2.098/6.255/15.093 - Recitation 9

Michael Frankovich and Shubham Gupta

November 20, 2009

1 Unconstrained Optimization

1.1 Optimality Conditions

Consider the unconstrained problem: $min_{x\in\mathbb{R}^n} f(x)$, where $f(x)$ is twice differentiable, the optimality conditions are:

1. Necessary conditions:
   If x^* is a local minimum, then $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is PSD.

2. Sufficient conditions:
   If $\nabla f(\overline{x}) = 0$ and $\exists \epsilon > 0$: $\nabla^2 f(x)$ is PSD for all $x \in B(\overline{x}, \epsilon)$, then $\overline{x}$ is a local optimum.

For a continuously differentiable convex function f, the sufficient and necessary conditions for x^* to be a global minimum is $\nabla f(x^*) = 0$.
Example 1 (Cauchy inequality) Given n positive numbers x_i, prove that

$$\left(\prod_{i=1}^{n} x_i\right)^{1/n} \leq \frac{1}{n}\sum_{i=1}^{n} x_i$$

Proof. First, we change the variables to $y_i = \ln(x_i)$. If we consider $\sum_{i=1}^n y_i = c$ then what we need to prove is

$$\min_{\sum_{i=1}^n y_i = c} \sum_{i=1}^{n} e^{y_i} \geq ne^{c/n}$$

This is a constrained optimization problem; however, we can transform it into an unconstrained problem by subsituting $y_n = c - \sum_{i=1}^{n-1} y_i$. The unconstrained problem is then

$$\min_{y\in\mathbb{R}^{n-1}} \sum_{i=1}^{n-1} e^{y_i} + e^{c-\sum_{i=1}^{n-1} y_i}$$

[1]Thanks Allison Chang for notes.

The necessary condition is $\nabla f(y) = 0$, where $f(y) = \sum_{i=1}^{n-1} e^{y_i} + e^{c-\sum_{i=1}^{n-1} y_i}$. We have:

$$\frac{\partial f(y)}{\partial y_i} = e^{y_i} - e^{c-\sum_{j=1}^{n-1} y_j}$$

Thus we obtain a system of equations

$$y_i = c - \sum_{j=1}^{n-1} y_j \quad \forall i = 1, \ldots, n-1$$

This system has a unique solution of $y_i = c/n$ for all i. If we know that the function $f(y)$ has global minima (which it does), then this solution is the unique global minimum with the optimal value of $ne^{c/n}$. Thus the Cauchy inequality is proved.

2 Gradient Methods

We are interested in solving the following nonlinear unconstrained problem: $min_{x \in \mathbb{R}^n} f(x)$. In general, gradient methods generate a sequence of iterates x_k that converge to an optimal solution x^*.
Generic algorithm elements:

1. Iterative update $x^{k+1} = x^k + \lambda^k d^k$
2. Descent direction $\nabla f(x^k)' d^k < 0$; for example, $d^k = -D^k \nabla f(x^k)$, where D^k is PSD.
3. Best step length $\lambda^k = argmin_{\lambda > 0} f(x^k + \lambda d^k)$.

3 Methods of Unconstrained Optimization

3.1 Steepest Descent

First of all, we might ask ourselves, why should the gradient be part of the direction in which we want to move? Suppose we are at a point x, and we want to move to a point $x + \lambda d$ such that $f(x + \lambda d) < f(x)$. The linear approximation of f at $x + \lambda d$ is

$$f(x + \lambda d) \approx f(x) + \lambda \nabla f(x)^T d.$$

Thus we want to find a d such that $\nabla f(x)^T d$ is as small (negative) as possible. Define

$$\tilde{d} = -\frac{\nabla f(x)}{\|\nabla f(x)\|}.$$

Note that $\|\tilde{d}\| = 1$. Let d be any other vector that satisfies $\|d\| = 1$. Then

$$\nabla f(x)^T \tilde{d} = \nabla f(x)^T \left(-\frac{\nabla f(x)}{\|\nabla f(x)\|} \right) = -\frac{\nabla f(x)^T \nabla f(x)}{\|\nabla f(x)\|} = -\frac{\|\nabla f(x)\|^2}{\|\nabla f(x)\|} = -\|\nabla f(x)\| = -\|\nabla f(x)\| \|d\|.$$

Now the Cauchy-Schwarz inequality says that for any vectors a and b, $|a^T b| \leq \|a\|\|b\|$, which implies $-a^T b \leq \|a\|\|b\|$, or $a^T b \geq -\|a\|\|b\|$. Thus $-\|\nabla f(x)\|\|d\| \leq \nabla f(x)^T d$, so we have

$$\nabla f(x)^T \tilde{d} \leq \nabla f(x)^T d.$$

We have shown that among all directions d with $\|d\| = 1$, $\tilde{d}$ makes $\nabla f(x)^T d$ the smallest (most negative). The unnormalized direction $-\nabla f(x)$ is called the direction of steepest descent at x.

