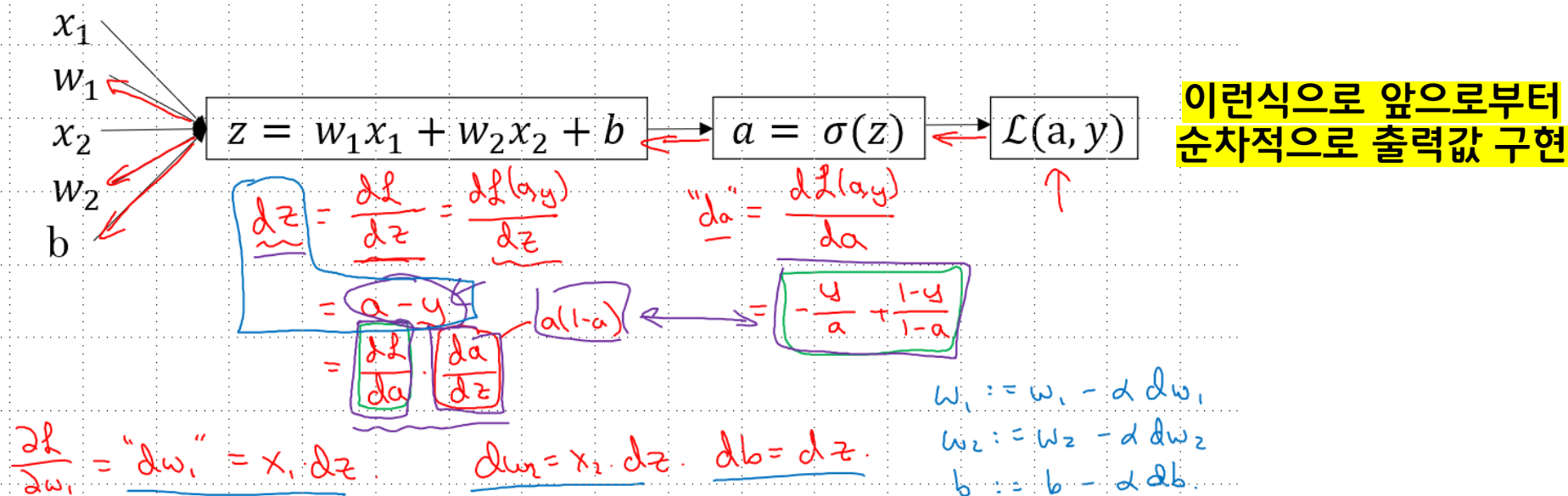
Differential

정방향 path(foward path) : 신경망의 출력값 계산

역방향 path (backward path) : 경사나 도함수 계산

Differential

Logistic regression derivatives



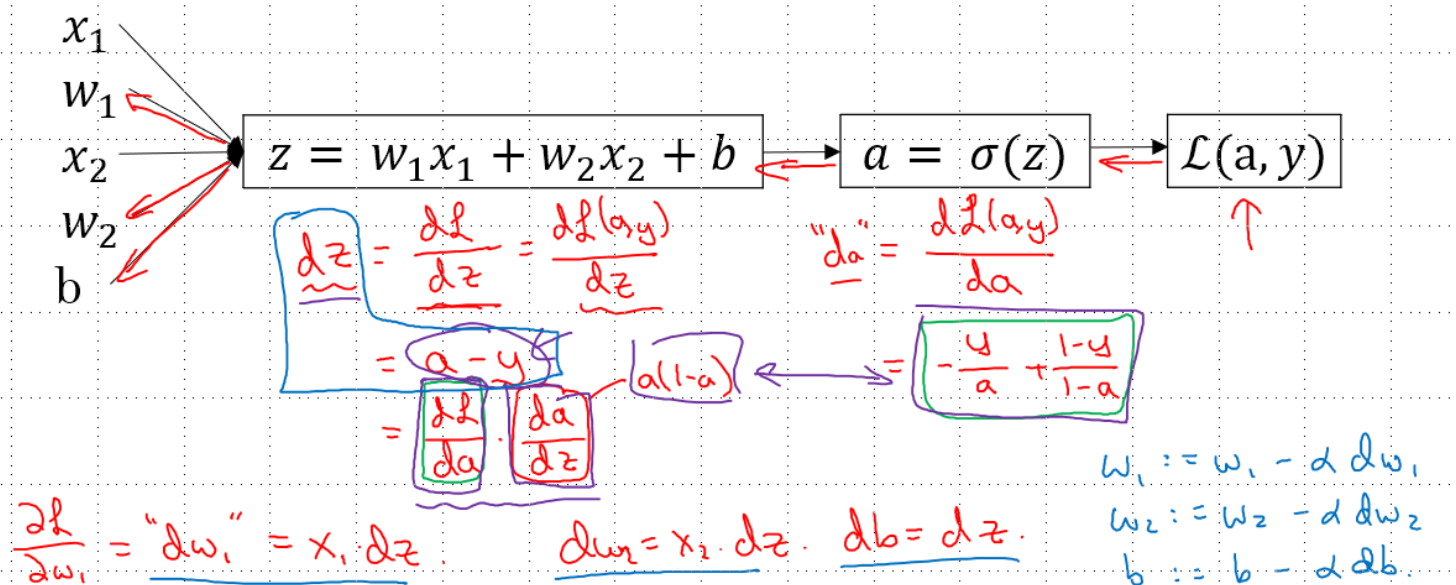
Andrew Ng

이러한 계산은 J같은 특정한 출력값 변수를 최적화 하는데 유용하다
이번엔 순차적으로 구했지만 왼쪽에서 오른쪽의 경로로 도함수를 구할 수 있다

Differential

기호쓰기 너무 불편하니까 간단하게만 설명하겠음
두개의 연쇄 도함수를 곱하면 최종 도함수 구할 수 있음

Logistic regression derivatives



a를 밀었을 때 J의 변화량은
a를 밀었을 때 v의 변화량과
v를 밀었을 때 J의 변화량의 곱

Andrew Ng

a가 바뀌면 v도 바뀌고 v가 바뀌면 J가 바뀐다
a를 조금 높였을 때 J에 일어나는 전체 변화는
 $\frac{dJ}{dv} * \frac{dv}{da} = \frac{dJ}{da}$

