# MAT 3007 – Optimization

## Algorithms for Unconstrained Optimization Problems

*Lecture 16* *July 14th*

Andre Milzarek SDS / CUHK-SZ

## Repetition

## Convex Problems:

- A minimization problem $\min_{x \in \Omega} f(x)$ is called convex if $\Omega$ is a convex set and $f$ is convex.
- Convexity/concavity plays a very important role in optimization problems!

## Calculus & Rules:

- A function $f$ is convex on a convex set $\Omega$ iff the Hessian $\nabla^2 f(x)$ is positive semidefinite for all $x \in \Omega$.
- Rich calculus: sum rule, composition, max-/min-rule.
- If $f$ is convex, then $L_{\leq c} = \{x : f(x) \leq c\}$ is a convex set $\rightsquigarrow$ can be used to check convexity of constraints.
- $\Omega = \{x : g(x) = 0,\ h(x) = 0\}$ is convex if all $g_i$ are convex and $h$ is an affine-linear function, i.e., $h(x) = Ax - b$.

## Convexity & Optimality

- Every local minimizer of a convex problem is a global minimizer.
- Every stationary point or KKT point of a (unconstrained/ constrained) convex problem is a global minimizer.

$\rightsquigarrow$ If $f$ is concave, we typically consider $\max_{x \in \Omega} f(x)$ or $\min_{x \in \Omega} -f(x)$.

# Algorithms for Unconstrained Problems

# Unconstrained Problems

We start with the unconstrained problem:

$$\mathrm{minimize}_{x\in\mathbb{R}^n} \quad f(x)$$

We are going to study the following methods:

- Bisection search and golden section search.
- Gradient descent method.
- Newton's method.

Optimization algorithms are iterative procedures:

- Starting from an initial point $x^0$, a sequence of iterates $\{x^k\}$ is generated.
- Goal: reduction of the function values and convergence to an optimal solution.

# Problems in $\mathbb{R}$

# Single Variable Problem

Assume $f : \mathbb{R} \to \mathbb{R}$ is a single variable function.

Our Objective: find a local minimizer of $f$.

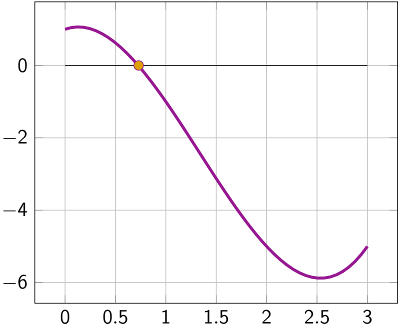
We introduce two methods:

- Bisection method.
- Golden section method.

Bisection method uses the idea that the local minimizer must satisfy the first-order necessary conditions: $f'(x) = 0$.

Therefore, the problem becomes a root-finding problem for

$$g(x) = f'(x) = 0.$$



Assume we can find $x_\ell$ and $x_r$ such that $g(x_\ell) < 0$ and $g(x_r) > 0$.

By the intermediate value theorem, if $g$ is continuous, there must exist a root of $g$ in $[x_\ell, x_r]$.

## Bisection Method

1. Define $x_m = \frac{x_\ell + x_r}{2}$.
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 $x_\ell = x_m$.
4. If $|x_r - x_\ell| < \epsilon$: stop and output $\frac{x_\ell + x_r}{2}$, otherwise go back to step 1.

One can also set the stopping criterion based on $|g(x)| < \epsilon$.

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 $\log_2 \frac{x_r - x_\ell}{\epsilon}$ many iterations.

Applying the bisection method to $f'$, we can find an approximate stationary point. If $f$ is convex, this is an (approximate) global minimizer of $f$.

