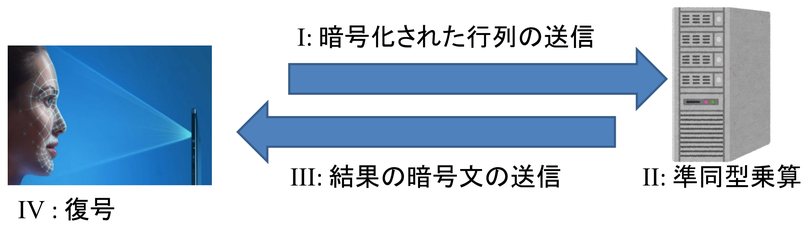
AIセキュリティ・プライバシーチーム
佐久間 淳
AI Security and Privacy Team　Jun Sakuma

■プライバシ保護機械学習のための行列乗算

準同型暗号上の行列乗算



- サーバにテストデータを見せずに機械学習で予測したい
  - データを準同型暗号化してサーバに予測評価(行列乗算)させる
- 現実的な応用では、クライアントは非力、サーバへの同時アクセスが多数発生
- 計算効率性と同時実行時性能のバランスがとれた行列乗算が必要

多項式環の性質を利用した効率的なパッキング

- k個の内積を一回の準同型乗算で実現
- 紛失送信ベースの手法よりも通信帯域を98%削減
- 計算能力の弱いクライアント(RasberryPie)において、AHEベースの手法よりも計算時間を23倍高速化(380sec -> 16sec)

行列乗算性能評価

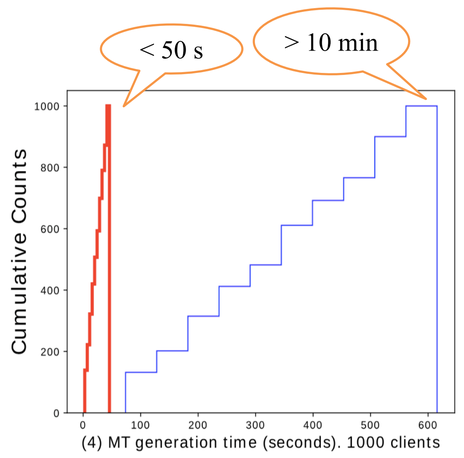
10x faster than AHE

100x faster than OT with 50 clients

Dimension n_1, n_2, n_3	Method	MT Generation Time (LAN/WAN) 1 client	10 clients	50 clients	Communication
128, 128, 128	AHE	26.3 sec/26.9 sec	26.1 sec/26.7 sec	33.2 sec/33.0 sec	3.98 MB
	OT	4.07 sec/60.8 sec	140 sec/168 sec	389 sec/485 sec	432 MB
	SwHE	123 sec/124 sec	123 sec/127 sec	146 sec/151 sec	32.3 MB
	Ours	2.49 sec/3.26 sec	2.54 sec/3.35 sec	3.53 sec/3.36 sec	8.25 MB
256, 128, 256	AHE	95.1 sec/95.3 sec	94.6 sec/95.0 sec	122 sec/123 sec	11.9 MB
	OT	16.1 sec/235 sec	588 sec/656 sec	0.426 hr/0.472 hr	1664 MB
	SwHE	486 sec/489 sec	495 sec/498 sec	545 sec/545 sec	129 MB
	Ours	7.38 sec/8.21 sec	7.41 sec/8.63 sec	10.8 sec/11.0 sec	32.5 MB
512, 128, 512	AHE	361 sec/361 sec	356 sec/359 sec	507 sec/510 sec	39.8 MB
	OT	57.5 sec/917 sec	0.723 hr/0.732 hr	1.59 hr/1.82 hr	6520 MB
	SwHE	0.522 hr/0.538 hr	0.556 hr/0.574 hr	0.560 hr/0.570 hr	513 MB
	Ours	26.6 sec/27.5 sec	26.8 sec/29.1 sec	41.2 sec/41.4 sec	129 MB

- OT-based and AHE-based methods from SecureML.
- Somewhat HE-based method (i.e., FV scheme) from MiniONN.

同時実行時性能評価



- Matrix Size: 128*128, LAN Network
- 5 machines sent 200 queries each

More Practical Privacy-Preserving Machine Learning as A Service via Efficient Secure Matrix Multiplication. WAHC@CCS 2018, pp. 25-36.

■離散分布におけるAdditive汎関数の最適推定

Additive汎関数を推定する最も良い方法は？

Additive汎関数) 関数ϕ: $[0,1] \to \mathbb{R}$, 離散分布$P = (p_1, \dots, p_k)$

$$\theta(P;\phi) = \sum_{i=1}^{k} \phi(p_i)$$

例: シャノンエントロピー$\phi(p) = -p \ln p$

問題: n個のPからの$\theta(P;\phi)$を推定

00年より前	現在	課題
kが固定 最尤推定量が最適	kが大きい 特定のϕの最適推定量	kが大きい 一般のϕでの最適推定量は？

貢献)
- 一般のϕにおいて最適推定誤差を特徴付け
- 一般のϕにおいて最適推定量を構築する方法論

結果) 発散速度による特徴づけ

ϕのℓ次発散速度が$p^\alpha \Leftrightarrow \exists W_\ell > 0, c_\ell, c'_\ell$,

$$W_\ell p^{\alpha-\ell} - c'_\ell \le \left|\phi^{(\ell)}(p)\right| \le W_\ell p^{\alpha-\ell} + c_\ell$$

