Optimization

Industrial AI Lab.

Some materials from Prof. Zico Kolter at CMU

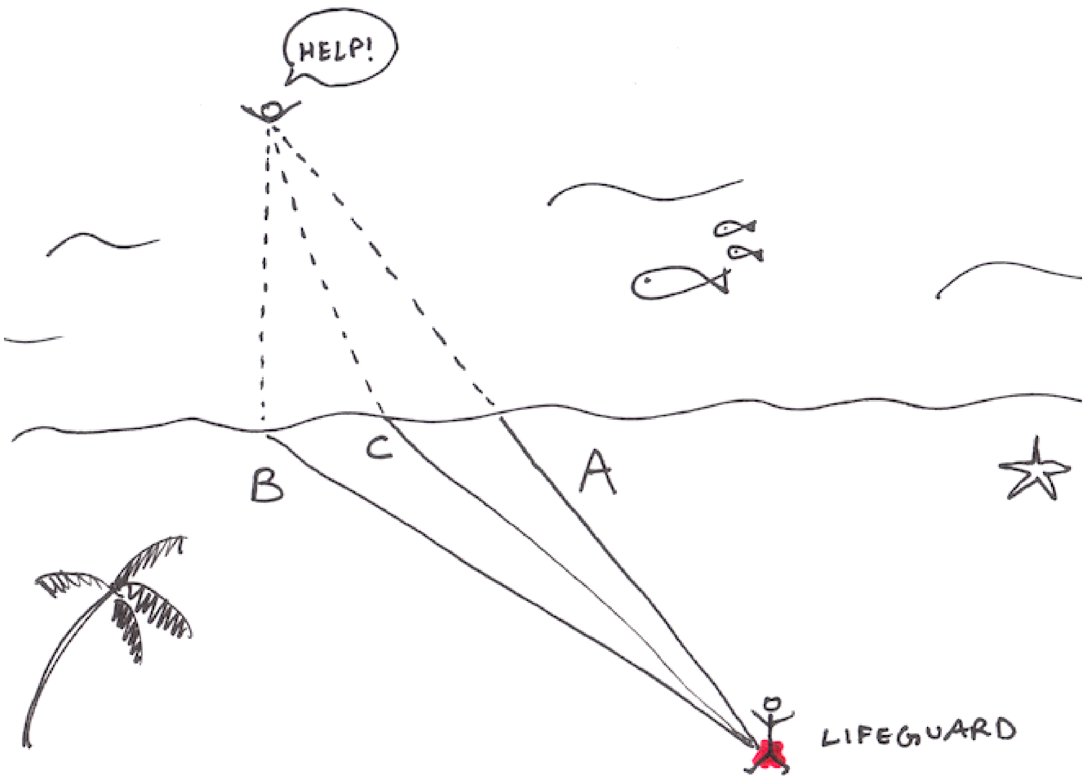
Optimization

- An important tool in
  1) Engineering problem solving and
  2) Decision science

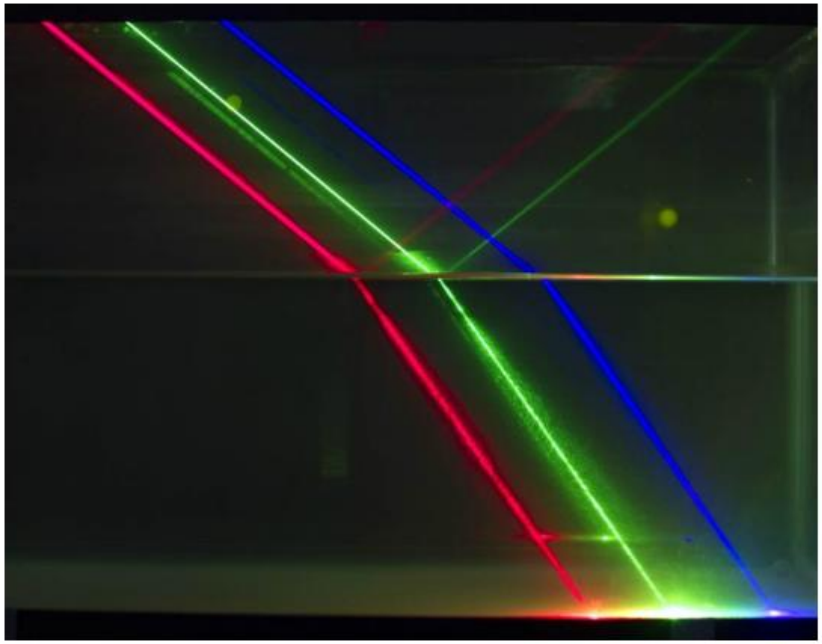
- People optimize

- Nature optimizes

People Optimize



Nature Optimizes



Optimization

- 3 key components
  1) Objective function
  2) Decision variable or unknown
  3) Constraints

- Procedures
  1) The process of identifying objective, variables, and constraints for a given problem (known as "modeling")
  2) Once the model has been formulated, optimization algorithm can be used to find its solutions

