Overview for Summer Research Activities in Differential Privacy and Dual Query.

Jason Suagee

July 28, 2016

1 Overview of Differential Privacy and the Dual Query Algorithm

A differentially private de-identification algorithm is an algorithm for releasing original data, perturbed by random noise, in a manner for which there is a well defined mathematical guarantee (in a probabilistic sense) that re-identification of any single record in the data set is impossible. For a precise definition see (Citation).

There exist several families of differentially private algorithms today, broadly falling into two main categories: the interactive paradigm and the ( ). In the interactive paradigm, a trusted source guards access to the original data set, and a database user issues queries to this trusted source and receives responses. These responses are issued according to an algorithm so as to stay within the privacy budget, which is a quantification of de-identifaction risk and for which the trusted source will attempt to not exceed. Typically each query will consume some of the privacy budget, so that the number of queries which a user can ask will necessarily be limited. Algorithms which fall under this paradigm include matrix method algorithms (Citation), and methods based on the Multiplicative Weights mechanism (Citation) such as MWEM (Citation).

The other paradigm involves creation of a synthetic database which as output of the

algorithm can be publically released at no additional loss to privacy. It is the responsibility of the algorithm to enforce that the synthetic database behave similarly to the actual database under the considered family of queries.

Dual Query (DQ) is a differentially private algorithm which falls under the second paradigm, and which will be the subject of these research activities. In particular, we would like to implement DQ in Python (because of the language's acessibility to the general data research public), and use our implementation to evaluate DQ's effectiveness for the purposes of de-identification and differential privacy. We will apply our implementation to a variety of data sets (both real world and synthetic), compare DQ with several other algorithms from the differential privacy literature using methodology in [6], and will attempt to handle a real world case study using publicaly available data from the Dallas Police Department.

2 Types of Data Sets, Queries, and Methods of Algorithm Evaluation

Data sets considered will be viewed as tables, where each row corresponds to a record in the data set and each column corresponds to an attribute. In general attributes can be either discrete or continuous, with the discrete category breaking down into binary and categorical attribute types. The current DQ algorithm depends on the fact that the attributes in the data set are binary, so we will either have to restrict ourselves to data sets with binary attributes, or find suitable ways of converting other classes of data sets to binary representations. Fortunately, both discrete and continuous data types can be represented as sets of binary attributes (bucketing in the case of continuous attributes). It is not clear that there is a best, or canonical, method for converting data sets to binary representations. This is a topic which will have to be considered during the project.

We will obtain our test data sets from three main sources. First, we will use the data sets that are used in the original article describing DQ [4]. These are the KDD99 data, which is network packet data, and "Census Income" data set from the Census department. Both data sets are available from the UCI repository (2).

Our second source will be data sets considered in the DPBench project (dpcomp.org), which include

Finally we will use data sets that have been released for public use by the Dallas Police Department, however, due to the large number of fields in these data sets we will restrict ourselves to considering only a small subset of the available fields.

The DQ algorithm is meant to handle only certain kinds of queries, which we will have to keep in mind in this project. These queries are counting queries, as opposed to queries which either take a numeric domain or range. Appropriate query classes for this study are k-marginals and parity queries, and possibly families of queries designed from more complicated boolean expressions.

Accuracy will be evaluated as described in [4, Pages 16-17], by both the average error and max error over a sampled subset of the allowed queries. For query sets we will allow families of k-marginals and parity queries as described in [1]. We will also report variance of the error, as in [6].

3 Timeline

- **August 1-12** Build the implementation of Dual Query in OCaml given in [1]. This implementation relies on a choice of an embedded integer linear program (ILP) solver (Cite here Wikipedia entry on ILP). The architects of [1] chose to use IBM's proprietary CPLEX software. If we can obtain a license to use CPLEX, then we will first configure the Ocaml implementation to use CPLEX as well as an open source ILP solver. We will then determine if the performance of the OCaml implementation depends significantly on the choice of embedded ILP solver (evaluated based on accuracy of query response and computational efficiency).

- **August 15-26** Implement in Python the Dual Query (DQ) algorithm as given in the orginal article [4]. As mentioned above, a major component of DQ requires an integer linear program solver. As an open source alternative to CPLEX, we plan to use the Branch and Cut solver (CBC) which is part of the Computational Infrastructure

for Operations Research (COIN-OR), a collection of open source software packages intended for use in operations research [3]. CBC can be used as a stand alone program, and there is a Python wrapper for it [8] which we will make use of.

- **Aug. 29-Sept. 9** The Ocaml implementation of DQ in [1] comes with several data sets which were used to evaluate it (Cite Data Set sources). Compare output of the Ocaml implementation with the python implementation of DQ by running a family of queries on these data sets. Each of these data sets is available on the UCI Machine Learning Repository (Citation). For the Census data set a data base of queries is already included with [1]. For the other data sets we must create our own query sets to base our evaluations on. Evaluation will proceed by comparing average difference between error on families of 3-way marginals as outlined in [4, Section 7].

- **Sept. 12-Sept. 23** DPBench is an evaluation platform for differentially private algorithms described in [6]. We will modify the methodology of DPBench in order to evaluate our version of Dual Query against the other algorithms presented in [6]. In particular we must configure DPBench to evaluate algorithms based on k-marginal queries rather than range queries, since DQ is not designed to be used with range queries.

- **Sept. 26-Oct. 7** To evaluate accuracy of our implementation we will look at a data set provided by the Dallas Police Department (DPD) which is publically available (Citation). Specifically, due to the size of this data set, we will concentrate on sub-datasets obtained by disregarding all but 2 or 3 fields, and retaining a very small proportion of the records (perhaps less than 100). All fields must be converted to a binary representation, which may expand the number of fields in the test database significantly. Accuracy will be evaluated as described in [4, Pages 16-17], by both the average error and max error over a sampled subset of the allowed queries. For query sets we will allow families of k-marginals and parity queries as described in [1]. (Also variance for error, as in [6].)

4 Other Research Questions

References

[1] E.J.G. Arias. Dualquery: Practical private query release algorithm. `https://github.com/ejgallego/dualquery`, 2014.

[2] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[3] John Forrest. Coin-or branch and cut library. `https://projects.coin-or.org/svn/Cbc/releases/2.9.8`, 2013–2016.

[4] Marco Gaboardi, Emilio Jesús Gallego Arias, Justin Hsu, Aaron Roth, and Zhiwei Steven Wu. Dual query: Practical private query release for high dimensional data. *CoRR*, abs/1402.1526, 2014.

[5] Moritz Hardt, Katrina Ligett, and Frank McSherry. A simple and practical algorithm for differentially private data release. *CoRR*, abs/1012.4763, 2010.

[6] Michael Hay, Ashwin Machanavajjhala, Gerome Miklau, Yan Chen, and Dan Zhang. Principled evaluation of differentially private algorithms using DPBench. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, pages 139–154, New York, NY, USA, 2016. ACM.

[7] Ilya Mironov. On significance of the least significant bits for differential privacy. ACM, October 2012.

[8] Ted et al. Ralphs. CyLP: A python interface to CLP, CBC, and CGL to solve LPs and MIPs. `https://github.com/coin-or/CyLP`, 2015.