VF-PS: How to Select Important participants in Vertical Federated Learning, Efficiently and Securely?

Seminar distributed learning systems

Presented at Neurips 2022, October 2022

Paper by

- Jiawei Jiang
- Lukas Burkhalter
- Fangcheng Fu
- Bolin Ding,Bo Du
- Anwar Hithnawi
- Bo Li
- Ce Zhang

Review by Maxime Geminiani Welcklen

Table of Contents

Table of figures

1. Analysis of the paper

The first section of this reviews starts by analyzing the paper itself. It does so by establishing the context of the paper and explains how VF-PS implements its solution.

1.1. Context

Federated learning (FL) is a machine learning paradigm that allows the training of a model over multiple stakeholders without exchange of training data. The strict compartmentalization of the data is at the core of FL: it allows secure training over sensitive data that would not be shared over untrusted participants. Federated learning is usually defined over two use cases: Vertical and horizontal.

Horizontal federated learning refers to a setup where the participants are usually numerous, with access to little computation power and each of them has a little sample of data. For example, training a text next-token predictor can be done on standard smartphones by using the texting history as private data. The learning is thus horizontal – it happens homogeneously over a large group of small contributors.

Vertical federated learning refers to a setup where the participants are fewer but hold a more important part of the data. In the case that is interesting for the paper, each participant has a part of the feature vector referring to the same entry list, and the user's labels are held at one of the participants. The learning is thus vertical: each of the few silos brings different information, or point of views, to the model.

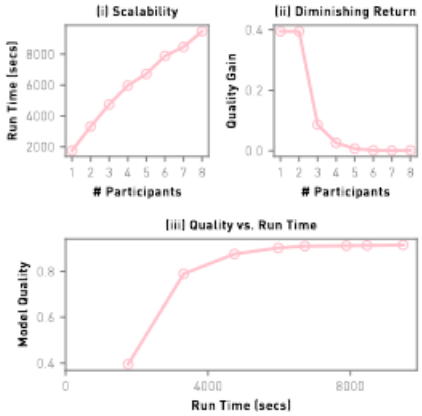

Figure 1 – Analysis of some properties of vertical federated learning

There are many challenges to vertical federated learning (VFL). An analysis of the training time over the number of participants in the Figure 1 shows that blindly adding more participants and taking all information into account increases the training time linearly with diminishing returns. This means that not all contributions are equals. As federated learning can involve a lot of data to train on, this begs the question: Would it be possible to apply data reduction and be more efficient ?

Data reduction would directly improve the scalability of VFL training. However, this improvement should not be done at the cost of accuracy or privacy.

1.2. VF-PS

The goal of the paper is to establish a protocol of participant selection for vertical federated learning tasks. This section describes the metric used to evaluate the relevance of a participant's contribution, then shows the process proposed to select the best N participants amongst them all, and finish by exposing the results of the process.

1.2.1. Principle

The statement that not all data are equal and that some samples are more significant than others is well established in the field of machine learning. It can be due to their inherent value/quality making a sample particularly impactful or to a mass of similar data brining the same information to the model. Therefore, data reduction has been widely studied for machine learning. For example, datasets can be pruned by using a simple 1-NN algorithm that can detect and remove samples that are too close to features already in the dataset.

However, in a setup of vertical learning where a few silos each hold a partial vector of the data, the data itself cannot be reduced horizontally at a sample level. VF-PS stands for Vertical learning – Participant selection because the data reduction is applied at the participant level. From a set of N participants, this process aims to select the l<<N best subset of participants that can train the model efficiently. In order to do that, the first step is to evaluate the relevance of a participant regarding the training of the model.

