Assignment - 1 Foundations

Q-o 1:

1.a) $x_1, ---, x_n$ be real no.

$w_1, ---, w_n$ be strictly positive no. representing importance of each of x_i.

$f(\theta) = \frac{1}{2} \sum_{i=1}^{n} w_i (\theta - x_i)^2$. What value of θ minimizes $f(\theta)$.

Soln:

$$f'(\theta) = \sum_{i=1}^{n} w_i (\theta - x_i) \cdot 1$$

$$\Rightarrow \sum_{i=1}^{n} w_i \theta - \sum_{i=1}^{n} w_i x_i = 0$$

~~then~~ θ 2

$$\Rightarrow \theta \cdot \sum_{i=1}^{n} w_i = \sum_{i=1}^{n} w_i x_i$$

~~$w_i \to 0$~~

~~So $\sum_{i=1}^{n} (\theta - x_i)$~~ $= 0$

~~So minimum will be when $\theta = x_i$~~

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

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

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

Does $f(x) \leq g(x)$, $f(x) = g(x)$ or $f(x) \geq g(x)$?

$\Rightarrow$ Results:

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

$$g(x) = \max \sum_{i=1}^{d} s \cdot x_i \quad \text{where } s \in \{-1, 1\}$$

Now, at every step f(x) can choose s to be either 1 or -1. But g(x) can choose s to be 1 or -1 only once because max is outside the summation. So depending on x_i

Scanned by CamScanner

If $x_i > 0$ always $\Rightarrow f(x) = \beta(x)$ as $\beta(x)$ will choose $s = 1$

If $x_i < 0$ always $\rightarrow f(x) = \beta(x)$ as $\beta(x)$ will choose $s = -1$

If $\exists x_i > 0 \wedge \exists x_i < 0 \Rightarrow$ ~~$\beta(x)$~~ $f(x) \geq \beta(x)$

Therefore $\boxed{f(x) \geq \beta(x)}$

1.c) Suppose you repeatedly roll a fair dice. until 1.

2 comes - a points loose.

6 comes - b points win.

3, 4, 5 — 0 points (no points)

Expected no. of points when you stop. in terms of a & b.

$\Rightarrow$ ~~Probability of 1 = 1/6~~

~~$2 = 1/6$~~

~~$6 = 1/6$~~

~~$\sum_{i=1}^{\infty} \left(\frac{1}{6}\right)^i \left(a \cdot \left(\frac{1}{6}\right) \times \frac{1}{2} + b \cdot \frac{1}{6} \times \frac{1}{2}\right)$~~

Probability of each no. $\{1, 2, 3, 4, 5, 6\} = 1/6$

→ So, if we don't roll 1 then Expected points : $-\frac{a}{6} + \frac{b}{6}$

$$= \frac{b-a}{6}$$

→ roll n - times:

'1' cannot occur in first n-1 rolls. nth one is '1'

$$P(1) = \left(\frac{5}{6}\right)^{n-1} \cdot \frac{1}{6}$$

So Expected score:

$$\cancel{\frac{5}{6}*}\frac{b-a}{6} + \frac{5}{6} \cdot \frac{b-a}{6} + \left(\frac{5}{6}\right)^2 \frac{b-a}{6} + \dots + \frac{5}{6}$$

↑
(n^{th})

1.c Suppose you roll a fair six sided dice until you roll 1. (and then you stop).

Everytime you roll 2 you get lose 'a' points

Everytime you roll 6 you get win~~lose~~ 'b points.

What is the expected no. of points you get when you stop?

Sol^n: First roll: You don't get one then,

$$-\frac{a}{6}+\frac{b}{6} = \frac{b-a}{6} \text{ points you get.}$$

Because P {1,2,3,4,5,6} = 1/6 (any one have 1/6 Prob)

Suppose: Your 2nd roll is one then your points are $\left(-\frac{a}{6}+\frac{b}{6}\right)\cancel{\tfrac{5}{6}}$ as $\frac{1}{6}$ for 2, $\frac{1}{6}$ for 6.

~~You rolled 3rd with~~

Now: If first roll is not one & 2nd roll is one : $\frac{b-a}{6}$

If 3rd roll is one : $\underbrace{\frac{b-a}{6}}_{1st} + \underbrace{\frac{5}{6}\left(\frac{b-a}{6}\right)}_{2nd} + \underbrace{0}_{3rd.}$

n=4 $\left(\frac{5}{6}\right)^{power = \boxed{n-2}}$ If 4th roll is one : $\frac{b-a}{6} + \frac{5}{6}\left(\frac{b-a}{6}\right) + \frac{5}{6}\left(\frac{5}{6}\left(\frac{b-a}{6}\right)\right) + 0$

⋮

For n rolls ;

$$\frac{b-a}{6} + \frac{5}{6}\left(\frac{b-a}{6}\right) + \left(\frac{5}{6}\right)^2\left(\frac{b-a}{6}\right) + \;---\; + \left(\frac{5}{6}\right)^{n-2}\frac{b-a}{6}$$

For $n \to \infty$

$$\left(\frac{b-a}{6}\right)\left[1 + \frac{5}{6} + \left(\frac{5}{6}\right)^2 - - + \cdots \infty\right]$$

$$= \frac{b-a}{6} \cdot \frac{1}{\left(1-\frac{5}{6}\right)} = \frac{b-a}{6} \times 6 = \underline{b-a}$$

d) You are playing a game with fair five sided die.

