Lecture 13 Deep Learning with DP

Yu-Xiang Wang

COMPUTER SCIENCE
UC SANTA BARBARA
Computing. ReInvented.

Recap: Last lecture

- Utility of NoisyGD mechanism
  - Main tools: the convergence of SGD.
- Privacy Amplification by Sampling and Noisy SGD
- Computational consideration

Recap: NoisyGD summary

	Lipschitz + convex	Lipschitz + Smooth + convex	Smooth + Lipschitz + convex + GLM
Output Pert	$\frac{d^{1/4}L\|\theta^*\|\log(\frac{1}{\delta})^{1/4}}{n^{1/2}\epsilon^{1/2}}$	$\frac{d^{1/3}\beta^{1/3}L^{2/3}\|\theta^*\|^{4/3}\log(\frac{1}{\delta})^{1/3}}{n^{2/3}\epsilon^{2/3}}$	Same as left
ObjPert	Not applicable	$\frac{dL\|\theta^*\|\sqrt{\log(\frac{1}{\delta})}}{n\epsilon}$	$\frac{\sqrt{d}L\|\theta^*\|\sqrt{\log(\frac{1}{\delta})}}{n\epsilon}$
NoisyGD	$\frac{\sqrt{d}L\|\theta^*\|\sqrt{\log(\frac{1}{\delta})}}{n\epsilon}$	$\frac{\sqrt{d}L\|\theta^*\|\sqrt{\log(\frac{1}{\delta})}}{n\epsilon}$	$\frac{\sqrt{d}L\|\theta^*\|\sqrt{\log(\frac{1}{\delta})}}{n\epsilon}$

Lower order terms and dependence on β hidden.

	Lipschitz + Strongly convex	Lipschitz + Smooth + Nonconvex
NoisyGD	$\frac{dL^2\log(1/\delta)}{n\lambda\epsilon^2}$	$\frac{\sqrt{n\beta dL^2(f(\theta_1)-f^*)\log(1/\delta)}}{n\epsilon}$

Stationary point convergence

Recap: Computational Complexity of NoisyGD

- General pattern: First term goes to 0, second term dominates with large T. (How large does T need to be?)
  - Convex + Lipschitz

$$T \geq \frac{n^2\rho}{d} \asymp \frac{n^2}{d}\epsilon^2$$

  - Convex + Smooth

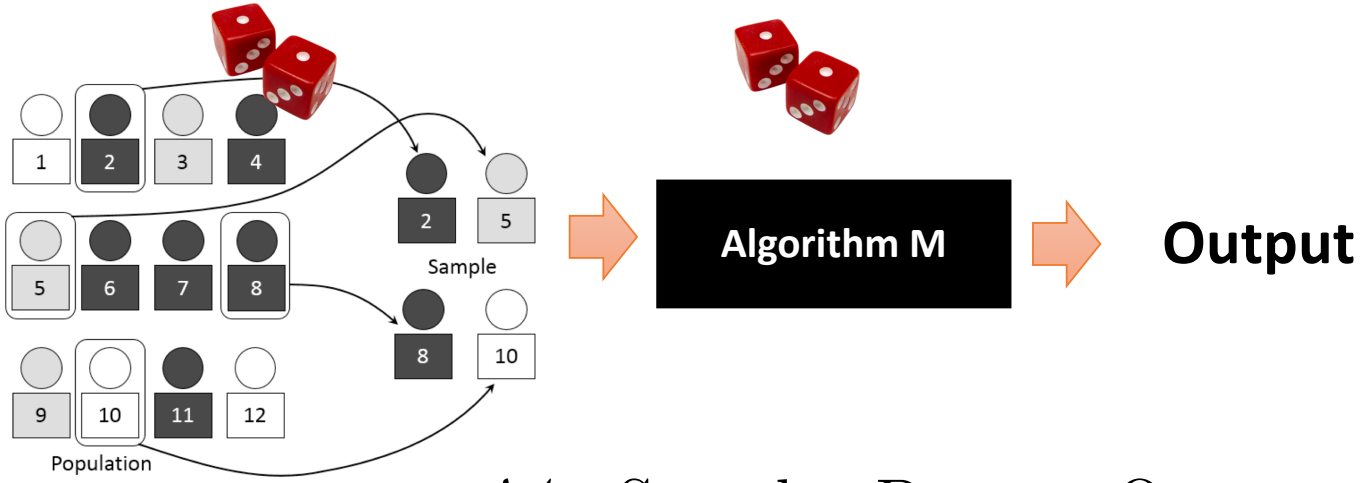
$$T \geq \frac{2n\beta\|x_1 - x^*\|\sqrt{\rho}}{\sqrt{d}L} \asymp n\epsilon/\sqrt{d})$$

