10-725: Optimization **Fall 2012**

Lecture 9: September 25

Lecturer: Geoff Gordon/Ryan Tibshirani Scribes: Xuezhi Wang, Subhodeep Moitra, Abhimanu Kumar

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

9.1 Review of Generalized Gradient Descent

Generalized Gradient Descent is used to solve the following problem,

$$\min_x \; f(x) = g(x) + h(x)$$

where g is convex and differentiable, $h(x)$ is convex and not necessarily differentiable. Particularly, here are some special cases of Generalized Gradient Descent (therefore all have $O(1/k)$ convergence rate):

- $h = 0$, which is simply the gradient descent.
- $h = I_C$, which is called projected gradient descent. The problem is defined as:
  Given closed, convex set $C \in \mathbb{R}^n$, we want:

  $$\min_{x \in C} g(x)$$

  which is equivalent to

  $$\min_x g(x) + I_C(x)$$

  where $I_C(x) = \begin{cases} 0 & x \in C \\ \infty & x \notin C \end{cases}$ is the indicator function of C. Hence

  $$\text{prox}_t(x) = \arg\min_z \frac{1}{2t}||x - z||^2 + I_C(z) = \arg\min_{z \in C} ||x - z||^2$$

  which is $P_C(x)$, the projection operator onto C. Therefore the generalized gradient update step is: $x^+ = P_C(x - t\nabla g(x))$, which performs usual gradient descent update and then project back to C. This is called **projected gradient descent**.
  Some of the easy-to-project-onto sets C are:

  1. Affine images $C = \{Ax + b : x \in \mathbb{R}^n\}$
  2. Solution set of linear system $C = \{x \in \mathbb{R}^n : Ax = b\}$
  3. Nonnegative orthant $C = \{x \in \mathbb{R}^n : x \succeq 0\} = \mathbb{R}^n_+$
  4. Norm balls $C = \{x \in \mathbb{R}^n : ||x||_p \leq 1\}$, for $p = 1, 2, \infty$
  5. Some simple polyhedra and simple cones

  However, it's worth to note that P_C can be very hard even for seemingly simple set C, like it's generally very hard to project onto solution set of arbitrary linear inequalities, i.e., arbitrary polyhedron $C = \{x \in \mathbb{R}^n : Ax \preceq b\}$.

- $g = 0$, which is called proximal minimization. Then generalized gradient update step is just a prox update

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

This is faster than subgradient method ($O(1/k)$ compared to $O(1/\sqrt{k})$), but it's not implementable unless we know prox function in a closed form.

Some issues regarding the Generalized Gradient Descent method:
In Generalized Gradient Descent, we assume that the minimization $\text{prox}_t(x) = \arg\min_z \frac{1}{2t}||x-z||^2 + h(z)$ can be done exactly, what if we cannot evaluate the prox function? In this case, if we just treat this as another minimization problem and obtain an approximate solution, all bets are off and the practical convergence rate can be very slow. But there are also some exceptions like partial proximation minimization [B94].
In the next section we'll be talking about acceleration, which you can get some flavor by looking at Fig 9.1.

