10-725/36-725: Convex Optimization **Spring 2015**

Lecture 21: April 6

Lecturer: Ryan Tibshirani *Scribes: Chiqun Zhang, Hanqi Cheng, Waleed Ammar*

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

This lecture's notes illustrate some uses of various LaTeX macros. Take a look at this and imitate.

21.1 Last time

For a generalized lasso problems:

$$\min_{\beta} f(\beta) + \lambda \|D\beta\|_1$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a smooth, convex function and $D \in \mathbb{R}^{m \times n}$ is a penalty matrix. We have derived its dual problem, and compared all algorithms we have learned so far to both its primal and dual problem. Also, after comparison, we found that different algorithms has different strengths and we should choose suitable algorithm for different situations.

Moreover, we have learned conjugate functions, which is defined as

$$f^*(y) = \max_x y^T x - f(x)$$

where $f : \mathbb{R}^n \to \mathbb{R}$. Conjugate function is very important because it appears in dual programs frequently since $-f^*(y) = \min_x f(x) - y^T x$. And there are two main properties for conjugate function.

First, if f is closed and convex, then $f^{**} = f$. Also,

$$x \in \partial f^*(y) \iff y \in \partial f(x) \iff x \in \arg\min_z f(z) - y^T z$$

Second, if f is strictly convex, then

$$\nabla f^*(y) = \arg\min_z f(z) - y^T z$$

21.2 Dual (sub)gradient methods

21.2.1 Method statement

In the case we cannot derive dual or conjugate in closed form, we want to utilize dual relationship. It turns out that we can still use dual-based subgradient or gradient methods.

Considering the problem

$$\min_x f(x) \quad \text{subject to} \quad Ax = b$$

Its dual problem is

$$\max_u -f^*(-A^T u) - b^T u$$

where f^* is conjugate of f. Defining $g(u) = f^*(-A^T u)$, then the dual problem becomes

$$\max_u -g(u) - b^T u$$

The subgradient of above dual problem is $\partial g(u) - b$. After applying chain rule, we have that $\partial g(u) = -A\partial f^*(-A^T u)$. And recall the definition of conjugate function,

$$x \in \partial f^*(-A^T u) \Longleftrightarrow x \in \arg\min_z f(z) + u^T Az$$

Thus, the dual subgradient method for maximizing the dual objective is as following:

- Start from an initial dual guess $u^{(0)}$
- For each $k = 1, 2, 3, ...,$

$$x^{(k)} \in \arg\min_x f(x) + (u^{(k-1)})^T Ax$$
$$u^{(k)} = u^{(k-1)} + t_k(Ax^{(k-1)} - b)$$

  where t_k are step sizes, chosen in standard ways.
- Repeat step two until reaching convergence criteria

In special case where f is strictly convex, then f^* is differentiable and so we get dual gradient ascent method, whose update equations are

$$x^{(k)} = \arg\min_x f(x) + (u^{(k-1)})^T Ax$$
$$u^{(k)} = u^{(k-1)} + t_k(Ax^{(k-1)} - b)$$

The only difference between dual gradient ascent method and dual subgradient method is that in each step k, $x^{(k)}$ is unique. Also, we can apply proximal gradients and accelerations in similar manner.

21.2.2 Convergence analysis

First, recall

Theorem 21.1 *If f is strongly convex with parameter d, then ∇f^* is Lipschitz with parameter $\frac{1}{d}$*

Proof: If f is strongly convex and x is its minimized, then

$$f(y) \geq f(x) + \frac{d}{2}\|y - x\|_2 \quad \text{for all } y$$

Hence defining $x_u = \nabla f^*(u)$, $x_v = \nabla f^*(v)$,

$$f(x_v) - u^T x_v \geq f(x_u) - u^T x_u + \frac{d}{2}\|x_u - x_v\|_2^2$$
$$f(x_u) - v^T x_u \geq f(x_v) - v^T x_v + \frac{d}{2}\|x_u - x_v\|_2^2$$

Adding these together, using Cauchy-Schwartz and rearranging shows that

$$\|x_u - x_v\|_2 \leq \frac{1}{d}\|u - v\|_2$$

Then, we have shown that ∇f^* is Lipschitz with parameter $\frac{1}{d}$. ■

Then, with this theorem, applying what we know about gradient descent convergence rate, we have that : if f is strongly convex with parameter d, then the dual gradient ascent with constant step size $t_k \leq d$ converges at rate $O(1/\epsilon)$.

