

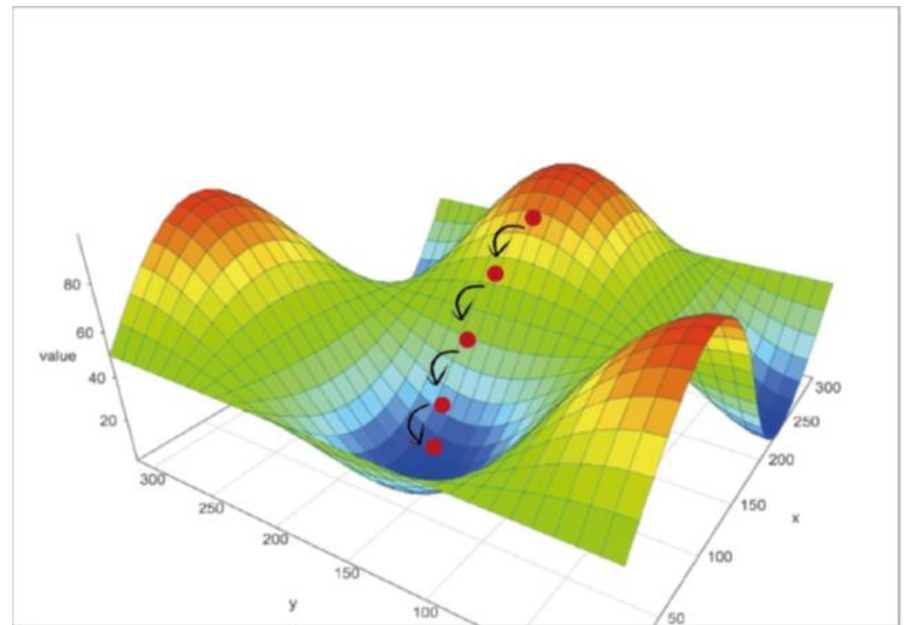
A rustic kitchen scene with a wooden shelf holding various teapots and cups. A framed picture hangs on the wall. The text 'Machine Learning' is overlaid in white, and 'Gradient Descent' is overlaid in yellow.

Machine Learning

Gradient Descent

김선녕(ksycafe@gmail.com)

- 최적화(optimization)
 - x 의 값을 바꾸어 가면서 어떤 함수 $f(x)$ 의 값을 최대화 또는 최소화 하는 것
- 최소화 또는 최대화 함수
 - 목적함수(*objective function*) 또는 판정기준(*criterion*)
 - 최소화 : 비용함수(*cost function*), 손실함수(*loss function*), 오차함수(*error function*)
- 경사 하강법(Gradient descent)
 - 배치(*batch*) 경사하강법
 - 확률적(*stochastic*)경사하강법
 - 미니배치(*mini - batch*) 경사하강법



$$\hat{y}(x) = wx + b$$

$$cost(w, b) = \frac{1}{n} \sum_{i=1}^n (\hat{y}(x_i) - y_i)^2 = \frac{1}{n} \sum_{i=1}^n (wx_i + b - y_i)^2$$

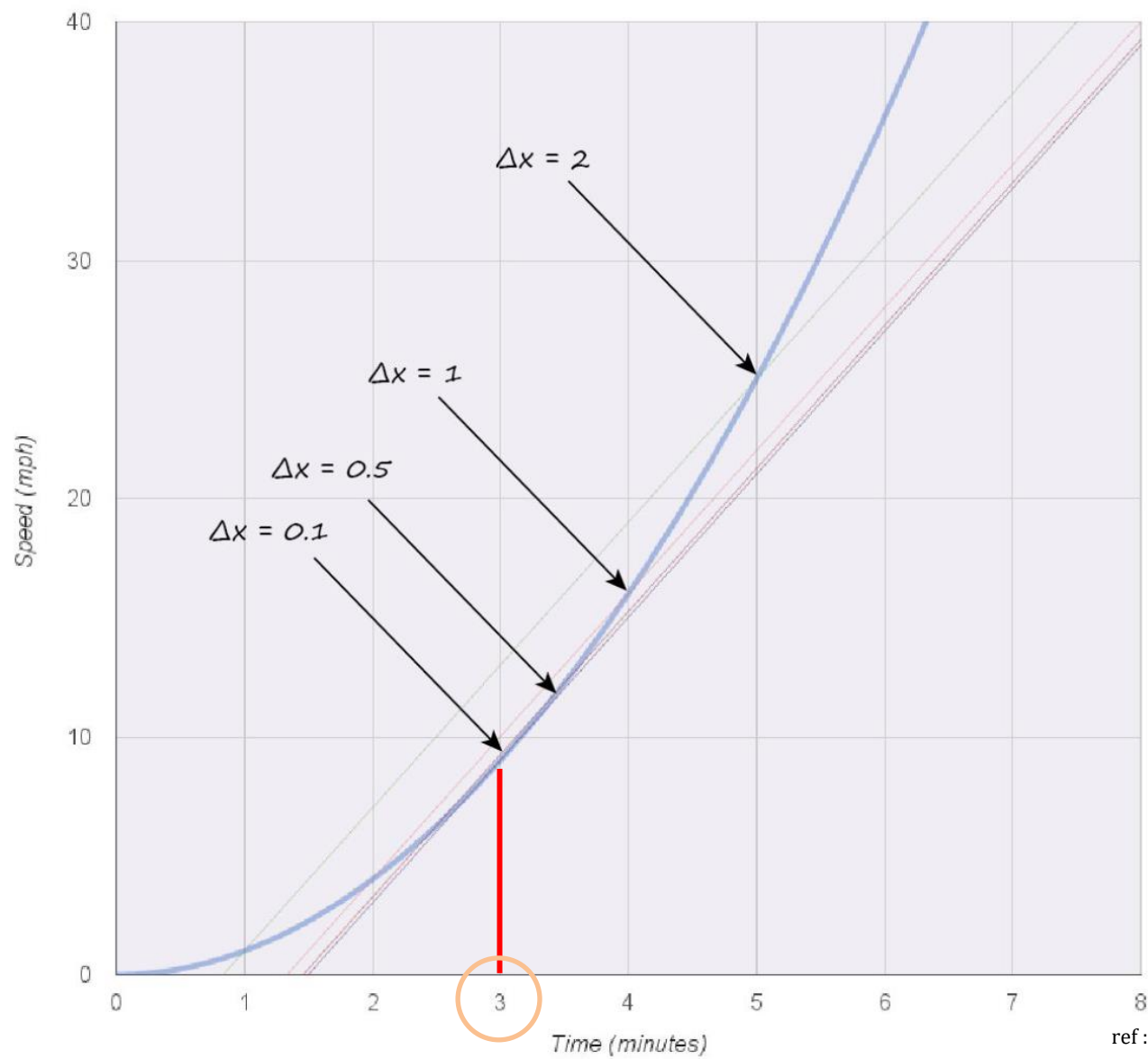
가장 작은 w, b 의 값을 구하려면(비용의 최소화): $\underset{w, b}{\text{minimize}} cost(w, b)$

