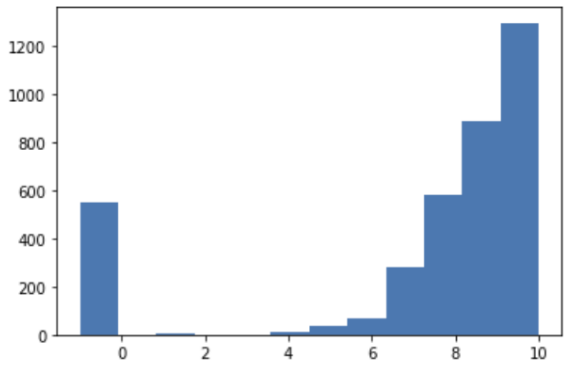
Doruk ÖZER 70192

Report

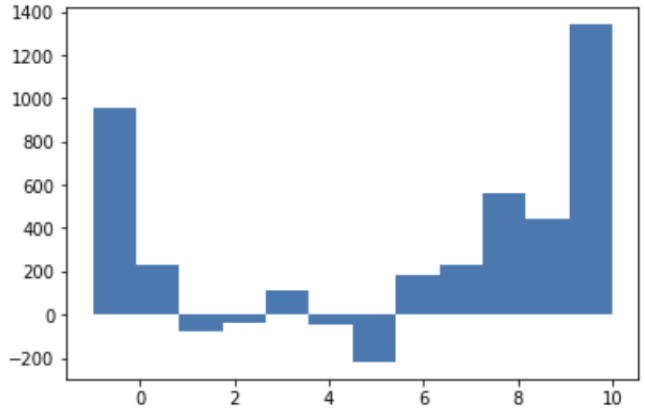
In [7]:
```
bins = np.array([-1,0,1, 2, 3, 4, 5, 6, 7, 8, 9,10]).astype(float)
counts = get_histogram(dataset).astype(float)

centroids = (bins[1:] + bins[:-1]) / 2
counts_, bins_, _ = plt.hist(bins, bins=len(counts),
                             weights=counts, range=(min(bins), max(bins)))
plt.show()
```



This is the histogram result, and how I calculated it

:
```
epsilon=0.01
bins = np.array([-1,0,1, 2, 3, 4, 5, 6, 7, 8, 9,10]).astype(float)
counts = get_dp_histogram(get_histogram(dataset),epsilon).astype(float)
centroids = (bins[1:] + bins[:-1]) / 2
counts_, bins_, _ = plt.hist(bins, bins=len(counts),
                             weights=counts, range=(min(bins), max(bins)))
plt.show()
```



This is the histogram result of differentially private histogram with epsilon 0.01 , and how I calculated it

Task 1

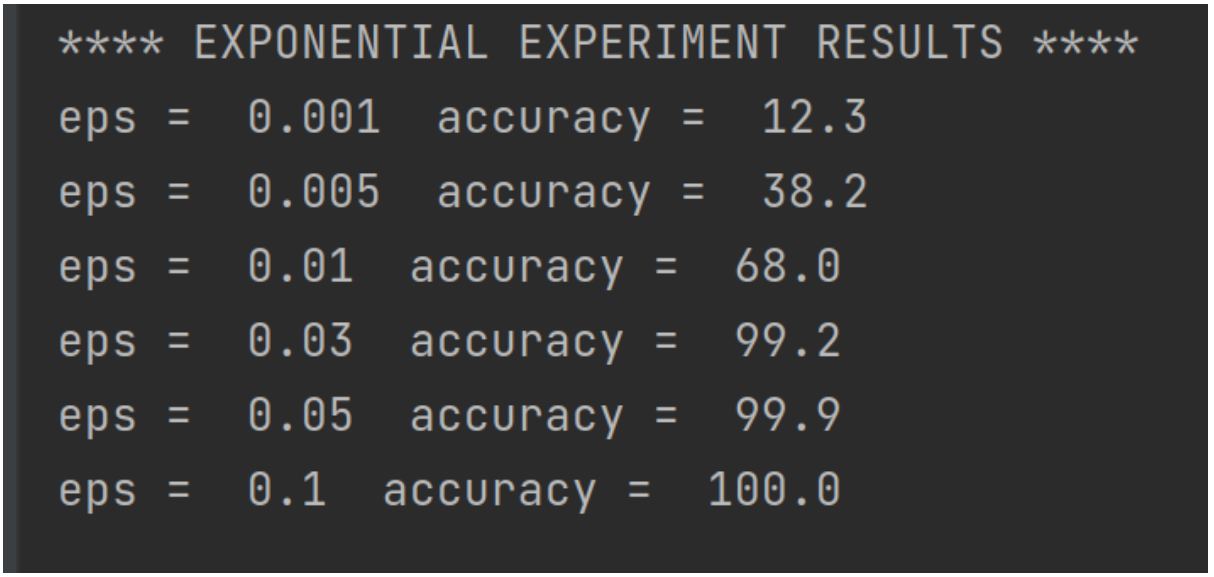
Table Of MSE and AvgErr For each Epsilon for Laplace Mechanism

```
**** LAPLACE EXPERIMENT RESULTS ****
**** AVERAGE ERROR ****
eps =  0.0001  error =  20500.0375
eps =  0.001  error =  1942.8499999999992
eps =  0.005  error =  422.04166666666663
eps =  0.01  error =  205.23958333333331
eps =  0.05  error =  39.20625
eps =  0.1  error =  19.875
eps =  1.0  error =  1.9104166666666664
**** MEAN SQUARED ERROR ****
eps =  0.0001  error =  906240357.6208336
eps =  0.001  error =  7433305.766666666
eps =  0.005  error =  351959.7083333334
eps =  0.01  error =  81972.13541666664
eps =  0.05  error =  2878.297916666667
eps =  0.1  error =  770.5416666666667
eps =  1.0  error =  7.447916666666664
```