Optimization

- In mathematical expression

$$\begin{array}{lll} \min\limits_{x} & f(x) & \\ \text{subject to} & g_i(x) \leq 0, & i = 1, \cdots, m \end{array}$$

  - $x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n$ is the decision variable
  - $f: \mathbb{R}^n \to \mathbb{R}$ is objective function
  - Feasible region: $C = \{x: g_i(x) \leq 0, \quad i = 1, \cdots, m\}$
  - $x^* \in \mathbb{R}^n$ is an optimal solution if $x^* \in C$ and $f(x^*) \leq f(x), \forall x \in C$

Optimization

- In mathematical expression

$$\begin{aligned} \min_x \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, \qquad i = 1, \cdots, m \end{aligned}$$

- Remarks: equivalent

$$\begin{aligned} \min_x f(x) \quad &\leftrightarrow \quad \max_x -f(x) \\ g_i(x) \leq 0 \quad &\leftrightarrow \quad -g_i(x) \geq 0 \\ h(x) = 0 \quad &\leftrightarrow \quad \begin{cases} h(x) \leq 0 & \text{and} \\ h(x) \geq 0 \end{cases} \end{aligned}$$

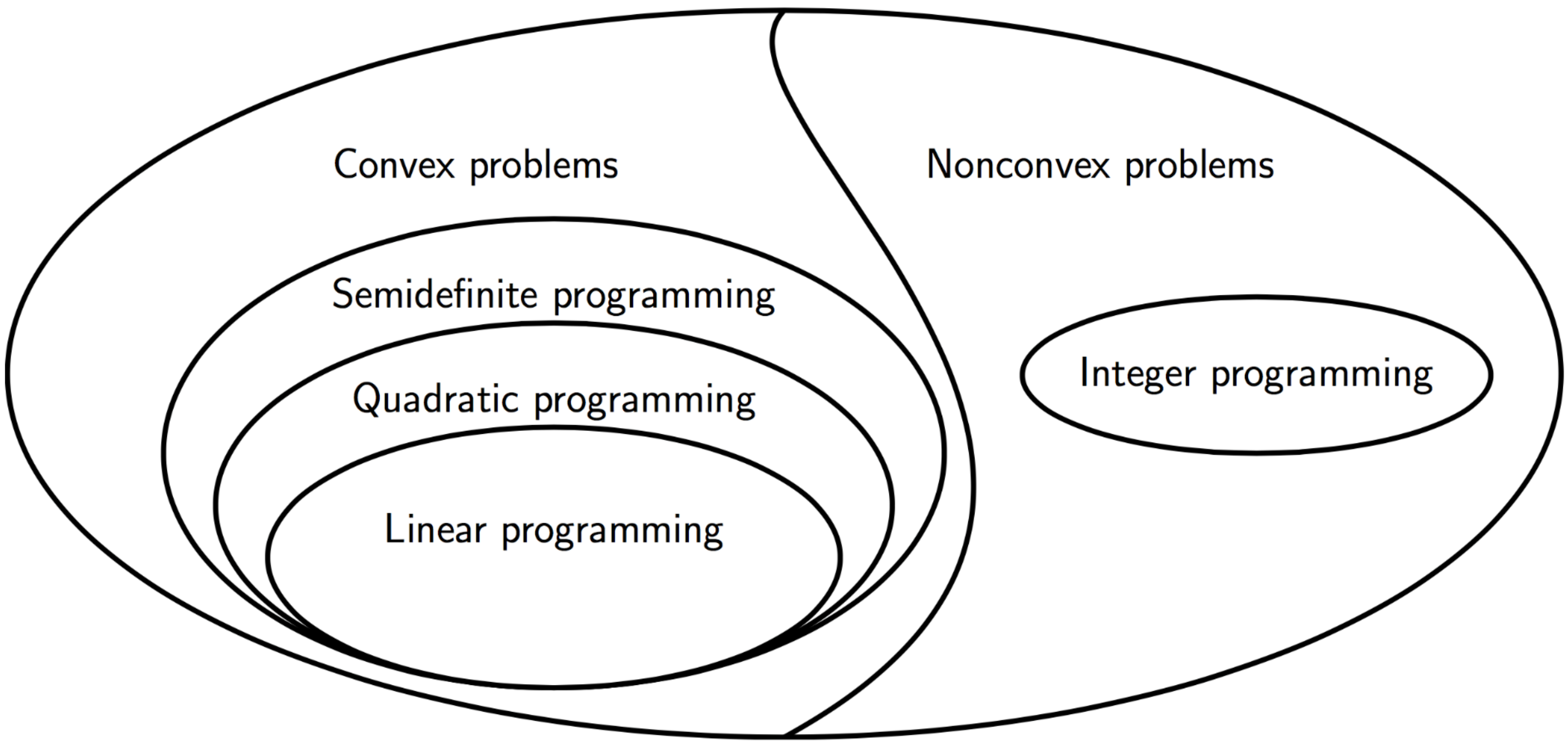
Unconstrained vs. Constrained

$$\underset{x}{\text{minimize}} \quad f(x)$$

vs.

