Gradient Descent Method

1 Overview

- GD
- SGD
- Mini Batched SGD

The parameter estimation method is shown below

$$\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta_t \cdot \frac{\partial f(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}}$$

2 GD

- For the gradient descendant (GD) method, the gradient will be updated as

$$\nabla L(h_{\boldsymbol{\theta}}, S) = \frac{1}{m}\sum_{i=1}^{m} \nabla R(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i)$$

- However, it is hard to update the gradient once by considering all samples.

3 SGD

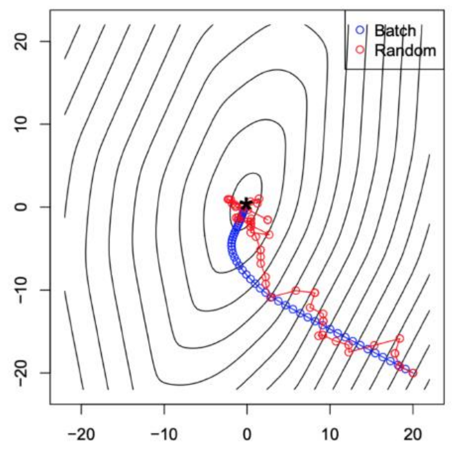
- The gradient can be viewed as an average of multiple samples. Each time the SGD method will learn the weight from one data sample.

$$\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta_t \cdot \nabla f_i(\boldsymbol{\theta})|_{\boldsymbol{\theta}^{(t)}}$$

where

- t: time step
- $\nabla f_i(\boldsymbol{\theta}^{(t)})$ is the gradient of the single-example loss L
- η_t is the learning rate (step size)

- SGD can be considered as GD with some randomness.

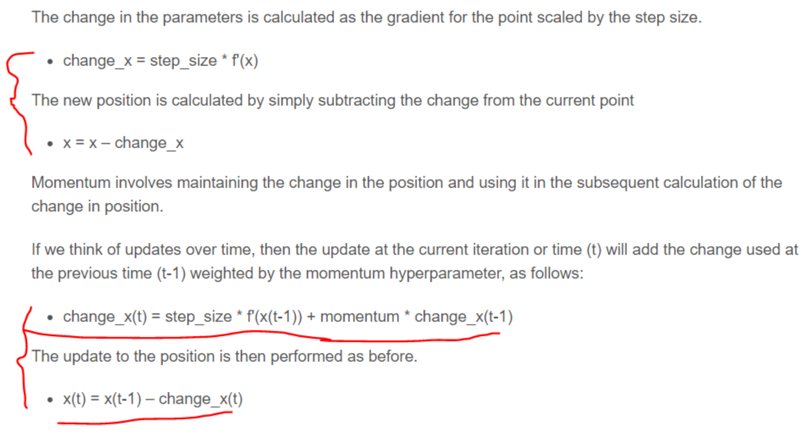


4 Batched SGD

- **Key idea**: each time SGD learns the weight from multiple data samples, which could reduce the number of iterations to learn the weight.

5 Momentum

- **Motivation**: Momentum is an extension to the gradient descent optimization algorithm, often referred to as gradient descent with momentum.
- **Challenge**: A problem with the gradient descent algorithm is that the progression of the search can bounce around the search space based on the gradient. For example, the search may progress downhill towards the minima, but during this progression, it may move in another direction, even uphill, depending on the gradient of specific points (sets of parameters) encountered during the search.
- **Updating rules**: consider part of the gradient in the previous step.



The change in the parameters is calculated as the gradient for the point scaled by the step size.

- change_x = step_size * f'(x)

The new position is calculated by simply subtracting the change from the current point

- x = x – change_x

Momentum involves maintaining the change in the position and using it in the subsequent calculation of the change in position.

If we think of updates over time, then the update at the current iteration or time (t) will add the change used at the previous time (t-1) weighted by the momentum hyperparameter, as follows:

- change_x(t) = step_size * f'(x(t-1)) + momentum * change_x(t-1)

The update to the position is then performed as before.

- x(t) = x(t-1) – change_x(t)

Reference

https://www.linkedin.com/pulse/pytorch-gradient-descent-stochastic-mini-batch-code-sobh-phd/

Yangfeng Ji, machine learning, lecture notes.

https://machinelearningmastery.com/gradient-descent-with-momentum-from-scratch/#:~:text=Momentum%20is%20an%20extension%20to,spots%20of%20the%20search%20space.