OPTIMIZING COMPOSITE FUNCTIONS

A *
A@gmail.com

June 23, 2015

CONTENTS

1 Introduction 2
2 Illustrations 2
2.1 CVX . . . 2
2.2 Proximal Gradient Descent . . . 3
2.3 ADMM . . . 6
2.4 Newton-type Methods . . . 8
2.5 Coordinate Descent . . . 9
2.6 SGD . . . 10
3 Experiments and Discussion 10
3.1 Figure Composed of Subfigures . . . 10

LIST OF FIGURES

Figure 1 Convergence comparison . . . 10

LIST OF TABLES

ABSTRACT

Many problems of relevance in bioinformatics, signal processing, and statistical learning can be formulated as minimizing a composite function: a smooth function and non-smooth function. There are some generic methods that could be used to solve this optimization problem theoretically, such as CVX, FISTA, ADMM and PNOPT. In addition, the stochastic methods including SGD and stochastic ADMM are well suited to handle large data sets. We will introduce these methods one by one, and analyze their strengths and weaknesses through the numerical experiments.

* *Department of Computer Sciences, SJTU*

1 INTRODUCTION

In this survey, we are concerned with the following optimization problem:

$$\min_{\mathbf{x}\in\mathbb{R}^p} \mathcal{F}(\mathbf{x}) \triangleq g(\mathbf{x}) + \Psi(\mathbf{x}), \tag{1}$$

where g is a convex and continuously differentiable loss function, and Ψ is usually defined as a (non-smooth) convex penalty function or regularizer. For example, $\Psi(\mathbf{x}) \triangleq \lambda_1\|\mathbf{x}\|_2$ is a ridge regression , and $\Psi(\mathbf{x}) \triangleq \lambda_1\|\mathbf{x}\|_1$ defines a LASSO penalty [1].

Indeed, there are plenty of machine learning models, which can be cast into this formulation in (1), such as the generalized lasso [2], the group lasso for logistic regression [3]. Moreover, many real-world problems benefit from these models such as Gene expression and time-varying network. In this paper we mainly study computational issue of the model in (1).

There are a variety of methods to solve these models, such as SpaRSA [4] and LARS [5]. We focus on the generic methods:

1. CVX. We can use CVX [6] to conveniently formulate and solve many complex convex programs. Constraints and objectives that are expressed using these rules are automatically transformed to a canonical form and solved.
2. Proximal Gradient Descent. The base operation of proximal methods is evaluating the proximal operator of a function. There are a large number of examples of proximal operators that commonly arise in practice like LASSO and elastic net.
3. ADMM. The alternating direction method of multipliers (ADMM) [7] is very similar to proximal methods but more general, and ADMM is well suited to distributed convex optimization.
4. Newton-type methods. The main idea is to approximate the smooth part of the objective function using the second order information. Then the computation is shift to solve the subproblem.
5. Coordinate Descent. When the objective is separable, we could Optimize the model according to each coordinate.
6. SGD. In stochastic gradient descent, the true gradient is approximated by a gradient at a single example. SGD is very efficient to obtain a less accuracy solutions.

We will introduce these methods one by one, and analyze their strengths and weaknesses through the numerical experiments.

2 ILLUSTRATIONS

In this section, we illustrate the basic concept of these methods in detail. To understand these methods more thoroughly, we give some comparative experiments.

2.1 CVX

2.1.1 *Basic Concept*

CVX is a Matlab-based modeling system for convex optimization. The convex optimization problems is constructed by *disciplined convex programming*. The default solver is currently SDPT3 [8] which is designed to solve

conic programming problems whose constraint cone is a product of semidefinite cones, second-order cones, and/or non-negative orthants. It employs a predictor-corrector primal-dual path-following method, with either the HKM or the NT search direction. CVX utilizes symmetric primal/dual solvers that simply cannot support the functions from the exponential family natively. CVX constructed a successive approximation heuristic that allows the symmetric primal/dual solvers to support the exponential family of functions. However, it will sometimes fail to converge even for problems known to have solutions. Even when it does converge, it is several times slower than the standard solver, due to its iterative approach.

