CS221 Spring 2019 Homework 1

SUNet ID: yijiez
Name: Yijie Zhuang
Last updated: April 9, 2019

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

(a) To minimize $f(\theta)$ given $w_1, \cdots, w_n$ be (strictly) positive , we need to make $f'(\theta) = 0$. Since $f(\theta) = \frac{1}{2}\sum_{i=1}^{n} w_i(\theta - x_i)^2$, $f'(\theta) = \sum_{i=1}^{n} w_i(\theta - x_i)$.

Given $\sum_{i=1}^{n} w_i(\theta - x_i) = 0$, we'll have

$$\theta = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i} \tag{1}$$

If some of the w_i's are negative, then the min may not exist. For example, in (1), if $\sum_{i=1}^{n} w_i = 0$, then min is not related to θ. If $\sum_{i=1}^{n} w_i < 0$, the min doesn't exist.

(b) Given $f(x) = \sum_{i=1}^{d} max_{s\in\{1,-1\}} s x_i$,

$$f(x) = \sum_{i=1}^{d} |x_i| \tag{2}$$

Given $g(x) = max_{s\in\{1,-1\}} \sum_{i=1}^{d} s x_i$,

$$g(x) = |\sum_{i=1}^{d} x_i| \tag{3}$$

From (2) and (3) we can conclude that $f(x) \geq g(x)$, and $f(x) = g(x)$ will be achieved only when all x_is are greater or equal to 0.

(c) Let $E(x)$ represent the expected value of points, assume we stop after y-th roll.

Stop after 1st roll: $E(x|y = 1) = 0$, where $P(y = 1) = \frac{1}{6}$

Stop after 2nd roll: $E(x|y = 2) = \frac{1}{5}(b - a)$, where $P(y = 2) = \frac{5}{6} * \frac{1}{6}$

Stop after 3rd roll: $E(x|y = 3) = \frac{2}{5}(b - a)$, where $P(y = 3) = \frac{5}{6} * \frac{5}{6} * \frac{1}{6}$
$\cdots$
Stop after nth roll: $E(x|y = n) = \frac{n-1}{5}(b - a)$, where $P(y = n) = \frac{5}{6} * \frac{5}{6} * \cdots * \frac{1}{6}$

$E(x) = E(x|y=1) * P(y=1) + E(x|y=2) * P(y=2) + \cdots + E(x|y=n) * P(y=n)$

$= \frac{1}{5}(b-a) * \frac{1}{6} \sum_{i=1}^{\infty} (n(\frac{5}{6})^n) =$

$$\frac{(b-a)}{30} \sum_{i=1}^{\infty} (i(\frac{5}{6})^i) \tag{4}$$

To solve (4), we can let:

$$S = \sum_{i=1}^{\infty} ip^i \tag{5}$$

Where $p = \frac{5}{6}$.
Expand (5), we'll have

$$S = p * 1 + p^2 * 2 + p^3 * 3 + \cdots + p^{n-1} * (n-1) \tag{6}$$
$$p * S = \quad p^2 * 1 + p^3 * 2 + \cdots + p^{n-1} * (n-2) + p^n * (n-1) \tag{7}$$

Deduct (7) from (6):

$$(1-p) * S = p + p^2 + p^3 + \cdots + p^n - p^{n-1} - p^n * (n-1) \tag{8}$$

$$S = p\frac{1-p^n}{(1-p)^2} - \frac{p^n(n-1)}{1-p} \tag{9}$$

When $n \to \infty$, $\frac{p^n*(n-1)}{1-p} \to 0$, as $p < 1$ and p^n converges faster than n.

$$E(x) = \frac{(b-a)}{30} S = \frac{(b-a)}{30} * S * \frac{1-p^n}{(1-p)^2} = b - a \tag{10}$$

Thus, the expected number of points (as a function of a and b) when stop is $(b-a)$.

(d) Assume our strategy is to stop after n-th roll, $n \in [0, \infty)$, Let $E(x)$ represent the expected value of points.

