Presentation-Only Demo - diffpriv: An R Package for Easy Differential Privacy

Benjamin I. P. Rubinstein BRUBINSTEIN@UNIMELB.EDU.AU
School of Computing and Information Systems
The University of Melbourne, Parkville, VIC 3010, Australia

Francesco Aldà FRANCESCO.ALDA@RUB.DE
Horst Görtz Institute for IT Security and Faculty of Mathematics
Ruhr-Universität Bochum, D-44801 Bochum, Germany

Abstract

The R package diffpriv provides tools for statistics and machine learning under differential privacy. Suitable for releasing analyses on privacy-sensitive data to untrusted third parties, differential privacy has become the framework of choice for privacy-preserving learning. diffpriv delivers: (a) implementations of generic mechanisms for privatizing non-private target functions, including the Laplace, Gaussian, exponential and Bernstein mechanisms; (b) a recent sensitivity sampler due to Rubinstein and Aldà (2017) that empirically estimates the sensitivity of non-private targets—obviating mathematical analysis for exact sensitivity bounds needed for most generic mechanisms; (c) an extensible framework for implementing differentially-private mechanisms. Together, the components of diffpriv permit easy high-utility privatization of complex analyses, learners and even black-box software programs.

Keywords: differential privacy, empirical process theory, R, open-source software

1. Introduction

Differential privacy (Dwork et al., 2006) has quickly become a key framework for semantic guarantees of data privacy when releasing analysis on privacy-sensitive data to untrusted third parties. A great deal of its popularity is owed to a suite of generic mechanisms for privatizing non-private target functions of data *i.e.*, statistics, estimation procedures, and learners. Common to these generic mechanisms is the requirement that the non-private target's sensitivity to dataset perturbation is known and bounded. In all except the most trivial analyses, bounding sensitivity is prohibitively involved. This paper describes the diffpriv R package that implements generic mechanisms for differential privacy, along with our recent sensitivity sampler that replaces exact sensitivity bounds with empirical estimates (Rubinstein and Aldà, 2017). As a result, diffpriv privatizes a wide range of procedures under random differential privacy (Hall et al., 2012), automatically without mathematical analysis and in many cases achieving high utility. diffpriv is available from `https://brubinstein.github.io/diffpriv/` under an open-source license.

2. Generic Mechanisms for Differential Privacy

Fundamental to differential privacy is a privacy-sensitive *dataset* (or *database*) $D \in \mathcal{D}^n$ on *domain* $\mathcal{D}$. In diffpriv a dataset can be a `list`, `matrix`, `data.frame`, `vector`. We say that a pair of databases $D, D' \in \mathcal{D}^n$ is *neighboring* if they differ on exactly one record. While individual records should not be revealed, we aim to release aggregate information on D with a mechanism. A *mechanism* $M : \mathcal{D}^n \to \mathcal{R}$ is a random-valued function of databases taking values in a response set $\mathcal{R}$; implemented in diffpriv as virtual class `DPMech`.

Definition 1 (**Dwork et al., 2006**) *For $\epsilon > 0$, mechanism $M : \mathcal{D}^n \to \mathcal{R}$ preserves ϵ-*differential privacy *if for all neighboring pairs $D, D' \in \mathcal{D}^n$, measurable response sets $R \subseteq \mathcal{R}$, $\Pr(M(D) \in R) \leq \exp(\epsilon) \cdot \Pr(M(D') \in R)$. Alternatively for $\delta \in (0,1)$, relaxed (ϵ, δ)-*differential privacy *holds if* $\Pr(M(D) \in R) \leq \exp(\epsilon) \cdot \Pr(M(D') \in R) + \delta$.

Mechanism M preserves differential privacy if its response distributions are close on neighboring pairs: an adversary cannot determine an unknown record from M responses, even with knowledge of the other $n-1$ records. Privacy parameters ϵ (ϵ, δ) are encapsulated in class `DPParamsEps` (`DPParamsDel` respectively). Virtual `DPMech` generic method `releaseResponse(mechanism, privacyParams, X)` takes privacy parameters along with sensitive dataset. Most generic mechanisms in differential privacy share a number of properties leveraged by the diffpriv package as follows.

