Privacy Preserving Data Publishing: A brief Survey and Study of A Differentially Private Algorithm For Histogram Release

Hitaxi Kalaria
Emory University, Atlanta GA 30322

August 19, 2010

Abstract

The problem of *statistical disclosure control* is to release important statistics from a large dataset which are useful of organizations and researches while ensuring to protect the privacy of individuals representing the dataset. Several fields have dedicated work and research to address issue of data privacy when releasing statistical information. This paper is a brief survey about certain weak privacy preserving principles like k-anonymity and l-diversity and differential privacy mechanism. We also discuss the problem of releasing differentially private histogram based on interactive differential privacy interface along with a cell-based partitioning algorithm and a kd-tree based algorithm to generate multidimensional partitions followed by the experimental results for the kd-tree based algorithm. Furthermore, we introduce and discuss a Health Information DE-identification framework (HIDE), to de-identify/anonymize heterogeneous medical data.

1 Introduction

With the advancement in information technology it has become feasible to store and collect enormous datasets. These datasets can be in the form of medical records, financial records, geographical data or census data. Data providers release such datasets to data miners and researchers who see great potential in such datasets as many useful statistics and information can be obtained from them. However since most of the data contain explicit information called personal identifiable information about an individual, the release of such dataset poses an attack on privacy of individuals. The simple solution would be to not release such information by removing or replacing personal information from the dataset. But this leaves the dataset less useful for research or to generate statistics. This is where data privacy community comes into picture. The prime work in the field of data privacy is to release useful datasets while preserving privacy of individuals. The research in this area is devoted to develop techniques and algorithms that guarantee privacy of a dataset. We will discuss a few principles and privacy mechanisms defined and developed to address the issue of privacy in released datasets [1, 4, 8].

2 Weak Privacy Preserving Principles

The primary goal of a data provider(curator) is to collect information from a large population(samples) and release statistical information about the population. However, the curator must release accurate statistics with a guarantee to protect the privacy of individuals. This is the called the problem of

statistical disclosure control [2, 1].Several fields of such as statistics, theoritical computer science, security, databases and cryptography have dedicated work and research to address this problem because of the valuable information provided by statistical databases. The information from statistical databases can be used to identify information such as common features within a group of people which is not obtained by analyzing individual data. Thus it is important for a data provider to guarantee the privacy of individuals who are a part of statistical databases. Data providers usually address the issue of privacy when releasing a dataset by removing all personal identifiable information such as name, address, age, etc. Since personal identifiable information is removed the resulting dataset is believed to be anonymous and is released by the data providers. However, simple removal of personal identifiable information is not enough for guaranteeing privacy of individuals. It has been shown that it is possible to re-identify an individual by linking information from two entirely different datasets [8]. A common solution to the problem could seem to have multiple level access databases but the problem remains when different organizations release datasets based on different privacy criteria. Also the area of computer security ensures that an authorized person receives the intended information but it does not protect against the attack of re-identifying using external/background information once the statistical data is released. In the following section we discuss k-anonymity and l-diversity principles to address the issue of privacy in released datasets.

2.1 k-anonymity

The problem of re-identifying by linking occurs when a set of attributes a common to both dataset A and B or a dataset A and some external information are used to successfully match an individual. The set of attributes a is not considered personal identifiable information in individual dataset but when combined with other information from both datasets or some other background information they successfully identify an individual.

Definition 1. *Let the dataset A be a table with finite number of tuples. Let a be the set of attributes of A. A quasi identifier q is defined as a set of attributes, where $q \subset a$, which when combined with external information is able to identify a tuple in A.*

Personal information such as name, address, zip code, birth date from a voters list are considered quasi identifiers. A data adversary is able to identify a tuple when he correctly identifies the value of the quasi identifier set that define the tuple. But if the same value of quasi identifier set also define few other tuples than it becomes difficult for the adversary to pinpoint on exactly one particular tuple using the given quasi identifier set. This is the basis for the k-anonymity principle.

Definition 2. *A table A is considered k-anonymous if and only if for a given set of quasi identifiers q, $A[q]$ defines a tuple than there must be at least $k-1$ other tuples defined by $A[q]$.*

k-anonymity can be achieved by using domain generalization on domains of attributes in the quasi identifier set. Thus for a table to be k-anonymous it is important to correctly identify the quasi identifier set. Even when careful consideration is given while identifying the correct quasi identifier, there are some possible attacks on k-anonymity principle that compromise privacy. Below are some of the possible attacks with measures to avoid them:

- If the order of tuples in both k anonymous releases of a table is exactly same, then it is possible for an adversary to use both releases to identify personal information. This can be prevented if the tuples are randomly ordered following each release.

- Adversary can use a previous release of the table along with other external information to re-identify a tuple from current release. This can be prevented if the attributes from previous release are considered to be quasi identifiers for future releases.
- A k-anonymized version b_1 of a table is released and after the release some more tuples are added to the table. Hence a new k-anonymized version b_2 of the table is released due to the addition of new tuples. However, an adversary can compare releases b_1 and b_2 and thus obtain information about the tuples that were added later on. To prevent this attributes from previous release must be considered when identifying quasi identifiers for future releases.

Note: Section on k-anonymity has been adapted from [8].