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

Example: Use bisection method to minimize:

$$f(x) = -\frac{xe^{-x}}{1+e^{-x}} \quad \rightsquigarrow \quad f'(x) = -\frac{e^{-x}(1-x+e^{-x})}{(1+e^{-x})^2}$$

# Bisection Method

```matlab
function [x,gx] = bisection(g,xl,xr,options)

% Compute intial function values
gr  = g(xr); gl  = g(xl); sl = sign(gl);

if gl*gr > 0
    fprintf(1,'The input data not suitable!');
    x = []; gx = []; return
end

for i = 1:options.maxit
    xm  = (xl + xr)/2; gm  = g(xm);

    if abs(gm) < options.tol || abs(xl-xr) < options.tol
        x = xm; gx = gm; return
    end

    if gm > 0
        if sl < 0, xr = xm; else, xl = xm; end
    else
        if sl < 0, xl = xm; else, xr = xm; end
    end
end
```

# Golden Section Method

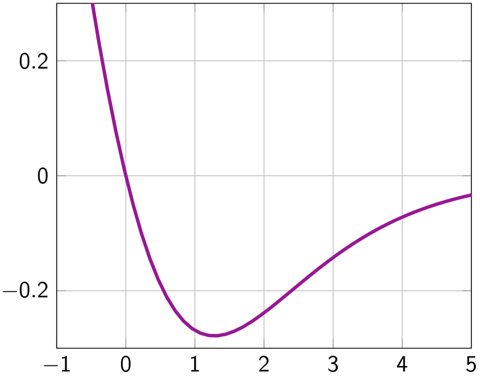
Drawback of the bisection method: When solving (single variable, unconstrained) optimization problems, we require the knowledge (and computation) of $f'$.

- Sometimes, $f'$ is not available. For example, $f$ 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 $f$ has a unique local minimum $x^*$ in the range $[x_\ell, x_r]$, then we still have a very efficient way to find $x^*$:

- We call $f$ unimodal if it only has one single stationary point (on $\mathbb{R}$).
- Unimodal functions have the property that the local minimum is already global. (Similarly, if the stationary point is a local maximum).

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




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

## Golden Section Method

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

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

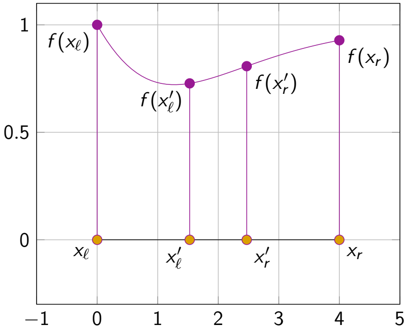
- Suppose we update $x_r = x'_r$. We want to choose $\phi$ such that $x'_r$ of the new iteration coincides with $x'_\ell$ of the old iteration.

$\rightsquigarrow$ This allows to save one function evaluation!

- This is true when

$$\phi = \frac{3-\sqrt{5}}{2} \quad \text{and} \quad 1-\phi = \frac{\sqrt{5}-1}{2} = 0.618.$$

# Illustration and Example




Both the bisection and golden section method can be easily adapted for maximization problems. (Just adjust the comparison).

Example Revisited: Use the Golden section method to maximize:

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

## Higher-Dimensional Problems

Next, we consider the $n$-dimensional problem:

$$\mathrm{minimize}_{x \in \mathbb{R}^n} \quad f(x)$$

- There is no clear bisection or golden section in that case.

Solution and General Idea:

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

Starting from the initial point $x^0$, we generate a sequence of points:

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

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

- The key is to choose a proper direction $d^k$ at each iteration.
- $d^k$ typically depends on $x^k$.
- The step size $\alpha_k$ may be chosen in accordance with some line (one-dimensional) search rules (later).

We will study two such methods:

- Gradient descent method and Newton's method.

In the following, we assume that $f$ is continuously differentiable.

**Definition: Descent Direction**

A vector $d \in \mathbb{R}^n$ is a descent direction of $f$ at $x$ if $\nabla f(x)^\top d < 0$.

