MAT3007 Optimization
Lecture 18 Algorithms

Yuang Chen

School of Data Scienc
The Chinese University of Hong Kong, Shenzhen

July 14, 2025

Outline

Outline

Last Topic: Algorithms

Discuss how to solve nonlinear optimization problems.

- We have shown that in many cases, KKT conditions 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:

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} \quad f(\boldsymbol{x})$$

We are going to study the following methods:

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

General Solution Idea

Typically, optimization algorithms are *iterative* procedures.

- Start from some feasible point $\boldsymbol{x}_0$, then generate a sequence of $\{\boldsymbol{x}_k\}$
- 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 $f(\boldsymbol{x}_{k+1}) < f(\boldsymbol{x}_k)$, i.e., each step we can improve the objective value.
- And hopefully, we want the sequence $\{\boldsymbol{x}_k\}$ to *converge* to a local minimizer $\boldsymbol{x}^*$ (or global minimizer).

Outline

Single Variable Problem

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

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

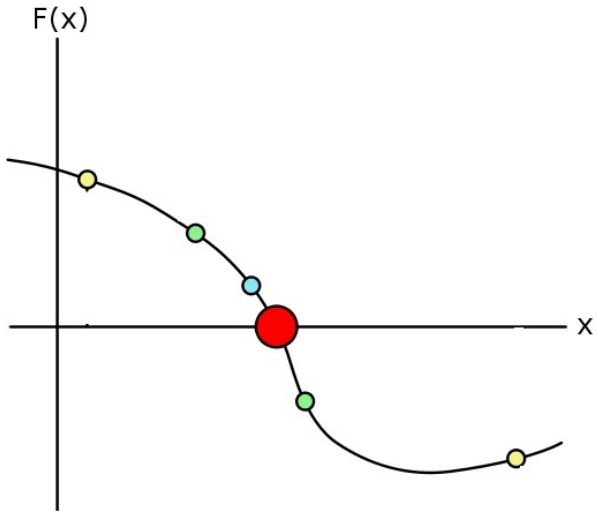
We introduce two methods:

- Bisection method
- Golden section method

Bisection Method

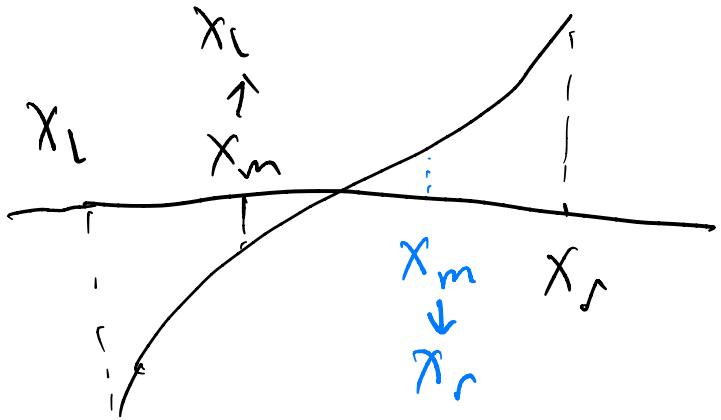
Bisection method uses the idea that the local minimizer must satisfy the FONC: $f'(x) = 0$.

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 $g(x_\ell) < 0$ and $g(x_r) > 0$. By intermediate value theorem, if $g(\cdot)$ is continuous, there must exist a root of $g(\cdot)$ in $[x_\ell, x_r]$.

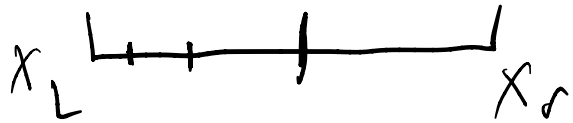


Bisection method:

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 stop criterion based on $|g(x)| \leq \varepsilon$

Bisection Method



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

$$\frac{x_r - x_l}{2^n} = \epsilon \Rightarrow n = \log_2 \frac{x_r - x_l}{\epsilon}$$

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

Applying bisection method to $g = f'$, one can find a critical point satisfying FONC (approximately).

- If f 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.

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 $f'(x)$.

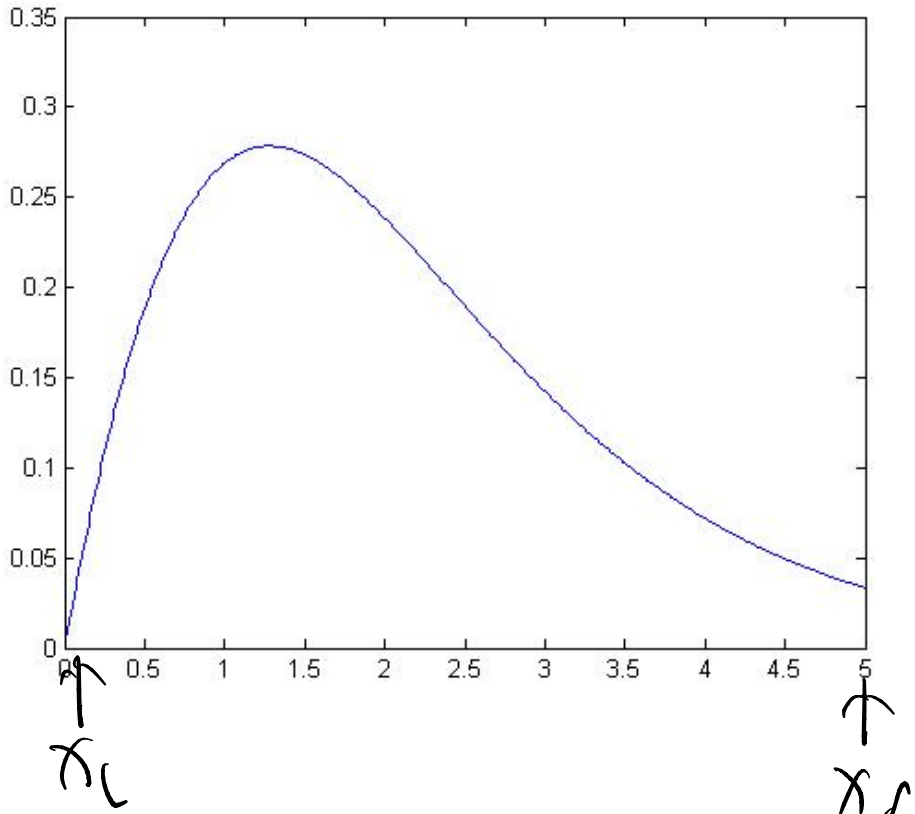
- Sometimes, we don't have $f'(x)$ available. For example, $f(x)$ sometimes is only a *black box*, which does not admit an analytical form (thus the derivative is hard to compute).

