Efficiency of CD methods on huge-scale optimization problems

Namit Katariya, Nishant Totla

November 11, 2011

Abstract

We studied some new methods for solving huge-scale optimization problems. For these problems, full-vector operations are expensive, so these methods propose techniques that make partial randomized updates of the state. And then it is shown that expected values converge to the optimal value. We studied and implemented such methods for unconstrained and constrained convex optimization problems, including an accelerated version. We then implemented the same on a simple example.

1 Introduction

A general convex optimization problem is of the following form

$$\min_{x\in\mathbb{R}^n} f(x)$$
$$\text{s.t. } x \in Q$$

where Q is a closed convex domain.
There are various methods to solve these problems. Some cases can be solved analytically, while others need numerical methods. The most popular among them is the standard gradient descent method. For each such method, it is important to prove convergence and evaluate the rate of convergence.

1.1 Standard gradient descent

Consider the unconstrained minimization problem

$$\min_{x\in\mathbb{R}^n} f(x)$$

Where f has component-wise Lipschitz continuous gradient with Lipschitz constant M. Then consider the following method.

Choose $x_0 \in \mathbb{R}^n$. For $k \geq 0$ iterate
1. Choose $i_k = arg\max_{1\leq i\leq n} |\nabla_i f(x_k)|$
2. Update $x_{k+1} = x_k - \frac{1}{M}\nabla_{i_k} f(x_k) e_{i_k}$

We note that this satisfies $f(x_k) - f^* \leq \frac{C}{k+4}$, $k \geq 0$, so it converges. But there are the following points that we should note:

1. At each test point, the whole gradient needs to be computed. This might typically be expensive, unless the vector is available.

2. For functions with Lipschitz continuous gradients, the Lipschitz constant for the entire function might be over-estimated, and hence, the step sizes will be small.

3. **Computational complexity** There has been increasing interest in optimization problems of huge size (eg. Internet applications, telecommunications). In these, even function value computation could be very expensive, and moreover, data could be distributed in space and time. Essentially, computation of a *coordinate directional derivative* can be much simpler than computing either a functional value or directional derivative along arbitrary direction.

In the following sections, we consider optimization problems on the space $\mathbb{R}^N$, and fix a decomposition of $\mathbb{R}^N$ on n subspaces:

$$\mathbb{R}^N = \bigotimes_{i=1}^{n} \mathbb{R}^{n_i},\ N = \sum_{i=1}^{n} n_i$$

We also define the partition of the unit matrix

$$I_N = (U_1, \ldots, U_n) \in \mathbb{R}^{N \times N},\ U_i \in \mathbb{R}^{N \times n_i},\ i = 1, \ldots, n$$

And the partial gradient of $f(x)$ in $x^{(i)}$ is defined as $f_i'(x) = U^T \nabla f(x) \in \mathbb{R}^{n_i},\ x \in \mathbb{R}^N$. We can also assume Lipschitz constants for each subspace separately.

2 Randomized coordinate descent methods

Now we describe some randomized gradient descent methods

2.1 Random Coordinate Descent Method (UCDM)

Here we consider the unconstrained minimization problem of a convex and differentiable objective function f over $\mathbb{R}^N$.

Method $RCDM(\alpha, x_0)$
For $k \geq 0$ iterate
1) Choose $i_k = \mathcal{R}_\alpha$
2) Update $x_{k+1} = T_{i_k}(x_k)$

The update is given as $T_i(x) \stackrel{def}{=} x - \frac{1}{L_i} U_i f_i'(x)^{\#},\ i = 1, \ldots, n$. $\mathcal{R}_\alpha$ is a random counter, and selects a subspace to move probabilistically. It assigns probabilities to each subspace based on α and the Lipschitz constant for that subspace. Large Lipschitz constants, in some sense, imply more steepness in that subspace. This is the intuition behind weighting the probabilities by Lipschitz constant since choosing a subspae with a high Lipschitz constant can give you large decrements in the function values.

We note the convergence result for this method.

$$\phi_k - f^* \leq \frac{2}{k+4} \cdot [\sum_{j=1}^{n} L_j^\alpha] \cdot R_{1-\alpha}^2(x_0)$$

Here ϕ_k is the expected value of the vector x after k iterations. $R_{1-\alpha}^2(x_0)$ is a constant. Please refer [1] for the complete details. The essence of this method is that at each step, only the partial gradient is required to be computed. Additionally, each subspace has its own Lipschitz constant.

When the function is strongly convex

- We don't perform the optimization a lot of times to talk of expected value
- So we need to talk about the "goodness" of a single run of our methods. Hence talking about confidence intervals make sense
- A result of the form "Probability that (function value at k^{th} iteration - optimal function value) is less than a small ϵ is greater than a confidence β" can be proven when f is strongly convex.

2.2 Uniform Coordinate Descent Method (UCDM)

We now consider the constrained minimization problem

$$\min_{x \in Q} f(x)$$

where $Q = \bigotimes_{i=1}^{n} Q_i$, and the sets $Q_i \subset \mathbb{R}^{n_i}$ are closed convex sets. The unconstrained coordinate update is given by

$$u^{(i)}(x) = arg\min_{u^{(i)} \in Q_i} [\langle f_i'(x), u^{(i)} - x^{(i)} \rangle + \frac{L_i}{2} ||u^{(i)} - x^{(i)}||_{(i)}]$$

$$V_i(x) = x + U_i^T (u^{(i)}(x) - x^{(i)})$$

Method $UCDM(x_0)$
For $k \geq 0$ iterate
1) Choose i_k randomly by uniform distribution on $\{1, \ldots, n\}$
2) Update $x_{k+1} = V_{i_k}(x_k)$

We note that to calculate $u^{(i)}(x)$, we need to find $arg\min$ over the restricted sets Q_i only. This often simplifies the problem considerably, as we shall present in an example later. The convergence rate, as earlier, is proportional to $\frac{1}{k}$, and interestingly depends on the number of dimensions also.

2.3 Accelerated Coordinate Descent Method (ACDM)

This is simply a randomized version of Nesterov's deterministic multi-step gradient descent method. This works for strongly convex objective functions, and has a convergence rate proportional to $\frac{1}{k^2}$. The method description is elaborate, so we have skipped that here.

3 Experiments

We implemented the above algorithms and tested our implementation on a few simple, elementary optimization problems - needless to say they give the optimal answer.

The results of the tests on a few non-trivial optimization problems and why that specific problem is hard to put down here and can be explained in detail during the presentation.

Although, the utility of these methods can be explained using the following example. Consider the function $f(x) = x^T Mx$ where $M \succ 0$. Then $f(x)$ is differentiable and the gradient is Lipschitz continuous. Clearly, Lipschitz constants can easily be found if we split the problem into n dimensions, and consider each dimension to be a subspace. Also, computations in each step become simple since we only have one-dimensional optimization problems to solve now.

4 Issues and problems

1. Nothing about how to choose the decomposition of the space is discussed, and we do not yet fully realize how changing the decomposition might suit the specific problem at hand.
2. Finding out Lipschitz constants in each subspace might not be trivial
3. The update rules of RCDM/UCDM themselves involve solving a optimization problem. This is considered an atomic step in the convergence analysis.
4. Complexity of one iteration of ACDM, for some cases, is rather high because one of the steps involves operations with full dimensional vectors.

References

[1] Yu. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. January 2010.