Convex Optimization

A light-speed introduction

Stephen Becker

Applied & Computational Mathematics
California Institute of Technology

October 23, 2009

Outline

Outline

Why optimization?

Why optimization?

- ACM: PDE, geometry, signal processing, *optimization*
- *convex* optimization, relaxations.
- *non-convex* optimization: heuristics (e.g. genetic algorithms)

Why this seminar?

- Taught haphazardly: high school (Lagrange multipliers, Newton's Method), college, grad school
- Field changing quickly

Definitions

Convexity

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y), \quad \forall\ 0 \leq \alpha \leq 1$$

If f is differentiable, then an equivalent definition is:

Convexity

$$f(x+h) \geq f(x) + \langle \nabla f(x), h \rangle$$

If f is twice differentiable, then an equivalent definition is:

Convexity

$$\nabla^2 f(x) \succeq 0$$

Subgradient and subderivative

Recall if f is differentiable and convex, then

Convexity

$$f(x+h) \geq f(x) + \langle \nabla f(x), h \rangle$$

This means the first order Taylor expansion is an *understimate* of the function. This is global, not just local!
If a vector g satisfies the following equation

Subgradient

$$f(x+h) \geq f(x) + \langle g, h \rangle$$

then we say that g is a subgradient of f (at x). If there is only one such g, then $g = \nabla f(x)$ and f is differentiable at this point x. The subdifferential of f at x, written $\partial f(x)$, is the set of all subgradients of f at x.

Optimization problem

Optimization

$$\min_x f_0(x)$$

subject to

$$f_i(x) \leq 0 \quad \forall i = 1, \ldots, p \quad \text{inequality constraints}$$

$$h_j(x) = 0 \quad \forall j = p+1, \ldots, m \quad \text{equality constraints}$$

f_0 is known as the objective, and often written as just f.

Assume f is a functional, i.e. $f : \mathbb{R}^n \to \mathbb{R}$

Convex Optimization problem

A *convex optimization* problem requires two things:

1. The *objective* f_0 is a convex function
2. The *inequality* constraints f_i are convex functions

Note that an equality constraint $h(x) = 0$ is the same as two inequality constraints. So h and $-h$ are both convex. So, h is affine.

Write $h(x) = 0$ as $Ax - b = 0$.

For concave maximization, what changes?

(NEW) Why convexity?

Consider

$$\min_x f(x) \quad \text{subject to} \quad x \in C$$

where both the function f and the set C are convex.

Theorem (All local minima are also global minima)

Proof: Let $x^\star$ be a local minimum, so there is some $\varepsilon > 0$ such that $f(y) \geq f(x^\star)$ for all $y \in B_\varepsilon(x^\star) \cap C$. Now suppose $x^\star$ is not a global minima, so there is some point $z \in C$ with $f(z) < f(x^\star)$. Then pick $\alpha > 0$ small enough so that $w \equiv \alpha z + (1-\alpha)x^\star$ is in $B_\varepsilon(x^\star)$. By convexity of C, we have $w \in C$. Then by convexity of f, we have

$$f(w) \leq \alpha f(z) + (1-\alpha) f(x^\star) < f(x^\star)$$

a contradiction to the fact $x^\star$ is a local minimum.

If $C = \{x : f_i(x) \leq 0,\ i = 1, \ldots, m\}$, then C is convex if f_i is convex for all i.

Conic programming, standard form

Conic Programming in standard form

$$\min c^T x \quad \text{subject to} \quad Ax = b, \quad x \in \mathcal{K}$$

Linear programming: $K = \mathbb{R}^n_+$, so $x \geq 0$. (what does this mean?)

Semidefinite programming: $K = S^n_+$, so x is positive semidefinite.

Second-order cone programming: $K = \{(y, t) : \|y\|_2 \leq t\}$ (Quadratic programming)

Tricks

equivalent problems are not identical

- slack variables

$$f_i(x) \leq 0 \iff f_i(x) + s_i = 0, \quad s_i \geq 0$$

- eliminate equality constraint (null space; discussed later)
  or, write as two inequalities (seldom useful in practice)
- introduce equality constraint

$$\min \|Ax - b\|_2 \iff \min \|r\|_2, \quad r = Ax - b$$

- equivalent differentiable form

$$\min \|Ax - b\|_2 \iff \min \|Ax - b\|_2^2$$

- epigraph

$$\min \|x\|_1 \iff \min t, \quad \|x\|_1 \leq t$$

Pervasive example: linear inverse problem

- Unknown signal $x^\star \in \mathbb{R}^n$
- Linear measurements $b = Ax^\star + z$, A is a m x n matrix.
- If A is *overcomplete* or invertible, then a good idea might be least squares: $\min_x \|b - Ax\|$.
- If A is *underdetermined*, ill-posed (because we don't have enough information)
- Need a prior assumption on $x^\star$ to reduce the degrees of freedom

Basis Pursuit

$$\min_x \|x\|_1 \quad \text{subject to} \quad Ax = b$$

Pervasive example: linear inverse problem

The exact form of the example that we will use is a simpler version. Most compressed sensing algorithms solve this version:

Basis Pursuit (Lagrangian relaxation)

$$\min_x \|x\|_1 + \frac{\mu}{2}\|Ax - b\|_2^2$$

Write $f(x) = \|x\|_1 + \frac{\mu}{2}\|Ax - b\|_2^2$. Then

$$\partial f(x) = \partial\|x\|_1 + \mu A^T(Ax - b)$$

Calculating $\partial\|x\|_1$ is fast ($\mathcal{O}(n)$), so the dominant cost is multiplying A and A^T times a vector. For a general matrix A, not using Strassen multiplication or anything fancy, the cost of Ax is $\mathcal{O}(mn)$. However, if A has a fast transform like the FFT, the cost is more like $\mathcal{O}(n \log n)$.

Outline

The Lagrangian and dual function

Consider the following problem (need not be convex)

$$\min_x f_0(x) \quad \text{subject to} \quad f_i(x) \leq 0 \ \forall i = 1, \ldots, m, \quad Ax = b$$

Let $p^\star$ denote the optimal value, $p^\star = f_0(x^\star)$.

