Towards Practical Differentially Private Convex Optimization

Roger Iyengar
Carnegie Mellon University

Joseph P. Near
University of California, Berkeley

Dawn Song
University of California, Berkeley

Om Thakkar
Boston University

Abhradeep Thakurta
University of California, Santa Cruz

Lun Wang
Peking University

***Abstract*—Building useful predictive models often involves learning from sensitive data. Training models with differential privacy can guarantee the privacy of such sensitive data. For convex optimization tasks, several differentially private algorithms are known, but none has yet been deployed in practice.**

In this work, we make two major contributions towards practical differentially private convex optimization. First, we present Approximate Minima Perturbation, a novel algorithm that can leverage any off-the-shelf optimizer. We show that it can be employed without any hyperparameter tuning, thus making it an attractive technique for practical deployment. Second, we perform an extensive empirical evaluation of the state-of-the-art algorithms for differentially private convex optimization, on a range of publicly available benchmark datasets, and real-world datasets obtained through an industrial collaboration. We release open-source implementations of all the differentially private convex optimization algorithms considered, and benchmarks on as many as nine public datasets, four of which are high-dimensional.

I. Introduction

Building useful predictive models often involves learning from sensitive data. To preserve the privacy of such sensitive data, many systems first carry out the learning task over the data, and then release just the final "learned" model. However, as many recent works [1], [2], [3] indicate, a model can leak information about the sensitive data it was trained on, even though the data might have never been made public. To prevent such information leakage, differential privacy (DP) [4], [5] has been recently used as a gold standard for performing learning tasks over sensitive data. It has also been adopted by large-scale corporations like Google [6], Apple [7], etc. Intuitively, DP prevents an adversary from confidently making any conclusions about whether a sample was used in training a model, even while having access to the model and any external side information.

The authors are ordered alphabetically.

With the recent advancements in machine learning and big data, private convex optimization has proven to be useful for large-scale learning over sensitive user data that has been collected by organizations. The objective of this work is to provide insight into practical differentially private convex optimization, with a specific focus on the classical technique of objective perturbation [8], [9]. Our main technical contribution is to design a new algorithm for private convex optimization that is amenable to real-world scenarios, and provide privacy and utility guarantees for it. In addition, we conduct a broad empirical evaluation of approaches for private convex optimization, including our new approach. Our evaluation is more extensive than prior works [8], [10], [11], [12]; it includes nine public datasets, four of which are *high-dimensional*. Apart from these, we also consider four real-world use cases, obtained in collaboration with Uber Technologies, Inc. We provide advice and resources for practitioners, including open-source implementations [13] of the algorithms evaluated, and benchmarks on all the public datasets considered.

A. Objective Perturbation and its Practical Feasibility

We focus our attention on the technique of objective perturbation, because prior works [8], [9] as well as our own preliminary empirical results have hinted at its superior performance. The standard technique of objective perturbation [8] consists of a two-stage process: "perturbing" the objective function by adding a random linear term, and releasing the minima of the perturbed objective. It has been shown [8], [9] that releasing such a minima is sufficient for achieving DP guarantees.

However, objective perturbation provides privacy guarantees *only if* the output of the mechanism is the *exact minima* of the perturbed objective. Practical algorithms for convex optimization often involve the use of first-order iterative methods, such as gradient descent or

for a very small proportion of the dataset), for this dataset we consider the task of predicting whether a sample is categorized under the most frequently used label or not.

The selected datasets include both *low-dimensional* and *high-dimensional* datasets. We define low-dimensional datasets to be ones where $n \gg p$ (where n is the number of samples and p is the number of dimensions). High-dimensional datasets are defined as those for which n and p are on roughly the same scale, i.e. $n \leq p$ (or nearly so). We consider the Synthetic-H, Gisette, Real-sim, and RCV1 datasets to be high-dimensional.

To obtain training and testing sets, we randomly shuffle the dataset, take the first 80% as the training set, and the remaining 20% as the testing set.

Sample clipping: Each of the algorithms we evaluate has the requirement that the loss have a Lipschitz constant. We can enforce this requirement for the loss functions we consider by bounding the norm for each sample. We can accomplish this by pre-processing the dataset, but it must be done carefully to preserve DP.

For all the algorithms except private Frank-Wolfe, to make the loss have an L_2-Lipschitz constant L, we bound the influence of each sample (x_i, y_i) by clipping the feature vector x_i to $\left(x_i \cdot \min\left(1, \frac{L}{\|x_i\|}\right)\right)$. This transformation is independent of other samples, and thus preserves DP; it has also been previously used, e.g. in [15]. As the private Frank-Wolfe algorithm requires the loss to have a relaxed L_1-Lipschitz constant L, it suffices (using Theorem 1 from [41]) to bound the L_∞-norm of each sample (x_i, y_i) by L. We achieve this by clipping each dimension $x_{i,j}$, where $j \in [d]$, to $\min(x_{i,j}, L)$.

Hyperparameters: Each of the evaluated algorithms has at least one hyperparameter. The values for these hyperparameters should be tuned to provide the best accuracy, but the tuning should be done privately in order to guarantee end-to-end differential privacy. Although a number of differentially private hyperparameter tuning algorithms have been proposed [8], [14], [15] to address this problem, they add more variance in the performance of each algorithm, thus making it more difficult to compare the performance across different algorithms.

In order to provide a fair comparison between algorithms, we use a grid search to determine the *best* value for each hyperparameter. Our grid search considers the hyperparameter values listed in Table II. In addition to the standard algorithm hyperparameters (Λ, η, T, k), we tune the clipping parameter L used in pre-processing the datasets, and the constraint on the model space used by private Frank-Wolfe, Private SGD when using regularized loss, and Private strongly convex PSGD. The parameter C controls the size of the L_1/L_2-ball from which models are selected by private Frank-Wolfe/the other algorithms respectively. For AMP, we set $\epsilon_2 = f \cdot \epsilon$, $\delta_2 = f \cdot \delta$, and tune for f. Here, f denotes the fraction of the budget (ϵ, δ) that is allocated to (ϵ_2, δ_2). Also, since the valid range of the hyperparameter ϵ_3 depends on the value of ϵ_1, we set $\epsilon_3 = f_1 \cdot \epsilon_1$, and tune for f_1. We also ensure that the constraint on ϵ_3 in Line 3 of Algorithm 1 is satisfied. Note that tuning hyperparameters may be non-private, but it enables a direct comparison of the algorithms themselves.

