Deep Learning With Differential Privacy

Chenhao Jiang
515021910104

1 Introduction

2 Related Work

3 My Work

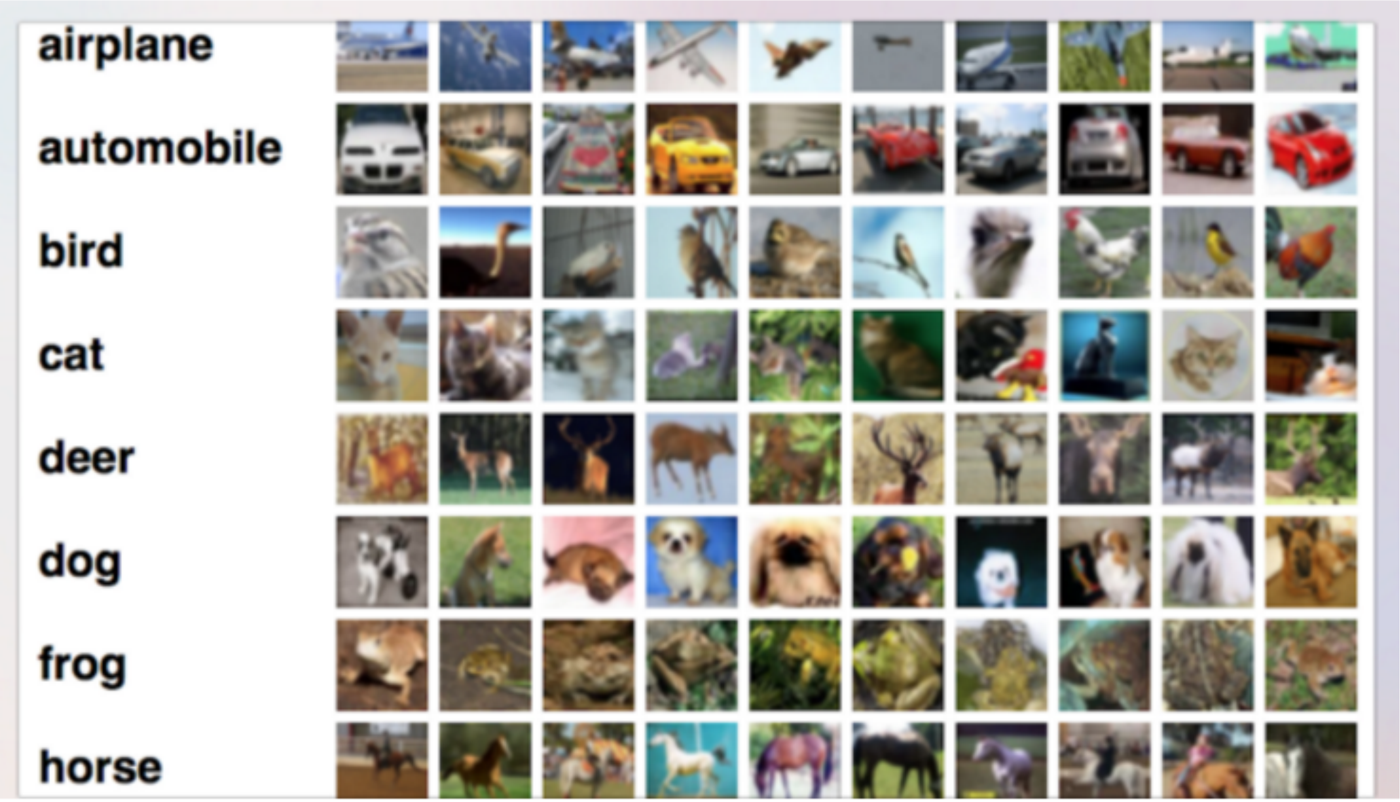
4 Result

PART 1

Introduction

1

Machine learning techniques based on neural networks are achieving remarkable results in a wide variety of domains. Often, the training of models requires large, representative datasets, which may be crowdsourced and contain sensitive information.



1

Privacy leaked

Name	Suffering from diabetes
Ross	1
Monica	1
Joey	0
Phoebe	0
Chandler	1

Name	Suffering from diabetes
Ross	1
Monica	1
Joey	0
Phoebe	0
Chandler	0

1

ε-Differential Privacy

A randomized mechanism $\mathcal{M} : \mathcal{D} \rightarrow \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies ϵ-differential privacy if for any two adjacent inputs $d, d' \in \mathcal{D}$ and for any subset of outputs $S \subseteq \mathcal{R}$ it hold that

$$\Pr[M(d) \in S] \leq e^{\epsilon} \Pr[M(d') \in S]$$

(ε,δ)-Differential Privacy

A randomized mechanism $\mathcal{M} : \mathcal{D} \rightarrow \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies (ϵ, δ)-differential privacy if for any two adjacent inputs $d, d' \in \mathcal{D}$ and for any subset of outputs $S \subseteq \mathcal{R}$ it hold that

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

PART 2

Related Work

2

Differentially Private SGD Algorithm

Algorithm 1 Differentially private SGD (Outline)

Input: Examples $\{x_1, \ldots, x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, x_i)$. Parameters: learning rate η_t, noise scale σ, group size L, gradient norm bound C.

