
# 손실함수와 경사하강법

수퍼스타를 키우는 곳
전주대학교

# 경사하강법

경사하강법 개요

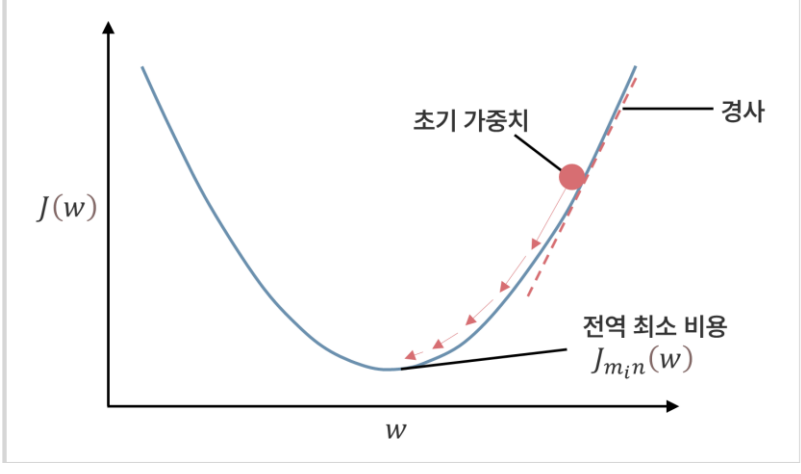
# 경사하강법 개요

## ❖ 경사하강법 개념

- 작동 원리
  - 함숫값이 낮아지는 방향으로 독립 변숫값을 변형시켜 가면서 최종적으로는 최소 함숫값을 갖도록 하는 독립 변숫값을 찾는 방법
  - 기울기
    - 해당 경사의 반대 방향으로 계속 이동시켜 극값에 이를 때까지 반복시키는 것

## ❖ 경사하강법 개요

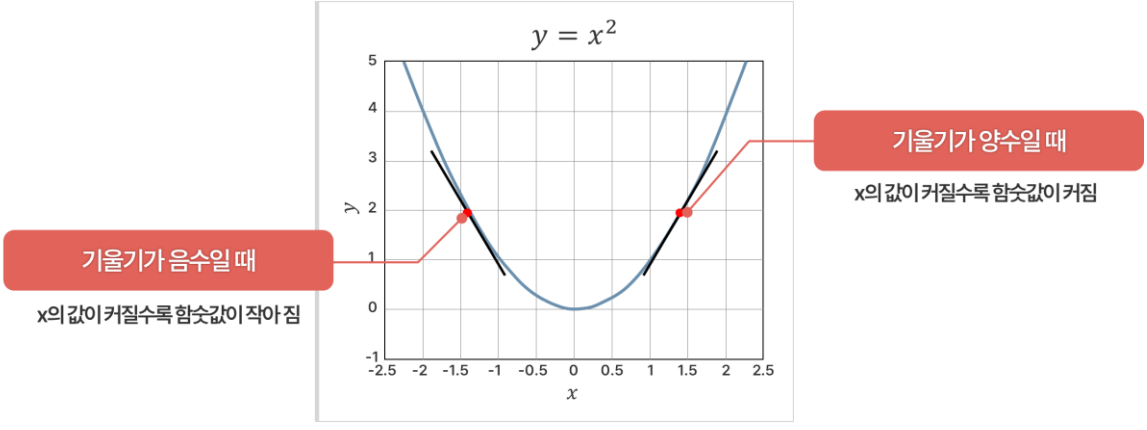
- 함수가 닫힌 형태가 아닌 경우
- 함수가 너무 복잡해 미분 계수를 구하기 어려운 경우
- Gradient Descent Method를 사용하는 것이 미분 계수를 구하는 것보다 더 쉬운 경우
- 데이터 양이 너무 많은 경우(효율적으로 계산하기 위해)
- 머신러닝 및 딥러닝 알고리즘을 학습시킬 때 사용하는 방법 중 하나
- 1차 근삿값 발견용 최적화 알고리즘