TABLE II
HYPERPARAMETER & PRIVACY PARAMETER VALUES

Hyperparameter	**Values Considered**
Λ (regularization factor)	$10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 0$
η (learning rate)	$0.001, 0.01, 0.1, 1$
T (number of iterations)	5, 10, 100, 1000, 5000
k (minibatch size)	50, 100, 300
L (clipping threshold)	0.1, 1, 10, 100
C (model constraint)	1, 10
f (output budget fraction)	0.001, 0.01, 0.1, 0.5
f_1 (privacy budget fraction)	0.9, 0.92, 0.95, 0.98, 0.99

Privacy Parameter	**Values Considered**
ϵ	$10^{-2}, 10^{-\frac{3}{2}}, 10^{-1}, 10^{-\frac{1}{2}}, 10^{0}, 10^{\frac{1}{2}}, 10^{1}$
δ	$\frac{1}{n^2}$

We consider a range of values for the privacy parameter ϵ. Following Wu et al. [12], we set the privacy parameter $\delta = \frac{1}{n^2}$, where n is the size of the training data. The complete set of values considered is listed in Table II. For multiclass classification datasets such as MNIST and Covertype, we implement the one-vs-all strategy by training a binary classifier for each class, and split ϵ and δ equally among the binary classifiers so that we can achieve an overall (ϵ, δ)-DP guarantee by using general composition [40].

Algorithm Implementations: The implementations used in our evaluation correspond to the pseudocode listings in Appendix C, are written in Python, and are available in our open source release [13]. For Approximate Minima Perturbation, we define the loss and gradient according to Algorithm 1, and leverage SciPy's `minimize` procedure to find the approximate minima.

For all datasets, our implementation is able to achieve $\gamma = \frac{1}{n^2}$, where n is the size of the training data. For low-dimensional datasets, our implementation of AMP uses SciPy's `BFGS` solver, for which we can specify the desired norm bound γ. The `BFGS` algorithm stores

the full Hessian of the objective, which does not fit in memory for the sparse high-dimensional datasets in our study. For these, we define an alternative low-memory implementation using SciPy's `L-BFGS-B` solver, which does not store the full Hessian.

Experiment procedure: Our experiment setup is designed to find the best possible accuracy achievable for a given setting of the privacy parameters. To ensure a fair comparison, we begin every run of each algorithm with the initial model 0^p. Because each of the evaluated algorithms introduces randomness due to noise, we train 10 independent models for each combination of the hyperparameter setting. We report the mean accuracy and standard deviation for the combination of the hyperparameter setting with the highest mean accuracy over the 10 independent runs.[4]

Differences with the setting in [12]: Although both the studies have 3 datasets in common (Covertype, KDDCup99, and MNIST), our setting is slightly different from [12] for all 3 of them. For Covertype, our study uses all 7 classes, while [12] uses a binary version. For KDDCup99, we use a 10% sample of the full dataset (as in [42]), while [12] uses the full dataset. For MNIST, we use all 784 dimensions, while [12] uses random projection to reduce the dimensionality to 50.

The results we obtain for both the variants of the Private PSGD algorithm [12] are based on faithful implementations of those algorithms. We tune the hyperparameters for both, using the grid search described earlier.

Non-private baseline: Note that one of the main objectives of this study is to determine the cost of privacy in practice for convex optimization. Hence, to provide a point of comparison for our results, we also train a non-private baseline model for each experiment. We use Scikit-learn's `LogisticRegression` class to train this model on the same training data as the private algorithms, and test its accuracy on the same testing data as the private algorithms. We do not perform sample clipping when training this model.

Strategy for Hyperparameter-free Approximate Minima Perturbation: Now, we describe a data-independent approach for setting Approximate Minima Perturbation's only hyperparameters, L, ϵ_2, δ_2, and ϵ_3, for *both* the loss functions we consider (see Section V-B). For L, we find that setting $L = 1$ achieves a good trade-off between the amount of noise added for perturbing the objective, and the information loss after sample clipping across all datasets. Next, we consider *only* the synthetically generated datasets for setting the hyperparameters specific to AMP. Fixing $\gamma = \frac{1}{n^2}$, we find that setting $\epsilon_2 = 0.01 \cdot \epsilon$ and $\delta_2 = 0.01 \cdot \delta$ achieves a good trade-off between the budget for perturbing the objective, and the amount of noise that its approximate minima can tolerate. For setting ϵ_3, we consider two separate cases:

- For $\epsilon_1 = 0.99 \cdot \epsilon$, and $\epsilon_3 = f_1 \cdot \epsilon_1$, we see that setting $f_1 = 0.99$ for $\epsilon_1 = 0.0099$, $f_1 = 0.95$ for $\epsilon_1 \in \{0.0313, 0.099\}$, and $f_1 = 0.9$ for $\epsilon_1 \in \{0.313, 0.99, 3.13, 9.99\}$ yields a good accuracy for Synthetic-L. Hence, we observe that for very low values of ϵ_1, a good accuracy is yielded by ϵ_3 close to ϵ_1 (i.e., most of the budget is used to reduce the scale of the noise, and the influence of regularization is kept large). As ϵ_1 increases, we see that it is more beneficial to reduce the effects of regularization. We fit a basic polynomial curve of the form $y = a + bx^{-c}$, where $a, b, c > 0$, to the above-stated values to get a dependence of f_1 (the privacy budget fraction) in terms of ϵ_1. We combine it with the lower bound imposed on f_1 by Theorem 1 (for instance, we require $f_1 \geq 0.9$ for $\epsilon_1 = 9.99$) to obtain the following data-independent relationship between ϵ_1 and ϵ_3 for low-dimensional datasets:

$$\epsilon_3 = \max\left\{\min\left\{0.887 + \frac{0.019}{\epsilon_1^{0.373}}, 0.99\right\}, 1 - \frac{0.99}{\epsilon_1}\right\} \cdot \epsilon_1$$

- For Synthetic-H, we see that setting $f_1 = 0.97$ yields a good accuracy for all the values of ϵ_1 considered. Thus, combining it with the lower bound imposed on f_1 by Theorem 1, we obtain the following relationship for high-dimensional datasets:

$$\epsilon_3 = \max\left\{0.97, 1 - \frac{0.99}{\epsilon_1}\right\} \cdot \epsilon_1$$

Note that the results for this strategy are consistent for both loss functions across all the public and the real-world datasets considered, *none* of which were used in defining the strategy except for setting the Lipschitz constant L of the loss. They can be considered to be effectively serving as test-cases for the strategy.

B. Loss Functions

Our evaluation considers the loss functions for two commonly used models: logistic regression and Huber SVM. This section contains results for logistic regression; results for Huber SVM are available in Appendix B.

Logistic regression: The L_2-regularized logistic regression loss function on a sample (x, y) with $y \in \{1, -1\}$ is $\ell(\theta, (x, y)) = \ln(1 + \exp(-y\langle\theta, x\rangle)) + \frac{\Lambda}{2}\|\theta\|^2$.

[4]The results shown are for hyperparameters tuned via the mean test set accuracy. Since all the considered algorithms aim to minimize the empirical loss, we also conducted experiments by tuning via the mean training set accuracy. Both settings provided visibly identical results.