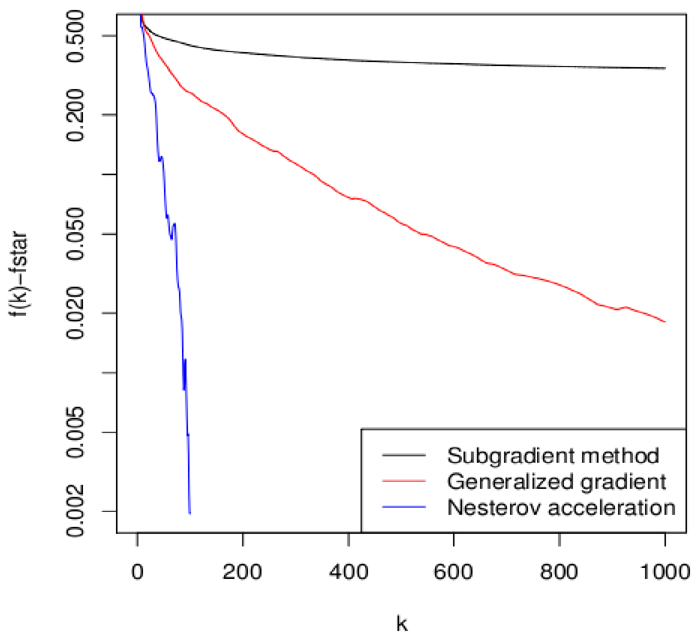

Figure 9.1: Comparison of different gradient methods

9.2 Acceleration for composite functions

There are four acceleration ideas proposed by Nesterov (1983, 1998, 2005, 2007). The first two are proposed for smooth functions. The third idea is smoothing techniques for nonsmooth functions, coupled with original acceleration idea. The fourth is the acceleration idea for composite functions, which requires entire history of previous steps and makes two prox calls in each step.
In 2008 Beck and Teboulle extend Nesterov's idea (1983) to composite functions, which uses only information from two last steps and makes one prox call. In this note we mainly focus on this idea.

9.2.1 Accelerated generalized gradient method

The problem is:

$$\min_{x\in\mathbb{R}^n} \; g(x)+h(x)$$

where $g(x)$ is convex and differentiable, $h(x)$ is convex and not necessarily differentiable.
The **accelerated generalized gradient descent method** works in this way:

- choose any initial $x^{(0)} = x^{(-1)} \in \mathbb{R}^n$
- repeat for $k = 1, 2, 3, \ldots$

$$y = x^{(k-1)} + \frac{k-2}{k+1}(x^{(k-1)} - x^{(k-2)})$$
$$x^{(k)} = \text{prox}_{t_k}(y - t_k\nabla g(y))$$

Some notes about the accelerated generalized gradient method are:

1. First step $k = 1$ is just usual generalized gradient update: $x^{(1)} = \text{prox}_{t_1}(x^{(0)} - t_1\nabla g(x^{(0)}))$
2. After the first step, the method carries some "momentum" from previous iterations
3. $h = 0$ gives accelerated gradient method
4. The method accelerates more towards the end of iterations, as shown in Fig 9.2.

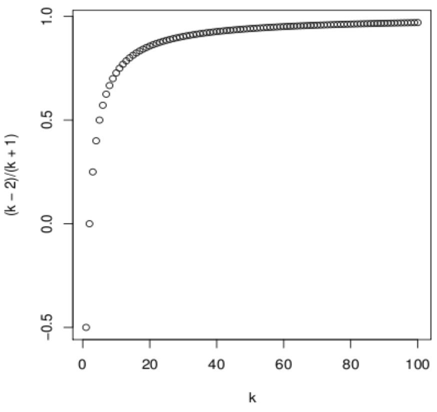

Figure 9.2: A figure showing the acceleration coefficient varying with number of iterations

Here are two examples (Fig 9.3) showing the performance of accelerated gradient descent compared with usual gradient descent.

9.2.2 Reformulation of the accelerated generalized gradient method

To make the convergence analysis easier, we can reformulate the accelerated generalized gradient method as:

- Initialize $x^{(0)} = u^{(0)}$

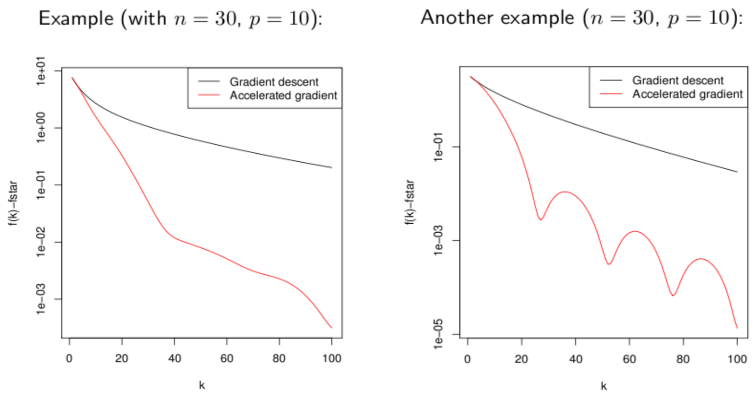

Figure 9.3: Performance of accelerated gradient descent compared with usual gradient descent