2.2 l-diversity

In the previous section we mentioned some attacks against k-anonymity and ways to prevent them. k-anonymity is accepted as a possible judging criteria to guarantee privacy as it is simple and easily achievable .However, even when careful measures are taken to identify the correct set of quasi identifier and also avoid the attacks discussed in section 2.1, k-anonymity does not guarantee privacy in all situation. In their work [6], Machanavajjhala et al. showed attacks based on background knowledge and homogeneity against k-anonymity. It was observed that in the homogeneity attack, due to lack of diversity in the domain of sensitive information attribute, a k-anonymous table creates groups of tuples that leak information. Also, k-anonymity provides no protection against attacks based on background information because a data provider has no control on how much does an adversary knows about an individual. Thus, a privacy principle must ensure enough diversity among the domain of sensitive information attribute and should not take into consideration the background information available to the adversary while providing privacy.

Definition 3. *A attribute is considered sensitive if its value is considered personal information of an individual and should be kept private. Medical condition is considered to be a sensitive attribute in a medical records database. Attributes that are not sensitive are called non-sensitive attributes. A set of non-sensitive attributes q make up the quasi identifier as they can be linked with external information to identify an individual.*

As we have seen that a table A is k-anonymous if and only if for possible value $A[q]$ for the given quasi identifier set q there are at least k tuples satisfying $A[q]$. Thus, a k-anonymous table creates groups of tuples, where each tuple in a group has the same value for their non sensitive attributes. Since, the grouping is not based on sensitive attribute, there exists some groups where the tuples have the same value for the sensitive attribute besides the non-sensitive attributes. Such groups lacking diversity within sensitive attributes are prone to privacy attacks as it becomes easier for an adversary to learn information about a group of individual sharing a common sensitive attribute value. e.g. adversary can learn that k people living in a specific area have a specific medical condition like cancer. To resolve the issue it is important to have at least more than one different value for the sensitive attribute within a group. This is the basis for l-diversity.

Definition 4. *A group b is considered l-diverse if and only if it contains at least l different values for the sensitive attribute. A table is l-diverse if every group within the table is l-diverse.*

Thus, a l-diverse table resolves the issue of homogeneity attack as the values for the sensitive attribute are diversified. Also, data provider does not need to assume what amount of knowledge

does an adversary posses. The parameter l is proportional to the amount of background information. Larger value of l means more background information is needed to successfully indentify a tuple in the released table.

Note: Section on l-diversity has been adapted from [6].

3 Differential Privacy

k-anonymity and l-diversity ensure privacy of a database by modifying and generalizing domains of some key attributes of the database. Moreover to use these principles it is important for a data provider to identify every possible quasi identifier to ensure privacy. However, in reality it is not feasible to find all quasi identifiers and the data provider can never know all the possible attacks that would be carried out by an adversary on the database. Also, there is no bound on how much external information an adversary possess. Thus, principles like k-anonymity and l-diversity can be used to judge if a released database is private but they do not guarantee against attacks on privacy due to background information. We will discuss a new notion of providing data privacy called differential privacy.

3.1 Background

In the problem of *statistical disclosure control* a data provider must release accurate statistics from a dataset while preserving the privacy of individual that represent the dataset. Below are some of suggestions to address the issue of privacy in statistical databases besides the two privacy principles discussed in section 2 are [1]:

- Providing privacy by rejecting queries that target specific individual or a group. This is not enough to guarantee privacy as the adversary can ask several similar queries and use the results to get specific information about individuals.
- Another approach is *query auditing* where each query is audited by the database and a history is maintained. If a query is deemed to be to much revealing it is denied. The problem with this approach is that a denial of query is enough to give out information about individuals within the database.
- A *sub sample* is a subset of a larger database which can be used as a representative of the larger set. This sub-sample can then be released but privacy for individuals who are part of the sub-sample is neglected.
- Another approach is *input perturbation*, in which data or queries are modified before a response is generated. Every time a same query is repeated the database gives the same output.
- In *randomized response*, the data is randomized permanently and statistics are calculated from the randomized response. The approach becomes more tedious for complex data.
- Adding *random noise* to the released data is another possibility to ensure privacy.

While one of the above approach can be used to provide privacy of information of individuals representing the database, sometimes the approach releases information about individuals who are

not a part of database. Thus an individual's privacy can be compromised even if he is not a part of any database. This leads us to discuss work by Cynthia Dwork on a new privacy mechanism termed *differential privacy*. The basic idea behind differential privacy is that the output of any query must not be affected by removal or addition of one single record from the database, i.e. joining or leaving a database should not affect privacy of an individual and thus encouraging participation in databases which results in increase of valuable statistical releases of data [2, 1].

3.2 Defining Differential Privacy

Consider two databases D_1 and D_2, where the only difference between them is a single record i.e one is a subset of other and larger database contains just one extra record. Let K be the randomized algorithm applied by the data provider when releasing information. Then differential privacy is defined as:

Definition 5. *[2] A randomized function K gives ϵ-differential privacy if for all data sets D_1 and D_2 differing by at most one element, and all $S \subseteq Range(K)$,*

$$Pr[K(D_1) \in S] \leq exp(\epsilon) \times Pr[K(D_2) \in S] \tag{1}$$

The probability is taken over the coin tosses of K.

A function K that satisfies the above definition guarantees that even if the information of an individual is removed from the database it does not affect the probability of an output to occur. The parameter ϵ is public and can assume values like 0.01, 0.1 or ln 2 or ln 3. The privacy of a dataset if based on ϵ. Thus differential privacy is a strong notion for guaranteeing privacy and it is independent of any background information.