At each turn:

You have an option to quit & win 15 points.

OR. win 3 points and roll the dice.

If at any point you roll an even number, then the game ends and you leave with your total winnings.

what is the optimal strategy & why?

what would change if the dice was weighted so that the probability of rolling an even number was 0?

e) Suppose the probability of a coin turning up head is $0 < p < 1$, & that we flip it 7 times and get H, H, T, H, T, T, H.

we know that probability of obtaining this sequence is $L(p) = p^4(1-p)^3$. what is value of p that maximises it?

Soln:

$$L(p) = p^4(1-p)^3$$

$$L'(p) = p^4 \; 3(1-p)^2(-1) + (1-p)^3 \cdot 3p^3$$

You would do this. But remember one thing -

If we maximise any ~~log~~ mentonically increasing function of $L(p)$ then it will give same result.

' The value of p that maximises $L(p)$ also maximizes $\log(L(p))$. '

$$\Rightarrow \quad \ln(L(p)) = 4 \ln p + 3 \ln(1-p)$$

$$\ln(L(p))' = \frac{4}{p} + \frac{3}{1-p}(-1) = 0$$

$$\frac{4}{p} = \frac{3}{1-p}$$

$$\Rightarrow \quad \frac{1-p}{p} = \frac{3}{4}$$

$$\Rightarrow \quad \frac{1}{p} - 1 = \frac{3}{4} \quad \Rightarrow \quad \frac{1}{p} = \frac{7}{4}$$

$$\Rightarrow \quad \boxed{p = \frac{4}{7}}$$

Our though will come - it is not a fair coin. But how come on 7 flips it can be fair?

f) for $w \in R^d$ (as a column vector) & constants $a_i, b_j \in R^d$ also represented as column vector. & $\lambda \in R$.

$$f(w) = \sum_{i=1}^{n} \sum_{j=1}^{n} (a_i^T w - b_j^T \cdot w)^2 + \lambda \|w\|_2^2$$

where the vector w is $w = (w_1, ---, w_d)^T$

$$\|w\|_2 = \sqrt{\sum_{k=1}^{d} w_k^2}$$ is l_2 norm.

Compute gradient $\nabla f(w)$

Sol^n: $$\nabla f(w) = \left(\frac{\partial f(w)}{\partial w_1}, ----, \frac{\partial f(w)}{\partial w_d} \right)^T$$

$$\frac{\partial f(w)}{\cancel{\partial w_1}} = \sum_{i=1}^{n} \sum_{j=1}^{n} 2 (a_i^T w - b_j^T w) \cdot (a_{i_1}^T - b_{j_1}^T) + 2\lambda w_1$$

$$\boxed{\text{Note: } \nabla_w \|w\|_2^2 = 2w = \nabla_w w \cdot w}$$

Thus any $k \in \{1, d\}$

$$\frac{\partial f(w)}{\partial w_k} = \sum_{i=1}^{n} \sum_{j=1}^{n} 2 (a_i^T w - b_j^T w) \cdot (a_{ik}^T - b_{jk}^T) + 2\lambda w_k.$$

Note: 1. $\nabla_w a.w = a$

2 $\nabla_w \|w\|_2^2 = 2w$

3. $\nabla_w w^T C w = (C + C^T) w$

$v^2 = v^T v = \|v\|_2^2$

$(A+B)^T = A^T + B^T$

$(AB)^T = B^T A^T$

Complexity

a) Suppose we have an image of human face n×n pixels. In our simplified setting a face consists of two eyes, two ears, one nose, one mouth, each represented as an arbitrary axis aligned rectangle. As we would like to handle picasso portraits too, there is no constraint on location or size of rectangles. How many possible faces (choices of its component rectangles) are there?

Soln: We have 6 rectangles (2eyes, 2ears, one nose, one mouth)

for one rectangle: in an n×n grid

```
for topX = 1 to n:
    for topY = 1 to n:
        for bottomX = 1 to n:
            for bottomY = 1 to n:
                rectangle = (topX, topY, bottomX,
                             bottomY)
```

This is $O(n^4)$

Now for 2 rectangles to generate all possible combinations, we will pass one rectange inside the for loop of other. That will be $O(n^8)$

for 3 rectangles $O(n^{12})$

Thus for n rectangles $O(n^{4n})$ or $O(n^c)$

where $c = 4 \times$ no. of rectangles)

Thus for 6 rectangles $O(n^{24})$

b) Suppose we have an $n \times n$ grid. We start in the upper-left corner (1,1) and would like to reach lower right corner (n,n) by taking single steps down & right.

