First Order Algorithms for Signal and Image Processing

Tarmizi Adam

February 12, 2021

Abstract

In this note we introduce first order optimization algorithms for optimization problems with a focus in signal and image processing. The notes will go trough gradient descent, proximal gradient descent algorithms and accelerated proximal gradient descent algorithms. We will also discuss other first order algorithms from time to time.

Contents

1 Introduction

1. *Inverse problems* in signal processing include:
   (a) Denoising.
   (b) Deconvolution (also known as channel equalization or dereverberation) .
   (c) Source separation.
   (d) Channel equilization.
   (e) System identification.
2. Inverse problems are *ill-posed* problems. Its problems and difficulty include
   (a) Ill conditioning of the mapping. Because to solve the inverse problem the inverse of the mapping is needed, this is usually impractical due to ill conditioning of the mapping.
   (b) The observation (observed signal) may itself be corrupted by noise.
   (c) No unique solution. Thus the need of regularization that can exploit prior information, find a unique solution and stabilize the ill-conditioning of the mapping inversion.

2 Gradient descent

Before discussing the proximal gradient algorithm that is widely used in signal and image processing, it would be a good idea to first refresh and discuss the basic ideas of gradient descent. In this subsection we will introduce the gradient descent algorithm, its convergence and some example applications where we can use gradient descent.

2.1 Basic Gradient descent

The gradient descent method is probably one of the earliest method in optimization. It traces back to the French mathematician Louis Augustin Cauchy [1]. It was developed in 1847 by Cauchy to do calculations related to heavenly bodies in astronomy [2]. Basically, the pure gradient decent is used to solve the following unconstrained minimization problem,

$$\underset{\mathbf{x}}{\text{minimize}}\, f(\mathbf{x}), \tag{1}$$

Table 1: Gradient descent algorithm

1. Choose initial $x^{(0)} \in \mathbb{R}^n$
2. **repeat**
3. $\quad \mathbf{x}^{k+1} = \mathbf{x}^k - t\nabla f\left(\mathbf{x}^k\right); \quad k = 1, 2, \cdots$
4. **stop** at some point

where $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function[1]. The gradient descent algorithm proceeds by iteratively solving

$$\mathbf{x}^{k+1} = \mathbf{x}^k - t\nabla f\left(\mathbf{x}^k\right), \quad k = 1, 2, \cdots \tag{2}$$

until convergence is achieved. In the gradient descent iteration, t is the stepsize and ∇f is gradient of f. Table 1 summarizes the gradient descent algorithm. We now state the convergence of gradient discent

Theorem 1. *Gradient descent with fixed step size $t \leq \frac{1}{L}$ satisfy*

$$f\left(\mathbf{x}^{k+1}\right) - f^\star \leq \frac{||\mathbf{x}^0 - \mathbf{x}^\star||}{2tk}, \tag{3}$$

that is, gradient descent with fixed step size has the convergence rate of $O\left(\frac{1}{k}\right)$ i.e. to get $f\left(\mathbf{x}^{k+1}\right) - f^\star \leq \epsilon_0$ we need $O\left(\frac{1}{\epsilon_0}\right)$ iterations.

In order to proof the convergence rate, we will give first give several lemmas and definitions.

Lemma 1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a C^1 function. Suppose f is convex. Then*

$$f\left(\bar{\mathbf{x}}\right) + \left\langle \nabla f\left(\bar{\mathbf{x}}\right), \mathbf{x} - \bar{\mathbf{x}} \right\rangle \leq f\left(\mathbf{x}\right), \ \forall \mathbf{x}, \bar{\mathbf{x}} \in \mathbb{R}^n. \tag{4}$$

Definition 1. *A function $f : \mathbb{R}^n \to \mathbb{R}$ is Lipschitz continuous with constant $L > 0$ if*

$$||f\left(\mathbf{x}\right) - f\left(\mathbf{y}\right)|| \leq L\,||\mathbf{x} - \mathbf{y}|| \ \ \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n. \tag{5}$$

Furthermore, if f is C^1, we have that ∇f is Lipschitz continuous with constant $L > 0$ if

$$||\nabla f\left(\mathbf{x}\right) - \nabla f\left(\mathbf{y}\right)|| \leq L\,||\mathbf{x} - \mathbf{y}|| \ \ \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n. \tag{6}$$

[1]Sometimes also referred as smooth convex function of type $C^{1,1}$. Recall that C^k function is that the function is continuous and the k^{th}order derivative exists. The class of C^1 function means that it is continuously differentiable while, if the function is $C^{1,1}$, it means that the function is continuously differentiable and has Lipschitz continuous gradient ∇f with $L > 0$ Lipschitz constant which satisfies $||\nabla f\left(\mathbf{x}\right) - f\left(\mathbf{z}\right)||_2 \leq L\,||\mathbf{x} - \mathbf{z}||_2$.

Lemma 2. *Suppose that f is a C^1 function whose gradient is Lipschitz continuous with constant $L > 0$, then*

$$f(\mathbf{x}) \leq f(\mathbf{x}) + \langle \nabla f(\bar{\mathbf{x}}), \mathbf{x} - \bar{\mathbf{x}} \rangle + \frac{L}{2} ||\mathbf{x} - \bar{\mathbf{x}}||^2 \ \forall \mathbf{x}, \bar{\mathbf{x}} \in \mathbb{R}^n. \tag{7}$$

Lemma 2 is called the descent lemma and the detailed proof can be obtained in texts such as [3]. With the given Lemmas and definition above, we are now ready to proof the convergence rate of the gradient descent.

Define a function $g : \mathbb{R}^n \to \mathbb{R}$ by

$$g(\mathbf{x}) = \mathbf{x} - t\nabla f, \ \mathbf{x} \in \mathbb{R}^n, \tag{8}$$

and choose $\mathbf{x}_k \in \mathbb{R}^n$

$$\mathbf{x}_{k+1} = g(\mathbf{x}_k), \ k = 0, 1, 2, \cdots \tag{9}$$

Theorem 2. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a C^1 convex function. Suppose that ∇f is Lipschitz continuous with constant $L > 0$ and $0 < t \leq \frac{1}{L}$ then,*

1. $f(g(\mathbf{x})) \leq f(\mathbf{x}) - \frac{t}{2} ||\nabla f(\mathbf{x})||^2$,
2. $f(g(\mathbf{x})) \leq f(\mathbf{x}^*) + \frac{1}{2t} \left(||\mathbf{x} - \mathbf{x}^*||^2 - ||g(\mathbf{x}) - \mathbf{x}^*||^2 \right)$,

for all $\mathbf{x}, \mathbf{x}^* \in \mathbb{R}^n$.

