Optimization

Will not be tested in final exam

What is Optimization?

- Optimization means finding the best solution among many feasible solutions that are available to us
- Feasible solutions are those that satisfy all the constraints in the optimization problem

Mathematical Formulation

Optimization Problem

- Objective Function (Cost Function, Performance Index): The function that is to be maximized or minimized --- $f(x_1, x_2, x_3, \cdots, x_n)$
- Decision Variables (Design Variables): Variables in the objective function --- $x_k,\ k = 1, 2, 3, \cdots n$
- It suffices to only study either Maximization or Minimization
- To turn Maximization into Minimization (and vice versa): $f \rightarrow -f$
- Unless otherwise specified, we study minimization in this course

Mathematical Formulation

- Objective function $f(\vec{x})$
- subject to constraints:
- $$g_i(\vec{x}) \geq 0 \qquad i = 1, 2, \cdots, m$$
  $$h_j(\vec{x}) = 0 \qquad j = 1, 2, \cdots, r$$
- where
- $$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \\ \vdots \\ x_n \end{bmatrix}$$

Remark

- The functions f, g_i, h_j are all assumed to be differentiable
- g_i: Inequality constraints
- h_j: Equality constraints

Mathematical Formulation

- Feasible point of variables: An $\vec{x}$ that satisfies all the constraints
- Feasible set of variables X: The set of all feasible points
- Unconstrained Optimization: Problem does not have any constraints $\rightarrow X = \mathbb{R}^n$
- Linear Programming Problem (LPP): Objective function and constraints are linear functions of all the variables

Optimal Solution(s) --- Unconstrained

- A vector $\vec{x}^*$ is an unconstrained local minimum of f if $\exists \varepsilon > 0$ s.t.

$$f\left(\vec{x}^*\right) \leq f\left(\vec{x}\right) \quad \forall \vec{x} \text{ with } \left|\vec{x} - \vec{x}^*\right| < \varepsilon$$

- A vector $\vec{x}^*$ is an unconstrained global minimum of f if

$$f\left(\vec{x}^*\right) \leq f\left(\vec{x}\right) \quad \forall \vec{x} \in \mathbb{R}^n$$

- A vector $\vec{x}^*$ is a strict unconstrained local minimum of f if $\exists \varepsilon > 0$ s.t.

$$f\left(\vec{x}^*\right) < f\left(\vec{x}\right) \quad \forall \vec{x} \text{ with } 0 < \left|\vec{x} - \vec{x}^*\right| < \varepsilon$$

- A vector $\vec{x}^*$ is a strict unconstrained global minimum of f if

$$f\left(\vec{x}^*\right) < f\left(\vec{x}\right) \quad \forall \vec{x} \in \mathbb{R}^n, \ \vec{x} \neq \vec{x}^*$$

Optimal Solution(s) --- Constrained

- A vector $\vec{x}^*$ is a constrained local minimum of f over X if $\exists \varepsilon > 0$ s.t.

$$f\left(\vec{x}^*\right) \leq f\left(\vec{x}\right) \quad \forall \vec{x} \in X \text{ with } \left|\vec{x} - \vec{x}^*\right| < \varepsilon$$

- A vector $\vec{x}^*$ is a constrained global minimum of f over X if

$$f\left(\vec{x}^*\right) \leq f\left(\vec{x}\right) \quad \forall \vec{x} \in X$$

- A vector $\vec{x}^*$ is a strict constrained local minimum of f over X if $\exists \varepsilon > 0$ s.t.

$$f\left(\vec{x}^*\right) < f\left(\vec{x}\right) \quad \forall \vec{x} \in X \text{ with } 0 < \left|\vec{x} - \vec{x}^*\right| < \varepsilon$$

- A vector $\vec{x}^*$ is a strict constrained global minimum of f if

$$f\left(\vec{x}^*\right) < f\left(\vec{x}\right) \quad \forall \vec{x} \in X,\ \vec{x} \neq \vec{x}^*$$

Review: Optimization of Single-Variable Functions

Local and Global Minimum

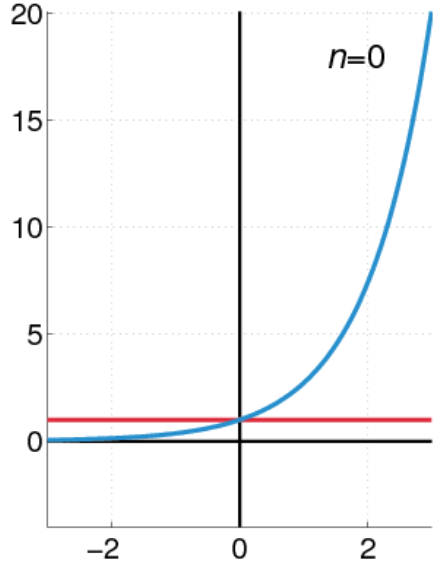
- Consider unconstrained optimization, $f: \mathbb{R} \to \mathbb{R}$
- Then a necessary condition for global minimum is that it must be a local minimum

Taylor's Theorem

Let $k \geq 1$ be an integer and let the function $f : \mathbb{R} \to \mathbb{R}$ be k times differentiable at the point $x \in \mathbb{R}$. Then

$$f(x+\Delta x) = f(x) + f'(x)\Delta x + \frac{1}{2!} f''(x)(\Delta x)^2 + \cdots + \frac{1}{k!} f^{(k)}(x)(\Delta x)^k + o\left((\Delta x)^k\right).$$

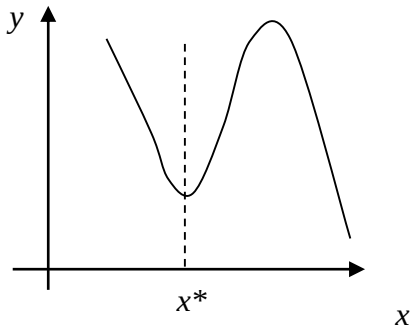
Proof omitted!



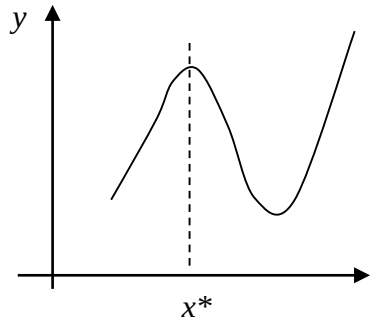
Necessary Condition on First Derivative for Optimality

- If f is differentiable at x^*, a necessary condition for $f(x^*)$ to be minimum is

$$f'\left(x^*\right)=0$$



Local minimum



Local maximum

Second Derivative Test

- The condition $f'(x^*) = 0$ is necessary but not sufficient
- Need to consider second derivative, if exists

Necessary Condition on Second Derivative for Optimality

- If $f''(x^*)$ exists, a necessary condition for $f(x^*)$ to be minimum is

$$f'\left(x^*\right)=0 \text{ and } f''\left(x^*\right)\geq 0$$

Sufficient Condition on Second Derivative for Optimality

- If $f''(x^*)$ exists, a sufficient condition for $f(x^*)$ to be minimum is

$$f'\left(x^*\right)=0 \text{ and } f''\left(x^*\right)>0$$

- In this case, $f(x^*)$ is a strict minimum

Second Derivative Test

- In summary, if $f'(x^*) = 0$ and
  - $f''(x^*) > 0$, it is strict minimum
  - $f''(x^*) < 0$, it is strict maximum
  - $f''(x^*) = 0$, cannot be determined
- When $f''(x^*) = 0$, need to look at higher order derivatives

Higher Order Derivative Tests

- If $f'(x^*) = f''(x^*) = 0$ and $f'''(x^*) \neq 0$, then it is neither minimum nor maximum
- If $f'(x^*) = f''(x^*) = f'''(x^*) = 0$ and
  - $f^{(4)}(x^*) > 0$, then it is a strict minimum
  - $f^{(4)}(x^*) < 0$, then it is a strict maximum
  - $f^{(4)}(x^*) = 0$, then it cannot be determined, need to look at higher order derivatives
- …

Derivative Tests in general:

A function does not attain local extremum (maximum or minimum) at a point x^* if the leading order term of the Taylor series expansion about this point is odd order

$$f'(x^*) = f''(x^*) = \cdots = f^{(2m)}(x^*) = 0 \quad \text{AND} \quad f^{(2m+1)}(x^*) \neq 0$$

for some non-negative integer m

A function attains local extremum (maximum or minimum) at a point x^* if the leading order term of the Taylor series expansion about this point is even order

$$f'(x^*) = f''(x^*) = \cdots = f^{(2m-1)}(x^*) = 0 \quad \text{AND} \quad f^{(2m)}(x^*) \neq 0$$

for some natural number m

It is a local maximum if $f^{(2m)}(x^*) < 0$

It is a local minimum if $f^{(2m)}(x^*) > 0$

Taylor's Theorem for Multivariate Functions

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a k-times-differentiable function at the point $\vec{x} \in \mathbb{R}^n$.

Then

$$f\left(\vec{x}+\Delta\vec{x}\right) = f\left(\vec{x}\right) + D^{(1)} f\left(\vec{x}\right)\Delta\vec{x} + \frac{1}{2!} D^{(2)} f\left(\vec{x}\right)\left(\Delta\vec{x}\right)^2 + \frac{1}{3!} D^{(3)} f\left(\vec{x}\right)\left(\Delta\vec{x}\right)^3 + \cdots \frac{1}{k!} D^{(k)} f\left(\vec{x}\right)\left(\Delta\vec{x}\right)^k + o\left(\left|\Delta\vec{x}\right|^k\right).$$

Taylor Series for Multivariable Functions

$$f\left(\vec{x}+\Delta\vec{x}\right)=f\left(\vec{x}\right)$$
$$+\frac{\partial f}{\partial x_1}\Delta x_1+\frac{\partial f}{\partial x_2}\Delta x_2+\cdots+\frac{\partial f}{\partial x_n}\Delta x_n$$
$$+\frac{1}{2}\frac{\partial^2 f}{\partial x_1^2}\left(\Delta x_1\right)^2+\frac{1}{2}\frac{\partial^2 f}{\partial x_1\partial x_2}\Delta x_1\Delta x_2+\cdots+\frac{1}{2}\frac{\partial^2 f}{\partial x_1\partial x_n}\Delta x_1\Delta x_n$$
$$+\frac{1}{2}\frac{\partial^2 f}{\partial x_2\partial x_1}\Delta x_2\Delta x_1+\frac{1}{2}\frac{\partial^2 f}{\partial x_2^2}\left(\Delta x_2\right)^2+\cdots+\frac{1}{2}\frac{\partial^2 f}{\partial x_2\partial x_n}\Delta x_2\Delta x_n$$
$$+\cdots$$
$$+\frac{1}{2}\frac{\partial^2 f}{\partial x_n\partial x_1}\Delta x_n\Delta x_1+\cdots+\frac{1}{2}\frac{\partial^2 f}{\partial x_n^2}\left(\Delta x_n\right)^2$$
$$+\cdots$$

Taylor Series for Multivariable Functions

$$f\left(\vec{x}+\Delta\vec{x}\right)=f\left(\vec{x}\right)$$

$$+\begin{bmatrix}\frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \cdots & \frac{\partial f}{\partial x_n}\end{bmatrix}\begin{bmatrix}\Delta x_1\\ \Delta x_2\\ \vdots\\ \Delta x_n\end{bmatrix}$$

$$+\frac{1}{2}\begin{bmatrix}\Delta x_1 & \Delta x_2 & \cdots & \Delta x_n\end{bmatrix}\begin{bmatrix}\frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1\partial x_2} & \frac{\partial^2 f}{\partial x_1\partial x_3}\\ \frac{\partial^2 f}{\partial x_2\partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \frac{\partial^2 f}{\partial x_2\partial x_3}\\ \frac{\partial^2 f}{\partial x_3\partial x_1} & \frac{\partial^2 f}{\partial x_3\partial x_2} & \frac{\partial^2 f}{\partial x_3^2}\end{bmatrix}\begin{bmatrix}\Delta x_1\\ \Delta x_2\\ \vdots\\ \Delta x_n\end{bmatrix}$$

$$+\cdots$$

$$f\left(\vec{x}+\Delta\vec{x}\right)=f\left(\vec{x}\right)+\vec{\nabla}f\left(\vec{x}\right)\cdot\Delta\vec{x}+\frac{1}{2}\Delta\vec{x}\cdot\mathbf{H}\Delta\vec{x}+\cdots$$

Schwarz's Theorem (Clairaut's Theorem)

If $(a_1, \cdots, a_n) \in \mathbb{R}^n$, $\Omega \subseteq \mathbb{R}^n$, some neighborhood of $(a_1, \cdots, a_n)$ is contained in Ω,

$f : \Omega \to \mathbb{R}$

and f has continuous second partial derivatives at the point $(a_1, \cdots, a_n)$, then $\forall i, j \in \{1, 2, \cdots, n\}$,

$$\frac{\partial^2}{\partial x_i \partial x_j} f(a_1, \cdots, a_n) = \frac{\partial^2}{\partial x_j \partial x_i} f(a_1, \cdots, a_n).$$

- AKA Clairaut's Theorem or Young's Theorem on equality of mixed partials

Symmetric Hessian Matrix

- In most “real-life” circumstances, f satisfies the Schwarz’s condition and hence by Schwarz’s Theorem, the Hessian Matrix of f is symmetric

Necessary Condition on Gradient Vector for Optimality

- If the partial derivatives of a function f at $\vec{x}^*$ exist, a necessary condition for $f(\vec{x}^*)$ to be minimum is

$$\vec{\nabla} f\left(\vec{x}^*\right) = \vec{0}$$

- We refer to a vector $\vec{x}^*$ at which the gradient is zero a stationary point

Taylor Expansion and Quadratic Form

- Because $\vec{\nabla} f(\vec{x}^*) = \vec{0}$, hence

$$f(\vec{x}) = f(\vec{x}^*) + \frac{1}{2}(\vec{x} - \vec{x}^*) \cdot \mathbf{H}(\vec{x}^*)(\vec{x} - \vec{x}^*) + o\left(\left|\vec{x} - \vec{x}^*\right|^2\right)$$

- One can shift the origin to $\vec{x}^*$ by defining $\vec{\tilde{x}} = \vec{x} - \vec{x}^*$, then

$$q(\vec{\tilde{x}}) = f(\vec{x}) - f(\vec{x}^*) \approx \frac{1}{2}\vec{\tilde{x}} \cdot \mathbf{H}(\vec{x}^*)\vec{\tilde{x}} = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} H_{ij}\tilde{x}_i\tilde{x}_j$$

  which is a quadratic form in $\tilde{x}_1, \tilde{x}_2, \cdots \tilde{x}_n$.

