GNU

Lab Seminar: 2022. 07. 19.

How to test Hypotheses and make Inferences & About Gradient Descent

Data Science from Scratch 2nd – Chapter 7, 8

IDEALAB

Improving lives through learning

ChanKi Kim

School of Computer Science/Department of AI Convergence Engineering

Gyeongsang National University (GNU)

GNU 경상국립대학교 Gyeongsang National University

Contents

- Additional Explanation of Last Seminar
- Introduction
- Hypothesis and Inference
- Extension Learning about Hypothesis and Inference
- Gradient Descent
- Extension Learning about Gradient Descent
- Conclusion & Realization

Additional Explanation of Last Seminar

- Matplotlib vs Seaborn

특징	Matplotlib	Seaborn
기능	기본 그래프 만들 때 사용	데이터 시각화를 위한 다양한 패턴 및 플롯 포함 / Matplotlib보다 테마의 다양성 / 내장 데이터 제공
다중 표 생성	여러 그림을 동시에 열고 사용 가능 또한 그림을 닫는 함수 존재 close()	그림 생성 시간 설정, 단 메모리 부족 문제 발생 가능 그림을 닫는 함수 X
시각화	Pyplot은 MATLAB에서와 유사한 기능과 구문을 제공하고있어 MATLAB 사용자는 쉽게 학습 가능	Pandas Data Frame을 처리하는데 Matplotlib보다 수월
작동	Ploting을 위한 상태 저장이 가능하여 plot()과 같은 함수가 매개 변수 없이 작동 가능	Ploting을 위한 상태 저장이 기능이 없기에 plot()과 같은 함수는 매개 변수 반드시 필요
문법	비교적 복잡하고 긴 코드 사용	비교적 간단한 코드 사용

Introduction

- Why should we learn Hypothesis and Inference?
  - 연구 및 개발을 할 때, 새로운 사실을 발견 혹은 발명했음을 입증
  - 입증을 위한 통계적 검증 필요

- Why should we learn Gradient Descent?
  - 실제 값과 예측 값 오차를 나타내는 손실 값을 최소로 하는 최적값 필요성
  - 최적값을 구하는 기법 중 하나

Hypothesis and Inference

- Null Hypothesis & Alternative Hypothesis
  - Null Hypothesis : "새로울 게 없다!"
  - Alternative Hypothesis : "무언가 새로운 것이 있다."
    - 귀무가설 검증 실패는 간접적으로 새 가설에 대한 검증

Hypothesis and Inference

Proving the Null Hypothesis Wrong

VS

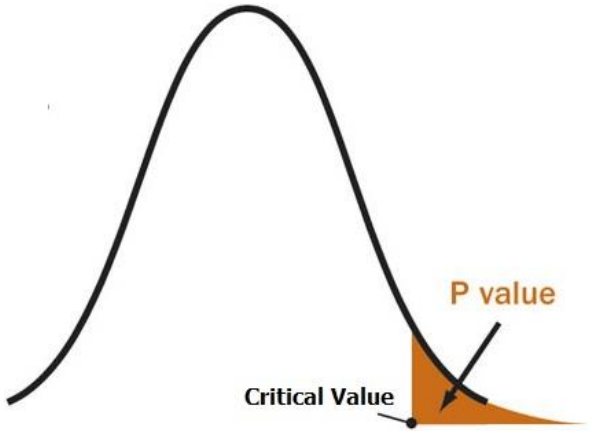
Proving the Alternative Hypothesis Correct

Hypothesis and Inference

- Why should we prove the Null Hypothesis wrong?
  - 참이 아님을 증명하는 것이 참임을 증명하는 것보다 수월
  - 귀무가설을 *올바르게* 서술하는 것보다 대립가설을 *정확하게* 서술하는 것이 어려움
    - But, 가설 검증은 귀무가설 기각으로만 가능한 것 X

Hypothesis and Inference

- P-value
  - 귀무가설에서 주장한 바가 옳을 확률
  - 0과 1사이로 표준화된 지표이자 확률값
    - p-value < 0.05(or 0.01) : 귀무가설 기각
    - p-value > 0.1 : 귀무가설 채택
    - 검정통계량 : 귀무가설이 참이라는 가정 아래 얻은 통계량



$$ex)\ t = \frac{\bar{x} - \mu_0}{\frac{s}{\sqrt{n}}}$$

Hypothesis and Inference

- P-hacking
  - 유의한 통계 수준이 나오도록 데이터 수집/선택/분석 과정 실행

- Why we use P-hacking?
  - 통계적으로 유의하지 않은 결과는 인정되지 않고, 이것은 논문 출판 여부와 직결
  - 기존 결과 재현 연구로 통계적으로 유의한 결론 내리는 연구가 인정