Our experiments consider both the regularized and unregularized (i.e., $\Lambda = 0$) settings. The un-regularized version has L_2-Lipschitz constant L when for each sample x, $\|x\| \leq L$. It is also L^2-smooth. The regularized version has L_2-Lipschitz constant $L+\Lambda C$ when for each sample x, $\|x\| \leq L$, and for each model θ, $\|\theta\| \leq C$. It is also $(L^2 + \Lambda)$-smooth, and Λ-strongly convex.

C. Experiment 1: Low-dimensional Datasets

We present the results of our experiments with logistic regression on low-dimensional data in Figure 2. All four algorithms perform better in comparison with the non-private baseline for binary classification tasks (Synthetic-L, Adult, and KDDCup99) than for multi-class problems (Covertype and MNIST), because ϵ and δ must be split among the binary classifiers built for each class.

Figure 1 contains precise accuracy numbers for each dataset for reasonably low values of ϵ. These results provide a more precise comparison between the four algorithms, and quantify the accuracy loss versus the non-private baseline for each one. Across all datasets, Approximate Minima Perturbation generally provides the most accurate models across ϵ values.

D. Experiment 2: High-dimensional Datasets

For this experiment, we repeat the procedure in Experiment 1 on high-dimensional data, and present the results in Figure 2. The results are somewhat different in the high-dimensional regime. We observe that although Approximate Minima Perturbation generally outperforms all the other algorithms, the private Frank-Wolfe algorithm performs the best on Synthetic-H. From prior works [11], [17], we know that both objective perturbation and the private Frank-Wolfe have near dimension-independent utility guarantees when the loss is of a GLM, and we indeed observe this expected behavior from our experiments. As in experiment 1, we present precise accuracy numbers for $\epsilon = 0.1$ in Figure 1.

Private Frank-Wolfe works best when the optimal *model* is sparse (i.e., a few important features characterize the classification task well), as in the Synthetic-H dataset, which is well-characterized by just ten important features. This is because private Frank-Wolfe adds at most a single feature to the model at each iteration, and noise increases with the number of iterations. However, noise does *not* increase with the *total* number of features, since it scales with the bound on the ℓ_∞-norm of the samples. This behavior is in contrast to Approximate Minima Perturbation (and the other algorithms considered in our evaluation), for which noise scales with the bound on the ℓ_2-norm of the samples. Private Frank-Wolfe therefore approaches the non-private baseline better than the other algorithms for high-dimensional datasets with sparse models, even at low values of ϵ.

Dataset	NP baseline	AMP	H-F AMP	P-SGD	P-PSGD	P-SCPSGD	P-FW
Low-Dimensional Binary Datasets ($\epsilon = 0.1$)							
Synthetic-L	94.9	**83.1**	80.6	81.6	81.7	76.4	81.8
Adult	84.8	**79.1**	78.7	78.5	77.4	77.2	76.9
KDDCup99	99.1	97.5	97.4	98.0	**98.1**	95.8	96.8
Low-Dimensional Multi-class Datasets ($\epsilon = 1$[5])							
Covertype	71.2	64.3	63.5	**65.0**	62.4	62.2	63.0
MNIST	91.5	**71.9**	70.5	68.6	68.0	63.2	65.0
High-Dimensional Datasets ($\epsilon = 0.1$)							
Synthetic-H	95.8	53.2	51.4	52.8	53.5	52.0	**57.6**
Gisette	96.6	**62.8**	59.7	61.5	62.3	61.3	58.3
Real-sim	93.3	**73.1**	71.9	66.3	66.1	65.6	69.8
RCV1	93.5	**64.2**	59.9	55.1	58.9	56.2	64.1
Real-World Datasets ($\epsilon = 0.1$)							
Dataset #1	75.3	**75.3**[6]	75.3	75.3	75.3	75.3	75.3
Dataset #2	72.2	**70.4**	70.1	69.8	69.5	68.9	68.6
Dataset #3	73.6	**71.9**	71.8	71.8	71.4	71.2	71.6
Dataset #4	82.1	**81.7**	81.7	81.7	81.5	81.3	81.0

Fig. 1. Accuracy results (in %) for logistic regression. For each dataset, the result in bold represents the DP algorithm with the best accuracy for that dataset. A key for the abbreviations used for the algorithms is provided in Table III.

TABLE III
LIST OF ABBREVIATIONS USED FOR ALGORITHMS

Abbreviation	Full-form
NP baseline	Non-private baseline
AMP	Approximate Minima Perturbation
H-F AMP	Hyperparameter-free AMP
P-SGD	Private SGD
P-PSGD	Private PSGD
P-SCPSGD	Private strongly convex PSGD
P-FW	Private Frank-Wolfe

E. Experiment 3: Real-world Use Cases

For this experiment, we repeat the procedure in Experiment 1 on real-world use cases, obtained in collaboration with Uber. These use cases are represented by four datasets, each of which has separately been used to train a production model deployed at Uber. The details of these datasets are listed in Table I. The results of this

[5]We report the accuracy for $\epsilon = 1$ for multi-class datasets, as compared to $\epsilon = 0.1$ for datasets with binary classification, because multi-class classification is a more difficult task than binary classification.

[6]For Dataset #1, AMP slightly outperforms even the NP baseline, as can been seen from Figure 2.