Differential

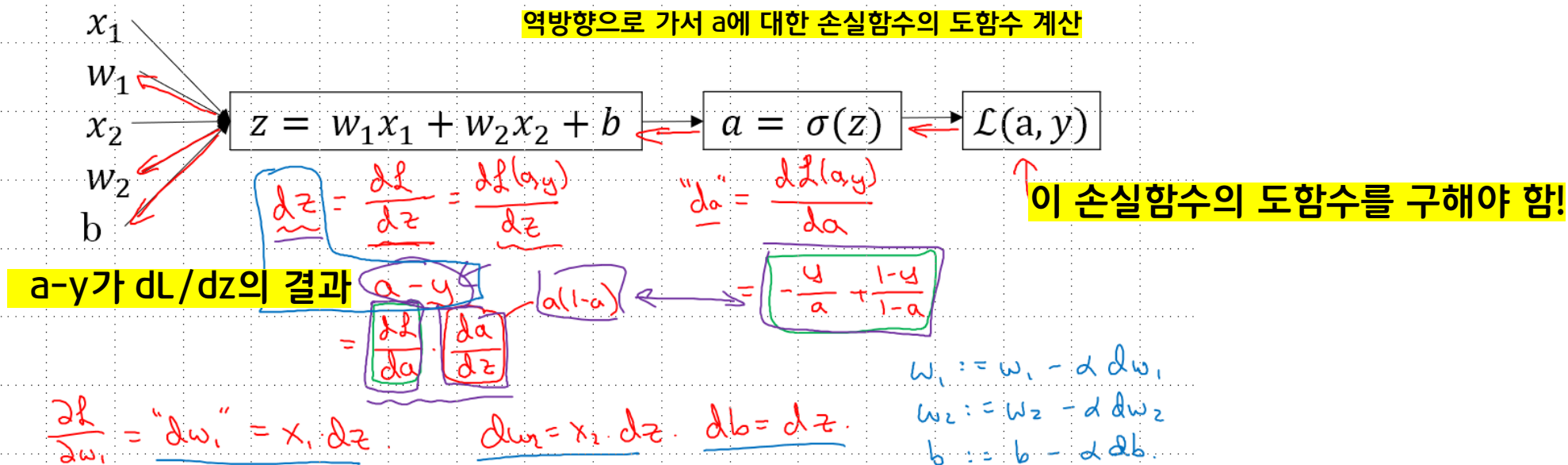
많은 계산을 통해 최종 출력값의 도함수를 계산하게 되는데,
Var라는 변수에 대한 도함수($d\text{최적화 하고싶은 아웃풋}/d\text{var}$) 계산의 많은 부분이
여러 중간 변수를 포함하는 최종 출력값 J의 도함수를 구하는 데 사용

구하고자 하는 최종 변수는 여러 중간값의 해를 계산한 도함수

Differential

기호쓰기 너무 불편하니까 간단하게만 설명하겠음
두개의 연쇄 도함수를 곱하면 최종 도함수 구할 수 있음

Logistic regression derivatives



Andrew Ng

lost function 미분을 computing하는 과정을 살펴보겠습니다. 총 m 개의 데이터가 있으니까 m 번의 loss를 구해서(m 번의 loop) 계속 더해가고 마지막에 m 으로 나누어 평균을 구하면 끝!

1. Logistic Regression as a Neural Network

Logistic regression on m examples

로지스틱 리그레션
Cost function 미분

$$J(w, b) = \frac{1}{m} \sum_{i=1}^m \ell(a^{(i)}, y^{(i)})$$
$$\rightarrow a^{(i)} = \hat{y}^{(i)} = \sigma(z^{(i)}) = \sigma(w^T x^{(i)} + b)$$

$$(x^{(i)}, y^{(i)})$$
$$\underline{dw_1^{(i)}}, \underline{dw_2^{(i)}}, \underline{db^{(i)}}$$

최적값 찾자!

로지스틱 회귀의 목적은
매개변수 w 와 b 를 변경해서

$$\frac{\partial}{\partial w_1} J(w, b) = \frac{1}{m} \sum_{i=1}^m \underbrace{\frac{\partial}{\partial w_1} \ell(a^{(i)}, y^{(i)})}_{\underline{dw_1^{(i)}} - (x^{(i)}, y^{(i)})}$$

$L(a, y)$ 를 줄이는 것!

loss function을 미분하는 법을 알았으니, cost function($J(w, b)$)도 미분할 수 있습니다..! loss를 평균내면 cost이니
까요! 위를 참고하시면 됩니다! 즉 cost를 미분한 것은, 각 데이터별로 loss의 경사(gradient)를 구해서(미분) 평균을
내면됩니다. cost의 경사를 구하는 거죠!!

1. Logistic Regression as a Neural Network

Logistic regression on m examples

$$J(w, b) = \frac{1}{m} \sum_{i=1}^m \ell(a^{(i)}, y^{(i)})$$

$$\rightarrow a^{(i)} = \hat{y}^{(i)} = \sigma(z^{(i)}) = \sigma(w^T x^{(i)} + b)$$

$$(x^{(i)}, y^{(i)})$$

$$\underline{dw_1^{(i)}}, \underline{dw_2^{(i)}}, \underline{db^{(i)}}$$

$$\frac{\partial}{\partial w_1} J(w, b) = \frac{1}{m} \sum_{i=1}^m \frac{\partial}{\partial w_1} \ell(a^{(i)}, y^{(i)})$$

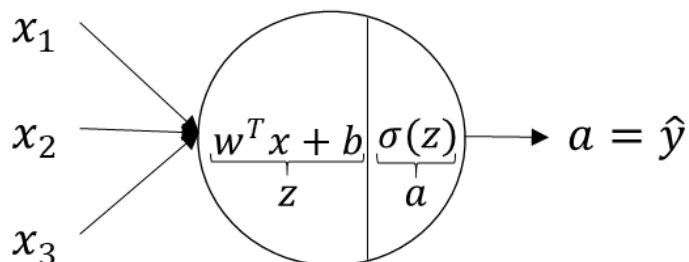
$$\underline{dw_1^{(i)}} - (x^{(i)}, y^{(i)})$$

단일 훈련 샘플의 도함수의 평균으로 경사하강법을 통해 전체적인 경사 구하는 것 가능

W1에 대한 전체 비용 함수의 도함수

Activation Function

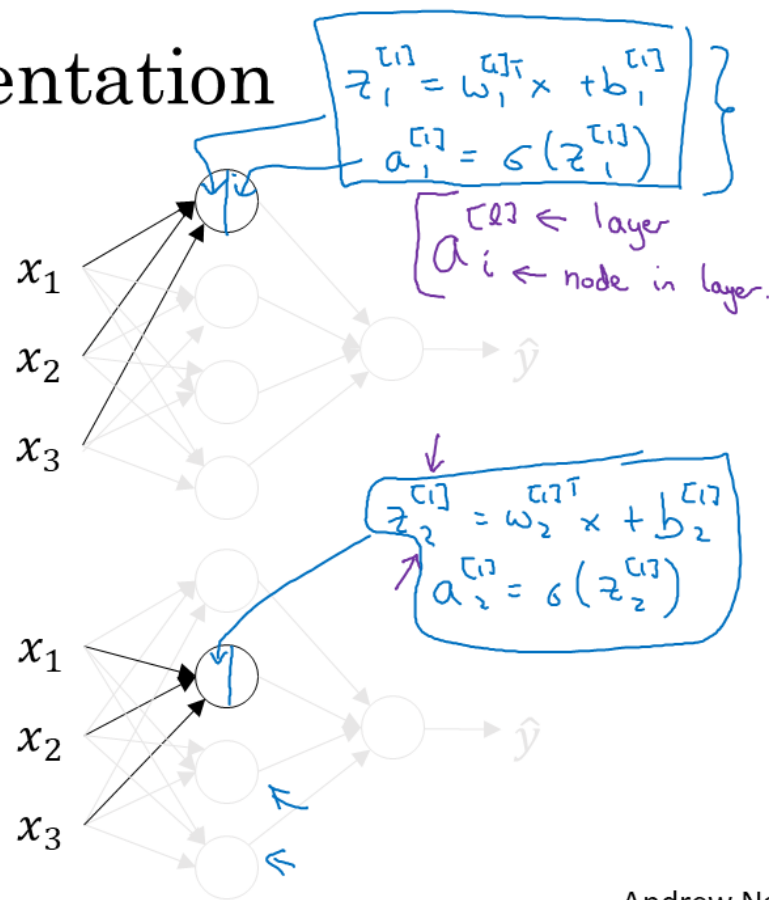
Neural Network Representation



로지스틱 회귀를 나타내는 이 원은
두 단계 계산을 나타냄

$$z = w^T x + b$$

$$a = \sigma(z)$$



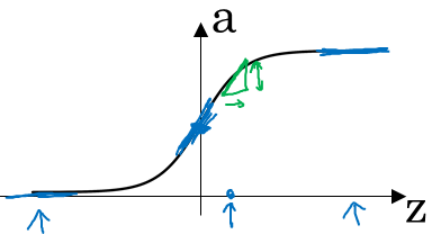
신경망은 이 원을
계속 반복

Andrew Ng

뉴런 하나를 살펴보면 선형적인 식($W^T x + b$)과 이를 비선형으로 만들어주는 activating function($\sigma(x)$)으로 이루어져 있습니다.
여기서 RL이므로 activating function은 sigmoid function입니다.

