CS215 Assignment 1 Report

Diwan Anuj Jitendra - 170070005
Soumya Chatterjee - 170070010

August 8, 2018

1 Problem 1

The definition of standard deviation, σ, is as follows:

$$\sigma^2 = \frac{\sum_{j=1}^{n}(x_j - \mu)^2}{n-1}$$

Let us take an i from $\{1, 2, \ldots, n\}$. Now consider that $\forall\ j \in \{1, 2, \ldots, n\}$, $(x_j - \mu)^2 \geq 0$. In particular, we have:

$$\forall\ j \in \{1, 2, \ldots, n\}, j \neq i, (x_j - \mu)^2 \geq 0$$

and

$$(x_i - \mu)^2 = (x_i - \mu)^2$$

Adding both, and recognising that the LHS is a sum over $(x_j - \mu)^2$ from $j = 1$ to n, we have:

$$\sum_{j=1}^{n}(x_j - \mu)^2 \geq (x_i - \mu)^2$$

Multiplying and dividing by $n - 1$ on LHS(since $n - 1 \geq 1$, this is valid):

$$(n-1) \times \frac{\sum_{j=1}^{n}(x_j - \mu)^2}{n-1} \geq (x_i - \mu)^2$$

This gives:

$$(n-1) \times \sigma^2 \geq (x_i - \mu)^2$$

Since both sides are non-negative, we can take square root, giving:

$$\sqrt{n-1} \cdot \sigma \geq |x_i - \mu|$$

i.e.

$$|x_i - \mu| \leq \sqrt{n-1} \cdot \sigma$$

Doing this for each i in $\{1, 2, \ldots, n\}$, we are done.

2 Problem 2

We have:

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

$$\sigma^2 = \frac{\sum_{j=1}^{n}(x_j - \mu)^2}{n-1}$$

And that the sum of absolute deviations from a value q,

$$\sum_{i=1}^{n} |x_i - q|$$

is minimized when $q = \tau$, the median. Now, let us prove: $|\mu - \tau| \leq \sigma$.

$$\begin{aligned}
|\mu - \tau| &= \left| \frac{\sum_{i=1}^{n} x_i}{n} - \tau \right| \\
&= \left| \frac{\sum_{i=1}^{n} x_i - n\tau}{n} \right| \\
&= \frac{1}{n} \times \left| \sum_{i=1}^{n} (x_i - \tau) \right| \\
&\leq \frac{1}{n} \times \sum_{i=1}^{n} |x_i - \tau| \quad \text{(Using triangle inequality)} \\
&\leq \frac{1}{n} \times \sum_{i=1}^{n} |x_i - \mu| \quad \text{(Using the fact that the sum of absolute deviations is minimized for } \tau\text{)} \\
&\leq \sqrt{\frac{1}{n} \times \sum_{i=1}^{n} |x_i - \mu|^2} \quad \text{(Using AM-QM inequality and observing that absolute values are non-negative)} \\
&\leq \sqrt{\frac{1}{n-1} \times \sum_{i=1}^{n} |x_i - \mu|^2} \quad \text{(Since } n > n-1 \text{ and the summation is a non-negative number)} \\
&= \sigma
\end{aligned}$$

Thus we are done.

3 Problem 3

3.1 Subproblem (a)

Observe that the event that the car is behind door i i.e. C_i is independent of the choice that the contestant makes i.e Z_1. Thus,

$$P(C_i|Z_1) = P(C_i) = \frac{1}{3} \quad \forall\ i \in \{1, 2, 3\}$$

3.2 Subproblem (b)

Let us find it for each i. We know that the host must choose a door behind which there is a stone and is not chosen by the contestant.

1. $P(H_3|C_1, Z_1)$:
   The host can open either door 2 or 3 since both doors 2 and 3 have a stone behind them and door 1 is chosen by the contestant. Since the host can choose either of the 2 doors with equal probability,

   $$P(H_3|C_1, Z_1) = \frac{1}{2}$$

2. $P(H_3|C_2, Z_1)$:
   The host can only open door 3 as door 1 is chosen by the contestant and door 2 has the car behind it. Thus,

   $$P(H_3|C_2, Z_1) = 1$$

3. $P(H_3|C_3, Z_1)$:
   The host can only open door 2 as door 1 is chosen by the contestant and door 3 has the car behind it. Thus,
$$P(H_3|C_3, Z_1) = 0$$

