A Study of Stochastic Gradient Descent Methods for L-1 and Elastic Net Regression

Limeng Cui

Pennsylvania State University

December 6, 2018

Introduction of SGD Methods

Stochastic Gradient Descent (SGD) is often used when the number of samples is large.

- Pro: it can reduce the computational cost of each update.
- Con: the convergence rate of SGD is not as good as GD.

In Stochastic Average Gradient (SAG), at the kth iteration, both the gradient at this step and the average of the previous $n-1$ gradients are taken into consideration.

- Pro: SAG is proved to be a linear convergence algorithm, which is much faster than SGD.
- Con: no difference between the computational complexity of SGD and SAG, but the memory of SAG is much larger

Stochastic Variance Reduced Gradient (SVRG) proposes a very important concept called "variance reduction".

- Pro: SVRG is proved to be a linear convergence algorithm, which is much faster than SGD. In addition, the memory is saved.

Proximal Operator

The proximal operator can be used as an approximation of the gradient for further gradient descent.
The proximal operator is an operator associated with a convex function h defined by:

$$prox_h(x) = arg \min_z h(x) + \frac{1}{2}||x - z||^2$$

where h is non differentiable. Note when h is smooth, the proximal operater is the gradient.

The LASSO problem is to minimize the following objective function:

$$g(x) + h(x) = \frac{1}{2}||Ax - b||_2^2 + \lambda||x||_1$$

where $g(x)$ denotes the L-2 norm of the residuals, $h(x)$ denotes the LASSO penalty.

As $g(x)$ is differentiable, we can obtain the following update rule by using the proximal gradient decent method:

$$\begin{aligned} x_{k+1} &= arg\min_x\{\lambda||x||_1 + \frac{1}{2\eta_k}||x - (x_k - \eta_k \nabla g(x_k))||_2^2\} \\ &= prox_{\eta_k h}(x_k - \eta_k \nabla g(x_k)) \end{aligned} \tag{1}$$

In each iteration, we first pick j uniformly at random and compute:

$$v_k = \frac{1}{b} \sum_{i_k \in I_k} (\nabla g_{i_k}(x_k)/n - \nabla g_{i_k}(\alpha_{i_k,k})/n) + \mu_k$$

Then, we can obtain the following update rule by using the proximal gradient decent method with v_k:

$$x_{k+1} = prox_{\eta_k h}(x_k - \eta_k v_k)$$

We update the intermediate values:

$$\alpha_{j,k+1} = x_k, \ \alpha_{-j,k+1} = \alpha_{-j,k}$$

To reduced the variance introduced by random sampling, SVRG computes the full batch periodically. Specifically, SVRG maintains an estimate $\tilde{x}$ of the optimal point x^*, which is updated after every m iterations.
In each iteration, we first randomly pick I_k of size b from $\{1, \ldots, n\}$ and compute:

$$v_k = \frac{1}{b} \sum_{i_k \in I_k} (\nabla g_{i_k}(x_k) - \nabla g_{i_k}(\tilde{x})) + \tilde{\mu}$$

Similarly, we can obtain the following update rule by using the proximal gradient decent method with v_k:

$$x_{k+1} = prox_{\eta_k h}(x_k - \eta_k v_k)$$

Extension to Elastic Net

The Elastic Net is a regularized regression method that linearly combines the L-1 and L-2 penalties of the LASSO and Ridge methods:

$$h(x) = \lambda_1 ||x||_1 + \frac{\lambda_2}{2} ||x||_2^2 \tag{2}$$

where the second term represents Ridge penalty. Thus, the proximal operator of $h(x)$ will be:

$$prox_{\eta_k h}(x) = (\frac{1}{1 + \eta_k \lambda_2}) prox_{\eta_k \lambda_1 ||x||_1}(x) \tag{3}$$

Numerical Experiments

Generate 100 samples with 2 dimensions randomly as data. Add noises $N(0, 0.25)$. (Step size: 0.1; Batch size: 10; Max epoch: 500.)
The real-world dataset ex1data2 contains a set of housing prices.

	ProxSGD	**ProxSAG**	**ProxSVRG**
Time(s)	0.123991	**0.039109**	0.052127
MSE	0.614088	0.615023	**0.613992**
Time(s)	8.746721	**8.717360**	9.466090
MSE	**0.392815**	0.405694	0.399983

Table: L-1 Regression Results (Top: Toy; Down: ex1data2)

	ProxSGD	**ProxSAG**	**ProxSVRG**
Time(s)	**7.089952**	7.335615	7.583075
MSE	3.080478	3.047427	**3.042382**
Time(s)	8.743878	4.679659	**0.505060**
MSE	**0.393503**	0.426037	0.424497

Table: Elastic Net Results

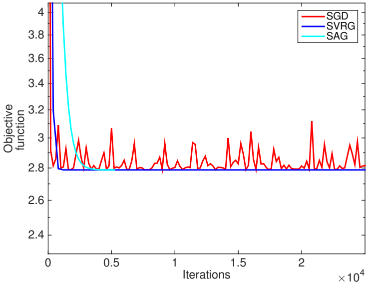

Figure: Objective function on ex1data2 for Elastic Net regression

Conclusion

- The cost of SGD descends fast at first. SAG and SVRG converge quickly.
- The max epoch and stopping criteria need to be selected carefully.
- As for SAG, the memory cost increases quickly as the dataset grows. This algorithm exchanges memory cost for time.
- SVRG converges quickly and achieves the best results in most cases among all three algorithms.