Hypothesis and Inference

- Bayesian Inference
  - "아는 것을 모르는 것으로 추론하는 것"
  - 베이지안 관점 확률은 믿음의 정도 -> 불확실성 측정 도구

$$P(Y|X) = \frac{P(X|Y)P(H)}{P(X)}$$

Extension Learning about Hypothesis and Inference

- How to prevent P-hacking?
  - 데이터들을 수집 및 가공 과정을 미리 정한 후, 결과에 따라서 변경 X
  - 통계적으로 유의한 결과보다 데이터 수집과 연구 방법론에 대한 강조 필요

- The recent trend of using P-hacking
  - P-value는 표본 수에 영향을 받아, 매우 많은 샘플링을 통해 P-hacking 사용

Extension Learning about Hypothesis and Inference

- Frequentist
  - 빈도주의 관점에서의 확률
  - -> 상대적 빈도의 극한 =(관심 있는 사건 발생 횟수) / (전체 시행 횟수)의 극한
  - 한계를 보여주는 예시 : 대선에서 당선될 확률

Gradient Descent

- Partial Derivative

$$f_x(x,y) = f_x = \frac{\partial f}{\partial x}$$

  - 하나의 변수에 대해서 미분하고, 나머지는 상수로 취급
  - 주로 다변수 함수에서 한 개의 변수의 값이 변화할 때의 변화율을 알기 위해 활용

- Chain Rule
  - 합성함수의 미분

$$p = f(x,y), x = x(t), y = y(t)$$

$$\frac{df}{dt} = \frac{\partial f}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial f}{\partial y} \cdot \frac{dy}{dt}$$

Extension Learning about Gradient Descent

- Optimizer
  - 총 손실 E를 최소화하는 최적의 w를 찾는 알고리즘
    - Grdient Descent : 경사를 따라 여러 번의 반복 과정을 통해 최적의 w 발견
    - Momentum : 하강하는 가속도 유지
    - AdaGrad : 자주 변하는 w값의 학습률은 작게, 자주 변하지 않는 w값은 학습률을 크게
    - AdaDelta : AdaGrad이지만 학습률이 작아져서 학습 안되는 문제 방지
    - Adam : RmsProp와 Momentum 합한 알고리즘
  - Optimizer는 정해진 것은 없고 가장 결과가 잘 나오는 것을 적용

Gradient Descent

- Optimization Theory
  - 총 손실 E를 최소화하는 최적의 w를 찾는 이론
    - Ex) Brute-Force, Gradient Descent

$$ex) \text{총 손실 } E = \frac{1}{n}(\text{예측값} - \text{실제값})^2$$

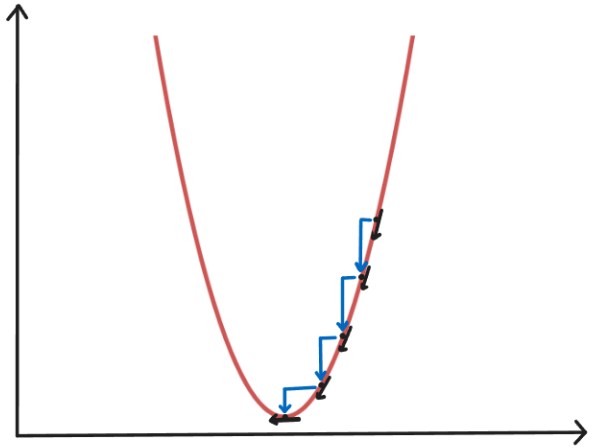
Gradient Descent

- Brute-Force
  - 가장 단순한 방법으로 가능한 모든 수를 대입해보는 기법
    - Problem 1) 최적값이 존재하는 범위 알아야 함
    - Problem 2) 최적값을 정확히 찾기 위해 무한히 촘촘하게 조사
    - Problem 3) 계산 복잡도가 매우 높음

"적게 대입하고 최적값을 찾을 수는 없을까?" -> Gradient Descent

Gradient Descent

- Gradient Descent
  - 경사를 따라 여러 번의 반복 과정을 통해 최적의 지점을 찾는 기법
  - 경사의 반대 방향으로 계속 이동시켜 극값에 이르게 하는 과정
  - 경사는 기울기를 이용해 계산