Initialize θ_0 randomly

for $t \in [T]$ **do**

Take a random sample L_t with sampling probability L/N

Compute gradient

For each $i \in L_t$, compute $\mathbf{g}_t(x_i) \leftarrow \nabla_{\theta_t} \mathcal{L}(\theta_t, x_i)$

Clip gradient

$\bar{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i) / \max\left(1, \frac{\|\mathbf{g}_t(x_i)\|_2}{C}\right)$

Add noise

$\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L}\sum_i (\bar{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I}))$

Descent

$\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{\mathbf{g}}_t$

Output θ_T and compute the overall privacy cost (ε, δ) using a privacy accounting method.

2

Related Work

They use MNIST and CIFAR to test their algorithm. Their implementation and experiments demonstrate that they can train deep neural networks with non-convex objectives, under a modest privacy budget, and at a manageable cost in software complexity, training efficiency, and model quality.

PART 3

My Work

3

I installed Tensorflow according to the guidance of a CSDN blog on the Internet. And I learned many deep learning algorithms, like CNN, RNN and LSTM. And I learned how to train a multi-layer convolutional neural network on TensorFlow. Then I tried to design a deep learning algorithm with differential privacy. However, I failed.

```
(tensorflow) C:\Users\john\Desktop>python
Python 3.6.4 |Anaconda, Inc.| (default, Mar 12 2018, 20:20:50) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> hello=tf.constant('hello,tensorflow')
>>> sess=tf.Session()
2018-05-27 16:33:21.809348: I T:\src\github\tensorflow\tensorflow\core\platform\cpu_feature_guard.cc:1
sorFlow binary was not compiled to use: AVX2
>>> print(sess.run(hello))
b'hello,tensorflow'
```

3

Linear Regression

Hypothesis:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_d x_d.$$

Loss Function:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$= \frac{1}{2m}\left[\sum_{0\le j,k\le d}\left(\sum_{i=1}^{m} x_j^{(i)} x_k^{(i)}\right)\theta_j\theta_k - 2\sum_{j=0}^{d}\left(\sum_{i=1}^{m} y^{(i)} x_j^{(i)}\right)\theta_j + \sum_{i=1}^{m} {y^{(i)}}^2\right]$$

3

Algorithm Design

I split the loss function into two child functions $g(\theta)$ and $t(\theta)$. Let Δ_1 indicates the sensitivity of $g(\theta)$, Δ_2 indicates the sensitivity of $b(\theta)$. Assume that two adjacent dataset are different in last training example. We can calculate Δ_1, Δ_2 according to the definition of sensitivity $\Delta_f = \max_{D_1,D_2} \|f(D_1) - f(D_2)\|$:

$$\Delta_1 = \sum_{0\leq j,k\leq d} \|x_j^{(n)} x_k^{(n)} - x_j^{'(n)} x_k^{'(n)}\| \leq 2(d+1)^2$$

$$\Delta_2 = \sum_{j=0}^{d} \|(-2y^{(n)} x_j^{(n)} - (-2y^{'(n)} x_j^{'(n)}\| \leq 4(d+1)$$

3

Algorithm Design

According to function $g(\theta)$ and $t(\theta)$, we assign privacy budget ϵ_1, ϵ_2. According to the characteristics of the Laplace mechanism, the Laplacian noise is proportional to the sensitivity of the function and inversely proportional to the privacy budget. In order to reduce the added noise, we allocate a larger privacy budget for the more sensitive functions. According to the computed Δ_1, Δ_2, when d is large, the sensitivity of $g(\theta)$ is larger than $t(\theta)$. Thus, we should make $\epsilon_1 \geq \epsilon_2$, i.e., for a fixed privacy budget ϵ, we can reduce noise by rationally allocating ϵ_1 and ϵ_2. And it is more accurate to predict linear regression models.