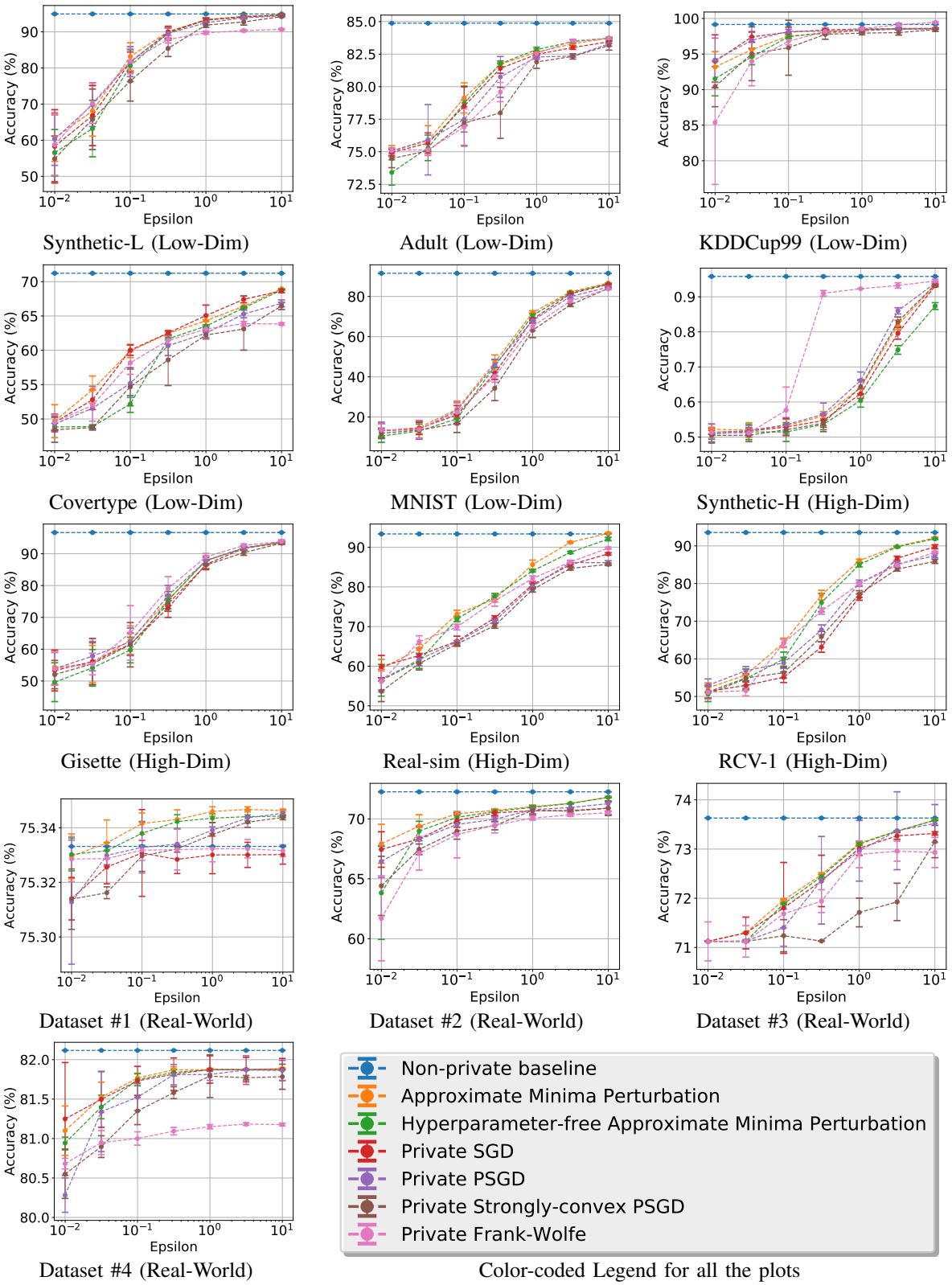

Fig. 2. Accuracy results for logistic regression on low-dimensional, high-dimensional and real-world datasets. Horizontal axis depicts varying values of ϵ; vertical axis shows accuracy (in %) on the testing set.

experiment are depicted in Figure 2, with more precise results for $\epsilon = 0.1$ in Figure 1.

The real-world datasets are much larger than the datasets considered in Experiment 1. The difference in scale is reflected in the results: all of the algorithms converge to the non-private baseline for very low values of ϵ. These results suggest that in many practical settings, the *cost of privacy is negligible*. In fact, for Dataset #1, some differentially private models exhibit a slightly *higher* accuracy than the non-private baseline for a wide range of ϵ. For instance, even Hyperparameter-free AMP, which is end-to-end differentially private as there is no tuning involved, yields an accuracy of 75.34% for $\epsilon = 0.1$ versus the non-private baseline of 75.33%. Some prior works [18], [19] have theorized that differential privacy could act as a type of regularization for the system, and improve the generalization error; this empirical result of ours aligns with this claim.

F. Discussion

For large datasets, the cost of privacy is low. Our results confirm the expectation that very accurate differentially private models exist for large datasets. Even for relatively small datasets like Adult and KDDCup99 (where $n < 100,000$), our results show that a differentially private model has accuracy within 6% of the non-private baseline even for a conservative privacy setting of $\epsilon = 0.1$.

For *all* the larger real-world datasets ($n > 1$m), the accuracy of the best differentially private model is within 4% of the non-private baseline even for the most conservative privacy value considered ($\epsilon = 0.01$). For $\epsilon = 0.1$, it is within 2% of the baseline for two of these datasets, essentially identical to the baseline for one of them, and even slightly higher than the baseline for one.

These results suggest that for realistic deployments on large datasets ($n > 1$m, and low-dimensional), a differentially private model can be deployed without much loss in accuracy.

Approximate Minima Perturbation almost always provides the best accuracy, and is easily deployable in practice. Our results in all the experiments demonstrate that among the available algorithms for differentially private convex optimization, our Approximate Minima Perturbation approach almost always produces models with the best accuracy. For four of the five low-dimensional datasets, and all the public high-dimensional datasets we considered, Approximate Minima Perturbation provided consistently better accuracy than the other algorithms. Under some conditions like high-dimensionality of the datasets, and sparsity of the optimal predictive model for it, private Frank-Wolfe does give the best performance. Unlike Approximate Minima Perturbation, however, no hyperparameter-free variant of private Frank-Wolfe exists—and suboptimal hyperparameter values can reduce accuracy significantly for this algorithm.

As mentioned earlier, Approximate Minima Perturbation also has important properties that enable its practical deployment. It can leverage any off-the-shelf optimizer as a black box, allowing implementations to use existing scalable optimizers (our implementation uses Scipy's `minimize`). None of the other evaluated algorithms have these properties.

Hyperparameter-free Approximate Minima Perturbation provides good utility. As demonstrated by our experimental results, AMP can be deployed without tuning hyperparameters, at little cost to accuracy. Our data-independent approach therefore enables deployment—without significant loss of accuracy—in practical settings where public data may not be available for tuning.

VI. Conclusion

This paper takes two important steps towards practical differentially private convex optimization. We have presented Approximate Minima Perturbation, a novel algorithm for differentially private convex optimization that does not require the optimization process to reach the true minima. It can leverage any off-the-shelf solver, and can be employed without hyperparameter tuning. Therefore, it is amenable to be deployed in practice.

We have also performed an extensive empirical evaluation of state-of-the-art approaches for differentially private convex optimization. To encourage the further development and deployment, we have released the implementations used in our evaluation, and the benchmarking scripts used to obtain the datasets and perform the experiments. This benchmark provides a standard point of comparison for further advances in differentially private convex optimization.

VII. Acknowledgements

The authors would like to thank Adam Smith for the discussions regarding the main privacy proof of AMP, and the anonymous reviewers for their helpful comments. This material is in part based upon work supported by NSF CCF-1740850, DARPA contract #N66001-15-C-4066, the Center for Long-Term Cybersecurity, and Berkeley Deep Drive. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors, and do not necessarily reflect the views of the sponsors.

REFERENCES

[1] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: ACM, 2015, pp. 1322–1333.

[2] X. Wu, M. Fredrikson, S. Jha, and J. F. Naughton, "A methodology for formalizing model-inversion attacks," in *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, June 2016, pp. 355–370.

