Visualization Error Metrics of Differentially Private Histograms

Diane Tam

College of Information and Computer Sciences

dztam@umass.edu

May 2017

Advisor: Gerome Miklau

Second Committee Member: Dan Zhang

Research Type: Thesis

1. Introduction

Differential privacy is a property first proposed in Cynthia Dwork's ICALP paper that in a nutshell ensures a query on two data sets return (nearly) identical results given that one data set includes a specific individual's information and the other does not. The Laplace mechanism is a baseline method commonly applied to ensure this property holds at some specified epsilon level of privacy. Maintaining differentially private data today has become a standard in analysis of sensitive data, with the main question being how do we do this? Much work has gone into developing algorithms that add mathematical noise to a sample data set in order to preserve individual privacy. While more recent research has focused on developing better differential privacy algorithms that can handle real-world data, another problem that comes into play is the visualization of data after this noise has been added. Traditional means of data visualization expose too much information about the data set and does not maintain differential privacy, while applying current differentially private algorithms to data sets before visualizing them often adds too much noise that renders the visualization useless for some certain tasks.

The first half of this two-semester thesis involved gathering knowledge on the concepts of differential privacy and visualization techniques, as well as the DPBench code base developed by Gerome Miklau's team. A working environment was configured in order to be able to use a range of pre-implemented DPBench differentially private algorithms to plot noisy histogram data. Experiments were run with varying parameters such that visualizations for different levels of epsilon differential privacy at different scales and shapes were comparable between algorithms. It was noted that certain algorithms have properties that may alter the effectiveness of the visualization in a broad sense. However, the formalization and application of certain properties of these differential privacy algorithms to different types of tasks has largely been unexplored. There is no such all-purpose differentially private algorithm that is perfect for all visualization tasks, and it becomes difficult to quantify or state which algorithm is "better-suited" for a task. In prior research, error metrics for the analysis of differentially private algorithms are based on normalized vector distances of answers to a given workload set of queries. However, the same error metric cannot be used to analyze a differential privacy algorithm for its visual effectiveness, as the definition of effectiveness will vary greatly among different tasks. As the experiments performed during the first semester have shown, there are many visual properties that are maintained in some algorithms but lost in others. Other visual artifacts (anomalies) that are outputted due to the added noise from differentially private algorithms often render a visualization useless for completing a task.

The research problem that the second part of this thesis project will focus on is a formal definition of a set of error metrics that can be used for differentially private visualization purposes. DPBench's current workload-based analysis of the differentially private algorithms measures the effectiveness of an algorithm in terms of how well the noisy data answers a workload, or a given set of queries. Some algorithms are designed to be workload aware, and thus dynamically alter their shapes and distribution depending on the workload for a lower "L2- error" metric score. This method of evaluation fails to capture the importance of maintaining certain visual aspects of the data which are evaluated on the noisy data itself. However, the issue arises in that there is also no ideal error metric that can be defined to capture all possible visualization tasks. Certain tasks will require a different type of differential privacy algorithm to be used such that the visual artifacts produced do not affect the goal of the task in hand. Thus, this work to attempt to formally define a set of visual error metrics and their respective tasks in order to confidently state the ideal differentially private algorithm to use in different task-based scenarios would work to the advancement of knowledge in the discipline of data visualization and differential privacy.

2. Background

There have been many studies focused on developing better algorithms for differential privacy since its first proposal. There are many algorithms that are fully developed and have been proven to work well enough in answering query workloads that differential privacy is now becoming a mainstream expectation when handling sensitive data. For instance, Apple has already begun using differential privacy techniques starting with their iOS 10 release in order to collect data on consumer usage patterns in order to apply machine learning techniques to aid query suggestions.