Recall that if we assume f is strongly convex as well as ∇f is Lipschitz, then the primal gradient descent converges at linear rate $O(log(1/\epsilon))$. Now we note that the converse statement of last theorem is also true: ∇f^* being Lipschitz with parameter $1/d$ implies that f is strongly convex with parameter d. And we assume that $f^{**} = f$. Then when f has Lipschitz gradient and its strongly convex, the same it is true about f^*. Thus, the dual gradient ascent also converges at linear rate $O(log(1/\epsilon))$.

Therefore, the convergence rates for primal gradient descent and dual gradient ascent are essentially the same given same condition on f.

21.3 Dual decomposition

21.3.1 Notion

One of the major advantages of the dual formulation above is dual decomposition. Consider the following optimization problem:

$$\min_x \quad \sum_{i=1}^{B} f_i(x_i) \quad \text{subject to} \quad Ax = b \tag{21.1}$$

Here the criterion is the sum of function f_i of variable x_i, subject to equality constraints. $x = (x_1, ...x_B) \in R^n$ denotes the block decomposition of n-dimensional variable x into B blocks, with $x_i \in R^{n_i}$ and $\sum_{i=1}^{B} n_i = n$. These blocks can be in different lengths. If we would not have the equality constraints, we can just minimize each of the B terms in the criterion independently, and solve the problem in a parallel fashion. But now we cannot do that because we have the equality constraints and the B blocks of variables are tied together in some way.

The powerful thing of the duals is that the duals decompose in these cases, even the primal doesn't. Here is how it works, partition A accordingly,

$$A = [A_1, ...A_B], \quad \text{where} \quad A_i \in R^{m \times n_i}$$

and we have $Ax = \sum_{i=1}^{B} A_i x_i$. The problem can be transformed into

$$\min_x \quad \sum_{i=1}^{B} f_i(x_i) \quad \text{subject to} \quad \sum_{i=1}^{B} A_i x_i = b \tag{21.2}$$

Assign

$$x^+ \in \arg\min_x \sum_{i=1}^B f_i(x_i) + u^T(\sum_{i=1}^B A_i x_i) \tag{21.3}$$

$$\iff x_i^+ \in \arg\min_{x_i} f_i(x_i) + u^T A_i x_i \quad \text{SEPARABLE over } i = 1, ...B \tag{21.4}$$

i.e., minimization *decomposes* into B separate problems.

21.3.2 Algorithm

Repeat for $k = 1, 2, 3, ...$

$$x_i^{(k)} \in \arg\min_{x_i} f_i(x_i) + (u^{(k-1)})^T A_i x_i, \quad i = 1, ...B$$

$$u^{(k)} = u^{(k-1)} + t_k(\sum_{i=1}^B A_i x_i^{(k-1)} - b)$$

The calculation of the (sub)gradient of the dual criterion can be separated over the B blocks and each minimization can be performed in parallel. Then once they are all done, we can just take the result of each one to form $\sum A_i x_i$ and get the gradient of the dual to take the dual update step.

Sometimes these steps are referred to as *Broadcast* and *Gather*, and here is a visualization of the process,

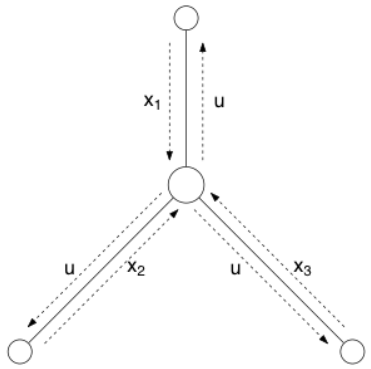

Figure 21.1: Visualization of broadcast and gather steps

Broadcast: At each step k, the processor in the center sends out the current value of the dual variable $u^{(k-1)}$ to each of the B parallel processors. In the figure above, $B = 3$. Each parallel processor i then solves the minimization problem in equation (1.4) to find $x_i^{(k)}$, independently.

Gather: $x_i^{(k-1)}$'s are sent back to the central processor to update the global dual variable $u^{(k)}$.

