《WEEK 2》 NN Implemential techniques 신경망 프로그래밍 기초.

OVER VIEW

ex) ① → m training set. → loop

⇒ explicit → **not for loop!** → for loop 쓰지 않고 training

② forward pause, forward propagation = 순방향 경로 = 순전파

computation in NN. 계산법. why 이거 사용?

why?

backward pause, backward propagation 역방향 경로 역전파

logistic regression. → 로지스틱 회귀.

↳ 이진 분류 (Binary Classification)를 위한 알고리즘

고양이 예제 Input x

output y.

이미지 - RGB. → 64 × 64 × 3 (64 × 64: 이미지 사이즈.)

그림의 5×4.

intensity → feature vector.
세고

→ unroll all pixel into a input feature vector x (unroll: 펼치다)

to unroll → find feature vector x
리스트에 일렬로 나열.

if 64×64

↳ 64 × 64 × 3. = 12288. (3: RGB)

$n = n_x = 12288.$

↳ dimension of input feature x 차원

나타내기 위해 n 사용.

이진분류

입력벡터 x → 입력으로 주어 분류라 학습
classifier

output y → 1 or 0 ?
cat noncat.

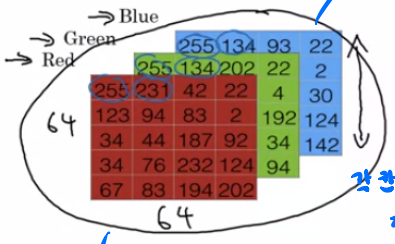
Binary Classification

64

64

1 (cat) vs 0 (non cat)

그림은 한 장의 특징 벡터. y



각 칸마다 채도값.

이미지 하나는 하나의 벡터.

$$x = \begin{bmatrix} 255 \\ 231 \\ \vdots \\ 255 \\ 134 \\ \vdots \end{bmatrix}$$

RGB

$64 \times 64 \times 3 = 12288$ - total dimension

$n = n_x = 12288$

입력 특징벡터의 차원

입력벡터 x 로 표현된 이미지.

$x \longrightarrow y$

Andrew Ng

$\hat{y}$ → y 확률. $0 \le \hat{y} \le 1$

$\hat{y} = P(y=1|x)$

y가 x가 1일 확률.

x -> 입력 => d:mensional vector $x \in \mathbb{R}^{n_x}$

$\hat{y}$ -> 고양이일 확률.

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

Real num.

Input x, para w, b. => $\hat{y}$?

output $\hat{y} = w^T x + b$

-> $0 \le \quad \le 1$ 범위 안에 좀이 가깝게 원리 X.

회귀를 가르키는데 쓰는 초기 알고리즘
신경망에 대한 초기법

Notation → 표기법

single train example → (x, y).

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

x 차원을 가진 특징벡터 └ lable.

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

m개의 training examples

└ m_{train}. → 트레이닝 예제 수.

m_{test} → 테스트 예제 수.

:implement - 구현하다

:input x - stack in columns

matrix

$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix} \quad n_x \text{ (↑)}, \quad m \text{ (←→)}$$

← 이게 더 쉽대

$$Y = \begin{bmatrix} y^{(1)}, y^{(2)} \cdots y^{(m)} \end{bmatrix}$$

$Y \in \mathbb{R}^{1 \times m}$ $Y.shape = (1, m)$.

$X \in \mathbb{R}^{n_x \times m}$ $X.shape\ (n_x, m)$ 행렬의 형태를 알려주는 명령어.

└ 파이썬.

다른 트레이닝과 관련된 레이블. ⌐ 같은 방식으로 진행.

Notation

(x 차원의 특징 벡터)

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

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

$m = m_{train}$ 트레이닝 set의 수 m_{test} = #test examples. └ 테스트 ex의 수

:input x를 행벡으로 나타내는 방법

보통 벡터로 생각

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

Matrix

입력값을 이리 쌓은 것

n_x, m (예제의 수)

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

$X.shape = (n_x, m)$ 행과 열의 알려주는 명령어.

$Y = \begin{bmatrix} y^{(1)} & y^{(2)} & \cdots & y^{(m)} \end{bmatrix}$

(Y의 Matrix 또한 이렇게 쌓임)

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

$Y.shape = (1, m)$

x를 이리 쌓으면 이게 더 쉬움.