Privatizing a target function. Many mechanisms $M : \mathcal{D}^n \to \mathcal{R}$ seek to privatize a non-private *target* function $f : \mathcal{D}^n \to \mathcal{B}$, with range $\mathcal{B}$ often (but not always!) coinciding with $\mathcal{R}$. Accordingly `DPMech` objects are initialized with `target` slot of type `function`. A mechanism's `releaseResponse()` method calls `target` to form privacy-preserving responses.

Normed target range space. Target f's output space $\mathcal{B}$ is typically imbued with a norm, denoted $\|\cdot\|_{\mathcal{B}}$, needed for measuring the sensitivity of f's outputs to input perturbation. diffpriv flexibly represents this norm within `DPMech` objects as described next.

Sensitivity-induced privacy. Many mechanisms achieve differential privacy by calibrating randomization to the sensitivity of the target function f. Targets that are relatively insensitive (sensitive) to perturbing input D to neighboring D' need relatively little (large) response randomization. On a pair of neighboring databases $D, D' \in \mathcal{D}^n$ the *sensitivity* of f is measured as $\Delta(D, D') = \|f(D) - f(D')\|_{\mathcal{B}}$. *Global sensitivity* is the largest such value $\overline{\Delta} = \sup_{D,D'} \|f(D) = f(D')\|_{\mathcal{B}}$ over all possible neighboring pairs. As we discuss in (Rubinstein and Aldà, 2017), a broad class of generic mechanisms, taking sensitivity Δ as a parameter, are *sensitivity-induced private*: for any neighboring pair $D, D' \in \mathcal{D}^n$ if $\Delta(D, D') \leq \Delta$ then the mechanism M_{Δ} run with parameter Δ achieves $\Pr(M_{\Delta}(D) \in R) \leq \exp(\epsilon) \cdot \Pr(M_{\Delta}(D') \in R)$ for all $R \subseteq \mathcal{R}$. When run with $\Delta = \overline{\Delta}$, this condition holds for all neighboring pairs: $M_{\overline{\Delta}}$ satisfies ϵ-differential privacy. Similarly for (ϵ, δ)-differential privacy. This is essentially how differential privacy is proved for most generic mechanisms, included those in diffpriv. `DPMech` objects can be initialized with a `sensitivity` argument, stored in an S4 slot of the same name. If the user provides a manually-derived global sensitivity bound $\overline{\Delta}$, then `releaseResponse()` responses preserve ϵ- or (ϵ, δ)-DP (depending on the specific mechanism). We now demonstrate this use case.

Example: Laplace mechanism. diffpriv implements Laplace (Dwork et al., 2006), Gaussian (Dwork and Roth, 2014), exponential (McSherry and Talwar, 2007), and Bernstein (Aldà and Rubinstein, 2017) mechanisms as `DPMech` subclasses `DPMechLaplace`, `DPMechGaussian`, `DPMechExponential`, and `DPMechBernstein`. An exponential example is included below. `DPMechLaplace` releases numeric vectors, and requires that $\mathcal{B}$ be numeric $\mathbb{R}^d$ for some d and adopts the L_1 norm (sum of absolutes) for $\|\cdot\|_{\mathcal{B}}$. The mechanism releases vectors in $\mathcal{R} = \mathcal{B} = \mathbb{R}^d$ by adding an i.i.d. sample of d Laplace-distributed random variables with means 0 and scales $\overline{\Delta}/\epsilon$ to $f(D)$ to achieve ϵ-differential privacy. `DPMechGaussian` also privatizes numeric responses but under L_2-sensitivity and weaker (ϵ, δ)-DP; `DPMechBernstein` leverages the Laplace mechanism to release multivariate real-valued functions.

