Order-preserving Encryption as a Tool for Privacy-Preserving Machine Learning

Nikita Lisin, Sergey Zapechnikov
Institute of Cyber Intelligence Systems, National Research Nuclear University (Moscow Engineering Physics Institute)
Moscow, Russia
lisin_ns@mail.ru, svzapechnikov@mephi.ru

***Abstract*—An order-preserving encryption is an encryption scheme based on strictly increasing functions. It allows mapping a set of plaintext into a set of ciphertexts with the order relation specified on a set of plaintexts. Thus, data comparisons and search for the minimum or maximum elements become possible without decrypting data. This type of encryption is mainly used to protect cloud databases in the case when it is necessary to make queries to them. However, it is almost not used in privacy-preserving data mining and machine learning. Nevertheless, it is possible to use this type of encryption in privacy-preserving machine learning, but only for certain algorithms. In this paper, we consider some existing order-preserving encryption schemes and suggest some cases of machine learning, where they can be applied to obtain correctly working privacy-preserving machine learning algorithms. An explanation is also given how and why it is possible to apply order-preserving encryption to these algorithms.**

Keywords—order-preserving encryptio;, machine learning; cloud service; privacy-preserving machine learning

I. Introduction

One of the main problems of machine learning is the need for a large amount of memory and long learning time. In this regard, at present, most companies are switching to Machine Learning as a Service technology, which allows placing the trained model and data in the cloud.

However, placing models and data in the cloud in the plain form is not always safe. Cloud storage is often attacked, which leads to data leakage. In addition to data leakage, there is a problem with their change. As a result, the user may be incorrectly identified and receive a refusal.

One way to deal with the above problems is to store data in an encrypted form whenever possible. The issue of training on encrypted data is especially acute for the banking and medical sectors.

In the paper [1], two main approaches to privacy-preserving machine learning (PPML) were considered:

- perturbation approach;
- cryptographic approach.

The perturbation approach is based on the addition of noise to the data. This leads to lower data accuracy than in the cryptographic approach. However, the throughput of the noise approach is much faster. This approach includes the following methods:

- dimension reduction;
- differential privacy;
- local differential privacy.

The cryptographic approach, in contrast to the noise approach, uses cryptographic primitives to preserve data privacy. This fact significantly slows down the speed of operation, but at the same time allows maintaining accuracy as for standard machine learning methods. The cryptographic approach includes the following methods:

- homomorphic encryption;
- garbled circuits;
- secure processor;
- order-preserving encryption.

Among all methods, the order-preserving encryption was chosen for further analysis. The reason for this is that order-preserving encryption is faster than other types of encryption and gives good accuracy results.

Order-preserving encryption (OPE) is a type of encryption that allows mapping the order relation specified on a set of plaintexts to the set of ciphertexts. It is widely used to encrypt databases located in a cloud environment. The reason for this is as follows:

- cloud services are often attacked, leading to data leakage, as stated above;
- OPE guarantees the correctness of the comparison operations and the search for minimum/maximum on encrypted data.

However, this encryption can be used in cloud services not only to protect databases but also for privacy-preserving machine learning. But on the other hand, since this method only saves comparison operations, it is very limited in application.

This work explains, for which machine learning algorithms and why these encryption schemes can be applied.

978-1-7281-5761-0/20/$31.00 ©2020 IEEE

II. Existing OPE Schemes

The first and simplest example of OPE is the addition of random numbers [2]. The essence of this method is that the encrypted number c is calculated by the formula:

$$c=\sum_{i=1}^{P} R_i \tag{1}$$

where R_i is the ith value of the random number generator.

A more complex version of OPE is the use of a sequence of strictly increasing polynomial functions [3]. During encryption, these functions are arranged sequentially, that is, the result of one function is the input to another. Polynomial functions can be of two types [4]:

- $ax+b,\ a>0$
- $ax^{2k+1}+b,\ a>0,\ k\geq 0$

The next improvement is the splitting of the initial segment of numbers into random segments and the use of each of them with its monotonically increasing linear function [3]. These functions are chosen in such a way that they form one piecewise linear monotonically increasing function over the entire initial segment.

A new OPE concept called MOPE was proposed by [5]. Its essence lies in the fact that when encrypting text, a secret modular shift is added to it. This improvement works well, provided that the attacker can only use an encrypted database. However, in the case when he has access to plaintext, he can calculate this shift by making the required number of queries.

