CS394S Assignment 2

Siyuan Chen 174817

Note: I select problem 1,2,4,5 to finish this assignment

1. Name and Shame Mechanism

Question 1:

For given Dataset $D: \{x_1, x_2 \dots x_n\}$ $\quad Y_i = \begin{cases} (i, x_i) & \text{wp. } \delta \\ \text{nothing} & \text{wp. } 1-\delta \end{cases}$

Proof:

① Consider two neighbouring datasets: D and D', assume they differ in the i-th element x_i,

② Consider S as a set of outputs from Y

③ Name the "BAD" with output nothing, and "Good" with output (i, x_i) wp. δ

Therefore:

$$\Pr(Y(D) \in S) = \Pr(Y(D) \in S \mid BAD) \cdot \Pr(BAD) + \Pr(Y(D) \in S \mid Good) \cdot \Pr(Good)$$

$$= \Pr(Y(D) \in S \mid BAD) \cdot (1-\delta) + \Pr(Y(D) \in S \mid Good) \cdot \delta$$

$$= \underbrace{\Pr(Y(D') \in S \mid BAD)}_{①} \cdot (1-\delta) + \underbrace{\Pr(Y(D) \in S \mid Good)}_{②} \cdot \delta$$

Because ② is a probability, therefore ② ≤ 1

$$\Pr(Y(D) \in S) \leq \Pr(Y(D') \in S) \cdot (1-\delta) + \delta$$

$$\leq \Pr(Y(D') \in S) + \delta$$

A(Name and Shame) is $(0, \delta)$-Dp.

2. Noisy-max with Laplace Noise

Question 2:

We use $Lap(\frac{\Delta}{\varepsilon})$ in the Noisy-Max mechanism

① We consider two neighbouring datasets D, and D'

② We fix the noise $Z_1, \dots Z_{i-1}, Z_{i+1}, \dots Z_d$, and denote as Z_{-i},

We want to show that for any D, D'.

$$P(Y=i \mid D, Z_{-i}) \leq e^{\varepsilon} P(Y=i \mid D', Z_{-i})$$

We define: $z^* = \underset{z}{\operatorname{argmin}}\ q(i;D) + z > q(j,D) + z_j \quad \forall j \neq i$

We have given D and fixed Z_{-i}, output $Y=i$ if and only if $z_i \geq z^*$

Thus:

$$q(i,D) + z^* \geq q(j,D) + z_j$$

$$\Rightarrow q(i;D) + \Delta + z^* \geq q(j,D') + z_j - \Delta$$

$$\Rightarrow q(i;D') + (z^* + 2\Delta) \geq q(j;D) + z_j$$

Thus: for D' with fixed Z_{-i}, if $z_i \geq z^* + 2\Delta$, the output for D' will also be i.

Thus we have: $P(Y=i \mid D', Z_{-i}) \geq P(z_1 \geq z^* + 2\Delta)$

$$P(z_i \geq z^* + 2\Delta) = \int_{z^*+2\Delta}^{\infty} \frac{\varepsilon}{2\Delta} \exp\left(-\frac{y\varepsilon}{\Delta}\right) dy$$

$$= e^{-\varepsilon} \int_{z^*}^{\infty} \frac{\varepsilon}{2\Delta} \exp\left(-\frac{y\varepsilon}{\Delta}\right) dy$$

$$= e^{-\varepsilon} P(Y=i \mid D, z_{-i})$$

4. Implementation of Noisy-max Mechanism and Exponential Mechanism

I select the Approval Voting problem

After generating the database D we can have the vote information as the follows:

```
d = [0,1,2,3,4]   # 5 candidates
n = 10 # 10 voters

# Vote Process
vote = []

for i in range(n):
    vote_each = sorted (random.sample(d,random.randint(1,5)))
    vote.append(vote_each)
    print('voter:{} vote for {}'.format(i,vote[i]))
```

```
voter:0 vote for [4]
voter:1 vote for [0, 1, 2, 3, 4]
voter:2 vote for [1, 3]
voter:3 vote for [1, 3]
voter:4 vote for [0, 1, 3]
voter:5 vote for [2, 3, 4]
voter:6 vote for [0, 3]
voter:7 vote for [1, 2, 3, 4]
voter:8 vote for [3]
voter:9 vote for [1, 3, 4]
```