21.3.3 Example with inequality constraints

Here we just change the equality constraints in (1.2) into inequality constraints, as

$$\min_x \quad \sum_{i=1}^B f_i(x_i) \quad \text{subject to} \quad \sum_{i=1}^B A_i x_i \leq b \tag{21.5}$$

The update rules are as follows,

Repeat for $k = 1, 2, 3, ...$

$$x_i^{(k)} \in \arg\min_{x_i} f_i(x_i) + (u^{(k-1)})^T A_i x_i, \quad i = 1, ...B$$

$$v^{(k)} = u^{(k-1)} + t_k(\sum_{i=1}^{B} A_i x_i^{(k-1)} - b)$$

$$u^{(k)} = (v^{(k)})_+$$

where $(\cdot)_+$ is componentwise thresholding, $(u_+)_i = max\{0, u_i\}$.

Notes:

- We are using projected subgradient method instead of subgradient method. The reason is we have inequality constraints and the dual variable u is constrained by $u \geq 0$.
- The dual criterion function has exactly the same form as before so the calculation of the subgradient has no difference from the equality constrained questions (i.e., the first update rule).
- Taking the update of the dual variable may move it outside the constraint set. So we always need to project the updated dual back onto the constraint set (i.e., the second and the third update rules).

Price coordination interpretation (from Vandenberghe's lecture notes):

- Have B units in a system, each unit chooses its own decision variable x_i (how to allocate goods, or the numbers/units of different goods in system i)
- Constraints are limits on shared resources (each row j of A represents the consumptions of resource j per unit goods for the n different kinds of goods), each component of dual variable u_j is price of resource j
- Dual update:

$$u_j^+ = (u_j - ts_j)_+, \quad j = 1, ...m$$

  where $s_j = b_j - (\sum_{i=1}^{B} A_i x_i)_j$ are slacks

  - $s_j < 0$, which means $b_j < (\sum_{i=1}^{B} A_i x_i)_j$ and the inequality constraint is not satisfied, then resource j is over-utilized and we need to increase price u_j
  - $s_j > 0$, which means $b_j > (\sum_{i=1}^{B} A_i x_i)_j$ and the inequality constraint is satisfied, then resource j is under-utilized and we can decrease price u_j and spend more of resource j
  - $(\cdot)_+$ makes sure we never let prices get negative

21.4 Augmented Lagrangian

Motivation: one disadvantage of these dual methods is that we require strong conditions to ensure the primal iterates converge to solutions. Instead of giving details here, it is helpful to remind that we are iterating between the primal and dual iterates (section 1.1.2). What we have proved for most of the first order methods and a lot of other methods is we get a convergence of the criterion values. If we want the individual iterates in the dual methods converge to solutions, it requires more conditions, and these are usually much

more stringent. Just because of the nature that we are using the dual, we cannot get the primal converges without very strong conditions. Even under some conditions we do get the dual variable converges, the primal iterates can still not converge or even be infeasible, which means $Ax = b$ is not satisfied.

To improve the convergence properties, we introduce the Augmented Lagrangian method. Transform primal:

$$\min_x \quad f(x) + \frac{\rho}{2}\|Ax - b\|_2^2 \quad \text{subject to} \quad Ax = b \tag{21.6}$$

The added term $\frac{\rho}{2}\|Ax - b\|_2^2$ does not change the problem or the solution as long as the equality constraints are satisfied.

Repeat for $k = 1, 2, 3, ...$

$$x^{(k)} = \arg\min_x f(x) + (u^{(k-1)})^T Ax + \frac{\rho}{2}\|Ax - b\|_2^2$$

$$u^{(k)} = u^{(k-1)} + \rho(Ax^{(k-1)} - b)$$

(When, e.g., $A \in R^{m\times n}$ has full column rank, which means A has n non-zero singular values, the hessian of $\frac{\rho}{2}\|Ax - b\|_2^2$, $\rho A^T A$ satisfies $\rho A^T A \succeq sI$ for parameter $s > 0$, then the primal is guaranteed strongly convex.

Note:

- The step size choice is $t_k = \rho$ for all k. Why? Since $x^{(k)}$ minimizes $f(x) + (u^{(k-1)})^T Ax + \frac{\rho}{2}\|Ax - b\|_2^2$ over x, we have

$$0 \in \partial f(x^{(k)}) + A^T(u^{(k-1)} + \rho(Ax^{(k)} - b))$$

$$= \partial f(x^{(k)}) + A^T u^{(k)}$$

- $0 \in \partial f(x^{(k)}) + A^T u^{(k)}$ is the stationary condition for the original primal problem $\min_x f(x)$ subject to $Ax = b$
- Under mild conditions $Ax^{(k)} - b$ approaches zero (i.e., primal iterates approach feasibility). Hence in the limit KKT conditions are satisfied and $x^{(k)}$, $u^{(k)}$ approach optimality.

