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

Lecture 18: March 23

Lecturer: Ryan Tibshirani *Scribes: James Duyck*

These notes refer to the documents `algs-table.pdf` and `midterm-practice.pdf`, which were used as reference during the lecture.

18.1 Algorithm Review

We assume all criteria are convex.

18.1.1 Gradient Descent

To add constraints, take gradient step, then project onto constraint set.

Gradient descent is a special case of the subgradient method.

For each iteration, we compute the gradient, then add a multiple of the gradient to the current argument, so the iteration cost is cheap.

The gradient is required to be Lipschitz for a convergence rate of $O(\frac{1}{\epsilon})$. There are ways to get faster convergence. If we use acceleration, we can get a rate of $O(\frac{1}{\sqrt{\epsilon}})$. With strong convexity, the convergence rate is linear $O(\log(\frac{1}{\epsilon}))$. The constants in convergence rates depend on the conditioning, for example of the Hessian, so we could have linear convergence, but it might not perform well in practice due to a large constant.

Q: Did we discuss projection onto a constraint set?

A: Not generically, but for some special cases, for example in Lecture 8. Linear constraints, some easy cones, boxes, L1 and L2 balls are easy to project onto (there are closed forms or efficient algorithms). For a generic constraint set, the projection is a separate optimization problem.

18.1.2 Subgradient Method

This works for any criterion, with no restrictions.

Again, we handle constraints by taking a step along the subgradient, then projecting onto the constraint set.

We could choose step sizes to be fixed, but then we wouldn't necessarily have a convergent algorithm. A typical choice of step size is $O(\frac{1}{k})$ where k is the step number.

Gradient descent is quite a bit slower than gradient descent, but more general.

18.1.3 Proximal Gradient Descent

For the criterion $f = g + h$, g is smooth. h has to be simple so that we can evaluate the prox operator.

$$\begin{aligned}
& \min_{x \in C} g(x) + h(x) \qquad g \text{ smooth, } h \text{ nonsmooth} && (18.1)\\
\Leftrightarrow \quad & \min_{x} g(x) + h(x) + \mathbb{I}_C(x) \qquad \text{where } \mathbb{I} \text{ is the } 0-\infty \text{ indicator function} && (18.2)\\
= \quad & \min_{x} g(x) + \tilde{h}(x) && (18.3)\\
& \text{prox}_{\tilde{h},t}(x) && (18.4)\\
= \quad & \arg\min_{z} \frac{1}{2t}\|x-z\|_2^2 + \tilde{h}(z) && (18.5)\\
= \quad & \arg\min_{z \in C} \frac{1}{2t}\|x-z\|_2^2 + h(z) && (18.6)\\
x^+ = \quad & \text{prox}_{\tilde{h},t}(x - t\nabla g(x)) && (18.7)
\end{aligned}$$

The prox operator of h does not necessarily give the prox operator of $\tilde{h}$. For example if h is the L1 norm, the prox_h is soft thresholding. If the constraint is $z \geq 0$, we have an easy fix (soft thresholding with negative numbers set to 0). If C is some polyhedron, $\text{prox}_{\tilde{h}}$ might be much harder to evaluate. You might have to use iterative optimization in an inner loop.

Sometimes the prox operator decomposes or decomposes approximately, mentioned in recent NIPS papers.

We tend to use prox when the prox operator is moderately cheap (e.x. soft thresholding costs $O(n)$ for length n input). For expensive methods like Newton's method, we are willing to pay the price of expensive iterations because they have fast convergence rates, but the prox operator does not.

Q: For gradient and proximal gradient, do we have the same convergence rate?

A: Yes, but remember that for proximal gradient descent we have to evaluate both the gradient and the prox operator.

18.1.4 Newton's Method

Add equality constraints by solving a bigger linear system (called the KKT matrix).

We typically choose the step size to be 1 (pure Newton's method), but this may not converge. The safest way is to use backtracking line search.

For each iteration, we have to compute the Hessian and solve the system (possibly $O(n^3)$). This may be cheaper for a structured (ex. banded) Hessian.

The convergence rate $O(\log\log(\frac{1}{\epsilon})) \approx 5$ where $\epsilon = 2^{-32}$. Remember that the quadratic $O(\log\log(\frac{1}{\epsilon}))$ convergence rate is only local. The global convergence rate is linear $O(\log(\frac{1}{\epsilon}))$, but usually it reaches the local region quickly.

18.1.5 quasi-Newton

It doesn't solve a linear system in the Hessian. It approximates the Hessian with a sum of low rank matrices, so the iteration cost is lower than for Newton's method.

The convergence rate is super-linear: somewhere between linear and quadratic. n steps are equivalent to one step of Newton's method.

Point: Constants typically depend on the problem size n. This can make a big difference to the complexity.

18.1.6 Barrier Method

We assume that the inequality constraints are doubly smooth. We handle the equality constraints as in Newton's method.

We have to choose a step size for the inner loop (typically by backtracking line search). We have to choose an outer loop barrier parameter which multiplies the criterion (ex. 10).

The iteration is very expensive because we have to solve another optimization problem (Newton's method) on each iteration.

The convergence rate is linear ($O(\log \frac{1}{\epsilon})$) in the outer loop, and also in the number of Newton's steps (under enough conditions) - we can use a constant number of steps on each iteration to get a desired accuracy.

Q: The barrier method doesn't require doubly-smooth if we didn't use Newton's method on the inner loop?

A: Using something like gradient descent on the inner loop would require a huge number of iterations to get the accuracy required. Newton's method can get close to the central path in few steps.

18.1.7 Primal-dual IPM

Compared to the Barrier Method, there is no outer or inner loop. There are the same parameters as in the barrier method. The iteration is a single Newton step, but it can be quite large.

