15기 정규세션

ToBig's 14기 김상현

Optimization

Contents

Unit 01 | 최적화 & 머신러닝

✓ 최적화

특정의 집합 위에서 정의된 실수값, 함수, 정수에 대해 그 값이 최대나 최소가 되는 상태를 해석하는 문제

✓ 머신러닝

기계가 일일이 코드로 명시하지 않은 동작을 데이터로부터 학습하여 실행할 수 있도록 하는 알고리즘을 개발하는 연구 분야

Unit 01 | 최적화 & 머신러닝

최적화

Unit 01 | 최적화 & 머신러닝

최적화

머신러닝은 최적화를 통해 최적의 모수parameter를 찾는다

Unit 01 | 최적화 & 머신러닝

Maximum likelihood Estimation(MLE)을 통해 모수(θ)를 추정

MLE에서 나온 목적함수(objective function)를 **최적화**하면서 모수 조합을 찾아보자

Unit 01 | 최적화 & 머신러닝

✓Likelihood

Linear regression	Logistic regression
$L(Y_i\|X_i;\theta) = \prod^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(y_i - \theta^T X_i)^2}{2\sigma^2})$ 최소제곱추정량(LSE)와 동일 ⇒ Closed form solution(정규 방정식)	$L(Y_i\|X_i;\theta) = \prod^{m} p(X_i)^{y_i}(1 - p(X_i))^{(1-y_i)}$ $p(X_i) = \frac{1}{1 + e^{-X_i\theta}}$ ⇒ Open form solution ∵비선형 함수

Unit 01 | 최적화 & 머신러닝

Open form solution의 최적화는 수치해석적 방법을 이용한다.

cf) closed form solution의 최적화도 수치해석적 방법 이용 가능

Unit 01 | 최적화 & 머신러닝

Open form solution의 최적화는 수치해석적 방법을 이용한다.

cf) closed form solution의 최적화는 수치해석적 방법 이용 가능

경사하강법(Gradient Descent)

Contents

Unit 02 | 경사하강법

✓즐거운 수학시간!

로그 함수Logarithm

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

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

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

✓즐거운 수학시간!

미분공식Differentiation Rules

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

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

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

Unit 02 | 경사하강법

✓즐거운 수학시간!

편미분Partial Derivative

다양한 변수 중 하나에 대해서만 미분을 하자!

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

Unit 02 | 경사하강법

✓즐거운 수학시간!

연쇄법칙Chain Rule

겉미분과 속미분

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

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

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

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

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

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

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

Unit 02 | 경사하강법

✓즐거운 수학시간!

