

CSE 6740 Lecture 23

How Do I Optimize Convex/Linear Functions? (Unconstrained Linear Optimization)

Alexander Gray
agray@cc.gatech.edu

Georgia Institute of Technology

- ① Convex Optimization Problems
- ② Unconstrained Optimization: Linear Algebraic
- ③ Unconstrained Optimization: Online

Convex Optimization Problems

Problems with a convex objective function and, if there are constraints, convex constraints.

Convex Functions

It used to be that optimization was divided into linear and nonlinear objective functions. Now the distinction is between convex and non-convex functions.

A *convex set* C has the property that for any two points x_1 and x_2 in C , $0 \leq \alpha \leq 1$, a line segment

$$\alpha x_1 + (1 - \alpha)x_2 \in C. \quad (1)$$

A *convex function* f has the property that $\text{dom}f$, the domain of f , is a convex set and for any two points in $\text{dom}f$,

$$f(\alpha x_1 + (1 - \alpha)x_2) \leq \alpha f(x_1) + (1 - \alpha)f(x_2). \quad (2)$$

This inequality is often called *Jensen's inequality*. f is *concave* if $-f$ is convex.

Convex Functions

Examples of convex functions:

- affine functions $f(x) = Ax + b$
- quadratic functions
- exponential $f(x) = e^{ax}$
- powers $f(x) = x^a$, $a \geq 1$ or $a \leq 0$
- logarithm $f(x) = \log x$
- all norms
- maximum $f(x) = \max(x_1, \dots, x_N)$
- nonnegative weighted sum of convex functions
 $f = \alpha_1 f_1 + \dots + \alpha_K f_m$, $\alpha_k \geq 0$
- maximum of convex functions $f(x) = \max(f_1(x), f_2(x))$

Convex Functions

Suppose f is differentiable (its gradient ∇f exists at each point in $\text{dom}f$). Then f is convex iff $\text{dom}f$ is convex and for any two points in $\text{dom}f$,

$$f(x_2) \geq f(x_1) + \nabla f(x_1)^T(x_2 - x_1). \quad (3)$$

Thus, if $\nabla f(x) = 0$, then for all $\tilde{x} \in \text{dom}f$, $f(\tilde{x}) \geq f(x)$, i.e. x is a *global* minimizer of f . This is what makes convex functions special.

If f is twice differentiable, f is convex iff $\text{dom}f$ is convex and its Hessian is positive definite (for all $x \in \text{dom}f$, $\nabla^2 f(x) \geq 0$), or in 1 dimension, $f''(x) \geq 0$.

Least-Squares

Consider the least-squares problem

$$\text{Find } x^* = \arg \min_{x \in \mathbb{R}^D} \|Ax - b\|_2^2 = x^T A^T A x - 2b^T A x + b^T b. \quad (4)$$

This is quadratic and unconstrained.

It can be solved analytically as $x^* = A^{-1}b$. So this is a special “easy” case.

Unconstrained Convex Optimization

The conditions for optimality of a convex function, in the unconstrained case, boil down to the necessary and sufficient condition

$$\nabla f(x) = 0. \tag{5}$$

So for convex functions without constraints, unconstrained optimization methods like Newton's method find the *global* minimum.

If we use L_1 or L_∞ minimization, we end up with a constrained optimization problem.

General Form

From now on, “convex optimization” refers to problems having convex objective functions and convex constraints. For such problems, we can find the global minimum, in polynomial time.

Recall the general form for an optimization problem:

$$\text{Find } x^* = \arg \min_{x \in \mathbb{R}^D} f(x) \quad (6)$$

$$\text{subject to } c_i(x) \geq 0, \quad i = 1, \dots, M \quad (7)$$

$$d_i(x) = 0, \quad i = 1, \dots, N. \quad (8)$$

The domain is the intersection of the domains of the constraint functions. A point is *feasible* if it satisfies the constraints.

Unconstrained Optimization: Linear Algebraic

Computational linear algebra can be seen as a special case of unconstrained optimization that is very well-studied.

Computational Linear Algebra

There is one main computation studied in numerical linear algebra: Solving a system of linear equations $Ax = b$ where A is $N \times N$, or perhaps $Ax_j = b_j$ for many j .