# 경사하강법 개요

## ❖ 경사하강법 개요

- 함수의 기울기를 이용




- 이동거리: 최솟값에 가까워질수록 기울기의 크기는 작아짐
  - 최솟값에 가까울수록 조심히 움직여야 발산하지 않음
  - $x_{i+1}=x_i$ −(이동거리 × 기울기의 부호)
  - $x_{i+1}=x_i$ −(기울기의 크기 × 기울기의 부호)
  - $x_{i+1}=x_i$ − 기울기

# 경사하강법 개요

## ❖ 경사하강법 수식

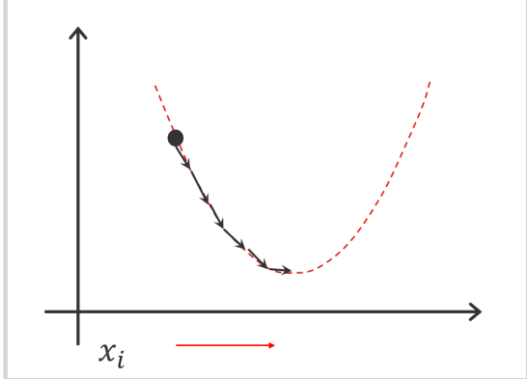
- Step Size

$$x_{i+1} = x_i - \text{기울기} \longrightarrow x_{i+1} = x_i - \eta \cdot \frac{df}{dx}(x_i)$$

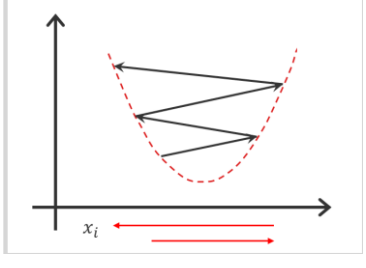
학습률

  - 사용자의 필요에 맞게 이동 거리를 조절할 수 있도록 해주기 위하여 조절 인자를 넣기도 함

- 인자 $\eta$ 값(학습률)이 너무 작을 경우
  - 학습률이 너무 적으면 최저점을 찾아 수렴하는 시간이 매우 느림
  - 반복해야 하는 값이 많으므로 학습 시간이 오래 걸림
  - 지역 최소값(Local Minimum)에 수렴할 수 있음



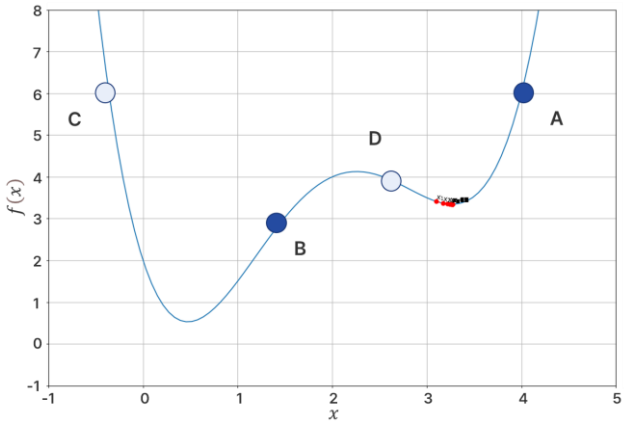

- 인자 $\eta$ 값이 너무 클 경우
  - 빠르게 수렴할 수 있지만, 최솟값으로 수렴되지 못하고 발산할 여지 있음
  - 스텝이 너무 커서 전역 최솟값(global minimum)을 가로질러 반대편으로 건너뛸 수 있음



