Optimization in Deep Learning

Zhang Jinxiong

December 21, 2017

Abstract

Stochastic gradient descent is to optimize the finite sum objective functions based on gradient descent. Stochastic gradient descent (SGD) is the popular optimization algorithm in deep learning. It is easy to implement and of much advantages in differentiable optimization within high dimension space.

1 Introduction

Gradient descent or steepest descent is the fundamental optimization only using the gradient of the object function. It is simple and especially efficient in convex optimization problem. But when the dimension of the objective function is too high, it is not easy to compute its gradient owing to the dimension curse. To void that, we just grasp some "local" or "partial" information instead of the full information in high dimension space.

For example, the coordinate descent algorithm is to optimize the multi-variable function in one specific variable or coordinate while the rest are fixed sequentially.

It is natural for us to take the advantage of "partial" gradient information in place of the full gradient information with affordable computation cost. This paper is aimed to review some stochastic gradient descent methods.

2 Determinant Gradient Method

The stochastic gradient descent is rooted in classical or determinant method.

2.1 Gradient Descent

Gradient descent is based on the local information - it is the negative gradient that is most speedy to decrease for a function in a given point. Let $f(x)$ be the objective function and $\nabla f(x) = g(x)$.

Note: α must be small enough to ensure that it is descent: $f(x_{k+1}) < f(x_k)$.

The step is key factor in the convergence proof of the gradient descent. One common setting is diminishing but not summable:

$$\sum_{k=0}^{\infty} \alpha_k = \infty, \sum_{k=0}^{\infty} \alpha_k^2 < \infty.$$

Algorithm 1 Gradient Descent

1. Give the initial value: x_0;
2. Update the parameters with small step:

$$x_{k+1} = x_k - \alpha g(x_k).$$

2.2 The Momentum Methods

Momentum is form the physics. It is used to accelerate the speed of convergence.

2.3 The Dynamics in Optimization

It is simple to prove the convergence rate of the above methods in a unified framework.

3 Stochastic Gradient Descent

Stochastic methods are not to compute the exact gradient but to estimate the gradient by sampling. It is always applied to the optimization in the finite sum form. For example, it is to solve the mean square error of least square method in large scale.

The object function is called empirical risk function with the form $f(x_i; \mathbf{x}) = \sum_{i=1}^{n} f_i(\mathbf{x})$, where x_i is constant parameters given by training sample and f_is have the same independent variables $\mathbf{x}$.

Let $\nabla f_i(x) = g_i(x)$. The stochastic gradient descent with constant step is shown below.

Algorithm 2 Primary Stochastic Gradient Descent

1. Give the initial value: x_0;
2. Randomly choose term index k_i in the kth iteration from $1, 2, \cdots, n$;
3. Update the parameters:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha g_{k_i}(\mathbf{x}_k).$$

Note the difference with gradient descent: the index k_i is a random variable! We can randomly choose a minibatch of training set to compute the gradient instead of the single sample: $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \frac{1}{m} \sum_{i=1}^{m} g_{k_i}(\mathbf{x}_k)$.

3.1 Adaptive Learning Rate

There are some strategies of choosing the step sizes.

3.1.1 Adagrad

It is based on the experience that infrequent parameters can give more information than the frequent parameters. Thus it is designed to perform larger updates for infrequent and smaller updates for frequent parameters.

It individually adapts the learning rates of all model parameters by scaling them inversely proportional to the square root of the sum of all the historical squared values of the gradient.

Algorithm 3 Adagrad Algorithm

1. Sample a minibatch of m examples from the training set $\{x_{k_1}, x_{k_2}, \ldots, x_{k_m}\}$ with corresponding targets y_{k_i};
2. Compute the gradient:$g_k = \frac{1}{m} \sum_{i=1}^{m} g_{k_i}(\mathbf{x})$;
3. Accumulate squared gradient: $r_{k+1} = r_k + g_{k+1} \odot g_{k+1}$, where $\odot$ is multiplication applied element-wise;
4. Update: $\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\varepsilon}{\delta + \sqrt{r_{k+1}}} \odot g_k$ (Division and square root applied element-wise).

3.1.2 RMSprop

RMSProp uses an **exponentially decaying average**to discard history from the extreme past.

Algorithm 4 RMSprop Algorithm

1. Sample a minibatch of m examples from the training set $\{x_{k_1}, x_{k_2}, \ldots, x_{k_m}\}$ with corresponding targets y_{k_i};
2. Compute the gradient:$g_k = \frac{1}{m} \sum_{i=1}^{m} g_{k_i}(\mathbf{x})$;
3. Accumulate squared gradient: $r_{k+1} = \rho r_k + (1-\rho) g_{k+1} \odot g_{k+1}$, where $\odot$ is multiplication applied element-wise;
4. Update: $\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\varepsilon}{\delta + \sqrt{r_{k+1}}} \odot g_k$ (Division and square root applied element-wise).

3.1.3 Adam

Adaptive Moment Estimation (Adam) is another method that computes adaptive learning rates for each parameter. It can be seen as a variant on the combination of RMSProp and momentum.

3.2 Variance Reduction

The stochastic gradient descnet is not always descent due to its inherent variance.