Proof: To proof the first point, by Lemma (2) we have

$$f(\mathbf{z}) \leq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{z} - \mathbf{x} \rangle + \frac{L}{2} ||\mathbf{z} - \mathbf{x}||^2$$

$$\begin{aligned}
f(g(\mathbf{x})) &= f\left(\underbrace{\mathbf{x} - t\nabla f(\mathbf{x})}_{\mathbf{z}} \right), \\
&\leq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), -t\nabla f(\mathbf{x}) \rangle + \frac{L}{2} ||t\nabla f(\mathbf{x})||^2, \\
&\leq f(\mathbf{x}) - t||\nabla f(\mathbf{x})||^2 + \frac{Lt^2}{2} ||\nabla f(\mathbf{x})||^2, \\
&\leq f(\mathbf{x}) + \left(\frac{Lt^2}{2} - t \right) ||\nabla f(\mathbf{x})||^2, \\
&\leq f(\mathbf{x}) - \frac{t}{2} ||\nabla f(\mathbf{x})||^2. \qquad \square
\end{aligned} \tag{10}$$

The final inequality in (10) is obtained by the stepsize being $0 < t \leq \frac{1}{L}$, multiplying both sides of this inequality by $\frac{t}{2}$ and then subtracting the resulting inequality by t. Point 1,

in Theorem (2) tels us that in each iteration of the gradient descent, the function value decreases at least by a constant times the norm of the function gradient that is

$$f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k) \leq -\frac{t}{2} ||\nabla f(\mathbf{x}_k)||^2, \tag{11}$$

or equivalently,

$$f(\mathbf{x}_k) - f(\mathbf{x}_{k+1}) \geq \frac{t}{2} ||\nabla f(\mathbf{x}_k)||^2. \tag{12}$$

Next, to prove the second point, recall that

$$f(\mathbf{x}) \leq f(\mathbf{x}^*) + \langle \nabla f(\mathbf{x}), \mathbf{x} - \mathbf{x}^* \rangle$$

substitute

$$\begin{aligned} f(g(\mathbf{x})) &\leq f(\mathbf{x}^*) + \langle \nabla f(\mathbf{x}), \mathbf{x} - \mathbf{x}^* \rangle - \frac{t}{2} ||\nabla f(\mathbf{x})||^2 \\ &\leq f(\mathbf{x}^*) - \frac{t}{2} \left(||\nabla f(\mathbf{x})||^2 - \frac{2}{t} \langle \nabla f(\mathbf{x}), \mathbf{x} - \mathbf{x}^* \rangle \right) \end{aligned} \tag{13}$$

By completing the square of the last inequality, with[2] $b = -\frac{2}{\alpha}(\mathbf{x} - \mathbf{x}^*)$, $\frac{b}{2} = -\frac{\mathbf{x}-\mathbf{x}^*}{\alpha}$ and

[2]from $ax^2 + bx + c = 0$.

$\left(\frac{b}{2}\right)^2 = \frac{||\mathbf{x}-\mathbf{x}^*||^2}{\alpha^2}$,

$$
\begin{aligned}
&= f(\mathbf{x}^*) - \frac{\alpha}{2}\left(||\nabla f(\mathbf{x})||^2 - \frac{2}{\alpha}\langle \nabla f(\mathbf{x}),\, \mathbf{x}-\mathbf{x}^*\rangle + \frac{||\mathbf{x}-\mathbf{x}^*||^2}{\alpha^2} - \frac{||\mathbf{x}-\mathbf{x}^*||^2}{\alpha^2}\right),\\
&= f(\mathbf{x}^*) - \frac{\alpha}{2}\left[\left|\left|\nabla f(\mathbf{x}) - \frac{1}{\alpha}(\mathbf{x}-\mathbf{x}^*)\right|\right|^2 - \frac{1}{\alpha^2}||\mathbf{x}-\mathbf{x}^*||^2\right],\\
&= f(\mathbf{x}^*) - \frac{\alpha}{2}\left[\left|\left|\frac{\alpha\nabla f(\mathbf{x}) - (\mathbf{x}-\mathbf{x}^*)}{\alpha}\right|\right|^2 - \frac{1}{\alpha^2}||\mathbf{x}-\mathbf{x}^*||^2\right],\\
&= f(\mathbf{x}^*) - \frac{\alpha}{2}\left[\frac{||\alpha\nabla f(\mathbf{x}) - \mathbf{x} + \mathbf{x}^*||^2}{\alpha^2} - \frac{||\mathbf{x}-\mathbf{x}^*||^2}{\alpha^2}\right], \qquad (14)
\end{aligned}
$$

$$
\begin{aligned}
&= f(\mathbf{x}^*) - \frac{\alpha}{2}\left(\frac{1}{\alpha^2}\right)\left[||\alpha\nabla f(\mathbf{x}) - \mathbf{x} + \mathbf{x}^*||^2 - ||\mathbf{x}-\mathbf{x}^*||^2\right],\\
&= f(\mathbf{x}^*) - \frac{1}{2\alpha}\left[||\mathbf{x}^* - (\mathbf{x} - \alpha\nabla f(\mathbf{x}))||^2 - ||\mathbf{x}-\mathbf{x}^*||^2\right],\\
&= f(\mathbf{x}^*) - \frac{1}{2\alpha}\left[||\mathbf{x}^* - g(\mathbf{x})||^2 - ||\mathbf{x}-\mathbf{x}^*||^2\right],\\
&= f(\mathbf{x}^*) - \frac{1}{2\alpha}\left[-||\mathbf{x}-\mathbf{x}^*||^2 + ||\mathbf{x}^* - g(\mathbf{x})||^2\right],\\
&= f(\mathbf{x}^*) + \frac{1}{2\alpha}||\mathbf{x}-\mathbf{x}^*||^2 - \frac{1}{2\alpha}||\mathbf{x}^* - g(\mathbf{x})||^2,\\
&= f(\mathbf{x}^*) + \frac{1}{2\alpha}\left[||\mathbf{x}-\mathbf{x}^*||^2 - ||\mathbf{x}^* - g(\mathbf{x})||^2\right],\\
&= f(\mathbf{x}^*) + \frac{1}{2\alpha}\left[||\mathbf{x}-\mathbf{x}^*||^2 - ||g(\mathbf{x}) - \mathbf{x}^*||^2\right]. \qquad (15)
\end{aligned}
$$

Thus,

$$f(g(\mathbf{x})) \leq f(\mathbf{x}^*) + \frac{1}{2\alpha}\left[||\mathbf{x}-\mathbf{x}^*||^2 - ||g(\mathbf{x}) - \mathbf{x}^*||^2\right]. \quad \square$$

completing the proof of the second point of Theorem 2.