  - Strongly convex + Lipschitz

$$T \geq \frac{n^2\rho}{\lambda} \asymp \frac{n^2\epsilon^2}{\lambda}$$

Each iteration requires n incremental gradient evaluation!

Recap: Privacy Amplification by Sampling



$$\mathcal{M} \circ \text{Sample} : \text{Data} \rightarrow \text{Output}$$

Subsampling Lemma: If M obeys (Ɛ,δ)-DP, then M ∘ Subsample obeys that (Ɛ',δ')-DP with $\delta' = \gamma\delta$

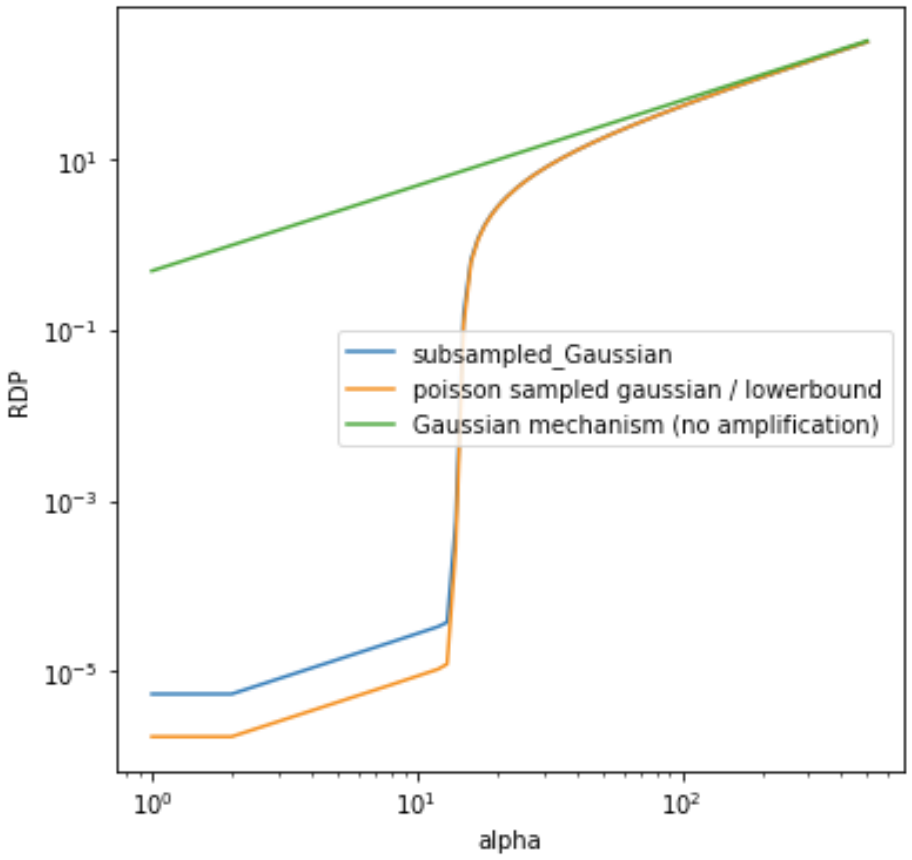
$$\epsilon' = \log(1 + \gamma(e^{\epsilon} - 1)) = O(\gamma\epsilon)$$

Recap: The Noisy Stochastic Gradient Descent Mechanism (NoisySGD)

- Privacy analysis:
  - A composition of T **subsampled** gaussian mechanism
  - Satisfying that:

- Utility analysis:
  - Same convergence theorem of SGD, but different parameters due to sampling.