[3] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*, May 2017, pp. 3–18.

[4] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation." in *EUROCRYPT*, 2006.

[5] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography Conference*. Springer, 2006, pp. 265–284.

[6] Ú. Erlingsson, V. Pihur, and A. Korolova, "Rappor: Randomized aggregatable privacy-preserving ordinal response," in *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. ACM, 2014, pp. 1054–1067.

[7] "Apple tries to peek at user habits without violating privacy," *The Wall Street Journal*, 2016.

[8] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization," *JMLR*, 2011.

[9] D. Kifer, A. Smith, and A. Thakurta, "Private convex empirical risk minimization and high-dimensional regression," *Journal of Machine Learning Research*, vol. 1, p. 41, 2012.

[10] S. Song, K. Chaudhuri, and A. D. Sarwate, "Stochastic gradient descent with differentially private updates," in *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*. IEEE, 2013, pp. 245–248.

[11] P. Jain and A. Thakurta, "(near) dimension independent risk bounds for differentially private learning," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ser. ICML'14. JMLR.org, 2014, pp. I–476–I–484.

[12] X. Wu, F. Li, A. Kumar, K. Chaudhuri, S. Jha, and J. Naughton, "Bolt-on differential privacy for scalable stochastic gradient descent-based analytics," in *Proceedings of the 2017 ACM International Conference on Management of Data*, ser. SIGMOD '17. New York, NY, USA: ACM, 2017, pp. 1307–1322.

[13] "Differentially Private Convex Optimization Benchmark," 2017. [Online]. Available: https://github.com/sunblaze-ucb/dpml-benchmark

[14] K. Chaudhuri and S. Vinterbo, "A stability-based validation procedure for differentially private machine learning," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'13. USA: Curran Associates Inc., 2013, pp. 2652–2660.

[15] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: ACM, 2016, pp. 308–318.

[16] R. Bassily, A. Smith, and A. Thakurta, "Private empirical risk minimization: Efficient algorithms and tight error bounds," in *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*. IEEE, 2014, pp. 464–473.

[17] K. Talwar, A. Thakurta, and L. Zhang, "Private empirical risk minimization beyond the worst case: The effect of the constraint set geometry," *CoRR*, vol. abs/1411.5417, 2014.

[18] R. Bassily, A. D. Smith, and A. Thakurta, "Private empirical risk minimization, revisited," *CoRR*, vol. abs/1405.7085, 2014.

[19] C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, and A. Roth, "Generalization in adaptive data analysis and holdout reuse," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, pp. 2350–2358.

[20] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[21] S. Bubeck *et al.*, "Convex optimization: Algorithms and complexity," *Foundations and Trends® in Machine Learning*, vol. 8, no. 3-4, pp. 231–357, 2015.

[22] L. Zhang, T. Yang, and R. Jin, "Empirical risk minimization for stochastic convex optimization: $O(1/n)$- and $O(1/n^2)$-type of risk bounds," in *Proceedings of the 2017 Conference on Learning Theory*, ser. Proceedings of Machine Learning Research, S. Kale and O. Shamir, Eds., vol. 65. Amsterdam, Netherlands: PMLR, 07–10 Jul 2017, pp. 1954–1979.

[23] V. Feldman, "Generalization of erm in stochastic convex optimization: The dimension strikes back," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 3576–3584.

[24] A. Smith and A. Thakurta, "Differentially private feature selection via stability arguments, and the robustness of the lasso," in *COLT*, 2013.

[25] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Research Logistics (NRL)*, vol. 3, no. 1-2, pp. 95–110, 1956.

[26] M. Jaggi, "Revisiting frank-wolfe: projection-free sparse convex optimization," in *Proceedings of the 30th International Conference on International Conference on Machine Learning-Volume 28*. JMLR. org, 2013, pp. I–427.

[27] S. Lacoste-Julien and M. Jaggi, "An affine invariant linear convergence analysis for frank-wolfe algorithms," *arXiv preprint arXiv:1312.7864*, 2013.

[28] ——, "On the global linear convergence of frank-wolfe optimization variants," in *Advances in Neural Information Processing Systems*, 2015, pp. 496–504.

[29] S. Lacoste-Julien, "Convergence Rate of Frank-Wolfe for Non-Convex Objectives," Jun. 2016, 6 pages.

[30] P. Jain, P. Kothari, and A. Thakurta, "Differentially private online learning." in *COLT*, vol. 23, 2012, pp. 24–1.

[31] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*. IEEE, 2013, pp. 429–438.

[32] A. G. Thakurta and A. Smith, "(nearly) optimal algorithms for private online learning in full-information and bandit settings," in *Advances in Neural Information Processing Systems 26*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds., 2013, pp. 2733–2741.

[33] P. Jain and A. Thakurta, "Differentially private learning with kernels," in *ICML*, 2013.

[34] J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett, "Functional mechanism: Regression analysis under differential privacy," *Proc. VLDB Endow.*, vol. 5, no. 11, pp. 1364–1375, Jul. 2012.

[35] X. Wu, M. Fredrikson, W. Wu, S. Jha, and J. F. Naughton, "Revisiting differentially private regression: Lessons from learning theory and their consequences," *CoRR*, vol. abs/1512.06388, 2015.

[36] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Sept 2015, pp. 909–910.

[37] Z. Ji, Z. C. Lipton, and C. Elkan, "Differential privacy and machine learning: a survey and review," *CoRR*, vol. abs/1412.7584, 2014.

[38] A. Nikolov, K. Talwar, and L. Zhang, "The geometry of differential privacy: The sparse and approximate cases," in *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, ser. STOC '13. New York, NY, USA: ACM, 2013, pp. 351–360.
[39] P. Billingsley, *Probability and Measure*, ser. Wiley Series in Probability and Statistics. Wiley, 1995.
[40] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2014.
[41] R. Paulavičius and J. Žilinskas, "Analysis of different norms and corresponding lipschitz constants for global optimization," *Ukio Technologinis ir Ekonominis Vystymas*, vol. 12, no. 4, pp. 301–306, 2006.
[42] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization," *Journal of Machine Learning Research*, vol. 12, no. Mar, pp. 1069–1109, 2011.
[43] S. Dasgupta and L. Schulman, "A probabilistic analysis of em for mixtures of separated, spherical gaussians," *JMLR*, 2007.
[44] K. Talwar, A. Thakurta, and L. Zhang, "Nearly optimal private lasso," in *NIPS*, 2015.

APPENDIX

A. Omitted Proofs

Here, we provide a proof for the utility guarantee of Algorithm 1, which is provided in Theorem 2. For bounding the expected risk of the algorithm, we first need to bound its empirical risk (Lemma A.1).