# 경사하강법 개요

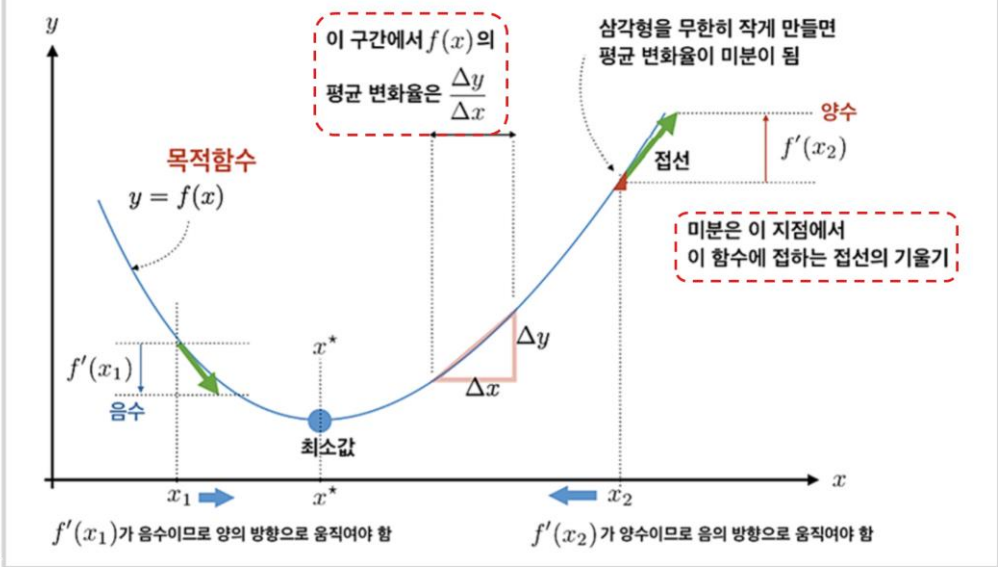
## ❖ 경사하강법 수식

- 방향과 스텝사이즈




## ❖ 미분과 기울기, 그리고 경사하강법의 개념

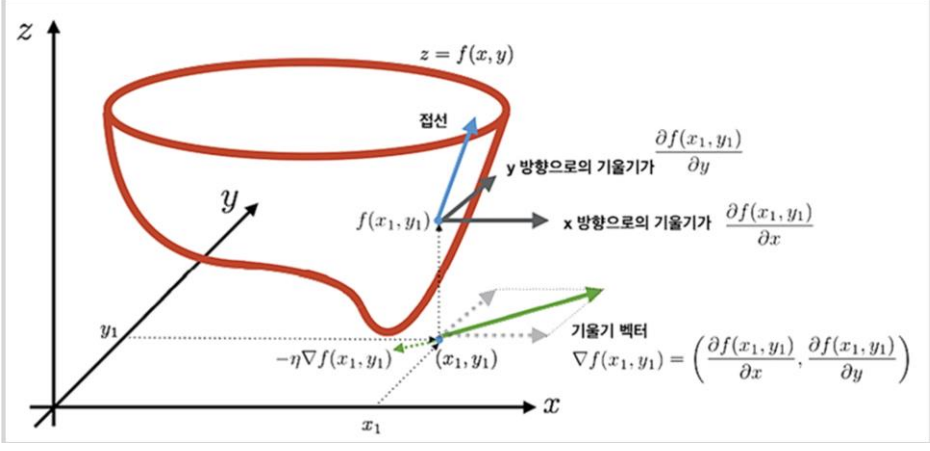
- 미분이란 순간 변화량을 구하는 것
- 독립 변수값의 변화량에 대한 함수값 변화량 비의 극한
- 경사하강



