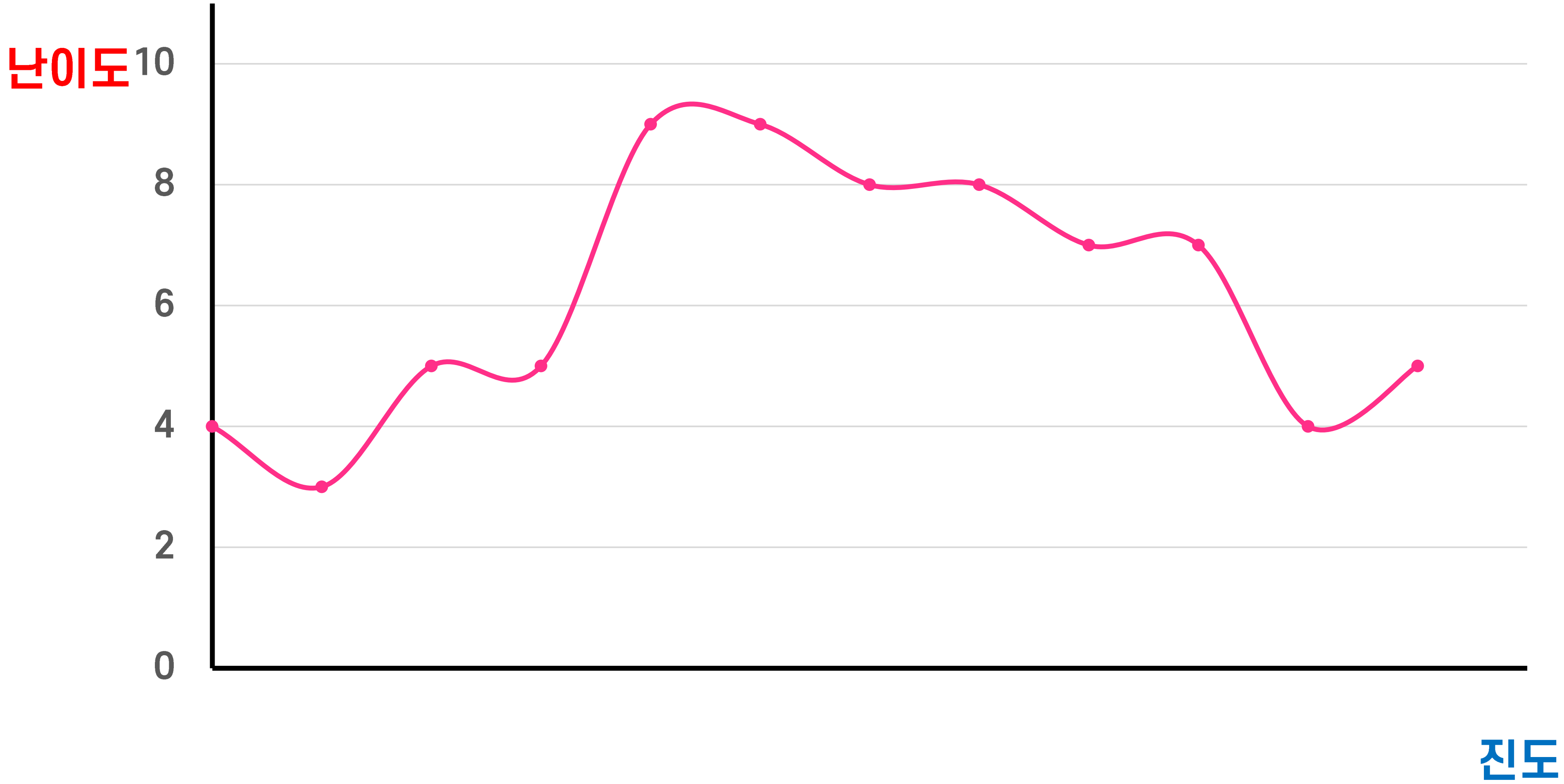
Optimization

14기 정규세션

TOBIG'S 13기 이지용

이번 수업 예상



지난 수업 간단한 리뷰

연속형 변수
Continuous Variable

선형 회귀
Linear Regression

이산형 변수
Discrete Variable

로지스틱 회귀
Logistic Regression

지난 수업 간단한 리뷰

모수Parameter를 추정하고 싶다! (베타β와 세타θ 등등)
-> Maximum likelihood EstimationMLE

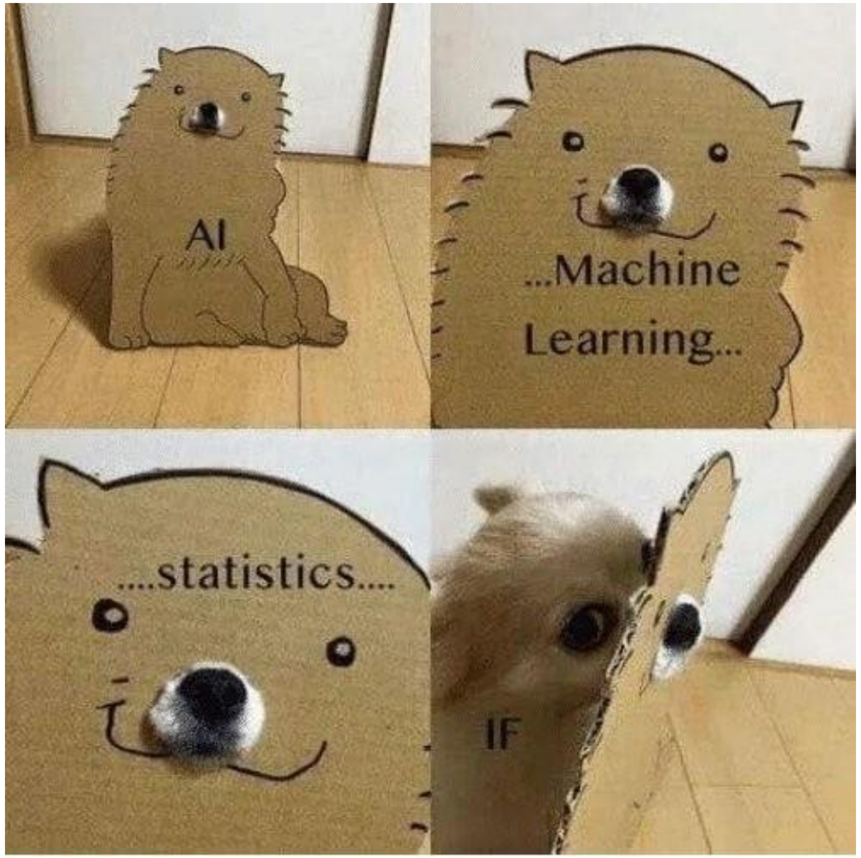
어떻게? 목적함수objective function를 **최적화**optimization하면서!
Mean Square ErrorMSE -> 최적값 계산 가능
Cross Entropy -> 최적값 계산 불가능 ㅠㅠ

Machine Learning

통계학?? 머신러닝??