Activation Function

Sigmoid activation function


$$g(z) = \frac{1}{1 + e^{-z}}$$
$$a = g(z) = \frac{1}{1 + e^{-z}}$$
$$g'(z) = \frac{d}{dz} g(z) = \text{slope of } g(z) \text{ at } z$$
$$= \frac{1}{1 + e^{-z}} \left(1 - \frac{1}{1 + e^{-z}} \right)$$
$$= g(z) (1 - g(z)) \leftarrow$$
$$= \boxed{a(1-a)} \quad \left| \begin{array}{l} g'(z) = a(1-a) \\ \uparrow \end{array} \right.$$
$$z = 10, \quad g(z) \approx 1$$
$$\frac{d}{dz} g(z) \approx 1(1-1) \approx 0$$
$$z = -10, \quad g(z) \approx 0$$
$$\frac{d}{dz} g(z) \approx 0(1-0) \approx 0$$
$$z = 0, \quad g(z) = \frac{1}{2}$$
$$\frac{d}{dz} g(z) = \frac{1}{2} \left(1 - \frac{1}{2} \right) = \frac{1}{4}$$

Andrew Ng

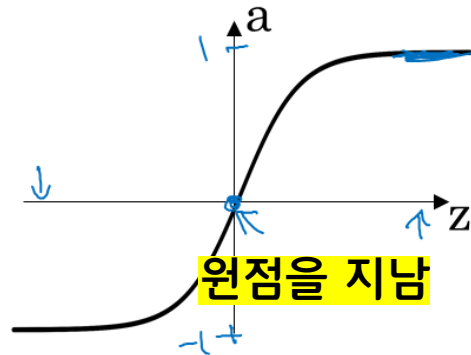
Sigmoid function은 RL에서 output layer에서 사용하고, 그 이외에는 잘 사용하지 않습니다. 그 이유는 z 가 너무 작거나 크면 기울기가 0에 가까워 지게 됩니다. 그러면 신경망의 학습이 너무 느려지게 됩니다. Gradient가 너무 작으니 update도 작게 되기 때문입니다. 따라서 대부분 RL에서 output layer에서만 사용합니다.

Activation Function

Tanh activation function

tanh

원점을 지나고 비율이 달라짐 1과 -1사이



$$g(z) = \tanh(z) \\ = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

Tanh 식

$$g'(z) = \frac{d}{dz} g(z) = \text{slope of } g(z) \text{ at } z \\ = 1 - (\tanh(z))^2 \leftarrow$$

$$a = g(z), \quad g'(z) = 1 - a^2$$

$$\begin{array}{l} z=10 \quad \tanh(z) \approx 1 \\ \quad \quad g'(z) \approx 0 \\ z=-10 \quad \tanh(z) \approx -1 \\ \quad \quad g'(z) \approx 0 \\ z=0 \quad \tanh(z) = 0 \\ \quad \quad g'(z) = 1 \end{array}$$

Tanh

-1에서 1사이

Andrew Ng

hyperbolic tangent function(Tanh function)의 식은 위와 같다. Tanh function은 거의 항상 sigmoid보다 좋은 성능을 낸다. 그 이유는 함수값의 평균이 0에 가깝기 때문이다.

Activation Function

은닉층에는 tanh 활성화함수를 쓰고 출력층에는 시그모이드 함수를 쓴다

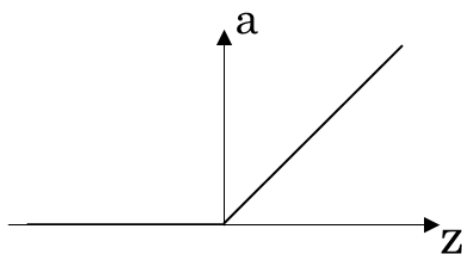
시그모이드 함수와 tanh 함수의 단점은 z 가 굉장히 크거나 작으면
함수의 도함수가 굉장히 작아진다는 것이다

인기있는 함수 : ReLU

이진 분류 출력층에는 시그모이드 함수를 많이 사용한다
활성화 함수에 ReLU가 기본값으로 많이 사용된다

Activation Function

ReLU and Leaky ReLU



ReLU

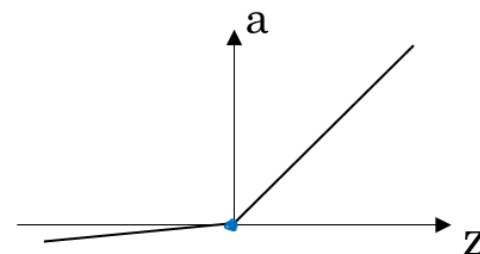
$$g(z) = \max(0, z)$$

공식

$$\rightarrow g'(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \\ \text{undefined} & \text{if } z = 0 \end{cases}$$

$z = 0.0000000000$

z가 양수면 도함수가 1이고 음수면 도함수가 0이다
0일때는 잘 정의하지 않는다 될 확률도 적다



Leaky ReLU

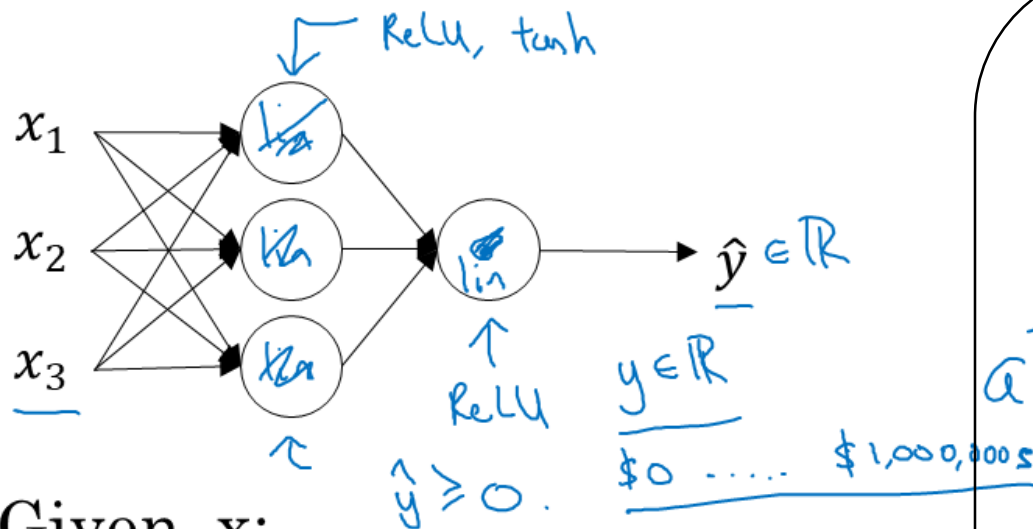
$$g(z) = \max(0.01z, z)$$
$$g'(z) = \begin{cases} 0.01 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