Lemma A.1 (Empirical Risk)**.** *Let $\hat{\theta}$ be the minimizer of the objective function $\mathcal{L}(\theta; D) = \frac{1}{n}\sum_{i=1}^{n} \ell(\theta; d_i)$, and θ_{min} be the minimizer of the objective function $\mathcal{L}_{priv}(\theta; D) = \mathcal{L}(\theta; D) + \frac{\Lambda}{2n}\|\theta\|^2 + b_1^T\theta$, where b_1 is as defined in Algorithm 1. Also, let θ_{out} be the output of Algorithm 1. We have:*

$$\mathcal{L}(\theta_{out}; D) - \mathcal{L}(\hat{\theta}; D) \leq L\left(\frac{n\gamma}{\Lambda} + \|b_2\|\right) + \frac{\Lambda\|\hat{\theta}\|^2}{2n} + \frac{2n\|b_1\|^2}{\Lambda}.$$

Proof. We have

$$\mathcal{L}(\theta_{out}; D) - \mathcal{L}(\hat{\theta}; D) = (\mathcal{L}(\theta_{out}; D) - \mathcal{L}(\theta_{min}; D)) + \left(\mathcal{L}(\theta_{min}; D) - \mathcal{L}(\hat{\theta}; D)\right) \tag{4}$$

First, we will bound $(\mathcal{L}(\theta_{out}; D) - \mathcal{L}(\theta_{min}; D))$. We have:

$$\begin{aligned}\mathcal{L}(\theta_{out}; D) - \mathcal{L}(\theta_{min}; D) &\leq |\mathcal{L}(\theta_{out}; D) - \mathcal{L}(\theta_{min}; D)| \\ &\leq L\|\theta_{out} - \theta_{min}\| \\ &= L\|\theta_{approx} - \theta_{min} + b_2\| \\ &\leq L\|\theta_{approx} - \theta_{min}\| + L\|b_2\| \\ &\leq L\left(\frac{n\gamma}{\Lambda} + \|b_2\|\right) \end{aligned} \tag{5}$$

The second inequality above follows from the Lipschitz property of $\mathcal{L}(; D)$. The first equality follows as $\theta_{out} = \theta_{min} + (\theta_{approx} - \theta_{min} + b_2)$, whereas the last inequality follows from inequality 3.

Next, we bound $\left(\mathcal{L}(\theta_{min}; D) - \mathcal{L}(\hat{\theta}; D)\right)$ on the lines of the proof of Lemma 3 in [9]. Let $\theta^{\#} = \arg\min_{\theta\in\mathbb{R}^p} \mathcal{L}^{\#}(\theta; D)$, where $\mathcal{L}^{\#}(\theta; D) = \mathcal{L}(\theta; D) + \frac{\Lambda}{2n}\|\theta\|^2$. As a result, $\mathcal{L}_{priv}(\theta; D) = \mathcal{L}^{\#}(\theta; D) + b_1^T\theta$. So, we have:

$$\begin{aligned}\mathcal{L}(\theta_{min}; D) - \mathcal{L}(\hat{\theta}; D) &= \mathcal{L}^{\#}(\theta_{min}; D) - \mathcal{L}^{\#}(\theta^{\#}; D) \\ &\quad + \mathcal{L}^{\#}(\theta^{\#}; D) - \mathcal{L}^{\#}(\hat{\theta}; D) \\ &\quad + \frac{\Lambda\|\hat{\theta}\|^2}{2n} - \frac{\Lambda\|\theta_{min}\|^2}{2n} \\ &\leq \mathcal{L}^{\#}(\theta_{min}; D) - \mathcal{L}^{\#}(\theta^{\#}; D) \\ &\quad + \frac{\Lambda\|\hat{\theta}\|^2}{2n} \end{aligned} \tag{6}$$

The inequality above follows as $\mathcal{L}^{\#}(\theta^{\#}; D) \leq \mathcal{L}^{\#}(\hat{\theta}; D)$.

Let us now bound $\mathcal{L}^{\#}(\theta_{min}; D) - \mathcal{L}^{\#}(\theta^{\#}; D)$. To this end, we first observe that since $\mathcal{L}_{priv}$ is $\frac{\Lambda}{n}$-strongly convex in θ, we have that

$$\begin{aligned}\mathcal{L}_{priv}(\theta^{\#}; D) &\geq \mathcal{L}_{priv}(\theta_{min}; D) \\ &\quad - \nabla\mathcal{L}_{priv}(\theta_{min}; D)^T(\theta_{min} - \theta^{\#}) \\ &\quad + \frac{\Lambda}{2n}\|\theta^{\#} - \theta_{min}\|^2 \\ &= \mathcal{L}_{priv}(\theta_{min}; D) + \frac{\Lambda}{2n}\|\theta^{\#} - \theta_{min}\|^2 \end{aligned} \tag{7}$$

The equality above follows as $\|\nabla\mathcal{L}_{priv}(\theta_{min}; D)\| = 0$.

Substituting the definition of $\mathcal{L}_{priv}(; D)$ in equality 7, we get that

$$\mathcal{L}^{\#}(\theta_{min}; D) - \mathcal{L}^{\#}(\theta^{\#}; D) \leq b_1^T(\theta^{\#} - \theta_{min}) - \frac{\Lambda}{2n}\|\theta^{\#} - \theta_{min}\|^2 \tag{8}$$

$$\leq \|b_1\| \cdot \|\theta^{\#} - \theta_{min}\| \tag{9}$$

Inequality 9 above follows by the Cauchy–Schwarz inequality.

Now, since $\mathcal{L}^{\#}(\theta_{min}; D) - \mathcal{L}^{\#}(\theta^{\#}; D) \geq 0$, it follows from inequalities 8 and 9 that

$$\begin{aligned}\|b_1\| \cdot \|\theta^{\#} - \theta_{min}\| &\geq \frac{\Lambda}{2n}\|\theta^{\#} - \theta_{min}\|^2 \\ \Rightarrow \|\theta^{\#} - \theta_{min}\| &\leq \frac{2n\|b_1\|}{\Lambda} \end{aligned} \tag{10}$$

We get the statement of the lemma from equation 4, and inequalities 5, 6, 9, and 10. □

Now, we are ready to prove Theorem 2.