Lagrangian

$$\mathcal{L}(x, \lambda, \nu) = f_0(x) + \sum_i \lambda_i f_i(x) + \nu^T (Ax - b)$$

Dual function

$$g(\lambda, \nu) = \inf_x \mathcal{L}(x, \lambda, \nu)$$

Weak duality and the dual problem

Pick an $\lambda \geq 0$ and any ν.

$$\begin{aligned} g(\lambda, \nu) &= \inf_x \left(f_0(x) + \sum_i \lambda_i f_i(x) + \nu^T (Ax - b) \right) \\ &\leq \inf_{x \text{ feasible}} \left(f_0(x) + \sum_i \lambda_i f_i(x) + \nu^T (Ax - b) \right) \\ &\leq \inf_{x \text{ feasible}} f_0(x) = p^\star \end{aligned}$$

So $\lambda \geq 0$ and ν give a bound on $p^\star$. To find the *best* bound, we can solve...

Dual problem

$$\max_{\lambda \geq 0} g(\lambda, \nu)$$

Weak duality and strong duality

Fact: dual problem is convex, even if primal isn't.

Let $d^\star$ be the value of to

$$\max_{\lambda \geq 0} g(\lambda, \nu)$$

We just saw that $d^\star \leq p^\star$. If in fact $d^\star = p^\star$, we call this strong duality. This is VERY useful. For convex problems, we *usually* have strong duality. Prove via constraint qualificiation.

Slater condition (for a convex problem)

If there is a point x that is *strictly* feasible, i.e. $f_i(x) < 0 \, \forall i = 1, \ldots, m$, then strong duality holds

KKT conditions, (1)

Assume strong duality holds. Let $x^\star$ be the optimal primal point, so $f_i(x^\star) \leq 0$ and $Ax^\star - b = 0$.

$$\begin{aligned} f_0(x^\star) &= g(\lambda^\star, \nu^\star) \\ &= \inf_x \left(f_0(x) + \sum_k \lambda_i^\star f_i(x) + \nu^\star (Ax - b) \right) \\ &\leq f_0(x^\star) + \sum_k \lambda_i^\star f_i(x^\star) + \nu^\star (Ax^\star - b) \\ &\leq f_0(x^\star) \end{aligned}$$

Hence must be equalities. Hence $\lambda_i^\star f_i(x^\star) = 0$ for all i. So this is a necessary condition. Called complementary slackness. *Only for inequality constraints!*
Also, this proves $x^\star = \text{argmin}_x \mathcal{L}(x, \lambda^\star, \nu^\star)$.

KKT conditions, (2)

Recall $x^\star = \text{argmin}_x \mathcal{L}(x, \lambda^\star, \nu^\star)$. So, by necesary first-order condition (assuming f_0 and f_i differentiable),

$$\begin{aligned} 0 &= \nabla_x \mathcal{L}(x^\star, \lambda^\star, \nu^\star) \\ &= \nabla_x f_0(x^\star) + \sum_i \lambda_i^\star \nabla_x f_i(x^\star) + A^T \nu^\star \end{aligned}$$

Called stationarity condition.
The other two KKT conditions are obvious:

- primal feasibility $Ax^\star = b$ and $f_i(x^\star) \leq 0$
- dual feasibility $\lambda^\star \geq 0$, no constraint on $\nu^\star$

Fact: if problem is convex (and differentiable), KKT are also *sufficient*.
Proof: Lagrangian is convex, so stationarity condition implies $x^\star$ is the global minimizer. Using same setup as last page, we see the duality gap is zero.

KKT conditions, inequality constraints only

Consider a 1D problem,

$$\min f(x) \quad \text{subject to} \quad a \leq x \leq b$$

Assume f differentiable.

Then KKT conditions reduce to the following:

either $f'(x^\star) = 0$, or $x = a$, or $x = b$

which should look familiar.

KKT conditions, equality constraints only

Assume equality constraints $Ax = b$ but no inequality constraints.
The 4 KKT conditions are now just 2:

$$0 = \nabla_x f_0(x^\star) + A^T \nu^\star$$

$$Ax^\star = b$$

This is the "Lagrange Multipliers" technique you may have learned in your first calculus class.

Lagrange Multipliers: connections with your past

Remember this from calculus?

Lagrange Multipliers (one constraint, 2D)

Let $f(x, y)$ and $g(x, y)$ be functions with continuous first-order partial derivatives. If the maximum (or minimum) value of f subject to the condition $g(x, y) = 0$ occurs at a point P where $\nabla g(P) \neq 0$, then

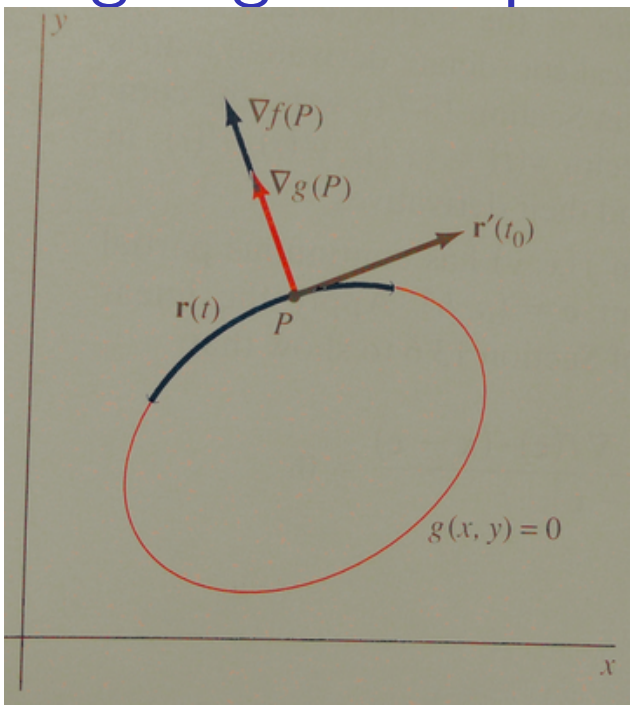
$$\nabla f(P) = \lambda \nabla g(P)$$

for some constant λ.

Edwards & Penney, 5th ed., *Calculus, with analytic geometry; Early Transcendentals*, Prentice Hall 1997.

Extension to 2 constraints, and 3D.

Lagrange Multipliers: connections with your past



Proof: By the implicit function theorem, and the nonzero gradient of g at P, we can represent the curve $g(x, y) = 0$ by a parametric curve $r(t)$ (locally, near P), such that r has a nonzero tangent vector r' near P. Let t_0 be the value such that $r(t_0) = P$.
Let $f(x, y)$ attain its max value at P.

If we parameterize f by $r(t)$, then $f(r(t))$ attains its maximum value at t_0, so $\partial_t f(r(t))$ is zero at $t = t_0$. This gives, by the vector chain rule,

$$\nabla f(P) \cdot r'(t_0) = 0$$

Since r is defined such that $g(r(t)) \equiv 0$, then $\partial_t g(r(t)) \equiv 0$, so

$$\nabla g(P) \cdot r'(t_0) = 0$$

Thus both $\nabla f(P)$ and $\nabla g(P)$ are perpendicular to the nonzero vector $r'(t_0)$, which means they must be linearly dependent.