It has some common special cases:

- Inverting a matrix A to obtain A^{-1} ($AA^{-1} = I$; think $x = A^{-1}b$).
- Solving a linear least-squares problem $\min_x \|Ax - b\|_2$.
- Solving an eigenvalue problem $Ax = \lambda x$.

All three of these come up commonly in statistics, e.g. respectively in evaluating a Gaussian density, linear regression, PCA.

Types of Matrices

There are several special cases of matrices for which there are often faster custom methods:

- banded (diagonal, tridiagonal, triangular, Hessenberg...)
- block
- symmetric ($A^T = A$)
- definite (positive, semi-)
- Vandermonde
- Toeplitz

Mostly, these allow different matrix decompositions, which lead to different types of solutions.

Types of Matrices

These lead to implicit storage, and matrix multiplication:

- sparse
- kernel

Examples:

- Covariance matrices: symmetric positive definite, sometimes diagonal.
- AR models: Toeplitz.
- Kernel PCA: kernel matrix, or sparse.

Linear Regression

Recall linear regression:

$$y_i = X_i\beta + \epsilon_i = \sum_j \beta_j X_{ij} + \epsilon_i. \quad (9)$$

We want the parameters β which minimize the squared error, or residual sum of squares

$$\text{RSS}(\beta) = \sum_{i=1}^N (y_i - \hat{f}(x_i))^2 \quad (10)$$

$$= (y - X\beta)^T (y - X\beta) \quad (11)$$

$$= \|y - X\beta\|_2. \quad (12)$$

Least Squares Problem

This has the form

$$\min_x \|Ax - b\|_2. \quad (13)$$

In linear algebra this is the standard linear *least-squares problem*.

This is basically solving $Ax = b$ when the system is *overdetermined* (more equations than unknowns). It can be put in the form of $A'x = b'$ with square A' as

$$(A^T A)x = A^T b, \quad (14)$$

called the *normal equations* of the least-squares problem.

Optimization Form

Consider the linear system $Ax = b$. Assume for the moment that A is symmetric and positive definite (thus square). Let's consider the best solution to the quadratic form

$$\phi(x) = \frac{1}{2}x^T Ax - x^T b. \quad (15)$$

Then the minimizer x^* is exactly the solution to $Ax = b$.

Optimization Form

To see this, suppose x^* is exactly the solution to $Ax = b$. Then by completing the square,

$$\phi(x) = \frac{1}{2}x^T Ax - x^T Ax^* \quad (16)$$

$$= \frac{1}{2}x^T Ax - x^T Ax^* + \frac{1}{2}x^{*T} Ax^* - \frac{1}{2}x^{*T} Ax^* \quad (17)$$

$$= \frac{1}{2}(x - x^*)^T A(x - x^*) - \frac{1}{2}x^{*T} Ax^*. \quad (18)$$

The last term does not depend on x , so $\phi(x)$ is minimized when $\frac{1}{2}(x - x^*)^T A(x - x^*)$ is minimized. Since A is positive definite, we know that this term is positive unless $x - x^*$ is 0. So $\phi(x)$ takes its minimum when and only when $x = x^*$.

More simply, looking at the derivative we find that

$$\nabla\phi = Ax - b. \quad (19)$$

Clearly the only point at which the gradient is zero is the solution of $Ax = b$. So the problem $Ax = b$ can be written as the optimization problem $\phi(x) = \frac{1}{2}x^T Ax - x^T b$.

Steepest Descent

At the current guess x_k , the function ϕ decreases most rapidly in the direction of the negative gradient: $-\nabla\phi(x_k) = b - Ax_k$. If the *residual*

$$r_k = b - Ax_k \quad (20)$$

of x_k is nonzero, there exists a positive α such that $\phi(x_k + \alpha_k r_k) < \phi(x_k)$.

The residual gives us our search direction, $p_k = r_k$. Now we need the step size α_k ; determining it is called *line search*.

Steepest Descent

Line search is difficult in general, but for our quadratic function we can actually do *exact* line search, in which we choose α_k so that

$$\phi(x_{k+1}) = \min_{\alpha} \phi(x_k + \alpha p_k), \quad (21)$$

in which case we set

$$\alpha_k = \frac{p_k^T r_k}{p_k^T A p_k} \quad (22)$$

to obtain steepest descent with exact line search.