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

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

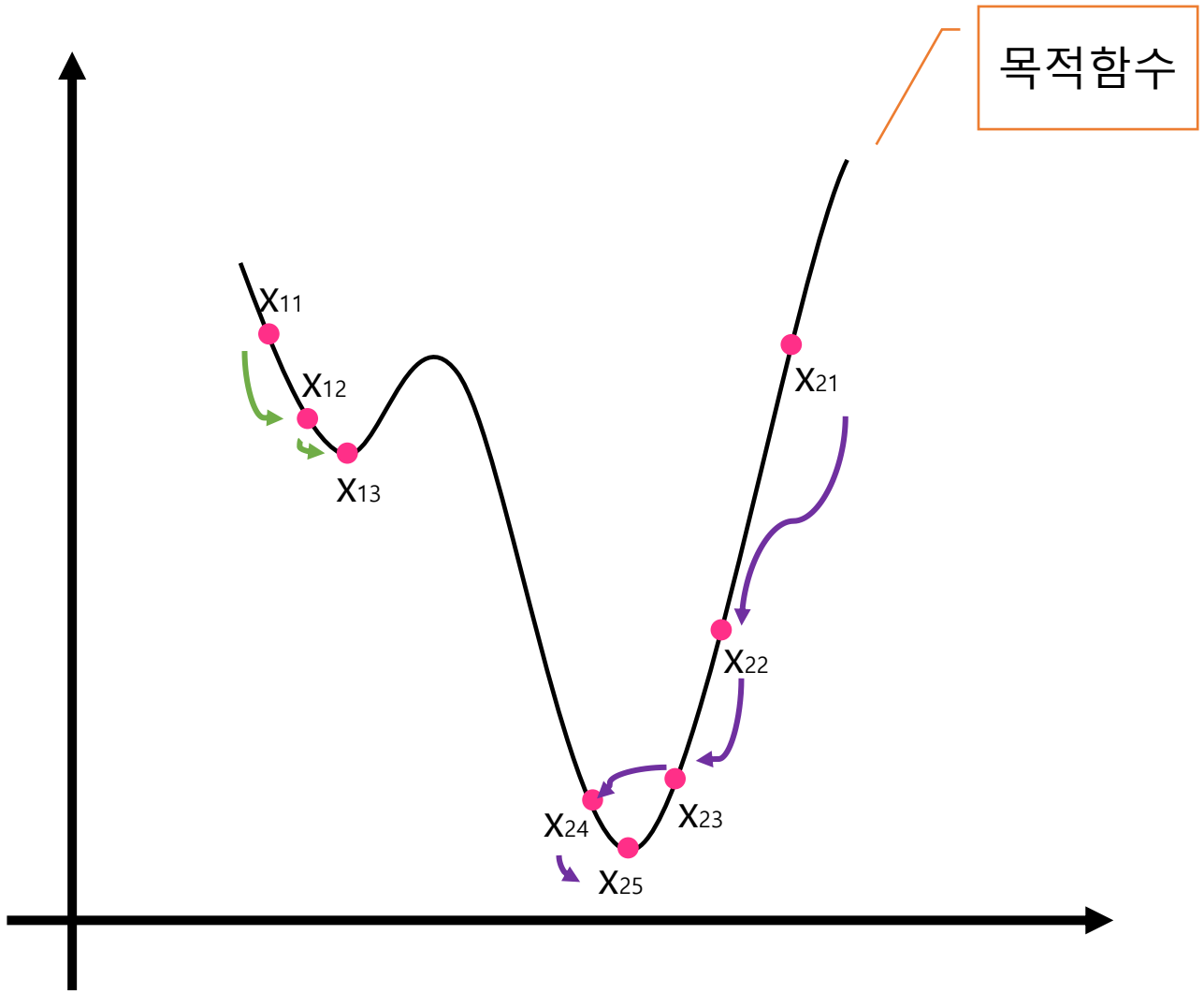
Unit 02 | 경사하강법

✓경사 하강법 비유

생존의 달인 대위 '인성 문제있어?'는 앞이 보이지 않는 안개가 낀 산을 내려가려 한다.

앞이 보이지 않는 상황이므로 모든 방향으로 산을 더듬어가며 산의 높이가 가장 낮아지는 방향으로 한 발씩 내딛는다.

Unit 02 | 경사하강법



기울기를 계산하고 학습률을 곱해서 다음 탐색위치를 결정하자

Gradient기울기 : 경사

Learning Rate학습률 : 보폭

Local Minimum극솟값 : 골짜기

Global Minimum최솟값 : 마을

✓경사 하강 알고리즘

1. 초기값 x_0 설정
2. $x_{t+1} \leftarrow x_t - \alpha \nabla f(x_t)$ 반복
3. 2번을 적당한 횟수로 반복하거나 수렴 조건을 이용해 최적의 x_t를 찾는다.

해당 알고리즘을 반복하면 극솟값 또는 최솟값으로 수렴한다.

cf) 극솟값과 최솟값 수렴 여부는 함수 f의 형태와 초기값 x_t에 따라 다르다.

Unit 02 | 경사하강법

✓경사 하강 알고리즘

1. 초기값 x_0 설정
2. $x_{t+1} \leftarrow x_t - \alpha \nabla f(x_t)$ 반복
3. 2번을 적당한 횟수로 반복하거나 수렴 조건을 이용해 최적의 x_t를 찾는다.

위치 최신화를 2번과 같이 하는 이유?

해당 알고리즘을 반복하면 극솟값 또는 최솟값으로 수렴한다.

cf) 극솟값과 최솟값 수렴 여부는 함수 f의 형태와 초기값 x_t에 따라 다르다.

Unit 02 | 경사하강법

$$x_{t+1} \leftarrow x_t - \alpha \nabla f(x_t)$$

α: learning rate
∇f: gradient

위와 같은 방법으로 위치를 최신화 할 때
기울기가 가장 크게 감소하는 방향으로 움직인다.

Point: 기울기 벡터의 반대 방향 = 기울기가 가장 크게 감소하는 방향

Unit 02 | 경사하강법

✓기울기의 반대방향?

✓정의 1.

다음의 극한이 존재할 때

$$D_u f(x_0, y_0) = \lim_{h \to 0} \frac{f(x_0 + ha, y_0 + hb) - f(x_0, y_0)}{h}$$

이 극한을 $\boldsymbol{u} =< \boldsymbol{a}, \boldsymbol{b} >$방향에 대한 점$(x_0, y_0)$에서 f의 **방향도함수**라고 한다.

$\boldsymbol{u}$:단위벡터

✓정리 1.

이변수함수 f가 x와 y의 미분가능한 함수이면, f는 모든 단위벡터 $\boldsymbol{u} =< \boldsymbol{a}, \boldsymbol{b} >$ 방향으로의 방향도함수가 존재하고

$$D_u f(x, y) = f_x(x, y)a + f_y(x, y)b$$

이다.

방향도함수: 함수 $\boldsymbol{f}$의 $\boldsymbol{u}$ 방향에 대한 변화율

Unit 02 | 경사하강법

✓기울기의 반대방향?

✓정의 2.

f가 두 변수 x와 y의 함수이면, f의 기울기 벡터는 벡터함수 ∇f이고 다음과 같이 정의된다.

$$\nabla f(x,y) = < f_x(x,y), f_y(x,y) >$$

✓식 1.

$$D_u f(x,y) = \nabla f(x,y) \cdot \boldsymbol{u}$$

Unit 02 | 경사하강법

✓기울기의 반대방향?

✓정리 2.

f가 이변수의 미분가능한 함수라고 하자. 그러면 방향도함수 $D_u f(\boldsymbol{x})$의 최댓값은 $|\nabla f(\boldsymbol{x})|$이고, 이것은 기울기 벡터 $\nabla f(\boldsymbol{x})$ 와 벡터 $\boldsymbol{u}$의 방향이 일치할 때 나타난다.