Andrew Ng

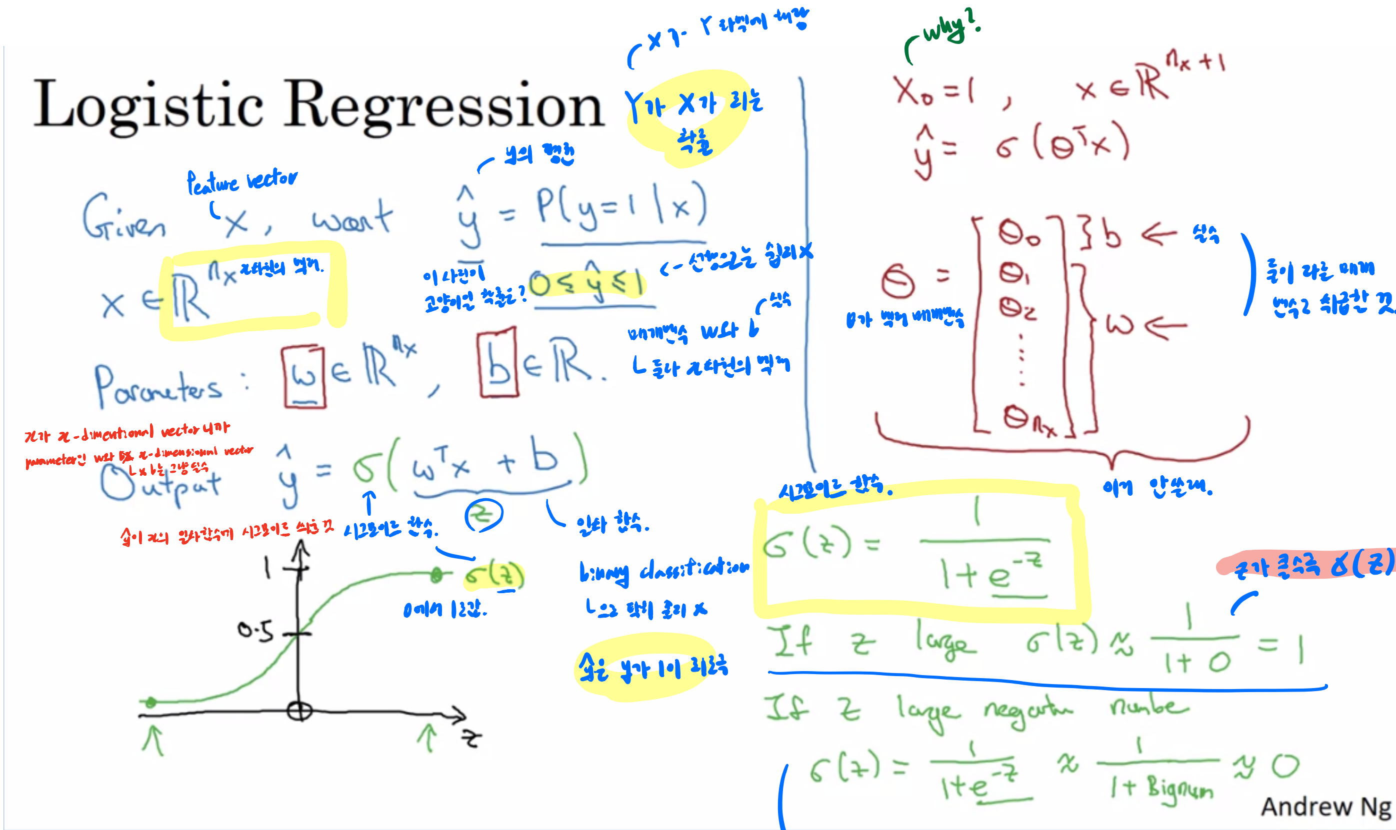
Logistic regression
└ binary classification을 위한 알고리즘.

supervised learning 지도학습

선형 회귀.

└ y → 지도학습에서 0이나 1. 이런 분류.

B = inter-spectrum



z가 클수록 σ(z)는 1에 가까움

z가 작으면 σ(z)는 0에 가까움

0 ≤ ŷ ≤ 1 이기 때문에 sigmoid 함수를 취한다.

ŷ이 y=1 라는 확률을 잘 측정했는지.

to train param W B → model

↑

define

cost func.

sigmoid ?

m개의 training set에서 param W와 B 찾음.

prediction $\hat{y}^{(i)}$

$\hat{y}^{(i)} \approx y^{(i)}$

sigmoid func

super

식 설명 : 괄호 안에 있는 것이 각각의 인덱스에 해당.

아 하라 괄호로 각각의 training set을 나타내는 소니

superscript = subscript

하라. 어깨 글자.

$\sigma(\quad)$

└ 시그모이드 함수인 듯?

괄호 하라 - 데이터 인덱스.

제곱 오차 squared error.

└ 작을수록 좋다.

<Loss func> ① $\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$. (X).

└→ 잘 없거나.

non-convex. → 비볼록.

gradient descent 랑 안 맞는.

└ 비슷한거 찾아서

볼록한 최적화를해.

⇒ optimization with multiple local optima.

복수 국부 최적의 optimization.

└→ gradient descent → may not find global optima

전역 최적값을 못 찾을 수 있음.

y값을 가리고 있을 때

얼마나 정확히 맞을

산출했는지 나타내는 함수.

Loss func ①로 (gradient descent가) 전역 최적값 못찾을 수 있기 때문에

Loss func ② 사용.

② $$\mathcal{L}(\hat{y}, y) = -(y \log \hat{y} + (1-y)\log(1-\hat{y})).$$

└ 얘도 작을수록 좋다.

if $y=1$: $\mathcal{L}(\hat{y}, y) = -\log \hat{y}$

└ 커야! 마이너스니까.

아 이거 부호 생각해 주는거 아니었나?

$\hat{y}$ 커야

↳ 시그모이드 func의 결과
1보다 클수 x.

=> $\hat{y}$는 1에 최대한 가깝게.

if $y=0$: $\mathcal{L}(\hat{y}, y) = -\log(1-\hat{y})$.

↳ 커야!

∴ $\hat{y}$이 가장 작아야.

=> 0에 가까워야.

↳ y가 1이면 $\hat{y}$는 가장 크게.
y가 0이면 $\hat{y}$는 가장 작게.

en 이걸 사용하는가? → 최적.

지금까지 single example2 였어.

Cost func : $J(w,b) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)})$

↳ entire set
cost of pumeters

↑ prediction output.

$= -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\hat{y}^{(i)} + (1-y^{(i)})\log(1-\hat{y}^{(i)})\right]$

Loss func → 비용 최소.

↳ cost func은 los func의 평균

=> Loss func는 sigle training example에 적용.
cost func은 파라미터의 비용을 나타낸 것.
Logistic regression 모델을 훈련
→ W와 B라는 파라미터를 찾고, cost func의 비용 J를 줄여나가는 것.

ŷ → 예측을 위한 선형 함수에 sigmoid 취한 것.

이걸 통해 w와 b를 찾아내기

Logistic Regression cost function

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

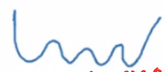
Given $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$, want $\hat{y}^{(i)} \approx y^{(i)}$. (ŷ: 예상치)

$x^{(i)}, y^{(i)}, z^{(i)}$ — i-th example : i번째 training example

Loss (error) function: $\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$ (ŷ: 알고리즘 결과, y: true label)

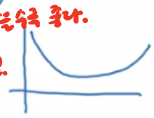
└ true label y를 갖는 경우 얼마나 정확히 출력값을 산출하는지 정의.

선형 회귀에서는 잘 안 쓴다. → 비볼록. └ 여러 지역최소 볼록하지 않기 때문임.



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

→ 값이 작을수록 좋다. 작을수록 좋다.