컴퓨터가 잘하는 일은? -> 같은 일 반복하기

머신러닝을 머신러닝답게



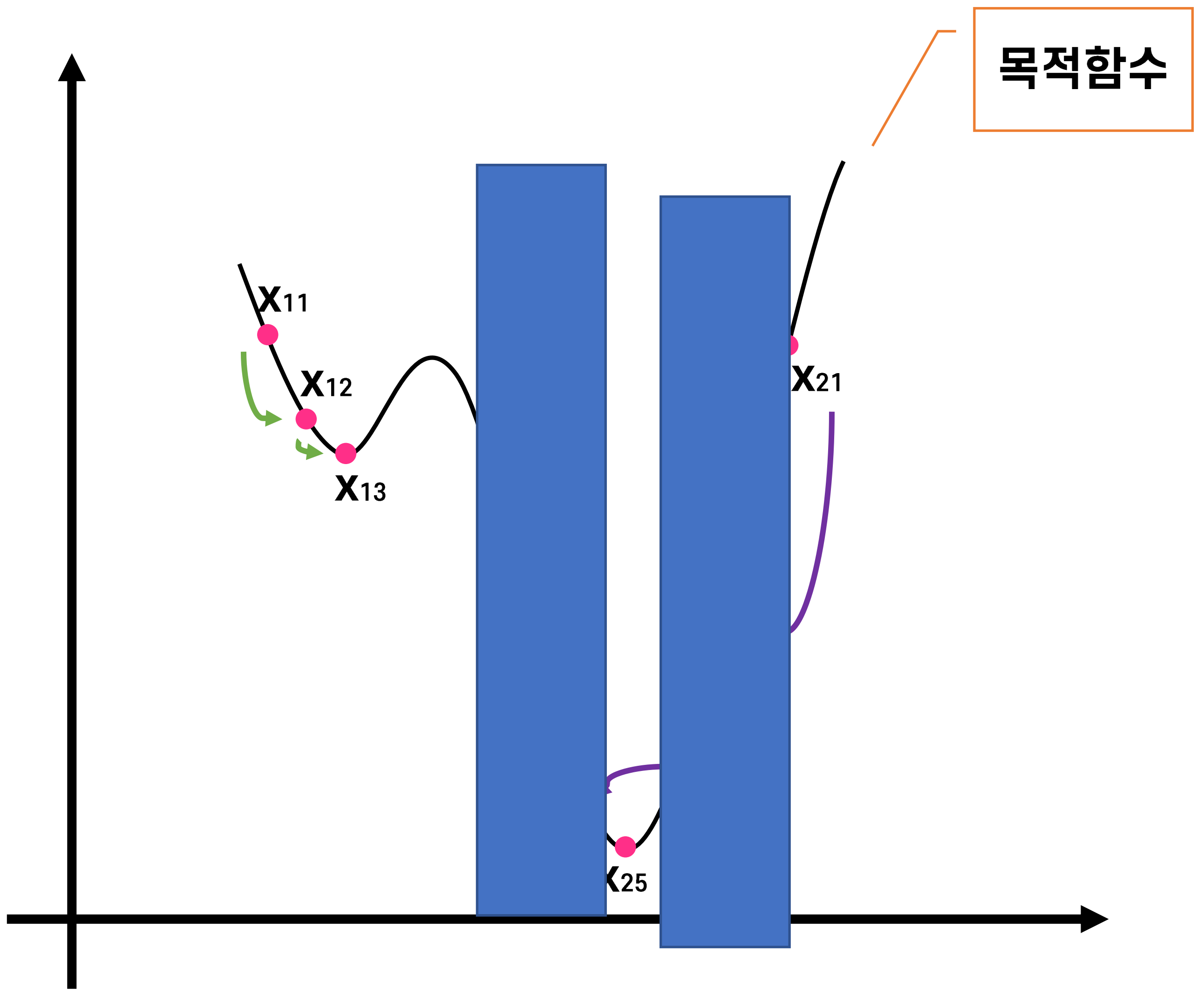
이것이 Gradient Descent다! – 개념편

목적함수를 최적화하는 모수를 찾을 때까지 계산을 반복하자

산에서 내려가기(But 정확한 지도가 없음)

경사길을 따라서 조금씩 내려가다 보면 평지가 나오지 않을까?

이것이 Gradient Descent다! – 개념편



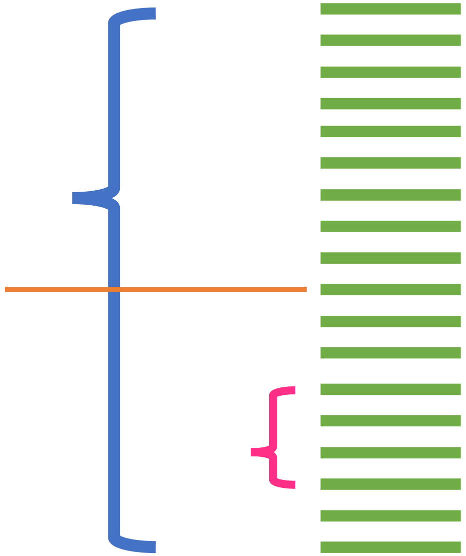
기울기를 계산하고 학습률을 곱해서 다음 탐색위치를 결정하자!