$\boldsymbol{x} =< x, y >$

증명

식 1로부터 다음을 얻는다.

$$D_u f = \nabla f \cdot \boldsymbol{u} = |\nabla f||\boldsymbol{u}| \cos\theta = |\nabla f| \cos\theta$$

이때 θ는 ∇f와 $\boldsymbol{u}$사이의 각이다. $\cos\theta$ 의 최댓값이 1이고, $\theta = 0$ 일 때 나타난다. 그러므로 $D_u f$의 최댓값은 $|\nabla f|$ 이고, $\theta = 0$ 일 때이다. 즉, $|\nabla f|$와 $\boldsymbol{u}$의 방향이 일치할 때이다.

Unit 02 | 경사하강법

✓기울기의 반대방향?

정리 2에 의해

$D_u f(\boldsymbol{x})$의 최솟값은 기울기 벡터 $\nabla f(\boldsymbol{x})$ 와 벡터 $\boldsymbol{u}$의 방향이 반대일때 나타난다.

$$-1 \leq \cos\theta \leq 1, 0 \leq \theta \leq 2\pi$$

Unit 02 | 경사하강법

✓기울기의 반대방향?

기울기 벡터의 반대 방향 = 기울기가 가장 크게 감소하는 방향

정리 2에 의해

$D_u f(\boldsymbol{x})$의 최솟값은 기울기 벡터 $\nabla f(\boldsymbol{x})$ 와 벡터 $\boldsymbol{u}$의 방향이 반대일때 나타난다.

$$-1 \leq \cos\theta \leq 1, 0 \leq \theta \leq 2\pi$$

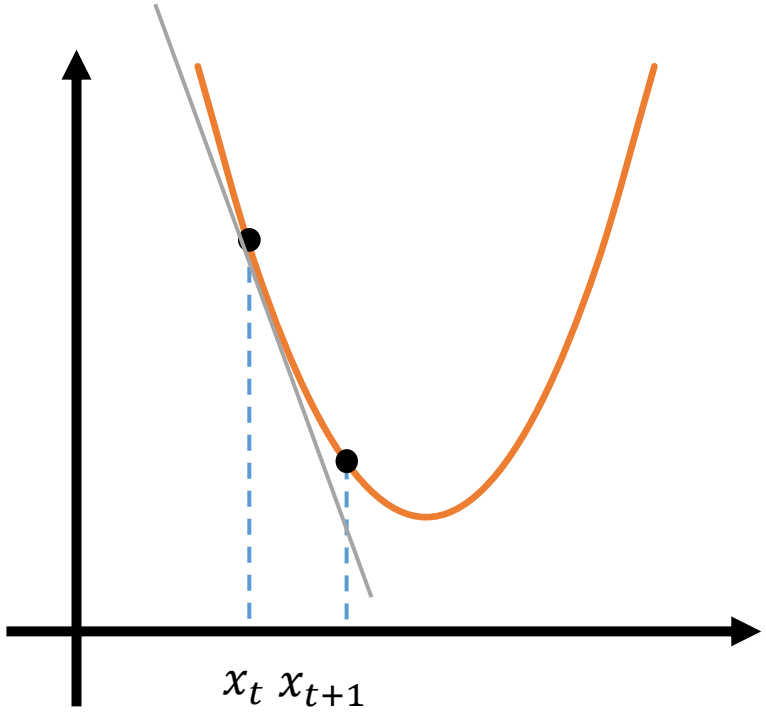
Unit 02 | 경사하강법

✓예시 $f(x) = x^2$

$$x_{t+1} \leftarrow x_t - \alpha \nabla f(x_t)$$

1) $x_t = 2, \alpha = 0.01, f'(x_t) = 4$
$1.96 \leftarrow 2 - 0.01 * 4 = 2 - 0.01 * 4$
$f(1.96) \leq f(2)$

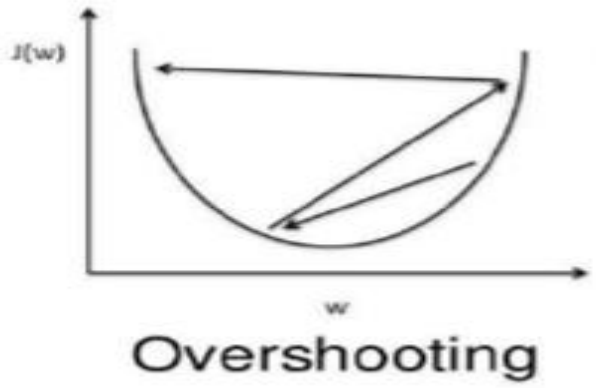
2) $x_t = -3, \alpha = 0.1, f'(x_t) = -6$
$-2.4 \leftarrow -3 - 0.1 * (-6)$
$f(-2.4) \leq f(-3)$



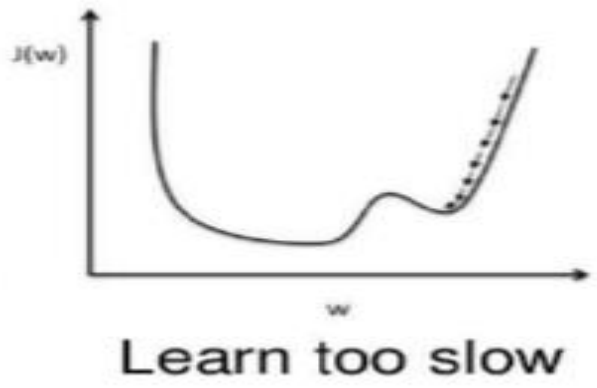
Unit 02 | 경사하강법

✓학습률(learning rate) & 초기치(initial point)

