10-725: Optimization **Fall 2012**

Lecture 24: August 28

Lecturer: Geoff Gordon/Ryan Tibshirani *Scribes: Jiaji Zhou,Tinghui Zhou,Kawa Cheung*

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

24.1 Review

24.1.1 Analytic Center and Dikin Ellipsoid

The Analytic Center is the minimum of a log barrier function for a set of inequalities, which can be thought as the equilibrium point of forces from each inequality pushing away from it.
The newton's method can find analytic center quickly, and the newton step satisfies:

$$y = 1./(Ax + b)$$
$$A^T Y^2 A\Delta x = A^T y$$

The Dikin Ellipsoid at a certain feasible point x_0 is defined as the unit ball of Hessian norm for log barrier function evaluated at x_0. It is always contained in the feasible region. In particular, if we scale the Dikin Ellipsoid at the analytic center by a factor of m (the number of rows in A), it will contain the whole feasible region.

24.1.2 Central Path and Affine Invariance

If we combine both the objective (optimality) term and the log barrier (centering) term (with a parameter t), then the central path would be the solution path with respect to t.

The central path problem can be represented as either the penalty form or the constraint form (optimality term as a constraint to be smaller than λ).

The dual problem of the central path problem will give us another central path, and there is a one to one mapping between each other, with duality gap m/t.

Note that both the Analytic Center, Central Path and Dikin Ellpsoid are all affine invariant, which means if we perform a linear transform on x, they will also be linear transformed accordingly.

24.2 Primal-dual Constraint Form

Let's look at the LP primal-dual pair:

$$primal: \quad \min c^T x \quad st \quad Ax + b \geq 0$$
$$dual: \quad \max -b^T y \quad st \quad A^T y = c \quad y \geq 0$$

And the KKT condition is

$$\begin{aligned} Ax + b \geq 0 \quad & \text{primal feasibility} \\ y \geq 0 \quad A^T y = c \quad & \text{dual feasibilty} \\ c^T x + b^T y \leq 0 \quad & \text{strong duality} \end{aligned}$$

if we relax strong duality by enlarging the gap between primal and dual, we have the relaxed strong duality

$$c^T x + b^T y \leq \lambda \quad \text{relaxed strong duality}$$

Now we think of the KKT condition as linear feasibility problem. Note that the original problem does not have strictly feasible point and the relaxed one has. Therefore we can run Newton's method to find the Analytic Center and get the solution the linear program.

24.3 Central Path for Relaxed KKT

We now formulate the central path for this relaxed LP KKT problem

$$\begin{aligned} Ax + b = s \geq 0 &\Rightarrow \min -\sum \ln s_i \\ y \geq 0 &\Rightarrow \min -\sum \ln y_i \\ A^T y &= c \\ c^T x + b^T y \leq \lambda &\Rightarrow \min -\ln(\lambda - c^T x - b^T y) \geq t(c^T x + b^T y - \lambda) + 1 + \ln t \end{aligned}$$

For the last inequality, we used the fact that $-\ln z \geq -tz + 1 + \ln t$. Now if we group $tc^T x$ and $-\sum \ln s_i$, we get the primal central path problem, and if we group $tb^T y$ and $-\sum \ln y_i$, we get the dual central path problem. The mapping between λ and t is $\lambda = \frac{m+1}{t}$, one can get this by differentiate $t(c^T x + b^T y - \lambda) + 1 + \ln t$ with respect to t and use the fact that for the primal and dual pair (x, y) on the central path, the duality gap is $c^T x - (-b^T)y = \frac{m}{t}$.

We can now run Newton's method for dual central path problem and form a simple algorithm as below:

- $t := 1$, $y := 1^m$, $x := 0^n$
- Repeat:
  - Use infeasible-start Newton to find point y on dual central path (and corresponding mulipliers x)
  - $t := \alpha t (\alpha > 1)$
- After any outer iteration:
  - Multipliers x are primal feasible; gap $c^T x + b^T y = \frac{m}{t}$
  - $s = I./ty$, $x = A \backslash (s - b)$

Figure **??** shows an example of running this algorithm.

We now split the KKT condition into feasible conditions and optimality conditions.