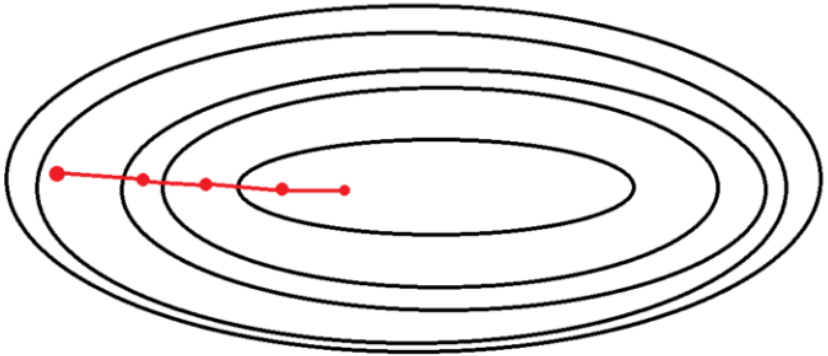
Gradient Descent

- What is Batch?
  - GPU가 한 번에 처리하는 데이터의 묶음

- Types of Gradient Descent
  - Batch Gradient Descent (BGD)
  - Stochastic Gradient Descent(SGD)
  - Mini-Batch Stochastic Gradient Descent(MSGD)

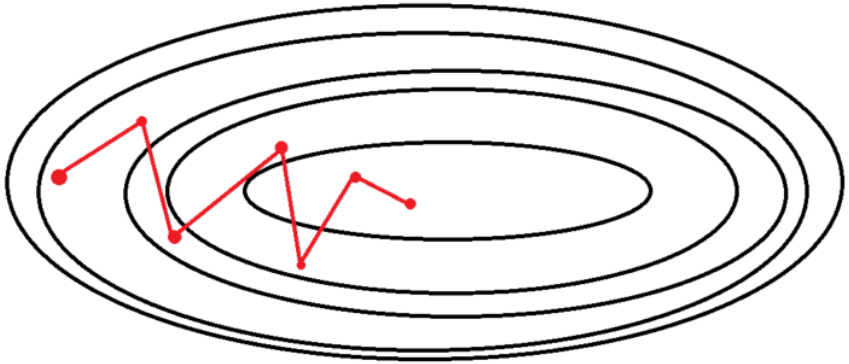
Gradient Descent

- Batch Gradient Descent
  - 전체 Train Data를 하나의 Batch로 묶어서 학습
  - 전체 Train Data를 한 번에 처리한다는 특징 때문에 많은 메모리를 요구하며 이에 따른 긴 학습 시간 요구
  - 항상 같은 Data에 대한 Descent를 구하기 때문에 안정적 수렴



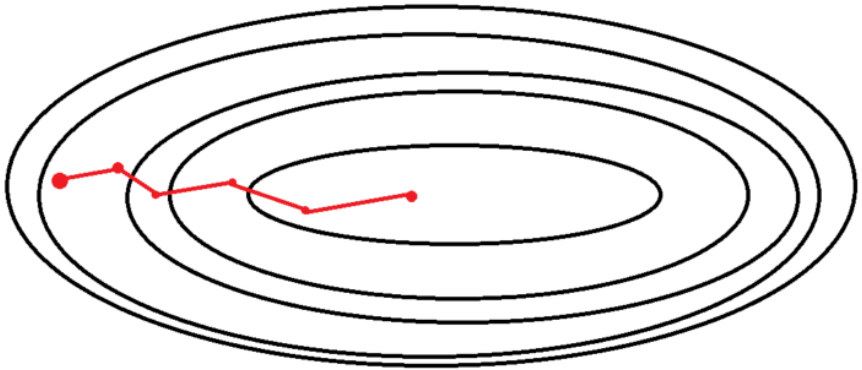
Gradient Descent

- Stochastic Gradient Descent
  - 랜덤으로 선택한 단 하나의 Train Data를 크기가 1인 Batch로 1회 학습 진행
  - 비교적 적은 수의 Data로 학습하여 빠른 학습 속도
  - 수렴에 Shooting 발생