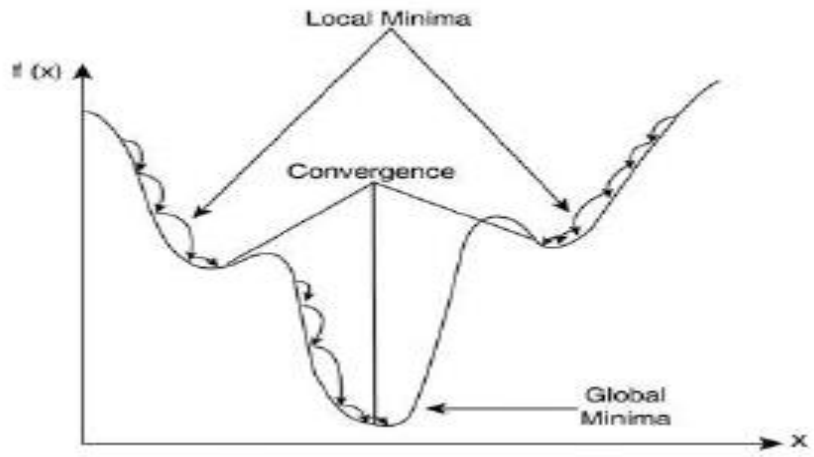
- 학습률이 너무 작을 경우 학습 시간이 오래 걸리고, 극솟값에 수렴함 수 있다.
- 학습률이 너무 클 경우 최솟값을 가로질러 반대편으로 건너뛰어 최솟값에서 멀어질 수 있다.
- 초기치에 따라 수렴 지점이 달라진다.(극솟값 or 최솟값)



Overshooting



Learn too slow



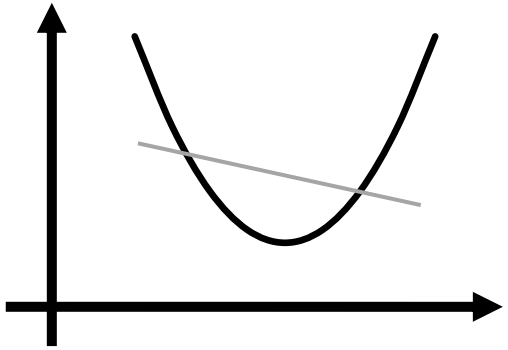
Unit 02 | 경사하강법

✓Convex function

함수 $f: I \rightarrow R$가 모든 $0 < \lambda < 1, x \in I, y \in I$ 에 대해

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y)$$

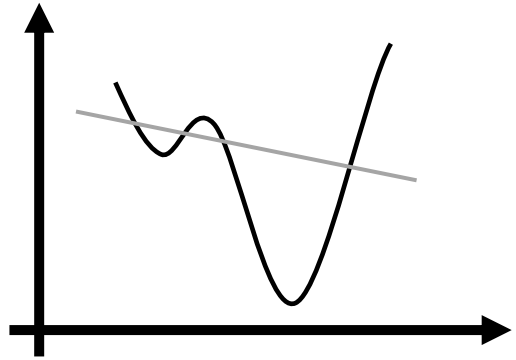
를 만족시키면 함수 f를 볼록(convex)이라고 한다.



Convex

함수 $f: (a, b) \rightarrow R$가 두 번 미분가능하면

f가 (a, b)에서 볼록이다. $\leftrightarrow$ (a, b)에서 $f'' \geq 0$이다.



Non-convex

Global Minimum이 존재하는 Convex function은 수렴정리에 의해 경사하강법을 이용해 Global Minimum을 찾을 수 있다.

convex함수 최적화 수렴:https://wikidocs.net/18086

Unit 02 | 경사하강법

✓Logistic Regression & Gradient Descent

Logistic Regression의 목적함수 Negative Log Likelihood는 볼록하다.

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

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

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

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

BCE convex증명: https://towardsdatascience.com/binary-cross-entropy-and-logistic-regression-bf7098e75559

Unit 02 | 경사하강법

✓Logistic Regression & Gradient Descent

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

$$l(y_i|X_i;\theta) = -\sum\{y_i \log p(X_i) + (1-y_i)\log(1-p(X_i))\}$$

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

$$= -\sum\left\{y_i \log \frac{1}{e^{-X_i\theta}} - \log\left(\frac{1+e^{-X_i\theta}}{e^{-X_i\theta}}\right)\right\}$$

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

Unit 02 | 경사하강법

✓Logistic Regression & Gradient Descent

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

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

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

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

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

Unit 02 | 경사하강법

✓Logistic Regression & Gradient Descent

미분을 통해 방향을 결정했으니 모수들을 각각 업데이트를 해주자

입력 데이터(batch)의 개수를 고려

$$x_{t+1} \leftarrow x_t - \alpha \nabla f(x_t)$$

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

Unit 02 | 경사하강법

✓Linear Regression & Gradient Descent

Feature가 많은 경우 행렬 연산을 수행하는 것보다 효율적이다.

$$MSE = \frac{1}{2}\Sigma(y_i - \theta^T X_i)^2$$

$$x_{t+1} \leftarrow x_t - \alpha\nabla f(x_t)$$

$$\beta_k^{t+1} \leftarrow \beta_k^t - \alpha\frac{1}{N}\frac{\partial}{\partial\beta_k^t}MSE(\beta_k^t) = \beta_k^t + \alpha\frac{1}{N}\sum(y_i - \theta^T X_i)X_{ik}$$

Contents

Unit 03 | 경사하강법과 배치학습 구현

✓Batch?

