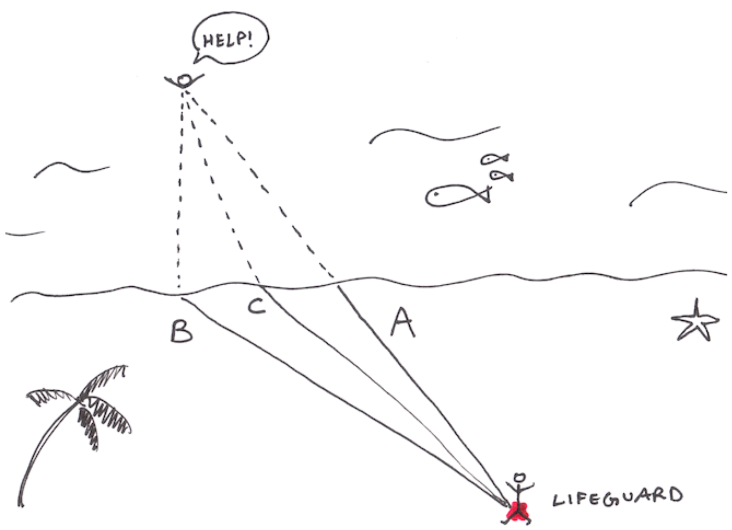
Optimization

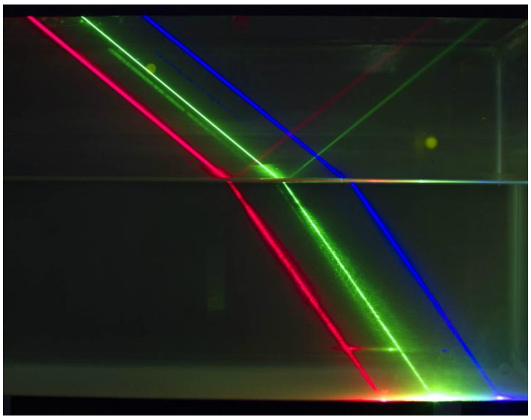
By Prof. Seungchul Lee
iSystems Design Lab
http://isystems.unist.ac.kr/
UNIST

Table of Contents

1. Optimization

- an important tool in 1) engineering problem solving and 2) decision science
- peolple optimize
- nature optimizes






(source: http://nautil.us/blog/to-save-drowning-people-ask-yourself-what-would-light-do (http://nautil.us/blog/to-save-drowning-people-ask-yourself-what-would-light-do))

3 key components

1. objective
2. decision variable or unknown
3. constraints

Procedures

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

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

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

The good news: for many classes of optimization problems, people have already done all the "hardwork" of developing numerical algorithms

- A wide range of tools that can take optimization problems in "natural" forms and compute a solution

2. Linear Programming (Convex)

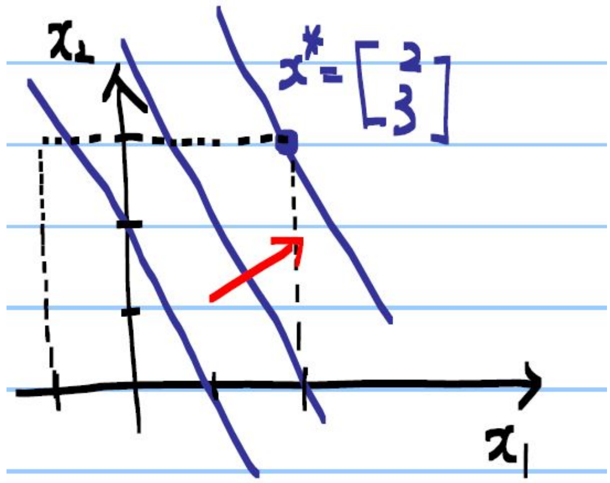
Objective function and Constraints are both linear

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

$$\text{subject to} \quad -1 \leq x_1 \leq 2 \qquad \leftarrow \text{constraints}$$
$$0 \leq x_2 \leq 3$$

Method 1: **graphical approach**

$$3x_1 + 1.5x_2 = C \qquad \Rightarrow \qquad x_2 = -2x_1 + \frac{2}{3}C$$



Method 2: **CVXPY-based solver**

CVXPY code

- Many examples will be provided throughout the class
- below example
    - http://cvxr.com/cvx/examples/ (http://cvxr.com/cvx/examples/) $\rightarrow$ Miscellaneous examples
    - http://inst.eecs.berkeley.edu/~ee127a/book/login/exa_lp_drug_manuf.html (http://inst.eecs.berkeley.edu/~ee127a/book/login/exa_lp_drug_manuf.html)

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

In [1]:

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

In [2]:

```
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)
```

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

3. Quadratic Programming (Convex)

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

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

In [3]:

```
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)
```

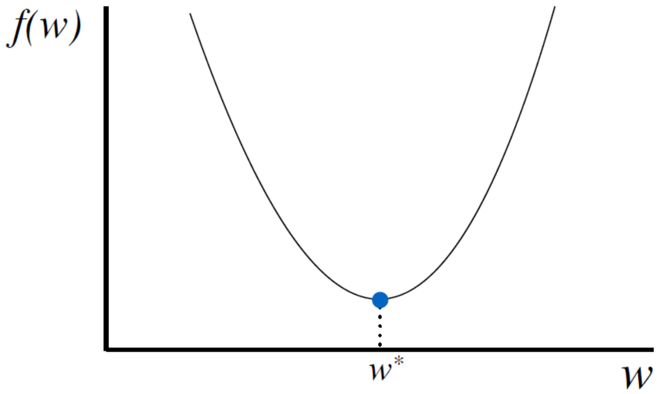
```
[[  7.37787558e-10]
 [  5.00000000e+00]]
20.00000000034897
```

4. Gradient Descent

Goal : Find ω^* that minimizes

$$f(\omega) = \|X\omega - y\|_2^2$$

- Closed form solution exists
- Gradient Descent is iterative (Intuition: go downhill !)



$$\text{Scalar objective: } f(\omega) = \|\omega x - y\|_2^2 = \sum_{j=1}^{n} \left(\omega x^{(j)} - y^{(j)}\right)^2$$

- gradient.pdf (./image_files/gradient.pdf)

In [2]:

```
%%html
<center><iframe src="./image_files/gradient.pdf", width=700, height = 400" frameborder
="0"></iframe></center>
```

Example

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

- update rule

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

In [6]:

```
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]]
```