- repeat for $k = 1, 2, 3, \ldots$

$$y = (1-\theta_k)x^{(k-1)} + \theta_k u^{(k-1)}$$
$$x^k = \text{prox}_{t_k}(y - t_k \nabla g(y))$$
$$u^{(k)} = x^{(k-1)} + \frac{1}{\theta_k}(x^{(k)} - x^{(k-1)})$$

where $\theta = 2/(k+1)$. Note this reformulation is equivalent to the accelerated generalized gradient method presented in section 9.2.1 since $u^{(k-1)} = x^{(k-2)} + \frac{1}{\theta_{k-1}}(x^{(k-1)} - x^{(k-2)})$, then we have:

$$\begin{aligned} y &= (1-\theta_k)x^{(k-1)} + \theta_k u^{(k-1)} \\ &= (1-\theta_k)x^{(k-1)} + \theta_k x^{(k-2)} + \frac{\theta_k}{\theta_{k-1}}(x^{(k-1)} - x^{(k-2)}) \\ &= x^{(k-1)} + (\frac{\theta_k}{\theta_{k-1}} - \theta_k)(x^{(k-1)} - x^{(k-2)}) \\ &= x^{(k-1)} + \frac{k-2}{k+1}(x^{(k-1)} - x^{(k-2)}) \end{aligned}$$

9.3 Convergence Analysis

Just as the generalized gradient method, we minimize $f(x) = g(x) + h(x)$ assuming that: g is convex, differentiable, ∇g is Lipschitz with constant $L > 0$, and h is convex, the prox function can be evaluated.

Theorem 9.1 *Accelerated generalized gradient method with fixed step size $t \leq 1/L$ satisfies:*

$$f(x^{(k)}) - f(x^*) \leq \frac{2||x^{(0)} - x^*||^2}{t(k+1)^2}$$

This theorem tells us that accelerated generalized gradient method can achieve the optimal $O(1/k^2)$ rate for first-order method, or equivalently, if we want to get $f(x^{(k)}) - f(x^*) \leq \epsilon$, we only need $O(1/\sqrt{\epsilon})$ iterations. Now we prove this theorem.

Proof: In the proof we focus on one iteration and drop k notation, so x^+, u^+ are updated versions of x, u. First we bound both $g(x^+)$ and $h(x^+)$.

- Since $t \leq 1/L$ and ∇g is Lipschitz with constant $L > 0$, we have
$$g(x^+) \leq g(y) + \nabla g(y)^T(x^+ - y) + \frac{L}{2}||x^+ - y||^2 \leq g(y) + \nabla g(y)^T(x^+ - y) + \frac{1}{2t}||x^+ - y||^2 \tag{9.1}$$
- Suppose
$$v = \text{prox}_t(w) = \arg\min_v \frac{1}{2t}||w - v||^2 + h(v)$$
we have
$$0 \in \partial(\frac{1}{2t}||w - v||^2 + h(v)) = -\frac{1}{t}(w - v) + \partial h(v) \quad \Rightarrow \quad -\frac{1}{t}(w - v) \in \partial h(v)$$
According to the definition of subgradient, we have for all z,
$$h(z) \geq h(v) - \frac{1}{t}(v - w)^T(z - v) \quad \Rightarrow \quad h(v) \leq h(z) + \frac{1}{t}(v - w)^T(z - v)$$
for all z, w and $v = \text{prox}_t(w)$. Since $x^+ = \text{prox}_t(y - t\nabla g(y))$, substitute x^+ in the above inequality we get for all z,
$$h(x^+) \leq h(z) + \frac{1}{t}(x^+ - y + t\nabla g(y))^T(z - x^+) = h(z) + \frac{1}{t}(x^+ - y)^T(z - x^+) + \nabla g(y)^T(z - x^+) \tag{9.2}$$

