The Complexity of Computing the Optimal Composition of Differential Privacy

Jack Murtagh* and Salil Vadhan**

*Graduate Student, Harvard John A. Paulson School of Engineering and Applied Sciences

**Professor of Computer Science and Applied Mathematics, Harvard John A. Paulson School of Engineering and Applied Sciences

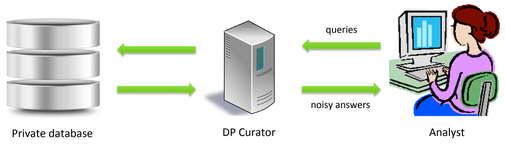
Privacy Tools for Sharing Research Data

A National Science Foundation Secure and Trustworthy Cyberspace Project

with additional support from the Sloan Foundation and Google, Inc.

Differential Privacy

- Promising avenue for answering statistical queries on sensitive datasets while minimizing privacy risks for individuals in the dataset.



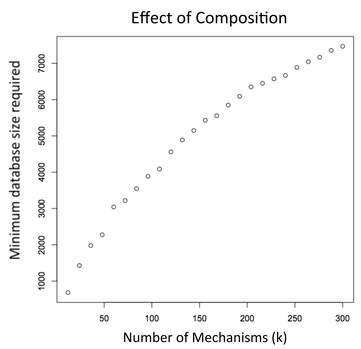
- An algorithm M is said to be (ε,δ)-Differentially Private if for all neighboring databases D, D' and all output sets S:

$$\Pr[M(D) \in S] \leq e^{\varepsilon} \Pr[M(D') \in S] + \delta$$

- Central Idea: A private algorithm should behave very similarly on a database with your data as it would on the **same** database **without** your data. This way, an adversary cannot distinguish the two cases.

Composition

- The more DP algorithms we run on a single database, the more privacy degrades.



- Simple composition: If we run k (ε,δ)-DP mechanisms on the same database, we maintain a global privacy of (kε, kδ).
- But we want to compute many statistics on our dataset!
- We can do better. Quantifying privacy degradation under composition as precisely as possible is important for the practice and theory of DP.

Optimal Composition: Special Case

- Recent result [3] characterized optimal composition exactly:

Theorem ([KOV15]). *For every $\epsilon \geq 0$ and $\delta \in [0,1)$, the class of (ϵ,δ)-DP algorithms satisfy $((k-2i)\cdot\epsilon, \delta_g)$-DP, where i is the largest integer in $\{0,1,\ldots,\lfloor k/2 \rfloor\}$ such that*

$$\frac{\sum_{l=0}^{i-1} \binom{k}{l} \left(e^{(k-l)\epsilon} - e^{(k-2i+l)\epsilon}\right)}{(1+e^{\epsilon})^k} \leq 1 - \frac{1-\delta_g}{(1-\delta)^k}$$

- Closed form, efficiently computable solution.
- Problem: This result only applies in the 'homogenous' case i.e. when every mechanism in the composition has identical privacy parameters.
- In practice, we often want to run algorithms with different privacy parameters on the same database.

Optimal Composition: General Case

- Characterization of the optimal composition of arbitrary differentially private algorithms.

Theorem. *For all $\epsilon_1,\ldots,\epsilon_k \geq 0$ and $\delta_1,\ldots,\delta_k,\delta_g \in [0,1)$, the optimal composition of $(\epsilon_1,\delta_1),\ldots,(\epsilon_k,\delta_k)$-DP algorithms is (ϵ_g,δ_g)-DP where ϵ_g is the smallest number satisfying*

$$\frac{1}{\prod_{i=1}^{k}(1+e^{\epsilon_i})} \sum_{S\subseteq\{1,\ldots,k\}} \max\left\{e^{\sum_{i\in S}\epsilon_i} - e^{\epsilon_g}\cdot e^{\sum_{i\notin S}\epsilon_i}, 0\right\} \leq 1 - \frac{1-\delta_g}{\prod_{i=1}^{k}(1-\delta_i)}$$

- Can compute in time exponential in k by summing over subsets S by brute force. In practice this is not satisfactory.

Question: Is there an efficient algorithm for computing optimal composition exactly?

Answer: Likely no.

Complexity

Theorem. *Computing* OptComp *is #P-complete, even on instances where $\delta_1 = \delta_2 = \ldots = \delta_k = 0$ and $\Sigma_{i\in[k]}\epsilon_i \leq \epsilon$ for any desired constant $\epsilon > 0$.*

- Efficient algorithm does not exist unless #P=FP.
- Problem turns out to be very closely related to counting solutions to partition and knapsack-type problems, known to be #P-complete [2].

Approximation Algorithm

- Idea: If we can't compute optimal composition exactly, maybe we can approximate it.

Theorem. *There is a polynomial-time algorithm that given $\epsilon_1,\ldots,\epsilon_k \geq 0, \delta_1,\ldots\delta_k,\delta_g \in [0,1)$, and $\eta > 0$, outputs (ϵ^*,δ^*) where $\epsilon_g \leq \epsilon^* \leq \epsilon_g + \eta$ and $\delta_g \leq \delta^* \leq e^{-\eta/2}\cdot\delta_g$. The algorithm runs in $O\left(\log\left(\frac{k}{\eta}\Sigma_{i=1}^{k}\epsilon_i\right)\frac{k^2}{\eta}\Sigma_{i=1}^{k}\epsilon_i\right)$ time assuming constant-time arithmetic operations.*

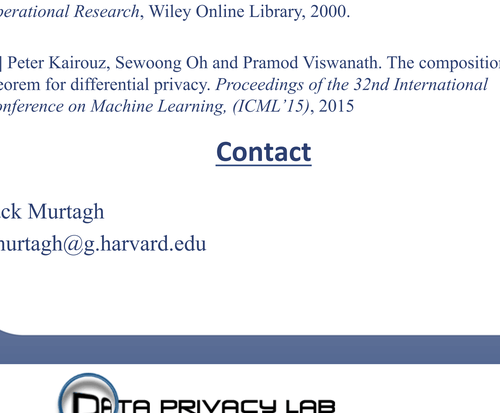
- There exists a polynomial time algorithm based on dynamic programming for approximately counting knapsack solutions [1]
- Note that we can tolerate a polynomial dependence on 1/η because we typically think of ε as a non-negligible constant.

Algorithm Ideas

1. A multiplicative variation of counting knapsack solutions can be solved exactly in pseudo-polynomial time
2. Discretize ε values, creating a slightly different input set
3. Compute optimal composition on discretized ε values using pseudo-polynomial time algorithm
4. Prove that the privacy guarantee of the rounded set of ε values is not too far from the privacy guarantee of the original input

Conclusions

- Since computing optimal composition is #P-complete, an algorithm that approximates it to arbitrary precision in polynomial time is essentially the best we can hope for.
- Algorithm outperforms bound provided in [3] in practice:



Future Work

- Quantify gap in image above.
- Characterize optimal composition when adversary may adaptively choose privacy parameters.
- Remove multiplicative error on δ in approximation algorithm.

References

[1] Martin Dyer. Approximate counting by dynamic programming. *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*. ACM, 2003.

[2] Matthias Ehrgott. Approximation algorithms for combinatorial multicriteria optimization problems. *International Transactions in Operational Research*, Wiley Online Library, 2000.

[3] Peter Kairouz, Sewoong Oh and Pramod Viswanath. The composition theorem for differential privacy. *Proceedings of the 32nd International Conference on Machine Learning, (ICML'15)*, 2015

Contact

Jack Murtagh

jmurtagh@g.harvard.edu