Proof of Theorem 2. The proof is on the lines of the proof of Theorem 4 in [9]. First, let us get a high probability bound on $\mathcal{L}(\theta_{out}; D) - \mathcal{L}(\hat{\theta}; D)$. To this end, we will first bound $\|b_1\|$ and $\|b_2\|$ w.h.p., where $b_s \sim \mathcal{N}\left(0, \sigma_s^2 I_{p\times p}\right)$ for $s \in \{1, 2\}$. Using Lemma 2 from [43], we get that w.p. $\geq 1 - \frac{\alpha}{2}$,

$$\|b_s\| \leq \sigma_s \sqrt{2p \log \frac{2}{\alpha}}.$$

Substituting this into Lemma A.1, we get that w.p. $\geq 1 - \alpha$,

$$\mathcal{L}(\theta_{out}; D) - \mathcal{L}(\hat{\theta}; D) \leq L\left(\frac{n\gamma}{\Lambda} + \sigma_2\sqrt{2p \log \frac{2}{\alpha}}\right) + \frac{\Lambda\|\hat{\theta}\|^2}{2n} + \frac{4n\sigma_1^2 p \log \frac{2}{\alpha}}{\Lambda}.$$

It is easy to see that by making $\epsilon_i = \frac{\epsilon}{2}$ for $i \in \{1, 2\}, \epsilon_3 = \max\left\{\frac{\epsilon_1}{2}, \epsilon_1 - 0.99\right\}, \delta_j = \frac{\delta}{2}$ for $j \in \{1, 2\}$, and setting $\Lambda = \Theta\left(\frac{L\sqrt{rp \log 1/\delta}}{\epsilon\|\hat{\theta}\|} + \frac{n}{\|\hat{\theta}\|}\sqrt{\frac{L\gamma\sqrt{p \log 1/\delta}}{\epsilon}}\right)$ such that it satisfies the constraint in Step 2 in Algorithm 1, we get the statement of the theorem. □

B. Results for Huber SVM

This section reports the results of experiments with the Huber SVM loss function. The Huber SVM loss function is a differentiable and smooth approximation of the standard SVM's hinge loss. We define the loss function as in [18]. Defining $z = y\langle x, \theta\rangle$, the Huber SVM loss function is:

$$\ell(\theta, (x, y)) = \begin{cases} 1 - z & 1 - z > h \\ 0 & 1 - z < -h \\ \frac{(1-z)^2}{4h} + \frac{1-z}{2} + \frac{h}{4} & \text{otherwise} \end{cases}$$

As with logistic regression, the Huber SVM loss function has L_2-Lipschitz constant L when for each sample x, $\|x\| \leq L$.

We repeat the experiments of Section V with the Huber SVM loss. To ensure that the experiments run to completion for Synthetic-H, we run the experiments on 2000 samples, each consisting of 2000 dimensions. For all the experiments, we obtain the non-private baseline using SciPy's `minimize` procedure with the Huber SVM loss function defined above. Following Wu et al. [12], we set $h = 0.1$. The results are shown in Figure 4, with more precise results in Figure 3. They demonstrate a similar trend to the earlier results for logistic regression, with our Approximate Minima Perturbation approach generally providing the highest accuracy. However, the advantage of Approximate Minima Perturbation is less pronounced in this setting.

Dataset	NP baseline	AMP	H-F AMP	P-SGD	P-PSGD	P-SCPSGD	P-FW
Low-Dimensional Binary Datasets ($\epsilon = 0.1$)							
Synthetic-L	94.9	**89.3**	87.8	85.6	86.2	79.4	86.8
Adult	84.8	**79.6**	77.5	79.0	76.5	76.0	77.8
KDDCup99	99.1	98.7	**98.7**[7]	98.5	98.5	98.1	98.0
Low-Dimensional Multi-class Datasets ($\epsilon = 1$[8])							
Covertype	71.5	**66.4**	65.3	64.3	62.3	62.7	63.3
MNIST	91.5	**74.7**	73.7	69.6	72.9	70.6	65.1
High-Dimensional Datasets ($\epsilon = 0.1$)							
Synthetic-H[9]	96.5	55.2	54.3	55.0	**56.6**	55.6	56.0
Gisette	96.6	69.9	67.9	65.7	**70.6**	66.8	66.8
Real-sim	93.6	**78.3**	76.7	73.6	71.8	69.7	78.3
RCV1[9]	93.8	74.5	72.9	71.3	70.1	69.7	**75.8**
Real-World Datasets ($\epsilon = 0.1$)							
Dataset #1	75.3	75.3	75.3	75.3	**75.3**[10]	75.3	75.3
Dataset #2	72.2	**70.8**	70.6	70.8	70.3	70.2	68.6
Dataset #3	73.6	**71.3**	71.2	71.2	71.1	71.1	71.1
Dataset #4[9]	81.9	81.5	81.3	**81.7**	81.5	81.2	81.2

Fig. 3. Accuracy results (in %) for Huber SVM. For each dataset, the result in bold represents the DP algorithm with the best accuracy for that dataset. A key for the abbreviations used for the algorithms is provided in Table III.

[7]H-F AMP can outperform AMP when the data-independent strategy provides a better value for the privacy budget fraction f_1 than the specific set of values we consider for tuning in AMP.

[8]We report the accuracy for $\epsilon = 1$ for multi-class datasets, as compared to $\epsilon = 0.1$ for datasets with binary classification, as multi-class classification is a more difficult task than binary classification.

[9]The numbers cited here do not reflect the trend for this dataset, as can be seen from Figure 3

[10]Slightly outperforms even the NP baseline, as can been seen from Figure 2.

