SGD: 최근 대부분의 딥러닝 알고리즘에 사용됨
gradient descent 의 확장판임
머신러닝에서 큰 training sets은 generalization에 좋다. 하지만 클수록 계산하는 데 비용이 증가함.

아래는 training data에 대한 negative conditional log-likelihood.

L에 대한 수식.
이 cost(loss) function에 따라서 gradient descent는 아래와 같습니다.

이곳에서는 batch size가 모두 m입니다. O(m).

SGD 관점에서 expectation은 gradient이다.
따라서 training sets을 모두 적용할 필요 없이 mini batch를 sampling하여 더 효율적으로 구할 수 있음.
minibatch size: 1 to a few hundred.

SGD의 장점은 위와 같음.

kernel을 사용하는 ML algorithm은 위와 같은 이유로 cost가 높음.
하지만 SGD는 m과 상관 없이 each step O(1).

단점은 추가로 찾아보았습니다.

비등방성 함수일 경우 시간이 오래 걸리는 문제와
최적화했을 때의 weight가 극솟값인지 최솟값인지 알 수 없습니다.

비등방성 함수 SGD를 수행하면 아래와 같이 y축에 대해서 지그재그로 움직여 처적해를 찾는 데 시간이 더 소요되는 문제가 발생합니다.
이를 해결하기 위해 Momentum과 AdaGrad를 사용한다고 합니다.

parameter w가 기울기를 따라 계속 진행하여 기울기가 0이 되는 지점에 도달했습니다.
하지만 이 지점이 손실함수의 최솟값이 아니라 극솟값일 경우 문제가 발생합니다.
손실함수를 최소화하지 못했으나 기울기가 0이 되어 학습이 종료됩니다.
이 또한 Momuntum을 통해 해결할 수 있다고 합니다.