A brief summary:

- Augmented Lagrangian has much better convergence properties compared to the dual (sub)gradient method.
- However, it loses decomposability! The term $\frac{\rho}{2}\|Ax - b\|_2^2$ is not separable...

21.5 Alternating direction method of multipliers

The idea of ADMM is quite old. It has the good convergence properties of augmented Lagrangians, along with decomposability.

Consider the following minimization problem:

$$\min_x f_1(x_1) + f_2(x_2) \quad \text{subject to } A_1x_1 + A_2x_2 = b \tag{21.7}$$

As before, we augment the criterion function

$$\min_x f_1(x_1) + f_2(x_2) + \frac{\rho}{2}\|A_1x_1 + A_2x_2 - b\|_2^2$$
$$\text{subject to } A_1x_1 + A_2x_2 = b$$

Write the augmented Lagrangian as

$$L_p(x_1, x_2, u) = f_1(x_1) + f_2(x_2) + u^T(A_1x_1 + A_2x_2 - b) + \frac{\rho}{2}\|A_1x_1 + A_2x_2 - b\|_2^2$$

And here is how ADMM works:

Repeat the steps for $k = 1, 2, 3, ...$

$$x_1^{(k)} = \arg\min_{x_1} L_p(x_1, x_2^{(k-1)}, u^{(k-1)})$$

$$x_2^{(k)} = \arg\min_{x_2} L_p(x_1^{(k)}, x_2, u^{(k-1)})$$

$$x_2^{(k)} = \arg\min_{u} L_p(x_1^{(k)}, x_2^{(k)}, u)$$

Notes:

- ADMM updates variable blocks simultaneously. That is, after updating x_1 by minimizing the Lagrangian over x_1, we take the new $x_1^{(k)}$ value and plug in the Lagrangian to update x_2.
- This method does not necessarily decompose the $\frac{\rho}{2}\|A_1x_1 + A_2x_2 - b\|_2^2$ term over x_1 and x_2, it is an approximated way.
- It may hurt decomposability since we cannot do the updates in parallel, at least we can solve smaller number of sub-problems more efficiently instead of solving the full problem. Nevertheless, ADMM does not have the full decomposable power of the dual method. (Go to the references at the end for recent studies on the parallelization of ADMM)
- For comparison, the usual method of multipliers would have replaced the first two steps by

$$(x_1^{(k)}, x_2^{(k)}) = \arg\min_{x_1, x_2} L_p(x_1, x_2, u^{(k-1)})$$

21.5.1 Convergence guarantees

Any questions about ADMM before we go to examples? No. OK.

We're going to talk about convergence guarantees just for a bit first. And again, there is more information in the reference at the end. So, this is what we get under modest assumptions on f_1 and f_2. All they have to be is convex and closed. There is no condition on differentiability for sure. And A_1 and A_2 don't need to be full column rank. For any fixed value of ρ, we get that primal iterates approach feasibility so we get residual convergence. We also get objective convergence, which shouldn't be surprising because that comes out from our analysis in first order methods. We do not generically get primal convergence, but this can be shown under more assumptions. So, it doesn't have as strong convergence guarantees as augmented Laugrangian but it comes close. A very interesting quesiton is about the convergence rate. It is moving very quickly so I

could be out of date with the state of the art at this point, but as far as I can tell, it is still not known in general. There are special cases which people studied in detail but there is nothing like f is convex and has Lipschitz gradient. The consensus seems to be both in theory and in practice, that it behaves similar to first order methods. You would need to iterate a lot for a very accurate solution but it gives you a rough solution in a few iterations. So it is certainly more similar to first order methods than second order methods.

[unintelligible question]

So we are going to see towards the end an example. There is not one way to set up ADMM. You can do it in several ways so there's kind of an art to it in a sense.