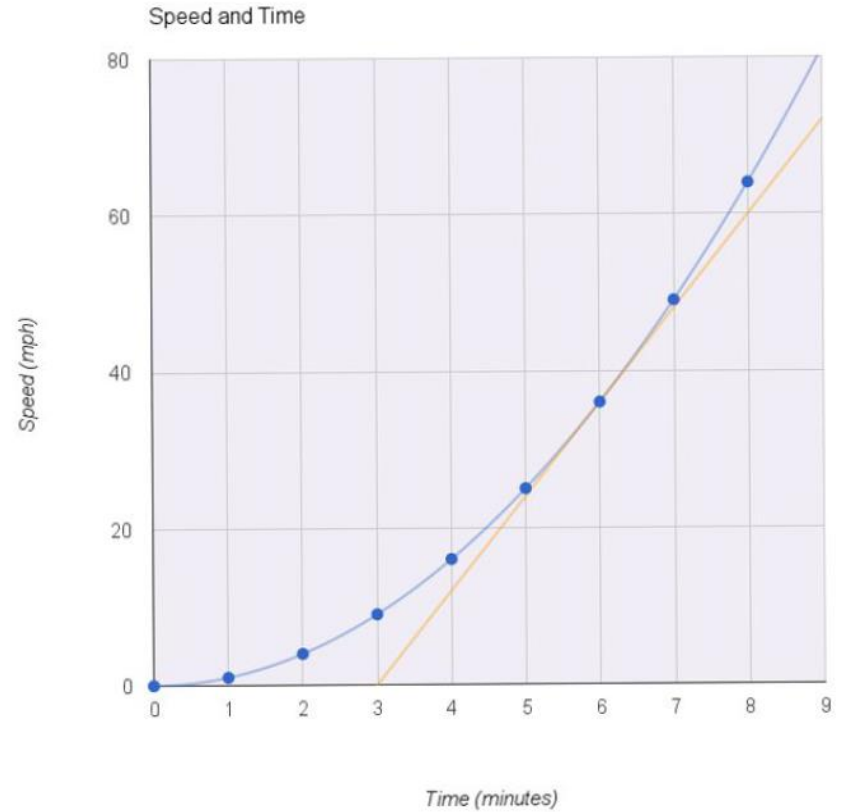
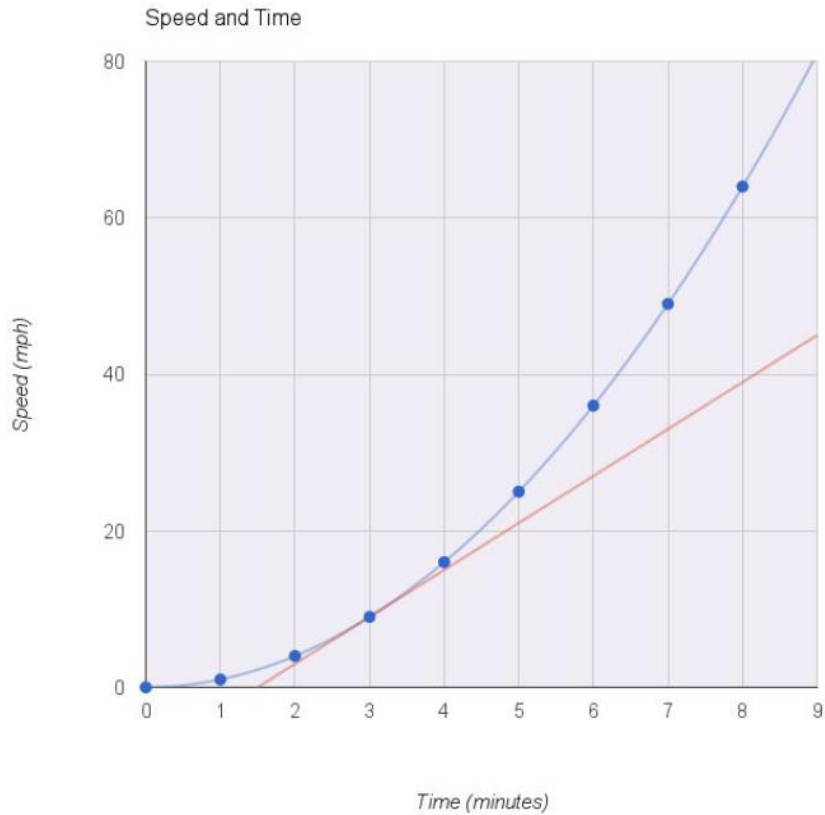
$$\frac{\partial}{\partial w} cost(w) = \frac{2}{n} \sum_{i=1}^n (wx_i - y_i) \cdot x_i$$

