How reparametrization trick broke differentially-private text representation learning

Ivan Habernal
Trustworthy Human Language Technologies
Department of Computer Science
Technical University of Darmstadt
ivan.habernal@tu-darmstadt.de
www.trusthlt.org

Abstract

As privacy gains traction in the NLP community, researchers have started adopting various approaches to privacy-preserving methods. One of the favorite privacy frameworks, differential privacy (DP), is perhaps the most compelling thanks to its fundamental theoretical guarantees. Despite the apparent simplicity of the general concept of differential privacy, it seems non-trivial to get it right when applying it to NLP. In this short paper, we formally analyze several recent NLP papers proposing text representation learning using DPText (Beigi et al., 2019a,b; Alnasser et al., 2021; Beigi et al., 2021) and reveal their false claims of being differentially private. Furthermore, we also show a simple yet general empirical sanity check to determine whether a given implementation of a DP mechanism almost certainly violates the privacy loss guarantees. Our main goal is to raise awareness and help the community understand potential pitfalls of applying differential privacy to text representation learning.

1 Introduction

Differential privacy (DP), a formal mathematical treatment of privacy protection, is making its way to NLP (Igamberdiev and Habernal, 2021; Senge et al., 2021). Unlike other approaches to protect privacy of individuals' text documents, such as redacting named entities (Lison et al., 2021) or learning text representation with a GAN attacker (Li et al., 2018), DP has the advantage of *quantifying* and *guaranteeing* how much privacy can be lost in the worst case. However, as Habernal (2021) showed, adapting DP mechanisms to NLP properly is a non-trivial task.

Representation learning with protecting privacy in an end-to-end fashion has been recently proposed in DPText (Beigi et al., 2019b,a; Alnasser et al., 2021). DPText consists of an auto-encoder for text representation, a differential-privacy-based noise adder, and private attribute discriminators, among others. The latent text representation is claimed to be differentially private and thus can be shared with data consumers for a given down-stream task. Unlike using a predetermined privacy budget ε, DPText takes ε as a learnable parameter and utilizes the reparametrization trick (Kingma and Welling, 2014) for random sampling. However, the downstream task results look too good to be true for such low ε values. We thus asked whether DPText is really differentially private.

This paper makes two important contributions to the community. First, we formally analyze the heart of DPText and prove that the employed reparametrization trick based on inverse continuous density function in DPText is wrong and the model violates the DP guarantees. This shows that extreme care should be taken when implementing DP algorithms in end-to-end differentiable deep neural networks. Second, we propose an empirical sanity check which simulates the actual privacy loss on a carefully crafted dataset and a reconstruction attack. This supports our theoretical analysis of non-privacy of DPText and also confirms previous findings of breaking privacy of another system ADePT.[1]

2 Differential privacy primer

Suppose we have a dataset (database) where each element belongs to an individual, for example Alice, Bob, Charlie, up to m. Each person's entry, denoted with a generic variable x, could be an arbitrary object, but for simplicity consider it a real valued vector $x \in \mathbb{R}^k$. An important premise is that this vector contains some sensitive information we aim to protect, for example an income ($x \in \mathbb{R}$), a binary value whether or not the person

[1] ADePT is a text-to-text rewriting system claimed to be differentially private (Krishna et al., 2021) but has been found to be DP-violating (Habernal, 2021).

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 2: Short Papers, pages 771 - 777

has a certain disease ($x \in \{0.0, 1.0\}$), or a dense representation from SentenceBERT containing the person's latest medical record ($x \in \mathbb{R}^k$). This dataset is held by someone we trust to protect the information, the trusted curator.[2]

This dataset is a set from which we can create 2^m subsets, for instance $X_1 = \{\text{Alice}\}$, $X_2 = \{\text{Alice}, \text{Bob}\}$, etc. All these subsets form a *universe* $\mathcal{X}$, that is $X_1, X_2, \cdots \in \mathcal{X}$, and each of them is also called (a bit ambiguously) a dataset.

Definition 2.1. *Any two datasets $X, X' \in \mathcal{X}$ are called* neighboring*, if they differ in one person.*

For example, $X = \{\text{Alice}\}, X' = \{\text{Bob}\}$ or $X = \{\text{Alice}, \text{Bob}\}, X' = \{\text{Bob}\}$ are neighboring, while $X = \{\text{Alice}\}, X' = \{\text{Alice, Bob, Charlie}\}$ are not.