There's not one answer to what you should use!

18.2 Midterm Practice

18.2.1 Problem 6

Derive the dual of

$$\min_{x\in\mathbb{R}^n} \sum_{i=1}^N \|A_i x + b_i\|_1 + \frac{1}{2}\|x\|_2^2 \tag{18.8}$$

The dual for the problem without constraints is a constant equal to f^* the optimal criterion value, from the minimum of the Lagrangian, but that's not very useful. You make a substitution that doesn't change the problem but allows you to find a dual. If you have a matrix that's composed with a non-smooth function, that is difficult. We want a substitution that pulls out the matrix. There are multiple ways to derive different duals, but they are equivalent.

Let $A_i x = z_i$.

$$\min_{x\in\mathbb{R}^n} \sum_{i=1}^N \|z_i + b_i\|_1 + \frac{1}{2}\|x\|_2^2 \text{ subject to } A_i x = z_i \quad i = 1, ..., N \tag{18.9}$$

We associate one dual variable u_i with each of the constraints. Then we can write the Lagrangian

$$L(x, z_i, u_i) = \sum \| \|z_i + b_i\|_1 + \frac{1}{2}\|x\|_2^2 + \sum u_i^\top (z_i - A_i x) \tag{18.10}$$

We minimize the Lagrangian over the primal variables to find the dual function. Rearrange so that we separate terms that only involve z_i and terms that only involve x.

$$\min_{x,z} L(x, z_i, u_i) = \min_z \sum(\|z_i + b_i\|_1 + u_i^\top z_i) + \min_x \frac{1}{2}\|x\|_2^2 - (\sum A_i^\top u_1)^\top x \tag{18.11}$$

Minimize terms involving x.

$$\min_x \frac{1}{2}\|x\|_2^2 - (\sum A_i^\top u_1)^\top x = -\frac{1}{2}\| \sum A_i^\top u_i\|_2^2 \tag{18.12}$$

We want to minimize terms involving z_i. We know how to minimize $\|y_i\|_1$, so we substitute.

$$\begin{aligned}
&\min_{y_i = z_i + b_i} \|z_i + b_i\|_1 + u_i^\top z_i && (18.13)\\
&= \min_{y_i} \|y_i\|_1 + u_i^\top y_i - u_i^\top b_i && (18.14)\\
&= -\mathbb{I}\{\| - u_i\|_\infty \le 1\} - u_i^\top b_i && (18.15)\\
&= -\mathbb{I}\{\|u_i\|_\infty \le 1\} - u_i^\top b_i && (18.16)\\
& && (18.17)
\end{aligned}$$

The indicators can be written as constraints. Then the dual problem is

$$\begin{aligned}
\max_{u_i} \quad & -\sum u_i^\top b_i - \tfrac{1}{2}\| \sum A_i^\top u\|_2^2 && (18.18)\\
\text{subject to} \quad & \|u_i\|_\infty \le 1 \quad \forall i && (18.19)
\end{aligned}$$

18.2.2 Problem 1

Write down the dual of

$$\begin{aligned}
\min y_1, y_2 \quad & -4y_1 + 2y_2 && (18.20)\\
\text{subject to} \quad & -y_1 + y_2 \ge 2 && (18.21)\\
& y_1 - y_2 \ge 1 && (18.22)\\
& y_1, y_2 \ge 0 && (18.23)
\end{aligned}$$

First note the primal is infeasible. If we add the constraints, we get $0 \ge 3$.

If we minimize something over an empty set of constraints, the criterion is infinite. We want to write the dual and specify the duality gap.

We have strong duality (duality gap 0) if Slater's condition holds for the primal constraints or equivalently the dual constraints. For a linear program, this holds if either is feasible. The only way in which the duality gap is nonzero is if both the primal and the dual are infeasible. The duality can be infinity if the dual is also infeasible, and is 0 otherwise.

There are multiple ways to solve this. You can assign a multiplier to each constraint and add them to get the criterion. You can use the Lagrangian. You can think of the problem as a general form LP.

First find the Lagrangian

$$\begin{aligned} y_1 - y_2 &\leq -2 & (18.24)\\ -y_1 + y_2 &\leq -1 & (18.25)\\ -y_1 &\leq 0 & (18.26)\\ -y_2 &\leq 0 & (18.27)\\ L(y_1, y_2, u_1, ..., u_4) &= -4y_1 + 2y_2 + u_1(y_1 - y_2 + 2) + u_2(-y_1 + y_2 + 1) - u_3y_1 - u_4y_2 & (18.28)\\ &= y_1(-4 + u_1 - u_2 - u_3) + y_2(2 - u_1 + u_2 - u_4) + 2u_1 + u_2 & (18.29) \end{aligned}$$

The Lagrangian is linear in y_1 and y_2, so when we minimize over y_1 and y_2 to find the dual, the dual will be undefined ($-\infty$) if the coefficient on y_1 or y_2 is nonzero. Then the dual has the two constraints that the coefficients on y_1 and y_2 are zero. The dual is

$$\begin{aligned} \max_{u_1,...,u_4} \quad & 2u_1 + u_2 & (18.30)\\ \text{subject to} \quad & -4 + u_1 - u_2 - u_3 = 0 & (18.31)\\ & 2 - u_1 + u_2 - u_4 = 0 & (18.32)\\ & u_1, ..., u_4 \geq 0 & (18.33) \end{aligned}$$

Add the top two constraints together.

$$\begin{aligned} -2 - u_3 - u_4 &= 0 & (18.34)\\ u_3 + u_4 &= -2 & (18.35) \end{aligned}$$

So the dual is infeasible and the duality gap is infinite.