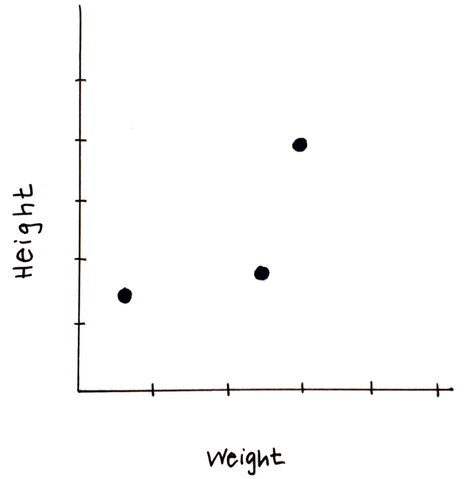
Stochastic Gradient Descent

박달님 04.01.2021

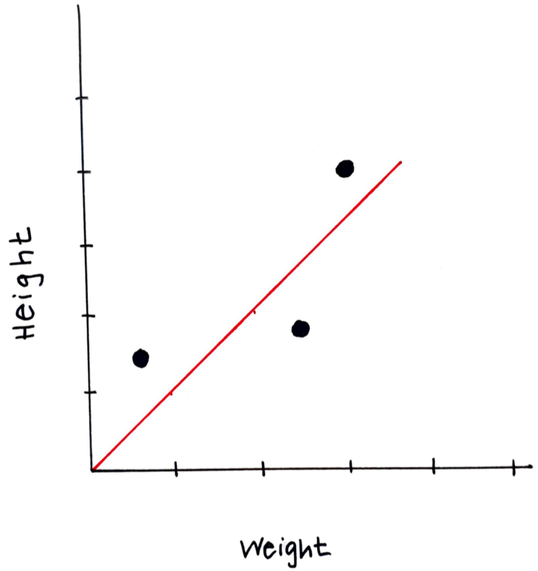
Gradient Descent

Predicted Height = intercept + slope x **Weight**

목표: 절편 및 기울기에 대한 최적 값 찾기



Ex 1) 만약 절편 = 0, 기울기 = 1이면
몸무게를 이용해서 키를 추정할 수 있다



sum of the squared residuals를 **Loss Function** 으로 이용해서 초기 선이 데이터에 얼마나 잘 맞는지 확인한다.

Sum of squared residuals = **(Observed Height - Predicted Height)^2**

절편과 기울기에 대한 최적값을 찾기 위해 Predicted Height 값을 sum of the squared residuals 식에 넣는다.

Sum of squared residuals = (**Observed Height** - (intercept + slope x **Weight**))^2
d/(d intercept) Sum of squared residuals = -2(Height - (intercept + slope x Weight))
그런 다음 절편에 대한 sum of the squared residuals에 미분을 취한다.