Definition 2.2. Numeric query *is any function f applied to a dataset X and outputting a real-valued vector, formally $f : X \to \mathbb{R}^k$.*

For example, numeric queries might return an average income ($f \to \mathbb{R}$), number of persons in the database ($f \to \mathbb{R}$), or a textual summary of medical records of all persons in the database represented as a dense vector ($f \to \mathbb{R}^k$). The query is simply something we want to learn from the dataset. A query might be also an identity function that just 'copies' the input, e.g., $f(X = \{(1,0)\}) \to (1,0)$ for a real-valued dataset $X = \{(1,0)\}$.

An attacker who knows everything about Bob, Charlie, and others would be able to reveal Alice's private information by querying the dataset and combining it with what they know already. Differentially private algorithm (or mechanism) $\mathcal{M}(X; f)$ thus randomly modifies the query output in order to minimize and quantify such attacks. Smith and Ullman (2021) formulate the principle of differential privacy as follows: *"No matter what they know ahead of time, an attacker seeing the output of a differentially private algorithm would draw (almost) the same conclusions about Alice whether or not her data were used."*

Let a DP-mechanism $\mathcal{M}(X; f)$ have an arbitrary range $\mathcal{R}$ (a generalization of our case of numeric queries, for which we would have $\mathcal{R} = \mathbb{R}^k$). Differential privacy is then defined as

$$\frac{\Pr(X|\mathcal{M}(X;f) = z)}{\Pr(X'|\mathcal{M}(X;f) = z)} \leq \exp(\varepsilon) \cdot \frac{\Pr(X)}{\Pr(X')} \quad (1)$$

for all neighboring datasets X, X' and all $z \in \mathcal{R}$, where $\Pr(X)$ and $\Pr(X')$ is our prior knowledge of X and X'. In words, our posterior knowledge of X or X' after observing z can only grow by factor $\exp(\varepsilon)$ (Mironov, 2017), where ε is a *privacy budget* (Dwork and Roth, 2013).[3]

3 Analysis of DPText

In the heart of the model, DPText relies on the standard Laplace mechanism which takes a real-valued vector and perturbs each element by a random draw from the Laplace distribution.

Formally, let $\mathbf{z}$ be a real-valued d-dimensional vector. Then the Laplace mechanism outputs a vector $\tilde{\mathbf{z}}$ such that for each index $i = 1, \ldots, d$

$$\tilde{z}_i = z_i + s_i \quad (2)$$

where each s_i is drawn independently from a Laplace distribution with zero mean and scale b that is proportional to the ℓ_1 sensitivity Δ and the privacy budget ε, namely

$$s_i \sim \text{Lap}\left(\mu = 0; b = \frac{\Delta}{\varepsilon}\right) \quad (3)$$

The Laplace mechanism satisfies differential privacy (Dwork and Roth, 2013).

3.1 Reparametrization trick and inverse CDF sampling

DPText employs the variational autoencoder architecture in order to directly optimize the amount of noise added in the latent layer parametrized by ε. In other words, the scale of the Laplace distribution becomes a trainable parameter of the network. As directly sampling from a distribution is known to be problematic for end-to-end differentiable deep networks, DPText borrows the reparametrization trick from Kingma and Welling (2014).

In a nutshell, the reparametrization trick decouples drawing a random sample from a desired distribution (such as Exponential, Laplace, or Gaussian) into two steps: First draw a value from another distribution (such as Uniform), and then transform it using a particular function, mainly the inverse continuous density function (CDF).

As a matter of fact, sampling using the inverse CDF is a well-known and widely used

[2] This is *centralized DP*, as opposed to *local-DP* where no such trusted curator exists.

[3] In this paper, we will use the basic form of DP, that is $(\varepsilon, 0)$-DP. There are various other (typically more 'relaxed') variants of DP, such (ε, δ)-DP, but they are not relevant to the current paper as DPText also claims $(\varepsilon, 0)$-DP.

method (Devroye, 1986; Ross, 2012) and forms the backbone of probability distribution generators in many popular frameworks.

