ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

Privacy-Preserving Kernel Computation For Vertically Partitioned Data

Mirko Polato, Alberto Gallinaro and Fabio Aiolli

University of Padova, Department of Mathematics, Italy

Abstract. In this paper, we propose a secure and privacy-preserving technique for computing dot-product kernels on vertically distributed data. Our proposal is based on secure multi-party computation which provides theoretical guarantees on both security and privacy. We also provide a practical application of the method by adapting a kernel-based collaborative filtering technique to the federated setting. An extensive experimental evaluation shows the effectiveness of the proposed approach.

1 Introduction

Federated learning [1, 2] (also known as collaborative learning) is a machine learning technique in which the training process is computed across multiple decentralized parties (e.g., devices or servers) holding private local data samples that are never exchanged with other parties. The general idea consists of training local models on the main local private data, and then sharing the parameters (e.g., the weights of a neural network) to generate a global model (usually on a centralized server). Thus, the learning is decentralized, but the global model stays on the server which has only a view of the model and not the data that generated it. It is important to underline that Federated learning differs from distributed learning in terms of the assumption made on the data. Distributed machine learning aims at parallelizing the computation, but the data are shared between the different parties. In this paper, we propose a secure and privacy-preserving method to compute dot-product kernels in the context of federated learning. The proposed method does not perform learning *per se*, but it can be injected in any federated kernel-based techniques.

Privacy-preserving computation of the kernel matrix has been studied almost exclusively in the context of Support Vector Machine (SVM). In [3], Rubinstein et al. proposed two variants of privacy-preserving SVM one that uses finite feature mapping (specifically, translation-invariant kernels), and one with infinite feature mapping (as in [4]). In the former case, the authors propose to add noise to the data for computing the kernel in a privacy-preserving manner. They also showed a bound of the empirical risk of adding such noise. In the latter case, privacy is achieved by using Raimi and Recht [5] random projection. However, both strategies are not suited for our purposes for two reasons: (*i*) data are highly sparse and adding noise work best with highly dense data as claimed in [6], and (*ii*) computing random projection (that is based on Fourier Transform) would require high computational power to the user. A similar idea based on random kernels has been proposed in [7] which seems only applicable to L1 SVM. In [6], a secure set intersection cardinality approach [8] is proposed. The method is

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence
and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

highly scalable and very easy to compute. Roughly speaking, users compute the entries of the linear kernel matrix using sets intersection that is made private via a particular scheme of one-way hashing functions applied in sequence by the users. The approach is really interesting, however, it fails anytime a user drops out.

Differently from the just mentioned techniques, in this paper we consider a scenario in which data are vertically distributed across the users, that is, each user owns a specific set of features of the training instances. We propose to compute the kernel using a protocol that is robust in case of dropping users. Besides the theoretical guarantees, we also show a collaborative filtering application adapting CF-KOMD [9], a kernel-based recommender system, to the federated setting.

2 Secure kernel computation

In this work we assume a (federated) scenario in which the data is vertically distributed, that is, each client owns a different set (possibly a single one) of features of the same data instances. This is pretty common, for example, in IoT systems, or more generally in sensors networks where different sensors/devices collect specific types of data (features) of the same overall set of observations. Another example is collaborative filtering based recommenders systems in which the data owned by a user are the ratings given (implicitly or explicitly) to the items and hence a user represents a single items' feature.

For simplicity, we stick with the assumption that each client/user has a single feature for all the instances, but the following considerations apply also with an arbitrary number of features. Let $\mathbf{X} \in \mathbb{N}^{n \times m}$ be m training instances with n integer features. Let $\mathbf{X}$ be vertically partitioned over a set of users $\mathcal{U}$ of cardinality n, and let $\mathbf{x}_u \in \mathbb{N}^{1 \times m}$ be the vector of features owned by the user u. Then, the linear kernel can be computed as

$$\mathbf{K} = \sum_{u \in \mathcal{U}} \mathbf{x}_u^\top \mathbf{x}_u \to \mathbf{K}_{ij} = \sum_{u \in \mathcal{U}} \left(\mathbf{x}_u^\top \mathbf{x}_u \right)_{ij} = \sum_{u \in \mathcal{U}} x_{ui} x_{uj}. \tag{1}$$

where $\mathbf{x}_u^\top \mathbf{x}_u \in \mathbb{N}^{m \times m}$ is the outer product. From (1), it is evident that the exact linear kernel can be computed as the sum over the outer product of the feature vectors of the users. Since we are in a federated setting, users can not directly share their contribution because it would mean sharing their private data. However, we can take advantage of the secure aggregation protocol [10] (SAP) to securely compute the sum $\sum_{u \in \mathcal{U}} \mathbf{x}_u^\top \mathbf{x}_u$.