The basics of differential privacy can be explained as the chances of whether a noisy released data result is nearly the same as the original data with or without some individual's information. This is formally defined as ε-differential privacy where for any $|D_{\pm i} - D| < 1 \text{ and } C \in Range(M) : \frac{Pr(M(D) = C}{Pr(M(D_{\pm i}) = C} < e^{\varepsilon}$. Epsilon is an empirically set parameter that controls the level and strictness of privacy for the noisy output. There are different ways of introducing mathematical noise to the data that ensures ε-differential privacy, but the base method involves the Laplace Mechanism which is based on choosing noise from the Laplace distribution $p(z) = exp(-|z|/b)/2b$, with a parameter b (denoted $[Lap(b)]$) that flattens the curve as the value increases. The sensitivity of a function essentially captures how much one person's data can affect the output and is represented by $\Delta f = max_{adjacent\ x,x'} |f(x) - f(x')|$. Putting these together we get the Laplace Mechanism where on any query f, in order to achieve ε-differential privacy, a scaled symmetric noise of Lap(b), where $b = \Delta f/\varepsilon$ on the data. This way, the noise that is added will always depend solely on f and ε, rather than the data set.

Carrying these concepts over to the DPBench code base, many of the pre-existing algorithms already implement the Laplace mechanism in order to add noise to data. DPBench also implements algorithms that take into account the workload set of queries that might be asked in order to dynamically change shape and improve error analysis ratings of that algorithm. The L1, L2, and Linf errors are then based off of the solution vectors to such workload queries and their normalized distance from the true solution vectors based on the original data. This type of error metric does not hold for analysis of the algorithms for visualization purposes according to the first semester's research and experimentation. It is also insufficient to simply judge private data by the naked eye simply for patterns and "visual utility", defined informally as "perceived visual similarity to the true data". In reality, the tasks that a data analyst must perform given such noisy data will require different properties that various differentially private algorithms will preserve.

The idea of task abstraction becomes very important as we dive into why a visualization might be used. In Chapter 3 of *Visualization Analysis & Design,* Munzner explains the actions and targets that may differ with each task and consequently why they all need slightly different visualizations. At the highest level, we must ask whether a user is trying to consume or produce new information from this visualization. A few key readings also proposed ideas of a task topology that groups tasks based on their time and space dimensionality such that each level is asking a fundamentally different question from the data. It follows that a visualization should be designed as data and a task. Visualization evaluation on the other hand says that the combination of data and visualizations together create a task. Clearly, it is impossible to declare one differentially private algorithm that will produce an all-encompassing visualization for every goal, so this thesis will attempt to begin to formulate new error metrics for task-based visualization analysis rankings based on a subset of differentially private algorithms in DPBench.

3. Methodology

The datasets and algorithms used for the first semester's experiments come from the DPBench code base. Each experiment involved plotting noisy histogram outputs using python plotting libraries including matplotlib1.4.3 and seaborn. The code was written and run in an iPython notebook on the UMass CS dbcluster.

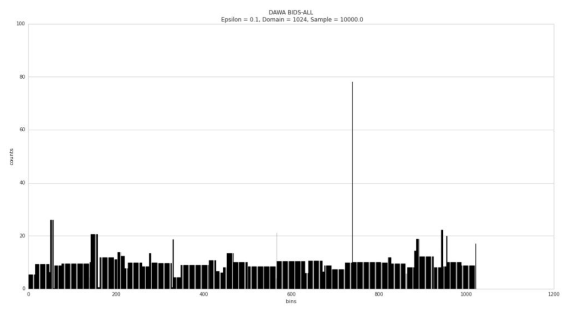
In terms of set-up, all experiments were run on two datasets, BIDS-ALL and HEPTH, with focus on the H_b, Identity, MWEM, and DAWA algorithms (the latter two being workload-aware). The workloads used are Identity and Prefix1D. The experiments were also tested on two different domains: 4096 and 1024. The general workflow for each experiment included first plotting the original histogram and cumulative density function of the data and then choosing a scale-epsilon pairing for the data generator with a domain of either 4096 or 1024 and then plotting the noisy histogram and cumulative density function outputted by the H_b and Identity algorithms, followed by the same plots for MWEM and DAWA under each of the two workloads. Thus, each experiment under a domain for a choice of scale-epsilon generates 6 noisy plots for visual analysis compared to the original.

For algorithms that produced negative outputs, the noisy data was cleaned up during post-processing via a normalized non-negative rounding by multiplying each positive bin count by a global weight of the [original noisy data sum]/[sum without negative values]. The negative bin counts are then rounded to 0. This method aims to maintain the overall distribution of the plot, while keeping the sum constant. However, while this worked well in terms of naked eye judgement of the visual utility for the histogram plot itself, there are some nuances in the cumulative distribution plot which are affected by this post-processing method. There are also clear artifacts of negative bin counts if the scale becomes too low, or epsilon is set too high. In an application sense, the negative outputs are meaningless in a histogram of data counts that might be used for a consumer task involving, say, statistical analysis, and this is a visual artifact of the differential privacy algorithm.