3.3 Subproblem (c)

We know
$$P(C_2|H_3 \cap Z_1) = \frac{P(H_3|C_2 \cap Z_1) \times P(C_2 \cap Z_1)}{P(H_3 \cap Z_1)}$$
$$= \frac{P(H_3|C_2 \cap Z_1) \times P(C_2|Z_1) \times P(Z_1)}{P(H_3|Z_1) \times P(Z_1)} = \frac{P(H_3|C_2 \cap Z_1) \times P(C_2|Z_1)}{P(H_3|Z_1)}$$

From subproblem (b):
$$P(H_3|C_2 \cap Z_1) = 1$$

From subproblem (a):
$$P(C_2|Z_1) = \frac{1}{3}$$

By marginalizing $H_3|Z_1$ by partitioning it by intersection with the three disjoint events C_1, C_2, C_3 and from subproblem (b),
$$P(H_3|Z_1) = \sum_{i=1}^{3} P(H_3|Z_1 \cap C_i)P(C_i) = (\frac{1}{2} + 1 + 0)\frac{1}{3} = \frac{1}{2}$$

Thus, we get:
$$P(C_2|H_3 \cap Z_1) = \frac{1 \times \frac{1}{3}}{\frac{1}{2}} = \frac{2}{3}$$

3.4 Subproblem (d)

We know
$$P(C_1|H_3 \cap Z_1) = \frac{P(H_3|C_1 \cap Z_1) \times P(C_1 \cap Z_1)}{P(H_3 \cap Z_1)}$$
$$= \frac{P(H_3|C_1 \cap Z_1) \times P(C_1|Z_1) \times P(Z_1)}{P(H_3|Z_1) \times P(Z_1)} = \frac{P(H_3|C_1 \cap Z_1) \times P(C_1|Z_1)}{P(H_3|Z_1)}$$

From subproblem (b):
$$P(H_3|C_1 \cap Z_1) = \frac{1}{2}$$

From subproblem (a):
$$P(C_1|Z_1) = \frac{1}{3}$$

By marginalizing $H_3|Z_1$ by partitioning it by intersection with the three disjoint events C_1, C_2, C_3 and from subproblem (b),
$$P(H_3|Z_1) = \sum_{i=1}^{3} P(H_3|Z_1 \cap C_i)P(C_i) = (\frac{1}{2} + 1 + 0)\frac{1}{3} = \frac{1}{2}$$

Thus, we get:
$$P(C_1|H_3 \cap Z_1) = \frac{\frac{1}{2} \times \frac{1}{3}}{\frac{1}{2}} = \frac{1}{3}$$

3.5 Subproblem (e)

Thus, the probability of winning if the contestant switches is $\frac{2}{3}$ and the probability of winning if the contestant doesn't switch is $\frac{1}{3}$. Hence it is more beneficial to switch.

4 Problem 4

4.1 How to run

1. Run `Problem4.m` on MATLAB. The command prompt area will print the relative mean squared errors corresponding to $f = 30$ % and $f = 60$ %.
2. Two figures will appear. These are the required plots.

4.2 Relative mean squared errors:

1. Relative mean squared error for f=30 %:
   (a) Moving median filtering:7.485420e+01
   (b) Moving average filtering:1.039800e+02
   (c) Moving quartile filtering:1.550093e-02
2. Relative mean squared error for f=60 %:
   (a) Moving median filtering:6.729354e+02
   (b) Moving average filtering:3.800406e+02
   (c) Moving quartile filtering:1.349047e+02

4.3 Explanation

Clearly, the 25^{th} percentile gives the least relative mean squared error and hence it is the best choice amongst the three to use for filtering the corrupted signal. The explanation for this is as follows:

1. The values of y in the original, clean sine wave vary from -5 to 5. We introduced corruption in this wave of magnitude 100 to 120, which is nearly 20 times larger than the original values. As a result, the corrupted sine wave has either extremely high values from 95 to 125 or extremely low values from -5 to 5.
2. The goal of the filtering method is to remove this corruption. Since the corruption introduced a large increase in value of y, the filtering method will work better if smaller values of y are given more importance. In particular, considering the set of values of y for the $[-8 : 8]$ neighbourhood of each x, the smaller values in this set will be closer to the correct values. This notion is captured best by setting the expected correct value $\hat{y}$ to the 25^{th} percentile. This choice will, with a higher probability, give a value near -5 to 5, as required. In contrast, the mean will be highly skewed by the higher values and will almost always give bad results, and the median being the 50^{th} percentile will give the middle value which is worse than the 25^{th} percentile.