Recap: The Renyi DP of sampled mechanisms



A concise asymptotic bound of the RDP for sampled Gaussian mechanism via truncated CDP.

Theorem: Assume $\mathcal{M}$ satisfies $\rho - \mathrm{CDP}$.

then

$$\epsilon_{\mathcal{M} \circ \mathrm{Sample}_\gamma}(\alpha) \leq 13\alpha\gamma^2\rho \text{ for } \alpha \leq \frac{\log(1/\gamma)}{4\rho}$$

More precise bound: https://arxiv.org/abs/1808.00087

Theorem 11 of Bun et al. "Composable and Versatile Privacy via Truncated CDP" https://projects.iq.harvard.edu/files/privacytools/files/bun_mark_composable_.pdf

Working out the constraint due to this phases transition in amplification by sampling

Today

- Computational complexity of NoisySGD
- Application to Deep Learning with Differential Privacy
- Other models in Deep Learning with DP

Readings

- NoisySGD: Bassily et al. https://arxiv.org/abs/1405.7085
- Deep Learning with DP: Abadi et al. https://arxiv.org/abs/1607.00133
- Amplification by sampling in RDP: https://arxiv.org/abs/1808.00087

NoisySGD's Convergence Bound under the smooth convex Case

$$\mathbb{E}\left[\frac{1}{T}\sum_{t}(f(x_t)-f^*)\right] \leq \frac{\|x_1-x^*\|^2}{T\eta} + \eta d\sigma^2$$

- Larger variance

NoisySGD's Convergence Bound under the non-smooth convex Case

$$\mathbb{E}\left[\frac{1}{T}\sum_{t}(f(x_t)-f^*)\right] \leq \frac{\|x_1-x^*\|^2}{T\eta} + \eta(\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}\|\partial f(x_t)\|^2\right] + d\sigma^2)$$

NoisySGD's convergence under the strongly convex case

$$\mathbb{E}\left[f(\frac{2}{T(T+1)}\sum_{t=1}^{T}tx_t)\right]-f(x^*)\leq\frac{4(n^2L^2+d\sigma^2)}{\lambda(T+1)}$$

Requires more advanced algorithm to obtain subquadratic time algorithm for non-smooth convex ERM

- Very interesting recent paper by
  - Kulkarni, Lee and Liu (2021): https://arxiv.org/pdf/2103.15352.pdf

- By a smoothing trick they obtained an algorithm with the following oracle calls

$$O(\frac{n^{3/2}}{d^{1/8}} + \frac{n^2}{d})$$

- i.e., slightly worse dimension dependence than the smooth + convex case that we derived.
- Also, if we are to consider minimizing population risk instead, then subquadratic-time algorithms are possible.

Checkpoint: Comparing NoisyGD and NoisySGD computationally

- Both optimal information-theoretically.
  - If we ignore computation and add very large noise, but use infinitesimal step-size
- Table to compare computation
  - in terms of the number of incremental gradient calls to achieve information theoretic limit up to a constant

	Lipschitz + Smooth + Convex	Lipschitz + Convex	Lipschitz + Strongly convex
NoisyGD	$\frac{n^2\beta\|x_1 - x^*\|\sqrt{\rho}}{\sqrt{d}L}$	$\frac{n^3\rho}{d}$	$\frac{n^3\rho}{\lambda}$
NoisySGD	$\frac{n^{3/2}\beta^{1/2}\|x_1 - x^*\|\rho^{1/2}}{d^{1/4}L^{1/2}} + \frac{n^2\rho}{d}$	$\frac{n^2\rho^{3/4}}{d^{1/2}} + \frac{n^2\rho}{d}$	$\frac{n^2\rho^{3/4}}{d^{1/2}} + \frac{n^2\rho}{d}$

Open problem: what is the optimal computational complexity?

