# CS221 Fall 2018 Homework [1]

SUNet ID: [chiwang]
Name: [Chi Wang]

By turning in this assignment, I agree by the Stanford honor code and declare that all of this is my own work.

## Problem 1: Optimization and probability

### 1.a: What value of $\theta$ minimizes $f(\theta)$?

**Answer**: $\theta = \frac{\sum_{i=1}^{n} x_i}{n}$
In order to get minimized $f(\theta) = \frac{1}{2}\sum_{i=1}^{n} w_i(\theta - x_i)^2$ when $w_1, , w_n$ are positive real numbers, we need to find a $\theta$ which close to $x$ in average, so we have $\theta = \frac{\sum_{i=1}^{n} x_i}{n}$.
When some $w_i$ is negative, this approach won't work well. ex, For given $n = 8, W = \{1, 1, 1, 1, 1, 1, 1, -100\}, X = \{1, 1, 1, 1, 1, 1, 1, 9\}$, $\frac{\sum_{i=1}^{n} x_i}{n} = 2$, but $\theta = 2$ doesn't give minimized $f(\theta)$ in this case.

### 1.b: Does $f(x) \leq (x), f(x) = g(x)$ or $f(x) \geq g(x)$ hold for all x? Prove it.

**Proof**:We will prove $f(x) \geq g(x)$ for $f(x) = \sum_{i=1}^{d} \max_{s\in\{1,-1\}} sx_i$ and $g(x) = \max_{s\in\{1,-1\}} \sum_{i=1}^{d} sx_i$
Let's convert $f(x)$ and $g(x)$ to following expressions by using $s = 1$ and $s = -1$:

$$f(x) = \sum_{i=1}^{d} \max(x_i, -x_i) \tag{1}$$

$$g(x) = \max(\sum_{i=1}^{d} x_i, \sum_{i=1}^{d} -x_i) \tag{2}$$

In order to show $f(x) \geq g(x)$, we have below three cases to consider:
First case, all $x_i \in \mathbf{x}$ are not negative, in this case, we find $f(x) = \sum_{i=1}^{d} x_i$ and $g(x) = \sum_{i=1}^{d} x_i$, we could get $f(x) = g(x)$.
Second case, all $x_i \in \mathbf{x}$ are negative, then we find $f(x) = \sum_{i=1}^{d} -x_i$ and $g(x) = \sum_{i=1}^{d} -x_i$, which is $f(x) = g(x)$.
Last case, part of $x \in \mathbf{x}$ are negative and rest of them are not, let's define this set as $\mathbf{x}' = \{x|x < 0, x \in \mathbf{x}\}$, $\mathbf{x}' \subset \mathbf{x}$, then we could get $f(x) = \sum_{i=1}^{d} |x_i|$ and $g(x) = \sum_{i=1}^{d} |x_i| - \sum_{i=1}^{m} |x_i{}'|$ where $m$ is the size of $\mathbf{x}'$, it's clear that $f(x) > g(x)$.
Since $f(x)$ is either equal or greater than $g(x)$ in all condition, we could conclude that $f(x) \geq g(x)$.

## 1.c What is the expected number of points (as a function of $a$ and $b$) you will have when you stop.

**Answer**:
According to probability expectation formula, we could define function $f(a,b)$ as $f(a,b) = \frac{1}{6}b - \frac{1}{6}a + 3 \times \frac{1}{6}f(a,b)$, after keep repeating $f(a,b)$, we could have

$$\begin{aligned}
a &= \frac{1}{6}b - \frac{1}{6}a + \frac{1}{2}(\frac{1}{6}b - \frac{1}{6}a + \frac{1}{2}f(a,b)) && (3)\\
&= \frac{1}{6}b - \frac{1}{6}a + \frac{1}{2}(\frac{1}{6}b - \frac{1}{6}a + \frac{1}{2}(\frac{1}{6}b - \frac{1}{6}a + \frac{1}{2}f(a,b))) && (4)\\
&= \frac{1}{6}(b-a)(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + ... + \frac{1}{2^n}) + \frac{1}{2^n}f(a,b) && (5)\\
&= \frac{1}{6}(b-a)(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + ... + \frac{1}{2^n}) && (6)\\
&= \frac{1}{6}(b-a) \times 2(1 - (\frac{1}{2})^n) && (7)\\
&= \frac{1}{3}(b-a) && (8)
\end{aligned}$$

Hence, we could say the expect number of points is $\frac{1}{3}(b-a)$.