Definiteness of a Real Symmetric Matrix

- A $n \times n$ real symmetric matrix $\mathbf{A}$ is said to be positive definite if

$$\vec{x} \cdot \mathbf{A}\vec{x} > 0 \quad \forall \vec{x} \in \mathbb{R}^n \setminus \vec{0}$$

- A $n \times n$ real symmetric matrix $\mathbf{A}$ is said to be positive semidefinite if

$$\vec{x} \cdot \mathbf{A}\vec{x} \geq 0 \quad \forall \vec{x} \in \mathbb{R}^n$$

Necessary Condition on Hessian for Optimality

- Let $\vec{x}^*$ be an unconstrained local minimum of $f: \mathbb{R}^n \to \mathbb{R}$, and assume that f is twice continuously differentiable at $\vec{x}^*$, then

$$\vec{\nabla} f\left(\vec{x}^*\right) = \vec{0}$$

$$\mathbf{H}\left(\vec{x}^*\right) = \vec{\nabla}^2 f\left(\vec{x}^*\right) \text{ is positive semidefinite}$$

- Corollary: All eigenvalues of the Hessian are non-negative in this case

Sufficient Condition on Hessian for Optimality

- If $f: \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable at $\vec{x}^*$ and

$$\vec{\nabla} f\left(\vec{x}^*\right) = \vec{0}$$

$$\mathbf{H}\left(\vec{x}^*\right) = \vec{\nabla}^2 f\left(\vec{x}^*\right) \text{ is positive definite}$$

  then $\vec{x}^*$ is a strict unconstrained local minimum of f

- Local minima that satisfy the above sufficiency conditions are called nonsingular, otherwise they are called singular
- Corollary: All eigenvalues of the Hessian are positive in this case

1-D Optimization Algorithms

1-D Optimization

- $f: \mathbb{R} \rightarrow \mathbb{R}$
- 1-D optimization algorithms form the basic building blocks for multivariable algorithms
- For multivariable functions, line search methods along a certain direction $\vec{d}$ by minimizing $F(\alpha) = f\left(\vec{x} + \alpha\vec{d}\right)$ → 1-D optimization
- Can be classified into gradient-based and non-gradient-based algorithms

1-D Optimization Methods

- Examples of gradient-based methods:
  - Relaxation Method
  - Bisection Method
  - Newton Raphson Method
- Examples of non-gradient-based methods:
  - Quadratic Interpolation
  - Golden Section Method

Relaxation method

- Find minimum of $g(x)$ by solving
$$g'(x) = 0$$
- Suppose it can be rewritten in the form
$$x = f(x)$$
- One elementary method that gives good answers in many cases is simply to iterate the equation. That is, we guess an initial value of the unknown variable x, plug it in on the right-hand side of the equation and get a new value x' on the left-hand side

Example

- Then we repeat the process, taking this value and feeding it in on the right again to get

$$x = 2 - e^{-x}$$

- If we keep on doing this, and if we are lucky, then the value will converge to a fixed point of the equation, meaning it stops changing. In this particular case, that's exactly what happened
- Here is a python code to perform the iterations of the calculations:

```
from math import exp
x = 1.0
for k in range(10):
    x = 2 – exp(-x)
    print(x)
```

- If a process like this converges to a fixed point, then the value of x you get is necessarily a solution to the original equation

Bisection Method

- Also known as the binary search method
- More robust than relaxation method that, given root(s) exists inside a specified interval, the method can always find at least one of the roots
- Find minimum of $g(x)$ by solving

$$g'(x) = 0$$

- Suppose it can be rewritten in the form

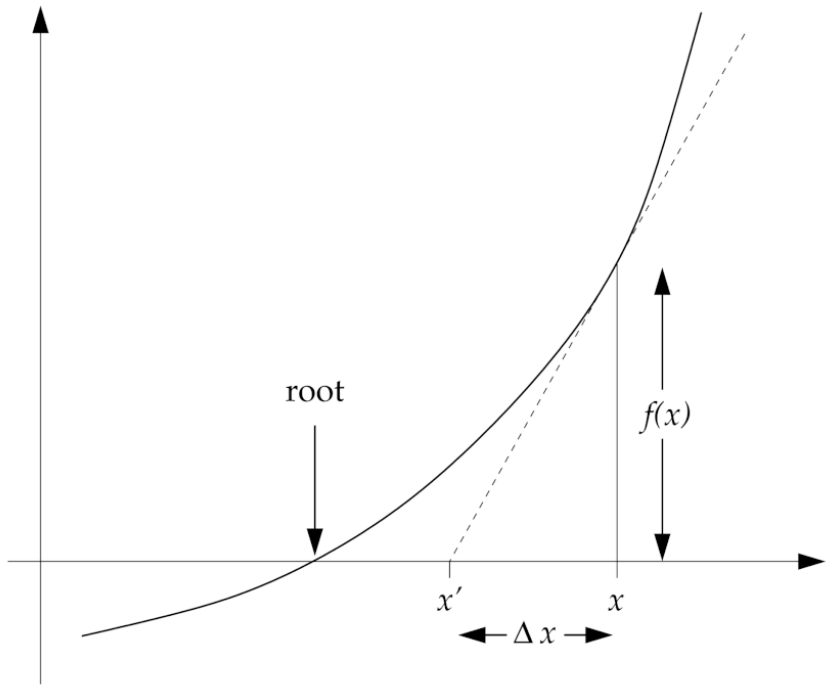
$$x = f(x)$$

Algorithm

1. Given an initial pair of points x_1, x_2, check that $f(x_1)$ and $f(x_2)$ have opposite signs. Also choose a target accuracy for the answer you want
2. Calculate the midpoint $x' = \frac{1}{2}(x_1 + x_2)$ and evaluate $f(x')$
3. If $f(x') = 0$, stop. We have found a root
4. If $f(x')$ has the same sign as $f(x_1)$ then set $x_1 = x'$. Otherwise set $x_2 = x'$
5. If $|x_1 - x_2|$ is greater than the target accuracy, repeat from step 2. Otherwise, calculate $\frac{1}{2}(x_1 + x_2)$ once more and this is the final estimate of the position of the root

Newton Raphson Method

- Find minimum of $g(x)$ by solving
$$g'(x) = 0$$
- Suppose it can be rewritten in the form
$$x = f(x)$$
- The figure shows a graphical representation of the Newton's method for a single variable
- We start with a single guess x and then use the slope at that position (dotted line) to extrapolate and make another guess x', which will usually be better than the first guess, although it is possible, in unlucky cases, for it to be worse.
- Form the figure we see that slope at x is
$$f'(x) = \frac{f(x)}{\Delta x}$$
and the formula for our new guess x' for the position of the root is thus
$$x' = x - \Delta x = x - \frac{f(x)}{f'(x)}$$



Newton Raphson Method

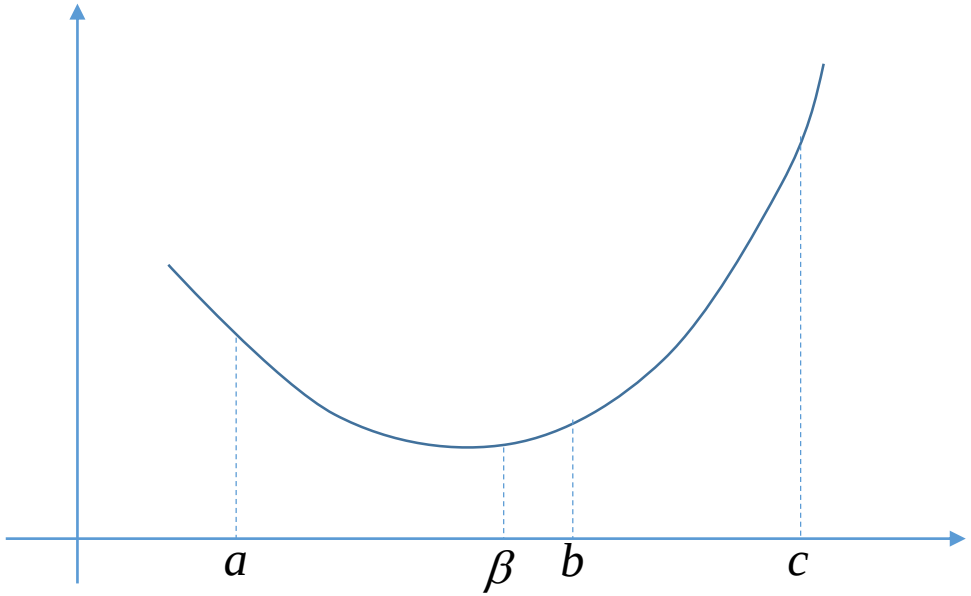
- This approach requires us to know the derivative $f'(x)$
- If we know $f'(x)$ then putting the method to work on the computer is straightforward: we just start with a guess x and use the equation repeatedly to get better and better estimates of the position of the root
- For functions with more than one root Newton's method typically converges to a root near the starting value of x, so one can use the method to find different roots by choosing different starting points

Quadratic Interpolation

- Minimize $f(x)$
- Three-Point-Pattern: $a < b < c$ and $f(a) > f(b) < f(c)$
- It can be seen that a local minimum of f must lie between a and c
- Fit a quadratic polynomial by Lagrange interpolation formula

$$q(x) = \frac{(x-b)(x-c)}{(a-b)(a-c)} f(a) + \frac{(x-a)(x-c)}{(b-a)(b-c)} f(b) + \frac{(x-a)(x-b)}{(c-a)(c-b)} f(c)$$

$$\Rightarrow \beta = \frac{1}{2} \frac{f(a)(c^2-b^2) + f(b)(a^2-c^2) + f(c)(b^2-a^2)}{f(a)(c-b) + f(b)(a-c) + f(c)(b-a)}$$



Quadratic Interpolation

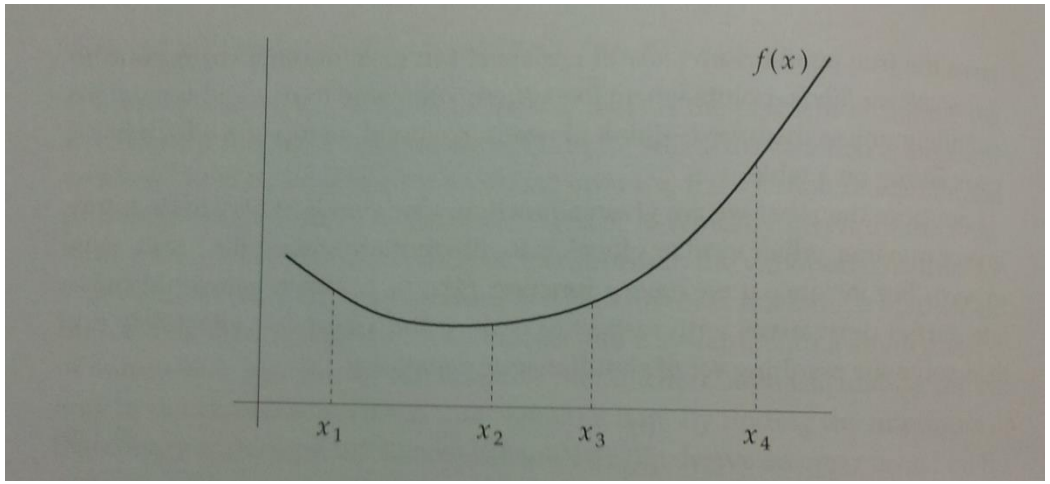
- Step 1: Determination of Initial Three-Point Pattern
  - We search along the line until we find three successive points a, b, and c with $a < b < c$ such that $f(a) > f(b)$ and $f(b) < f(c)$
  - We assume that this stage can be carried out
- Step 2: Updating the Current Three-Point Pattern
  - Given the current three-point pattern a, b and c, we fit a quadratic polynomial and determine its unique minimum $\beta \in (a, c)$ by

$$\beta = \frac{1}{2} \frac{f(a)\left(c^2 - b^2\right) + f(b)\left(a^2 - c^2\right) + f(c)\left(b^2 - a^2\right)}{f(a)(c - b) + f(b)(a - c) + f(c)(b - a)}$$

  - Form a new three-point pattern as follows:
    - If $\beta < b$ and $f(\beta) > f(b)$, then replace a by β
    - If $\beta > b$ and $f(\beta) > f(b)$, then replace c by β
    - If $\beta < b$ and $f(\beta) < f(b)$, then replace c by b and b by β
    - If $\beta > b$ and $f(\beta) < f(b)$, then replace a by b and b by β
- The computation is terminated when the length of the three-point pattern is smaller than a certain tolerance

Golden Section Method

- Minimize $f(x)$
- Golden section method is similar in spirit to bisection method for finding the roots of a function
- In bisection method we used two points to bracket a root and a third point in the middle to narrow the search
- The equivalent construction for a minimum involves three points that bracket the minimum plus a fourth as shown in the figure



Golden Section Method

- Suppose that at least one of the values $f(x_2)$ and $f(x_3)$ at the two intermediate points is less than the values $f(x_1)$ and $f(x_4)$ at the outer points
- In that case we know that there must be at least one minimum of the function between x_1 and x_4, because the function goes down and up again
- We can narrow down the location of this minimum by comparing the values at x_2 and x_3 to see which is smaller
- If $f(x_2)$ is smaller, then the minimum must lie between x_1 and x_3 because, again, the function goes down and up again
- Conversely, if $f(x_3)$ is smaller, then the minimum must lie between x_2 and x_4
- By this process we can narrow down our search for the minimum to a smaller range, encompassing three of the previous four points
- We then add a fourth point to those three, and repeat the process again
- We keep on repeating until the range in which the root falls becomes smaller than our required accuracy, then stop and take the middle of the range as our result for the position of the minimum

Golden Section Method

- Since we hope that the efficiency of searching steps to be constant, we need to control the distribution of the four points
- It can be shown that

$$z = \frac{x_3 - x_1}{x_2 - x_1}$$

where

$$z = \frac{1 + \sqrt{5}}{2} = 1.618\ldots$$

- This value of z is called the golden ratio, and it is after this ratio that golden ratio search is named
- The golden ratio crops up repeatedly in many branches of physics and mathematics, as well as finding uses in art and architecture