Another version of OPE was proposed by the authors of [6]. It involves adding noise to a simple linear expression. In this regard, encryption is as follows:

$$Enc(x)=ax+b+noise. \tag{2}$$

Noise is determined by the sensitivity of the input values. The secret parameter a must be greater than zero, but there are no special requirements for the parameter b. The disadvantage of this approach is that if there are duplicates in the initial values, these secret parameters can be calculated.

To combat duplicates in [6], it is proposed [7] to modify the formula by adding f(x):

$$Enc(x)=af(x)x+b+noise \tag{3}$$

To the requirements of equation (3), the demand is added to the function f(x). It must be positive and strictly monotonically increasing. The encryption algorithm is more complex; however, it is resistant to duplicates in the source text.

The fastest calculation option was proposed in [8]. This algorithm is based on the general approximate common divisor problem [9]. Due to this, its computational complexity is O(1). However, the level of security is the same as for other linear algorithms.

All previous OPE variations were anyway based on linear functions. The model proposed by the authors [10] represents client-server architecture. The client has some data that needs to be encrypted and stored on an untrusted side of the server. Encrypted data is stored as a binary search tree. For a binary search tree for each node v, all nodes in the left subtree are strictly less than v, and all nodes in the right subtree are strictly greater than v. Moreover, B-trees are used in the implementation, since they have advantages such as the logarithmic time of insertion, deletion, search, and allow efficient ciphertext updates. Each node of such a tree stores plaintext encrypted with a client key according to the corresponding order relation.

III. Decision trees and gradient boosting

A decision tree is a machine learning algorithm that reproduces a logical scheme that allows getting a final decision on the classification of an object using answers to a hierarchically organized system of questions. When constructing a decision tree at each step, one question is asked, which opens up the greatest new information about objects. There are many ways to measure the amount of new information. Most commonly used is the concept of information entropy:

$$S=-\sum_{i=1}^{N} p_i*\log_2(p_i) \tag{4}$$

The decision tree construction algorithm is as follows [11]:

- calculate the entropy in the current state (S_0);
- if S_0=0 then stop;
- otherwise, start looking for a partition according to some feature from the set of features X, which most of all reduces the heterogeneity of the node (E):

$$E=S_{old}-S_{new}=S_{old}-\frac{n_{left}}{n}S_{left}-\frac{n_{right}}{n}S_{right} \tag{5}$$

- do the partitioning and repeat the procedure first for the resulting subsets.

When the decision tree has built, all the elements of the training set pass through it and end up in some leaves. The answer is assigned to the sheet by the principle of voting: which class of elements dominates in the sheet, such will be the answer on it. Since machine learning algorithms can work only with numbers, and not with textual information, in the nodes of the trained decision tree, the feature is compared with the number obtained as a result of training, according to which the partition was performed in this node. Due to this fact, this algorithm is stable against non-standard distributions and does not require preliminary normalization of data, and can also work with data encrypted using OPE.

However, one decision tree does not work well. But if you combine several decision trees into an ensemble, then the results will be much better. There are several ways to combine decision trees into ensembles. The two most famous are Random Forest and Gradient Boosting.

In a Random Forest, each tree is trained in parallel and its result does not depend on the results of other trees. In gradient boosting, training takes place sequentially. Thus, each subsequent decision tree takes into account the errors of the previous one.

We denote the ith decision tree by b_i, then the ensemble of N decision trees for gradient boosting will look as follows:

$$a(x)=\sum_{i=1}^{N} w_i b_i(x) \tag{6}$$

To account for errors, it is necessary to calculate the loss function L for all elements in a dataset (let the dataset consist of M elements):

$$\text{error}(a)=\sum_{j=1}^{M} L(y_j,a(x_j)) \tag{7}$$

The gradient boosting algorithm begins with the ensemble initialization:

$$a_0(x)=\text{argmin}_c\left(\sum_{j=1}^{M} L(y_j,C)\right) \tag{8}$$

Then the gradient is calculated for all i from 1 to N:

$$g_{ij}=-\left[\frac{\partial L(y_j,a(x_j))}{\partial a(x_j)}\right],a=a_{i-1} \tag{9}$$

After that a tree is built with the target variable:

$$b_i=\text{argmin}_b\left(\sum_{j=1}^{M} (g_{ij}-b(x_j))^2\right) \tag{10}$$

We consider the weight for this tree:

$$w_i=\text{argmin}_w\left(\sum_{j=1}^{M} L(y_j,a_{i-1}(x)+wb_i(x_j))\right) \tag{11}$$

And finally, we add the tree to the ensemble:

$$a_i(x)=a_{i-1}(x)+w_i b_i(x) \tag{12}$$

Each subsequent tree tries to predict the gradient of this function. A simplified diagram of the gradient boosting operation is presented in Figure 1. Due to this structure, the gradient boosting algorithm works better than many other machine learning algorithms. This is confirmed by [12].