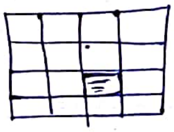
Define a function C_{ij} to be the cost of touching (i,j), & assume it takes constant time to compute.

C_{ij} can be negative.

Give an algorithm for computing the minimum cost in most efficient way. What is runtime (In Big-O)?

Soln: Rule : Only down or Right one step at a time.

Suppose we land up at (x,y) location.



Then

↙ either came from (x-1,y)

↘ OR came from (x,y-1)

Thus, minCost(x,y) = minCost(x-1,y)+minCost(x,y-1) + Cost(x,y)

But we need to handle two cases

↙ If we are going only down

↘ If we are going only right.

Because in these two cases: there will be no left in Case(1) & there will be no left in Case (2).

So,

min(Cost(x,1) = minCost(x-1,1) + Cost(x,1)

minCost(1,y) = minCost(1,y-1) + Cost(1,y)

We can store the minCost into an array of size $n \times n$

Code: // Assume minCost be an array of nxn as zeros.
// populate the cost of (1,1)

```
minCost(1,1) = cost(1,1)

for i=2 to n:
    minCost(i,1) = minCost(i-1,1) + cost(i,1)
    minCost(1,i) = min cost(1,i-1) + cost(1,i)

for i=2to n:
    for j=2 to n:
        minCost(i,j) = ~~minCost~~
                min(minCost(i-1,1) , minCost(i,j-1))
                    + cost(i,j)
```

Thus complexity $O(n^2)$

2.c: Suppose we have a staircase with n steps. (We start on the ground, so we need n total steps to reach the top) We can take as many steps forward at a time, but will never step backwards. How many ways are there to reach the top? Give your answer as a function of n.

Given Case: for n=3 total 4 ways.

	step	step	step
1)	~~step~~ 1,	1	1
2)	2	1	0
3)	1	2	0
4)	3	0	0

Soln: Affectively we are trying to see how to effectively write n as a sum of numbers. Right! Composition of n is a way of writting n as a sequence of strictly positive numbers.

The no. of compositions of n into (k-1)

Therefore for all possible Combinations of steps we have,

$$\sum_{k=1}^{n} \binom{n-1}{k-1} = 2^{n-1}$$

Therefore there are 2^{n-1} ways to reach to the top.

Assuming $\underline{n \geq 1}$

Consider the scalar valued function from 2 (1.f).

$\underline{2.d)}$ Consider the scalar valued function from 2 (1.f). 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 $f(w)$.

Hint: Refractor the algebric Expression. This is classic trick used in machine learning.

From 1.f. 1 $\quad f(w) = \sum_{i=1}^{n} \sum_{j=1}^{n} (a_i^T w - b_j^T w)^2 + \lambda |w|_2^2 \quad (1)$

$a_i, b_j \in R^{d \times 1} \therefore a_i^T, b_j^T \in R^{1 \times d}$

Expand (1): $\quad f(w) = \sum_{i=1}^{n} \sum_{j=1}^{n} \underline{(a_i^T w)^2} + \underline{(b_j^T w)^2} - \underline{2 a_i^T w b_j^T w} + \lambda |w|_2^2 \quad (2)$

$(a_i^T w)^2 = (a_i^T w) \cdot (a_i^T w) = w^T (a_i a_i^T) w$

$(b_j^T w)^2 = (b_j^T w) \cdot (b_j^T w) = w^T (b_j b_j^T) w$

$2 a_i^T w b_j^T w = 2 w^T (a_i b_j^T) w$

We can preprocess ~~$a_i^T w$~~ $a_i \cdot a_i^T$, $b_j b_j^T$ & $a_i b_j^T$. (as they are constants)

This will take $O(d^2)$.

After putting back into (2):

$$\sum_{i=1}^{n} \sum_{j=1}^{n} w^T (a_i a_i^T) w + w^T (b_j b_j^T) w - 2 w^T (a_i b_j^T) w + \lambda |w|_2^2$$

$$= \underbrace{n \cdot \sum_{i=1}^{n} w^T (a_i a_i^T) w}_{O(nd^2)} + \underbrace{n \sum_{j=1}^{n} w^T (a_j a_j^T) w}_{O(nd^2)} - \underbrace{2 \sum_{i=1}^{n} \sum_{j=1}^{n} w^T (a_i b_j^T) w}_{O(n^2 d^2)}$$

$\Downarrow$

We need to improve it.

Transform double summation into two single summations.

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_i b_j^T = \sum_{i=1}^{n} a_i \sum_{j=1}^{n} b_j$$

$\rightarrow$ Now $O(nd^2)$

Lastly: $a_i b_j^T$ is $d \times d$ so $w^T (a_i b_j^T) w$ is also $O(d^2)$

Therefore we have time to compute w as $O(d^2)$.