In this protocol, we assume that all parties (i.e., server and clients) possess pair-wise secure communication channels with (relatively) ample bandwidth. Following SAP [10], we define the following steps: (*i*) **One-time pad masking**: this is the core step of the whole protocol which guarantees perfect privacy, but it fails in case of dropping users; (*ii*) **Dropped users recovery**: adds a recovery mechanism in case of dropping users; (*iii*) **Double masking**: adds a further level of security; (*iv*) **Communication speed improvement**: improves the communication's efficiency. In the following, we will detail only the first step (i.e., One-time pad masking) since the rest of the protocol is essentially the same as described in [10].

2.1 One-time pad masking

Using their secure channels, each pair of users first agree on a matched pair of perturbations. Specifically, every user u samples, for each other user v, an integer (sparse) matrix $\mathbf{S}_{uv} \in [0, R)^{m \times m}$, for some R, and exchange it with its counterpart $\mathbf{S}_{vu}$. Then, each user u computes the perturbation $\mathbf{P}_{uv} = \mathbf{S}_{uv} - \mathbf{S}_{vu}$, with $\mathbf{P}_{uu} = \mathbf{0}$, and noting that $\mathbf{P}_{uv} = -\mathbf{P}_{vu} (\text{mod } R)$. Once the perturbations are computed, each user sends to the server her contribution $\mathbf{Y}_u = \mathbf{x}_u^\top \mathbf{x}_u + \sum_{v \in \mathcal{U}} \mathbf{P}_{uv}$. Finally, the server sums up the perturbed user contributions to obtain the kernel

$$\mathbf{K} = \sum_{u \in \mathcal{U}} \left(\mathbf{x}_u^\top \mathbf{x}_u + \sum_{v \in \mathcal{U}} \mathbf{P}_{uv} \right) = \sum_{u \in \mathcal{U}} \left(\mathbf{x}_u^\top \mathbf{x}_u + \sum_{v \in \mathcal{U}} \mathbf{S}_{uv} - \mathbf{S}_{vu} \right) = \sum_{u \in \mathcal{U}} \mathbf{x}_u^\top \mathbf{x}_u (\text{mod } R).$$

The correctness is guaranteed since "opposite" perturbations cancel out. One-time pad masking guarantees perfect privacy for the users as well since the users' contributions are masked until the final aggregation. The secure aggregation protocol is also designed to handle the potential drop of users during the execution of the aggregation. The details are described in [10]. Nonetheless, if users drop during the aggregation, it means that the computed kernel is an approximation of the full kernel (some users/features are missing). However, as we will see in the experimental section, if a reasonable number of users participate in the computation, the kernel approximation is good enough to not harm the overall kernel machine effectiveness.

2.2 Computing dot-product kernels

Even though we showed how to secure aggregate only the linear kernel, this is enough to allow the computation of any dot-product kernel, i.e., a kernel that is a function of the dot-product between the inputs. As discussed by [11], under mild conditions, any dot-product kernel of the form $\kappa(\mathbf{x}, \mathbf{z}) = f(\langle \mathbf{x}, \mathbf{z} \rangle)$ can be seen as a dot-product polynomial, that is $\kappa(\mathbf{x}, \mathbf{z}) = \sum_{d=0}^{+\infty} a_d \langle \mathbf{x}, \mathbf{z} \rangle^d$ for some specific coefficients a_d. Thus, the server, starting from the linear kernel, can easily compute any dot-product polynomial. Dot-product kernel is a big family of kernels that contains many of the most used kernels, such as the RBF kernel, the polynomial kernel, and the boolean kernels [12].