We are now ready to prove Theorem 1.

$$f(\mathbf{x}_{k+1}) = f(g(\mathbf{x}_k)) \leq f(\mathbf{x}^*) + \frac{1}{2\alpha}\left[||\mathbf{x}-\mathbf{x}^*||^2 - ||\mathbf{x}_{k+1}-\mathbf{x}^*||^2\right], \qquad (16)$$

for all $k = 0,\, 1,\, 2,\, 3,\, \cdots,\, k-1$. If we apply for all values of k,

$$
\begin{aligned}
f\left(\mathbf{x}_1\right) &\leq f\left(\mathbf{x}^*\right)+\frac{1}{2\alpha}\left[\left|\left|\mathbf{x}_0-\mathbf{x}^*\right|\right|^2-\left|\left|\mathbf{x}_1-\mathbf{x}^*\right|\right|^2,\right] \\
f\left(\mathbf{x}_2\right) &\leq f\left(\mathbf{x}^*\right)+\frac{1}{2\alpha}\left[\left|\left|\mathbf{x}_1-\mathbf{x}^*\right|\right|^2-\left|\left|\mathbf{x}_2-\mathbf{x}^*\right|\right|^2\right], \\
&\vdots \qquad\qquad \vdots \\
f\left(\mathbf{x}_k\right) &\leq f\left(\mathbf{x}^*\right)+\frac{1}{2\alpha}\left[\left|\left|\mathbf{x}_{k-1}-\mathbf{x}^*\right|\right|^2-\left|\left|\mathbf{x}_k-\mathbf{x}^*\right|\right|^2\right]. \qquad (17)
\end{aligned}
$$

Summing up for all k we get the following

$$
\sum_{i=1}^{k} f\left(\mathbf{x}_i\right) \leq k f\left(\mathbf{x}^*\right)+\frac{1}{2\alpha}\left[\left|\left|\mathbf{x}_0-\mathbf{x}^*\right|\right|^2-\left|\left|\mathbf{x}_k-\mathbf{x}^*\right|\right|^2\right], \qquad (18)
$$

which can be simplified into

$$
\begin{aligned}
k f\left(\mathbf{x}_k\right) &\leq k f\left(\mathbf{x}^*\right)+\frac{\left|\left|\mathbf{x}_0-\mathbf{x}^*\right|\right|^2}{2\alpha} \\
k f\left(\mathbf{x}_k\right)-k f\left(\mathbf{x}^*\right) &\leq \frac{\left|\left|\mathbf{x}_0-\mathbf{x}^*\right|\right|^2}{2\alpha} \\
f\left(\mathbf{x}_k\right)-f\left(\mathbf{x}^*\right) &\leq \frac{\left|\left|\mathbf{x}_0-\mathbf{x}^*\right|\right|^2}{2\alpha k}, \qquad \square \qquad (19)
\end{aligned}
$$

which completes the proof of the $O\left(\frac{1}{k}\right)$ convergence rate of gradient descent.

2.1.1 Stepsize rule strategies

The rate of convergence for the gradient method discussed above was obtained when we have prior knowledge of the Lipschitz constant L. Unfortunately in many applications, the Lipschitz constant L is unknown no practical method exist to estimate it [4].

Instead of using *a constant stepsize* $t=\frac{1}{L}$, a more practical stepsize is the *backtracking line search* method to find the stepsize t. This method is used when no Lipschitz constant L is known or provided. In the bakctracking line search, if the following condition is true [5]

$$
f\left(\mathbf{x}+t\Delta\mathbf{x}\right)-f\left(\mathbf{x}\right) > \alpha t \nabla f\left(\mathbf{x}\right)^{\top}\Delta\mathbf{x}, \qquad (20)
$$

then update the stepsize t as

$$
t=\beta t. \qquad (21)
$$

Where in the above, $0<\alpha<0.5$, $0<\beta<1$ and $\Delta\mathbf{x}$ is the descent direction which in the gradient descent method $\Delta\mathbf{x}=\nabla f\left(\mathbf{x}\right)$. The condition (20) is called the *Armijo* condition [6].

Table 2: Gradient descent algorithm for solving (22)

1. Choose initial $x^{(0)} \in \mathbb{R}^n$, Lipschitz constant, L
2. **repeat**
3. $\quad \mathbf{x}^{k+1} = \mathbf{x}^k - \frac{1}{L}\mathbf{A}^\top \left(\mathbf{A}\mathbf{x}^k - \mathbf{y}\right); \quad k = 1, 2, \cdots$
4. **stop** at some point

2.1.2 Applications

In order to understand more the gradient descent algorithm, let us use it to solve a simple least square problem. Here, we would like to solve the following optimization problem.

$$\underset{\mathbf{x}}{\text{minimize}}\, f(\mathbf{x}) = \frac{1}{2}\left|\left|\mathbf{A}\mathbf{x} - \mathbf{y}\right|\right|^2. \tag{22}$$

By assuming $\mathbf{A} \in \mathbb{R}^{m\times m}$ is symmetric positive definite, this problem is quadratic and we can re-write it as

$$\underset{\mathbf{x}}{\text{minimize}}\, f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top\mathbf{A}^\top\mathbf{A}\mathbf{x} - \mathbf{b}^\top\mathbf{A}\mathbf{x} + \frac{1}{2}\left|\left|\mathbf{b}\right|\right|. \tag{23}$$

It is also worth mentioning that this least squares problem is also equivalent to solving a system of linear equations

$$\mathbf{A}\mathbf{x} = \mathbf{y}, \tag{24}$$

where we minimize the error $\mathbf{r} = \mathbf{A}\mathbf{x} - \mathbf{y}$ in the ℓ_2-norm sense i.e., $||\mathbf{A}\mathbf{x} - \mathbf{y}||^2$. Now, all we need is the gradient of f,

$$\nabla f(\mathbf{x}) = \mathbf{A}^\top\left(\mathbf{A}\mathbf{x} - \mathbf{y}\right), \tag{25}$$

and also the a suitable step size t. If we know the Lipschitz constant L, we can use the constant step size rule by taking $t = \frac{1}{L}$. The Lipschitz L is taken to be the largest eigenvalue of $\mathbf{A}^\top\mathbf{A}$. However, if the matrix $\mathbf{A}$ is to large, this might be impractical and hence, other rules are used to find the step size[3]. The other rules will be discussed as we progress trough these notes. Now with $t = \frac{1}{L}$ and (25) the iteration for solving (22) is listed in Table 2.

In order to visualize the steps taken by the gradient descent algorithm, we will minimize a simpler quadratic problem which is also an instance of problem (23),

$$\underset{\mathbf{x}}{\text{minimize}}\, f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top\mathbf{A}\mathbf{x} - \mathbf{b}^\top\mathbf{x}, \tag{26}$$

with gradient

$$\nabla f(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{y}, \tag{27}$$

for $\mathbf{x}, \mathbf{b} \in \mathbb{R}^{2\times 1}$ and $\mathbf{A} \in \mathbb{R}^{2\times 2}$. For this small example, we have

$$\mathbf{A} = \begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix},$$

[3]In general, if we can know the Lipschitz constant L beforehand, it is always a good idea to just use the constant step size with $t = \frac{1}{L}$. Unfortunately, in real applications this might not be the case.