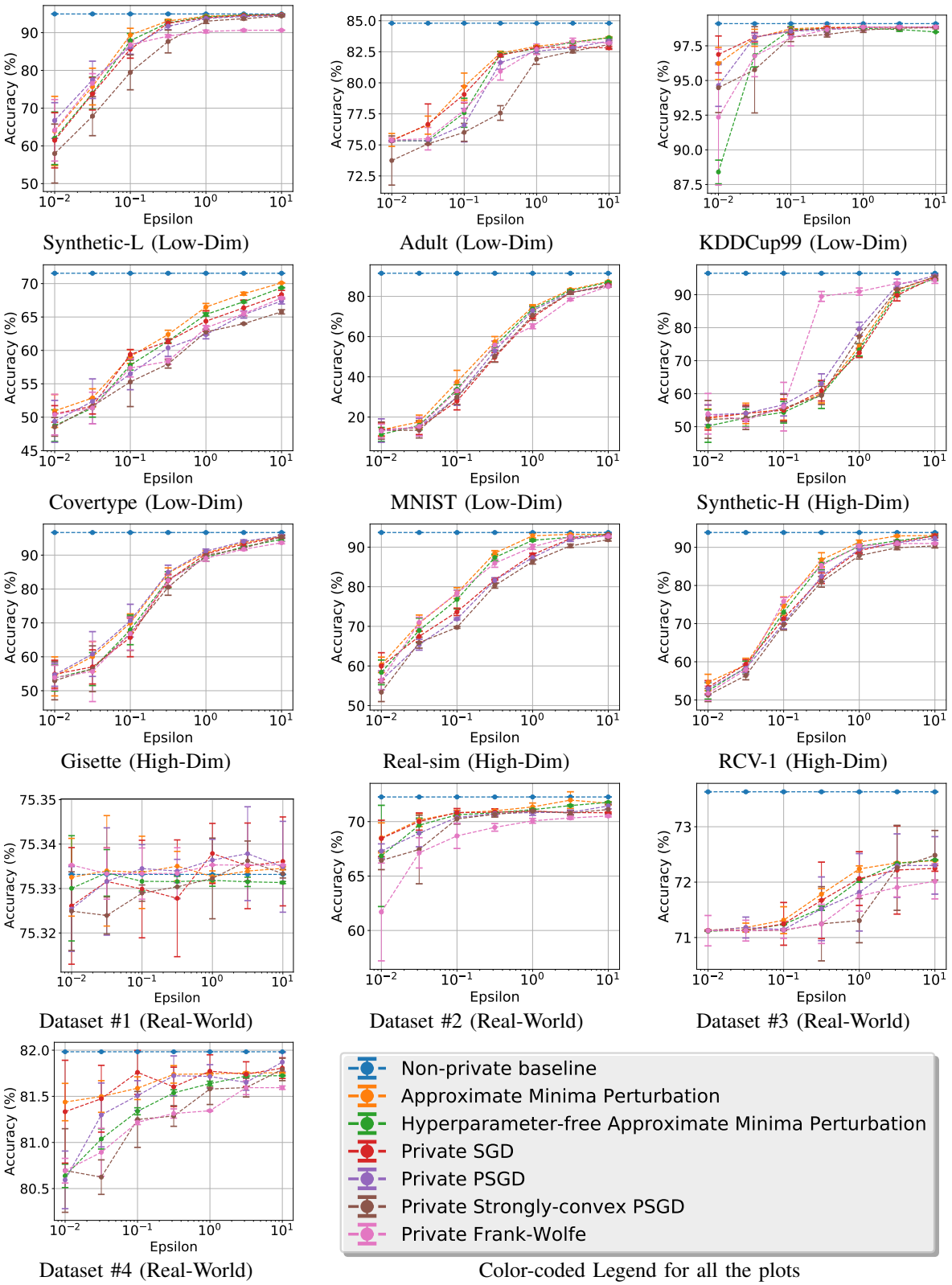

Fig. 4. Accuracy results for Huber SVM. Horizontal axis depicts varying values of ϵ; vertical axis shows accuracy (in %) on the testing set.

C. Pseudocodes for Algorithms evaluated in Section V

Algorithm 2: Differentially Private Minibatch Stochastic Gradient Descent [16], [15]

Input: Data set: $D = \{d_1, \cdots, d_n\}$, loss function: $\ell(\theta; D_i)$ with L_2-Lipschitz constant L, privacy parameters: (ϵ, δ), number of iterations: T, minibatch size: k, learning rate function: $\eta : [T] \to \mathbb{R}$.

$\sigma^2 \leftarrow \frac{16L^2T\log\frac{1}{\delta}}{n^2\epsilon^2}$
$\theta_1 = 0^p$
for *t = 1 to T-1* **do**
  $s_1, \cdots, s_k \leftarrow$ Sample k samples uniformly with replacement from D
  $b_t \sim \mathcal{N}(0, \sigma^2 I_{p\times p})$
  $\theta_{t+1} = \theta_t - \eta(t)[(\frac{1}{k}\sum_{i=1}^{k}\nabla\ell(\theta; s_i)) + b_t]$
end
Output θ_T

Algorithm 3: Differentially Private Permutation-based Stochastic Gradient Descent [12]

Input: Data set: $D = \{d_1, \cdots, d_n\}$, loss function: $\ell(\theta; D_i)$ with L_2-Lipschitz constant L, privacy parameters: (ϵ, δ), number of passes: T, minibatch size: k, constant learning rate: η.

$\theta \leftarrow 0^p$
Let τ be a random permutation of $[n]$
for *t = 1 to* $T-1$ **do**
  for *b = 1 to* $\frac{n}{k}$ **do**
    Let $s_1 = d_{\tau(bk)}, \cdots, s_k = d_{\tau(b(k+1)-1)}$
    $\theta \leftarrow \theta - \eta(\frac{1}{k}\sum_{i=1}^{k}\nabla\ell(\theta; s_i))$
  end
end
$\sigma^2 \leftarrow \frac{8T^2L^2\eta^2\log(\frac{2}{\delta})}{k^2\epsilon^2}$
$b \sim \mathcal{N}(0, \sigma^2 I_{p\times p})$
Output $\theta_{priv} = \theta + b$

Algorithm 4: Differentially Private Strongly Convex Permutation-based Stochastic Gradient Descent [12]

Input: Data set: $D = \{d_1, \cdots, d_n\}$, loss function: $\ell(\theta; D_i)$ that is ξ-strongly convex and β-smooth with L_2-Lipschitz constant L, privacy parameters: (ϵ, δ), number of passes: T, minibatch size: k.

$\theta \leftarrow 0^p$
Let τ be a random permutation of $[n]$
for *t = 1 to* $T-1$ **do**
  $\eta_t \leftarrow \min\frac{1}{\beta}, \frac{1}{\xi t}$
  for *b = 1 to* $\frac{n}{k}$ **do**
    Let $s_1 = d_{\tau(bk)}, \cdots, s_k = d_{\tau(b(k+1)-1)}$
    $\theta \leftarrow \theta - \eta_t(\frac{1}{k}\sum_{i=1}^{k}\nabla\ell(\theta; s_i))$
  end
end
$\sigma^2 \leftarrow \frac{8L^2\log(\frac{2}{\delta})}{\xi^2n^2\epsilon^2}$
$b \sim \mathcal{N}(0, \sigma^2 I_{p\times p})$
Output $\theta_{priv} = \theta + b$

Algorithm 5: Differentially Private Frank-Wolfe [44]

Input: Data set: $D = \{d_1, \cdots, d_n\}$, loss function: $L(\theta; D) = \frac{1}{n}\sum_{i=1}^{n}\ell(\theta; d_i)$ (with L_1-Lipshitz constant L for ℓ), privacy parameters: (ϵ, δ), convex set: $C = conv(S)$ with $\|C\|_1$ denoting $max_{s\in S}\|s\|_1$ and S being the set of corners.

Choose an arbitrary θ_1 from C
$\sigma^2 \leftarrow \frac{32L^2\|C\|_1^2Tlog(1/\delta)}{n^2\epsilon^2}$
for *t = 1 to T-1* **do**
  $\forall s \in S, \alpha_s \leftarrow \langle s, \bigtriangledown L(\theta_t; D)\rangle + Lap(\sigma)$, where $Lap(\lambda) \sim \frac{1}{2\lambda}e^{-|x|/\lambda}$
  $\widetilde{\theta}_t \leftarrow \arg\min_{s\in S}\alpha_s$
  $\theta_{t+1} \leftarrow (1-\eta_t)\theta_t + \eta\widetilde{\theta}_t$, where $\eta_t = \frac{1}{t+1}$
end
Output $\theta^{priv} = \theta_T$