The axis scales of a plot are another source of a less obvious visual artifact that arises from the noisy algorithm histogram output. In the case of the non-post-processed noisy H_b and Identity plots, or MWEM, the original axis y-scales do not fit the new noisy data scales. The H_b and Identity plots have many negative bin counts that cause the maximum y-axis to increase drastically as well in order to maintain a constant sum. MWEM often captures one high outlier bin count and underestimates lower bin counts for the rest of the data. Ideally, if the distribution of the noisy data was more accurate, the axis scales should remain consistent with the original. This is the case with the normalized rounding post-processed noisy plots, but we cannot test this for MWEM because of the inherent workload-aware algorithm design.

When using two different versions of seaborn and matplotlib python libraries to plot these noisy histograms, some vast differences in the visual appearance of the original plots seemed to appear. The maximum bins seem to vary slightly between a plot of the original HEPTH dataset generated with a domain = 4096 and sample = 10^6. Therefore, in order to accurately compare noisy histograms to their original counterpart, each plot must be compared to the original plot that was produced by the same version of matplotlib. For this paper, all graphs are plotted with matplotlib version 1.4.3 unless otherwise noted.

Declaring whether or not scale-epsilon exchangeability holds also becomes a very subjective question when referencing the visual utility of the data. Theoretically, changing the scale-epsilon parameter should present the same plot visually, scaled down or up by the correct magnitude. In an experiment comparing the pair (A) $\varepsilon = 0.1$, $sample = 10^4$ and (B) $\varepsilon = 0.001$, $sample = 10^6$ (note the two are scale-epsilon rank equivalent), we should be able to compare the plots generated by each algorithm under otherwise constant circumstances and find them to be visually exchangeable. However, this is not quite the case. It is arguable that in some cases, they are similar to a certain general degree, but not the the level of detail that would aid someone hoping to do statistical analysis with these plots. For instance, the DAWA plots generated under the Identity workload for the BIDS-ALL dataset appear similar at first glance for overall distribution. However, the y-axis scale of the outliers becomes skewed from experiment (A) to (B). The highest outlier in (Fig. 1) approaches 80, and we would expect the highest outlier in (Fig. 2) to approach 8000, consequently. However, the highest outlier instead passes 10000 in experiment (B). There are also more outliers in experiment (B) that cause a more uniform distribution compared to the drops in experiment (A).



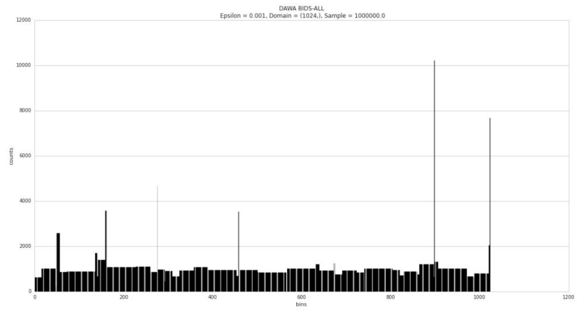
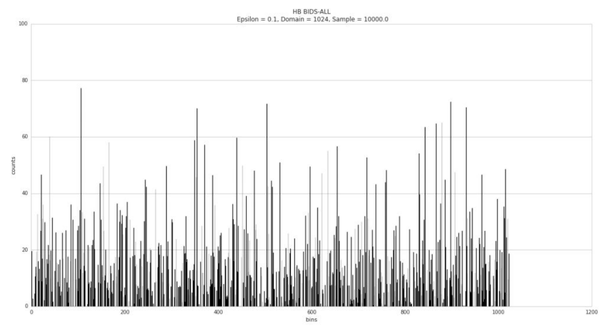


Figure 1: DAWA on BIDS-ALL (A) Figure 2: DAWA on BIDS-ALL (B)

For the same experiments (A) and (B), but instead looking at the H_b plots (post-processed to remove all negative counts), these issues seem to be harder to discern. The peaks of most bins lie between 60 and 80 (Fig. 3) for experiment (A), and as theorized, these lie between 6000 and 8000 for experiment (B) (Fig. 4). The general visual utility of the distribution also seems accurate to the naked eye. Again, this degree of accuracy might depend on the level of statistical analysis to be performed on the histogram.