ϕのℓ次微分の0点付近の挙動が$p^{\alpha-\ell}$と同じ

α	ℓ	Minimax risk
≤ 0	1	一致推定量なし
$(0, 1/2]$	4	$\frac{k^2}{(n \ln n)^{2\alpha}}$
$(1/2, 1)$	4	$\frac{k^2}{(n \ln n)^{2\alpha}} + \frac{k^{2-2\alpha}}{n}$
1	4	$\frac{k^2}{(n \ln n)^2} + \frac{\ln^2 k}{n}$
$(1, 3/2)$	6	$\frac{k^2}{(n \ln n)^{2\alpha}} + \frac{1}{n}$
$[3/2, 1]$	2	$1/n$

Kazuto Fukuchi and Jun Sakuma, “Minimax Optimal Additive Functional Estimation with Discrete Distribution: Slow Divergence Speed Case”, ISIT, pp. 1041-1045, 2018.
Kazuto Fukuchi and Jun Sakuma, “Minimax Optimal Additive Functional Estimation with Discrete Distribution”, arxiv:1812.00001, 2018.

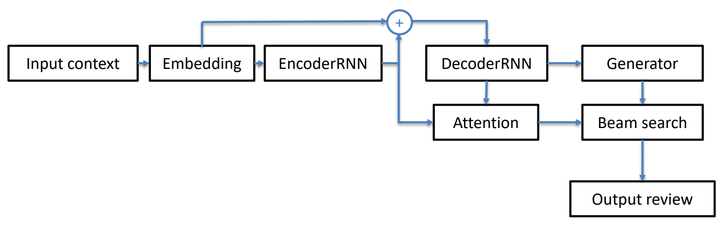
■Stay On-Topic: Generating Context-specific Fake Restaurant Reviews

★やったこと

日本での食べログのような、レストランの評判情報をターゲットとしたフェイクレビューを自動生成する攻撃手法の提案と、そのようなフェイクレビューの検出方法を提案

★アイディア

レストラン評判情報のコンテクスト（星の数、レストラン名、カテゴリ、場所等）とそのレストランのレビューを多数収集して対訳コーパスとする。ニューラル機械翻訳技術により、コンテクストをレビューに機械翻訳する⇒フェイクレビューの生成。



★結果

Type	Detection rate	F-score
Human-written	63%	59%
Machine-generated	**50%**	53%

通常の被験者、懐疑的な被験者ともに、機械翻訳で生成したフェイクレビューと人間が生成したレビューの見分けがつかなかった。
特徴量を工夫することで、フェイクと人間のレビューの区別をつけることは可能（ただし万能な対策ではない）

“Stay On-Topic: Generating Context-specific Fake Restaurant Reviews,” Proceedings of ESORICS 2018, pp. 132–151, September 2018

■高レベルプログラムの値依存時相仕様検証

高レベル言語で記述されたプログラムが、プログラム実行時の値に依存した時相仕様（プログラム状態の時間発展に関する仕様）を満たすことを機械的に証明する方法を提案（例えば、プログラムのならし計算量の解析が可能に）

高レベルプログラムの例：（OCaml言語による）
リストの対を用いたキューの実装

```
(* リストの反転関数 *)
let rev l = let rec aux l acc = match l
with
    | [] -> acc | h::t -> event[Tick];
aux t (h::acc)
  in aux l []
(* エンキュー関数 *)
let enqueue e (l1,l2) = event[Enq];
(l1,e::l2)
(* デキュー関数 *)
let rec dequeue (l1,l2) = match l1 with
  | e::l1' -> event[Deq]; (e, (l1', l2))
  | [] -> dequeue (rev l2, [])
(* キューが空になるまで非決定的に
     エンキューかデキューを繰り返すメイン関数 *)
let main () = let rec loop (l1,l2) =
    if * then loop (enqueue 42 (l1,l2))
    else if is_empty (l1,l2) then ()
    else loop (snd (dequeue (l1,l2)))
  in loop ([], [])
```

プログラム中のevent[a]を実行するとイベントaが発生

左のプログラムの main () を実行すると例えば以下のようなイベント列が非決定的に生成される：
Enq, Enq, Tick, Tick, Deq, Deq
Enq, Tick, Deq, Enq, Tick, Deq
Enq, Enq, Tick, Tick, Deq, Deq, Enq, Tick, Deq

値依存時相仕様の例：

$$\lambda x \in \Sigma^*.\, x \neq \epsilon \Rightarrow \frac{\#_{\mathrm{Tick}}(x)}{\#_{\mathrm{Deq}}(x)} = 1$$

ここで $\#_a(x)$ はイベント列 x にイベント aが出現する回数を表す
つまり、上記仕様はプログラムの実行によって生成されるイベント列 x が空でなければ、デキューによって引き起こされるリストの反転にかかる計算回数とデキューの回数の比が定数で抑えられることを表す

$$P \models^{?} \Phi$$

プログラムP が仕様 Φ を満たすことを検査するために以下の2つの形式体系を提案
(1) 依存リファインメント型・エフェクトシステム
- プログラムP が仕様 Φ を満たすことの判定問題を不動点論理$\mathcal{L}$の妥当性判定問題に帰着

(2) 不動点論理 $\mathcal{L}$ の妥当性判定のための演繹体系
- 論理式中の不動点を右表の技術を用いて過大・過小近似して消去
- 不動点を含まない論理式の妥当性を既存の定理証明器により判定

	過大近似	過小近似
最小不動点	不変条件 (帰納法)	不変条件 + 整礎関係
最大不動点	不変条件 + 整礎関係	不変条件 (余帰納法)

現在、本成果をベースに確率的プログラムのプライバシー検証法を研究中

A Fixpoint Logic and Dependent Effects for Temporal Property Verification. LICS 2018, pp. 759-768.