$$w := w - \alpha \sum_{i=1}^n (wx_i - y_i) \cdot x_i$$

※ α : learning rate

Speed and Time



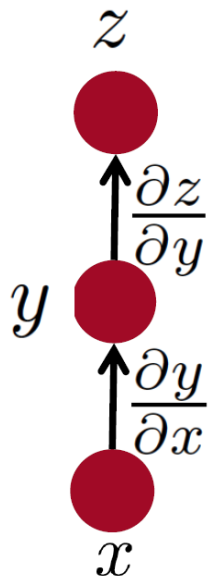


$$y' = f'(x) = \frac{dy}{dx} = \frac{d}{dx} y = \frac{df(x)}{dx} = \frac{d}{dx} f(x) = \frac{\partial f}{\partial x} = \frac{\delta f}{\delta x}$$

합성함수(Chain Rule)의 미분

6

Simple chain rule

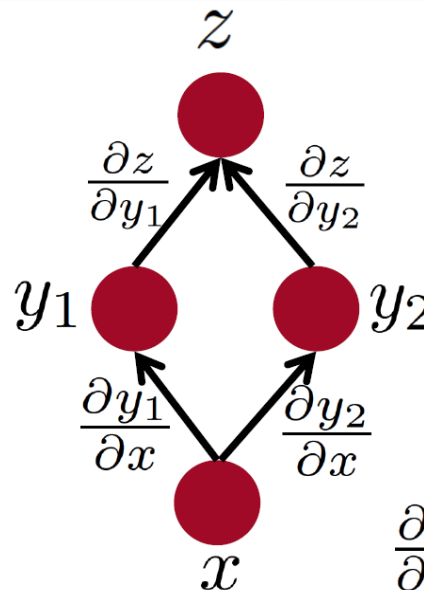


$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x}$$

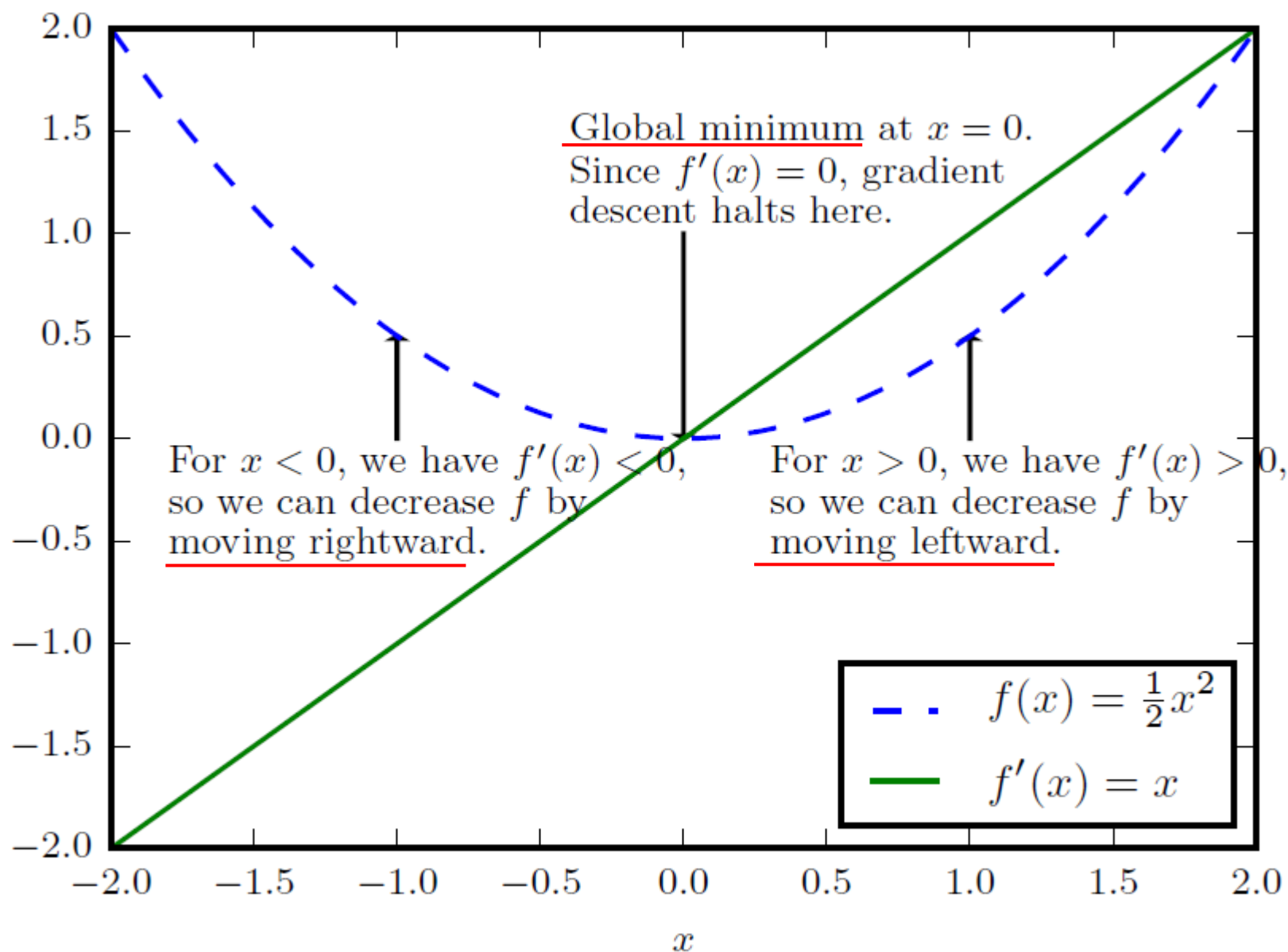
$$y = f(u), \quad u = g(v), \quad v = h(x)$$