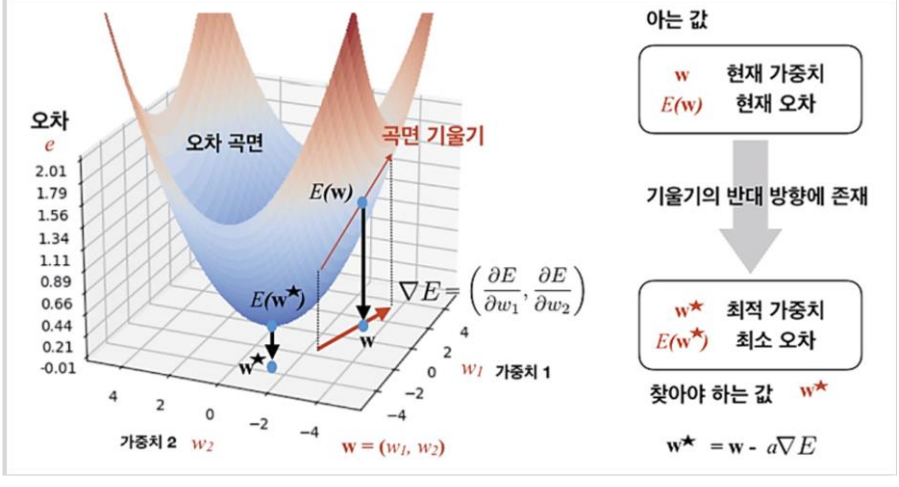
# 경사하강법 개요

## ❖ 미분과 기울기, 그리고 경사하강법의 개념

- 편미분과 기울기와 벡터
  - 편미분(partial derivative)이란 둘 이상의 변수들을 가지는 함수 $f$가 있을 경우, 이 함수를 각각의 변수에 대해서 독립적으로 미분을 하는 방식