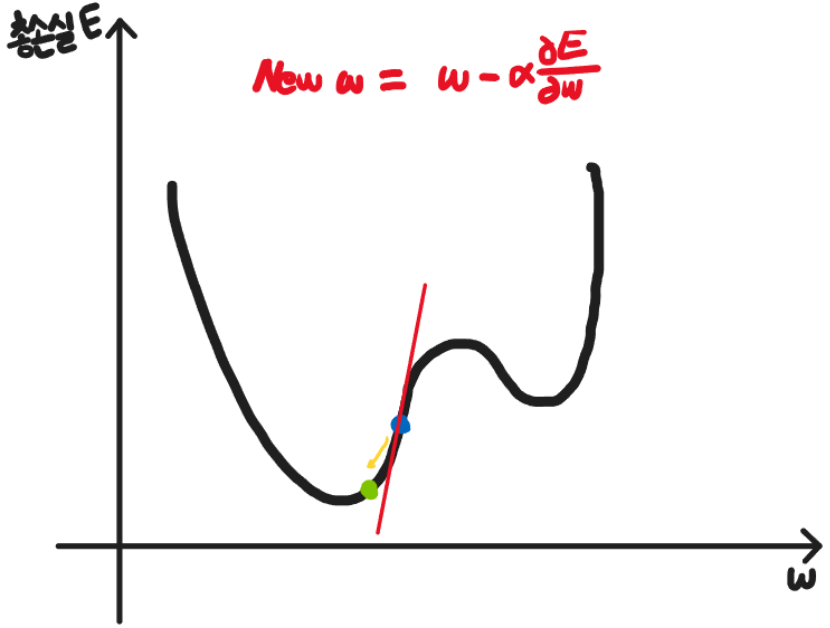
Gradient Descent

- Mini-Batch Stochastic Gradient Descent
  - 전체 Train Data를 Batch size(사용자 지정)개씩 나눈 후 학습 진행
  - SGD와 같이 수렴에 shooting이 발생하긴 하지만, 한 배치의 손실 값의 평균으로 경사 하강을 진행해 Shooting의 발생 정도 ↘
  - SGD와 BGD의 절충안