Andrew Ng

대부분의 z에 대해 기울기가 더 빠르게 학습
학습을 느리게 하는 원인인 함수의 기울기가 0에
가까워지는 것을 막음

Why Non-linear Activation Function

Activation function



Given x :

→ $z^{[1]} = W^{[1]}x + b^{[1]}$ **Z1로 대체**

→ $a^{[1]} = g^{[1]}(z^{[1]})$ $z^{[1]}$

→ $z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$

→ $a^{[2]} = g^{[2]}(z^{[2]})$ $z^{[2]}$

$g(z) = z$
"linear activation function"

선형 활성화 함수

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

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

$$a^{[2]} = W^{[2]}(W^{[1]}x + b^{[1]}) + b^{[2]}$$

$$= (W^{[2]}W^{[1]})x + (W^{[2]}b^{[1]} + b^{[2]})$$

$$= W'x + b'$$

$$g(z) = z$$

**선형 활성화 함수만 쓰면,
신경망은 층을 쌓아도 선형식만 출력하게 된다
→ 은닉층이 없는 것과 같다**

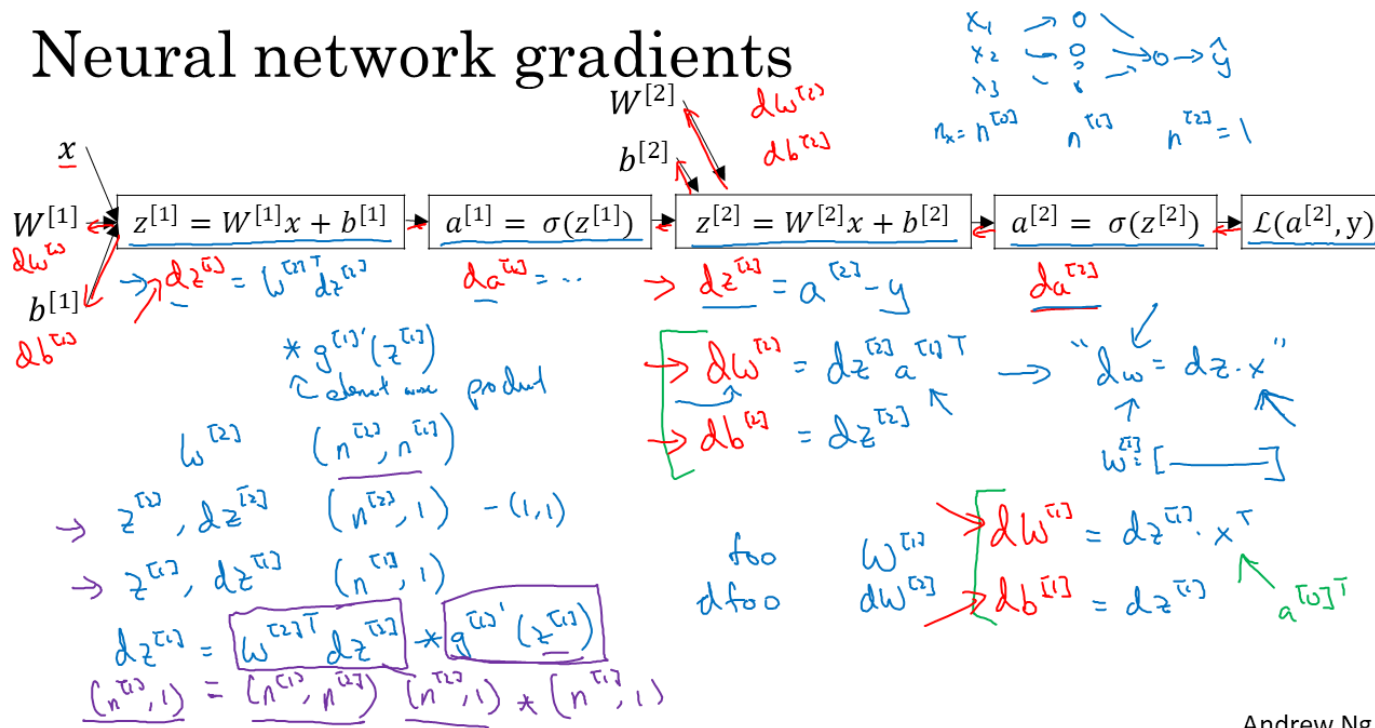
Why Non-linear Activation Function

선형 활성화 함수를 쓸 수 있는 곳은 대부분 출력층
(출력값인 y 값이 0부터 무한대까지 실수값이 되도록)

만약 activation function이 선형이면 이는 activation function이 없는 것과 같습니다. 단순 k 배에 상수를 더하는 것은 W 와 b 로 해줄 수 있기 때문입니다. 그러므로 위에서 볼 수 있듯이 input과 output은 단순한 선형관계가 되어, 선형회귀가 되어버립니다. 여러층의 layer도 필요가 없어지게 됩니다. 이와 같은 이유로 activation function으로 non-linear를 쓰는 것입니다.

Gradient Descent

Neural network gradients



Andrew Ng

[2] layer(여기선 마지막 layer, 즉 output layer)부터는 단층 RL Neural Network의 GD와 식이 같습니다. 한가지 다른점은 $dW^{[2]}$ 의 경우 행벡터라 $dw = adz$ 의 식을 transpose 해준 $dw = dza^T$ 의 식을 만족시킨다는 것입니다. [1] layer의 미분값은 chain rule을 사용하여 앞에서 구한 [2] layer의 미분값들 이용하면 구할 수 있습니다. 또한 행렬의 성분 하나에 대해서 생각하면 좀 더 쉽게 생각할 수 있습니다. 예를들어 $dz[1] = da[1]/dz[1] * dz[1]/da[2] * dz[2] = g'(z[1])w[2]dz[2]$ 이고 (recap에서 다루었습니다.)이를 행렬로 나타내면 $W^T[2]dz[1] g'(z[1])$ 가 됩니다.(g 는 activation function을 뜻합니다. 는 element wise product를 뜻합니다.) 여기서 element wise product는 같은 위치의 성분끼리 곱하는 것을 뜻합니다. 예를 들어 $[1,2]*[2,3] = [2, 6]$ 입니다.

Gradient Descent

요약

Summary of gradient descent

$$\underline{dz^{[2]}} = \underline{a^{[2]}} - \underline{y}$$

$$dW^{[2]} = dz^{[2]} a^{[1]T}$$

$$db^{[2]} = dz^{[2]}$$

$$\underset{(n^{[1]}, 1)}{dz^{[1]}} = W^{[2]T} dz^{[2]} * g^{[1]'}(z^{[1]})$$

$$dW^{[1]} = dz^{[1]} x^T$$

$$db^{[1]} = dz^{[1]}$$

$$\underline{dZ^{[2]}} = \underline{A^{[2]}} - \underline{Y}$$

$$dW^{[2]} = \frac{1}{m} dZ^{[2]} A^{[1]T}$$

$$db^{[2]} = \frac{1}{m} \text{np.sum}(dZ^{[2]}, \text{axis} = 1, \text{keepdims} = \text{True})$$

$$\underset{(n^{[1]}, m)}{dZ^{[1]}} = \underset{(n^{[1]}, m)}{W^{[2]T} dZ^{[2]}} * \underset{(n^{[1]}, m)}{g^{[1]'}(Z^{[1]})}$$

↓ elementwise product

$$dW^{[1]} = \frac{1}{m} dZ^{[1]} X^T$$

$$db^{[1]} = \frac{1}{m} \text{np.sum}(dZ^{[1]}, \text{axis} = 1, \text{keepdims} = \text{True})$$

$$J(\cdot) = \frac{1}{m} \sum_{i=1}^n \mathcal{L}(\hat{y}_i, y_i)$$

Initialization

신경망에서 변수를 임의값으로 초기화하는 것은 굉장히 중요하다
로지스틱 회귀의 경우 모두 0으로 초기화하여도 괜찮지만, 신경망에서 모두 0으로 초기화하고
경사하강법을 적용할 경우 올바르게 동작하지 않는다

Initialization

parameter를 update하려면 초기값을 지정해주어야 하는데, 편하게 모두 0으로 시작하면 안될까? 라는 생각을 할 수 있습니다. 결론은 안된다!입니다. 그 이유는 모든 unit의 파라미터가 0이면 unit이 여러개가 필요하지 않게 됩니다. 그냥 unit한개와 0인 $W(\text{parameter})$ 한개만 있어도 같은 작용을 합니다. ReLU를 사용할 경우 학습도 되지 않습니다.