2.1.2 *Matlab Implementation*

CVX turns Matlab into a modeling language, allowing constraints and objectives to be specified using standard Matlab expression syntax. For example, consider the following convex optimization model (LASSO):

$$\begin{aligned} \min \ & \|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2 \\ \text{s.t. } & \|\mathbf{x}\|_1 \leqslant d. \end{aligned}$$

or equivalent,

$$\min \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2 + \lambda\|\mathbf{x}\|_1 \tag{2}$$

We can use CVX to solve this model via the following matlab code:

Listing 1: Solve LASSO using CVX

```
cvx_begin
  variable x(n)
  minimize(norm(A * x - b, 2))
  subject to
    norm(x, 1) <= d
cvx_end
```

CVX will solve many medium and large scale problems, provided they have exploitable structure (such as sparsity), eliminated for loops and functions like log and exp that require successive approximation. We test the performance of CVX in several datasets, the Matlab code is located in folder *matlab/cvx_test*.

2.2 Proximal Gradient Descent

2.2.1 *Basic Concept*

Proximal algorithms can be viewed as a standard tool for non-smooth, constrained, large-scale, or distributed problems. They are very generally applicable, but they turn out to be especially well-suited to problems of recent and widespread interest involving large or high-dimensional datasets. The base operation of proximal algorithms is evaluating the proximal operator of a function, which involves solving a small convex optimization problem. These subproblems can be solved with standard methods, but they often admit closed form solutions or can be solved very quickly with simple specialized methods.

The proximal operator $\mathbf{prox}_{\lambda f} : \mathbb{R}^n \to \mathbb{R}^n$ of the scaled function λf, where $\lambda > 0$, is defined by

$$\mathbf{prox}_{\lambda f}(v) = \underset{x}{\operatorname{argmin}} \left(f(x) + \frac{1}{2\lambda}\|x - v\|_2^2 \right). \tag{3}$$

There are various interpretations of the proximal operator illustrated in [9].

We could evaluate the proximal operator (3) using iterative methods such as gradient methods and L-BFGS when f is smooth. If the minimization is over a convex constrained set, we can use CVX to obtain the proximal operator. A fully separable function f reduces to evaluating the proximal operator of a scalar convex function. For a non-smooth example, if $f(x) = |x|$, then we have that

$$S_\lambda(v) = \mathbf{prox}_{\lambda f}(v) = \begin{cases} v - \lambda, & v > \lambda \\ 0, & |v| \leqslant \lambda \\ v + \lambda, & v < -\lambda, \end{cases} \tag{4}$$

which is called *soft thresholding*.

From the the Moreau envelope perspective, that Moreau envelope or Moreau-Yosida regularization $M_{\lambda f}$ of the function λf is define as,

$$M_{\lambda f}(v) = \inf_x \left(f(x) + \frac{1}{2\lambda}\|x - v\|_2^2 \right).$$

The Moreau envelope M_f is essentially a smoothed or regularized form of f: It has domain $\mathbb{R}^n$, even when f does not, and it is continuously differentiable, even when f is not. In addition, the sets of minimizers of f and M_f are the same. The problems of minimizing f and M_f are thus equivalent, and the latter is always a smooth optimization problem. Because $M_f^{**} = M_f$ and the infimal convolution is dual to addition, it follows that

$$\begin{aligned} M_f &= ((M_f)^*)^* \\ &= (f^* + \frac{1}{2}\|\cdot\|_2^2)^*. \end{aligned} \tag{5}$$

In general, the conjugate ϕ^* of a closed proper convex function ϕ is smooth when ϕ is strongly convex. This suggests that the Moreau envelope M_f can be interpreted as obtaining a smooth approximation to a function by taking its conjugate, adding regularization, and then taking the conjugate again.