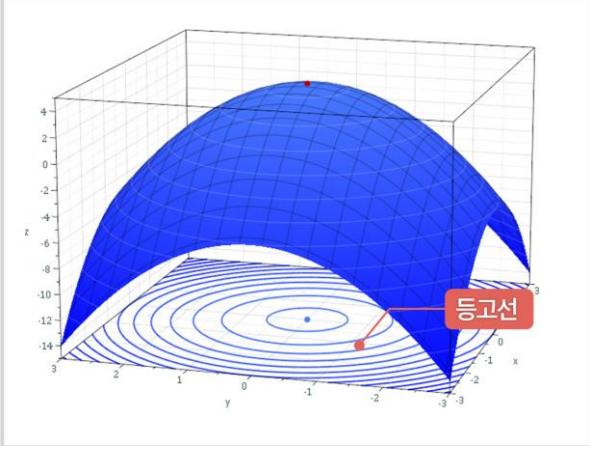

- 오차를 감소시키는 방향으로 가중치를 조정하는 방법



# 경사하강법 개요

## ❖ 미분과 기울기, 그리고 경사하강법의 개념

- 함수가 여러 개 있는 경우
  - 지역 최솟값만 찾거나 전역 최솟값을 찾는 경우가 발생

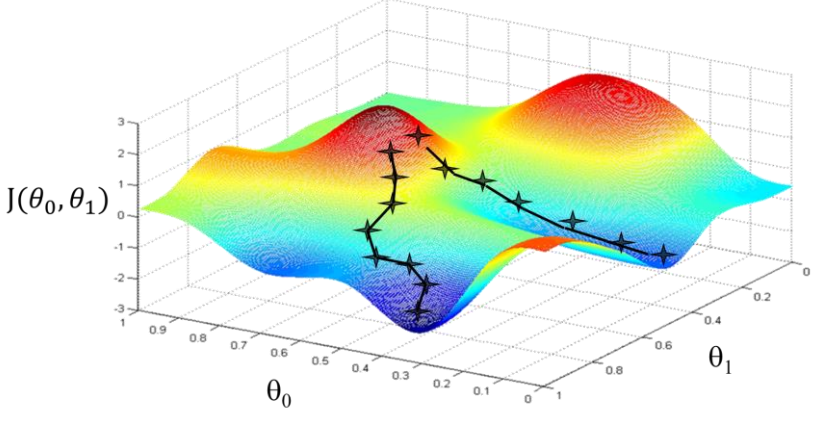



- 경사하강법
  - 새로운 가중치를 찾을 때 현재 값의 기울기에 학습률을 곱해서 작아지는 형태로 찾는 방법

$$w' = w - \eta \cdot \nabla f(w)$$

## ❖ 경사하강법의 약점

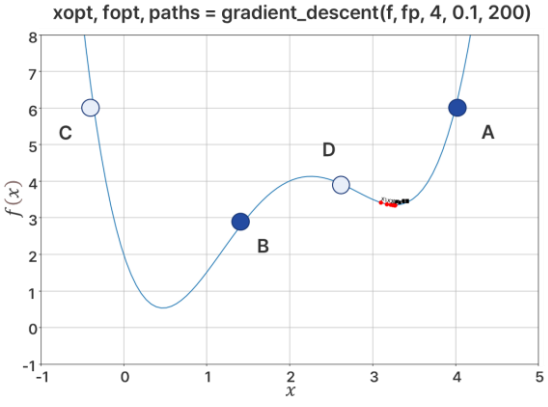
- 함수가 여러 개 있는 경우

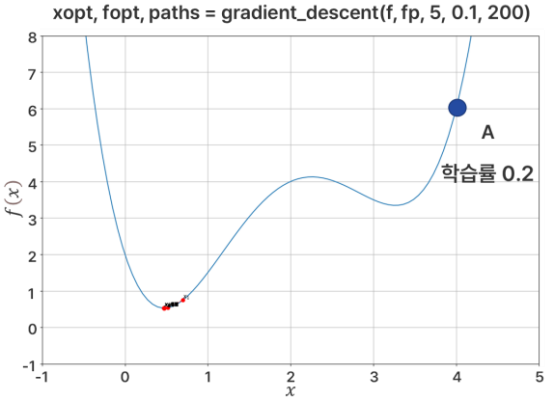


# 경사하강법 개요

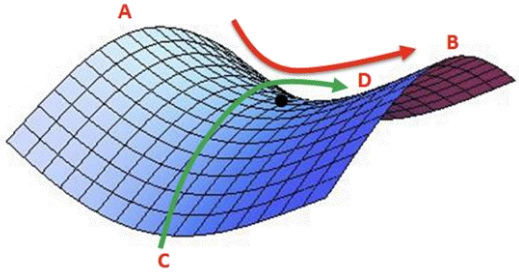
## ❖ 경사하강법의 약점

- 시작점의 차이에 따라 다른 결과를 가져올 수 있음




- 경사하강법은 현재 위치에서의 기울기를 사용하기 때문에 지역 최솟값에 빠질 수 있음




- Saddle Point
  - 안장점은 기울기가 0이지만 극값이 아닌 지점을 의미함

# 경사하강법 개요

## ❖ 예제

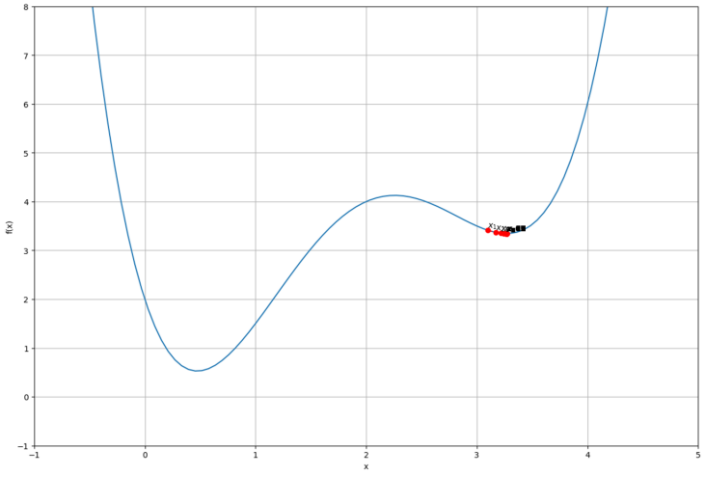
```
def gradient_descent(f, fp, x0, learning_rate,
max_Iter):
   paths = []   # 비어 있는 list

   print(\"{0:02d} : {1:5.5f}, {2:6.5f}\".format(0, x0,
f(x0)))

   for i in range(max_Iter):
      x1 = x0 - learning_rate * fp(x0)
      # x를 학습률로 지정한 거리만큼 이동
      print('{0:02d} : {1:5.5f}, {2:6.5f}'.format(i+1, x1,
f(x1)))

      # x0에 x1의 값을 대입해주며 갱신해주고, 다시 반복
      x0 = x1
      paths.append(x0)   # 최적값을 찾아가는 각 점을 모은 것
   return(x0, f(x0), np.array(paths))
```

```
def f(x):
   return  0.5*x**4 - 4*x**3 + 10*x**2 - 7*x + 2
def fp(x):
   return 2*x**3 - 12*x**2 + 20*x – 7

xopt, fopt, paths = gradient_descent(f, fp, 4, 0.1, 200)

x = np.linspace(-1, 5, 100)
y = f(x)
plt.figure(figsize=(15, 10))
plt.plot(x, f(x))
plt.plot(paths, f(paths), 'ro:')
for k, point in enumerate(paths):
   plt.text(point, f(point), '$x_{0}$'.format(k+1),
verticalalignment='bottom')
plt.grid()
plt.xlabel('x')
plt.ylabel('f(x)')
plt.xlim(-1, 5)
plt.ylim(-1.0, 8.0)
plt.show()
```