3.3 Achieving Differential Privacy

The idea of differential private mechanism is to generate the same output regardless if a particular record is present or not present in the database. This can be achieved by adding Laplace random noise. Let m be the true answer and r be the possible response to a query. Then the random noise added to the query must be of magnitude $r-m$. Similarly if the true answer is $m-1$ and response is r, the noise magnitude is $r-m+1$. Thus, for response r to be differentially private it is sufficient for

$$exp(-\epsilon) \leq \frac{Pr[noise = r-m]}{Pr[noise = r-m+1]} \leq exp(\epsilon) \tag{2}$$

For two arbitrary databases D_1 and D_2 differing in at most one record, sensitivity, ΔQ of query Q is defined as the maximum difference between the query results of D_1 and D_2.

$$\Delta Q = max[Q(D_1) - Q(D_2)] \tag{3}$$

Thus to achieve ϵ-differential privacy for a given query Q on dataset D it is sufficient to return the response as $Q(D) + noise$ instead of returning the true answer $Q(D)$ where *noise* is derived from $Lap(\Delta Q/\epsilon)$. Noise is dependent on ΔQ and ϵ. Increasing $Lap(\Delta Q/\epsilon)$ flattens the noise curve resulting into more privacy. Noise must grow with the number of queries posed to a database.

Note: Section 4, 4.1 and 4.2 have been adapted from [2, 1].

3.4 Interactive and Non-Interactive Release of a Database

There are two settings to release statistics from a statistical database using differential privacy mechanism and a fixed privacy budget. In a *interactive* setting like Privacy INtegrated Queries platform (PINQ) [7], the data provider uses a differential privacy mechanism that adds random noise to each query posed based on the privacy budget. As the number of queries increases each query gets a lower privacy budget which results in increase in the amount of noise added with each response. Once the privacy budget has been exhausted the interface comes to a complete shutdown. In a $non-interactive$ setting the data provider releases a 'sanitized' version of output based on the privacy budget and query sequence. Once a 'sanitized' database is generated the original data is never used again. Thus, a data provider needs to design an algorithm that will minimize the random noise added to the true response for a particular query sequence [2, 1, 9].

3.5 Algorithms

There are several algorithms based on differential privacy for problems like $Histogram\ Queries$, $k-Means\ Clustering$ and $Statistical\ Data\ Inference$ [2, 1]. In this section we will discuss two algorithms for the problem of differentially private histogram release based on an interactive differential privacy interface [9].

3.5.1 Introduction and Background

A *histogram* is created by partitioning the database into number of partitions with a certain number of points in each partition. A interactive mechanism provides an algorithm that uses a certain partitioning strategy to create a differentially private histogram using specific queries to query the database. Once the histogram is generated it can be treated and released as a 'sanitized' version of the original data for queries like count and sum. For a sequence of differentially private computations, the composability of differential privacy ensures privacy guarantee for that sequence. For a series of analysis the privacy parameter values add up. Thus, more information exposed implies less privacy. If the analysis operate on disjoint subsets of the data, the final privacy guarantee depends only on the worst case and not the sum.

Definition 6. *Consider a set M_i of queries, each providing ϵ_i-differential privacy. If M_i is sequential, i.e. the queries are related then the sequence M_i provides $(\sum_i \epsilon_i)$-differential privacy. This is called sequential composition of differential privacy.*

Definition 7. *If each M_i provide ϵ-differential privacy for D_i, where D_i is a subset of database, then the sequence M_i provides ϵ-differential privacy. This is called parallel composition of differential privacy. Thus in worst case privacy guarantees by a sequence of queries is additive reductions of the overall privacy budget.*

The parameter ϵ determines the level of differential privacy and its choice is public. Since, the entire privacy budget is based on ϵ it is important to choose the right value for it. A sufficient bound for ϵ has been proposed by Xioa, Xiong and Yuan [9], based on the analysis of prior and posterior probability of a response.

Corollary 1. *Let P_0 be the prior probability of a response and P_l be the posterior probability of a response after l queries. Then, $\epsilon < ln(x)/l$ is sufficient bound for guaranteeing $P_l/P_0 < x$ and $\epsilon < -ln(P_0)/l$ is a sufficient bound for guaranteeing $P_l < 1$.*

Definition 8. *A "data cube" is a representation of database with N dimensions, in N-dimensional cube. All the records in the database are points in the data cube. A "partition" is any sub-cube in the data cube and a "cell" is the smallest partition that cannot be divided any further.*

Definition 9. *A database mechanishm A is (ϵ, δ)-useful for queries in class C if with probability $1 - \delta$, for every $Q \in C$, and every database D, $A(D) = \hat{D}, |Q(\hat{D}) - Q(D)| \leq \epsilon$.*

Aggregate queries functions can be of two forms. A distributive aggregate query function can be computed by partitioning the dataset into small subsets, computing the aggregate function on each subset and finally combining the results from each subsets to get the result for the entire dataset. e.g. sum, count queries. A linear distributive query function can be computed as linear function of the results from each subset. e.g avg can be computed by first computing sum and count.

Definition 10. *Absolute count query AC and relative count RC in a multi-dimensional database D are defined as:*

$$AC_{P(x)}(D) = \sum_{x \in D} P(x) \qquad RC_{P(x)}(D) = \frac{\sum_{x \in D} P(x)}{n} \tag{4}$$

where P(x) returns 1 or 0 depending on the predicate.