$$\begin{aligned} &\underset{x}{\text{minimize}} && f(x) \\ &\text{subject to} && g_i(x) \leq 0, \quad i = 1, \dots, m \end{aligned}$$

Convex vs. Nonconvex



Convex Optimization

- An extremely powerful subset of all optimization problems

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & x \in \mathcal{C} \end{aligned}$$

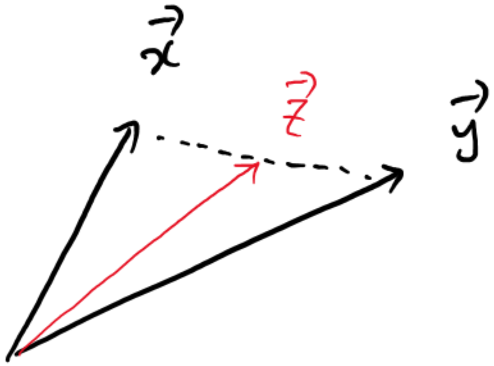
- $f: \mathbb{R}^n \to \mathbb{R}$ is a convex function and
- Feasible region C is a convex set

- Key property of convex optimization: all local solutions are global solutions

- We will use CVX (or CVXPY) as a convex optimization solver
  - Many examples later

Linear Interpolation between Two Points

- $\vec{z} = \theta\vec{x} + (1-\theta)\vec{y}$ and $\theta \in [0,1]$

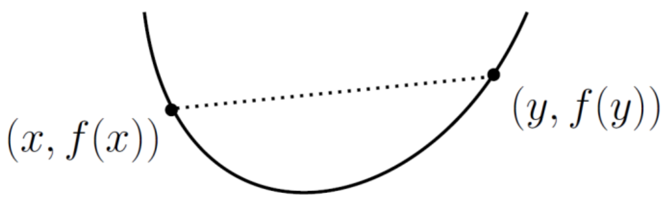


$$\vec{z} = \vec{y} + \theta\left(\vec{x} - \vec{y}\right), \qquad 0 \leq \theta \leq 1$$
$$= \theta\,\vec{x} + (1-\theta)\,\vec{y}$$

$$\vec{z} = \alpha\,\vec{x} + \beta\,\vec{y}, \qquad \alpha + \beta = 1 \quad \text{and} \quad 0 \leq \alpha, \beta$$

Convex Function and Convex Set

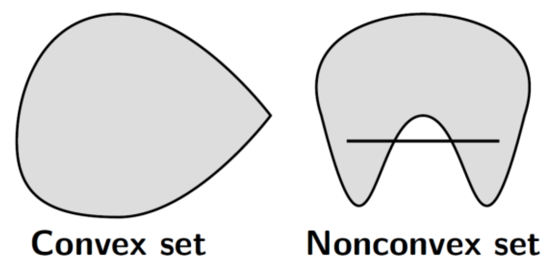
convex function



for any $x, y \in \mathbb{R}^n$ and $\theta \in [0, 1]$

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y)$$

convex set

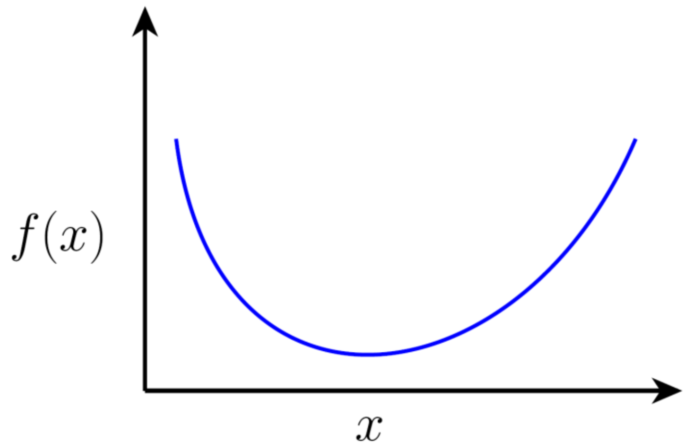


for a $x, y \in \mathcal{C}$ and $\theta \in [0, 1]$,

$$\theta x + (1-\theta) y \in \mathcal{C}$$

Solving Optimization Problems

- Starting with the unconstrained, one dimensional case



  – To find minimum point x^*, we can look at the derivative of the function $f(x)$: any location where $f'(x) = 0$ will be a “flat” point in the function

- For convex problems, this is guaranteed to be a minimum

- Generalization for multivariate function $f: \mathbb{R}^n \rightarrow \mathbb{R}$
  - the gradient of f must be zero

$$\nabla_x f(x) = 0$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

- For defined as above, *gradient* is a *n*-dimensional vector containing partial derivatives with respect to each dimension

$$\nabla_x f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix}$$

- For continuously differentiable f and unconstrained optimization, optimal point must have $\nabla_x f(x^*) = 0$

