GD-based algorithms for model parameter optimization

Alex Tse
University College London

Motivation

- Suppose we have fixed a machine learning model (e.g. a deep neural network) $f = f(x; \theta)$ parameterized by a vector of model parameters θ
- Goal of model training: find the parameters θ to minimize the average losses across all training data $\{(x_i, y_i)\}_{i=1}^N$. i.e. to solve

$$\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f(x_i; \boldsymbol{\theta}))$$

  where $\ell(\cdot, \cdot)$ is our choice of loss function
- Impossible to solve the problem analytically because f can have a complicated form $\Longrightarrow$ efficient numerical methods required

Gradient Descent (GD)

Goal: Find the optimal θ to attain the local minimum of the function $L(\theta)$, where $L : \mathbb{R}^p \to \mathbb{R}$ is continuously differentiable.

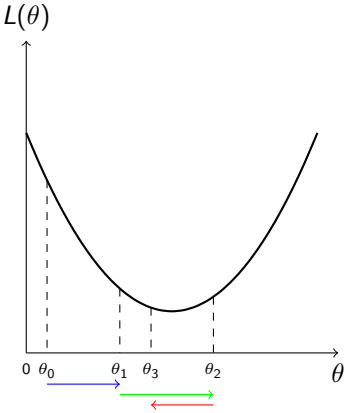
Algorithm: Initialize θ_0. Fix the maximum number of iterations M. For any integer $n \in \{0, 1, \cdots, M-1\}$,

$$\boxed{\theta_{n+1} = \theta_n - \eta \nabla L(\theta_n)}$$

where η is called the learning rate, $\eta > 0$ and $\theta = (\theta^1, \cdots, \theta^p) \in \mathbb{R}^p$ and $\nabla L(\theta) = (\partial_{\theta^1} L(\theta), \cdots, \partial_{\theta^p} L(\theta))$.

Intuition behind GD

- Consider a simple one-dimensional minimization problem $\min_{\theta} L(\theta)$



- Choose an arbitrary starting point θ_0
- Make a step to right (resp. left) if the function is downward (resp. upward) slopping at the current location
- A simple update rule is:

$$\theta_{n+1} = \theta_n - \eta L'(\theta_n)$$

for $n = 0, 1, 2...$

  - Direction of step depends on the sign of $L'(\theta_n)$
  - Make larger adjustment when the current slope is steep
  - $\eta > 0$ is a parameter controlling the step size

Batch Gradient Descent (BGD)

Model Parameter Optimization

Find optimal θ to minimize $L(\theta|\mathcal{D})$ in the following additive form:

$$L(\theta|\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N} Q_\theta(x_i, y_i),$$

where N is the sample size, θ is the model parameter and $Q_\theta : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$.

Batch Gradient Descent

Direct application of GD to empirical loss function. The update rule is given as follows:

$$\theta_{n+1} = \theta_n - \eta \underbrace{\nabla L(\theta_n|\mathcal{D})}_{\text{Gradient Term}} = \theta_n - \eta \underbrace{\frac{1}{N}\sum_{i=1}^{N} \nabla Q_{\theta_n}(x_i, y_i)}_{\text{Gradient Term}}.$$

Variants of Gradient Descent Methods (I)

Stochastic Gradient Descent (SGD)

Algorithm: At each step n, we uniformly choose a random index i_n from $\{1, \cdots, N\}$ without replacement, and update the weights θ_n as follows:

$$\theta_{n+1} = \theta_n - \eta_n \underbrace{\nabla Q_{\theta_n}(x_{i_n}, y_{i_n})}_{\text{Stochastic Gradient Term}} .$$

for a suitably chosen decreasing sequence $\{\eta_n\}_n$ with the limit 0.
Idea:

$$\mathbb{E}_{x,y}[\nabla Q_\theta(x_{i_n}, y_{i_n})] = \nabla L(\theta|\mathcal{D})$$

where (x_{i_n}, y_{i_n}) is sampled from the empirical distribution of $(x_i, y_i)_{i=1}^N$.

Variants of Gradient Descent Methods (II)

Mini-Batch Gradient Descent (Mini-Batch GD)

The Mini-Batch Gradient Descent computes the gradients on small random sets of instances called mini-batches.

$$\theta_{n+1} = \theta_n - \eta_n \underbrace{\frac{1}{m}\sum_{i=1}^{m} \nabla Q_{\theta_n}(x_i, y_i)}_{\text{Mini-batch Gradient Term}} ,$$

where m is called the mini-batch size. [a]

[a] To distinguish mini-batch GD from BGD and SGD, m in Mini-Batch GD is not equal to 1 or N).

An epoch refers to one complete pass of the training dataset through the gradient updates.

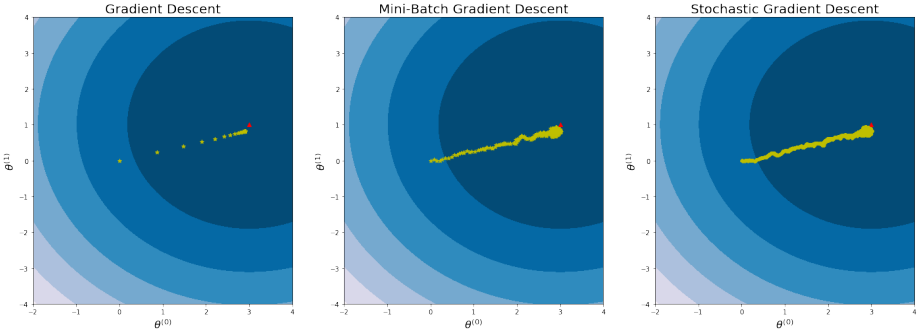
Algorithm (Mini-Batch GD)

Input: θ_0 (initial value); N_{epoches} (the number of epochs); m (the batch size).
Output: θ (estimated model parameter).

1: $N_{\text{batches}} = N/m$.[1]
2: Initialize $\theta = \theta_0$;
3: **for** $n = 1 : N_{\text{epoches}}$ **do**
4: Randomly reshuffle the whole dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$;
5: **for** $q = 1 : N_{\text{batches}}$ **do**
6:
$$\theta = \theta - \frac{1}{m}\sum_{i=1}^{m}\eta \nabla Q_{\theta_n}(x_{(q-1)m+i}, y_{(q-1)m+i}),$$
where m is the size of mini-batch.
7: **end for**
8: **end for**
9: Return θ.

[1]For simplicity, we assume that the sample size N is a multiplier of the mini-batch size m.

Method Comparison



- BGD: It has the stable training process and easier to converge. But it is computational expensive per iteration, and likely to be stuck at the local minima.
- SGD: It is very quick to evaluate each iteration. Randomness helps to escape local minimun, but makes the settling of the minimum difficult.
- Mini-batch GD: A combination of batch GD and SGD.

Method Comparison

	BGD	SGD	Mini-Batch GD
Update Frequency	Low	High	Medium
Update Complexity	High	Low	Medium
Fidelity of loss gradient	High	Low	Medium
Stuck the local minimum	Easy	Difficult	Difficult
Easy to Converge	Yes	No	No

Table: Comparison of various GD based methods.

Thanks for your attention!