Gradient기울기 : 경사
Learning Rate학습률 : 보폭
Local Minimum극솟값 : 골짜기
Global Minimum최솟값 : 마을

Optimizer

Gradient Descent경사하강법

- **Batch Gradient DescentBGD**
  - 학습 한 번에 모든 데이터셋에 대해 기울기를 구한다
- **Stochastic Gradient DescentSGD**
  - 학습 한 번에 임의의 데이터에 대해서만 기울기를 구한다
- **Mini batch Gradient DescentMGD**
  - 학습 한 번에 데이터셋의 일부에 대해서만 기울기를 구한다
- **그 외 NAG, Adagrad, Adam…**



Newton Method

Lagrange Multiplier

들어가기에 앞서 간단한 수학공부

로그 함수Logarithm

$$\log ab = \log a + \log b$$

$$\log \frac{a}{b} = \log a - \log b$$

$$\log a^k = k \log a, \qquad \log \frac{1}{a} = -\log a$$

들어가기에 앞서 간단한 수학공부

미분공식Differentiation Rules

$$f'(x) = \frac{d}{dx}f(x) = \lim_{h\to 0}\frac{f(x+h)-f(x)}{h}$$

$$\frac{d}{dx}\frac{1}{f(x)} = \frac{f'(x)}{\{f(x)\}^2}$$

$$\frac{d}{dx}\log x = \frac{1}{x}, \qquad (e^x)' = e^x$$

들어가기에 앞서 간단한 수학공부

편미분Partial Derivative

다양한 변수 중 하나에 대해서만 미분을 하자!

$$\frac{\partial}{\partial x} f(x, y) = \lim_{h \to 0} \frac{f(x+h, y) - f(x, y)}{h}$$

들어가기에 앞서 간단한 수학공부

연쇄법칙Chain Rule

겉미분과 속미분

$$y = f(u), u = g(x),$$

$$\{f(g(x))\}' = \frac{d}{dx} f(g(x))$$

$$= \frac{d}{du} f(u) \frac{d}{dx} g(x) \quad \left(= \frac{dy}{du}\frac{du}{dx}\right)$$

$$= f'(u)g'(x)$$

$$= f'(g(x))g'(x)$$

$$\frac{d}{dx} e^{-2x} = \frac{d}{d(-2x)} e^{-2x} \frac{d}{dx}(-2x)$$

$$= -2e^{-2x}$$

들어가기에 앞서 간단한 수학공부

벡터Vector 표현

$$\boldsymbol{x} = \{x_1, x_2, x_3, x_4, \dots, x_n\}, \qquad \frac{\partial \boldsymbol{y}}{\partial x_i} = \{\frac{\partial y_1}{\partial x_i}, \frac{\partial y_2}{\partial x_i}, \frac{\partial y_3}{\partial x_i}, \dots\}$$

스칼라 곱Scalar product

벡터로부터 실수 스칼라를 얻는 연산

길이와 각도를 통해 다음과 같이 정의된다

$$\vec{\boldsymbol{a}} \cdot \vec{\boldsymbol{b}} = |\vec{\boldsymbol{a}}||\vec{\boldsymbol{b}}| \cos\alpha = a_1 b_1 + a_2 b_2$$

이것이 Gradient Descent다! - 수학편

TOBIG'S 13기 이지용

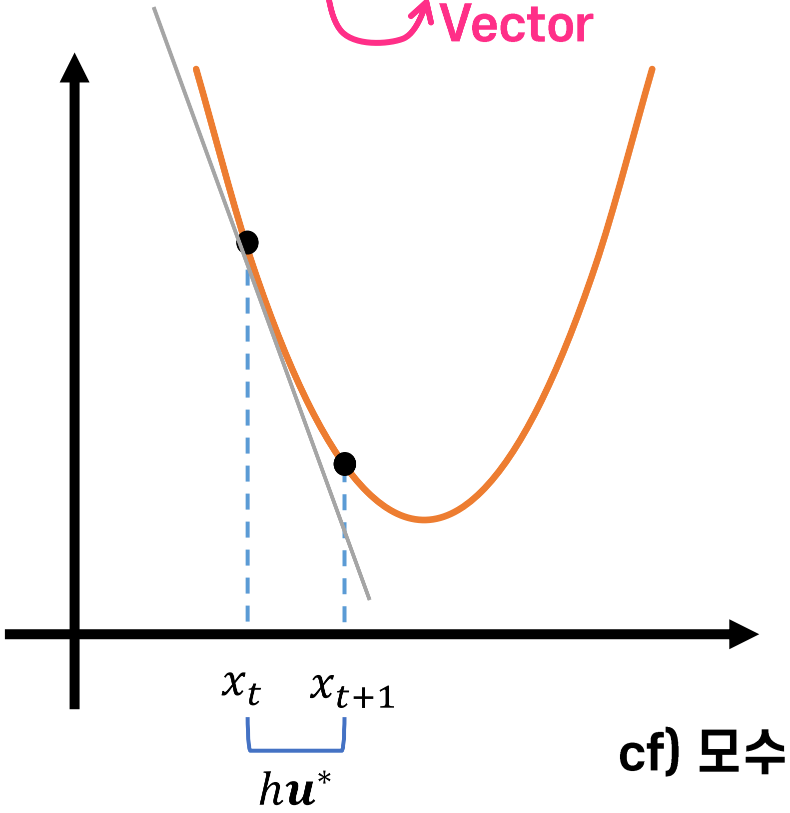
기울기와 반대방향

Step size

Gradient

$$x_{t+1} \leftarrow x_t + h\boldsymbol{u}^* = x_t - h\frac{f'(x_1)}{|f'(x_1)|}$$

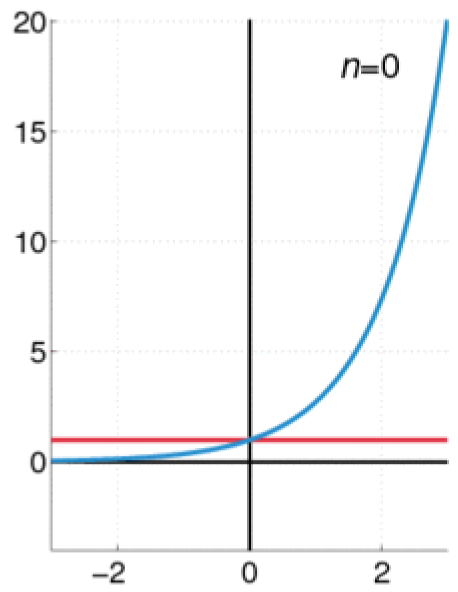
Unit Vector



cf) 모수가 n개면 n차원 그래프로 나타남

테일러 급수Taylor Series

-> 무한히 미분 가능한 함수를 무한급수로 표현
미분이 어려운 함수를 다항함수로 바꿔서
연산을 쉽게 하기 위해 사용



$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

$$e^x = \frac{x^0}{0!} + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots, (a = 0)$$

