A Note on Our Submission to Track 4 of iDASH 2019

Marcel Keller, Ke Sun
CSIRO's Data61

March 30, 2020

Abstract

iDASH is a competition soliciting implementations of cryptographic schemes of interest in the context of biology. In 2019, one track asked for multi-party computation implementations of training of a machine learning model suitable for two datasets from cancer research. In this note, we describe our solution submitted to the competition. We found that the training can be run on three AWS `c5.9xlarge` instances in less then one minute using MPC tolerating one semi-honest corruption, and less than ten seconds at a slightly lower accuracy. After seeing some winning solutions, we have lowered this figure to less than a second.

1 Introduction

In this section, we summarize the task.[1] Participants were invited to implement machine learning (ML) training in multi-party computation (MPC). MPC allows a set of parties to jointly compute on data hold among each other without revealing anything other than the result of the computation. In the current context this means that a set of healthcare providers holding measurements about cancer patients and healthy individuals can jointly compute a ML model detecting cancer without sharing the measurements.

The training algorithm must be suitable to the GSE2034 [17] and BC-TCGA [14] breast cancer datasets. The organizers provided a subset of both. For GSE2034, this contains 142 positive (recurrence tumor) and 83 negative (no recurrence normal) samples, each with 12,634 features. On the other hand, the subset of BC-TCGA contains 422 positive (breast cancer tissue) and 48 negative (normal tissue) samples of 17,814 features each. The organizers provided a reference model, and the submissions were expected to perform similarly.

In terms of security, the competition asked for three-party computation with one semi-honest corruption. This security model has received widespread attention because it does not require relatively expensive cryptographic primitives such as oblivious transfer or homomorphic encryption. Instead, so-called replicated secret sharing suffices, where every party holds two out of three random shares which sum up to a secret value [2]. Furthermore, semi-honest security requires that even corrupted parties follow the protocol, which allows the creation of optimized protocols for specific purposes, for example probabilistic truncation [6].

For the evaluation, the organizers asked for Docker containers that would be run on three hosts in a local cluster. The submissions were required to finish within 24 hours, and they were ranked on accuracy, performance, and communication.

2 The Model

We evaluated the performance of the following baseline models using plaintext computation

[1] `http://www.humangenomeprivacy.org/2019/competition-tasks.html`

Logistic Logistic regression based on mini-batch stochastic gradient descent (SGD) with constant learning rate and momentum. The model is trained for a fixed number of 100 epochs. In each mini-batch of 16 samples, we re-balance the positive and negative classes by re-sampling the same number of samples from these two classes;

MLP A multilayer perceptron with two hidden layers of 256+64 ReLU [8] neurons. We apply dropout [16] with probability 0.5 to the hidden layers to avoid over-fitting. The model is trained by SGD for 100 epochs using the same mini-batches as Logistic;

Linear SVM A linear SVM classifier with L_2 regularization and typical settings. We use scikit-learn's [15] implementation with a stopping tolerance of 10^{-3} and other default settings;

Random Forest A random forest classier [10] with 100 trees based on scikit-learn's [15] implementation;

Reference The reference model provided by the iDASH committee, which is a deep 1D convolutional network composed of Residual blocks [9]. We train the model using the Adam optimizer [12] with a fixed learning rate of 10^{-4} for 30 epochs.

In these models, both Logistic and Linear SVM correspond to a single-neuron model. The main difference between these two methods is the loss function: Logistic minimizes the cross-entropy between the target label and the prediction, while Linear SVM minimizes the squared Hinge loss. The detailed hyper-parameter configurations are omitted for brevity. We tried to achieve a representative accuracy score for each method. Further tuning these methods can give marginal improvement. We tried to compare all methods using similar settings such as how the mini-batches are constructed and how the performance is evaluated.

Denote TP (true positive), FN (false negative) to be the population of the ground truth positive class; denote TN (true negative), and FP (false positive) to be the negative class. Then the F_1 score of these two classes are given by the harmonic mean of the precision and recall, that is,

$$F_1^{\mathrm{P}} = \frac{2}{\frac{1}{\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}} + \frac{1}{\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}}} = \frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}, \quad F_1^{\mathrm{N}} = \frac{2\mathrm{TN}}{2\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}. \tag{1}$$

Then we evaluate the classification performance based on the weighted F_1 score

$$F_1 = \frac{\mathrm{TP}+\mathrm{FN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}} F_1^{\mathrm{P}} + \frac{\mathrm{TN}+\mathrm{FP}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}} F_1^{\mathrm{N}}, \tag{2}$$

which is a population-weighted mean of F_1^{P} and F_1^{N}.