한 번 기울기를 계산하여 파라미터를 업데이트하는 작업, 또는 그 작업을 위해 사용하는 데이터 셋을 의미한다. 이때 데이터 셋의 크기(데이터 수)를 배치 크기(batch size)라고 한다.

Unit 03 | 경사하강법과 배치학습 구현

✓배치 학습

-**Batch Gradient Descent**BGD

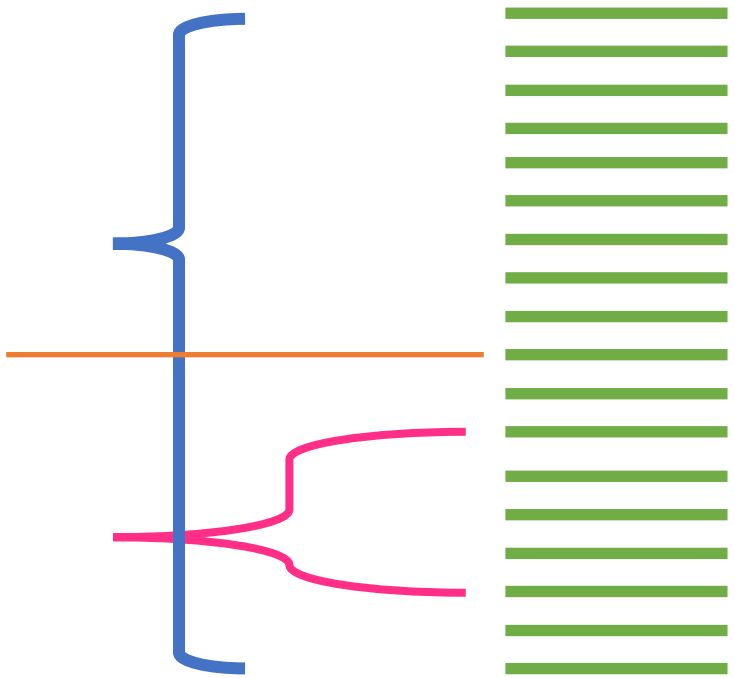
-학습 한 번에 모든 데이터셋에 대해 기울기를 구한다

-**Stochastic Gradient Descent**SGD

-학습 한 번에 임의의 데이터에 대해서만 기울기를 구한다

-**Mini batch Gradient Descent**MGD

-학습 한 번에 데이터셋의 일부에 대해서만 기울기를 구한다



QUIZ! 각각의 batch size는 어떻게 될까?

Unit 03 | 경사하강법과 배치학습 구현

✓배치 학습

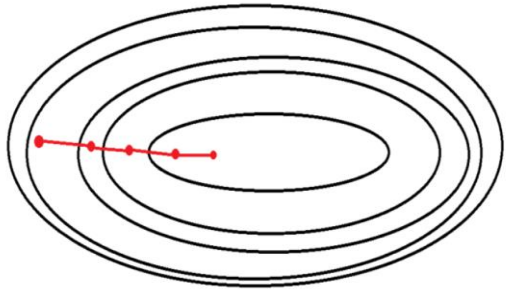
	BGD	SGD
장점	-안정적인 수렴 -병렬 처리 가능	-일부 문제에서 빠른 최적화 -지역 최적화 회피
단점	-메모리 문제 -지역 최적화 문제	-병렬 처리 불가 -안정적으로 수렴 어려움

→ BGD와 SGD의 장단점을 합친 MGD

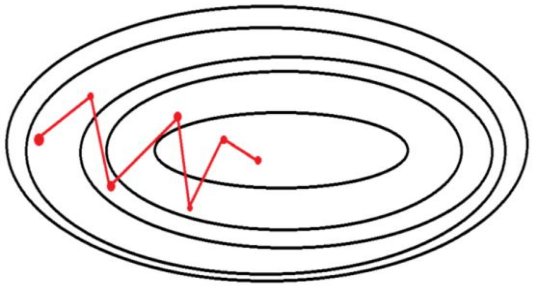
Unit 03 | 경사하강법과 배치학습 구현

✓배치 학습

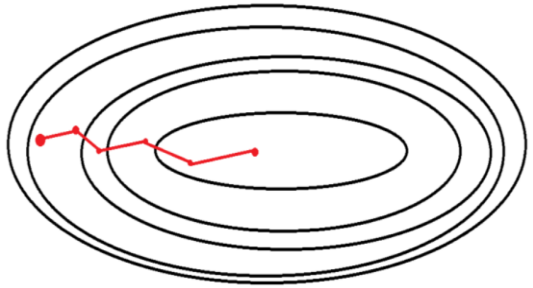
수렴과정 시각화



BGD



SGD



MGD

Unit 03 | 경사하강법과 배치학습 구현

✓구현

```
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
    for epoch in range(반복횟수) :                     # 반복횟수만큼 반복한다
        for X, y in [Batch X list], [Batch y list] :   # Batch 데이터의 개수만큼 for loop 반복
            gradients = [0 k개]                        # 기울기를 저장할 리스트
            for i in range(batch_size):
                gradient_i = 데이터 i의 gradient        #각각 데이터 i에 대해 목적함수를 미분
                gradients += gradient_i                # 기울기 리스트에 더해준다.
            parameters -= gradients*학습률             # 학습률 만큼 모수를 업데이트

    return parameters                                  # 학습이 완료되면 모수를 반환
```

Unit 03 | 경사하강법과 배치학습 구현

✓구현

자주 사용하는 코드는 함수로 정리!