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

- We call such f *unimodal* on $[x_\ell, x_r]$
- Unimodal function has the property that the local optimal is global optimal (convex function is always unimodal).

Example of Unimodal Function (Maximization)

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



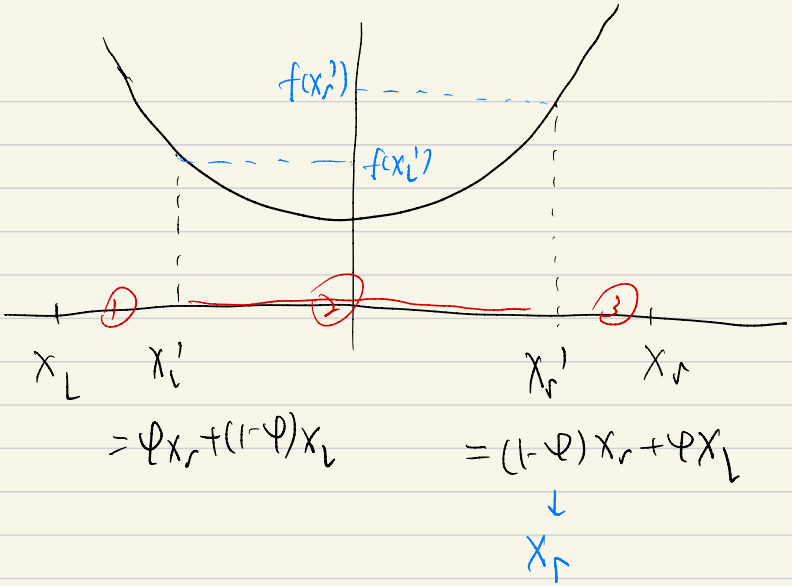
This is a unimodal function, but not a concave function.

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.

If we want to reuse the computation (i.e.,$x''_r = x'_\ell$), then we set $\phi = \frac{3-\sqrt{5}}{2}$, and $1-\phi = \frac{\sqrt{5}-1}{2} = 0.618$ (where the name comes from).



Current iteration: $f(x_l') < f(x_r)$

next iteration: $[x_L, x_r']$

Calculate out x_l'', x_r''

Goal: $x_r'' = x_l'$

$$x_r'' = (1-\varphi) x_r' + \varphi x_L$$
$$= (1-\varphi)((1-\varphi) x_r + \varphi x_L) + \varphi x_L$$
$$= (1-\varphi)^2 x_r + (\varphi(1-\varphi) + \varphi) x_L$$
$$= (1-\varphi)^2 x_r + (\varphi(2-\varphi)) x_L$$

$$x_l' = \varphi x_r + (1-\varphi) x_L$$

$(1-\varphi)^2 = \varphi$

$\varphi(2-\varphi) = 1-\varphi$

$$\Rightarrow \varphi = \frac{3-\sqrt{5}}{2} = 0.382$$

$$1-\varphi = \frac{\sqrt{5}-1}{2} = 0.618$$

Outline

Higher Dimensional Problems

Next we consider the high-dimensional problem

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

- 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.

General Framework for High Dimensional Search

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

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

step size; direction

$$\begin{bmatrix} -x_1^k \\ x_2^k \\ \vdots \\ x_n^k \end{bmatrix} + \alpha_k \begin{bmatrix} -d_1^k \\ d_2^k \\ \vdots \\ d_n^k \end{bmatrix}$$

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

- The key is to choose proper $\boldsymbol{d}^k$ at each iteration, such that $f(\boldsymbol{x}^{k+1}) < f(\boldsymbol{x}^k)$. → descent direction
- $\boldsymbol{d}^k$ typically depends on $\boldsymbol{x}^k$
- Then α_k may be chosen in accordance with some line (one-dimension) search rules.

Our goal is to find the proper $\boldsymbol{d}^k$ and α_k at each iteration so that the sequence $\{x^k\}$ converges to a local minimizer x^* (or global minimizer or something meaningful).

