CS221, Spring 2019, PS1 Foundations

Haiyuan Mei (hmei0411@stanford.edu)

Problem 1: Optimization and probability

1.a

Since $w_1, \ldots, w_n$ are strictly positive, $f(\theta)$ is a convex function which takes minimum value when $f'(\theta) = 0$.
Take the derivative of $f(\theta)$:

$$\begin{aligned} f'(\theta) &= \sum_{i=1}^{n} w_i(\theta - x_i) \\ &= \sum_{i=1}^{n} w_i\theta - \sum_{i=1}^{n} w_i x_i \end{aligned}$$

Set the above to be 0 and solve the equation:

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

If some of the w_i are negative, but $\sum_{i=1}^{n} w_i$ is positive, the result is still the same.
If some of the w_i are negative, and $\sum_{i=1}^{n} w_i$ is negative too, $f(\theta)$ will become concave and has no minumum.

1.b

Answer should be: $f(\mathbf{x}) \geq g(\mathbf{x})$
Proof: For $g(\mathbf{x}) = \max_{s\in\{1,-1\}} \sum_{i=1}^{d} sx_i$, depending on the sign of $\sum_{i=1}^{d} sx_i$, $g(\mathbf{x})$ takes the maximum on either $s = 1$ or $s = -1$. Suppose it takes the maximum when $s = 1$, we have:

$$\begin{aligned} g(\mathbf{x}) &= \max_{s\in\{1,-1\}} \sum_{i=1}^{d} sx_i \\ &= \sum_{i=1}^{d} x_i \\ &\leq \sum_{i=1}^{d} \max_{s\in\{1,-1\}} sx_i \\ &= f(\mathbf{x}) \end{aligned}$$

When the maximum is with $s = -1$, simple change the above equation to be:

$$\begin{aligned} g(\mathbf{x}) &= \max_{s\in\{1,-1\}} \sum_{i=1}^{d} sx_i \\ &= \sum_{i=1}^{d} -x_i \\ &\leq \sum_{i=1}^{d} \max_{s\in\{1,-1\}} sx_i \\ &= f(\mathbf{x}) \end{aligned}$$

Question proved.

1.c

Solution 1: Suppose the expected number of points is x, the following equation is true:

$$x = \frac{5}{6}x + \frac{1}{6}(b-a)$$

The above equation says that the expected number of points x is equal to half of x which comes from 3,4 or 5, and $\frac{1}{6}(b-a)$ which is from 2 or 6; The probability of taking each of the value is $\frac{1}{6}$. The solution of above gives $x = b - a$.

Solution 2: At each time the expected number of points should be $\frac{1}{6}(b-a)$, and the probability of moving to next time is $\frac{5}{6}$. So the expected number of points can be written as:

$$x = \sum_{i=0}^{\infty} \frac{5}{6}^{i} \frac{b-a}{6} = b - a$$

1.d

If stop at nth turn, the expected point is: $E_n = \sum_{i=1}^{n-1} \frac{3}{5}^i 3 + \frac{3}{5}^n 15$;
when n = 1, $E_1 = 9$;
when n = 2, $E_2 = 7.2$;
when n = 3, $E_3 = 6.12$;
...
when n = ∞, $E_\infty = 4.5$;
Which means the earlier we stop the game the higher the expected points. So the strategy should be stop at begining.

If the probability of rolling an even number is 0, then the strategy should be never stop.

1.e

In order to maximize the $L(p)$, consider the derivative w.r.t. $\log L(p)$:

$$\log L(p) = 4 \log p + 3 \log(1-p)$$

take the derivative with respect to p and set it to 0 we have:

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

the solution should be $p = \frac{4}{7}$.
The intuitive interpretation of value p is that the expected probability of getting a head when flipping this coin is 4/7 if the result array is {H,H,T,H,T,T,H}

