# Lecture 21: Algorithms for Unconstrained Optimization

Zizhuo Wang

Institute of Data and Decision Analytics (iDDA)
Chinese University of Hong Kong, Shenzhen

Nov 21, 2018

# Announcements

- Homework 7 due today
- Homework 8 posted, due next Wednesday (11/28)

# Recap: Convexity/Convex Optimization Problem

We defined convex functions and convex optimization problems

- Can identify whether a given function is convex, concave or neither
- Can identify whether an optimization problem is a convex optimization problem
- Can convert problems into convex optimization problems

Implications of Convexity

**Theorem**

*For a convex optimization problem, any local minimizer is also a global minimizer.*

**Theorem**

*For convex optimization problems, KKT conditions are sufficient for global optimality.*

# Recap: Duality for Nonlinear Optimization

There are dual problems for nonlinear optimization problems

- The dual problems are derived by considering the Lagrangian function and switching the order of max and min.
- Dual problems for nonlinear optimization do not necessarily have an explicit form.

Duality theorems:

- Weak duality always holds
- Strong duality holds when the problem is convex and the Slater's condition holds (otherwise, there could be a positive *duality gap*)

# Upcoming Agenda

Discuss how to solve nonlinear optimization problems.

- We have shown that in many cases, KKT condition can be used to solve the optimization problem
- However, those are ad hoc situations. In most cases, we cannot directly find the optimal solution from the KKT conditions
- We want to have a robust procedure (an algorithm) that guarantees to solve the optimization problem.

# Unconstrained Problems

We start with the unconstrained problem:

$$\text{minimize}_{\mathbf{x}} \quad f(\mathbf{x})$$

We are going to study the following methods:

- Bisection search
- Golden section search
- Gradient descent method
- Newton's method

# General Solution Idea

Typically, optimization algorithms are *iterative* procedures.

- Start from some point $\mathbf{x}_0$, then generate a sequence of $\{\mathbf{x}_k\}$
- The sequence terminates when either no progress can be made or when we know that the current solution is already satisfactory
- Typically, we want to have $f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$, i.e., each step we can improve the objective value.
- And hopefully, we want the sequence $\{\mathbf{x}_k\}$ to *converge* to a local minimizer $\mathbf{x}^*$ (or global minimizer).

Recall the only algorithms we have studied: the simplex method and the interior point method. They both follow the above paradigm.

### Definition

Let $\{\mathbf{x}_k\}$ be a sequence of real vectors. Then $\{\mathbf{x}_k\}$ converges to $\mathbf{x}^*$ if and only if for all real numbers $\epsilon > 0$, there exists a positive integer $K$ such that $||\mathbf{x}_k - \mathbf{x}^*|| < \epsilon$ for all $k \geq K$.

In all our discussions, we assume $||\mathbf{x}||$ is the 2-norm of $\mathbf{x} = (x_1, ..., x_n)$, which means:

$$||\mathbf{x}|| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

Example of convergence:

- $x_k = 1/k \to 0$
- $x_k = (1/2)^k \to 0$

# Single Variable Problem

Assume $f(x)$ is a single variable function.

Objective: find a local minimizer of $f(x)$.

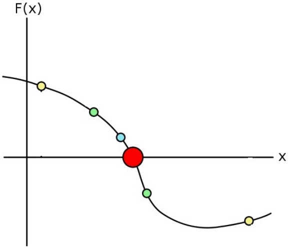
We introduce two methods:

- Bisection method
- Golden section method

# Bisection Method

Bisection method uses the idea that the local minimizer must satisfy the FONC: $f'(x) = 0$.

Therefore, the problem becomes a root-finding problem for $g(x) = f'(x)$.



# Root Finding Algorithm: Bisection Method

Assume one can find $x_l$ and $x_r$ such that $g(x_l) < 0$ and $g(x_r) > 0$. By intermediate value theorem, if $g(\cdot)$ is continuous, there must exist a root of $g(\cdot)$ in $[x_l, x_r]$.

Bisection method:

1. Define $x_m = \frac{x_l + x_r}{2}$
2. If $g(x_m) = 0$, then output $x_m$
3. Otherwise
   - If $g(x_m) > 0$, then let $x_r = x_m$
   - If $g(x_m) < 0$, then let $x_l = x_m$
4. If $|x_r - x_l| < \epsilon$. stop and output $\frac{x_l + x_r}{2}$, otherwise go back to Step 1

One can also set the stop criterion based on $g(x)$

# Bisection Method

In the bisection method, each iteration will divide the search interval to half.

Therefore, to find an $\epsilon$ approximation of $x^*$, we need at most $\log_2 \frac{x_r - x_l}{\epsilon}$ iterations

Applying bisection method to $f'(\cdot)$, one can find a point satisfying FONC (approximately), and if $f(\cdot)$ is convex, we can find the global minimizer of $f(x)$ (approximately)

- Although simple, the bisection method is very useful in practice because it is easy to implement.

Example: Use bisection method to maximize

$$f(x) = \frac{xe^{-x}}{1 + e^{-x}}$$

# Golden Section Method

One drawback of using the bisection method to solve (single variable, unconstrained) optimization problems is that it requires the knowledge (and computation) of $f'(x)$.

