Stochastic gradient descent

Like usual gradient descent, Stochastic gradient descent is motivated to minimize an average of functions:

$$\min_{\boldsymbol{\beta}} \frac{1}{n} \sum_{i=1}^{n} f_i(\boldsymbol{\beta}).$$

[For regression: $f_i(\boldsymbol{\beta}) = (y_i - \mathbf{z}_i^T \boldsymbol{\beta})^2$, where $x_i = (y_i, z_i)$.]

The 'exact' gradient of the above objective function will be $\frac{1}{n} \sum_{i=1}^{n} \frac{\partial f_i(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}$ and thus the gradient descent algorithm will repeat the following steps:

$$\boldsymbol{\beta}^{(k)} = \boldsymbol{\beta}^{(k-1)} - t_k \cdot \frac{1}{n} \sum_{i=1}^{n} \frac{\partial f_i(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} \Big|_{\boldsymbol{\beta}=\boldsymbol{\beta}^{(k-1)}}$$

However, gradient descent will be very costly if we have n very large or have streaming data with increasing n.

Then, we can apply the following algorithm:

$$\boldsymbol{\beta}^{(k)} = \boldsymbol{\beta}^{(k-1)} - t_k \cdot \frac{\partial f_{i_k}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} \Big|_{\boldsymbol{\beta}=\boldsymbol{\beta}^{(k-1)}}$$

This algorithm is called the stochastic gradient descent or SGD (or incremental gradient descent). The main appeal and motivations of using SGD are: 1) The iteration cost of SGD is independent of m 2) SGD can be a big savings in terms of memory usage.

Note: SGD is not necessarily a descent method!

Two rules for choosing index i_k at iteration k:

- Randomized rule: choose $i_k \in 1 \ldots, n$ uniformly at random.
- Cyclic rule: choose $i_k = 1, 2, \ldots, n, 1, 2, \ldots, n, \ldots$.

Randomized rule is more common in practice. For randomized rule, note that $\mathbb{E}\left(\frac{\partial f_{i_k}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}\right) = \mathbb{E}\left(\frac{\partial f_i(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}\right) \approx \frac{1}{n} \sum_i \frac{\partial f_i(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}$ using the sample mean-population mean argument. So we can view SGD as using an unbiased estimate of the gradient at each step.

1 Step Sizes

It is standard to use diminishing step sizes in SGD (e.g., $t_k = 1/k$). The short (and intuitive) explanation for why we do not use fixed step sizes is the following. Suppose we take cyclic rule for simplicity. We have $t_k = t$ for n updates in a row, so for SGD we get $\boldsymbol{\beta}^{(k+n)} = \boldsymbol{\beta}^{(k)} + t\sum_{i=1}^{n} \frac{\partial f_{i_k}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} \Big|_{\boldsymbol{\beta}=\boldsymbol{\beta}^{(k+i-1)}}$.

Meanwhile, for full gradient method with step size nt, we have $\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} + t\sum_{i=1}^{n} \frac{\partial f_{i_k}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} \Big|_{\boldsymbol{\beta}=\boldsymbol{\beta}^{(k)}}$.

Consider the error between the above two updates: $t[\sum_{i=1}^{n} \frac{\partial f_{i_k}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} \Big|_{\boldsymbol{\beta}=\boldsymbol{\beta}^{(k+i-1)}} - \sum_{i=1}^{n} \frac{\partial f_{i_k}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} \Big|_{\boldsymbol{\beta}=\boldsymbol{\beta}^{(k)}}]$.

If we keep t constant, this difference will in general not go to zero. Hence, usually using a diminishing step sizes will make stochastic update a more accurate estimate of the full gradient estimate.

2 Mini-batch SGD

Since people in machine learning often want better performance on the samples outside the training sets, which makes fully optimization of the objectives unnecessary, therefore we can always stop early on an "okay" solution. To people actually want the optimization, we show a common way to improve the convergence rate of SGD, which the Mini-batch SGD. Mini-batch SGD For the optimization problem described early, during the k-th update, we choose a random subset $I_k \subseteq \{1, \ldots, n\}$, where $|I_k| = b \ll n$, and use the following update rule:

$$\boldsymbol{\beta}^{(k)} = \boldsymbol{\beta}^{(k-1)} - t_k \cdot \frac{1}{b} \sum_{i \in I_k} \frac{\partial f_{i_k}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} \Big|_{\boldsymbol{\beta}=\boldsymbol{\beta}^{(k-1)}}$$

As before, the gradient estimate is unbiased as $\frac{1}{b}\sum_{i\in I_k} \frac{\partial f_{i_k}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \mathbb{E}\left(\frac{\partial f_i(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}\right) \approx \frac{1}{n}\sum_i \frac{\partial f_i(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}$ using the sample mean-population mean argument. and its variance is reduced by a factor $1/b$, though at the cost of b times more expensive running time at each iteration. Therefore, one may regard Mini-batch SGD as a compromise between SGD and full gradient descent.

Theoretically, it does not lead to any computational gain in terms of convergence rate. Though it is not convincing to use Mini-batch SGD from the theory aspect, Mini-batch SGD turns out performing not too bad in practice.

3 SGD with momentum

SGD is often augmented with a momentum term, so that the update in,

$$\boldsymbol{\beta}^{(k)} = \boldsymbol{\beta}^{(k-1)} - t_k \cdot \frac{\partial f_{i_k}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} \Big|_{\boldsymbol{\beta}=\boldsymbol{\beta}^{(k-1)}} + \theta(\boldsymbol{\beta}^{(k-1)} - \boldsymbol{\beta}^{(k-2)})$$

where θ is an exponential decay factor between 0 and 1 that determines the relative contribution of the current gradient and earlier gradients to the weight change.

Remark As we have discussed, SGD can be very effective in terms of resources (per iteration cost, memory), and for this reason it is still the standard for solving large problems. However, SGD is slow to converge, and does not benefit from strong convexity unlike full gradient descent. Mini-batch SGD can be an approach to trade off effectiveness and convergence rate: it is not that beneficial in terms of ops, but it can still be useful in practice in settings where the cost of computation over a batch can be brought closer to the cost for a point. Still, while mini-batches help reducing the variance, they do not necessarily lead to faster convergence in the theoretical domain. Due to these properties, it was thought for a while that SGD would not be commonly useful. It was recognized very recently that these lower bounds (which were established for a more general stochastic problem) do not apply to finite sums, and with the new wave of variance reduction methods, it was shown that we can modify SGD to converge much faster for finite sums.

4 Few more extensions that you may look into in future

There are some Newton-like adjustments to set parameter-specific different learning rates. You may check those out. Specifically, these methods use the sequence of updates to quantify different aspects of the loss functions.

1. AdaGrad (adaptive gradient algorithm)
2. RMSProp (Root Mean Square Propagation)
3. ADAM (Adaptive Moment Estimation)