Conjugate Gradient

It turns out that this isn't that efficient. Taking the steepest direction from x_k is not the same as taking the direction which goes to the bottom of a long trough with steep sides, say.

We will do a smoothing of the directions over the iterations, by using

$$p_k = r_{k-1} + \beta_k p_{k-1} \quad (23)$$

where

$$\beta_k = \frac{\langle r_{k+1}, r_{k+1} \rangle}{\langle r_k, r_k \rangle}. \quad (24)$$

This is *conjugate gradient*.

Conjugate Gradient

What's good about this:

- Inside each iteration, we need only perform a multiplication of the matrix A with a different vector each time. We do not need to make a new matrix the size of A . This is good if A is large.
- It turns out this hones in quickly on the top eigenvalues.

It is the method of choice for large, sparse matrices.

Conjugate Gradient

The same basic approach can be derived this way, from the viewpoint of optimization, or from a different viewpoint involving methods called *Lanczos methods*. These viewpoints can be unified using the framework of *Krylov subspaces*.

There is a version of this approach for the least-squares problem, called *LSQR*, and also called *CGNE*. There is a version of this approach for the eigenvalue problem, called *Arnoldi iteration*.

There are versions for cases which are not symmetric positive definite.

Decomposition-based Methods

The downside of the Krylov-type approaches, also called *iterative methods*, is that their numerical stability is not the best possible.

For full-rank matrices, the methods of choice are *Gaussian elimination* or orthogonalization approaches such as *QR factorization*.

For rank-deficient or unknown-rank matrices, the method of choice is *singular value decomposition*.

Unconstrained Optimization: Online

One data point at a time.

Sherman-Morrison Formula

Suppose you have already obtained, after a lot of work, a matrix inverse A^{-1} . Now you want to make a small change to A , for example change one element, or one row, or one column:

$$A \rightarrow A + u \otimes v. \quad (25)$$

($u \otimes v$ is a matrix whose $(i, j)^{th}$ element is the product of the i^{th} component of u and the j^{th} component of v .) If u is a unit vector e_i , this adds v to the i^{th} row; if v is a unit vector e_j , this adds u to the j^{th} column; if both are proportional to unit vectors then a term is only added to one element.

Sherman-Morrison Formula

The *Sherman-Morrison* formula gives the inverse after the change

$$(A + u \otimes v)^{-1} = A^{-1} - \frac{(A^{-1}u) \otimes (vA^{-1})}{vA^{-1}u + 1}. \quad (26)$$

The advantage of this is that it only requires two matrix multiplications and a dot product, which is $O(N^2)$ rather than $O(N^3)$ to redo the entire inverse.

This can be used for obtaining

$$(A + u \otimes v)x = b \quad (27)$$

in an online fashion.

Stochastic Gradient Descent

Consider the batch optimization problem

$$\arg \min_w \left(\frac{1}{N} \sum_{i=1}^N \phi(w^T x_i, y_i) + \frac{\lambda}{2} \|w\|_2^2 \right). \quad (28)$$

where ϕ is convex. *Stochastic gradient descent* updates the parameters one datum at a time via

$$\hat{w}_k = \hat{w}_{k-1} - \alpha_k (\lambda \hat{w}_{k-1} + \phi'(\hat{w}_{k-1}^T x_k, y_k) x_k) \quad (29)$$

where $\phi' = \frac{\partial}{\partial w} \phi$.

Stochastic Gradient Descent

It can be shown that as $k \rightarrow \infty$ and $\alpha_k \rightarrow 0$,

$$\mathbb{E}_{x,y} \phi(\hat{w}_k^T x, y) + \frac{\lambda}{2} \|\hat{w}_k\|_2^2 \rightarrow e^* \quad (30)$$

where

$$e^* = \arg \min_w \mathbb{E}_{x,y} \phi(w^T x, y) + \frac{\lambda}{2} \|w\|_2^2, \quad (31)$$

i.e. the online estimate converges to the true optimum value.

This method can often reach the optimum quickly, perhaps without even touching all of the data. However it may also bounce around the optimum erratically with no clear sense of convergence.

Main Things You Should Know

- What convex functions and optimization problems are
- The connection between linear algebraic problems and optimization
- What stochastic gradient descent is