4.4 Plots

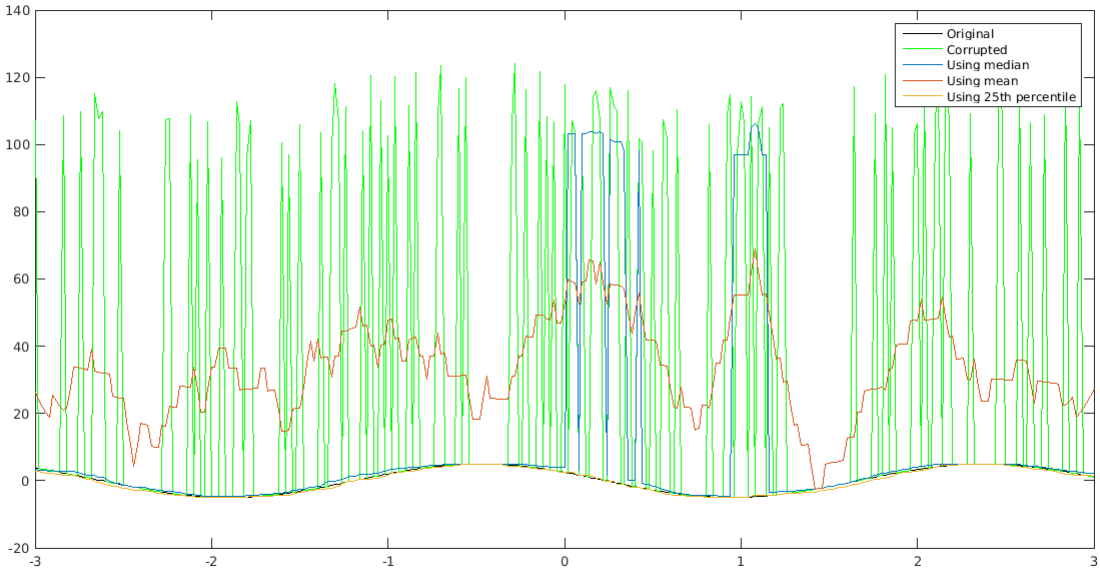

Figure 1: For $f = 30\%$

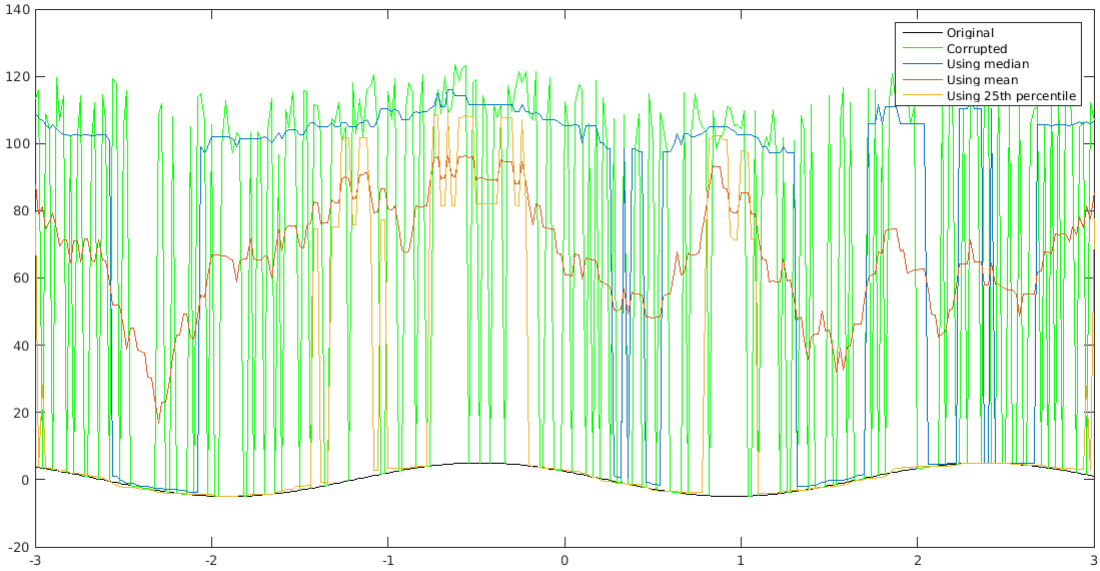

Figure 2: For $f = 60\%$

5 Problem 5

5.1 How to run

1. There are three `.m` files, namely `UpdateMean.m`, `UpdateMedian.m`, `UpdateStd.m`. Each contains the required functions, and can be used in the format specified in the question.