3.2 Inverse CDF of Laplace distribution

The inverse cumulative distribution function of Laplace distribution $\mathrm{Lap}(\mu; b)$ is:

$$F^{-1}(u) = \mu - b\,\mathrm{sgn}(u - 0.5)\,\ln(1 - 2|u - 0.5|) \tag{4}$$

where $u \sim \mathrm{Uni}(0, 1)$ is drawn from a standard uniform distribution (Sugiyama, 2016, p. 210), (Nahmias and Olsen, 2015, p. 303). An equivalent expression without the sgn and absolute functions is derived, e.g., by Li et al. (2019, p. 166) as

$$F^{-1}(u) = \begin{cases} b\ln(2u) + \mu & \text{if } u < 0.5 \\ \mu - b\ln(2(1-u)) & \text{if } u \geq 0.5 \end{cases} \tag{5}$$

where again $u \sim \mathrm{Uni}(0, 1)$.[4]

An alternative sampling strategy, as shown, e.g., by Al-Shuhail and Al-Dossary (2020, p. 62), assumes that the random variable is drawn from a shifted, zero-centered uniform distribution

$$v \sim \mathrm{Uni}\,(-0.5, +0.5) \tag{6}$$

and transformed through the following function

$$F^{-1}(v) = \mu - b\,\mathrm{sgn}(v)\ln(1 - 2|v|) \tag{7}$$

While both (4) and (7) generate samples from $\mathrm{Lap}(\mu; b)$, note the substantial difference between u and v, since each is drawn from a different uniform distribution (see visualizations in Fig. 1).

3.3 Proofs of DPText violating DP

According to Eq. 3 in (Alnasser et al., 2021), Eq. 9 in (Beigi et al., 2019a) which is an extended version of (Beigi et al., 2019b), in Eq. 14 in (Beigi et al., 2021), and personal communication to confirm the formulas, the main claim of DPText is as follows (rephrased):

> DPText utilizes the Laplace mechanism, which is DP (Dwork and Roth, 2013). It implements the mechanism as follows: Sampling a value from standard uniform
>
> $$v \sim \mathrm{Uni}(0, 1) \tag{8}$$
>
> and transforming using
>
> $$F^{-1}(v) = \mu - b\,\mathrm{sgn}(v)\ln(1 - 2|v|) \tag{9}$$
>
> is equivalent to sampling noise from $\mathrm{Lap}(b)$.

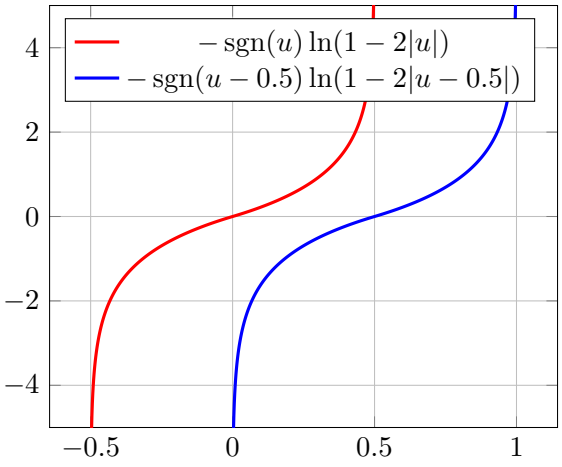

Figure 1: Inverse CDFs for Laplace sampling.

This claim is unfortunately false, as it mixes up both approaches introduced in Sec. 3.2. As a consequence, the Laplace mechanism using such sampling is not DP, which we will first prove formally.