점근표기법(big-o)
이후의 표현식 합의 상한

$$\boxed{f(x)} = f(a) + \frac{f'(a)}{1!}(x-a) + \boxed{O\left(||x-a||^2\right)}$$

목적함수를 테일러 급수로 표현

목적함수가 무한히 미분 가능한 경우도
단순한 1차식으로 바꿔 표현 가능해진다

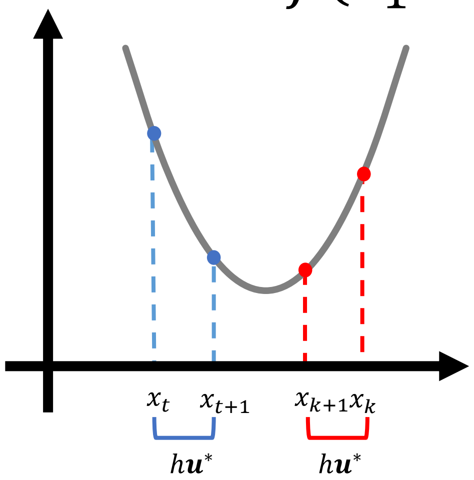
기울기의 반대방향으로 가는 이유

$$a = x_1, x = x_1 + h\boldsymbol{u}, |\boldsymbol{u}|=1$$

$$f(x_1 + h\boldsymbol{u}) = f(x_1) + \frac{f'(x_1)}{1!}(x_1 + h\boldsymbol{u} - x_1) + O\left(||x_1 + h\boldsymbol{u} - x_1||^2\right)$$

$$f(x_1 + h\boldsymbol{u}) = f(x_1) + hf'(x_1)\boldsymbol{u} + \boxed{h^2 O(1)}$$

h가 작으므로 무시 가능
(h= 0.01, 0.001)



$$\underline{f(x_1 + h\boldsymbol{u}) - f(x_1) \approx hf'(x_1)\boldsymbol{u}}$$

$$\boldsymbol{u}^* = argmin_{\boldsymbol{u}}\{f(x_1 + h\boldsymbol{u}) - f(x_1)\} = argmin_{\boldsymbol{u}} hf'(x_1)\boldsymbol{u} = -\frac{f'(x_1)}{|f'(x_1)|}$$

다음 번 학습의 결과값은
지금보다 작아야 한다.

$$\boxed{\because f(x_1 + h\boldsymbol{u}) \leq f(x_1),}$$

$$\boxed{\vec{\boldsymbol{a}} \cdot \vec{\boldsymbol{b}} = |\vec{\boldsymbol{a}}||\vec{\boldsymbol{b}}| \cos\alpha}$$

$-1 \leq \cos\alpha \leq 1,\ 0 \leq \alpha \leq 2\pi$
따라서 $f'(x_1)$ 와 $\boldsymbol{u}$ 는
반대방향일 때 내적이 최소

예시) 목적함수가 $f(x) = x^2$인 경우

$$x_{t+1} \leftarrow x_t + h\boldsymbol{u}^* = x_t - h\frac{f'(x_1)}{|f'(x_1)|}$$

1) $x_t = 2, h = 0.01, f'(x_t) = 4$

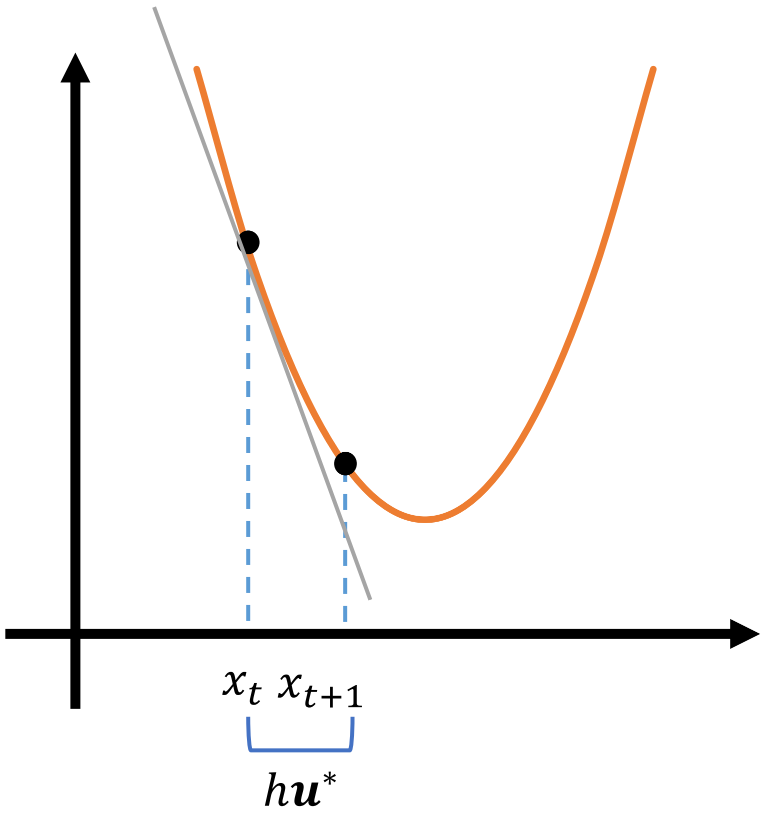
$$1.99 \leftarrow 2 + 0.01 \times (-1) = 2 - 0.01\frac{4}{|4|}$$

$$f(1.99) \leq f(2)$$

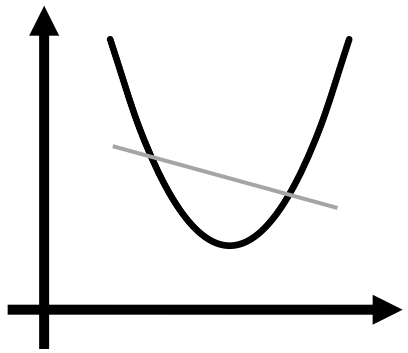
2) $x_t = -3, h = 0.1, f'(x_t) = -9$

$$-2.9 \leftarrow -3 + 0.1 \times (+1) = -3 - 0.1\frac{-9}{|-9|}$$

$$f(-2.9) \leq f(-3)$$



Convex Function



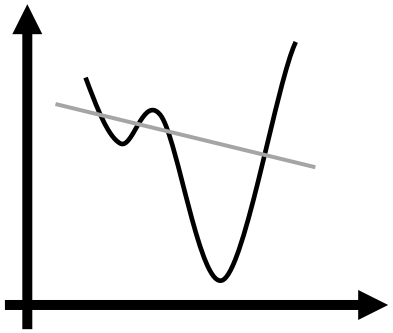
Convex볼록하다?