3.5.2 Multidimensional Partioning Approach

For differentially private histogram release, the data points are partitioned into disjoint subsets based on a set of attributes to create a multi dimensional histogram. The frequencies of each partition is released and the histogram can be used to answer random count queries. A Laplace noise or *perturbation error* is added to each partition by the differential privacy interface. When a query is posed to the histogram, perturbation error is added if the query covers multiple partitions. The result is estimated if a query falls within a partition. It is assumed that all partitions have uniform frequencies thus introducing *approximation error*. Thus the total error added to the query response is perturbation error plus approximation error. However, if the assumption of uniform partition is realized practically, approximation error can be eliminated. If we perform careful uniform partitions to generate the histogram we can minimize the perturbation error and also approximation error.

Cell-based Algorithm A dataset can be partitioned into cells based on the domain of all attributes and then release the count for each cell. Since each cell is disjoint subset of the original dataset, according to definition 5, the algorithm provides ϵ-differential privacy.

kd-tree based Algorithm A kd-tree is a data structure for organizing data points in k-dimensional space. The construction of kd-tree usually start from the root node that covers the entire space. At each subsequent step, the space is divided into two subspaces based on a splitting criteria and value. The algorithm continues splitting until a desired height or number of data points in each space is achieved. The resulting structure is a balanced kd-tree. In the following kd-tree based algorithm the main goal is to generate uniform partitions so that approximation error is minimized. To achieve this a variance like metric H is used with a specified threshold xi_1 as a check condition for continuing partitioning of the dataset until a desired sized partition is achieved.

Definition 11. *Let D_0 be a sub-cube with β cells, then the average count AC_0 is given as, $AC_0 =$*

$\sum_{c_i \in D_0} count(c_i)/\beta$ *where* c_i *is each cell in* D_0*. The variance like metric* H *is defined as:*

$$H(D_0) = \sum_{c_i \in D_0} |count(c_i) - AC_0| \tag{5}$$

If $H > \xi_1$*. where* ξ_1 *is a threshold for* H*, then stop partitioning.*

The *kd*-tree algorithm is a two step process each using ϵ differential privacy budget. The*kd*-tree algorithm first generates a 'sanitized' database from the cell-based algorithm. The algorithm then recursively partitions the 'sanitized' dataset using *kd*-tree partitioning technique and uses the resulting keys to partition the original database. The original database is not queried while performing *kd* partitioning instead it uses an approximation of the original database, thus saving the privacy budget.

Note: section 4.5.1 has been adapted from [9].

4 Contributions

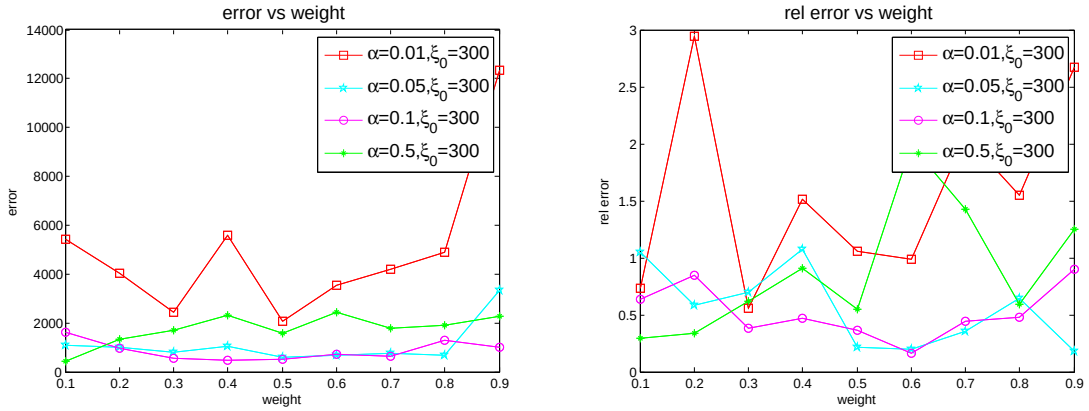
The utility of the *kd*-tree based partitioning algorithm is directly decided by the parameter ξ_0, ξ_1 and data distribution. The overall privacy guarantee offered by the algorithm is based on the privacy budget used. For experiment purposes, the CENSUS data(http://www.ipsums.org) has been used. The data has 1 million tuples and 4 attributes: Age, Education, Occupation and Income, with domain sizes 79, 14, 23 and 100 respectively. The initial experiments on the *kd*-tree based partitioning algorithm to perform absolute count and relative count had the following specifics for its parameters:

parameter	value	Summary
α	0.1	Overall privacy budget. Cell-based partitioning uses $\alpha/2$ privacy budget and *kd*-tree partitioning uses $\alpha/2$ privacy budget.
ξ_1	80	Threshold for variance like function H.
ξ_0	[100:900] step 100	Threshold for generating kd tree.
query	1000	Number of random queries.

we tested the algorithm by dividing the overall privacy budget α into α_1 and α_2 budgets, where α_1 is used by the cell-based partitioning and α_2 is used by kd-tree based partitioning. Below are the specifics for each parameter used in our experiments:

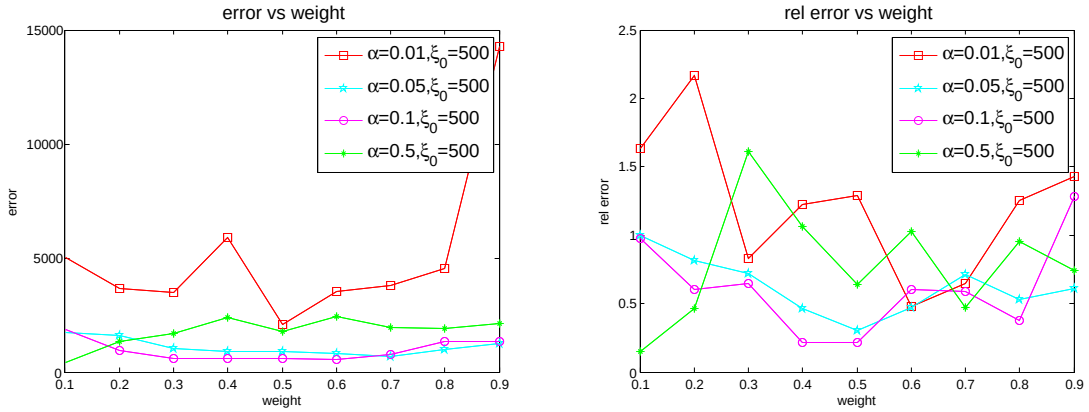
parameter	value	Summary
α	[0.01, 0.05, 0.1, 0.5]	Overall privacy budget.
weight	[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]	Weight is used to determine α_1 and α_2 from α.
α_1	weight $\times\alpha$	Privacy budget for cell-based partitioning.
α_2	$((1 - \text{weight}) \times\alpha)$	Privacy budget for kd-tree based partitioning.
ξ_1	[60, 80, 100, 120]	Threshold for variance like function H.
ξ_0	[100:900] step 100	Threshold for generating kd tree.
query	[1000, 2000, 4000, 6000, 8000]	Number of random queries.

- Query error vs different α
  We analyze the effect of α on the absolute count error and relative count error by varying α. The values of α_1 and α_2 is calculated using *weight* as described in the table above. Figure 1 to 2 show that $\alpha = 0.1$ provides best results. We can see that the value of α significantly affects the query error.



(a) absolute query error for different α when ξ_0=300 (b) relative query error for different α when ξ_0=300

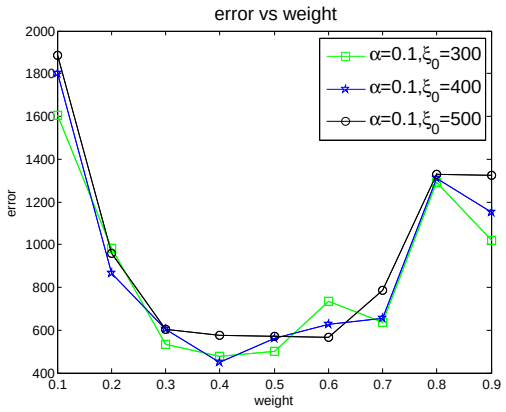
Figure 1: query error for different α when ξ_0=300



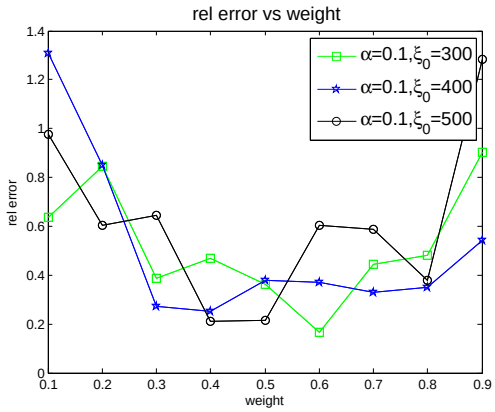
(a) absolute query error for different α when ξ_0=500 (b) relative query error for different α when ξ_0=500

Figure 2: query error for different α when ξ_0=500

- Query error vs different ξ_0
  We choose a fixed value of α and analyzed the results on query error by varying the value for threshold ξ_0 used to generate *kd*-tree. Figure 3 and 4 show the results

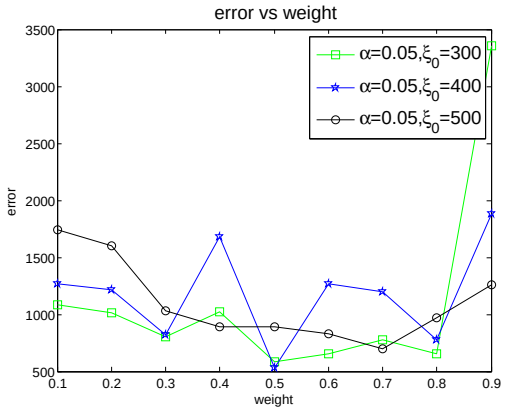


(a) absolute query error for different ξ_0 when $\alpha = 0.1$

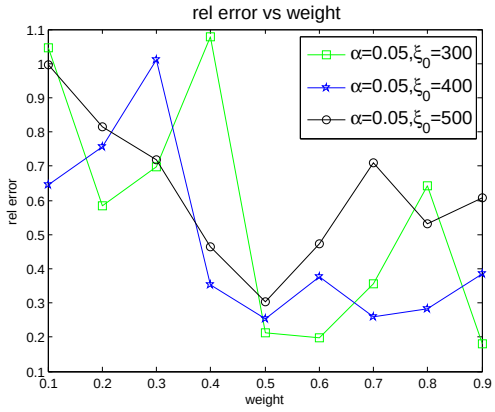


(b) relative query error for different ξ_0 when $\alpha = 0.1$

Figure 3: query error for different ξ_0 when $\alpha = 0.1$



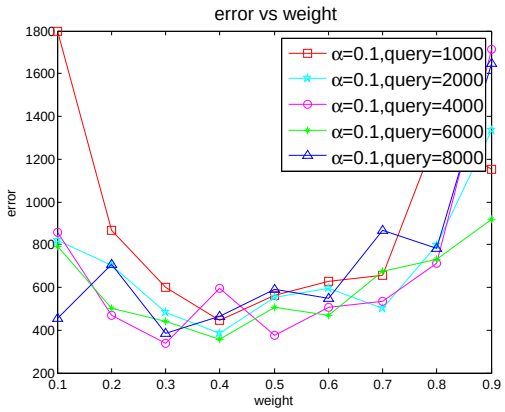
(a) absolute query error for different ξ_0 when $\alpha = 0.05$



