这次介绍蚂蚁、山东师范大学和阿里一起发表在KDD'21上的论文《When Homomorphic Encryption Marries Secret Sharing: Secure Large-Scale Sparse Logistic Regression and Applications in Risk Control》。在该论文中，Chaochao Chen等人提出了 CAESAR 在数据纵向分布的场景下，安全两方的Logistic Regression模型的训练。并且针对实际应用场景中的数据稀疏性问题（Sparse）做了优化，使得 CAESAR 比 SecureML （基于秘密分享的方案）高效 $130\times$。

0. 背景和基础知识

数据纵向分布场景下的联合建模比数据横向分布更加具有挑战性。已有的方案例如 SecureML，虽然可以使用秘密分享将数据分布到不合谋的服务器上，能够实现可证明安全。但是分享过程中会将原本系数的数据变为稠密数据（Dense），从而会大大降低性能（如图1）。而目前的基于同态加密的纵向解决方案虽然能够利用稀疏性加速，但是训练过程中需要公开模型更新参数，这会导致隐私泄露的风险。本文将同态加密和秘密分享结合起来，在保证可证明安全的要求下充分利用数据的稀疏性加速优化。
考虑到实际场景中的应用，本文考虑两方下（$\mathcal{A}$ 和 $\mathcal{B}$）的建模，并且抵抗半诚实敌手。

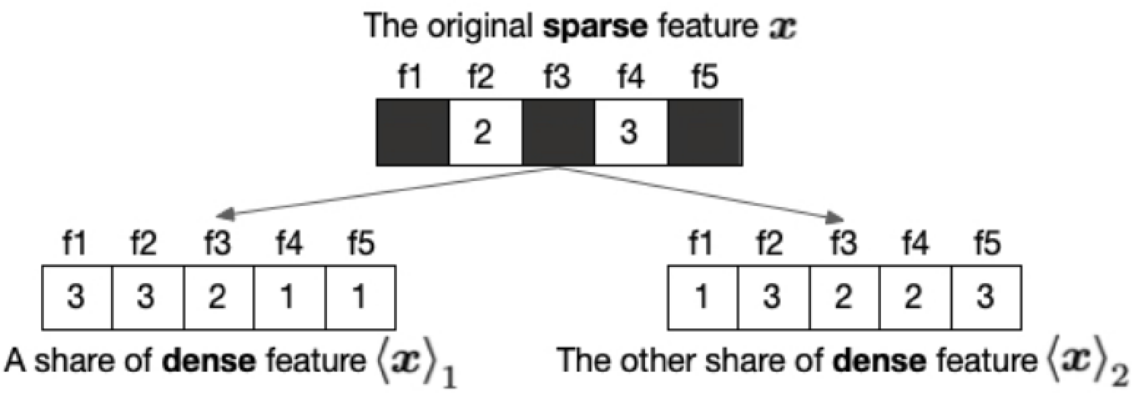


Figure 1: Sparse feature becomes dense features after using secret sharing in $\mathbb{Z}_4$.

本文需要的基础知识如下：

1. 加法秘密分享：取大整数 $\phi=2^\ell$。在环 $\mathbb{Z}_\phi$ 中的加法秘密分享可以记作 $\langle\cdot\rangle$。$\langle\cdot\rangle$ 下的加法可以本地计算，乘法则需要利用Beaver三元组交互计算得到；
2. 同态加密：本文只需要半同态加密即可，例如 Okamoto-Uchiyama encryption (OU) 和 Paillier方案。同态加密的密文记作 $[\![\cdot]\!]$；
3. 在整数环下编码浮点数：和之前的方案一致，取 $\lfloor 10^c x \rceil \mod \phi$。
4. Logistic Regression: 对于数据集 $\mathcal{D}=\{(\mathbf{x}_i,y_i)\}_{i=1}^n$，Logistic Regreesion 的目标是学的 $\mathbf{w}$ 从而使得损失函数 $\mathcal{L}=\sum_{i=1}^n l(y_i,\widehat{y}_i)$ 最小。其中 $l(y_i,\widehat{y}_i)=-y\cdot\log(\widehat{y}_i)-(1-y)\cdot\log(1-\widehat{y}_i)$，$\widehat{y}_i=1/(1+e^{-\mathbf{x}_i\cdot\mathbf{w}})$；在实际优化中，常取一个batch的数据 $(\mathbf{X}_B,\mathbf{Y}_B)$ 来优化 $\mathbf{w}$ 如下：

$$\mathbf{w} \leftarrow \mathbf{w} - \frac{\alpha}{|\mathbf{B}|} \cdot \frac{\partial \mathcal{L}}{\partial \mathbf{w}},$$

