MPC Techniques for Machine Learning

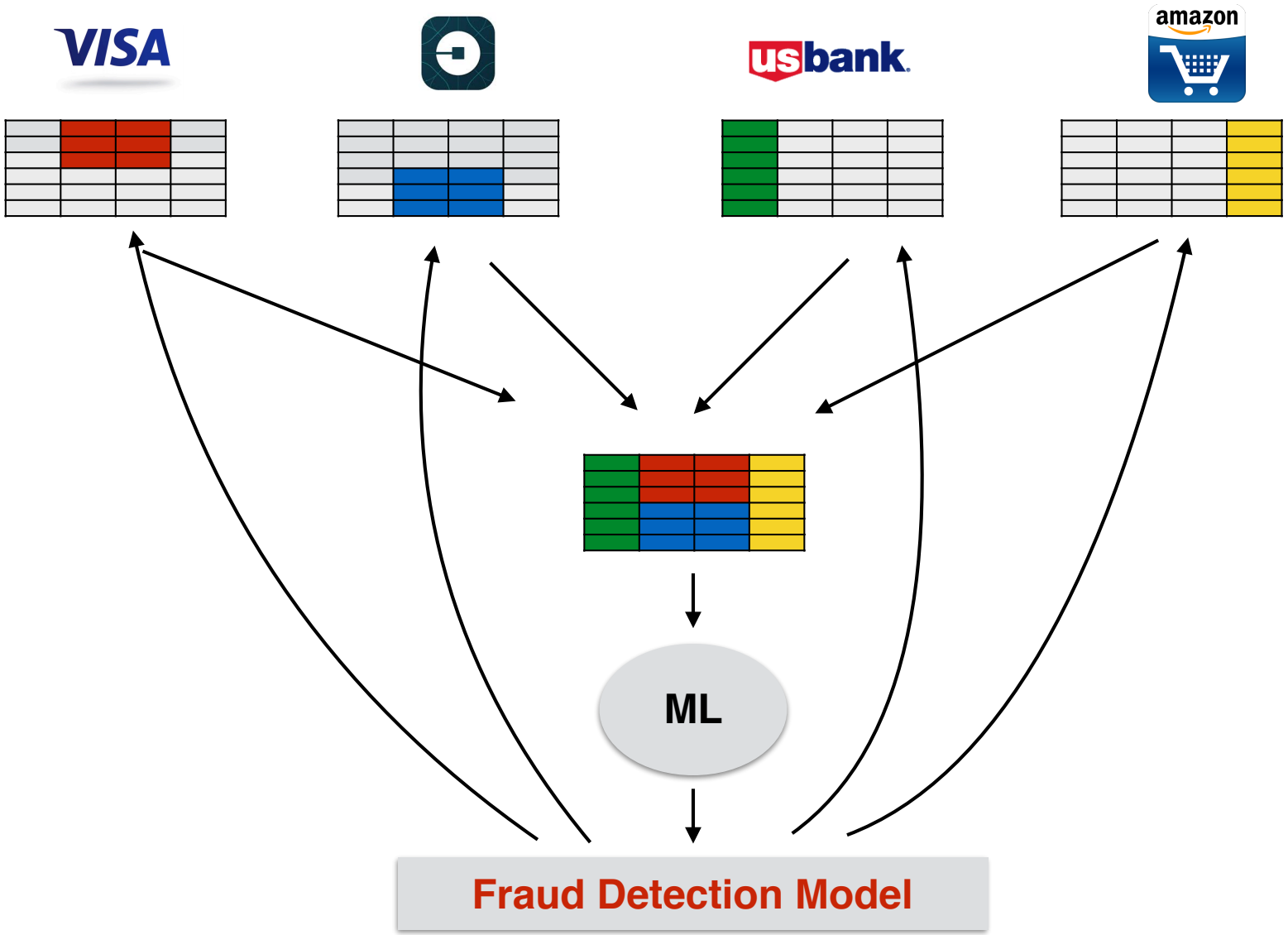
Payman Mohassel
Visa Research

with Yupeng Zhang and Peter Rindal
UMD OSU

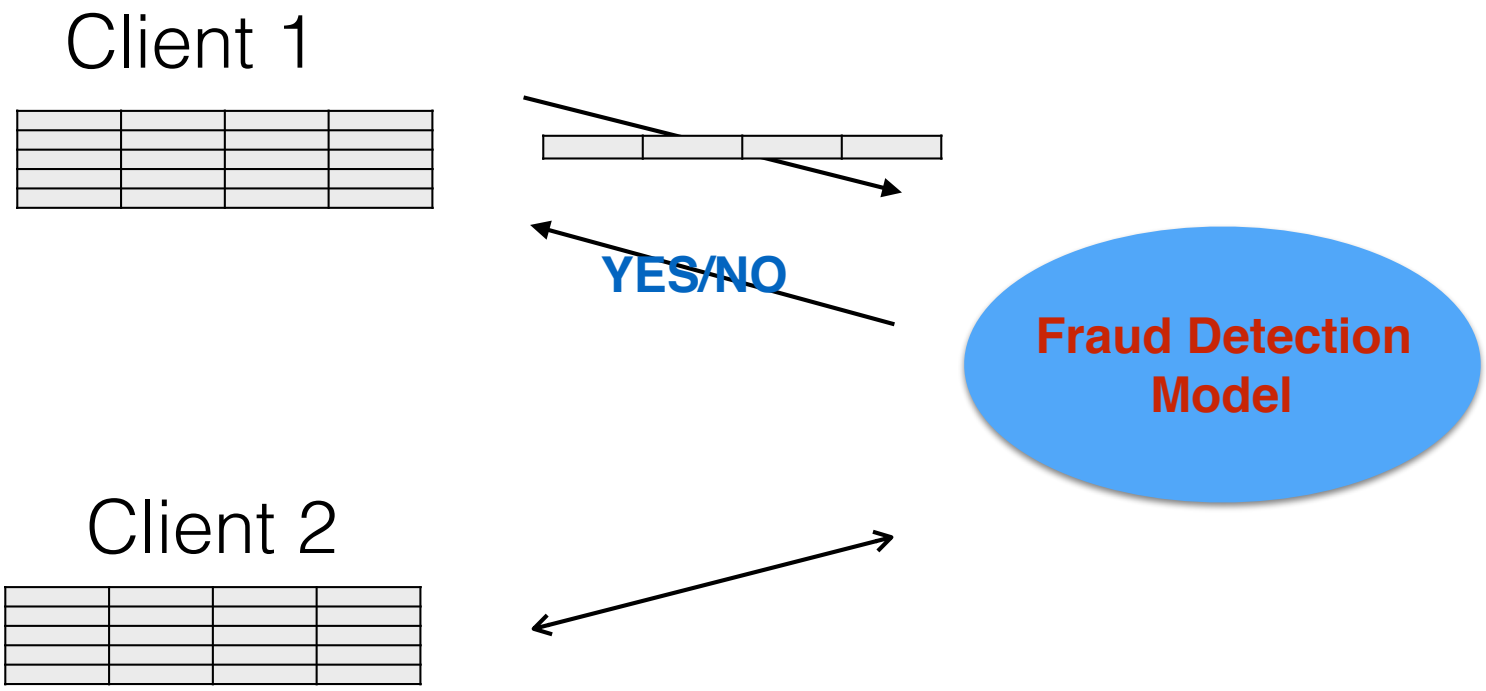
More Data ⇝ Better Models

- Deep learning
  - Breakthroughs in speech, image …
  - Thrives on big data
- Pool datasets