Algorithm

1. Choose two initial outside points x_1 and x_4, then calculate the interior points x_2 and x_3 according to the golden ratio rule above. Evaluate $f(x)$ at each of the four points and check that at least one of the values at x_2 and x_3 is less than the values at both x_1 and x_4. Also choose a target accuracy for the position of the minimum
2. If $f(x_2) < f(x_3)$ then minimum lies between x_1 and x_3. In this case, x_3 becomes the new value of x_4, x_2 becomes the news x_3, and there will be a new value for x_2, chosen once again according to the golden ratio rule. Evaluate $f(x)$ at this new point
3. Otherwise, the minimum lies between x_2 and x_4. Then x_2 becomes the new x_1, x_3 becomes the new x_2, and there will be a new value for x_3. Evaluate $f(x)$ at this new point
4. If $x_4 - x_1$ is greater than the target accuracy, repeat from step 2. Otherwise, calculate $\frac{1}{2}(x_2 + x_3)$ and this is the final estimate of the position of the minimum[2]

Unconstrained Optimization

Introduction

- Most optimization problems are constrained
- Example of unconstrained optimization problem: Data fitting --- fitting a curve on measured data
- Algorithms of unconstrained optimization can be used to solve constrained optimizations problems as well
- Example: Modifying the objective functions by including a penalty term in case constraints are violated

Introduction

- Solution methods can be broadly classified into
  - Gradient-based search methods
  - Non-gradient-based search methods
- Gradient-based methods: Require gradient and/or Hessian information in determining the search direction
  - Steepest descent method
  - Newton's method
  - Quasi-Newton methods
    - Davidon-Fletcher-Powell (DFP) method
    - Broyden-Fletcher-Goldfarb-Shanno (BFGS) method
  - Conjugate-gradient method
    - Levenberg-Marquardt method
  - Gauss-Newton method
- Non-gradient-based methods: Do not require gradient or Hessian information in determining the search direction. Instead usually guided by function evaluations and search directions computed from earlier iterations
  - Simplex method (Nelder-Mead algorithm)
  - Conjugate-direction method (Powell)

Unidirectional Search (Line Search)

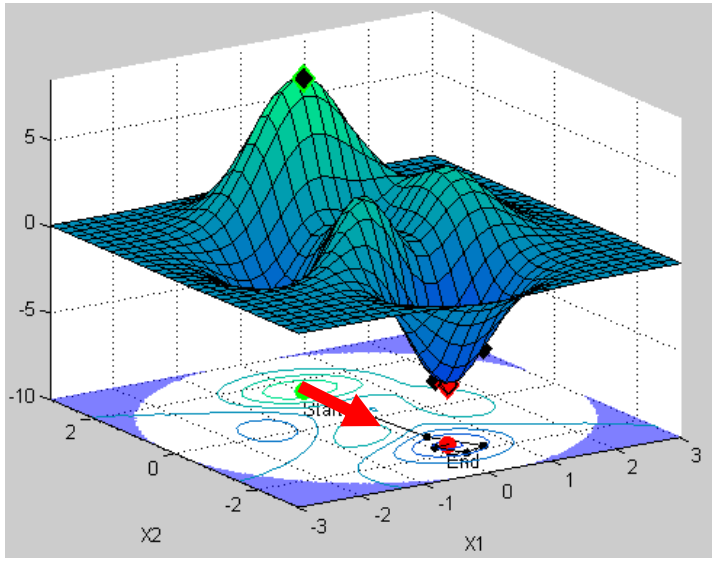
- Initial starting point $\vec{x}_0 \in \mathbb{R}^n$
- Search for minimum along the line in direction $\vec{d} \neq \vec{0}$
- Define

$$\vec{x}(\alpha) = \vec{x}_0 + \alpha\vec{d}, \qquad \alpha \in \mathbb{R}$$

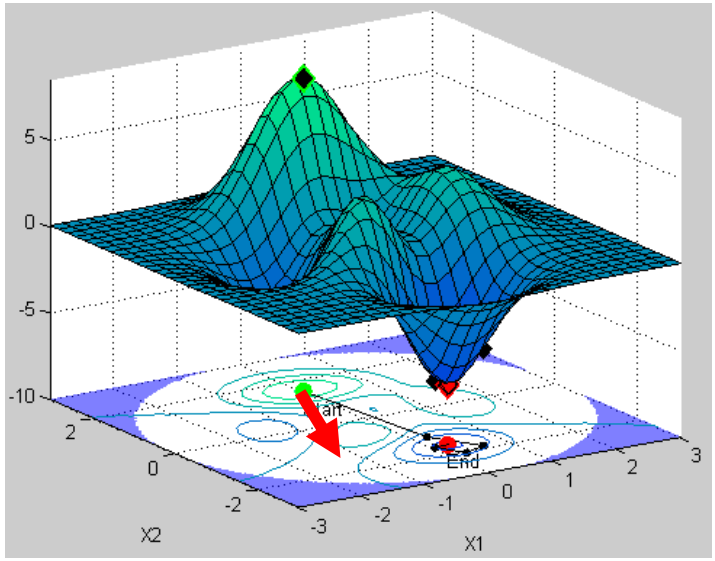
- Reduced to 1-D minimization of the single-variable function $F: \mathbb{R} \to \mathbb{R}$

$$F(\alpha) = f\left(\vec{x}(\alpha)\right) = f\left(\vec{x}_0 + \alpha\vec{d}\right)$$

- Cannot get actual minimum unless $\vec{d}$ is in the correct direction



YES



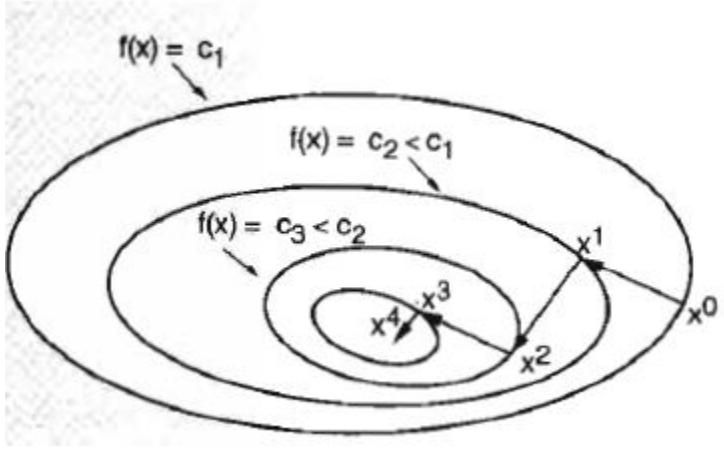
NO

Iterative Descent

- Continuously differentiable function $f: \mathbb{R}^n \to \mathbb{R}$
- Start at some point $\vec{x}_0$ (an initial guess)
- Successively generate $\vec{x}_1, \vec{x}_2, \ldots$, such that

$$f(\vec{x}_{k+1}) < f(\vec{x}_k), \qquad k = 0, 1, \ldots,$$

- Hope that f will decrease all the way to its minimum



Step Directions and Stepsizes

- Given $\vec{x} \in \mathbb{R}^n$ with $\vec{\nabla} f(\vec{x}) \neq \vec{0}$, consider the step going from $\vec{x}$ to
$$\vec{x}(\alpha) = \vec{x} + \alpha \vec{d}, \qquad \alpha \geq 0, \vec{d} \neq \vec{0}$$
- When $\alpha > 0$, the direction is $\vec{d}$ and the stepsize is $\alpha \left| \vec{d} \right|$
- First order Taylor series expansion around $\vec{x}$:
$$f\left(\vec{x}(\alpha)\right) = f(\vec{x}) + \vec{\nabla} f(\vec{x})^T (\vec{x}(\alpha) - \vec{x}) + o(|\vec{x}(\alpha) - \vec{x}|)$$
$$= f(\vec{x}) + \alpha \vec{\nabla} f(\vec{x})^T \vec{d} + o\left(\alpha \left| \vec{d} \right|\right)$$
$$= f(\vec{x}) + \alpha \vec{\nabla} f(\vec{x})^T \vec{d} + o(\alpha)$$
- If $\vec{\nabla} f(\vec{x})^T \vec{d} < 0$ → Descent
- In many of the algorithm we will study, the direction is set to be
$$\vec{d} = -B^{-1} \vec{\nabla} f$$
for some symmetric and nonsingular matrix B, and thus
$$\Delta f = -\alpha \vec{\nabla} f(\vec{x})^T B^{-1} \vec{\nabla} f(\vec{x}) + o(\alpha)$$
- In this case, descent is guaranteed for sufficiently small positive α if B is positive definite

- In selecting the stepsize α_k, we face a tradeoff
- We would like to choose α_k to give a substantial reduction of f, but at the same time, we do not want to spend too much time making the choice
- Minimization Rule: The ideal choice would be the global minimizer of the univariate function

$$\phi(\alpha) = f\left(\vec{x}_k + \alpha \vec{d}_k\right)$$

$$f\left(\vec{x}_k + \alpha_k \vec{d}_k\right) = \min_{\alpha \geq 0} \phi(\alpha)$$

- But in general, it is too expensive to identify this value
- To find even a local minimizer of ϕ to moderate precision generally requires too many evaluations of the objective function f and possibly the gradient $\vec{\nabla} f$
- More practical strategies perform an inexact line search to identify a stepsize that achieves adequate reductions in f at minimal effort

- Typical line search algorithms try out a sequence of candidate values for α_k, stopping to accept one of these values when certain conditions are satisfied
- The line search is done in two stages:
  1) A bracketing phase finds an interval $\mathcal{I}$ containing desirable stepsizes
  2) A 1-D algorithm to compute a good stepsize within this interval
- We have learned a number of such 1-D algorithms in this course
- We now discuss various termination conditions for the line search algorithm, under which the stepsize is considered "good enough"
- Limited Minimization Rule: $f\left(\vec{x}_k + \alpha_k \vec{d}_k\right) = \min_{\alpha \in \mathcal{I}} \phi(\alpha)$
- Note that the good stepsize may or may not satisfy the limited minimization rule

Non-gradient-based Methods

Requiring neither analytical nor numerical differentiation

Powell Method

- Cost function $f: \mathbb{R}^n \to \mathbb{R}$ which is not required to be differentiable because derivatives are not needed
- Given an initial set of n linearly independent vectors $\{\vec{u}_1, \vec{u}_2, \cdots, \vec{u}_n\}$
- Set starting point $\vec{x}_0$
- Successive line searches along $\{\vec{u}_1, \vec{u}_2, \cdots, \vec{u}_n\}$ in order, yielding

$$\left\{\vec{x}_0 + \alpha_1\vec{u}_1, \vec{x}_0 + \alpha_1\vec{u}_1 + \alpha_2\vec{u}_2, \cdots, \vec{x}_0 + \sum_{j=1}^{i}\alpha_j\vec{u}_j, \cdots, \vec{x}_0 + \sum_{j=1}^{n}\alpha_j\vec{u}_j\right\}$$

- Set

$$\vec{x}_1 = \vec{x}_0 + \sum_{j=1}^{n}\alpha_j\vec{u}_j$$

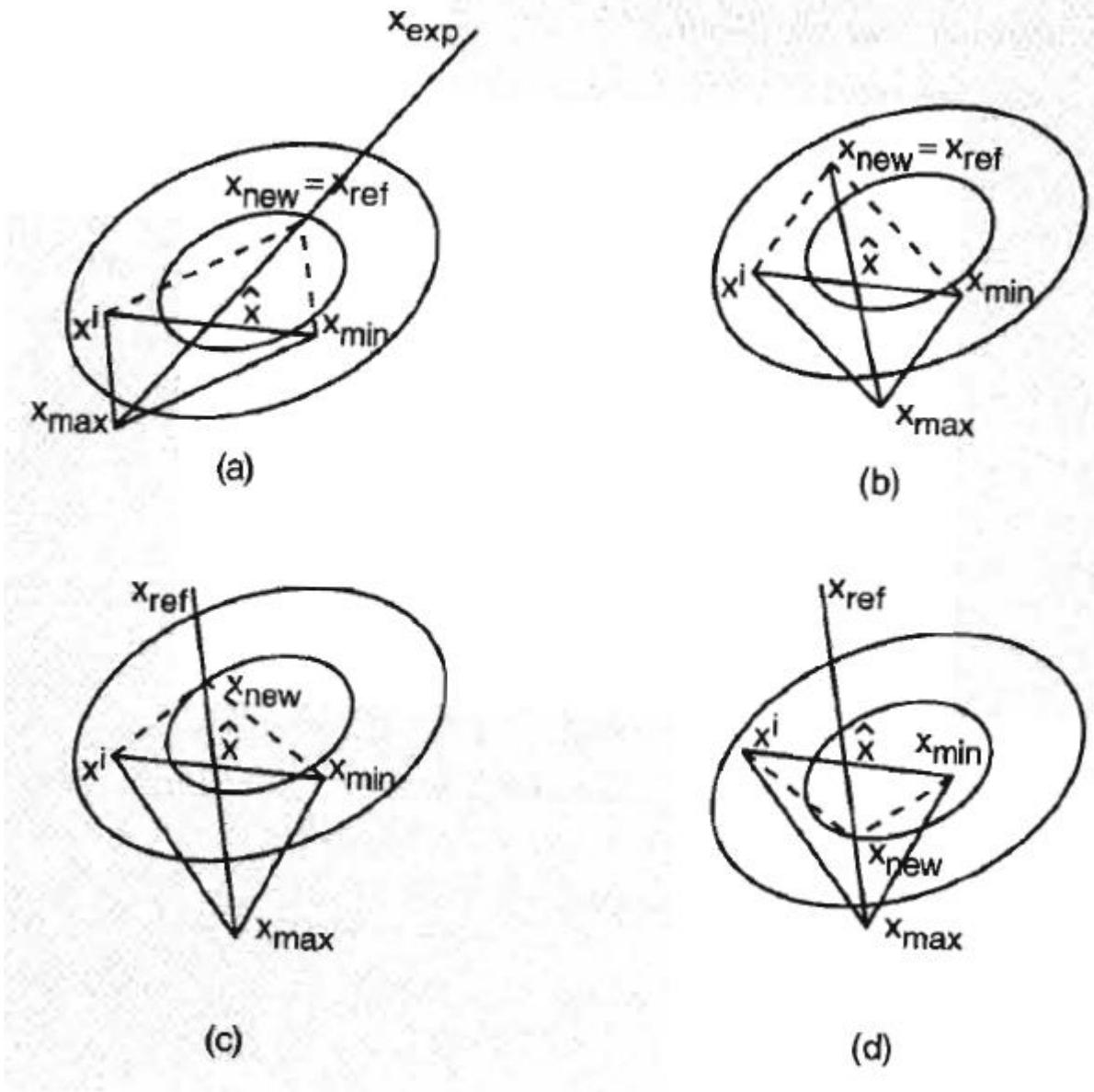
- The direction which yielded the greatest magnitude of displacement ($\max_j \left|\alpha_j\vec{u}_j\right|$) is then deleted from the list
- The displacement vector

$$\vec{x}_1 - \vec{x}_0 = \sum_{j=1}^{n}\alpha_j\vec{u}_j$$

  is added as the last search direction in the next iteration

Nelder–Mead Algorithm

- The Nelder–Mead algorithm is a direct search method and uses function information only (no gradient computation is required) to move from one iteration to the next
- In each step the objective function is computed at all vertices of an n-simplex, denoted by $\{\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_n, \vec{x}_{n+1}\}$ (not to be confused by the same notation for decision variables in different iteration steps in the previous methods)
- Based on the evaluated function values, search and construct a better simplex in the next step
- Three basic operations are used in the this construction: reflection, contraction, and expansion


x_{exp}
$x_{new} = x_{ref}$
x^i
$\hat{x}$
x_{min}
x_{max}
(a)
$x_{new} = x_{ref}$
x^i
$\hat{x}$
x_{min}
x_{max}
(b)
x_{ref}
x_{new}
x^i
$\hat{x}$
x_{min}
x_{max}
(c)
x_{ref}
x^i
$\hat{x}$
x_{min}
x_{new}
x_{max}
(d)