(NEW) Euler Conditions

Consider a convex problem with differentiable objective f and feasible set C. Then

Theorem (Euler condition)

$x^\star$ is optimal if and only if $x^\star \in C$ and

$$\nabla f(x^\star)^T (y - x^\star) \geq 0 \quad \forall y \in D$$

i.e. $-\nabla f(x^\star)$ defines a supporting hyperplane to C.

Unconstrained minimization

$x^\star$ is optimal if and only if

$$\nabla f(x^\star) = 0$$

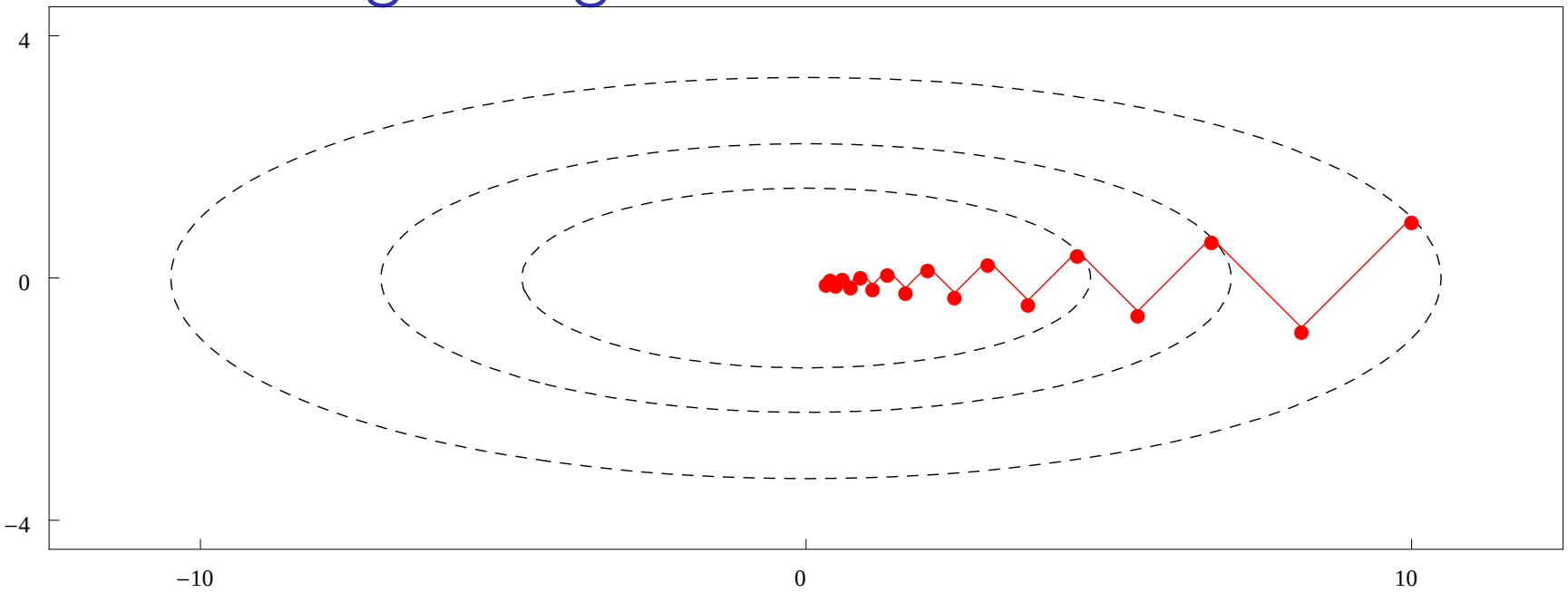
Equality constraints, $Ax = b$

$\nabla f(x^\star) \perp \mathcal{N}(A)$, or

$$\nabla f(x^\star) \in \mathcal{R}(A^T)$$

Outline

What's wrong with gradient descent?



Boyd & Vandenberghe

- $\min_x \frac{1}{2}(x^2 + \gamma y^2)$
- Gradient Descent

$$x_{k+1} = x_k - t_k \nabla f(x_k)$$

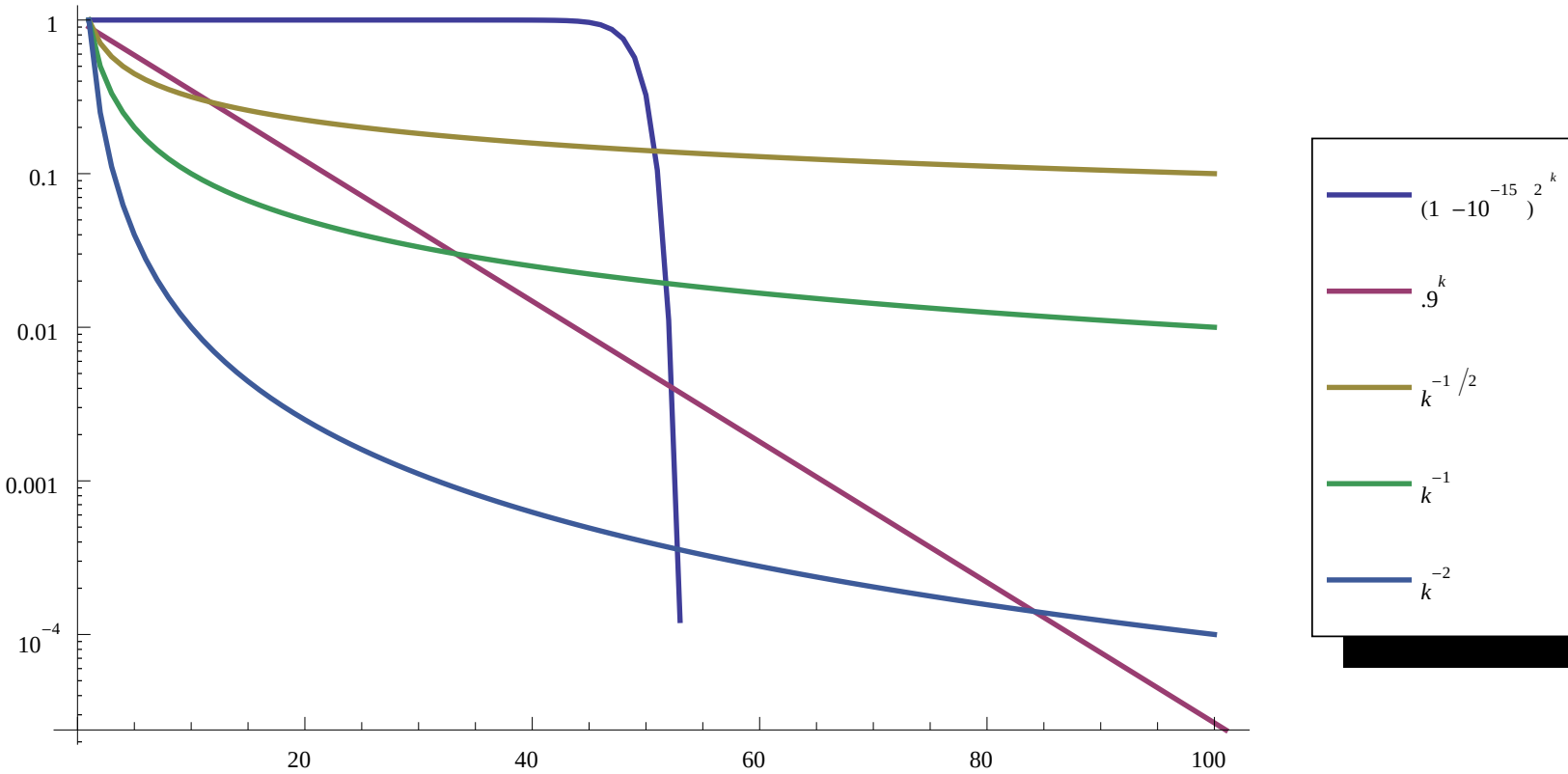
- Convergence rate (assume ∇f is Lipshitz)