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dv} \cdot \frac{dv}{dx}$$

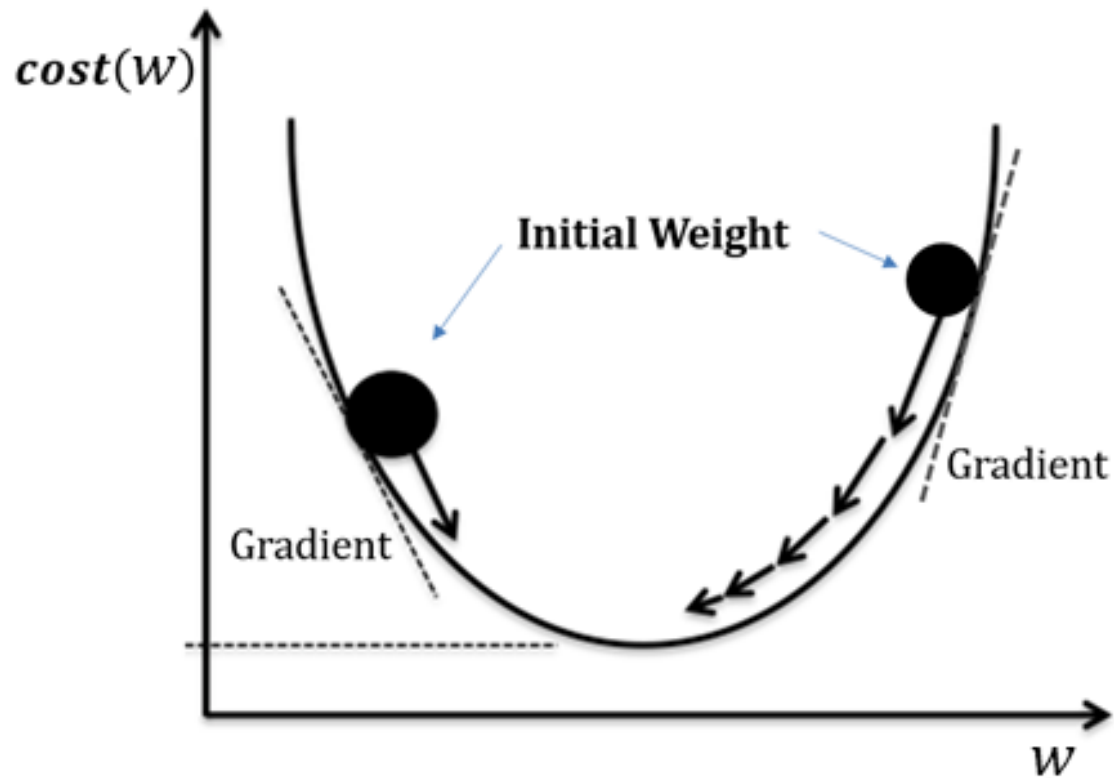
Multiple path chain rule

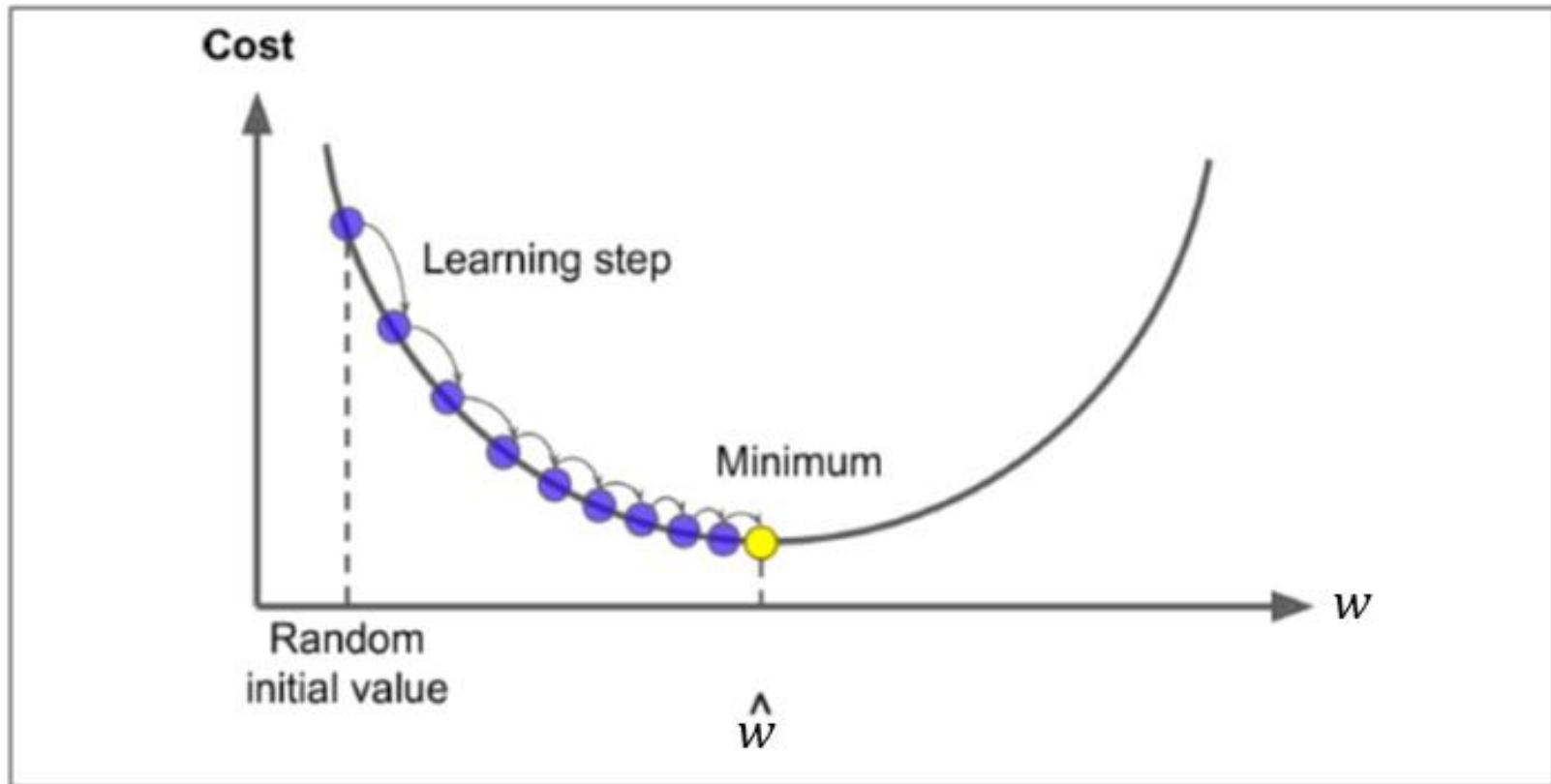


$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1} \frac{\partial