10-725/36-725: Convex Optimization **Fall 2016**

Lecture 9: September 28

Lecturer: Ryan Tibshirani *Scribes: Yiming Wu, Ye Yuan, Zhihao Li*

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

9.1 Example: matrix completion

The *prox* operator is complicated, but the algorithm we get out is quite interesting. One example is matrix completion. This problem got attention starting from around 8 years ago. One example of matrix completion problem is predicting rates on Netflix. Assuming we have an incomplete matrix M where M_{ij} denotes User i's rate of movie j on Netflix. In practice this matrix has lots of blanks. Our mission is to try to predict the rate of unrated movie j' from User i'

Formally, given a matrix $Y \in \mathbb{R}^{m \times n}$, and only observe entries Y_{ij}, $(i,j) \in \Omega$. Suppose we want to fill in missing entries (e.g., for a recommender system), so we solve a matrix completion problem:

$$\min_B \underbrace{\frac{1}{2} \sum_{(i,j) \in \Omega} (Y_{ij} - B_{ij})^2}_{close\ to\ observed\ entries} + \underbrace{\lambda ||B||_{tr}}_{low\ rank}$$

Here B is the missing part of matrix we are going to predict. Formally, B will be the entries that are not in Ω. $||B||_{tr}$ is the trace (or nuclear) norm of B,

$$||B||_{tr} = \sum_{i=1}^{r} \sigma_i(B)$$

where $r = rank(B)$ and $\sigma_0(X) \geq \cdots \geq \sigma_r(X) \geq 0$ are the singular values. What we are doing here is that we want the prediction B close to observed entries Y and be low-rank as well.

Another prospect of trace norm is like the $l1$ norm in lasso. For a diagonal matrix, taking trace norm is like taking an 1-norm of the diagonal vector.

This is a convex problem because the first part $\frac{1}{2} \sum_{(i,j) \in \Omega} (Y_{ij} - B_{ij})^2$ is quadratic. The second half is a norm, which is convex. You can check some classic matrix analysis textbook for that.

So how do we solve this question? Before people start using proximal gradient, this problem is pretty hard. This is a semi-definite program. We used to solve this with interior point method, which is very slow for problem like this. Proximal gradient fires a much faster algorithm compared to alternative solutions.

So here is the way we set it up. Define P_Ω, projection operator onto observed set:

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

Then the criterion is

$$f(B) = \underbrace{\frac{1}{2}||P_\Omega(Y) - P_\Omega(B)||_F^2}_{g(B)} + \underbrace{\lambda||B||_{tr}}_{h(B)}$$

This is the same loss we saw at the beginning of this section. Here we call the first part $g(B)$, which is convex and smooth. The second part $h(B)$, which is also convex but not smooth.

Then we can think about two ingredients needed for proximal gradient descent:

- The first is $\nabla g(B)$. Here subgradient is just the gradient.

$$\nabla g(B) = -(P_\Omega(Y) - P_\Omega(B))$$

- Then is the *prox* operator.

$$prox_t(B) = \arg\min_Z \frac{1}{2t}||B - Z||_F^2 + \lambda||Z||_{tr}$$

Lemma 9.1 *$prox_t(B) = S_{\lambda t}(B)$, matrix soft-thresholding at the level λ. Here $S_\lambda(B)$ is defined by*

$$S_\lambda(B) = U\Sigma_\lambda V^T$$

where $B = U\Sigma V^T$ is an SVD, and Σ_λ is diagonal with

$$(\Sigma_\lambda)_{ii} = \max\{\Sigma_{ii} - \lambda, 0\}$$

This matrix soft-thresholding is nothing but element-wise soft-thresholding of the matrix. With a soft-threaded singular matrix Σ_λ, this is giving us a low-rank *prox* result.

Proof: Note that $prox_t(B) = Z$. If we apply subgradient optimality, then Z satisfies

$$0 \in Z - B + \lambda t \cdot \partial||Z||_{tr}$$

The main difficulty is in computing the subgradient of trace norm of the matrix. We are not going to prove it here. If you are interested in proving it yourself, you can try to prove the following fact from both sides.

Fact: if $Z = U\Sigma V^T$, then

$$\partial||Z||_{tr} = \{UV^T + W : ||W||_op \leq 1, U^TW = 0, WV = 0\}$$

Now plug in $Z = S_{\lambda t}(B)$ and check that we can get 0 ■

To illustrate more about the subgradient of trace norm, you can start with using it's dual, max norm. To get subgradient of $||Z||_{tr}$, use

$$\begin{aligned} ||Z||_* &= \max_{||y||_{op} \leq 1} Z \cdot Y \\ &= \max_{||y||_{op} \leq 1} tr(Z^TY) \end{aligned}$$

Hence proximal gradient update step is:

$$\begin{aligned} B^+ &= prox_t(B - t\nabla g(B)) \\ &= S_{\lambda t}(B + t(P_\Omega(Y) - P_\Omega(B))) \end{aligned}$$

Note that $\nabla g(B)$ is Lipschitz continuous with $L = 1$, so we can choose fixed step size $t = 1$. Update step is now:

$$B^+ = S_\lambda(P_\Omega(Y) + P_\Omega^\perp(B))$$

where $P_\Omega^\perp(B)$ projects onto unobserved set, $P_\Omega(B) + P_\Omega^\perp(B) = B$.