    - Similar data (Horizontal)
    - Complementary data (Vertical)
  - Aggregate user data

Training on Joint Data



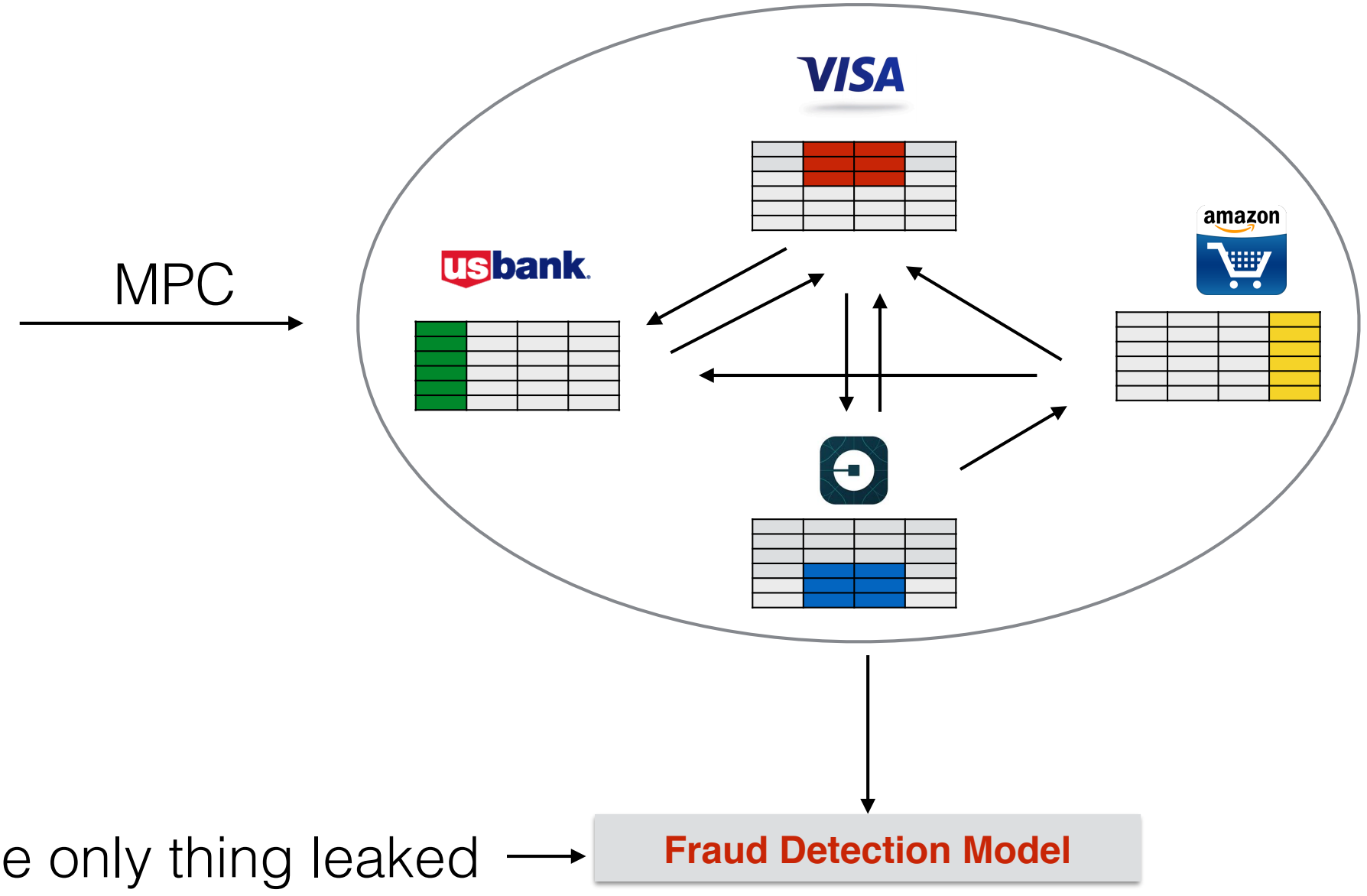
Prediction as a Service



Data Sharing

- Privacy concerns
  - Hard to control once you share
- Regulatory restrictions
- Competitive advantage
- Data sovereignty