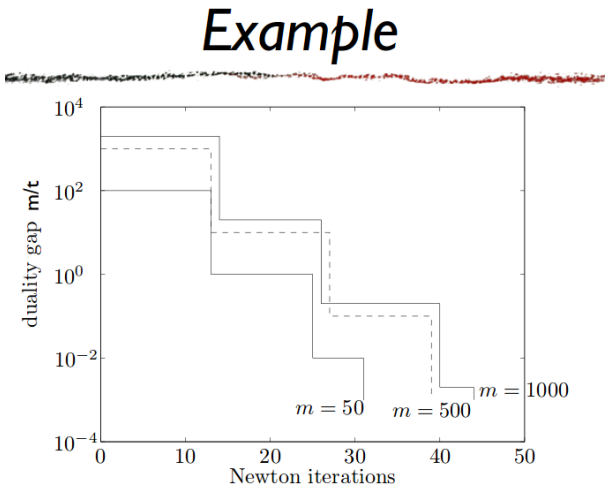

Figure 24.1: Newton

- Feasible conditions:
  - $Ax + b = s \geq 0$
  - $y \geq 0$
  - $A^T y = c$
- Optimality conditions:
  - $c^T x + b^T y \leq 0 \Leftrightarrow s^T y \leq 0$

, where we use the fact that $s^T y = (Ax + b)^T y = c^T x + b^T y$. Next, we will formulate an optimization objective that combines both feasibility and optimality.

24.4 Potential Reduction

We can define a potential combining feasibility and optimality for the KKT conditions as follows:

$$\begin{aligned} p(s, y) &= (m + k) \ln y^T s - \Sigma \ln y_i - \Sigma \ln s_i \\ &= k \ln y^T s + [m \ln y^T s - \Sigma \ln y_i - \Sigma \ln s_i] \\ &\geq k \ln y^T s \ . \end{aligned}$$

Strategy of the potential reduction algorithm:

- Start with a strictly feasible (x, y, s).
- Update by $(\Delta x, \Delta y, \Delta s)$: to maintain strict feasibility, we have $A^T \Delta y = 0$, and $\Delta s = A\Delta x + b$.

- Reduce $p(s, y)$ by at least δ per iteration. Therefore,

$$\begin{aligned} & p \leq p_0 - T\delta \\ \Longleftrightarrow & k \ln y^T s \leq p_0 - T\delta \\ \Longleftrightarrow & y^T s \leq exp(\frac{p_0 - t\delta}{k}), \end{aligned}$$

  where T is the number of iterations, and p_0 is the initial potential.

We upper bound $p(s, y)$ locally with a quadratic, which will look like Hessian from Newton's method. Let $p(s, y) = p_1(s, y) + p_2(s, y)$, where $p_1(s, y) = (m + k) \ln y^T s$, and $p_2(s, y) = -\Sigma \ln y_i - \Sigma \ln s_i$. Then we have

$$\begin{aligned} p_1(s + \Delta s, y + \Delta y) &\leq (m + k)[\ln y^T s + \frac{1}{y^T s}\left((y + \Delta y)^T (s + \Delta s) - y^T s\right)] \\ &= (m + k)[\ln y^T s + \frac{1}{y^T s}(\Delta y^T s + \Delta s^T y)] \\ &= \bar{p_1} \end{aligned}$$

and

$$\begin{aligned} p_2(s + \Delta s, y + \Delta y) &\leq -\Sigma \ln y_i - \Sigma \ln s_i + \frac{\tau}{2}\Delta y^T Y^{-2} \Delta y + \frac{\tau}{2}\Delta s^T S^{-2} \Delta s \\ &= \bar{p}_2, \end{aligned}$$

where $\tau > 1$ is a constant, and $\frac{dp_2}{dy} = -\frac{1}{y}$, $\frac{dp_2}{ds} = -\frac{1}{s}$, $\frac{d^2 p_2}{dy^2} = Y^{-2}$, and $\frac{d^2 p_2}{ds^2} = S^{-2}$.

The algorithm then chooses $(\Delta x, \Delta y, \Delta s)$ to minimize $\bar{p}_1 + \bar{p}_2$ such that $A^T \Delta y = 0$, $\Delta s = A\Delta x$, and $\Delta y^T Y^{-2} \Delta y + \Delta s^T S^{-2} \Delta s \leq (2/3)^2$ (in the case of $\tau = 2$). The claim is that if the algorithm steps along $(\Delta x, \Delta y, \Delta s)$ while keeping $y > 0, s > 0$, then the potential always decreases by $\delta = 1/4$ per iteration.

