

# 3. 경사하강법

## Gradient Descent

- 정의 및 기능
- 문제점 및 유의점
- 회귀 분석과 경사하강법
- 경사하강법의 유형

## 본 수업의 내용

---

- 경사하강법의 정의와 기능
- 배치 경사하강법
- 확률적 경사하강법
- 미니배치 경사하강법

# 경사하강법 Gradient Descent

---

## ■ 경사하강법:

- 여러 종류의 문제에서 최적의 해법을 찾을 수 있는 일반적인 **최적화(Optimization) 알고리즘**
- **손실(비용) 함수를 최소화**하기 위해 반복해서 파라미터를 조정해가는 것
- 선형 회귀의 경우 손실함수(MSE(평균오차제곱))를 최소화하는 파라미터  **$w_1, w_0$** 에 대해 함수의 현재 기울기(그라디언트)를 계산한 후 기울기가 감소하는 방향으로 진행하고, 기울기가 0이 되면 최솟값에 도달한 것.

## ■ 비유

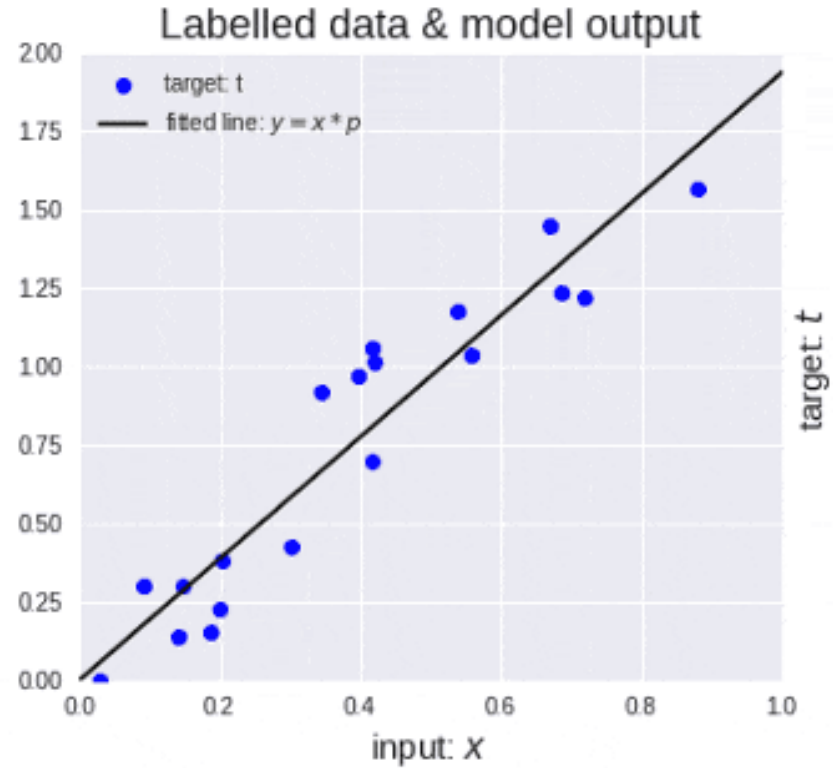
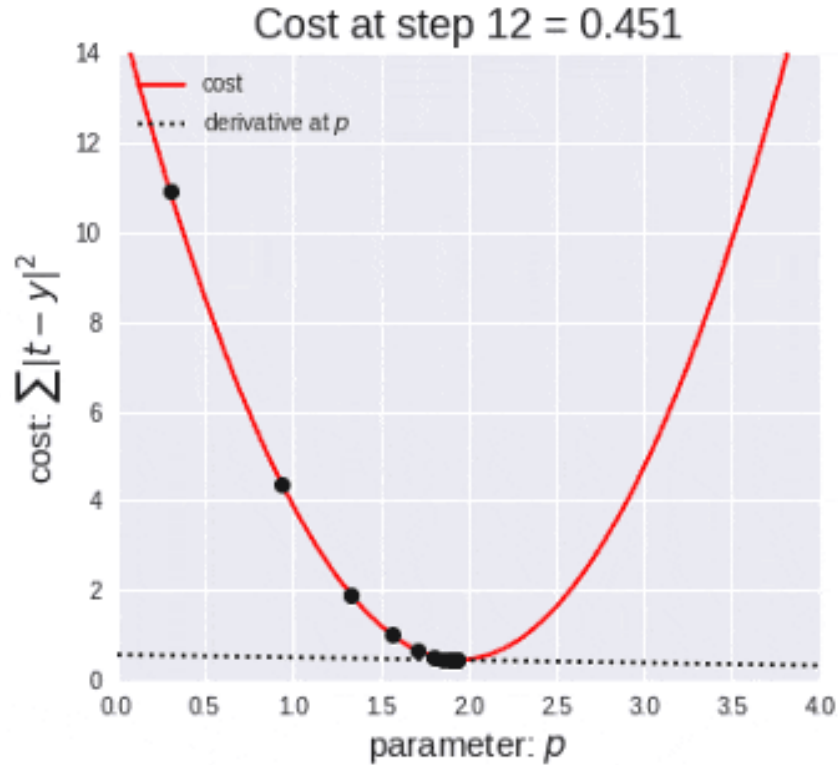
- 앞이 보이지 않는 안개가 낀 산을 내려올 때는 모든 방향으로 산을 더듬어가면서 산의 높이가 가장 낮아지는 방향으로 한 발 씩 내딛어 내려올 수 있다.

