# PyTorch로 딥러닝 제대로 배우기

## - 중급편 -

Part2-2. 머신러닝 개론

강사: 김 동 희

# 3. 오차 측정

## ❑ 목표 (Target)

- 데이터($X$)를 통해 구하고자 하는 해답

## ❑ Supervised Learning

- 데이터($X$)에 해당 Task 해결을 위한 정답 label이 존재하는 경우
- 이때, 오차 측정은 정답 label과 도출한 예측 값과 차이로 산출

## ❑ Unsupervised Learning

- 데이터($X$)에 정답 label이 존재하지 않음
- 데이터간에 거리, 유사도 등을 비교하여 오차 측청을 수행

## ❑ Self-supervised Learning

- 데이터($X$)에 정답 label이 존재하지 않지만, 강제로 정답 label을 만들어서 학습하는 방법
- 데이터($X$)를 Rotation, Random Cut 등을 수행하여 만들어진 데이터($X'$)와 오차 측정을 하는 방법

# 3. 오차 측정

## ❑ 손실 함수 (Loss Functions)

- L1Loss

$$l_n = |x_n - y_n|,$$

- MSELoss

$$l_n = (x_n - y_n)^2$$

- CrossEntropyLoss

$$l_n = -w_{y_n} \log \frac{\exp(x_{n,y_n})}{\sum_{c=1}^{C} \exp(x_{n,c})} \cdot 1\{y_n \neq \text{ignore\_index}\}$$

- KLDivLoss

$$L(y_{\text{pred}},\ y_{\text{true}}) = y_{\text{true}} \cdot \log \frac{y_{\text{true}}}{y_{\text{pred}}} = y_{\text{true}} \cdot (\log y_{\text{true}} - \log y_{\text{pred}})$$