See Table 1 for our cross-validation accuracy scores and timing results. Observe that MLP does not improve over Logistic, because the training data is linearly separable and prone to over-fitting. The poor performance of the reference model, despite its expensive training cost, is due to over-fitting. One may improve further its performance with a carefully designed optimization procedure, e.g., based on dynamic learning rates. However, the benefit is quite limited in light of the high complexity to implement it.

Eventually, we decided to go ahead with logistic regression because its accuracy comes close the other models while its simplicity promised a more efficient implementation. In the next section, we will show that our MPC implementation slightly surpasses the plaintext implementation of logistic regression and comes even closer to the linear SVM classifier in terms of accuracy.

3 Our Implementation

We implemented our solution in MP-SPDZ [7]. It features fixed-point computation, that is, the fractional number x is represented as an integer near $x \cdot 2^k$ for some k. Addition and subtraction are straight-forward due to linearity while multiplication is implemented as integer multiplication followed by truncation. This truncation can either mean rounding to the nearest integer or the more efficient probabilistic truncation where

	GSE2034		BC-TCGA	
Model	F_1	Time (sec)	F_1	Time (sec)
Logistic	0.666 ± 0.062	3	0.995 ± 0.006	8
MLP	0.664 ± 0.066	35	0.994 ± 0.007	39
Linear SVM	0.677 ± 0.058	1	0.996 ± 0.006	0.3
Random Forest	0.592 ± 0.055	0.6	0.988 ± 0.011	0.8
Reference	0.650 ± 0.075	$68^\star$	0.987 ± 0.008	$139^\star$

Table 1: Weighted F_1 score and computational time in seconds with plaintext computation. All reported number are averages over 20 runs of five-fold cross validation (100 folds in total). The reported time in seconds is measured on an Intel Core i5-7300U CPU. The upper-script "$\star$" means that the timing is performed instead on a NVidia Tesla P100 because the experiments could not finish in reasonable time.

Dataset	Duration	Truncation	F_1	Time (sec)
GSE2034	100	Probabilistic	0.670 ± 0.070	8
		Exact	0.666 ± 0.068	20
	200	Probabilistic	0.674 ± 0.063	14
		Exact	0.670 ± 0.066	38
	Variable	Probabilistic	0.657 ± 0.091	6
		Exact	0.650 ± 0.091	22
BC-TCGA	100	Probabilistic	0.994 ± 0.007	21
		Exact	0.994 ± 0.008	43
	200	Probabilistic	0.994 ± 0.007	38
		Exact	0.994 ± 0.009	77
	Variable	Probabilistic	0.995 ± 0.010	8
		Exact	0.994 ± 0.011	17

Table 2: Five-fold cross-validation accuracy and running times of our implementation

rounding down is the more likely the closer the input number is to the floor. See Catrina and Saxena [3] for more details.

Standard logistic regression uses the sigmoid function based on the exponential, and computing the loss requires the logarithm function. Our implementation of these is based on the code provided in SCALE-MAMBA [5] by Aly and Smart [1].

A particular helpful feature of MP-SPDZ in some protocols including the one used here is the implementation of dot products of fixed-point numbers with constant communication. See Dalskov et al. [6] for more details. Due to this optimization, we decided not to use mini-batches but the whole training set at once for simplicity. The benefit of the optimization increases with the size of the batch. Our implementation can be straightforwardly generalized to mini-batch training.

Given the decisions above, we ran our implementation with a few parameters on AWS `c5.9xlarge`, namely the precision of fixed-point truncation and the duration. For the former, there is a choice of probabilistic and exact truncation. Considering the latter, we saw that it takes about 100 epochs for the loss to get close to zero without using mini-batches. Therefore, we ran our implementation either for 100 or for 200 epochs, or until the loss was below 10^{-4}.

Table 2 shows the five-fold cross-validation accuracy and running time for each combination of parameters and dataset. Each values is averaged over 100 runs. Note that the running times are taken from the same run as the accuracies and therefore use only 80% of the respective dataset.

The implementation used for these timings is different to the submitted version in two points: The submitted version would only use a single thread because the evaluation criterion was changed from a

Sigmoid	Comparison	F_1	Time (sec)
3-piece	A	0.670 ± 0.064	0.94
	AB	0.672 ± 0.071	0.95
5-piece	A	0.673 ± 0.070	1.00
	AB	0.666 ± 0.071	0.96
Accurate	A	0.666 ± 0.073	2.83

Table 3: Improved results for GSE2034 with 100 epochs and probabilistic truncation

single host to several shortly before the deadline. Furthermore, the improved version uses the dot product optimization more consequently. We have seen that this reduces the running time by about one third.