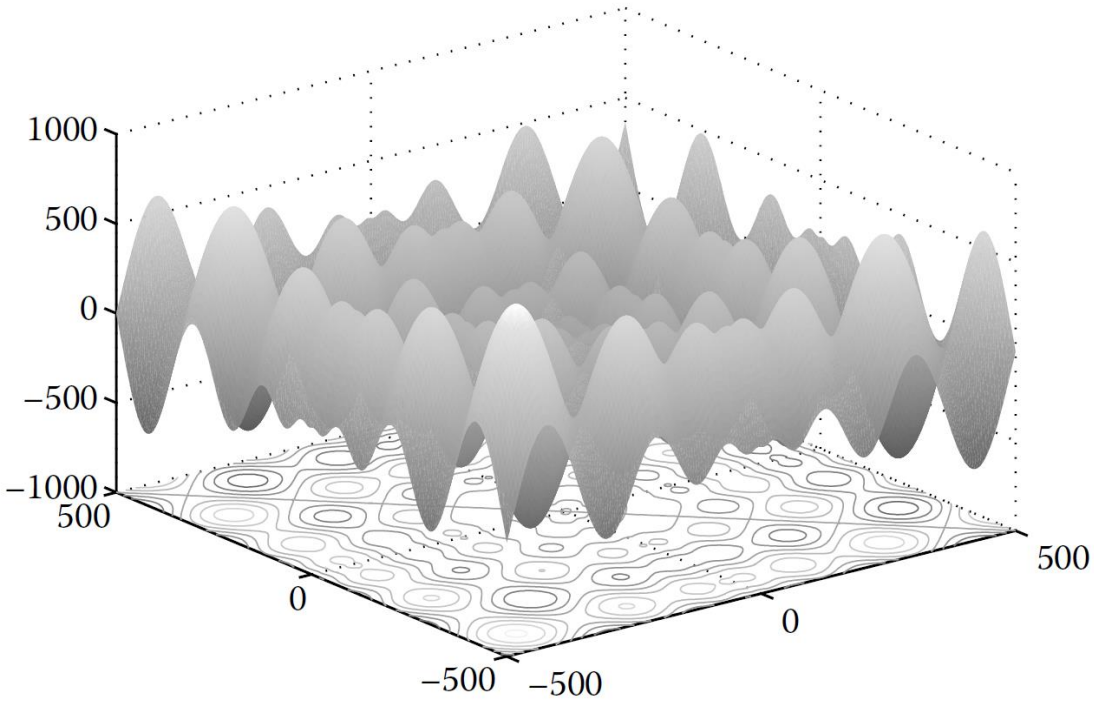
Random Search Methods

Introduction

- Most of the solution techniques for unconstrained optimization problems we have covered use the gradient information to locate the optimum
- Such methods require the objective function to be differentiable (and therefore continuous)
- We will explore five different random search methods of which four do not require the objective function to be either continuous or differentiable
- The solution techniques are
  - Stochastic Gradient Descent (SGD) (w/wo Momentum)
  - Genetic algorithm (GA)
  - Particle swarm optimization (PSO)
  - Tabu search
  - Simulated annealing (SA)

Introduction

- One major problem of the gradient methods is that they may converge to local but not global minimum
- This may happen depending on the initial guess and the “landscapes” of the objective function
- In the random search methods, worsening moves can be accepted
- This allows a large region of the search space to be explored to locate the global minimum



Stochastic Gradient Descent

- Applied when the objective function is the sum of a large number of terms
- For example, in deep learning, it is the average of the loss functions for each example in the training dataset. Given a training dataset of m examples, we assume that $f_i(\vec{x})$ is the loss function with respect to the training example of index i, where $\vec{x}$ is the parameter vector. Then we arrive at the objective function

$$f(\vec{x}) = \frac{1}{m}\sum_{i=1}^{m} f^{(i)}(\vec{x})$$

- The gradient of the objective function is

$$\vec{\nabla} f(\vec{x}) = \frac{1}{m}\sum_{i=1}^{m} \vec{\nabla} f^{(i)}(\vec{x})$$

- If gradient descent is used, the computational cost for each independent variable iteration is $O(m)$, which grows linearly with m.
- Therefore, when the training dataset is larger, the cost of gradient descent for each iteration will be higher

Stochastic Gradient Descent

- Stochastic gradient descent (SGD) reduces computational cost at each iteration. At each iteration of stochastic gradient descent, we uniformly sample an index $i \in \{1,2,3,\cdots,m\}$ for data examples at random, and compute the gradient $\vec{\nabla} f^{(i)}(\vec{x})$ to update $\vec{x}$:

$$\vec{x}_{k+1} = \vec{x}_k - \alpha_k \vec{\nabla} f^{(i)}(\vec{x})$$

- The computational cost for each iteration drops from $O(m)$ to $O(1)$
- Moreover, the stochastic gradient is an unbiased estimate of the full gradient because

$$\mathbb{E}_i\left[\vec{\nabla} f^{(i)}(\vec{x})\right] = \frac{1}{m}\sum_{i=1}^{m} f^{(i)}(\vec{x}) = \vec{\nabla} f(\vec{x})$$

- This means that, on average, the stochastic gradient is a good estimate of the gradient

Minibatch Stochastic Gradient Descent

- Gradient descent is not particularly data efficient whenever data is very similar
- Stochastic gradient descent is not particularly computationally efficient since CPUs and GPUs cannot exploit the full power of vectorization
- This suggests that it may be a good idea to use some hybrid in between

Minibatches

- We can increase the computational efficiency by applying SGD to a minibatch of observations at a time
- That is, we replace the gradient over a single observation by one over a small batch $\mathcal{B}_k$

$$\vec{x}_{k+1} = \vec{x}_k - \alpha_k \frac{1}{|\mathcal{B}_k|} \sum_{i \in \mathcal{B}_k} \vec{\nabla} f^{(i)}(\vec{x})$$

- Let's see what this does to the statistical properties of $\vec{\nabla} f(\vec{x})$
- Since $\mathcal{B}_k$ is drawn uniformly at random from the dataset, the expectation of the gradient remains unchanged
- The variance, on the other hand, is reduced significantly since the minibatch gradient is composed of independent gradients which are being averaged, its standard deviation is reduced by a factor of $1/\sqrt{|\mathcal{B}_k|}$
- This, by itself, is a good thing, since it means that the updates are more reliably aligned with the full gradient
- Naively this would indicate that choosing a large minibatch would be universally desirable
- However, it turns out that after some point, the additional reduction in standard deviation is minimal when compared to the linear increase in computational cost
- In practice we pick a minibatch that is large enough to offer good computational efficiency while still fitting into the memory of a GPU

Momentum

- In stochastic methods we need to be extra cautious when it comes to choosing the stepsizes in the face of noise
- If we decrease it too rapidly, convergence stalls
- If we are too lenient, we fail to converge to a good enough solution since random fluctuations keep on driving us away from optimality
- Here we will explore more effective optimization algorithms, especially for certain types of optimization problems that are common in practice

Leaky Averages

- First let us redefine the notation of minibatch stochastic gradient descent:

$$\vec{\nabla} f_{k,k-1}(\vec{x}) = \frac{1}{|\mathcal{B}_k|} \sum_{i \in \mathcal{B}_k} \vec{\nabla} f_{k-1}^{(i)}(\vec{x}) = \frac{1}{|\mathcal{B}_k|} \sum_{i \in \mathcal{B}_k} h_{i,k-1}(\vec{x})$$

- That is, $h_{i,k-1}(\vec{x})$ denotes the stochastic gradient descent for sample i using the weights updated at time $k-1$
- It would be nice if we could benefit from the effect of variance reduction even beyond averaging gradients on a minibatch
- One option to accomplish this task is to replace the gradient computation by a "leaky average" using "velocity" $\vec{v}$:

$$\vec{v}_k = \beta \vec{v}_{k-1} + \vec{\nabla} f_{k,k-1}(\vec{x})$$

for some $\beta \in (0,1)$ and with $\vec{v}_0 \equiv \vec{0}$

Leaky Averages

- This effectively replaces the instantaneous gradient by one that is been averaged over multiple past gradients
- The velocity $\vec{v}$ accumulates past gradients similar to how a heavy ball rolling down the objective function landscape integrates over past forces
- To see what is happening in more detail let's expand $\vec{v}_k$ recursively into

$$\begin{aligned}\vec{v}_k &= \beta\left(\beta\vec{v}_{k-2} + \vec{\nabla} f_{k-1,k-2}(\vec{x})\right) + \vec{\nabla} f_{k,k-1}(\vec{x}) \\ &= \beta^2\vec{v}_{k-2} + \beta\vec{\nabla} f_{k-1,k-2}(\vec{x}) + \vec{\nabla} f_{k,k-1}(\vec{x}) \\ &\cdots \\ &= \sum_{l=0}^{k-1} \beta^l \vec{\nabla} f_{k-l,k-l-1}(\vec{x})\end{aligned}$$

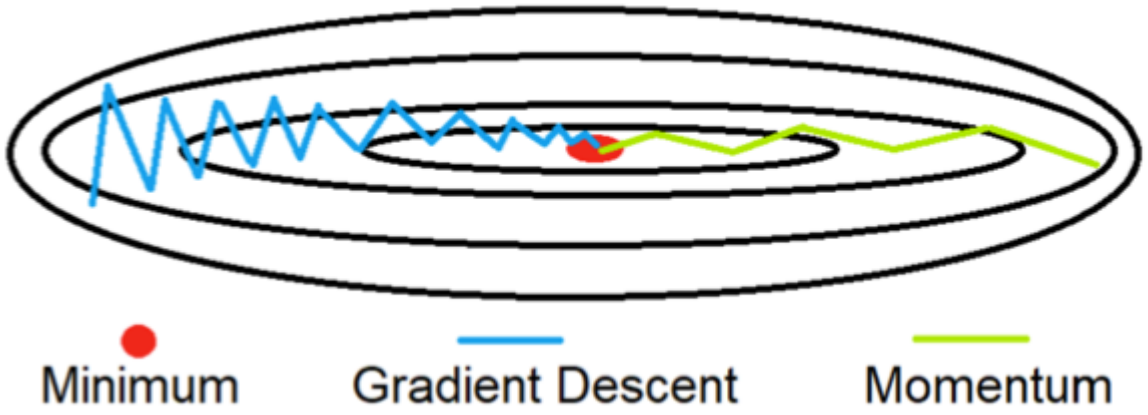
- Large β amounts to a long-range average, whereas small β amounts to only a slight correction relative to a gradient method
- The new gradient replacement no longer points into the direction of steepest descent on a particular instance but rather in the direction of a weighted average of past gradients
- This allows us to realize most of the benefits of averaging over a batch without the cost of actually computing the gradients on it

Leaky Averages

- The above reasoning formed the basis for what is now known as accelerated gradient methods, such as gradients with momentum
- They enjoy the additional benefit of being much more effective in cases where the optimization problem is ill-conditioned (i.e., where there are some directions where progress is much slower than in others, resembling a narrow canyon)
- Furthermore, they allow us to average over subsequent gradients to obtain more stable directions of descent

Momentum Method

- Looking at the optimization trace above we might intuit that averaging gradients over the past would work well
- After all, in the direction this will aggregate well-aligned gradients, thus increasing the distance we cover with every step
- Conversely, in the direction where gradients oscillate, an aggregate gradient will reduce step size due to oscillations that cancel each other out



Momentum Method

- Using $\vec{v}_k$ instead of the gradient $\vec{\nabla} f_k(\vec{x})$ yields the following update equations:

$$\begin{cases} \vec{v}_k = \beta \vec{v}_{k-1} + \vec{\nabla} f_{k,k-1}(\vec{x}) \\ \vec{x}_k = \vec{x}_{k-1} - \alpha_{k-1} \vec{v}_k \end{cases}$$

- Note that for $\beta = 0$ we recover regular gradient descent
- As mentioned, momentum method can reduce sluggishness of the gradient decent method
- Assuming this is true, for constant stepsize α in gradient descent, we have

$$\alpha \vec{v}_k \approx \alpha \sum_{l=0}^{k-1} \beta^l \vec{\nabla} f_{k-1}(\vec{x}) \approx \frac{\alpha}{1-\beta} \vec{\nabla} f_{k-1}(\vec{x})$$

- Comparing with gradient descent, the stepsize is increased by a factor of $\frac{1}{1-\beta}$
- This is yet another advantage of the method when large stepsize is desired.