## ■ 경사하강법의 장점

- 함수가 너무 복잡해 미분 계수를 구하기 어려운 경우
- 경사하강법을 구현하는게 미분 계수를 구하는 것보다 더 쉬운 경우
- 데이터 양이 너무 많아 효율적인 계산이 필요한 경우
- “데이터를 기반으로 알고리즘이 스스로 학습한다” 는 머신러닝의 개념을 가능하게 해준 핵심 기법 중 하나

# 경사하강법 Gradient Descent

- 경사하강법을 사용하여 비용함수의 최적화 파라미터를 찾고 회귀식에 적용하는 예

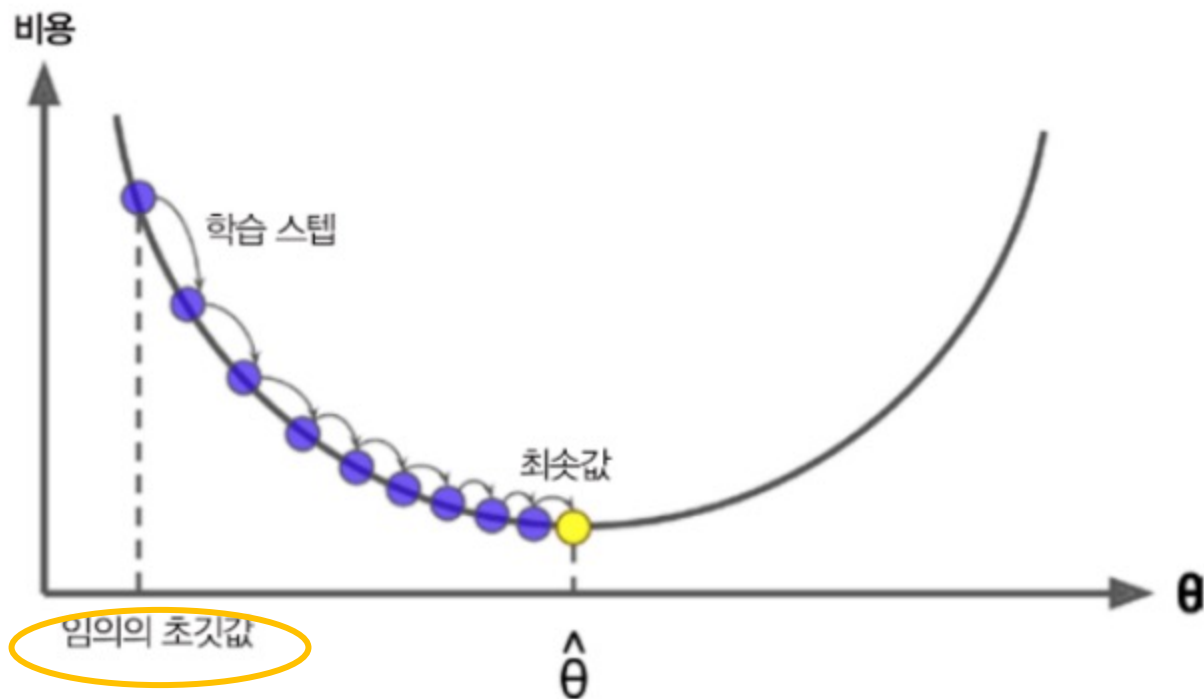


<출처: <https://jonhyuk0922.tistory.com/129>>

# 경사하강법 Gradient Descent

- 학습률(Learning Rate)