y_1}{\partial x} + \frac{\partial z}{\partial y_2} \frac{\partial y_2}{\partial x}$$



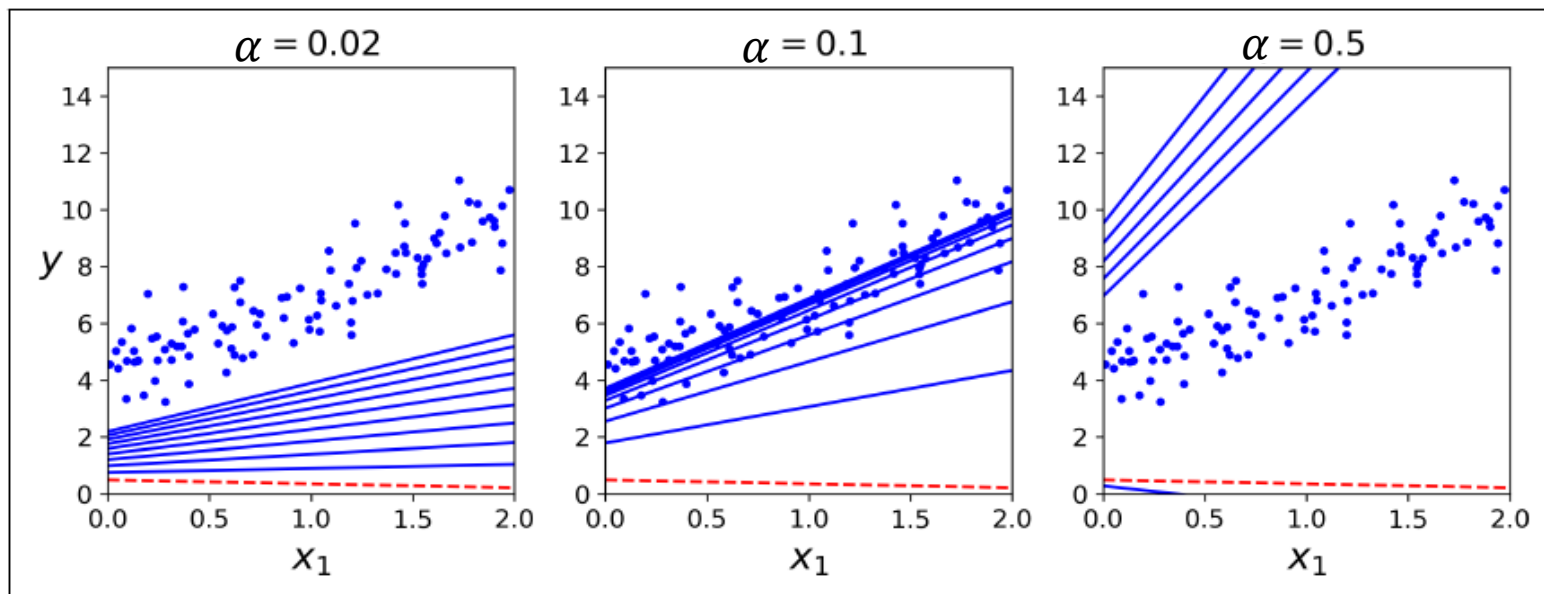
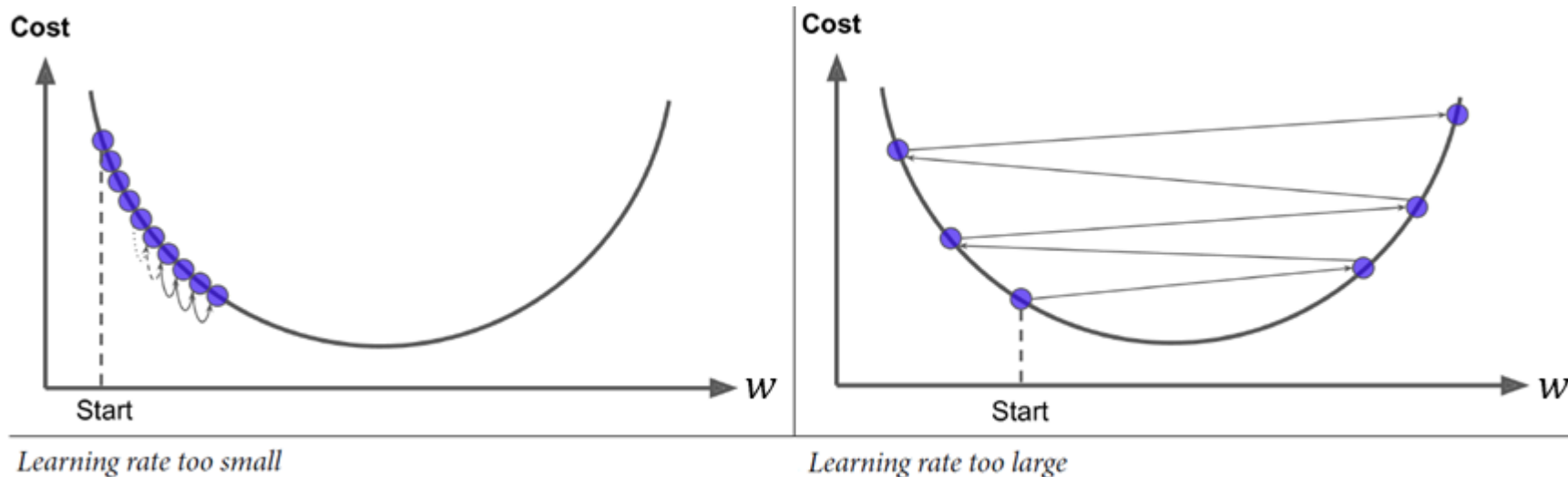
$$w := w - \alpha \sum_{i=1}^n (wx_i - y_i) \cdot x_i$$





