Transfer learning for regression under differential privacy

Zhenhao Zhang and Yuquan Song

East China University of Petroleum

upc.edu.cn

September 15, 2022

Overview

Differential Privacy

Definition 4 (Classical differential privacy [50]). *An algorithm $\mathcal{A}$ is (ϵ, δ)-differential private if for any two neighborhood datasets $\mathbf{X}$ and $\mathbf{X}'$ with $\mathbf{X}, \mathbf{X}' \in \mathbb{R}^{N \times d}$, and for all measurable sets $\mathcal{O} \subseteq Range(\mathcal{A})$, the following holds:*

$$\Pr(\mathcal{A}(\mathbf{X}) \in \mathcal{O}) \leq e^{\epsilon} \Pr\big(\mathcal{A}(\mathbf{X}') \in \mathcal{O}\big) + \delta. \tag{7}$$

Here the neighborhood datasets $\mathbf{X}$ and $\mathbf{X}'$ refer that the number of rows in $\mathbf{X}$ that need to be modified (e.g., moved) to get the $\mathbf{X}'$ is one.

Algorithm 2 $\mathcal{A}_{\mathsf{Noise-FW(polytope)}}$: Differentially Private Frank-Wolfe Algorithm (Polytope Case)

Input: Data set: $\mathcal{D} = \{d_1, \cdots, d_n\}$, loss function: $\mathcal{L}(\theta; D) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(\theta; d_i)$ (with ℓ_1-Lipschitz constant L_1 for $\mathcal{L}$), privacy parameters: (ϵ, δ), convex set: $\mathcal{C} = conv(S)$ with $\|\mathcal{C}\|_1$ denoting $\max_{s \in S} \|s\|_1$.

1: Choose an arbitrary θ_1 from $\mathcal{C}$
2: **for** $t = 1$ to $T - 1$ **do**
3: $\forall s \in S, \alpha_s \leftarrow \langle s, \triangledown \mathcal{L}(\theta_t; D)\rangle + \mathsf{Lap}\left(\frac{L_1 \|\mathcal{C}\|_1 \sqrt{8T \log(1/\delta)}}{n\epsilon}\right)$, where $\mathsf{Lap}(\lambda) \sim \frac{1}{2\lambda} e^{-|x|/\lambda}$.
4: $\widetilde{\theta}_t \leftarrow \arg\min_{s \in S} \alpha_s$.
5: $\theta_{t+1} \leftarrow (1 - \mu_t)\theta_t + \mu_t \widetilde{\theta}_t$, where $\mu_t = \frac{2}{t+2}$.
6: Output $\theta^{priv} = \theta_T$.

for dataset $\mathrm{A}_0(\mathit{the \quad original \quad dataset})A_2.A_3...A_n(\mathit{To \quad be \quad migrated})$

model: $y_i^{(0)} = (x_i^{(0)})^\intercal \beta + \epsilon_i^{(0)},\ i = 1, \ldots, n_0,$

$$\delta^{(k)} = \beta - w^{(k)}$$

$\mathrm{A}_0, A_1...A_n$:

$$\mathcal{A}_q = \{1 \leq k \leq K : \|\delta^{(k)}\|_q \leq h\},$$

Our Work

where the data set is

$$A = [A_1, A_2 ... A_n]$$

where A1 represents the original data set, A2...An represents the dataset to be migrated, and all the datasets satisfy uniform distribution

$$A_1 \sim U(a1, b1), A_2 \sim U(a2, b2) ... A_n \sim U(an, bn),$$

the following regression model is established.

$$y = \omega^T x + b$$

where y is the output quantity, ω is the coefficient matrix, x is the input quantity, and b is the constant residual term. The loss function of this regression can be expressed as

$$L(\theta) = \frac{1}{m} \sum_{i=0}^{m} (y^i - \omega^T (x^i))^2 + \lambda ||\omega^T||_q$$

Our Work

step1:add noise

```
for i in range(50):
    grad_a, grad_b = [0 for i in range(len(x[0]))], 0
    grad_a+=add_laplace_noise(0,sum(a)/len(a),len(grad_a))
    for i in range(m):

        common=0
```

step2:transfer learning

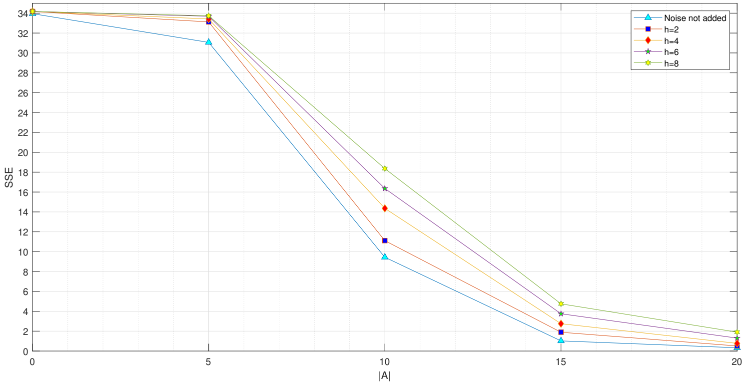
```
for i in range(6,11):
    x.append([float(i),float(i)])
for i in range(6,11):
    y.append(float(2*i+1))

sumx=0.0
for i in range(100):
    sumx+=solve_by_gradient(x,y)
print("Migrate 5 dataset,sse=",sumx/100)
```

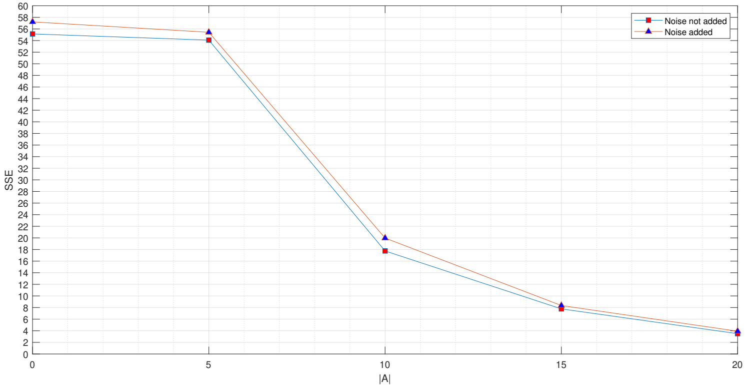
Our Work

```
Migrate 0 dataset,sse= 34.17541071316984
Migrate 5 dataset,sse= 33.12849629296133
Migrate 10 dataset,sse= 11.112535857610235
Migrate 15 dataset,sse= 1.8909878040817576
Migrate 20 dataset,sse= 0.5093283949462145
```

Simulation Studies



Simulation Studies



End

Thanks for listening