Therefore, the total vote pool will be :

```
[[4], [0, 1, 2, 3, 4], [1, 3], [1, 3], [0, 1, 3], [2, 3, 4], [0, 3], [1, 2, 3,
4], [3], [1, 3, 4]]
[3. 6. 3. 9. 5.]
```

The score function of th candidate j in dataset D is

$$q(j;D) = |i : j \in x_i|$$

```
def Global_Sensitivity(score_function,score_list,candidate_list,Dataset,
Parallel_Datasets):
    sensitivity = 0

    for D in Parallel_Datasets:
        score_list_parallel = score_function(candidate_list,D)
        diff = np.max(abs(score_list_parallel - score_list))
        if diff >= sensitivity:
            sensitivity = diff
    return sensitivity
```

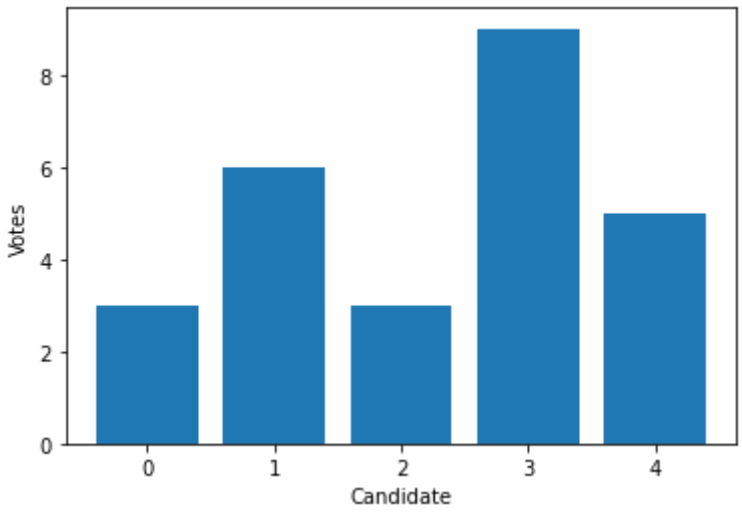
After calculating the global sensitivity of this scoring function, we have the sensitivity is 1:

$$\Delta = 1$$

The python implementation of the Noisy-max Mechanism and Exponential Mechanism

```
def laplace_mech(v, sensitivity, epsilon):
    return v + np.random.laplace(loc=0, scale=sensitivity / epsilon)

def report_noisy_max(score_list, sensitivity, epsilon):
```

```
    # Add noise to each score
    noisy_scores = [laplace_mech(score, sensitivity, epsilon) for score in
score_list]

    # Find the index of the maximum score
    max_idx = np.argmax(noisy_scores)

    # Return the element corresponding to that index
    return max_idx


def exponential(score_list, sensitivity , epsilon):

    # Calculate the probability for each element, based on its score
    probabilities = [np.exp(epsilon * score / (2 * sensitivity)) for score in
score_list]

    # Normalize the probabilties so they sum to 1
    probabilities = probabilities / np.linalg.norm(probabilities, ord=1)

    # Choose an element from R based on the probabilities
    return np.random.choice(d, 1, p=probabilities)[0]
```



The number of votes for each candidate is [3. 6. 3. 9. 5.]

Study of the effect of epsilon on the Exponential Mechanism

epsilon = 0.01, exponential mechanism gives Candidate 4
epsilon = 0.1, exponential mechanism gives Candidate 3
epsilon = 0.2, exponential mechanism gives Candidate 4
epsilon = 0.3, exponential mechanism gives Candidate 1
epsilon = 0.5, exponential mechanism gives Candidate 3
epsilon = 1, exponential mechanism gives Candidate 3
epsilon = 5, exponential mechanism gives Candidate 3
epsilon = 10, exponential mechanism gives Candidate 3
epsilon = 20, exponential mechanism gives Candidate 3
epsilon = 100, exponential mechanism gives Candidate 3