- 매 계산(step) 마다 적용되는 이동거리
- 기울기 갱신의 폭 결정
- 하이퍼파라미터로 조절 가능함
- $\theta_{i+1} = \theta_i - \alpha \frac{\partial f}{\partial \theta}(\theta_i)$



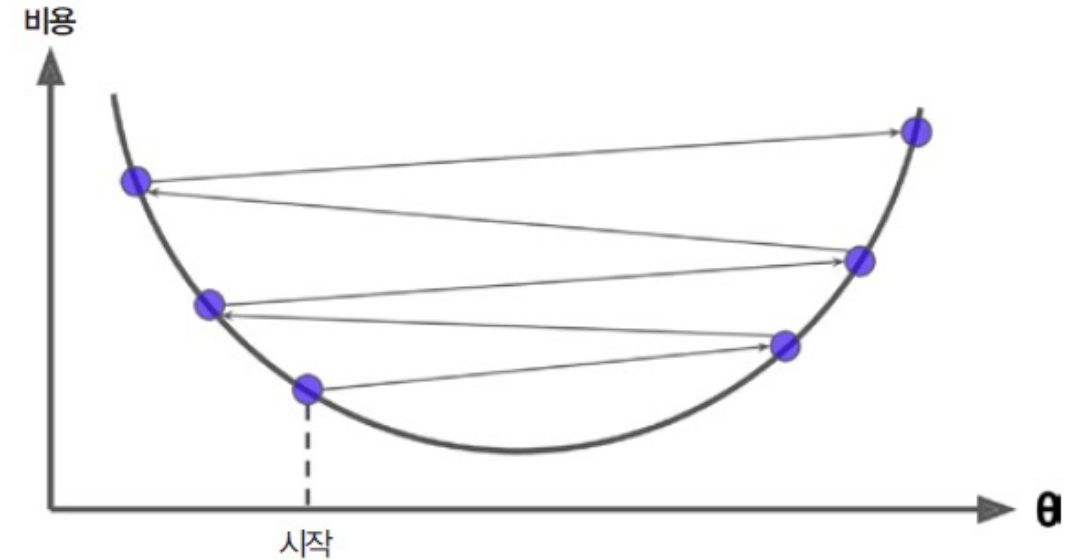
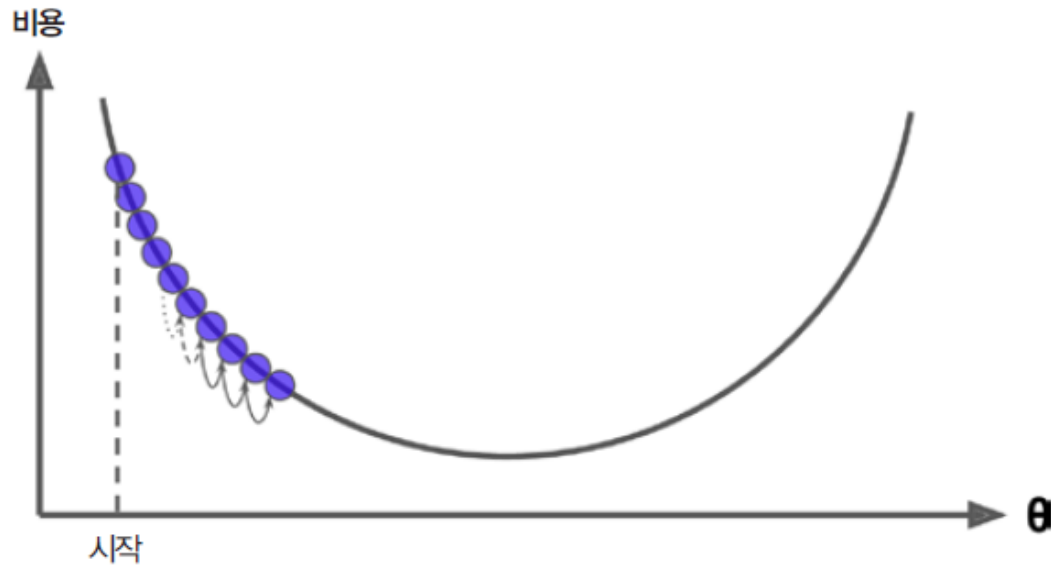
<출처: 핸드온 머신러닝, p165, 한빛미디어>