If $y = 1$: $\mathcal{L}(\hat{y}, y) = -\log \hat{y}$ ← want $\log \hat{y}$ large, want $\hat{y}$ large. y=1 일때 ŷ이 커야한다. └ 1에 가장 가까워야 한다.

If $y = 0$: $\mathcal{L}(\hat{y}, y) = -\log(1-\hat{y})$ ← want $\log(1-\hat{y})$ large …. want $\hat{y}$ small. y=0 일때 ŷ이 작아야한다. └ 0에 가장 가까워야

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

└ entire training set → Loss func의 평균

Andrew Ng

Loss func는 single training example을 바탕으로 정의.

Rafidah's effect

cost는 Loss 값의 평균.

└ cost of parameter

find w and b that minimize cost func J's value

Clarification about Upcoming Gradient Descent Video

Please note that in the next video at the second slide, there is a missing parenthesis.

The negative sign should apply to the entire cost function (both terms in the summation).

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

Also Andrew mentions: "To make this easier to draw, I'm going to ignore b for now, just to make this a *one-dimensional* plot instead of a high-dimensional plot.". T*his plot is actually two-dimensional,* it shows the loss J(w) for values of w.

복습: Loss - single set 에서 성능

cost - 전체 set 에서 파라미터가 얼마나 잘하는지

↳ cost는 m개 set 에서 Loss의 평균

원래 ½( )² 쓰는데 gradient descent에서 전역 최적값 안 나올 수 있어서 log 쓰는 Loss func 쓴다.

<Gradient descent Algorithm>

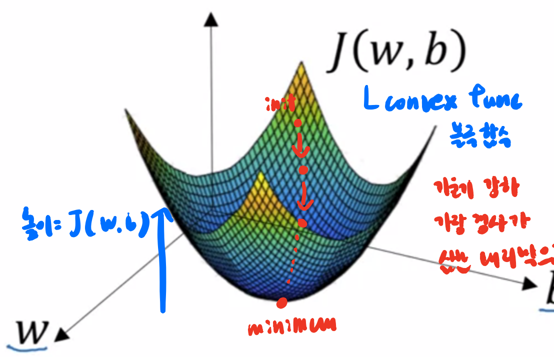
Gradient Descent

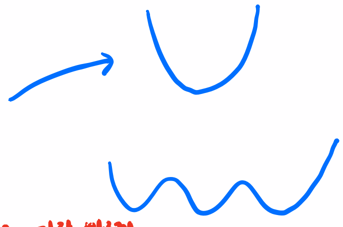
Recap: $\hat{y} = \sigma(w^T x + b)$, $\sigma(z) = \frac{1}{1+e^{-z}}$ ← 0과 1사이의 값에 들도록 하기 위한 sigmoid 함수.

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

cost func / average of loss func

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



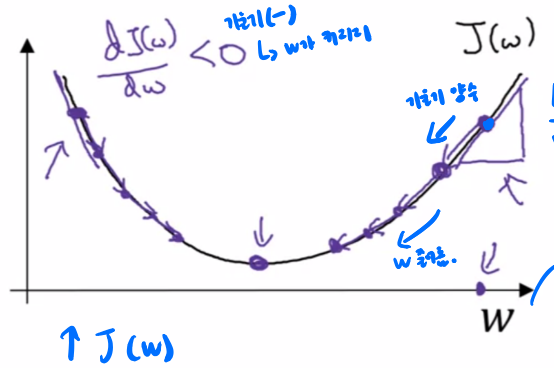


볼록하기 때문에 사용 가능.
L 그래서 위에서 loss 바꾼거.

Andrew Ng

w, b 실수로 가정

Gradient Descent



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

update

w가 점점 어디로 가는지

learning rate : 한번에 얼마만큼 기울기 강하?

미분하여.. derivative.

"dw"

$w := w - \alpha\, dw$

w에 얼마나 변화를 줄지 기울기로 보여 slope.

원래 w, b 2차원 지금 1차원으로 가정, $\frac{dJ(w)}{dw} = ?$ w만

$J(w,b)$

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

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

이렇게 써야 함.

∂ "partial derivative"

d ← J

J (함수)가 2개 이상의 변수로 이루어진 경우 사용하는 표기. d와 동일한 의미.

Andrew Ng

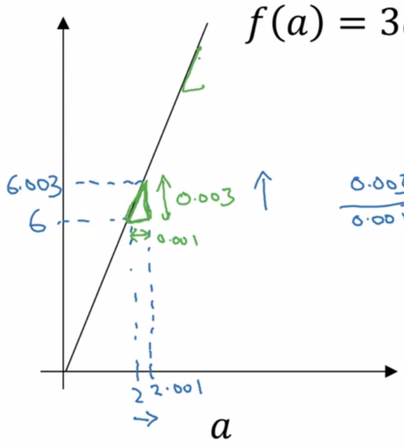
순간 변화율.

< Derivative (미분학) >

프로그래밍 과제 할 수 있는 정도. + forward backward 함수에 사용.

Intuition about derivatives

$f(a) = 3a$



$\frac{0.003}{0.001}$ $\frac{\text{height}}{\text{width}}$

→ $a = 2$ $f(a) = 6$

$a = 2.001$ $f(a) = 6.003$

→ slope (derivative) of $f(a)$ at $a = 2$ is 3 – 기울기 3.

→ $a = 5$ $f(a) = 15$

$a = 5.001$ $f(a) = 15.003$

slope at $a=5$ is also 3

$\frac{d f(a)}{da} = 3 = \frac{d}{da} f(a)$

0.001 ←
0.00000001
0.00000000001

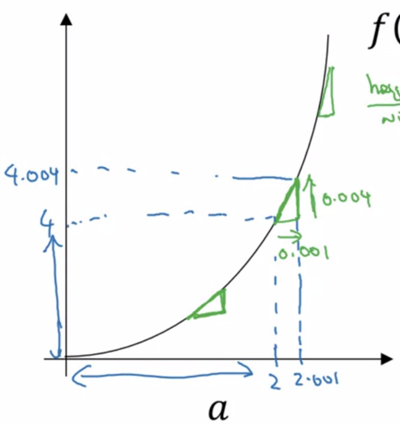
(극한 얘기하는 거겠지

Andrew Ng

기울기가 변하는 경우.