1.2.2. VF-Mine

VF-Mine stands for Vertical federated [learning] – Mutual Information Estimator. Mutual information is an already established metric of evaluation for the information contained in a dataset. It does so by using a K-NN algorithm detailed in the following section.

$$\max_{\alpha_1 \ldots . \alpha_l} MI([X_{\alpha_1} \ldots X_{\alpha_l}]; Y);\ s.t, \forall i \in [l] : \alpha_i \in [m] \text{ and } \forall i, j \in [l] : \alpha_i \neq \alpha_j. \qquad (1)$$

Figure 2 – Equation of the goal of VF-PS

The mutual information is computed over a set of data features and labels. It evaluates the correlation of the features with regards to their respective label. As each silo or participant holds a subset of features, the mutual information (MI) computed on a group of participants gives a good idea of their ability to solve the classification problem, and thus, their importance during the training of the model. The problem of participant selection based on MI can then be summarized as the equation in the Figure 2: The goal is to maximize the MI held by a subset of features contained in l participants – a subset of the original m participants.

It is thus important to understand the computation of the MI.

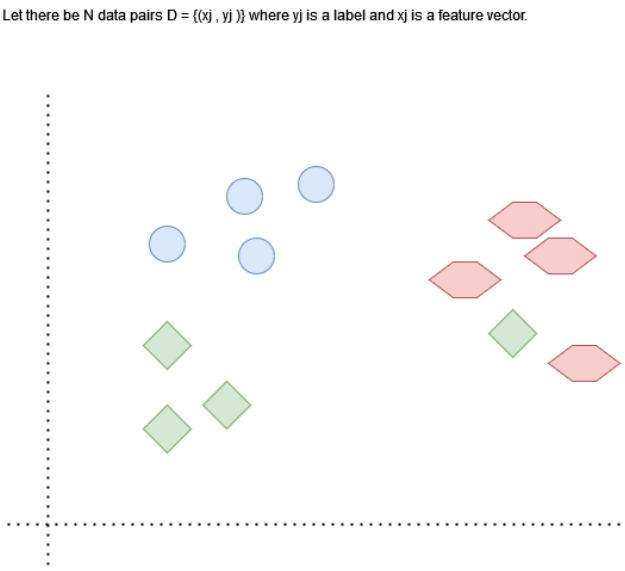

Figure 3 – First step of computing the MI over a dataset

Let a standard dataset of 2-dimensional features with three distinct classes as shown in the Figure 3. This represents the initial N data-pairs of the MI computation. This dataset has one rhombus whose features are located close to the hexagon class.

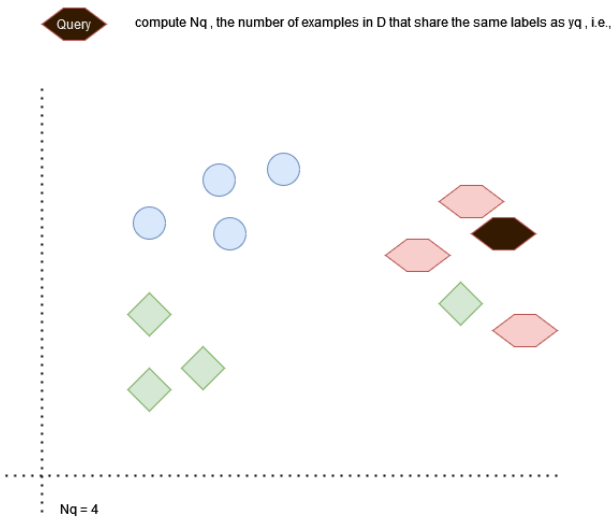

Figure 4 - Second step of computing the MI over a dataset

The MI is computed by selecting a number of queries: Data-pairs from the dataset that will be used as the basis of computation. Let's select one of the hexagons as the query and mark it black in the Figure 4.

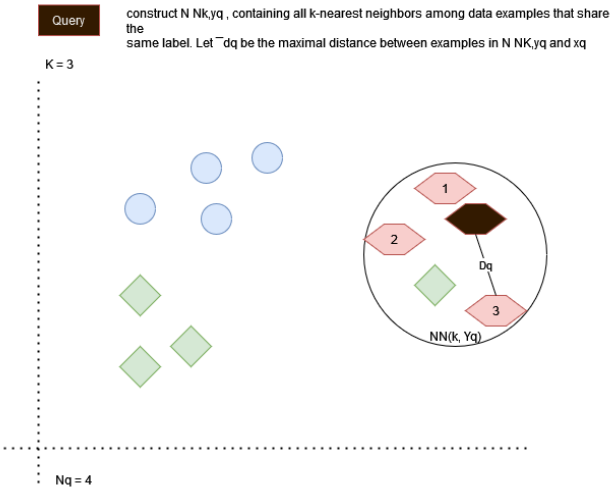

Figure 5 - Third step of computing the MI over a dataset

The next step is to count the amount of data with the same label as the query, this is Nq.