Theorem 3.1. *Sampling using inverse CDF as in DPText using (8) and (9) does not produce Laplace distribution.*

Proof. We will rely on the standard proof of sampling from inverse CDF (see Appendix A). The essential step of that proof is that the CDF is increasing on the support of the uniform distribution, that is on $[0, 1]$. However, F^{-1} as used in (9) is increasing only on interval $[0, 0.5]$ (Fig. 1). For $v \geq 0.5$, we get negative argument to ln which yields a complex function, whose real part is even decreasing. Therefore (9) is not CDF of any probability distribution, if used with $\mathrm{Uni}(0, 1)$. □

As a consequence, the output $\ln(v \leq 0)$ arbitrarily depends on the particular implementation. In `numpy`, it is `NaN` with a warning only. Therefore this function samples only positive or `NaN` numbers. Since DPText sources are not publicly available, we can only assume that `NaN` numbers

[4] This implementation is used in `numpy`, see https://github.com/numpy/numpy/blob/maintenance/1.21.x/numpy/random/src/distributions/distributions.c#L469

are either replaced by zero, or the sampling proceeds as long as the desired number of samples is reached (discarding `NaNs`). In either case, no negative values can be obtained. See Fig. 3 in the Appendix for various Laplace-based distributions sampled with different techniques including possible distributions sampled in DPText.

Theorem 3.2. *DPText with private mechanism based on (8) and (9) fails to guarantee differential privacy.*

Proof. We rely on the standard proof of the Laplace mechanism as shown, e.g, by Habernal (2021). Let $X = 0$ and $X' = 1$ be two neighboring datasets, and the query f being the identity query, such that it outputs simply the value of X. Let the DPText mechanism $\mathcal{M}(X; f)$ outputs a particular value z.

In order to being differentially private, mechanism $\mathcal{M}(X; f)$ has to fulfill the following bound of the privacy loss:

$$\left| \frac{\Pr(\mathcal{M}(X) = z)}{\Pr(\mathcal{M}(X') = z)} \right| \leq \exp(\varepsilon) \tag{10}$$

for all neighboring datasets $X, X' \in \mathcal{X}$ and all outputs $z \in \mathcal{R}$ from the range of $\mathcal{M}$, provided that our priors over X and X' are uniform (cf. Eq. 1).

Fix $z = 0.1$. Then $\Pr(\mathcal{M}(X) = 0.1)$ will have a positive probability (recall it takes the query output $f(X = 0) = 0$ and adds a random number drawn from the probability distribution, which is always positive as shown in Theorem 3.1.) However $\Pr(\mathcal{M}(X') = 0.1)$ will be zero, as the query output $f(X' = 1) = 1$ will be added again only a positive random number and thus never be less than 1. By plugging this into (10), we obtain

$$\left| \frac{\Pr(\mathcal{M}(X) = 0.1)}{\Pr(\mathcal{M}(X') = 0.1)} \right| = \frac{\Pr > 0}{\Pr = 0} \not\leq \exp(\varepsilon) \tag{11}$$

which results in an infinity privacy loss and violates differential privacy. □

4 Empirical sanity check algorithm

It is impossible to empirically verify that a given DP-mechanism implementation is actually DP (Ding et al., 2018). However, it is possible to detect a DP-violating mechanism with a fair degree of certainty. We propose a general sanity check applicable to any real-valued DP mechanism, such as the Laplace mechanism, DPText, or any other.[5]

We start by constructing two neighboring datasets X (Alice) and X' (Bob) such that $X = (0, \ldots, 0_n)$ consists of n zeros and $X' = (1, \ldots, 1_n)$ consists of n ones. The dimensionality $n \in \{1, 2, \ldots\}$ is a hyperparameter of the experiment. We employ a synthetic data release mechanism (also called local DP). The mechanism takes X or X' and outputs its privatized version of the same dimensionality n, so that the zeros or ones are 'noisified' real numbers. The query sensitivity Δ is n.[6]

Thanks to the post-processing lemma, any post-processing of DP output remains DP. We can thus turn the output real vector back to all zeros or all ones, simply by rounding to closest 0 or 1 and applying majority voting. This process is in fact our reconstruction attack: given a privatized vector, we try to guess what the original values were, either all zeros or all ones.

What our attacker is doing, and what DP protects, is that if Alice gives us her privatized data, we cannot tell whether her private values were all zeros or all ones (up to a given factor); the same for Bob.