Algorithm 5 Adam

Sample a minibatch of m examples from the training set $\{x_{k_1}, x_{k_2}, \ldots, x_{k_m}\}$ with corresponding targets y_{k_i};
Compute the gradient:$g_k = \frac{1}{m}\sum_{i=1}^{m} g_{k_i}(\mathbf{x})$;
Update biased first moment estimate: $s_{k+1} = \rho_1 s_k + (1-\rho_1)g_k$;
Update biased second moment estimate: $r_{k+1} = \rho_2 r_k + (1-\rho_2)g_k \odot g_k$;
Correct bias in first moment: $\hat{s}_{k+1} = \frac{s_{k+1}}{1-\rho_1^k}$;
Correct bias in second moment: $\hat{r}_{k+1} = \frac{r_{k+1}}{1-\rho_2^k}$;
Update: $\mathbf{x}_{k+1} = \mathbf{x}_k - \varepsilon \frac{\hat{s}_{k+1}}{\sqrt{\hat{r}_{k+1}}+\delta}$.

3.2.1 Stochastic Average Gradient

Like stochastic gradient (SG) methods, the SAG method's iteration cost is independent of the number of terms in the sum. However, by incorporating a memory of previous gradient values the SAG method achieves a faster convergence rate than black-box SG methods.

The SAG iterations take the form:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\alpha_k}{n}\sum_{i=1}^{n} y_{i_k},$$

where at each iteration a random index i_k is selected and we set:

$$y_{i_k} = \begin{cases} g_i(\mathbf{x}_k), & \text{if } i = i_k; \\ y_{i_{k-1}}, & \text{otherwise.} \end{cases}$$

3.3 SVRG

Rie Johnson and Tong Zhang introduced an explicit variance reduction method for stochastic gradient descent. It is proved that this method enjoys the same fast convergence rate as those of stochastic dual coordinate ascent (SDCA) and Stochastic Average Gradient (SAG) for *smooth and strongly convex* functions.

3.4 SVRG++

It is improved SVRG for non-strongly convex or sum-of-non-convex settings.The objective function is

$$F(x) = \sum_{i=1}^{n} f_i(x) + h(x),$$

where $h(x)$ is the regularization term. Let $g_i(x) = \nabla_x f_i(x)$.

3.5 SCSD

As a member of the SVRG family of algorithms, SCSG makes use of gradient estimates at two scales, with the number of updates at the faster scale being governed by a geometric random variable. Unlike most existing algorithms in this family,

Algorithm 6 SVRG

Parameters:s update frequency m and learning rate α.
Initialize:$\tilde{\mathbf{x}}$
Iterate for $s = 1, 2, \cdots$ $\tilde{\mathbf{x}} = \tilde{\mathbf{x}}_{s-1}$
$\tilde{\mu} = \frac{1}{n}\sum_{i=1}^{n} g_i$
$\mathbf{x}_0 = \tilde{\mathbf{x}}$
Iterate Randomly pick $i_k \in \{1, 2, \cdots, m\}$ and update weight:

$$\mathbf{x}_k = \mathbf{x}_{k-1} - \alpha(g_{i_k}(\mathbf{x}_{k-1}) - g_{i_k}(\tilde{\mathbf{x}}) + \tilde{\mu}).$$

end
Option I: set $\tilde{\mathbf{x}}_{s-1} = \mathbf{x}_m$
Option II:set $\tilde{\mathbf{x}}_{s-1} = \mathbf{x}_t$ for randomly chosen $t \in \{1, 2, \cdots, m-1\}$

Algorithm 7 SVRG++

1. During the sth epoch, $\hat{\mu}_{s-1} = \frac{1}{m_{s-1}}\sum_{i=1}^{m} g_i(\hat{x}_{s-1})$;
2. $m_s = 2^s m_0$
3. For $k \in \{0, 1, 2, \cdots, m_s - 1\}$

Randomly pick $i_k \in \{1, 2, \cdots, n\}$ and update weight:

$$\hat{w} = g_{i_k}(x_k^s) - g_{i_k}(\hat{x}_{s-1}) + \hat{\mu}_s,$$

$$x_{k+1}^s = argmin_x\{h(x) + \frac{1}{2\gamma}\|x - x_k^s\| + < x, \hat{w} >\}$$

4. $\hat{x}_s = \frac{1}{m_s}\sum_{k=1}^{m_s} x_k^s$;
5. $x_0^{s+1} = x_{m_s}^s$

both the computation cost and the communication cost of SCSG do not necessarily scale linearly with the sample size n; indeed, these costs are independent of n when the target accuracy is low.

Algorithm 8 SCSD

1.During the sth epoch and $s \in \{1, 2, \cdots, T\}$, $\hat{\mathbf{x}} = \hat{\mathbf{x}}_{s-1}$, $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{m} g_i(\hat{\mathbf{x}})$,$\mathbf{x}_0 = \hat{\mathbf{x}}$;
2.For $k \in \{1, 2, \cdots, M\}$, where $M \sim Geom(\frac{B}{B+1})$ Randomly pick $i_k \in \{1, 2, \cdots, n\}$ and update weight:

$$\mathbf{x}_k = \mathbf{x}_{k-1} - \alpha(g_{i_k}(\mathbf{x}_{k-1}) - g_{i_k}(\hat{\mathbf{x}}) + \hat{\mu}).$$

3.In this epoch:$\hat{\mathbf{x}}_s = \mathbf{x}_M$
4.(Strongly convex case)$\hat{\mathbf{x}} = \mathbf{x}_T$ or (Not strongly convex case)$\hat{\mathbf{x}} = \frac{1}{T}\sum_{k=1}^{T} \mathbf{x}_k$