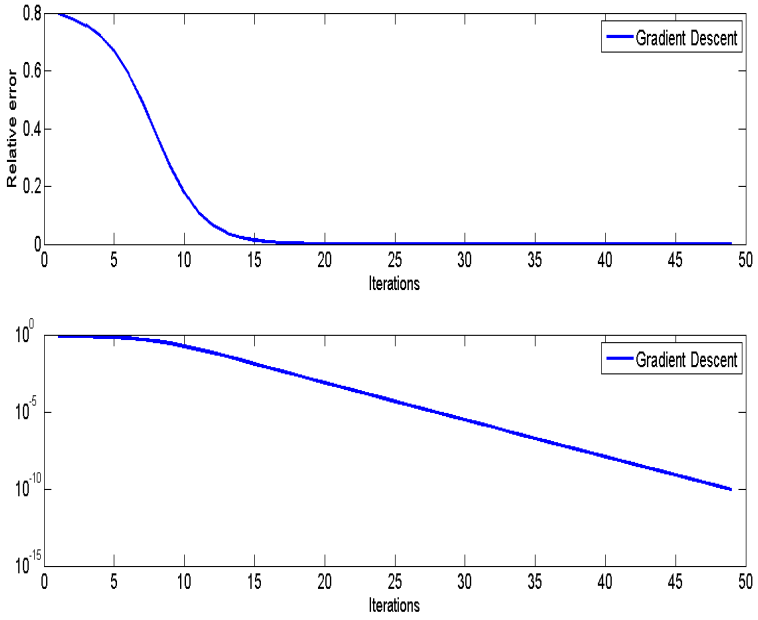


Figure 1: Convergence of the iterates of gradient descent on a least squares problem. The bottom plot has the y-axis on a logarithmic scale. The bottom plot also shows that the gradient descent converges at a linear rate.

and

$$\mathbf{b} = \begin{bmatrix} 2 \\ -8 \end{bmatrix}.$$

The initial value $\mathbf{x}_0$ is chosen to be

$$\mathbf{x}_0 = \begin{bmatrix} -6 \\ -6 \end{bmatrix}.$$

To further show the steps of gradient descent, we minimize a more difficult objective function. The function is called the Rosenbrock function. This function is very ill-conditioned and hence difficult to minimize. The Rosenbrock function is defines as

2.1.3 The Heavy Ball Method

To achieve faster convergence than the gradient descent several other first order algorithms have been proposed. The heavy ball method by Polyak proposed in 1964 is a multi step first order method for this purpose. The algorithm is presented in Table 3. Note that the heavy ball method uses two previous step for its $\mathbf{x}$ update.

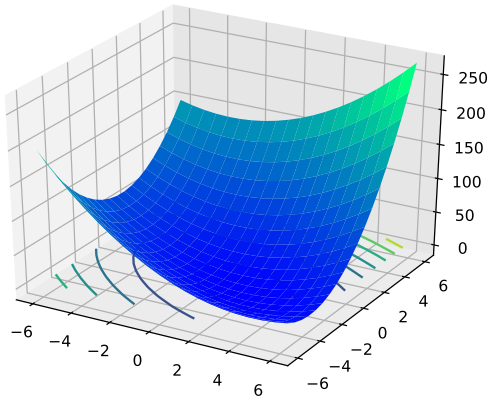


Figure 2: The surface plot of the simple quadratic function (26).

Table 3: Heavy Ball Method

1. Choose initial $\mathbf{x}_{k-1}$, $\mathbf{x}_k$, $\mathbf{x}_{k+1} \in \mathbb{R}^n$, $\alpha > 0$, $\beta > 0$
2. **repeat**
3. $\quad \mathbf{y}_{k+1} = \mathbf{x}_k + \beta\left(\mathbf{x}_k - \mathbf{x}_{k-1}\right)$
4. $\quad \mathbf{x}^{k+1} = \mathbf{y}_{k+1} - \alpha \nabla f\left(\mathbf{x}^k\right); \quad k = 1, 2, \cdots$
5. **stop** at some point

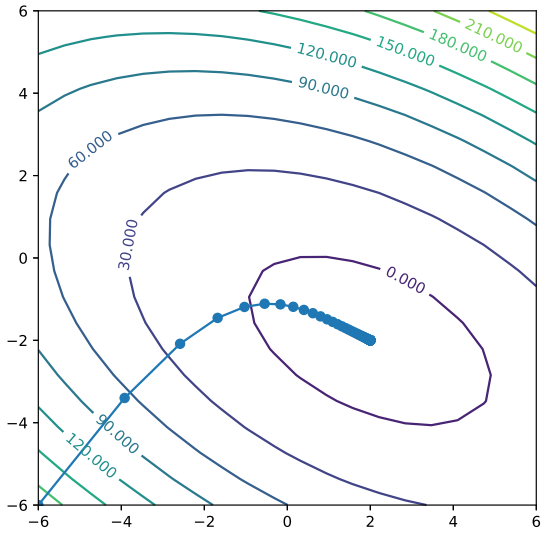
2.1.4 Nesterov Accelerated Gradient Descent

The gradient method discussed previously has a convergence rate of $O\left(\frac{1}{k}\right)$ iterations which is quite slow. In a seminal paper [7], Nesterov proposed an accelerated variant of the gradient method which can achieve an iteration complexity of $O\left(\frac{1}{k^2}\right)$ which is a huge improvement. This method is usually referred as Nesterov accelerated gradient descent. This convergence rate is "optimal" in the sense of complexity analysis of Nemirovski and Yudin [8] i.e., first order methods cannot converge any faster than this rate [9]. The algorithm is presented in Table 4.

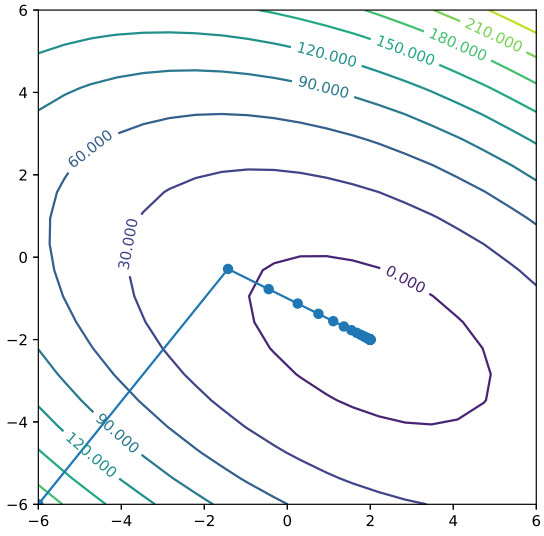
2.1.5 Performance comparison

With the introduction of the above first order algorithms, let us now compare the performance of each algorithm for minimizing the quadratic minimization problem (22),

$$\underset{\mathbf{x}}{\text{minimize}}\, f\left(\mathbf{x}\right) = \frac{1}{2}\left|\left|\mathbf{A}\mathbf{x} - \mathbf{y}\right|\right|^2,$$



(a) Gradient descent path. Step size, $t = 0.065$.



(b) Gradient descent path. Step size, $t = \frac{1}{L}$.