其中 $\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = (\widehat{\mathbf{Y}}_B - \mathbf{Y}_B)^T \cdot \mathbf{X}_B$；

5. 为了近似计算Sigmoid函数，本文采用了 Minimax近似法，三次多项式。

1. CAESAR Design

本文首先基于同态加密和秘密分享提出了稀疏矩阵的乘法，然后基于乘法协议提出了模型训练协议。

1.1 Secure Sparse Matrix Multiplication

该协议的构造和两方下利用同态加密生成Beaver三元组的交叉项很相似：首先 $\mathcal{B}$ 加密将自己的输入 $\mathbf{Y}$ 得到 $[\![\mathbf{Y}]\!]_b$，将密文传给 $\mathcal{A}$；$\mathcal{A}$ 在本地利用同态加密计算明文-密文乘积得到 $[\![\mathbf{Z}]\!]_b$，并利用随机数茫化 $[\![\mathbf{Z}]\!]_b$ 得到使得两方得到 $\mathbf{Z} = \mathbf{X} \cdot \mathbf{Y}$ 的秘密分享：$\langle \mathbf{Z} \rangle_1$ 和 $\langle \mathbf{Z} \rangle_2$。具体协议如下：

Protocol 1: Secure Sparse Matrix Multiplication

Input: A sparse matrix $\mathbf{X}$ hold by $\mathcal{A}$, a matrix $\mathbf{Y}$ hold by $\mathcal{B}$, HE key pair for $\mathcal{A}$ ($\{pk_a, sk_a\}$), HE key pair for $\mathcal{B}$ ($\{pk_b, sk_b\}$)

Output: $\mathbf{Z}_1$ for $\mathcal{A}$ and $\mathbf{Z}_2$ for $\mathcal{B}$ thus that $\mathbf{Z}_1 + \mathbf{Z}_2 = \mathbf{X} \cdot \mathbf{Y}$

1 $\mathcal{B}$ encrypts $\mathbf{Y}$ with pk_b and sends $[\![\mathbf{Y}]\!]_b$ to $\mathcal{A}$

2 $\mathcal{A}$ calculates $[\![\mathbf{Z}]\!]_b = \mathbf{X} \cdot [\![\mathbf{Y}]\!]_b$

3 $\mathcal{A}$ secretly shares $[\![\mathbf{Z}]\!]_b$ using Protocol 2, and after that $\mathcal{A}$ gets $\mathbf{Z}_1$ and $\mathcal{B}$ gets $\mathbf{Z}_2$

4 **return** $\mathbf{Z}_1$ for $\mathcal{A}$ and $\mathbf{Z}_2$ for $\mathcal{B}$

Protocol 2: Secret Sharing in Homomorphically Encrypted Field

Input: Homomorphically encrypted matrix $[\![\mathbf{Z}]\!]_b$ for $\mathcal{A}$, HE key pair for $\mathcal{B}$ $(\{pk_b, sk_b\})$

Output: $\langle\mathbf{Z}\rangle_1$ for $\mathcal{A}$ and $\langle\mathbf{Z}\rangle_2$ for $\mathcal{B}$

1. $\mathcal{A}$ locally generates share $\langle\mathbf{Z}\rangle_1$ from $\mathbb{Z}_\phi$
2. $\mathcal{A}$ calculates $[\![\langle\mathbf{Z}\rangle_2]\!]_b = [\![\mathbf{Z}]\!]_b - \langle\mathbf{Z}\rangle_1 \mod \psi$ and sends $[\![\langle\mathbf{Z}\rangle_2]\!]_b$ to $\mathcal{B}$
3. $\mathcal{B}$ decrypts $[\![\langle\mathbf{Z}\rangle_2]\!]_b$ and gets $\langle\mathbf{Z}\rangle_2$
4. **return** $\langle\mathbf{Z}\rangle_1$ for $\mathcal{A}$ and $\langle\mathbf{Z}\rangle_2$ for $\mathcal{B}$