To get some intuition, suppose $s = y = \mathbf{1}^m$. Then

$$\begin{aligned} \bar{p_1} &= p_1 + [(m + k)/m][\mathbf{1}^T \Delta y + \mathbf{1}^T \Delta s], \\ \bar{p_2} &= p_2 - \mathbf{1}^T \Delta y - \mathbf{1}^T \Delta s + \Delta y^T \Delta y + \Delta s^T \Delta s, \\ \Delta p &\leq \frac{k}{m} I^T \Delta y + \frac{k}{m} \mathbf{1}^T \Delta s + \Delta y^T \Delta y + \Delta s^T \Delta s. \end{aligned}$$

The feasible set in this case is $A^T \Delta y = 0, \Delta s = A\Delta x$. Furthermore, consider projecting the gradient g onto the two subspaces $range(S \setminus A)$ and $null(A^T Y)$ as α and β, respectively. Then we have $\| \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \| = \|g\| = \frac{k}{\sqrt{m}}$, which determines how much decrease in potential is possible.

Without further proof, it is known that the norm of the projection of the gradient vector onto the feasible set is lower bounded by $k/\sqrt{m}$.

24.5 Step Size

So far, we have proved for a very simple case and inferred for more complicated cases that the norm of the projection of the gradient vector onto the set of feasible step directions is at least $\frac{k}{\sqrt{m}}$. Then the step size

problem is equivalent to solving:

$$\min \quad t^2 - \frac{k}{\sqrt{m}}t,$$

where the quadratic part has a hessian equals to twice the identity.

Consider the case where $\frac{k}{\sqrt{m}} = 1$:

$$\begin{aligned} t^2 - t &= t^2 - 2(\frac{1}{2})(t) + (\frac{1}{2})^2 - (\frac{1}{2})^2 \\ &= (t - \frac{1}{2})^2 - \frac{1}{4} \end{aligned}$$

where the minimum of this function is $-\frac{1}{4}$ and is achieved when $t = \frac{1}{2}$.

Since we can solve this case nice and easily, we can go ahead and pick $k = \sqrt{m}$ and guarantee lower bounds on the step size and the decrease of the quadratic bound by $\frac{1}{2}$ and $-\frac{1}{4}$ respectively after every iteration. On another note, recall the ellipsoid constraint before, which essentially restricted every step to less than $\frac{2}{3}$. As shown above, if the norm of the projection of the gradient achieves its lower bound at $\frac{k}{\sqrt{m}}$, the step size will be $\frac{1}{2}$, which is less than $\frac{2}{3}$ and hence feasible. But if the norm is greater than $\frac{k}{\sqrt{m}}$, then the step size might violate the ellipsoid constraint, in which case, we can just scale it back until the constraint is fulfilled and still guarantee our decrease of $-\frac{1}{4}$. The choice of the ellipsoid constraint over the box constraint facilitates the scaling process.

24.6 Algorithm Summary

1. Start with parameters $k = \sqrt{m} > 0, \tau = 2 > 1$ and a feasible initial point (x_0, y_0, s_0).
2. Repeat
   (a) Find step direction $(\Delta x, \Delta y, \Delta z)$ by solving:

$$\begin{aligned} min \quad & (\frac{(m+k)s}{y^T s} - \frac{1}{y})^T \Delta y + (\frac{(m+k)y}{y^T s} - \frac{1}{s})^T \Delta s + \frac{\tau}{2}\Delta y^T Y^{-2} \Delta y + \frac{\tau}{2}\Delta s^T S^{-2} \Delta s \\ s.t. \quad & \Delta s = A\Delta x \\ & A^T \Delta y = 0 \\ & \frac{\tau}{2}\Delta y^T Y^{-2} \Delta y + \frac{\tau}{2}\Delta s^T S^{-2} \Delta s \leq f(\tau) \end{aligned}$$

   (b) Line Search: Find optimal step length with s>0, y>0.
   (c) Update: Calculate new (x,y,s) with our step direction and step length.
3. Stop when $y^T s$ is small enough.

Note:

- The objective function in 2(a) is the quadratic upper bound on the potential function and is tangent to it at our current point.

- The step direction will look like Newton's method.
- Since the ellipsoid constraint shares the same hessian as the quadratic terms in the objective function of 2(a), this allows us to first solve for the minimization problem without considering the constraint and then project the step back onto the ellipsoid if it violates the constraint.

24.7 Examples

24.7.1 Example 1

- Infeasible initializer
- $k = \sqrt{m}$
- $\tau = 2$
- $A \in R^{7\times 2}$

Figure **??** shows an example of running this algorithm with the parameters specified above.