This research compared various machine learning algorithms for the data classification problem. The comparison involved more than 160 known data sets for this task. The results were presented in the form of a heat map, which shows that gradient boosting was the best. In most tasks, he gave accuracy higher than other algorithms.

CONCLUSION

Machine learning is becoming increasingly popular as a service, which involves placing the model and data in the cloud. However, the data there needs to be protected. The fastest and most accurate way to use order-preserving encryption. However, this imposes restrictions on the machine learning algorithm that we can use.

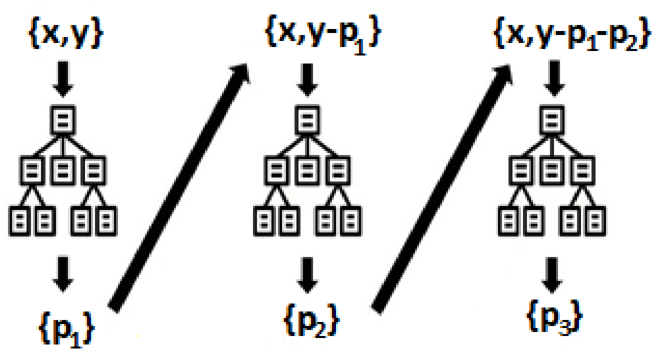

Fig. 1. Gradient Boosting Visualization.

As shown in the article, one of the algorithms compatible with order-preserving encryption is gradient boosting. This algorithm in many cases gives accuracy higher than others. Therefore, a good idea for developing a privacy-preserving machine learning algorithm that will work faster and give better accuracy is to use gradient boosting along with order-preserving encryption. In further researches, we plan to conduct experiments and numerically confirm this statement.

REFERENCES

[1] Lisin, N. Methods and Approaches for Privacy-Preserving Machine Learning / N. Lisin, S. Zapechnikov // Proc. of Intelligent Technologies in Robotics Conference. Moscow. 2019. 7 pp.

[2] Bebek, G. Anti-tamper database research: Inference control techniques. Technical Report EECS 433 Final Report, Case Western Reserve University, November 2002.

[3] Ozsoyoglu G. Anti-tamper databases: Querying encrypted databases / G. Ozsoyoglu, D. Singer // Proc. of the 17th Annual IFIP WG 11.3 Working Conference on Database and Applications Security, Estes Park, Colorado, August 2003

[4] Krendelev, S. Order-preserving encryption. Main concepts, High availability systems, №3 2013

[5] Boldyreva, A. Order-preserving encryption revisited: improved security analysis and alternative solution. / A. Boldyreva, N. Chenette, A. O'Neill // Advances in Cryptology-CRYPTO 2011. Berlin, Heidelberg: Springer; 2011, p. 578–95.

[6] Liu, D. Programmable order-preserving secure index for encrypted database query / D. Liu, S. Wang // Proceedings of the 5th IEEE International Conference on Cloud Computing, Honolulu, Hawaii, USA, 2012; 502–509.

[7] Liu, D. Nonlinear order-preserving index for the encrypted database query in service cloud environments / D. Liu, S. Wang // Concurrency and Computation Practice and Experience 25(13), September 2013.

[8] Dyer, J. Order-Preserving Encryption using approximate integer common divisors / J. Dyer, M. Dyer, J. Xu // Springer International Publishing, 2017.

[9] Howgrave-Graham, N. Approximate Integer Common Divisors. Cryptography and Lattices. Springer, 2001, pp. 51–66.

[10] Popa, R. An Ideal-Security Protocol for Order-Preserving Encoding / R. Popa, F. Li, N. Zeldovich // MIT CSAIL, 2011.

[11] Navlani, A. Decision Tree Classification in Python, https://www.datacamp.com/community/tutorials/decision-tree-classification-python, last accessed 2019/09/01.

[12] Gough, J. Alteryx Machine Learning Theory: Gradient boosted model [Электронный ресурс]. — URL: https://www.thedataschool.co.uk/jamie-gough/alteryx-machine-learning-theory-gradient-boosted-model, last accessed 2019/09/10.