- ※ 경사하강 코드 구현

- 기울기가 충분히 작아질 때까지 갱신하다가 기울기( $\theta$ )가 평평한 곳에 도달하면 갱신을 종료한다.
- 기울기 갱신 횟수, 기울기 최솟값을 사전에 지정(하이퍼파라미터)

## 경사하강법 Gradient Descent > 학습률

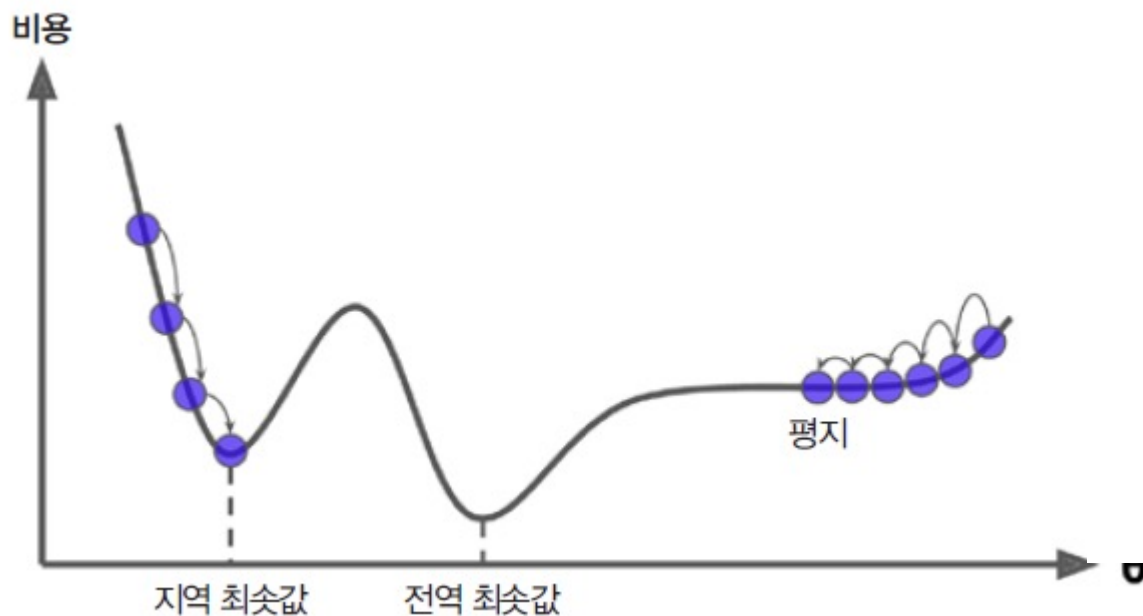
- **학습률(learning rate)**: 매 계산마다 적용되는 이동거리에 해당.
  - 학습률이 너무 작은 경우
  - 학습률이 너무 큰 경우



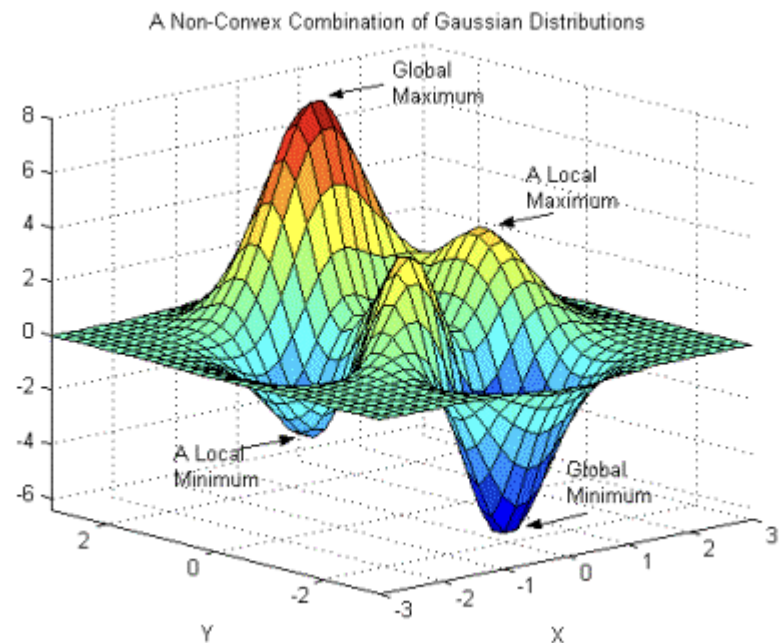
<출처: 핸즈온 머신러닝, p166, 한빛미디어>

## 경사하강법 Gradient Descent > 유의점

- 모든 손실함수가 볼록 함수(Convex) 함수가 아니며, 비볼록 함수일 경우, 최적값(global optimum)을 찾지 못하고, 지역 최적값(local optimum)을 구하게 될 수도 있음.



