CS221 Fall 2018 Homework 1

SUNet ID: 05794739
Name: Luis Perez
Collaborators:

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

Problem 1

(a) The problem asks us to find the value of θ that minimizes the function:

$$f(\theta) = \frac{1}{2}\sum_{i=1}^{n} w_i(\theta - x_i)^2$$

We can compute the above by simply taking the derivate of the function with respect to θ and solving for θ when equal to 0.

$$\begin{aligned} \frac{df}{d\theta} &= \sum_{i=1}^{n} w_i(\theta - x_i) && \text{(using product rule } (uv)' = u'v + uv'\text{)} \\ &= 0 \implies \theta = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i} \end{aligned}$$

We note that $\frac{d^2 f}{d\theta^2} = \sum_{i=1}^{n} w_i$, so the above solution is a minimum if and only if $\sum_{i=1} w_i > 0$. This is guaranteed if $w_i > 0, \forall i$. However, if we have some $w_i < 0$, then we could run into an issue where the function does not have a minimum.

(b) We can show what we want rather directly. In particular, we will show that $f(\boldsymbol{x}) \geq g(\boldsymbol{x})$.

$$\begin{aligned} f(\boldsymbol{x}) &= \sum_{i=1}^{d} \max_{s \in \{1,-1\}} s x_i && \text{(definition of } f\text{)} \\ &= \max_{\boldsymbol{s} \in \{1,-1\}^d} \boldsymbol{s}^T \boldsymbol{x} && \text{(definition of dot product)} \\ &\geq \max_{\boldsymbol{s} \in \{[1]^d, [-1]^d\}} \boldsymbol{s}^T \boldsymbol{x} && \text{(}\boldsymbol{s}\text{ can take on fewer values now)} \\ &= \max_{s \in \{1,-1\}} \sum_{i=1}^{d} s x_i && \text{(definition of dot product)} \\ &= g(\boldsymbol{x}) \end{aligned}$$

(c) TODO

(d) As the hint indicates, we can calculate the value of p that maximizes $L(p)$ by taking the derivate of $\log L(p)$ and setting it to zero (further nothing that $L(p)$ is concave, and therefore the critical point is a maximum).

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

The intuitive interpretation of this value of p is that this is the probability of the coin which we used of landing heads. Given the data, it appears to be slightly biased in favor of heads.

(e) We're given the function $f(\boldsymbol{w}) = \sum_{i=1}^{n}\sum_{j=1}^{n}(\boldsymbol{a}_i^T\boldsymbol{w} - \boldsymbol{b}_j^T\boldsymbol{w})^2 + \lambda||w||$ and we wish to compute $\nabla f(\boldsymbol{w})$. We can do it by parts, first.

$$\begin{aligned}\frac{\partial}{\partial w_k}||\boldsymbol{w}||_2^2 &= \frac{\partial}{\partial w_k}\sum_{k=1}^{d} w_k^2\\ &= 2w_k && \text{(all terms except } w_k \text{ are zero)}\end{aligned}$$

We therefore have $\nabla\lambda||\boldsymbol{w}||_2^2 = 2\lambda\boldsymbol{w}$. Continuing:

$$\begin{aligned}\frac{\partial}{\partial w_k}\left[\sum_{i=1}^{n}\sum_{j=1}^{n}(\boldsymbol{a}_i^T\boldsymbol{w} - \boldsymbol{b}_j^T\boldsymbol{w})^2\right] &= \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\partial}{\partial w_k}\left[\boldsymbol{a}_i^T\boldsymbol{w} - \boldsymbol{b}_j^T\boldsymbol{w}\right]^2\\ &= 2\sum_{i=1}^{n}\sum_{j=1}^{n}(\boldsymbol{a}_i^T\boldsymbol{w} - \boldsymbol{b}_j^T\boldsymbol{w})\frac{\partial}{\partial w_k}[\boldsymbol{a}_i^T\boldsymbol{w} - \boldsymbol{b}_j^T\boldsymbol{w}]\\ & && \text{(chain rule)}\\ &= 2\sum_{i=1}^{n}\sum_{j=1}^{n}(\boldsymbol{a}_i^T\boldsymbol{w} - \boldsymbol{b}_j^T\boldsymbol{w})[\boldsymbol{a}_i - \boldsymbol{b}_j]_k\end{aligned}$$

We therefore have the result that:

$$\nabla f(\boldsymbol{w}) = 2\sum_{i=1}^{n}\sum_{j=1}^{n}(\boldsymbol{a}_i^T\boldsymbol{w} - \boldsymbol{b}_j^T w)[\boldsymbol{a}_i - \boldsymbol{b}_i] + 2\lambda\boldsymbol{w}$$

Problem 2

(a) We first note that there are a total of $\binom{n^2}{2} = O(n^4)$ locations to place a rectangle in an $n \times n$ grid (think about it as if you're choosing the top-left and bottom-right corners, which will fully determine an axis-aligned rectangle). To fully determine a face, we need to place 2 (eyes) + 2 (ears) + 1 (nose) + 1 (mouth) = 6 rectangles total. To simplify things (this won't affect the final answer), we assume that the same rectangle can use used for multiple face parts. We therefore have a total of $O(n^4) \times O(n^4) \times O(n^4) \times O(n^4) \times O(n^4) \times O(n^4) = O(n^24)$ choices of 6 rectangles each choise defining one face for a total of $O(n^24)$ possible faces.[1]

(b) This is essentially a DP problem. First, define $T(i, j)$ to be the minimum cost for reaching the lower-right corner (n, n) given that we're currently at position (i, j) for $1 \leq i, j \leq n$. We can define this recursively as follows:

$$\begin{aligned}
T(n,n) &= c(n,n) \\
T(i,n) &= c(i,n) + T(i+1,n) \\
T(n,j) &= c(n,j) + T(n,j+1) \\
T(i,j) &= c(i,j) + \min\{T(i,j+1), T(i+1,j)\}
\end{aligned}$$

To answer the question asked in the problem statement, we would simply need to return $T(1, 1)$. We can memoize intermediate results and thereby achieve a running time of $O(n^2)$ – n^2 possible inputs, each taking contant time to compute – with space complexity of $O(n^2)$. We can further minimize the space-complexity to be O(n) if we do an iterative matrix-filling approach in an intelligent way, but this is not required for this problem.

(c) We can compute this rather directly by thinking of the problem slightly differently. Basically, there are n steps. Each "way" to reach the top can be encoded uniquely by a binary string of length $n - 1$ indicating whether or not we step on the i-th step or not on our way to the n-th step.

As such, the total number of ways is simply $2^{\max\{0,n-1\}}$, where we take $\max\{0, n - 1\}$ to handle the $n = 0$ case nicely.

[1] Technically, we're double counting the eyes and the ears here – however, note that this will add at most a constant factor of 4, so it does not affect the big-O solution.