Supervised Text Classification with Leveled Homomorphic Encryption

Do we need to see data to use it?

Zachary Zanussi*, Benjamin Santos, Saeid Molladavoudi

Official Statistics and Methodology Symposium
September 29th, 2021

Delivering insight through data for a better Canada

Outline

- Introduction:
  - Encrypted machine learning
  - Homomorphic encryption

- Text classification task:
  - Scanner data product descriptions → NAPCS codes

- Methods:
  - Standard model
  - Issues
  - Ensemble model

- Conclusion

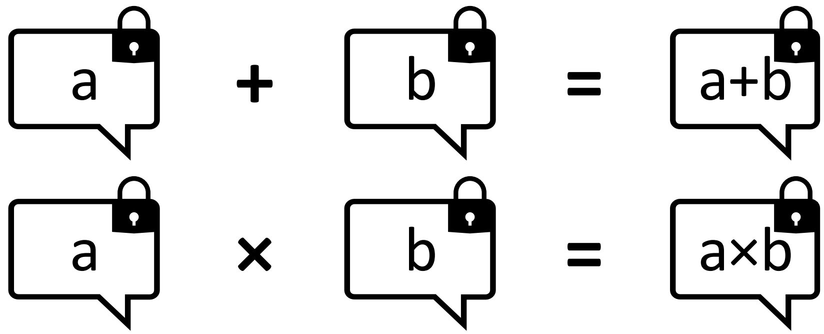
Introduction

- Machine learning has seen myriad uses across applications such as text classification
- Considerations of data privacy and security pose additional challenges to data science methods
- Emerging Privacy Preserving Techniques (PPTs) allow us to extract analytics while protecting respondent privacy
- In this work, we explore the application of Homomorphic Encryption (HE) to train a neural network on sensitive data

What is Homomorphic Encryption (HE)?

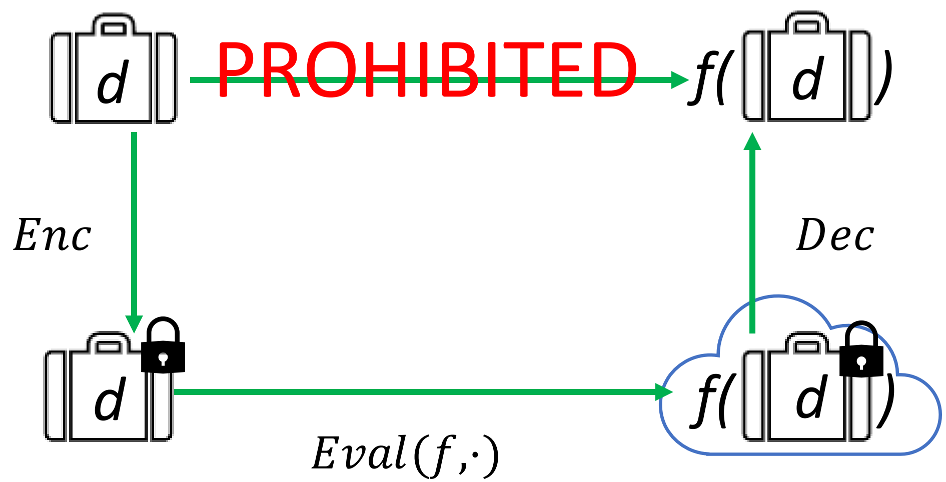
- Allows to perform arithmetic operations on encrypted data

- Make the encryption map a *ring homomorphism*: it should preserve addition and multiplication



- Application: delegated computing!