Table 4: Nesterov's Accelerated Gradient descent algorithm

1. Choose initial $t_0 = 1$, $\mathbf{x}_0 = \mathbf{y}_0 \in \mathbb{R}^n$, $\alpha = \frac{1}{L}$
2. **repeat**
3. $\quad \mathbf{x}_{k+1} = \mathbf{y}_k - \alpha \nabla f(\mathbf{x}_k)\,; \quad k = 1, 2, \cdots$
4. $\quad t_{k+1} = \frac{\left(1+\sqrt{4 \cdot t_k^2 + 1}\right)}{2}$
5. $\quad \mathbf{y}_{k+1} = \mathbf{x}_{k+1} + \frac{t_k - 1}{t_{k+1}} (\mathbf{x}_{k+1} - \mathbf{x}_k)$
6. **stop** at some point

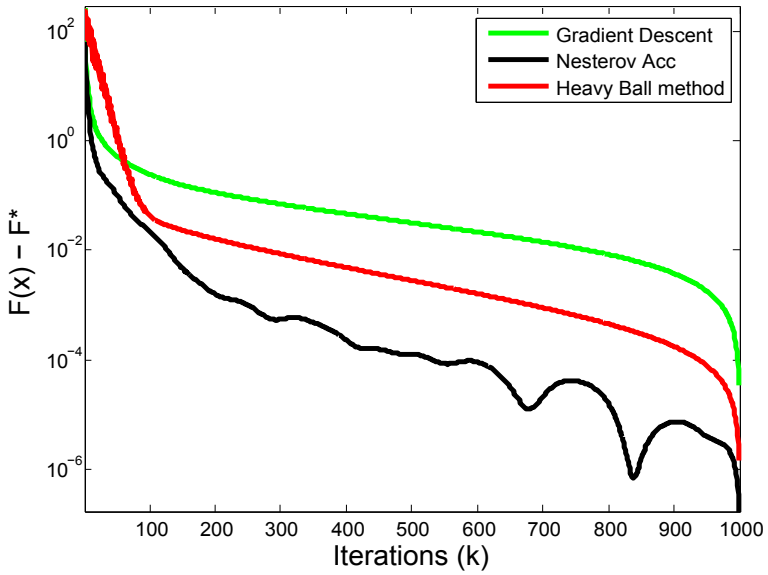


Figure 4

with $\mathbf{A} \in \mathbb{R}^{n \times n}$ as

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & -1 & 2 & -1 \\ 0 & 0 & \cdots & -1 & 2 \end{bmatrix},$$

and $\mathbf{y} \in \mathbb{R}^n$ a random vector. We wish to recover or obtain $\mathbf{x}$ by minimizing the above problem. The performance of gradient descent, Nesterov's accelerated descent and Heavy ball method is shown in Figure 4.

2.2 Anderson Acceleration

Anderson acceleration (AA) was first introduce in the 1960's to speed up solutions related to nonlinear integral equations [10]. This extrapolation technique has then been modified for accelerating fixed point iteration schemes [11]. An advantage of the Anderson acceleration

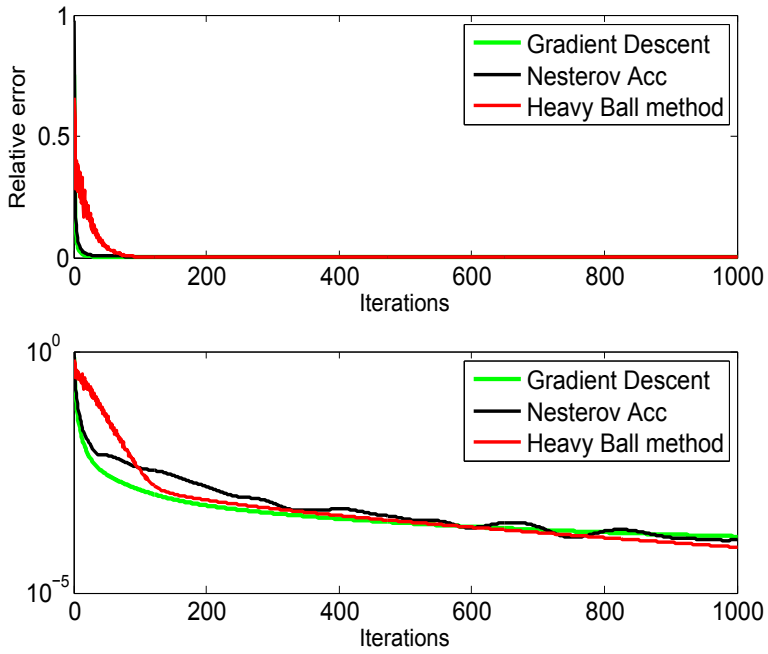


Figure 5

technique is the estimation of the solution directly from the available previous sequence without needing any of problem parameters. Before delving further into the idea of (AA) we give some preliminaries of fixed point iterations.

2.2.1 Fixed point iterations

The fixed point iteration (FPI) or also called the Picard iteration [12] is an algorithm to find a fixed point of a function $g(\cdot)$. Recall that $\mathbf{x}$ is a fixed point of the function $g(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^n$ if

$$\mathbf{x} = g(\mathbf{x}) \tag{28}$$

Let $\mathbf{r} \in \mathbb{R}^n$ be the root of the function $g(\cdot)$, this means

$$g(\mathbf{r}) = 0, \tag{29}$$

by using (28) this can be written as

$$g(\mathbf{r}) = \mathbf{r} - g(\mathbf{r}). \tag{30}$$

Because $\mathbf{r}$ is the root and $g(\mathbf{r}) = 0$,

$$\begin{aligned} g(\mathbf{r}) &= \mathbf{r} - 0 \\ &= \mathbf{r}. \end{aligned} \tag{31}$$

Hence, we can write $g(\mathbf{x}) = \mathbf{x}$. In terms of iterations the FPI goes as follows

$$\mathbf{x}^{k+1} = g\left(\mathbf{x}^{k}\right), \tag{32}$$

Algorithm 1. Anderson Acceleration

Input: $\mathbf{x}_0$, $m \geq 1$
1. $\mathbf{x}_1 = g(\mathbf{x}_0)$
2. **for** $k = 1, 2, \cdots, K$
3. $\quad m_k = \min(m, k)$
4. $\quad \mathbf{r}_i = g(\mathbf{x}_i) - \mathbf{x}_i$
5. $\quad \mathbf{R}_k = [\mathbf{r}_{k-m_k}, \cdots, \mathbf{r}_k]$
6. $\quad \alpha^k = \underset{\alpha^\top \mathbf{1}=\mathbf{1}}{\operatorname{argmin}} ||\mathbf{R}_k \alpha||$
7. $\quad \mathbf{x}_{k+1} = \sum_{i=0}^{m_k} \alpha_i^k g(\mathbf{x}_{k-m_k+i})$
8. **end for**

Output: $\mathbf{x}_K$

---

with $\mathbf{x}^0 \in \mathbb{R}^n$ as some starting point. We will not delve into details of FPI, rather the interested reader should consult related references on numerical methods. The main point for this small section on FPI is to show the reader that in the FPI iteration (32), the current iterate $\mathbf{x}^{k+1}$ is only obtained by one previous iteration $\mathbf{x}^k$. Different from AA which we shall see later, in AA the current iterate is obtained by a linear combination of several past iterations.

2.2.2 Anderson Accelerated Gradient Descent

As stated earlier, AA is used to speed up convergence of FPI. In order to apply AA for gradient descent, we will manipulate the gradient descent step as a FPI. First, let us introduce the AA algorithm. The full AA algorithm is presented as Algorithm 1.