需要注意的是，上述乘法不能利用SIMD技术优化。因此，只使用数据特别稀疏的场景。如果数据的稀疏性不强，则效果不如利用支持SIMD的全同态或者Level-同态方案。

1.2 安全训练协议

CAESAR的安全训练协议如下所示：首先是 line 5-6 对各自拥有的部分初始化模型进行秘密分享；在T轮训练中，两方在 line 10-13 安全计算 $\mathbf{X}\cdot\mathbf{w}$；而 line14-15 则利用多项式近似计算 Sigmoid；line 16 计算输出的秘密分享 $\langle\hat{\mathbf{y}}\rangle$；接下来 line 18-19 计算 error；line 21-24 计算更新梯度；尔后 line 26-27 在秘密分享下更新模型。训练得到的模型两方各自恢复自己的那一部分（line 30-31）。

Algorithm 1: CAESAR: seCure lArge-scalE SpArse logistic Regression

Input: features for party $\mathcal{A}$ ($\mathbf{X}_a$), features for party $\mathcal{B}$ ($\mathbf{X}_b$), labels for $\mathcal{B}$ ($\mathbf{y}$), HE key pair for $\mathcal{A}$ ($\{pk_a, sk_a\}$), HE key pair for $\mathcal{B}$ ($\{pk_b, sk_b\}$), max iteration number (T), and polynomial coefficients (q_0, q_1, q_2)

Output: models for party $\mathcal{A}$ ($\mathbf{w}_a$) and models for party $\mathcal{B}$ ($\mathbf{w}_b$)

1 **Initialization:**

2 $\mathcal{A}$ and $\mathcal{B}$ initialize their logistic regression models, i.e., $\mathbf{w}_a$ and $\mathbf{w}_b$, respectively

3 $\mathcal{A}$ and $\mathcal{B}$ exchange their public key pk_a and pk_b

4 **Secretly share models:**

5 $\mathcal{A}$ locally generates shares $\langle \mathbf{w}_a \rangle_1$ and $\langle \mathbf{w}_a \rangle_2$, keeps $\langle \mathbf{w}_a \rangle_1$, and sends $\langle \mathbf{w}_a \rangle_2$ to $\mathcal{B}$

6 $\mathcal{B}$ locally generates shares $\langle \mathbf{w}_b \rangle_1$ and $\langle \mathbf{w}_b \rangle_2$, keeps $\langle \mathbf{w}_b \rangle_2$, and sends $\langle \mathbf{w}_b \rangle_1$ to $\mathcal{A}$

7 **Training model:**

8 **for** $t = 1$ *to* T **do**

9 **Calculate prediction:**

10 $\mathcal{A}$ calculates $\langle \mathbf{z}_a \rangle_1 = \mathbf{X}_a \cdot \langle \mathbf{w}_a \rangle_1$

11 $\mathcal{A}$ and $\mathcal{B}$ securely calculate $\langle \mathbf{z}_a \rangle_2 = \mathbf{X}_a \cdot \langle \mathbf{w}_a \rangle_2$ using Protocol 1, and after that $\mathcal{A}$ gets $\langle\langle \mathbf{z}_a \rangle_2\rangle_1$ and $\mathcal{B}$ gets the result $\langle\langle \mathbf{z}_a \rangle_2\rangle_2$

12 $\mathcal{B}$ calculates $\langle \mathbf{z}_b \rangle_2 = \mathbf{X}_b \cdot \langle \mathbf{w}_b \rangle_2$

13 $\mathcal{A}$ and $\mathcal{B}$ securely calculate $\langle \mathbf{z}_b \rangle_1 = \mathbf{X}_b \cdot \langle \mathbf{w}_b \rangle_1$ using Protocol 1, and after that $\mathcal{A}$ gets $\left\langle\langle \mathbf{z}_b \rangle_1\right\rangle_1$ and $\mathcal{B}$ gets the result $\left\langle\langle \mathbf{z}_b \rangle_1\right\rangle_2$