We next demonstrate Laplace privatization of the sample mean on bounded data in $\mathcal{D}^n = [0,1]^n$, for which $\mathcal{B}$ dimension `dims` d is one. Global sensitivity is readily bounded as $1/n$: For any neighboring pair $D, D' \in [0,1]^n$, $\Delta(D, D') = n^{-1} \left|\sum_{i=1}^n D_i - \sum_{i=1}^n D'_i\right|$. Since $n-1$ records are identical, and all records are in $[0,1]$, this is $|D_n - D'_n|/n \leq 1/n$.

```r
library(diffpriv)
f <- function(X) mean(X) ## target function
n <- 100 ## dataset size
mechanism <- DPMechLaplace(target = f, sensitivity = 1/n, dims = 1)
D <- runif(n, min = 0, max = 1) ## the sensitive database in [0,1]^n
pparams <- DPParamsEps(epsilon = 1) ## desired privacy budget
r <- releaseResponse(mechanism, privacyParams = pparams, X = D)
cat("Private response r$response:", r$response,
  "\nNon-private response f(D):  ", f(D))

#> Private response r$response: 0.5244495
#> Non-private response f(D):   0.5343454
```

3. Sensitivity Sampling for Random Differential Privacy

When `target` global sensitivity is supplied as `sensitivity` within `DPMech` construction, responses are differentially private. Global sensitivity is known for *idealizations* of *e.g.*, coefficients for regularized logistic regression (Chaudhuri and Monteleoni, 2009) and the SVM (Rubinstein et al., 2012; Chaudhuri et al., 2011). In complex applications such as privatizing a software function, however, `target`'s global sensitivity may not be readily available. For such cases, diffpriv implements the sensitivity sampler of Rubinstein and Aldà (2017) which forms a high-probability estimate of `target` sensitivity by repeated probing of sensitivity on random neighboring database pairs, leveraging tools from empirical process theory. Like sensitivity estimates, resulting privacy holds with high probability.

Definition 2 (Hall et al., 2012) *A mechanism M preserves* (ϵ, γ)-random differential privacy *(with a corresponding form for ϵ, δ, γ) if $\forall R \subseteq \mathcal{R}, \Pr(M(D) \in R) \leq \exp(\epsilon) \cdot \Pr(M(D') \in R)$ holds with probability at least $1-\gamma$ over random database pairs D, D'.*

While weaker than ϵ-DP, RDP is arguably more natural than (ϵ, δ)-DP: The later safeguards all databases but not unlikely responses, while RDP protects against all responses but not pathological databases (as defined by the database sampling distribution). The sampling distribution can be anything meaningful *e.g.*, uniform, a Bayesian prior, a density from data privately fit by the Bernstein mechanism (Aldà and Rubinstein, 2017), etc.

The `DPMech` method `sensitivitySampler(object, oracle, n, m, gamma)` requires: a mechanism `object`; a function `oracle` which outputs i.i.d. random databases of given size `n` which should match the size of input data supplied later in calls to `releaseResponse()`; a sensitivity sample size `m`; and desired privacy confidence `gamma`. Either (but not both) of `m`, `gamma` can be omitted: the omitted resource will be optimized automatically. For example `m` taken small (few hundred) reflects limited sampling time; small `gamma` (*e.g.*, 0.05) prioritizes privacy. The sensitivity sampler calls appropriate `DPMech sensitivityNorm()` which implements $\Delta(D, D')$ for the mechanism's norm and stored `target`. New subclasses of `DPMech` need only implement this method in order to take advantage of the sensitivity sampler. After `sensitivitySampler()`, subsequent `releaseResponse()` results have a privacy parameter slot of type `DPParamsGam` indicating response RDP.

Example: Exponential mechanism. All `DPMech` subclasses are sensitivity-induced private and can be sensitivity sampled; we demonstrate the exponential mechanism here. Exponential privately optimizes an application-specific objective (or score, utility) $s(r)$ of candidate response $r \in \mathcal{R}$, with response distribution proportional to $\exp(\epsilon \cdot s(r)/(2\Delta))$. Typically s depends on input D, and so `DPMechExponential` is initialized with `target` that takes D and outputs a score function. That is, $\mathcal{B} = \mathbb{R}^{\mathcal{R}}$ is a real-valued function space on $\mathcal{R}$ and the class's `sensitivityNorm()` implements the sup-norm (*cf.* Rubinstein and Aldà, 2017). In practice, users supply `target` as an R closure as demonstrated below. Given global `sensitivity` of `target`, the mechanism preserves ϵ-DP; if `sensitivitySampler()` estimates sensitivity with some `gamma`, then RDP is preserved at confidence $\gamma =$`gamma`.

