Difference between SGD and random reshuffling

Monde, Diego; Voracek, Vojtech and Wohlleben, Kilian
Faculty of Computer Science, University of Vienna, Vienna

***Abstract*—In this document we will present the difference between stochastic gradient descent and random reshuffling and provide empirical evidence for the latters superiority.**

I. Motivation

We consider unconstrained, finite-sum minimization problems or an empirical risk minimization:

$$\min_{x\in\mathbb{R}} f(x) = \sum_{i=1}^{n} f_i(x), \tag{1}$$

where f is a strongly convex function which ensures that there exists a unique optimal solution whhich is denoted by x^*. We also assume that each individual function f_i is smooth with Lipschitz gradients and L-Lipschitz on a bounded domain. This assumption helps us to make a particular convergence analysis. These types of problems are common in many areas of machine learning, one example being Linear Regression.

Gradient descent [3]: Traditional approach to minimizing convex functions.

Algorithm 1: Gradient descent

$x := x_0$
for *epochs* `t=1,...,T` **do**
  $x_{t+1} := x_t - \alpha \bigtriangledown f(x_t)$
end

A initial vector x_0, a number of epochs T, and a step size α are defined by the user.

Drawback of gradient descent: For large n it is computationally expensive to evaluate the full gradient.

Instead of GD one can use the Stochastic gradient descent [4] since $\bigtriangledown f(x) = \sum_{i=1}^{n} \bigtriangledown f_i(x)$.

Drawbacks of SGD and also the motivation to introduce Random reshuffling [6]:
The specific sample may be chosen more frequently than others. On the other hand, Random reshuffling guarantees that all samples are selected at the same frequency.

Algorithm 2: Stochastic gradient descent

$x := x_0$
for *epochs* `t=1, ..., T` **do**
  for `i=1, ..., n` **do**
    Sample $j \in \{1, ..., n\}$ uniformly.
    $x_t^{i+1} := x_t^i - \alpha \bigtriangledown f_j(x_t^i)$
  end
  $x_{t+1} = x_t^n$
end

Random reshuffling:
In each epoch t, we sample indices $[\pi_1, \pi_2, ..., \pi_n]$ without replacement from $\{1, 2, ..., n\}$, in other words, $[\pi_1, \pi_2, ...\pi_n]$ is a random permutation of the set $\{1, 2, ..., n\}$ and than perform n iterations of the following form

$$x_t^{i+1} := x_t^i - \alpha \bigtriangledown f_{\pi_i}(x_t^i)$$

Algorithm 3: Random reshuffling

$x := x_0$
for *epochs* `t=1, ..., T` **do**
  Sample a permutation
  $[\pi_1, \pi_2, ..., \pi_n]$ of $\{1, 2, ..., n\}$
  for `i=1, ..., n` **do**
    $x_t^{i+1} := x_t^i - \alpha \bigtriangledown f_{\pi_i}(x_t^i)$
  end
  $x_{t+1} = x_t^{n+1}$
end

II. Optimization problems

We start with some benchmark functions to analyze the difference between SGD and RR. Than we used those algorithms to solve a real-life problem.

A. *Sphere function*

Definition [5]:

$$f(x) = \sum_{i=1}^{n} x^2$$

Components of gradient:

$$\bigtriangledown f_i(x) = [0, ..., 0, 2 \cdot x_i, 0, ..., 0]$$

Global minimum:

$$f(x^*) = 0$$

$$x^* = [0, ..., 0]$$

Strong-convexity constant $\mu = 2$, Lipschitz constant $L = 2$, number of components of the gradient: n. It is presumable one of the easiest continuous domain optimization problem.

B. The component function

Definition [6]:

$$f_1(x) = \frac{1}{2}(x-1)^2, f_2(x) = \frac{1}{2}(x+1)^2 + \frac{x^2}{2}$$

$$f(x) = f_1(x) + f_2(x) = \frac{3}{2}x^2 + 1$$

Components of gradient:

$$\bigtriangledown f_1(x) = x - 1, \bigtriangledown f_1(x) = 2x + 1$$

Global minimum:

$$f(x^*) = 1$$

$$x^* = 0$$

Strong-convexity constant $\mu = 3$, Lipschitz constant $L = 3$, number of components of the gradient: 2. This is an example of a function where RR is provably better than SGD. [6]

C. Inverse Bell Curve

Definition :

$$f(x) = \sum_{i=1}^{n} 1 - e^{-\frac{x_i^2}{2\alpha}},$$

Where α is a scaling parameter. Components of gradient:

$$\bigtriangledown f_i(x) = \frac{x_i}{\alpha} e^{-\frac{x_i^2}{2\alpha}}$$

Global minimum:

$$f(x^*) = 0$$

$$x^* = [0, ..., 0]$$

This is an example of a non-convex function. We want to see, whether SGD and RR behave differently here.

