Convex Optimization for Statistics and Machine Learning

Part IV: Advanced Methods

Ryan Tibshirani

Depts. of Statistics & Machine Learning
Carnegie Mellon University

`http://www.stat.cmu.edu/~ryantibs/talks/cuso-part4-2019.pdf`

Outline for Part IV

- Part A. Newton's method
- Part B. Interior-point methods
- Part C. Coordinate descent
- Part D. ADMM

Part IV: Advanced methods

A. Newton's method

Newton's method

Given unconstrained, smooth convex optimization

$$\min_x \; f(x)$$

where f is convex, twice differentable, and $\mathrm{dom}(f) = \mathbb{R}^n$. Recall that gradient descent chooses initial $x^{(0)} \in \mathbb{R}^n$, and repeats

$$x^{(k)} = x^{(k-1)} - t_k \cdot \nabla f(x^{(k-1)}), \quad k = 1, 2, 3, \ldots$$

In comparison, Newton's method repeats

$$x^{(k)} = x^{(k-1)} - \left(\nabla^2 f(x^{(k-1)})\right)^{-1} \nabla f(x^{(k-1)}), \quad k = 1, 2, 3, \ldots$$

Here $\nabla^2 f(x^{(k-1)})$ is the Hessian matrix of f at $x^{(k-1)}$

Newton's method interpretation

Recall the motivation for gradient descent step at x: we minimize the quadratic approximation

$$f(y) \approx f(x) + \nabla f(x)^T (y - x) + \frac{1}{2t}\|y - x\|_2^2$$

over y, and this yields the update $x^+ = x - t\nabla f(x)$

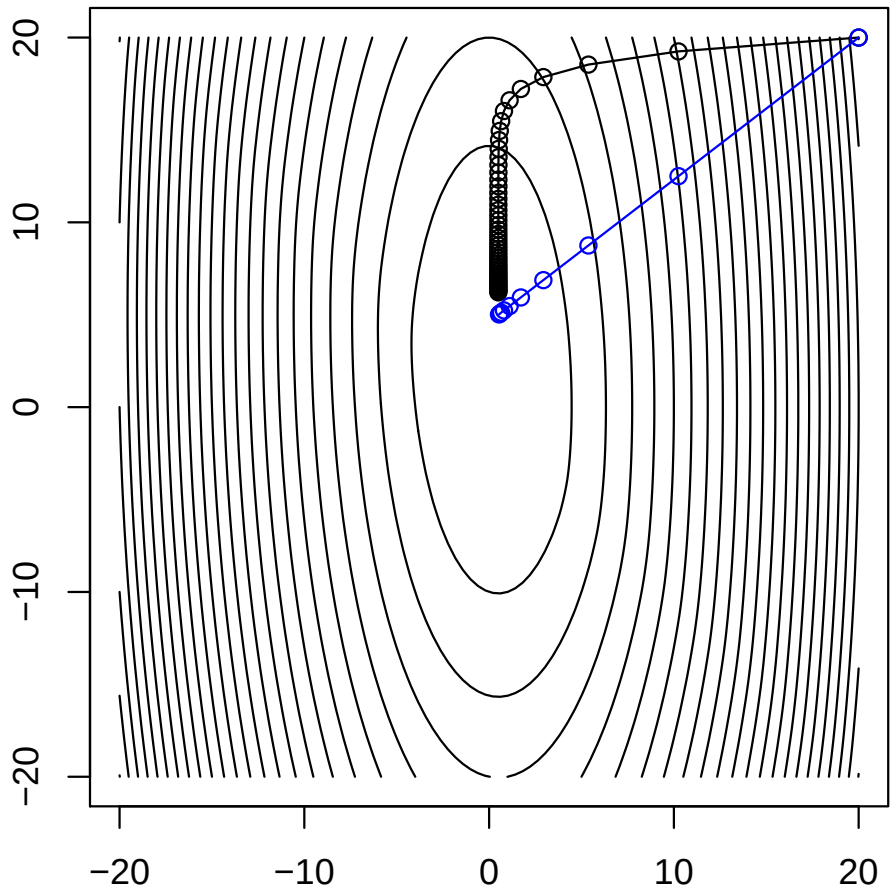
Newton's method uses in a sense a better quadratic approximation

$$f(y) \approx f(x) + \nabla f(x)^T (y - x) + \frac{1}{2}(y - x)^T \nabla^2 f(x)(y - x)$$

and minimizes over y to yield $x^+ = x - (\nabla^2 f(x))^{-1} \nabla f(x)$

Consider minimizing $f(x) = (10x_1^2 + x_2^2)/2 + 5\log(1 + e^{-x_1 - x_2})$
(this must be a nonquadratic ... why?)

We compare gradient descent (black) to Newton's method (blue), where both take steps of roughly same length

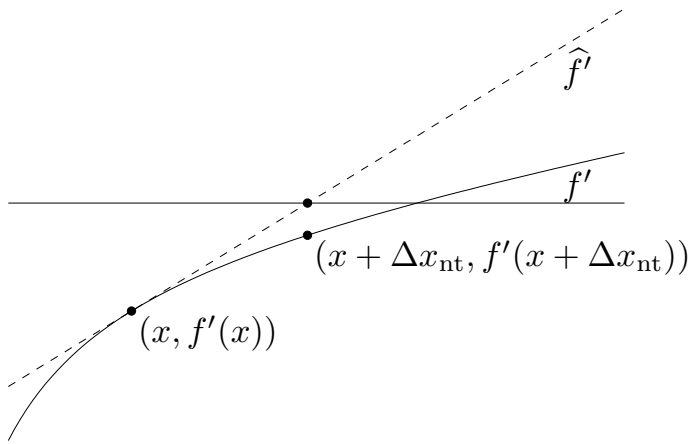


Linearized optimality condition

Aternative interpretation of Newton step at x: we seek a direction v so that $\nabla f(x+v) = 0$. Let $F(x) = \nabla f(x)$. Consider linearizing F around x, via approximation $F(y) \approx F(x) + DF(x)(y-x)$, i.e.,

$$0 = \nabla f(x+v) \approx \nabla f(x) + \nabla^2 f(x) v$$

Solving for v yields $v = -(\nabla^2 f(x))^{-1} \nabla f(x)$



(From B & V page 486)

History: work of Newton (1685) and Raphson (1690) originally focused on finding roots of polynomials. Simpson (1740) applied this idea to general nonlinear equations, and minimization by setting the gradient to zero

Affine invariance of Newton's method

Important property Newton's method: affine invariance. Given f, nonsingular $A \in \mathbb{R}^{n \times n}$. Let $x = Ay$, and $g(y) = f(Ay)$. Newton steps on g are

$$\begin{aligned} y^+ &= y - \big(\nabla^2 g(y)\big)^{-1} \nabla g(y) \\ &= y - \big(A^T \nabla^2 f(Ay) A\big)^{-1} A^T \nabla f(Ay) \\ &= y - A^{-1} \big(\nabla^2 f(Ay)\big)^{-1} \nabla f(Ay) \end{aligned}$$

Hence

$$Ay^+ = Ay - \big(\nabla^2 f(Ay)\big)^{-1} \nabla f(Ay)$$

i.e.,

$$x^+ = x - \big(\nabla^2 f(x)\big)^{-1} f(x)$$

So progress is independent of problem scaling. This is not true of gradient descent!

Backtracking line search

So far we've seen pure Newton's method. This need not converge. In practice, we use damped Newton's method (typically just called Newton's method), which repeats

$$x^+ = x - t\big(\nabla^2 f(x)\big)^{-1} \nabla f(x)$$

Note that the pure method uses $t = 1$

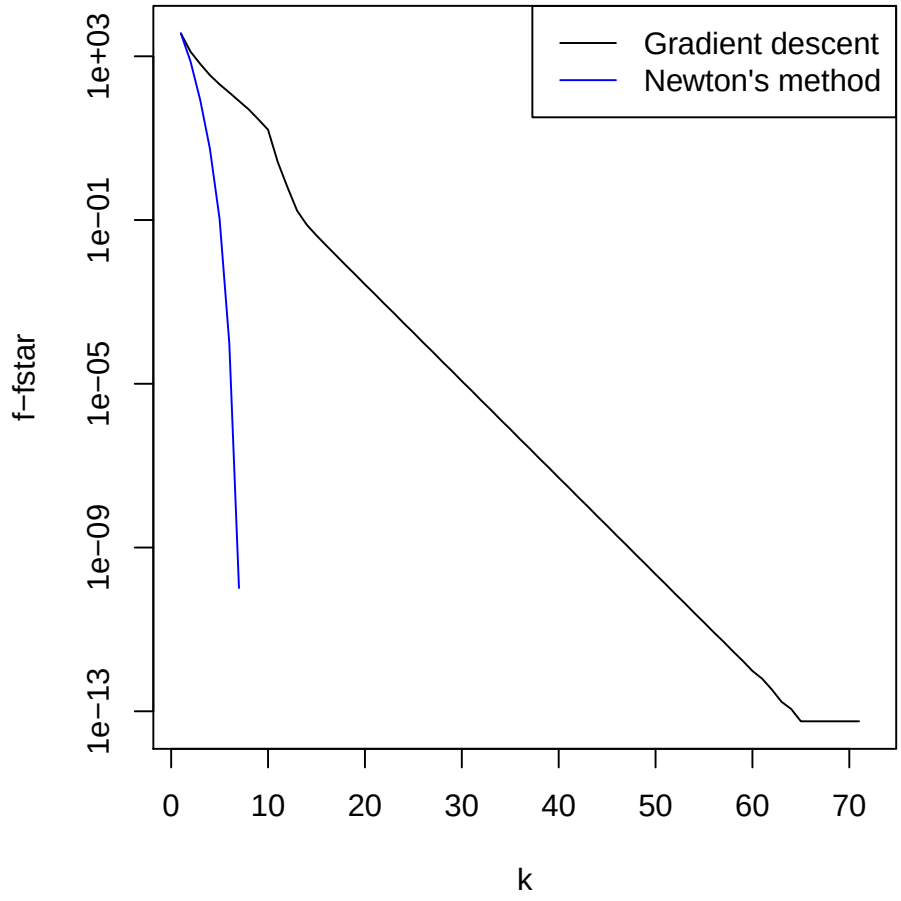
Step sizes here are chosen by backtracking search, with parameters $0 < \alpha \leq 1/2$, $0 < \beta < 1$. At each iteration, start with $t = 1$, while

$$f(x + tv) > f(x) + \alpha t \nabla f(x)^T v$$

we shrink $t = \beta t$, else we perform the Newton update

Example: logistic regression

Logistic regression example, with $n = 500$, $p = 100$: we compare gradient descent and Newton's method, both with backtracking



Newton's method: in a totally different regime of convergence...!