- Sometimes, we don't have $f'(x)$ available. For example, $f(x)$ sometimes is only a *black box*, which does not admit an analytical form (thus the derivative is hard to compute)

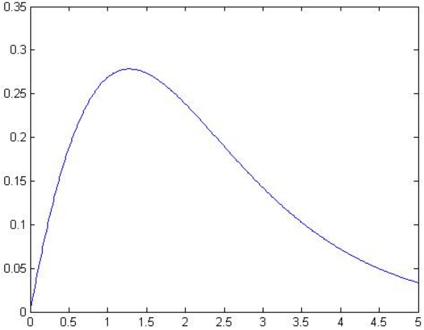
However, if we know that $f(x)$ has a unique local minimum $x^*$ in the range $[x_l, x_r]$, then we still have very efficient way to find it out

- We call such $f$ *unimodal* on $[x_l, x_r]$
- Unimodal function has the property that the local minimum is global minimum (convex function is always unimodal)

In the context of maximization, we say a function is unimodal if it has a unique local maximum

# Example of Unimodal Function

Consider $f(x) = \frac{xe^{-x}}{1+e^{-x}}$ (the function in last homework set):




This is a unimodal function, but not a concave function.

# Golden Section Method

Assume we start with $[x_l, x_r]$. Assume $0 < \phi < 0.5$.

1. Set $x_l' = \phi x_r + (1-\phi)x_l$ and $x_r' = (1-\phi)x_r + \phi x_l$.
2. If $f(x_l') < f(x_r')$, then the minimizer must lie in $[x_l, x_r']$, so set $x_r = x_r'$
3. Otherwise, the minimizer must lie in $[x_l', x_r]$, so set $x_l = x_l'$.
4. If $x_r - x_l < \epsilon$, output $\frac{x_l+x_r}{2}$, otherwise go back to Step 1

If we want to reuse the computation in the last step, we want to set

$$\frac{1-2\phi}{1-\phi} = \phi$$

i.e., $\phi = \frac{3-\sqrt{5}}{2}$, and $1-\phi = \frac{\sqrt{5}-1}{2} = 0.618$ (where the name comes from)

# For Maximization Problem

Both the bisection and golden section method can be easily adapted for maximization problem.

- Just change the way the comparison works (or just change the interval to be cut off in each iteration).

Example Revisited: Use Golden section method to maximize:

$$f(x) = \frac{xe^{-x}}{1 + e^{-x}}$$

# Higher Dimensional Problems

Next we consider the high-dimensional problem

$$\text{minimize}_{\mathbf{x}} \quad f(\mathbf{x})$$

- There is not a clear bisection or golden section in that case

Solution idea:

- Each time, we first find a search direction
- Then we search for a good solution along that direction (which reduces to a one-dimensional problem)

# General Framework for High Dimensional Search

From $\mathbf{x}^0$, we generate a sequence of points:

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{d}^k.$$

We call $\mathbf{d}^k$ the search direction (a vector) and $\alpha_k$ the step size (a scalar).

- Then the key is to choose proper $\mathbf{d}^k$ at each iteration.
- $\mathbf{d}^k$ typically depends on $\mathbf{x}^k$
- Then $\alpha_k$ may be chosen in accordance with some line (one-dimension) search rules.

We will study two such methods:

- Gradient descent method and Newton's method

# Gradient Descent Method

In the following discussion, we assume that $f(\mathbf{x})$ is continuously differentiable (differentiable and the derivative is continuous).

We know by Taylor expansion, for small $\alpha$

$$f(\mathbf{x}+\alpha\mathbf{d}) \approx f(\mathbf{x}) + \alpha\nabla f(\mathbf{x})^T\mathbf{d}$$

Choosing $\mathbf{d} = -\nabla f(\mathbf{x})$ can decrease the objective value.

In the gradient descent method, when at point $\mathbf{x}^k$, we choose $\mathbf{d} = -\nabla f(\mathbf{x}^k)$

# The Step Size

Now we choose the step size $\alpha_k$.

- An intuitive idea is to choose $\alpha_k$ to achieve the largest descent

That is, to choose $\alpha_k$ such that

$$\alpha_k = \text{argmin}_\alpha f(\mathbf{x}^k + \alpha \mathbf{d}^k) \tag{1}$$

- If we get the exact $\alpha_k$ in (**??**), we say we used an *exact line search* method to find the step size
- We can use the golden section method to perform the exact line search
- In some situations, we can even find the exact $\alpha$ analytically

# Example of Exact Line Search

Consider

$$f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T Q\mathbf{x} \quad (Q \text{ positive definite})$$