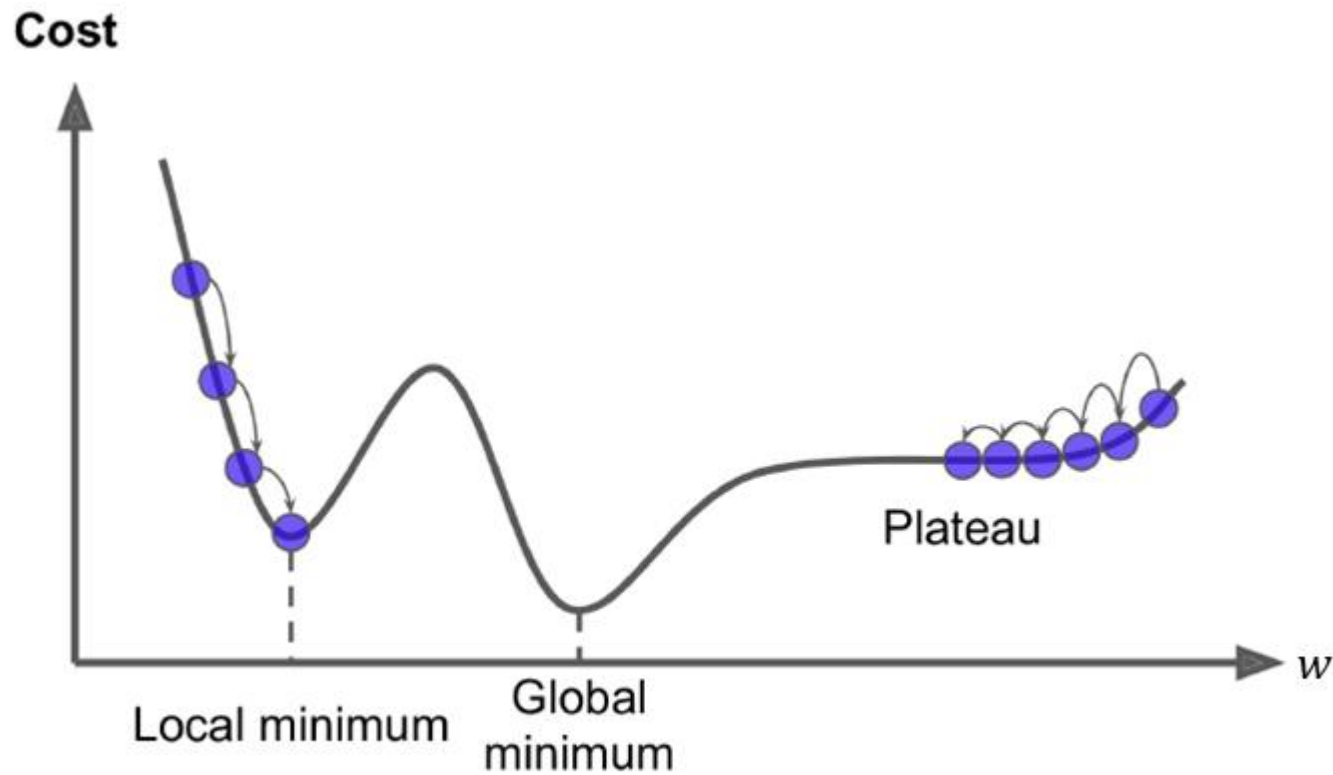
Learning rate

10



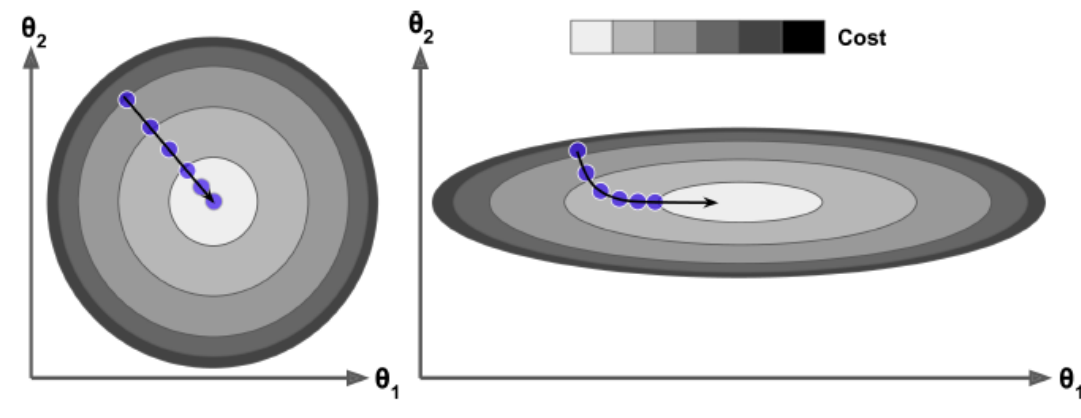
Local minimum vs Global minimum

11



Gradient Descent pitfalls

- 특성1(*feature1*)과 특성 2(*feature2*)의 스케일의 크기의 차이
 - 최소값에 도달하는 시간
- 경사 하강법을 사용할 때는 데이터의 모든 특성이 같은 스케일을 갖도록 처리
 - (예) 사이킷런의 StandardScaler를 사용