From Eqn. (3), we know that proximal minimizing is smooth the primal objective. For example, applying this technique to $|x|$, gives the Huber function.

$$(|x|^* + \frac{1}{2}\|x\|_2^2)^* = \phi^{\text{huber}}(x)$$

This perspective is very related to recent work by Nesterov[10]. Then minimize the non-smooth objective is equivalent to minimize the smoothed objective M_f.

There is a wide literature on applying various proximal algorithms to particular problems or problem domains, we describe the proximal minimization algorithm and proximal gradient method respectively.

2.2.2 *Proximal Minimizing Algorithm*

The proximal minimization algorithm, also called proximal iteration or the proximal point algorithm, is

$$x^{k+1} = \mathbf{prox}_{\lambda f}(x^k), \tag{6}$$

where f is a closed proper convex function, and x^k denotes the kth iterate of the algorithm. If f has a minimum, then x^k converges to the set of minimizers of f and $f(x^k)$ converges to its optimal value [11]. A simple interpretation is as disappearing Tikhonov regularization, that the quadratic (Tikhonov) regularization 'goes away' in the limit. Moreover, the proximal operator of the second-order approximation can be viewed as the trust-region method with a Levenberg-Marquardt update, that

$$\mathbf{prox}_{\lambda \hat{f}_v}(v) = v - (\nabla^2 f(v) + \frac{1}{\lambda}\mathbf{I})^{-1}\nabla f(v), \tag{7}$$
$$\text{where } \hat{f}_v = f(v) + \nabla f(v)^T(x-v) + \frac{1}{2}(x-v)^T\nabla^2 f(v)(x-v).$$

The (sub)problem (6) becomes easier for gradient method as more quadratic regularization is added. Here, 'easier' can mean fewer iterations, faster convergence, or higher reliability. The proximal algorithm would be useful in a situation where it is hard to minimize the function f (our goal), but easy (or at least easier) to minimize f plus a quadratic like iterative refinement for solving linear equations. Besides, it will play an important role in solving ill-conditioned smooth minimization problems.

2.2.3 *Proximal Gradient Method*

Consider the problem (1), where $g : \mathbb{R}^n \to \mathbb{R}$ and $\Psi : \mathbb{R}^n \to \mathbb{R} \cup +\infty$ are closed proper convex and g is differentiable. In this form, we split the objective into two terms, one of which is differentiable. The proximal gradient method is

$$\mathbf{x}^{k+1} = \mathbf{prox}_{\lambda^k \Psi}(\mathbf{x}^k - \lambda^k \nabla g(\mathbf{x}^k)), \tag{8}$$

where $\lambda^k > 0$ is a step size. The proximal gradient can be interpreted as majorization-minimization algorithm or fixed point iteration [9]. This method can be shown to converge with rate $\mathcal{O}(\frac{1}{k})$, and the accelerated version [12] (optimal first-order method) converges with rate $\mathcal{O}(\frac{1}{k^2})$. We could solve LASSO problem (2) directly using Eqn. (4) and (8). There is an excellent software package called TFOCS [13] is based on and contains several implementations of such accelerated proximal gradient methods. One of these accelerated proximal gradient methods (No7[14]) is described in Algorithm 1, the proximal operator of Ψ should be easy to evaluate. There are several ways to estimate L as discussed in [13].