To further understand the steps of AA as listed in Algorithm (**??**), we will elaborate on steps 4, 5, 6 and 7. Step 4, is the residual computation of the k^{th} iteration. This residual is then used as the columns for the k^{th} residual matrix $\mathbf{R}_k$ in step 5. To elaborate further, lets say that we are on the 7^{th} iteration (i.e., $k = 7$) and our chosen $m = 5$. From step 3, this would yield $m_k = 5$ thus, $\mathbf{r}_{k-m_k} = \mathbf{r}_{7-5} = \mathbf{r}_2$ and

$$\begin{aligned} \mathbf{R}_7 &= [\mathbf{r}_2, \cdots, \mathbf{r}_7], \\ &= [\mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4, \mathbf{r}_5, \mathbf{r}_6, \mathbf{r}_7]. \end{aligned} \tag{33}$$

More clearly, $\mathbf{R}_k$ would be a matrix as follows,

$$\mathbf{R}_k = \begin{bmatrix} \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \mathbf{r}_2 & \mathbf{r}_3 & \cdots & \mathbf{r}_7 \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

Let us clarify with another example. Consider now iteration 10 i.e., $k = 10$. The residual matrix would be

$$\mathbf{R}_k = \begin{bmatrix} \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \mathbf{r}_5 & \mathbf{r}_6 & \cdots & \mathbf{r}_{10} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

3 Proximal gradient

1. Proximal gradient algorithm is to solve the following optimization problem

$$\min_{\mathbf{x}} F(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x}), \tag{34}$$

   where $f(\cdot)$ is smooth and differentiable (its gradient ∇f, is Lipschitz continuous) and $g(\cdot)$ is possibly non-smooth.

2. Problem of the form 34 plays an important role in various signal processing and machine learning task. Some examples

   (a) Basis pursuit denosing (BPD): $\min_{\mathbf{u}} \frac{1}{2} ||\mathbf{u} - \mathbf{f}||_2^2 + \lambda ||\mathbf{u}||_1$

3. Due to the non-smooth term $g(\cdot)$ methods such as gradient descent, nonlinear conjugate gradient and Newton method are not possible due to the non-differentiable nature of non-smooth functions.

4. Other methods like splitting methods i.e., ADMM or Douglas-Rachford (DR) can be used but leads to solving a linear system which needs a solver of its own. Therefore, we have inner iterations within these splitting algorithms.

5. To deal with the non-smooth function $g(\cdot)$ the *proximal* operator is used. It is defined as

$$\mathbf{z}^* = \text{prox}_{\mu g}(\mathbf{u}) = \arg\min_{\mathbf{z}} \frac{1}{2\mu} ||\mathbf{z} - \mathbf{u}||_2^2 + g(\mathbf{u}) \tag{35}$$

6. In the proximal operator, we add a quadratic proximal term to the function $g(\cdot)$. By minimizing the proximal operator, the solution $\mathbf{z}^*$ is regulated to be in the proximity of the solution to the minimization of $g(\cdot)$.

7. The scalar μ controls the closeness of the solution $\mathbf{z}$ to $\mathbf{u}$. Small values will result $\mathbf{z}$ being close to $\mathbf{u}$ while large values will have $\mathbf{z}$ close to the minimum of $g(\cdot)$.

8. Solution to the proximal map (35) are usually cheap to compute. They usually have analytic solution or closed form solution.

9. Some examples: If $g(\cdot) = ||\cdot||_2^2$, the proximal amount to solving a linear system of equation. Closed form with matrix inversion is possible in this situation but iterative methods are preferable if the problem is of large scale.

3.1 Iterative Shrinkage and Thresholding Algorithm

Consider the unconstrained minimization problem

$$\underset{\mathbf{x}}{\text{minimize}}\, f(\mathbf{x}), \tag{36}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function. The classical gradient descent solves the above unconstrained optimization problem in each iteration as:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - t\nabla f\left(\mathbf{x}^k\right), \quad k = 1, 2, \cdots \tag{37}$$

where t is a step size. The above gradient iteration of $\mathbf{x}^{k+1}$ can be viewed as a proximal regularization of the linearized function f at $\mathbf{x}^k$

$$\mathbf{x}^{k+1} = \underset{\mathbf{x}}{\text{argmin}}\, f\left(\mathbf{x}^k\right) + \langle \nabla f\left(\mathbf{x}^k\right), \mathbf{x} - \mathbf{x}^k \rangle + \frac{1}{2t}\left|\left|\mathbf{x} - \mathbf{x}^k\right|\right|_2^2 \tag{38}$$

Recall the following composite objective function (34)

$$\underset{\mathbf{x}}{\text{minimize}}\, F(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x}), \tag{39}$$

the iterative shrinkage and thresholding algorithm (ISTA) solves this by adopting the same linearization technique. At each iteration of ISTA, $\mathbf{x}$ is updated as[4]:

$$\begin{aligned}\mathbf{x}^{k+1} &= \underset{\mathbf{x}}{\text{argmin}}\, f\left(\mathbf{x}^k\right) + \langle \nabla f\left(\mathbf{x}^k\right), \mathbf{x} - \mathbf{x}^k \rangle + \frac{1}{2t}\left|\left|\mathbf{x} - \mathbf{x}^k\right|\right|_2^2 + g(\mathbf{x}) \\ &= \underset{\mathbf{x}}{\text{argmin}}\, \frac{1}{2t}\left|\left|\mathbf{x} - \left(\mathbf{x}^k - t\nabla f\left(\mathbf{x}^k\right)\right)\right|\right|_2^2 + g(\mathbf{x}).\end{aligned} \tag{40}$$

Note that (40) is actually a proximal operator defined as defined in (35). Here, the argument to the proximal operator is the gradient step[5] $\mathbf{x}^k - t\nabla f\left(\mathbf{x}^k\right)$. Depending on the function $g(\mathbf{x})$, the proximal step (40) may have simple closed form solutions. As an example, if $g(\mathbf{x}) = ||\mathbf{x}||_1$ i.e., the ℓ_1-norm, the proximal step (40) is the well known soft thresholding or shrinkage operator which can be solved by

$$\text{soft}(\mathbf{x}) = \text{sgn}(\mathbf{x}) \odot \max\left(|\mathbf{x}| - \lambda\right). \tag{41}$$

More examples will be discussed in the subsequent sections. Next, we will state the convergence of ISTA.

[4]Refer to appendix to see the derivations

[5]Hence, the name *proximal gradient* method.

The convergence rate of ISTA has the same convergence rate of $O\left(\frac{1}{k}\right)$ as the classical gradient method (gradient decent). The following theorem states the convergence rate of ISTA.

Theorem 3. *Let $\{\mathbf{x}_k\}$ be the sequence generated by ISTA. Then for any $k \geq 1$, we have that*

$$F\left(\mathbf{x}_k\right) - F\left(\mathbf{x}^*\right) \leq \frac{\alpha L \left|\left|\mathbf{x}_0 - \mathbf{x}^*\right|\right|^2}{2k}, \tag{42}$$

where $\mathbf{x}^*$ is a minimizer of F, $\alpha = 1$ for constant stepsize and $\alpha = \eta$ for backtracking stepsize.

This theorem states that in order to obtain an ϵ-optimal solution $\overline{\mathbf{x}}$ i.e., $F\left(\overline{\mathbf{x}}\right) - F\left(\mathbf{x}^*\right)$ the iteration required is at most $\frac{C}{\epsilon}$, where $C = \frac{\alpha L \left|\left|\mathbf{x}_0 - \mathbf{x}^*\right|\right|^2}{2}$. Proof of the convergence can be found in the excellent paper by Beck and Teboulle [13].