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y), \qquad (0 \leq t \leq 1)$$

-> Optimal Value가 하나다 = Global Minimum을 찾을 수 있다

Convex하지 않은 경우?

-> Local minimum, Saddle Point안장점 등 함정이 존재



Logistic Regression의 목적함수 Log Likelihood는 볼록하다

$$L(X) = \prod p(X_i)^{y_i}\left(1 - p(X_i)\right)^{(1-y_i)}$$

여전히 단조증가, 감소

$$l(X) = -\log L(X) = -\sum\left\{y_i \log p(X_i) + (1 - y_i)\log\left(1 - p(X_i)\right)\right\}$$

Descent를
진행하기 위해

$$p(X_i) = \frac{1}{1 + e^{-X_i\theta}} \qquad \left(= \phi(z) = \frac{1}{1 + e^{-z}}\right)$$

$$\hat{\theta} = argmin_{\theta}\, l(X)$$

Log Likelihood 미분하기 1

Logistic Regression의 목적함수를 미분한 값을 알아야
다음 학습에 필요한 모수를 업데이트할 수 있다

$$p(X_i) = \phi(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-X_i\theta}}$$

$$\begin{aligned}
l(y_i|X_i;\theta) &= -\sum\{y_i \log p(X_i) + (1-y_i)\log(1-p(X_i))\} \\
&= -\sum\{y_i \log p(X_i) - y_i \log(1-p(X_i)) + \log(1-p(X_i))\} \\
&= -\sum\left\{y_i \log\frac{p(X_i)}{1-p(X_i)} + \log(1-p(X_i))\right\} \\
&= -\sum\left\{y_i \log\frac{\frac{1}{1+e^{-X_i\theta}}}{1-\frac{1}{1+e^{-X_i\theta}}} + \log\left(1-\frac{1}{1+e^{-X_i\theta}}\right)\right\} \\
&= -\sum\left\{y_i \log\frac{\frac{1}{1+e^{-X_i\theta}}}{\frac{1+e^{-X_i\theta}-1}{1+e^{-X_i\theta}}} + \log\left(\frac{1+e^{-X_i\theta}-1}{1+e^{-X_i\theta}}\right)\right\} \\
&= -\sum\left\{y_i \log\frac{1}{e^{-X_i\theta}} \times \frac{e^{X_i\theta}}{e^{X_i\theta}} - \log\left(\frac{1+e^{-X_i\theta}}{e^{-X_i\theta}} \times \frac{e^{X_i\theta}}{e^{X_i\theta}}\right)\right\} \\
&= \underline{-\sum\{y_i X_i\theta - \log(1+e^{X_i\theta})\}}
\end{aligned}$$

Log Likelihood 미분하기 1

$$X_i\theta = (x_{i1}\theta_1 + x_{i2}\theta_2 + \cdots + x_{ij}\theta_j + \cdots)$$

$$\frac{\partial}{\partial\theta_j} l(y_i|X_i;\theta) = -\frac{\partial}{\partial\theta_j}\sum\{y_i X_i\theta - \log(1+e^{X_i\theta})\}$$

$$= -\sum(y_i x_{ij} - \frac{e^{X_i\theta}}{1+e^{X_i\theta}} x_{ij})$$

$$= -\sum(y_i x_{ij} - \frac{1}{1+e^{-X_i\theta}} x_{ij})$$

$$= -\sum(y_i - p_i)x_{ij}$$

Log Likelihood 미분하기 2

14기 정규세션
TOBIG'S 13기 이지용

함수를 합성함수로 표현하고 연쇄법칙을 이용해 미분하는 방법

$$l(\phi)' = -\frac{dl}{d\phi}\sum\{y_i \log\phi + (1-y_i)\log(1-\phi)\}$$

$$= -\sum\left(\frac{y_i}{\phi} - \frac{1-y_i}{1-\phi}\right) \quad = -\sum\frac{\boldsymbol{y_i-\phi}}{\boldsymbol{\phi(1-\phi)}}$$

$$\phi(z) = \frac{1}{1+e^{-z}},\ z = X\theta \qquad \left(\frac{d}{dx}\frac{1}{f(x)} = \frac{f'(x)}{\{f(x)\}^2}\right)$$

$$l(\theta) = -\sum\{y_i \log\phi + (1-y_i)\log(1-\phi)\}$$

$$\phi'(z) = \left(\frac{1}{1+e^{-z}}\right)'$$

$$= -\frac{(1+e^{-z})'}{(1+e^{-z})^2}$$

$$= \frac{e^{-z}}{(1+e^{-z})^2}$$

$$= \frac{1+e^{-z}-1}{(1+e^{-z})^2}$$

$$= \frac{1+e^{-z}}{(1+e^{-z})^2} - \frac{1}{(1+e^{-z})^2}$$

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

$$= \phi(z)(1-\phi(z))$$

Log Likelihood 미분하기 2

$$\frac{\partial}{\partial \theta_j} l\left(y_i \middle| X_i;\theta\right) = \frac{\partial l}{\partial \phi}\frac{\partial \phi}{\partial z}\frac{\partial z}{\partial \theta_j}$$

$$= -\sum \frac{y_i - \phi}{\phi(1-\phi)}\ \phi(z)\big(1 - \phi(z)\big)x_{ij}$$

$$= -\sum (y_i - \phi(z))x_{ij}$$

$$= -\sum (y_i - p_i)x_{ij}$$

$$l(\theta) = -\sum\{y_i \log\phi + (1-y_i)\log(1-\phi)\}$$

$$l'(\phi) = -\sum \frac{y_i-\phi}{\phi(1-\phi)}$$

$$\phi(z) = \frac{1}{1+e^{-z}}, \quad z = X\theta(= \{x_1\theta_1, x_2\theta_2, \dots\})$$

$$\phi'(z) = \phi(z)(1-\phi(z)) \qquad \frac{dz}{d\theta_j} = X_{ij}$$

Parameter Update

미분을 통해 방향을 결정했으니 모수들을 각각 업데이트를 해주자

$$x_{t+1} \leftarrow x_t - h\frac{f'(x_1)}{|f'(x_1)|}$$

$$\theta_j^{t+1} \leftarrow \theta_j^t - \frac{h}{C}\frac{\partial}{\partial\theta_j^t}l(\theta_j^t) \quad = \theta_j^t + \frac{h}{C}\sum(y_i - p_i)X_{ij}$$