Initialization

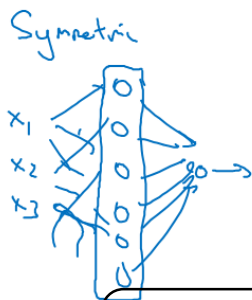
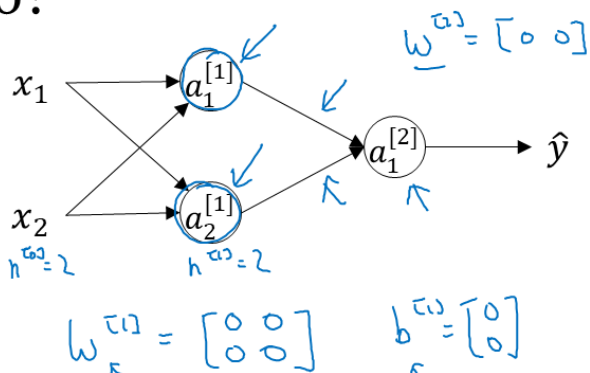
두 유닛이 같은 함수를 계산하는 것으로 시작

→ 출력 유닛에도 항상 같은 영향

→ 많은 훈련을 해도 같은 함수, 은닉 유닛 실제로는 하나

What happens if you initialize weights to zero?

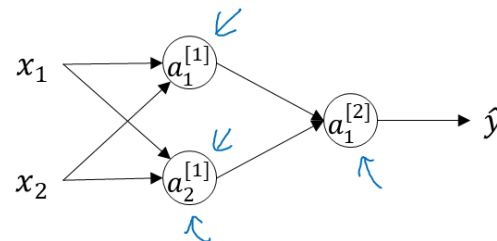
Random initialization



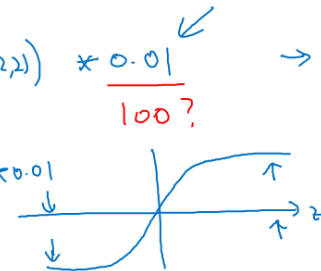
각 열이 같은 값

$$W^0 = \begin{bmatrix} \dots \\ \dots \end{bmatrix}$$

여떤 샘플의 경우에도 같은 값
두 은닉 유닛 모두 정확히 같은 함수



$$\begin{aligned} w^{00} &= \text{np.random.randn}(2,2) * 0.01 \\ b^{00} &= \text{np.zeros}(2,1) \\ w^{01} &= \text{np.random.randn}(1,2) * 0.01 \\ b^{01} &= 0 \end{aligned}$$



$$\begin{aligned} z^{00} &= W^{00}x + b^{00} \\ a^{00} &= g^{00}(z^{00}) \end{aligned}$$

Andrew Ng

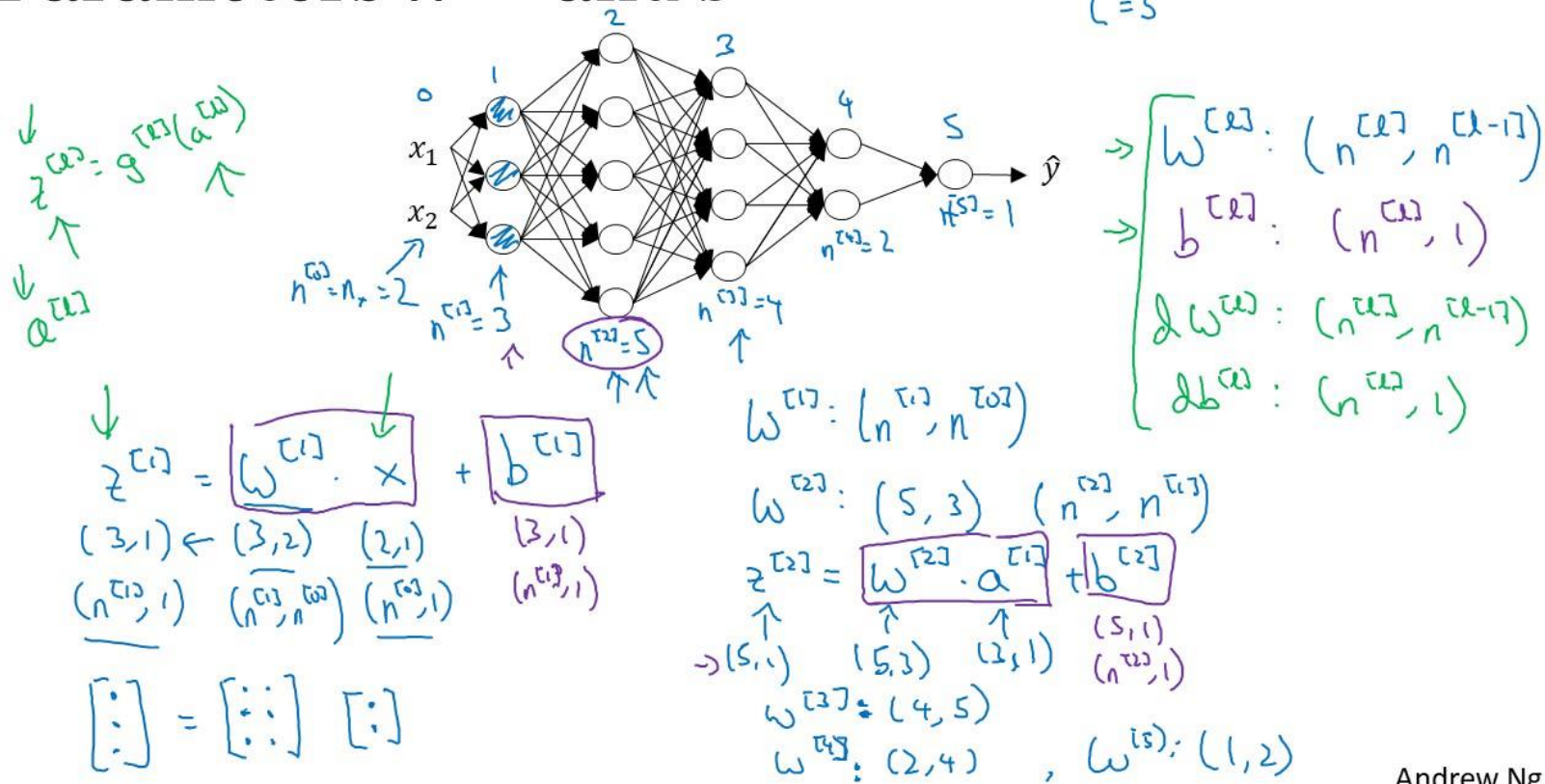
Andrew Ng

따라서 변수를 random하게 파라미터를 초기화합니다. 여기서 정규분포에서 값을 뽑아서 0.01을 곱해줬습니다. 0.01을 곱하는 이유는 파라미터를 작게 하기위해서인데, sigmoid나 Tanh 함수는 z 가 조금만 크거나 작아져도 기울기가 0에 가까워져 모델의 학습이 느려지게 됩니다. 따라서 parameter를 작게해서 z 값을 너무 크지 않게 하기 위하여 0.01을 곱하는 것입니다.