Gradient Descent(Xset, yset, 학습률, 반복횟수)	**{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}**

Unit 03 | 경사하강법과 배치학습 구현

Gradient Descent

✓구현

Gradients

```
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
    for epoch in range(반복횟수) :                      # 반복한다
        for X, y in [Batch X list], [Batch y list] :   # 개수만큼 for loop 반복
            gradients = [0 k개]                        # 기울기를 저장할 리스트
            for i in range(batch_size):
                gradient_i = 데이터 i의 gradient       #각각 데이터 i에 대해 목적함수를 미분
                gradients += gradient_i                # 기울기 리스트에 더해준다.
        parameters -= gradients*학습률                 # 학습률 만큼 모수를 업데이트

    return parameters                                  # 학습이 완료되면 모수를 반환
```

Logistic

Gradient

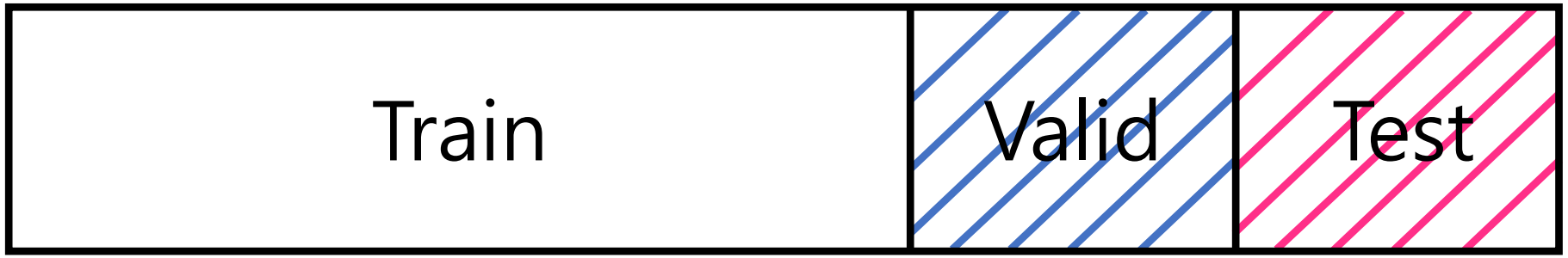
Step

Unit 03 | 경사하강법과 배치학습 구현

✓Train, Validation, Test

데이터를 미리 나눠 놓기
-> 모델이 적절히 학습됐는지 검증

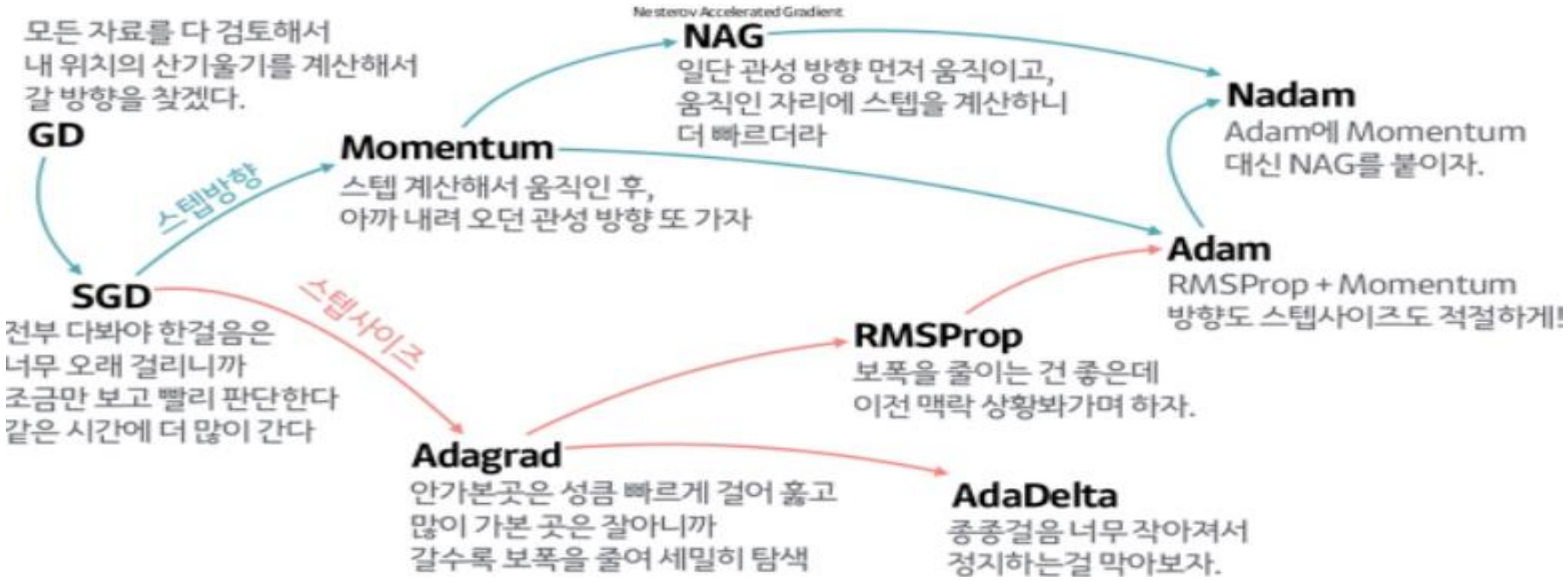
Train Set : 학습에 사용
Validation Set : 학습 중에 지속적으로 모델 검증(Overfitting과적합 방지)
Test Set : 최종 평가에 사용