This can be interpreted as: at every step, we look at all unobserved entries from our estimate. We also take everything from the observed entries. And we do matrix soft-thresholding on this combined matrix. This is the soft-impute algorithm[CW88], a simple and effective method for matrix completion

9.2 Special cases of proximal gradient descent

Recall that proximal mapping is defined as

$$\text{prox}_t(x) = \arg\min_z \frac{1}{2t}\|x - z\|_2^2 + h(z). \tag{9.1}$$

Consider the problem

$$\min g(x) + h(x), \tag{9.2}$$

where g is convex and differentiable, and h is convex but possibly non-differentiable.

Proximal gradient descent will choose an initial $x^{(0)}$ and repeat the following step:

$$x^{(k)} = \text{prox}_{t_k}\left(x^{(k-1)} - t_k \nabla g(x^{(k-1)})\right), \quad k = 1, 2, 3, \cdots \tag{9.3}$$

Proximal gradient descent is also called composite gradient descent or generalized gradient descent. We will see some special cases to understand why it is generalized.

9.2.1 Gradient descent

When $h(x) = 0$, proximal mapping becomes

$$\begin{aligned} \text{prox}_t(x) &= \arg\min_z \frac{1}{2t}\|x - z\|_2^2 & (9.4)\\ &= x & (9.5) \end{aligned}$$

Thus, the update rule becomes

$$x^{(k)} = x^{(k-1)} - t_k \nabla g(x^{(k-1)}), \quad k = 1, 2, 3, \cdots \tag{9.6}$$

which is just the normal gradient descent.

9.2.2 Projected gradient descent

When $h(x) = I_C$, where I_C is the indicator function of set C, proximal mapping becomes

$$\begin{aligned} \text{prox}_t(x) &= \arg\min_z \frac{1}{2t}\|x - z\|_2^2 + I_C & (9.7)\\ &= \arg\min_{z \in C} \frac{1}{2t}\|x - z\|_2^2 & (9.8)\\ &= P_C(x) & (9.9) \end{aligned}$$

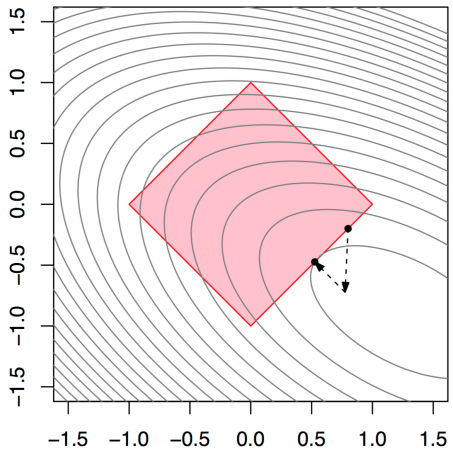

Figure 9.1: Projected gradient descent.

where $P_C(x)$ is the projection operator onto set C.

Hence, the update rule becomes

$$x^{(k)} = P_C\left(x^{(k-1)} - t_k \nabla g(x^{(k-1)})\right), \quad k = 1, 2, 3, \cdots \tag{9.10}$$

which is the projected gradient descent. It performs usual gradient update and then project x back onto C.

9.2.3 Proximal minimization algorithm

When $g(x) = 0$, the update rule becomes

$$x^{(k)} = \arg\min_z \frac{1}{2t}\|x^{(k-1)} - z\|_2^2 + h(z), \quad k = 1, 2, 3, \cdots \tag{9.11}$$

which is called proximal minimization algorithm. It is mainly used for $h(x)$ that is convex but not necessarily differentiable. It is faster than subgradient method but is only implementable when the proximal operator is in closed form.

9.2.4 What happens if we cannot evaluate proximal operator?

When proximal operator cannot be evaluated exactly, we can still recover the original convergence rate if we can precisely control the errors in approximating it. In another words, if proximal operator is done approximately, it should be done to decently high accuracy.