Some Useful Concepts: Convergent Sequences

Definition

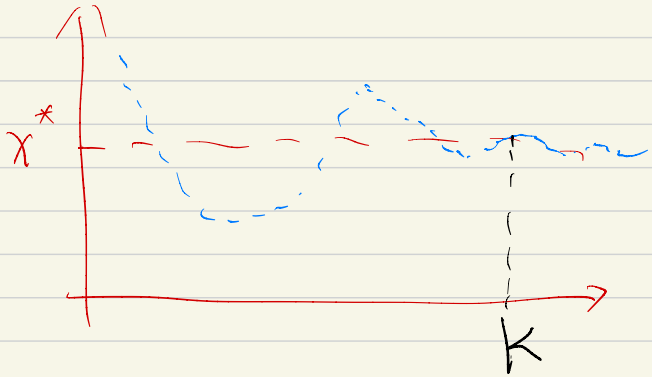
Let $\{\boldsymbol{x}_k\}$ be a sequence of real vectors. Then $\{\boldsymbol{x}_k\}$ converges to $\boldsymbol{x}^*$ if and only if for all real numbers $\epsilon > 0$, there exists a positive integer K such that $||\boldsymbol{x}_k - \boldsymbol{x}^*|| < \epsilon$ for all $k \geq K$. $\forall \epsilon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

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

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

Example of convergence:

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


x^*
K

Gradient Descent Method

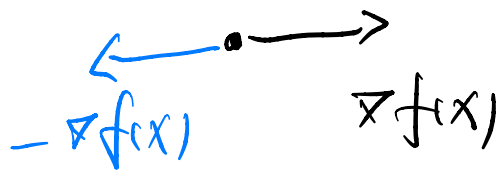
$\vec{a} \cdot \vec{b} = |\vec{a}||\vec{b}| \cos\theta \Rightarrow \vec{b} = -\vec{a}$

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

We know by Taylor expansion, for small α

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

small — next iteration; large — current iteration; < 0



- Descent direction: $\{\boldsymbol{d} : \nabla f(\boldsymbol{x})^T \boldsymbol{d} < 0\}$
- Choose $\boldsymbol{d} = -\nabla f(\boldsymbol{x})$ can decrease the objective value.
- In the gradient descent method, when at point $\boldsymbol{x}^k$, we choose $\boldsymbol{d} = -\nabla f(\boldsymbol{x}^k)$.

steepest descent

The Step Size

Now we choose the step size α_k.

- The easiest idea is to choose α_k to be fixed, i.e., $\alpha_k = \bar{\alpha}$ for all k.
- An intuitive idea is to choose α_k to achieve the largest descent

That is, to choose α_k such that

$$\alpha_k = \arg\min_{\alpha} f(\boldsymbol{x}^k + \alpha \boldsymbol{d}^k) \tag{1}$$

- If we get the exact α_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 α analytically

Example of Exact Line Search

Consider

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

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

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

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

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

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

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

Line Search Methods

However, in general, one cannot expect that

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

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

- Moreover, it is not clear how much benefit there is to solve (2) exactly. After all, it is just one iteration, it doesn't mean $\boldsymbol{x}^k + \alpha_k \boldsymbol{d}^k$ 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 / Armijo Line Search

Assume we have found a descent direction $\boldsymbol{d}^k$ and we want to choose step size α_k.

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

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

This condition is called **Armijo condition.**

Backtracking / Armijo line search procedure:

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

α_k can be determined after finitely many steps once $\boldsymbol{d}^k$ is a descent direction (see next slide).

Armijo Line Search: Discussion

Why does this work?

- By Taylor expansion, if α is sufficiently small, we have

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

  Therefore, as long as α is small enough, the Armijo condition must be satisfied.

Illustration:

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

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

- The Armijo condition is then equivalent to:

  find α *with* $\phi_k(\alpha) < \gamma\alpha \cdot \phi_k'(0)$.

Stopping Criterion for Gradient Descent Method

Remember that for local optimality, we need

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

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

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

$\|x^{k+1} - x^k\| < \varepsilon$

$f(x^{k+1}) - f(x^k) < \varepsilon$

Theorem

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

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

where $\tilde{\boldsymbol{x}}$ is the final solution from gradient descent method and $\boldsymbol{x}^*$ is the global minimum of $f(\boldsymbol{x})$.