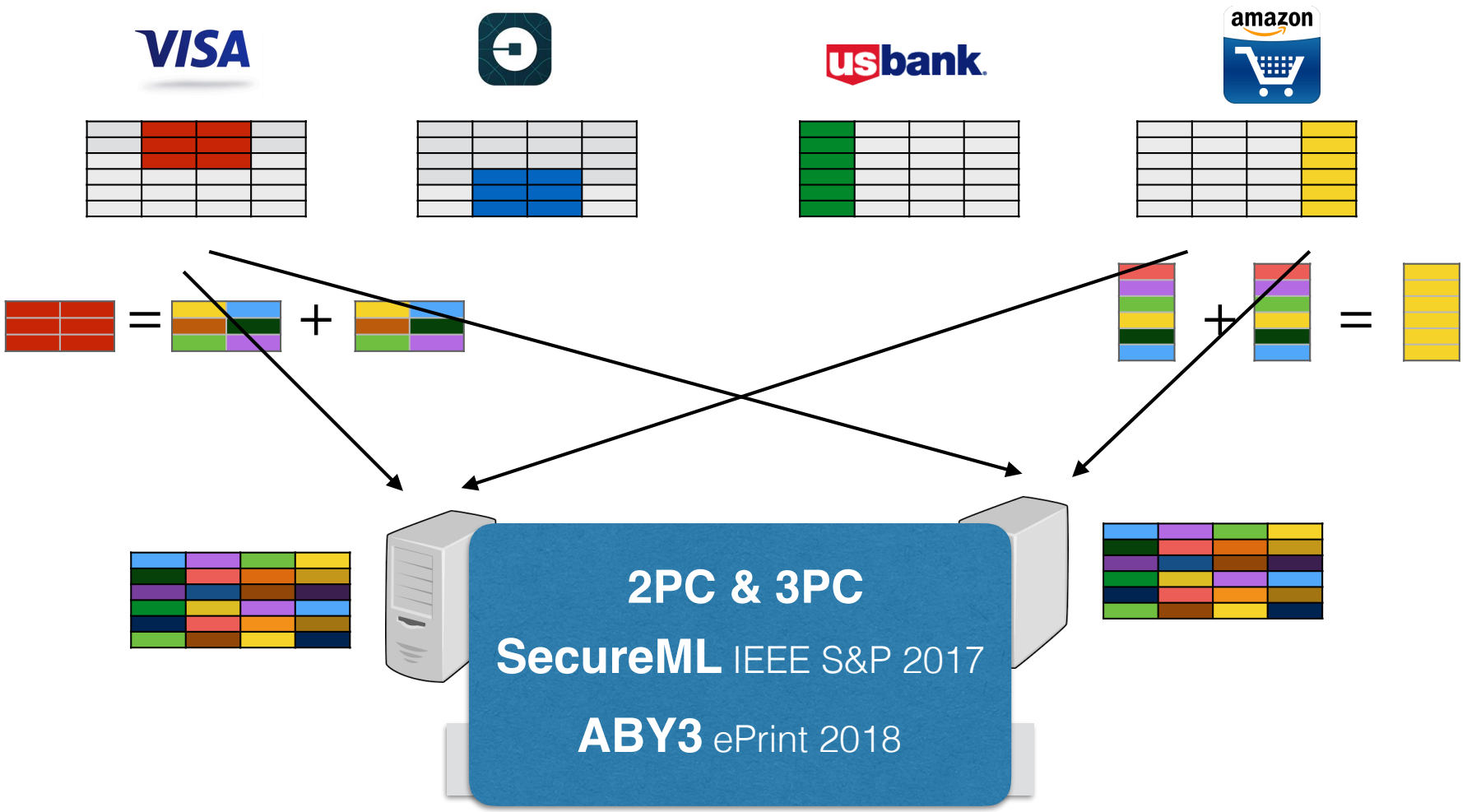
Secure Multiparty Computation



Hard to Scale

- Connect: network, broadcast
- Coordinate: online at same time
- Complexity: all parties run/manage software
- Efficiency: MPC for large n is expensive

Server-Aided Model



used in many prior work
[NWI+13, NIW+13,GSB+16, …]

Supervised Learning

Data

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1d} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \dots & x_{nd} \end{bmatrix} \quad \mathbf{x_1} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Model

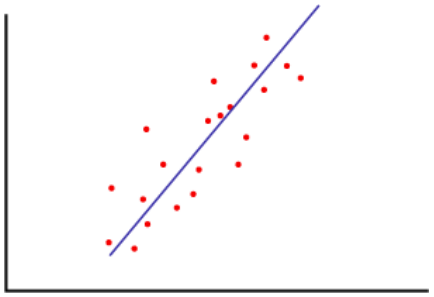
$$g(\mathbf{x_i}) \approx \mathbf{y_i}$$

Cost Function

$$C_i(g) = |g(\mathbf{x_i}) - \mathbf{y_i}|$$

$$C(g) = \sum_{i=1}^{n} C_i$$

Linear Regression



Model $$g(\mathbf{x}_i) = \sum_{j=1}^{d} x_{ij} w_j = \mathbf{x}_i \cdot \mathbf{w}$$

Cost function $$C_i(\mathbf{w}) = \frac{1}{2}(\mathbf{x}_i \cdot \mathbf{w} - y_i)^2$$