Let's fix the KNN's hyper-parameter at K=3. The k-neighborhood of the query considering only samples of the class Y_q is referred to as $NN_{k, Yq}$. From that group, the circle to the furthest same-class example can be drawn. The radius of this circle constitutes D_q as shown in the Figure 5.

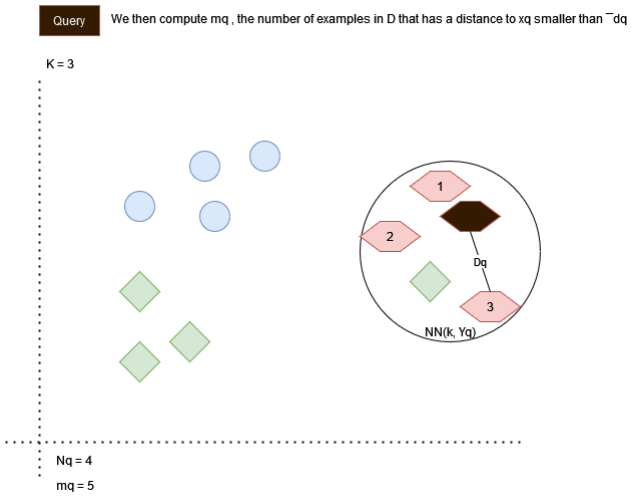

Figure 6 - Forth step of computing the MI over a dataset

Finally, the total amount of samples over the $NN_{k, Yq}$ neighborhood is counted. In the Figure 6, the 3-neighborood of the hexagon query includes a rhombus. This variable is denoted as M_q, with M_q=5.

lows: $\frac{1}{|Q|}\sum_q(\psi(N)-\psi(N_q)+\psi(K)-\psi(m_q))$, where ψ is the digamma function $\psi(x) = \frac{d}{dx}ln(\Gamma(x)) \sim lnx - \frac{1}{2x}$.

Figure 7 – Computation of the MI

All four variables required for the computation of the MI as shown in the equation of the Figure 7 are now known: The total dataset N, the cardinality of the class data N_q, the hyperparameter K, and the amount of total sample in the (k,y)-neighborhood M_q. The equation is the average over all queries of the sum of the digamma function of these four variables for each query.

The MI can be computed from a set of features and label and gives a good information over the correlation of the dataset features to their label – akin to silhouette computation regularly used to score unsupervised algorithms such as K-means. Although this is promising, the computation of the distances for the KNN is not a straightforward job: Each participant holds a subset of the features and these subsets are not to be shared across an aggregator party. However, some distances such as the Euclidian distance are very convenient as they can be computed on multiple disjointed instances, allowing each silo to compute a partial distance.

In order to preserve security over the network, the partial distances are exchanged as encrypted through Homomorphic encryption. This encryption allows arithmetic computation on encrypted data – allowing the aggregation of partial distances over encrypted messages. However, the aggregation of all the data samples coming from each participant is very costly. VF-PS proposes a way to further reduce the number of computations to obtain the neighborhood of the queries.

1.2.3. Overview of KNN Fagin-optimized

The idea behind the Fagin optimization is to see the problem of partial distances for the KNN's MI resolution as a partial ranking problem. There is a list of N data samples, each of them being represented

partially by each participant. The computation of the partial distances is done at the participant's level and result in |P| lists. The basic aggregation method could be to sum all the partial distances inside of an aggregator and to sort the resulting list, thus finding the top-k neighbors. However, as stated previously, this is costly and could hamper the scalability of the selection.