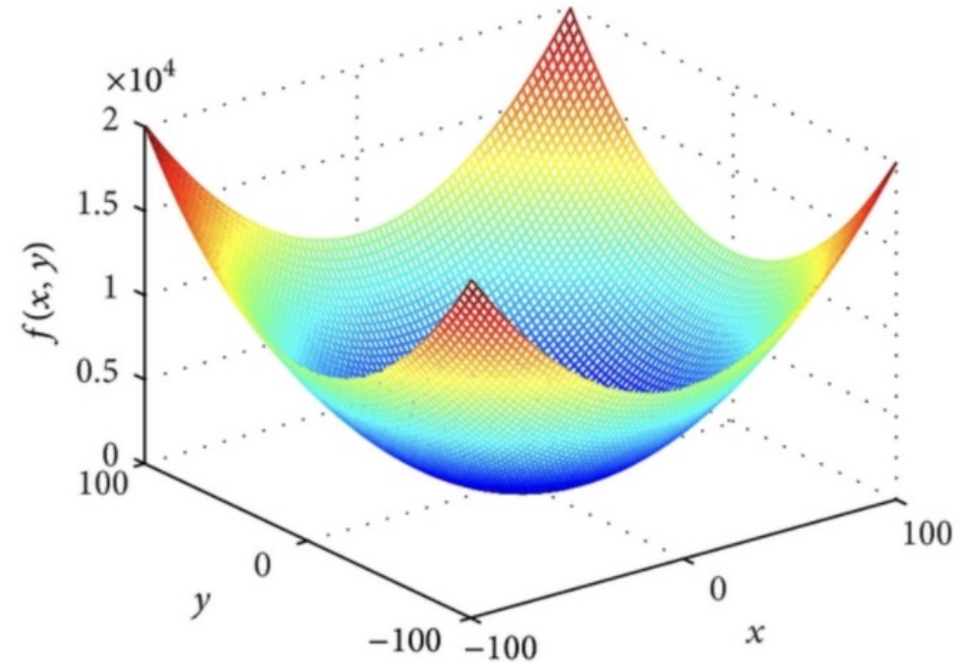
<출처: 핸드온 머신러닝, p167, 한빛미디어>



<출처: [https://commons.wikimedia.org/wiki/File:Non-Convex\\_Objective\\_Function.gif](https://commons.wikimedia.org/wiki/File:Non-Convex_Objective_Function.gif) >

## 경사하강법 > 선형회귀 손실함수

- 선형 회귀를 위한 MSE 손실(비용)함수는 볼록 함수(Convex) 함수이다
- 지역 최솟값이 없고, 하나의 전역 최솟값만 있다
- 연속된 함수이고 기울기가 갑자기 변하지 않는다.
- 반드시 경사하강법을 통해 전역 최솟값에 도달할 수 있다



[https://www.researchgate.net/figure/Sphere-function-D-2\\_fig8\\_275069197](https://www.researchgate.net/figure/Sphere-function-D-2_fig8_275069197)



# 경사 하강법과 최소제곱법과의 차이

---

## 경사 하강법과의 비교

- 경사 하강법과 최소 제곱법 모두 예측 알고리즘에 해당한다.
- 경사 하강법이 수학적 최적화 알고리즘으로서 적절한 학습비율(learning rate)를 설정해야 하고 많은 연산량이 필요하지만 정규방정식에는 그와 같은 단점이 없다는 장점이 있다.
- 정규방정식은 행렬 연산에 기반하기 때문에 특성의 개수가 엄청나게 많을 경우 연산이 느려지는 것을 피할 수 없다.
- 경사 하강법은 아무리 많은 특성이 존재하더라도 일정한 시간 내에 해법을 찾는 것이 가능하다.
- 그러므로 예측 알고리즘을 선택할 때 있어 특성의 개수에 따라 알맞은 것을 선택하여야 한다.