Closed form $$(\mathbf{X}^T \times \mathbf{X}) \times \mathbf{w} = \mathbf{X}^T \times \mathbf{Y}$$

Solve the linear system using 2PC [NWI+13,GSB+16]

expensive for large dim

fail for non-linear models

SGD Method

Update function

learning rate

$$w_j := w_j - \alpha \frac{\partial C_i(\mathbf{w})}{\partial w_j}$$

$$w_j := w_j - \alpha(\mathbf{x}_i \cdot \mathbf{w} - y_i)x_{ij}$$

MiniBatch update
2<|B|<500

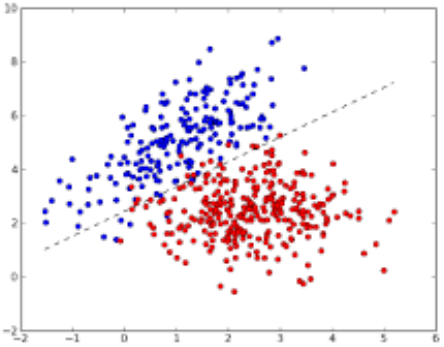
$$\mathbf{w} := \mathbf{w} - \frac{1}{|B|}\alpha \mathbf{X}_B^T \times (\mathbf{X}_B \times \mathbf{w} - \mathbf{Y}_B)$$

Epochs
2<e<15

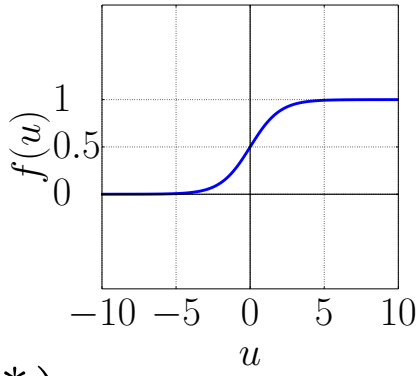
1. Shuffle dataset
2. Iterate through mini-batches
3. Terminate if change is small
4. Adjust rate and go to 1

1K to 75K iterations for 1M samples

Logistic Regression



Model $$g(\mathbf{x}_i) = f(\mathbf{x}_i \cdot \mathbf{w}), f(u) = \frac{1}{1+e^{-u}}$$

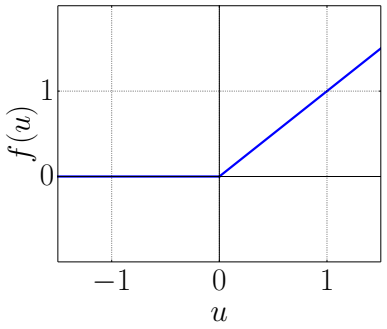


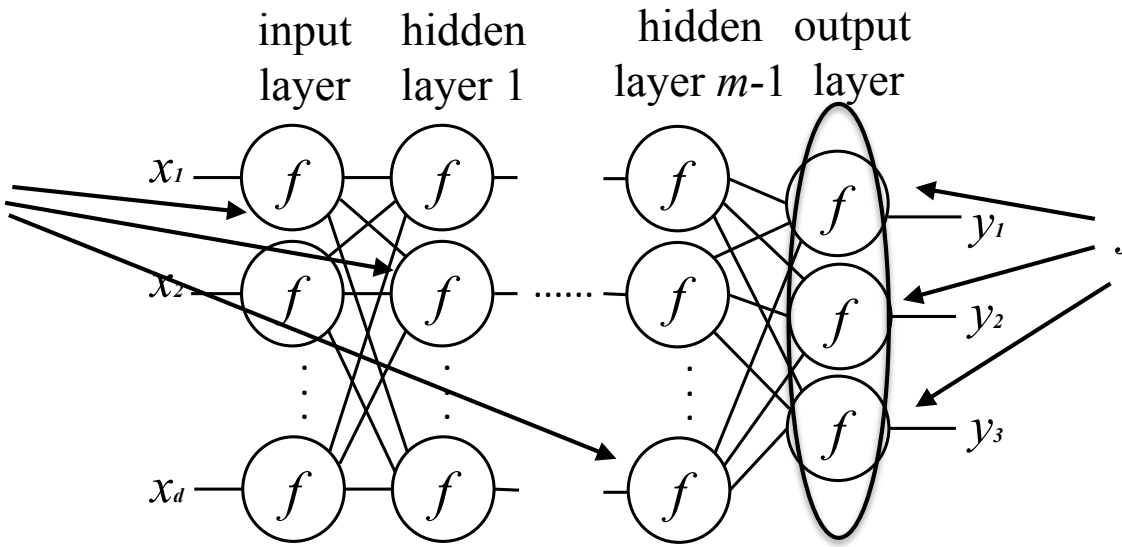
Cost Function $$C_i(\mathbf{w}) = -y_i \log y_i^* - (1-y_i)\log(1-y_i^*)$$
$$y_i^* = f(\mathbf{x}_i \cdot \mathbf{w})$$

Update function $$\mathbf{w} := \mathbf{w} - \frac{1}{|B|}\alpha \mathbf{X}_B^T \times (f(\mathbf{X}_B \times \mathbf{w}) - \mathbf{Y}_B)$$

Neural Networks

RELU





Softmax

$$f(u_i) = \frac{e^{-u_i}}{\sum_{i=1}^{d_m} e^{-u_i}}$$

Forward Propagation

$$\mathbf{U}_i = \mathbf{X}_{i-1} \times \mathbf{W}_i$$

$$\mathbf{X_0} = \mathbf{X_B} \qquad \mathbf{X_i} = \mathbf{f}(\mathbf{U_i}) \qquad \rightarrow \quad \mathbf{U}_m \;\; \mathbf{X}_m$$