$$f(x_k) - f(x^\star) \leq \frac{C}{k}$$

Convergence Rates

e_k is error at kth iteration

- (Q) linear convergence: $\frac{e_{k+1}}{e_k} \leq \beta$ eventually, i.e. $\lim_{k\to\infty} e_k = \beta^k$
  $\beta = 0$ is *superlinear*, $\beta = 1$ is *sublinear*.
- Quadratic convergence: $\frac{e_{k+1}}{e_k^2} \leq C$



Convergence Rates

ε-solution: x feasible and $f(x) - f(x^\star) \leq \varepsilon$

- $e_k = \frac{1}{\sqrt{k}}$. ε-solution in $\frac{1}{\varepsilon^2}$ iterations. Sub-linear convergence.
- $e_k = \frac{1}{k}$. ε-solution in $\frac{1}{\varepsilon}$. Sub-linear.
- $e_k = \frac{1}{k^2}$. ε-solution in $\frac{1}{\sqrt{\varepsilon}}$. Sub-linear.
- $e_k = \beta^k$. ε-solution in $\frac{\log(1/\varepsilon)}{\log(1/\beta)}$. Linear.
- $e_k = \frac{1}{k}^k$. Super-linear convergence, but not quadratic.
- $e_k = \beta^{2^k}$. ε-solution in $C \log(\log(1/\varepsilon))$
  $\log_{10}(\log_{10}(\frac{1}{10^{-100}})) = 2$. Quadratic convergence.

Convergence rate of first-order methods

(sub)-gradient descent

$$x_{k+1} = x_k - t_k d$$

$d = \nabla f(x_k)$ or $d \in \partial f(x_k)$
$\mathcal{O}(\frac{1}{k})$ for gradient descent, $\mathcal{O}(\frac{1}{\sqrt{k}})$ for sub-gradient descent
Can we do better?

Nesterov's optimal method

$$x_k = y_{k-1} - t_k \nabla f(y_{k-1})$$
$$y_k = x_k + \frac{k}{k+3}(x_k - x_{k-1})$$

Many variants Nesterov '83,'04,'05; Auslender, Teboulle '06; Tseng '08; Lan, Lu, Monteiro '08; Beck, Teboulle '09
Need f differentiable, with ∇f Lipshitz.
$\mathcal{O}(\frac{1}{k^2})$ convergence!

First order methods with constraints

Can sometimes handle very easily by projecting point x onto the feasible set Q via $\mathcal{P}_Q$. Might destroy some proofs of convergence. Projection may not need to be exact

gradient descent

$$x_{k+1} = x_k - t_k \nabla f(x_k)$$

projected gradient descent

$$x_{k+1} = \mathcal{P}_Q(x_k - t_k \nabla f(x_k)$$

or

$$x_{k+1} = x_k + s_k(\mathcal{P}_Q(x_k - t_k \nabla f(x_k)) - x_k)$$

Have accelerated versions with $\mathcal{O}(\frac{1}{k^2})$ convergence

Basic Newton's Method

Maybe you remember this from high school as a technique to find the intercept of a function, i.e. to solve $g(x) = 0$. In this "root-finding" mode, Newton's method just forms a linearization to f and solves this linearization:

$$g(x + h) \simeq g(x) + g'(x)h$$

Given a current point x, the next point $x + h$ is calculated so that $g(x) + g'(x)h = 0$, which means $h = -g(x)/g'(x)$.

Basic Newton's Method

Now suppose we want to minimize a function $f(x)$. If there are no constraints, and if f is differentiable, this is equivalent to finding all the roots of f'. So, if f also has a second derivative, then we can apply Newton's method to $g = f'$. Specifically, we move to the new point $x + h$ where $h = -f'(x)/f''(x)$. Because we use the second derivative, this is known as a second-order method.

In general, x is a vector, in which case we solve the equation

Newton's Method

$$h_{Newton} = -(\nabla^2 f(x))^{-1} \nabla f(x)$$

and update $x \leftarrow x + h$. Repeat until convergence. Another way to think about this: we are minimizing the second-order Taylor series of f:

$$f(x + h) \simeq f(x) + \langle \nabla f(x), h \rangle + \frac{1}{2} \langle h, \nabla^2 f(x) h \rangle$$

Basic Newton's Method

There are several implications to Newton's method. The general principle is that the Hessian information is very useful, but it comes at a price.

- The major cost of each iteration is inverting the Hessian (or, more accurately, solving a linear system). If the Hessian has no special structure or sparsity, this costs $\mathcal{O}(n^3)$ operations, which is slow (we need to do this at every iteration).
- However, we will not need many iterations. Newton's method enjoys local quadratic convergence. This basically means that once x_k is close enough to the solution, the error $f(x_k) - f(x^\star)$ will become an *order of magnitude smaller* at every iteration.
- Newton's method is at the heart of Interior Point Methods.

Quasi-Newton Methods

Think: secant method, i.e. use finite differences to approximate a derivative.