H가 충분히 작으므로
구현때는 무시

Linear Regression & Gradient Descent

선형회귀에서도 Gradient Descent를 사용할 수 있다!
특히 feature가 많으면 행렬연산보다 효율적

$$x_{t+1} \leftarrow x_t - h\frac{f'(x_1)}{|f'(x_1)|}$$

$$\beta_k^{t+1} \leftarrow \beta_k^t - h\frac{\partial}{\partial \beta_k^t} MSE(\beta_k^t) \quad = \beta_k^t + \frac{h}{N}\sum 2(y_i - \hat{y}_i)X_{ik}$$

이것이 Gradient Descent다! - 구현편

14기 정규세션
TOBIG'S 13기 이지용

[X1, X2, X3, …, Xi, …] [xi1, xi2, xi3, …, xij, …] [y1, y2, y3, …, yi,…]

```
def Gradient Descent(독립변수X 데이터셋, 종속변수y 데이터셋, 학습률, 학습횟수) :
    k = 독립변수 개수, count = 0
    parameters = [임의값 k개]                          # 모수값을 임의로 초기화
    while count < 학습횟수 :                           # 학습횟수만큼 반복한다
        gradients = [0 k개]                            # 기울기를 저장할 리스트
        for X, y in [X list], [y list] :               # 데이터의 개수만큼 for loop 반복
            for j in range(len(parameters)) :          # 모수의 개수만큼 for loop 반복
                gradients[j] += Θj에 대한 기울기       # 목적함수를 편미분해서 더해준다.
        parameters -= gradients*학습률                 # 학습률 만큼 모수를 업데이트
        count += 1
    return parameters                                  # 학습이 완료되면 모수를 반환
```

함수 모듈화

반복적으로 자주 쓰이는 코드는 따로 함수로 만들어주자
-> 재활용, 수정 용이

Gradient Descent(Xset, yset, 학습률, 학습횟수)	{return Optimized Parameter}
Gradients(Xset, yset)	{return Gradients}
Gradient_ij(xij, y, yhat)	{return Gradient_ij}
Logistic(Xi)	{return p}
Step(Parameter, Gradients, 학습률)	{return New Parameter}
Loss(X, y)	{return Loss}

함수 모듈화

Gradient Descent

Gradients

```
def Gradient Descent(독립변수X 데이터셋, 종속변수y 데이터셋, 학습률, 학습횟수) :
    k = 독립변수 개수, count = 0
    parameters = [임의값 k개]                          # 모수값을 임의로 초기화
    while count < 학습횟수 :                           # 학습횟
        gradients = [0 k개]                            # 기울기
        for X, y in [X list], [y list] :               # 데이터의 개수만큼 for loop 반복
            for j in range(len(parameters)) :          # 모수의 개수만큼 for loop 반복
                gradients[j] += Θj 에 대한 기울기        # 목적함수를 편미분해서 더해준다.
        parameters -= gradients*학습률                  # 학습률 만큼 모수를 업데이트
        count += 1
    return parameters                                  # 학습이 완료되면 모수를 반환
```

Logistic

Gradient

Step

Stochastic Gradient Descent

학습 한 번에 임의의 데이터 하나에 대해서만 기울기를 구하자

```
def Gradient Descent(독립변수X 데이터셋, 종속변수y 데이터셋, 학습률, 학습횟수) :
    k = 독립변수 개수, count = 0
    parameters = [임의값 k개]                              # 모수값을 임의로 초기화
    while count < 학습횟수 :                                      # 학습횟수만큼 반복한다
        gradients = [0 k개]                                # 기울기를 저장할 리스트
        for j in range(len(parameters)) :                  # 모수의 개수만큼 for loop 반복
            ~~for X, y in [X list], [y list] :                # 데이터의 개수만큼 for loop 반복~~
            gradients[j]=R번째 데이터의 θj 에 대한 기울기# 목적함수를 미분해서 더해준다. R은 랜덤값
        parameters -= gradients*학습률                     # 학습률 만큼 모수를 업데이트
        count += 1
    return parameters                                      # 학습이 완료되면 모수를 반환
```

Mini Batch Gradient Descent

학습 한 번에 전체 데이터의 일부에 대해서만 기울기를 구하자

```
def Gradient Descent(독립변수X 데이터셋, 종속변수y 데이터셋, 학습률, 학습횟수, Batch Size) :
    k = 독립변수 개수, count = 0, Batch=[[B1], [B2],…]
    parameters = [임의값 k개]                              # 모수값을 임의로 초기화
    while count < 학습횟수 :                                # 학습횟수만큼 반복한다
        gradients = [0 k개]                                # 기울기를 저장할 리스트
        for X, y in [Batch X list], [Batch y list] :        # Batch 데이터의 개수만큼 for loop 반복
            for j in range(Batch[count%Batch Size]) :       # 모수의 개수만큼 for loop 반복
                gradients[j] += Θj 에 대한 기울기            # 목적함수를 미분해서 더해준다.
        parameters -= gradients*학습률                      # 학습률 만큼 모수를 업데이트
        count += 1
    return parameters                                      # 학습이 완료되면 모수를 반환
```