D. Least squares linear regression

Definition [7]:

$$f(x) = \sum_{i=1}^{n} (a_i^T x - b_i)^2 = ||Ax - b||^2,$$

where $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^{n \times 1}$ are user-specified. Components of gradient:

$$\bigtriangledown f_i(x) = 2a_i(a_i^T x - b_i)$$

Global minimum:

$$f(x^*) = ||A(A^T A)^{-1} A^T b - b||$$

$$x^* = (A^T A)^{-1} A^T b$$

Strong-convexity constant $\mu = \lambda_{min}(2A^T A)$, Lipschitz constant $L = \lambda_{max}(2A^T A)$, where $\lambda_{min}(\cdot), \lambda_{max}(\cdot)$ denotes the smallest and the largest eigenvalues. respectively. The number of components of the gradient: n.

III. Experiments

In this work, we use the step size $\alpha = c/(t+1)^s$, where $c > 0, s \in (0, 1]$ for t-th epoch. If we consider q-suffix[1] averages of the iterates for some $q \in (0, 1]$ and step size $\alpha = c/(t+1)^s$, one can show that those iterates of both algorithms (SGD, RR) converge almost surely at rate $\mathcal{O}(1/t^s)$ to the optimal solution [6]. The specific requirements on the functions f_i decribed in the I section are necessary to prove this.

We say that the algorithm converged if:

$$||x - x^*|| \leq \epsilon$$

or alternatively:

$$||\overline{x}_{q,k} - x^*|| \leq \epsilon$$

for a user-specified ϵ.

Settings

- $\epsilon = 10^{-7}$
- $c = 3$ ($c = 50$ for linear regression, $c = 30$ for Inverse Bell Curve)
- $s = 0.9$
- $q = 0.2$
- $T = 2000$
- $x_0 = \sim \mathcal{U}^d_{[-10,10]}$
- $\alpha = 10$

The aim of this work is to compare performance of RR and SGD on specific functions 1. For this purpose, we measured the performance of those two algorithms on the Sphere function of different dimensions, on the component function, on the real-world linear regression

[1]q-suffix average is obtained by averaging the last qk iterates at iteration k [6]

- the Diabetes dataset provided by sklearn [8], [9] and for a self posed non-convex problem. For the diabetes dataset, the dimension is 10, and the optimized function is a sum of 442 independent functions.

IV. Results

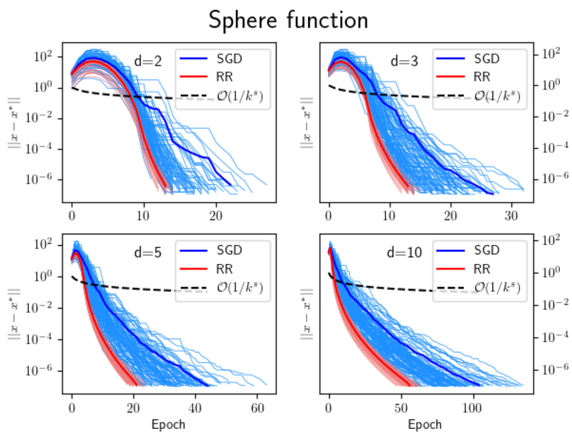

Figure 1. 100 runs of SGD and RR on the sphere function of different dimensions (2, 3, 5, 10). The bold curve captures the mean over all runs of certain algorithm. The graph shows the distance of x to the optimal solution x^* over epochs.Moreover, the curve capturing the convergence rate $\mathcal{O}(1/k^s)$ is added.

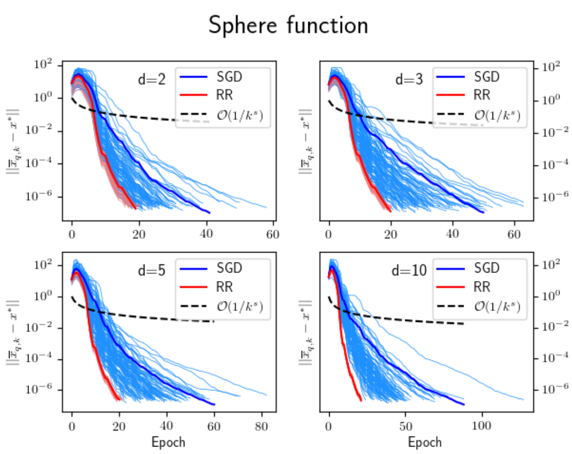

Figure 2. The graph is similar to 1. In contrast, this graph shows the distance of q-suffix average $\overline{x}_{q,k}$ to the optimal solution x^* over iterations. Moreover, the curve capturing the convergence rate $\mathcal{O}(1/k^s)$ is added.

V. Conclusion

RR has outperformed SGD in all our experiments. We have specifically seen, that RR is much more stable and yields faster convergence. In the case of the sphere function and the inverse bell curve that is due to the fact that all components of x are optimized over equally often, so that no dimension is "neglected". In the case of the component function we could confirm the theoretical superiority of RR proven in [6]. For the real world dataset, we have also seen, that optimizing over the entire dataset uniformly is better than leaving the frequency to chance.