- Idea: approximate exact Hessian $\nabla^2 f$ with H_k, using gradient information
- $H_k = LI$ gives first-order method
- Need to invert H_k every step. Update to H_k is probably simple.
- Brilliant idea: keep track of H_k^{-1}, and update this quantity. Woodbury formula: rank 1 update to inverse is easy
- Best implementation: BFGS. For large systems, use L-BFGS

Convergence rules-of-thumb:

- First-order method: linear convergence at best. Use Nesterov acceleration.
- Quasi-newton method: superlinear convergence sometimes
- Newton method: quadratic convergence, if close to solution

Outline

Equality constraints vs inequality constraints

There are nice tricks for converting equality constraints to inequality constraints, and vice-versa. In general, inequality constraints may be trickier, and/or it may be easier to project onto a set defined by equality constraints. But really, it's problem dependent.

Basic form of equality constrained minimization

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

Basic form of inequality problem

$$\min_x f(x) \quad \text{subject to} \quad g(x) \leq 0$$

f and g convex (if we write $g(x) \geq 0$, then g concave). Easy to extend to more than one g.

Method 1: adapt Newton's method; equality constraints only

Adapt Newton's method to incorporate $Ax = b$. Define $\hat{f}$ to be the second-order Taylor approximation of f about x:

$$\hat{f}(x+h) = f(x) + \langle \nabla f(x), h \rangle + \frac{1}{2}\langle h, \nabla^2 f(x) h \rangle$$

Then we solve the following problem

$$\min_h \hat{f}(x+h) \quad \text{subject to} \quad A(x+h) = b$$

If we assume that $Ax = b$ already, then the constraint is just $Ah = 0$. Turns out we can solve this minimization problem exactly.

Method 1: adapt Newton's method; equality constraints only

The Lagrangian is

$$\mathcal{L}(h, \nu) = \hat{f}(x + h) + \nu^T (Ah)$$

so the KKT conditions reduce to the following two equations

$$\nabla_h \mathcal{L} \equiv \nabla f(x) + \langle \nabla^2 f(x), h \rangle + A^T \nu = 0 \qquad \text{(stationarity)} \quad (1)$$

$$Ah = 0 \qquad \text{(primal feasibility)} \quad (2)$$

This is a system of linear equations, so it can be solved easily (though it may be expensive). If we had no constraints, so A is gone, then this reduces to solving $h = -\nabla^2 f(x)^{-1} \nabla f(x)$, i.e. basic Newton's Method.

Method 2: trickery; equality constraints only

To enforce the constraint $Ax = b$, we find a matrix G such that $AG = 0$ and the columns of G span the null space of A. In *coding theory*, this is called a *parity-check* matrix. Now, borrowing terminology from PDE, let $\hat{x}$ be a *particular solution*, meaning that it is any solution to $A\hat{x} = b$ (there are an infinite choice of these). Thus, for any feasible x that satisfied $Ax = b$, we can uniquely write

$$x = \hat{x} + Gy$$

for some y. The minimization problem is now unconstrained:

Null space method for equality constraints

$$\min_y f(\hat{x} + Gy)$$

Of course, $f(\hat{x} + Gy)$ could be difficult to work with, or it may be hard to find G...

Method 3: solve the dual, equality constraints

[the dual function is usually g, but I am using g for the inequality constraint, so use G for the dual]
The dual problem of $\min_x f(x)$ subject to $Ax = b$ is

$$\max_{\nu} G(\nu)$$

where

$$G(\nu) = -b^T \nu + \inf_x (f(x) + \nu^T A x)$$

which can be expressed in terms of the conjugate function
$f^*(z) \equiv \sup_x z^T x - f(x)$.
Caveats:

- f smooth doesn't imply G smooth
- even if we have a dual optimal point $\nu^\star$, it's not always possible to recover the primal optimal point $x^\star$
- it may be hard or impossible to calculate f^*.

Method 3: solve the dual, inequality constraints

[the dual function is usually g, but I am using g for the inequality constraint, so use G for the dual]
The dual problem of $\min_x f(x)$ subject to $g(x) \leq 0$ is

$$\max_{\lambda} G(\lambda) \quad \text{subject to} \quad \lambda \geq 0$$

Advantages:

- Projecting x onto $\{x : g(x) \leq 0\}$ may be difficult, but projecting λ onto $\{\lambda : \lambda \geq 0\}$ is easy.
- Dual problem may for some reason be easier to solve

Caveats: same as for the equality constrained version.
Can of course combine equality and inequality constraints with this method.

Method 4: log-barrier, inequality constraints

One of the great ideas from the last 20 years is the development of the log-barrier methods, which are a fundamental part of many IPM. The idea is simple: replace the inequality constraint $g(x) \leq 0$ with a term in the objective function that penalizes the objective whenever $g(x)$ is close to 0. If the penalizing term is chosen well (i.e. self-concordant), then the method has provably nice properties. Most choices involve the logarithm, hence the name *log-barrier*.

Log-barrier form of inequality problem

$$\min_x f(x) + t \log(-g(x))$$

A log-barrier IPM solves the above problem many times (usually, about 15 to 50 times), each time with a smaller value of t, using the previous solution as a *warm-start*. Note: x is *always* feasible, and interior.

Method 5a: penalty functions. Equality and inequality constraints

Similar to the log-barrier idea, we put a penalizing term into the objective. There are a few variants to this. Let our problem be

$$\min f(x) \quad \text{subject to} \quad Ax = b, \quad g(x) \leq 0$$

Variant a: Quadratic Penalty Method

Solve

$$\min f(x) + \frac{\mu}{2} \left(\|Ax - b\|_2^2 + (\lfloor g(x) \rfloor_+)^2 \right)$$

For a finite value of μ, the solution will likely be infeasible. So, solve several times, with $\mu \to \infty$.

If we have no inequality constraints, this is smooth, which is good.

Problems: (1) nonsmooth $\lfloor \cdot \rfloor_+$ term may cause problems. (2) Becomes ill-conditioned as $\mu \to \infty$. Note: x is *never* feasible.

Method 5a: penalty functions. Example

Basis Pursuit

$$\min_x \|x\|_1 \quad \text{subject to} \quad Ax = b$$

Basis Pursuit denoising

$$\min_x \|x\|_1 \quad \text{subject to} \quad \|Ax - b\|_2 \leq \varepsilon$$

To solve Basis Pursuit, we can solve this for $\mu \to \infty$:

Basis Pursuit (quadratic penalty)

$$\min_x \|x\|_1 + \frac{\mu}{2}\|Ax - b\|_2^2$$

If we want to solve Basis Pursuit denoising, then can still solve the same quadratic penalty, but now $\mu \to \mu_\varepsilon$ for some $\mu_\varepsilon < \infty$. If $\varepsilon \to 0$, then $\mu_\varepsilon \to \infty$. In general, $\mu(\varepsilon)$ not easy to compute.