For the steepest descent method, we set D_k to be the identity matrix I for all k. Thus the iterative step is just

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

The algorithm stops when $\nabla f(x_k) = 0$, or when $\|\nabla f(x_k)\|$ is very small. The only unspecified parameter in this algorithm is the stepsize λ_k. There are various methods for choosing a stepsize. If $f(x)$ is a convex function, then one way to pick a stepsize is an exact line search. Since we already determined that the new point will be $x_k + \lambda_k d_k$, where $d_k = -\nabla f(x_k)$, we just want to find λ_k to minimize $f(x_k + \lambda_k d_k)$. Let $h(\lambda) = f(x_k + \lambda d_k)$. We want to find λ such that $h'(\lambda) = \nabla f(x_k + \lambda d_k)^T d_k = 0$. In some cases, we can find an analytical solution to this equation. If not, recognize that $h(\lambda)$ is convex since it is the composition of a convex function with a linear function. Thus $h''(\lambda) \geq 0$ for all λ, which implies $h'(\lambda)$ is increasing. Notice that $h'(0) = \nabla f(x_k)^T d_k = -\nabla f(x_k)^T \nabla f(x_k) = -\|\nabla f(x_k)\|^2 < 0$. Since $h'(\lambda)$ is increasing, we can find some $\bar{\lambda} > 0$ such that $h'(\bar{\lambda}) > 0$. Then we can keep bisecting the interval $\left[0, \bar{\lambda}\right]$ until we find λ^* such that $h'(\lambda^*) = 0$.

3.2 Newton's Method

Suppose we are at a point x and move to $x + d$. The second-order approximation of f at $x + d$ is

$$h(d) = f(x) + \nabla f(x)^T d + \frac{1}{2} d^T H(x) d,$$

where $H(x)$ is the Hessian of f at x. We minimize h by finding d such that $\nabla h(d) = \nabla f(x) + H(x)d = 0$, i.e., $d = -H(x)^{-1} \nabla f(x)$, which is called the Newton direction or Newton step at x. This motivates Newton's method, in which the iterative step is

$$x_{k+1} = x_k - H(x_k)^{-1} \nabla f(x_k).$$

Here the stepsize is $\lambda_k = 1$ in every iteration, and $D_k = H(x_k)^{-1}$. Note that the Newton direction is not necessarily a descent direction, though it is as long as $H(x_k)^{-1}$ is positive definite.

3.3 Rates of Convergence

We want to analyze the convergence rate, or the rate at which the error $e_k = \|x_k - x^*\|$ is decreasing, for the two methods described above. Suppose, for example, that the error was $e_k = 0.1^k$ in iteration k. Then we would have errors $10^{-1}, 10^{-2}, 10^{-3}, \ldots$. This error is decreasing linearly. As another example, suppose the error was $e_k = 0.1^{2^k}$. In this

case, the errors would be $10^{-2}, 10^{-4}, 10^{-8}, \ldots$ (much faster!). This error is decreasing quadratically.
It can be shown that the convergence rate is linear for steepest descent and (locally) quadratic for Newton's method. Thus Newton's method typically converges in fewer iterations than steepest descent, but the computation can be much more expensive because Newton's method requires second derivatives.

3.4 Example

Suppose we want to minimize the one-dimensional function $f(x) = 7x - \ln x$. We have $\nabla f(x) = f'(x) = 7 - \frac{1}{x}$ and $H(x) = f''(x) = \frac{1}{x^2}$. We can initialize $x_0 = 1$. The steepest descent iteration is then

$$x_{k+1} = x_k - \lambda_k \left(7 - \frac{1}{x_k}\right),$$

and the Newton step is

$$x_{k+1} = x_k - x_k^2 \left(7 - \frac{1}{x_k}\right) = x_k + \left(x_k - 7x_k^2\right) = 2x_k - 7x_k^2.$$

15.093J / 6.255J Optimization Methods

Fall 2009