weekly report 2019-2-2

Zhiyao Sheng
yao907110651@gmail.com

Abstract

Recently I tried the idea of the encrypted neural network. The dataset, weight, and bias are all encrypted during training. However I am not very positive about its application – it's too slow, maybe thousands of times slower than the usual training process. Anyway, I proved its viability and overfitted a simple 2-layer-network with random input and target. I still used the vector homomorphic encryption.

1 How does it work?

This process is suitable for encrypted data and network. First the client will have to compute some matrices using the plain text and secret keys and send these matrices to the server. There are a lot of secret keys for different weight, bias and processed data at different layers during the propagation because we should encrypt vectors of different length with different secret keys, which is inconvenient. Those matrices are only computed once.

The server will accept these matrices and encrypted dataset. It also holds the encrypted weight and bias of the network. It compute the encrypted derivatives of the weight and bias and update the encrypted weight and bias using these encrypted derivatives. Because of the peculiarity of this process, the encryption strategy I used last semester doesn't work anymore. I have to use special methods provided by the encryption scheme to perform forward and backward propagation instead of using the convolutional layers and fully-connected layers directly. According to my calculation, the time complexity of the propagation of an usual fully-connected layer is $\underline{mn^2}$ where the input is vector of length n and the matrix's shape is $m \times n$. For the encrypted fully-connected layer, the time complexity is m^3n^4l(l is a parameter of the encryption, usually being 6 to 100). So that I don't think this method is suitable for training a complex network. Maybe it's better to train it on your own slower computer without encryption.

2 The encryption strategy

Because the weight and bias are encrypted and I have to calculate the inner product of the weight and input and both of them are encrypted, The strategy I used last semester isn't suitable. I can't encrypt the input in the batch dimension. Instead, I encrypted them on the other dimension. Take an example, there is an input of shape $m \times n$ whose batch size is m. In the past, I encrypted it in the firt dimension so that it becomes a ciphertext of shape $m' \times n$ and now I encrypt it on the second dimension so that its shape becomes $m \times n'$.

And to compute the forward propagation of a layer is calculating the inner product of the weight and input, 2 ciphertext. The inner product process is quite slow using vector homomorphic encryption. I don't know if other encryption scheme would provide a faster inner product calculation but it will not change the fact that the calculation is over hundreds of times slower since we can't use the optimized algorithms of layers provided by tensorflow and pytorch.

3 Conclusion

Well, I have to admit that I feel negative about the prospect of the encrypted neural network and I am not very enthusiastic about it.