Method 5b: exact penalty functions. Equality and inequality constraints

We use a different penalty function, this time with the property that there is some $\mu_0 < \infty$ such that the constrained and unconstrained versions are the same for $\mu > \mu_0$. So no longer need $\mu \to \infty$. Good.
Problem: in general, an exact penalty function is necessarily nonsmooth.
Ex.

Variant b: Exact Penalty Method

Solve

$$\min f(x) + \mu\big(\|Ax - b\|_1 + \lfloor g(x) \rfloor_+ \big)$$

Can also use $\|\cdot\|_\infty$ or $\|\cdot\|_2$ (not $\|\cdot\|_2^2$). If $\mu > \mu_0 = \max(\|\lambda^\star\|_\infty, \|\nu^\star\|_\infty)$, then it is exact.
Problems: (1) nonsmooth (2) μ_0 may be large, and is unknown *a priori*

Method 5c: Augmented Lagrangian. Equality constraints only

Want to solve equality constrainted problem

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

Motivation: The Lagrangian for this problem is

$$\mathcal{L}(x, \nu) = f(x) + \nu^T (Ax - b)$$

Strong duality is equivalent to the following saddle-point inequality:

$$\begin{aligned} d^\star = \max_\nu g(\nu) &= \max_\nu \min_x \mathcal{L}(x, \nu) \\ &= \min_x \max_\nu \mathcal{L}(x, \nu) \quad \text{by strong duality} \\ &= \min_{x: Ax = b} f(x) = p^\star \end{aligned}$$

since

$$\max_\nu \mathcal{L}(x, \nu) = \begin{cases} f(x) & Ax = b \\ +\infty & \text{else} \end{cases}$$

Method 5c: Augmented Lagrangian. Equality constraints only

if $\nu^\star$ is dual optimal, then by the previous equalities

$$\min_{x:Ax=b} f(x) = \max_{\nu} \min_{x} \mathcal{L}(x,\nu) = \min_{x} \mathcal{L}(x,\nu^\star)$$

So if we knew $\nu^\star$, we should solve this:

$$\min_{x} f(x) + (\nu^\star)^T (Ax - b)$$

But we don't know $\nu^\star$. Instead combine with the quadratic penalty method to solve:

$$\min_{x} f(x) + \underbrace{\nu^T (Ax-b)}_{\text{Lagrangian term}} + \underbrace{\frac{\mu}{2}\|Ax-b\|_2^2}_{\text{quadratic penalty term}}$$

ν and μ are updated dynamically

Method 5c: Augmented Lagrangian. Equality constraints only

Benefits:

- like quadratic penalty, it's smooth
- like the nonsmooth penalties, it's exact. Don't need $\mu \to \infty$

Why is it exact? Problem with quadratic penalty is that we never have feasibility for $\mu < \infty$. We have roughly

$$Ax - b = -\nu^\star/\mu$$

For Augmented Lagrangian, we get

$$Ax - b = -(\nu^\star - \nu)/\mu$$

which goes to zero much faster, if we update ν well.

Problems:

- need robust updating scheme for parameters
- can handle inequality constraints but not very naturally

Note: aka method of multipliers

Method 6: projection

For both equality and inequality constraints.
We keep the constraints and design a method around them. In these methods, the advantage is that the iterates of x are actually feasible. Usually, involves a projection, which may or may not be easy. Sometimes, don't need an exact projection.

The simplest method is a modification of gradient descent, called projected gradient descent.

We already discussed this. This is a huge class of algorithms.

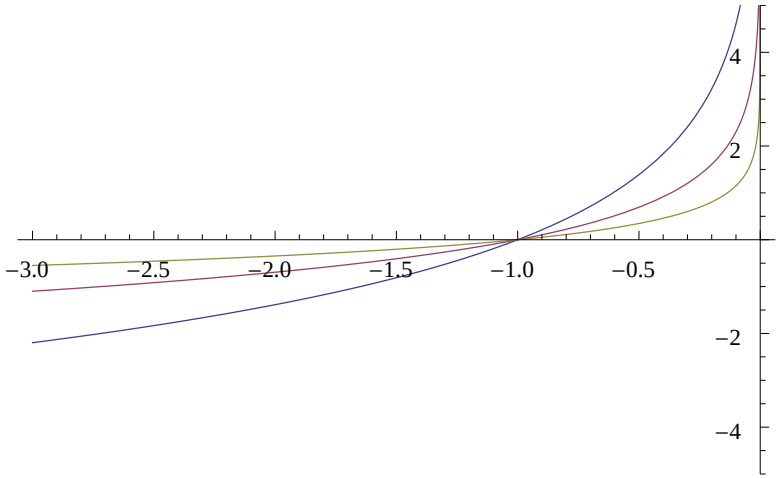
Outline

Interior Point Methods (IPM)

- Basic idea: apply Newton's method. We already saw that Newton's method can deal with equality constraints. So, now deal with inequalities somehow.
- Similar to the log-penalties we already discussed, but some subtleties. Can be motivated either by the log-penalty formulation or by trying to solve a (slightly modified) version of the KKT conditions.
- Have primal and primal-dual variants (and dual variants).
- Key property for analysis: self-concordance. Gives bounds on number of Newton steps. The log-penalty is self-concordant. Most common *cones* have known self-concordant penalties (usually involving a logarithm). Nesterov and Nemirovskii.
- IPM are both theoretically nice (polynomial complexity, building on Khachiyan '79 and Karmarker '84) and work well in practice (Mehrotra's predictor-corrector method, '89)
- IPM first in 1980s; theory matured around 1994.

Interior Point Methods

A barrier method (primal). Iterates stay feasible. Pick t large.



$$\min_x f(x) + \frac{1}{t}\Big(\underbrace{-log(-g(x))}_{\phi(x)}\Big)$$

such that $Ax = b$. ϕ is the log-barrier term. Equivalently, solve

$$\min_x \; t\, f(x) + \phi(x), \; Ax = b$$

Note: $\nabla\phi(x) = \frac{1}{-g(x)}\nabla g(x)$. KKT stationarity condition is

$$0 = \nabla\mathcal{L} = t\nabla f(x) + \frac{1}{-g(x)}\nabla g(x) + A^T\nu$$

If $x^\star$ is optimal, then $\lambda^\star = -\frac{1}{tg(x^\star)}$.