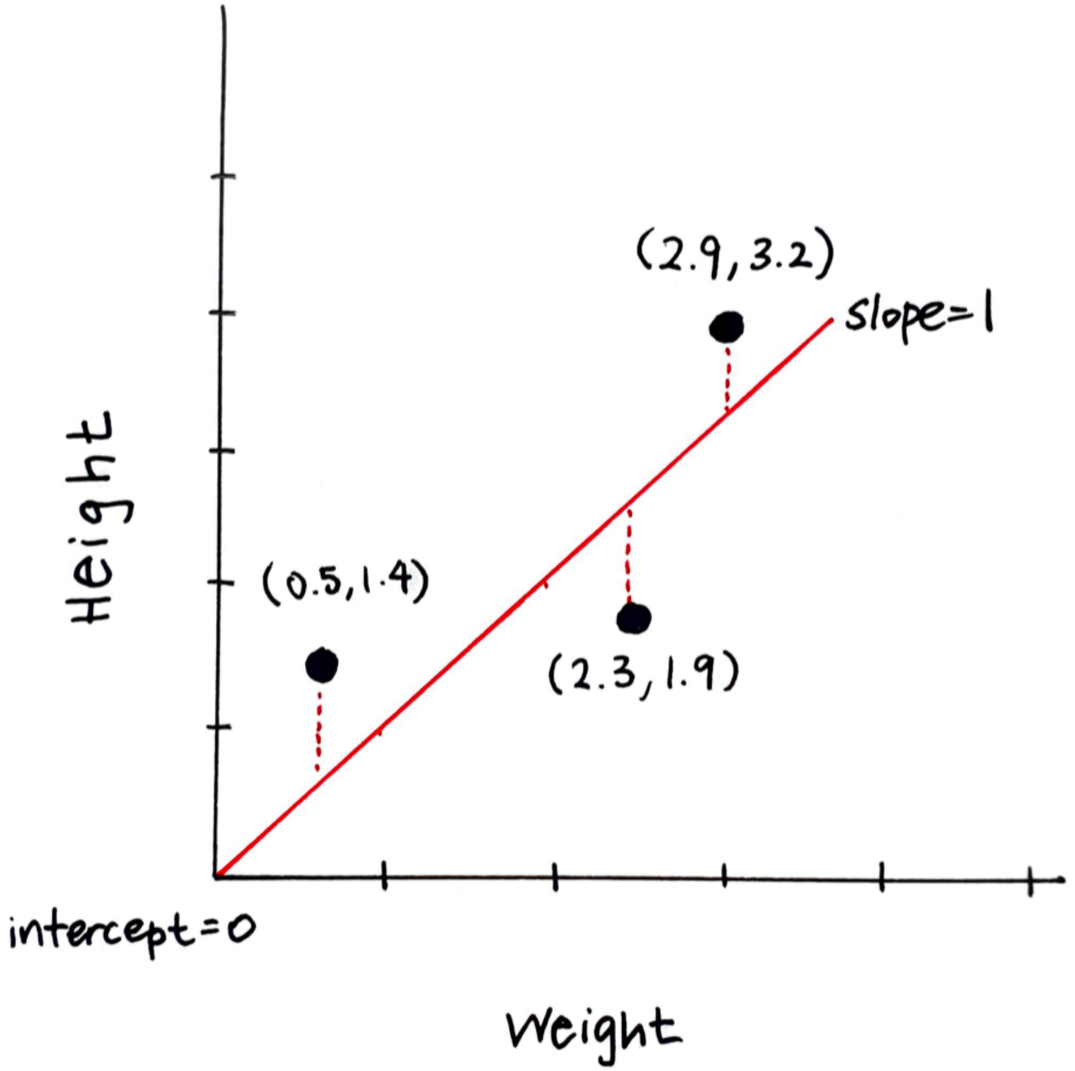
slope에 관련하여:
d/d slope Sum of squared residuals = -2 x Weight(Height - (intercept + slope x Weight))

Observed 데이터의 값을 slope에 대한 미분, 절편에 대한 initial guess = 0, 기울기에 대한 initial guess = 1을 넣는다.

d/d intercept Sum of squared residuals =
-2(1.4 - (0 + 1 x 0.5)
+ -2(1.9 - (0 + 1 x 2.3))
+ -2(3.2 - (0 + 1 x 2.9)) = **-1.6**

그리고 기울기에 대한 미분에 대해서도 똑같이 넣어준다.

d/d slope Sum of squared residuals =
-2 x 0.5(1.4 - (0 + 1 x 0.5))
+ -2 x 2.9(3.2 - (0 + 1 x 2.9))
+ + -2 x 2.3(1.9 - (0 + 1 x 2.3)) = **-0.8**



Plug in the **slopes** into the **step size** formulas.

Step Size intercept = -1.6 x **Learning Rate**
Step Size slope = -0.8 x **Learning Rate**

Learning Rate만큼 곱해준다. 처음은 0.01

Step Size intercept = -1.6 x 0.01 = -0.016
Step Size slope = -0.8 x 0.01 = -0.008

Old intercept와 Old Slope을 이용하여 New intercept와 New slope을 계산한다.
New Intercept = Old Intercept - Step Size = 0 - (-0.016) **= 0.016**
New Slope = Old Slope - Step Size = 1 - (-0.008) **= 1.008**

다시 미분으로 돌아가서 최대 number of steps를 얻거나 steps가 매우 작아질때까지 이 과정을 반복한다.

Stochastic Gradient Descent

지금은 절편과 기울기에 대한 각 계산을 하는데 3개의 항목만 있어서 수학이 간단했지만 만약에 누가 질병에 걸릴지 예측하기 위해 23,000개의 유전자를 사용하여 계산하는 회귀 모델이 있다면 수학이 너무 복잡해질 것이다.