Remainder of the lecture

- A survey of differentially private deep learning

- Reference:
  - NoisySGD: Abadi et al. (2016)
  - PATE: Papernot et al. (2018), PrivateKNN: Zhu et al. (2020)

  - Autodp tutorial on private deep learning: https://github.com/yuxiangw/autodp/blob/master/tutorials/tutorial_private_deep_learning.ipynb

Deep Learning with Differential Privacy

- Main challenge: It is difficult to enforce global sensitivity

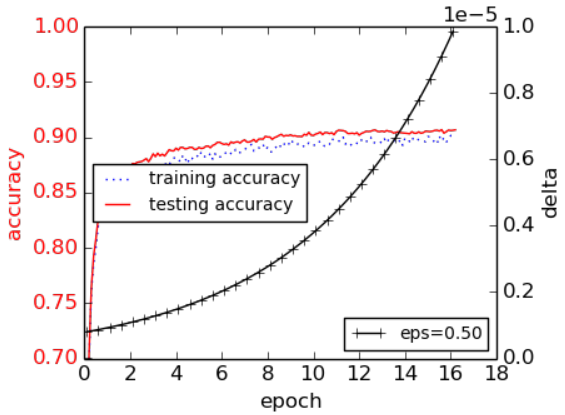
- Abadi et al: Per-example Gradient Clipping

- Other tricks:
  - Use small neural networks
  - Private-PCA to reduce dimension in the original data

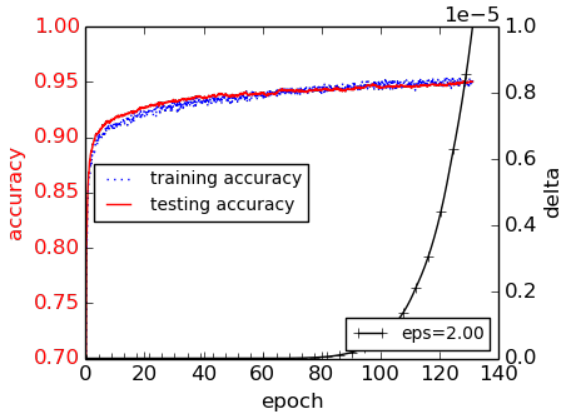
What does Deep Learning with Differential Privacy look like in autodp?

```
from autodp.autodp_core import Mechanism
from autodp.transformer_zoo import Composition
from autodp import mechanism_zoo, transformer_zoo

class NoisySGD_mech(Mechanism):
    def __init__(self,prob,sigma,niter,name='NoisySGD'):
        Mechanism.__init__(self)
        self.name=name
        self.params={'prob':prob,'sigma':sigma,'niter':niter}

        # create such a mechanism as in previously
        subsample = transformer_zoo.AmplificationBySampling() # by default this is using poisson sampling
        mech = mechanism_zoo.GaussianMechanism(sigma=sigma)
        prob = prob
        # Create subsampled Gaussian mechanism
        SubsampledGaussian_mech = subsample(mech,prob,improved_bound_flag=True)

        # Now run this for niter iterations
        compose = transformer_zoo.Composition()
        mech = compose([SubsampledGaussian_mech],[niter])

        # Now we get it and let's extract the RDP function and assign it to the current mech being constructed
        rdp_total = mech.RenyiDP
        self.propagate_updates(rdp_total,type_of_update='RDP')
```

