Adaptive Granularity Mechanism

High Level Description

At a high level, the mechanism we propose is very simple. It consists of the following steps.

1) **Select** a collection of low-dimensional marginals to be measured. (These are selected using the public data, and are hard-coded into the mechanism)
2) **Measure** a noisy linear transformation of each marginal by introducing Gaussian noise to satisfy the required differential privacy level.
3) **Post-process** the noisy measurements using Private-PGM and generate synthetic data that preserves the measured information well.

Important notes:

- All queries measured by our mechanism are about trips, and not taxis. We pay the full 200X sensitivity price for these queries. We introduce clipping to reduce this to up to 150X by introducing some bias.
- We assign taxi ids by leveraging the public data. This step does not consume the privacy budget at all.

Adaptive Granularity Example

The key innovation in this round of the competition is the introduction of the linear transformation described in the Measure step above. In previous rounds of the competition, we would typically just measure each marginal directly (this can be seen as an “Identity” linear transformation), and is a special case the mechanism we propose.

In this section, we give an intuitive description of the key ideas behind how this linear transformation is defined and why we expect it to help. We defer the formal treatment of the mechanism to later sections.

Suppose for the sake of the example that we had a 2-dimension domain, where the first attribute can assume the values {W,X,Y,Z} and the second attribute can assume the values {1,2,3}. We begin by answering each 1-way marginal via the Gaussian mechanism. After inspecting the output, we observe that “X” has a small count for attribute and “3” has a small count for attribute 2. We want to measure the two-way marginal, but it only make sense to measure things which have the potential to have large counts, as adding noise to many near-zero counts can skew the mechanism in undesirable ways. Thus, in this case, we should measure the count for { W, Y, Z } x { 1, 2 }, giving us a finer granularity estimate of these values,

but skip measuring everything else (i.e., we don't want to measure the count for (W,3) because we know that count will be small). By doing this naively, we would actually be wasting some privacy budget. In particular, by parallel composition, we can measure things about "X" and "3" without spending additional privacy budget, as long as we do so carefully.

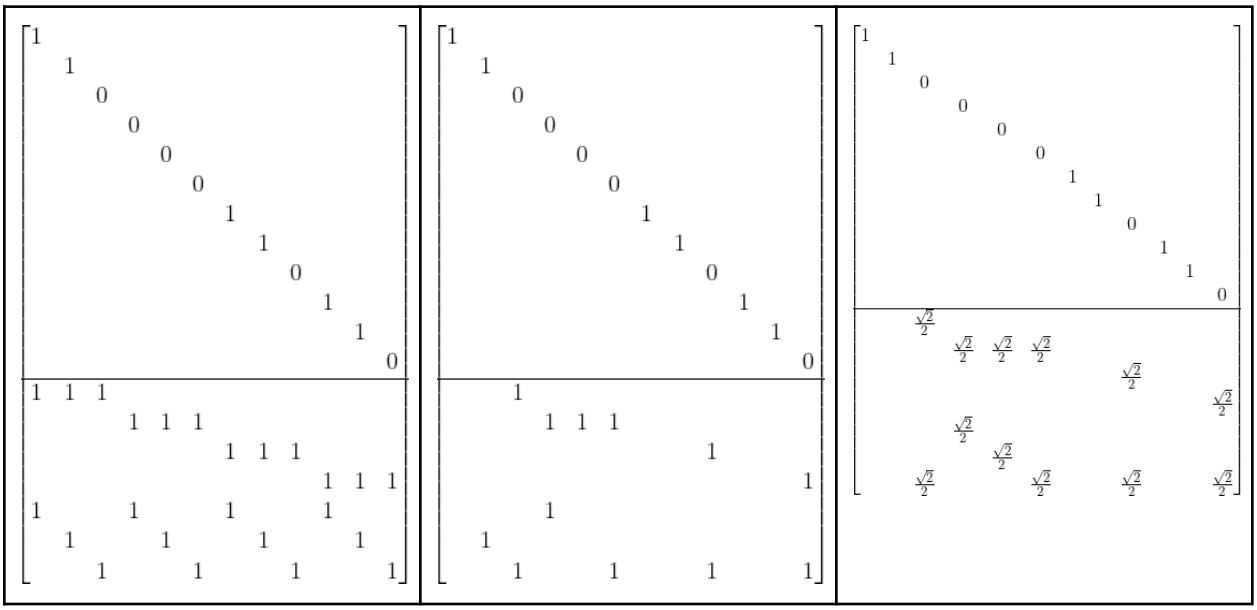
W	10000
X	300
Y	5000
Z	6000

1	15000
2	6100
3	200

Let mu be the marginal vector, which contains the true count for each possible combination of attribute values for the two attributes. Then this vector clearly has size 4*3 = 12. Using the standard indexing scheme: 1 → (W,1); 2 → (W,2); 3 → (W,3); …; 11 → (Z,2); 12 → (Z,3) we can view the queries over this marginal vector in matrix form as follows:

$$\begin{bmatrix} 1 & & & & & & & & & & & \\ & 1 & & & & & & & & & & \\ & & 0 & & & & & & & & & \\ & & & 0 & & & & & & & & \\ & & & & 0 & & & & & & & \\ & & & & & 0 & & & & & & \\ & & & & & & 1 & & & & & \\ & & & & & & & 1 & & & & \\ & & & & & & & & 0 & & & \\ & & & & & & & & & 1 & & \\ & & & & & & & & & & 1 & \\ & & & & & & & & & & & 0 \end{bmatrix}$$

This is a diagonal matrix with exactly 6 non-zero entries corresponding exactly to the tuples in the domain corresponding to {W,Y,Z} x {1,2} (i.e., the count we think could be sufficiently large). The sensitivity of this "query matrix" is the largest L2 norm of each column [4]. Clearly, the L2 norm of some columns is 1, and the L2 norm of other columns is 0. Thus, this is in some sense "wasting" the privacy budget. We thus include 7 additional queries to make sure we are taking advantage of the parallel composition and using the privacy budget effectively. In particular, we will use the remaining budget to essentially re-measure the 1-way marginals. We can intuitively think of this as three steps: (1) Add the one-way marginal queries back into the matrix, expressed over the 2D domain. (2) modify the queries to exclude those domain elements that we are measuring at a fine granularity. (3) reweight the new queries so that the total L2 norm of each column is exactly 1 (and hence the L2 sensitivity is also 1).



Detailed Description of the Algorithm

Data preprocessing. We begin by loading and discretizing the given dataset. For each numeric column, we discretize it using the bin widths provided in metric.py, mapping them to integer values 0, 1, …. For each categorical column, we map the possible values to integer bins 0, 1, …, as well. This leads to a discrete dataset where every attribute can assume a finite number of possible values.

Shift	21
Company_id	57
pickup_community_area	78
dropoff_community_area	78
payment_type	9
fare	11
tips	11
trip_total	11
trip_seconds	11
trip_miles	11

As part of this preprocessing step, we assign each record a weight, to limit the magnitude of contribution that can come from any single individual taxi_id. By default these weights are 1, in which case a single individual impacts any marginal by at most 200 (in either L1 or L2 distance). Let C be a clipping value and let R be the number of records contributed by a given taxi_id. Then we assign a weight of min(1.0, C/R) to all records in the dataset for the given taxi_id. If R <= C, then the assigned weight is simply 1, and no clipping is done. If R > C, then the weights will be reduced to limit the impact of individuals with more records. This introduces some bias, but allows us to reduce the necessary magnitude of noise when invoking the Gaussian mechanism later.

Marginal Selection. We begin by selecting a carefully chosen set of marginals that (1) depend on the stated evaluation criteria and (2) exploit correlations we observed in the provisional dataset. These are hard-coded into the mechanism. There are a total of 7 "outer-level" marginals, which always include **pickup_community_area** and **shift**, and additionally include the following combinations of other attributes

1. **fare x trip_total**
2. **trip_total x trip_seconds**
3. **dropoff_community_area x fare**
4. **payment_type x trip_total**
5. **fare x trip_miles**
6. **company_id**
7. **tips x trip_total**

With these outer-level marginals selected, we then compute the *downward closure* of the set. That is, for each outer-level set of attributes, we include all of its subsets as well (without duplicates). Heuristically, we also add 7 extra copies of the **dropoff_community_area** x **pickup_community_area** marginal due to the extra importance of those statistics in the scoring criteria. This leads to a total of 66 marginals that will be measured.

Measurements. Given a gaussian noise scale sigma (defined later), we proceed in 4 rounds. For round k=1,2,3,4 we measure the k-way marginals from the selected set described above. For each marginal we construct a query matrix Q that has max L2 column norm of 1.0. The construction of this query matrix depends on the noisy answers from previous rounds. We use the adaptive grid methodology exemplified in the previous section to construct this query matrix Q. If mu is the marginal vector, and Q has m rows then we compute y = Q*mu + N(0, sigma^2)^m.