- Unparalleled cryptographic security at the cost of higher computational and storage requirements
  - *Levelled* nature of CKKS limits the number of consecutive operations



CKKS Homomorphic Encryption Scheme

- Data is encoded as a vector $v \in \mathbb{R}^N$, and then encrypted to get $[v]$
- Given two ciphertexts $[v]$ and $[u]$, we have
  - $[v] \oplus [u] = [v+u]$ and $[v] \otimes [u] = [vu]$
- Addition and multiplication are performed *componentwise*;

$$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix} \otimes \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{bmatrix} = \begin{bmatrix} v_1u_1 \\ v_2u_2 \\ \vdots \\ v_Nu_N \end{bmatrix}$$

- *Rotation* is a relatively costly operation that lets us change slots;

$$rot\left(\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix}\right) = \begin{bmatrix} v_N \\ v_1 \\ \vdots \\ v_{N-1} \end{bmatrix}$$

- The structure of our ciphertexts affects performance and is referred to as *packing*
- *Leveled* scheme means we want to minimize our multiplications
  - We have access to 13 – 31 multiplications and unlimited additions/rotations

Scanner data

- Consists of retailer's product description, some identifiers, and price of transaction
- One use is to compute the Consumer Price Index
- Use ML to convert product description into North American Product Classification System (NAPCS) codes
- Separate into lists based on first three columns; compute statistics on each
- Pipeline at StatCan:

Description	ID1	ID2	Value
"mochi ice cream bon bons"	054	78	$5.31
"chipotle barbeque sauce"	201	34	$3.80

ML TC

NAPCS	ID1	ID2	Value
5611121	054	78	$5.31
5611132	201	34	$3.80

Statistics

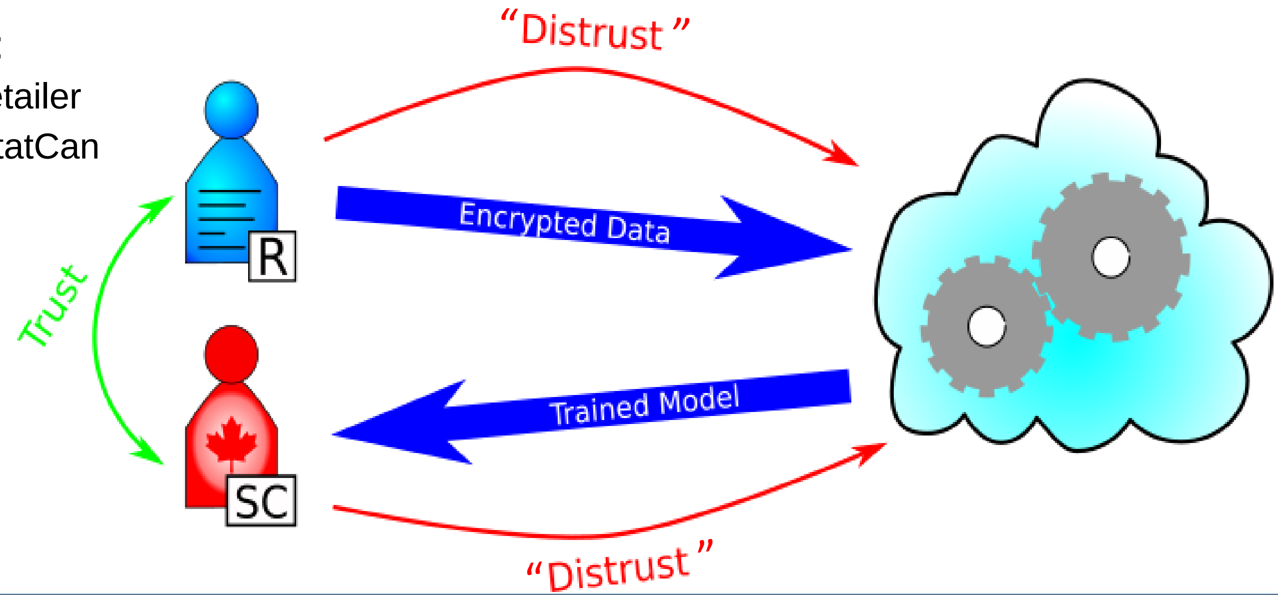
Task: Private Text Classification

- The standard scanner data workflow needs to be moved to the cloud, but the data is sensitive
- The workflow includes two phases – training and prediction
- As a proof of concept, we test the feasibility of securing the training routine with HE
- Synthetic data is sampled from USDA FoodData
  - 50,000 entries from 5 different NAPCS codes