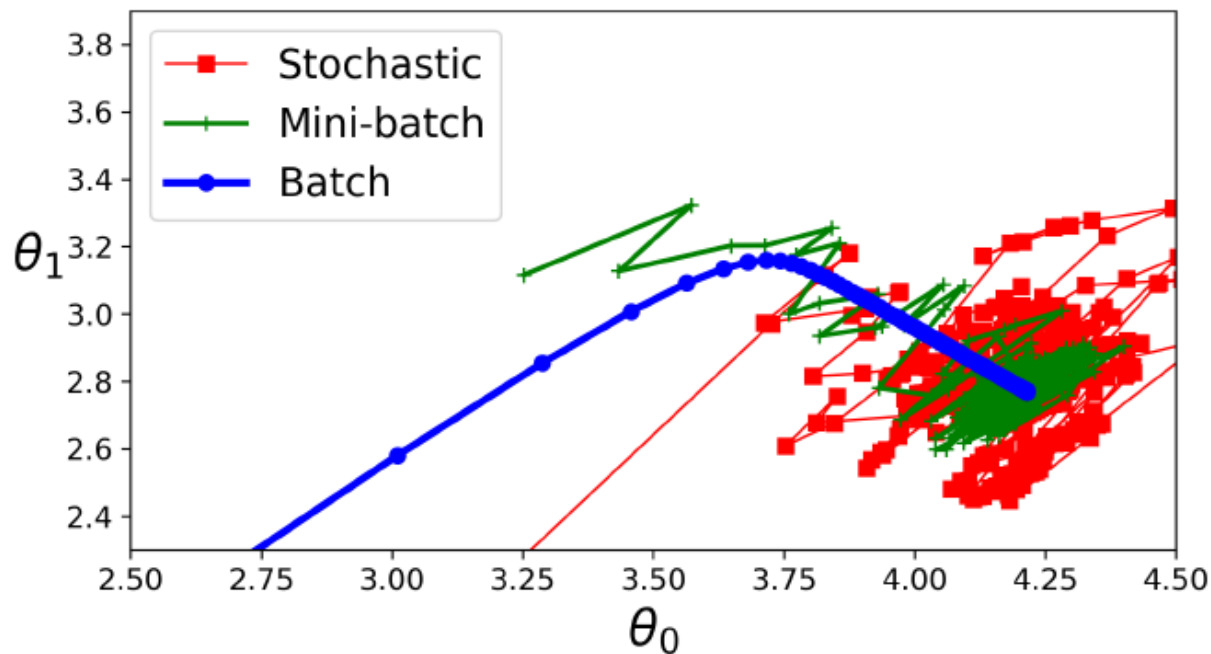


Gradient Descent with and without feature scaling

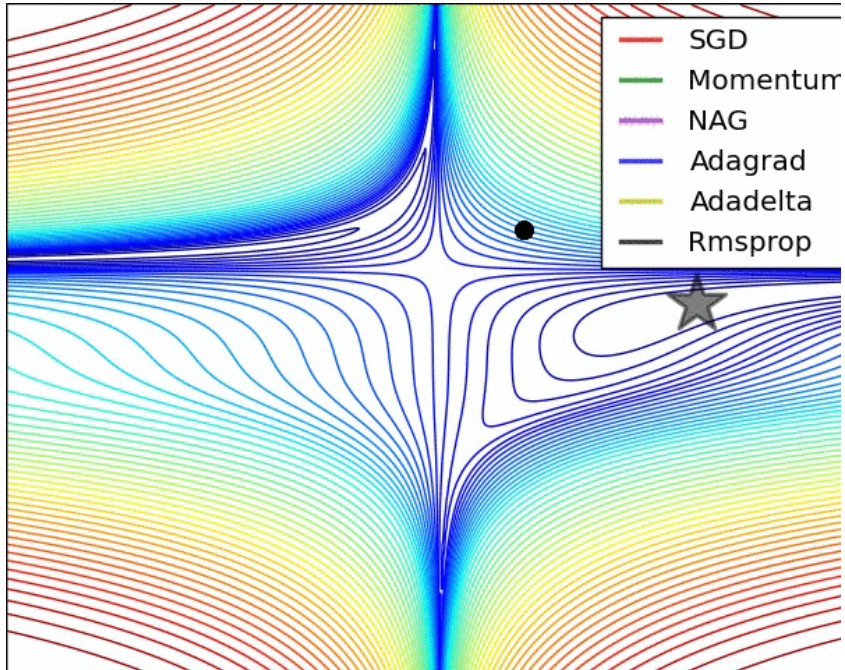
- one **epoch** : one forward pass and one backward pass of all the training examples
- **batch size** : the number of training examples in one forward/backward pass. The higher the batch size, the more memory space you'll need.
- number of **iterations** : number of passes, each pass using [batch size]number of examples.
To be clear, one pass = one forward pass + one backward pass
 - we do not count the forward pass and backward pass as two different passes.
- Example
 - if you have 1000 training examples, and your batch size is 500, then it will take 2 iterations to complete 1 epoch

- 배치 경사하강법(*batch gradient descent*)
 - 전체 훈련 데이터셋을 사용해 매 스텝에서 사용
 - 큰 훈련 데이터 셋에서 속도가 느림(비용이 많이 든다)
- 확률적 경사하강법(*stochastic gradient descent, SGD*)
 - 일정량의 훈련 데이터셋을 무작위로 선택하여 매 스텝에서 사용
 - 데이터량이 적기 때문에 속도가 빠름
 - 새로운 데이터가 도착하는 대로 훈련(온라인 학습으로 사용)