3

Algorithm Design

In general, I want to summarize my algorithm. First, I assign privacy budget ϵ_1 and ϵ_2. And then I split the loss function into two child functions $g(\theta)$ and $t(\theta)$, where $g(\theta) = a\theta^2$, $t(\theta) = b\theta$. After doing this, I add Laplacian noise to the coefficient a and b as $\hat{a} = a + Lap(\frac{\Delta_1}{\epsilon_1})$, $\hat{b} = b + Lap(\frac{\Delta_2}{\epsilon_2})$. And then optimize the new loss function to get $\hat{\theta}$.

3

Algorithm Design

I first want to prove that function $\hat{g}(\theta)$ satisfies ϵ_1-differentially privacy, where $\hat{g}(\theta) = \hat{a}\theta^2$.

Proof: Given two adjacent datasets D_1, D_2 different in last training example. We have

$$
\begin{aligned}
\frac{\Pr\{\hat{g}(\theta)|D_1\}}{\Pr\{\hat{g}(\theta)|D_2\}} &= \frac{\prod_{0\le j,k\le d} exp(\frac{\epsilon_1 \| \sum_{D_1}(x_j^{(i)}x_k^{(i)} - \hat{a}\|}{\Delta_1})}{\prod_{0\le j,k\le d} exp(\frac{\epsilon_1 \| \sum_{D_2}(x_j^{(i)}x_k^{(i)} - \hat{a}\|}{\Delta_1})} \\
&\le \prod_{0\le j,k\le d} exp(\frac{\epsilon_1}{\Delta_1}\|(\sum_{D_1} x_j^{(i)}x_k^{(i)}) - (\sum_{D_2} x_j^{(i)}x_k^{(i)})\|) \\
&= \prod_{0\le j,k\le d} exp(\frac{\epsilon_1}{\Delta_1}\|x_j^{(n)}x_k^{(n)} - x_j^{'(n)}x_k^{'(n)}\|) \\
&= exp(\frac{\epsilon_1}{\Delta_1}\sum_{0\le j,k\le d}\|x_j^{(n)}x_k^{(n)} - x_j^{'(n)}x_k^{'(n)}\|) \\
&\le exp(\epsilon_1)
\end{aligned}
$$

Also, we can prove that $\hat{t}(\theta)$ satisfies ϵ_2-differentially privacy. Therefore, the algorithm satisfies ϵ-differentially privacy, where $\epsilon = \epsilon_1 + \epsilon_2$.