- TripletMarginLoss

$$L(a, p, n) = \max\{d(a_i, p_i) - d(a_i, n_i) + \text{margin}, 0\}$$

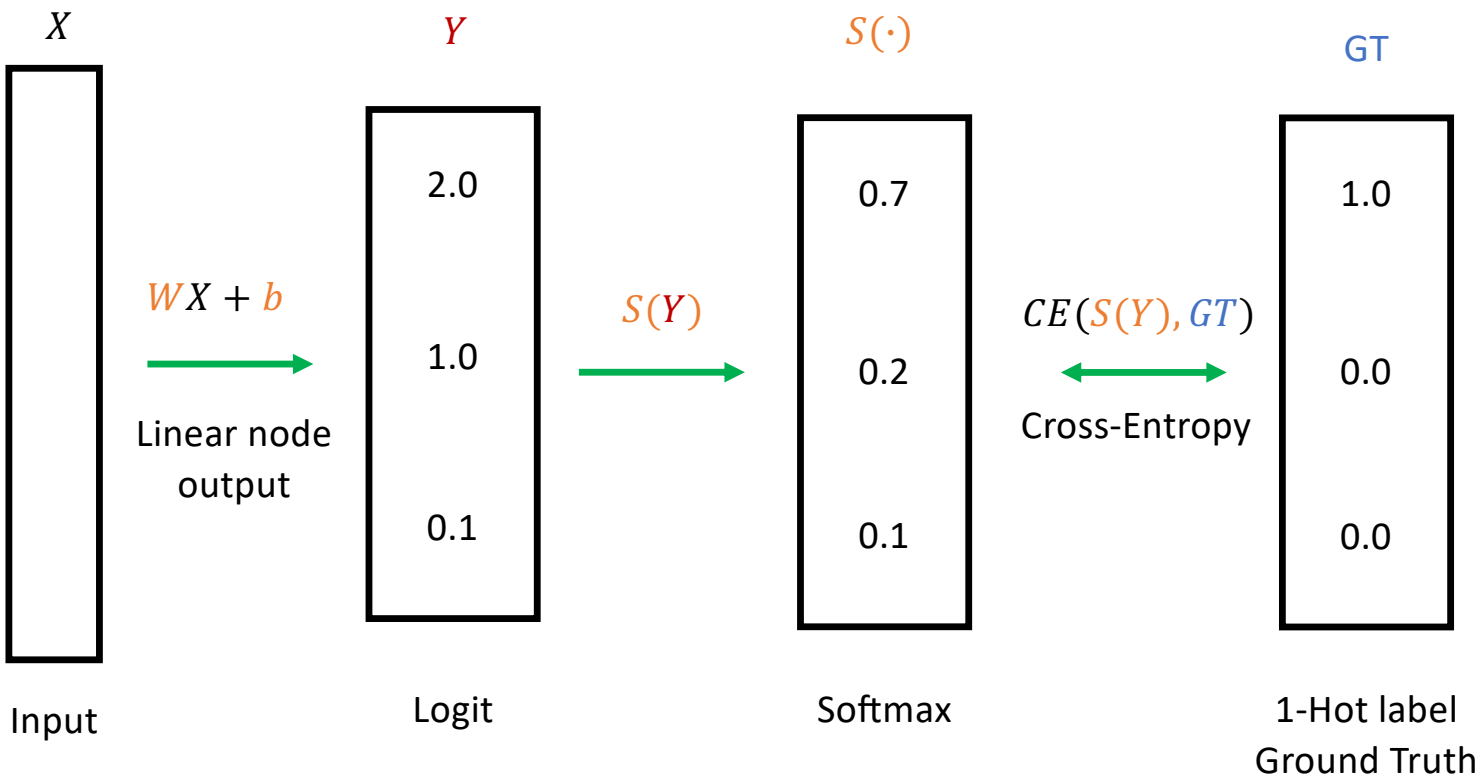
# 3. 오차 측정

## ❑ Cross Entropy

- $H(p,q) = -\sum_{c=1}^{C} \mathrm{p}(y_c)\log(q(y_c))$
- 예측 모델(q)은 실제 분포 p를 모르는 상태에서 구분 분포를 따라가고자 함
- 즉, entropy를 줄이는 쪽으로 minimize 함 (불확실성을 줄이겠다)
- Loss 함수에서는 **분포의 차이를 줄이는 것이 목표**
- PyTorch에서 CrossEntropyLoss는 Softmax가 포함되어 구현
- $l_n = -w_{y_n} \log \frac{\exp(x_{n,y_n})}{\sum_{c=1}^{C} \exp(x_{n,c})} \cdot 1\{y_n \neq ignore_{index}\}$

# 3. 오차 측정

## ❑ Cross Entropy



# 3. 오차 측정

## ❑ KLDivLoss

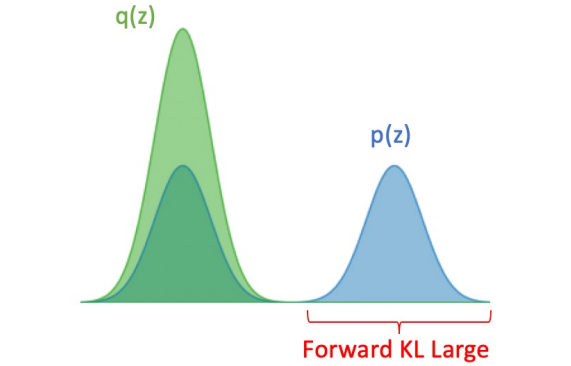
- 서로 다른 두 분포의 차이(dissimilarity)를 측정하는데 쓰이는 measure
- $D_{KL}(P||Q) = \sum_{x \in \chi} P(x) log\left(\frac{P(x)}{Q(x)}\right) = -\sum_{x \in \chi} P(x)[\log(Q(x) - log\big(P(x)\big)] = H(P,Q) - H(P)$
- Cross-entropy는 entropy 보다 항상 0보다 크거나 같은 값을 같기 때문에 P의 entropy(정보량)과 P,Q의 cross-entropy가 같을 때 최소 값을 가짐
- P를 실제분포라하고 Q를 예측 분포라고 한다면, H(P)는 항상 상수 값을 갖기 때문에 Cross-entropy를 최소화하는 방법으로 loss를 줄임
- 단, $D_{KL}(P||Q) \neq D_{KL}(Q||P)$

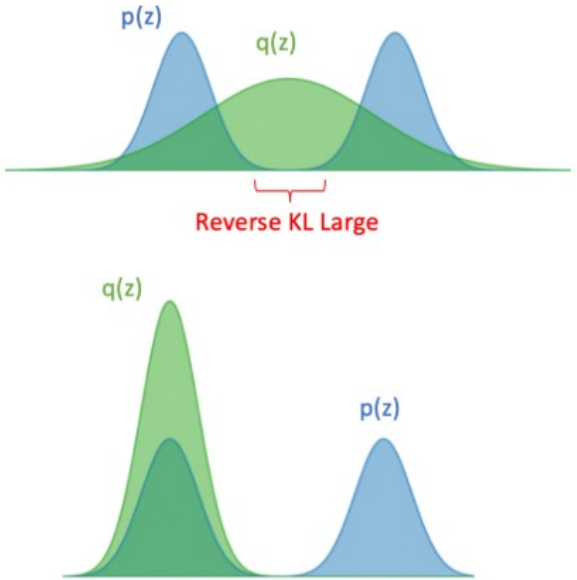
https://3months.tistory.com/436
https://towardsdatascience.com/why-is-cross-entropy-equal-to-kl-divergence-d4d2ec413864