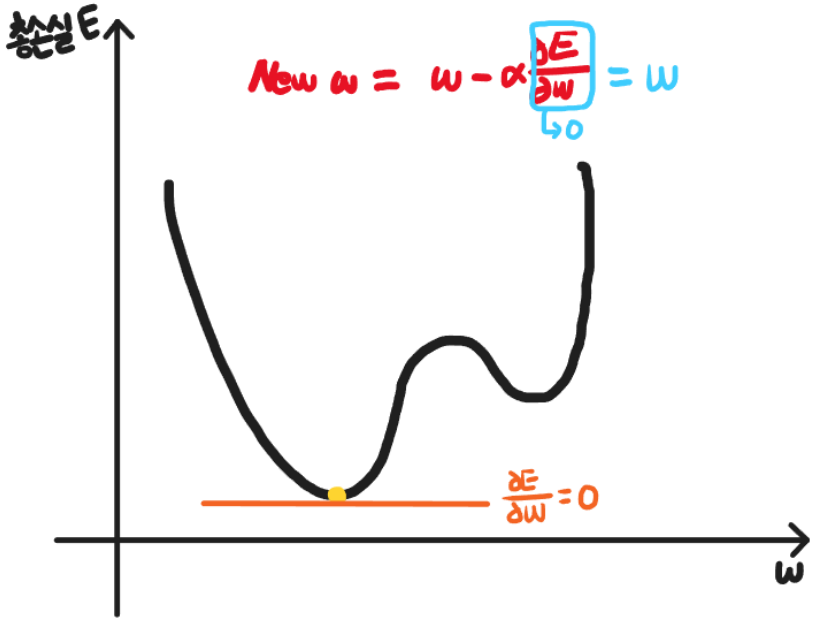
Gradient Descent

- How to find the optimal w value?
  - 현재 w값에서의 접선의 기울기를 w값에서 빼서 w값 갱신



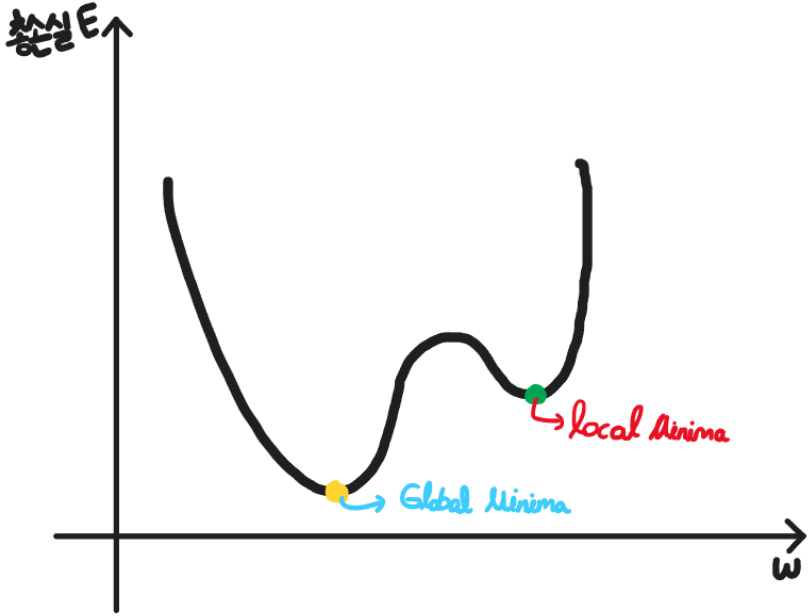
Gradient Descent

- How to find the optimal w value?
  - 극점에서의 접선의 기울기는 0이기 때문에 더 이상 w 값 갱신 X



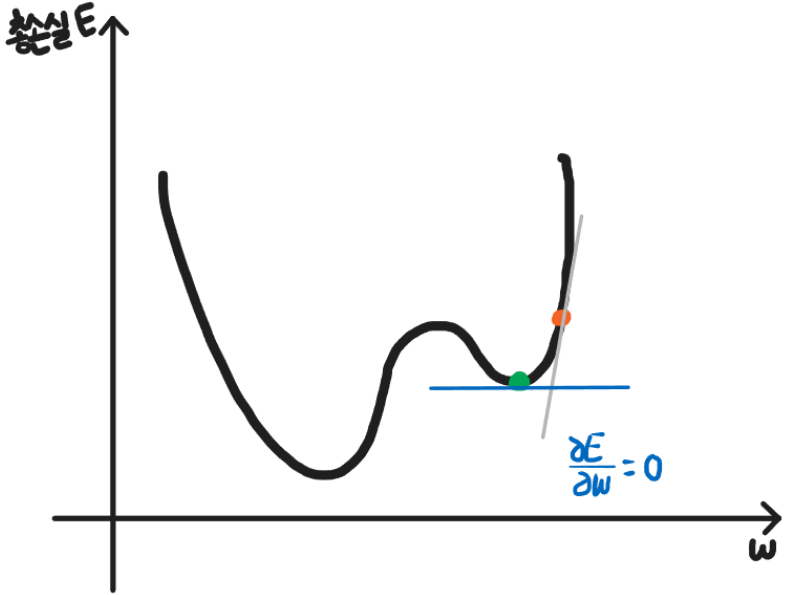
Gradient Descent

- Global Minima vs Local Minima



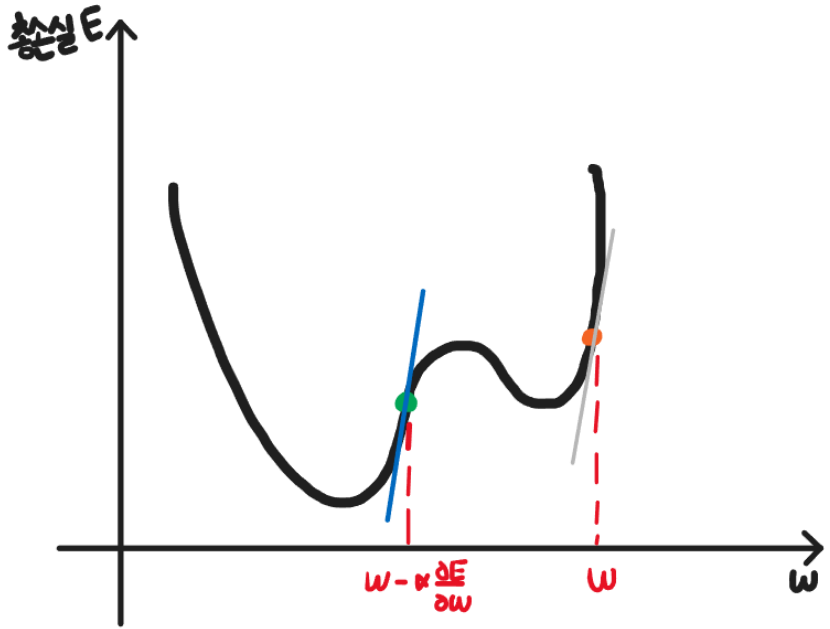
Gradient Descent

- If we only subtract the Descent
  - 손실을 최소로 하는 w를 찾지 못할 가능성 존재 (Local Minima)