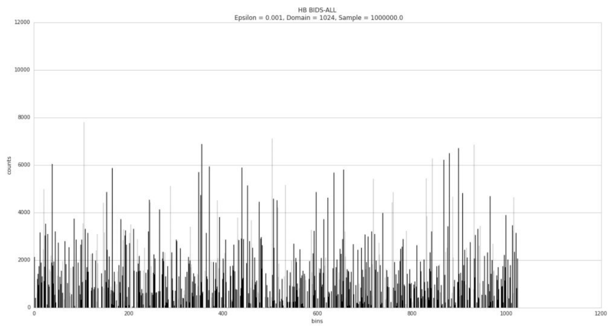


Figure 3: H_b on BIDS-ALL (A) Figure 4: H_b on BIDS-ALL (B)

Although the normalized non-negative rounding worked quite well in maintaining overall visual shape for the original histogram in negative-count generating plots such as H_b and Identity, the plot of the cumulative distribution function suffered visually from this post-processing routine. Take the CDF plot of a non-post-processed H_b noisy output run on the HEPTH dataset $(\varepsilon = 0.001,\ domain = 1024,\ sample = 10^6)$ and its post-processed counterpart run on the same parameters as an example. The green line represented the CDF of the noisy histogram, and the blue represents the CDF of the original histogram. As expected, the noisy CDF still sums up to the original count of 10^6 samples. However, the non-post-processed CDF line (Fig 5) matches the original line more closely than that of the post-processed CDF line (Fig 6). The post-processed CDF line appears to rise much faster than the original until they intersect, where the CDF then slows down to meet the original. The post-processing gets rid of any negative bin counts, and as a result, allows the CDF to grow much faster than original. This could pose an issue if the statistical analysis of the noisy data were not focused on the distributions of the original data, but rather the growth rate or changes within the data. Again, we would want to determine a metric that could capture this error and quantify it against other algorithms for a task involving the CDF.

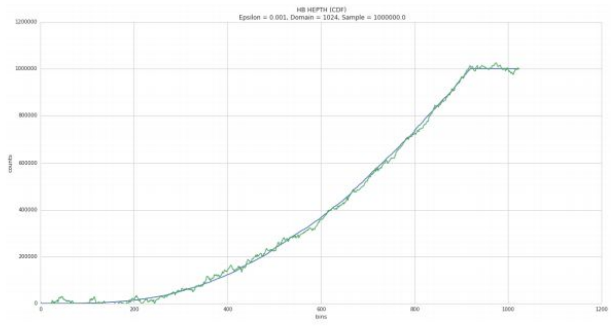


Figure 5: H_b on HEPTH (no post-processing)

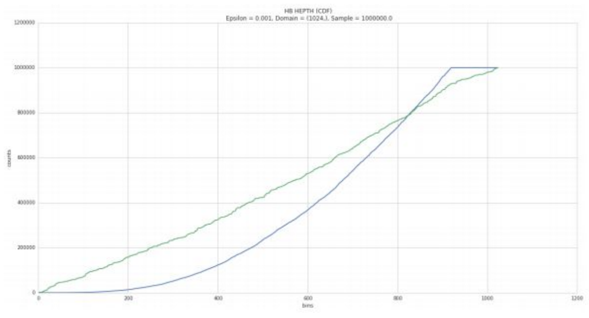


Figure 6: H_b on HEPTH (with post-processing)

4. Evaluation

Milestone 1: Implement robust simple error metrics and integrate with DPBench source code

Design a system in which error metrics can be easily defined and run on different algorithms by changing simple parameters. This code should be robust and used in conjunction with the existing DPBench metrics such that they can be analyzed on the implemented differential privacy algorithms. Implement any simple error metric (ie. bin distance of maximum bar count of original and noisy histogram) as a proof of concept. Also integrate DPComp code for calculating L1, L2, and Linf error which accounts for workload-aware algorithms.