Adding inequation 9.1 and 9.2 we get for all z,
$$f(x^+) \leq g(y) + h(z) + \frac{1}{t}(x^+ - y)^T(z - x^+) + \frac{1}{2t}||x^+ - y||^2 + \nabla g(y)^T(z - y)$$
Using $g(z) \geq g(y) + \nabla g(y)^T(z - y)$ since g is convex, we further get
$$f(x^+) \leq f(z) + \frac{1}{t}(x^+ - y)^T(z - x^+) + \frac{1}{2t}||x^+ - y||^2$$

Now take $z = x$ and $z = x^*$ and multiply both sides by $(1 - \theta)$ and θ respectively,
$$(1 - \theta)f(x^+) \leq (1 - \theta)f(x) + \frac{1 - \theta}{t}(x^+ - y)^T(x - x^+) + \frac{1 - \theta}{2t}||x^+ - y||^2$$
$$\theta f(x^+) \leq \theta f(x^*) + \frac{\theta}{t}(x^+ - y)^T(x^* - x^+) + \frac{\theta}{2t}||x^+ - y||^2$$
Adding these two inequalities together, we get
$$f(x^+) - f(x^*) - (1 - \theta)(f(x) - f(x^*)) \leq \frac{1}{t}(x^+ - y)^T((1 - \theta)x + \theta x^* - x^+) + \frac{1}{2t}||x^+ - y||^2 \tag{9.3}$$
Using $u^+ = x + \frac{1}{\theta}(x^+ - x)$ and $y = (1-\theta)x + \theta u$, we have $(1-\theta)x + \theta x^* - x^+ = \theta(x^* - u^*)$ and $x^+ - y = \theta(u^+ - u)$, substitute these equations to the RHS of inequation 9.3 we have
$$\begin{aligned} f(x^+) - f(x^*) - (1 - \theta)(f(x) - f(x^*)) &\leq \frac{\theta}{2t}(u^+ - u)^T[2\theta(x^* - u^+) + \theta(u^+ - u)] \\ &= \frac{\theta^2}{2t}[(x^* - u) - (x^* - u^+)]^T[(x^* - u^+) + (x^* - u)] \\ &= \frac{\theta^2}{2t}(||x^* - u||^2 - ||x^* - u^+||^2) \end{aligned}$$

Back to k notation we have:
$$\frac{t}{\theta_k^2}(f(x^{(k)}) - f(x^*)) + \frac{1}{2}||u^{(k)} - x^*||^2 \leq \frac{t(1 - \theta_k)}{\theta_k^2}(f(x^{(k-1)}) - f(x^*)) + \frac{1}{2}||u^{(k-1)} - x^*||^2$$

Using $\frac{1-\theta_k}{\theta_k^2} \le \frac{1}{\theta_{k-1}^2}$ we have

$$\frac{t}{\theta_k^2}(f(x^{(k)}) - f(x^*)) + \frac{1}{2}||u^{(k)} - x^*||^2 \le \frac{t}{\theta_{k-1}^2}(f(x^{(k-1)}) - f(x^*)) + \frac{1}{2}||u^{(k-1)} - x^*||^2$$

Iterate this inequality and use $\theta_1 = 1, u^{(0)} = x^{(0)}$ we get

$$\frac{t}{\theta_k^2}(f(x^{(k)}) - f(x^*)) + \frac{1}{2}||u^{(k)} - x^*||^2 \le \frac{t(1-\theta_1)}{\theta_1^2}(f(x^{(0)}) - f(x^*)) + \frac{1}{2}||u^{(0)} - x^*||^2 = \frac{1}{2}||x^{(0)} - x^*||^2$$