PART 4

Result

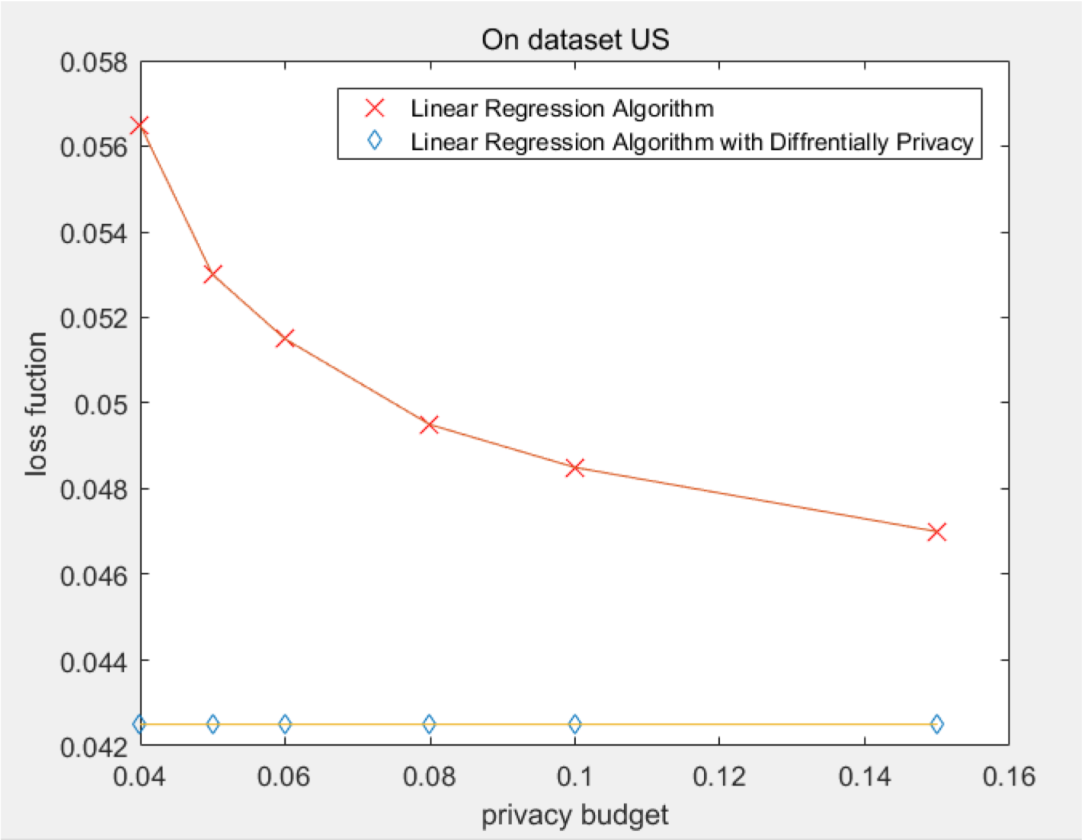
4

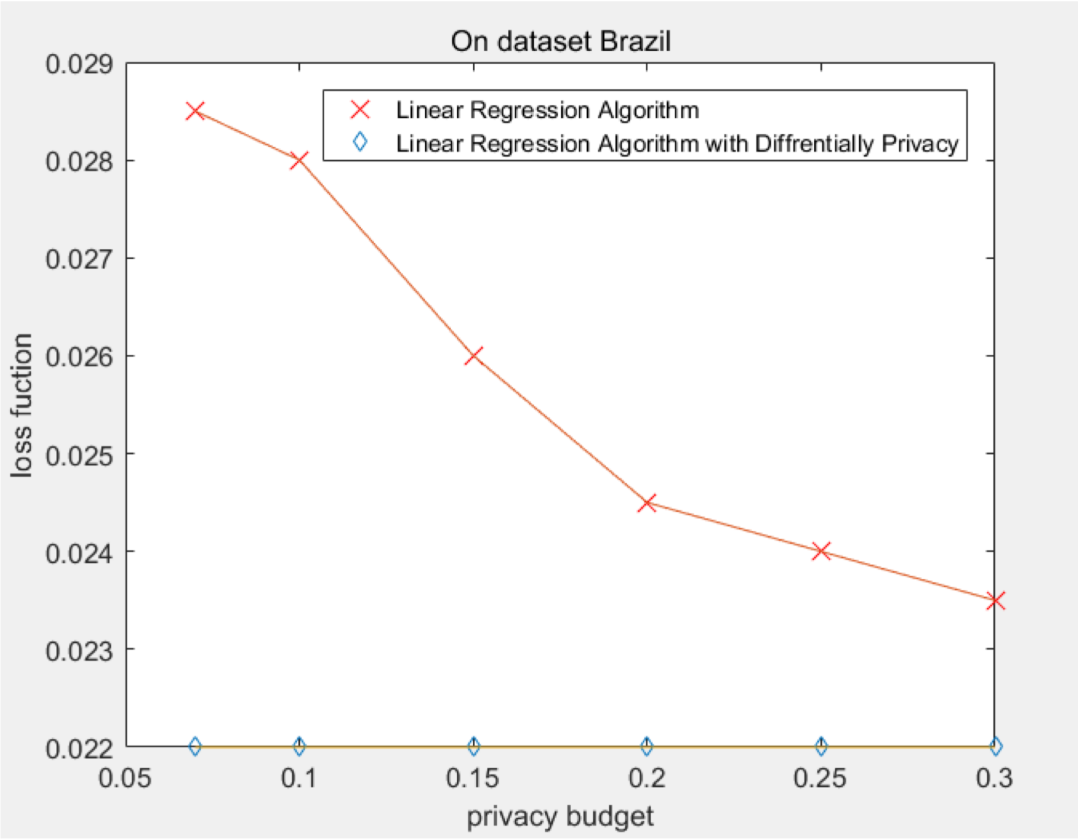
Algorithm Performance Test

I use Matlab to test the algorithm, the dataset is US and Brazil from Integrated Public Use Microdata, including 370000 examples and 190000. I first use feature scale to process the data. And then I randomly take 80% examples as training examples and 20% as test examples.

4

Algorithm Performance Test





4

Algorithm Performance Test

As we can see, the loss function of linear regression algorithm with differential privacy is larger than the loss function of linear regression algorithm on both datasets. And the loss function of linear regression algorithm with differential privacy reduces with the increase of privacy budget.

References

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I.Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. CoRR, abs/1607.00133, 2016.

[2] CSDN Blog
https://blog.csdn.net/cs_hnu_scw/article/details/79695347

[3] TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[4] C. Dwork and J. Lei. Differential privacy and robust statistics. In STOC, pages 371–380. ACM, 2009.

[5] C. Dwork and G. N. Rothblum. Concentrated differential privacy. CoRR, abs/1603.01887, 2016.

[6] Minnesota Population Center. Ingrated public use micodata series-inter-nation: Version 5.0[OL].[2009]. https://internationnal.ipums.org.

THANK YOU!