Stop at the beginning: $E = 15$

Stop after 1st roll: $E(n=1) = 3 * 1 + 15 * \frac{3}{5} = 12$

Stop after 2nd roll: $E(n=2) = 3 * (1 + \frac{3}{5}) + 15 * (\frac{3}{5})^2 = 10.2$

Stop after 3rd roll: $E(n=3) = 3 * (1 + \frac{3}{5} + \frac{3}{5}^2) + 15 * (\frac{3}{5})^3$
...
Stop after nth roll: $E(n) = 3 * (1 + \frac{3}{5} + \frac{3}{5}^2 + \cdots + (\frac{3}{5})^{n-1}) + 15 * (\frac{3}{5})^n = \frac{15}{2}(1 + (\frac{3}{5})^n)$

If there exists n to make $E(n) > 15$, then $(\frac{3}{5})^n > 1$, which won't happen assuming $n \in [0, \infty)$.

So, the best strategy is to stop at the beginning and win 15 points. If the probability of rolling an even number was 0, then we should keep playing and the expectation of total points will be ∞.

(e) Given $L(p) = p^4(1-p)^3$, let $G(p) = \ln L(p)$, we can get $G(p) = \ln p^4 + \ln (1-p)^3 = 4 \ln p + 3 \ln (1-p)$.

Let $G'(p) = \frac{4}{p} - \frac{3}{1-p} = 0$, we can get $p = \frac{4}{7}$. As $0 < p < 1$, this value will maximize the result.

An intuitive interpretation of this value of p is the coin head appears 4 times when flipping 7 times.

(f) Given $f(\mathbf{w}) = \sum_{i=1}^{n} \sum_{j=1}^{n} (a_i^\top w - b_i^\top w)^2 + \lambda \|w\|_2^2$, assume $g(w) = \sum_{i=1}^{n} \sum_{j=1}^{n} (a_i^\top w - b_j^\top w)^2$ and $t(w) = \lambda \|w\|_2^2$. $f(w) = g(w) + t(w)$.

$g(w) = \sum_{i=1}^{n} \sum_{j=1}^{n} (a_{i1} * w_1 + a_{i2} * w_2 + \cdots + a_{id} * w_d - b_{j1} * w_1 - b_{j2} * w_2 + \cdots - b_{jd} * w_d)^2$

$\frac{\partial g(w)}{\partial w_1} = 2 * \sum_{i=1}^{n} \sum_{j=1}^{n} (a_{i1} * w_1 + a_{i2} * w_2 + \cdots + a_{id} * w_d - b_{j1} * w_1 - b_{j2} * w_2 + \cdots - b_{jd} * w_d)(a_{i1} - b_{j1})$

$$\nabla g(w) = (\frac{\partial g(w)}{\partial w_1}, \cdots, \frac{\partial g(w)}{\partial w_d})^\top = 2 * \sum_{i=1}^{n} \sum_{j=1}^{n} (a_i^\top w - b_j^\top w)(a_i - b_j) \tag{11}$$

Similarly,

$t(w) = \lambda(w_1^2 + w_2^2 + \cdots + w_k^2)$

$\frac{\partial t(w)}{\partial w_1} = 2\lambda w_1$

$$\nabla t(w) = 2\lambda w \tag{12}$$

Given (11) and (12),

$$\nabla f(w) = 2 * \sum_{i=1}^{n} \sum_{j=1}^{n} (a_i^\top w - b_j^\top w)(a_i - b_j) + 2\lambda w \tag{13}$$

Problem 2: Complexity

(a) In a $n * n$ grid, there are $n * n$ vertices for placing a rectangle. For each vertex, possible number of rectangles are different.

For vertex (0,0), possible number of rectangles = $n * n$,
For vertex (1,0), possible number of rectangles = $(n-1) * n$,
For vertex (n - 1,0), possible number of rectangles = $1 * n$,
For vertex (n - 1,n - 1), possible number of rectangles = $1 * 1$.

Consider vertices (0, 0) $\rightarrow$ (n - 1, 0), possible number of rectangles = $n * n + n * (n-1) + n = n * \frac{n(n+1)}{2}$.
Consider vertices (0, 1) $\rightarrow$ (n - 1, 1), possible number of rectangles = $(n-1) * n + (n-1) * (n-1) + (n-1) * 1 = (n-1) * \frac{n(n+1)}{2}$.
Thus, when consider all vertices, the possible number of rectangles is $6 * \frac{n(n+1)}{2} * \frac{n(n+1)}{2} = \frac{3}{2}(n^4 + 2n^3 + n^2)$

Consider asymptotic complexity, since there are 6 components, the answer is $O((n^4)^6) = O(n^{24})$.