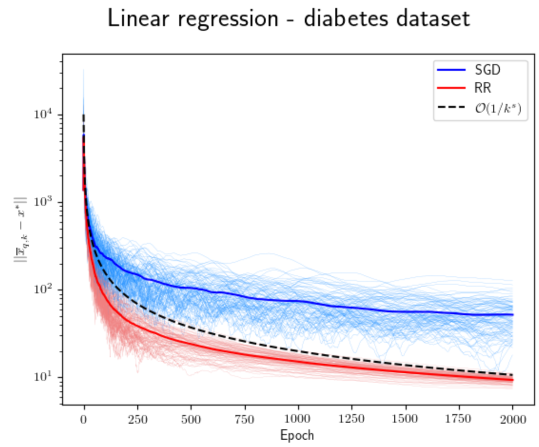

Figure 3. 100 runs of SGD and RR on the linear regression problem. The bold curve captures the mean over all runs of certain algorithm. The graph shows the distance of q-suffix average $\overline{x}_{q,k}$ to the optimal solution x^* over epochs. Moreover, the curve capturing the convergence rate $\mathcal{O}(1/k^s)$ is added.

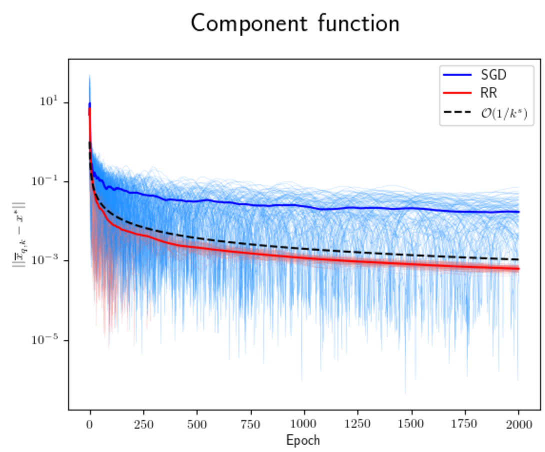

Figure 4. 100 runs of SGD and RR on the component function. The bold curve captures the mean over all runs of certain algorithm. The graph shows the distance of q-suffix average $\overline{x}_{q,k}$ to the optimal solution x^* over epochs. Moreover, the curve capturing the convergence rate $\mathcal{O}(1/k^s)$ is added.

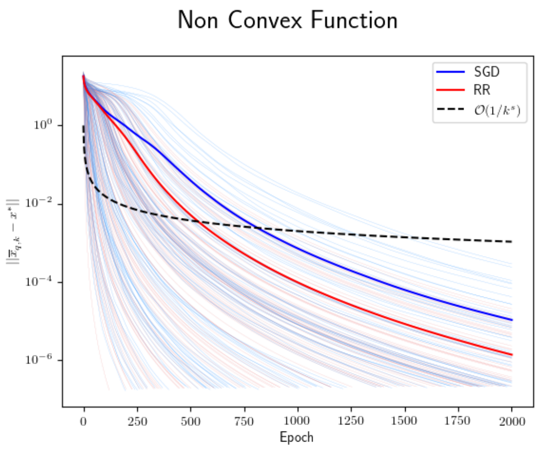

Figure 5. 100 runs of SGD and RR on the non-convex function. The bold curve captures the mean over all runs of certain algorithm. The graph shows the distance of q-suffix average $\overline{x}_{q,k}$ to the optimal solution x^* over epochs. Moreover, the curve capturing the convergence rate $\mathcal{O}(1/k^s)$ is added.

References

[1] D. P. Bertsekas, A. Nedic, and A. E. Ozdaglar, *Convex analysis and optimization*. Athena Scientific, 2003.

[2] M. O'Searcoid, *Metric Spaces*, ser. Springer Undergraduate Mathematics Series. Springer London, 2006. [Online]. Available: https://books.google.cz/books?id=aP37I4QWFRcC

[3] M. A. Cauchy, "Methode g en erale pour la r esolution des syst emes ‘ d'equations simultan ees," 1847.

[4] B. T. Poliak, *Introduction to optimization*. New York: Optimization Software, Publications Division, 1987.

[5] K. A. De Jong, "Analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, The University of Michigan, 1975.

[6] M. Gürbüzbalaban, A. Ozdaglar, and P. A. Parrilo, "Why random reshuffling beats stochastic gradient descent," *Mathematical Programming*, vol. 186, no. 1-2, p. 49–84, Oct 2019. [Online]. Available: http://dx.doi.org/10.1007/s10107-019-01440-w

[7] F. Galton, "Regression towards mediocrity in hereditary stature." *The Journal of the Anthropological Institute of Great Britain and Ireland*, vol. 15, pp. 246–263, 1886. [Online]. Available: http://www.jstor.org/stable/2841583

[8] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *The Annals of Statistics*, vol. 32, no. 2, pp. 407 – 499, 2004. [Online]. Available: https://doi.org/10.1214/009053604000000067

[9] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. Vanderplas, A. Joly, B. Holt, and G. Varoquaux, "Api design for machine learning software: experiences from the scikit-learn project," 2013.