Algorithm 1 Solve composite functions (1) via proximal gradient method

Require: $\mathbf{z}^0 \in \mathbb{R}^n$, Lipschitz estimate L
1: $\theta^0 \leftarrow 1, \mathbf{v}^0 \leftarrow \mathbf{z}^0, j \leftarrow 0$
2: **repeat**
3: $\quad \mathbf{y}^j \leftarrow (1-\theta^j)\mathbf{z}^j + \theta^j \mathbf{v}^j$
4: $\quad \mathbf{v}^{j+1} \leftarrow \underset{\mathbf{z}}{\operatorname{argmin}}\ \langle (\theta^j)^2 \sum_{i=0}^{j} (\theta^i)^{-1}\nabla g(\mathbf{y}^i), \mathbf{z}\rangle + \frac{1}{2}(\theta^j)^2 L\|\mathbf{z}-\mathbf{z}_0\|^2 + \Psi(\mathbf{z})$
5: $\quad \mathbf{z}^{j+1} \leftarrow \underset{\mathbf{z}}{\operatorname{argmin}}\ \langle \nabla g(\mathbf{y}^j), z\rangle + \frac{1}{2}L\|\mathbf{z}-\mathbf{y}^j\|^2 + \Psi(\mathbf{z})$
6: $\quad \theta^{j+1} \leftarrow 2/(1+(1+4/(\theta^j)^2)^{\frac{1}{2}})$
7: $\quad j \leftarrow j+1$
8: **until** some stopping condition is satisfied

2.2.4 *Dual Approach*

Unlike Nesterov [10]'s approach which applied the smoothing technique to the primal objective, we consider the dual approach presented in TFOCS[13]. The dual approach smooth the dual objective, specifically,

$$\begin{aligned} M_g(\mathbf{v}) &= \inf_{\mathbf{x}}(g(\mathbf{x}) + \frac{1}{2}\|\mathbf{x}-\mathbf{v}\|_2^2) \\ &= (g(\mathbf{x})^* + \frac{1}{2}\|\mathbf{x}-\mathbf{v}\|_2^2)^* \\ &= (f(\mathbf{x}) + \frac{1}{2}\|\mathbf{x}-\mathbf{v}\|_2^2)^*. \end{aligned}$$

Minimizing $f(\mathbf{x}) + \frac{1}{2}\|\mathbf{x}-\mathbf{x}_0\|_2^2$ is equivalent to maximizing the dual $M_g(\mathbf{v})$. There is a more general model called smoothed conic dual (SCD) model introduced in TFOCS [13] which could solve the more general non-smooth models like the fused LASSO. It's easy to show that the gradient of the dual is Lipschitz continuous if the primal is strongly convex[10], then provably convergent, accelerated gradient methods in the Nesterov style are possible.

Consider the statistical model that $\Psi(\mathbf{x}) = \|\mathbf{F}\mathbf{x}\|_1$ in Eqn. (1), we could solve this model via SCD as shown in Algorithm 2. However, it does not solve the problem exactly because the proximal function $\frac{\rho}{2}\|\mathbf{x}-\mathbf{x}_0\|_2^2$, by applying proximal iteration (6) (which called continuation in TFOCS), we could obtain the exact solutions. The main burden of Algorithm 2 is line 4 which may be inefficient to find the minimizer.

Algorithm 2 Solve Composite Function via SCD

Require: $\mathbf{z}^0, \mathbf{x}^0, \rho > 0$

1: $\theta^0 \leftarrow 1, \mathbf{v}^0 \leftarrow \mathbf{z}^0, j \leftarrow 0$
2: **repeat**
3: $\quad \mathbf{y}^j \leftarrow (1-\theta^j)\mathbf{v}^j + \theta^j\mathbf{z}^j$
4: $\quad \hat{\mathbf{x}} \leftarrow \underset{\mathbf{x}}{\operatorname{argmin}}\ g(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{x}-\mathbf{x}_0\|_2^2 - \mathbf{x}^T\mathbf{F}^T\mathbf{y}^j$
5: $\quad \mathbf{z}^{j+1} \leftarrow \underset{\mathbf{z}:\|\mathbf{z}\|_\infty \leqslant 1}{\operatorname{argmin}}\ \frac{\theta^j}{2\delta^j}\|\mathbf{z}-\mathbf{z}^j\|_2^2 + \mathbf{z}^T\mathbf{F}\hat{\mathbf{x}}$
6: $\quad \mathbf{v}^{j+1} \leftarrow (1-\theta^j)\mathbf{v}^j + \theta^j\mathbf{z}^{j+1}$
7: $\quad \theta^{j+1} \leftarrow 2/(1+(1+4/(\theta^j)^2)^{\frac{1}{2}})$
8: $\quad j \leftarrow j+1$
9: **until** some stopping condition is satisfied