Intuition about derivatives

$f(a) = a^2$



0.001 ←
0.00000....01 ←

$\frac{d}{da} a^2 = 2a$

0.001
$(2a) \times 0.001$

$a = 2$ $f(a) = 4$

$a = 2.001$ $f(a) \approx 4.004$ (4.004001)

slope (derivative) of $f(a)$ at $a=2$ is 4.

$\frac{d}{da} f(a) = 4$ when $a = 2$ → 지점마다 기울기가 다르다

$a = 5$ $f(a) = 25$

$a = 5.001$ $f(a) \approx 25.010$

$\frac{d}{da} f(a) = 10$ when $a = 5$

$\frac{d}{da} f(a) = \frac{d}{da} a^2 = 2a$

Andrew Ng

More derivative examples

$f(a) = a^2$ $\frac{d}{da} f(a) = 2a$ (4) $a = 2$ $f(a) = 4$; $a = 2.001$ $f(a) \approx 4.004$

순간의 기울기,
순간의 변화율.

$f(a) = a^3$ $\frac{d}{da} f(a) = 3a^2$ ($3 \times 2^2 = 12$) $a = 2$ $f(a) = 8$; $a = 2.001$ $f(a) \approx 8.012$

$f(a) = \log_e(a)$ ($\ln(a)$) $\frac{d}{da} f(a) = \frac{1}{a}$ → $\frac{d}{da} f(a) = \frac{1}{2}$

$a = 2$ $f(a) \approx 0.69315$

$a = 2.001$ $f(a) \approx 0.69365$

0.0005 ↔ 0.0005

Andrew Ng

여기서부터 필기 정리된 부분. 중복 없는 부분

deeplearning.ai

Basics of Neural Network Programming

Computation Graph

computations of NN → forward pass / propagation step
└ 전방향
신경망 결과값 계산 후 backward pass 통해 기울기 계산

↓ why? example.

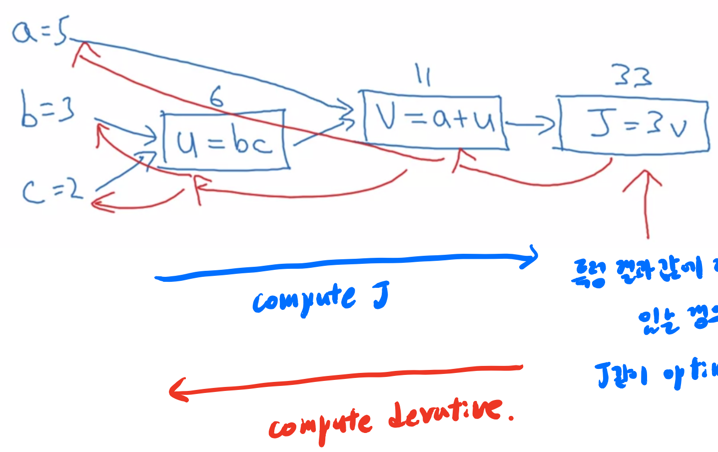
Computation Graph → compute J

$J(a,b,c) = 3(a + bc) = 3(5 + 3 \times 2) = 33$
└ cost func

(In the handwritten annotation, bc is marked as u, $a + u$ as v, and $3v$ as J.)

계산 순서 ↓

$u = bc$
$v = a + u$
$J = 3v$



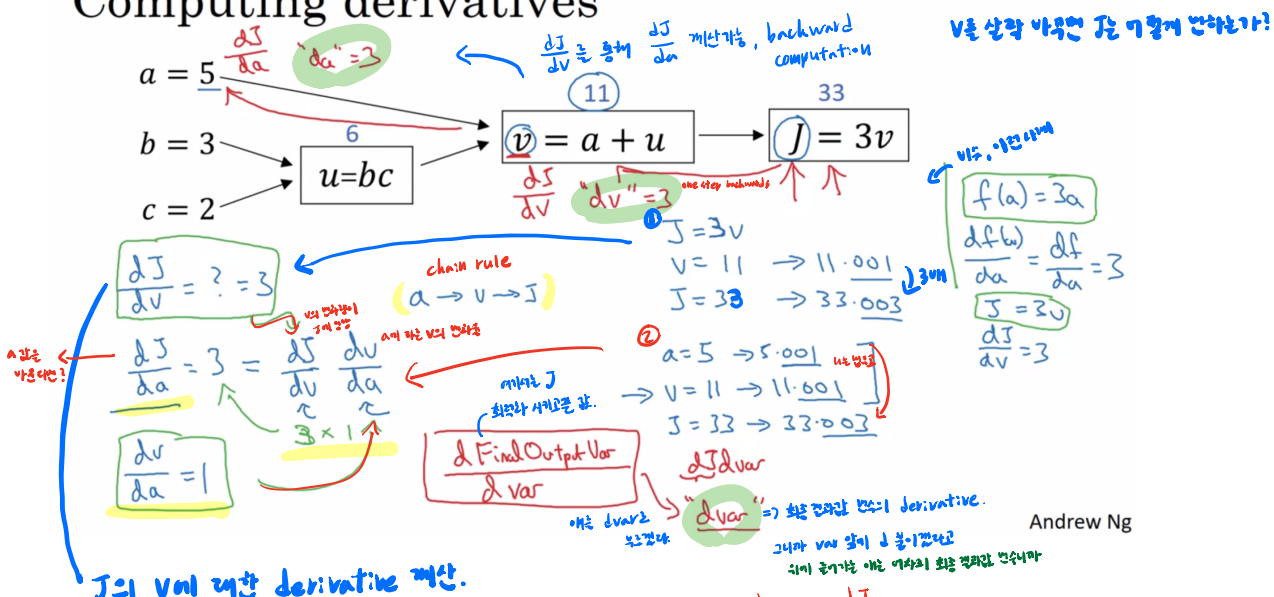
특정 결과값에 대한 변수가 있는 경우
J값이 optimize 하고싶은 경우

