# Privacy Preserving Multi-party Machine Learning with Homomorphic Encryption

**Hassan Takabi**
University of North Texas
Denton, TX, USA
takabi@unt.edu

**Ehsan Hesamifard**
University of North Texas
Denton, TX, USA
ehsanhesamifard@my.unt.edu

**Mehdi Ghasemi**
University of Saskatchewan
Saskatoon, Canada
mehdi.ghasemi@usask.ca

## Abstract

Privacy preserving multi-party machine learning approaches enable multiple parties to train a machine learning model from aggregate data while ensuring the privacy of their individual datasets is preserved. In this paper, we propose a privacy preserving multi-party machine learning approach based on homomorphic encryption where the machine learning algorithm of choice is deep neural networks. We develop theoretical foundation for implementing deep neural networks over encrypted data and utilize it in developing efficient and practical algorithms in encrypted domain.

## 1 Introduction

In many settings, multiple parties would benefit from training precise machine learning models on the data aggregated from datasets collected by different parties. For example, multiple medical institutions might share patient data to collaboratively train a machine learning model that helps in diagnosing a disease. The aggregate data would allow machine learning algorithms to produce better models as more data is available to the algorithms and it provides a set of features that would not otherwise be available to any of the parties. However, the parties may be hesitant to share their potentially sensitive data due to privacy concerns. On the other hand, machine learning algorithms based on deep neural networks are achieving remarkable results and are extensively used for data analytics in a variety of domains such as spam detection, traffic analysis, intrusion detection, medical or genomics predictions, face recognition, and financial predictions. However, training the models require access to the raw data which is often privacy sensitive.

Motivated by these two issues, in this paper, we focus on the problem of privacy preserving multi-party machine learning where the machine learning algorithm of choice is neural networks. We aim to provide solutions to run deep learning algorithms based on neural networks on encrypted data and allow the parties to jointly train a model without having to reveal their sensitive data to the other parties. Recent advances in fully homomorphic encryption (FHE) enable a limited set of operations to be performed on encrypted data. This will allow us to apply machine learning models directly to encrypted data and return encrypted results without compromising security and privacy concerns. However, due to a number of constraints associated with these cryptographic schemes, designing practical efficient solutions to run deep learning models on encrypted data is a challenging task.

Although implementing machine learning algorithms in encrypted domain has been studied in recent years (1; 2; 3; 5; 6; 11), there is not much work on multi-party machine learning in encrypted domain. In order to have efficient and practical solutions for computations in encrypted domain, we typically need to use somewhat homomorphic schemes or leveled 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. Having a real valued elementary function $f$, we are interested in finding polynomials with the lowest possible degree that estimate $f$ within a certain error range.

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

We first provide theoretical foundation to show that it is possible to find lowest degree polynomial approximation of a function within a certain error range and provide an approach to generate those approximations. We use Chebyshev polynomials for approximating the functions and make some changes in Chebyshev basis to achieve a better performance in neural networks. Next, we propose two methods for approximating a continuous function such as sigmoid with low degree polynomials and utilize these polynomials in neural network and analyze the performance of the new algorithms. Finally, we implement the neural networks with polynomial approximation as activation function over encrypted data. Our preliminary results are promising and show that neural networks could be trained over encrypted data where the datasets are distributed across multiple data owners.

## 2 Theoretical Foundation: Polynomial Approximation

Let us denote the family of all continuous real valued functions on a non-empty compact space $X$ by $\mathrm{C}(X)$. Suppose that among elements of $\mathrm{C}(X)$, a subfamily $A$ of functions are of particular interest. For simplicity, one can think of $X$ as a closed, bounded interval $[a, b]$ in $\mathbb{R}$ and $A$ as the set of polynomials in a single variable with real coefficients. Since linear combination and product of polynomials are also polynomials, we assume that $A$ is closed under addition, scalar multiplication and product and also a non-zero constant function belongs to $A$ (This actually implies that $A$ contains all constant functions).

We say an element $f \in \mathrm{C}(X)$ 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 when every $f \in \mathrm{C}(X)$ 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 tolerance $\epsilon$. To address this issue, and the search for an object begins.