2.3 ADMM

2.3.1 *Basic Concept*

ADMM was first introduced in the mid-1970s by Gabay, Mercier, Glowinski, and Marrocco, though similar ideas emerged as early as the mid-1950s. It turns out to be equivalent or closely related to many other algorithms as well, such as Douglas-Rachford splitting from numerical analysis, Spingarn's method of partial inverses, Dykstra's alternating projections method, Bregman iterative algorithms for ℓ_1 problems in signal processing, proximal methods, and many others.

ADMM is an algorithm that is intended to blend the decomposability of dual ascent with the superior convergence properties of the method of multipliers. The algorithm solves problems in the form

$$\begin{aligned} &\min \ g(\mathbf{x}) + \Psi(\mathbf{z}) \\ &\text{s.t. } \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c}, \end{aligned} \tag{9}$$

with $\mathbf{x} \in \mathbb{R}^n$ and $z \in \mathbb{R}^m$, where $\mathbf{A} \in \mathbb{R}^{p \times n}$, $\mathbf{B} \in \mathbb{R}^{p \times m}$, and $c \in \mathbb{R}^p$. Assuming that f and g are convex, we form the augmented Lagrangian

$$L_\rho(\mathbf{x}, \mathbf{z}, \mathbf{y}) = g(\mathbf{x}) + \Psi(\mathbf{z}) + \mathbf{y}^T(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}\|_2^2.$$

ADMM consists of the iterations

$$\begin{aligned} \mathbf{x}^{k+1} &\leftarrow \operatorname*{argmin}_{\mathbf{x}} L_\rho(\mathbf{x}, \mathbf{z}^k, \mathbf{y}^k) \\ \mathbf{z}^{k+1} &\leftarrow \operatorname*{argmin}_{\mathbf{z}} L_\rho(\mathbf{x}^{k+1}, \mathbf{z}, \mathbf{y}^k) \\ \mathbf{y}^{k+1} &\leftarrow \mathbf{y}^k + \rho(\mathbf{A}\mathbf{x}^{k+1} + \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c}), \end{aligned}$$

where $\rho > 0$. The algorithm is very similar to dual ascent and the method of multipliers where the augmented Lagrangian is minimized jointly with respect to the two primal variables. In ADMM, on the other hand, $\mathbf{x}$ and $\mathbf{z}$ are updated in an alternating or sequential fashion, which accounts for the term alternating direction. Boyd et al. [7] explained the ADMM in detail.

There are many convergence results for ADMM discussed in the literature, under some assumptions, ADMM converges linearly [15]. However simple examples show that ADMM can be very slow to converge to high accuracy. Fortunately, it is often the case that ADMM converges to modest accuracy within a few tens of iterations which is sufficient for many applications.

2.3.2 *Proximal Operator In ADMM*

Consider the simple case where $\mathbf{A} = \mathbf{I}$, which appears frequently. Then the $\mathbf{x}$-update issue

$$\mathbf{x}^+ = \operatorname*{argmin}_{\mathbf{x}} f(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{x} - \mathbf{v}\|_2^2,$$

which is exactly the proximal operator as (3). The same method discussed in Section 2.2 could be applied to evaluating the update of ADMM. For instance, if $f(\mathbf{x}) = |\mathbf{x}|$, then the $\mathbf{x}$-update could be calculated by $S_{1/\rho}(\mathbf{v})$ as shown in Eqn. (4).