Interior Point Methods

A barrier method (primal).
So, we have primal and dual points. Get a duality gap $\frac{C}{t}$. Idea: solve for some t using Newton's method. Increase t and repeat. So, inner and outer iterations.
KKT interpretation:

$$\begin{aligned} Ax = b, g(x) \leq 0 \quad & \text{primal feasibility} \\ \lambda \geq 0 \quad & \text{dual feasibility} \\ \nabla f(x) + \lambda \nabla g(x) + A^T \nu = 0 \quad & \text{stationarity} \\ -\lambda g(x) = \frac{1}{t} \quad & \textit{centrality}\text{, formerly complementary slackness} \end{aligned}$$

It's a primal method, meaning we force $\lambda = \frac{-1}{tg(x)}$, as opposed to letting λ evolve on its own.

Interior Point Methods

A primal-dual method.

$$Ax = b,\ g(x) \leq 0,\ \lambda \geq 0$$

$$\nabla f(x) + \lambda \nabla g(x) + A^T \nu = 0 \quad \text{stationarity}$$

$$-\lambda g(x) = \frac{1}{t} \quad \textit{centrality}, \text{ formerly complementary slackness}$$

Don't force $\lambda = \frac{-1}{tg(x)}$. To compute update $(\Delta x, \Delta\lambda, \Delta\nu)$, linearize. Ignore $\lambda \geq 0$ and $g(x) \leq 0$ for now (will use line search later). Ignore all $\Delta x^T \Delta x$ and $\Delta\lambda\Delta x$ terms. Leads to a system like

$$\begin{pmatrix} \nabla^2 f(x) + \lambda \nabla^2 g(x) & \nabla g(x) & A^T \\ -\lambda \nabla g(x) & -g(x) & 0 \\ A & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta\lambda \\ \Delta\nu \end{pmatrix} = \begin{pmatrix} r_{\text{dual}} \\ r_{\text{cent}} \\ r_{\text{primal}} \end{pmatrix}$$

Basically, Newton's method. For linear programming, simplifies a lot.

Interior Point Methods

Differences between the primal log-barrier and the primal-dual methods we just introduced:

- Both methods: need line-search to keep feasible. Not a problem, because biased toward interior. Backtracking works.
- Log-barrier: solved outer loop to find a point on central path
- Primal-dual version has no inner-outer iteration. t is updated every step.
- Instead, primal-dual aims roughly for central path
- Primal-dual version has no exactly feasible dual point, so no exact duality gap (can approximate it)

Need to start feasible, i.e. in the interior. Can get around this, e.g. embedding into a larger problem. "Phase 1", "Phase 2"

Interior Point Methods

There are two kinds of distinctions (an algorithm can be in any of the four possible types):

1. *primal* vs *primal-dual*
2. *log-barrier* vs *path-following*

Method 1 was a primal log-barrier method ("log-barrier" is also known as "potential reduction").

Method 2 was a primal-dual path-following method.

There are also primal-dual log-barrier methods. Use a log-barrier like

$$\phi(x) = -\log(-g(x)\lambda)$$

Interior Point Methods

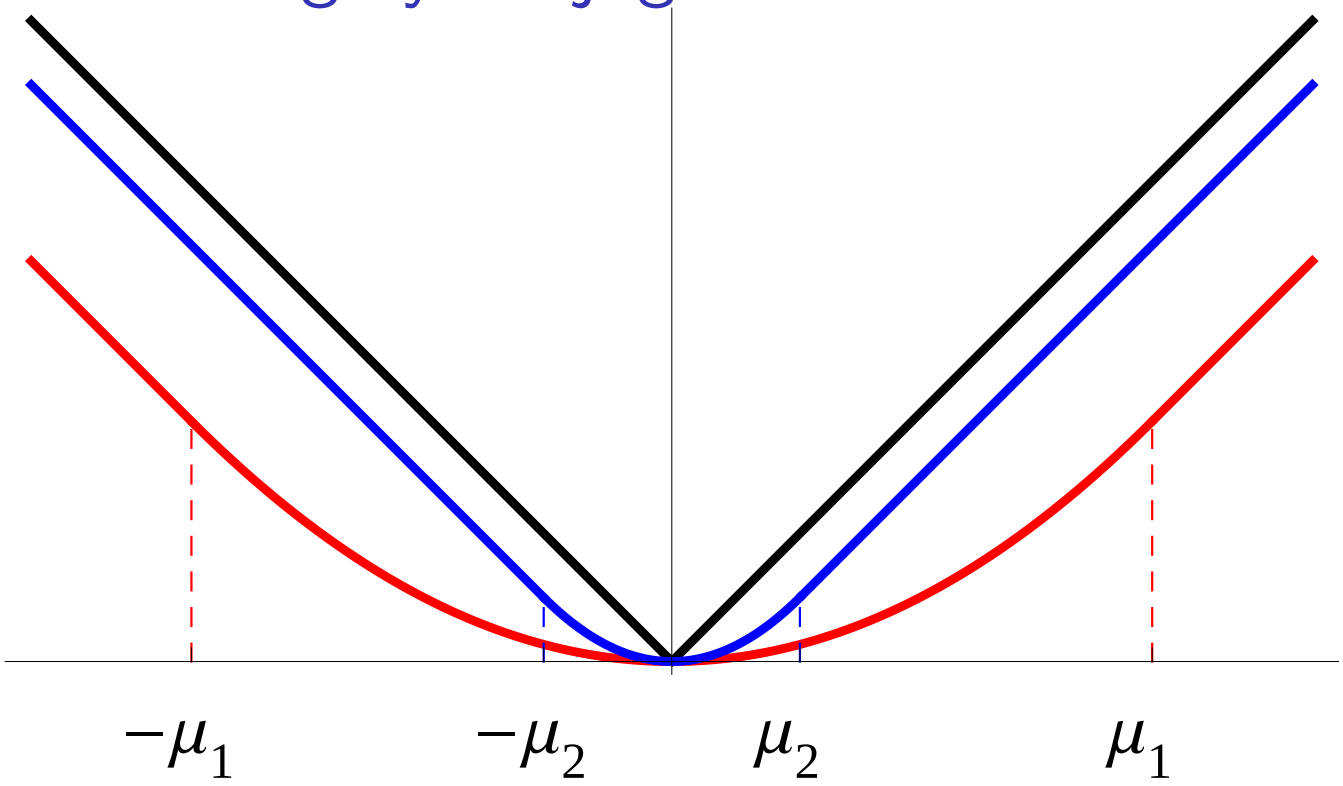
- For the "path-following" algorithms (those that try to solve the KKT conditions, with the complementary slackness condition changed), there are many variations. Can control how conservative (stay very close to central path, so update t only a little) or aggressive they are.
- Usually, force solutions to be in some neighborhood of the central path. Tight neighborhoods lead to short-step algorithms; loose neighborhoods lead to long-step algorithms.
- Can also make predictor-corrector algorithms, that alternate steps between increasing g and getting closer to the central path. Sort of like the inner-outer iterations, with only one inner iteration.
- Most common algorithm in practice is Mehotra's 1989 algorithm, which controls some parameters dynamically, and makes a second-order correction. (This is ideal for non-sparse problems, since it makes a Cholesky factorization and uses it twice; for Lanczos-based solvers, this benefit is gone).