How to Find $\nabla_x f(x) = 0$

- Direct solution
  - In some cases, it is possible to analytically compute x^* such that $\nabla_x f(x^*) = 0$

$$f(x) = 2x_1^2 + x_2^2 + x_1 x_2 - 6x_1 - 5x_2$$

$$\Longrightarrow \nabla_x f(x) = \begin{bmatrix} 4x_1 + x_2 - 6 \\ 2x_2 + x_1 - 5 \end{bmatrix}$$

$$\Longrightarrow x^\star = \begin{bmatrix} 4 & 1 \\ 1 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 6 \\ 5 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Gradients

- Matrix derivatives

$\mathbf{y}$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$
$\mathbf{Ax}$	$\mathbf{A}^T$
$\mathbf{x}^T\mathbf{A}$	$\mathbf{A}$
$\mathbf{x}^T\mathbf{x}$	$2\mathbf{x}$
$\mathbf{x}^T\mathbf{Ax}$	$\mathbf{Ax} + \mathbf{A}^T\mathbf{x}$

How to Find $\nabla_x f(x) = 0$

- Direct solution
  - In some cases, it is possible to analytically compute x^* such that $\nabla_x f(x^*) = 0$

$$f(x) = 2x_1^2 + x_2^2 + x_1 x_2 - 6x_1 - 5x_2$$

$$\Longrightarrow \nabla_x f(x) = \begin{bmatrix} 4x_1 + x_2 - 6 \\ 2x_2 + x_1 - 5 \end{bmatrix}$$

$$\Longrightarrow x^\star = \begin{bmatrix} 4 & 1 \\ 1 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 6 \\ 5 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$\mathbf{y}$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$
$\mathbf{Ax}$	$\mathbf{A}^T$
$\mathbf{x}^T\mathbf{A}$	$\mathbf{A}$
$\mathbf{x}^T\mathbf{x}$	$2\mathbf{x}$
$\mathbf{x}^T\mathbf{Ax}$	$\mathbf{Ax} + \mathbf{A}^T\mathbf{x}$

Examples

$\mathbf{y}$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$
$\mathbf{A}\mathbf{x}$	$\mathbf{A}^T$
$\mathbf{x}^T\mathbf{A}$	$\mathbf{A}$
$\mathbf{x}^T\mathbf{x}$	$2\mathbf{x}$
$\mathbf{x}^T\mathbf{A}\mathbf{x}$	$\mathbf{A}\mathbf{x} + \mathbf{A}^T\mathbf{x}$

- affine function $g(x) = a^T x + b$

$$\nabla g(x) = a,$$

- quadratic function $g(x) = x^T P x + q^T x + r$, $\qquad P = P^T$

$$\nabla g(x) = 2Px + q,$$

Revisit: Least-Square Solution

Scalar Objective: $J = \|Ax - y\|^2$

$\mathbf{y}$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$
$\mathbf{Ax}$	$\mathbf{A}^T$
$\mathbf{x}^T\mathbf{A}$	$\mathbf{A}$
$\mathbf{x}^T\mathbf{x}$	$2\mathbf{x}$
$\mathbf{x}^T\mathbf{Ax}$	$\mathbf{Ax} + \mathbf{A}^T\mathbf{x}$

$$\begin{aligned} J(x) &= (Ax - y)^T (Ax - y) \\ &= \left(x^T A^T - y^T\right)(Ax - y) \\ &= x^T A^T A x - x^T A^T y - y^T A x + y^T y \end{aligned}$$

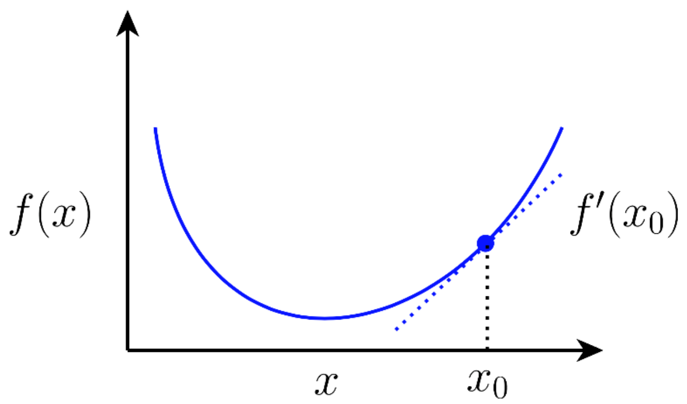
$$\begin{aligned} \frac{\partial J}{\partial x} &= A^T A x + \left(A^T A\right)^T x - A^T y - \left(y^T A\right)^T \\ &= 2A^T A x - 2A^T y = 0 \end{aligned}$$

$$\Longrightarrow \quad \left(A^T A\right) x = A^T y$$

$$\therefore \quad x^* = \left(A^T A\right)^{-1} A^T y$$

How to Find $\nabla_x f(x) = 0$

- Iterative methods
  - More commonly the condition that the gradient equal zero will not have an analytical solution, require iterative methods