9.3 Acceleration of proximal gradient method

Consider the same problem:

$$\min \quad g(x) + h(x). \tag{9.12}$$

Accelerated proximal gradient chooses initial point $x^{(0)} = x^{(-1)} \in \mathbb{R}^n$ and repeat the following steps:

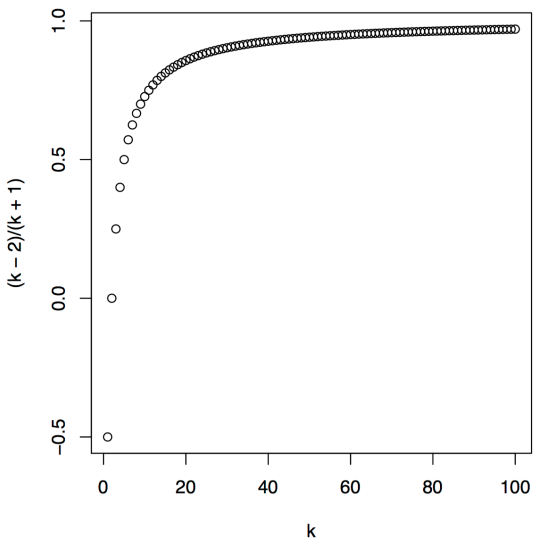

Figure 9.2: momentum weights.

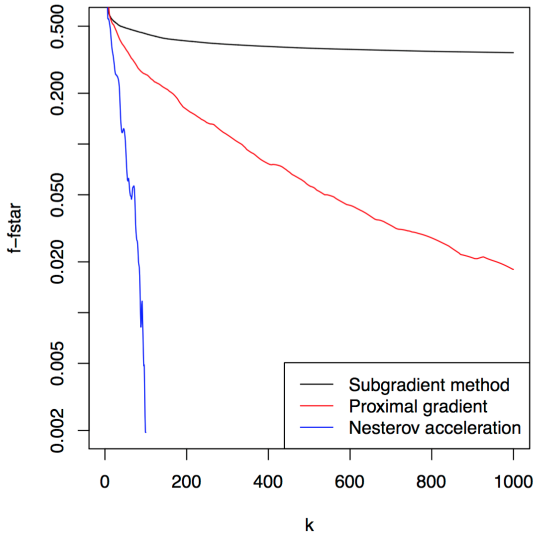

Figure 9.3: convergence rate comparison for lasso.

1. calculate v by adding momentum from previous iterations,

$$v = x^{(k-1)} + \frac{k-2}{k+1}(x^{(k-1)} - x^{(k-2)}) \tag{9.13}$$

2. use v to update x,

$$x^{(k)} = \text{prox}_{t_k}\left(v - t_k \nabla g(x^{(k-1)})\right) \tag{9.14}$$

The first step $k = 1$ is just usual proximal gradient update. After that, we will use v to substitute $x^{(k-1)}$ to carry some momentum from previous iterations. When $h(x) = 0$, it becomes accelerated gradient method.

As we can see from figure 9.3, accelerated proximal gradient is no longer a descent method.

9.3.1 Convergence analysis

As usual, we are minimizing $f(x) = g(x) + h(x)$. we assume the following:

- $\text{dom}(f) = \mathbb{R}^n$.
- g is convex, differentiable, and ∇g is Lipschitz continuous with constant $L > 0$.
- h is convex, and its proximal operator can be evaluated.

Theorem 9.2 *Accelerated proximal gradient method with fixed step size $t \leq 1/L$ satisfies*

$$f(x^{(k)}) - f^\star \leq \frac{2\|x^{(0)} - x^\star\|_2^2}{t(k+1)^2} \tag{9.15}$$

Note that accelerated proximal gradient method can achieve the optimal rate $O(1/k^2)$ for first-order methods.

9.3.2 Acceleration with backtracking line search

A simple way to accelerate proximal gradient method with acceleration is to fix $\beta < 1, t_0 = 1$, and at iteration k, do the following steps:

1. calculate $x^{(k)} = \text{prox}_t\left(v - t\nabla g(x^{(k-1)})\right)$
2. check if $g(x^{(k)}) > g(v) + \nabla g(v)^T(x^{(k)} - v) + \frac{1}{2t}\|x^{(k)} - v\|_2^2$. If true, go to step 2. Else, end current iteration.
3. shrink t to βt, and go back to step 2.

Theorem 9.3 *Accelerated proximal gradient method with back- tracking line search satisfies*

$$f(x^{(k)}) - f^\star \leq \frac{2\|x^{(0)} - x^\star\|_2^2}{t_{min}(k+1)^2} \tag{9.16}$$

where $t_{min} = \min\{1, \beta/L\}$.