(b) To compute the minimum cost, go through all cells and call cost(i,j) to compute the cost, while adding the minimum cost between the left cell and the upper cell. More precisely, we could use the following pseudocode.

Algorithm 1: ComputeMinCost returns the minimum cost.

Input: An $n * n$ grid.
Output: The minimum cost from (1, 1) to (n, n).
tmp $\leftarrow int[n][n]$;
for $i \in \{1, \ldots, n\}$ **do**
  for $j \in \{1, \ldots, n\}$ **do**
    tmp $[i][j] \leftarrow$ `Cost(`i, j`)` $+ min($tmp$[i-1][j],$ tmp$[i][j-1])$;
return $tmp[n][n]$

Given the pseudocode, the runtime is $O(n^2)$.

(c) let $f(n)$ represent how many ways are there to reach the top for a staircase with n steps, we'll have:
$f(0) = 1$
$f(1) = f(0) = 1$
$f(2) = f(0) + f(1) = 2$, means (1) take one step, take one step; (2) take two
$f(3) = f(0) + f(1) + f(2) = 4$, means (1) take one step, take one step, take one step (2) take two steps, take one step (3) take one step, take two steps (4) take three steps.
...
$f(n) = f(0) + f(1) + \cdots + f(n-1) = 2 * f(n-1)$. This then becomes a problem to sum a Geometric Sequence.

Use the formula: $\sum_{i=0}^{n-2} 2^i = \frac{1-2^{n-1}}{1-2} = 2^{n-1} - 1 \Rightarrow$

$$f(n) = 1 + \sum_{i=0}^{i=n-2} 2^i = 2^{n-1} (n \in [1, \infty)) \tag{14}$$

(d)

$$f(\mathbf{w}) = \sum_{i=1}^{n} \sum_{j=1}^{n} (a_i^\top w - b_i^\top w)^2 + \lambda \|w\|_2^2 \tag{15}$$

In $f(\mathbf{w})$:

$$(a_i^\top w - b_i^\top w)^2 = [(a_i^\top - b_j^\top)w]^\top [(a_i^\top - b_j^\top)w] = w^\top * (a_i^\top - b_j^\top)^\top * (a_i^\top - b_j^\top) * w \tag{16}$$

So $\sum_{i=1}^{n} \sum_{j=1}^{n} (a_i^\top w - b_i^\top w)^2$ can be expanded to:

$$\lambda = \sum_{i=1}^{n} \sum_{j=1}^{n} (a_i^\top - b_j^\top)^\top (a_i^\top - b_j^\top) = \sum_{i=1}^{n} a_i a_i^\top - \sum_{i=1}^{n} a_i \sum_{j=1}^{n} b_j^\top - \sum_{j=1}^{n} b_j \sum_{i=1}^{n} a_i^\top + \sum_{j=1}^{n} b_j b_j^\top \tag{17}$$

The complexity for $\sum_{i=1}^{n} a_i a_i^\top$ and $\sum_{j=1}^{n} b_j b_j^\top$ are both $O(nd^2)$, while the complexity for $\sum_{i=1}^{n} a_i \sum_{j=1}^{n} b_j^\top$ and $\sum_{j=1}^{n} b_j \sum_{i=1}^{n} a_i^\top$ are both $O(2nd + d^2)$

Thus, the overall complexity of (17) is $O(nd^2)$.

After (16) has been computed, we can compute (16) easily by doing two multiplications: $w^\top * \lambda * w$. Since w is a $d \times 1$ matrix, this will add an extra $O(2d^2)$ complexity to (17).

Now, consider the second part of $f(\mathbf{w})$: $\lambda \|\mathbf{w}\|_2^2 = \lambda(w_1^2 + w_2^2 + \cdots + w_d^2)$, its complexity is d.

To summarize, for $f(\mathbf{w})$, by computing (17) first then (16), we can achieve $O(nd^2) + O(d^2)$ time.