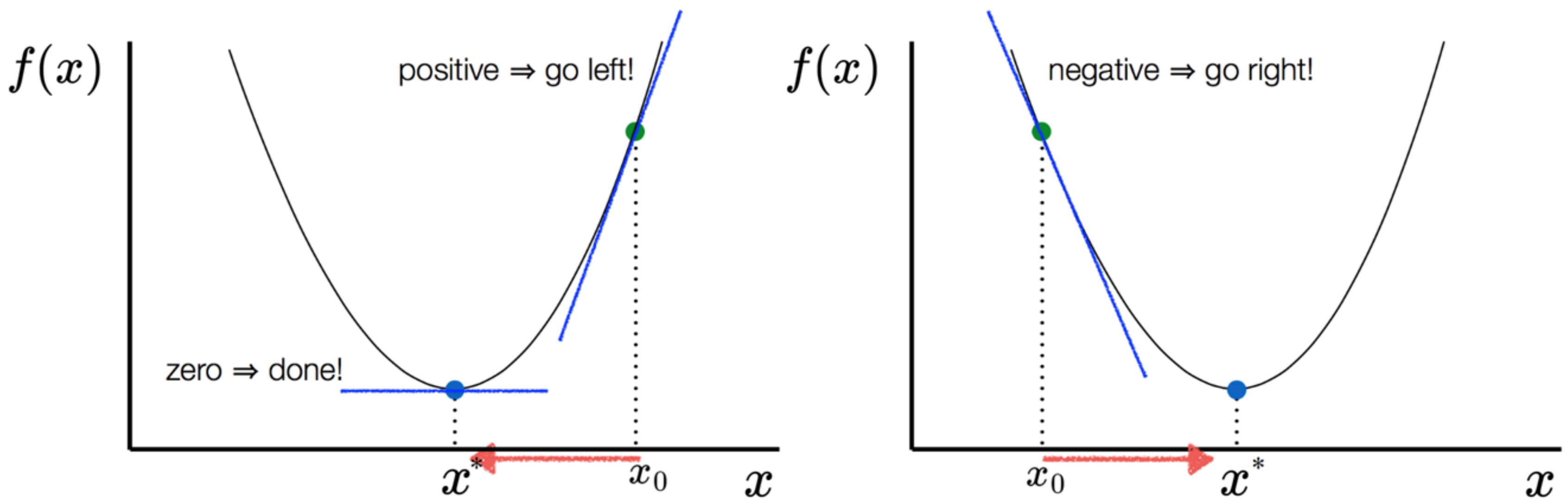


  - The gradient points in the direction of "steepest ascent" for function f

Descent Direction (1D)

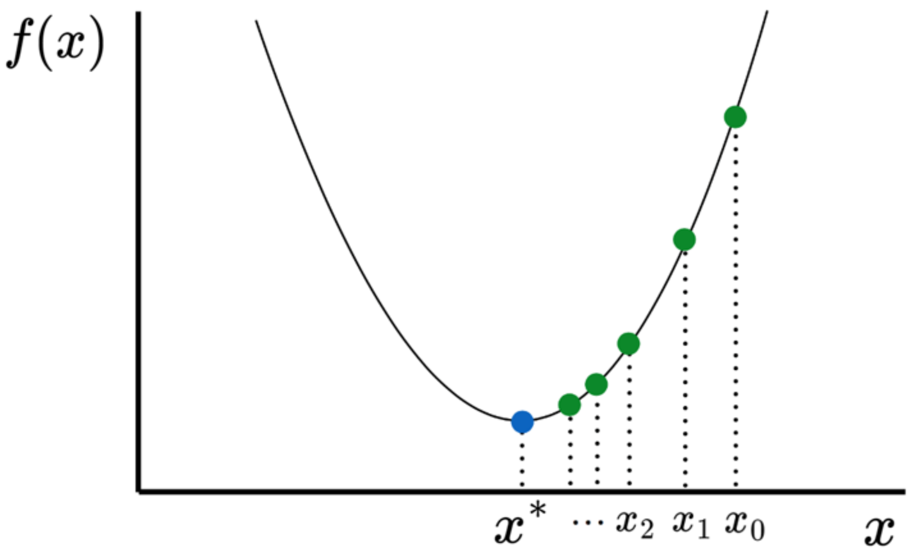
- It motivates the *gradient descent* algorithm, which repeatedly takes steps in the direction of the negative gradient