deeplearning.ai

Basics of Neural Network Programming

Derivatives with a Computation Graph

Computing derivatives



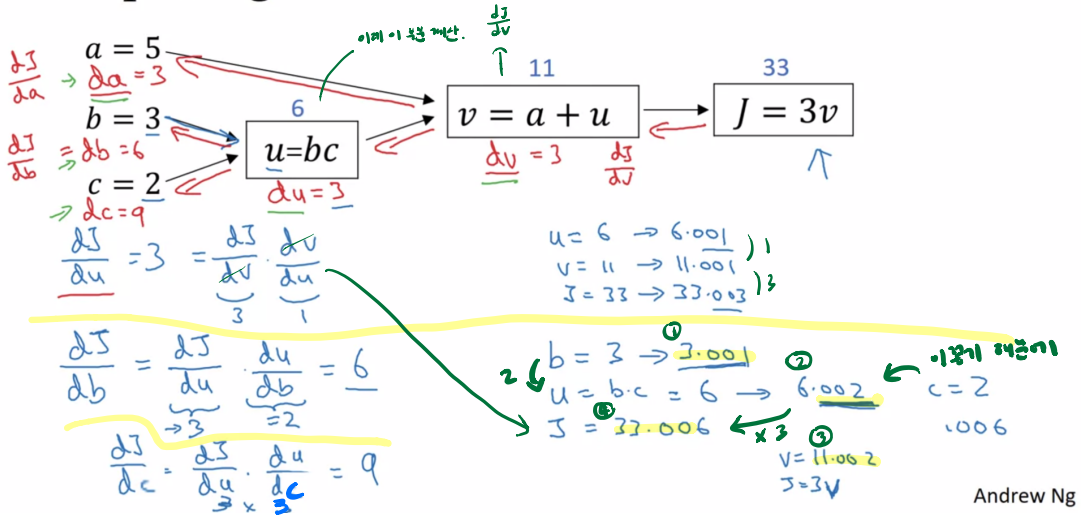
J의 var에 대한 derivative 계산.

② change a → change v → change J $\frac{dJ}{da} = \frac{dv}{da} \times \frac{dJ}{dv}$

review

$\frac{dJ}{dv} = 3$ $\frac{dJ}{da} = \frac{dJ}{dv} \times \frac{dv}{da}$ $3 \quad 1$

Computing derivatives



chain rule 쉽게 생각하면 되게 간단하다.

⇒ chain rule 활용하여 compute derivative backward pass

dv를 찾으면 da와 du를 찾을 수 있다. du → db, dc compute.

J를 계산하는 방법은 forward, derivative를 계산하는 방법은 backward.

→ Logistic regression model의 derivative 계산.