14 $\mathcal{A}$ calculates $\langle \mathbf{z} \rangle_1 = \langle \mathbf{z}_a \rangle_1 + \langle\langle \mathbf{z}_a \rangle_2\rangle_1 + \left\langle\langle \mathbf{z}_b \rangle_2\right\rangle_1$, $\langle \mathbf{z} \rangle_1^2$, and $\langle \mathbf{z} \rangle_1^3$ and sends ciphertext $\llbracket \langle \mathbf{z} \rangle_1 \rrbracket_a$, $\llbracket \langle \mathbf{z} \rangle_1^2 \rrbracket_a$, and $\llbracket \langle \mathbf{z} \rangle_1^3 \rrbracket_a$ to $\mathcal{B}$

15 $\mathcal{B}$ calculates $\langle \mathbf{z} \rangle_2 = \langle \mathbf{z}_b \rangle_1 + \langle\langle \mathbf{z}_a \rangle_2\rangle_2 + \left\langle\langle \mathbf{z}_b \rangle_2\right\rangle_2$, $\llbracket \mathbf{z} \rrbracket_a = \llbracket \langle \mathbf{z} \rangle_1 \rrbracket_a + \langle \mathbf{z} \rangle_2$, and $\llbracket \mathbf{z}^3 \rrbracket_a = \llbracket \langle \mathbf{z} \rangle_1^3 \rrbracket_a + 3\llbracket \langle \mathbf{z} \rangle_1^2 \rrbracket_a \odot \langle \mathbf{z} \rangle_2 + 3\llbracket \langle \mathbf{z} \rangle_1 \rrbracket_a \odot \langle \mathbf{z} \rangle_2^2 + \langle \mathbf{z} \rangle_2^3$

16 $\mathcal{B}$ calculates $\llbracket \hat{\mathbf{y}} \rrbracket = q_0 + q_1 \llbracket \mathbf{z} \rrbracket_a + q_2 \llbracket \mathbf{z}^3 \rrbracket_a$, and secretly shares $\llbracket \hat{\mathbf{y}} \rrbracket_a$ using Protocol 2, and after that $\mathcal{A}$ gets $\langle \hat{\mathbf{y}} \rangle_1$ and $\mathcal{B}$ gets $\langle \hat{\mathbf{y}} \rangle_2$

17 **Calculate shared error:**

18 $\mathcal{A}$ calculates error $\langle \mathbf{e} \rangle_1 = \langle \hat{\mathbf{y}} \rangle_1$

19 $\mathcal{B}$ calculates error $\langle \mathbf{e} \rangle_2 = \langle \hat{\mathbf{y}} \rangle_2 - \mathbf{y}$

20 **Calculate gradients:**

21 $\mathcal{B}$ locally calculates $\llbracket \mathbf{g}_b \rrbracket_a = \llbracket \mathbf{e} \rrbracket_a^T \cdot \mathbf{X}_b$

22 $\mathcal{B}$ secretly shares $\llbracket \mathbf{g}_b \rrbracket_a$ using Protocol 2, and after that $\mathcal{A}$ gets $\langle \mathbf{g}_b \rangle_1$ and $\mathcal{B}$ gets $\langle \mathbf{g}_b \rangle_2$

23 $\mathcal{A}$ calculates $\langle \mathbf{g}_a \rangle_1 = \langle \mathbf{e} \rangle_1^T \cdot \mathbf{X}_a$

24 $\mathcal{A}$ and B securely calculate $\langle \mathbf{g}_a \rangle_2 = \langle \mathbf{e} \rangle_2^T \cdot \mathbf{X}_A$ using Protocol 1, and after that $\mathcal{A}$ gets $\langle\langle \mathbf{g}_a \rangle_2\rangle_1$ and $\mathcal{B}$ gets $\langle\langle \mathbf{g}_a \rangle_2\rangle_2$