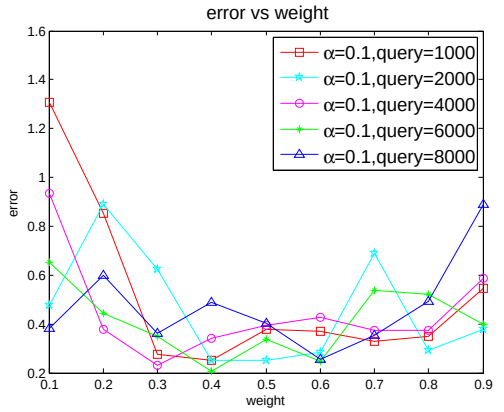
(b) relative query error for different ξ_0 when $\alpha = 0.05$

Figure 4: query error for different ξ_0 when $\alpha = 0.05$

- Query error vs Number of queries
  In the initial setting for experiments, we choose 1000 random queries to analyze the effects of varying α or ξ_0. For this part, we fixed the value of α and ξ_0 and varied the number of random queries generated. Figure 5 and 6 show that the query error is significantly affected by the number of random queries. For $\alpha = 0.1$ and ξ_0, 4000 random queries gives the best result.

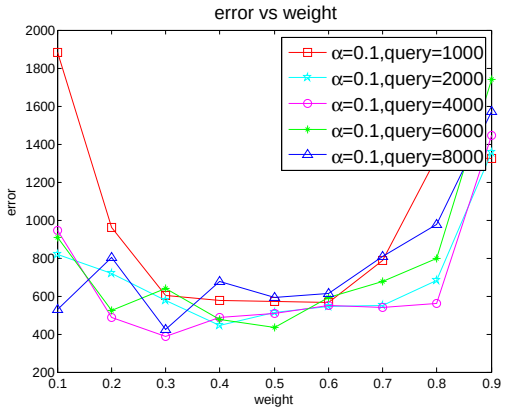


(a) absolute query error for varying number of random queries when $\alpha = 0.1$ and $\xi_0 = 400$

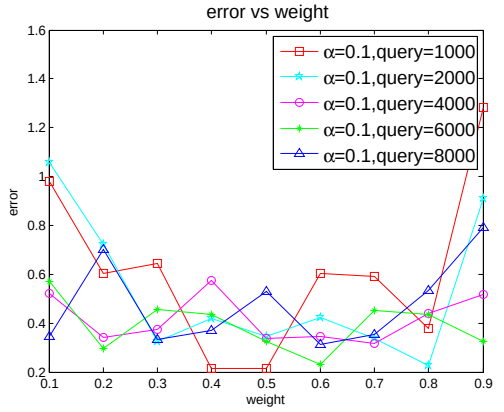


(b) relative query error for varying number of random queries when $\alpha = 0.1$ and $\xi_0 = 400$

Figure 5: query error for varying number of random queries when $\alpha = 0.1$ and $\xi_0 = 400$



(a) absolute query error for varying number of random queries when $\alpha = 0.1$ and $\xi_0 = 500$

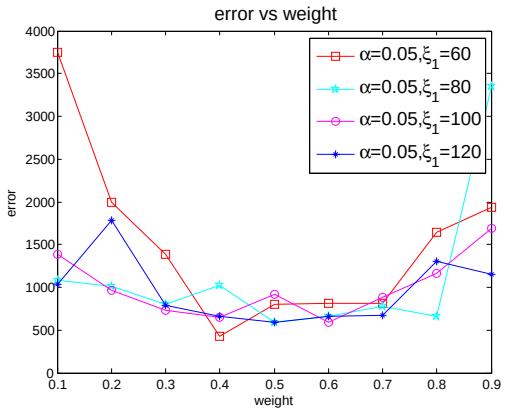


(b) relative query error for varying number of random queries when $\alpha = 0.1$ and $\xi_0 = 500$

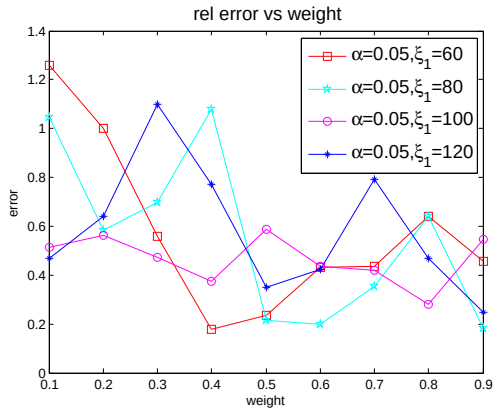
Figure 6: query error for varying number of random queries when $\alpha = 0.1$ and $\xi_0 = 500$

itemQuery error vs different ξ_1

We choose a fixed value of α and ξ_0 and analyzed the results on query error by varying the value for threshold ξ_1 used a threshold for variance like metric H. Figure 7 and 8 show that for different α the behaviour of a particular ξ_1 is dependent on α. Figures 7 to 10 show the results.