## 경사하강법 Gradient Descent > 배치 batch와 에폭 epoch

---

### ■ 배치 batch:

- 1회의 경사 업데이트(스텝)에 사용되는 데이터 집합
- 이 때 사용되는 데이터 집합의 개수를 배치 크기(배치 사이즈)라고 함
- 예) 전체 데이터 세트 : 100개, 배치 사이즈 : 20
  - 배치 개수: 5, 경사(기울기) 업데이트 수: 5회

### ■ 에폭 epoch:

- 전체 데이터들을 한 번 모두 사용하는 것
- 일반적으로 경사하강법은 수십, 수백 번 이상 에폭을 수행
- 예) 전체 데이터 세트: 100개, 배치 사이즈 : 20, 에폭: 1000
  - 에폭 1회 -> 배치 개수: 5, 경사(기울기) 업데이트 수 : 5회
  - 에폭 1000회 -> 경사 업데이트 수(학습횟수): 5000

## 경사하강법 Gradient Descent > 종류

---

### 1. 배치 경사하강법(batch gradient descent)(full gradient descent)

- 매 스텝(1회의 기울기 갱신, 가중치 갱신)에서 훈련 데이터 전체를 사용(배치의 크기: 훈련 데이터 전부)하여 계산한 후 최적의 스텝을 나아감.(Full gradient descent)
- 매우 큰 훈련 세트에서는 아주 느림
- 한 개의 배치에 전체 학습 데이터가 모두 들어간다

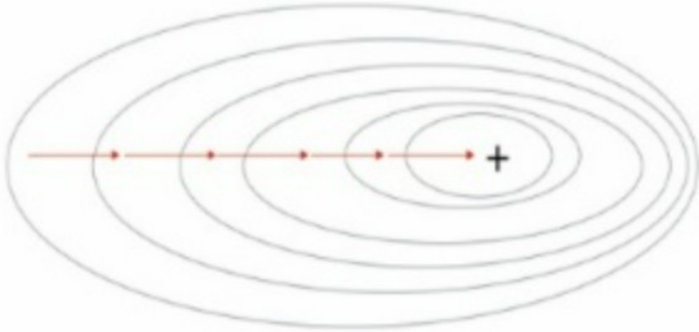
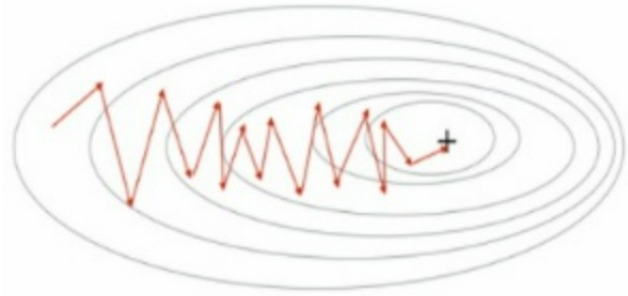
### 2. 확률적 배치 경사하강법(SGD:stochastic gradient descent)

- 매 스텝에서 한 개의 샘플(훈련 데이터)을 무작위로 선택하고 그 하나의 샘플에 대한 그래디언트(기울기)를 계산함
- 한 개의 배치에 임의의 학습 데이터 1개만 들어간다
- 매 스텝에서 다루어야 할 데이터가 매우 작으므로(배치의 크기 1) 처리 속도가 빠름
- 빠르게 최적점을 찾을 수 있지만 정확도는 낮아짐
- 매우 큰 훈련 데이터도 학습 가능함
- 데이터가 들어오는 즉시 가중치를 갱신할 수 있어 온라인 학습이 가능하며 즉각적인 시스템 대응이 가능

## 3. 미니 배치mini-batch 경사하강법

- 확률적 경사하강법 및 배치 경사하강법의 절충안
- 각 스텝에서 전체 훈련 세트(배치 경사하강법)이나 하나의 샘플(확률적 경사 하강법)을 기반으로 기울기를 계산하지 않고, 미니 배치라고 부르는 임의의 샘플 세트를 기반으로 기울기를 계산함.
- 1개의 배치에 임의의 학습 데이터 여러 개가 들어감
- 배치 경사하강법보다 효율적이고, 확률적 경사하강법 보다 노이즈가 적다.
- 행렬 연산에 최적화된 하드웨어(GPU를 사용하는 경우)에서 성능이 향상됨.