Contents

Unit 04 | Optimizer

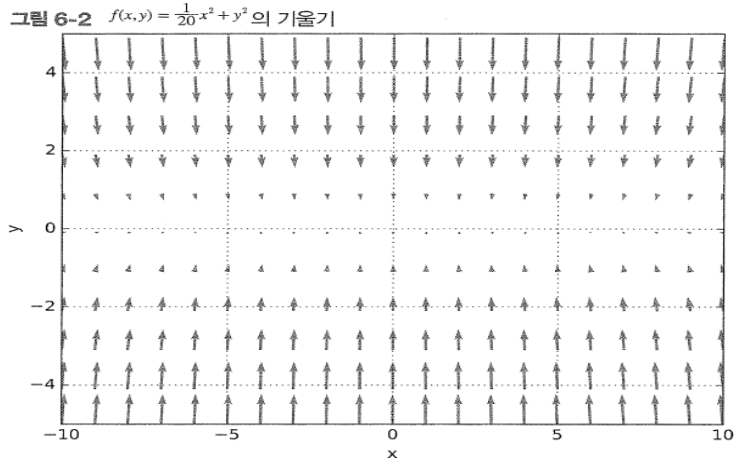
✓종류



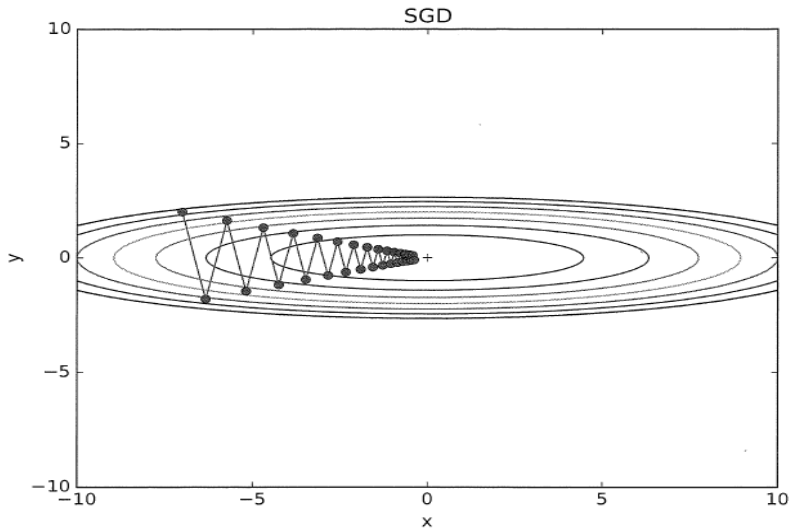
Unit 04 | Optimizer

✓SGD

$$W \leftarrow W - \eta g$$



- 기울기가 달라지는 함수에서
  무작정 기울어진 방향으로만 학습하므로
  탐색경로가 비효율적

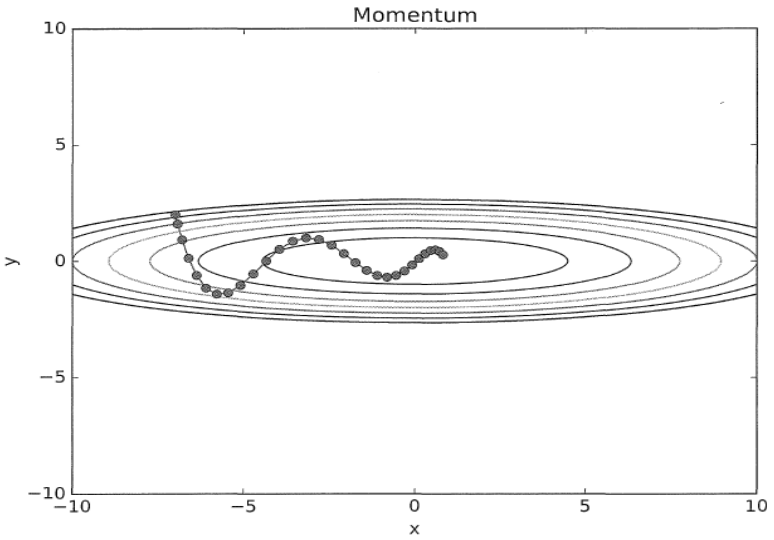


Unit 04 | Optimizer

✓Momentum

$$v \leftarrow \alpha v - \eta g$$
$$W \leftarrow W + v$$

- v: 기울기 방향으로 힘을 받게 해서 가속시킨다
- α: 과거 기울기의 영향을 줄이기 위해 0.9등의 값을 사용
- 기울기의 변화가 작은 축도 일정한 방향을 유지하면 가속하여 수렴