Backward Propagation

$$\mathbf{Y}_i = \frac{\partial C(\mathbf{W})}{\partial \mathbf{U}_i}$$

$$\mathbf{Y}_i = (\mathbf{Y}_{i+1} \times \mathbf{W}_i^T) \odot \frac{\partial f(\mathbf{U}_i)}{\partial \mathbf{U}_i} \qquad \mathbf{Y}_m = \frac{\partial C}{\partial \mathbf{X}_m} \odot \frac{\partial f(\mathbf{U}_m)}{\partial \mathbf{U}_m}$$

Update Function

$$\mathbf{W}_i := \mathbf{W}_i - \frac{\alpha}{|B|} \cdot \mathbf{X}_i \times \mathbf{Y}_i$$

Nature of Computation

- Floating point arithmetic
- Vectorized computation
- Non-linear functions
  - ReLU, sigmoid, softmax, …
- Arithmetic ⟷ non-Arithmetic

Integer Multiplication

Secret Sharing over $\mathbb{Z}_{2^\ell}$

$$[\![x]\!]^{\mathsf{A}} \times [\![y]\!]^{\mathsf{A}}$$

2PC

$$[\![x]\!]^{\mathsf{A}} = (x_1, x_2)$$

$$x = x_1 + x_2 \bmod 2^\ell$$

Multiplication triplets
OT or LHE

3PC
[AFLNO16]

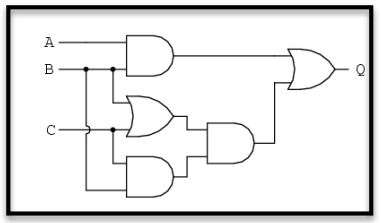
$$[\![x]\!]^{\mathsf{A}} = \{(x_1, x_2), (x_2, x_3), (x_3, x_1)\}$$

$$x = x_1 + x_2 + x_3 \bmod 2^\ell$$

Binary/Yao MPC

$$f([\![x]\!]^{\mathsf{B}})$$

over $\mathbb{Z}_2$



2PC

$$[\![x]\!]^{\mathsf{Y}} = \{(k^0_{\mathsf{w}}, k^1_{\mathsf{w}}), k^x_{\mathsf{w}}\}$$

3PC
[MRZ15]

$$[\![x]\!]^{\mathsf{Y}} = \{(k^0_{\mathsf{w}}, k^1_{\mathsf{w}}), (k^0_{\mathsf{w}}, k^1_{\mathsf{w}}), k^x_{\mathsf{w}}\}$$

Decimal Multiplication

1. Decimal -> Integer
2. Choose a large ℓ
3. Embed Integer in $\mathbb{Z}_{2^\ell}$
4. Use arithmetic MPC

ℓ too large
Fails for training ML

1. Decimal -> Integer
2. Fixed decimal points
3. Multiply/truncate integers
4. Use binary/Yao MPC

High communication
or
High round

Fixed Point 2PC

1. Decimal -> Integers
2. Secret share integers: $[\![x]\!]^{\mathsf{A}}, [\![y]\!]^{\mathsf{A}}$
3. Arithmetic MPC: $[\![z]\!]^{\mathsf{A}} = [\![x]\!]^{\mathsf{A}} \times [\![y]\!]^{\mathsf{A}}$
4. Truncate shares: $z_1/2^d, z_2/2^d$

a = 0001, a/2 = 0000
b = 0011, b/2 = 0001
c = 0100, c/2 = **0010**
a/2 + b/2 = **0001** = c/2 - **1**

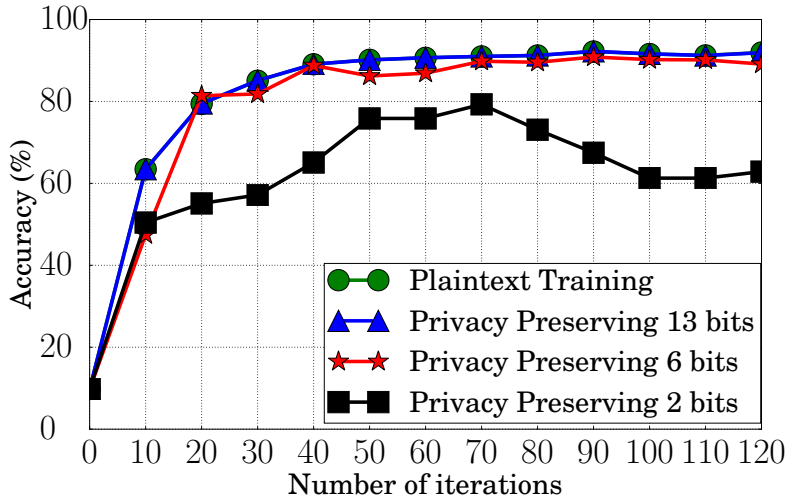
a =0111, a/2 = 0011
b = 0111, b/2 = 0011
c = 1110, c/2 = **1111**
a/2 + b/2 = **0110**

If $0 \leq z \leq 2^{\ell_z}, \ell \geq \ell_z + 1$ with prob. $1 - 2^{\ell_z+1}/2^{\ell}$

$$z_1/2^d + z_2/2^d \approx z/2^d$$

Accuracy and Efficiency

MNIST: n= 60k d=784



Ours

FP with GC

		Total (OT)	Total (LHE)	GC
LAN	$k = 1000$	0.028s	5.3s	0.13s
	$k = 10,000$	0.16s	53s	1.2s
	$k = 100,000$	1.4s	512s	11s
WAN	$k = 1000$	1.4s	6.2s	5.8s
	$k = 10,000$	12.5s	62s	68s
	$k = 100,000$	140s	641s	552s