# 3. 오차 측정

## ❑ KLDivLoss

- $D_{KL}(P||Q) = H(P,Q) - H(P) = Forward$
- $D_{KL}(Q||P) = H(Q,P) - H(Q) = Reverse$



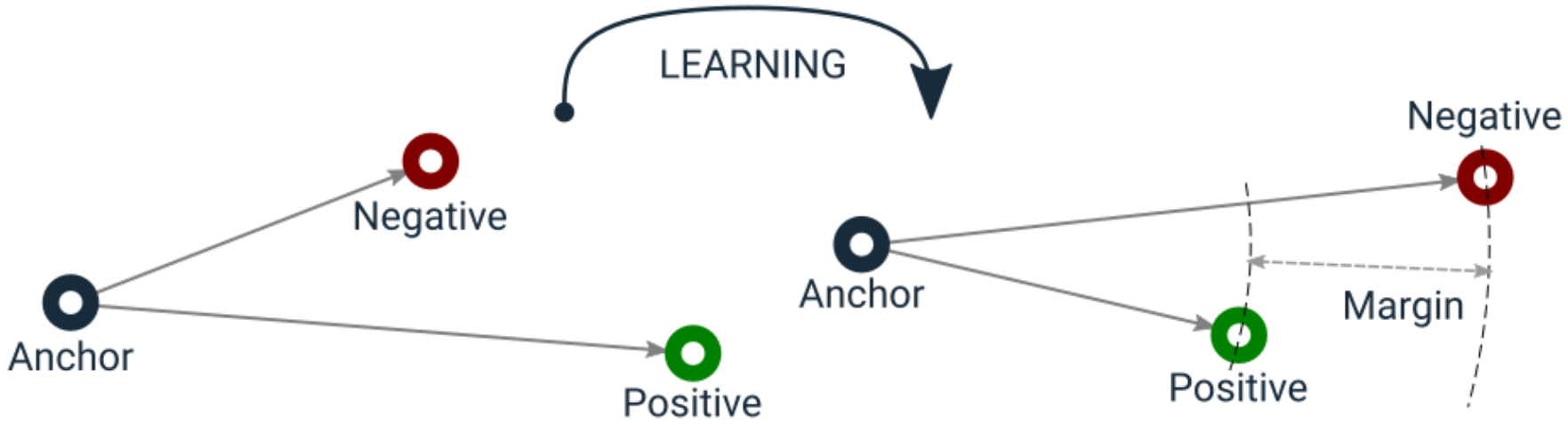




http://adamlineberry.ai/vae-series/kl-divergence

# 3. 오차 측정

## ❑ Triplet Margin Loss

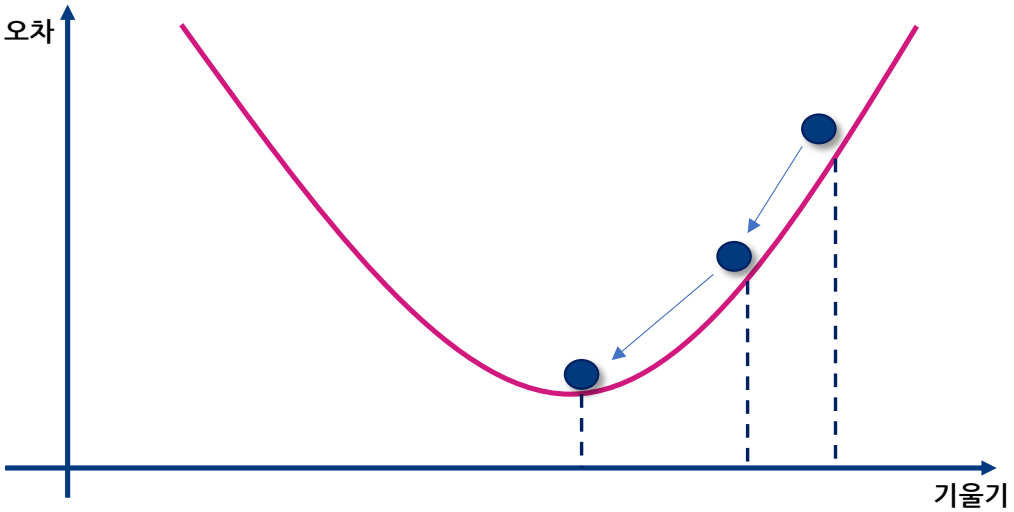
- Anchor를 기준으로 Positive 또는 Negative하게 학습을 유도
- Anchor에 positive한 샘플들은 가까이에, Anchor와 negative 한 샘플은 멀리 embedding 되도록 유도