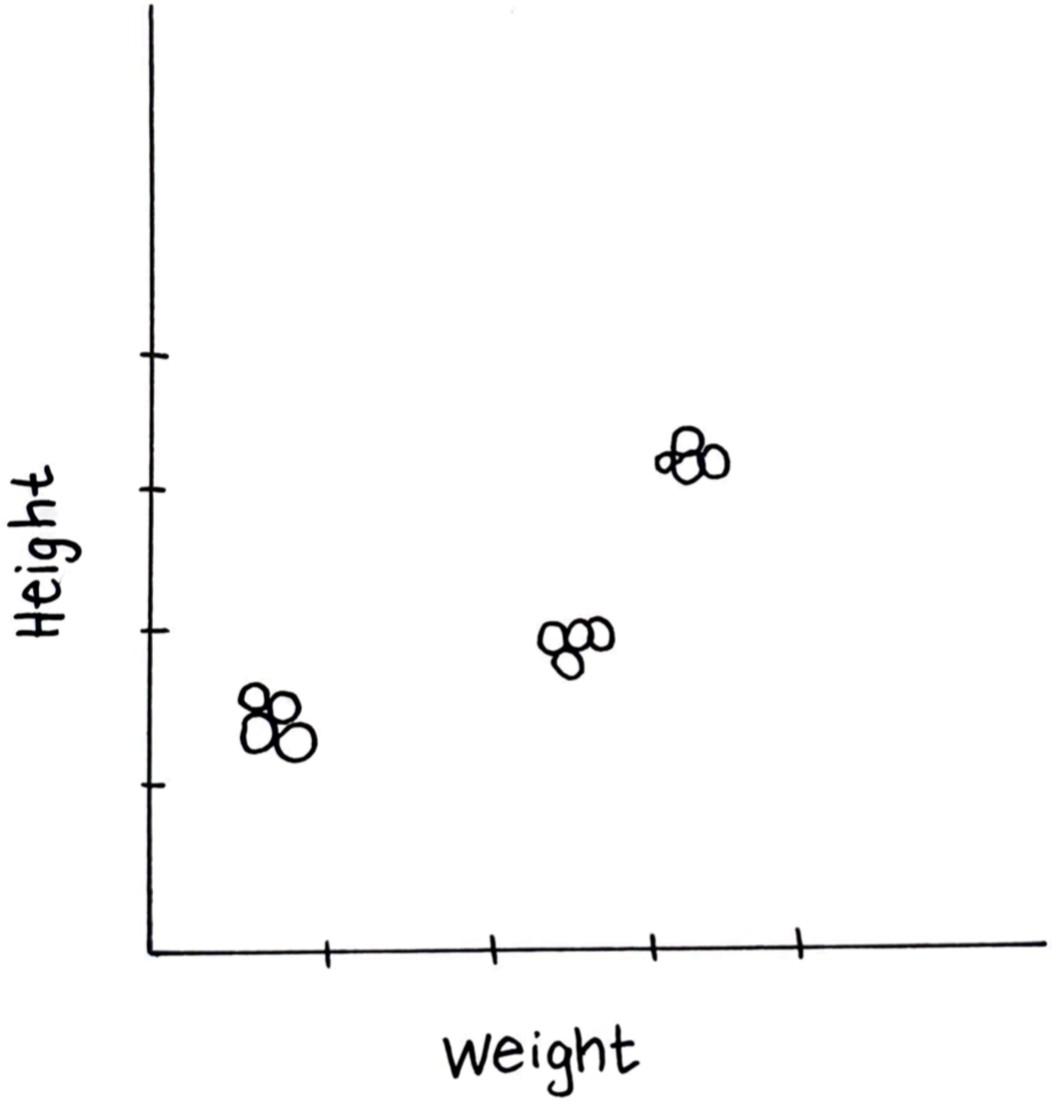
Ex) 23,000 항목, 각 1,000,000개의 샘플이 있고 각각 1000개의 step이 있다면 계산할때
2,300,000,000,000개의 항목이 있을 것이다.

따라서 크기가 큰 데이터의 경우 gradient descent 방법은 너무 느리다.