25 **Update model:**

26 $\mathcal{A}$ updates $\langle \mathbf{w}_a \rangle_1$ and $\langle \mathbf{w}_b \rangle_1$ by $\langle \mathbf{w}_a \rangle_1 \leftarrow \langle \mathbf{w}_a \rangle_1 - \alpha \cdot (\langle \mathbf{g}_a \rangle_1 + \langle\langle \mathbf{g}_a \rangle_2\rangle_1)$ and $\langle \mathbf{w}_b \rangle_1 \leftarrow \langle \mathbf{w}_b \rangle_1 - \alpha \cdot \langle \mathbf{g}_b \rangle_1$

27 $\mathcal{B}$ updates $\langle \mathbf{w}_a \rangle_2$ and $\langle \mathbf{w}_b \rangle_2$ by $\langle \mathbf{w}_a \rangle_2 \leftarrow \langle \mathbf{w}_a \rangle_2 - \alpha \cdot \langle\langle \mathbf{g}_a \rangle_2\rangle_2$ and $\langle \mathbf{w}_b \rangle_2 \leftarrow \langle \mathbf{w}_b \rangle_2 - \alpha \cdot \langle \mathbf{g}_b \rangle_2$

28 **end**

29 **Reconstructing models:**

30 $\mathcal{A}$ sends $\langle \mathbf{w}_b \rangle_1$ to $\mathcal{B}$

31 $\mathcal{B}$ sends $\langle \mathbf{w}_a \rangle_2$ to $\mathcal{A}$

32 $\mathcal{A}$ reconstructs $\mathbf{w}_a = \langle \mathbf{w}_a \rangle_1 + \langle \mathbf{w}_a \rangle_2$

33 $\mathcal{B}$ reconstructs $\mathbf{w}_b = \langle \mathbf{w}_b \rangle_1 + \langle \mathbf{w}_b \rangle_2$

34 **return** models for party $\mathcal{A}$ ($\mathbf{w}_a$) and models for party $\mathcal{B}$ ($\mathbf{w}_b$)

上述协议比较简单，在每一个batch的迭代更新中，line 11 和 13 一共需要通信 $d + 2|\mathbf{B}|$ 个密文；line 14 需要传输 $3|\mathbf{B}|$ 个密文；line 16 需要传输 $|\mathbf{B}|$ 个密文；line 22 需要 d 个密文； line 24 需要 $d + |\mathbf{B}|$ 个密文。一共需要传输 $O(7|\mathbf{B}| + 3d)$ 个密文。对整个数据集跑一次则需要 $O(7n + 3nd/|\mathbf{B}|)$ 的通信。

1.3 Distributed Deployed

在部署的过程中，两方分别使用两个集群。集群中对应的服务器并行协作计算一个局部的预测值分享；然后一个集群中的所有服务器将所有局部预测值合并，再进行后续的更新。而每个服务器又有多个worker，可以加速加密计算。

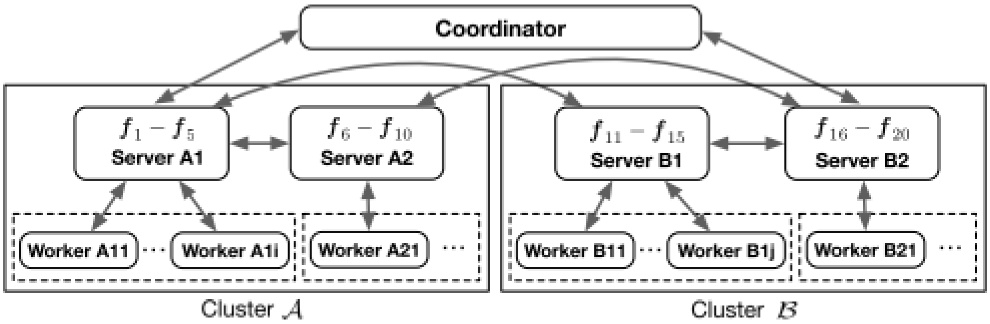


Figure 4: Implementation framework of CAESAR.