Define $\|f\|$ (the $\sup$ norm of $f$) of a given function $f \in \mathrm{C}(X)$ by $\|f\|_\infty = \sup_{x\in X} |f(x)|$,

Then, the above argument can be read as: *For every $f \in C(X)$ and every $\epsilon > 0$, there exists $p \in A$ such that* $\|f - p\|_\infty < \epsilon$. It is easy to see that $\|0\|_\infty = 0$, $\|\lambda f + g\|_\infty \leq |\lambda|\|f\|_\infty + \|g\|_\infty$ (subadditivity) and $\|f \times g\|_\infty \leq \|f\|_\infty \times \|g\|_\infty$. The function $\|\cdot\|_\infty$ on $\mathrm{C}(X)$ is an instance of *norm* on the function space $\mathrm{C}(X)$, which also resembles the structure of inner product spaces which have nice geometry and one can define and develop intuitive concepts over them easily. Let $V$ be an $\mathbb{R}$-vector space, an *inner product* on $V$ is a function $\langle\cdot,\cdot\rangle : V \times V \to \mathbb{R}$ satisfying the following requirements:

1. $\langle f, f\rangle \geq 0$;
2. $\langle f, f\rangle = 0$ if and only if $f = 0$;
3. $\langle \alpha f + \beta g, h\rangle = \alpha\langle f, h\rangle + \beta\langle g, h\rangle \geq 0$ for every $\alpha, \beta \in \mathbb{R}$;
4. $\langle f, g\rangle = \langle g, f\rangle$.

The pair $(V, \langle\cdot,\cdot\rangle)$ is called inner product space and the function $\|v\| = \langle v, v\rangle^{\frac{1}{2}}$ induces a norm on $V$. A basis $\{v_\alpha\}_{\alpha\in I}$ is called an orthonormal basis for $V$ if $\langle v_\alpha, v_\beta\rangle = \delta_{\alpha\beta}$, where $\delta_{\alpha\beta} = 1$ if and only if $\alpha = \beta$ and is equal to 0 otherwise. Every given set of linearly independent vectors can be turned into a set of orthonormal vectors that spans the same sub vector space as the original. The following well-known result gives us an algorithm for producing such orthonormal vectors from a set of linearly independent vectors:

**Theorem 2 (Gram–Schmidt)** *Let $(V, \langle\cdot,\cdot\rangle)$ be an inner product space. Suppose $\{v_i\}_{i=1}^n$ is a set of linearly independent vectors in $V$. Let $u_1 := \frac{v_1}{\|v_1\|}$ and (inductively) let $w_k := v_k - \sum_{i=1}^{k-1}\langle v_k, u_i\rangle u_i$ and $u_k := \frac{w_k}{\|w_k\|}$, then $\{u_i\}_{i=1}^n$ is an orthonormal collection, and for each $k$, $span\{u_1, u_2, \cdots, u_k\} = span\{v_1, v_2, \cdots, v_k\}$.*

Note that in the above theorem, we can even assume that $n = \infty$.

Let $B = \{v_1, v_2, \dots\}$ be an ordered basis for $(V, \langle \cdot, \cdot \rangle)$. For any given vector $w \in V$ and any initial segment of $B$, say $B_n = \{v_1, \dots, v_n\}$, there exists a unique $v \in \text{span}(B_n)$ such that $\|w - v\|$ is the minimum:

**Theorem 3** *Let $w \in V$ and $B$ a finite orthonormal set of vectors (not necessarily a basis). Then, for $v = \sum_{u \in B} \langle u, w \rangle u$, we have $\|w - v\| = \min_{z \in span(B)} \|w - z\|$.*

