

# Foundations

Prabhjot Singh Rai

October 1, 2018

## 1 Optimization and probability

**Solution 1a:** In order to check whether the minima exists for the function,  $f(\theta)$ , we calculate  $f''(\theta)$ .

$$\begin{aligned}\frac{df(\theta)}{d\theta} &= \sum_{i=1}^n w_i(\theta - x_i) \\ g(\theta) &= \sum_{i=1}^n w_i\theta - \sum_{i=1}^n w_ix_i\end{aligned}$$

Differentiating  $g(\theta)$ , we get

$$\frac{dg(\theta)}{d\theta} = \sum_{i=1}^n w_i \quad (1)$$

Since it is given that  $w_1, \dots, w_n$  are positive real numbers, therefore  $f''(\theta) > 0$ , which proves that the graph of this quadratic function is convex and minima exists at value of  $\theta$  when  $f'(\theta) = 0$ .

$$\begin{aligned}\frac{df(\theta)}{d\theta} &= \sum_{i=1}^n w_i(\theta - x_i) \\ 0 &= \sum_{i=1}^n w_i\theta - \sum_{i=1}^n w_ix_i \\ \theta &= \frac{\sum_{i=1}^n w_ix_i}{\sum_{i=1}^n w_i}\end{aligned}$$

Therefore, the value of  $\theta$  which minimizes  $f(\theta)$  is  $\frac{\sum_{i=1}^n w_ix_i}{\sum_{i=1}^n w_i}$ . If some of the  $w_i$ 's are negative, then from equation (1), it's not necessary that  $f''(\theta)$  is always positive. If  $f''(\theta) < 0$ , the function will be a concave function and minima will not be defined. And if a point of inflection exists ( $f''(\theta) = 0$ ), then the function would be a straight line and minima won't be defined.

**Solution 1b.** Let  $n$  vectors out of  $x_1, \dots, x_d$  be negative where  $n \leq d$ . Let sum of negative vectors be  $-\beta$  and sum of positive vectors be  $\alpha$ , where  $\alpha$  and  $\beta$  are both positive. Evaluating  $f(x)$ ,

$$\begin{aligned}f(x) &= \sum_{i=1}^d \max_{s \in \{1, -1\}} s x_i \\ &= \max_{s \in \{1, -1\}} s(\alpha) + \max_{s \in \{1, -1\}} s(-\beta) \\ &= \alpha + \beta\end{aligned}$$

$\max_{s \in \{1, -1\}}$  is 1 for  $\alpha$  and -1 for  $\beta$

Evaluating  $g(x)$ ,

$$\begin{aligned}
g(x) &= \max_{s \in \{1, -1\}} \sum_{i=1}^d s x_i \\
&= \max_{s \in \{1, -1\}} s \sum_{i=1}^d x_i \\
&= \max_{s \in \{1, -1\}} s(\alpha - \beta) \\
g(x) &= \begin{cases} \alpha - \beta & \text{for } \alpha > \beta \\ \beta - \alpha & \text{for } \beta > \alpha \\ 0 & \text{for } \alpha = \beta \end{cases}
\end{aligned}$$

Comparing  $f(x)$  and  $g(x)$  for different cases:

$$f(x) - g(x) = \begin{cases} 2\beta & \text{for } \alpha > \beta \\ 2\alpha & \text{for } \beta > \alpha \\ 2\alpha = 2\beta & \text{for } \beta = \alpha \end{cases}$$

Therefore, if  $\alpha$  and  $\beta$  are non-zero,  $f(x) > g(x)$ , else  $f(x) = g(x)$ . Hence, for all  $\mathbf{x}$ ,  $f(x) \geq g(x)$ .

**Solution 1c.** Let  $E(a, b)$  denote the expected number of points when we stop. Only cases to consider would be 1, 2 and 6. Let's divide the sequences into 3 cases.

One third times we get a 1 first, and then we stop.

One third times we get a 2 first, and then we repeat.

One third times we get a 6 first, and then we repeat.

Expressing in equation form, it would be

$$\begin{aligned}
E(x) &= \frac{1}{3}(0) + \frac{1}{3}(-a + E(x)) + \frac{1}{3}(b + E(x)) \\
3E(x) &= -a + b + 2E(x) \\
E(x) &= b - a
\end{aligned}$$

**Solution 1d.** Since we know that value of  $p$  for which  $\log L(p) = 0$  must also satisfy  $L(p) = 0$ , checking the point where the slope of the function is zero

$$\begin{aligned}
\log L(p) &= 4 \log p + 3 \log(1 - p) \\
\frac{d \log L(p)}{dp} &= \frac{4}{p} - \frac{3}{1 - p} \\
0 &= -4 + 4p + 3p \\
p &= \frac{4}{7}
\end{aligned}$$

In order to check if it's a maxima or a minima at  $p = \frac{4}{7}$ , let's check the sign on substituting  $p$  in  $L''(p)$

$$\begin{aligned}\frac{dL(p)}{dp} &= 4x^3(1-x)^3 - 3x^4(1-x)^2 \\ \frac{d^2L(p)}{dp^2} &= (1-x)^2(12x^2 - 28x^3) - 2(1-x)(4x^3 - 7x^4)\end{aligned}$$

Substituting  $p = \frac{4}{7}$  we get an approximate value of  $-0.238$ , therefore, the function has a maxima at this point.