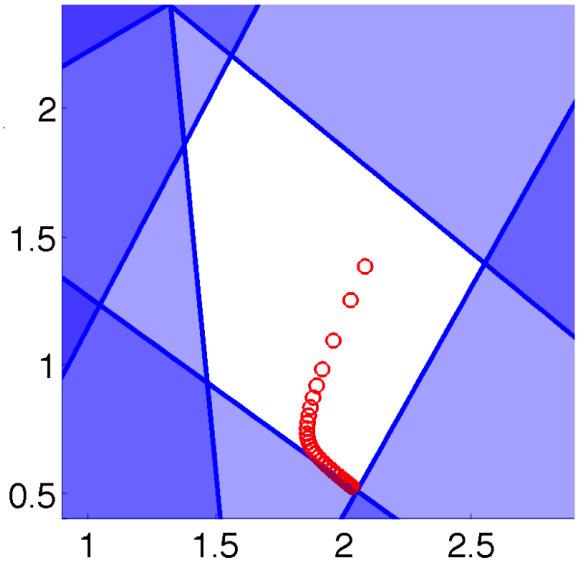

Figure 24.2: Plot of central path

In this example, we started with an infeasbile initializer that wasn't guaranteed to work but the results of the algorithm showed that it worked just fine. The parameters for k and τ were set to their theoretically valid values, which turned out to be quite conservative for a matrix with 7 inequality constraints and 2 real variables. The trail of red circles shows the central path, which we find on the first iteration and follow it down nice and smoothly to the optimal point.

Figure **??** shows some diagnostics for the example above.

```
  1: step 1.0000, mean gap 10^ -0.4057, pot  17.2774
  2: step 1.0000, mean gap 10^ -0.5184, pot  15.9290
  3: step 1.0000, mean gap 10^ -0.6024, pot  15.1694
  4: step 1.0000, mean gap 10^ -0.6908, pot  14.5695
…
 11: step 1.0000, mean gap 10^ -1.3256, pot  10.6940
 12: step 1.0000, mean gap 10^ -1.4165, pot  10.1402
 13: step 1.0000, mean gap 10^ -1.5074, pot   9.5864
 14: step 1.0000, mean gap 10^ -1.5983, pot   9.0327
…
 21: step 1.0000, mean gap 10^ -2.2340, pot   5.1602
 22: step 1.0000, mean gap 10^ -2.3248, pot   4.6069
 23: step 1.0000, mean gap 10^ -2.4157, pot   4.0534
 24: step 1.0000, mean gap 10^ -2.5066, pot   3.4997
…
 30: step 1.0000, mean gap 10^ -3.0522, pot   0.1756
```

Figure 24.3: Diagnostics

Looking at some diagnostics, we are always taking a step length of one with these conservative parameters. The calculated potentials after every iteration show a decrease of 10 from steps 11 to 30, which is equivalent to a decrease of 0.5 after each iteration instead of 0.25 as predicted by the proof. So even though the algorithm outperformed the proof, the proof still did a good job of analysising the behaviour of the algorithm. The gap starts out at single digits and goes down to the order of 10^{-3} after 30 iterations, which means the things that are suppose to be zero stay around 10^{-3} and are hard to distinguish from one another in the plot above.

24.7.2 Example 2

- Same initializer
- $k = 0.999m$
- $\tau = 1.95$
- $A \in R^{7\times 2}$

Figure **??** shows an example of running this algorithm with the parameters specified above.

Rerunning the algorithm with slightly less conservtive parameters, we now only see 7 to 8 distinct points.

Figure **??** shows some diagnostics for the example above.

Looking at the diagnostics, we can see from the gap that the algorithm achieves the same solution quality as the previous example in just 7 iterations. After 30 iterations, the algorithm achieves around machine-precision for the gap at about 10^{-13}. For comparison, a 'double' has precision of about 10^{-12}. Further iterations result in the algorithm appearing to be achieve a better value, when in fact it is only making use of the roundoff error to its favor.