4-8x improvement
fast online phase

Fixed Point 3PC

Approach 1

$$[\![z]\!]^{\mathsf{A}} = (z_1, z_2), (z_2, z_3), (z_3, z_1)$$

$$\downarrow$$

$$z_1/2^d, (z_2 + z_3)/2^d$$

$$\downarrow$$

$$(z'_1, z'_2, z'_3) = (z_1/2^d, (z_2 + z_3)/2^d - r, r)$$

$$\downarrow$$

$$(z'_1, z'_2), (z'_2, z'_3), (z'_3, z'_1)$$

Extra round
Semi-honest or

Approach 2

$$[\![r]\!]^{\mathsf{A}}, [\![r/2^d]\!]^{\mathsf{A}}$$

$$\downarrow$$

$$\text{Reveal } z' = z - r$$

$$\downarrow$$

$$z'/2^d + [\![r/2^d]\!]^{\mathsf{A}}$$

Binary 3PC
In parallel

Combine
With mult

Invoke
2pc theorem

No extra rounds
icious security
eralize to MPC

50x throughput
24x latency

2PC Vectorization

Matrix-Vector: (|B| x d) (d x 1), |B| = 128
OTs: Same receiver bits
LHE: Matrix multiplication triplets

	d	Online	Online Vec	OT	OT Vec	LHE	LHE Vec
LAN	100	0.37ms	0.22ms	0.21s	0.05s	67s	1.6s
	500	1.7ms	0.82ms	1.2s	0.28s	338s	5.5s
	1000	3.5ms	1.7ms	2.0s	0.46s	645s	10s
WAN	100	0.2s	0.09s	14s	3.7s	81s	2s
	500	0.44s	0.20ss	81s	19s	412s	6.2s
	1000	0.62s	0.27s	154s	34s	718s	11s

OT: 4-6x improvement
LHE: 40-60x improvement

3PC Vectorization

$$[\![x]\!]^{\mathsf{A}} \times [\![y]\!]^{\mathsf{A}}$$

$$\begin{aligned} z_1 &:= x_1y_1 + x_1y_2 + x_2y_1 \quad + \alpha_1 \\ z_2 &:= x_2y_2 + x_2y_3 + x_3y_2 \quad + \alpha_2 \\ z_3 &:= x_3y_3 + x_3y_1 + x_1y_3 \quad + \alpha_3 \end{aligned}$$

$$\downarrow$$

$$\{(z_1, z_2), (z_2, z_3), (z_3, z_1)\}$$

$$[\![\vec{x}]\!] \cdot [\![\vec{y}]\!]$$

$$\downarrow$$

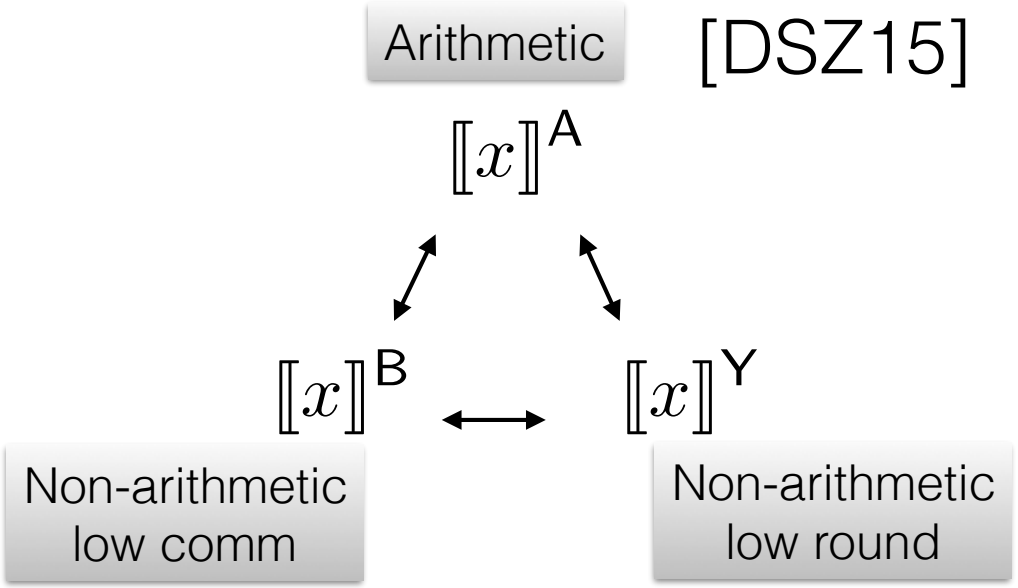
$$z_1 = \sum_{i=1}^{n} \vec{z}[i]_1, z_2 = \sum_{i=1}^{n} \vec{z}[i]_2, z_3 = \sum_{i=1}^{n} \vec{z}[i]_3$$