Applying these ideas to find the most frequent a–z character within a dataset of top-10 computer scientists from Semantic Scholar, subject to privacy of individuals. The exponential mechanism privately maximizes total frequency. But without bounded name lengths, this function has unbounded global sensitivity. We therefore use sensitivity sampler for $(1, 0.1)$-RDP, with an oracle that samples representative U.S. names based on `randomNames`.

```r
library(randomNames) ## a package that generates representative random names
oracle <- function(n) randomNames(n)
D <- c("Michael Jordan", "Andrew Ng", "Andrew Zisserman","Christopher Manning",
       "Jitendra Malik", "Geoffrey Hinton", "Scott Shenker",
       "Bernhard Scholkopf", "Jon Kleinberg", "Judea Pearl")
n <- length(D)
f <- function(X) { function(r) sum(r == unlist(base::strsplit(X, ""))) }
rSet <- as.list(letters) ## the response set, letters a--z, must be a list
mechanism <- DPMechExponential(target = f, responseSet = rSet)
mechanism <- sensitivitySampler(mechanism, oracle = oracle, n = n, gamma = 0.1)
pparams <- DPParamsEps(epsilon = 1)
r <- releaseResponse(mechanism, privacyParams = pparams, X = D)
cat("Private response r$response: ", r$response,
  "\nNon-private f(D) maximizer:  ", letters[which.max(sapply(rSet, f(D)))])

#> Private response r$response:  e
#> Non-private f(D) maximizer:   e
```

Example: Bernstein mechanism. Our final example demonstrates DPMechBernstein which implements the Bernstein mechanism for private function release (Aldà and Rubinstein, 2017). Here the non-private target function must itself (depending on a sensitive dataset) return a real-valued function on $[0, 1]^d$ for some dims d. The mechanism operates by fitting a Bernstein polynomial approximation to this released function, perturbing (via Laplace mechanism) the coefficients of this approximation, and then multiplying the dataset-independent Bernstein basis polynomials by these perturbed coefficients to reconstruct a private approximation. An additional argument to mechanism construction is the Bernstein polynomial degree latticeK which is also the size of the grid (in each dimension) on which the target release is evaluated.

Consider fitting sensitive dataset D with Priestly-Chao kernel regression, using the Gaussian kernel with a bandwidth hyperparameter specifying kernel smoothness. For simplicity, we'll consider a single co-variate. A fitting function for the estimator is as follows. It takes D a 2-column matrix with examples in rows, and returns a function for making predictions on new data.

```r
pck_regression <- function(D, bandwidth = 0.1) {
  K <- function(x) exp(-x^2/2)
  ids <- sort(D[,1], decreasing = FALSE, index.return = TRUE)$ix
  D <- D[ids, ]
  n <- nrow(D)
  ws <- (D[2:n,1] - D[1:(n-1),1]) * D[2:n,2]
  predictor <- function(x) {
    sum(ws * sapply((x - D[2:n,1]) / bandwidth, K)) / bandwidth
  }
  return(predictor)
}
```

We have the following (synthetic) sensitive dataset, as a 250×2 matrix with the first column representing co-variates/features and the second column representing dependent variables/labels.

```r
N <- 250
D <- runif(N)
D <- cbind(D, sin(D*10)*D + rnorm(N, mean=0, sd=0.2))
```

Let's fit three models for comparison: A non-private exact Priestly-Chao regression given by model; A non-private Bernstein approximation of the exact regression bmodel; and A privatized regression produced by DPMechBernstein, pmodel.

```r
## Non private fitting
model <- pck_regression(D)

## Bernstein non private fitting
K <- 25
bmodel <- bernstein(model, dims=1, k=K)

## Private Bernstein fitting
```