2.3.3 *Generalized LASSO*

The lasso problem can be generalized to

$$\min \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{F}\mathbf{x}\|_1 \tag{10}$$

where $\mathbf{F}$ is an arbitrary linear transformation. An important special case is fused LASSO when $\mathbf{F} \in \mathbb{R}^{(n-1)\times n}$ is the difference matrix,

$$\mathbf{F}_{ij} = \begin{cases} 1 & \text{if } i = j \\ -1 & \text{if } j = i + 1 \\ 0 & \textit{otherwise.} \end{cases}$$

In ADMM form, the problem (10) can be written as

$$\begin{aligned} \min \ & \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2+\lambda\|\mathbf{z}\|_1 \\ \text{s.t. } & \mathbf{F}\mathbf{x}-\mathbf{z}=0, \end{aligned}$$

which yields the ADMM iteration

$$\begin{aligned} \mathbf{x}^{k+1} &\leftarrow (\mathbf{A}^T\mathbf{A}+\rho\mathbf{F}^T\mathbf{F})^{-1}(\mathbf{A}^T\mathbf{b}+\rho\mathbf{F}^T(\mathbf{z}^k-\mathbf{u}^k)) \\ \mathbf{z}^{k+1} &\leftarrow S_{\frac{\lambda}{\rho}}(\mathbf{F}\mathbf{x}^{k+1}+\mathbf{u}^k) \\ \mathbf{u}^{k+1} &\leftarrow \mathbf{u}^k+\mathbf{F}\mathbf{x}^{k+1}-\mathbf{z}^{k+1}. \end{aligned}$$

It easy to see that ADMM is much powerful than proximal algorithm. Besides, ADMM is readily to do distributed optimization via global variable consensus [7].

2.4 Newton-type Methods

2.4.1 *Basic Concept*

In order to solve the problem more precisely in modest time, we resort to Newton approach which use the second information. More specifically, we use information from the second derivative to form a quadratic approximation to a function like (7). Consider the problem (1), we appropriate the smooth part g, which results

$$\hat{\mathcal{F}}(\mathbf{x})=\hat{g}(\mathbf{x})+\Psi(\mathbf{x})=g(\mathbf{x}_k)+\nabla g(\mathbf{x}_k)^T(\mathbf{x}-\mathbf{x}_k)+\frac{1}{2}(\mathbf{x}-\mathbf{x}_k)^T\mathbf{H}_k(\mathbf{x}-\mathbf{x}_k)+\Psi(\mathbf{x}), \quad (11)$$

where $\mathbf{H}_k$ is a $p\times p$ (semi)definite positive matrix as approximation to the Hessian of g at $\mathbf{x}=\mathbf{x}_k$. There are many strategies for choosing $\mathbf{H}_k$, such as BFGS and LBFGS [16]. There has been a flurry of activity about developments of Newton-type methods for minimizing composite functions in the literature. In particular, Lee et al. [17] focused on minimizing a composite function, which contains a convex smooth function and a convex nonsmooth function with a simple proximal mapping. They also analyzed the convergence rate of various proximal Newton-type methods. Schmidt et al. [18, 19] discussed a projected quasi-Newton algorithm, but the sub-iteration costs too much.

Compare to the proximal operator (3), Eqn. (11) could be expressed as the generalized proximal operator

$$\mathbf{gprox}_{\mathbf{H}\Psi}(\mathbf{v})=\underset{\mathbf{x}}{\operatorname{argmin}}\left(\Psi(\mathbf{x})+\frac{1}{2}(\mathbf{x}-\mathbf{v})^T\mathbf{H}(\mathbf{x}-\mathbf{v})\right).$$

Usually, **gprox** is not as easy as **prox** operator to calculate. It would be equivalent to **prox** operator when $\mathbf{H}$ is a diagonal matrix, and there's an straightforward method called spectral projected gradient (SPG) [20] adopted this strategy. There also have some more complex and efficient ways like the "diagonal+rank-1 case" studied by Becker and Fadili [21]. Here we focus on the most general case when $\mathbf{H}$ is approximated via BFGS, which proved to converge superlinearly [17].