Iteratively reweighted least squares

Given $y \in \mathcal{Y}^n$, $X \in \mathbb{R}^{n \times p}$, and a convex function b, consider fitting a generalized linear model:

$$\min_{\beta} \; -y^T X\beta + b(X\beta)$$

where b is applied componentwise. Examples include:

- Linear regression: $b(u) = u^2/2$
- Logistic regression: $b(u) = \log(1 + e^u)$
- Poisson regression: $b(u) = e^u$

Recall b the is cumulant generating function, hence

$$\mu = b'(X\beta) \quad \text{and} \quad V = \mathrm{diag}(b''(X\beta))$$

are the mean vector and diagonal matrix of variances

Gradient calculation:

$$\nabla f(\beta) = X^T \big(y - \underbrace{b'(X\beta)}_{\mu}\big)$$

Hessian calculation:

$$\nabla^2 f(\beta) = X^T \underbrace{\operatorname{diag}(b''(X\beta))}_{V} X$$

Note the similarities to the linear model case! Moreover, Newton's method becomes iteratively reweighted least squares:

$$\beta^+ = \beta - \big(\nabla^2 f(\beta)\big)^{-1} \nabla f(\beta) \iff \beta^+ = (X^T V X)^{-1} X^T V z$$

where $z = X\beta - V^{-1}(y - \mu)$. (These iterations are also central to classical statistical inference in GLMs)

Convergence analysis

Assume that f convex, twice differentiable, having $\mathrm{dom}(f) = \mathbb{R}^n$, and additionally

- ∇f is Lipschitz with parameter L
- f is strongly convex with parameter m
- $\nabla^2 f$ is Lipschitz with parameter M

Theorem: Newton's method with backtracking line search satisfies the following two-stage convergence bounds

$$f(x^{(k)}) - f^\star \leq \begin{cases} (f(x^{(0)}) - f^\star) - \gamma k & \text{if } k \leq k_0 \\ \dfrac{2m^3}{M^2}\left(\dfrac{1}{2}\right)^{2^{k-k_0+1}} & \text{if } k > k_0 \end{cases}$$

Here $\gamma = \alpha\beta^2\eta^2 m/L^2$, $\eta = \min\{1, 3(1-2\alpha)\}m^2/M$, and k_0 is the number of steps until $\|\nabla f(x^{(k_0+1)})\|_2 < \eta$

In more detail, convergence analysis reveals $\gamma > 0$, $0 < \eta \leq m^2/M$ such that convergence follows two stages

- Damped phase: $\|\nabla f(x^{(k)})\|_2 \geq \eta$, and

$$f(x^{(k+1)}) - f(x^{(k)}) \leq -\gamma$$

- Pure phase: $\|\nabla f(x^{(k)})\|_2 < \eta$, backtracking selects $t = 1$, and

$$\frac{M}{2m^2}\|\nabla f(x^{(k+1)})\|_2 \leq \left(\frac{M}{2m^2}\|\nabla f(x^{(k)})\|_2\right)^2$$

Note that once we enter pure phase, we won't leave, because

$$\frac{2m^2}{M}\left(\frac{M}{2m^2}\eta\right)^2 \leq \eta$$

when $\eta \leq m^2/M$

Unraveling this result, what does it say? To get $f(x^{(k)}) - f^\star \leq \epsilon$, we need at most

$$\frac{f(x^{(0)}) - f^\star}{\gamma} + \log\log(\epsilon_0/\epsilon)$$

iterations, where $\epsilon_0 = 2m^3/M^2$

- This is called quadratic convergence. Compare this to linear convergence (which, recall, is what gradient descent achieves under strong convexity)
- The above result is a local convergence rate, i.e., we are only guaranteed quadratic convergence after some number of steps k_0, where $k_0 \leq \frac{f(x^{(0)}) - f^\star}{\gamma}$
- Somewhat bothersome may be the fact that the above bound depends on L, m, M, and yet the algorithm itself does not ...

Self-concordance

A scale-free analysis is possible for self-concordant functions: on $\mathbb{R}$, a convex function f is called self-concordant if

$$|f'''(x)| \leq 2f''(x)^{3/2} \quad \text{for all } x$$

and on $\mathbb{R}^n$ is called self-concordant if its projection onto every line segment is so

Theorem (Nesterov and Nemirovskii): Newton's method with backtracking line search requires at most

$$C(\alpha, \beta)\big(f(x^{(0)}) - f^\star\big) + \log\log(1/\epsilon)$$

iterations to reach $f(x^{(k)}) - f^\star \leq \epsilon$, where $C(\alpha, \beta)$ is a constant that only depends on α, β

What kind of functions are self-concordant?

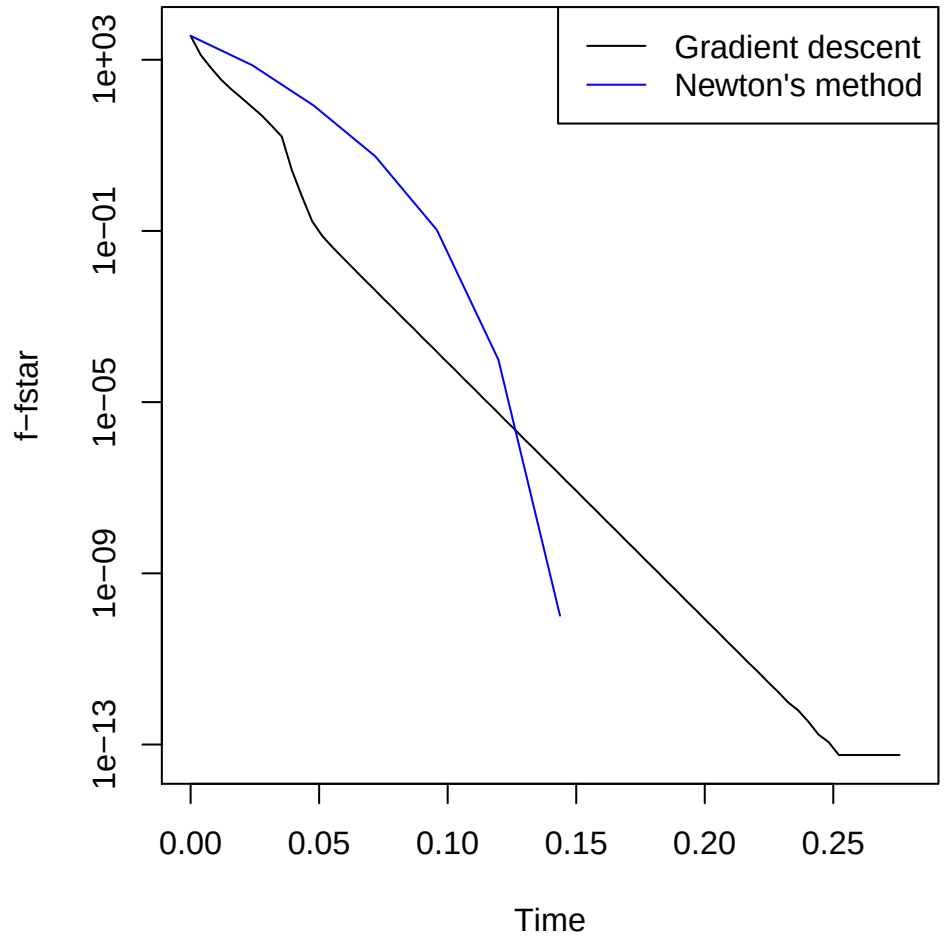
- Linear and quadratic functions
- $f(x) = -\sum_{i=1}^{n} \log(x_i)$ on $\mathbb{R}^n_{++}$
- $f(X) = -\log(\det(X))$ on $\mathbb{S}^n_{++}$
- If g is self-concordant, then so is $f(x) = g(Ax + b)$
- In the definition of self-concordance, we can replace factor of 2 by a general $\kappa > 0$
- If g is κ-self-concordant, then we can rescale: $f(x) = \frac{\kappa^2}{4} g(x)$ is self-concordant (2-self-concordant)

Comparison to first-order methods

At a high-level:

- Memory: each iteration of Newton's method requires $O(n^2)$ storage ($n \times n$ Hessian); each gradient iteration requires $O(n)$ storage (n-dimensional gradient)
- Computation: each Newton iteration requires $O(n^3)$ flops (solving a dense $n \times n$ linear system); each gradient iteration requires $O(n)$ flops (scaling/adding n-dimensional vectors)
- Backtracking: backtracking line search has roughly the same cost, both use $O(n)$ flops per inner backtracking step
- Conditioning: Newton's method is not affected by a problem's conditioning, but gradient descent can seriously degrade

Back to logistic regression example: now x-axis is parametrized in terms of time taken per iteration



Each gradient descent step is $O(p)$, but each Newton step is $O(p^3)$

Equality-constrained Newton's method

Consider now:

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

Equality-constrained Newton: start with $Ax^{(0)} = b$, and repeat

$$x^+ = x + tv, \quad \text{where}$$

$$v = \underset{Az=0}{\operatorname{argmin}} \ \nabla f(x)^T (z - x) + \frac{1}{2}(z - x)^T \nabla^2 f(x)(z - x)$$

This keeps x^+ in feasible set, since $Ax^+ = Ax + tAv = b + 0 = b$. Furthermore, v is the solution to minimizing a quadratic subject to equality constraints. KKT conditions:

$$\begin{bmatrix} \nabla^2 f(x) & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} -\nabla f(x) \\ 0 \end{bmatrix}$$

for some w. Direction v is again given by solving a linear system

Quasi-Newton methods

If the Hessian is too expensive (or singular), then a quasi-Newton method can be used to approximate $\nabla^2 f(x)$ with $H \succ 0$, and we update according to

$$x^+ = x - tH^{-1}\nabla f(x)$$

- Approximate Hessian H is recomputed at each step. Goal is to make H^{-1} cheap to apply (possibly, cheap storage too)
- Convergence is fast: superlinear, but not the same as Newton. Roughly n steps of quasi-Newton make same progress as one Newton step
- Very wide variety of quasi-Newton methods; common theme is to "propogate" computation of H across iterations

Davidon-Fletcher-Powell or DFP:

- Update H, H^{-1} via rank 2 updates from previous iterations; cost is $O(n^2)$ for these updates
- Since it is being stored, applying H^{-1} is simply $O(n^2)$ flops
- Can be motivated by Taylor series expansion

Broyden-Fletcher-Goldfarb-Shanno or BFGS:

- Came after DFP, but BFGS is now much more widely used
- Again, updates H, H^{-1} via rank 2 updates, but does so in a "dual" fashion to DFP; cost is still $O(n^2)$
- Also has a limited-memory version, L-BFGS: instead of letting updates propogate over all iterations, only keeps updates from last m iterations; storage is now $O(mn)$ instead of $O(n^2)$

Part IV: Advanced methods

B. Interior-point methods

Hierarchy of second-order methods

Assuming all problems are convex, you can think of the following hierarchy that we've worked through:

- Quadratic problems are the easiest: closed-form solution
- Equality-constrained quadratic problems are still easy: we use KKT conditions to derive closed-form solution
- Equality-constrained smooth problems are next: use Newton's method to reduce this to a sequence of equality-constrained quadratic problems
- Inequality-constrained (and also equality-constrained) smooth problems are what we cover now: use interior-point methods to reduce this to a sequence of equality-constrained problems

Log barrier function

Consider the convex optimization problem

$$\begin{array}{ll} \min\limits_{x} & f(x) \\ \text{subject to} & h_i(x) \leq 0,\ i = 1, \ldots, m \\ & Ax = b \end{array}$$

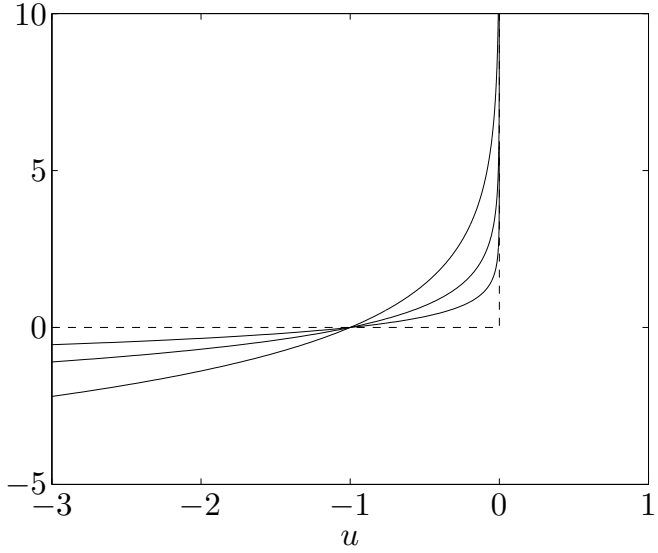
We will assume that f, $h_1, \ldots, h_m$ are convex, twice differentiable, each with domain $\mathbb{R}^n$. The function

$$\phi(x) = -\sum_{i=1}^{m} \log(-h_i(x))$$

is called the log barrier for the above problem. Its domain is the set of strictly feasible points, $\{x : h_i(x) < 0,\ i = 1, \ldots, m\}$, which we assume is nonempty. (Note this implies strong duality holds)

Ignoring equality constraints for now, our problem can be written as

$$\min_x \; f(x) + \sum_{i=1}^m I_{\{h_i(x) \leq 0\}}(x)$$



We can approximate the sum of indicators by the log barrier:

$$\min_x \; f(x) - \frac{1}{t} \sum_{i=1}^m \log(-h_i(x))$$

where $t > 0$ is a large number

This approximation is more accurate for larger t. But for any value of t, the log barrier approaches ∞ if any $h_i(x) \to 0$

Log barrier calculus

For the log barrier function

$$\phi(x) = -\sum_{i=1}^{m} \log(-h_i(x))$$

we have for its gradient:

$$\nabla\phi(x) = -\sum_{i=1}^{m} \frac{1}{h_i(x)} \nabla h_i(x)$$

and for its Hessian:

$$\nabla^2\phi(x) = \sum_{i=1}^{m} \frac{1}{h_i(x)^2} \nabla h_i(x) \nabla h_i(x)^T - \sum_{i=1}^{m} \frac{1}{h_i(x)} \nabla^2 h_i(x)$$