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

Gradient Descent Algorithm

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

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

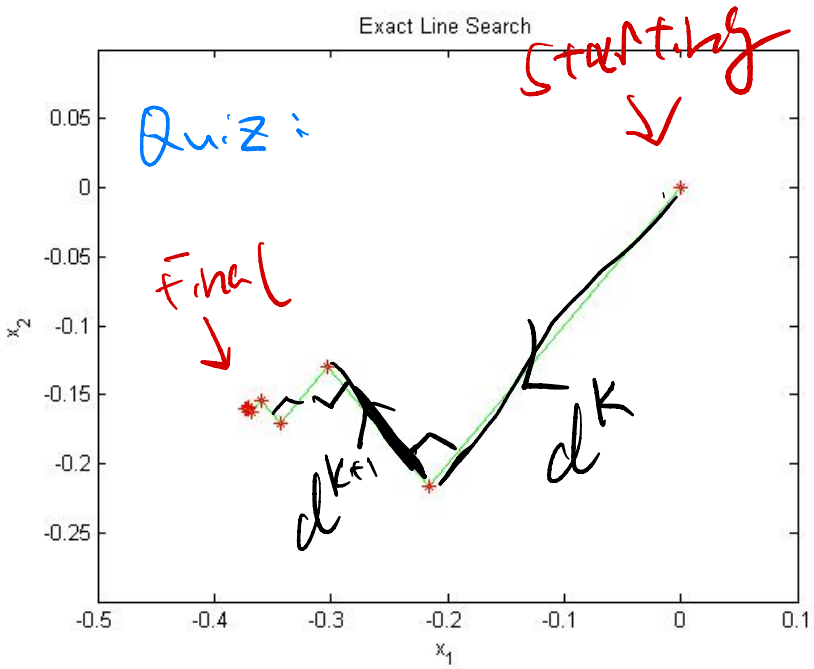


Figure 1: Exact Line Search

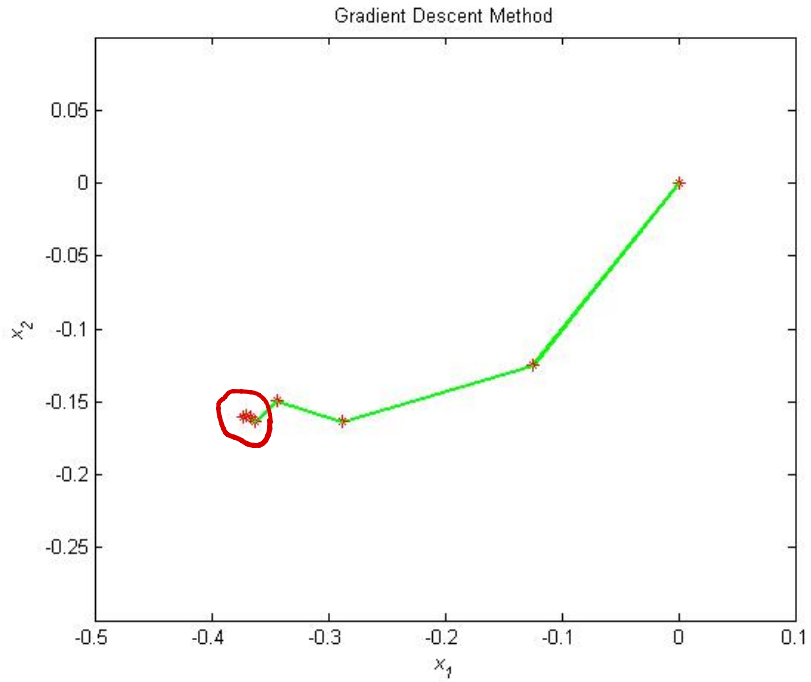
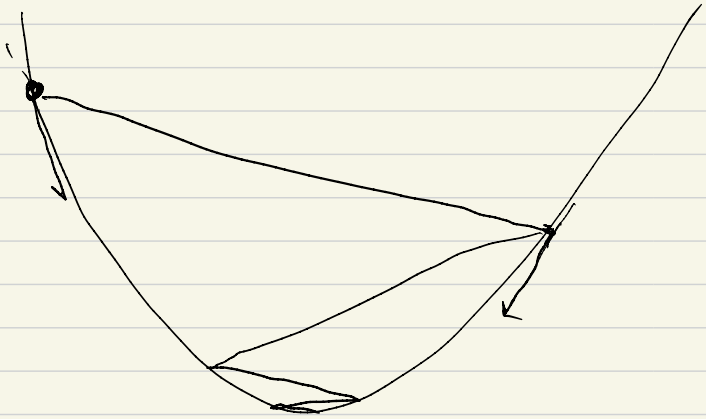
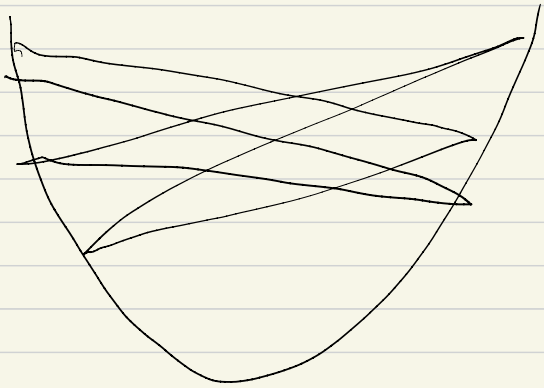


Figure 2: Backtracking Line Search





Property of Exact Line Search

We have seen that when using exact line search, the directions between consecutive steps are perpendicular, i.e.,

$$(\boldsymbol{d}^{k+1})^T \boldsymbol{d}^k = 0$$

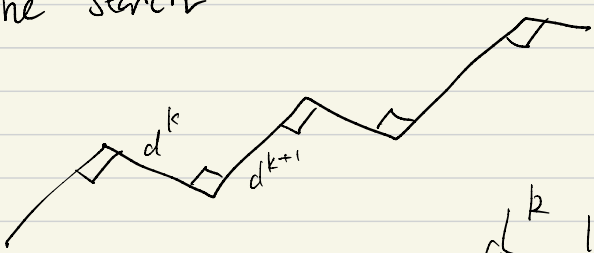
In fact, this is always true when using exact line search.

Why?