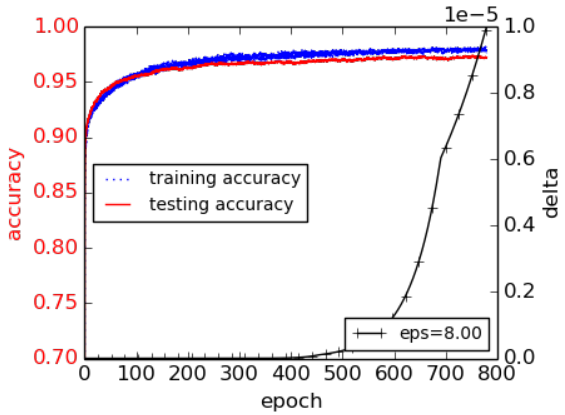
Empirical results on MNIST



(1) Large noise



(2) Medium noise

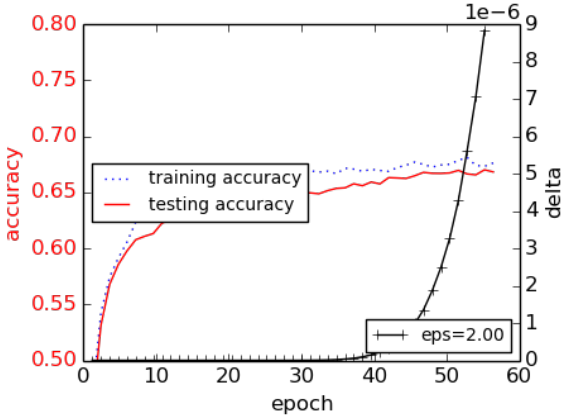


(3) Small noise

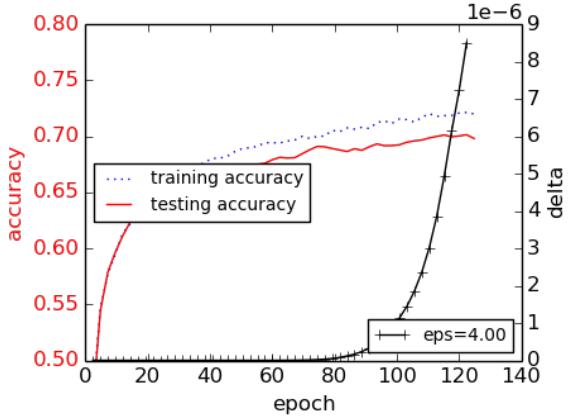
Figure 3: Results on the accuracy for different noise levels on the MNIST dataset. In all the experiments, the network uses 60 dimension PCA projection, 1,000 hidden units, and is trained using lot size 600 and clipping threshold 4. The noise levels (σ, σ_p) for training the neural network and for PCA projection are set at (8, 16), (4, 7), and (2, 4), respectively, for the three experiments.

Abadi et al. (2016) Deep Learning with Differential Privacy: https://arxiv.org/pdf/1607.00133.pdf

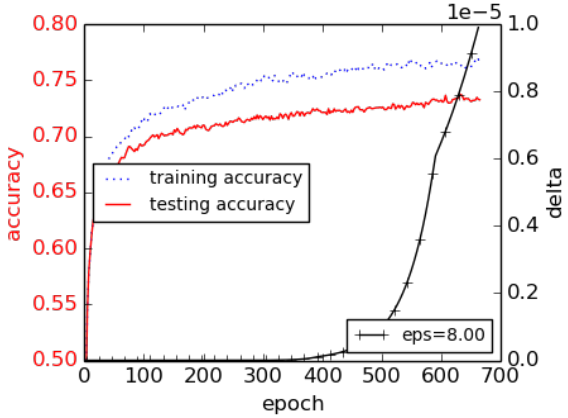
Empirical results on CIFAR10



(1) $\varepsilon = 2$



(2) $\varepsilon = 4$



(3) $\varepsilon = 8$

Figure 6: Results on accuracy for different noise levels on CIFAR-10. With δ set to 10^{-5}, we achieve accuracy 67%, 70%, and 73%, with ε being 2, 4, and 8, respectively. The first graph uses a lot size of 2,000, (2) and (3) use a lot size of 4,000. In all cases, σ is set to 6, and clipping is set to 3.