## 1.d what value of p maximizes $L(p)$? What is an intuitive interpretation of this value of $p$?

**Answer**: when $p = \frac{4}{7}$, it will maximize $L(p)$. From value $p$, we could interpret it as $p = \frac{Count(H)}{TotalDiceTimes}$.
Let's calculate derivative for $log(L(p))$, since $L(p) = p^4(1-p)^3$, we have $log(L(p))$'s derivative as:

$$\begin{aligned}
log'(L(p)) &= \frac{1}{L(P)ln2}L'(p) && (9)\\
&= \frac{1}{ln2 \times p^4(1-p)^3} \times (p^4 \times (1-p)^{3'} + p^{4'}(1-p)^3) && (10)\\
&= \frac{1}{ln2 \times p^4(1-p)^3} \times (p^4 \times 3(1-p)^2 \times (-1) + 4p^3(1-p)^3) && (11)\\
&= \frac{1}{ln2 \times p^4(1-p)^3} \times p^3 \times (1-p)^2 \times (4-7p) && (12)\\
&= \frac{4-7p}{ln2 \times p(1-p)} && (13)
\end{aligned}$$

Base on expression (13), when $p = \frac{4}{7}$, $log'(L(p)) = 0$, which represents $log(L(p))$'s max value.

### 1.e Compute the gradient of $f(x) = \sum_{i=1}^{n}\sum_{j=1}^{n}(a_i^\top w - b_j^\top w)^2 + \lambda\|w\|_2^2$

**Answer**:
For gradient of $f(w)$, let's calculate partial derivative $\bigtriangledown f(w) = \{\frac{\partial f(w)}{\partial w_1}, ..., \frac{\partial f(w)}{\partial w_d}\}$, for $w_k$ where $1 \leq k \leq d$, we have:

$$\frac{\partial f(w)}{\partial w_k} = \sum_{i=1}^{n}\sum_{j=1}^{n} 2(a_i^\top w - b_j^\top w) \times (a_k - b_k) + \frac{\partial \lambda \|w\|_2^2}{\partial w_k} \tag{14}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} 2(a_i^\top w - b_j^\top w) \times (a_k - b_k) + 2\lambda w_k \tag{15}$$

# Problem 2

### 2.a How many possible faces (choice of its component rectangles) are there?

**answer**: Consider to put an arbitrary organ on face first, it has roughly $n \times n$ choices for its location, and has roughly $n \times n$ choices for its size, then we could see there are nearly $n^4$ possible approach for the first organ.
According to problem description, requirement for the total 6 organs are the same, so we could see there are $O(n^{24})$ possible face choices.

### 2.b Give an algorithm for computing the minimum cost in the most efficient way. What is the runtime (just give the big-O)?

**answer**: List python code implementation below, algorithm runtime is $O(n^2)$.

```python
import sys

def ComputeMinCost(n, c):
    dp = [sys.maxint for i in range(n)]
    dp[0] = 0

    for i in range(n):
        pre = dp[0]
        for j in range(n):
            # cost(i,j) = min (move_from_top, move_from_left) + c(i,j)
            # dp array is used for caching cost of cells in above row.
            # pre is used for caching cost of the left cell
            dp[j] = min(dp[j], pre) + c(i,j)
            pre = dp[j]

    return dp[n-1]
```

## 2.c How many ways are there to reach the top?

**answer**: according to the problem description, we could find the following expression hold for number of ways to the 'x'th ($1 \leq x \leq n$) stair: num[x] = num[x-1] + num[x-2] + ... num[0], num[0] means move cur steps at once, num[x-2] means move 2 steps at once at stair x-2. So we could get below generic expression for $f(n)$:

$$\begin{aligned} f(n) &= f(n-1) + f(n-2) + ... + f(1) + f(0) && (16) \\ &= \sum_{i=1}^{n} f(n-i) && (17) \\ f(0) &= 1 && (18) \end{aligned}$$

## 2.d Devise a strategy that first does preprocessing in $O(nd^2)$ time, and then for any given vector w, takes $O(d^2)$ time instead to compute $f(w)$.

**answer**: let's transform $f(x) = \sum_{i=1}^{n} \sum_{j}^{n} (a_i^\top w - b_j^\top w)^2 + \lambda \|w\|_2^2$ to separate $w$ out of summation computing, see below expressions:

$$\begin{aligned} f(n) &= \sum_{i=1}^{n} \sum_{j=1}^{n} ((a_i^T w)(a_i^T w) - (a_i^T w)(b_j^T w) - (b_j^T w)(a_i^T w) + (b_j^T w)(b_j^T w)) + \lambda \|w\|_2^2 && (19) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} (w^T (a_i a_i^T) w - 2 w^T (a_i b_j) w + w^T (b_j a_j^T) w) + \lambda \|w\|_2^2 && (20) \\ &= w^T w (\sum_{i=1}^{n} (a_i^T a_i) + \sum_{j=1}^{n} (b_j^T b_j) - 2 \sum_{i=1}^{n} \sum_{j=1}^{n} (a_i b_j^T)) + \lambda \|w\|_2^2 && (21) \end{aligned}$$

from expression (21), we could see, if we could pre compute $K = \sum_{i=1}^{n} (a_i^T a_i) + \sum_{j=1}^{n} (b_j^T b_j) - 2 \sum_{i=1}^{n} \sum_{j=1}^{n} (a_i b_j^T)$, which takes $O(nd^2)$, then we could have below expression for $f(n)$:

$$f(n) = K w^T w + \lambda \|w\|_2^2$$

, this will take $O(d^2)$ time to compute if $k$ is pre-computed.