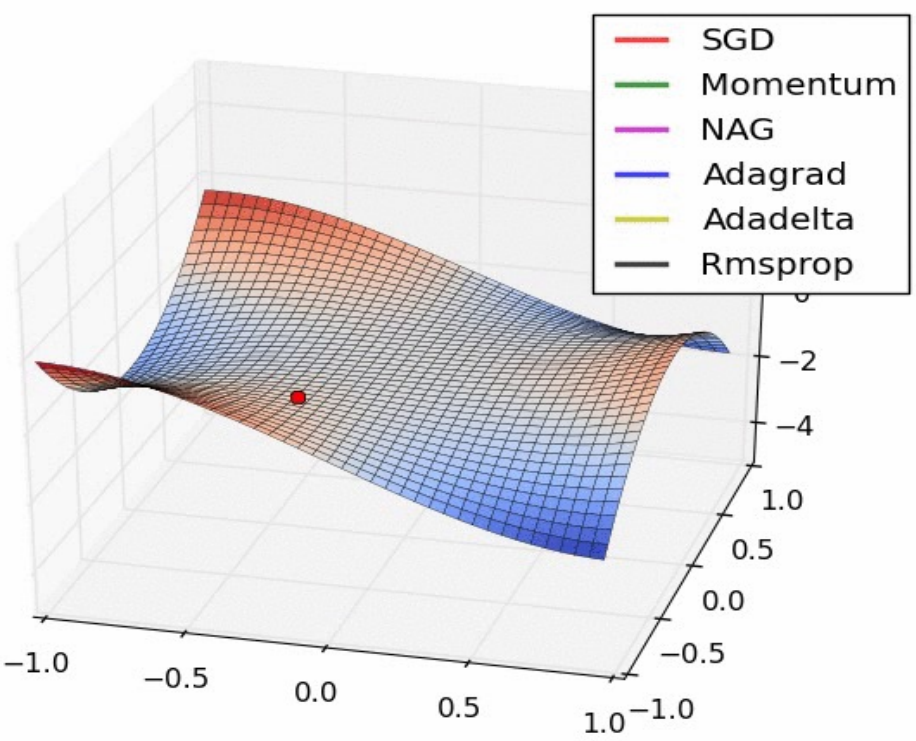
# 4. 최적화

## ❑ Overview

"오차가 가장 작은 점을 찾자"



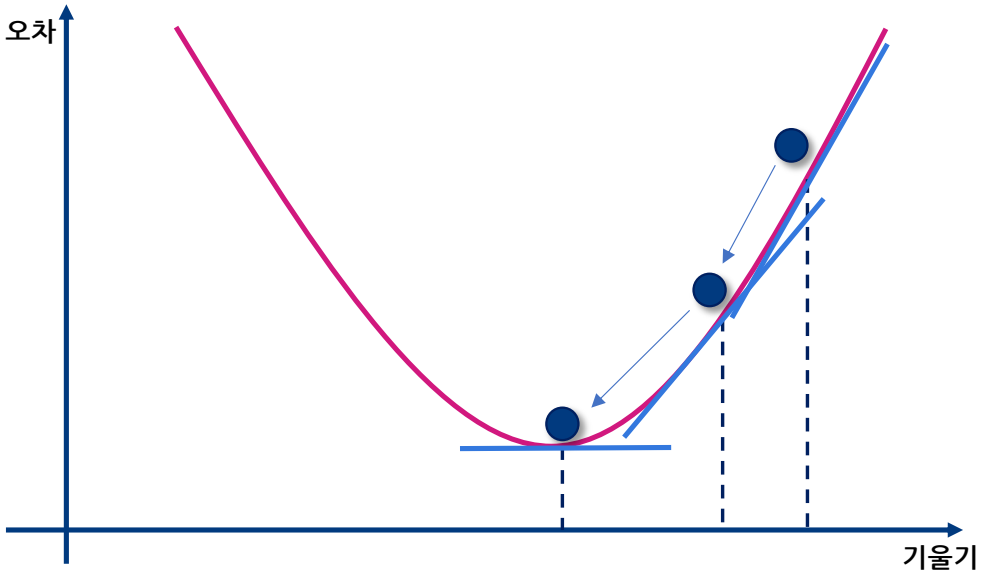

But, 현재 point가 minima인 것을 어떻게 알지?