여기서 stochastic gradient descent가 유용해진다.

Stochastic gradient descent은 각 step에 대해 하나의 샘플을 무작위로 선택한다. 그리고 그 하나의 샘플을 사용하여 미분을 계산한다. 따라서 이 예제에서는 stochastic gradient descent가 계산 된 항의 수를 3배로 줄였다.

Stochastic gradient descent는 특히 중복이 있을때 유용하다. 예를 들어 12개의 데이터 포인트가 있지만 3개의 클러스터를 형성하는 많은 중복성이 있다.



처음 시작:

intercept = 0 and the slope = 1

아무 데이터나 고른다.

그 데이터의 수치인 Weight, 3 and Height, 3.3를 넣는다.

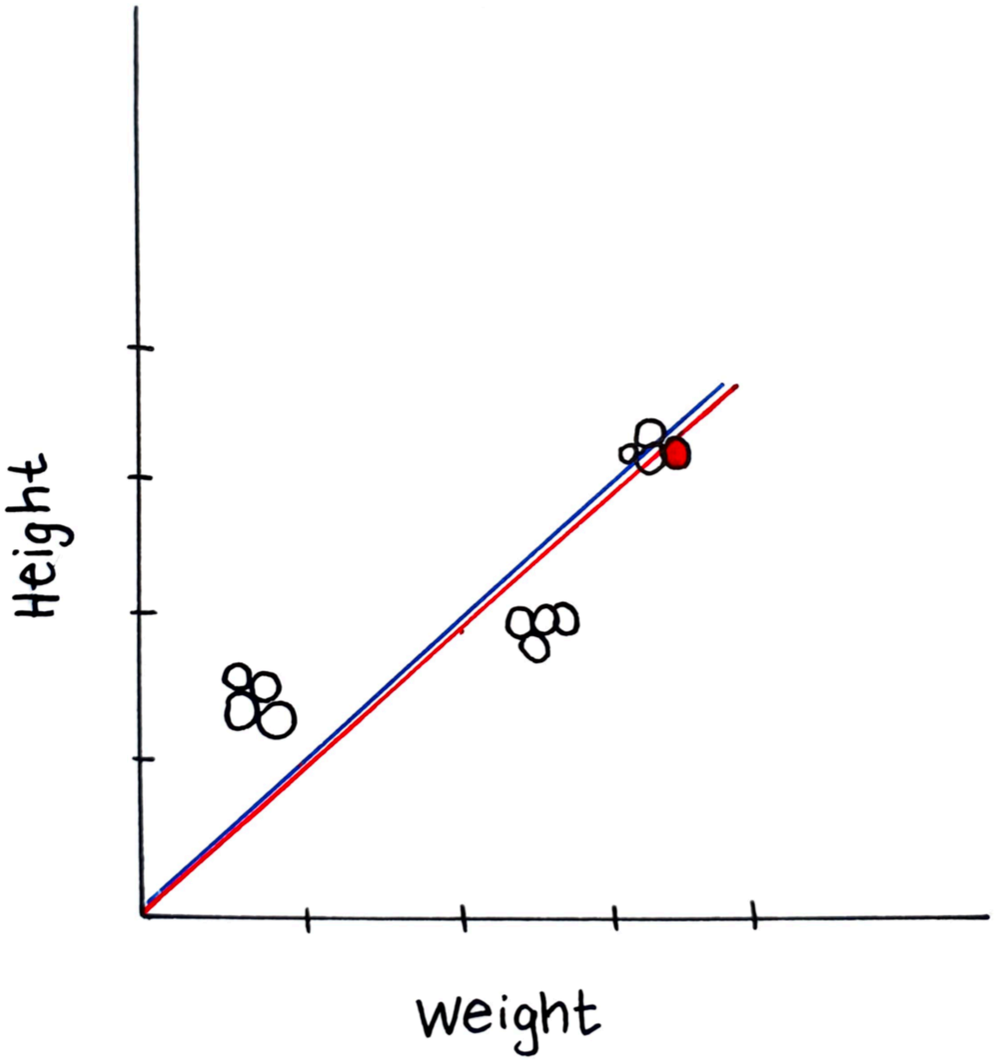
d/d intercept Sum of squared residuals = -2(3.3-(0+1x3) = -0.6

d/d slope Sum of squared residuals = -2 x 3(3.3-(0+1x3)) = -1.8

slope을 넣고 learning rate만큼 곱한다.