Hence we conclude

$$f(x^{(k)}) - f(x^*) \le \frac{\theta_k^2}{2t}||x^{(0)} - x^*||^2 = \frac{2||x^{(0)} - x^*||^2}{t(k+1)^2}$$

■

9.4 Accelerated Backtracking Line Search

In the proof for accelerated general gradient descent we saw an $O(1/k^2)$ convergence rate which optimal. Gradient descent on the other hand has a $O(1/k)$ convergence rate. The proofs for these convergence rates are very different and are made under different assumptions. There are a number of different accelerated backtracking schemes and these are made under different criteria for the same reason. We will examine one of the simpler schemes.

The assumptions that this scheme needs to make are :

- Lipschitz Gradient condition
$$g(x^+) \le g(y) + \nabla g(y)^T(x^+ - y) + \frac{1}{2t}\|x^+ - y\|^2$$
- Condition on θ_k
$$\frac{(1-\theta_k)t_k}{\theta_k^2} \le \frac{t_{k-1}}{\theta_{k-1}^2}$$
- t_k is monotonically decreasing i.e. $t_k \le t_{k-1}$. The problem with this condition is that if you choose a small step size initially, you will need to continue to use small step sizes further on as well.

9.4.1 Algorithm

```
Choose $\beta < 1$
$t_0 = 1$
**for** $k = 1, 2, 3, \ldots$ till convergence
    $t_k = t_{k-1}$
    $x^+ = \text{prox}_{t_k}(y - t_k \nabla g(y))$
    **while** $g(x^+) > g(y) + \nabla g(y)^T(x^+ - y) + \frac{1}{2t_k}\|x^+ - y\|^2$ **repeat**
        $t_k = \beta t_k$
        $x^+ = \text{prox}_{t_k}(y - t_k \nabla g(y))$
    **endwhile**
**endfor**
```

This method checks if $g(x^+)$ is small enough, otherwise it shrinks t_k by a factor β and updates x^+. This method satisfies the required conditions and is thus able to achieve a $O(1/k^2)$ convergence rate.

9.4.2 Convergence rates

From the above discussion, the following theorem summarizes the $O(1/k^2)$ convergence rate

Theorem 9.2 *Accelerated generalized gradient method with backtracking satisfies*

$$f(x^{(k)}) - f(x^*) \leq \frac{2\|x^{(0)} - x^*\|^2}{t_{\min}(k+1)^2}$$

9.5 FISTA

Fast Iterative Soft Thresholding Algorithm(FISTA) is an accelerated version of ISTA which is applied to problems containing convex differentiable objectives with L1 norm such as Lasso. The Lasso problem is defined as :

$$\min_x \frac{1}{2}\|y - Ax\|^2 + \lambda\|x\|_1$$

ISTA solution This is the solution by using the normal generalized gradient also known as the iterative soft thresholding algorithm (ISTA). See Lecture 8.

$$x^{(k)} = S_{\lambda t_k}(x^{(k-1)} + t_k A^T(y - Ax^{(k-1)})) \quad k = 1, 2, 3, \ldots$$

where $S_\lambda(\cdot)$ is the soft-thresholding operator

$$[S_\lambda(x)]_i = \begin{cases} x_i - \lambda & \text{if } x_i > \lambda \\ 0 & \text{if } -\lambda \leq x_i \leq \lambda \\ x_i + \lambda & \text{if } x_i < -\lambda \end{cases}$$

This is obtained by solving the prox function

$$prox_t(x) = \arg\min_{z \in \mathbb{R}^n} \frac{1}{2t}\|x - z\|^2 + \lambda\|z\|_1 = S_{\lambda t}(x)$$

FISTA solution The accelerated version involves solving the same prox problem but with an additional vector added to the input to the prox function.

$$\begin{aligned} v &= x^{(k-1)} + \frac{k-2}{k+1}(x^{(k-1)} - x^{(k-2)}) \\ x^{(k)} &= S_{\lambda t_k}(v + t_k A^T(y - Ax^{(k-1)})) \quad k = 1, 2, 3, \ldots \end{aligned}$$

Here we show two images comparing the performance of ISTA vs FISTA. We can see that FISTA clearly beats ISTA by an order of magnitude faster.

