Gradient Descent

Vanilla Gradient Descent

Compute gradient for the entire training set and then update parameters

$$\theta^t = \theta^{t-1} - \alpha \nabla L(\theta^{t-1})$$

Stochastic Gradient Descent

Compute gradient for one training data and then update parameters subsequently.

$$\theta^t = \theta^{t-1} - \alpha \nabla L_i(\theta^{t-1})$$

where L_i means the loss function applies on only one instance.

Randomly Choose and Return

$\{x_1, ..., x_N\}$, $L(\theta) = \frac{1}{N} \sum_{i=1}^{N} L_i(\theta)$. Choose x_i uniformly randomly from $\{x_1, ..., x_N\}$.

$$\theta^1 = \theta^0 - \alpha \nabla L_i(\theta^0)$$

x_i will not be pulled out from instances. Choose new x_j uniformly randomly again.

$$\theta^2 = \theta^1 - \alpha \nabla L_j(\theta^1)$$

Shuffle and Not Return

Randomly shuffle $\{1, ..., N\}$ into $\{k_1, ..., k_N\}$. For i for 1 to N,

$$\theta^t = \theta^{t-1} - \alpha \nabla L_{k_i}(\theta^{i-1})$$

One process go trough all instances is an "epoch".

Mini-Batch Gradient Descent

Split instances into m subsets, each subset (or mini-batch) has n instances. Update θ per mini-batch.

$$\theta^t = \theta^{t-1} - \alpha \frac{1}{m} \sum_{i=1}^{m} \nabla L_i(\theta)$$

After going through all batches, we complete one "epoch".