Abadi et al. (2016) Deep Learning with Differential Privacy: https://arxiv.org/pdf/1607.00133.pdf

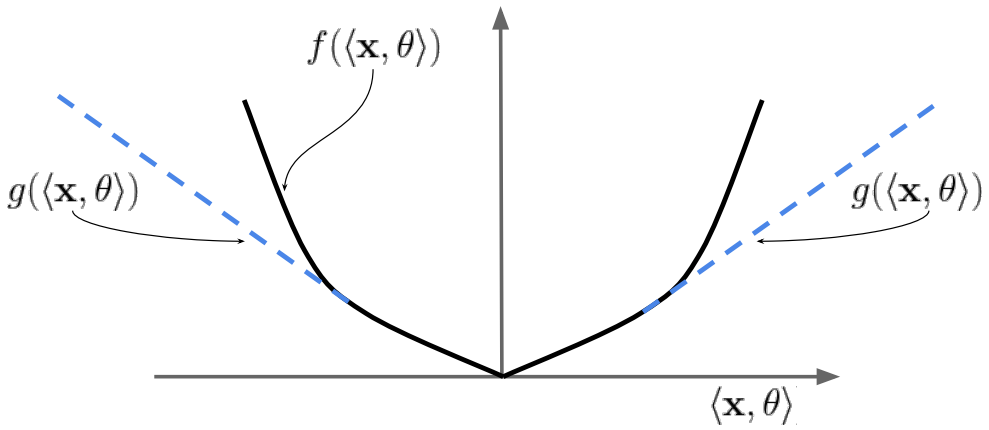
Per-Example Clipped Gradient Descent may not converge

- Example: Two data points / linear losses

Effect of gradient clipping in Generalized Linear Model

- Fixing clipping threshold B, there exists another convex GLM that the algorithm is optimizing.

$$g_{\mathbf{x}}(y) = \begin{cases} -\frac{B}{\|\mathbf{x}\|_2}(y - y_1) + f(y_1) & \text{for } y \in (-\infty, y_1) \\ f(y) & \text{for } y \in [y_1, y_2] \cap \mathbf{R} \\ \frac{B}{\|\mathbf{x}\|_2}(y - y_2) + f(y_2) & \text{for } y \in (y_2, \infty) \end{cases}.$$



Song et al (2020) https://arxiv.org/abs/2006.06783

Knowledge transfer model via the “Private Aggregation of Teacher Ensemble” (PATE)

- Assume a large private dataset
- A small public dataset (can be unlabeled)

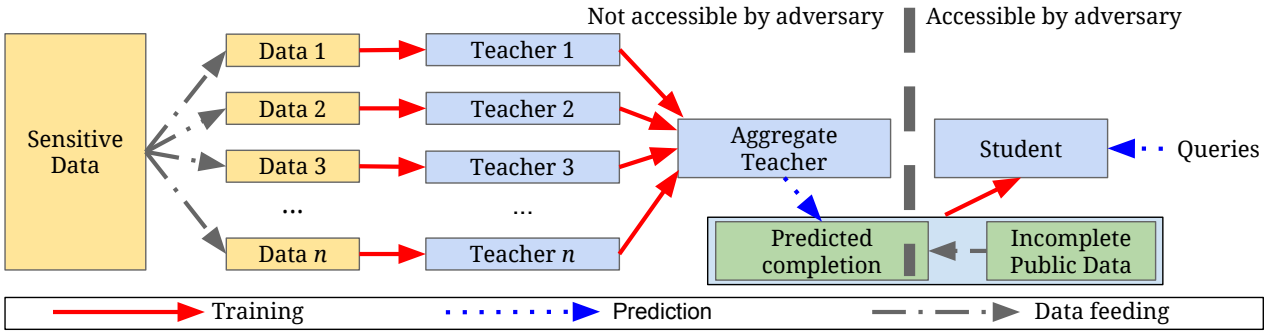


Figure 2: Overview of the approach: (1) an ensemble of teachers is trained on disjoint subsets of the sensitive data, (2) a student model is trained on public data labeled using the ensemble.