[unintelligible questions]

21.5.2 Scaled form

Let me introduce a slightly different form for ADMM. I didn't do anything that is meaningful here except for replacing the dual variable u with a $w = \frac{u}{\rho}$:

$$x_1^{(k)} = \arg\min_{x_1} f_1(x_1) + \frac{\rho}{2}\|A_1 x_1 + A_2 x_2^{(k-1)} - b + w^{(k-1)}\|_2^2 \tag{21.8}$$

$$x_2^{(k)} = \arg\min_{x_2} f_2(x_2) + \frac{\rho}{2}\|A_1 x_1^{(k)} + A_2 x_2^{(k-1)} - b + w^{(k-1)}\|_2^2 \tag{21.9}$$

$$w^{(k)} = w^{(k-1)} + A_1 x_1^{(k)} + A_2 x_2^{(k)} - b \tag{21.10}$$

Note that there is no ρ in the dual update (last equation). You just take w which is the new parameterization for the dual variable and update it by adding the residual terms. And the minimization steps, you can check with simple calculus, reduce to the following: the function (e.g., f_1) and the squared L_2 norm of the constraints violations plus the dual variable. To show that this is equivalent to what we have before, just expand this and throw terms which don't depend on x_1 and x_2 and you will see that it's the same steps.

21.5.3 Practicalities and tricks

I want to save some time for examples but we are running out of time so I'll go through this quickly. Practically, you need many iterations of ADMM for an accurate solution. This is an empirical consensus. Choice of ρ can influence convergence of ADMM. So this is a good question asked earlier. Here's one strategy that people take in practice for how to choose ρ. But let me first give you the two ends of the spectrum: if ρ is too large, and there is not enough emphasis on minimizing the primal. So may be we don't get convergence to the primal solution as fast as we would like. If ρ is too small then we don't have enough emphasis on feasibility. So we don't get that the iterations approach feasibility fast enough. So there is a strategy that is more of a heuristic, that's given by a paper by Stephen Boyd. People use in practice, fairly often, but it doesn't come with convergence guarantees. If ρ is fixed, we are guaranteed to converge (eventually). When people use ADMM in practice, they tune ρ. You look at the primal residual and dual residual. If the primal residual is too big, you make ρ smaller. If the dual residual is too big, you make ρ larger, which ends up emphasizing feasibility more. Now that is not going to converge guaranteed to the solution so it is a kind of funny situation where I think more people use in practice but we don't know that it converges. One thing you could do to be careful is to use this strategy for 5000 iterations then fix it. If you do that so ρ will be

fixed eventually, so you get the convergence guarantees.

So there's a bit of trickery that goes into transforming problems that fit in the ADMM form. And like deriving duals, you can do it in a few different ways. I want to talk about sum-of-norms regularization as an example before we head out today. The problem we think about solving is a problem where we decompose our primal variable into different blocks and we are going to call that β_{I_g} and we are placing an L_2 norm on that block of variables. We could be doing other things than regression; I just put regression here because it is familiar:

$$min_{\beta \in R^p} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda \sum_{g=1}^{G} \|\beta_{I_g}\|_2 \tag{21.11}$$

We can use different norms, but if we use L_2 norm, we get sparsity in the groups of variables which is why we call it a group lasso problem.

How do we apply ADMM here? There is no constraints. We saw a similar situation when we wanted to derive a dual. Similarly, we introduce an auxiliary variable $\alpha = \beta$, and add the constraint that $\beta - \alpha = 0$:

$$min_{\beta,\alpha \in R^p} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda \sum_{g=1}^{G} \|\alpha_{I_g}\|_2 \text{ subject to } \beta - \alpha = 0 \tag{21.12}$$

The two blocks of variables now are α and β. The ADMM updates are not complicated at all:

$$\beta^{(k)} = (X^\top X + \rho I)^{-1}(X^\top y + \rho(\alpha^{(k-1)} - w^{(k-1)})) \tag{21.13}$$

$$\alpha_{I_g}^{(k)} = R_{\frac{\lambda}{\rho}}(\beta_{I_g}^{(k)} + w_{I_g}^{(k-1)}), g = 1, \ldots, G \tag{21.14}$$

$$w^{(k)} = w^{(k-1)} + \beta^{(k)} - \alpha^{(k)} \tag{21.15}$$

Each step is very simple and fairly efficient. Doesn't converge to accuracy quickly but if we want a rough solution it is certainly a good candidate. One thing to say is that a similar logic carries even if the groups are overlapping. If the groups are overlapping we can introduce a separate variable for each group of variables. Outside ADMM, optimizing group lasso is a very hard problem. I want to close by saying that you can think of ADMM as a very simple way to get an algorithm for a problem that may be complicated. So it is good for cases where you exhausted faster options and you want to use something that can do with simple updates which you can typically do if you parameterize ADMM properly.