 - 최소값에 다다를 때까지 불규칙하여 Local minimum을 건너뛰도록 도와주게 되어 *Global minimum*을 찾을 가능성이 높다
- 미니배치 경사하강법(*mini – batch gradient descent*)
 - 배치 경사하강법과 확률적 경사하강법의 절충
 - 임의의 작은 훈련데이터셋을 선택하여 매 스텝에서 사용
 - 배치를 어느정도 크게 하면 SGD보다 덜 불규칙하지만 *Local minimum*에서 빠져나 오기는 더 힘들 수 있다.

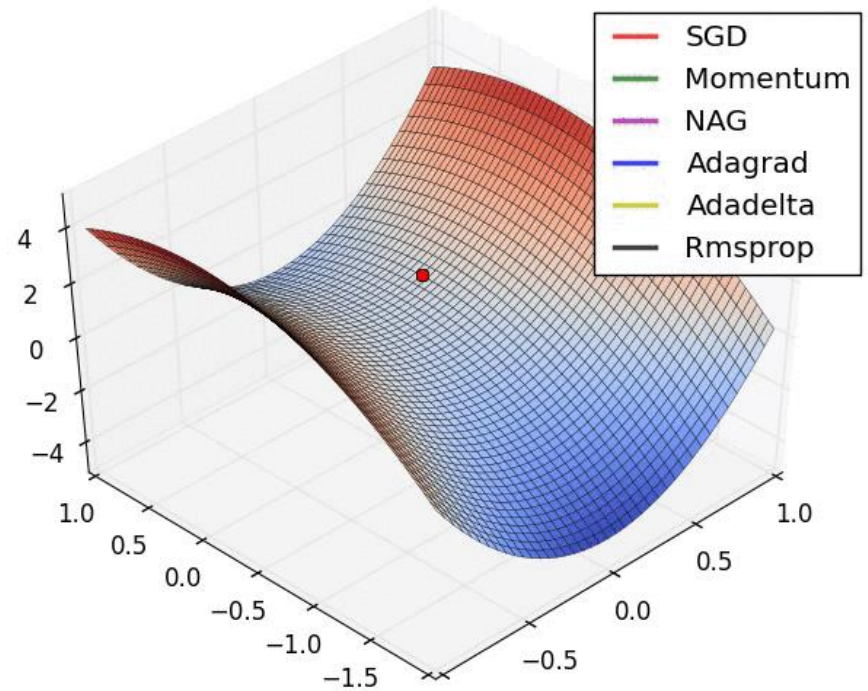
- 전부 최소값 근처에 도달
- 배치 경사하강법의 경로가 실제로 최솟값에 멈춘 반면(많은 시간 소요)
- 확률적/미니배치 경사하강법은 근처에서 맴돌고 있다



- *Momentum*
 - Gradient Descent를 통해 이동하는 과정에서 일종의 관성(momentum)값을 더해 주는 것
- *NAG(Nesterov Accelerated Gradient)*
 - Momentum 값이 적용된 지점에서 gradient값이 계산
 - 한 단계를 미리 예측함으로써 불필요한 이동을 줄인다.
- *Adagrad(Adaptive Gradient)*
 - SGD를 개선한 형태
 - 변수의 업데이트 횟수에 따라 학습률(Learning rate)를 조절하는 옵션이 추가된 방법
- *RMSProp*
 - AdaGrad를 개선한 형태
- *AdaDelta(Adaptive Delta)*
 - RMSProp과 유사하게 AdaGrad의 단점을 보완하기 위해 제안된 방법
- *Adam(Adaptive Moment Estimation)*
 - Momentum/AdaGrad/RMS prop의 장점만을 모아 하나로 결합한 형태



SGD optimization on loss surface contours



SGD optimization on saddle point