$$x \leftarrow x - \alpha \nabla_x f(x)$$ for some *step size* $\alpha > 0$

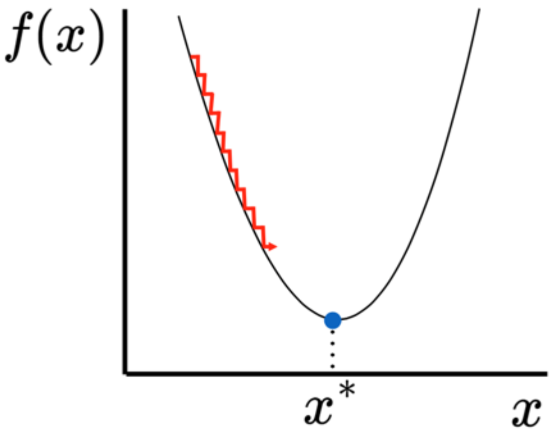


Gradient Descent

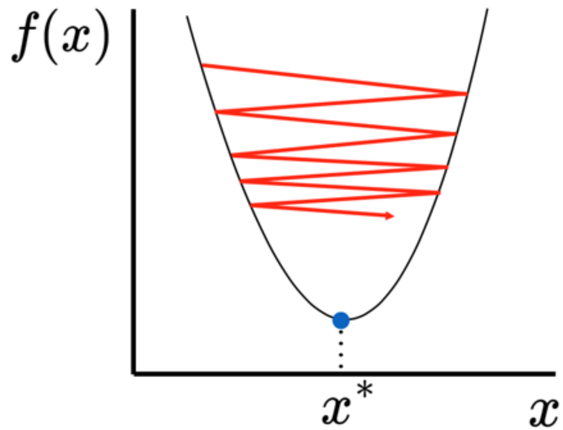
Repeat: $x \leftarrow x - \alpha \nabla_x f(x)$ for some *step size* $\alpha > 0$



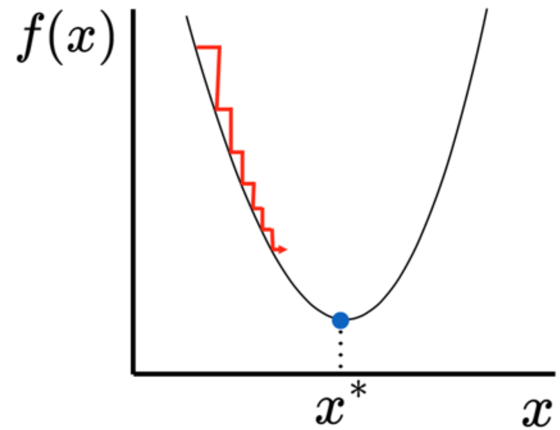
Choosing Step Size α



Too small: converge very slowly

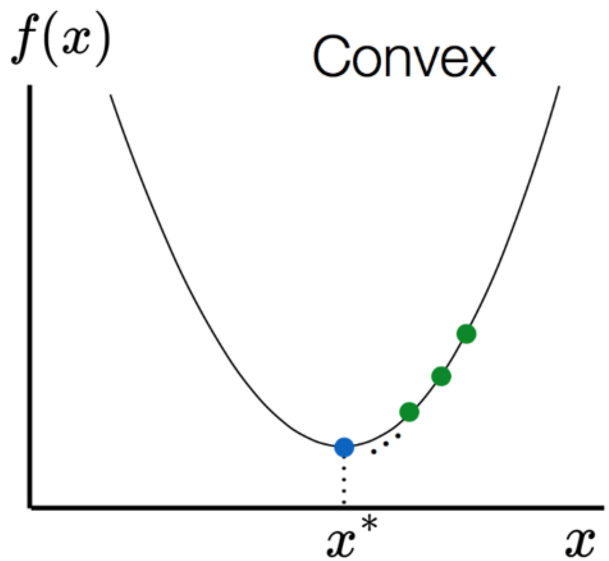


Too big: overshoot and even diverge

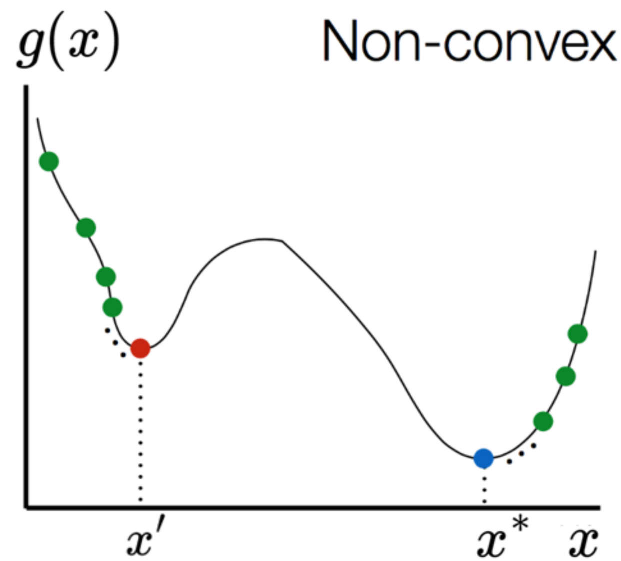


Reduce size over time

Where will We Converge?