Step Size intercept = -0.6 x Learning Rate

Step Size slope = -1.8 x Learning Rate



전략: 상대적으로 큰 학습률로 시작하고 각 단계마다 더 작게 만든다.

학습률이 상대적으로 큰 것에서 상대적으로 작은 것으로 변경

If set the learning rate = 0.01:

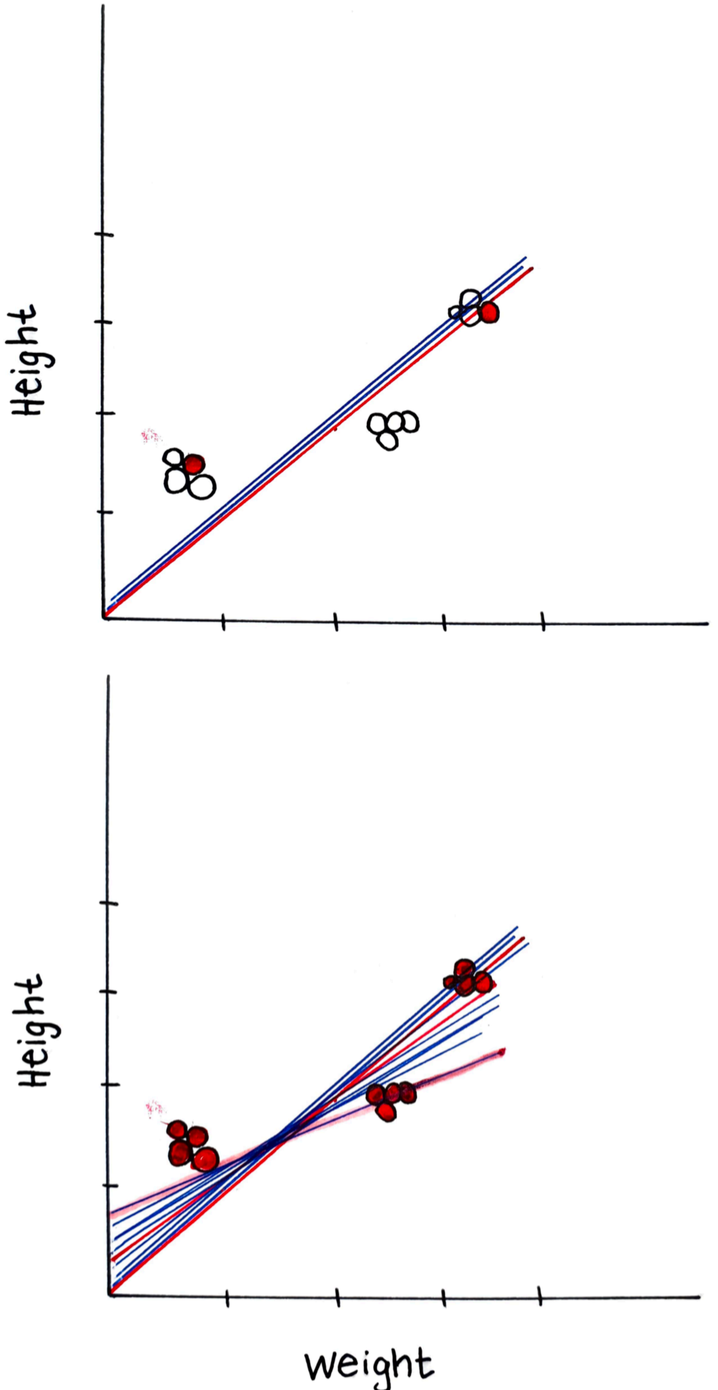
Step Size intercept = -0.6 x 0.01 = -0.006
Step Size slope = -1.8 x 0.01 = -0.018

Calculate the new intercept

New intercept = 0 - step size = 0 - (-0.006) = 0.006
New Slope = Old Slope = Step Size = 1 - (-0.018) = 1.018

임의의 점 선택, 다른 선에 대한 절편과 기울기 계산, 반복

Ultimately, we end up with a line where the intercept = 0.85
and the slope = 0.68