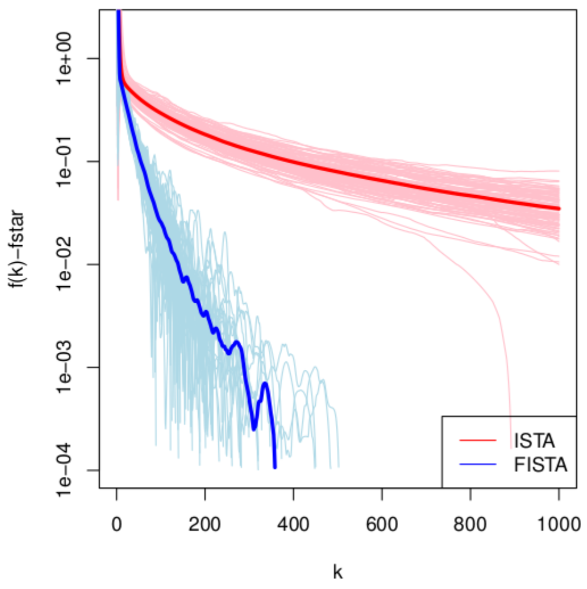

Figure 9.4: Performance of ISTA vs FISTA for Lasso Regression (n=100,p=500)

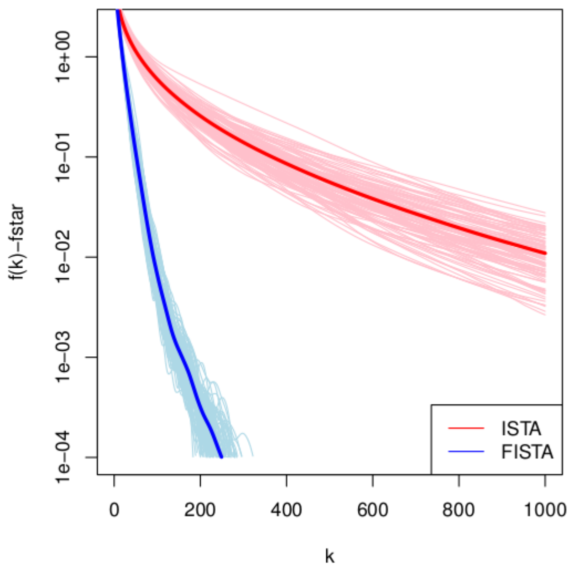

Figure 9.5: Performance of ISTA vs FISTA for Lasso Logistic Regression (n=100,p=500)

9.6 Failure Cases for Acceleration

Acceleration achieves the optimal $O(1/k^2)$ convergence rate for gradient based methods. However, it does not do so under all conditions. In some cases it might perform similar to non-accelerated methods and in some others it might actually hurt performance. Some cases are presented wherein acceleration fails to do well.

9.6.1 Warm Starts

In iterative algorithms warm starting can be an effective strategy to speed up convergence. For e.g. in cross validation runs for Lasso , one can use the optimal $\hat{x}$ found in the previous iteration as a warm start for the next iteration. Let the tuning parameters for lasso be

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_r$$

When solving for λ_1, initialize $x^{(0)} = 0$ and record solution $\hat{x}(\lambda_1)$. Now, reuse this value such that when solving for λ_j, initialize $x^{(0)}(\lambda_j) = \hat{x}(\lambda_{j-1})$. It has been observed that over a fine grid of values, generalized gradient descent can perform just as well as accelerated version when using warm starts.