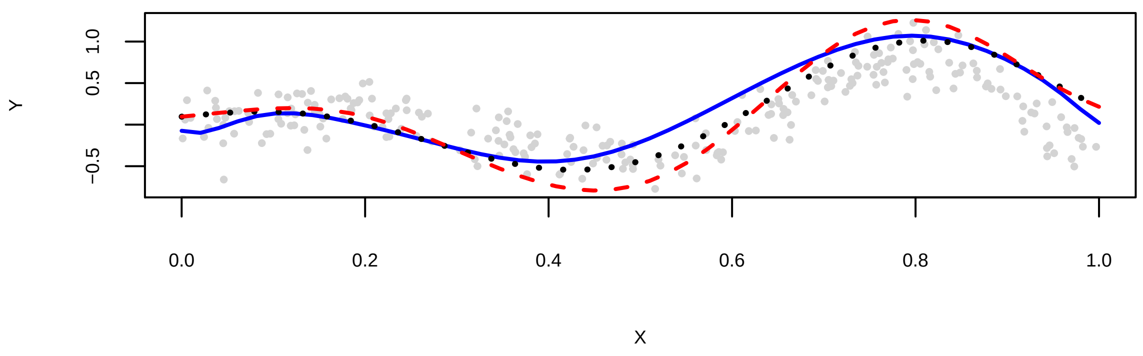

Figure 1: Kernel regression on 1D training data (gray points): non-private model (red dashed); non-private Bernstein polynomial approximation (black dotted); private Bernstein mechanism (blue solid).

```r
m <- DPMechBernstein(target=pck_regression, latticeK=K, dims=1)
P <- function(n) {  # a sampler of random, "plausible", datasets
  Dx <- runif(n)
  Dy <- rep(0, n)
  if (runif(1) < 0.95) Dy <- Dy + Dx
  if (runif(1) < 0.5) Dy <- Dy * sin(Dx)
  if (runif(1) < 0.5) Dy <- Dy * cos(Dx)
  cbind(Dx, Dy + rnorm(n, mean=0, sd=0.2))
}
m <- sensitivitySampler(m, oracle=P, n=N, gamma=0.20, m=500)
R <- releaseResponse(m, privacyParams=DPParamsEps(epsilon=5), X=D)
pmodel <- R$response
```

The private model is produced as described above. `sensitivitySampler()` probes the non-private model with 500 random pairs of datasets, sampled from `P()`, to estimate the target's sensitivity. The resulting perturbed private model preserves random differential privacy with level $\epsilon = 5$ and confidence $\gamma = 0.2$. In practice we could easily take γ much smaller (much higher confidence) by increasing sensitivity sample size `m`.

Let's now take our three fitted models, and predict the dependent variable/label across a range of covariates/features. These values are displayed in Figure 1.

```r
xs <- seq(from=0, to=1, length=50)
yhats   <- sapply(xs, model)
yhats.b <- predict(bmodel, xs)
yhats.p <- R$response(xs)
```

Acknowledgments

B. Rubinstein and F. Aldà acknowledge the support of the Australian Research Council (DE160100584) and the DFG Research Training Group GRK 1817/1 respectively.

References

Francesco Aldà and Benjamin I. P. Rubinstein. The Bernstein mechanism: Function release under differential privacy. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI'2017)*, pages 1705–1711, 2017.

Kamalika Chaudhuri and Claire Monteleoni. Privacy-preserving logistic regression. In *Advances in Neural Information Processing Systems*, pages 289–296, 2009.

Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(Mar):1069–1109, 2011.

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

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, pages 265–284. Springer, 2006.

Rob Hall, Alessandro Rinaldo, and Larry Wasserman. Random differential privacy. *Journal of Privacy and Confidentiality*, 4(2):43–59, 2012.

Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science, 2007 (FOCS'07)*, pages 94–103. IEEE, 2007.

Benjamin I. P. Rubinstein and Francesco Aldà. Pain-free random differential privacy with sensitivity sampling. In *Proceedings of the 34th International Conference on Machine Learning (ICML'2017)*, 2017. to appear; `https://arxiv.org/abs/1706.02562`.

Benjamin I. P. Rubinstein, Peter L. Bartlett, Ling Huang, and Nina Taft. Learning in a large function space: Privacy-preserving mechanisms for SVM learning. *Journal of Privacy and Confidentiality*, 4(1):65–100, 2012.