## 배치 경사하강법과 확률적 경사하강법 비교

	경사 하강법	확률적 경사하강법
1회학습에 사용되는 데이터(배치크기)	모든 훈련 데이터	랜덤 추출 1개 데이터
반복에 따른 정확도	학습이 반복될수록 최적값에 근접	학습이 반복될 수록 최적값에 근접
노이즈	거의 없음	비교적 노이즈 심함
		

---

<실습>

사이킷런 SGDRegressor를 사용하여 머신러닝 구현

## [응용] 보스턴 집값 예측

---

1. OLS 방식으로 보스턴 집값 예측 (RM VS Price)
  - `LinearRegression()`
2. `SGDRegressor` with default hyperparameter
3. `SGDRegressor` with hyperparameter tuning
4. `SGDRegressor` with scaling
5. `SGDRegressor` with `StandardScaler()`
6. Pipeline with `StandardScaler`, `LinearRegressor`, `SGDRegressor`

## [참고] 특성 스케일링feature Scaling

---

데이터 변환 중 가장 중요한 변환 중 하나

대부분의 머신러닝 알고리즘은 입력 숫자 특성들의 스케일이 많이 다르면 잘 작동하지 않음

(Decision Tree 예외)

### [정규화Normalization]

- 모든 값이 0 ~ 1 사이에 들도록 **범위를 조정**(feature\_range로 조정 가능)
- sklearn.preprocessing.MinMaxScaler

### [표준화Standardization]

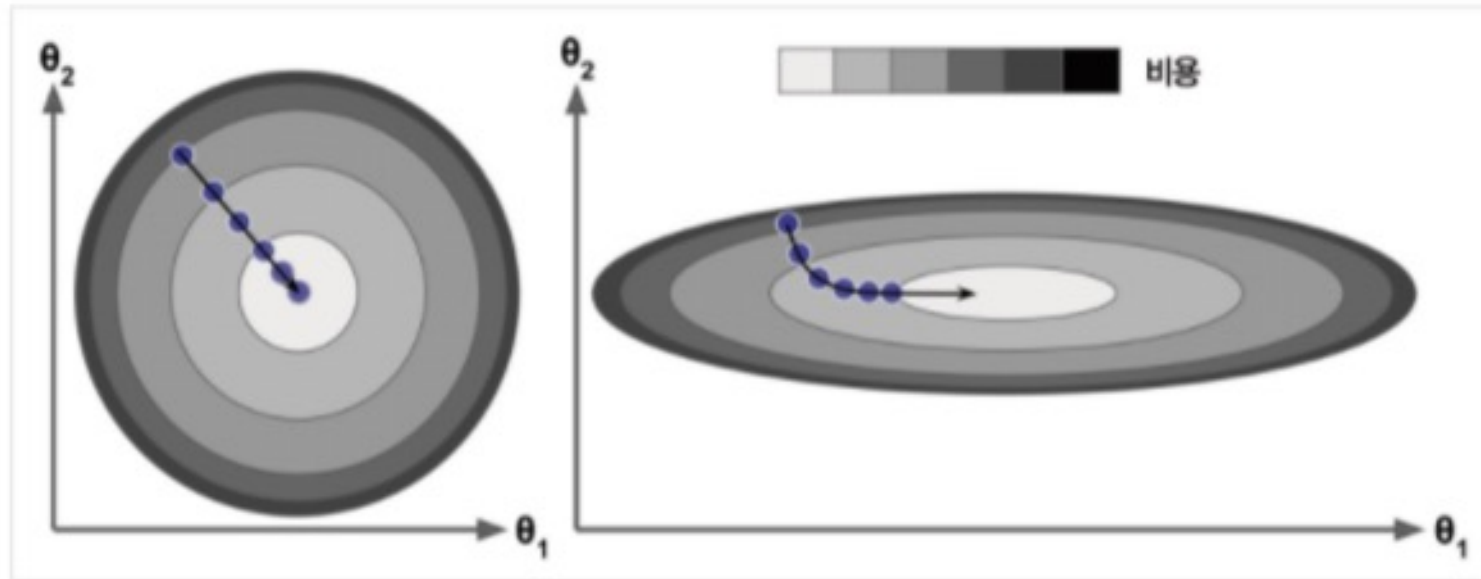
- 평균을 뺀 후 표준편차로 나누어 평균 0, 분산 1이 되는 분포로 전환
- 각 특성값이 0에서 표준 편차의 몇 배만큼 떨어져 있는가
- Min-max 스케일링과 달리 표준화는 범위의 상한과 하한이 없음
- 신경망의 경우 입력값의 범위를 0~1로 기대함
- 표준화는 이상치에 영향을 덜 받음(vs. min-max 스케일링)
- sklearn.preprocessing.StandardScaler

※ 타겟 스케일링은 필요하지 않음.