By definition (1) and having no prior knowledge over X and X' apart from the fact that the values are correlated, our attacker cannot exceed the guaranteed privacy loss $\exp(\varepsilon)$:

$$\frac{\Pr(X|\mathcal{M}(X; f) = z)}{\Pr(X'|\mathcal{M}(X; f) = z)} \leq \exp(\varepsilon) \tag{12}$$

We can estimate the conditional probability $\Pr(X|\mathcal{M}(X; f) = z)$ using maximum likelihood estimation (MLE) simply as our attacker's precision: How many times the attacker reconstructed true X values given the observed privatized vector. We can do the same for estimating the conditional probability of X'. In particular, we repeatedly run each DP mechanism over X and X' 10 million times each, which gives very precise MLE estimates even for small ε.[7]

[5] Some related works along these lines also utilize statistical analysis of the source code written in a C-like language (Wang et al., 2020).

[6] See (Dwork and Roth, 2013) for ℓ_1-sensitivity definition.

[7] For example, we repeated the full experiment on ADePT ($n = 2$, $\varepsilon = 0.1$) 100 times which results in standard deviation 0.0008 from the mean value 0.195. Better MLE precision can be simply obtained by increasing the 10 million repeats per experiment. Source codes available at `https://github.com/trusthlt/`

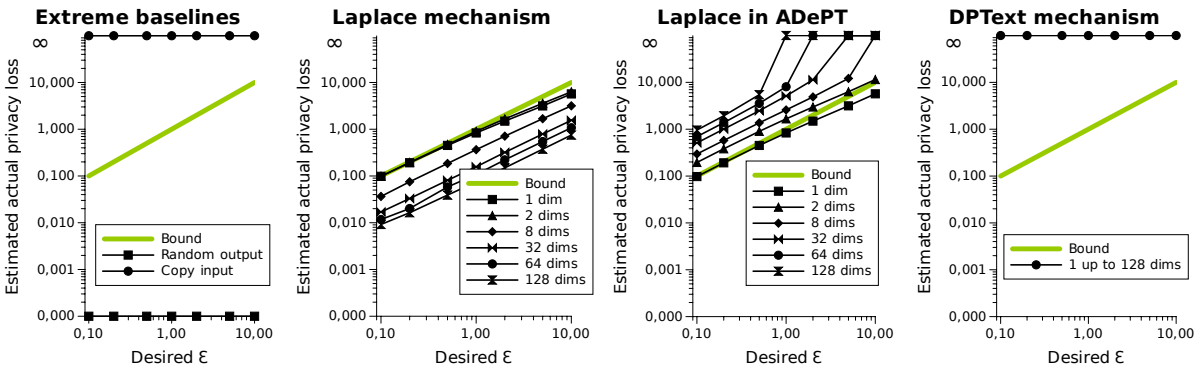

Figure 2: Area under the green line: Our attack does not reveal more than allowed by the desired privacy budget. Note that it does not guarantee DP, the reconstruction attack might be just weak. Area above the green line: The algorithm almost certainly violates DP as our attack caused bigger privacy loss than allowed by ε. *Extreme baselines* show two extreme scenarios, as *random output* is absolutely private (but provides zero utility) and *copy input* provides maximal utility but no privacy by revealing the data in full.

5 Results and discussion

For the sake of completeness, we implemented two extreme baselines: One that simply copies input (no privacy) and other one completely random regardless of the input (maximum privacy); these are shown in Figure 2 left. The vanilla Laplace mechanism behaves as expected; all empirical losses for all dimensions (1 up to 128) are bounded by ε. We re-implemented the Laplace mechanism from ADePT (Krishna et al., 2021) that, due to wrong sensitivity, has been shown theoretically as DP-violating (Habernal, 2021). We empirically confirm that ADePT suffered from the curse of dimensionality as the privacy loss explodes for larger dimensions. The last panel confirms our previous theoretical DPText results, which (regardless of dimensionality) has infinite privacy loss.

Note that we constructed the dataset carefully as two neighboring multidimensional correlated data that are as distant from each other as possible in the $(0, 1)^n$ space. However, DP must guarantee privacy for any datapoints, even the worst case scenario, as shown by the correct Laplace mechanism.

6 Conclusion

We formally proved that DPText (Beigi et al., 2019b,a; Alnasser et al., 2021; Beigi et al., 2021) is not differentially private due to wrong sampling in its reparametrization trick. We also proposed an empirical sanity check that confirmed our findings and can help to reveal potential errors in DP mechanism implementations for NLP.