1.f

The original function can be written with as:

$$f(w) = \sum_{i=0}^{n} \sum_{j=0}^{n} ((a_i^T - b_j^T)w)^2 + \lambda w^T w$$

Apply $\nabla trAB = B^T$, it's easy to see that $\nabla w^T w = 2w$;
also consider $(a_i^T - b_j^T)w = tr(a_i^T - b_j^T)w = tr(w(a_i^T - b_j^T))$, it's gradient is $(a_i - b_j)$;
so that $\nabla((a_i^T - b_j^T)w)^2 = 2(a_i^T - b_j^T)w \nabla tr((a_i^T - b_j^T)w) = 2(a_i^T - b_j^T)w(a_i - b_j)$
put things together we have:

$$\nabla f(w) = \sum_{i=0}^{n} \sum_{j=0}^{n} 2(a_i^T - b_j^T)w(a_i - b_j) + 2\lambda w$$

Problem 2: Complexity

2.a

Suppose each component resides in an rectangle starting from (x_1, y_1) to (x_2, y_2). Since x and y are ranging from 1 to n, the starting point has n^2 possibilities, the ending point also has n^2 possibilities, so each component has n^4 possibilities. There are 5 components, so there are $O(n^{20})$ possibile faces.

2. b

The problem can be solved by dynamic programming.
Suppose the minimum cost of reaching point (x, y) is denoted as $f(x, y)$ where both $x, y \in [1, n]$; The following are true:

$$
\begin{aligned}
f(x, y) &= \min(f(x-1, y), f(x, y-1)) + c(x, y) \\
f(1, y) &= f(1, y-1) + c(1, y) \\
f(x, 1) &= f(x-1, 1) + c(x, 1) \\
f(1, 1) &= c(1, 1)
\end{aligned}
$$

Using dynamic programming, we need to compute for each grid the minimum cost, which is just $O(n^2)$ run time complexity.
Apart from that, we also need $O(n^2)$ storage complexity for all the intermediate costs.

2.c

It is obvious to see that the problem can be written as: $f(n) = f(n-1) + (n-1)$
This gives the following:

$$
\begin{aligned}
f(2) - f(1) &= 1 \\
f(3) - f(2) &= 2 \\
f(4) - f(3) &= 3 \\
&\dots \\
f(n) - f(n-1) &= (n-1)
\end{aligned}
$$

Sum the above we have: $f(n) - f(1) = 1 + 2 + \ldots + (n-1)$, so:

$$
f(n) = \frac{n(n-1)}{2} + 1
$$

There are $\frac{n(n-1)}{2} + 1$ ways to reach the top.

2.d

Rewrite $f(w)$ and use the fact that $trABC = trCAB = trBCA$, we have:

$$\begin{aligned} f(w) &= \sum_{i=0}^{n}\sum_{j=0}^{n}(a_i^T - b_j^T)ww^T(a_i - b_j) + \lambda w^T w \\ &= \sum_{i=0}^{n}\sum_{j=0}^{n} w^T(a_i - b_j)(a_i^T - b_j^T)w + \lambda w^T w \\ &= w^T\left(\sum_{i=0}^{n}\sum_{j=0}^{n}(a_i - b_j)(a_i^T - b_j^T)\right)w + \lambda w^T w \end{aligned}$$

We can preprocess the double summation part denoted by M, which is just an $n \times n$ matrix:

$$\begin{aligned} M &= \sum_{i=0}^{n}\sum_{j=0}^{n}(a_i - b_j)(a_i^T - b_j^T) \\ &= \sum_{i=0}^{n}\sum_{j=0}^{n}(a_i a_i^T + b_j b_j^T - a_i b_j^T - b_j a_i^T) \\ &= n\sum_{i=0}^{n} a_i a_i^T + n\sum_{j=0} b_j b_j^T - AB^T - (AB^T)^T \end{aligned}$$

- The part $\sum_{i=0}^{n} a_i a_i^T$ and $\sum_{i=0}^{n} a_i a_i^T$ both takes d^2 steps(multiplication) to compute.
- For AB^T part ($A = [a_1, \ldots, a_n], B = [b_1, \ldots, b_n]$), they both are $n \times d$ matrix, so AB^T ends up with a $d \times d$ matrix, each element is a sum of n values comes from n multiplications. So the total steps of calculation AB^T is nd^2.
- Summarize the above two we can conclude that the preprocess takes complexity of $O(nd^2)$.
- After preprocessing, for any given w, $w^T Mw$ takes d^2 multiplications, and λww^T takes d multiplications, which concludes that $f(w)$ has complexity of $O(d^2)$.