The Fagin algorithm [1] proposes that the full scan is not strictly required to obtain a final ranking composed of k sub-rankings. The principle is simple.

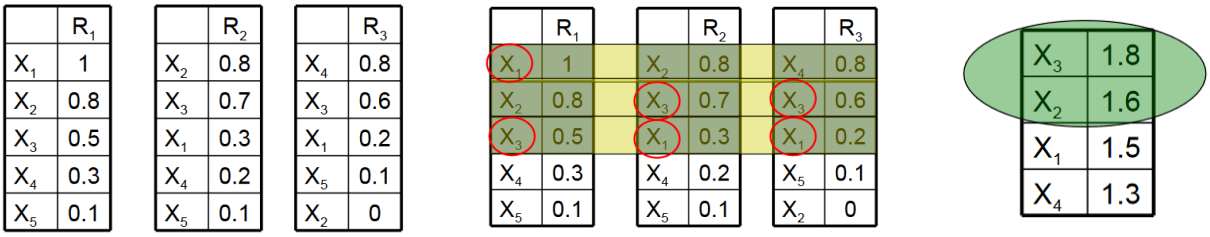

Figure 8 – multi-step Fagin algorithm to solve top-k rankings

Instead of iterating over each list and aggregating the results, the algorithm iterates over the indexes of each list. It stores the ID of the data it goes through. The scan stops once K distinct IDs have been met in all lists. All IDs that have been met at retrieved from the list and only those entries are aggregated. The resulting – shorter – list of distances can be sorted to obtain the top-k results.

The Figure 8 shows a run of the Fagin algorithm to find the top-2 best rank in three steps: First (a), there is a dataset containing five samples of data over three participants. Then, the algorithm scans each list index-by-index (b). The scan stops at index 2 as two distinct IDs have been met: X1 and X3. As this steps, IDs X1, X2, X3, and X4 have been met during the scans. Lastly, all the entries met during the scans are aggregated (c), resulting in the final sorted list. The top-2 can be extracted. This example shows that two optimizations took place: firstly, the scan only spanned three out of five indexes followed by a few constant accesses to retrieve all the values of the entries met. Secondly, only four values were aggregated, reducing the number of costly operations to run.

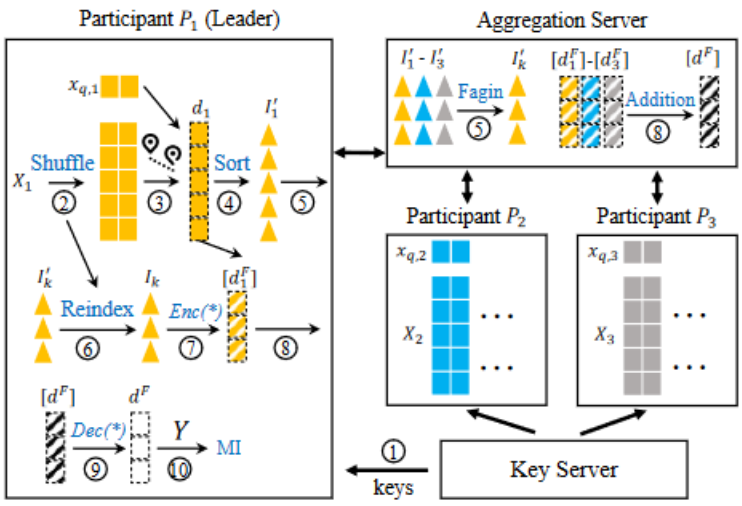

Figure 9 – Overview of the full MI run with Fagin optimization

The final process to compute an MI with the Fagin optimization is illustrated in the Figure 9. It shows a step-by-step run of the algorithm with one additional property: to ensure security, a shuffle takes places over the IDs.