Any local minimum is a global minimum

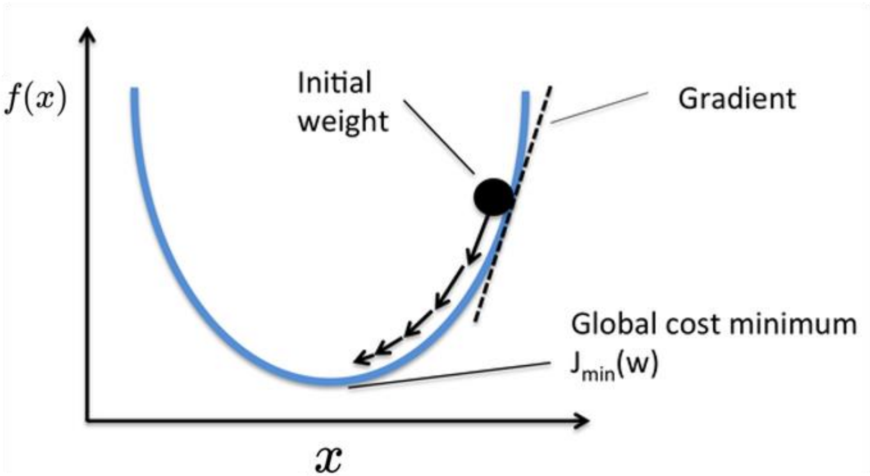


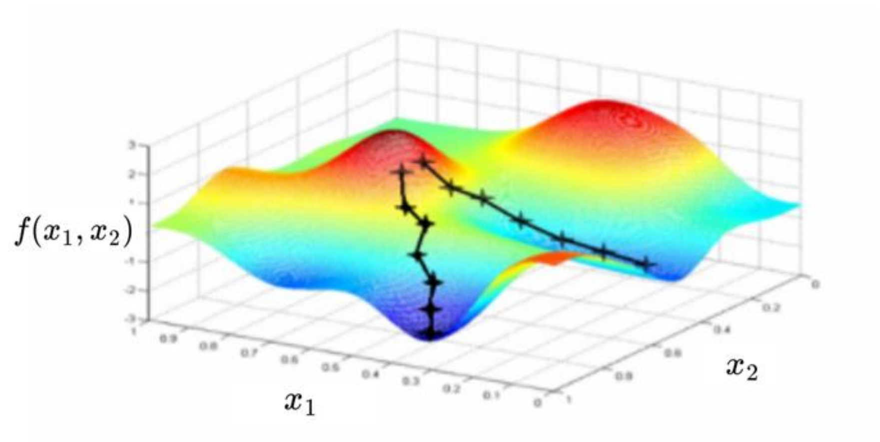
Multiple local minima may exist

- Random initialization
- Multiple trials

Gradient Descent in Higher Dimension

$$\text{Repeat: } x \leftarrow x - \alpha \nabla_x f(x)$$





Practically Solving Optimization Problems

- The good news: for many classes of optimization problems, people have already done all the “hard work” of developing numerical algorithms
  - A wide range of tools that can take optimization problems in “natural” forms and compute a solution

- We will use CVX (or CVXPY) as an optimization solver
  - Only for convex problems
  - Download: http://cvxr.com/cvx/

- Gradient descent
  - Neural networks/deep learning
  - (won’t be covered in this semester)

Linear Programming (Convex)

- Objective function and constraints are both linear
- Convex

$$\max \quad 3x_1 + \frac{3}{2}x_2 \qquad \leftarrow \text{objective function}$$

$$\begin{aligned} \text{subject to} \quad -1 \le x_1 \le 2 & \qquad \leftarrow \text{constraints} \\ 0 \le x_2 \le 3 & \end{aligned}$$

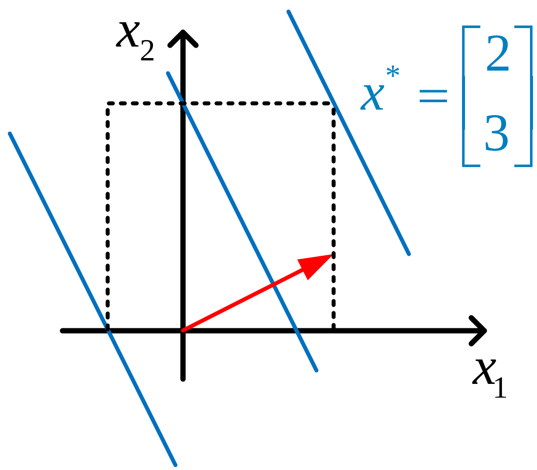
Method 1: Graphical Approach

$$\max \quad 3x_1 + \frac{3}{2}x_2$$

$$\text{subject to} \quad -1 \le x_1 \le 2$$
$$0 \le x_2 \le 3$$

$$3x_1 + 1.5x_2 = C \qquad \Rightarrow$$

$$x_2 = -2x_1 + \frac{2}{3}C$$



Method 2: CVXPY-based Solver

- CVXPY code
  - Many examples will be provided throughout the class

$$\begin{aligned} \max_x \quad & x_1 + x_2 \\ \text{subject to} \quad & 2x_1 + x_2 \le 29 \\ & x_1 + 2x_2 \le 25 \\ & x_1 \ge 2 \\ & x_2 \ge 5 \end{aligned} \quad \Longrightarrow \quad \begin{aligned} \min_x \quad & -\begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ \text{subject to} \quad & \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \le \begin{bmatrix} 29 \\ 25 \end{bmatrix} \\ & \begin{bmatrix} 2 \\ 5 \end{bmatrix} \le \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \le \begin{bmatrix} \; \end{bmatrix} \end{aligned}$$