- If α_k is the minimizer of $f(\boldsymbol{x}^k + \alpha \boldsymbol{d}^k)$. Then the gradient of $f(\boldsymbol{x}^k + \alpha \boldsymbol{d}^k)$ with respect to α must be 0 at α_k, which means

$$\nabla f(\boldsymbol{x}^k + \alpha_k \boldsymbol{d}^k)^T \boldsymbol{d}^k = -(\boldsymbol{d}^{k+1})^T \boldsymbol{d}^k = 0$$

Exact line search



$d^k \perp d^{k+1}$

$$(d^k)^T d^{k+1} = 0$$

$$\alpha_k = \arg\min_{\alpha} \underbrace{f(x^k + \alpha d^k)}_{g(\alpha)}$$

FONC: $g'(\alpha) = 0$

$$\frac{d}{d\alpha} f(x^k + \alpha d^k) = 0$$

$$\Rightarrow \left(\nabla f(\overbrace{x^k + \alpha d^k}^{x^{k+1}})\right)^T d^k = 0$$

$(f(g(\alpha)))' = f'(g(\alpha)) \cdot g'(\alpha)$

$$\Rightarrow (\nabla f(x^{k+1}))^T d^k = 0$$

$$\Rightarrow (-d^{k+1})^T d^k = 0$$

$$\Rightarrow (d^k)^T d^{k+1} = 0$$

Outline

Lipschitz Continuity

Definition

We say that ∇f is **Lipschitz continuous** over $\mathbb{R}^n$ if

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \quad \forall x, y \in \mathbb{R}^n,$$

where $L \geq 0$ is the **Lipschitz constant**. The function f satisfying this property is called **Lipschitz smooth** (with constant L).

Examples:

1. The linear function $f(x) := b^\top x + c$
2. Consider the quadratic function $f(x) := \frac{1}{2}x^\top Ax + b^\top x + c$

1. $\|\nabla f(x) - \nabla f(y)\|$

$= \|b - b\|$

$= 0 \leq 0 \cdot \|x - y\|$

$L = 0$

2. $f(x) = \frac{1}{2} x^T A x + b^T x + c = \frac{1}{2} Ax^2 + bx + c$

$\|\nabla f(x) - \nabla f(y)\|$

$= \|(Ax + b) - (Ay + b)\|$

$= \|A(x - y)\|$

$\leq \|A\| \|x - y\|$ $\quad L = \|A\|$

$\|A\|$ ↓

largest singular value of A

$= \sqrt{\lambda_{max}(A^T A)}$

Descent Lemma

Descent Lemma

Let f be Lipschitz smooth with constant L. Then, it follows:

$$f(y) \leq f(x) + \nabla f(x)^\top (y - x) + \frac{L}{2}\|y - x\|^2 \quad \forall x, y \in \mathbb{R}^n.$$

- The descent lemma provides a quadratic upper bound for f. For fixed x, we have:

$$f(y) \leq q(y) := f(x) + \nabla f(x)^\top (y - x) + \frac{L}{2}\|y - x\|^2 \quad \forall y.$$

- This lemma shows that gradient descent with a constant stepsize has a descent property. Note that $x^{k+1} - x^k = -\bar{\alpha}\nabla f(x^k)$, we have

$$f(x^{k+1}) \leq f(x^k) - \bar{\alpha}\left(1 - \frac{L\bar{\alpha}}{2}\right)\|\nabla f(x^k)\|^2.$$

- Thus, once we choose $\bar{\alpha} \in \left(0, \frac{2}{L}\right)$, the gradient descent has descent property.

Lemma: $f(y) \le f(x) + \nabla f(x)^T (y-x) + \frac{L}{2}\|y-x\|^2$

Let $y = x^{k+1} \overset{GD}{=} x^k - \bar{\alpha}\nabla f(x^k)$

$x = x^k$

$$f(x^{k+1}) \le f(x^k) + \nabla f(x^k)^T (x^{k+1} - x^k) + \frac{L}{2}\|x^{k+1} - x^k\|^2$$

$$= f(x^k) + \nabla f(x^k)^T \cdot (-\bar{\alpha}\nabla f(x^k)) + \frac{L}{2}\|-\bar{\alpha}\nabla f(x^k)\|^2$$

$$= f(x^k) - \bar{\alpha}\|\nabla f(x^k)\|^2 + \frac{L\bar{\alpha}^2}{2}\|\nabla f(x^k)\|^2$$

$$= f(x^k) + \left(\frac{L}{2}\bar{\alpha}^2 - \bar{\alpha}\right)\|\nabla f(x^k)\|^2$$

$f(x^{k+1}) \le$ small $=$ $f(x^k)$ large $+$ $\left(\frac{L}{2}\bar{\alpha}^2 - \bar{\alpha}\right)\|\nabla f(x^k)\|^2$ (< 0; $\|\nabla f(x^k)\|^2 \ge 0$)

If I want $f(x^{k+1}) < f(x^k)$,

I need $\frac{L}{2}\bar{\alpha}^2 - \bar{\alpha} < 0$

$\frac{L}{2}\bar{\alpha} < 1$

$0 < \bar{\alpha} < \frac{2}{L}$

Lipschitz Continuity via the Hessian

Theorem

Let f be a twice continuously differentiable function. Then, the following two conditions are equivalent:

- *f is Lipschitz smooth with constant L.*
- *$\|\nabla^2 f(x)\| \leq L$ for any $x \in \mathbb{R}^n$, where $\|\nabla^2 f(x)\|$ denotes the largest eigenvalue of the Hessian matrix $\nabla^2 f(x)$.*

$$\nabla^2 f(x) \preceq L I$$

Convergence analysis

Theorem

Suppose the function $f : \mathbb{R}^n \to \mathbb{R}$ is convex and differentiable, and that its gradient is Lipschitz continuous with constant $L > 0$, i.e. we have that $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2$ for any x, y. Then if we run gradient descent for k iterations with a fixed step size $\alpha \leq 1/L$, it will yield a solution $f(x^k)$ which satisfies

$$f(x^k) - f(x^\star) \leq \frac{\|x^0 - x^\star\|_2^2}{2\alpha k},$$

where $f(x^\star)$ is the optimal value. Intuitively, this means that gradient descent is guaranteed to converge and that it converges with rate $\mathcal{O}(1/k)$.