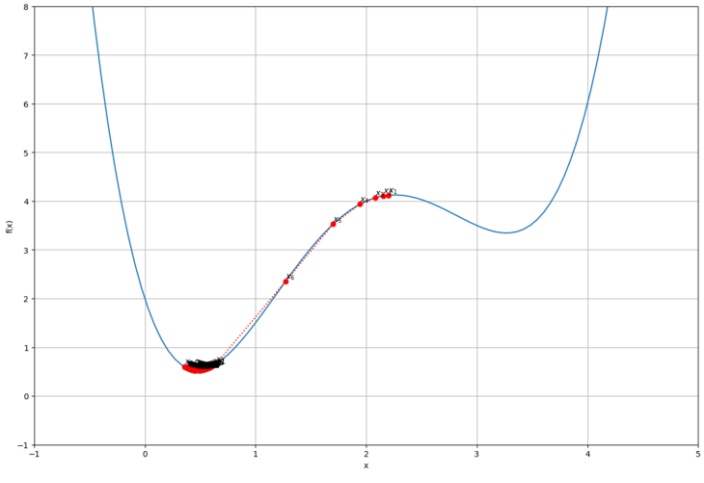
# 경사하강법 개요

## ❖ 예제

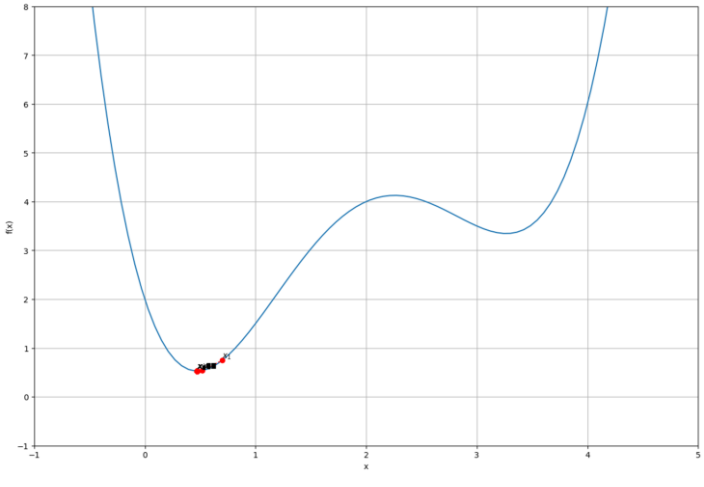
xopt, fopt, paths = gradient_descent(f, fp, 4, 0.1, 200)



xopt, fopt, paths = gradient_descent(f, fp, 4, 0.2, 200)



xopt, fopt, paths = gradient_descent(f, fp, 5, 0.1, 200)