Genetic Algorithm (GA)

- Genetic algorithm (GA) is a search algorithm based on the mechanism of natural selection
- It relies on one of the most important principles of Darwin: survival of the fittest
- Globally the population is submitted to selection and various genetic transformations
- After some generations, when the population is enduring no more, the best individual in the population is assumed to represent the optimal solution
- GA mimics the genetic process in which hereditary characteristics are transmitted from a parent to an offspring

Genetic Algorithm (GA)

- The basic unit of inheritance are genes in chromosomes
- For example, humans have 23 pairs of chromosomes
- One chromosome in each pair is derived from the maternal and one from the paternal parent
- As a result of the crossover operation of chromosomes, some characteristics of each parent can be seen in the offspring
- In the natural hereditary process, some genes also randomly mutate
- For instance, if the gene corresponding to eye color mutates, the offspring can have blue eyes even if both of the parents' eyes are brown
- The mutation in a sense brings variety into the offspring and improves his survivability in a changing environment

Genetic Algorithm (GA)

- In gradient-based methods, the solution moves from one point to another using the gradient and the Hessian information
- In GA, one works with a population of points in the search space rather than a single point
- The fitness of each individual (point) is defined by the value of the objective function at this point:
  - Minimization problems: Smaller function value → Higher fitness
  - Maximization problems: Larger function value → Higher fitness
- Individuals who have high fitness value undergo crossover and mutation with the hope that they produce even better offspring (even higher fitness value) as compared to their parents
- The population will be replaced by the offspring in the next generation (next iteration step)

Particle Swarm Optimization

- In the particle swarm optimization (PSO) technique, similar to GA, a number of search points are simultaneously explored in the iteration
- The PSO technique is inspired by the collective wisdom of a group of individuals such as a flock of birds, animals moving in herds, or schools of fish moving together
- The PSO algorithm keeps track of the best position of the individual as well as that of the population in terms of the objective function
- The best objective function value of the individual and that of the group is denoted by $p\text{best}$ and $g\text{best}$ respectively
- Each individual in the group moves with a velocity that is a function of $p\text{best}$, $g\text{best}$ and its initial velocity
- The new position of the individual is updated based on its initial position and the velocity
- The objective function value is again computed for the new positions and the PSO steps are repeated

Tabu Search

- The tabu search is another meta-heuristic technique employing local search methods used for mathematical optimization
- Created by Fred W. Glover in 1986 and formalized in 1989
- It can be employed to tackle complex combinatorial problems such as job scheduling and traveling salesman problems
- Local search methods may become stuck in suboptimal regions or on plateaus where many solutions are equally fit
- Tabu search enhances the performance of local search by relaxing its basic rule
- First, at each step worsening moves can be accepted if no improving move is available (like when the search is stuck at a strict local minimum)
- In addition, prohibitions (henceforth the term tabu) are introduced to discourage the search from coming back to previously-visited solutions

Tabu Search

- The implementation of tabu search uses memory structures that describe the visited solutions or user-provided sets of rules
- For example, a tabu-list of solutions visited in the previous n iterations is stored and updated
- Potential solutions that are on the list are said to be tabu-active and are banned
- They can be revisited when they reach the expiration points and are removed from the list
- Aspiration criteria can be employed that override a solution's tabu state, thereby including the otherwise-excluded solution in the allowed set (provided the solution is “good enough” according to a measure of quality or diversity)
- A simple and commonly used aspiration criterion is to allow solutions which are better than the currently-known best solution

Simulated Annealing (SA)

- It has been well known for centuries, among glassworkers, metalworkers, ceramicists, and others, that when you are working with hot materials, like molten glass or metal, one must cool the materials slowly to produce a hardy, solid final product
- Glass cooled too quickly will develop defects and weaknesses, or even shatter
- Metal cooled too quickly will be soft or brittle
- If one cools slowly (anneals the material), on the other hand, the end result will be rigid and sturdy
- The reason is that when materials are hot their atoms or molecules are in rapid motions and are disordered
- The motions randomizes their positions and prevent them coming to rest in an orderly arrangement
- If one cools such a material rapidly, that disorder gets frozen into the material, and this creates flaws in the structure, cracks and defects that weaken the material
- In the language of thermal physics, the system has become trapped at a local minimum of the energy
- If, on the other hand, one cools the material slowly then the disorder no longer gets frozen in and the atoms have time to rearrange and reach an ordered state, such as the regular lattice seen in the crystal structure of metals, which is the ground state of the system and contains no flaws, and is correspondingly stronger
- Thus while a rapidly cooled system can get stuck in a local energy minimum, an annealed system can find its way to the ground state

Simulated Annealing (SA)

- Simulated annealing (SA) is an optimization technique that has derived its name from the physical process of annealing of solids
- Similar to Tabu Search, SA methodology allows “hill climbing” when the temperature is high. That is, those points that are in the near vicinity of the search point, but have a higher objective function value can still be selected with certain probability
- This allows the algorithm to escape from local optima. Thus, SA is a powerful technique in locating the global optimum solution

Constrained Optimization

Mathematical Formulation

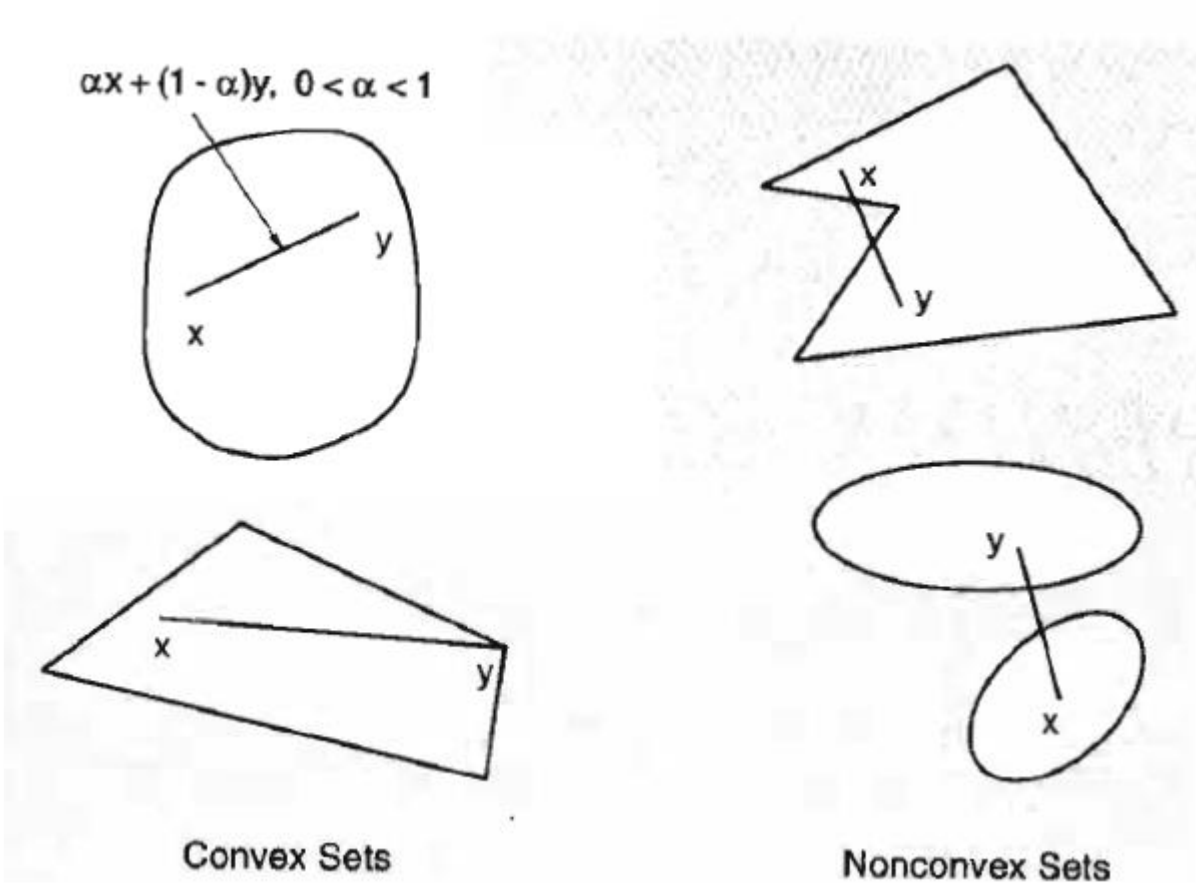
$$\min_{\vec{x}\in\mathbb{R}^n} f\left(\vec{x}\right) \quad \text{subject to} \begin{cases} c_i\left(\vec{x}\right)=0 & i\in\mathcal{E} \\ c_i\left(\vec{x}\right)\geq 0 & i\in\mathcal{I} \end{cases}$$

$$\text{Feasible Set: } X=\left\{\vec{x}: c_i\left(\vec{x}\right)=0 \ \ \forall i\in\mathcal{E},\, c_i\left(\vec{x}\right)\geq 0 \ \ \forall i\in\mathcal{I}\right\}$$

Convex Sets

- Def: Let C be a subset of $\mathbb{R}^n$. We say that C is <u>convex</u> if

$$\alpha\vec{x} + (1-\alpha)\vec{y} \in C, \qquad \forall \vec{x}, \vec{y} \in C, \forall \alpha \in [0,1]$$



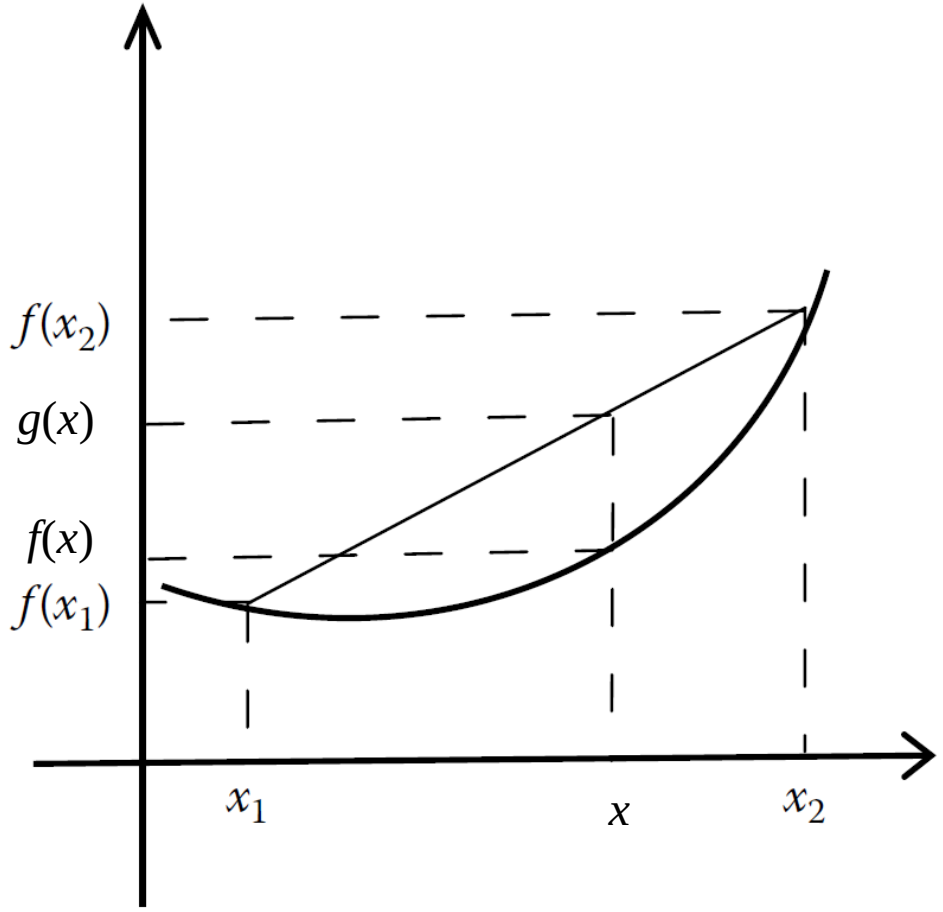
Convex Functions

- A single-variable function $f(x)$ is called convex if for any $x_1, x_2,$ and $x \ni [x_1, x_2]$

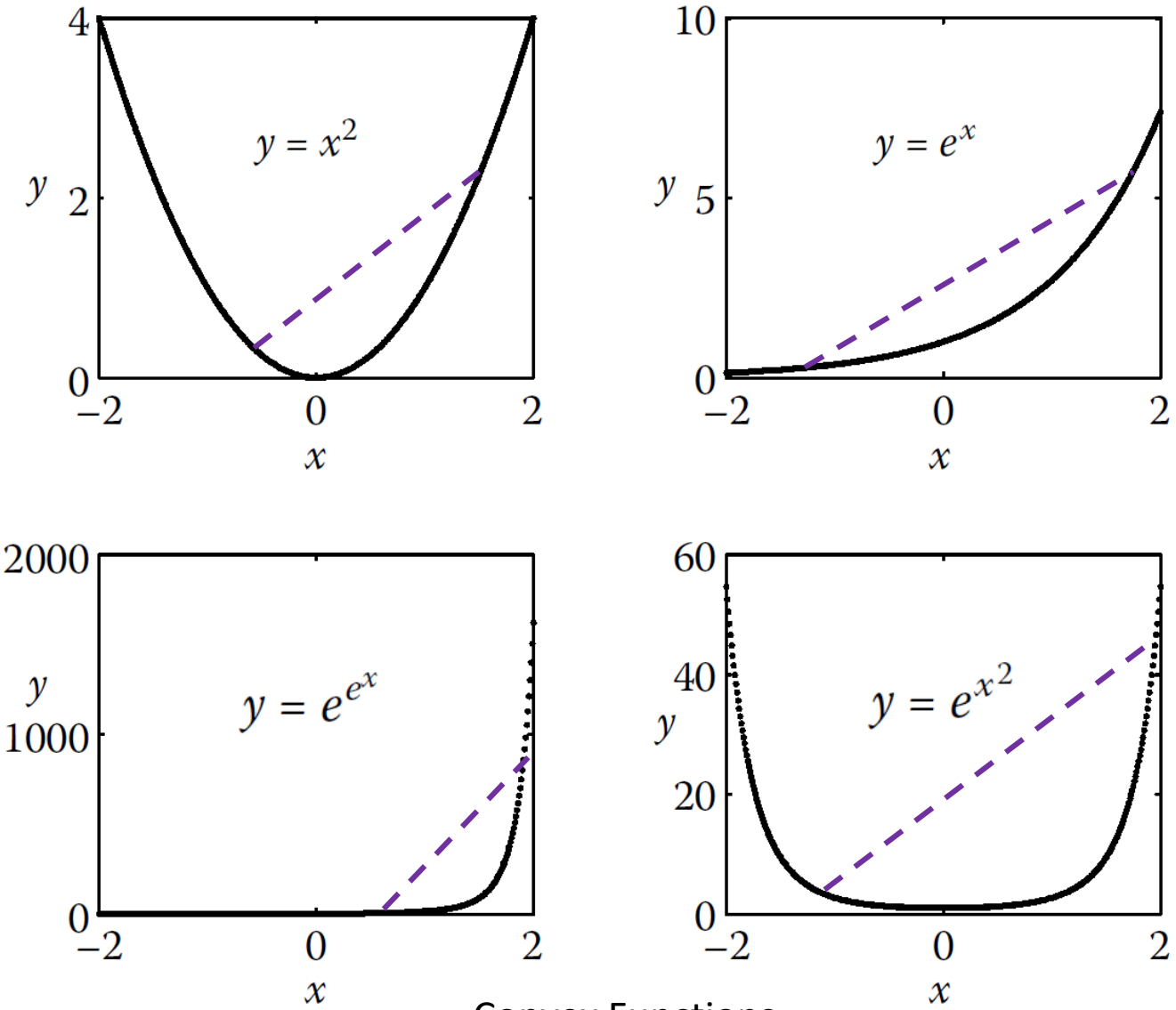
$$f(x) \le g(x) = \frac{f(x_2) - f(x_1)}{x_2 - x_1} x + \frac{x_2 f(x_1) - x_1 f(x_2)}{x_2 - x_1}$$