5.2 Formulae and their Derivation

1. `UpdateMean` :
   Code: `newMean=(OldMean*n+NewDataValue)/(n+1)`
   Formula:
   $$\texttt{newMean} = \frac{n \times \texttt{oldMean} + \texttt{NewDataValue}}{n+1}$$
   Derivation: Let A be the original array containing n distinct numbers $\{a_i\}_{i=1}^{n}$ and let a_{n+1} be the new data value. Then:
   $$\begin{aligned}
   \texttt{newMean} &= \frac{\sum_{i=1}^{n+1} a_i}{n+1} \\
   &= \frac{\sum_{i=1}^{n} a_i + a_{n+1}}{n+1} \\
   &= \frac{n \times \texttt{oldMean} + a_{n+1}}{n+1} \quad (\text{Using } \texttt{oldMean} = \frac{\sum_{i=1}^{n} a_i}{n}\ ) \\
   &= \frac{n \times \texttt{oldMean} + \texttt{NewDataValue}}{n+1}
   \end{aligned}$$
   And thus we are done.

2. `UpdateMedian` :
   Let A be the original array containing n distinct numbers in ascending order, $\{a_i\}_{i=1}^{n}$, and let x be the new data value. We must take care of several cases here. If $n = 1$, then `newMedian` is simply $\frac{a_1+x}{2}$. Otherwise:

   (a) Suppose n is odd. Then it has one median, $a_{(n+1)/2}$, and after adding x, we must take the mean of the middle two values. After adding x, if x is between $a_{(n+1)/2}$ and $a_{[(n+1)/2]+1}$, then the new median is the mean of $a_{(n+1)/2}$ and x, since these two are the middle values in the new array. If x is greater than $a_{[(n+1)/2]+1}$, then the new median is the mean of $a_{(n+1)/2}$ and $a_{[(n+1)/2]+1}$. Similarly, if x is between $a_{(n+1)/2}$ and $a_{[(n+1)/2]-1}$, then the new median is the mean of $a_{(n+1)/2}$ and x, since these two are the middle values in the new array. If x is lesser than $a_{[(n+1)/2]-1}$, then the new median is the mean of $a_{(n+1)/2}$ and $a_{[(n+1)/2]-1}$.

   (b) Now, suppose n is even. Then after adding x, we will have one unique median, namely the middle value. $a_{n/2}$ and $a_{(n/2)+1}$ are the two middle values. Let there be k values lesser than $a_{n/2}$ and k values greater than $a_{(n/2)+1}$. If x is between $a_{n/2}$ and $a_{(n/2)+1}$, then x is the new median since there are $k+1$ number of values on either side. If x is greater than $a_{(n/2)+1}$, then $a_{(n/2)+1}$ is the new median using a similar argument. Similarly, if x is lesser than $a_{n/2}$ then $a_{n/2}$ is the new median.

3. `UpdateStd`
   Let A be the original array containing n distinct numbers in ascending order, $\{a_i\}_{i=1}^{n}$, and let x be the new data value. Let σ_2 be the new standard deviation, σ_1 be the old standard deviation, μ_2 be the new mean, μ_1 be the old mean. Formula:
   $$\sigma_2 = \sqrt{\frac{{\sigma_1}^2(n-1) + n(\frac{x-\mu_1}{n+1})^2 + (x-\mu_2)^2}{n}}$$
   Derivation: We know:
   $$\mu_2 = \frac{n\mu_1 + x}{n+1}$$
   $${\sigma_1}^2 = \frac{\sum_{i=1}^{n} (a_i - \mu_1)^2}{n-1}$$