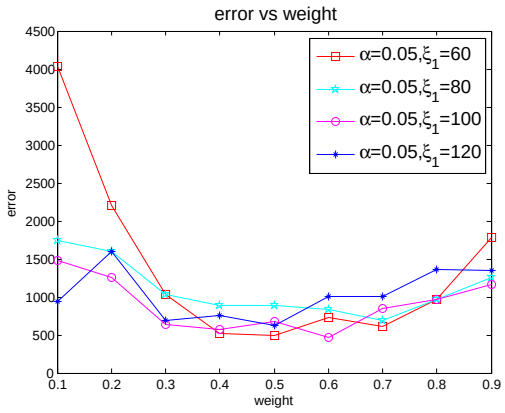


(a) absolute query error for different ξ_1 when ξ_0=300 and $\alpha = 0.05$

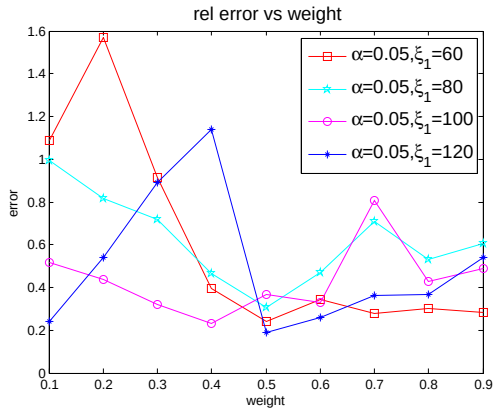


(b) relative query error for different ξ_1 when ξ_0=300 and $\alpha = 0.05$

Figure 7: query error for different ξ_1 when ξ_0=300 and $\alpha = 0.05$

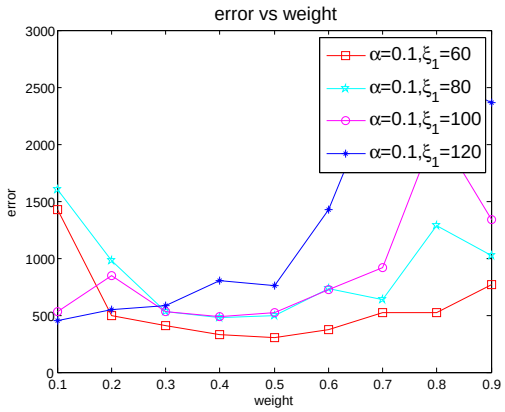


(a) absolute query error for different ξ_1 when ξ_0=500 and $\alpha = 0.05$

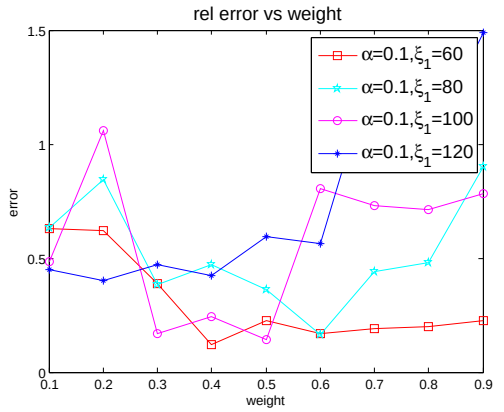


(b) relative query error for different ξ_1 when ξ_0=500 and $\alpha = 0.05$

Figure 8: query error for different ξ_1 when ξ_0=500 and $\alpha = 0.05$

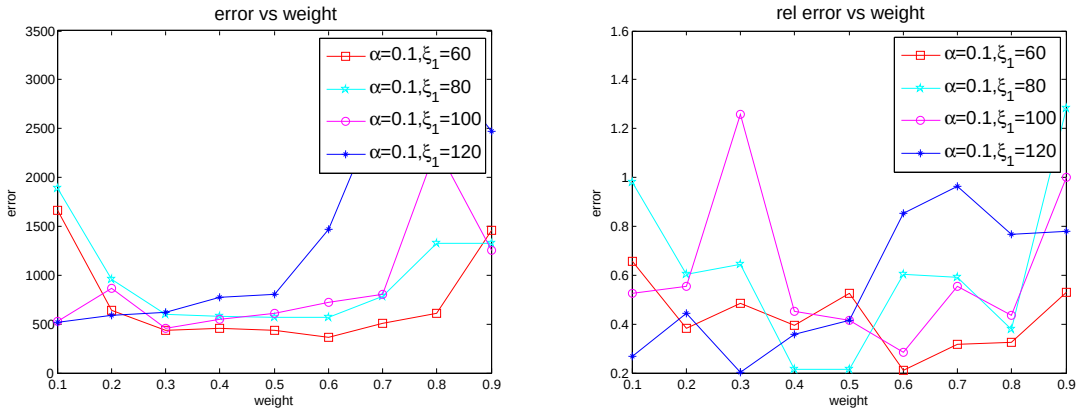


(a) absolute query error for different ξ_1 when ξ_0=300 and $\alpha = 0.1$



(b) relative query error for different ξ_1 when ξ_0=300 and $\alpha = 0.1$

Figure 9: query error for different ξ_1 when ξ_0=300 and $\alpha = 0.1$



(a) absolute query error for different ξ_1 when ξ_0=500 and $\alpha = 0.1$

(b) relative query error for different ξ_1 when ξ_0=500 and $\alpha = 0.1$

Figure 10: query error for different ξ_1 when ξ_0=500 and $\alpha = 0.1$

5 HIDE

So far we have seen techniques to address privacy issues when data is organized in form of a table or structured data. However this is not always the case because large amount of personal data is in the form of unstructured text. e.g. most medical notes, lab reports are unstructured text and contain personal identifiable information about individuals. Sharing of such unstructured data is as important and valued as releasing of statistical databases. Here too it is important to protect the privacy of individuals before releasing the data. Personal identifiable information is protected under the Health Insurance Portability and Accountability Act (HIPAA). HIDE: An Integrated System for Health Information DE-identification is a framework developed to anonymize both structured and unstructured data.