2.4.2 *Proximal Newton-type Method*

Though it is not straightforward to solve the subproblem (11), we could apply the iterative method. Obviously, (11) is a quadratic problem which

could be solved via CVX, FISTA, ADMM and SPG. We have found that FISTA performs best empirically. Given Algorithm 1, we only need to obtain $\nabla g(\mathbf{x})$, and $\nabla g(\mathbf{x}) = \nabla g(\mathbf{x}_k) + \mathbf{H}_k(\mathbf{x} - \mathbf{x}_k)$ in this subproblem, so the LBFGS is appreciate and we could solve this subproblem with $\mathcal{O}(\frac{1}{k^2})$ complexity. The whole proximal quasi-Newton method is illustrated in Algorithm 3.

Algorithm 3 The proximal quasi-Newton method

Require: $\mathbf{x}_0$ and $\mathbf{H}_0$
Ensure: $\mathbf{x}_0 \in \mathbf{dom}\ f$, and $\mathbf{H}_0$ is a scaled identity matrix (positive definite).
1: $S \leftarrow \{\}$, $Y \leftarrow \{\}$
2: **repeat**
3: Update $\mathbf{H}_k$ using LBFGS, where $\mathbf{H}_k$ is symmetric positive definite.
4: Solve the problem in (11) for a descent direction:

$$\Delta\mathbf{x}_k \leftarrow \underset{\Delta}{\operatorname{argmin}}\ \hat{\mathcal{F}}(\mathbf{x}_k + \Delta) \quad \text{(Alogrithm 1)}$$

5: Search t_k with backtracking method.
6: Update $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + t_k\Delta\mathbf{x}_k$
7: **if** $(\mathbf{x}_{k+1} - \mathbf{x}_k)^T(\nabla g(\mathbf{x}_{k+1}) - \nabla g(\mathbf{x}_k)) > 0$ **then**
8: Update $S \leftarrow \{S, \mathbf{x}_{k+1} - \mathbf{x}_k\}$, $Y \leftarrow \{Y, \nabla g(\mathbf{x}_{k+1}) - \nabla g(\mathbf{x}_k)\}$
9: Adjust $\mathbf{H}_0$
10: **end if**
11: **until** stopping condition is satisfied

2.5 Coordinate Descent

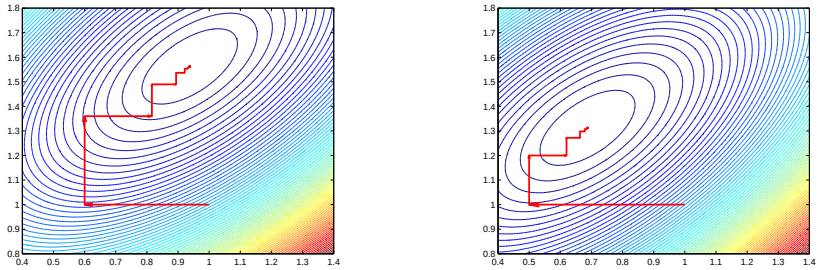
2.5.1 *Basic Concept*

Coordinate descent is a non-derivative optimization algorithm. To find a local minimum of a function, one does line search along one coordinate direction at the current point in each iteration. One uses different coordinate directions cyclically throughout the procedure. It follows that, if $\mathbf{x}_k$ is given, the ith coordinate of $\mathbf{x}_{k+1}$ is given by

$$\mathbf{x}_{k+1}(i) = \underset{v\in\mathbb{R}}{\operatorname{argmin}}\ f(\mathbf{x}_{k+1}(1), ..., \mathbf{x}_{k+1}(i-1), v, \mathbf{x}_k(i+1), ..., \mathbf{x}_k(p)),$$

where $\mathbf{x}_k(i)$ is ith coordinate of $\mathbf{x}_k$. Obviously, it's very efficient. Note that coordinate descent iteration may get stuck at a non-stationary point if the level curves of a function are not smooth.