Deep L-layer neural network

what is a deep neural network?

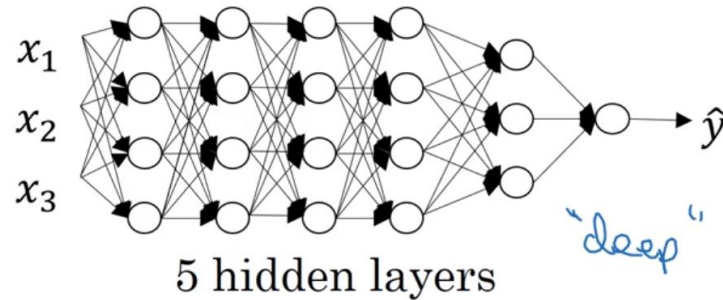
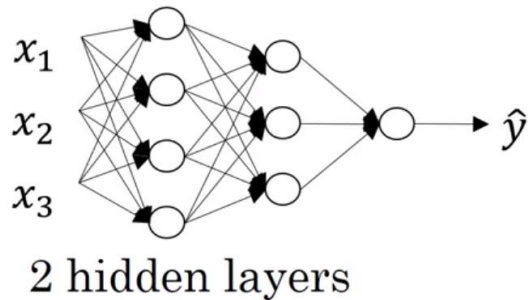
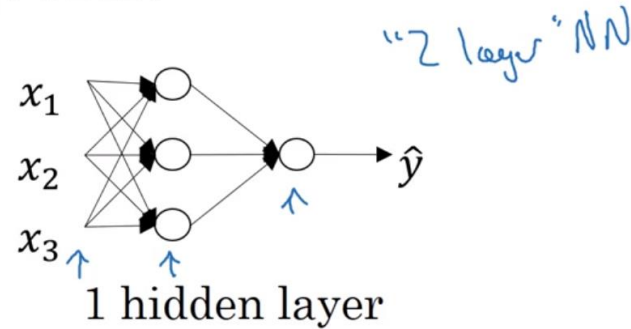
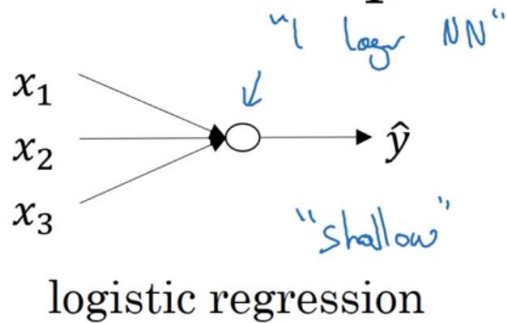
Parameters $W^{[l]}$ and $b^{[l]}$



Deep L-layer neural network

what is a deep neural network?

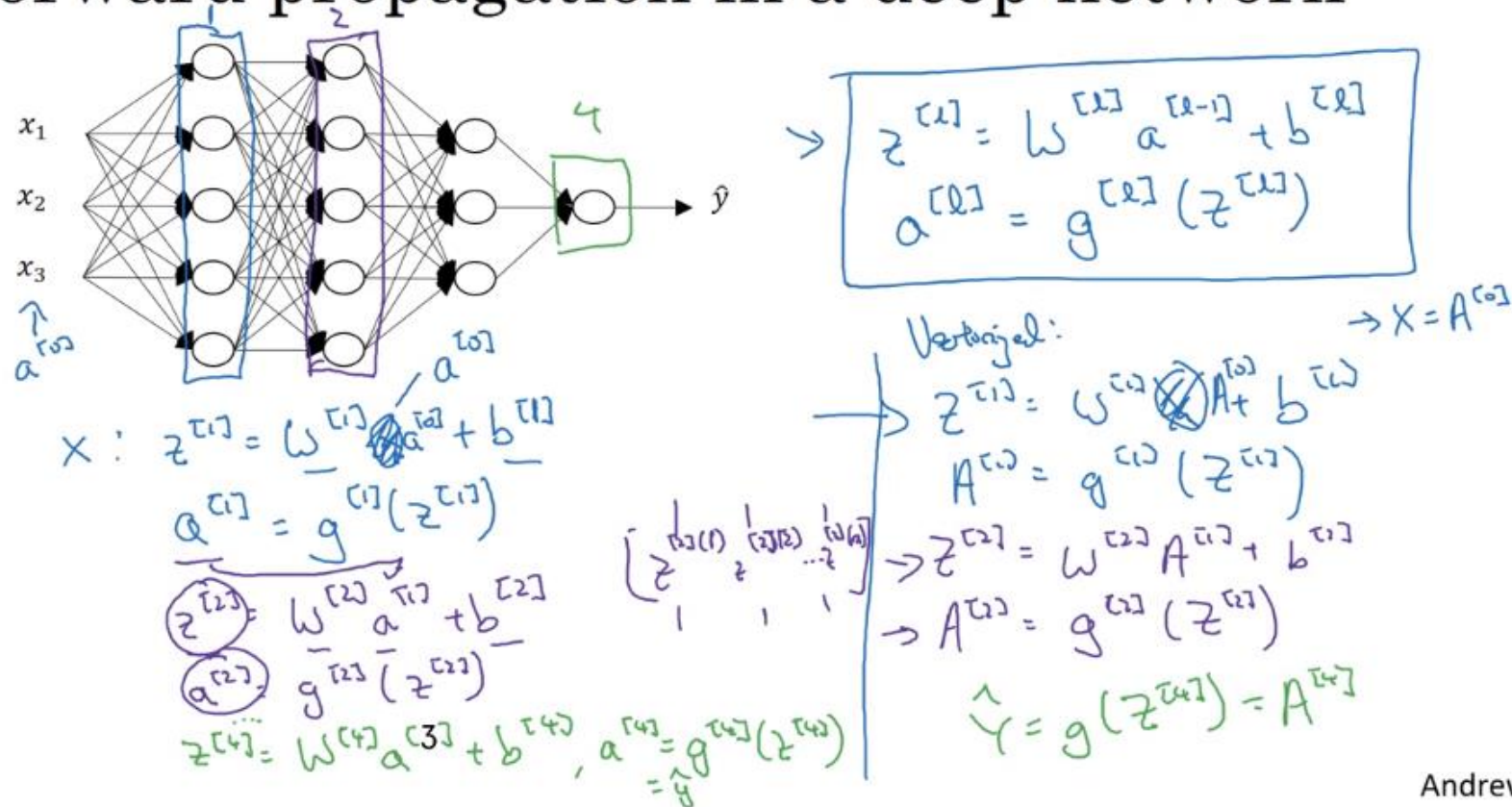
What is a deep neural network?



Layer이 1인 경우를 Shallow 하다고 하며, 이는 logistic regression과 크게 다를바가 없다. 현재의 모델들과 같이 Layer가 많은 경우를 Deep하다고 한다.