Example 1: Basis pursuit denoising (BPD)

We will solve the basis pursuit denoising for sparse signal recover using ISTA. The following problem is to be minimized

$$\underset{\mathbf{x}}{\text{minimize}}\, F\left(\mathbf{x}\right) = \frac{1}{2}\left|\left|\mathbf{A}\mathbf{x} - \mathbf{y}\right|\right|_2^2 + \lambda \left|\left|\mathbf{x}\right|\right|_1, \tag{43}$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}$ is the measurement matrix, $\mathbf{y} \in \mathbb{R}^{n \times 1}$ is the observed corrupted signal and $\mathbf{x} \in \mathbb{R}^{n \times 1}$ is the sparse signal to be estimated. Obviously, the BPD fits into the model (39) with $f\left(\mathbf{x}\right) = \frac{1}{2}\left|\left|\mathbf{A}\mathbf{x} - \mathbf{y}\right|\right|_2^2$ and $g\left(\mathbf{x}\right) = \lambda \left|\left|\mathbf{x}\right|\right|_1$.

The first step is to compute the gradient of the smooth and differentiable part $\nabla f\left(\mathbf{x}\right)$ and leaving alone the nonsmooth part $g\left(\mathbf{x}\right)$. The gradient of $f\left(\mathbf{x}\right)$ is

$$\nabla f\left(\mathbf{x}\right) = \mathbf{A}^\top\left(\mathbf{A}\mathbf{x} - \mathbf{y}\right).$$

By the iteration update of ISTA (40) we have

$$\begin{aligned}
\mathbf{x}^{k+1} &= \underset{\mathbf{x}}{\text{argmin}}\, \frac{1}{2t}\left|\left|\mathbf{x} - \left(\mathbf{x}^k - t\nabla f\left(\mathbf{x}^k\right)\right)\right|\right|_2^2 + \lambda g\left(\mathbf{x}\right) \\
&= \underset{\mathbf{x}}{\text{argmin}}\, \frac{1}{2t\lambda}\left|\left|\mathbf{x} - \left(\mathbf{x}^k - t\mathbf{A}^\top\left(\mathbf{A}\mathbf{x}^k - \mathbf{y}\right)\right)\right|\right|_2^2 + g\left(\mathbf{x}\right)
\end{aligned} \tag{44}$$

The value of the stepsize t, is taken to be $t = \frac{1}{L}$, where $L \geq \text{maxeig}\left(\mathbf{A}^\top\mathbf{A}\right)$ is the Lipschitz constant of the gradient ∇f. The regularization parameter λ, controls the degree of sparseness of the solution. With these parameter, we can re-write (44) as

$$\begin{aligned}
\mathbf{x}^{k+1} &= \underset{\mathbf{x}}{\text{argmin}}\, \frac{1}{2\left(\frac{\lambda}{L}\right)}\left|\left|\mathbf{x} - \left(\mathbf{x}^k - \frac{1}{L}\mathbf{A}^\top\left(\mathbf{A}\mathbf{x}^k - \mathbf{y}\right)\right)\right|\right|_2^2 + g\left(\mathbf{x}\right) \\
&= \underset{\mathbf{x}}{\text{argmin}}\, \frac{L}{2\lambda}\left|\left|\mathbf{x} - \left(\mathbf{x}^k - \frac{1}{L}\mathbf{A}^\top\left(\mathbf{A}\mathbf{x}^k - \mathbf{y}\right)\right)\right|\right|_2^2 + g\left(\mathbf{x}\right)
\end{aligned} \tag{45}$$

Algorithm 2. ISTA for basis pursuit denoising (constant stepsize)

Input: L (Lipschitz constant), λ (regularization parameter)
1. **while** not converged
2. $\quad \mathbf{v} = \mathbf{x}^k - \frac{1}{L}\mathbf{A}^\top (\mathbf{A}\mathbf{x} - \mathbf{y})$
3. $\quad \mathbf{x}^{k+1} = \text{prox}_{\frac{\lambda}{L}g}(\mathbf{v})$
4. $\quad k = k + 1$
5. **end while**

For perfect reconstruction or recovery of the sparse signal, we require that the sensing matrix $\mathbf{A}$ satisfy the restricted isometric property (RIP) as follows,

$$(1-\delta)\,||\mathbf{x}||_2 \leq ||\mathbf{A}\mathbf{x}|| \leq (1-\delta)\,||\mathbf{x}||_2\,. \tag{46}$$

Ensuring that the sensing matrix $\mathbf{A}$ satisfy RIP is NP-hard. Fortunately, this RIP property usually holds with very high probability when the entries of the sensing matrix is formed from some random distribution. For example, if the entries of $\mathbf{A}$ has independent and identically distributed (i.i.d) Gaussian entries then $\mathbf{A}$ satisfy RIP with high probability. We will not delve into the details of the theoretical results that govern the perfect reconstruction in this note. These theoretical results are discussed with high level of details in papers related to compressed sensing.

Example 2: Matrix Completion

We are interested in the recovery of a low rank matrix from the following linear model

$$\mathbf{Y}_\Omega = \mathcal{P}_\Omega(\mathbf{X}) + \mathbf{N}, \tag{47}$$

where $\mathbf{Y}_\Omega$, $\mathbf{X}, \mathbf{N} \in \mathbb{R}^{m\times n}$. Therefore we minimize the following objective

$$\underset{\mathbf{X}\in\mathbb{R}^{\mathbf{m}\times\mathbf{n}}}{\text{minimize}}\, F(\mathbf{X}) = \frac{1}{2}\,||\mathbf{Y}_\Omega - \mathcal{P}_\Omega(\mathbf{X})||_2^2 + \lambda\,||\mathbf{X}||_*\,. \tag{48}$$

Observe that in this case $f(\mathbf{X}) = \frac{1}{2}\,||\mathbf{Y}_\Omega - \mathcal{P}_\Omega(\mathbf{X})||_2^2$ and $g(\mathbf{X}) = \lambda\,||\mathbf{X}||_*$. The projection operator $\mathcal{P}(\cdot)$ applied to $\mathbf{X}$ sets $\mathbf{X}_{i,j}$, if $(i,\,j) \in \Omega$ and 0 otherwise. Therefore, we can re-write problem (48) as

$$\underset{\mathbf{X}\in\mathbb{R}^{\mathbf{m}\times\mathbf{n}}}{\text{minimize}}\, F(\mathbf{X}) = \frac{1}{2}\,||\mathbf{Y}_\Omega - \mathbf{X}_\Omega||_2^2 + \lambda\,||\mathbf{X}||_*\,. \tag{49}$$