## 경사하강법 Gradient Descent > 유의 사항

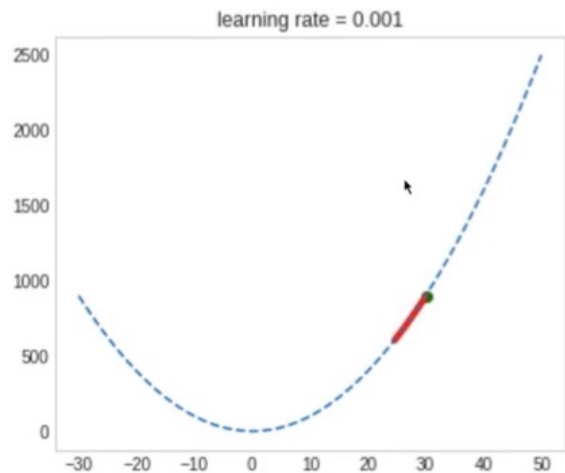
경사하강법 유의점 2) : 특성 스케일 고려



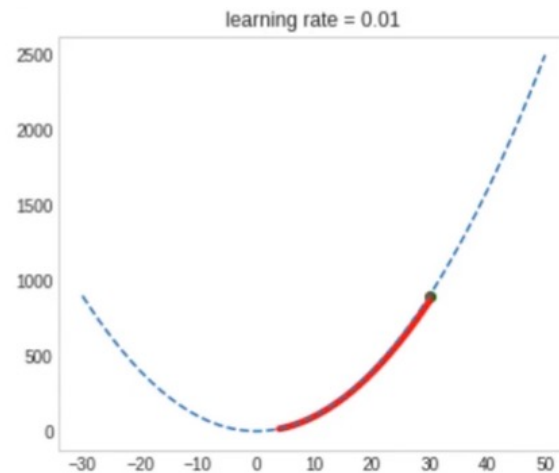
<출처: 핸드온 머신러닝, p167, 한빛미디어>

# 학습률의 변화 VS 최솟값 확인

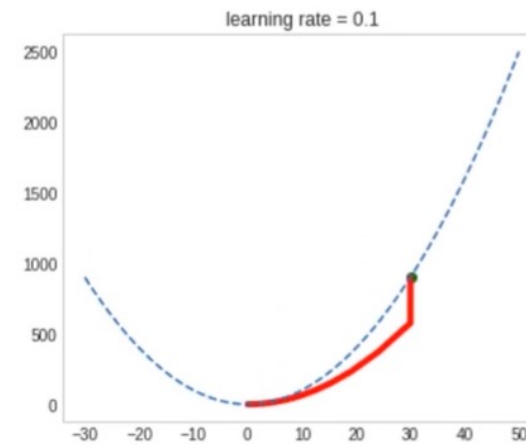
- 설정한 학습률(LR): [0.001, 0.01, 0.1, 1.01]
- epoch(iteration) : 100



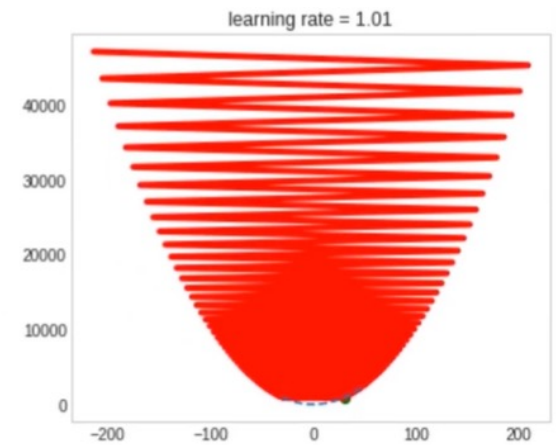
LR=0.001



LR=0.01



LR=0.1



LR=1.01

출처: <https://www.youtube.com/watch?v=GwR1ivsUiAI&t=290s>