Important Observation:

- Taking a small enough step along a descent direction reduces the objective function value.
- By Taylor: there exists $\epsilon > 0$ such that

$$f(x + \alpha d) < f(x) \quad \forall\, \alpha \in (0, \epsilon].$$

### Schematic Descent Directions Method

1. Initialization: Select an initial point $x^0 \in \mathbb{R}^n$.

**For** $k = 0, 1, \ldots$:

2. Pick a descent direction $d^k$.
3. Find a stepsize $\alpha_k$ satisfying $f(x^k + \alpha_k d^k) < f(x^k)$.
4. Set $x^{k+1} = x^k + \alpha_k d^k$.
5. If a stopping criterion is satisfied, then STOP and $x^{k+1}$ is the output.

Open questions and missing details:

- What is the initial point $x^0$?
- How to choose the descent direction? What step size should be taken?
- What is the stopping criterion?

## Gradient Descent:

- One simple and possible descent direction is $d^k = -\nabla f(x^k)$. This direction satisfies:

$$\nabla f(x^k)^\top d^k = -\|\nabla f(x^k)\|^2 < 0$$

as long as $\nabla f(x^k) \neq 0$.

- Choosing $d^k = -\nabla f(x^k)$, the abstract descent method becomes the gradient descent method.

## Stopping Criterion:

- A popular stopping criterion is: $\|\nabla f(x^{k+1})\| \leq \epsilon$ with tolerance $\epsilon > 0$.

$\rightsquigarrow$ We stop if $x^{k+1}$ is an approximate stationary point.

Constant Step Size:

- Choose $\alpha_k = \bar{\alpha}$ for all $k$.

Exact Line Search:

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

That is, choose $\alpha_k$ such that:

$$\alpha_k = \mathrm{argmin}_{\alpha \geq 0} f(x^k + \alpha d^k). \tag{1}$$

- If we get the exact $\alpha_k$ in (1), 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.

Consider

$$f(x) = b^\top x + \frac{1}{2}x^\top Ax \quad (A \text{ positive definite})$$

At $x^k$, the gradient descent method will choose:

$$d^k = -\nabla f(x^k) = -(b + Ax^k).$$

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

$$\begin{aligned} f(x^k + \alpha d^k) &= b^\top (x^k + \alpha d^k) + \frac{1}{2}(x^k + \alpha d^k)^\top A(x^k + \alpha d^k) \\ &= \frac{1}{2}\alpha^2 (d^k)^\top Ad^k + \alpha(b^\top d^k + (x^k)^\top Ad^k) + f(x^k) \end{aligned}$$

This is a quadratic function of $\alpha$ with positive second-order term! We can find the optimal $\alpha \geq 0$ minimizing $\phi(\alpha) = f(x^k + \alpha d^k)$:

$$\alpha_k = \frac{(d^k)^\top d^k}{(d^k)^\top Ad^k}.$$

# Example: Gradient Method for Quadratic Functions

```
function [x,obj] = gm_quadratic(A,b,x0,eps)

x       = x0;       iter    = 0;
g       = A*x + b;  ng      = norm(g);

fprintf(1,'--- grad. method ; n = %g\n',length(b));
fprintf(1,'ITER ; OBJ.VAL ; G.NORM ; STEP.SIZE\n');

while ng > eps && iter < 10000
 iter    = iter + 1;
 alpha   = ng^2 / (g'*A*g);
 x       = x - alpha*g;
 g       = A*x + b;
 ng      = norm(g);
 obj     = 0.5*x'*A*x + b'*x;
 fprintf(1,'[%4i] ; %2.6f ; %2.6f ; %1.2f\n',iter,obj,ng,
     alpha);
end
```

# Example: A Quadratic Problem

We now want to test the method and solve the problem:

$$\min_x \; f(x) = x_1^2 + 2x_2^2 = \frac{1}{2} x^\top \begin{pmatrix} 2 & 0 \\ 0 & 4 \end{pmatrix} x.$$

We use the initial point $x^0 = (2,1)^\top$ and the tolerance $\varepsilon = 10^{-5}$.

The method stops after 13 iterations with a solution that is already very close the optimal value $x = 10^{-5} \cdot (0.1254, -0.0627)^\top$.