Study of the effect of epsilon on the Noisy-max Mechanism

epsilon = 0.01, noisy_max mechanism gives Candidate 3
epsilon = 0.1, noisy_max mechanism gives Candidate 2
epsilon = 0.2, noisy_max mechanism gives Candidate 1
epsilon = 0.3, noisy_max mechanism gives Candidate 2
epsilon = 0.5, noisy_max mechanism gives Candidate 3
epsilon = 1, noisy_max mechanism gives Candidate 3
epsilon = 5, noisy_max mechanism gives Candidate 3
epsilon = 10, noisy_max mechanism gives Candidate 3
epsilon = 20, noisy_max mechanism gives Candidate 3
epsilon = 100, noisy_max mechanism gives Candidate 3

My findings

The exponential mechanism provides differential privacy by *approximately* maximizing the score of the element it returns When the epsilon is very low (<0.3) the exponential mechanism sometimes returns an element from the set which does *not* have the highest score.

For the noisy-max mechanism, by adding a Laplace noise on each score, we guarantee differential privacy. As can be seen in the study of comparing different epsilons, we can see that when the epsilon is set low (<1) the noisy-max mechanism sometimes do not returns the candidate with the highest score.

5. Comparison with Gaussian and Laplace Mechanism

Generate a dataset D = {x1, · · · , xn} where each xi ∈ {0, 1}d. Consider answering the average query

$$f(D) = (1/n)\sum_{i=1}^{n} x_i$$

via Laplace and Gaussian mechanism. Implement these two mechanisms with variate n, d, " (You cases must at least included = 1 and d ≫ 1). Write a report on your findings.

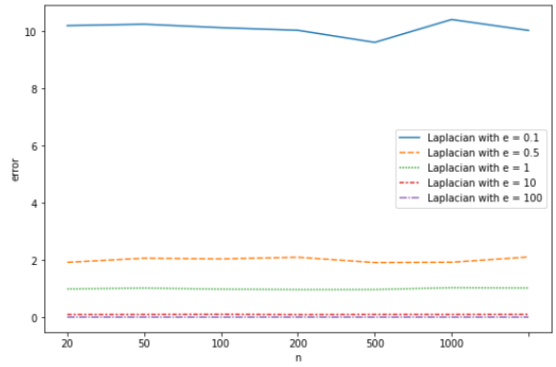
Dataset Generation

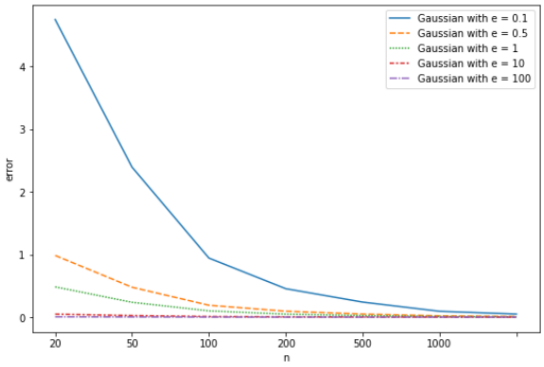
```
data_pool = [0,1]

def generate_one_data(d):
    data_point = []
    for i in range(d):
        data_point.append(random.choice(data_pool))
    return data_point

def generate_dataset(n,d):
    dataset = []
    for n_i in range(n):
        dataset.append(generate_one_data(d))
    return dataset

def f(D):
    return np.mean(D)
```

Laplace and Gaussian Mechanism Definition

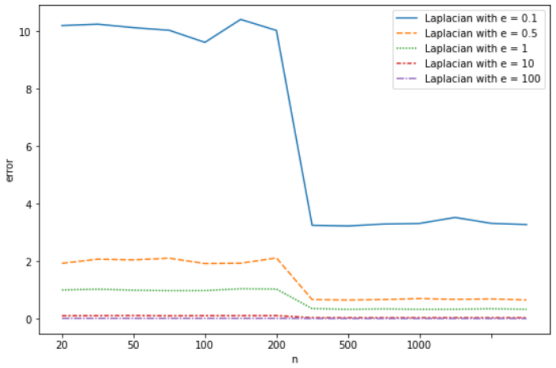
```
import math

def laplace_mech(v, sensitivity, epsilon,n):
    return v + np.random.laplace(0, sensitivity / epsilon*n)

def Gaussian_mech(v, sensitivity, delta, epsilon):

    sigma = math.sqrt( (32*sensitivity**2*math.log(2/delta)) / (9* epsilon**2) )
    return v + np.random.normal(0,sigma)
```