In the lectures we discussed that the lower the value of epsilon means more privacy added to the algorithm. In Laplace mechanism we add Laplace noise with scale sensitivity of query divided by epsilon and with zero mean. From the equation of Laplace noise when we increase the epsilon then the amount of laplace noise added decreases. This is also what I observed in my experiments as epsilon increased error for both MSE and AvgErr decreased.

Task 2

Table Of MSE and AvgErr For each Epsilon for Exponential Mechanism



In exponential mechanism i defined the scoring function as number of people who voted 10 and I computed it for each anime. The sensitivity of this scoring function is 1 according to my scoring function. It is known that the probability of choosing the correct result increases as the epsilon increases for exponential mechanism's probability function. This is also what I observed in my experiments as the epsilon increased the accuracy is increased.

Part 3 Analysis

Results

```
GRR EXPERIMENT
e=0.1, Error: 14678.12
e=0.5, Error: 4433.41
e=1.0, Error: 1454.65
e=2.0, Error: 650.76
e=4.0, Error: 112.59
e=6.0, Error: 36.00
**************************************************
RAPPOR EXPERIMENT
e=0.1, Error: 11525.88
e=0.5, Error: 2360.00
e=1.0, Error: 1025.47
e=2.0, Error: 558.41
e=4.0, Error: 222.71
e=6.0, Error: 154.76
**************************************************
OUE EXPERIMENT
e=0.1, Error: 12966.59
e=0.5, Error: 2147.65
e=1.0, Error: 1026.59
e=2.0, Error: 496.94
e=4.0, Error: 188.59
e=6.0, Error: 159.65
```

Table of Results

Epsilons	GRR EXPERIMENT	RAPPOR EXPERIMENT	OUE EXPERIMENT
0.1	14678.12	11525.88	12966.59
0.5	4433.41	2360.00	2147.65
1.0	1454.65	1025.47	1026.59
2.0	650.76	558.41	496.94
4.0	112.59	222.71	188.59
6.0	36.00	154.76	159.65

Analysis

General results shows that as the epsilon increases the error decreases

GRR: Experiments showed that the error of GRR decreases as the epsilon increases. The GRR depends on the length of the domain so we can conclude that the probability that is used in perturbation part and, estimation, decreases as the length of domain increases. This means, for long domains we would have poorer results for the same epsilon values.

Based on the experiment results the GRR results performed much better in the range epsilon>=4. The results and estimations are less distorted compared to estimations calculated by RAPPOR and OUE in that certain domain. For the range epsilon<4 it performed worse than RAPPOR and OUE.

Since the GRR depends on the length of the domain RAPPOR and OUE is proposed to get rid of the dependency.

RAPPOR: In RAPPOR, one hot encoding is used to represent the data of the user, and the data is converted in bit vector. The bits on the bit vector is perturbed (conversion of bit 1-> bit 0, bit 0-> bit 1) with certain probability. For the epsilon values 1.0, and 6.0, 0.1 it performed better than OUE, but I would say it is approximately same with OUE.

OUE: OUE differs from RAPPOR, as it uses different probabilities for perturbation of zero bits and one bits, so it treats zero bits and one bits differently compared to RAPPOR, although they use the same encoding. In addition, the estimator of OUE is different than RAPPOR since the perturbation operation is different and it depends on the value of the bit. As I emphasized in the RAPPOR part, it performs nearly same as RAPPOR.