In this setup, the key server is used to exchange public and private keys used for asymmetric encryption. The aggregation server is only used to perform the aggregation function of the KNN's MI algorithm. The participants each hold their sample of the data's features and one is elected the leader and holds the ground truth. Each of these stakeholders is partitioned such that no one should have the possibility of retrieving the data of another. However, the participants and the server are assumed to be "honest-but-curious": this threat model means that they will not run malicious code or disrupt processes, but if given the choice, will retrieve information. Because of that, the security model is made such that the key server delivers unique keys to each participant. The participants each compute their partial distances for the KNN algorithm and no feature is transmitted over the network. The bulk of the exchanges is composed of:

- The sub-rankings given by the participants to the aggregation server (5) holding the full dataset's ranking of closest distances for the Fagin computation
- The partial distances themselves (8) after the pre-selection of the Fagin algorithm

As every communication is secured, the most important security threat is the aggregation server: the only party that receives information directly from a participant. It is assumed and necessary that an IDs exists that binds the disjointed features together to form a coherent dataset. However, exchanging the partial distances to the aggregation server using the actual IDs of the dataset could open a breach of security from the server that could link the partial ranking of each participant to their features, and eventually retrieve/estimate said features with multiple runs. For this reason, the leader shuffle the IDs of the list according to a seed that is propagated through the participants only. Therefore, the partial distances are computed over ephemeral IDs that can be replaced once the Fagin algorithm is complete. The aggregation server computes the Fagin algorithm using the shuffled sub-ranking. This means that given it has no access to the seed (no server-participant collusion), it is impossible to make any conclusion regarding the original features of each participant. It then aggregates the distance over the reduced set of data that has been selected by the Fagin algorithm, once again unable to link the data to their original IDs.

The next challenge is to efficiently and securely find the best participants from their partial feature dataset.

1.2.4. Participant selection using batching

A naïve participant selection would be to create X groups of participants randomly and select the best performing out of them. Going to the extrema, the maximum number of possible subgroups of k participants amongst a total of N is bounded by the classical combinatory equation of $x = \binom{n}{k} = \frac{n!}{k!(n-k)!}$. As VF-PS is focused on making VFL scalable, having a complexity in the order of O(n!) participants for the selection algorithm is unacceptable.

To be more efficient, the VF-PS process select a collection of test groups containing random participants. Each of the groups has its MI evaluated via the VF-mine estimator. Each participant's score is then averaged over all its groups scores, giving it an importance score. The K most important participants are then selected – thus reducing the data used in the training.

To go deeper into optimization, the inclusion property of the Fagin algorithm states that the optimized top-k rankings obtained after the Fagin algorithm for any group will always be included in the top-k ranking obtained for the global dataset of all groups. Therefore, for a cardinality of G groups, the initial process would perform G times the algorithm. Instead of having this linear complexity, VF-PS proposes to perform the computation over the full group. This allows a "batching" of all the Fagin's computation for all the groups in one single operations, as each sub-ranking is derived from this global group.

1.2.5. Results

This section briefly shows the results of the VF-PS process applied to standard model trainings. The research team has used standard datasets used in the field to test this system. All the models used have undergone a simple hyper-parameter fine-tuning.

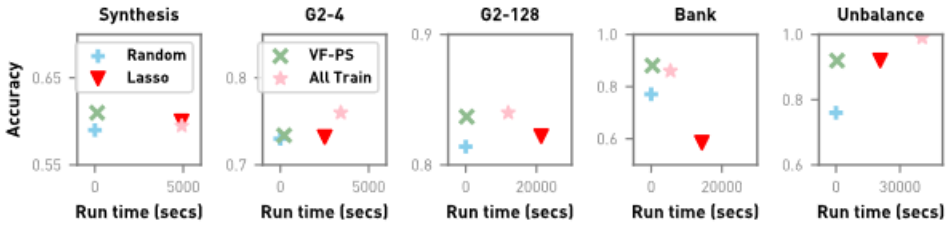

Figure 10 – Comparative results of different participant selection methods

The first test in the Figure 10 compares the accuracy as per the running time of four participant selection algorithms: Random, Lasso, VF-PS and all-train. The last one refers to a training over all participants without selection. The results are clear: Over the five examples, VF-PS performs as good or better than the two others participant selection. Moreover, it performs close or better to the all-train on most of the results, proving its efficiency in reducing the training time of the models.