Now, let $\mu$ be a finite measure on $X$ and for $f, g \in \mathrm{C}(X)$ define $\langle f, g \rangle = \int_X fg d\mu$. This defines an inner product on the space of functions. The norm induced by the inner product is denoted by $\| \cdot \|_{2,\mu}$. It is evident that $\|f\|_{2,\mu \leq} \|f\|_\infty \mu(X),\ \forall f \in \mathrm{C}(X)$ which implies that any good approximation in $\| \cdot \|_\infty$ gives a good $\| \cdot \|_{2,\mu}$-approximation. But generally, our interest is the other way around. Employing Gram–Schmidt procedure, we can find $\| \cdot \|_{2,\mu}$ within any desired accuracy, but this does not guarantee a good $\| \cdot \|_\infty$-approximation. The situation is favorable in finite dimensional case. Take $B = \{p_1, \dots, p_n\} \subset \mathrm{C}(X)$ and $f \in \mathrm{C}(X)$, then there exists $K_f > 0$ such that for every $g \in \text{span}(B \cup \{f\})$,

$\mathrm{K}_f \|g\|_\infty \leq \|g\|_{2,\mu \leq} \|g\|_\infty \mu(X)$. (1)

Since $X$ is assumed to be compact, $\mathrm{C}(X)$ is separable, i.e., $\mathrm{C}(X)$ admits a countable dimensional dense subvector space (e.g. polynomials for when $X$ is a closed, bounded interval). Thus for every $f \in \mathrm{C}(X)$ and every $\epsilon > 0$, one can find a big enough finite $B$, such that (1) holds. In other words, *"good enough $\| \cdot \|_{2,\mu}$-approximations of $f$ give good $\| \cdot \|_\infty$-approximations"*, as desired.

In practice, $X = [a, b]$ and the countable dimensional subspace is the algebra of polynomials which satisfies the assumption of the Stone–Weierstrass theorem and the set of monomials is admissible in Gram–Schmidt process. Different choices of $\mu$, gives different systems of orthogonal polynomials. For example, if we choose $d\mu = dx$ on $[-1, 1]$, then we get Legendre polynomials and if we choose $d\mu = \frac{dx}{\sqrt{1-x^2}}$ on $[-1, 1]$, then we get Chebyshev 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 orthogonal system of polynomials. As the first choice, we consider Chebyshev polynomials on the stretched interval which come from the measure

$$d\mu = \frac{dx}{l\sqrt{1 - (x/l)^2}}. \tag{2}$$

Our second choice comes from the measure.

$$d\mu = e^{-(l/x)^2} dx. \tag{3}$$

We note that the measure for Chebyshev polynomials mainly concentrates at the end points of the interval which causes interpolation at mostly initial and end points with two singularities at both ends. While the second measure evens out through the whole real line and puts zero weight at the center. This behavior causes less oscillation in the resulting approximation and hence more similarities of derivatives with sigmoid function. We use SageMath (4) to find coefficients of Chebyshev polynomial for the sigmoid function $f(x) = \frac{1}{1+e^{-x}}$.

The results show that the higher degrees give a better approximation and we expect that a better approximation will be a better replacement for the sigmoid function in the neural network.

## 3 Experimental Results

### 3.1 Polynomial Approximation in Neural Networks

Once the polynomial approximations are calculated, we replace the activation function in neural network with approximation polynomials and analyze the performance of neural network. We utilize Neural Network Toolbox (10) to implement the neural network and use 15 datasets from UC Irvine Machine Learning Repository (8).

We compute polynomial approximations for sigmoid function based on four different parameters: degree, error, intervals and precision of coefficients. To compare the performance of our polynomial approximations with other studies, we use several activation functions for training: sigmoid function $f_1(x) = \frac{1}{1+e^{-x}}$, another variation of sigmoid function $f_2(x) = \frac{2}{1+e^{-4x}} - 1$, and square function $f_3 = x^2$ which was proposed in (9) and (5).

The results show that our polynomial approximation is able to achieve the best accuracy if we choose the interval properly based on the dataset.

### 3.2 Polynomial Computation over Encrypted Values

We run the polynomials over encrypted data and measure the computation cost of this step. We used HELib for implementation. For generating schemes in HELib, we set some values for input parameters, $k = 80$, $s = 1$, $w = 64$ (for details about these parameters see (7)). The other important value is $L$. Small values for $L$ decrease the computation time, however the degree of polynomial also decreases. Large values of $L$ lead to slower computations and larger degrees for polynomials. Size of ciphertext and the time for generating encryption scheme increased by increasing the value for $L$.