In general, we can not expect that

$$\alpha_k = \mathrm{argmin}_{\alpha \geq 0} f(x^k + \alpha d^k) \tag{2}$$

can be solved explicitly. It can be very time-consuming!

- Computing $\alpha_k$ is an optimization problem on its own!
- It is also not clear how much benefit there is when solving (2) exactly. After all, it is just one iteration and it does not imply that $x^k + \alpha_k d^k$ is optimal.

Agenda: Let us consider approximate and cheaper techniques!

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

Assume we have found a descent direction $d^k$ and we want to choose step size $\alpha_k$.

Let $\sigma, \gamma \in (0,1)$ be given. Choose $\alpha_k$ as the largest element in $\{1, \sigma, \sigma^2, \sigma^3, ...\}$ such that

$$f(x^k + \alpha_k d^k) - f(x^k) \leq \gamma \alpha_k \cdot \nabla f(x^k)^\top d^k.$$

- This condition is called Armijo condition.
- $\alpha_k$ can be determined after finitely many steps if $d^k$ is a descent direction.

Procedure:

1. Start with $\alpha = 1$.
2. If $f(x^k + \alpha d^k) \leq f(x^k) + \gamma\alpha \cdot \nabla f(x^k)^\top d^k$, choose $\alpha_k = \alpha$. Otherwise, set $\alpha = \sigma\alpha$ and repeat this step.

Why does this work?

- By Taylor expansion, if $\alpha$ is sufficiently small, we have

$$f(x^k + \alpha d^k) \approx f(x^k) + \alpha \nabla f(x^k)^\top d^k < f(x^k) + \gamma\alpha \cdot \nabla f(x^k)^\top d^k.$$

  Therefore, as long as $\alpha$ is small enough, the Armijo condition must be satisfied (recall $\nabla f(x^k)^\top d^k = -\|\nabla f(x^k)\|^2 < 0$).

Illustration:

- Define $\phi_k(\alpha) := f(x^k + \alpha d^k) - f(x^k)$. Then, we have

$$\phi_k'(\alpha) = \nabla f(x^k + \alpha d^k)^\top d^k, \quad \phi_k'(0) = \nabla f(x^k)^\top d^k.$$

- The Armijo condition is then equivalent to:

$$\textit{find } \alpha \textit{ with } \phi_k(\alpha) \leq \gamma\alpha \cdot \phi_k'(0).$$

- Notice that $\phi_k'(0) < 0$ (since $d^k$ is a descent direction).

# Armijo Line Search: Visualization

### Gradient Descent Method

1. Initialization: Select an initial point $x^0 \in \mathbb{R}^n$.

**For** $k = 0, 1, \ldots$:

2. Pick a stepsize $\alpha^k$ by a line search procedure (exact line search or backtracking) on the function

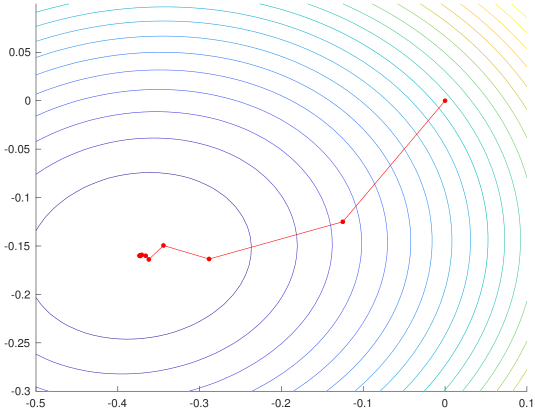
$$\phi(\alpha) = f(x^k - \alpha \nabla f(x^k)).$$

3. Set $x^{k+1} = x^k - \alpha_k \nabla f(x^k)$.
4. If $\|\nabla f(x^{k+1})\| \leq \varepsilon$, then STOP and $x^{k+1}$ is the output.

Minimize

$$f(x) = \exp(x_1 + x_2) + x_1^2 + 3x_2^2 - x_1x_2$$

using the gradient method with Armijo line search.



Questions?