3 FedCFK: Federated CF-KOMD

We now introduce the Federated CF-KOMD [9] algorithm, dubbed FedCFK, that extends CF-KOMD to the federated paradigm. CF-KOMD is a kernel-based recommender system that learns (independently for each user) a ranking over the items. The ranking is induced by the scores computed solving an optimization problem based on the kernel between the items that the user has rated. CF-KOMD fits our setting since users own their ratings (items' features) and the features (ratings) are binary, i.e., $\mathbf{x}_u \in \{0, 1\}^m$.

Let us divide the FedCFK learning process into two distinct phases: (1) kernel matrix computation; (2) users' optimization and scores computation. To compute the kernel (phase 1) in a distributed and privacy-preserving fashion, FedCFK employs the protocol described in Section 2. Phase 2 instead fits in the federated framework straightforwardly: the optimization can be independently computed *on device* by the clients without the need of sharing their ratings.

3.1 Server-side kernel computation

As previously mentioned, usually not all users participate in a single round of secure aggregation harming the quality of the approximated kernel. However, the server can continuously improve its kernel estimate. Specifically, after the secure kernel aggregation, the server computes $\hat{\mathbf{K}}$ where each entry $\hat{\mathbf{K}}_{ij}$ can be interpreted as (an estimate of) the number of users who rated both i and j. The server can improve the quality of $\hat{\mathbf{K}}$ w.r.t. the optimal $\mathbf{K}$ by keeping the maximum value of each entry $\hat{\mathbf{K}}_{ij}$ (that is a better estimate of the similarity between i and j) anytime a new secure kernel computation takes place. In the long run, these incremental improvements guarantee that the kernel $\hat{\mathbf{K}}$ becomes closer and closer to $\mathbf{K}$.

3.2 Kernel request-response

An important aspect that needs to be discussed is that each user only needs a portion (usually really small) of the overall kernel between items. Once the server has notified users that the kernel is ready, the users have to ask for the portion of the kernel needed to solve their optimization problem. For computing the items' score, a user only needs a "stripe" of the kernel corresponding to her positive items. However, the user cannot ask directly for those specific rows, otherwise, she would disclose her ratings to the server. To address this problem, a user asks for the rows corresponding to a bigger set of items. To avoid any possible leak of information, the user can also perform a noisy request by not including some of her positive items. The idea is to always ask for different sets of items and keeping track of the most up-to-date kernel's rows. In this way, the server can not infer any information about the users' ratings, and users can rely on a sufficiently updated kernel matrix.

4 Empirical evaluation

In this section, we assess the quality of FedCFK against its non-federated counterpart CF-KOMD. We evaluate the performance of the methods on MovieLens 100k (`ml-100k`[1]) and MovieLens 1M (`ml-1m`[2]). The performance of the methods

[1] `https://grouplens.org/datasets/movielens/100k/`

[2] `https://grouplens.org/datasets/movielens/1m/`

is measured using standard ranking metrics, namely AUC, normalized Discount Cumulative Gain (nDCG), and recall. For the last two metrics, we use a cutoff value of 100. In both datasets, the test set size is 10%.

The first set of experiments aims at showing the robustness of FedCFK when the kernel is computed on a single aggregation round. Table 1 and 2 shows the performance varying the number of users participating in the round. The reported results are an average of 10 repetitions.

Model	**% of users**	**AUC**	**nDCG@100**	**recall@100**
CF-KOMD	100	0.808	0.207	0.325
FedCFK	90	0.808	0.206	0.325
FedCFK	70	0.801	0.196	0.308
FedCFK	50	0.797	0.194	0.314
FedCFK	30	0.772	0.169	0.270

Table 1: Performance of FedCFK on `ml-100k`.

Model	**% of users**	**AUC**	**nDCG100**	**recall@100**
CF-KOMD	100	0.823	0.129	0.191
FedCFK	90	0.822	0.128	0.190
FedCFK	70	0.821	0.126	0.188
FedCFK	50	0.819	0.127	0.189
FedCFK	30	0.813	0.127	0.190

Table 2: Performance of FedCFK on `ml-1m`.