Additional Hyper Parameter

14기 정규세션
TOBIG'S 13기 이지용

실제 코드 구현에서는 부가적인 Hyper Parameter가 더 들어가야 한다

Step이 너무 작아서 더 이상의 학습이 무의미할 때
-> 학습을 멈추는 Tolerance 조건 추가

특정 변수에 집중하고 싶다
-> Weight 추가

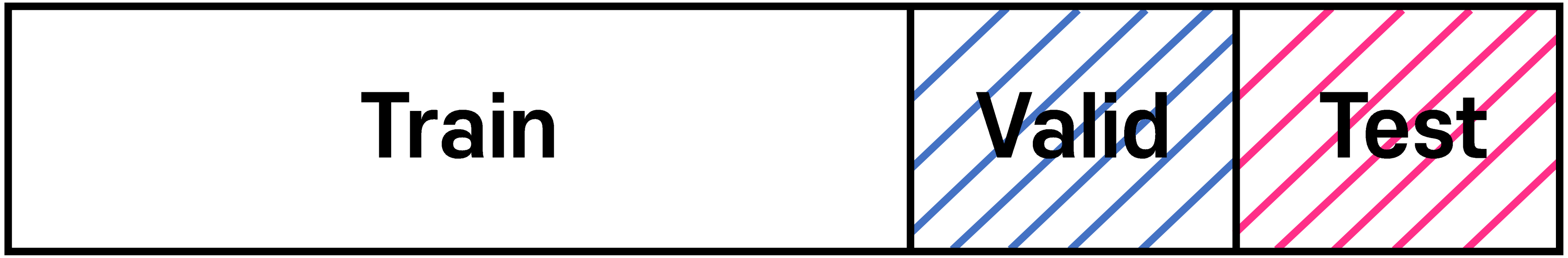
모델의 복잡도를 제한하고 싶다
-> Penalty 조건 추가

* Sklearn의 문서를 천천히 읽어보세요 (링크)

Train, Validation, Test

데이터를 미리 나눠 놓기
-> 모델이 적절히 학습됐는지 검증

Train Set : 학습에 사용
Validation Set : 학습 중에 지속적으로 모델 검증(Overfitting과적합 방지)
Test Set : 최종 평가에 사용



마무리

머신러닝에서 목적함수를 최적화하는 모수를 찾는 과정을 Optimization 이라 한다

Gradient Descent는 천천히 목적함수를 최소화하는 Optimization이다

Gradient Descent는 반복적인 학습을 통해 진행된다

Gradient Descent는 기울기의 반대방향으로 진행된다

Logistic Regression과 Linear Regression의 목적함수는 볼록하다

볼록함수는 최소의 모수를 찾을 수 있다

다음 모수는 현재 모수에서 (목적함수를 편미분한 값*학습률)을 빼준 값이다

Gradient Descent를 효율적으로 진행하기 위해 Stochastic, Mini Batch등의 방법을 사용한다

모듈화는 코드를 효율적으로 관리하기 위해 필요하다

머신러닝 학습시에는 데이터를 Train/Valid/Test로 분할한다

과제

optimization_assignment.ipynb를 완성해주세요

1) 필요한 함수 모듈들의 빈칸을 채워주세요
2) 모듈을 조립해 로지스틱 회귀의 모수를 최적화하는 Gradient Descent 함수를 완성해주세요
3) 주어진 데이터를 완성한 모듈에 적용해주세요(Train, Test 분할해서)
4) 회귀분석 시간에 배웠던 평가지표를 활용해 모델을 평가해주세요

* 이해한 만큼 코드주석, 부가설명 열심히 달아 주세요
* 어려운 게 당연한 수업이었으니 혼자 고민하지 말고 질문해주세요

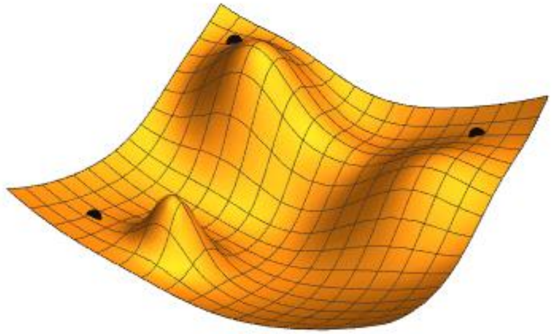
Q&A

추가자료

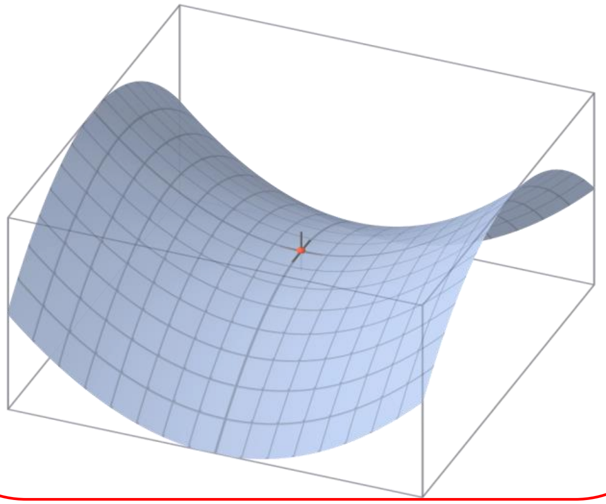
* 추가자료는 '그냥 이런 게 있다', '이런 게 궁금했는데 이렇다'
정도로만 알아 두시고 아직은 이해 못하셔도 됩니다.
이해를 돕기 위해서 첨부했으니 부담 갖지 말고 읽어주세요
수업범위를 벗어나거나, 생략해도 괜찮은 내용들입니다