9.6.2 Matrix Completion

In the case of matrix completion, acceleration and even backtracking can hurt performance. The matrix completion problem is described in Lecture 8. Briefly, Given a matrix A, only some entries $(i,j) \in \Omega$ of which are visible to you, you want to fill in the rest of entries, while keeping the matrix low rank. We solve,

$$\min_X \frac{1}{2}\|P_\Omega(A) - P_\Omega(X)\|_F^2 + \lambda\|X\|_*$$

where $\|X\|_* = \sum_{i=1}^r \sigma_i(X)$ is the nuclear norm, r is the rank of X and $P_\Omega(\cdot)$ is the projection operator,

$$[P_\Omega(X)]_{ij} = \begin{cases} X_{ij} & (i,j) \in \Omega \\ 0 & (i,j) \notin \Omega \end{cases}$$

the gradient descent updates, also known as the *soft-impute* algorithm are

$$X^+ = S_\lambda(P_\Omega(A) + P_\Omega^\perp(X))$$

where $S_\lambda(\cdot)$ is the matrix soft-thresholding operator which requires the SVD to compute as $S_\lambda(X) = U\Sigma_\lambda V^T$ where $(\Sigma_\lambda)_{ii} = \max\{\Sigma_{ii} - \lambda, 0\}$. Calculating the SVD can be expensive and can cost upto $O(mn^2)$ operations.

This can be expensive for the backtracking search method since each backtracking loop evaluates the generalized $G_t(X)$ at various values of t and this involves solving the prox function. This equates to calling the SVD several times and can be slow.

The matrix completion problem does not work well with acceleration as well, since acceleration involves changing the argument passed to the prox function. We pass $y - t\nabla g(y)$ instead of $x - t\nabla g(x)$. This can make the matrix high rank which makes the computation of SVD more expensive.

Here is an example where using acceleration performs worse than soft-impute [M11].

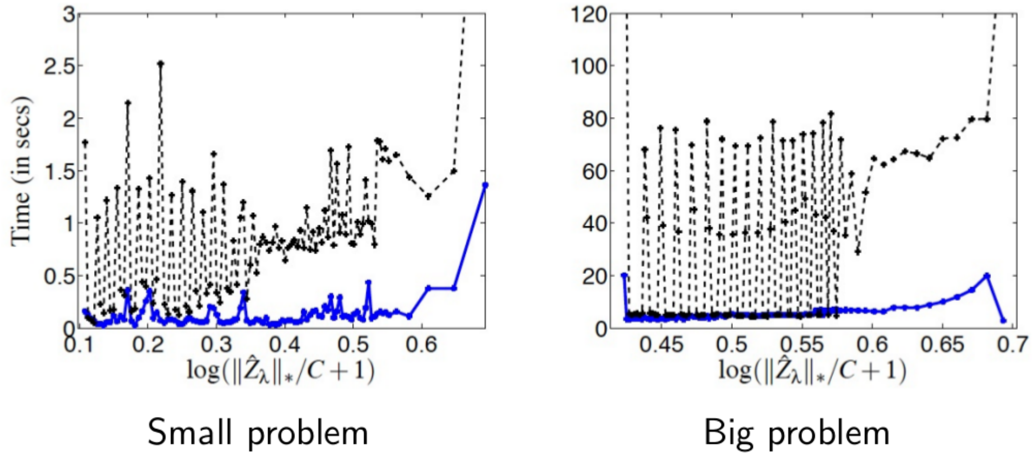

Figure 9.6: Performance of Soft impute vs Accelerated grad descent on a small and a big problem

References

[M11] R. Mazumder and T. Hastie and R. Tibshirani "Spectral Regularization Algorithms for Learning Large Incomplete Matrices," *The Journal of Machine Learning Research*, 2011, pp. 2287–2322.

[B94] Bertsekas and Tseng "Partial proximal minimization algorithms for convex programming" 1994.