Papernot et al. (2018) Scalable private learning with PATE: https://arxiv.org/abs/1802.08908

What does PATE look like from autodp?

- Just a composition of Gaussian mechanisms!

```python
from autodp.mechanism_zoo import GaussianMechanism

import numpy as np

class PATE(GaussianMechanism):
    def __init__(self,sigma, m, Binary,name='PATE'):
        # sigma is the std of the Gaussian noise added to the voting scores
        if Binary:
            # This is a binary classification task
            sensitivity = 1
        else: # for the multiclass case, the L2 sensitivity is sqrt(2)
            sensitivity = np.sqrt(2)
        GaussianMechanism.__init__(self, sigma=sigma/sensitivity/np.sqrt(m),name=name)

        self.params = {'sigma':sigma}
```

- Ideas / tricks that improve to PATE
  - Noisy-Screening and semi-supervised learning (Papernot et al 2017/2018)
  - Release only the argmax rather than voting scores? (Papernot et al 2018)
  - Sparse Vector Technique (Bassily et al., 2018; Liu et al. 2020)
  - Active learning (Liu et al. 2020)

Private K-Nearest Neighbor

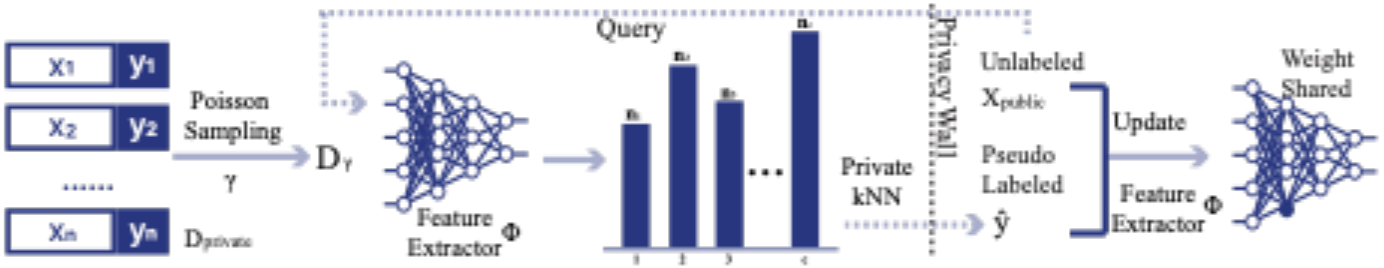


Figure 2. The overview of the proposed framework. Given the unlabeled public data X_{public}, we query through privacy wall for pseudo labels, where the private data and the queried public data are sent through feature extractor Φ and "Private-kNN" to assign pseudo labels. Combining the public data and the pseudo labels, the feature extractor Φ is further updated. This procedure can be iterated for rounds to achieve satisfied privacy-accuracy trade-off.

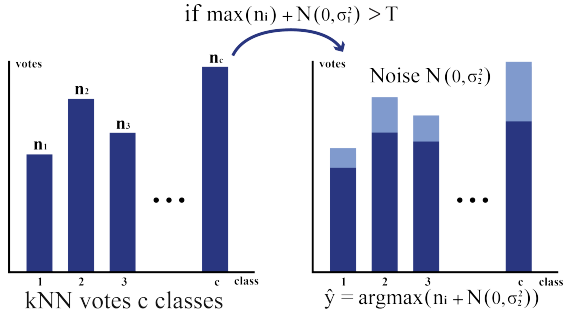


Figure 3. Illustration on the noisy screening and noisy aggregation procedure.

