3S-06

A Privacy Preserving Protocol for Cloud-based Machine Learning Implementation

橋本雅人† 趙強福†

会津大学コンピュータ理工学部†

Abstract

This paper proposes a privacy preserving protocol for implementing machine learning based on cloud server. Since the mobile terminals have very limited computational resources, cloud server-based implementation is considered more efficient for various mobile applications. But cloud-based approaches may have information leakage and privacy issues. The server or third person can see and collect the user's sensitive private data easily. Our proposed protocol can be useful to solve these problems. The protocol implements a neural network using both a mobile device and the server; and the part on the server is only a black box. Experimental results reveal that the protocol works well while preserving both the user data and the prediction model.

1. Introduction

As mobile terminals such as smartphones become more and more popular, a lot of smartphone applications are developed and used for various purposes. To make the applications more intelligence and effective, machine learning (ML) technologies can be used. But, it might be difficult to implement an ML model because it is usually designed for large-scale computing environments. A mobile device has limited resources (e.g., CPU, memory, battery and storage). To solve this problem, client-server implementation model can be used. In this approach, a client is used as input/output, and the server is used for ML-based prediction. However, this method poses information leakage and privacy invasion problems. If the ML models are conducted in the server, it will be easy to analyze the user's requests, intentions and sensitive information. Even if the server is trustable, some malicious third person may attack and see the personal information easily. Therefore, the server-based prediction services may not be trustable. To solve the problem, we proposed a privacy preserving protocol using extreme learning machine (ELM) [1]. The protocol can reduce the calculation cost on the mobile terminal by using server calculation while preserving the privacy. The server can only see the encrypted user data and a random matrix (black-box) calculation. The final decision of the ML model may not be seen from server-side. The protocol will make the server trustable.

2. Privacy preserving protocol

The privacy preserving protocol is proposed by us to solve limited resources problem and privacy problem. The main idea is to realize an ML model by ELM that is basically a single hidden layer feedforward neural network (NN), and to divide the ELM into server side and client (mobile terminals) side. The server has a weight matrix for the hidden layer; and the client has a weight matrix for the output layer. The hidden weight matrix for the ELM is generated at random and is fixed. Only the output weight is trained by using the training data. This means server-side has only random numbers that does not contain user's sensitive data. Even if the model on the server is stolen, nothing will be leaked to malicious person because the server has only random numbers.

The basic protocol works as follows. First, the mobile terminal gets a datum (feature vector) from the user. Then, the terminal sends the datum to the sever. The server calculates the hidden layer outputs with the random matrix, and sends back the outputs of the hidden neurons. The mobile terminal then calculates the final output.

We proposed the following three methods to improve the usefulness and security. 1) User input encryption based on a transposition cipher; 2) Random weight matrix sharing by multiple applications; and 3) Decision making with part of the results obtained from the server.

A privacy preserving protocol for machine learning prediction on mobile terminals

† Masato Hashimoto ・ University of Aizu

‡ Qiangfu Zhao ・ University of Aizu

By using these methods, the server input, server calculation, and server output are all protected. (See Fig. 1)

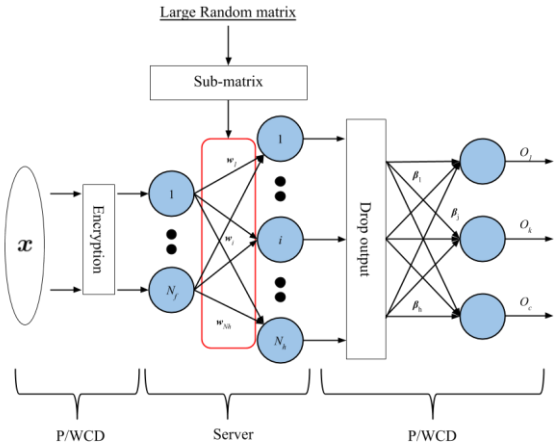

Fig. 1. Neural network diagram of the protocol

The classification flow of the protocol is as follows. First, the user input is encrypted using transposition cipher with a "trans-key". The terminal then sends the encrypted input along with a "pos-key". The server gets a sub-matrix of the large random weight matrix using the pos-key. The matrix on the server can be shared by multiple applications. Then the server calculates the outputs of the hidden neurons and sends back the results. Here the number of hidden neurons was set larger than the required value. After that the client terminal drops some elements of the output from server by using a "drop-key" which defines the indices of the elements to drop. This method adjusts the number of hidden neurons to required value. Then, the terminal gets a vector that is a part of output vector of the hidden layer. By this method, the server cannot see which element of the hidden layer are used to obtain the final decision. Finally, the mobile terminal calculates output layer and find a result.

In the training phase, multiple trans-keys, pos-keys and drop-keys are randomly generated and used as training parameters. The output weight is generated through learning based on training data corresponding to these keys. The classification performance of the protocol is the same as that of the original ELM.

3. Experiments and Discussions

First, we compared the accuracy of ELM and the proposed protocol by 10 times 5-fold cross validation using the same parameters. The public dataset MNIST was used. The results are 90.7% for the ELM, and 90.5% for the protocol. There is no significant difference.

The next experiment is to compare the classification times for the proposed protocol and a method that implements the ML model using only the mobile device (all-in-device). As the mobile device, Android Nexus 6 was used. The experiment environment is one smartphone, one router and one server. The classification time is the time between capturing an input datum and obtaining the result. We measured the classification times of 30 runs. Fig.2 shows the averaged classification time of MNIST. The vertical axis shows the time in second (smaller is better). The horizontal axis shows 3 type of methods MLP (all-in-device) trained by Adam, ELM (all-in-device) and the protocol. The size (number of hidden neurons) of all ML models are 2000. These values are found by grid-search. We can see that our protocol is the fastest in these three methods. So, the protocol is effective if the application NN is relatively large.

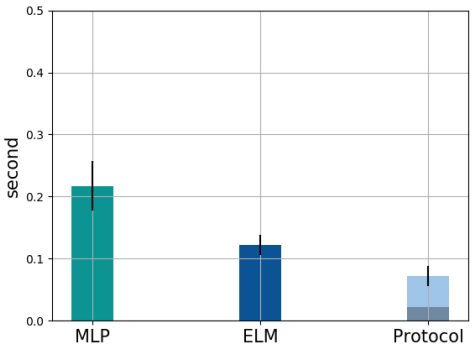

Fig. 2 The classification time of MNIST dataset

4. Conclusions

In this study we proposed a privacy preserving protocol to implement ML-based applications using mobile terminal and cloud server. Experimental results show that our protocol can keep the classification performance and make decisions more quickly for the MNIST dataset.

References

[1] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," Neuro computing, vol. 70, no. 1, pp. 489–501, 2006.