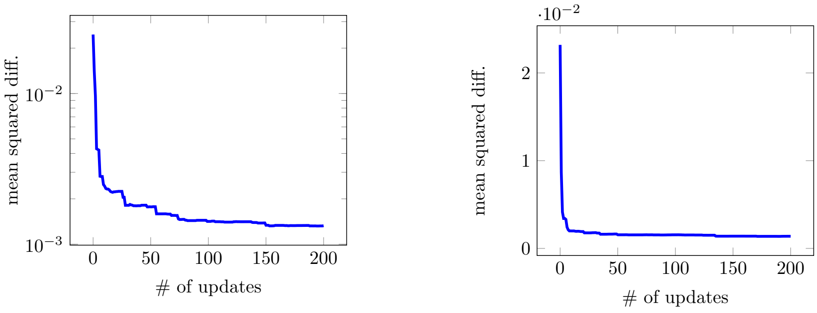
From the tables is evident that FedCFK is robust w.r.t. the number of users participating in the single round of the kernel computation. When users are less than 50% the performance sensibly decreases but still remains reasonably close to the one of CF-KOMD.

In the second set of experiments, we assess the quality of the server side kernel approximation in terms of the mean squared difference between the normalized version of $\mathbf{K}$ and $\hat{\mathbf{K}}$ computed over an increasing number of rounds in which a random set of users (with a size between 1% and 50%) participate.

The plots show that in 100 rounds the kernel approximations are good even if in each round less than 50% of the users participate.

5 Conclusions

We propose a secure and privacy-preserving method to compute dot-product kernels on vertically partitioned data. Our methodology is based on the secure aggregation protocol that provides theoretical guarantees for both security and privacy. One limitation of our proposal is the nature of the features that have to be integers. However, discretization techniques may help in mitigating this limitation.



(a) `ml-100k`.

(b) `ml-1m`.

Fig. 1: Normalized kernel approximation over 200 rounds.

References

[1] Jakub Konecny, H. Brendan McMahan, Felix X. Yu, Peter Richtarik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. In *NIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[2] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Agüera y Blaise Arcas. Communication-efficient learning of deep networks from decentralized data. *AISTATS*, pages 1273–1282, 2017.

[3] Benjamin I. P. Rubinstein, Peter L. Bartlett, Ling Huang, and Nina Taft. Learning in a large function space: Privacy-preserving mechanisms for svm learning. *Journal of Privacy and Confidentiality*, 4(1), Jul. 2012.

[4] Haoran Li, Li Xiong, Lucila Ohno-Machado, and Xiaoqian Jiang. Privacy preserving rbf kernel support vector machine. *BioMed Research International*, 2014, 2014.

[5] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, pages 1177–1184, Red Hook, NY, USA, 2007. Curran Associates Inc.

[6] Hwanjo Yu, Xiaoqian Jiang, and Jaideep Vaidya. Privacy-preserving svm using nonlinear kernels on horizontally partitioned data. In *Proceedings of the 2006 ACM Symposium on Applied Computing*, SAC '06, pages 603–610, New York, NY, USA, 2006. Association for Computing Machinery.

[7] Olvi L. Mangasarian and Edward W. Wild. *Privacy-Preserving Random Kernel Classification of Checkerboard Partitioned Data*, pages 375–387. Springer US, Boston, MA, 2010.

[8] Jaideep Vaidya and Chris Clifton. Secure set intersection cardinality with application to association rule mining. *J. Comput. Secur.*, 13(4):593–622, July 2005.

[9] Mirko Polato and Fabio Aiolli. Exploiting sparsity to build efficient kernel based collaborative filtering for top-n item recommendation. *Neurocomputing*, 268:17 – 26, 2017. Advances in artificial neural networks, machine learning and computational intelligence.

[10] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for federated learning on user-held data. In *NIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[11] Michele Donini and Fabio Aiolli. Learning deep kernels in the space of dot product polynomials. *Machine Learning*, pages 1–25, 2016.

[12] Mirko Polato and Fabio Aiolli. Boolean kernels for collaborative filtering in top-n item recommendation. *Neurocomputing*, 286:214 – 225, 2018.