Milestone 2: Present alternative ways of modeling error metrics such that they are comparable across algorithms

Error metrics may not be in comparable units. Is there a way to normalize each error metric so that they are also comparable across scales and domain shapes? (ie. a bin distance error of 1 is a much greater effect for a histogram with only 5 bins versus one with 1000). Is a horizontal bar plot the best way of representing this data? Look into presenting algorithm rankings with line plots instead with varying epsilon or scale. Determine whether or not it is worth quantifying “how much” better one algorithm might be compared to another based on a certain metric, or if we just need to rank them.

Milestone 3: Determine new error metrics and their respective visualization task

Create formal error metrics that can be used to confidently rank algorithms for their utility given a task. Clearly define the properties that the visualization should have or maintain for each task and capture this in the error metric.

Milestone 4: Formulate task topologies and the correct error metric to use for determining the best algorithm

Using defined error metrics, can we organize tasks into a topology that points to specific error metric classes when presented with certain task goals? Or is there no clear answer such that each task needs to be closely analyzed and a new error metric needs to be determined for each individual task at hand?

Milestone 5: Decide how to rank each algorithm for each task

Formalize theories. How exactly is it proved that this error metric successfully ranks the optimal algorithm for a given task. Is there a pattern to this?

5. Communication

Meetings will be run similar to those of the previous semester with both Gerome Miklau and Dan Zhang, the full committee. These will occur on a weekly basis for at least 30 minutes per session. During these meetings, I present the work I have done for that week and clarify any issues or questions I ran into. The committee chair and member reviews this work and determines what the next best possible direction is regarding the project. Between each weekly meeting, I expect to be applying about 10 hours of time to the thesis with coding, reading materials, writing up current findings, etc.

6. Timeline

Meeting 1 | Sept 5- 8

Milestone 1: Implement robust simple error metrics and integrate with DPBench source code

Meeting 2 | Sept 11-15

Continue Milestone 1

Meeting 3 | Sept 18-22

Milestone 2: Present alternative ways of modeling error metrics such that they are comparable across algorithms

Meeting 4 | Sept 25-29

Continue Milestone 2

Meeting 5 | Oct 2-6

Milestone 3: Determine new error metrics and their respective visualization task

Meeting 6 | Oct 10-13

Continue Milestone 3

Meeting 7 | Oct 16-20

Continue Milestone 3

Milestone 4: Formulate task topologies and the correct error metric to use for determining the best algorithm

Meeting 8 | Oct 23-27

Continue Milestone 4

Meeting 9 | Oct 30-Nov 3

Milestone 5: Decide how to rank each algorithm for each task

Meeting 10 | Nov 6-10

Continue Milestone 5

Meeting 11 | Nov 13-17

1st Draft of Thesis submitted to committee chair and advisor, Gerome Miklau

Meeting 12 | Nov 27- Dec 1

2nd Draft of Thesis submitted to Honors Program Director, Ben Marlin

Meeting 13 | Dec 4 - 8

Oral Defense

Meeting 14 | Dec 11-12

Final submission of thesis to CHC

7. References

C. Dwork. A Firm Foundation for Private Data Analysis, Communications of the ACM, 2011.
https://goo.gl/1wsKz4

C. Dwork and A. Roth. The algorithmic foundations of differential privacy. Foundations and Trends in Theoretical Computer Science, 2014.
https://www.cis.upenn.edu/~aaroth/Papers/privacybook.pdf

M. Hay, A. Machanavajjhala, G. Miklau, Y. Chen. D. Zhang. Principled evaluation of differentially private algorithms using dpbench, 2016.
https://people.cs.umass.edu/~miklau/assets/pubs/dp/hay16principled.pdf

D. Zhang, M. Hay, G. Miklau. B. O'Connor. Visualizing differentially private data, 2016.
https://people.cs.umass.edu/~miklau/assets/pubs/viz/zhang16challenges.pdf

T Munzner. Visualization Analysis and Design. AK Peters Visualization Series, 2014.
https://www.crcpress.com/Visualization-Analysis-and-Design/Munzner/p/book/9781466508910

H Schuls, T Nocke, M Heitzler, H. Schumann. A Design Space of Visualization Tasks, 2013.
https://pdfs.semanticscholar.org/b768/fd79df84859a93d397e8ac17c346fada0f59.pdf