CryptoDL: Towards Deep Learning over Encrypted Data

Ehsan Hesamifard
Department of Computer Science and Engineering
University of North Texas
Denton, TX, USA
ehsanhesamifard@my.unt.edu

Hassan Takabi
Department of Computer Science and Engineering
University of North Texas
Denton, TX, USA	takabi@unt.edu

Mehdi Ghasemi
Department of Mathematics and Statistics
University of Saskatchewan
Saskatoon, Saskatchewan, Canada
mehdi.ghasemi@usask.ca

ABSTRACT

With increasing growth of cloud services, machine learning services can be run on cloud providers’ infrastructure where training and deploying machine learning models are performed on cloud servers. However, machine learning solutions require access to the raw data, which can create potential security and privacy risks. In this work, we take the first steps towards developing theoretical foundation for implementing deep learning algorithms in encrypted domain and propose a method to adopt deep neural networks (NN) within the practical limitations of current homomorphic encryption schemes. We first design two methods for approximation of activation functions with low degree polynomials. Then, we train NNs with the generated polynomials and analyze the performance of the trained models. Finally, we run the low degree polynomials over encrypted values to estimate the computation costs and time.

1. INTRODUCTION

Machine learning algorithms based on deep neural networks have achieved remarkable results and are extensively used for analyzing big data in a variety of domains. However, training the models require access to the raw data which is often privacy sensitive. Recent advances in fully homomorphic encryption (FHE) allow us to apply deep learning models directly to encrypted data and return encrypted results without compromising security and privacy concerns. However, despite the advantages of using HE schemes, only additions and multiplications can be efficiently implemented over encrypted data and complex functions like sigmoid functions that are used in neural networks are not practical in current HE schemes.

Furthermore, in order to have efficient and practical solutions for computations in encrypted domain, we typically need to use somewhat homomorphic schemes instead of fully homomorphic encryption. However, a solution that builds upon these encryption schemes has to be restricted to computing low degree polynomials in order to be practical.

Approximating a function with low degree polynomials is an important issue for computations over encrypted data when we use homomorphic encryption. We propose two methods in section 2 for approximating activation functions used in neural networks with low degree polynomials.

The goal of this work is to provide solutions to run deep learning algorithms on encrypted data and allow the parties to provide/receive the service without having to reveal their sensitive data to the other parties. The application of machine learning in encrypted domain has attracted significant interest in recent years from both academia and industry [1, 2, 3, 4]. However, majority of previous work uses secure multiparty computation techniques which require communication between parties and are not efficient. We focus on homomorphic encryption which requires much less communication and reduces the communication cost overhead is removed significantly. Closely related to our work, Xie et al. discuss theoretical aspects of using polynomial approximation for implementing neural network in encrypted domain [4]. Building on this work, Dowlin et al. implement a neural network classifier on encrypted data by replacing some steps like max pooling with scaled mean-pooling or the activation function with square function $f(z):=z^{2}$ [2]. They build the model using plain data and then use this model for classifying encrypted data. Our proposed work on the other hand, performs both training and classification in encrypted domain.

In this work, we take first steps towards developing solutions for implementing deep neural networks within limitations of current HE schemes. We propose two methods for approximating a continuous function with low degree polynomials and use them in NN algorithms and analyze the performance of the new algorithms.

2. PROPOSED APPROACH

Among continuous functions, perhaps polynomials are the most well-behaved and easiest to compute. Let us denote the family of all continuous real valued functions on a non-empty compact space $X$ by $C(X)$. Suppose that among elements of $C(X)$, a subfamily $A$ of functions are of particular interest. We say that an element $f \in C(X)$ can be approximated by elements of $A$, if for every $\epsilon > 0$, there exists $p \in A$ such that $|f(x) - p(x)| < \epsilon$ for every $x \in X$. The following classical results guarantee every $f \in C(X)$ can be approximated by elements of $A$.

Theorem 1 (Stone–Weierstrass). Every element of $C(X)$ can be approximated by elements of $A$ if and only if for every $x \neq y \in X$, there exists $p \in A$ such that $p(x) \neq p(y)$.
Despite the strong and important implications of the Stone-Weierstrass theorem, it leaves computational details out and does not give a specific algorithm to generate an estimator for $f$ with elements of $A$, given an error bound $\epsilon$. Now, let $\mu$ be a finite measure on $X$ and for $f, g \in C(X)$ define $(f, g) = \int_X f g \mu$. Different choices of $\mu$, give different systems of orthogonal polynomials. In this work, we experiment with polynomial approximations of the Sigmoid function $\frac{1}{1+e^{-x}}$ over a symmetric interval $[-l, l]$ using two different measure: $d\mu = \frac{dx}{\sqrt{1-(x/l)^2}}$ and $d\mu = e^{-(l/x)} \frac{dx}{x}$ (Table 1).

2.1 Experimental Results

We train models using a polynomial approximation as an activation function in the NN for the implementation and set encrypted values. All computations were run on a virtual machine with 4GB RAM, Core i5 and Ubuntu 14.04. We use HELib for the implementation and set $k = 80$, $s = 1$, and $w = 64$ and $p = 10^{18} + 37$. The other parameter is $L$ which generally shows the allowed depth of algorithm. As shown in Table 3, small values for $L$ decrease the computation time, size of ciphertext and the degree of polynomial. For higher degree polynomials, we should increase the value of $L$ which leads to increasing computation and size of ciphertexts and also the time for generating encryption scheme.

<table>
<thead>
<tr>
<th>Degree</th>
<th>Running Times (seconds)</th>
<th>$L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.122829</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>0.305238</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>1.81771</td>
<td>21</td>
</tr>
<tr>
<td>5</td>
<td>2.19533</td>
<td>21</td>
</tr>
<tr>
<td>6</td>
<td>2.26061</td>
<td>25</td>
</tr>
<tr>
<td>7</td>
<td>2.71306</td>
<td>25</td>
</tr>
<tr>
<td>8</td>
<td>3.27217</td>
<td>25</td>
</tr>
<tr>
<td>9</td>
<td>4.59763</td>
<td>27</td>
</tr>
</tbody>
</table>

3. CONCLUSION

In this work, we developed theoretical foundation for implementing deep learning algorithms over encrypted data. We showed that polynomial approximations are suitable alternatives for activation functions to adopt NNs within HE schemes limitations. However, polynomials should be chosen carefully based on the dataset to achieve the best results.

4. REFERENCES