deeplearning.ai

Basics of Neural Network Programming

Logistic Regression Gradient descent

What you need to implement

로지스틱 회귀모델을 위한 가중치 업데이트 하는 주요 공식

Logistic regression recap

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

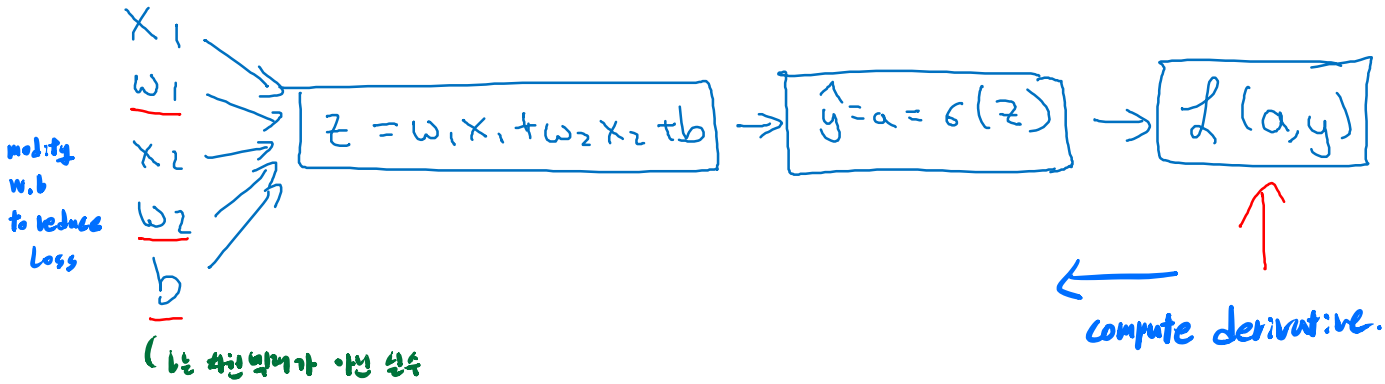
y: ground truth lable

$$\hat{y} = a = \sigma(z)$$

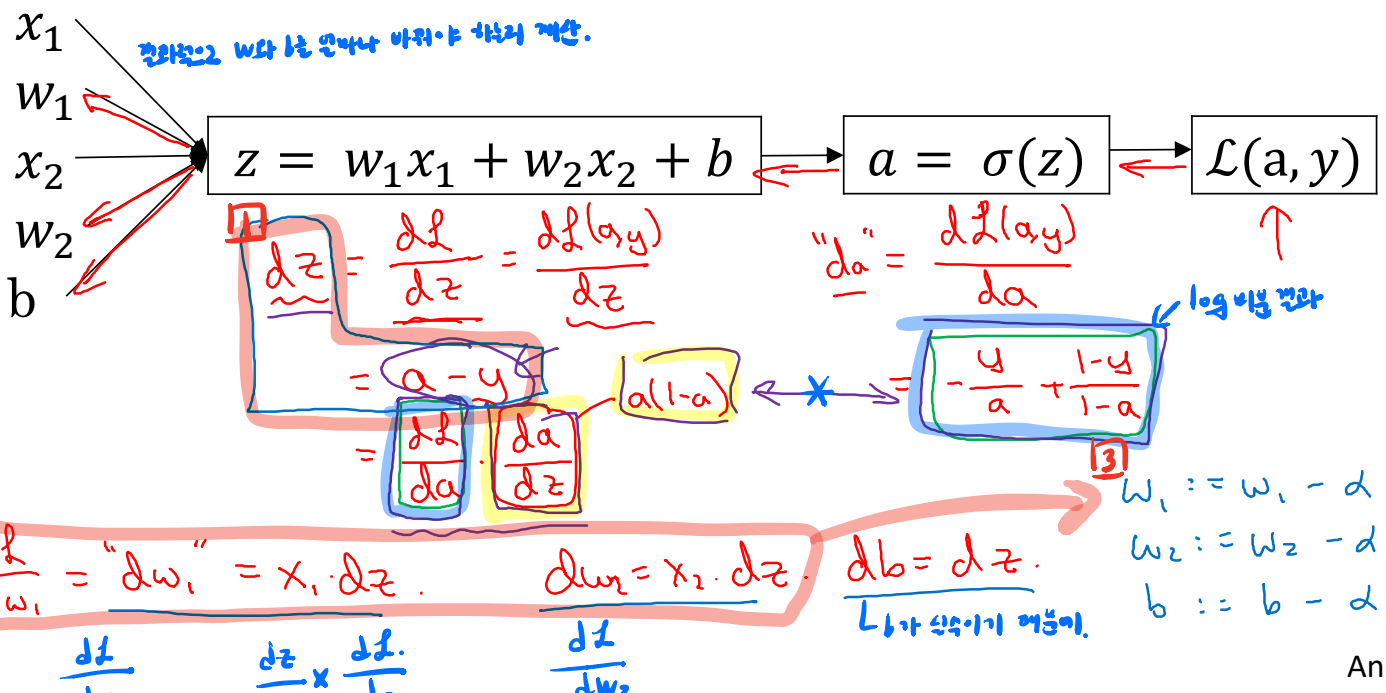
prediction

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

Loss



Logistic regression derivatives



Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Gradient descent on m examples

Logistic regression on m examples

cost func

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

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

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

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

1 training example

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

dw_1은 모든 $dw_1^{(i)}$의 평균값이 된다는 뜻.

$dw_1^{(i)}$ — $(x^{(i)}, y^{(i)})$

기울기 강하와 로지스틱 회귀 손익이 같이 계산된다.

Logistic regression on m examples

$J=0;\ dw_1=0;\ dw_2=0;\ db=0$

→ For $i=1$ to m

$z^{(i)} = w^T x^{(i)} + b$

$a^{(i)} = \sigma(z^{(i)})$ 누적해서 계산해야 하니까

$J += -\left[y^{(i)} \log a^{(i)} + (1-y^{(i)}) \log(1-a^{(i)})\right]$

$dz^{(i)} = a^{(i)} - y^{(i)}$

② $dw_1 += x_1^{(i)} dz^{(i)}$

$dw_2 += x_2^{(i)} dz^{(i)}$ ← $n=2$ 레에 확장한 경우 가정하에

$db += dz^{(i)}$

dw_3 ⋮ dw_n

$J /= m$ ←

$dw_1 /= m$; $dw_2 /= m$; $db /= m$. ←

↑ ↑ ↑

나머지 정의값.

$dw_1 = \frac{\partial J}{\partial w_1}$

dw_2와 db도 정의

gradient descent에 적용

$w_1 := w_1 - \alpha\, dw_1$

$w_2 := w_2 - \alpha\, dw_2$

$b := b - \alpha\, db$.

지금까지 1개의 단계의 기울기 강하를 적용한 것.

Vectorization

↓

이것 여러번 반복 multiple steps of gradient descent

Andrew Ng

계산에서의 2가지 관점.

2개의 for loop 사용 ① m training example → to compute average.

② all the features n개의 features

=> 알고리즘 비효율적 수행. 데이터 세트 처리능력.

↓

Vectorization → for loop 제거.

↑ speedy. \ 데이터 및 처리능력 증대시켰다.

deeplearning.ai

Basics of Neural Network Programming

Vectorization

└ 코드에서 for loop 제거

레이어 별 최적화에 따른 코딩 중요

↑ 중요!

What is vectorization?

$z = w^T x + b$

column vector.

$w = \begin{bmatrix} \vdots \\ \vdots \\ \vdots \end{bmatrix}$ $x = \begin{bmatrix} \vdots \\ \vdots \\ \vdots \end{bmatrix}$ $w \in \mathbb{R}^{n_x}$ $x \in \mathbb{R}^{n_x}$

Non-vectorized:

```
z = 0
for i in range(n-x):
    z += w[i] * x[i]
z += b
```

0으로 초기화 후 각각 더해 나가는 방식

1 ~ nx

slow!

Vectorized

```
z = np.dot(w,x) + b
```

$w^T x$

→ GPU
→ CPU
} SIMD - single instruction multiple data.

for문 <

parallelization instruction 이 있음

└ numpy가 이런 연산을 더 빠르게. 즉슨. GPU가 더 최적화된 것.

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

More vectorization examples

Neural network programming guideline

Whenever possible, avoid explicit for-loops.

↳ using built-in func.

for speed.

Neural network programming guideline

Whenever possible, avoid explicit for-loops.

$u = Av$ A matrix의 곱셈.

행렬곱의 정의 $u_i = \sum_i \sum_j A_{ij} * v_j$

```
u = np.zeros((n, 1))
for i ...
    for j ...
        u[i] += A[i][j] * v[j]
```

2개의 for 제거.

```
u = np.dot(A, v)
```

Vectors and matrix valued functions

Say you need to apply the exponential operation on every element of a matrix/vector.

$$v = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} \rightarrow u = \begin{bmatrix} e^{v_1} \\ e^{v_2} \\ \vdots \\ e^{v_n} \end{bmatrix}$$

vector 0으로 초기화

```
u = np.zeros((n,1))
for i in range(n):
    u[i]=math.exp(v[i])
```

```
import numpy as np
u = np.exp(v)
```

벡터의 산술연산 : 같은 차원에 있는 원소에 연산.

```
np.log(v)
np.abs(v)
np.maximum(v,0)
v**2
1/v
```

np.log(v) → element-wise log.

np.maximum(v,0) → element-wise maximum, v의 모든 원소에 대한 최대값을 0으로

v**2, 1/v → v과의 원소에 대한 element-wise의 적용.