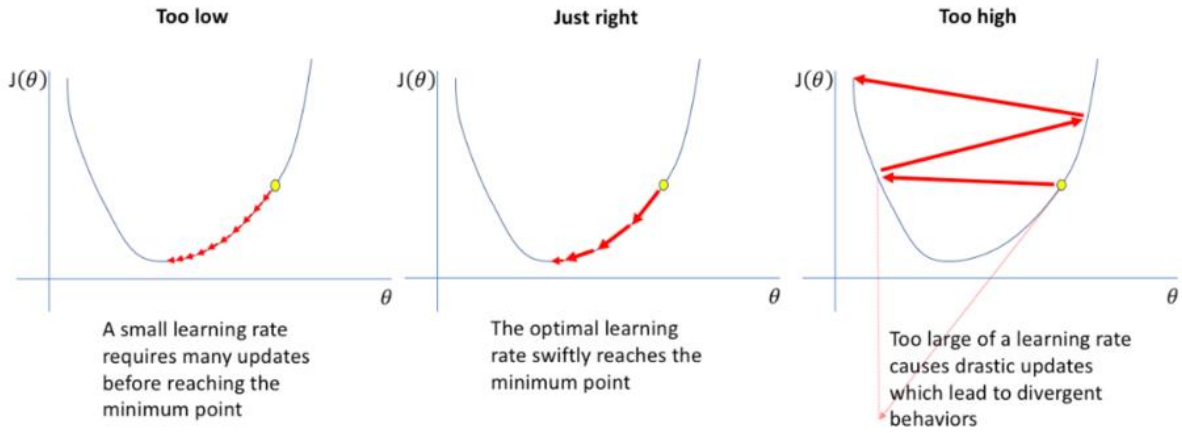
Gradient Descent

- Learning Rate
  - 최적의 w 값을 찾는 과정에서 오류를 범하지 않게 해주는 값



Extension Learning about Gradient Descent

- Fixed Learning Rate
  - 고정된 값만 주면 복잡한 문제의 경우 학습이 느리거나 일어나지 않는 경우 존재
  - Learning Rate Scheduler가 이러한 문제 해결



Extension Learning about Gradient Descent

- Learning Rate Scheduler
  - 학습 과정에서 Learning Rate를 조정
  - 초기에는 Learning Rate를 크게 설정하여 최적화를 하고, 찾고자 하는 최적값에 근사해질수록 Learning Rate를 줄이는 미세한 조정을 하는 것이 학습 결과 우수

Extension Learning about Gradient Descent

- Lambda LR
  - Epoch에 따른 가중치를 이용해 Learning Rate 감소

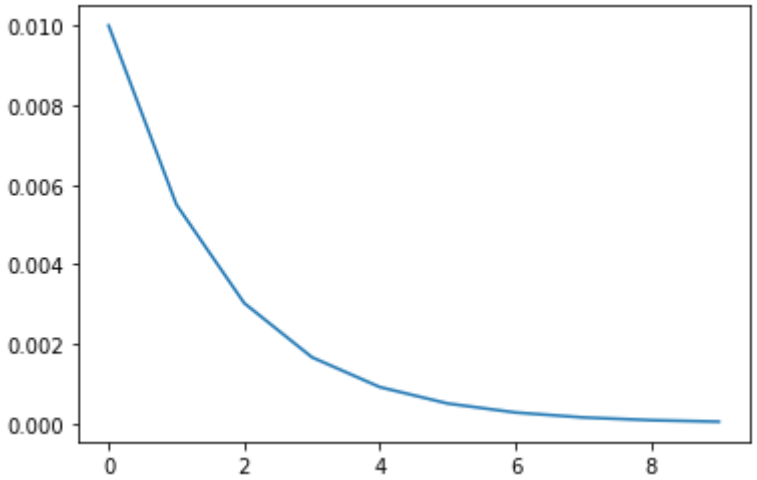
$$lr_{\text{epoch}} = lr_{\text{initial}} * Lambda(epoch)$$

```
import torch
import matplotlib.pyplot as plt

model = torch.nn.Linear(2, 1)
optimizer = torch.optim.SGD(model.parameters(), lr=1e-2)
first_lambda = lambda epoch: 0.55 ** epoch
scheduler = torch.optim.lr_scheduler.LambdaLR(optimizer, lr_lambda=first_lambda)

lr_adjust = []

for i in range(10):
    optimizer.step()
    lr_adjust.append(optimizer.param_groups[0]["lr"])
    scheduler.step()

print(lr_adjust)
plt.plot(range(10),lr_adjust)
```