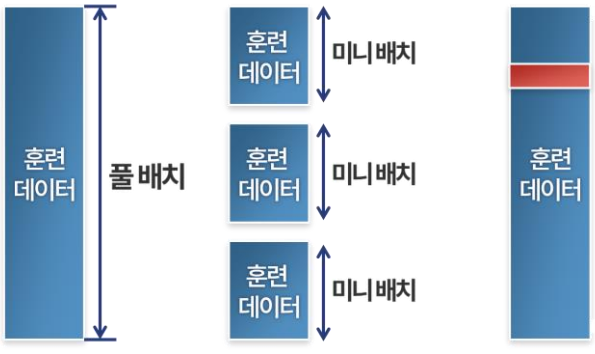
# 경사하강법

경사하강법 변형

# 경사하강법 변형

## ❖ 경사하강법의 변형

- 데이터양에 따라서 Gradient 계산을 바꿔보자는 아이디어
- Gradient 계산에 사용하는 데이터의 양에 따라 구분
- Batch Gradient Descent
  - 한 번의 업데이트에 전체 훈련 데이터세트 사용
  - 매우 느림
  - Full Gradient Descent
- Mini-batch Gradient Descent
  - 임의의 작은 샘플(50-256) 데이터세트 사용
- 확률적 경사하강법(Stochastic Gradient Descent)
  - 매 step에서 딱 한 개의 샘플을 무작위로 선택 사용
    - Batch Gradient Descent
    - Stochastic Gradient Descent
    - Mini-batch Gradient Descent



# 경사하강법 변형

## ❖ Batch Gradient Descent

- 배치 사이즈(Batch size)
- 에포크(Epoch)
  - 1 epoch
    - 전체 학습 데이터셋이 한 신경망에 적용되어 순전파와 역전파를 통해 신경망을 한번 통과했다는 의미
  - 지나치게 Epoch를 높이게 되면, 그 학습 데이터셋에 과적합(Overfitting)되어 다른 데이터에 대해선 제대로 된 예측을 하지 못할 수 있음

**Repeat {**

**모든** $x^i$**에 대해 미분값** $\nabla^1, \nabla^2, \nabla^3, \ldots, \nabla^m$ **계산**
**미분값의 평균 계산 :** $\nabla = \frac{1}{m}\sum_{i=1}^{m}\nabla^i$
**미분값의 평균을 이용하여 업데이트 :** $\theta = \theta - \alpha\nabla$

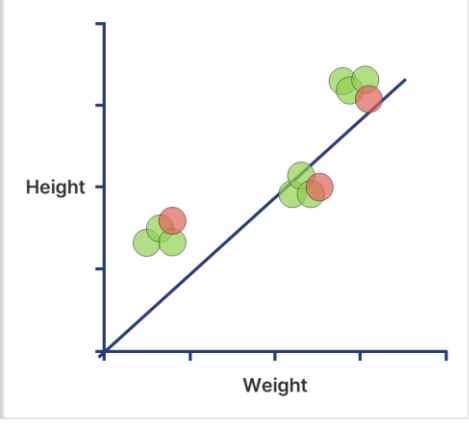
**} until convergence**

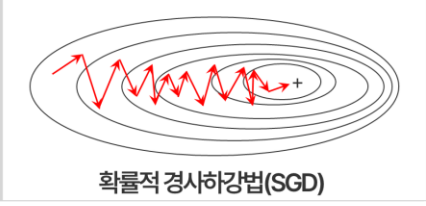
$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta)$$

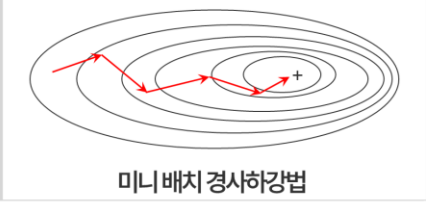
# 경사하강법 변형

## ❖ Mini-batch Gradient Descent

- 예를 들어, 1,000개인 학습 데이터 셋에서 batch_size를 100으로 잡았으면 총 10개의 mini batch가 나오게 됨
  - 이 100개씩의 mini batch를 갖고 한 번씩 SGD를 진행