Central path

Consider barrier problem:

$$\begin{array}{ll} \min\limits_{x} & tf(x) + \phi(x) \\ \text{subject to} & Ax = b \end{array}$$

The central path is defined by solution $x^\star(t)$ with respect to $t > 0$

- Hope is that, as $t \to \infty$, we will have $x^\star(t) \to x^\star$, solution to our original problem
- Why don't we just set t to be some huge value, and solve the above problem? Directly seek solution at end of central path?
- Problem is that this is seriously inefficient in practice
- Much more efficient to traverse the central path, as we will see

An important special case: barrier problem for a linear program:

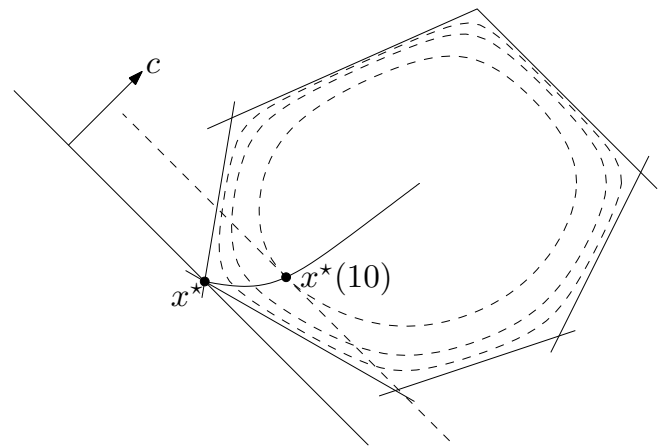
$$\min_x \quad tc^T x - \sum_{i=1}^{m} \log(e_i - d_i^T x)$$

The barrier function corresponds to polyhedral constraint $Dx \leq e$

Gradient optimality condition:

$$0 = tc - \sum_{i=1}^{m} \frac{1}{e_i - d_i^T x^\star(t)} d_i$$

This means that gradient $\nabla\phi(x^\star(t))$ must be parallel to $-c$, i.e., hyperplane $\{x : c^T x = c^T x^\star(t)\}$ lies tangent to contour of ϕ at $x^\star(t)$



(From B & V page 565)

KKT conditions and duality

Central path is characterized by its KKT conditions:

$$t\nabla f(x^\star(t)) - \sum_{i=1}^{m} \frac{1}{h_i(x^\star(t))}\nabla h_i(x^\star(t)) + A^T w = 0,$$
$$Ax^\star(t) = b, \quad h_i(x^\star(t)) < 0, \quad i = 1, \ldots, m$$

for some $w \in \mathbb{R}^m$. But we don't really care about dual variable for barrier problem ...

From central path points, we can derive feasible dual points for our original problem. Given $x^\star(t)$ and corresponding w, we define

$$u_i^\star(t) = -\frac{1}{th_i(x^\star(t))}, \quad i = 1, \ldots, m, \quad v^\star(t) = w/t$$

We claim $u^\star(t), v^\star(t)$ are dual feasible for original problem, whose Lagrangian is

$$L(x, u, v) = f(x) + \sum_{i=1}^{m} u_i h_i(x) + v^T(Ax - b)$$

Why?

- Note that $u_i^\star(t) > 0$ since $h_i(x^\star(t)) < 0$ for all $i = 1, \ldots, m$
- Further, the point $(u^\star(t), v^\star(t))$ lies in domain of Lagrange dual function $g(u, v)$, since by definition

$$\nabla f(x^\star(t)) + \sum_{i=1}^{m} u_i(x^\star(t)) \nabla h_i(x^\star(t)) + A^T v^\star(t) = 0$$

I.e., $x^\star(t)$ minimizes Lagrangian $L(x, u^\star(t), v^\star(t))$ over x, so $g(u^\star(t), v^\star(t)) > -\infty$

Duality gap

This allows us to bound suboptimality of $f(x^\star(t))$, with respect to original problem, via the duality gap. We compute

$$\begin{aligned} g(u^\star(t), v^\star(t)) &= f(x^\star(t)) + \sum_{i=1}^m u_i^\star(t) h_i(x^\star(t)) + \\ &\qquad v^\star(t)^T (Ax^\star(t) - b) \\ &= f(x^\star(t)) - m/t \end{aligned}$$

That is, we know that $f(x^\star(t)) - f^\star \leq m/t$

This will be very useful as a stopping criterion; it also confirms the hope that $x^\star(t) \to x^\star$ as $t \to \infty$

Barrier method

The barrier method solves a sequence of problems

$$\begin{array}{ll} \min\limits_{x} & tf(x) + \phi(x) \\ \text{subject to} & Ax = b \end{array}$$

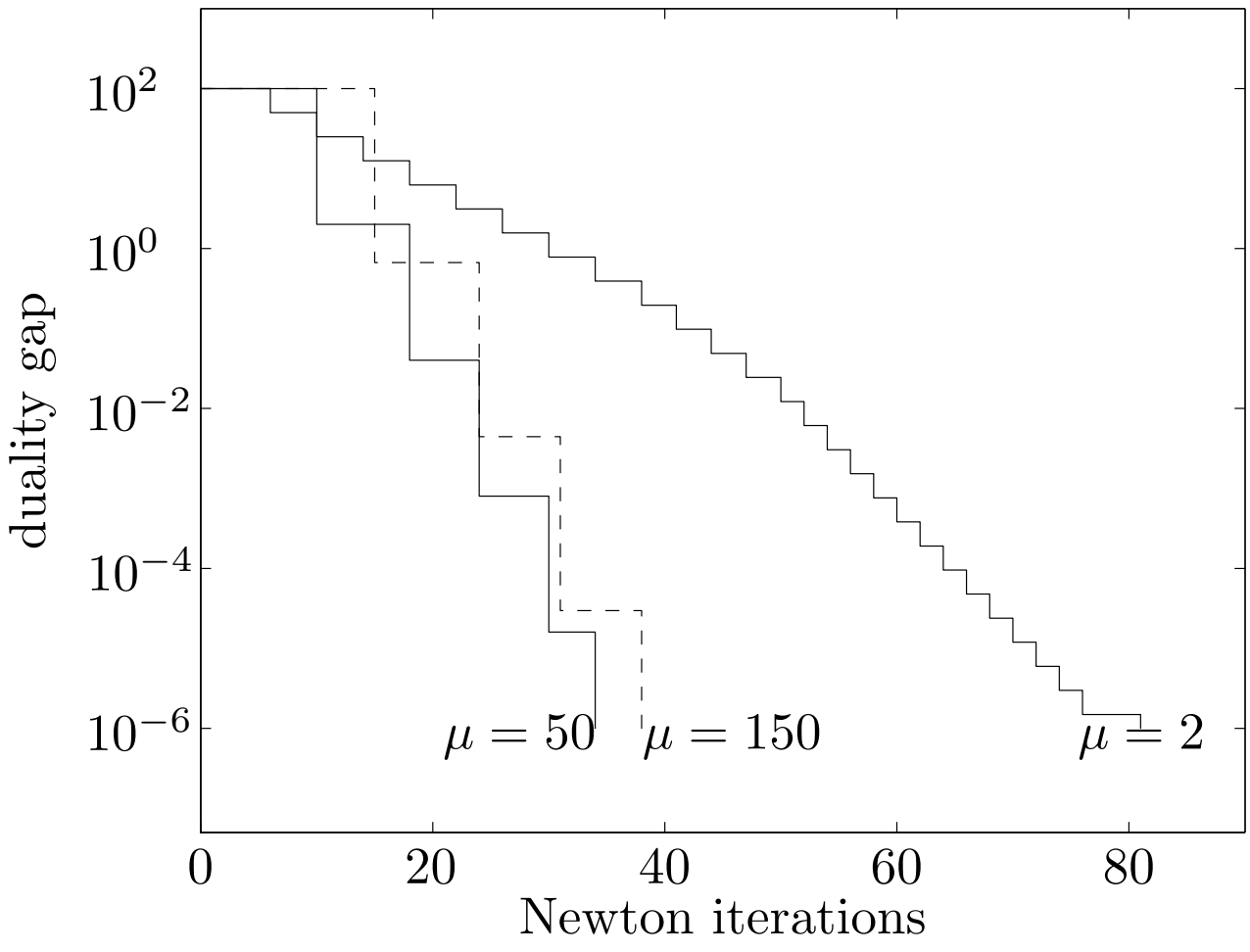
for increasing values of $t > 0$, until duality gap satisfies $m/t \leq \epsilon$

We fix $t^{(0)} > 0$, $\mu > 1$. We use Newton to compute $x^{(0)} = x^\star(t)$, a solution to barrier problem at $t = t^{(0)}$. For $k = 1, 2, 3, \ldots$

- Solve the barrier problem at $t = t^{(k)}$, using Newton initialized at $x^{(k-1)}$, to yield $x^{(k)} = x^\star(t)$
- Stop if $m/t \leq \epsilon$, else update $t^{(k+1)} = \mu t$

The first step above is called a centering step (since it brings $x^{(k)}$ onto the central path)

Example of a small LP in $n = 50$ dimensions, $m = 100$ inequality constraints (from B & V page 571):



Convergence analysis

Assume that we solve the centering steps exactly. The following result is immediate

Theorem: The barrier method after k centering steps satisfies

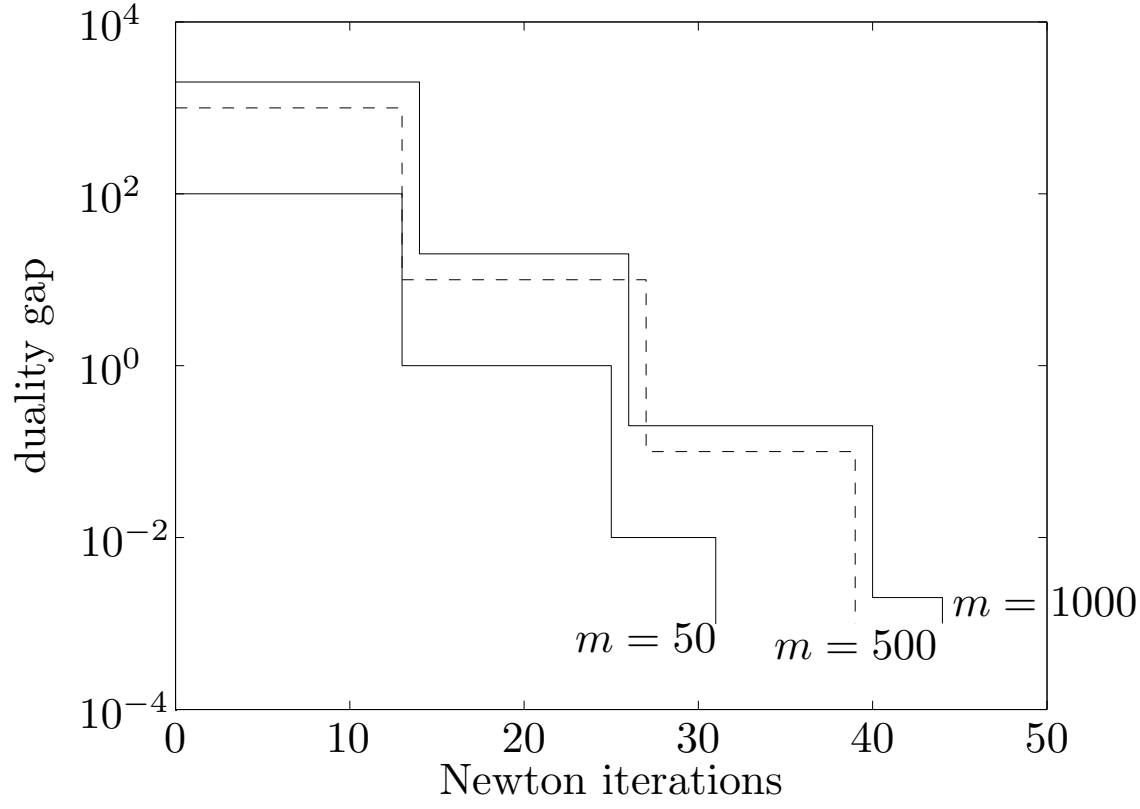
$$f(x^{(k)}) - f^\star \leq \frac{m}{\mu^k t^{(0)}}$$

In other words, to reach a desired accuracy level of ϵ, we require

$$\frac{\log(m/(t^{(0)}\epsilon))}{\log \mu}$$

centering steps with the barrier method (plus initial centering step)

Example of barrier method progress for an LP with m constraints (from B & V page 575):



Can see roughly linear convergence in each case, and logarithmic scaling with m

How many Newton iterations?