Zhu, Y., Yu, X., Chandraker, M., & Wang, Y. X. (2020). Private-KNN: Practical differential privacy for computer vision. In *CVPR'2020*

What does private KNN look like from autodp?

- Just a composition of Subsampled Gaussian mechanisms (same as NoisySGD!)

```
In [4]: # We will reuse the noisySGD class

        # number of data points

        gamma = 0.01
        m=1000
        sigma = 5.0

        privateKNN = NoisySGD_mech(prob=gamma,sigma=sigma,niter=m)

        # compute epsilon, as a function of delta
        privateKNN.get_approxDP(delta=1e-6)

Out[4]: 0.27105623043762284
```

- Isn't there a NoisyScreening step?

Empirical results in the Knowledge Transfer model

- Private-KNN and PATE:

Table 1. Utility and privacy of semi-supervised student model

Dataset	Methods	#Queries	ϵ	Acc.	NP Acc.
MNIST	LNMAX	1000	8.03	98.1%	99.2%
	GNMAX	286	1.97	98.5%	
	Ours	735	**0.47**	98.8%	
SVHN	LNMAX	1000	8.19	90.1%	92.8%
	GNMAX	3098	4.96	91.6%	
	Ours	2939	**0.49**	91.6%	
CIFAR-10	GNMAX			$\leq$ 50%	80.5%
	Noisy SGD		4	70%	
	Ours	3877	**2.92**	70.8%	

Theory behind PATE and algorithmic improvements

Table 1: Summary of our results: excess risk bounds for PATE algorithms.

Algorithm	**PATE** (Gaussian Mech.) Papernot et al. [2017]	**PATE** (SVT-based) Bassily et al. [2018a]	**PATE** (SVT-based) This paper	**PATE** (Active Learning) This paper
Realizable	$\tilde{O}\left(\frac{d}{(n\epsilon)^{2/3}} \vee \frac{d}{m}\right)$	$\tilde{O}\left(\frac{d}{(n\epsilon)^{2/3}} \vee \sqrt{\frac{d}{m}}\right)$	$\tilde{O}\left(\frac{d^{3/2}}{n\epsilon} \vee \frac{d}{m}\right)$	$\tilde{O}\left(\frac{d^{3/2}\theta^{1/2}}{n\epsilon} \vee \frac{d}{m}\right)$
τ-TNC	$\tilde{O}\left(\left(\frac{d^{3/2}}{n\epsilon}\right)^{\frac{2\tau}{4-\tau}} \vee \frac{d}{m}\right)$	same as agnostic	$\tilde{O}\left(\left(\frac{d^{3/2}}{n\epsilon}\right)^{\frac{\tau}{2-\tau}} \vee \frac{d}{m}\right)$	$\tilde{O}\left(\left(\frac{d^{3/2}\theta^{1/2}}{n\epsilon}\right)^{\frac{\tau}{2-\tau}} \vee \frac{d}{m}\right)$
Agnostic (vs h^*)	$\Omega(\texttt{Err}(h^*))$ required.	$13\texttt{Err}(h^*)+$ $\tilde{O}\left(\frac{d^{3/5}}{n^{2/5}\epsilon^{2/5}} \vee \sqrt{\frac{d}{m}}\right)$	$\Omega(\texttt{Err}(h^*))$ required.	$\Omega(\texttt{Err}(h^*))$ required.
Agnostic (vs $h_\infty^{\texttt{agg}}$)	-	-	Consistent under weaker conditions.	-

- What do these results say:
  - Sparse-Vector-Technique-based online query release helps
  - Active learning helps
  - Label-noise models help

Liu, Zhu, Chaudhuri and W. (2020) "Revisiting model-agnostic private learning". AISTATS and JMLR. https://arxiv.org/pdf/2011.03186.pdf

Next lecture

- Going beyond the worst case!
- Smoothed Sensitivity and the Median