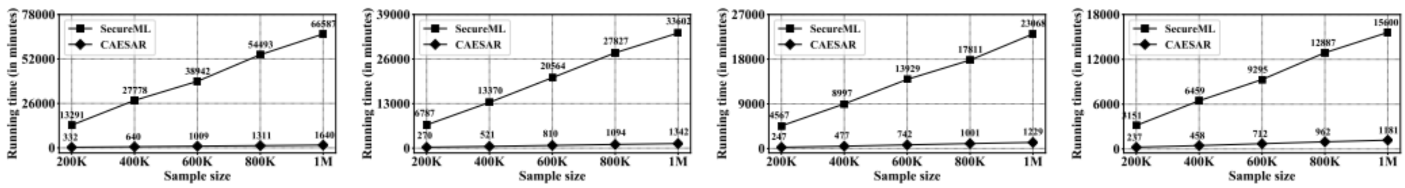
2. Experiments

最后，本文首先验证了联合建模相比于局部建模带来的增益：

Table 1: Comparison results

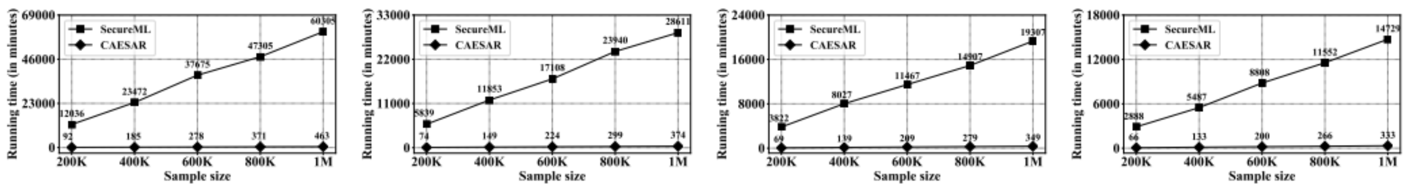
Metric	AUC	KS	F1	Recall@0.9precision
Ant-LR	0.9862	0.9018	0.5350	0.2635
SecureML	0.9914	0.9415	0.6167	0.3598
CAESAR	**0.9914**	**0.9415**	**0.6167**	**0.3598**

进一步，罗列了和SecureML相比，带来的性能提升：



(a) *Bandwidth=10Mbps* (b) *Bandwidth=20Mbps* (c) *Bandwidth=30Mbps* (d) *Bandwidth=40Mbps*

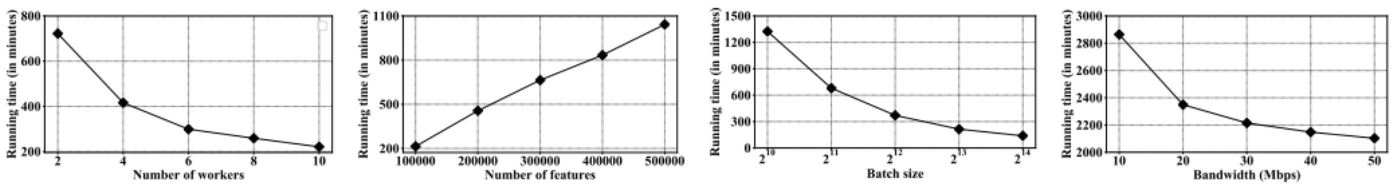
Figure 5: Running time (per epoch) comparison with respect to sample size by varying bandwidth (batch size = 1,024).



(a) *Bandwidth=10Mbps* (b) *Bandwidth=20Mbps* (c) *Bandwidth=30Mbps* (d) *Bandwidth=40Mbps*

Figure 6: Running time (per epoch) comparison with respect to sample size by varying bandwidth (batch size = 4,096).

最后，探索了不同参数设置对于整体性能的影响：



(a) *Effect of worker number* (b) *Effect of feature number* (c) *Effect of batch size* (d) *Effect of network bandwidth*

Figure 7: Effects of parameters on CAESAR.

3. Conclusion

纵向建模一直是一个难点，之前的方法要不利用纯安全多方计算进行，从而无法利用数据稀疏性等加速计算；另一方面的方法则折衷安全性，公开中间的计算结果来加速计算。本文提出了两方下的可证明安全的纵向建模方法。