Visualization

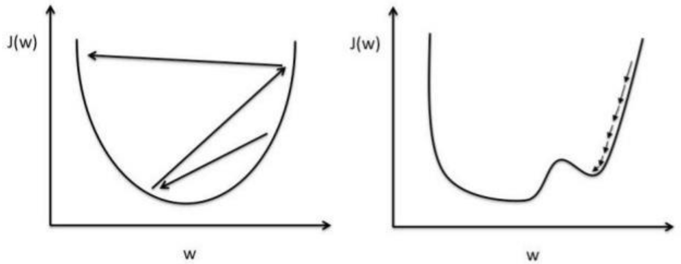
3차원 Gradient Descent



Saddle Point

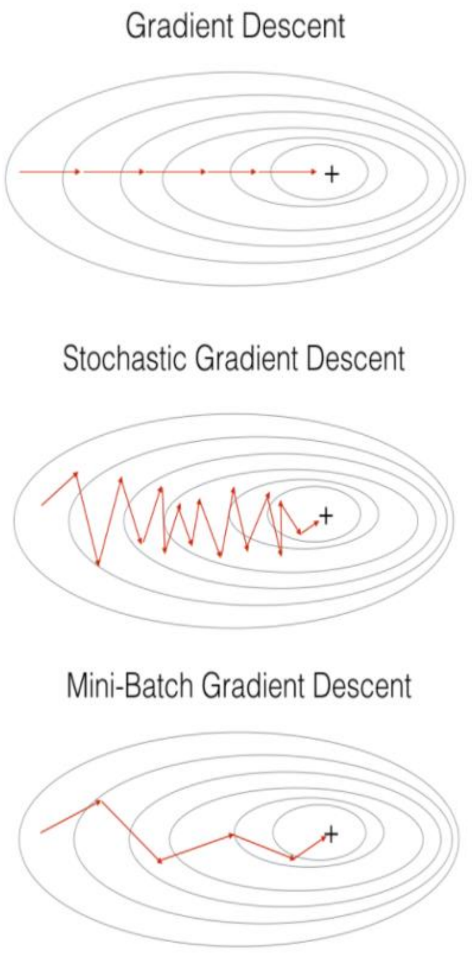


학습률이 너무 크거나 작을 때



Overshooting　Learn too slow

Optimizer별 수렴 방식



Ascent? Descent?

이번 수업에서는 Descent만 알아보았지만, 사실 Gradient Ascent는 Descent와 똑같은 개념입니다.

단지 방향만 바뀐 셈입니다. 최솟값을 찾는 문제가 아닌 최댓값을 찾는 문제에서는

기울기와 같은 방향으로 모수를 업데이트 해야 합니다. 만약 Log Cross Entropy에 음수를 붙이지 않고

그대로 갖다 쓴다면 Gradient Descent가 아닌 Gradient Ascent를 구현하는게 맞습니다

다만 일반적으로, 편의상 목적함수의 최솟값을 찾는 방향으로 구현하기 때문에

Descent 알고리즘이 더 유명합니다.

목적함수? 비용함수? 손실함수?

사실상 세 용어가 같은 의미로 쓰이지만, 굳이 구체적으로 의미의 차이를 따지자면 다음과 같습니다.

손실함수(Loss Function) : Single Data Set에 대한 예측 값이 실제 값과 얼마나 다른 지를 측정하는 함수

비용함수(Cost Function) : 손실함수를 일반화해서 전체 데이터셋에 대한 비용을 계산한 함수
(합친다든지, 평균을 낸다든지)

목적함수(Objective Function) : 더 일반화된 개념으로 최대화, 최소화하고자 하는 함수

로지스틱? 시그모이드?

로지스틱 함수처럼 S자 곡선를 그리는 함수들을 시그모이드 함수라고 부릅니다.
즉, 로지스틱 함수는 시그모이드의 한 종류인 셈입니다.

시그모이드 함수에는 로지스틱 함수 외에도 Hyperbolic tangent, Arctangent 등이 있습니다.

머신러닝에서는 시그모이드 함수를 활성화 함수로 많이 사용합니다.
그리고 활성화 함수 중에서 가장 유명한 Softmax 함수는 로지스틱 함수를 일반화한 버전입니다.

점근 표기법

알고리즘에 익숙하신 분들은 많이들 보셨을 텐데요,

점근 표기법은 다항식을 간단하게 표기할 때 많이 사용합니다.

간단하고 추상적인 방식으로 설명하자면,

Big O Notation($f(x) = O(n)$)은 특정 함수의 상한을,

Big Theta Notation($f(x) = \Theta(n)$)은 특정 함수 상한/하한을,

Big Omega Notation($f(x) = \Omega(n)$)은 특정 함수의 하한을 표현합니다.

주로 다항식의 가장 높은 차수를 참조해서

$O(1)$, $O(n)$, $O(\log n)$ 등으로 많이 표현한다고만 알고 있어도 됩니다.

보다 엄밀하고 구체적인 해석학적 의미는 검색해보세요!

Cross Entropy Convexity 증명

* 어려우니까 그냥 넘어가세요

Cross Entropy의 Convexity를 증명하기 위해서도 Taylor Series가 사용됩니다
t,1-t를 변수로 넣은 함수를 2차함수로 근사한 다음 다변수 함수의 특성을 적용하기 위해
이계도함수의 행렬인 Hessian 행렬을 사용하고, 도출한 Hessian 행렬의 고윳값에 따라
극소, 극대, 안장점을 판정할 수 있습니다(2차함수는 극소=최소)
(Hessian 행렬은 선형대수학에 나옵니다)

보다 자세한 내용은 https://taeoh-kim.github.io/blog/crossent/를 참고하세요

Neural Net 예고

오늘 배웠던 Gradient Descent는 하나의 Loss function에만 적용되었지만,
Neural Network에서는 수많은 Parameter를 최적화하기 위해
각각의 함수에 대해 Gradient Descent를 진행합니다.
때문에 연산량이 아주 많고, 이를 효율적으로 처리하기 위한 기법들을 사용합니다.
그리고 Chain Rule은 이 연산에 핵심을 담당합니다.

다시 말해, Gradient Descent와 Derivative, Chain Rule, Vector, Matrix에
친숙해지면 NN을 더 잘 이해할 수 있습니다.

그럼 6주차에 뵙겠습니다!

참고자료

Tobig’s 12기 이유진, 11기 이영전 강의자료

http://sanghyukchun.github.io/63/

https://hwiyong.tistory.com/7

https://wikidocs.net/17206

https://taeoh-kim.github.io/blog/crossent/

https://ratsgo.github.io/machine%20learning/2017/07/02/logistic/

https://www.geeksforgeeks.org/ml-mini-batch-gradient-descent-with-python/

https://rfriend.tistory.com/188

https://light-tree.tistory.com/133

카이스트 문일철 교수 유튜브 강의
https://www.youtube.com/watch?v=coTT9X_ovtk&list=PLbhbGI_ppZISMV4tAWHlytBqNq1-lb8bz&index=18

The hundred-page machine learning book

처음 배우는 딥러닝 수학

파이썬 머신러닝