7 Ethics Statement

We declare no conflict of interests with the authors of DPText, we do not even know them personally. The purpose of this paper is strictly scientific.

Acknowledgements

The independent research group TrustHLT is supported by the Hessian Ministry of Higher Education, Research, Science and the Arts. Thanks to Cecilia Liu, Haau-Sing Li, and the anonymous reviewers for their helpful feedback. A special thanks to Condor airlines, whose greed to make passengers pay for everything resulted in the most productive transatlantic flights I've ever had.

References

Abdullatif Al-Shuhail and Saleh Al-Dossary. 2020. *Robust Filter—Dealing with Impulse Noise*, pages 61–80. Springer International Publishing.

Walaa Alnasser, Ghazaleh Beigi, and Huan Liu. 2021. Privacy Preserving Text Representation Learning Using BERT. In *Proceedings of the 14th International Conference on Social, Cultural, and Behavioral Modeling (SBP-BRiMS)*, pages 91–100, Virtual event. Springer International Publishing.

John E. Angus. 1994. The Probability Integral Transform and Related Results. *SIAM Review*, 36(4):652–654.

acl2022-reparametrization-trick-broke-differential-privacy

Ghazaleh Beigi, Kai Shu, Ruocheng Guo, Suhang Wang, and Huan Liu. 2019a. I Am Not What I Write: Privacy Preserving Text Representation Learning. *arXiv preprint*.

Ghazaleh Beigi, Kai Shu, Ruocheng Guo, Suhang Wang, and Huan Liu. 2019b. Privacy Preserving Text Representation Learning. In *Proceedings of the 30th ACM Conference on Hypertext and Social Media*, pages 275–276, Hof, Germany. ACM.

Ghazaleh Beigi, Kai Shu, Ruocheng Guo, Suhang Wang, and Huan Liu. 2021. Systems and methods for a privacy preserving text representation learning framework. U.S. Patent, Pending Application US20210342546A1, Application filed by Arizona Board of Regents of ASU.

Luc Devroye. 1986. *Non-uniform random variate generation*. Springer-Verlag New York Inc.

Zeyu Ding, Yuxin Wang, Guanhong Wang, Danfeng Zhang, and Daniel Kifer. 2018. Detecting Violations of Differential Privacy. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 475–489, Toronto, Canada. ACM.

Cynthia Dwork and Aaron Roth. 2013. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3-4):211–407.

Ivan Habernal. 2021. When differential privacy meets NLP: The devil is in the detail. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1522–1528, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Timour Igamberdiev and Ivan Habernal. 2021. Privacy-Preserving Graph Convolutional Networks for Text Classification. *arXiv preprint*.

Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, pages 1–14, Banff, Canada.

Satyapriya Krishna, Rahul Gupta, and Christophe Dupuy. 2021. ADePT: Auto-encoder based Differentially Private Text Transformation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2435–2439, Online. Association for Computational Linguistics.

Xianxian Li, Huaxing Zhao, Dongran Yu, Li-e Wang, and Peng Liu. 2019. Multidimensional Correlation Hierarchical Differential Privacy for Medical Data with Multiple Privacy Requirements. In *Proceedings of the 2nd International Conference on Healthcare Science and Engineering*, pages 153–173, Guilin, China. Springer Singapore.

Yitong Li, Timothy Baldwin, and Trevor Cohn. 2018. Towards Robust and Privacy-preserving Text Representations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 25–30, Melbourne, Australia. Association for Computational Linguistics.

Pierre Lison, Ildikó Pilán, David Sanchez, Montserrat Batet, and Lilja Øvrelid. 2021. Anonymisation Models for Text Data: State of the art, Challenges and Future Directions. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4188–4203, Online. Association for Computational Linguistics.

Ilya Mironov. 2017. Rényi Differential Privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275, Santa Barbara, piiCA, USA. IEEE.

Steven Nahmias and Tava Lennon Olsen. 2015. *Production and Operations Analysis*, 7th edition. Waveland Press, Inc.

Sheldon Ross. 2012. *Simulation*, 5th edition. Academic Press.

Manuel Senge, Timour Igamberdiev, and Ivan Habernal. 2021. One size does not fit all: Investigating strategies for differentially-private learning across NLP tasks. *arXiv preprint*.