Appendix

A Derivation of proximal gradient update

The proximal gradient update (40) is[6]

$$\begin{aligned}\mathbf{x}^{k+1} &= \underset{\mathbf{x}}{\operatorname{argmin}}\, f\left(\mathbf{x}^k\right) + \left\langle \nabla f\left(\mathbf{x}^k\right), \mathbf{x}-\mathbf{x}^k\right\rangle + \frac{1}{2t}\left|\left|\mathbf{x}-\mathbf{x}^k\right|\right|_2^2 + g(\mathbf{x}) \\ &= \underset{\mathbf{x}}{\operatorname{argmin}}\, \frac{1}{2t}\left|\left|\mathbf{x}-\left(\mathbf{x}^k - t\nabla f\left(\mathbf{x}^k\right)\right)\right|\right|_2^2 + g\left(\mathbf{x}\right).\end{aligned}$$

We will show how to get the latter equation. First, expand

$$\begin{aligned}&= f\left(\mathbf{x}^k\right) + \left\langle \nabla f\left(\mathbf{x}^k\right), \mathbf{x}-\mathbf{x}^k\right\rangle + \frac{1}{2t}\left|\left|\mathbf{x}-\mathbf{x}^k\right|\right|_2^2 \\ &= f\left(\mathbf{x}^k\right) + \mathbf{x}\nabla f\left(\mathbf{x}^k\right) - \mathbf{x}_k\nabla f\left(\mathbf{x}^k\right) + \frac{1}{2t}\left(\mathbf{x}^\top\mathbf{x} - 2\mathbf{x}^\top\mathbf{x}_k + \mathbf{x}_k^\top\mathbf{x}_k\right) \\ &= f\left(\mathbf{x}^k\right) + \mathbf{x}\nabla f\left(\mathbf{x}^k\right) - \mathbf{x}_k\nabla f\left(\mathbf{x}^k\right) + \frac{1}{2t}\mathbf{x}^\top\mathbf{x} - \frac{1}{t}\mathbf{x}^\top\mathbf{x}_k + \frac{1}{2}\mathbf{x}_k^\top\mathbf{x}_k,\end{aligned} \tag{50}$$

by discarding constant terms:

$$\begin{aligned}&= \frac{1}{2t}\mathbf{x}^\top\mathbf{x} + \mathbf{x}\nabla f\left(\mathbf{x}^k\right) - \frac{1}{t}\mathbf{x}^\top\mathbf{x}_k \\ &= \frac{1}{2t}\mathbf{x}^\top\mathbf{x} - \left(\frac{1}{t}\mathbf{x}_k - \nabla f\left(\mathbf{x}^k\right)\right)\mathbf{x} \\ &= \frac{1}{2t}\left[\underbrace{\mathbf{x}^\top\mathbf{x} - 2t\left(\frac{1}{t}\mathbf{x}_k - \nabla f\left(\mathbf{x}_k\right)\right)\mathbf{x}}_{ax^2+bx+c,\,\text{form}}\right],\end{aligned} \tag{51}$$

we will now need to complete the square for expression (51). With $a = 1$ and

$$b = -2t\left(\frac{1}{t}\mathbf{x}_k - \nabla f\left(\mathbf{x}_k\right)\right)$$

Elementary algebra for completing the square is to express the quadratic form $ax^2 + bx + c$ in the form of

$$a\left[\left(x - \frac{b}{2}\right)^2 + \frac{b^2}{4} + c\right],$$

[6]This is actually an identity as stated in chapter 9 of [3].

which we will do for (51). We have

$$\begin{aligned}\frac{b}{2} &= -\frac{2t\left(\frac{1}{t}\mathbf{x}_k - \nabla f\left(\mathbf{x}_k\right)\right)}{2} \\ &= -\left(\mathbf{x}_k - t\nabla f\left(\mathbf{x}_k\right)\right) \end{aligned} \tag{52}$$

and

$$\left(\frac{b}{2}\right)^2 = \left[-\mathbf{x}_k + t\nabla f\left(\mathbf{x}_k\right)\right]^2$$

Applying the elementary completing the square formula,

$$\begin{aligned}&\frac{1}{2t}\left[\mathbf{x}^\top\mathbf{x} - 2t\left(\frac{1}{t}\mathbf{x}_k - \nabla f\left(\mathbf{x}_k\right)\right)\mathbf{x} + \left(-\mathbf{x}_k + t\nabla f\left(\mathbf{x}_k\right)\right)^2 - \left(-\mathbf{x}_k + t\nabla f\left(\mathbf{x}_k\right)\right)^2\right] \\ &= \frac{1}{2t}\left[\left(\mathbf{x} - \left(\mathbf{x}_k - t\nabla f\left(\mathbf{x}_k\right)\right)\right)^2 - \left(-\mathbf{x}_k + t\nabla f\left(\mathbf{x}_k\right)\right)^2\right] \\ &= \frac{1}{2t}\left[\left(\mathbf{x} - \left(\mathbf{x}_k - t\nabla f\left(\mathbf{x}_k\right)\right)\right)^2 - \text{constant}\right] \\ &= \frac{1}{2t}\left(\mathbf{x} - \left(\mathbf{x}_k - t\nabla f\left(\mathbf{x}_k\right)\right)\right)^2 - \frac{1}{2t}\text{constant} \\ &= \frac{1}{2t}\left|\left|\mathbf{x} - \left(\mathbf{x}_k - t\nabla f\left(\mathbf{x}_k\right)\right)\right|\right|_2^2 + \text{constant}. \end{aligned} \tag{53}$$

Combining with the function $g\left(\mathbf{x}\right)$, we get the desired equation

$$\frac{1}{2t}\left|\left|\mathbf{x} - \left(\mathbf{x}_k - t\nabla f\left(\mathbf{x}_k\right)\right)\right|\right|_2^2 + g\left(\mathbf{x}\right) \tag{54}$$

B Lyapunov Functions and Gradient Methods

B.1 Differential equation of the gradient descent

References

[1] J. C. Meza, "Steepest descent," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 6, pp. 719–722, 2010.

[2] C. Lemaréchal, "Cauchy and the gradient method."

[3] A. Beck, *First-order methods in optimization*. SIAM, 2017, vol. 25.

[4] T. Goldstein, C. Studer, and R. Baraniuk, "A field guide to forward-backward splitting with a fasta implementation," *arXiv preprint arXiv:1411.3406*, 2014.

[5] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[6] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[7] Y. E. Nesterov, "A method for solving the convex programming problem with convergence rate $o\left(\frac{1}{k^2}\right)$," in *Dokl. akad. nauk Sssr*, vol. 269, 1983, pp. 543–547.

[8] A. Nemirovsky, "Problem complexity and method efficiency in optimization."

[9] E. Ghadimi, H. R. Feyzmahdavian, and M. Johansson, "Global convergence of the heavy-ball method for convex optimization," in *2015 European Control Conference (ECC)*. IEEE, 2015, pp. 310–315.

[10] D. G. Anderson, "Iterative procedures for nonlinear integral equations," *Journal of the ACM (JACM)*, vol. 12, no. 4, pp. 547–560, 1965.

[11] V. V. Mai and M. Johansson, "Anderson acceleration of proximal gradient methods," *arXiv preprint arXiv:1910.08590*, 2019.

[12] E. K. Ryu and S. Boyd, "Primer on monotone operator methods," *Appl. Comput. Math*, vol. 15, no. 1, pp. 3–43, 2016.

[13] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.