5.1 De-identification Models

Currently HIDE provides three de-identification models to address the issue of data privacy. Information that can be used explicitly to identify a person is termed as Personal Health Information(PHI) by HIPAA. Direct identifiers such as name, social security, medical record number and indirect identifiers such as age, gender etc all count as PHI. It is mandatory for any data provider to remove PHI from the data before releasing it to public.

- Full de-identification
  In full de-identification model all HIPAA identifiers are removed. Since most of the valuable information is removed from dataset the resulting data provides minimum data utility.

- Partial de-identification
  In partial de-identification only certain number of HIPAA identifiers are removed thus resulting in better data utility from the release data.

- Statistical de-identification
  Statistical de-identification model attempts to provide maximum data utility while guaranteeing a statistically acceptable data privacy.

5.2 Conceptual Framework

The HIDE framework consists of three important components.

- Attribute Extraction
  HIDE takes heterogeneous data as input, which is fed to the attribute extraction component where personal identifiable information is tagged using statistical learning. A conditional random field based named entity recognizer is used to label and tag sensitive information. A CRF is trained on sample data files after which it automizes the process of labeling sensitive information thus avoiding manual labeling. Once all the sensitive information is labeled and linked it creates an identifier view.

- Data linking
  Once sensitive information is labeled it is fed to the data linking component where records belonging to same individual are linked together. Once the linking is done the linked data is fed back into the attribute extraction component to check for sensitive attributes. This iterative step is unique feature of the HIDE framework.

- Anonymization
  Once identifier view is fed to the anonymization component, different privacy models are used to obtain full de-identification, partial de-identification of statistical de-identification by attribute removal or suppression. Anonymization techniques based on attribute generalization which guarantees data privacy based on privacy principles like k-anonymity, l-diversity can be used.

5.3 Future Works

At present anonymization techniques based on k-anonymity and l-diversity through attribute generalization are used to perform anonymization and thus guarantee privacy. Future works on HIDE framework is to incorporate differential private mechanism in the Anonymization component to guarantee stronger privacy for released information.

Note: Section 5 has been adapted from [3, 4, 5].

6 Conclusion

We studied the importance of preserving privacy of individuals in a statistical release of a dataset by reviewing weak and strong privacy preserving mechanism. Starting with k-anonymity we learned several mechanisms to preserve privacy and eventually builded our study to differential privacy which is considered to be a powerful privacy preserving mechanism available at present. Differential privacy is prefered because it guarantees privacy of individual regardless of whether an individual participates in a database or not. This aprroach will encourage more individuals to take part in databases that can later be used by researchers and analysts to better understand ways of human society and thus help towards betterment of commnunity at large. For, the differentially private histogram release algorithm we can add the Avg aggregate query in the implementation to increase the utlility of the algorithm. By adding the Avg aggregate query we will be able to get a better understanding of the distribution in original database as well as we can compare the results of Avg query on different datasets and thus rate the usefulness of each dataset. For future works, I am interested in learning and understanding methods and techniques that will introduce differential

private mechanism in the HIDE framework which will make it more robust and powerful and thus act as an interactive/non-interactive interface which guranatees differential privacy.

7 Acknowledgements

We thank the CRA-W/CDC program Distributed Research Experience Undergraduates (DREU) for providing this opportunity to participate in Summer 2010, DREU.

References

[1] C. Dwork. A firm foundation for private data analysis. to appear.

[2] C. Dwork. Differential privacy: A survey of results. In M. Agrawal, D.-Z. Du, Z. Duan, and A. Li, editors, *Theory and Applications of Models of Computation TAMC*, volume 4978 of *Lecture Notes in Computer Science*, pages 1–19. Springer Verlag, 2008.

[3] J. Gardner and L. Xiong. Hide: An integrated system for health information de-identification. 21st IEEE International Symposium on Computer-Based Medical Systems (CBMS), June 2008.

[4] J. Gardner and L. Xiong. An integrated framework for de-identifying unstructured medical data. *Data and Knowledge Engineering*, 68:1441–1451, 2009.

[5] J. Gardner, L. Xiong, K. Li, and J. J. Lu. Hide: Heterogeneous information de-identification (demo track). 12th International Conference on Extending Database Technology (EDBT), March 2009.

[6] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. 22nd International Conference on Data Engineering (ICDE), 2006.

[7] F. McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. New York, NY, USA, 2009. 35th SIGMOD international conference on Management of data, ACM.

[8] L. Sweeney. k anonymity: A model for protecting privacy. *International Journal on Uncertainty Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

[9] Y. Xiao, L. Xiong, and C. Yuan. Differentially private data release through mutlidimensional partitioning. In *Secure Data Management*, volume 6358 of *Lecture Notes in Computer Science*. 7th VLDB Workshop, Springer Verlag, 2010.