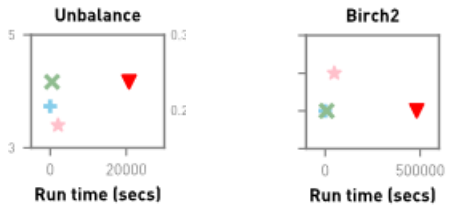

Figure 11 – Interesting comparative results

Some of the extensive results of the paper are interesting. For example, the Figure 11 shows that reducing the number of participants can result in a much higher accuracy (a). On the opposite, the accuracy can be lowered due to participant selection (b), hinting that some information can be lost in the selection process.

	Synthesis	G2-4	G2-128	Bank	Unbalance	Letter	Birch1	Birch2
VF-PS/50%	0.60	0.68	1.0	0.89	0.93	0.89	0.1	0.99
VF-PS/25%	0.59	0.61	1.0	0.8	-	0.83	-	-
Brute-force/50%	0.60	0.68	1.0	0.93	0.93	0.94	0.1	0.99
Brute-force/25%	0.59	0.63	1.0	0.93	-	0.86	-	-

Figure 12 – Comparative result of participant selection compared to brute-force

A further test is conducted and reported in the Figure 12 regarding the effectiveness of group selection. As discussed in the relevant section, group selection is optimized as brute-forcing all the possible group of participants has a factorial complexity over the number of participants due to combination. This test compares the final accuracy result of a training obtained with the group-based selection and the full brute-force selection. The results are very promising: Although some residual accuracy can be lost between the best possible selection (brute force testing all combination) and the efficient selection, the results are very close even for a 25% reduction. This trade-off seems acceptable considering the scalability of brute-force and proves the selection to be efficient and close to the best possible.

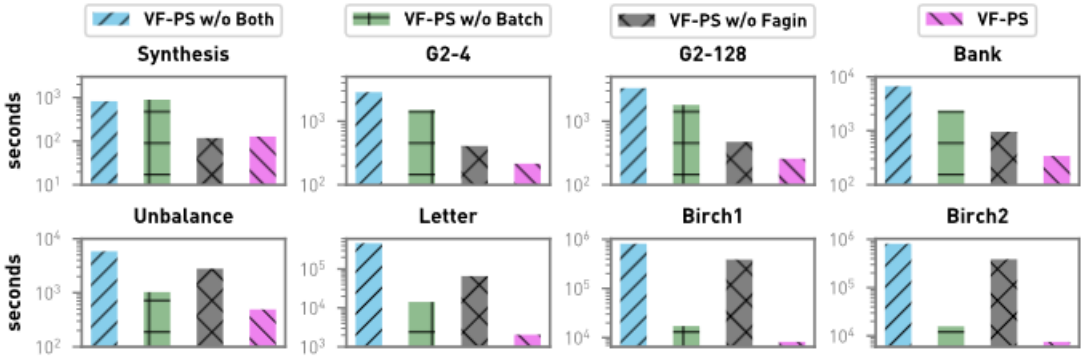

Figure 13 – Ablation study regarding Fagin and batching

The last test reported in this review in the Figure 13 shows an ablation study analyzing the impact of the batching and the Fagin optimization over the training time of VF-PS. In all examples, the VF-PS with both optimizations greatly outperforms either of the three other setups. Fagin steadily reduces training time

meanwhile batching has an uneven impact depending on the datasets. Overall, the combination of both is very impactful on the resulting time.

2. Critical discussion

This section proposes a point of view of the paper's strengths and weaknesses.

2.1. Strengths

As a research paper, VF-PS provides new ideas regarding vertical learning. This section highlights the quality or innovations of the paper.

2.1.1. Data reduction

VF-PS brings a novelty to the field of federated learning by proposing data reduction in the form of participant selection. However, the concept of data reduction itself is thoroughly studied since it applies to a lot of other machine learning algorithms. This means that successors or iterations of the algorithm can be inspired by the many techniques used in classical data reduction methods, opening a lot of room for innovation while picking from a steady and known pool of established ideas.