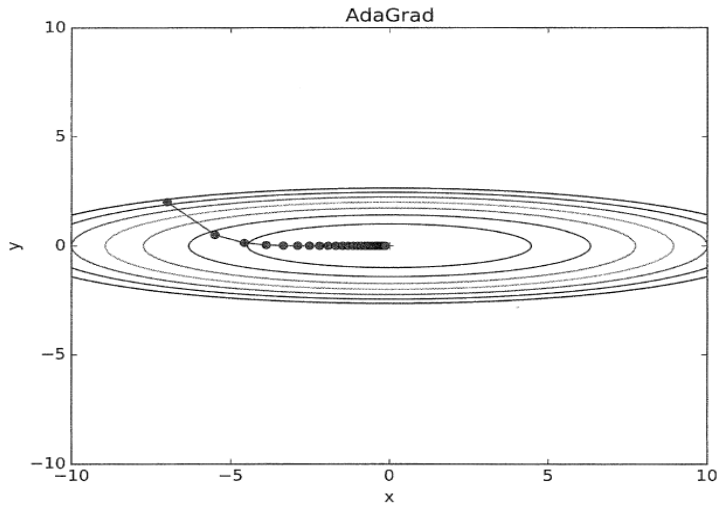


Unit 04 | Optimizer

✓AdaGrad

$$h \leftarrow h + g \cdot g$$
$$W \leftarrow W - \eta \frac{1}{\sqrt{h}} g$$

- 개별 매개변수에 적응적으로 학습률을 조정하면서 학습
- 매개변수 원소 중 크게 갱신된 원소는 학습률이 낮아진다
- 문제점으로 갱신회수가 증가하면 $G \rightarrow \infty$이므로 학습이 진행되지 않는다.



Unit 04 | Optimizer

✓RMSprop

$$h \leftarrow \alpha h + (1-\alpha) g \cdot g$$
$$W \leftarrow W - \eta \frac{1}{\sqrt{h}} g$$

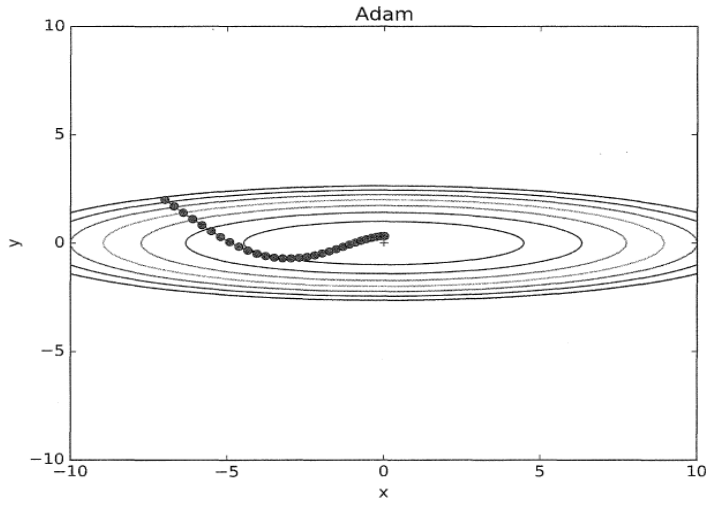
- AdaGrad의 문제인 $G \rightarrow \infty$를 해결
- 가중치 기울기를 단순히 누적시키는 것이 아니라 최신 기울기들이 더 반영되도록 한다.
- α는 주로 0.9 사용

Unit 04 | Optimizer

✓Adam

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)



Unit 04 | Optimizer

✓Adam

- Momentum과 RMSprop을 합친 형태
- Momentum과 Adagrad는 각각 v와 h가 처음에 0으로 초기화되면 W가 학습 초반에 0으로 biased되는 문제 존재
- 따라서 Adam에서는 bias-corrected moment estimate를 사용하여 biased를 해결
- 현재 많은 딥러닝 최적화 문제에서 사용한다
- 여담(근거x): optimizer를 모르면 그냥 Adam을 사용하라는 말도 있다

Assignment

wk2_optimization_assignment.ipynb를 완성해주세요

1) 빈칸을 채워주세요 (마크다운, 코드)
2) 완성된 함수로 Gradient Descent 실행해주세요
3) Hyper Parameter를 변경하며 실행해주세요

*이해한 만큼 코드주석 및 부가설명(마크다운)!
*과제에 대해 이해가 안 되거나 어렵다면 망설이지 말고 연락주세요~

Reference

Tobig's 13기 이지용, 이재빈, 12기 이유진 강의자료

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

LR with MLE: https://machinelearningmastery.com/linear-regression-with-maximum-likelihood-estimation/

convex함수 최적화 수렴:https://wikidocs.net/18086

BCE convex증명: https://towardsdatascience.com/binary-cross-entropy-and-logistic-regression-bf7098e75559

헤시안 행렬: https://angeloyeo.github.io/2020/06/17/Hessian.html

배치 학습: https://skyil.tistory.com/68

미니배치 학습: https://www.youtube.com/watch?v=lQftsyAk6V8&t=308s

Optimizer: https://hiddenbeginner.github.io/deeplearning/2019/09/22/optimization_algorithms_in_deep_learning.html

Adam 논문 리뷰: https://dalpo0814.tistory.com/29

밑바다부터 시작하는 딥러닝1

핵심 미적분학 7th edition

들어주셔서 감사합니다.