Given the result, one would choose 200 epochs with probabilistic rounding for the best accuracy and variable duration with probabilistic rounding for faster inference at the cost of accuracy. However, we did not have time to run these evaluations before the deadline. From the limited information we had at the time, we decided to submit the training with 200 epochs and exact rounding. We achieved slightly better accuracy than logistic regression with plaintext computation, because a limited precision helps improve generalization on these two datasets.

4 Lessons Learned From Winning Solutions

Publications about two of three winning solutions [4, 11] broadly suggest two more optimizations compared to our submission, both of them inspired by Mohassel and Rindal [13]:

1. Approximate the sigmoid function by a piece-wise linear function. Both use the three-piece one by Mohassel and Rindal while Hong et al. [11] also consider a five-piece one.
2. The piece-wise approximation requires to compute comparisons, for which both switch from arithmetic to binary computation.

We have improved our solution accordingly and present results for the GSE2034 dataset in Table 3. We restrict ourselves to 100 epochs and probabilistic truncation because that setting represents a good trade-off in Table 2. We evaluated every option of sigmoid (three-piece, five-piece, and computed accurately) and purely arithmetic (A) vs. arithmetic-binary (AB) for piece-wise approximations. Note that timing with the accurate sigmoid function differs from the one in the previous section because we used various software optimizations for the results in this section.

We observed that both sigmoid approximations considerably improve the speed while keeping the accuracy intact. Furthermore, there is not much difference between either sigmoid approximation or whether or not binary computation is used for comparisons. This contradicts the observation by Hong et al., who found an improvement using the five-piece approximation. This might be the case because they use a domain-specific feature selection while we simply use all available features.

Mini-batches. Another difference to Hong et al.'s solution is that we do not use mini-batches but learn on the whole dataset at once. This is because the dataset is small and we are essentially training on large batches. However, running the first variant above with 15 epochs and mini-batches of size 32 takes 4.62 seconds while achieving an F_1 score of 0.644 ± 0.081. We estimate that the decrease in performance is due to the increased number of communication rounds.

References

[1] A. Aly and N. P. Smart. Benchmarking privacy preserving scientific operations. In R. H. Deng, V. Gauthier-Umaña, M. Ochoa, and M. Yung, editors, *ACNS 19*, volume 11464 of *LNCS*, pages 509–529. Springer, Heidelberg, June 2019.

[2] T. Araki, J. Furukawa, Y. Lindell, A. Nof, and K. Ohara. High-throughput semi-honest secure three-party computation with an honest majority. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *ACM CCS 2016*, pages 805–817. ACM Press, Oct. 2016.

[3] O. Catrina and A. Saxena. Secure computation with fixed-point numbers. In R. Sion, editor, *FC 2010*, volume 6052 of *LNCS*, pages 35–50. Springer, Heidelberg, Jan. 2010.

[4] M. D. Cock, R. Dowsley, A. C. A. Nascimento, D. Railsback, J. Shen, and A. Todoki. High performance logistic regression for privacy-preserving genome analysis. Cryptology ePrint Archive, Report 2020/171, 2020. `https://eprint.iacr.org/2020/171`.

[5] COSIC, KU Leuven. SCALE-MAMBA. `https://github.com/KULeuven-COSIC/SCALE-MAMBA`, 2019.

[6] A. Dalskov, D. Escudero, and M. Keller. Secure evaluation of quantized neural networks. Cryptology ePrint Archive, Report 2019/131, 2019. `https://eprint.iacr.org/2019/131`.

[7] Data61. MP-SPDZ. `https://github.com/data61/MP-SPDZ`, 2019.

[8] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323. PMLR, 2011.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[10] T. K. Ho. Random decision forests. In *International Conference on Document Analysis and Recognition*, pages 278–282, 1995.

[11] C. Hong, Z. Huang, W. jie Lu, H. Qu, L. Ma, M. Dahl, and J. Mancuso. Privacy-preserving collaborative machine learning on genomic data using TensorFlow, 2020. `https://arxiv.org/abs/2002.04344`.

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. `https://arxiv.org/abs/1412.6980`.

[13] P. Mohassel and P. Rindal. ABY^3: A mixed protocol framework for machine learning. In D. Lie, M. Mannan, M. Backes, and X. Wang, editors, *ACM CCS 2018*, pages 35–52. ACM Press, Oct. 2018.

[14] C. G. A. Network et al. Comprehensive molecular portraits of human breast tumours. *Nature*, 490(7418):61, 2012.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[17] Y. Wang, J. G. Klijn, Y. Zhang, A. M. Sieuwerts, M. P. Look, F. Yang, D. Talantov, M. Timmermans, M. E. Meijer-van Gelder, J. Yu, et al. Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer. *The Lancet*, 365(9460):671–679, 2005.