24.8 When is IP useful?

Interior point algorithms are based on Newton's method. Newton's method, at least naively, is cubic in the smaller of the number variables and the number of constraints. So unless we can take advantage of some structure in the problem, we're limited to only a thousand or a few thousand variables. For a 1000×1000

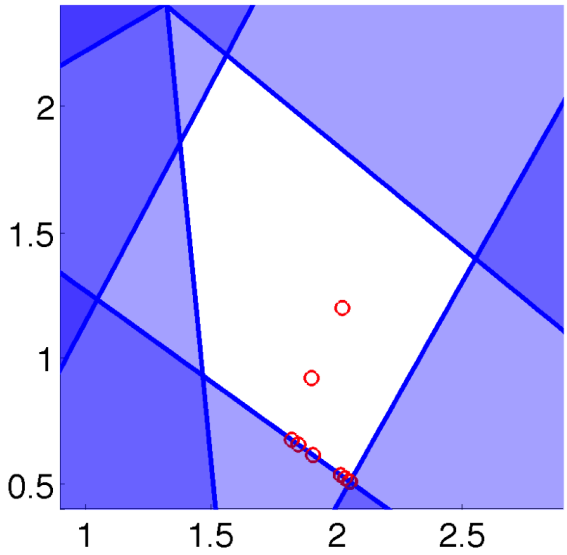

Figure 24.4: Plot of central path

```
  1: step 1.0000, mean gap 10^ -0.6266, pot  18.1732
  2: step 0.9109, mean gap 10^ -0.9666, pot  13.4386
  3: step 0.9997, mean gap 10^ -1.4694, pot  10.6936
  4: step 0.7258, mean gap 10^ -1.9010, pot   2.4038
  5: step 0.6761, mean gap 10^ -2.2711, pot  -4.7473
  6: step 0.9258, mean gap 10^ -2.8463, pot -14.3558
  7: step 0.6785, mean gap 10^ -3.3540, pot -24.9006
…
 17: step 0.9767, mean gap 10^ -8.1569, pot -98.7712
…
 30: step 1.0000, mean gap 10^-13.7609, pot -193.9617
```

Figure 24.5: Diagnostics

matrix, it takes a billion floating operations to invert and goes up cubically from there. Another problem is memory for storage. However, structure is often present and we are able to accomodate millions of variables and quickly solve the Newton system. In this sense, quickly could mean that we know of a particular sparsity structure in our constraint matrix, or a low rank structure, or even both, and we are able to invert that system and solve it faster than the naive cubic.

The other thing to point out is that the convergence rate of the IP method is on a totally different level than the first-order methods we discussed before. Even with acceleration, the best we could have hoped for was $\frac{1}{\sqrt{\epsilon}}$ iterations to achieve an accuracy of ϵ, whereas here we are able to achieve $ln(\frac{1}{\epsilon})$. Figure **??** shows a comparisons between convergence rates. Here we note that for an accuracy of one part in a million, we require close to a thousand iterations for first-order methods, compared to the mere tens needed for the IP method. Futhermore, for the accelerated method to achieve the rate above, smooth functions are required. In contrast, the IP method has no need for such requirements since it is a linear program and can deal with sharp boundaries that are non-differentiable. The main trade-off in using IP methods over first-order methods is the cost per iteration. Hence for a particular level of accuracy, the tradeoff is between the costs of computing the more expensive Newton steps and the costs of incurring thousands of iterations more using the cheaper first-order steps. So in particular, for a moderate sized problem that has tight accuracy require-

ments or fairly bad conditioning, the IP method is going to dominate. In contrast, for a situation requiring only mediocre accuracy, but has a million variables with no special structure in our constraint matrix, we are stuck with first-order methods.

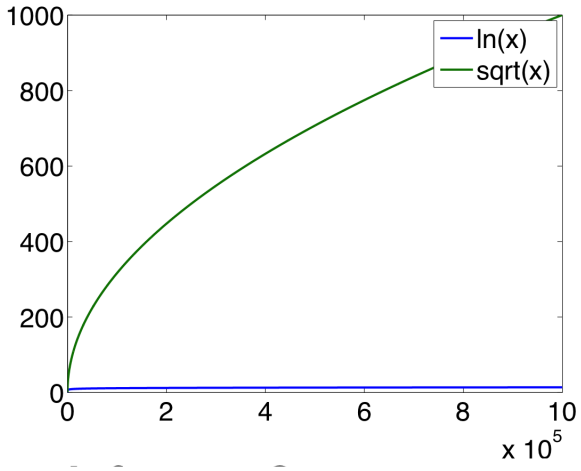

Figure 24.6: Convergence rates

The last thing to point is that even for large scale cases where we can't afford to do Newton's method, intuition from duality can help with algorithm design. So it's nice to know about tools like Dikin ellipsoids and Central Paths because they let us know a little bit more about the geometry of what the feasible region looks like.