Intuitively, the maximum probability would be when  $p$  is equivalent to the number of times the desired result came up divided by the total number of events.

**Solution 1e.**

$$\begin{aligned}f(w) &= \sum_{i=1}^n \sum_{j=1}^n (a_i^\top w - b_j^\top w)^2 + \lambda \|w\|_2^2 \\ &= \sum_{i=1}^n \sum_{j=1}^n ((a_i - b_j)^\top w)^2 + \lambda \sum_{k=1}^d w_k^2 \quad (\dots \text{using distributive property})\end{aligned}$$

Calculating  $f'(w)$ ,

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

Thus, the gradient  $\nabla f(w)$  can be expressed as

$$\begin{aligned}\nabla f(w) &= \left( \frac{\partial f(w)}{\partial w_1}, \dots, \frac{\partial f(w)}{\partial w_d} \right)^\top \\ &= \left( 2 \sum_{i=1}^n \sum_{j=1}^n ((a_i^\top - b_j^\top)w_1)(a_i^\top - b_j^\top) + \lambda 2w_1, \dots, 2 \sum_{i=1}^n \sum_{j=1}^n ((a_i^\top - b_j^\top)w_d)(a_i^\top - b_j^\top) + \lambda 2w_d \right)\end{aligned}$$

## 2 Complexity

**Solution 2a.** This problem can be thought of creating a rectangle inside a rectangle of width  $m$  pixels and height  $n$  pixels ( $m = n$  for this problem, but we will use it later). Therefore, the number of points which can be selected to create a rectangle are  $m + 1$  and  $n + 1$ .

Total number of points = (Number of ways first point is selected) (Number of ways next point is selected on the remaining points on width) (Number of ways next point is selected on remaining points on height) / (Number of points of rectangle, since for each point, the remaining points are selected again)

$$\begin{aligned}\text{No of ways} &= \frac{\binom{(m+1)(n+1)}{1} \binom{m}{1} \binom{n}{1}}{4} \\ &= \frac{m(m+1)n(n+1)}{4}\end{aligned}$$

For the given problem,  $m = n$ . Therefore

$$\text{No of ways 1 rectange can be created} = \frac{n^2(n+1)^2}{4}$$

This way, the number of possible faces are

$$\text{No of ways 6 rectange can be created} = \left(\frac{n^2(n+1)^2}{4}\right)^6$$

Therefore, asymptotic complexity for this function is  $O(n^{24})$ .

**Solution 2b.** Let  $v(i, j)$  represent the cost at a particular point. Cost of touching the point  $(i, j)$  can be defined as the sum of cost at  $(i, j)$  and (cost of touching point at  $(i - 1, j)$  or cost of touching point at  $(i, j - 1)$ ).

$$c(i, j) = v(i, j) + c(i - 1, j) \text{ or } c(i, j) = v(i, j) + c(i, j - 1)$$

Minimum cost to reach point  $(i, j)$  is can be defined as

$$c_{min}(i, j) = v(i, j) + \min_{(i,j) \in \{(i-1,j), (i,j-1)\}} c_{min}(i, j)$$

where  $\min_{(i,j) \in \{(i-1,j), (i,j-1)\}} c_{min}(i, j)$  is the minimum of the cost between reaching point  $(i - 1, j)$  and  $(i, j - 1)$ .

Let `cost_matrix` be a matrix consisting of costs at all the points. Algorithm to compute this efficiently is defined as follows

```
def min_cost(i, j, cost_matrix):
    # iterating over first column and computing cost
    # for traversing till the element (i, 0)
    for m in range(1, i + 1):
        cost_matrix[m][0] = cost_matrix[m - 1][0] + cost_matrix[m][0]

    # iterating over first row and computing cost
```

```

# for traversing till the element (j, 0)
for n in range(1, j + 1):
    cost_matrix[0][n] = cost_matrix[0][n - 1] + cost_matrix[0][n]

# iterating over remaining entries for which the
# cost for top and left elements have already
# been computed
for m in range(1, i + 1):
    for n in range(1, j + 1):
        cost_matrix[m][n] = min(cost_matrix[m - 1][n], cost_matrix[m][n - 1]) \
            + cost_matrix[m][n]

return cost_matrix[i][j]

```

Since there are two nested loops for each dimension, the runtime of the function is  $O(n^2)$ .

**Solution 2c.** Total number of ways to reach to the top can be calculated by counting the number of ways the person can select a step on moving forward towards his way to the top, given that he has to land on the final step.

There can be two different cases for each step, either the step is selected on the way to the top or is skipped.

$$\begin{aligned}
 \text{Total ways} &= (\text{1st Selected or Skipped})(\text{2nd Selected or Skipped}) \dots (\text{Last Selected}) \\
 &= 2 * 2 * 2 \dots 1 \\
 &= 2^{n-1}
 \end{aligned}$$

Therefore, total ways are  $2^{n-1}$ .

**Solution 2d.**