At $\mathbf{x}^k$, the gradient descent method will choose

$$\mathbf{d}^k = -\nabla f(\mathbf{x}^k) = -(\mathbf{c} + Q\mathbf{x}^k)$$

To choose the step size, notice that we can explicitly compute

$$\begin{aligned} f(\mathbf{x}^k + \alpha\mathbf{d}^k) &= \mathbf{c}^T(\mathbf{x}^k + \alpha\mathbf{d}^k) + \frac{1}{2}(\mathbf{x}^k + \alpha\mathbf{d}^k)^T Q(\mathbf{x}^k + \alpha\mathbf{d}^k) \\ &= \frac{1}{2}\alpha^2(\mathbf{d}^k)^T Q\mathbf{d}^k + \alpha(\mathbf{c}^T\mathbf{d}^k + (\mathbf{x}^k)^T Q\mathbf{d}^k) + f(\mathbf{x}^k) \end{aligned}$$

This is a quadratic function of $\alpha$ with positive second-order term. Thus we can find the optimal $\alpha$ that minimizes $f(\mathbf{x}^k + \alpha\mathbf{d}^k)$:

$$\alpha_k = \frac{(\mathbf{d}^k)^T\mathbf{d}^k}{(\mathbf{d}^k)^T Q\mathbf{d}^k}$$

# Line Search Methods

However, in general, one cannot expect that

$$\alpha_k = \text{argmin}_\alpha f(\mathbf{x}^k + \alpha \mathbf{d}^k) \tag{2}$$

can be solved explicitly. It is sometimes time-consuming.

- Moreover, it is not clear how much benefit there is to solve (**??**) exactly. After all, it is just one iteration, it doesn't mean $\mathbf{x}^k + \alpha_k \mathbf{d}^k$ is optimal

Therefore, it might be good enough to get an approximately good point.

- There are multiple ways to do it, here we introduce the *backtracking line search*

# Backtracking Line Search

Assume we have found $\mathbf{d}^k$ and we want to choose step size $\alpha_k$.

1. We first choose a small $\alpha \in (0, 0.5)$. Also choose a constant $0 < \beta < 1$
2. Let $t = 1$
3. If $f(\mathbf{x}^k + t\mathbf{d}^k) \le f(\mathbf{x}^k) + \alpha t \nabla f(\mathbf{x}^k)^T \mathbf{d}^k$, then choose $\alpha_k = t$. Otherwise, set $t = \beta t$ and repeat this step.

Why this works?

- We know by Taylor expansion, if $t$ is sufficiently small, we must have

$$f(\mathbf{x}^k + t\mathbf{d}^k) \approx f(\mathbf{x}^k) + t\nabla f(\mathbf{x}^k)^T \mathbf{d}^k < f(\mathbf{x}^k) + \alpha t \nabla f(\mathbf{x}^k)^T \mathbf{d}^k$$

Therefore, as long as $t$ is small enough, the condition in Step 3 must be satisfied (remember $\nabla f(\mathbf{x}^k)^T \mathbf{d}^k = -||\nabla f(\mathbf{x}^k)||^2 < 0$)

# Stopping Criterion for Gradient Descent Method

Remember that for local optimality, we need

$$\nabla f(\mathbf{x}) = 0$$

Since we don't know the optimal value, we use the gradient as the stopping criterion:

- We stop when $||\nabla f(\mathbf{x})|| < \epsilon$ for a pre-chosen $\epsilon$

**Theorem**

*Suppose $f(\mathbf{x})$ is convex and the smallest eigenvalue of $\nabla^2 f(\mathbf{x})$ is $m$. Then*

$$||\mathbf{x} - \mathbf{x}^*|| \leq \frac{2}{m} ||\nabla f(\mathbf{x})||$$

*where $\mathbf{x}^*$ is the global minimum of $f(\mathbf{x})$*

- Therefore, when $||\nabla f(\mathbf{x})||$ is small enough, the solution is guaranteed to be close to the optimal solution

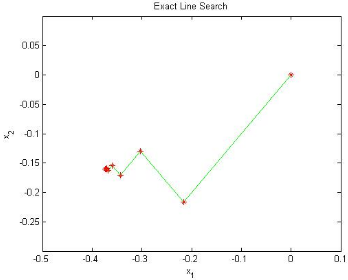
# Gradient Descent Algorithm

Start with any point $\mathbf{x}^0$. Set $k = 0$ and stopping criterion $\epsilon > 0$

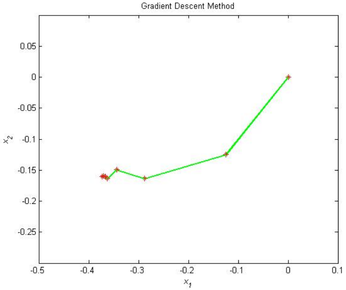
1. Check $||\nabla f(\mathbf{x}^k)||$. If $||\nabla f(\mathbf{x}^k)|| \leq \epsilon$, stop and output $\mathbf{x}^k$. Otherwise, continue to Step 2
2. Let $\mathbf{d}^k = -\nabla f(\mathbf{x}^k)$
3. Use either exact line search or backtracking line search to find $\alpha_k$
4. Let $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{d}^k$, let $k = k + 1$. Go back to step 1.

# Illustration

Minimize $f(x) = \exp(x_1 + x_2) + x_1^2 + 3x_2^2 - x_1 x_2$ using gradient method.




(a) Exact Line Search




(b) Backtracking Line Search