Extension Learning about Gradient Descent

- Lambda LR

```
[0.01, 0.0055000000000000005, 0.0030250000000000003, 0.0016637500000000005, 0.0009150625000000003, 0.0005032843750000001, 0.00027680640625
000016, 0.0001522435234375001, 8.373939789062506e-05, 4.6053665839843786e-05]
```



Extension Learning about Gradient Descent

- MultiplicativeLR

$$lr_{\text{epoch}} = lr_{\text{epoch - 1}} * Lambda(epoch)$$

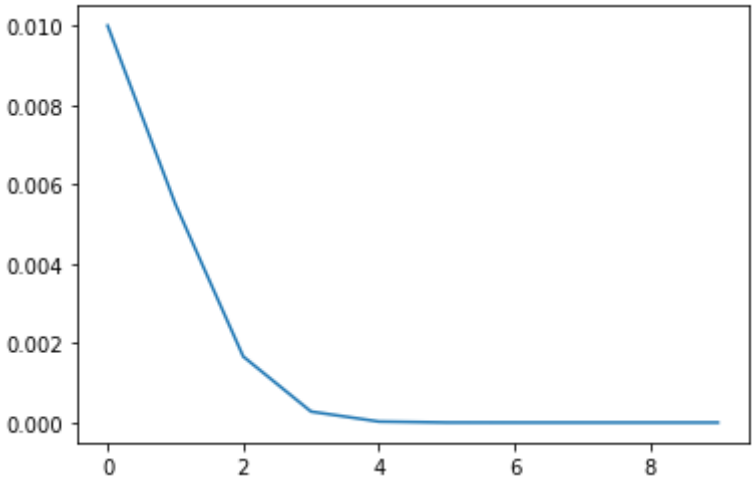
  - 초기 Learning Rate에 Lambda 함수에서 나온 값을 누적 곱하여 Learning Rate 계산

```
import torch
import matplotlib.pyplot as plt

model = torch.nn.Linear(2, 1)
optimizer = torch.optim.SGD(model.parameters(), lr=1e-2)
first_lambda = lambda epoch: 0.55 ** epoch
scheduler = torch.optim.lr_scheduler.MultiplicativeLR(optimizer, lr_lambda=first_lambda)

lr_adjust = []

for i in range(10):
    optimizer.step()
    lr_adjust.append(optimizer.param_groups[0]["lr"])
    scheduler.step()

print(lr_adjust)
plt.plot(range(10),lr_adjust)
```

Extension Learning about Gradient Descent

- MultiplicativeLR

```
[0.01, 0.005500000000000005, 0.001663750000000005, 0.0002768064062500016, 2.5329516211914083e-05, 1.2747949735765551e-06, 3.528714153412901e-08, 5.372238759193552e-10, 4.498387065959216e-12, 2.071672147539610 6e-14]
```



Extension Learning about Gradient Descent

- StepLR

$$lr_{\text{epoch}} = \begin{cases} Gamma * lr_{\text{epoch}-1}, & \text{if epoch } \% \text{ step\_size} = 0 \\ lr_{\text{epoch}-1}, & \text{otherwise} \end{cases}$$