# 4. 최적화

## ❑ 미분의 개념

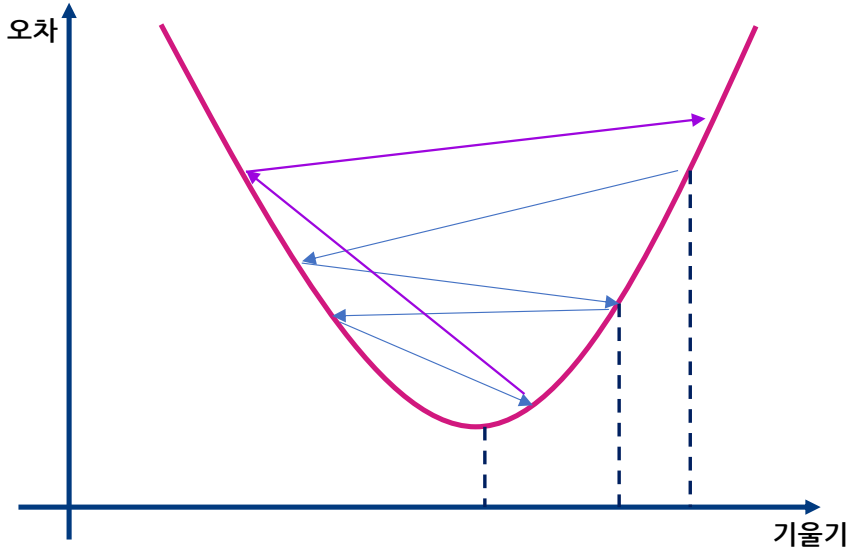
- 미분은 순간 변화율
- $\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h)-f(x)}{h}$
- x의 작은 변화가 함수 f(x)를 얼마나 변화시키느냐?
- 즉, 기울기를 의미함
- 기울기가 0에 가까울 수록, 최저점을 의미함

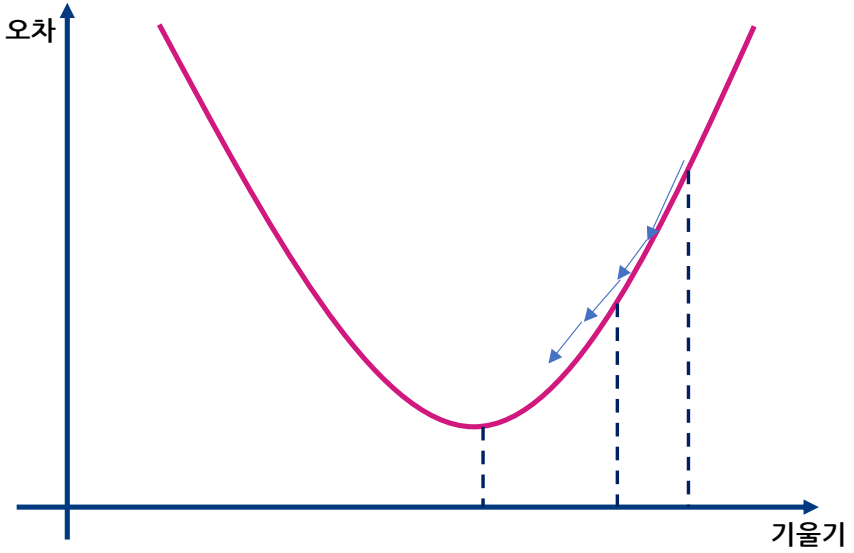


# 4. 최적화

## ❑ Learning rate

- 얼마만큼 움직일 것인지 결정하는 문제가 학습률(Learning rate)라 함
- 학습률이 너무 크면 최저점을 찾지 못하고 발산하고,
- 학습률이 너무 작으면 최저점을 찾는데 오래 걸림