CVXPY

```
import numpy as np
import matplotlib.pyplot as plt
import cvxpy as cvx
```

```
f = np.array([[1, 1]])
A = np.array([[2, 1], [1, 2]])
b = np.array([[29], [25]])
lb = np.array([[2], [5]])

x = cvx.Variable(2,1)

objective = cvx.Minimize(-f*x)
constraints = [A*x <= b, lb <= x]

prob = cvx.Problem(objective, constraints)
result = prob.solve()

print (x.value)
print (result)
```

$$\begin{aligned} \min_x \quad & -\begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ \text{subject to} \quad & \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 29 \\ 25 \end{bmatrix} \\ & \begin{bmatrix} 2 \\ 5 \end{bmatrix} \leq \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} \ \end{bmatrix} \end{aligned}$$

```
[[ 11.]
 [  7.]]
-17.999999998643816
```

Quadratic Form

$$q(x_1, \cdots, x_n) = \sum_{i=1}^{n} \sum_{j=1}^{n} h_{ij} x_i x_j = x^T H x$$

Quadratic Programming (Convex)

$$\begin{aligned} \min \quad & \frac{1}{2}x^2 + 3x + 4y \\ \text{subject to} \quad & x + 3y \geq 15 \\ & 2x + 5y \leq 100 \\ & 3x + 4y \leq 80 \\ & x, y \geq 0 \end{aligned} \quad \Longrightarrow \quad \begin{aligned} \min_X \quad & \tfrac{1}{2}X^T H X + f^T X \\ \text{subject to} \quad & AX \leq b \\ & A_{eq}X = b_{eq} \\ & LB \leq X \leq UB \end{aligned}$$

The problem can be found at
http://courses.csail.mit.edu/6.867/wiki/images/e/ef/Qp-quadprog.pdf

Quadratic Programming (Convex)

$$\begin{aligned} \min \quad & \frac{1}{2}x^2 + 3x + 4y \\ \text{subject to} \quad & x + 3y \geq 15 \\ & 2x + 5y \leq 100 \\ & 3x + 4y \leq 80 \\ & x, y \geq 0 \end{aligned} \quad \Longrightarrow \quad \begin{aligned} \min_X \quad & \tfrac{1}{2}X^T H X + f^T X \\ \text{subject to} \quad & AX \leq b \\ & A_{eq}X = b_{eq} \\ & LB \leq X \leq UB \end{aligned}$$

```
f = np.array([[3], [4]])
H = np.array([[1, 0], [0, 0]])
A = np.array([[-1, -3], [2, 5], [3, 4]])
b = np.array([[-15], [100], [80]])
lb = np.array([[0], [0]])

x = cvx.Variable(2,1)

objective = cvx.Minimize(cvx.quad_form(x, H) + f.T*x)
constraints = [A*x <= b,lb <= x]

prob = cvx.Problem(objective, constraints)
result = prob.solve()

print(x.value)
print(result)
```

```
[[  7.37787558e-10]
 [  5.00000000e+00]]
20.00000000034897
```

The problem can be found at
http://courses.csail.mit.edu/6.867/wiki/images/e/ef/Qp-quadprog.pdf

Gradient Descent

$$
\begin{aligned}
&\min \quad (x_1-3)^2+(x_2-3)^2 \\
= &\min \quad \frac{1}{2}\begin{bmatrix} x_1 & x_2 \end{bmatrix}\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} 6 & 6 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + 18
\end{aligned}
$$

$$\boxed{\min_X \qquad \frac{1}{2}X^T H X + f^T X}$$

$$\boxed{\nabla f(X_i)}$$

Gradient Descent

$$\begin{aligned} \min \quad & (x_1 - 3)^2 + (x_2 - 3)^2 \\ = \min \quad & \frac{1}{2} \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} 6 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + 18 \end{aligned}$$

- Update rule

$$X_{i+1} = X_i - \alpha_i \nabla f(X_i)$$

```
H = np.array([[2, 0],[0, 2]])
f = -np.array([[6],[6]])

x =  np.zeros((2,1))
alpha = 0.2

for i in range(25):
    g = H.dot(x) + f
    x = x - alpha*g

print(x)
```

```
[[ 2.99999147]
 [ 2.99999147]]
```

$$\boxed{\min_X \quad \frac{1}{2} X^T H X + f^T X}$$

$$\boxed{\nabla f(X_i)}$$

$\mathbf{y}$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$
$\mathbf{Ax}$	$\mathbf{A}^T$
$\mathbf{x}^T\mathbf{A}$	$\mathbf{A}$
$\mathbf{x}^T\mathbf{x}$	$2\mathbf{x}$
$\mathbf{x}^T\mathbf{Ax}$	$\mathbf{Ax} + \mathbf{A}^T\mathbf{x}$