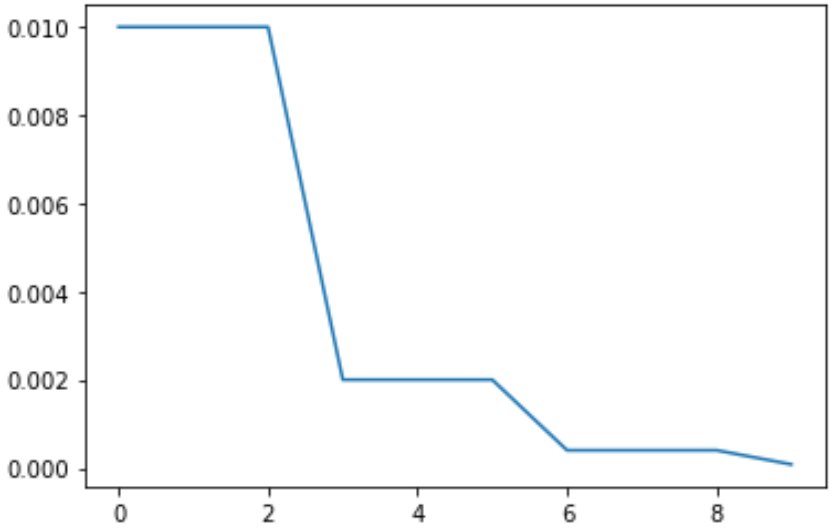
  - Step size마다 Gamma 비율로 Learning Rate 감소

```
import torch
import matplotlib.pyplot as plt

model = torch.nn.Linear(2, 1)
optimizer = torch.optim.SGD(model.parameters(), lr=1e-2)
scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=3, gamma=0.2)

lr_adjust = []

for i in range(10):
    optimizer.step()
    lr_adjust.append(optimizer.param_groups[0]["lr"])
    scheduler.step()

print(lr_adjust)
plt.plot(range(10),lr_adjust)
```

Extension Learning about Gradient Descent

- StepLR

[0.01, 0.01, 0.01, 0.002, 0.002, 0.002, 0.0004, 0.0004, 0.0004, 8e-05]



Extension Learning about Gradient Descent

- MultiStepLR

$$lr_{\text{epoch}} = \begin{cases} Gamma * lr_{\text{epoch}-1}, & \text{if epoch in [milestones]} \\ lr_{\text{epoch}-1}, & \text{otherwise} \end{cases}$$

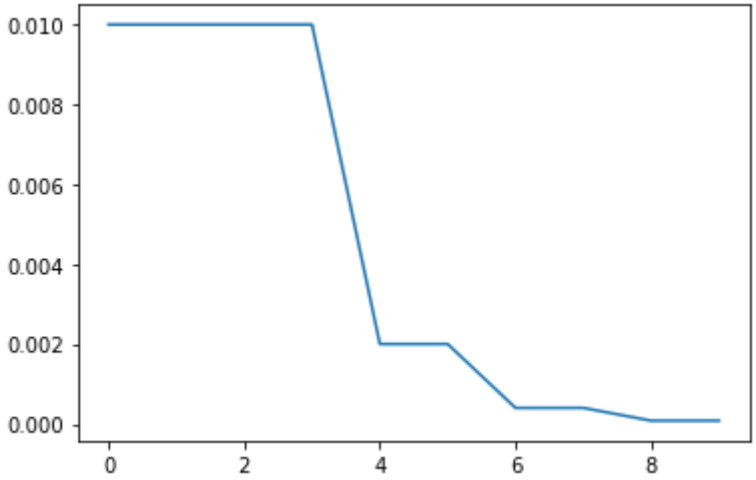
  - Learning Rate 감소시킬 Epoch를 지정

```
import torch
import matplotlib.pyplot as plt

model = torch.nn.Linear(2, 1)
optimizer = torch.optim.SGD(model.parameters(), lr=1e-2)
scheduler = torch.optim.lr_scheduler.MultiStepLR(optimizer, milestones=[4,6,8], gamma=0.2)

lr_adjust = []

for i in range(10):
    optimizer.step()
    lr_adjust.append(optimizer.param_groups[0]["lr"])
    scheduler.step()

print(lr_adjust)
plt.plot(range(10),lr_adjust)
```

Extension Learning about Gradient Descent

- MultiStepLR

[0.01, 0.01, 0.01, 0.01, 0.002, 0.002, 0.0004, 0.0004, 8e-05, 8e-05]



Conclusion & Realization

- The Direction of Future Learning
  - P-value 개념을 추후에 모델 학습할 때 잘 활용해볼 예정
  - 경사 하강법의 개념을 추후에 직접 적용해보며 학습해볼 예정
  - Optimizer의 예시들을 열거하고 간단한 개념에 대해서 알아보았으니 Gradient Descent 뿐만 아니라 다른 Optimizer 또한 추후에 좀 더 깊게 공부하고 직접 모델을 돌려보며 하나씩 적용해볼 예정

Thank you for your Attention!

경상국립대학교

Gyeongsang National University

Improving lives through learning

IDEALAB