“mochi ice cream bon bons” → 5611121

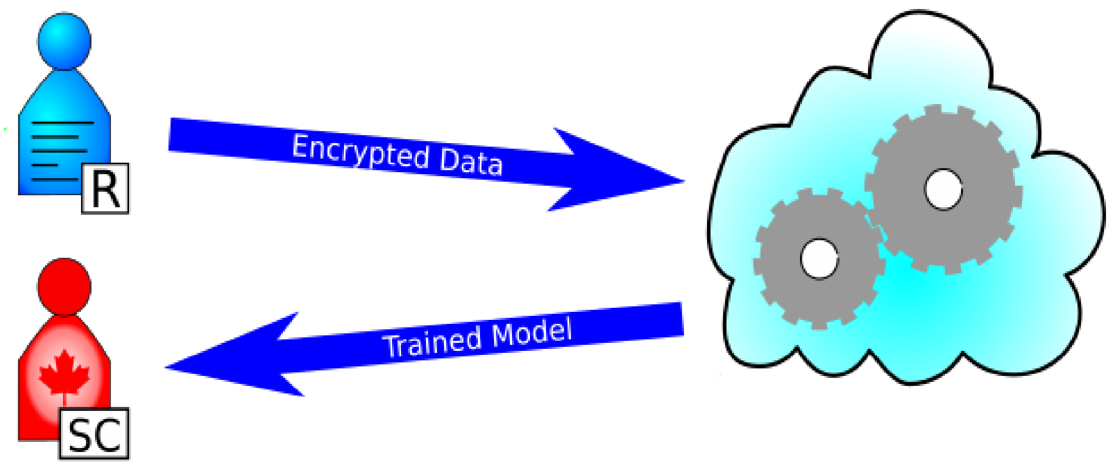
Official Statistics Trust Model

- Slightly relaxed semi-honest trust model takes advantage of the relationship between NSOs and data holders:
  - These parties can share their data, but have limited compute power
- Ciphertext “refreshing” is allowed
- Would like to minimize retraining, and guarantee a performant model after the first training session
- Three party protocol:
  - Data holder, the retailer
  - Data consumer, StatCan
  - The Cloud

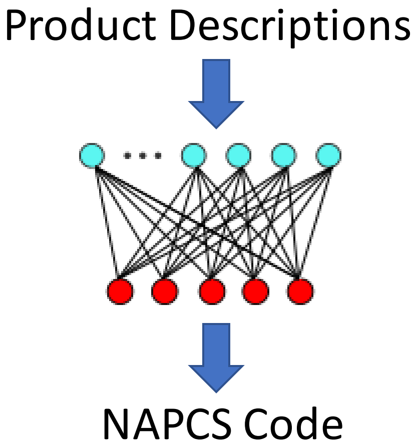


Protocol

- StatCan creates and distributes encryption keys and designs encoding and training circuit
- Retailer encodes and encrypts their data and sends it to the cloud
- Cloud evaluates training circuit homomorphically
- StatCan ends up with trained model, which can be run on premise or returned to the cloud for encrypted prediction



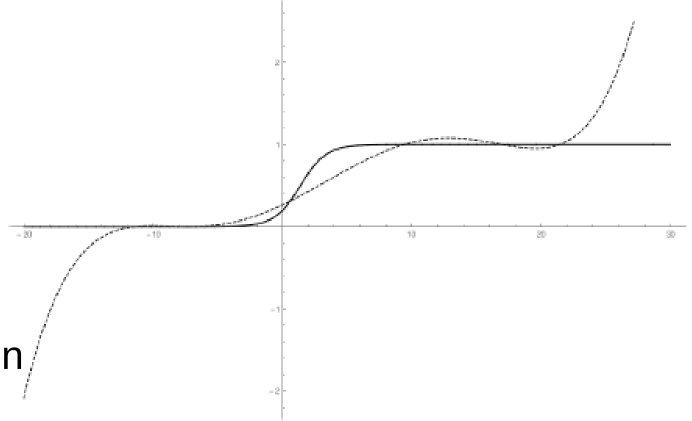
First Model



- Single layer NN with (shifted) sigmoid activation layer
  - Softmax is hard to approximate with a polynomial!
  - Sigmoid is approximated by degree 5 polynomial
- Use n-gram bag of words encoding:
  - $n \in \{3, 4, 5, 6\}$, only keep n-gram if it appears > 3 times, leaving 14,212 n-grams
- Use hashing vectorizer to reduce input dimension to 8,192