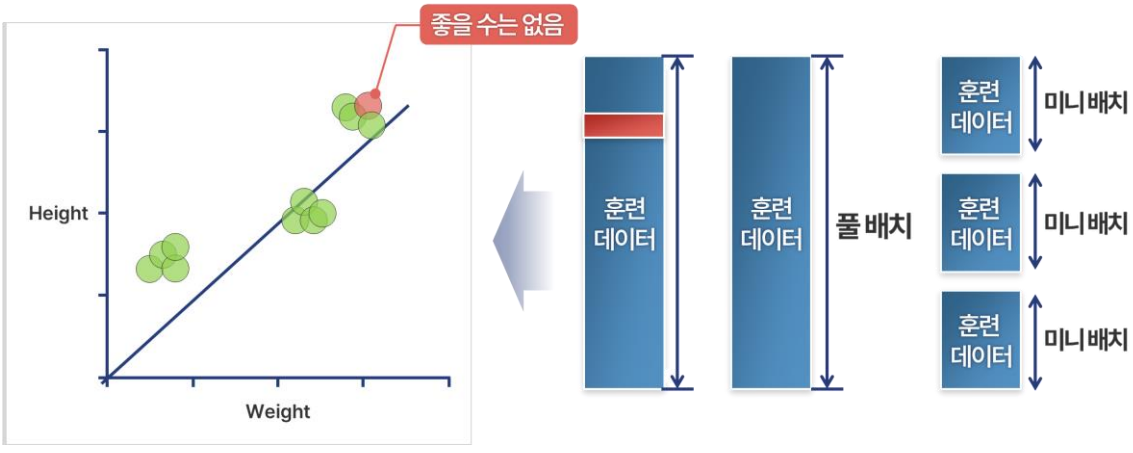




  - 지그재그 하는 것의 중간 점이 미니 배치 경사하강법

# 경사하강법 변형

## ❖ 확률적 경사하강법

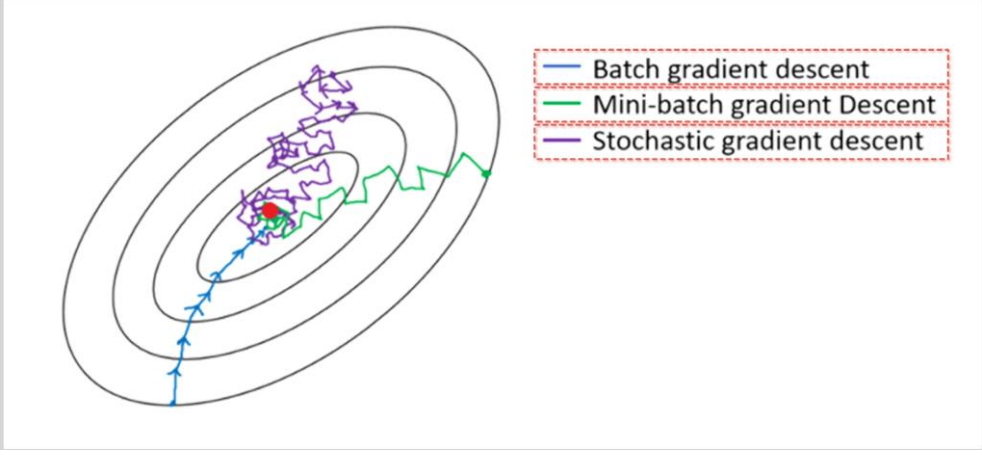
- 매 step에서 딱 한 개의 샘플을 무작위로 선택하고 그 하나의 샘플에 대한 기울기 계산
  - 매우 불규칙하게감
- 특징
  - 샘플의 선택이 확률적이기 때문에 배치 경사하강법에 비해 불안정




- 확률적 경사하강법 특징
  - 샘플 선택이 확률적이기 때문에 배치 경사하강법에 비해 불안정
  - Cost Function이 매우 불규칙할 경우 Local Minimum을 건너뛰도록 도와주어 Global Minimum 찾을 가능성 높음
  - 부드럽게 감소하지 않고 위아래로 요동치며 평균적으로 감소

# 경사하강법 변형

## ❖ 경사하강법




## ❖ 경사하강법의 문제

- 적당한 학습률을 선택하기 어려움
- 학습률 Schedule로 학습률 조정 노력
  - 미리 정해진 스케쥴
- 똑같은 학습률이 모든 파라미터 업데이트에 적용
- Local Minimum의 함정(특히, Saddle Points)