Remark:

- We say gradient descent has a convergence rate $\mathcal{O}(1/k)$.
- This convergence rate is called *sublinear convergence*.

Linear Convergence

- In addition to sublinear convergence, we can also have **linear convergence** of gradient descent under stronger conditions.

Let us first define the notion of linear convergence.

Definition: Linear Convergence

We say that $\{x^k\}_k$ *converges linearly (linear convergence)* with rate $\eta \in (0,1)$ to $x^* \in \mathbb{R}^n$ if there exists $\ell \geq 0$ such that

$$\|x^{k+1} - x^*\| \leq \eta \cdot \|x^k - x^*\|, \quad \forall k \geq \ell.$$

Strongly Convex Function

Definition: Strongly Convex

f is *strongly convex* with parameter $m > 0$ if $\mathrm{dom}(f)$ is convex and

$$f(\lambda x + (1-\lambda)y) \le \lambda f(x) + (1-\lambda) f(y) - \frac{m}{2}\lambda(1-\lambda)\|x-y\|^2$$

holds for all $x, y \in \mathrm{dom}(f)$, $\lambda \in [0,1]$.

- f is strongly convex if and only if

$$f(x + t(y-x)) - \frac{m}{2} t^2 \|x-y\|^2$$

  is a convex function of t, for all $x, y \in \mathrm{dom}\, f$.
- f is strongly convex if and only if the smallest eigenvalue of $\nabla^2 f(x)$ is greater than or equal to m.
- Without loss of generality, we can take $\|\cdot\| = \|\cdot\|_2$.
- However, the strong convexity parameter m depends on the norm used.

Linear Convergence of GD in Strongly Convex Case

Theorem

Suppose that ∇f is Lipschitz continuous with parameter L and f is strongly convex with parameter $m > 0$. Let x^* be the unique global minimizer of $\min_x f(x)$ (since strongly convex). Then the gradient descent update sequence $\{x^k\}_k$ converges linearly to x^*. It further follows

$$f(x^{k+1}) - f(x^*) \leq \eta \cdot \Big(f(x^k) - f(x^*) \Big), \quad \forall k \geq 0.$$

for some $\eta \in (0, 1)$ depending on step size methods.

Summary

① f is convex + ∇f Lipschitz continuous

$\Rightarrow$ GD converges sublinearly ($\bar{\alpha} < \frac{1}{L}$)

② f is strongly convex + ∇f Lipschitz continuous

$\Rightarrow$ GD converges linearly (constant / exact step size / backtracking)

Nice Property of Gradient Descent Method

The convergence of gradient descent method doesn't depend on the initial point

- No matter where it starts, it will always converge to a local minimizer (when $f(\cdot)$ is convex, it converges to a global minimizer).
- We call this property *global convergence* property.