Adam Smith and Jonathan Ullman. 2021. Privacy in Statistics and Machine Learning. Lecture 5: Differential Privacy II.

Masashi Sugiyama. 2016. *Introduction to Statistical Machine Learning*. Morgan Kaufmann.

Yuxin Wang, Zeyu Ding, Daniel Kifer, and Danfeng Zhang. 2020. CheckDP: An Automated and Integrated Approach for Proving Differential Privacy or Finding Precise Counterexamples. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 919–938, Online. ACM.

A Proof of sampling from inverse CDF

Important fact 1: A random variable U is uniformly distributed on $[0, 1]$ if the following holds

$$U \sim \text{Uni}(0, 1) \iff \Pr(U \leq u) = u. \quad (13)$$

Important fact 2: For any function $g(\cdot)$ with an inverse function $g^{-1}(\cdot)$, the following holds

$$g(g^{-1}(x)) = x; \quad g^{-1}(g(x)) = x. \quad (14)$$

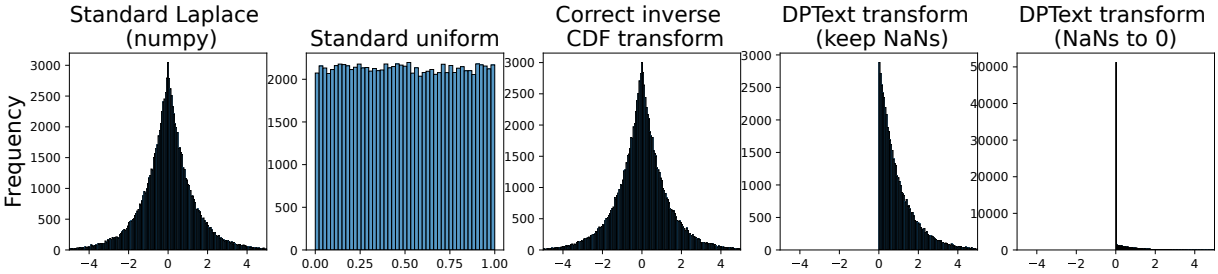

Figure 3: Comparing sampling strategies. Left: Sampling using vanilla `numpy` implementation. Second from the left: Uniform sample as basis for the following three inverse CDF transformations. Generated with 100k samples.

Important fact 3: For any increasing function $g(\cdot)$, we have by definition

$$x \leq y \implies g(x) \leq g(y). \tag{15}$$

We know that $\Pr(X \leq a)$ is a shortcut for probability of event E_1 defined using the set-builder notation as $E_1 = \{s \in \Omega : X(s) \leq a\}$. Then by plugging (15) into the predicate of E_1, we obtain an equal set, namely event $E_2 = \{s \in \Omega : g(X(s)) \leq g(a)\}$, for which the probability must be the same. Therefore for any random variable X and increasing function $g(\cdot)$ we have

$$\Pr(X \leq a) = \Pr(g(X) \leq g(a)). \tag{16}$$

Theorem A.1. *Let U be a uniform random variable on $[0, 1]$. Let X be a continuous random variable with CDF (cumulative distribution function) $F(\cdot)$. Let Y be defined such that $Y = F^{-1}(U)$. Then Y has CDF $F(\cdot)$.*

Proof. Function $F(\cdot)$ is the CDF of a continuous random variable X, and as a CDF its range is $[0, 1]$. Also, if $F(\cdot)$ is strictly increasing, it has a unique inverse function $F^{-1}(\cdot)$ defined on $[0, 1]$.

We defined $Y = F^{-1}(U)$, so consider

$$\Pr(Y \leq y) = \Pr\big(F^{-1}(U) \leq y\big). \tag{17}$$

Since $F(\cdot)$ is increasing, using (16) we get

$$\Pr(Y \leq y) = \Pr\big(F(F^{-1}(U)) \leq F(y)\big). \tag{18}$$

Now plugging (14) we obtain

$$\Pr(Y \leq y) = \Pr(U \leq F(y)), \tag{19}$$

and finally by (13)

$$\Pr(Y \leq y) = F(y). \tag{20}$$

□

For an overview of proofs of Theorem A.1 see (Angus, 1994).