Andrew Ng

for 쓰기전에 numpy 내장 기능 있는지 찾아보기

Logistic regression derivatives

$J = 0$, ~~$dw_1 = 0$, $dw_2 = 0$~~, $db = 0$

$dw = np.zeros((n_x, 1))$

n_x를 1차원 벡터로.

이게 훨씬 빠르다.

dw를 계산

for $i = 1$ to n:

$z^{(i)} = w^T x^{(i)} + b$

$a^{(i)} = \sigma(z^{(i)})$

$J \mathrel{+}= -[y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})]$

예제는 $dz^{(i)} = \frac{da}{dz}$

$dz^{(i)} = a^{(i)}(1 - a^{(i)})$

for $j = 1 \dots n_x$:

~~$dw_1 \mathrel{+}= x_1^{(i)} dz^{(i)}$~~

~~$dw_2 \mathrel{+}= x_2^{(i)} dz^{(i)}$~~

$n_x = 2$

$dw \mathrel{+}= x^{(i)} dz^{(i)}$

$db \mathrel{+}= dz^{(i)}$

$J = J/m$, ~~$dw_1 = dw_1/m$, $dw_2 = dw_2/m$~~, $db = db/m$

$dw /= m$.

deeplearning.ai

Basics of Neural Network Programming

Vectorizing Logistic Regression

to compute forward propagation =>

Vectorizing Logistic Regression

$$z^{(1)} = w^T x^{(1)} + b$$
$$a^{(1)} = \sigma(z^{(1)})$$

$$z^{(2)} = w^T x^{(2)} + b$$
$$a^{(2)} = \sigma(z^{(2)})$$

$$z^{(3)} = w^T x^{(3)} + b$$
$$a^{(3)} = \sigma(z^{(3)})$$

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

(n_x, m)

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

w^T (1 × n_x row vector) $\begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$

$$Z = \begin{bmatrix} z^{(1)} & z^{(2)} & \cdots & z^{(m)} \end{bmatrix} = w^T X + \begin{bmatrix} b & b & \cdots & b \end{bmatrix} = \begin{bmatrix} w^T x^{(1)} + b & w^T x^{(2)} + b & \cdots & w^T x^{(m)} + b \end{bmatrix}$$

$1 \times m$ $\quad$ $1 \times m$

```
Z = np.dot(w.T, X) + b
```

(1, 1) $\mathbb{R}$

"Broadcasting"

$$A = \begin{bmatrix} a^{(1)} & a^{(2)} & \cdots & a^{(m)} \end{bmatrix} = \sigma(Z)$$

Andrew Ng

programming에서 vector valued sigmoid

대문자 Z를 변수로 대문자 A 산출.

=> Vectorization을 사용하여 효율적으로 모든 activation을 산출하는 방법.

deeplearning.ai

Basics of Neural Network Programming

Vectorizing Logistic Regression’s Gradient Computation

z는 선형회귀식

b는 실수 파라미터 W는 벡터 파라미터.

Vectorizing Logistic Regression

$dz^{(1)} = a^{(1)} - y^{(1)}$ $\quad dz^{(2)} = a^{(2)} - y^{(2)}$ $\quad \dots$

$dZ = [dz^{(1)} \; dz^{(2)} \dots dz^{(m)}]$ $\leftarrow$ $(1 \times m)$

$A = [a^{(1)} \dots a^{(m)}]$, $\quad Y = [y^{(1)} \dots y^{(m)}]$

$\rightarrow dZ = A - Y = [a^{(1)} - y^{(1)} \quad a^{(2)} - y^{(2)} \quad \dots]$

$\rightarrow dw = 0$

$dw \mathrel{+}= x^{(1)} dz^{(1)}$

$dw \mathrel{+}= x^{(2)} dz^{(2)}$

$\vdots$

$dw /= m$

~~dw_1~~

~~dw_2~~

$db = 0$

$db \mathrel{+}= dz^{(1)}$

$db \mathrel{+}= dz^{(2)}$

$\vdots$

$db \mathrel{+}= dz^{(m)}$

$db /= m.$

$$db = \frac{1}{m} \sum_{i=1}^{m} dz^{(i)}$$

$$= \frac{1}{m} \, np.sum(dZ)$$

전치 행렬

$$dw = \frac{1}{m} X \, dZ^T$$

$$= \frac{1}{m} \begin{bmatrix} | & & | \\ x^{(1)} & \dots & x^{(m)} \\ | & & | \end{bmatrix} \begin{bmatrix} dz^{(1)} \\ \vdots \\ dz^{(m)} \end{bmatrix}$$

$$= \frac{1}{m} \left[x^{(1)} dz^{(1)} + \dots + x^{(m)} dz^{(m)} \right]$$

$n \times 1$

dw - parameter.

Andrew Ng

Implementing Logistic Regression

$J = 0,\ dw_1 = 0,\ dw_2 = 0,\ db = 0$

for i = 1 to m:

$z^{(i)} = w^T x^{(i)} + b$

$a^{(i)} = \sigma(z^{(i)})$

$J \mathrel{+}= -\left[y^{(i)} \log a^{(i)} + (1 - y^{(i)}) \log(1 - a^{(i)})\right]$

$dz^{(i)} = a^{(i)} - y^{(i)}$

$dw_1 \mathrel{+}= x_1^{(i)} dz^{(i)}$

$dw_2 \mathrel{+}= x_2^{(i)} dz^{(i)}$

$dw \mathrel{+}= x^{(i)} * dz^{(i)}$ vector value.

$db \mathrel{+}= dz^{(i)}$

$J = J/m,\ dw_1 = dw_1/m,\ dw_2 = dw_2/m$

$db = db/m$

for iter in range(1000):

$Z = w^T X + b$

$= np.dot(w.T, X) + b$

$A = \sigma(Z)$

$dZ = A - Y$

$dw = \frac{1}{m} X dZ^T$

$db = \frac{1}{m} np.sum(dZ)$

$w := w - \alpha dw$

$b := b - \alpha db$

Loop again for single iteration.