- Model (both ciphertext and cleartext) implemented by hand in C++
- Use big batches (usually the entire dataset)
- Train for 2 model updates (epochs)
- Implemented Nesterov momentum and l2 regularization

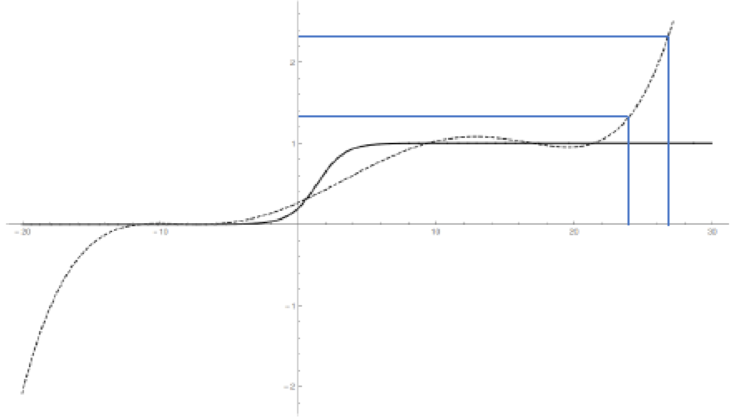
Figure 1: Comparison of sigmoid σ (solid) to the sigmoid approximation s (dashed).



Problems Identified in the First Model

- First model attained an **accuracy of 67%** after **47 hours** of training
- **Slow**, not particularly accurate
- Sensitive to **explosion** due to sigmoid approximation

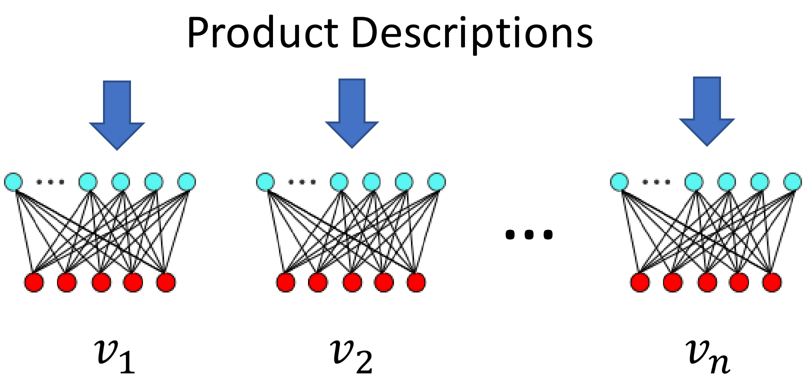
Figure 1: Comparison of sigmoid σ (solid) to the sigmoid approximation s (dashed).



- **Delicate** in terms of hyperparameters
  - In production, training should be performant on the first try, without peeking
- The model is **inflexible**
  - Inputs must be exactly 8192 dimensional!
- Idea: Train an ensemble!

Ensemble Model

- Idea: Train an *ensemble* of weak learners.
- Each learner takes less time to train, and at prediction time each learner votes
- Strip out bells and whistles (activation functions, regularization, …)
- Keep Nesterov's accelerated gradient to maximize training on limited epochs
- Results in a faster, more accurate model, which should *generalize* better to different datasets
- A smart ciphertext packing strategy allows us to train several models in parallel



Ensemble prediction:

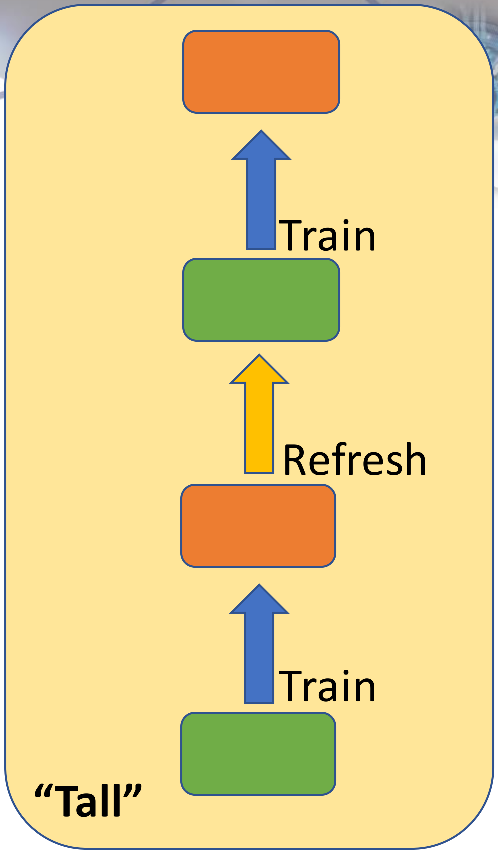
$$v = \sum_i v_i$$

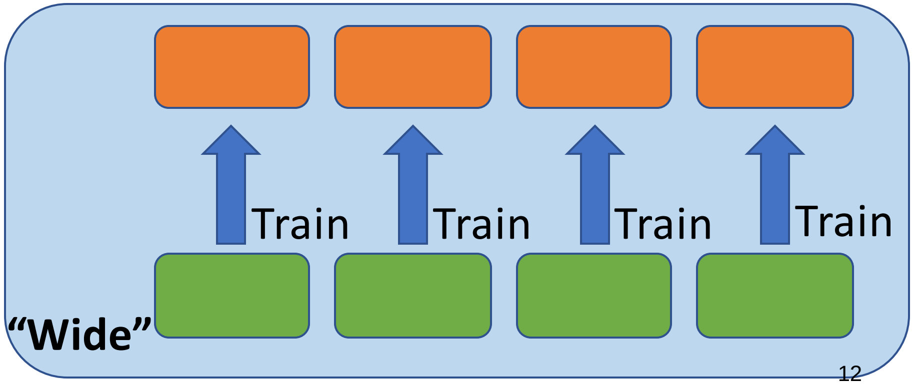
Two modes of operation

- We need to somehow mitigate the problem of limited training levels

- **<u>"Tall" model:</u>**
  - Use one set of model ciphertexts
  - When levels are expended, send back to StatCan to decrypt and re-encrypt
  - Allows for unlimited levels, but adds a back-and-forth IT/communication cost

- **<u>"Wide" model:</u>**
  - Use several sets of model ciphertexts
  - Train each only until levels are expended
  - No communication cost





Results

- Dataset Expansion: 14.9 MB → 78.5 GB
- Best cleartext results: 87% (80 epochs, 3,200 model updates)

Network	Submodels	Model Updates	Model Refreshes	Training Time	Test Accuracy
Cleartext	1	80	NA	15 s	74.3%
"Tall" Ciphertext	4	18	2	5.03 hr	74.2%
"Wide" Ciphertext	16	6×4	0	6.97 hr	74.4%

- Experiments performed on 8 core, 32 GB Azure VM
- Encrypted model implemented with Microsoft SEAL

Conclusion

- Homomorphic Encryption is an appropriate method for surmounting privacy issues in machine learning

- Accuracy loss is in this application is non-negligible but not prohibitively high

- Ensemble models help maximize learning and improves chances of a one-shot successful training run

- HE has progressed to a stage where a small team can perform a non-trivial real-world encrypted ML task in a reasonable amount of time and cloud cost

Thank you! Merci!

saeid.molladavoudi@canada.ca

benjamin.santos@canada.ca

zachary.zanussi@canada.ca

Zanussi, Z. and Santos, B. and Molladavoudi, S. (2021), Supervised text classification with leveled homomorphic encryption. To appear in: Proceedings of the 63rd ISI World Statistics Congress, Virtual.