Informally, due to careful central path traversal, in each centering step, Newton is already in quadratic convergence phase, so takes nearly constant number of iterations

This can be formalized under self-concordance. Suppose:

- The function $tf + \phi$ is self-concordant
- Our original problem has bounded sublevel sets

Then we can terminate each Newton solve at appropriate accuracy, and the total number of Newton iterations is still $O(\log(m/(t^{(0)}\epsilon))$ (where constants do not depend on problem-specific conditioning). See Chapter 11.5 of B & V

Importantly, $tf + \phi = tf - \sum_{i=1}^m \log(-h_i)$ is self-concordant when f, h_i are all linear or quadratic. So this covers LPs, QPs, QCQPs

Primal-dual interior-point methods

Centering step in the barrier method: can interpret as Newton's method for nonlinear system of "perturbed" KKT conditions

Primal-dual interior-point methods are defined similarly. Overview:

- Both can be motivated in terms of perturbed KKT conditions, primal-dual is more direct
- Primal-dual interior-point methods take one Newton step, and move on (no separate inner and outer loops)
- Primal-dual interior-point iterates are not dual feasible
- Primal-dual interior-point methods are often more efficient, as they can exhibit better than linear convergence
- Primal-dual interior-point methods are less intuitive ...

Part IV: Advanced methods

C. Coordinate descent

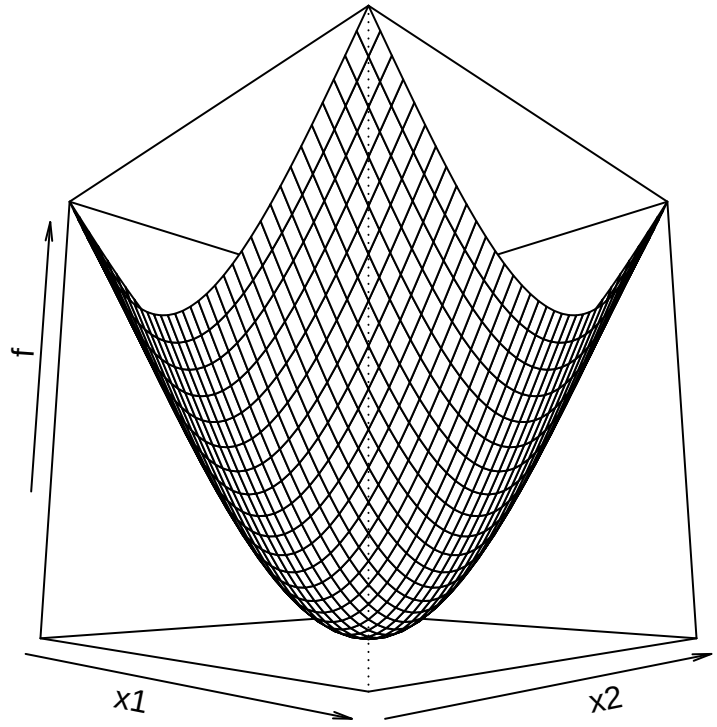
Coordinatewise optimality

Let's start with a motivating question (apparently has been around since the "birth" of optimization as a discipline)

Q: Given convex, differentiable $f : \mathbb{R}^n \to \mathbb{R}$, if we are at a point x such that $f(x)$ is minimized along each coordinate axis, then *have we found a global minimizer?*

That is, does $f(x + ve_i) \geq f(x)$ for all $v, i \Rightarrow f(x) = \min_z f(z)$?

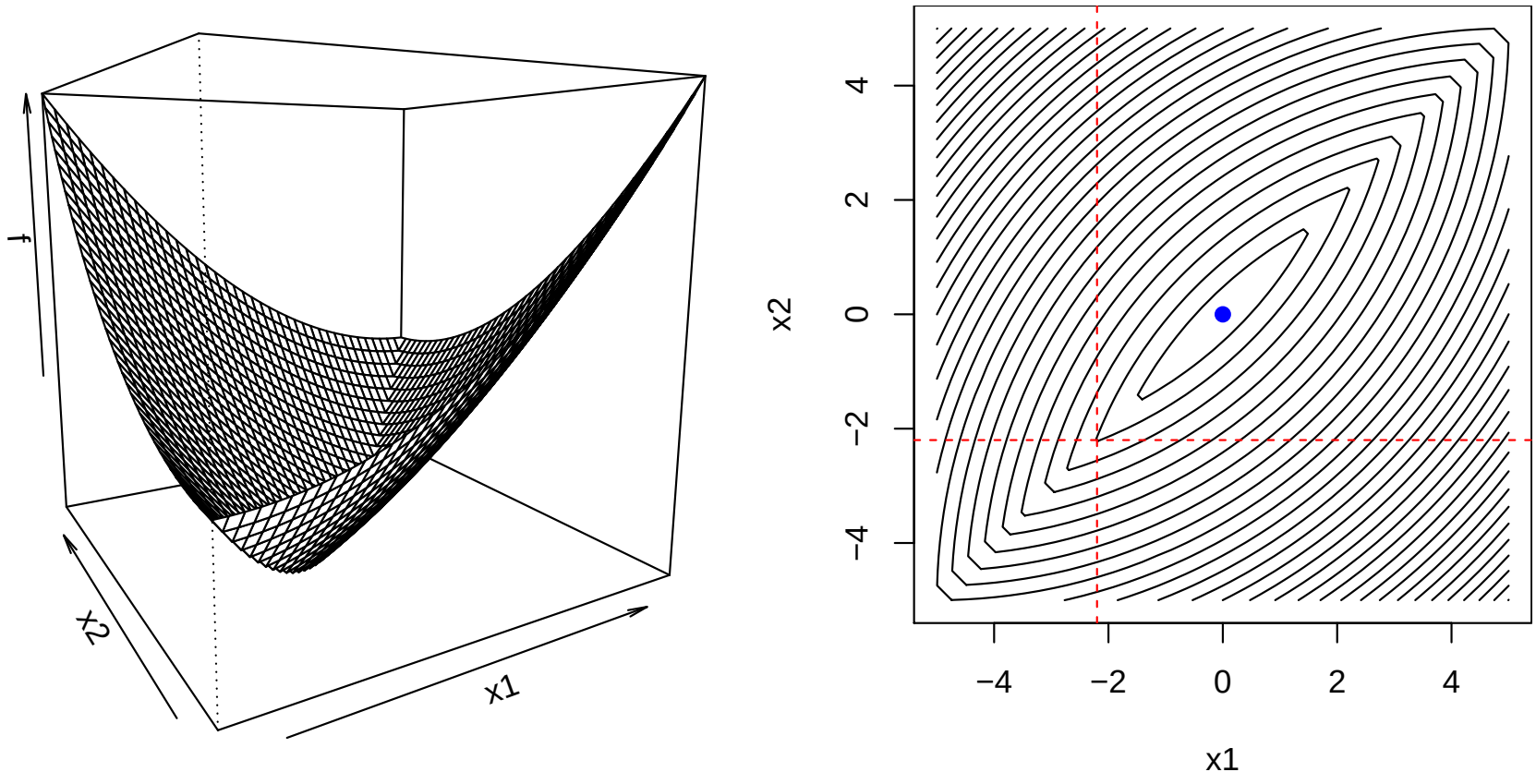
(Here $e_i = (0, \ldots, 1, \ldots, 0) \in \mathbb{R}^n$ is the ith standard basis vector)



A: Yes! Proof:

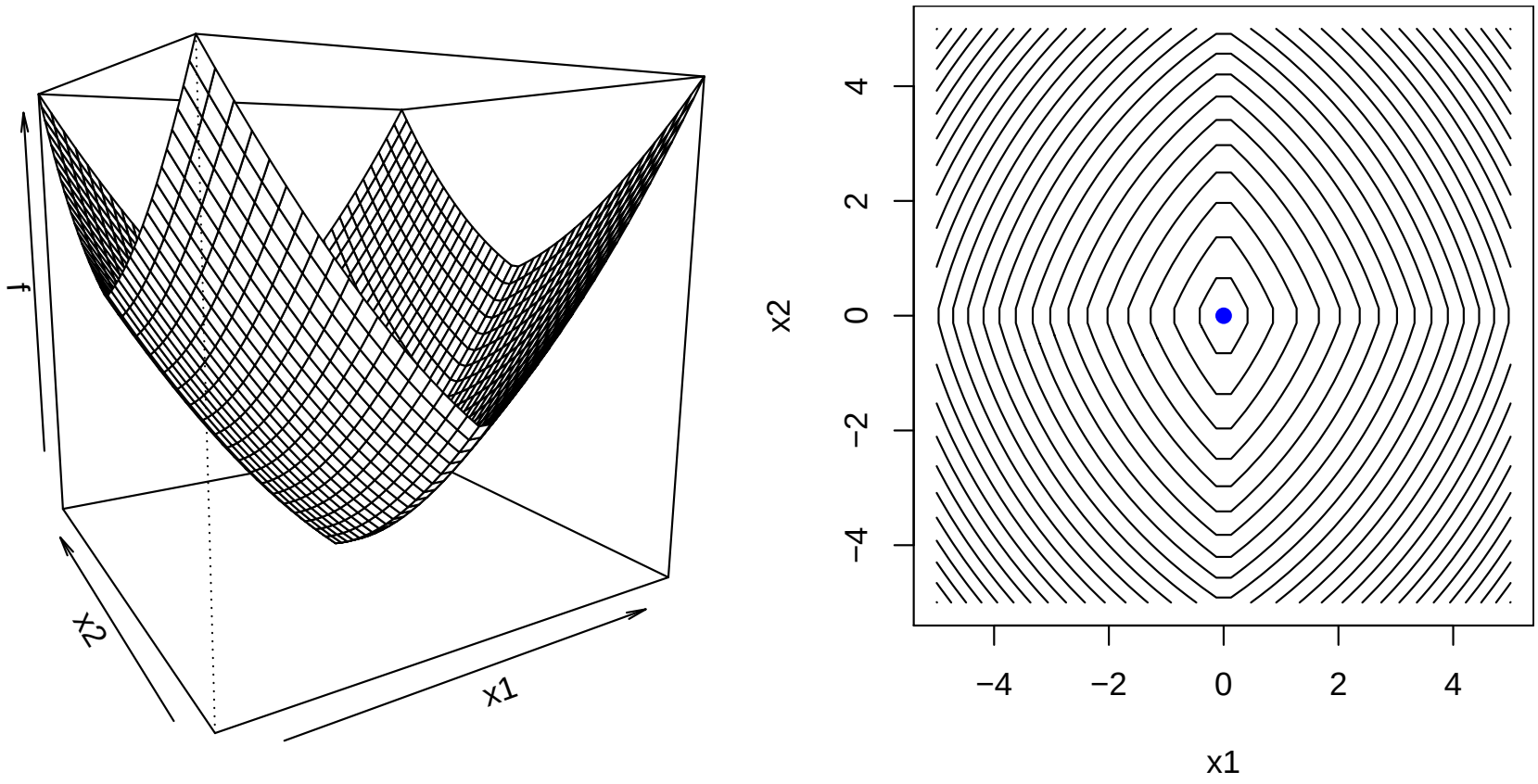
$$0 = \nabla f(x) = \left(\frac{\partial f}{\partial x_1}(x), \ldots, \frac{\partial f}{\partial x_n}(x) \right)$$

Q: Same question, but now for f convex, and not differentiable?