Loop again for α

deeplearning.ai

Basics of Neural Network Programming

Broadcasting in Python

Broadcasting example

Calories from Carbs, Proteins, Fats in 100g of different foods:

	Apples	Beef	Eggs	Potatoes
Carb	56.0	0.0	4.4	68.0
Protein	1.2	104.0	52.0	8.0
Fat	1.8	135.0	99.0	0.9

= A (3,4)

59cal

$\frac{56}{59} \approx 94.9\%$

Calculate % of calories from Carb, Protein, Fat. / Can you do this without explicit for-loop?

칼로리

세로로 값을 더해라, 1이면 가로로.

```
cal = A.sum(axis = 0)
percentage = 100*A/(cal.reshape(1,4))
```

얘는 1x4

모양 헷갈리면 그냥 써라. 비용 얼마 안든다.

(3,4) / (1,4)

Broadcasting example

3x4 matrix를 1x4 매트릭스로 나눔.
ㄴ A ㄴ cal

ㄴ파이썬 자체에서 연산하는 행렬에 발생
다중으로 확장해주는 기능
그래서 아까 + 상수 해도 다 더해진게 모든 요소에 (벡터)

$$\begin{bmatrix}1\\2\\3\\4\end{bmatrix} + \begin{bmatrix}100\\100\\100\\100\end{bmatrix} \;(\text{printed: } 100) = \begin{bmatrix}101\\102\\103\\104\end{bmatrix}$$

다중으로 4x1 벡터로 확장.
ㄴbroadcasting

$$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix} + \begin{bmatrix}100 & 200 & 300\\100 & 200 & 300\end{bmatrix} = \begin{bmatrix}101 & 202 & 303\\104 & 205 & 306\end{bmatrix}$$

(m, n) $(2,3)$ $(1, n) \rightsquigarrow (m, n)$ $(2,3)$

행은 m번 복사

$$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix} + \begin{bmatrix}100 & 100 & 100\\200 & 200 & 200\end{bmatrix} = \begin{bmatrix}101 & 102 & 103\\204 & 205 & 206\end{bmatrix}$$

(m, n) $(m, 1) \to (m, n)$

가로로 n번 복사

General Principle

신경망을 조건할 때 사용하는 주요 broadcasting 규칙.

행은 m번 복사

(m,n) $+ - * /$ $(1,n) \rightsquigarrow (m,n)$

matrix

열은 n번 복사

$(m,1) \rightsquigarrow (m,n)$

$(m,1)$ $+$ $\mathbb{R}$ (m,1) matrix가 나올 때까지 n번 복사

$\begin{bmatrix}1\\2\\3\end{bmatrix}$ $+$ 100 $=$ $\begin{bmatrix}101\\102\\103\end{bmatrix}$

$[1\ 2\ 3]$ $+$ 100 $=$ $[101\ \ 102\ \ 103]$

Matlab/Octave: bsxfun

similar as broadcasting.

deeplearning.ai

Basics of Neural Network Programming

A note on python/ numpy vectors

느슨함↑ but bug 발생 가능.

Python Demo

Python / numpy vectors

```
import numpy as np

a = np.random.randn(5)

a = np.random.randn((5,1))

a = np.random.randn((1,5))

assert(a.shape = (5,1))
```

a.shape(5,) "rank 1 array" } Don't use.

a.shape = (5,1). col vector

a.shape = (1,5) row vector.

└ 벡터의 dimension은 모르는 경우. (5,1) 벡터가 아님.

rank 1 array인 경우. a = a.reshape((5,1))

Explanation of Logistic regression cost function

(왜 이걸 써야하는지.

Logistic regression cost function

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

Interpret $\hat{y} = p(y=1|x)$ $\hat{y}$이 x에 대해 y가 1인 경우의 확률.

If $y=1$: $p(y|x) = \hat{y}$

If $y=0$: $p(y|x) = 1 - \hat{y}$ y가 0인 확률.

Logistic regression cost function

y는 반드시 0또는 1이어야 한다.

$p(y|x)$

If $y = 1$: $p(y|x) = \hat{y}$

If $y = 0$: $p(y|x) = 1 - \hat{y}$

둘을 한번에

$p(y|x) = \hat{y}^y (1-\hat{y})^{(1-y)}$

If $y=1$: $p(y|x) = \hat{y}^1 (1-\hat{y})^0 = \hat{y}$ ($(1-\hat{y})^0 = 1$)

If $y=0$: $p(y|x) = \hat{y}^0 (1-\hat{y})^{(1-0)} = 1 \times (1-\hat{y}) = 1-\hat{y}$ ($\hat{y}^0 = 1$)

$\uparrow \log p(y|x) = \log \hat{y}^y (1-\hat{y})^{(1-y)} = y \log \hat{y} + (1-y) \log(1-\hat{y})$

$= -\mathcal{L}(\hat{y}, y)$ ↓ Loss 함수의 마이너스 값과 같음. → loss는 최소화

log 함수는 값손실 없고 증가하는 함수 (목표에서로)

$\log(p(y|x))$를 최대화하면 optimizing $p(y|x)$과 비슷한 결과.

Andrew Ng 하고 싶은 것.

확률의 log를 최대화 하는 것과 같음.

확률 추정의 최대화 → 그렇게 만들어주는 parameter를 찾아야.

Cost on m examples

에러의 가능성은 곱?

[곱셈 형태로 표현.

$$\log p(\text{labels in training set}) = \log \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)}) \leftarrow$$

$$\log p(\dots) = \sum_{i=1}^{m} \log p(y^{(i)} \mid x^{(i)})$$

$$- \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

$$= -\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

최대 확률 측정법.

Maximum likelihood estimation ↖

Cost: (minimize)

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

=> $J(w,b)$를 최소화하여 로지스틱 회귀 분석 모델을 바탕으로 최대확률 측정법 진행.

Andrew Ng