Accelerated convergence

Nesterov's accelerated method. We already discussed. Need f to be continuously differentiable. So, how can we apply this to non-differentiable problems?

By smoothing...
Many ways to smooth. Next slide shows just one possible way, due to Nesterov's 2005 paper (for the case of the ℓ_1 norm and the Huber function, there are many formulations to derive it).

Smoothing by conjugation



- $\|x\|_1 = \max_{u \in \mathcal{Q}_d} \langle u, x \rangle$ where $\mathcal{Q}_d = \{u : \|u\|_\infty \leq 1\}$
- Define

$$f_\mu(x) = \max_{u \in \mathcal{Q}_d} \langle u, x \rangle - \frac{\mu}{2}\|u\|^2$$

- Then ∇f_μ exists and Lipschitz, $L_\mu \propto \mu^{-1}$
- This is just the Huber function

Why smooth?

The ℓ_1 function is non-smooth, so we have been smoothing it to get faster convergence. But can't we write out a smooth formulation of BP_ε?

Basis Pursuit Denoising BP_ε

$$\min_x \|x\|_1 \quad \text{subject to} \quad \|Ax - b\| \leq \varepsilon$$

Think of $x = x^{(+)} - x^{(-)}$.

Basis Pursuit, bound-constrained BP_ε

$$\min_{x^{(+)}, x^{(-)}} \mathbb{1}^T (x^{(+)} + x^{(-)}) \quad \text{subject to}$$
$$\|A(x^{(+)} - x^{(-)}) - b\| \leq \varepsilon, x^{(+)} \geq 0, x^{(-)} \geq 0$$

Problem: projection step is in general hard to solve

$$\mathcal{P}(u, v) = \operatorname*{argmin}_{x^{(+)}, x^{(-)}} \|u - x^{(+)}\|^2 + \|v - x^{(-)}\|^2 \quad \text{subject to}$$
$$\|A(x^{(+)} - x^{(-)}) - b\| \leq \varepsilon, \; x^{(+)} \geq 0, \; x^{(-)} \geq 0$$

Outline

software

General purpose Interior Point Method solvers

- SeDuMi `http://sedumi.ie.lehigh.edu/`. Solves all conic style problems, including LP, QP, and SDP. Free, but based on MATLAB
- SDPT3 `http://www.math.nus.edu.sg/~mattohkc/sdpt3.html`. Basically interchangeable with SeDuMi.
- cvxopt `http://abel.ee.ucla.edu/cvxopt/`. Vandeberghe's conic solver, in python. Has some modelling support. See also cvxmod

Environments and modelling languages

- cvx `http://www.stanford.edu/~boyd/cvx/`. Michael Grant and Stephen Boyd's MATLAB front-end. Super easy to use, and recommended. Calls either SeDuMi or SDPT3.
- yalmip `http://control.ee.ethz.ch/~joloef/wiki/pmwiki.php`. Has support for about a dozen or more solvers (including some commercial solvers, which are not free and don't come bundled with it). I don't use this, but many people do.

Doing it yourself not so bad (main cost: linear algebra). MATLAB and Mathematica's builtin functions not so good. GSL ?

Online resources

These two textbooks are by well-known authors in the community and are also available online in PDF or PS form.

- Convex Optimization, Stephen Boyd and Lieven Vandenberghe. `http://www.stanford.edu/~boyd/cvxbook`
- Convex Analysis and Nonlinear Optimization: Theory and Examples, Jonathan Borwein and Adrian Lewis, 2nd ed 2005. `www.cecm.sfu.ca/~jborwein/text.ps`

Much of the material in these slides has come from Boyd and Vandenberghe. (Other refs I used for these notes: Wright, and Nocedal and Wright).

Online resources

Other good internet resources:

- Stephen Boyd has videotapes of many of his lectures
  `http://www.stanford.edu/~boyd/teaching.html`
- Lieven Vandenberghe has many notes on his website, including short course notes. I have used his EE326C notes many times, as they are applicable to my interests.
  `http://www.ee.ucla.edu/~vandenbe/index.html`

(continued...)

Online resources

Other good internet resources:

- Dimitri Bertsekas has written about a dozen books, many related to optimization. I use his "Convex Analysis and Optimization" and "Nonlinear programming" (2nd ed) books from time-to-time. As far as I know, these are not available online. He has a new book, "Convex Optimization Theory", out in 2009, and there is a free supplemental chapter online called "Convex Optimization Algorithms": `http://www.athenasc.com/convexdualitychapter.pdf`
- Bertsekas does have an old book of his from the 1980's online for free: `http://web.mit.edu/dimitrib/www/lagr_mult.html`
- Bertsekas has his "Convex Analysis and Optimization" class on MIT's OCW. The lecture notes are available here: `http://ocw.mit.edu/OcwWeb/Electrical-Engineering-and-Computer-Science/6-253Spring2004/CourseHome/index.htm`

Textbooks

Books (in addition to the free online textbooks already mentioned)

- *Primal-Dual Interior-Point Methods*, S. Wright, 1997.
- *Numerical Optimization*, Wright, Nocedal, 2nd ed, 2006.
- *Introductory Lectures on Convex Optimization*, Nesterov, 2004.
- *Lectures on Modern Convex Optimization*, Ben-Tal and Nemirovski, 2001
- *Interior-Point Polynomial Algorithms in Convex Programming*, Nesterov, Nemirovskii, 1994. These are the guys who developed the self-concordant theory for IPM.
- *Convex Analysis and Minimization Algorithms, vols I and II*, Hiriart-Urrity, Lemarechal, 1993.
- *Interior Point Methods for Linear Optimization*, Terlaky, Vial, 2nd ed, 2006.
- *A Mathematical View of Interior-Point Methods in Convex Optimization*, James Renegar, 2001.
- *Convex Analysis*, Rockafellar, 1970.