Noise calibration and proof of privacy

We will discuss the privacy of our mechanism in the language of Gaussian differential privacy [1], and use that analysis to reason about the noise magnitude required to achieve (epsilon, delta)-differential privacy. First note that the L2 sensitivity of the statistic Q*x is || Q ||_2 * C, where C is the clipping value (e.g., C = 200), and represents the amount a single individual can contribute to any marginal. Here || Q ||_2 is the maximum L2 norm of columns of Q, and is 1.0 by construction. Thus, the sensitivity of this computation is C. By Theorem 2.7 in [1], adding Gaussian noise with scale sigma gives a privacy guarantee of mu-GDP for mu=C/sigma[1]. Moreover, by Theorem 3.2 and in particular Corollary 3.3, the n-fold composition of mu_i-GDP mechanisms is sqrt(mu_1^2 + … mu_n^2)-GDP. In our setting, we have a 66-fold composition since we are measuring 66 marginals. Thus, the privacy guarantee is sqrt(66)*C/sigma. By proposition 2.12 nad in particular Corollary 2.13, we can see that the resulting mechanism is (epsilon, delta)-DP as long as delta = Phi(epsilon/mu - mu/2) - exp(epsilon) * Phi(-epsilon/mu - mu/2). Here, Phi is the cdf of the standard normal distribution. This is exactly the formula for the analytic gaussian mechanism [2], and hence, we can calibrate noise using their open source library [3].

In short, Gaussian privacy is necessary for the analysis because it shows that n-fold *adaptive* composition behaves nicely when each mechanism adds Gaussian noise. Adaptive composition is necessary in our setting because the matrices chosen in later rounds of measurement depend on earlier rounds. If we didn't have to worry about adaptive composition, and the measurements didn't depend on each other, the analysis would have been much simpler: the overall L2 sensitivity of all measurements combined is sqrt(66)*C, which leads to the same answer.

For epsilon=1.0 and delta=2.5e-5, we use a clipping value of C=150. This leads to a gaussian noise scale of 5739.36. For epsilon=10.0 and delta=2.5e-5, we use a clipping value of C=200 (no clipping). This leads to a Gaussian noise scale of 895.

Post-processing. In the post-processing step, we collect all measured information about the marginals, and pass it into Private-PGM [5,6], which learns a graphical model that fits the observations well. We then use the model to generate synthetic data using the method provided in Private-PGM. Once the data is generated, we undo the discretization to return the data to its original domain. We finally use a heuristic taxi_id assignment strategy that leverages the public data.

[1] Dong, Jinshuo, Aaron Roth, and Weijie J. Su. "Gaussian differential privacy." *arXiv preprint arXiv:1905.02383* (2019).

[1] Note that this theorem covers a special case where the computation outputs a scalar values. It also applies to vector valued functions when using L2 sensitivity. Please refer to the discussion of SGD in section 5, as well as the analysis in [2].

[2] Balle, Borja, and Yu-Xiang Wang. "Improving the Gaussian mechanism for differential privacy: Analytical calibration and optimal denoising." *International Conference on Machine Learning*. PMLR, 2018.

[3] https://github.com/yuxiangw/autodp

[4] Li, Chao, et al. "The matrix mechanism: optimizing linear counting queries under differential privacy." *The VLDB journal* 24.6 (2015): 757-781.

[5] McKenna, Ryan, Daniel Sheldon, and Gerome Miklau. "Graphical-model based estimation and inference for differential privacy." *International Conference on Machine Learning*. PMLR, 2019.

[6] https://github.com/ryan112358/private-pgm

Code Outline

Important parts of the code are commented. Here is a general overview of the main pieces:

- **adagrid** is the main mechanism implementation. It consumes the data, privacy parameters, and hyper-parameters. It produces synthetic data for a single epsilon.
- **Get_identity** and **get_aggregate** are used to construct the query matrix for each marginal. The L2 sensitivity is printed to the terminal to show that it is indeed a sensitivity 1 query matrix.
- **assign_taxi_ids** is our heuristic method to assign a taxi_id to every row in the dataset. It does not depend on the sensitive data at all.
- **discretize** and **undo_discretize** pre-process the data as described above.
- **run _mechanism** consumes the dataset, schema, and runtime parameters, and automatically selects the hyperparameters to call **adagrid**.
- **main** just runs the mechanism for all provided runtime/privacy parameters.