Deep L-layer neural network

Forward propagation in a deep network

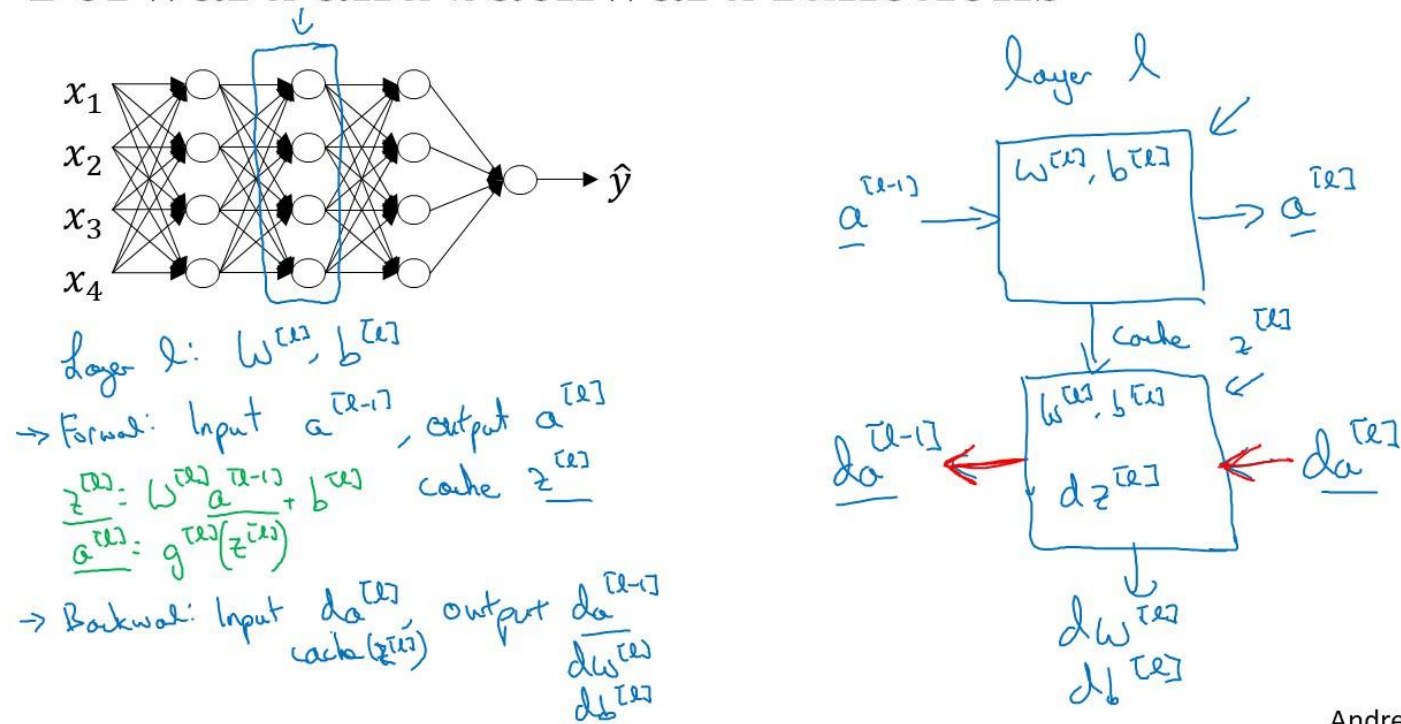


Andrew Ng

Forward propagation을 할 경우는 위 그림과 같이 표현 가능하다.

Forward and backward propagation

Forward and backward functions



Andrew Ng

각 레이어에서의 Forward 와 Backward propagation 계산은 위와 같다. 즉, Forward는 앞서 말해온 바와 같이 계산하고, Backward는 Forward의 최종 결과값의 loss를 역방향으로 input으로 두어, 계산해 나가게 된다.

Forward and backward propagation

Forward propagation for layer l

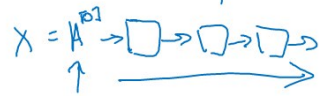
→ Input $a^{[l-1]}$ ←

→ Output $a^{[l]}$, cache $(z^{[l]})$

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

$$a^{[l]} = g^{[l]}(z^{[l]})$$

$$\begin{matrix} a^{[l]} \\ A^{[l]} \end{matrix}$$



Vectorized:

$$z^{[l]} = W^{[l]} \cdot A^{[l-1]} + b^{[l]}$$

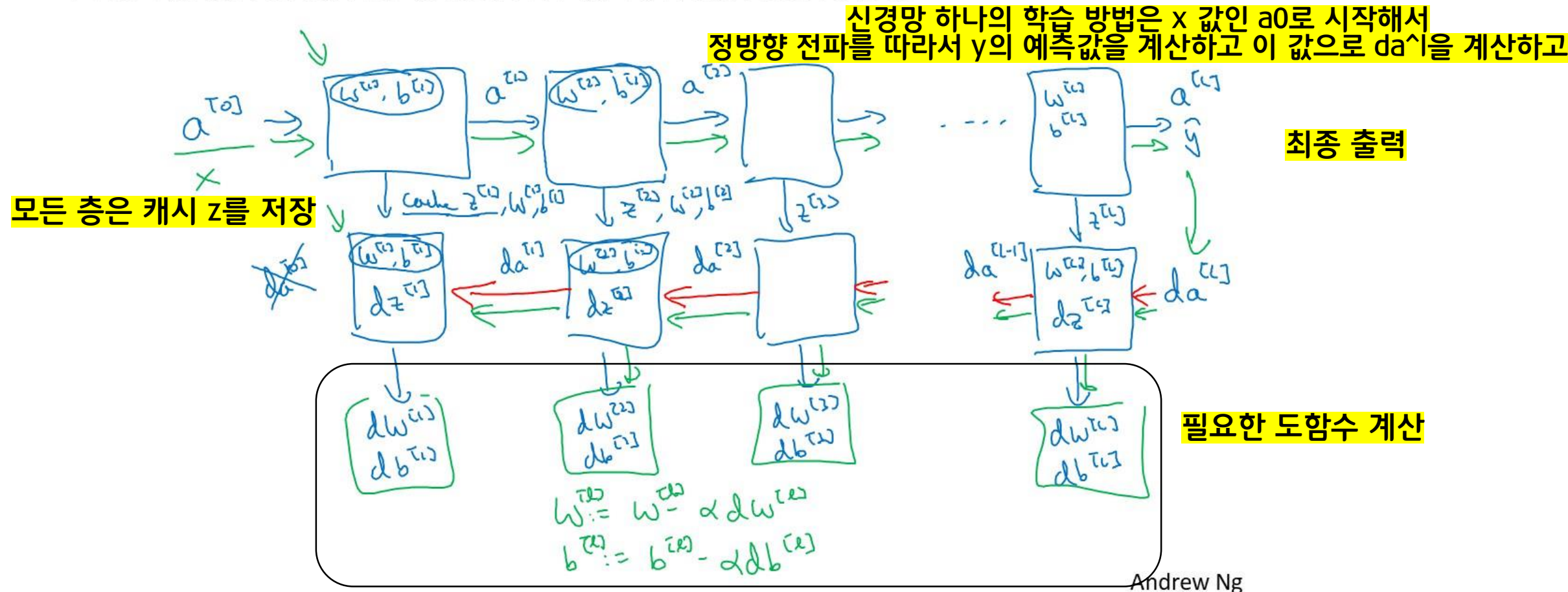
$$A^{[l]} = g^{[l]}(z^{[l]})$$

Andrew Ng

벡터화, 별 차이 없음

Forward and backward propagation

Forward and backward functions

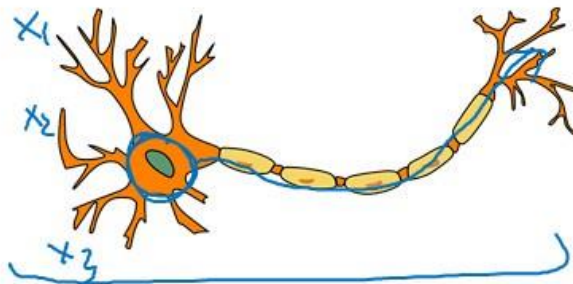


Forward and backward propagation

Forward and backward propagation

$$\begin{aligned}Z^{[1]} &= W^{[1]}X + b^{[1]} \\A^{[1]} &= g^{[1]}(Z^{[1]}) \\Z^{[2]} &= W^{[2]}A^{[1]} + b^{[2]} \\A^{[2]} &= g^{[2]}(Z^{[2]}) \\&\vdots \\A^{[L]} &= g^{[L]}(Z^{[L]}) = \hat{Y}\end{aligned}$$

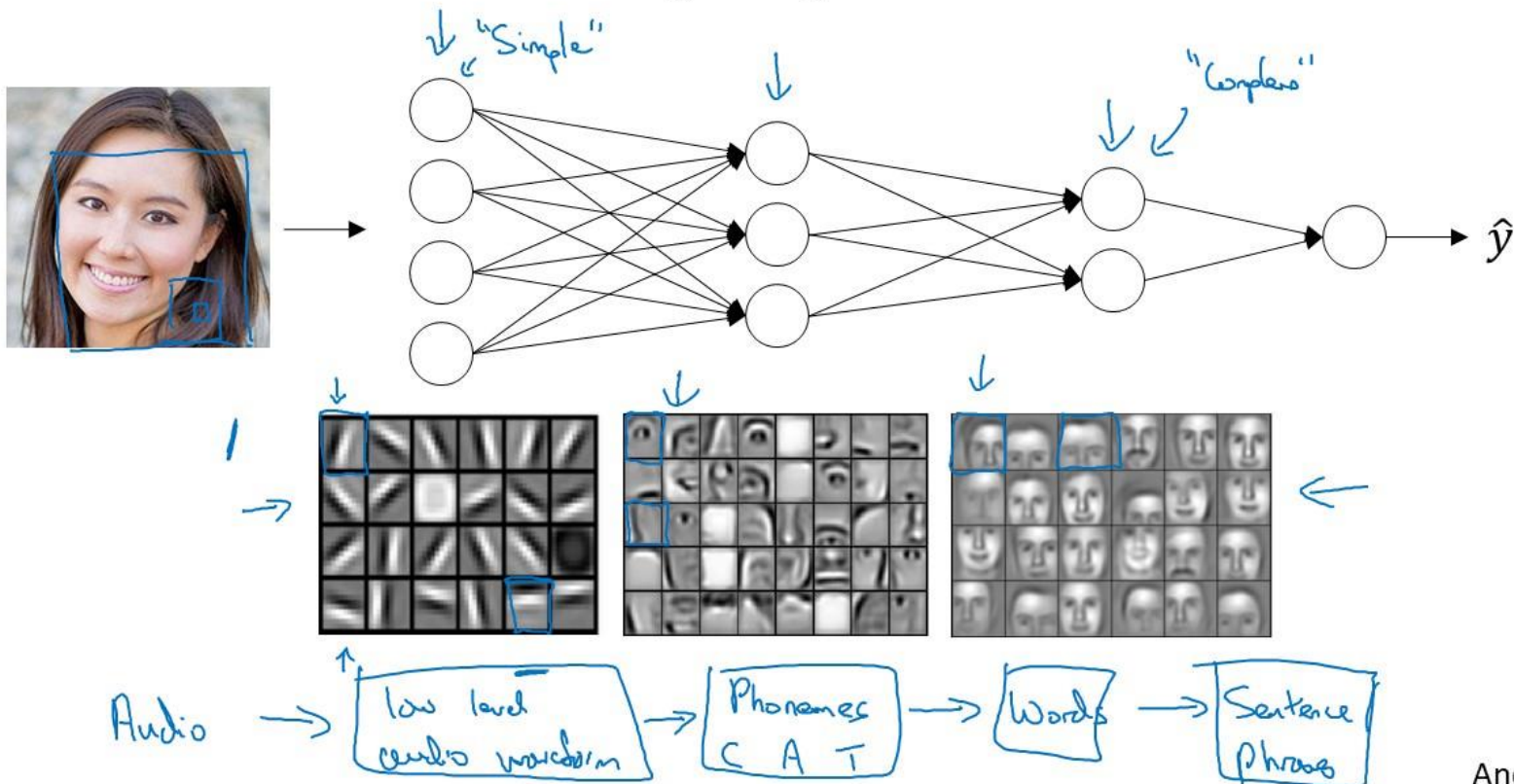
"It's like the brain."



$$\begin{aligned}dZ^{[L]} &= A^{[L]} - Y \\dW^{[L]} &= \frac{1}{m} dZ^{[L]} A^{[L]T} \\db^{[L]} &= \frac{1}{m} np.sum(dZ^{[L]}, axis = 1, keepdims = True) \\dZ^{[L-1]} &= dW^{[L]T} dZ^{[L]} g'^{[L]}(Z^{[L-1]}) \\&\vdots \\dZ^{[1]} &= dW^{[L]T} dZ^{[2]} g'^{[1]}(Z^{[1]}) \\dW^{[1]} &= \frac{1}{m} dZ^{[1]} A^{[1]T} \\db^{[1]} &= \frac{1}{m} np.sum(dZ^{[1]}, axis = 1, keepdims = True)\end{aligned}$$

Why deep representations?

Intuition about deep representation



Andrew Ng

첫번째 층은 특성 탐지거나 모서리탐지기, 모서리를 형성하기 위해 픽셀을 그룹화해서 얼굴의 일부 형성, 어떤 뉴런에서는 눈의 일부 어떤 뉴런에서는 입의 일부를 찾음, 많은 모서리를 한데 모아서 얼굴의 일부 감지

Why deep representations?

서로 다른 얼굴의 일부를 최종적으로 모아서 눈, 코, 귀, 뺨을 만들어

서로 다른 종류의 얼굴을 감지함

신경망 초기 층에서는 모서리와 같은 간단한 함수를 감지하고 그 후 신경망 층에서 이것을 감지해

더 복잡한 함수를 학습

Parameters vs Hyper parameters

하이퍼파라미터

: 학습률

: iterations

: 히든 레이어 수

: 히든 유닛 개수

: activation function 종류

이 매개변수들은 궁극적으로 매개변수 w, b 를 통제

Parameters vs Hyper parameters

딥러닝을 적용하는 것은 매우 경험적인 과정이다

아이디어와 코딩(실행)과 경험이 계속 반복되는 과정

어떤 값의 학습률을 사용해야 할지 확실하지 않다면
비용함수 J 가 내려가는 것을 확인해야 함

Parameters vs Hyper parameters

딥러닝을 적용하는 것은 매우 경험적인 과정이다

아이디어와 코딩(실행)과 경험이 계속 반복되는 과정

어떤 값의 학습률을 사용해야 할지 확실하지 않다면
비용함수 J 가 내려가는 것을 확인해야 함