ECE 473 Homework 1

Grant Julian

January 26, 2020

Contents

1 Optimization and Probability

1.1 problem a

(a) The process of minimizing the function $f(\theta)$ is simply achieved by minimizing the 'distances' from θ to all x_i. This function would be minimized by choosing a θ that is most near the mean of all x after being weighted by w_i. To find this, you can simply set the derivative of $f(\theta)$ to be equal to zero, and solve for θ. Doing so provides the following formula:

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

If some of the ω_i were negative, then they would incorrectly be causing a minimization of $f(\theta)$, i.e. an unfit equation might be considered fit.

1.2 problem b

(b) Given that

$$f(x) = \sum_{i=1}^{d} \max_{s\in\{1,-1\}} s x_i \tag{2}$$

and

$$g(x) = \max_{s\in\{1,-1\}} \sum_{i=1}^{d} s x_i \tag{3}$$

where $x = (x_1, ..., x_d) \in R^d$ is a real vector, it holds for all values of x that $f(x) \geq g(x)$. In the case of $f(x)$, an s value is chosen such that each individual x_i is made to be positive (i.e. maximized). For example if x = 5 then s = 1 or if x = -5 then s = -1. In simple terms, $f(x)$ is the summation of the magnitudes of all x. In contrast $g(x)$, is the magnitude of the summation of all x. This difference is significant because any summation of positive and negative numbers will effectively reduce the magnitude. This means that if x is a mix of negative and positive numbers, $g(x)$ will be smaller in magnitude than $f(x)$. However if x is purely negative or purely positive, then $g(x)$ will be equal to $f(x)$.

1.3 problem c

(c) To find how many points we would end up with at the end of the game, let us first consider the probabilities of each event happening. There is a 1/6 chance that we roll a 1 (-a), a 2 (+b), and a 3 (end of game). There is also a 1/2 chance that nothing happens and that we should just roll again- meaning that 5/6 roles we will continue the game. Knowing this we can write out the equation for the expected amount of points P:

$$p_1 = 1/6 * (-a) + 1/6 * b + 5/6(p_2) \tag{4}$$

In this equation, I recursively define the expected number of points p_1 to be a function of the next role p_2. After performing this computation, it can be seen that expected number of points approaches $b - a$.

1.4 problem d

(d) Suppose the probability of a coin turning up heads is $0 < p < 1$ and that we flip it 7 times and get {H,H,T,H,T,T,H}. The probability of getting this sequence is:

$$L(p) = pp(1-p)p(1-p)(1-p)p = p^4(1-p)^3 \tag{5}$$

We can find the value p that maximizes $L(p)$ by first taking the natural log of $L(p)$.

$$\ln L(p) = \ln(p^4(1-p)^3) \tag{6}$$

and then take the derivative

$$\frac{d}{dp} \ln L(p) = \frac{d}{dp} \ln(p^4(1-p)^3) \tag{7}$$

using the chain rule to reduce and setting the derivative equal to 0 we get that

$$0 = \frac{4}{p} - \frac{3}{1-p} \tag{8}$$

finally solving for p, we get that the probability of getting the sequence is maximized when p = 4/7.

1.5 problem e

2 Problem 2: Complexity

2.1 problem a

Given a the task to draw 6 unique rectangular features (two eyes, two ears, one nose, one mouth) onto an image of size n x n the complexity would be as follows. First, to find every configuration of one element onto the n x n image, we would need a complexity of $O(n^4)$. This is because it would take $O(n^2)$ to go through every single 2-dimensional position and then another $O(n^2)$ to go through every single 2-dimensional size. Since the second process would be nested within the first, the complexity to draw one element is $O(n^4)$. Since we need to draw 6 features, and every feature is dependent on the placement and size of the previous features, we can find that the total comlexity is $O((n^4)^6)$ or $O(n^{24})$.

2.2 problem b

Suppose we have an n x n grid and we would like to create a function $v(i, j)$ that can calculate the minimum cost of reaching (i, j) from $(1, 1)$. We can define the function $v(i, j)$ recursively as

$$v(i, j) = min(v(i-1, j), v(i, j-1)) + c(i, j) \tag{9}$$

Without any optimization, this process would have an exponential run-time complexity. However, if we were to implement memoization we would be able to reach $O(n^2)$. This is because we would keep track of every tile we visit and thus would eliminate any repetitive calculations.

2.3 problem c

Suppose we had a staircase with n steps and we can ascend it by taking as many steps at one time as we would like. We define the function for the total number of possible ways to reach the top as

$$s(n) = 2^{n-1} \tag{10}$$

Simply put, this problem can be represented by a bit vector, where each bit represents either taking that step (1) or stepping over it (0). For example '1001' would represent stepping up the first step and then skipping the next two to reach the top step. I write it as 2^{n-1} because the last step must always be taken (1).

2.4 problem d