$$\sigma_2{}^2 = \frac{\sum_{i=1}^{n}(a_i - \mu_2)^2 + (x - \mu_2)^2}{n}$$

Now,

$$\begin{aligned}
\sum_{i=1}^{n}(a_i - \mu_2)^2 &= \sum_{i=1}^{n}\left(a_i - \frac{n\mu_1 + x}{n+1}\right)^2 \\
&= \sum_{i=1}^{n}\left(a_i - \frac{(n+1)\mu_1 + x - \mu_1}{n+1}\right)^2 \\
&= \sum_{i=1}^{n}\left((a_i - \mu_1) - \frac{x - \mu_1}{n+1}\right)^2 \\
&= \sum_{i=1}^{n}(a_i - \mu_1)^2 - 2\left(\frac{x - \mu_1}{n+1}\right)\sum_{i=1}^{n}(a_i - \mu_1) + \sum_{i=1}^{n}\left(\frac{x - \mu_1}{n+1}\right)^2 \\
&= \sigma_1{}^2(n-1) - 2\left(\frac{x - \mu_1}{n+1}\right)(n\mu_1 - n\mu_1) + n\left(\frac{x - \mu_1}{n+1}\right)^2 \\
&= \sigma_1{}^2(n-1) + n\left(\frac{x - \mu_1}{n+1}\right)^2
\end{aligned}$$

Thus,

$$\sigma_2{}^2 = \frac{\sigma_1{}^2(n-1) + n(\frac{x-\mu_1}{n+1})^2 + (x - \mu_2)^2}{n}$$

Giving

$$\sigma_2 = \sqrt{\frac{\sigma_1{}^2(n-1) + n(\frac{x-\mu_1}{n+1})^2 + (x - \mu_2)^2}{n}}$$

4. Updating histogram:
   When a new value is received to be added to A, to update the histogram, one must find out which bin the value must be put into. We traverse through all the bins and determine which bin to put the value into. We then increase the occupancy of the determined bin by 1 and update the histogram by raising the y-value by 1. If no such bin can be found(typically when the value is greater than the maximum right bin bound of all the bins or smaller than the minimum left bin bound of all the bins), we create a new bin such that it contains the new value(e.g. suppose the new value is greater than the maximum right bin bound r_{max}, create a new bin $(r_{max}, \texttt{newValue}]$ and set its size to 1, creating a new entry on the histogram with y-value 1.
   If the new value is much larger or much smaller than typical values in A, it might be a good idea to change the bounds of the bins themselves, and resize the bins, to better capture the properties of the array A.

6 Problem 6

6.1 How to run

Run `Problem6.m` on MATLAB. It will generate the required plot and output the minimum number of people n such that the probability is at least p % for each p as: `At least p%:  n`

6.2 Formula and its derivation

We assume that there are 365 days in a year. The code has a variable called `daysInYear` set to 365 that can be changed manually to change the number of days in a year. In a group of n people, the probability

that some two people have the same birthday is:

$$1 \text{ if } n \geq 366,$$

and

$$1 - \frac{(366-1) \times (366-2) \times \cdots \times (366-n)}{365^n}$$

otherwise.

This formula has been implement iteratively in MATLAB, as one can easily obtain $P(n+1)$ from $P(n)$ as

$$P(n+1) = 1 - (1 - P(n)) \times \frac{366 - (n+1)}{365}$$

And by checking these values against the required values of P, we get the minimum number of people.

Derivation is as follows:

For $n >= 366$, we use the pigeonhole principle. Imagine each of the 365 days as bins and put each person into the bin corresponding to their birthday. Since there are strictly more people than bins($n \geq 366 > 365$), some bin necessarily has two or more people. Thus, some two people have the same birthday and hence this event's probability is 1.

For $n < 366$: Consider the sample space consisting of pairs of people and their birthdays. Every outcome in this sample space is equally likely as any person can have his/her birthday on any day with equal probability. The total number of outcomes is the total number of possibilities for the birthdays of all n people. The first person has 365 choices for the birthday, the second has 365 choices, and so on for each person. Thus the total number of outcomes is 365^n.

Now the total number of outcomes such that no two people have the same birthday is as follows: The first person has 365 choices for his/her birthday, the second person has 364 choices for his/her birthday(all days except the first person's birthday), and so on. The n^{th} person has $366-n$ choices for his/her birthday. Thus the number of outcomes is $365 \times 364 \times \cdots \times (366-n)$. Thus, the probability that no two people have the same birthday is:

$$\frac{(366-1) \times (366-2) \times \cdots \times (366-n)}{365^n}$$

and hence the probability that some two people have the same birthday is:

$$1 - \frac{(366-1) \times (366-2) \times \cdots \times (366-n)}{365^n}$$

6.3 Values

Probability in %	Minimum number of people
5	7
10	10
15	12
20	14
30	17
40	20
50	23
60	27
70	30
80	35
90	41
95	47
99	57
99.99	80
99.9999	97
100	366

6.4 Plot