9.3.3 Example: FISTA

Recall that lasso problem has the following form:

$$\min_\beta \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1. \tag{9.17}$$

And Iterative Soft-thresholding Algorithm (ISTA) has the following update rule:

$$\beta^{(k)} = S_{\lambda t_k}\left(\beta^{(k-1)} + t_k X^T(y - X\beta^{(k-1)})\right), \quad k = 1, 2, 3, \cdots \tag{9.18}$$

where $S_\lambda(\cdot)$ is vector soft-thresholding. Applying acceleration gives us Fast Iterative Soft-thresholding Algorithm (FISTA): for $k = 1, 2, 3, \cdots$,

$$v = \beta^{(k-1)} + \frac{k-2}{k+1}(\beta^{(k-1)} - \beta^{(k-2)}) \tag{9.19}$$

$$\beta^{(k)} = S_{\lambda t_k}\left(v + t_k X^T(y - X\beta^{(k-1)})\right) \tag{9.20}$$

9.3.4 Is acceleration always helpful?

Acceleration can be a very effective speedup tool, but it may not be suitable for some cases.

9.3.4.1 Warm starts

In practice the speedup of using acceleration is diminished in the presence of warm starts. Suppose we want to solve lasso problem for tuning parameters values

$$\lambda_1 > \lambda_2 > \cdots > \lambda_r$$

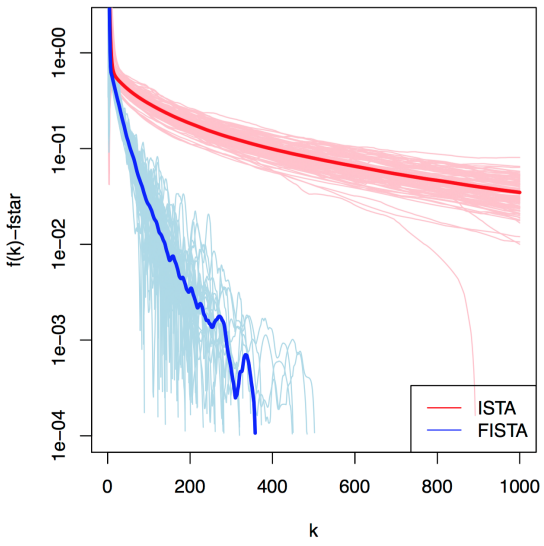

Figure 9.4: lasso regression.

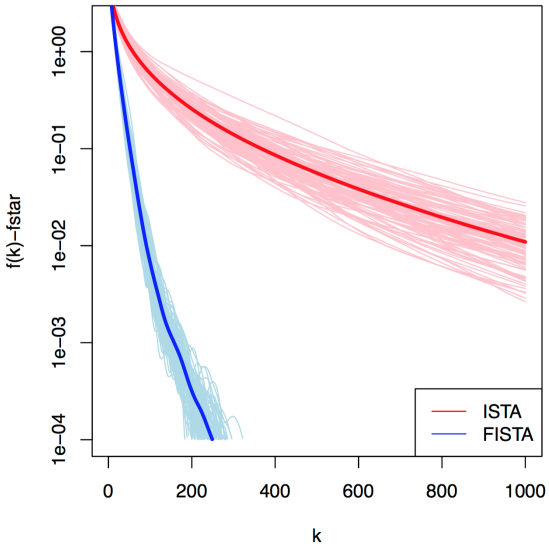

Figure 9.5: lasso logistic regression.

- When solving for λ_1, initialize $x^{(0)} = 0$, record solution $\hat{x}(\lambda_1)$.
- When solving for λ_j, initialize $x^{(0)} = \hat{x}(\lambda_{j-1})$, record solution $\hat{x}(\lambda_j)$.

Over a fine enough grid of values, proximal gradient descent can often perform just as well without acceleration.

9.3.4.2 Matrix completion

Recall that the proximal gradient update for matrix completion problem is

$$B^+ = S_\lambda\left(B + t\left(P_\Omega(Y) - P^\perp(B)\right)\right) \tag{9.21}$$

where S_λ is the matrix soft-thresholding operator, which requires SVD.

If we use backtracking line search, we will evaluate generalized gradient $G_t(x)$, *i.e*, evaluate proximal operator across various values of t. It means multiple SVDs, which is very computationally expensive.

If we use acceleration, it will change the argument we pass to proximal operator. For matrix completion, we were originally passing

$$B - \nabla g(B) = \underbrace{P_\Omega(Y)}_{\text{sparse}} + \underbrace{P_\Omega^\perp(Y)}_{\text{low rank}},$$

which means fast SVD. But now we are passing

$$V - \nabla g(V) = \underbrace{P_\Omega(Y)}_{\text{sparse}} + \underbrace{P_\Omega^\perp(V)}_{\text{not necessarily low rank}},$$

which means slow SVD.

Hence, for matrix completion, acceleration and backtracking line search will be disadvantageous.

References

[CW88] Mazumder, Rahul, Trevor Hastie, and Robert Tibshirani. "Spectral regularization algorithms for learning large incomplete matrices." *Journal of machine learning research* 11.Aug (2010): 2287-2322.