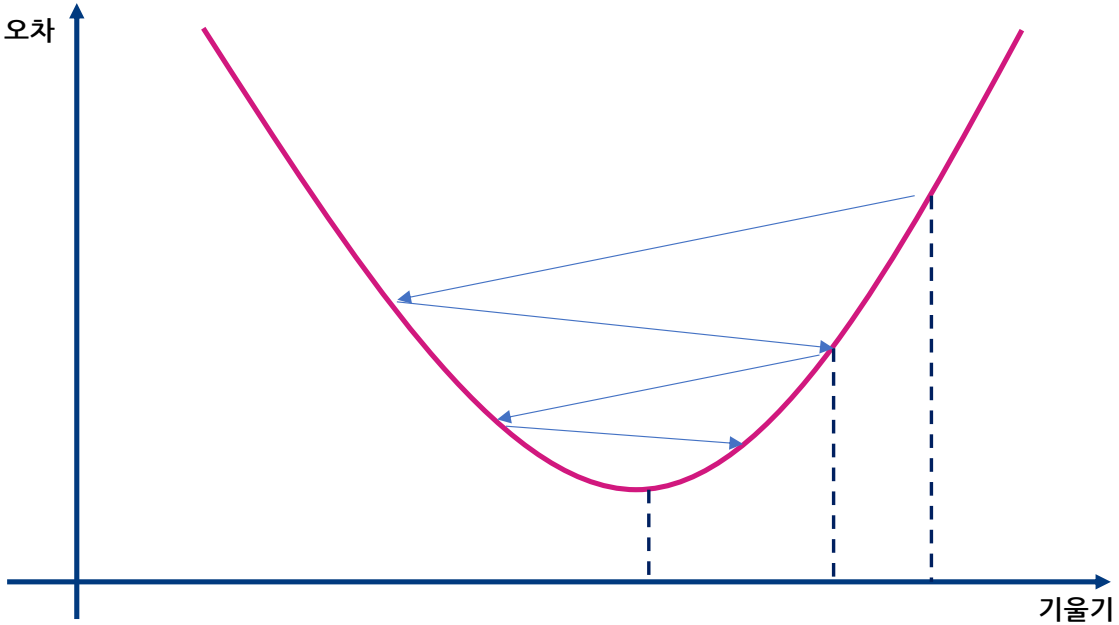


# 4. 최적화

## ❑ Stochastic Gradient Decent

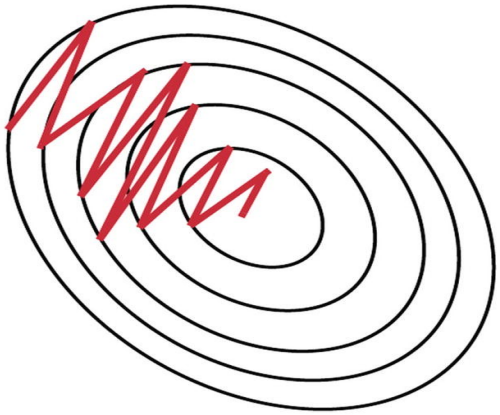
- $\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta \frac{\partial L}{\partial \boldsymbol{W}}$
- 일정한 거리를 움직이면서 오차 함수에서의 최저점을 찾는다.
- 경사를 내려가는 것과 같다고 하여, Gradient Decent라 부른다.



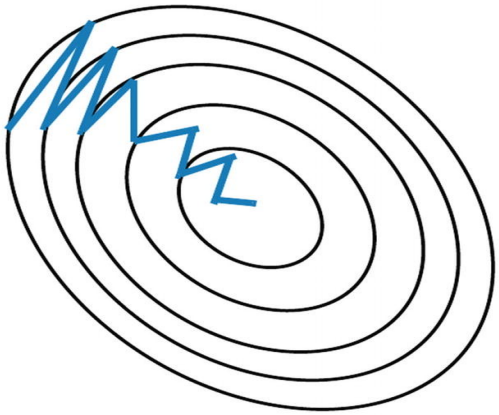
# 4. 최적화

## ❑ Momentum

- SGD는 상당히 비 효율적인 움직임을 보임
- $v \leftarrow \alpha v - \eta \frac{\partial L}{\partial W}$
- $W \leftarrow W + v$
- 기울기 방향으로 물체가 가속된다는 물리 법칙을 적용



Stochastic Gradient Descent **withhout** Momentum



Stochastic Gradient Descent **with** Momentum

감사합니다.