We used polynomials from degrees 2 to 9 and increased the value of $L$ from 5 to 25 as the degree of the polynomials increased. All the computations are in mod $10^{18} + 37$, which is a prime number. For example, for degree 2 running time is 0.1 seconds with $L = 9$ and for degree 3 running time is 0.3 seconds with $L = 15$.

### 3.3 Neural Network over Encrypted Data

For implementing neural networks over encrypted data, we need to revise the structure of neural network to have an efficient and compatible structure with HE schemes. In this work, we focus on fully connected feed-forward neural networks. We performed experiments for learning from datasets with different numbers of features and different numbers of classes.

To address the noise problem, instead of bootstrapping, we use an alternative approach where the server checks the level of noise in the ciphertext after each operation and if the noise level is lower than the threshhold (i.e. 2), the server sends the ciphertext to the client and the client decrypts and encrypts it again and sends the fresh ciphertext back to the server. Our results show that communication between the client and the server is the most time consuming part of the process. Therefore, we should choose our parameters in the encryption schemes in a way to have fewer communications. Two values that have impact on the number of interaction between the client and the server are $L$ and $p$. If we choose a proper prime number $p$, we could use smaller values for $L$ which consequently decrease the size of ciphertext and time of computations.

Although our approach requires some communications between the client and the server, it has several advantages over multi-party computation (MPC). The main advantage is the number of communications. In MPC protocols, we need a number of interactions between client and server for each operation whereas in our approach, the server does not need to interact with the client unless the amount of noise reaches a threshold. If the same neural network is implemented using MPC protocols, number of communications grows enormously. Another advantage is that unlike MPC approaches, the structure of the client does not need to be changed when the learning algorithm is changed. The only operations in client side are encryption and decryption and server can perform different learning algorithms with the same client.

## 4 Conclusion

In order to provide privacy preserving multi-party machine using homomorphic encryption, we developed a theoretical foundation for approximating activation functions with low degree polynomials and provided an approach to generate those approximations based on Chebyshev polynomials. We then used these approximation to train neural networks with polynomial approximation as activation function over encrypted data. We are working on extending our approach to scenarios where the datasets are distributed across multiple parties and computations are decentralized. We also plan to study approximation of non-continuous functions used in deep learning algorithms.

## References

[1] L. J. M. Aslett, P. M. Esperança, and C. C. Holmes, “A review of homomorphic encryption and software tools for encrypted statistical machine learning,” University of Oxford, Tech. Rep., 2015.

[2] L. J. M. Aslett, P. M. Esperança, and C. C. Holmes, “Encrypted statistical machine learning: new privacy preserving methods,” *CoRR*, vol. abs/1508.06845, 2015.

[3] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, “Machine learning classification over encrypted data,” in *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2014*, 2015.

[4] T. S. Developers, *SageMath, the Sage Mathematics Software System (Version 7.1)*, 2016, `http://www.sagemath.org`.

[5] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. L. M. Naehrig, and J. Wernsing, “Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy,” Tech. Rep. MSR-TR-2016-3, 2016. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=260989

[6] T. Graepel, K. Lauter, and M. Naehrig, “ML confidential: Machine learning on encrypted data,” in *Information Security and Cryptology–ICISC 2012*. Springer, 2013, pp. 1–21.

[7] S. Halevi and V. Shoup, “Algorithms in helib,” in *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, 2014, pp. 554–571.

[8] M. Lichman, “UCI machine learning repository,” 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[9] R. Livni, S. Shalev-Shwartz, and O. Shamir, “On the computational efficiency of training neural networks,” *CoRR*, vol. abs/1410.1141, 2014.

[10] P. Pilotte, “Neural Network Toolbox,” 2016. [Online]. Available: http://www.mathworks.com/products/neural-network/

[11] P. Xie, M. Bilenko, T. Finley, R. Gilad-Bachrach, K. Lauter, and M. Naehrig, “Cryptonets: Neural networks over encrypted data,” Tech. Rep., 2014.