$$\downarrow$$

$$\text{reveal}([\![z]\!] + [\![r]\!])/2^d - [\![r/2^d]\!]$$

O(1) comm
1 round
1 truncation

ABY3



Conversion	Semi-honest		Malicious	
	Comm.	Rounds	Comm.	Rounds
$[\![x]\!]^{\mathsf{A}} \to [\![x]\!]^{\mathsf{B}}$	$k + k \log k$	$1 + \log k$	$k + k \log k$	$1 + \log k$
$([\![x]\!]^{\mathsf{A}}, i) \to [\![x[i]]\!]^{\mathsf{B}}$	k	$1 + \log k$	$2k$	$1 + \log k$
$[\![x]\!]^{\mathsf{B}} \to [\![x]\!]^{\mathsf{A}}$	$k + k \log k$	$1 + \log k$	$k + \log k$	$1 + \log k$
$[\![b]\!]^{\mathsf{B}} \to [\![b]\!]^{\mathsf{A}}$	$2k$	1	$2k$	2
$[\![b]\!]^{\mathsf{Y}} \to [\![b]\!]^{\mathsf{B}}$	$1/3$	1	$2\kappa/3$	1
$[\![b]\!]^{\mathsf{B}} \to [\![b]\!]^{\mathsf{Y}}$	$2\kappa/3$	1	$4\kappa/3$	1
$[\![x]\!]^{\mathsf{Y}} \to [\![x]\!]^{\mathsf{A}}$	$4k\kappa/3$	1	$5k\kappa/3$	1
$[\![x]\!]^{\mathsf{A}} \to [\![x]\!]^{\mathsf{Y}}$	$4k\kappa/3$	1	$8k\kappa/3$	1

Arithmetic to Binary

$$[\![x]\!]^{\mathsf{A}} = (x_1, x_2, x_3)$$

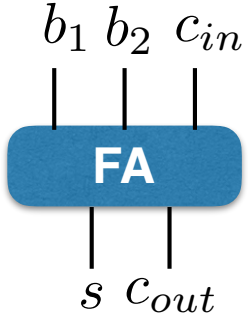
$$\downarrow$$

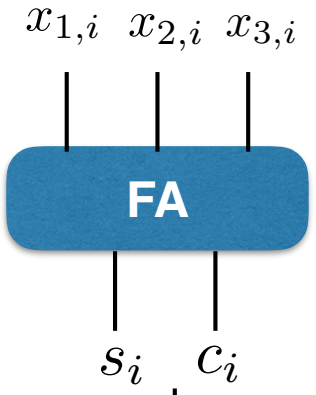
$$[\![x_1]\!]^{\mathsf{B}} := (x_1, 0, 0),$$
$$[\![x_2]\!]^{\mathsf{B}} := (0, x_2, 0),$$
$$[\![x_3]\!]^{\mathsf{B}} := (0, 0, x_3)$$

$$\downarrow$$

$$[\![x]\!]^{\mathsf{B}} = [\![x_1]\!]^{\mathsf{B}} + [\![x_2]\!]^{\mathsf{B}} + [\![x_3]\!]^{\mathsf{B}}$$

2 Ripple Carry Adders:
2k rounds
O(k) gates
2 Parallel Prefix Adders:
O(logk) rounds
O(klogk) rounds





$$\downarrow$$

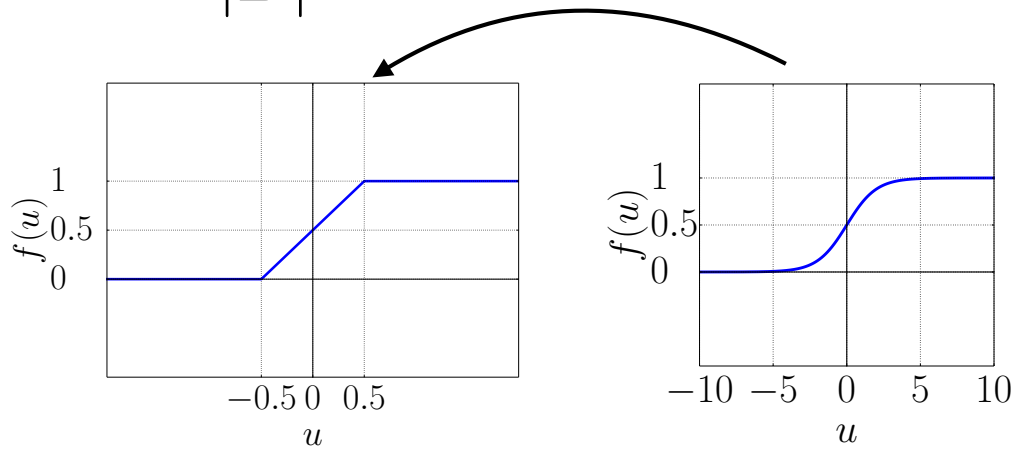
$$C = (c_1, \ldots, c_k)$$
$$S = (s_1, \ldots, s_k)$$

$$x_1 + x_2 + x_3 = 2C + S$$

1 Parallel Prefix Adder
k Parallel FAs
logk rounds
O(klogk) gates

Approximation 1

$$\mathbf{w} := \mathbf{w} - \frac{1}{|B|}\alpha \mathbf{X}_B^T \times (f(\mathbf{X}_B \times \mathbf{w}) - \mathbf{Y}_B)$$



Accuracy

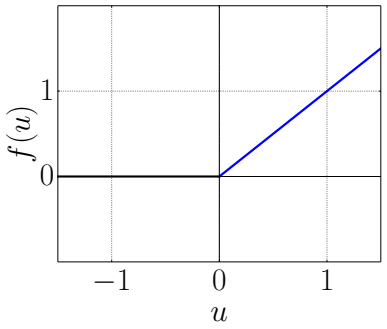
	Logistic	Our approaches		Polynomial Approx.		
		first	second	deg. 2	deg. 5	deg. 10
MNIST	98.64	98.62	97.96	42.17	84.64	98.54
Arcene	86	86	85	72	82	86