A: No! Look at the above counterexample

Q: Same, now $f(x) = g(x) + \sum_{i=1}^n h_i(x_i)$, with g convex, smooth, and each h_i convex? (Here the nonsmooth part is called separable)



A: Yes! Intuition is that separability condition "rotates the difficult parts" to be parallel to the coordinate axes

Proof: recall $f(x) = g(x) + \sum_{i=1}^n h_i(x_i)$. Using convexity of g and subgradient optimality

$$f(y) - f(x) \geq \nabla g(x)^T (y - x) + \sum_{i=1}^n [h_i(y_i) - h_i(x_i)]$$

$$= \sum_{i=1}^n \underbrace{[\nabla_i g(x)(y_i - x_i) + h_i(y_i) - h_i(x_i)]}_{a_i}$$

However, because each x_i is coordinatewise optimal, we must have $0 \in \partial_i f(x) = \partial_i (g(x) + \sum_{i=1}^n h_i(x_i))$, i.e.,

$$-\nabla_i g(x) \in \partial h_i(x_i)$$

By definition of a subgradient, for any y_i,

$$h_i(y_i) \geq h_i(x_i) - \nabla g_i(x_i)(y_i - x_i)$$

and so each $a_i \geq 0$, therefore $f(y) \geq f(x)$

Coordinate descent

This suggests that for the problem

$$\min_x \ f(x)$$

where $f(x) = g(x) + \sum_{i=1}^n h_i(x_i)$, with g convex and differentiable and each h_i convex, we can use coordinate descent: let $x^{(0)} \in \mathbb{R}^n$, and repeat

$$x_i^{(k)} = \underset{x_i}{\operatorname{argmin}} \ f\big(x_1^{(k)}, \ldots, x_{i-1}^{(k)}, x_i, x_{i+1}^{(k-1)}, \ldots, x_n^{(k-1)}\big),$$

$$i = 1, \ldots, n$$

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

Important note: we always use most recent information possible

Tseng (2001) showed that for such f (provided f is continuous on compact set $\{x : f(x) \leq f(x^{(0)})\}$ and f attains its minimum), any limit point of $x^{(k)}$, $k = 1, 2, 3, \ldots$ is a minimizer of f[1]

Notes:

- Order of cycle through coordinates is arbitrary, can use any permutation of $\{1, 2, \ldots, n\}$
- Can everywhere replace individual coordinates with blocks of coordinates
- "One-at-a-time" update scheme is critical, and "all-at-once" scheme does not necessarily converge
- The analogy for solving linear systems: Gauss-Seidel versus Jacobi method

[1] Using basic real analysis, we know $x^{(k)}$ has subsequence converging to $x^\star$ (Bolzano-Weierstrass), and $f(x^{(k)})$ converges to $f^\star$ (monotone convergence)

Example: linear regression

Given $y \in \mathbb{R}^n$, and $X \in \mathbb{R}^{n \times p}$ with columns $X_1, \ldots, X_p$, consider the linear regression problem:

$$\min_{\beta} \ \frac{1}{2}\|y - X\beta\|_2^2$$

Minimizing over β_i, with all β_j, $j \neq i$ fixed:

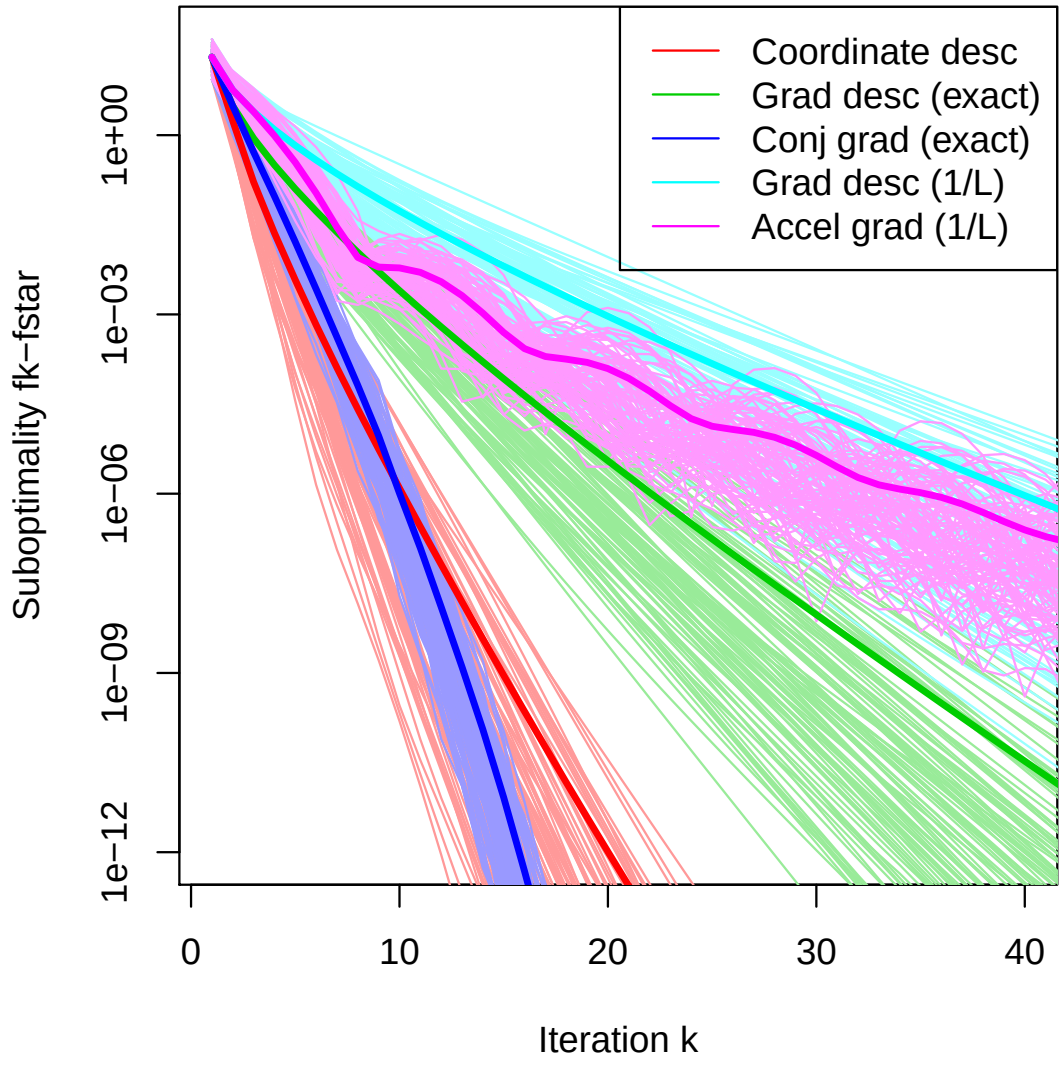
$$0 = \nabla_i f(\beta) = X_i^T(X\beta - y) = X_i^T(X_i\beta_i + X_{-i}\beta_{-i} - y)$$

i.e., we take

$$\beta_i = \frac{X_i^T(y - X_{-i}\beta_{-i})}{X_i^T X_i}$$

Coordinate descent repeats this update for $i = 1, 2, \ldots, p, 1, 2, \ldots$.
Note that this is exactly Gauss-Seidl for the system $X^T X\beta = X^T y$

Coordinate descent vs gradient descent for linear regression: 100 random instances with $n = 100$, $p = 20$



Is it fair to compare 1 cycle of coordinate descent to 1 iteration of gradient descent? Yes, if we're clever

- Gradient descent: $\beta \leftarrow \beta + tX^T(y - X\beta)$, costs $O(np)$ flops
- Coordinate descent, one coordinate update:

$$\beta_i \leftarrow \frac{X_i^T(y - X_{-i}\beta_{-i})}{X_i^T X_i} = \frac{X_i^T r}{\|X_i\|_2^2} + \beta_i$$

  where $r = y - X\beta$
- Each coordinate costs $O(n)$ flops: $O(n)$ to update r, $O(n)$ to compute $X_i^T r$
- One cycle of coordinate descent costs $O(np)$ operations, same as gradient descent

Example: lasso regression

Consider the lasso problem:

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

Note that nonsmooth part here is separable: $\|\beta\|_1 = \sum_{i=1}^p |\beta_i|$. Minimizing over β_i, with β_j, $j \neq i$ fixed:

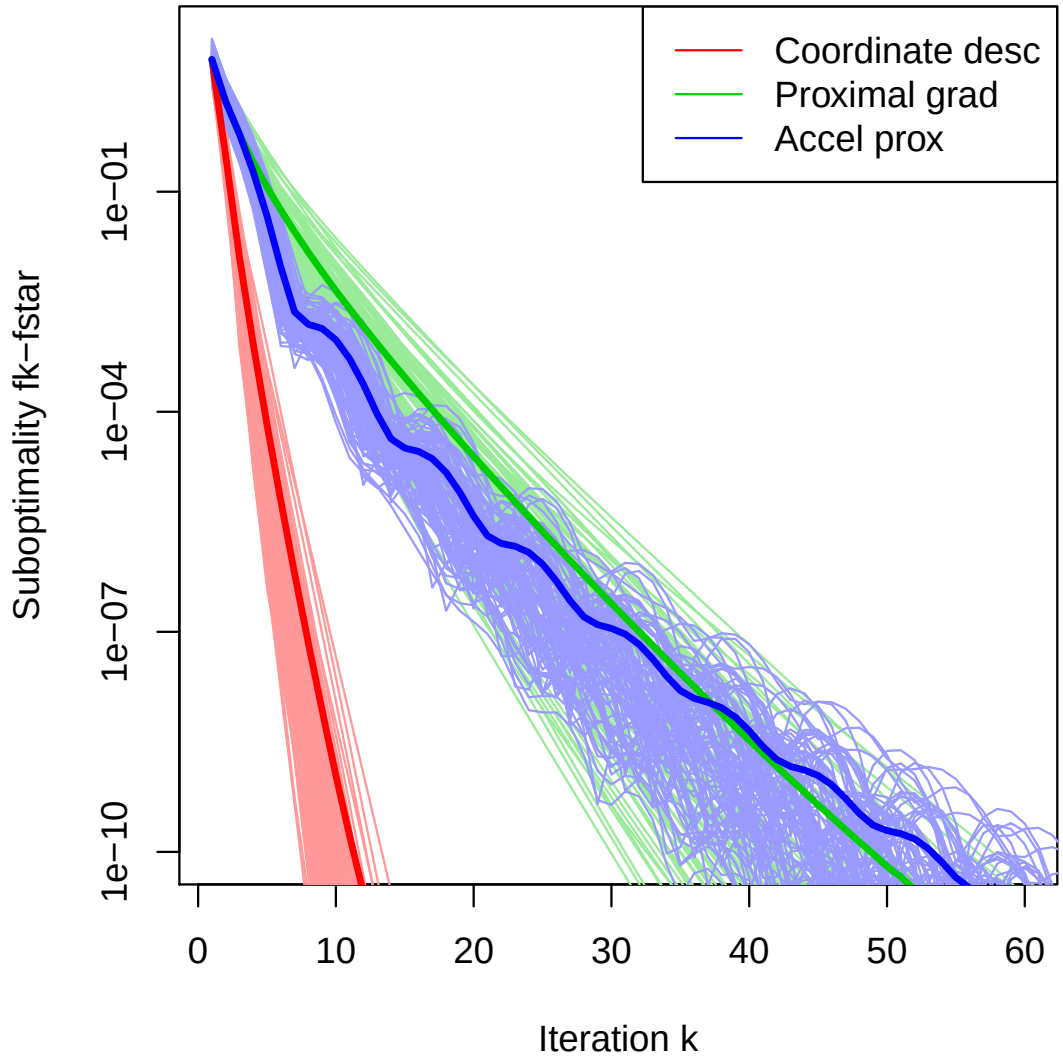
$$0 = X_i^T X_i \beta_i + X_i^T (X_{-i}\beta_{-i} - y) + \lambda s_i$$

where $s_i \in \partial|\beta_i|$. Solution is simply given by soft-thresholding

$$\beta_i = S_{\lambda/\|X_i\|_2^2}\left(\frac{X_i^T(y - X_{-i}\beta_{-i})}{X_i^T X_i}\right)$$

Repeat this for $i = 1, 2, \ldots, p, 1, 2, \ldots$

Coordinate descent vs proximal gradient for lasso regression: 100 random instances with $n = 200$, $p = 50$ (all methods cost $O(np)$ per iter)



Example: box-constrained QP

Given $b \in \mathbb{R}^n$, $Q \in \mathbb{S}^n_+$, consider a box-constrained QP:

$$\min_x \ \frac{1}{2} x^T Q x + b^T x \ \text{ subject to } \ l \leq x \leq u$$

Fits into our framework, as $I\{l \leq x \leq u\} = \sum_{i=1}^n I\{l_i \leq x_i \leq u_i\}$

Minimizing over x_i with all x_j, $j \neq i$ fixed: same basic steps give

$$x_i = T_{[l_i, u_i]} \left(\frac{b_i - \sum_{j \neq i} Q_{ij} x_j}{Q_{ii}} \right)$$

where $T_{[l_i, u_i]}$ is the truncation (projection) operator onto $[l_i, u_i]$:

$$T_{[l_i, u_i]}(z) = \begin{cases} u_i & \text{if } z > u_i \\ z & \text{if } l_i \leq z \leq u_i \\ l_i & \text{if } z < l_i \end{cases}$$