This method could be generalized to handle composite function with a separable non-smooth part. Tseng [22] analyzed the convergence of the block coordinate descent method for non-differentiable minimization. Tseng and Yun [23] further improved the coordinate descent method by quadratic approximation. Friedman et al. [24] extended the this method to non-separable case to solve the fused LASSO. It's easy to implement the coordinate descent methods, we apply this method to a smooth function and a composite function with separable non-smooth term. The experiments results are show in Figure 1



(a) smooth objective

(b) non-smooth objective

Figure 1: Convergence comparison

NUMERICAL EXPERIMENTS

2.6 SGD

2.6.1 *Basic Concept*

In stochastic gradient descent, the true gradient is approximated by a gradient at a single example. The convergence of stochastic gradient descent has been analyzed using the theories of convex minimization and of stochastic approximation. Briefly, when the learning rates decrease with an appropriate rate, and subject to relatively mild assumptions, stochastic gradient descent converges almost surely to a global minimum when the objective function is convex. SGD could be generalized to SPGD [25] to deal with regularized objectives. Recently, SGD with importance sampling [26] has been developed.

NUMERICAL EXPERIMENTS

3 EXPERIMENTS AND DISCUSSION

3.1 Figure Composed of Subfigures

REFERENCES

[1] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[2] Ryan Joseph Tibshirani. *The solution path of the generalized lasso*. Stanford University, 2011.

[3] Lukas Meier, Sara Van De Geer, and Peter Bühlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):53–71, 2008.

[4] Stephen J Wright, Robert D Nowak, and Mário AT Figueiredo. Sparse reconstruction by separable approximation. *Signal Processing, IEEE Transactions on*, 57(7):2479–2493, 2009.

[5] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[6] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx, March 2014.

[7] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[8] Kim-Chuan Toh, Michael J Todd, and Reha H Tütüncü. Sdpt3-a matlab software package for semidefinite programming, version 1.3. *Optimization methods and software*, 11(1-4):545–581, 1999.

[9] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in optimization*, 1(3):123–231, 2013.

[10] Yu Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005.

[11] Heinz H Bauschke and Patrick L Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. Springer Science & Business Media, 2011.

[12] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[13] SR Becker, EJ Candès, and MC Grant. TFOCS: Templates for first-order conic solvers, 2012.

[14] Yu Nesterov. Gradient methods for minimizing composite objective function, 2007.

[15] Robert Nishihara, Laurent Lessard, Benjamin Recht, Andrew Packard, and Michael I. Jordan. A general analysis of the convergence of ADMM. In *International Conference on Machine Learning 32*, 2015.

[16] Jorge Nocedal and Stephen Wright. Numerical optimization, series in operations research and financial engineering. *Springer, New York, USA*, 2006.

[17] Jason D Lee, Yuekai Sun, and Michael A Saunders. Proximal Newton-type methods for minimizing composite functions. *SIAM Journal on Optimization*, 24(3):1420–1443, 2014.

[18] Mark W Schmidt, Ewout Berg, Michael P Friedlander, and Kevin P Murphy. Optimizing costly functions with simple constraints: A limited-memory projected quasi-Newton algorithm. In *International Conference on Artificial Intelligence and Statistics*, 2009.

[19] Mark Schmidt, Dongmin Kim, and Suvrit Sra. Projected Newton-type methods in machine learning. 2011.

[20] Ernesto G Birgin, José Mario Martínez, and Marcos Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization*, 10(4):1196–1211, 2000.

[21] Stephen Becker and Jalal Fadili. A quasi-Newton proximal splitting method. In *Advances in Neural Information Processing Systems*, pages 2618–2626, 2012.

[22] Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475–494, 2001.

[23] Paul Tseng and Sangwoon Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117 (1-2):387–423, 2009.

[24] Jerome Friedman, Trevor Hastie, Holger Höfling, Robert Tibshirani, et al. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332, 2007.

[25] Atsushi Nitanda. Stochastic proximal gradient descent with acceleration techniques. In *Advances in Neural Information Processing Systems*, pages 1574–1582, 2014.

[26] Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling. *arXiv preprint arXiv:1401.2753*, 2014.