- A single-variable function $f(x)$ is called strictly convex if for any $x_1, x_2,$ and $x \ni (x_1, x_2)$

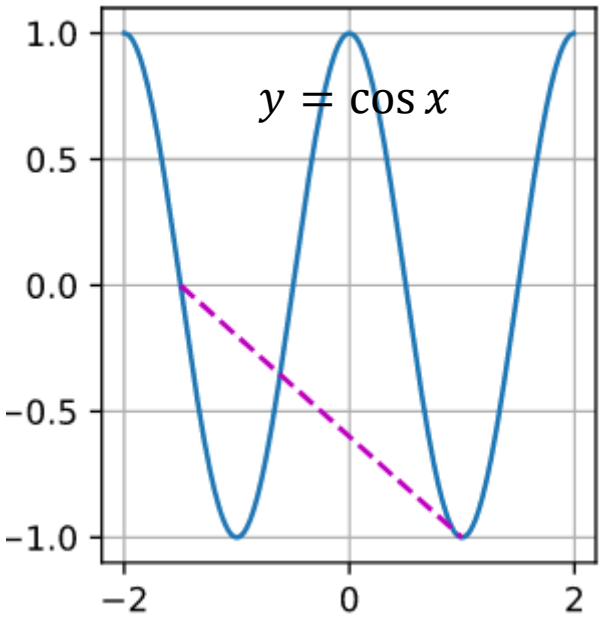
$$f(x) < g(x) = \frac{f(x_2) - f(x_1)}{x_2 - x_1} x + \frac{x_2 f(x_1) - x_1 f(x_2)}{x_2 - x_1}$$



Examples



Convex Functions



Non-convex Function

Convex Functions

- Def: Let C be a convex subset of $\mathbb{R}^n$. A function $f: C \to \mathbb{R}$ is called convex if

$$f(\alpha\vec{x} + (1-\alpha)\vec{y}) \leq \alpha f(\vec{x}) + (1-\alpha) f(\vec{y}), \qquad \forall \vec{x}, \vec{y} \in C, \forall \alpha \in [0,1]$$

- The function is called concave if $-f$ is convex
- The function f is called strictly convex if

$$f(\alpha\vec{x} + (1-\alpha)\vec{y}) < \alpha f(\vec{x}) + (1-\alpha) f(\vec{y}), \qquad \forall \vec{x}, \vec{y} \in C, \forall \alpha \in (0,1)$$

Jensen's Inequality

- The definition of convex functions can be generalized to the Jensen's Inequality:

$$f\left(\sum_i \alpha_i \vec{x}_i\right) \leq \sum_i \alpha_i f(\vec{x}_i)\,, \qquad \forall \text{ collinear } \vec{x}_i, \forall \alpha_i \geq 0 \text{ s. t.} \sum_i \alpha_i = 1$$

- The above, together with its generalization from α_i to continuous probability distribution $\alpha(x)$, lead to the following probabilistic form of the inequality in terms of expectations $E_x[\quad]$:

$$f(E_x[x]) \leq E_x[f(x)], \qquad \forall \text{ random variable } x$$

Characterizations of Differentiable Convex Functions

- Let $C \subset \mathbb{R}^n$ be a convex set and let $f: \mathbb{R}^n \to \mathbb{R}$ be differentiable over C, then

a) The function is convex over C if and only if

$$f(\vec{z}) \geq f(\vec{x}) + \vec{\nabla} f(\vec{x})^T (\vec{z} - \vec{x}), \qquad \forall \vec{x}, \vec{z} \in C$$

b) The function is strictly convex over C if and only if

$$f(\vec{z}) > f(\vec{x}) + \vec{\nabla} f(\vec{x})^T (\vec{z} - \vec{x}), \qquad \forall \vec{x}, \vec{z} \in C, \vec{x} \neq \vec{z}$$

Characterizations of Twice Differentiable Convex Functions

- Let $C \subset \mathbb{R}^n$ be a convex set, let $f: \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable over C, and let Q be a real symmetric $n \times n$ matrix

a) If $\nabla^2 f(\vec{x})$ is positive semidefinite for all $\vec{x} \in C$, then f is convex over C

b) If $\nabla^2 f(\vec{x})$ is positive definite for all $\vec{x} \in C$, then f is strictly convex over C

c) If $C = \mathbb{R}^n$ and f is convex, then $\nabla^2 f(\vec{x})$ is positive semidefinite for all $\vec{x} \in C$

d) The quadratic function $f(\vec{x}) = \vec{x}^T Q \vec{x}$ is convex if and only if Q is positive semidefinite. Furthermore, f is strictly convex if and only if Q is positive definite

Convex Optimization

- A special case of constrained optimization problem in which
  1) the objective function is convex;
  2) the equality constraint functions are linear;
  3) the inequality constraint functions are convex
- By (2) and (3), the feasible set is convex

Theorem

- If f is a convex function, then a local minimum of f over C is a global minimum.
- If in addition f is strictly convex over C, then there exists at most one global minimum of f over C

Theorem

- If $\vec{x}^*$ is a local minimum of f over C, then

$$\vec{\nabla} f(\vec{x}^*)^T(\vec{x} - \vec{x}^*) \geq 0, \qquad \forall \vec{x} \in C$$

- If f is convex over C, then the above condition is also sufficient for $\vec{x}^*$ to minimize f over C

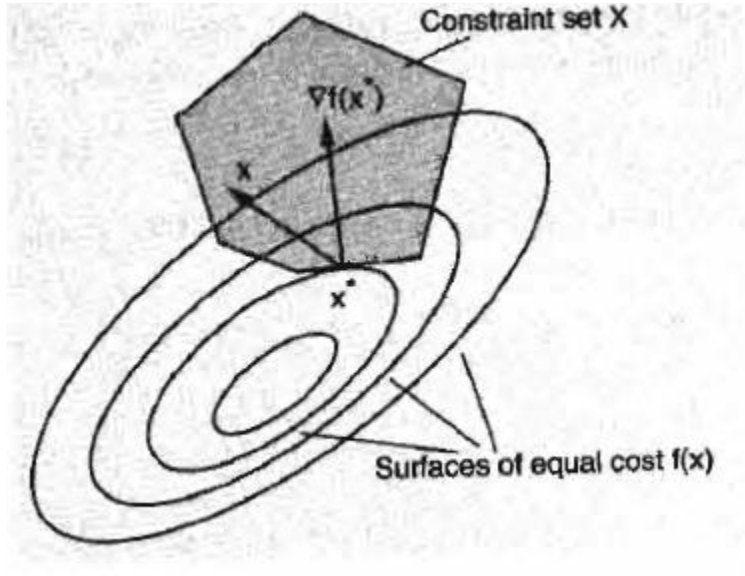


Figure 2.1.2. Geometric interpretation of the optimality condition of Prop. 2.1.2. At a local minimum x^*, the gradient $\nabla f(x^*)$ makes an angle less than or equal to 90 degrees with all feasible variations $x - x^*$, $x \in X$.

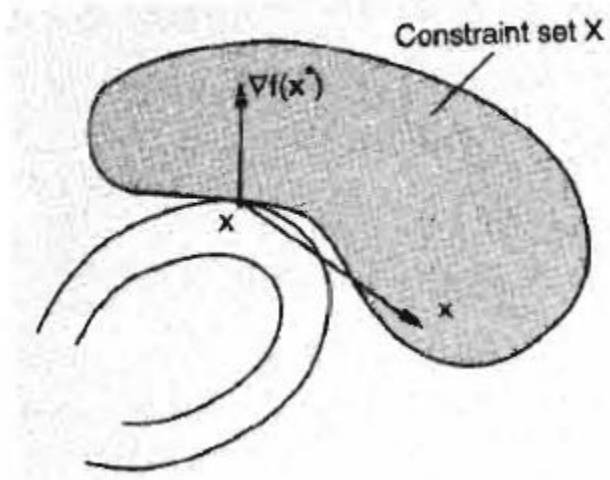


Figure 2.1.3. Illustration of how the necessary optimality condition may fail when X is not convex. Here x^* is a local minimum but we have

$$\nabla f(x^*)'(x - x^*) < 0$$

for the feasible vector x shown.

Lagrange Multiplier Theory

Lagrange Multiplier

- The constraint set of an optimization problem is usually specified in terms of equality and inequality constraints
- If we take into account this structure, we obtain a sophisticated collection of optimality conditions, involving some auxiliary variables called Lagrange multipliers
- These variables facilitate the characterization of optimal solutions, but also provide valuable sensitivity information, quantifying up to first order the variation of the optimal cost caused by variations in problem data

Equality Constraints

- Consider problems with equality constraints of the form

$$\text{minimize } f(\vec{x})$$
$$\text{subject to } h_i(\vec{x}) = 0, \qquad i = 1, 2, \cdots, m$$

- We assume that $f: \mathbb{R}^n \to \mathbb{R}$, $h_i: \mathbb{R}^n \to \mathbb{R}$, $i = 1, 2, \cdots, m$, are continuously differentiable functions over $\mathbb{R}^n$ (an open set containing the local minimum also works)
- For notational convenience, we introduce the vector constraint function $\vec{h}: \mathbb{R}^n \to \mathbb{R}^m$, where

$$\vec{h} = (h_1, \cdots, h_m)^T$$

- We can then write the constraints in the more compact form

$$\vec{h}(\vec{x}) = \vec{0}$$

- A feasible vector $\vec{x}$ at which the constraint gradients $\vec{\nabla} h_1(\vec{x})$, …, $\vec{\nabla} h_m(\vec{x})$ are linearly independent is called regular
- The set of first order feasible variations of a feasible point $\vec{x}$ (i.e., $\vec{h}(\vec{x}) = \vec{0}$) is the subspace of variations $\Delta\vec{x}$ for which the vector $\vec{x} + \Delta\vec{x}$ satisfies the constraints up to first order, viz.,

$$V(\vec{x}) = \left\{ \Delta\vec{x} \middle| \vec{\nabla}\vec{h}(\vec{x})^T \Delta\vec{x} = \vec{0} \right\}$$

Lagrange Multiplier Theorem --- Necessary Conditions

- Let $\vec{x}^*$ be a local minimum of f subject to $\vec{h}(\vec{x}) = \vec{0}$, and assume that $\vec{x}^*$ is regular. Then there exists a unique vector $\vec{\lambda}^* = (\lambda_1^*, \cdots, \lambda_m^*)^T$, called a Lagrange multiplier vector, such that

$$\vec{\nabla} f(\vec{x}^*) - \sum_{i=1}^{m} \lambda_i^* \vec{\nabla} h_i(\vec{x}^*) = \vec{0}$$

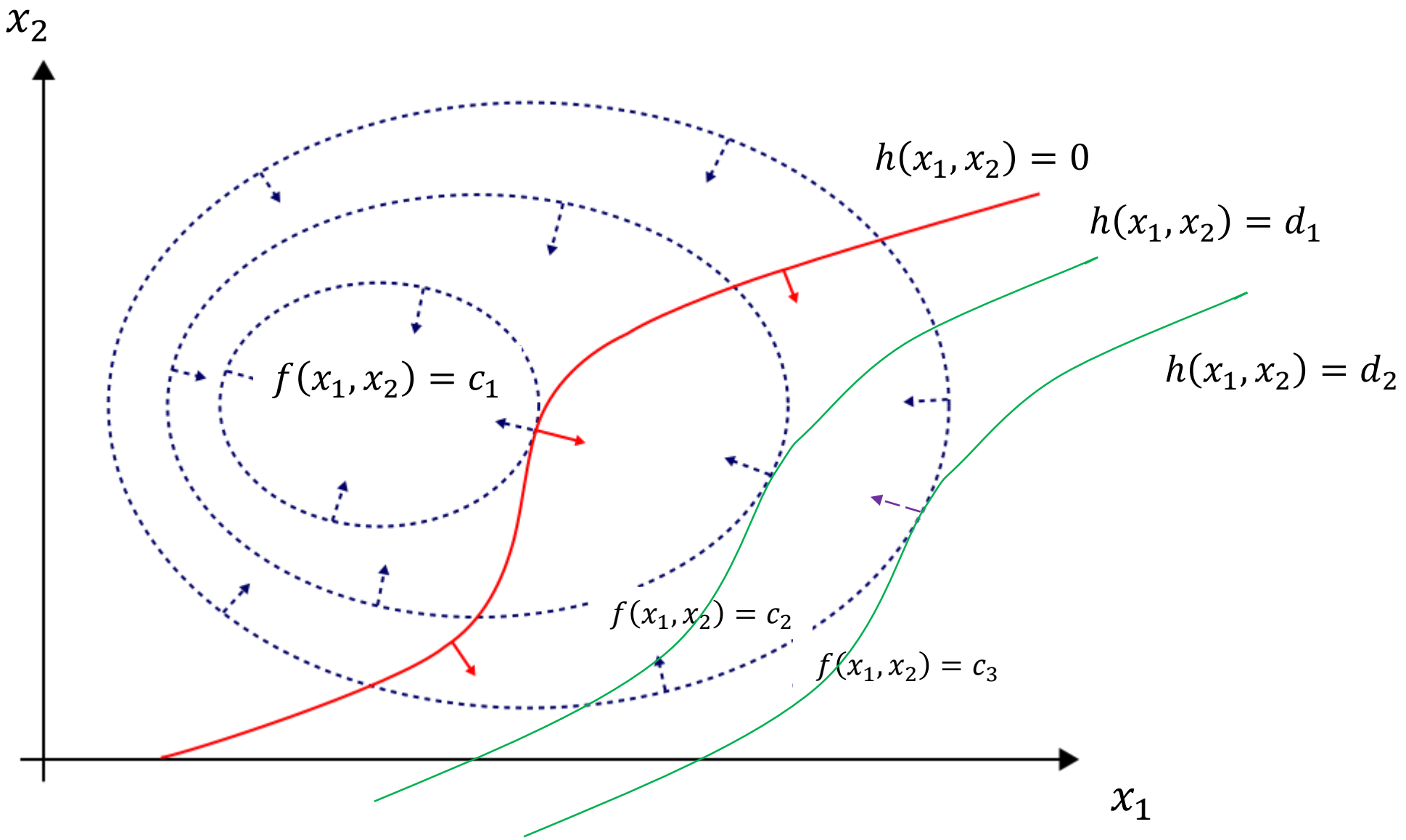
- If in addition f and $\vec{h}$ are twice continuously differentiable, we have

$$\vec{y}\left(\nabla^2 f(\vec{x}^*) - \sum_{i=1}^{m} \lambda_i^* \nabla^2 h_i(\vec{x}^*) \right)\vec{y} \geq 0, \qquad \forall \vec{y} \in V(\vec{x}^*)$$