Example: support vector machines

A coordinate descent strategy can be applied to the SVM dual:

$$\min_{\alpha} \ \frac{1}{2}\alpha^T \tilde{X}\tilde{X}^T\alpha - 1^T\alpha \ \text{ subject to } \ 0 \leq \alpha \leq C1, \ \alpha^T y = 0$$

Sequential minimal optimization or SMO (Platt 1998) is basically blockwise coordinate descent in blocks of 2. Instead of cycling, it chooses the next block greedily

Recall the complementary slackness conditions

$$\alpha_i\big(1 - \xi_i - (\tilde{X}\beta)_i - y_i\beta_0\big) = 0, \quad i = 1, \ldots, n \tag{1}$$

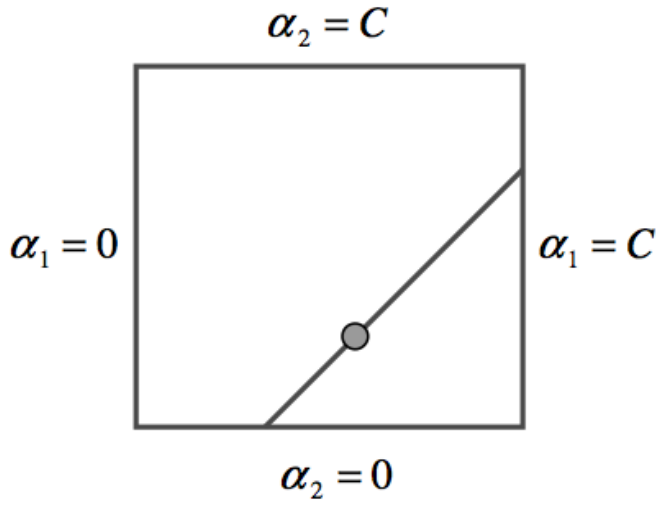
$$(C - \alpha_i)\xi_i = 0, \quad i = 1, \ldots, n \tag{2}$$

where β, β_0, ξ are the primal coefficients, intercept, and slacks. Recall that $\beta = \tilde{X}^T\alpha$, β_0 is computed from (1) using any i such that $0 < \alpha_i < C$, and ξ is computed from (1), (2)

SMO repeats the following two steps:

- Choose α_i, α_j that violate complementary slackness, greedily (using heuristics)
- Minimize over α_i, α_j exactly, keeping all other variables fixed

Using equality constraint, reduces to minimizing univariate quadratic over an interval (From Platt 1998)



Note this does not meet separability assumptions for convergence from Tseng (2001), and a different treatment is required

Many further developments on coordinate descent for SVMs have been made; e.g., a recent one is Hsiesh et al. (2008)

Coordinate descent in statistics and ML

History in statistics/ML:

- Idea appeared in Fu (1998), and then again in Daubechies et al. (2004), but was inexplicably ignored
- Later, three papers in 2007, especially Friedman et al. (2007), really sparked interest in statistics and ML communities

Why is it used?

- Very simple and easy to implement
- Careful implementations can achieve state-of-the-art
- Scalable, e.g., don't need to keep full data in memory

Examples: lasso regression, lasso GLMs (under proximal Newton), SVMs, group lasso, graphical lasso (applied to the dual), additive modeling, matrix completion, regression with nonconvex penalties

Pathwise coordinate descent for lasso

Structure for pathwise coordinate descent, Friedman et al. (2009):

Outer loop (pathwise strategy):

- Compute the solution over a sequence $\lambda_1 > \lambda_2 > \ldots > \lambda_r$ of tuning parameter values
- For tuning parameter value λ_k, initialize coordinate descent algorithm at the computed solution for λ_{k+1} (warm start)

Inner loop (active set strategy):

- Perform one coordinate cycle (or small number of cycles), and record active set A of coefficients that are nonzero
- Cycle over only the coefficients in A until convergence
- Check KKT conditions over all coefficients; if not all satisfied, add offending coefficients to A, go back one step

Coordinate gradient descent

For a smooth function f, the iterations

$$x_i^{(k)} = x_i^{(k-1)} - t_{ki} \cdot \nabla_i f\big(x_1^{(k)}, \ldots, x_{i-1}^{(k)}, x_i, x_{i+1}^{(k-1)}, \ldots, x_n^{(k-1)}\big), \quad i = 1, \ldots, n$$

for $k = 1, 2, 3, \ldots$ are called coordinate gradient descent, and when $f = g + h$, with g smooth and $h = \sum_{i=1}^n h_i$, the iterations

$$x_i^{(k)} = \mathrm{prox}_{h_i, t_{ki}}\Big(x_i^{(k-1)} - t_{ki} \cdot \nabla_i g\big(x_1^{(k)}, \ldots, x_i^{(k-1)}, \ldots, x_n^{(k-1)}\big)\Big), \quad i = 1, \ldots, n$$

for $k = 1, 2, 3, \ldots$ are called coordinate proximal gradient descent

When g is quadratic, (proximal) coordinate gradient descent is the same as coordinate descent under proper step sizes

Convergence analyses

Theory for coordinate descent moves quickly. Each combination of the following cases has (probably) been analyzed:

- Coordinate descent or (proximal) coordinate gradient descent?
- Cyclic rule, permuted cyclic, or greedy rule, randomized rule?

Roughly speaking, results are similar to those for proximal gradient descent: under standard conditions, get standard rates

But constants differ and this matters! Much recent work is focused on improving them

Some references are Beck and Tetruashvili (2013), Wright (2015), Sun and Hong (2015), Li et al. (2016)

Graphical lasso

Consider $X \in \mathbb{R}^{n \times p}$, with rows $x_i \sim N(0, \Sigma)$, $i = 1, \ldots, n$, drawn independently. Suppose Σ is unknown. It is often reasonable (for large p) to seek a sparse estimate of Σ^{-1}

Why? For $z \sim N(0, \Sigma)$, normality theory tells us

$$\Sigma^{-1}_{ij} = 0 \iff z_i, z_j \text{ conditionally independent given } z_\ell,\ \ell \neq i, j$$

Graphical lasso (Banerjee et al. 2007, Friedman et al. 2007):

$$\min_{\Theta \in \mathbb{S}^p_+} -\log\det \Theta + \mathrm{tr}(S\Theta) + \lambda \|\Theta\|_1$$

where $S = X^T X/n$, and $\|\Theta\|_1 = \sum_{i,j=1}^p |\Theta_{ij}|$. Observe that this is a convex problem. Solution $\hat{\Theta}$ serves as estimate for Σ^{-1}

Glasso algorithm

Graphical lasso KKT conditions (stationarity):

$$-\Theta^{-1} + S + \lambda\Gamma = 0$$

where $\Gamma_{ij} \in \partial|\Theta_{ij}|$. Let $W = \Theta^{-1}$. Note $W_{ii} = S_{ii} + \lambda$, because $\Theta_{ii} > 0$ at solution. Now partition:

$$W = \begin{bmatrix} W_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \quad \Theta = \begin{bmatrix} \Theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \end{bmatrix} \quad S = \begin{bmatrix} S_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix} \quad \Gamma = \begin{bmatrix} \Gamma_{11} & \gamma_{12} \\ \gamma_{21} & \gamma_{22} \end{bmatrix}$$

where $W_{11} \in \mathbb{R}^{(p-1)\times(p-1)}$, $w_{12} \in \mathbb{R}^{(p-1)\times 1}$, and $w_{21} \in \mathbb{R}^{1\times(p-1)}$, $w_{22} \in \mathbb{R}$; same with others

Glasso algorithm (Friedman et al., 2007): solve for w_{12} (recall that w_{22} is known), with all other columns fixed; then solve for second-to-last column, etc., and cycle around until convergence

Glasso block update

Consider $(1, 2)$-block of KKT conditions:

$$-w_{12} + s_{12} + \lambda\gamma_{12} = 0$$

Because $\begin{bmatrix} W_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \begin{bmatrix} \Theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & 1 \end{bmatrix}$, we know that $w_{12} = -W_{11}\theta_{12}/\theta_{22}$. Substituting this into the above,

$$W_{11}\frac{\theta_{12}}{\theta_{22}} + s_{12} + \lambda\gamma_{12} = 0$$

Letting $\beta = \theta_{12}/\theta_{22}$ and recalling that $\theta_{22} > 0$ at solution, this is

$$W_{11}\beta + s_{12} + \lambda\rho = 0$$

where $\rho \in \partial\|\beta\|_1$. What does this condition look like?

Hidden lasso problem

These are exactly the KKT conditions for

$$\min_{\beta} \ \beta^T W_{11}\beta + s_{12}^T\beta + \lambda\|\beta\|_1$$

which is (basically) a lasso problem and can be itself solved quickly via coordinate descent

From β we get $w_{12} = -W_{11}\beta$, and set $w_{21} = w_{12}^T$. Then θ_{12}, θ_{22} are obtained from $\begin{bmatrix} W_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \begin{bmatrix} \Theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & 1 \end{bmatrix}$, and we set $\theta_{21} = \theta_{12}^T$

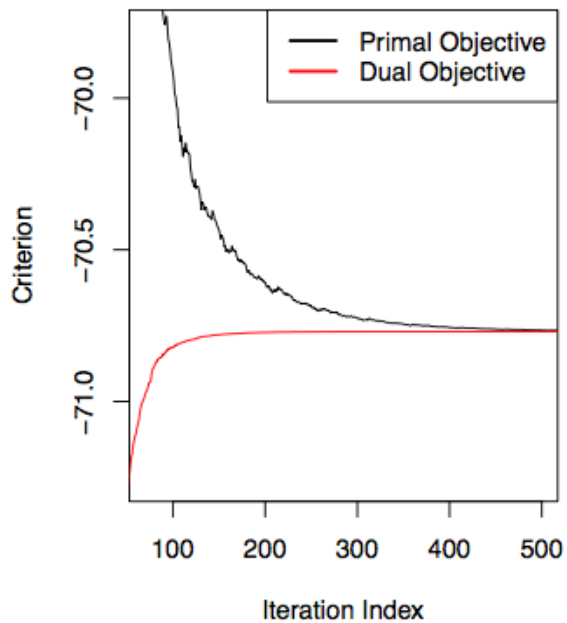
The next step moves on to a different column of W, and so on; hence we have reduced the graphical lasso problem to a repeated sequence of lasso problems

Coordinate descent?

The glasso algorithm is efficient and scales well. It also has the feel of coordinate descent. But, people have noticed that the criterion doesn't decrease monotonically—so it can't be coordinate descent?

The glasso algorithm makes a variable transformation and solves in terms of coordinate blocks of W; these are not coordinate blocks of original variable Θ, so strictly speaking it is not a coordinate descent algorithm

However, it can be shown that glasso is doing coordinate ascent on the dual problem! (Mazumder et al. 2011)



Part IV: Advanced methods

D. ADMM

Reminder: conjugate functions

Recall that given $f : \mathbb{R}^n \to \mathbb{R}$, the function

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

is called its conjugate

- Conjugates appear frequently in dual programs, since

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

- If f is closed and convex, then $f^{**} = f$. Also,

$$x \in \partial f^*(y) \iff y \in \partial f(x) \iff x \in \operatorname*{argmin}_z \; f(z) - y^T z$$

- If f is strictly convex, then $\nabla f^*(y) = \operatorname*{argmin}_z \; f(z) - y^T z$

Dual ascent

Even if we can't derive dual (conjugate) in closed form, we can still use dual-based gradient or subgradient methods

Consider the problem

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

Its dual problem is

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

where f^* is conjugate of f. Defining $g(u) = -f^*(-A^T u) - b^T u$, note that

$$\partial g(u) = A\partial f^*(-A^T u) - b$$

Therefore, using what we know about conjugates

$$\partial g(u) = Ax - b \quad \text{where} \quad x \in \operatorname*{argmin}_z \; f(z) + u^T Az$$

The dual subgradient method (for maximizing the dual objective) starts with an initial dual guess $u^{(0)}$, and repeats for $k = 1, 2, 3, \ldots$

$$x^{(k)} \in \operatorname*{argmin}_x \; f(x) + (u^{(k-1)})^T Ax$$
$$u^{(k)} = u^{(k-1)} + t_k(Ax^{(k)} - b)$$