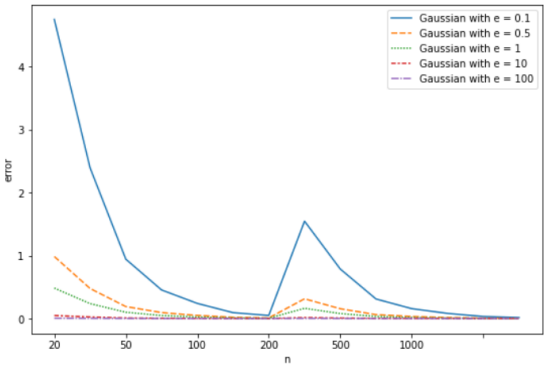
d = 1



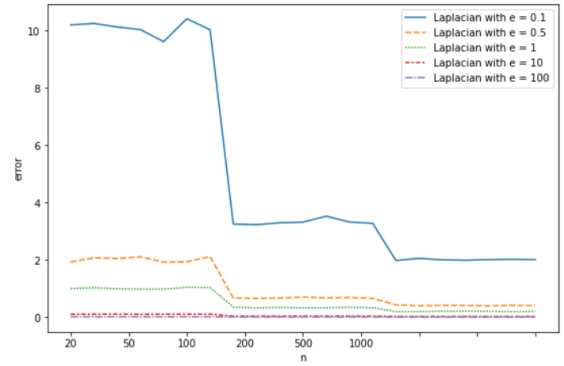


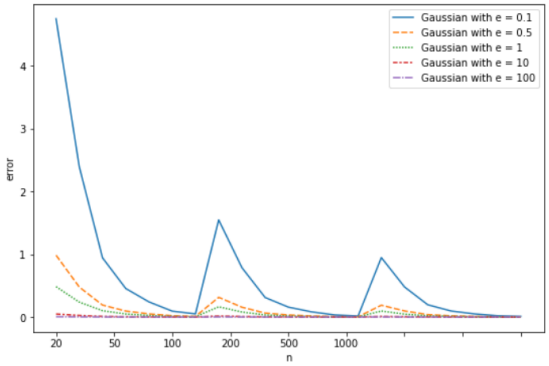
d = 3



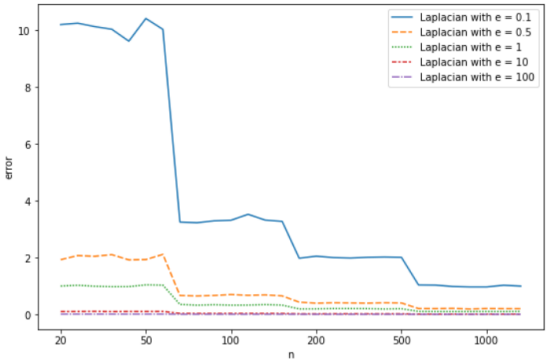


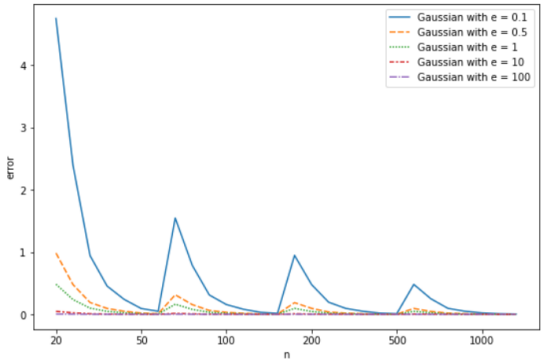
d = 5





d = 10





Comment

After doing the experiment on setting different d and different epsilon, I find that the error of Laplace mechanism is much

higher than that of Gaussian mechanism. However, it does not prove the utility of the approximate DP is better than pure

DP. As I'm setting the delta of the Gaussian Mechanism to 10e-5, which is a very low value.

Also, when having n being a very large value (>100), the average error is reduced, proving that more data points in one dataset would help preserve the privacy.