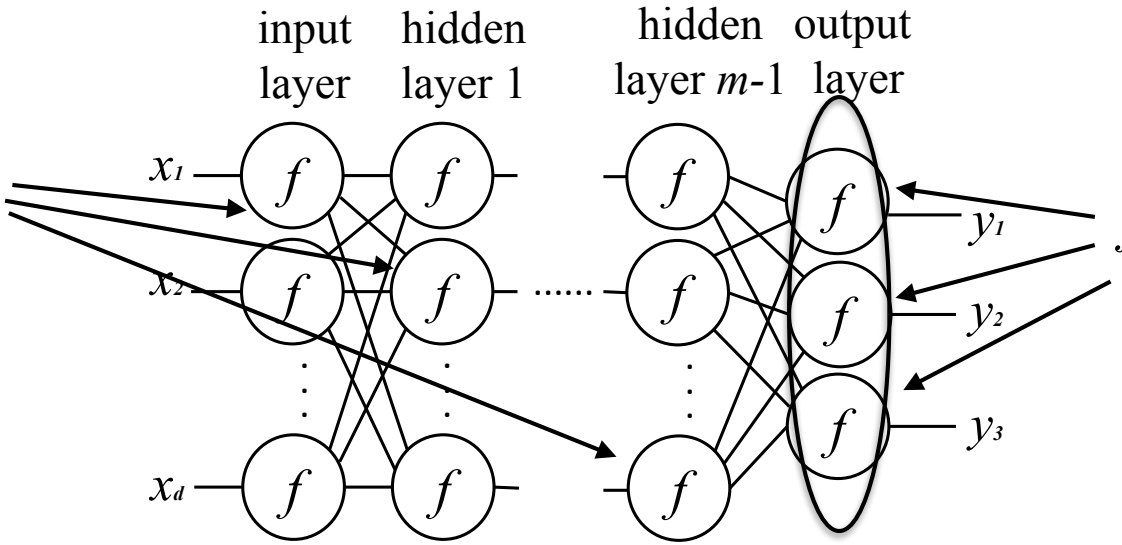
Efficiency

	New Logistic	Poly Total OT	Poly Total LHE
LAN	0.0045s	0.025s	6.8s
WAN	0.2s	2.5s	8.5s

Approximation 2

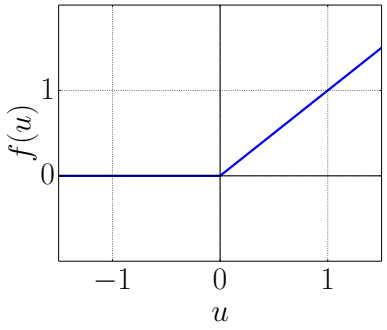
RELU





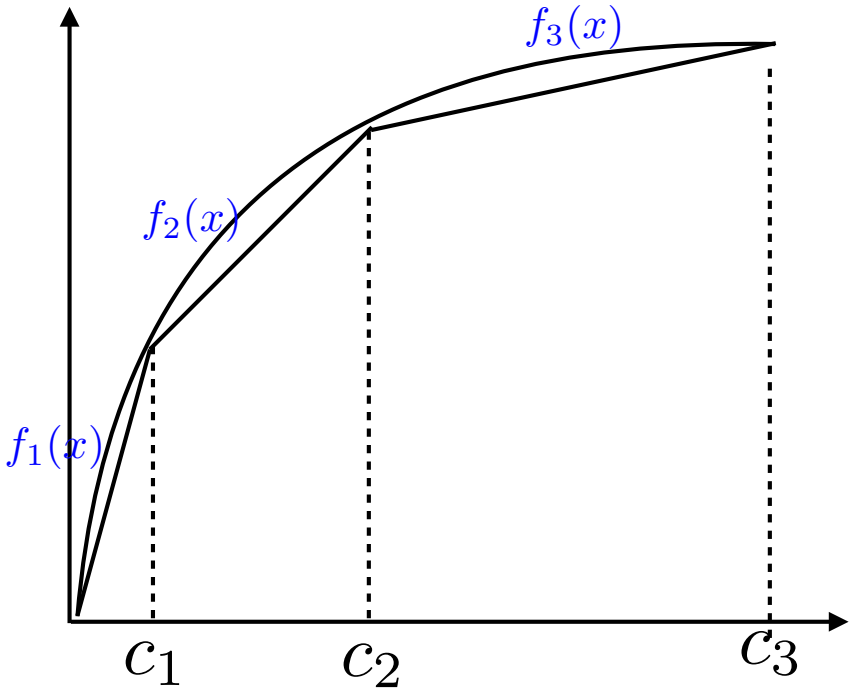
Softmax

$$f(u_i) = \frac{e^{-u_i}}{\sum_{i=1}^{d_m} e^{-u_i}}$$



- Use RELU for e^{-u_i}
- Addition/division circuit
- MNIST dataset
  - Tensorflow: 94.5%
  - Ours: 93.4%

Piecewise Polynomial Approximation



$$\sum b_i f_i(x) \text{ where } b_i = (c_{i-1} \leq x < c_i)$$

Addition circuit
+
customized oblivious transfer

Prediction Experiments

Model	Protocol	Batch Size	Running Time (ms) Online	Running Time (ms) Total	Comm. (MB)
Linear	This	1	0.1	3.8	0.002
		100	0.3	4.1	0.008
	SecureML	1	0.2	2.6	1.6
		100	0.3	54.2	160
Logistic	This	1	0.2	4.0	0.005
		100	6.0	9.1	0.26
	SecureML	1	0.7	3.8	1.6
		100	4.0	56.2	161
NN	This	1	3	8	0.5
	SecureML	1	193	4823	120.5
CNN	This*	1	6	10	5.2
	Chameleon	1	1360	2700	12.9
	MiniONN	1	3580	9329	657.5

NN: 3 layers, 266 nodes

CNN: 3 layers, 1090 nodes

Conclusion & Future

- Network is bottleneck
  - Parallelization is hard
- Formal Treatment of MPC for ML
- Combined with differential privacy
- Analyze approximation impact
- Toolbox for ML programmers