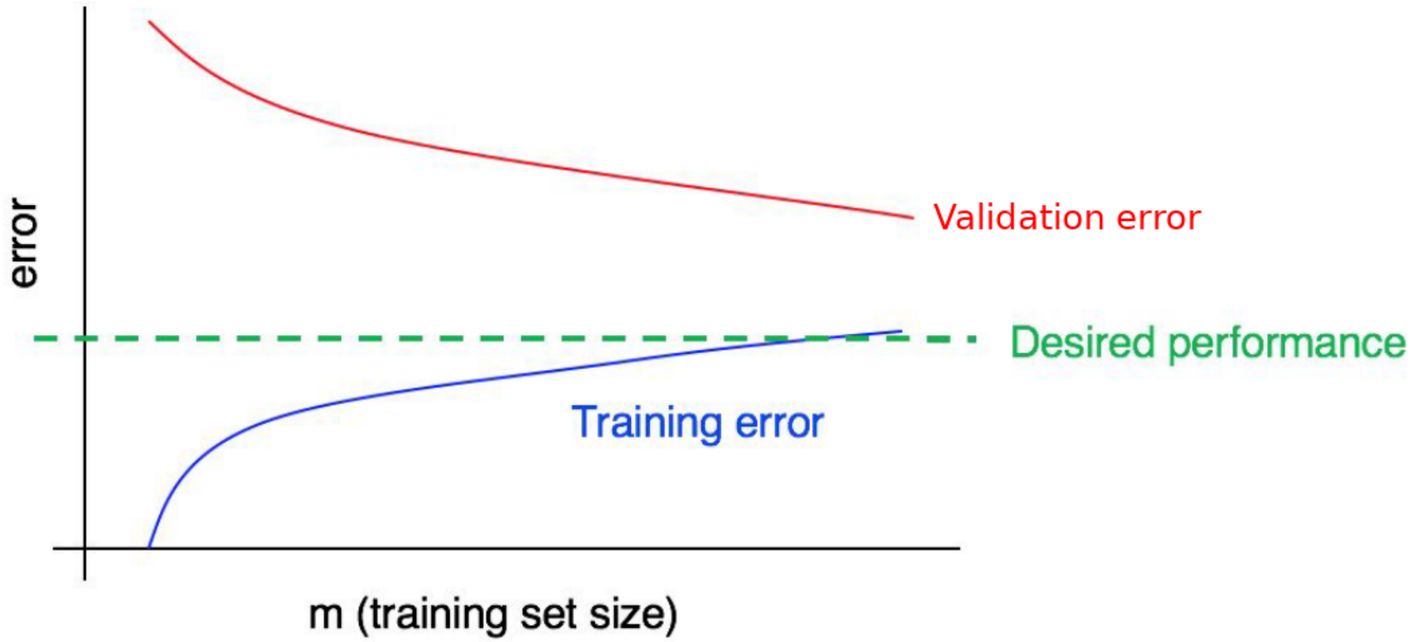
Strict definition of Stochastic Gradient Descent: to only use 1 sample per step
But it is more common to select a small subset of data, or **mini-batch** for each step
Ex) could use 3 samples per step instead of just 1

Using mini-batch benefits:
Can result in more **stable** estimates of the parameters in fewer steps
Using a mini-batch is much **faster** than using all of the data

Stochastic Gradient Descent is just like regular Gradient Decent, except it only looks at one sample per step or a small subset, of mini-batch for each step.

Stochastic Gradient Descent is great when we have tons of data and a lot of parameters, and can easily update the parameters when new data shows up.

일반화


error
Validation error
Desired performance
Training error
m (training set size)

Equations

Cost function $$J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x},\mathrm{y}\sim\hat{p}_{\mathrm{data}}} L(\boldsymbol{x}, y, \boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$

$$L \text{ is the per-example loss } L(\boldsymbol{x}, y, \boldsymbol{\theta}) = -\log p(y \mid \boldsymbol{x}; \boldsymbol{\theta})$$

Stochastic Gradient $$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$

Estimate of the Gradient $$\boldsymbol{g} = \frac{1}{m'}\nabla_{\boldsymbol{\theta}} \sum_{i=1}^{m'} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$

Estimated Gradient Downhill $$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \boldsymbol{g}$$