Interpretation

- By definition, the subspace of first order feasible variations at $\vec{x}^*$ is the space to vectors perpendicular to all $\vec{\nabla} h_i(\vec{x}^*)$
- Denote the true set of feasible variations by F, the span of the constraints' gradients by G, and the orthogonal complement of G by $G^\perp$
- Assume that the set of first order feasible variations is the true set of feasible variations
- Then $F = V = G^\perp$
- This implies $\vec{\nabla} f(\vec{x}^*) \in F^\perp = G$
- Thus there are scalars $\lambda_1^*, \cdots, \lambda_m^*$ such that

$$\vec{\nabla} f(\vec{x}^*) = \sum_{i=1}^{m} \lambda_i^* \vec{\nabla} h_i(\vec{x}^*) \Leftrightarrow \vec{\nabla} f(\vec{x}^*) - \sum_{i=1}^{m} \lambda_i^* \, \vec{\nabla} h_i(\vec{x}^*) = \vec{0}$$


x_2
$h(x_1, x_2) = 0$
$h(x_1, x_2) = d_1$
$h(x_1, x_2) = d_2$
$f(x_1, x_2) = c_1$
$f(x_1, x_2) = c_2$
$f(x_1, x_2) = c_3$
x_1

Example: A Problem with no Lagrange Multipliers

- Consider the problem of minimizing

$$f(x_1, x_2, x_3) = x_1 + x_2 + x_3^2$$

  subject to the two constraints

$$h_1(x_1, x_2, x_3) = (x_1 - 1)^2 + x_2^2 - 1 = 0$$
$$h_2(x_1, x_2, x_3) = (x_1 - 2)^2 + x_2^2 - 4 = 0$$

- The feasible set can be obtained by

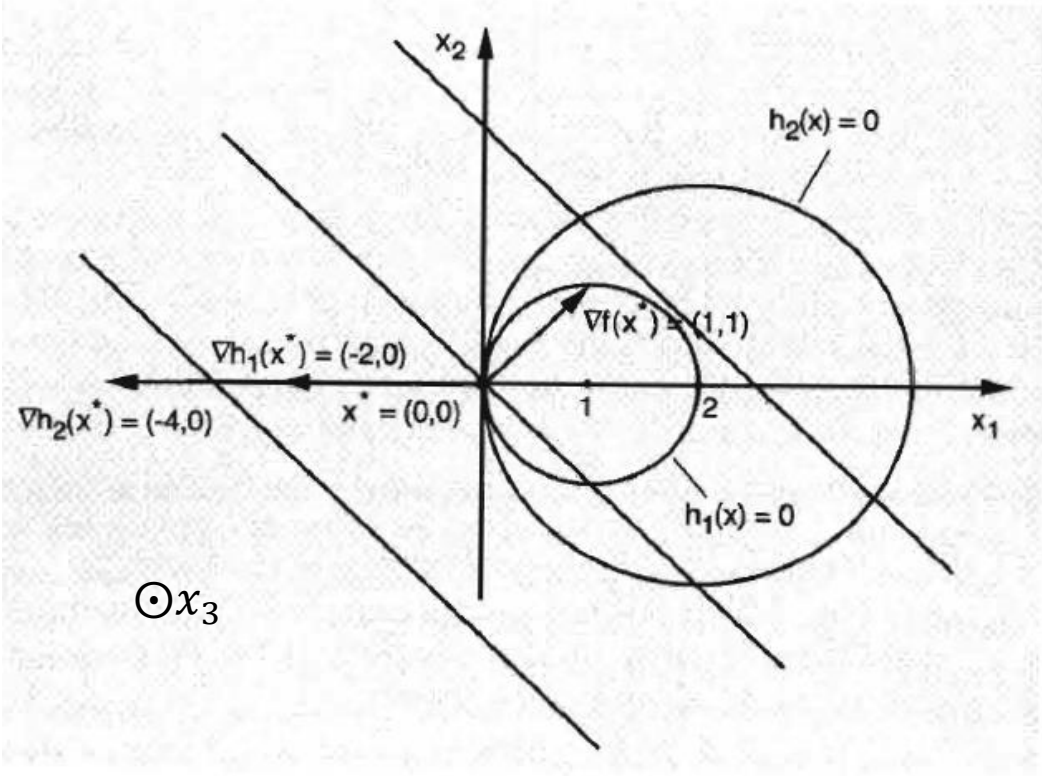
$$h_1(x_1, x_2, x_3) - h_2(x_1, x_2, x_3) = (x_1 - 1)^2 - (x_1 - 2)^2 + 3 = 0$$
$$\Rightarrow 2x_1 - 3 + 3 = 0 \Rightarrow x_1 = 0$$

- Substituting into h_1 or h_2, we obtain $x_2 = 0$
- Hence the feasible set is the x_3-axis
- Therefore,

$$f(x_3) = x_3^2 \Rightarrow \vec{x}^* = (0, 0, 0)^T$$

Example: A Problem with no Lagrange Multipliers

- The contour surfaces of h_1 and h_2 are cylinders
- Consider the point $\vec{x}^* = (0,0,0)^T$
- Here $\vec{\nabla} h_1(\vec{x}^*) = (-2,0,0)^T$ and $\vec{\nabla} h_2(\vec{x}^*) = (-4,0,0)^T$, which are not linear independent
- $\vec{x}^* = (0,0,0)^T$ is not a regular feasible point
- In fact, $\vec{\nabla} f(\vec{x}^*) = (1,1,0)^T$ cannot be expressed as a linear combination of $\vec{\nabla} h_1(\vec{x}^*)$ and $\vec{\nabla} h_2(\vec{x}^*)$
- The problem here is that the subspace $V(\vec{x}^*)$ is the x_2x_3-plane, which has larger dimension than the true set of feasible variations, which is just the x_3-axis



The Lagrangian Function

- We have seen that assuming regularity, solving $\vec{\nabla} f(\vec{x}) = \vec{0}$ restricted to the feasible set X is equivalent to solving

$$\begin{cases} \vec{\nabla} f(\vec{x}) - \sum_{i=1}^{m} \lambda_i \vec{\nabla} h_i(\vec{x}) = \vec{0} \\ \vec{h}(\vec{x}) = \vec{0} \end{cases}$$

- If we define the Lagrangian function L: $\mathbb{R}^{n+m} \to \mathbb{R}$ by

$$L\left(\vec{x}, \vec{\lambda}\right) = f(\vec{x}) - \sum_{i=1}^{m} \lambda_i h_i(\vec{x})$$

- Then, if $\vec{x}^*$ is a local minimum which is regular, the same conditions can be written compactly as

$$\begin{cases} \vec{\nabla}_{\vec{x}} L\left(\vec{x}^*, \vec{\lambda}^*\right) = \vec{0} \\ \vec{\nabla}_{\vec{\lambda}} L\left(\vec{x}^*, \vec{\lambda}^*\right) = \vec{0} \end{cases}$$

- The condition on second derivatives is

$$\vec{y} \nabla_{\vec{x}}^2 L\left(\vec{x}^*, \vec{\lambda}^*\right) \vec{y} \geq 0, \qquad \forall \vec{y} \in V(\vec{x}^*)$$

The Lagrangian Function

- The first order necessary conditions represent a system of $n + m$ equations with $n + m$ unknowns --- the coordinates of $\vec{x}^*$ and $\vec{\lambda}^*$
- Every local minimum $\vec{x}^*$ which is regular, together with its associated Lagrange multiplier vector, will be a solution of this system
- The problem is turned into a problem of finding stationary points of $L\left(\vec{x}, \vec{\lambda}\right)$ under no constraints!

Sufficiency Conditions

- Assume that f and $\vec{h}$ are twice continuously differentiable, and let $\vec{x}^* \in \mathbb{R}^n$ and $\vec{\lambda}^* \in \mathbb{R}^m$ satisfy

$$\begin{cases} \vec{\nabla}_{\vec{x}} L\left(\vec{x}^*, \vec{\lambda}^*\right) = \vec{0} \\ \vec{\nabla}_{\vec{\lambda}} L\left(\vec{x}^*, \vec{\lambda}^*\right) = \vec{0} \\ \vec{y} \nabla_{\vec{x}}^2 L\left(\vec{x}^*, \vec{\lambda}^*\right) \vec{y} > 0, \qquad \forall \vec{y} \neq \vec{0} \text{ with } \vec{\nabla} \vec{h} \vec{y} = \vec{0} \end{cases}$$

- Then $\vec{x}^*$ is a strict local minimum of f subject to $\vec{h}(\vec{x}) = \vec{0}$
- Note that the sufficient conditions do not include regularity of the vector $\vec{x}^*$

Sensitivity Theorem

- Let $\vec{x}^*$ and $\vec{\lambda}^*$ be a local minimum and Lagrange multiplier, respectively, satisfying the preceding sufficiency conditions
- Consider the family of problems parameterized by the vector $\vec{u} \in \mathbb{R}^m$

$$\text{minimize } f(\vec{x})$$
$$\text{subject to } \vec{h}(\vec{x}) = \vec{u}$$

- Then there exists an open sphere S centered at $\vec{u} = \vec{0}$ such that for every $\vec{u} \in S$, there is an $\vec{x}(\vec{u}) \in \mathbb{R}^n$ and a $\vec{\lambda}(\vec{u}) \in \mathbb{R}^m$, which are a local minimum-Lagrange multiplier pair of the above problem
- Furthermore, $\vec{x}(\cdot)$ and $\vec{\lambda}(\cdot)$ are continuously differentiable functions within S and we have $\vec{x}(\vec{0}) = \vec{x}^*$ and $\vec{\lambda}(\vec{0}) = \vec{\lambda}^*$
- In addition, for all $\vec{u} \in S$ we have

$$\vec{\nabla} p(\vec{u}) = \vec{\lambda}(\vec{u})$$

where $p(\vec{u}) = f\big(\vec{x}(\vec{u})\big)$ is the optimal cost parametrized by $\vec{u}$

Inequality Constraints

- Consider a problem involving both equality and inequality constraints

$$\text{minimize } f(\vec{x})$$
$$\text{subject to } h_1(\vec{x}) = 0, \cdots, h_m(\vec{x}) = 0$$
$$g_1(\vec{x}) \geq 0, \cdots, g_r(\vec{x}) \geq 0$$

  where $f, h_i, g_j \colon \mathbb{R}^n \to \mathbb{R}$ are continuously differentiable functions

- More succinctly, we can write this problem as

$$\text{minimize } f(\vec{x})$$
$$\text{subject to } \vec{h}(\vec{x}) = \vec{0}, \vec{g}(\vec{x}) \succcurlyeq \vec{0}$$

  where $\vec{h} = (h_1, \cdots, h_m) \colon \mathbb{R}^n \to \mathbb{R}^m$ and $\vec{g} = (g_1, \cdots, g_r) \colon \mathbb{R}^n \to \mathbb{R}^r$,

  and $\vec{g}(\vec{x}) \succcurlyeq \vec{0}$ is defined to mean $g_j(\vec{x}) \geq 0$ for all $j = 1, \cdots, r$

Inequality Constraints

- For any feasible point $\vec{x}$, the set of active inequality constraints is denoted by

$$A(\vec{x}) = \{j | g_j(\vec{x}) = 0\}$$

- If $j \notin A(\vec{x})$, we say that the jth constraint is inactive at $\vec{x}$
- We note that if $\vec{x}^*$ is a local minimum of the an inequality constrained problem (ICP), then $\vec{x}^*$ is also a local minimum for the ICP identical to the original ICP except that the inactive constraints at $\vec{x}^*$ have been discarded
- Thus, in effect, inactive constraints at $\vec{x}^*$ don't matter; they can be ignored in the statement of optimality conditions

Inequality Constraints

- On the other hand, at a local minimum, active inequality constraints can be treated to a large extent as equalities
- In particular, if $\vec{x}^*$ is a local minimum of the ICP, then $\vec{x}^*$ is also a local minimum for the equality constrained problem

$$\begin{gathered}\text{minimize } f(\vec{x}) \\ \text{subject to } h_1(\vec{x}) = 0, \cdots, h_m(\vec{x}) = 0 \\ g_j(\vec{x}) = 0, \qquad \forall j \in A(\vec{x}^*)\end{gathered}$$

- Thus, if $\vec{x}^*$ is regular for the latter problem, there exists Lagrange multipliers $\lambda_1^*, \cdots, \lambda_m^*$, and μ_j^*, $j \in A(\vec{x}^*)$ such that

$$\vec{\nabla} f(\vec{x}^*) - \sum_{i=1}^{m} \lambda_i^* \vec{\nabla} h_i(\vec{x}^*) - \sum_{j \in A(\vec{x}^*)} \mu_j^* \vec{\nabla} g_j(\vec{x}^*) = \vec{0}$$