Step sizes t_k, $k = 1, 2, 3, \ldots$, are chosen in standard ways

Recall that if f is strictly convex, then f^* is differentiable, and so this becomes dual gradient ascent, which repeats for $k = 1, 2, 3, \ldots$

$$x^{(k)} = \operatorname*{argmin}_x \; f(x) + (u^{(k-1)})^T Ax$$
$$u^{(k)} = u^{(k-1)} + t_k(Ax^{(k)} - b)$$

(Difference is that each $x^{(k)}$ is unique, here.) Again, step sizes t_k, $k = 1, 2, 3, \ldots$ are chosen in standard ways

Lastly, proximal gradients and acceleration can be applied as they would usually

Dual decomposition

Consider

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

Here $x = (x_1, \ldots, x_B) \in \mathbb{R}^n$ divides into B blocks of variables, with each $x_i \in \mathbb{R}^{n_i}$. We can also partition A accordingly

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

Simple but powerful observation, in calculation of (sub)gradient, is that the minimization decomposes into B separate problems:

$$x^+ \in \operatorname*{argmin}_x \sum_{i=1}^B f_i(x_i) + u^T Ax$$

$$\iff x_i^+ \in \operatorname*{argmin}_{x_i} f_i(x_i) + u^T A_i x_i, \quad i = 1, \ldots, B$$

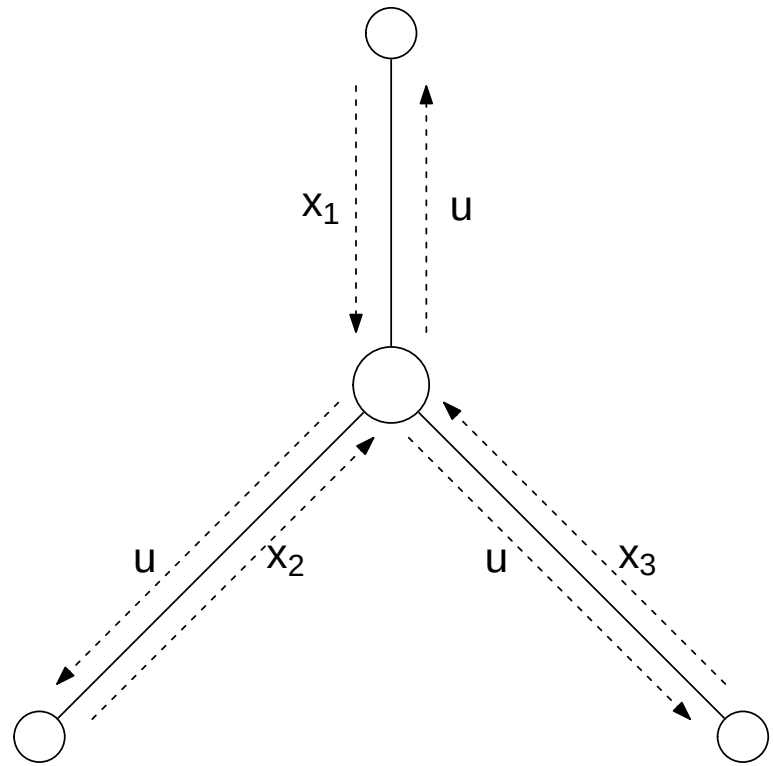
Dual decomposition algorithm: repeat for $k = 1, 2, 3, \ldots$

$$x_i^{(k)} \in \operatorname*{argmin}_{x_i} \; f_i(x_i) + (u^{(k-1)})^T A_i x_i, \quad i = 1, \ldots, B$$

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

Can think of these steps as:

- Broadcast: send u to each of the B processors, each optimizes in parallel to find x_i
- Gather: collect $A_i x_i$ from each processor, update the global dual variable u



Dual decomposition with inequality constraints

Consider

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

Dual decomposition, i.e., projected subgradient method:

$$x_i^{(k)} \in \operatorname*{argmin}_{x_i} \; f_i(x_i) + (u^{(k-1)})^T A_i x_i, \quad i = 1, \ldots, B$$

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

where u_+ denotes the positive part of u, i.e., $(u_+)_i = \max\{0, u_i\}$, $i = 1, \ldots, m$

Price coordination interpretation (Vandenberghe):

- Have B units in a system, each unit chooses its own decision variable x_i (how to allocate its goods)
- Constraints are limits on shared resources (rows of A), each component of dual variable u_j is price of resource j
- Dual update:

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

  where $s = b - \sum_{i=1}^{B} A_i x_i$ are slacks

  - Increase price u_j if resource j is over-utilized, $s_j < 0$
  - Decrease price u_j if resource j is under-utilized, $s_j > 0$
  - Never let prices get negative

Augmented Lagrangian method

(also known as: method of multipliers)

Disadvantage of dual ascent: require strong conditions to ensure convergence. Improved by augmented Lagrangian method, also called method of multipliers. We transform the primal problem:

$$\begin{array}{ll} \min\limits_{x} & f(x) + \dfrac{\rho}{2}\|Ax - b\|_2^2 \\ \text{subject to} & Ax = b \end{array}$$

where $\rho > 0$ is a parameter. Clearly equivalent to original problem, and objective is strongly convex when A has full column rank. Use dual gradient ascent:

$$\begin{aligned} x^{(k)} &= \operatorname*{argmin}_x \; f(x) + (u^{(k-1)})^T Ax + \frac{\rho}{2}\|Ax - b\|_2^2 \\ u^{(k)} &= u^{(k-1)} + \rho(Ax^{(k)} - b) \end{aligned}$$

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

$$\begin{aligned} 0 &\in \partial f(x^{(k)}) + A^T\Big(u^{(k-1)} + \rho(Ax^{(k)} - b)\Big) \\ &= \partial f(x^{(k)}) + A^T u^{(k)} \end{aligned}$$

This is the stationarity condition for original primal problem; under mild conditions $Ax^{(k)} - b \to 0$ as $k \to \infty$ (primal iterates become feasible), so KKT conditions are satisfied in the limit and $x^{(k)}, u^{(k)}$ converge to solutions

- Advantage: much better convergence properties
- Disadvantage: lose decomposability! (Separability is ruined by augmented Lagrangian ...)

Alternating direction method of multipliers

Alternating direction method of multipliers or ADMM: try for best of both worlds. Consider the problem

$$\min_{x,z} \; f(x) + g(z) \;\; \text{subject to} \;\; Ax + Bz = c$$

As before, we augment the objective

$$\begin{aligned} \min_{x} \quad & f(x) + g(z) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2 \\ \text{subject to} \quad & Ax + Bz = c \end{aligned}$$

for a parameter $\rho > 0$. We define augmented Lagrangian

$$L_\rho(x, z, u) = f(x) + g(z) + u^T(Ax + Bz - c) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2$$

ADMM repeats the steps, for $k = 1, 2, 3, \ldots$

$$
\begin{aligned}
x^{(k)} &= \operatorname*{argmin}_x \; L_\rho(x, z^{(k-1)}, u^{(k-1)}) \\
z^{(k)} &= \operatorname*{argmin}_z \; L_\rho(x^{(k)}, z, u^{(k-1)}) \\
u^{(k)} &= u^{(k-1)} + \rho(Ax^{(k)} + Bz^{(k)} - c)
\end{aligned}
$$

Note that the usual method of multipliers would have replaced the first two steps by a joint minimization

$$
(x^{(k)}, z^{(k)}) = \operatorname*{argmin}_{x,z} \; L_\rho(x, z, u^{(k-1)})
$$

Convergence guarantees

Under modest assumptions on f, g (these do not require A, B to be full rank), the ADMM iterates satisfy, for any $\rho > 0$:

- Residual convergence: $r^{(k)} = Ax^{(k)} - Bz^{(k)} - c \to 0$ as $k \to \infty$, i.e., primal iterates approach feasibility
- Objective convergence: $f(x^{(k)}) + g(z^{(k)}) \to f^\star + g^\star$, where $f^\star + g^\star$ is the optimal primal objective value
- Dual convergence: $u^{(k)} \to u^\star$, where $u^\star$ is a dual solution

For details, see Boyd et al. (2010). Note that we do not generically get primal convergence, but this is true under more assumptions

Convergence rate: roughly, ADMM behaves like first-order method. Theory still being developed, see, e.g., in Hong and Luo (2012), Deng and Yin (2012), Iutzeler et al. (2014), Nishihara et al. (2015)

Scaled form ADMM

Scaled form: denote $w = u/\rho$, so augmented Lagrangian becomes

$$L_\rho(x, z, w) = f(x) + g(z) + \frac{\rho}{2}\|Ax - Bx + c + w\|_2^2 - \frac{\rho}{2}\|w\|_2^2$$

and ADMM updates become

$$\begin{aligned}
x^{(k)} &= \underset{x}{\operatorname{argmin}}\ f(x) + \frac{\rho}{2}\|Ax + Bz^{(k-1)} - c + w^{(k-1)}\|_2^2 \\
z^{(k)} &= \underset{z}{\operatorname{argmin}}\ g(z) + \frac{\rho}{2}\|Ax^{(k)} + Bz - c + w^{(k-1)}\|_2^2 \\
w^{(k)} &= w^{(k-1)} + Ax^{(k)} + Bz^{(k)} - c
\end{aligned}$$

Note that here kth iterate $w^{(k)}$ is just a running sum of residuals:

$$w^{(k)} = w^{(0)} + \sum_{i=1}^{k} \left(Ax^{(i)} + Bz^{(i)} - c\right)$$

Practicalities

In practice, ADMM usually obtains a relatively accurate solution in a handful of iterations, but it requires a large number of iterations for a highly accurate solution (like a first-order method)

Choice of ρ can greatly influence practical convergence of ADMM:

- ρ too large $\rightarrow$ not enough emphasis on minimizing $f + g$
- ρ too small $\rightarrow$ not enough emphasis on feasibility

Boyd et al. (2010) give a strategy for varying ρ; it can work well in practice, but does not have convergence guarantees

Like deriving duals, transforming a problem into one that ADMM can handle is sometimes a bit subtle, since different forms can lead to different algorithms

Example: lasso regression

Given $y \in \mathbb{R}^n$, $X \in \mathbb{R}^{n \times p}$, recall the lasso problem:

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

We can rewrite this as:

$$\min_{\beta,\alpha} \ \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\alpha\|_1 \ \text{ subject to } \ \beta - \alpha = 0$$

ADMM steps:

$$\begin{aligned}
\beta^{(k)} &= (X^T X + \rho I)^{-1}\big(X^T y + \rho(\alpha^{(k-1)} - w^{(k-1)})\big) \\
\alpha^{(k)} &= S_{\lambda/\rho}(\beta^{(k)} + w^{(k-1)}) \\
w^{(k)} &= w^{(k-1)} + \beta^{(k)} - \alpha^{(k)}
\end{aligned}$$

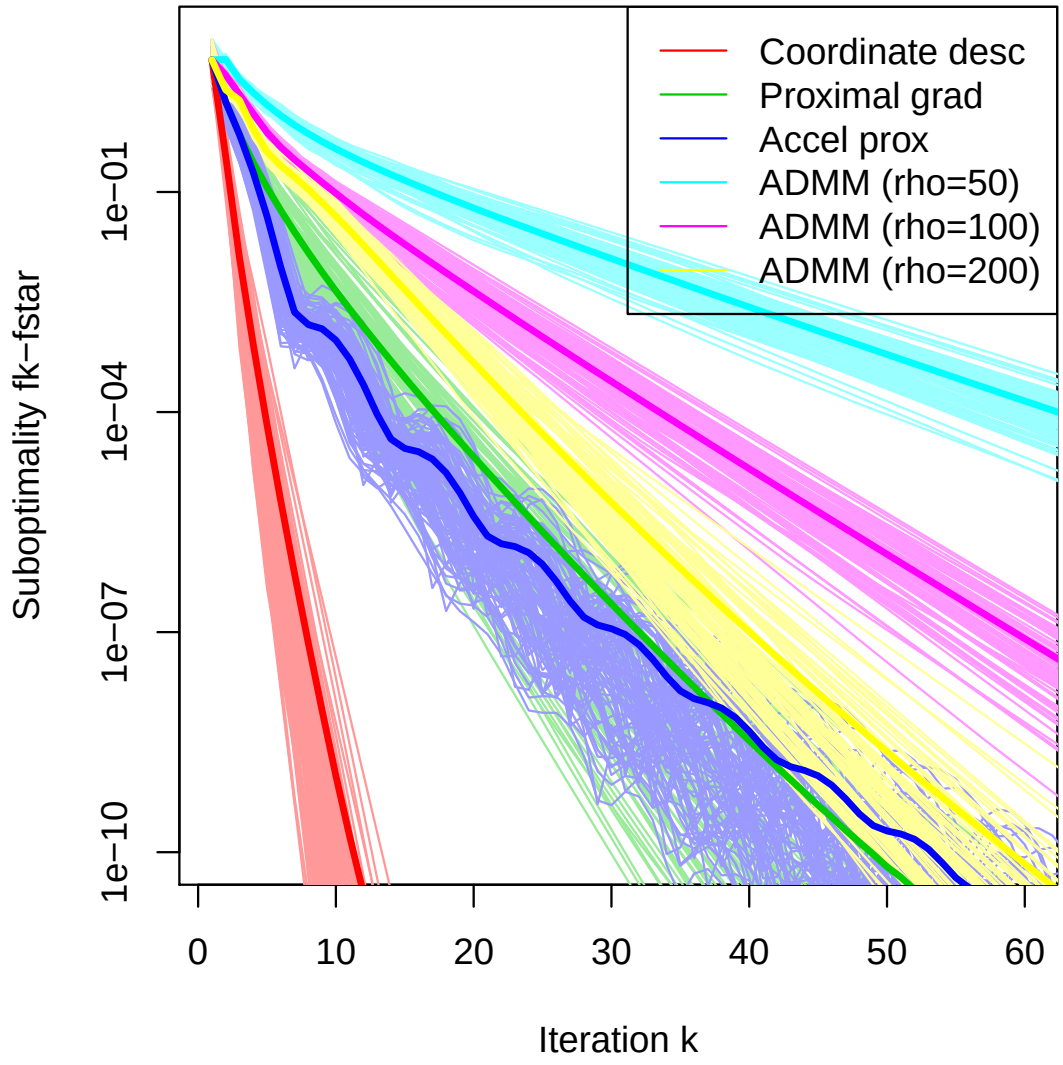
Notes:

- The matrix $X^T X + \rho I$ is always invertible, regardless of X
- If we compute a factorization (say Cholesky) in $O(p^3)$ flops, then each β update takes $O(p^2)$ flops
- The α update applies the soft-thresolding operator S_t, which recall is defined as

$$[S_t(x)]_j = \begin{cases} x_j - t & x > t \\ 0 & -t \leq x \leq t \\ x_j + t & x < -t \end{cases}, \quad j = 1, \ldots, p$$

- ADMM steps are "almost" like repeated soft-thresholding of ridge regression coefficients

Comparison of various algorithms for lasso regression: 100 random instances with $n = 200$, $p = 50$



Example: group lasso regression

Given $y \in \mathbb{R}^n$, $X \in \mathbb{R}^{n \times p}$, recall the group lasso problem:

$$\min_{\beta} \ \frac{1}{2}\|y - X\beta\|_2^2 + \lambda \sum_{g=1}^{G} c_g \|\beta_g\|_2$$

Rewrite as:

$$\min_{\beta,\alpha} \ \frac{1}{2}\|y - X\beta\|_2^2 + \lambda \sum_{g=1}^{G} c_g \|\alpha_g\|_2 \ \text{ subject to } \ \beta - \alpha = 0$$

ADMM steps:

$$\begin{aligned}
\beta^{(k)} &= (X^T X + \rho I)^{-1}\big(X^T y + \rho(\alpha^{(k-1)} - w^{(k-1)})\big) \\
\alpha_g^{(k)} &= R_{c_g \lambda/\rho}\big(\beta_g^{(k)} + w_g^{(k-1)}\big), \quad g = 1, \ldots, G \\
w^{(k)} &= w^{(k-1)} + \beta^{(k)} - \alpha^{(k)}
\end{aligned}$$

Notes:

- The matrix $X^T X + \rho I$ is always invertible, regardless of X
- If we compute a factorization (say Cholesky) in $O(p^3)$ flops, then each β update takes $O(p^2)$ flops
- The α update applies the group soft-thresolding operator R_t, which recall is defined as

$$R_t(x) = \left(1 - \frac{t}{\|x\|_2}\right)_+ x$$

- Similar ADMM steps follow for a sum of arbitrary norms of as regularizer, provided we know prox operator of each norm
- ADMM algorithm can be rederived when groups have overlap (hard problem to optimize in general!). See Boyd et al. (2010)

Example: sparse subspace estimation

Given $S \in \mathbb{S}_p$ (typically $S \succeq 0$ is a covariance matrix), consider the sparse subspace estimation problem (Vu et al., 2013):

$$\max_Y \ \mathrm{tr}(SY) - \lambda \|Y\|_1 \ \ \text{subject to} \ \ Y \in \mathcal{F}_k$$

where $\mathcal{F}_k$ is the Fantope of order k, namely

$$\mathcal{F}_k = \{Y \in \mathbb{S}^p : 0 \preceq Y \preceq I, \ \mathrm{tr}(Y) = k\}$$

Note that when $\lambda = 0$, the above problem is equivalent to ordinary principal component analysis (PCA)

This above is an SDP and in principle solveable with interior-point methods, though these can be complicated to implement and quite slow for large problem sizes

Rewrite as:

$$\min_{Y,Z} \; -\mathrm{tr}(SY) + I_{\mathcal{F}_k}(Y) + \lambda \|Z\|_1 \;\; \text{subject to} \;\; Y = Z$$

ADMM steps are:

$$\begin{aligned} Y^{(k)} &= P_{\mathcal{F}_k}(Z^{(k-1)} - W^{(k-1)} + S/\rho) \\ Z^{(k)} &= S_{\lambda/\rho}(Y^{(k)} + W^{(k-1)}) \\ W^{(k)} &= W^{(k-1)} + Y^{(k)} - Z^{(k)} \end{aligned}$$

Here $P_{\mathcal{F}_k}$ is Fantope projection operator, computed by clipping the eigendecomposition $A = U\Sigma U^T$, $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_p)$:

$$P_{\mathcal{F}_k}(A) = U\Sigma_\theta U^T, \quad \Sigma_\theta = \mathrm{diag}(\sigma_1(\theta), \ldots, \sigma_p(\theta))$$

where each $\sigma_i(\theta) = \min\{\max\{\sigma_i - \theta, 0\}, 1\}$, and $\sum_{i=1}^p \sigma_i(\theta) = k$

Example: sparse + low rank decomposition

Given $M \in \mathbb{R}^{n \times m}$, consider the sparse plus low rank decomposition problem (Candes et al., 2009):

$$\begin{array}{ll} \min\limits_{L,S} & \|L\|_{\mathrm{tr}} + \lambda \|S\|_1 \\ \text{subject to} & L + S = M \end{array}$$

ADMM steps:

$$\begin{aligned} L^{(k)} &= S^{\mathrm{tr}}_{1/\rho}(M - S^{(k-1)} + W^{(k-1)}) \\ S^{(k)} &= S^{\ell_1}_{\lambda/\rho}(M - L^{(k)} + W^{(k-1)}) \\ W^{(k)} &= W^{(k-1)} + M - L^{(k)} - S^{(k)} \end{aligned}$$

where, to distinguish them, we use $S^{\mathrm{tr}}_{\lambda/\rho}$ for matrix soft-thresolding and $S^{\ell_1}_{\lambda/\rho}$ for elementwise soft-thresolding

Example from Candes et al. (2009):

(a) Original frames (b) Low-rank $\hat{L}$ (c) Sparse $\hat{S}$

Consensus ADMM

Consider a problem of the form: $\min_x \sum_{i=1}^{B} f_i(x)$

The consensus ADMM approach begins by reparametrizing:

$$\min_{x_1,\ldots,x_B,x} \sum_{i=1}^{B} f_i(x_i) \text{ subject to } x_i = x,\ i = 1,\ldots B$$

This yields the decomposable ADMM steps:

$$x_i^{(k)} = \underset{x_i}{\operatorname{argmin}}\ f_i(x_i) + \frac{\rho}{2}\|x_i - x^{(k-1)} + w_i^{(k-1)}\|_2^2, \quad i = 1,\ldots,B$$

$$x^{(k)} = \frac{1}{B}\sum_{i=1}^{B}\left(x_i^{(k)} + w_i^{(k-1)}\right)$$

$$w_i^{(k)} = w_i^{(k-1)} + x_i^{(k)} - x^{(k)}, \quad i = 1,\ldots,B$$

Write $\bar{x} = \frac{1}{B}\sum_{i=1}^B x_i$ and similarly for other variables. Not hard to see that $\bar{w}^{(k)} = 0$ for all iterations $k \geq 1$

Hence ADMM steps can be simplified, by taking $x^{(k)} = \bar{x}^{(k)}$:

$$x_i^{(k)} = \underset{x_i}{\operatorname{argmin}} \; f_i(x_i) + \frac{\rho}{2}\|x_i - \bar{x}^{(k-1)} + w_i^{(k-1)}\|_2^2, \quad i = 1, \ldots, B$$

$$w_i^{(k)} = w_i^{(k-1)} + x_i^{(k)} - \bar{x}^{(k)}, \quad i = 1, \ldots, B$$

To reiterate, the x_i, $i = 1, \ldots, B$ updates here are done in parallel

Intuition:

- Try to minimize each $f_i(x_i)$, use (squared) ℓ_2 regularization to pull each x_i towards the average $\bar{x}$
- If a variable x_i is bigger than the average, then w_i is increased
- So the regularization in the next step pulls x_i even closer

General consensus ADMM

Consider a problem of the form: $\min_x \sum_{i=1}^B f_i(a_i^T x + b_i) + g(x)$

For consensus ADMM, we again reparametrize:

$$\min_{x_1,\ldots,x_B,x} \sum_{i=1}^B f_i(a_i^T x_i + b_i) + g(x) \text{ subject to } x_i = x,\ i = 1,\ldots B$$

This yields the decomposable ADMM updates:

$$x_i^{(k)} = \operatorname*{argmin}_{x_i}\ f_i(a_i^T x_i + b_i) + \frac{\rho}{2}\|x_i - x^{(k-1)} + w_i^{(k-1)}\|_2^2,$$

$$i = 1,\ldots,B$$

$$x^{(k)} = \operatorname*{argmin}_{x}\ \frac{B\rho}{2}\|x - \bar{x}^{(k)} - \bar{w}^{(k-1)}\|_2^2 + g(x)$$

$$w_i^{(k)} = w_i^{(k-1)} + x_i^{(k)} - x^{(k)}, \quad i = 1,\ldots,B$$

Notes:

- It is no longer true that $\bar{w}^{(k)} = 0$ at a general iteration k, so ADMM steps do not simplify as before
- To reiterate, the x_i, $i = 1, \ldots, B$ updates are done in parallel
- Each x_i update can be thought of as a loss minimization on part of the data, with ℓ_2 regularization
- The x update is a proximal operation in regularizer g
- The w update drives the individual variables into consensus
- A different initial reparametrization will give rise to a different ADMM algorithm

See Boyd et al. (2010), Parikh and Boyd (2013) for more details on consensus ADMM, strategies for splitting up into subproblems, and implementation tips

References and further reading

Part A:

- S. Boyd and L. Vandenberghe (2004), "Convex optimization", Chapters 9 and 10
- Y. Nesterov (1998), "Introductory lectures on convex optimization: a basic course", Chapter 2
- Y. Nesterov and A. Nemirovskii (1994), "Interior-point polynomial methods in convex programming", Chapter 2
- J. Nocedal and S. Wright (2006), "Numerical optimization", Chapters 6 and 7
- L. Vandenberghe, Lecture notes for EE 236C, UCLA, Spring 2011-2012

Part B:

- S. Boyd and L. Vandenberghe (2004), "Convex optimization", Chapter 11
- A. Nemirovski (2004), "Interior-point polynomial time methods in convex programming", Chapter 4
- J. Nocedal and S. Wright (2006), "Numerical optimization", Chapters 14 and 19
- J. Renegar (2001), "A mathematical view of interior-point methods"
- S. Wright (1997), "Primal-dual interior-point methods," Chapters 5 and 6

Part C:

- D. Bertsekas and J. Tsitsiklis (1989), "Parallel and distributed domputation: numerical methods", Chapter 3
- Z. Luo and P. Tseng (1992), "On the convergence of the coordinate descent method for convex differentiable minimization"
- P. Tseng (2001), "Convergence of a block coordinate descent method for nondifferentiable minimization"

Part D:

- S. Boyd and N. Parikh and E. Chu and B. Peleato and J. Eckstein (2010), "Distributed optimization and statistical learning via the alternating direction method of multipliers"
- N. Parikh and S. Boyd (2013), "Proximal algorithms"