Global Convergence

We use the same function as example

$$f(\boldsymbol{x}) = \exp(x_1 + x_2) + x_1^2 + 3x_2^2 - x_1 x_2$$

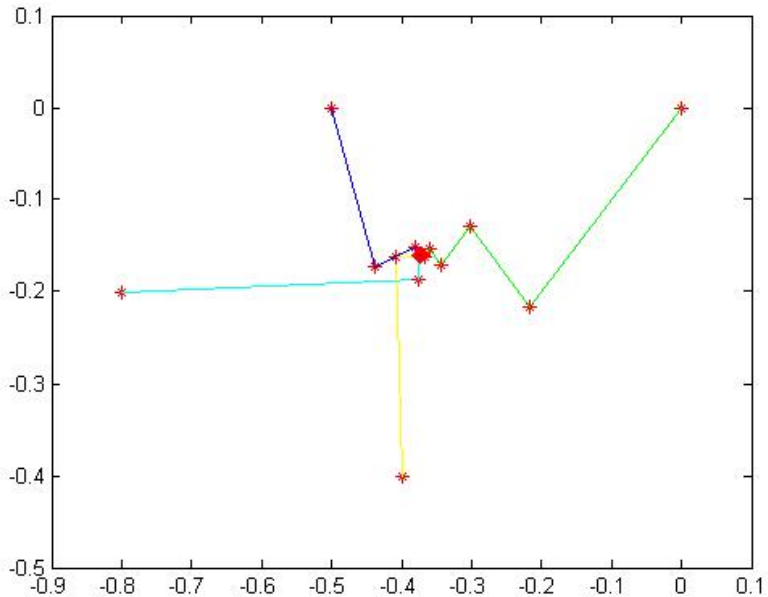


Figure 3: Exact Line Search

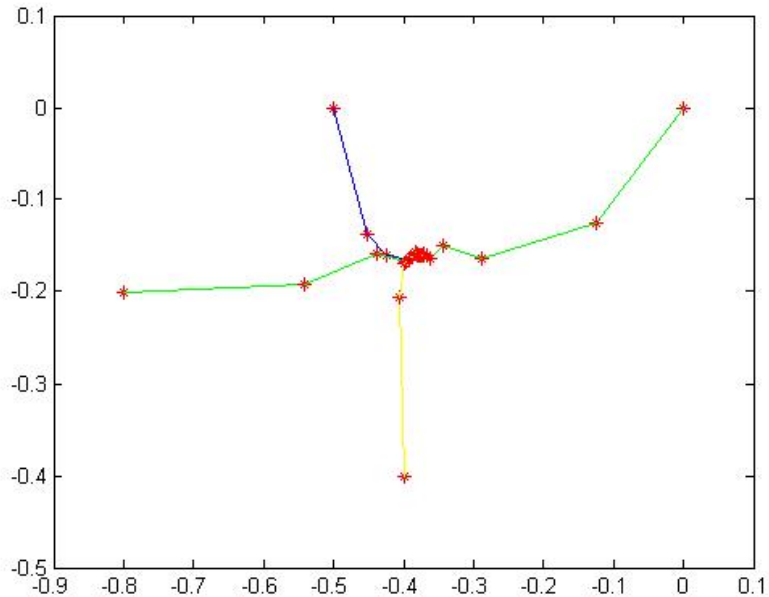


Figure 4: Backtracking Line Search

Pros and Cons of Gradient Descent Method

Pros:

- Easy to understand and implement
- Only need to know the first-order (gradient) information
- Globally convergent, doesn't depend on the initial point

Cons:

- Calculating Gradient takes time ← derive gradient / calculate gradient ⇒ subGradient descent
- Convergence speed may not be fast enough: Linear convergence

↓ Newton's Method

References and further reading

- S. Boyd and L. Vandenberghe (2004), *Convex Optimization*, Chapter 9
- T. Hastie, R. Tibshirani and J. Friedman (2009), *The Elements of Statistical Learning*, Chapters 10 and 16
- Y. Nesterov (1998), *Introductory Lectures on Convex Optimization: A Basic Course*, Chapter 2
- L. Vandenberghe, Lecture notes for EE 236C, UCLA

Outline

Intuition 1

- First order Taylor expansion for linearly approximating $f(x)$:

$$f(x + \Delta x) = f(x) + \nabla f(x)\Delta x + o(|\Delta x|).$$

- Take gradient on both sides:

$$\nabla f(x + \Delta x) = \nabla f(x) + \nabla^2 f(x)\Delta x + o(|\Delta x|).$$

- If we want $x^* = x + \Delta x$, then

$$\nabla f(x^*) = \nabla f(x + \Delta x) = \nabla f(x) + \nabla^2 f(x)\Delta x = 0$$

$$\Rightarrow \quad x^\star - x = \Delta x = -[\nabla^2 f(x)]^{-1}\nabla f(x)$$

- Use this direction in each iteration

$$x^{k+1} = x^k - [\nabla^2 f(x)]^{-1}\nabla f(x)$$

Intuition 2

- Let x^k be the current iterate and consider a second-order Taylor approximation of $f(x)$ at x^k, i.e.

$$g(x) = f(x^k) + \nabla f(x^k)^\top (x - x^k) + \frac{1}{2}(x - x^k)^\top \nabla^2 f(x^k)(x - x^k).$$

- Choose the next iterate as the solution that minimizes the approximate function $g(x)$.
- Setting $\nabla g(x) = 0$, we get the linear system

$$\nabla f(x^k) + \nabla^2 f(x^k)(x - x^k) = 0.$$

- If the Hessian is non-singular, a solution to the above system is well-defined, and we set

$$x^{k+1} = x^k - [\nabla^2 f(x_k)]^{-1} \nabla f(x^k).$$

Newton's Method in High Dimensional Case

- Newton's step:

$$\boldsymbol{x} = \boldsymbol{x}^k - (\nabla^2 f(\boldsymbol{x}^k))^{-1} \nabla f(\boldsymbol{x}^k)$$

- Direction: $\boldsymbol{d}^k = -(\nabla^2 f(\boldsymbol{x}^k))^{-1} \nabla f(\boldsymbol{x}^k)$.
- Then Newton's step direction is a descent direction when f is convex.
- Newton's method may not converge unless the starting point is close.
- One way to help convergence is to use a step size parameter α_k in

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

  where $\boldsymbol{d}^k = -(\nabla^2 f(\boldsymbol{x}^k))^{-1} \nabla f(\boldsymbol{x}^k)$ is the Newton's step.
- One can use backtracking line search to determine α_k.
- In all above derivations, we assume that $\nabla^2 f(\boldsymbol{x}^k)$ is invertible.

Complete Procedure of Newton's Method

Start from a point $\boldsymbol{x}^0$, set tolerance $\epsilon > 0$ and $k = 0$.

1. Compute the Newton's step:

$$\boldsymbol{d}^k = -(\nabla^2 f(\boldsymbol{x}^k))^{-1} \nabla f(\boldsymbol{x}^k)$$

   If $||\nabla f(\boldsymbol{x}^k)|| < \epsilon$, then output $\boldsymbol{x}^k$, otherwise continue;
2. Choose step size α_k by backtracking line search
3. $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \alpha_k \boldsymbol{d}^k$, $k = k + 1$, go back to step 1

Linear Convergence vs Quadratic Convergence

Remember the gradient descent method can have linear convergence rate (strongly convex case):

$$|x^{k+1} - x^*| \leq \eta |x^k - x^*|.$$

Now, Newton's method has quadratic convergence rate:

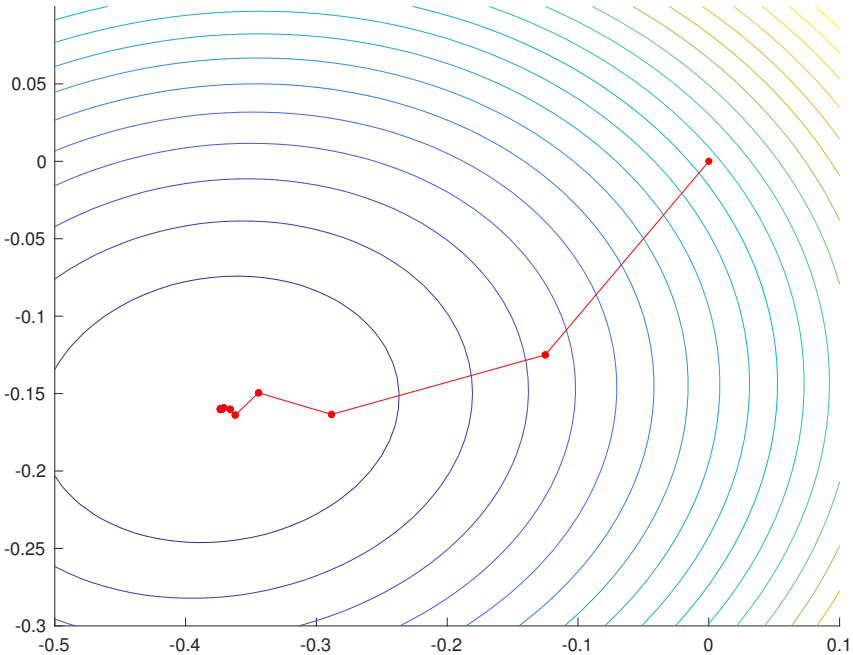
$$|x^{k+1} - x^*| \leq C |x^k - x^*|^2.$$

Convergence of Newton's Method

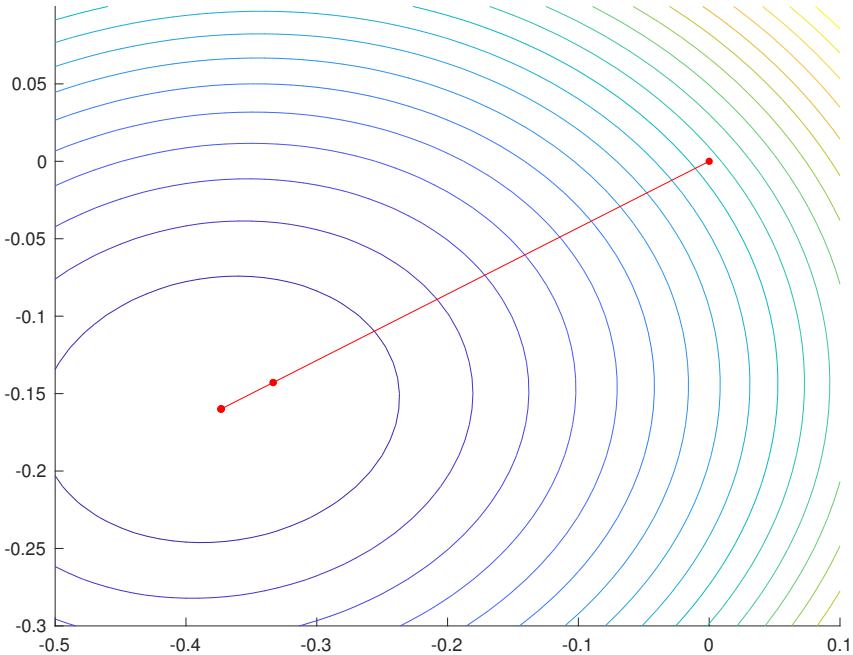
- Quadratic convergence in the neighborhood of a strict local minimum (under some conditions). Quadratic convergence is much faster than linear convergence in gradient descent.

Example: Minimize

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



(a) Gradient Descent



(b) Newton's Method

Newton's Method is Fast in Iteration Count, But ...

It requires computing the second-order derivative (Hessian) in each iteration.

- It might be computationally expensive to do so, especially if the second-order derivative doesn't have a closed-form (which is the case for many useful applications)
- It also requires a lot of space to store the Hessian matrix (If it is an n-dimensional problem, then we need n^2 space, comparing to n space required for gradient descent method).
- It requires much more matrix computation in each iteration
- If the Hessian is singular (or close to singular) at some iteration, we cannot proceed.
- Newton's method may diverge, even for convex functions.

Quasi-Newton Methods

- Goal: Avoids computation of the Hessian and its inverse.
- Update iterate using

$$x^{k+1} = x^k - \alpha_k H_k \nabla f(x^k),$$

where α_k is the step size and H_k is an approximation to $[\nabla^2 f(x^k)]^{-1}$.

- Since H_{k+1}^{-1} approximates the Hessian,

$$H_{k+1}(\nabla f(x^{k+1}) - \nabla f(x^k)) = x^{k+1} - x^k.$$

- A widely used formula is the Broyden–Fletcher–Goldfarb–Shanno (BFGS) formula:

$$H_{k+1} = H_k - \frac{d_k g_k^\top H_k + H_k g_k d_k^\top}{d_k^\top g_k} + \left(1 + \frac{g_k^\top H_k g_k}{d_k^\top g_k}\right) \frac{d_k d_k^\top}{d_k^\top g_k},$$

where $g_k = \nabla f(x^{k+1}) - \nabla f(x^k)$ and $d_k = x^{k+1} - x^k$.

Quiz

What are the update equations in each iteration for gradient descent and Newton's method for a maximization problem?