- Assigning zero Lagrange multipliers to the inactive constraints, we obtain

$$\vec{\nabla} f(\vec{x}^*) - \sum_{i=1}^{m} \lambda_i^* \vec{\nabla} h_i(\vec{x}^*) - \sum_{j=1}^{r} \mu_j^* \vec{\nabla} g_j(\vec{x}^*) = \vec{0}$$

which can be viewed as an analog of the first order optimality condition for the equality constrained problem

Inequality Constraints

- There is one more important fact about the Lagrange multipliers μ_j^*: they are nonnegative
- For by sensitivity theorem $\vec{\nabla} p(\vec{u}) = \vec{\lambda}(\vec{u}) \Rightarrow \frac{\partial f\left(\vec{x}(\vec{u})\right)}{\partial u_j} = \lambda_j$
- If one relaxes the jth constraint by $\Delta u_j < 0$, the optimal cost must not be worsened
- Therefore

$$\frac{\partial f\left(\vec{x}(\vec{u})\right)}{\partial u_j} \geq 0 \Rightarrow \lambda_j \geq 0$$

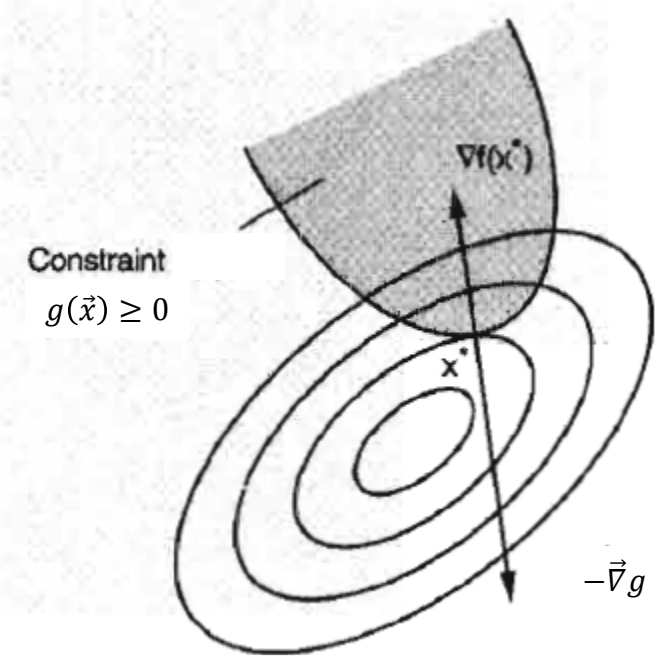


Figure 3.3.1. Illustration of the nonnegativity of the Lagrange multiplier for a problem with a single inequality constraint. If the constraint is inactive, then $\mu^* = 0$. Otherwise, $\nabla f(x^*)$ is normal to the constraint surface and points to the inside of the constraint set, and $\nabla g(x^*)$ is normal to the constraint surface and points to the inside of the constraint set. Thus $\nabla f(x^*)$ and $\nabla g(x^*)$ are collinear and have the same sign implying that the Lagrange multiplier is nonnegative.

Karush-Kuhn-Tucker Necessary Conditions

- Let $\vec{x}^*$ be a local minimum of the problem

$$\begin{gathered}\text{minimize } f(\vec{x}) \\ \text{subject to } h_1(\vec{x}) = 0, \cdots, h_m(\vec{x}) = 0 \\ g_1(\vec{x}) \geq 0, \cdots, g_r(\vec{x}) \geq 0\end{gathered}$$

  where f, h_i, g_j: $\mathbb{R}^n \to \mathbb{R}$ are continuously differentiable functions,

  and assume that $\vec{x}^*$ is regular

- Then there exist unique Lagrange multiplier vectors $\vec{\lambda}^* = (\lambda_1^*, \cdots, \lambda_m^*)^T$, $\vec{\mu}^* = (\mu_1^*, \cdots, \mu_r^*)^T$, such that

$$\begin{aligned}\vec{\nabla}_{\vec{x}} L\left(\vec{x}^*, \vec{\lambda}^*, \vec{\mu}^*\right) &= \vec{0} \\ \mu_j^* \geq 0, \qquad & j = 1, \cdots, r \\ \mu_j^* = 0, \qquad & \forall j \notin A(\vec{x}^*)\end{aligned}$$

- If in addition f, $\vec{h}$, and $\vec{g}$ are twice continuously differentiable, there holds

$$\vec{y} \nabla_{\vec{x}}^2 L\left(\vec{x}^*, \vec{\lambda}^*, \vec{\mu}^*\right) \vec{y} \geq 0$$

  for all $\vec{y} \in \mathbb{R}^n$ such that

$$\begin{aligned}\vec{\nabla} h_i(\vec{x}^*)^T \vec{y} = 0, \qquad & \forall i = 1, \cdots, m \\ \vec{\nabla} g_j(\vec{x}^*)^T \vec{y} = 0, \qquad & \forall j \in A(\vec{x}^*)\end{aligned}$$

Example

- Consider the problem

$$\text{minimize } f(x_1, x_2) = (x_1 - 2)^2 + (x_2 - 3)^2$$
$$\text{subject to } g(x_1, x_2) = x_1 - 3 \geq 0$$

- The Lagrangian function is

$$L(x_1, x_2, \mu) = f(x_1, x_2) - \mu g(x_1, x_2) = (x_1 - 2)^2 + (x_2 - 3)^2 - \mu(x_1 - 3)$$

- Case 1: If g is inactive, we have $\mu^* = 0$ and hence

$$L(x_1, x_2, \mu) = f(x_1, x_2) = (x_1 - 2)^2 + (x_2 - 3)^2$$

- Then

$$\vec{\nabla}_{\vec{x}} L(x_1^*, x_2^*, \mu^*) = \vec{0} \Rightarrow \begin{cases} 2(x_1^* - 2) = 0 \\ 2(x_2^* - 3) = 0 \end{cases}$$

$$\Rightarrow \begin{cases} x_1^* = 2 < 3 \\ x_2^* = 3 \end{cases}$$

- Hence no solution
- Case 2: If g is active, then $x_1^* = 3$ and

$$\vec{\nabla}_{\vec{x}} L(x_1^*, x_2^*, \mu^*) = \vec{0} \Rightarrow \begin{cases} 2(x_1^* - 2) - \mu^* = 0 \\ 2(x_2^* - 3) = 0 \end{cases}$$

$$\Rightarrow \begin{cases} x_1^* = 2 + \mu^*/2 \\ x_2^* = 3 \end{cases} \Rightarrow \begin{cases} \mu^* = 2 \\ x_2^* = 3 \end{cases}$$

- Hence $(x_1^*, x_2^*, \mu^*) = (3, 3, 2)$ and the minimum value is $f(3, 3) = (3 - 2)^2 + (3 - 3)^2 = 1$

Karush-Kuhn-Tucker Sufficient Conditions

- It can be shown that if for convex problems, the KKT conditions are also sufficient for optimality

Duality

Duality

- The lower bounds of the solution can be obtained by “duality”
- Because an equality constraint $h(\vec{x}) = 0$ problem can always be considered as two inequality constraints $h(\vec{x}) \geq 0$ and $-h(\vec{x}) \geq 0$, it suffices to consider only problems with inequality constraints

Infinite Step Penalty

- One way of converting a constrained problem

$$\begin{aligned} &\text{minimize } f(\vec{x}) \\ &\text{subject to } \vec{g}(\vec{x}) \succcurlyeq \vec{0} \end{aligned}$$

  where $\vec{g} = (g_1, \cdots, g_r)$: $\mathbb{R}^n \to \mathbb{R}^r$

  into an unconstrained one is to add a penalty function, to penalize the cost if the constraint is violated

- One obvious, but numerically not practical, penalty function to use is the indicator function

$$J(\vec{x}, \vec{\mu}) = f(\vec{x}) + \sum_{i=1}^{r} \mathbf{1}\big(g_i(\vec{x})\big)$$

  where $\mathbf{1}(z)$ is an infinite step function

$$\mathbf{1}(z) = \begin{cases} 0 & \text{if } z \geq 0 \\ \infty & \text{otherwise} \end{cases}$$

- This gives infinite penalty if the constraint is not satisfied, and hence would provide the same solution

Lagrangian

- However, this infinite step function is equally difficult to optimize
- The idea of Lagrange multipliers is to replace the infinite step function with a linear function

$$L(\vec{x}, \vec{\mu}) = f(\vec{x}) - \sum_{i=1}^{r} \mu_i g_i(\vec{x}) = f(\vec{x}) - \vec{\mu}^T \vec{g}$$

where the Lagrange multipliers satisfy

$$\vec{\mu} \succcurlyeq \vec{0}$$

Lagrangian Duality

- In general, duality in optimization is the idea of converting an optimization problem in one set of variables $\vec{x}$ (called the primal variables), into another optimization problem in a different set of variables $\vec{\mu}$ (called the dual variables)
- In this course, we will only discuss Lagrangian duality
- Now we may reformulate the primal problem in terms of Lagrangian

Lagrangian Duality

- Consider the original primal problem

$$\min_{\vec{x}} f(\vec{x})$$
$$\text{subject to } \vec{g}(\vec{x}) \succcurlyeq \vec{0}$$

- Define

$$L(\vec{x}, \vec{\mu}) = f(\vec{x}) - \vec{\mu}^T \vec{g}$$

  and

$$D(\vec{\mu}) = \min_{\vec{x}} L(\vec{x}, \vec{\mu})$$

- The associated Lagrangian dual problem is given by

$$\max_{\vec{\mu}} D(\vec{\mu})$$
$$\text{subject to } \vec{\mu} \succcurlyeq \vec{0}$$

Lagrangian Duality

- Subject to $\vec{\mu} \succcurlyeq \vec{0}$, we have

$$\max_{\vec{\mu}} L(\vec{x}, \vec{\mu}) = \max_{\vec{\mu}} (f(\vec{x}) - \vec{\mu}^T \vec{g}) = f(\vec{x}) + \sum_{i=1}^{r} \mathbf{1}\big(g_i(\vec{x})\big)$$

- Hence the primal problem can be written as

$$\min_{\vec{x}} \left(f(\vec{x}) + \sum_{i=1}^{r} \mathbf{1}\big(g_i(\vec{x})\big) \right) = \min_{\vec{x}} \max_{\vec{\mu}} L(\vec{x}, \vec{\mu})$$

Minimax Inequality

- For any function with two arguments $\varphi(\vec{x}, \vec{y})$, the maximin is less than or equal to the minimax, i.e.,

$$\max_{\vec{y}} \min_{\vec{x}} \varphi(\vec{x}, \vec{y}) \leq \min_{\vec{x}} \max_{\vec{y}} \varphi(\vec{x}, \vec{y})$$

- Proof: For all $\vec{x}, \vec{y}$

$$\min_{\vec{x}} \varphi(\vec{x}, \vec{y}) \leq \varphi(\vec{x}, \vec{y}) \leq \max_{\vec{y}} \varphi(\vec{x}, \vec{y})$$

  Since the above holds for all $\vec{y}$, we have

$$\max_{\vec{y}} \min_{\vec{x}} \varphi(\vec{x}, \vec{y}) \leq \max_{\vec{y}} \varphi(\vec{x}, \vec{y})$$

  Since the above holds for all $\vec{x}$, we have

$$\max_{\vec{y}} \min_{\vec{x}} \varphi(\vec{x}, \vec{y}) \leq \min_{\vec{x}} \max_{\vec{y}} \varphi(\vec{x}, \vec{y})$$

- Remark: Similarly, we also have

$$\min_{\vec{x}} \varphi(\vec{x}, \vec{y}) \leq \min_{\vec{x}} \max_{\vec{y}} \varphi(\vec{x}, \vec{y})$$

Lagrangian Duality

- By the minimax inequality, when subject to $\vec{\mu} \succcurlyeq \vec{0}$,

$$\min_{\vec{x}} \max_{\vec{\mu}} L(\vec{x}, \vec{\mu}) = \min_{\vec{x}} \left(f(\vec{x}) + \sum_{i=1}^{r} \mathbf{1}\big(g_i(\vec{x})\big) \right) \geq \max_{\vec{\mu}} \min_{\vec{x}} L(\vec{x}, \vec{\mu}) = \max_{\vec{\mu}} D(\vec{\mu})$$

- Note that in the above,
  - $\min_{\vec{x}} L(\vec{x}, \vec{\mu})$ is an unconstrained minimization
  - $\max_{\vec{\mu}} D(\vec{\mu})$ is a convex maximization because L is a linear (and hence convex) function of $\vec{\mu}$ and the constraint $\vec{\mu} \succcurlyeq \vec{0}$ is also convex. This is true even when the primal problem is not convex
- By the above, the solution of the dual problem is a lower bound of the solution of the primal problem
- This is called weak duality
- When the two solutions equal, we say that it satisfies strong duality and one may obtain the primal solution by finding the dual solution if that is easier
- It can be shown that when the primal problem is convex (and satisfying some additional technical conditions), strong quality holds

Example

- Consider the same problem

$$\text{minimize } f(x_1, x_2) = (x_1 - 2)^2 + (x_2 - 3)^2$$
$$\text{subject to } g(x_1, x_2) = x_1 - 3 \geq 0$$

- The above primal problem is convex because f is a convex function and g is linear (and hence convex).
- We will show that the same solution can be obtained by strong duality
- The Lagrangian function is

$$L(x_1, x_2, \mu) = f(x_1, x_2) - \mu g(x_1, x_2) = (x_1 - 2)^2 + (x_2 - 3)^2 - \mu(x_1 - 3)$$
$$= \left(x_1 - \frac{\mu}{2} - 2\right)^2 - \left(\frac{\mu}{2} + 2\right)^2 + 4 + (x_2 - 3)^2 + 3\mu$$
$$= \left(x_1 - \frac{\mu}{2} - 2\right)^2 + (x_2 - 3)^2 - \left(\frac{\mu}{2} + 2\right)^2 + 3\mu + 4$$

$$D(\mu) = \min_{\vec{x}} L(x_1, x_2, \mu) = L\left(\frac{\mu}{2} + 2, 3, \mu\right) = -\left(\frac{\mu}{2} + 2\right)^2 + 3\mu + 4 = \mu - \frac{\mu^2}{4}$$

$$=$$

$$\max_{\mu} D(\mu) = \max_{\mu}\left(\mu - \frac{\mu^2}{4}\right) = \max_{\mu}\left(1 - \left(\frac{\mu}{2} - 1\right)^2\right) = D(2) = L(3, 3, 2) = 1$$