2.1.2. Security model

Federated learning is often centered around the concept of compartmentalization of data, with most threat models having one of the stakeholders (participant, aggregation server) try to retrieve the data of other participants. It could be expected that some research project based on a new algorithm can neglect this aspect, letting privacy be implemented on the fly by the deployment team.

The process shown in VF-PS has every computation on feature done locally on the respective participants. The message exchanged with the Fagin sub rankings and the partial distances are encrypted end-to-end and the aggregation itself is done on encrypted data. Lastly, the only party that has access to the partial data from participants and could try to estimate the single features from it, the aggregation server, only ever sees shuffled lists that cannot be ordered from an execution to another. This further ensures that it cannot reconstruct the features. This makes VF-PS a very secure algorithm respecting this core principle, bringing it closer to a deployable state out-of-the-box.

2.1.3. Extensive results

The choice of experiments and the result of said experiments is very well explained and displayed in the paper. For example, the results have been made on models with multiple runs taking hyperparameter fine-tuning into account. Short of a novel machine learning model paper, a lot of scheduling-related paper do use the accuracy of the model(s) as a result metric without making a search on hyper-parameter. However, fine-tuning the hyper-parameters is a staple of ML training and is therefore to be expected in real life conditions. Furthermore, the design of the experiments is sensible and well done. It covers all parts of the paper is the selection of participants better than the extremes (all-train and random); what is the training gain compared to the accuracy cost, is group selection better than brute force, ablation study to determine whether the batching and/or the Fagin are doing the heavy lifting with regards to the training time gain.

2.1.4. Optimizations

There are three optimizations that particularly shine. Without any major change or loss of accuracy, VF-PS can increase its efficiency by a large margin.

Firstly, it leverages the Fagin algorithm to treat the KNN neighborhood detection as a sub-ranking algorithm, leading to a gain in both scanning and aggregation computation. Secondly, it implements batching during the group selection to process every sub-group of participants Fagin / KNN algorithm at once by using the sub-ranking property. Finally, it implements a smart group selection that is way better than brute-forcing every combination of groups while being very close to the results of the best possible subgroup.

2.1.5. Rhetorical questions

Lot of research paper are very standardized in their writing style by design. VF-PS sparingly uses rhetorical questions at opportune times to convey ideas to the reader. Question such as :*"Can we identify $l \ll m$ participants that are most important, and only train expensive VFL models on them instead?"* are good at capturing the attention of the reader. This is a refreshing take that brings a little bit of interactivity to an activity that is otherwise very conventional and academic.

2.2. Weaknesses

Despite its qualities, this section tries to investigate possible weaknesses in the paper's implementation. The paper does a good and thorough job at listing its own shortcomings, for this reason, this following list can have this section of the paper in common.

2.2.1. Fairness

One of the most problematic aspects of VF-PS is fairness. Fairness is a property of selection algorithm that is important in some parts of machine learning such as recommender systems. Some selection algorithms, due to human or mathematically induced bias, will tend to exclude part of the database with little chance to be covered. However, that can be problematic depending on the problem that is being solved.

For example, let's take a VFL use case where multiple universities put fauna databases in common in order to train an advanced animal recognition model. If one of the participants has a very specific database with detailed features over an under-represented class, it could be excluded on the basis of a little importance to the overall accuracy of the model. However, in this specific example, ignoring a class could be very problematic: the training would ignore it as it would not have a great impact on the computed MI and the model would not be train with this quality data.

VF-PS's core mechanism is to exclude part of the participants from the training to reduce the data used and shorten the training time. For this reason, it seems like ensuring fairness will be a difficult challenge and VF-PS shall be employed only on models and datasets that do not have fairness requirements.

2.2.2. Accuracy loss

VF-PS's goal is to be used to shorten the training time of a federated model. This ensures a better scalability of vertical federated learning and lowers resource cost of training models. However, some of the examples given in the result report a considerable loss of accuracy after the participant selection. Some other examples report the same accuracy for 50% or 25% of participant after the algorithm.

If no participant has redundant information, then all participants are relevant to the training and the best group is the all-train. On the opposite, it seems to happen that a very few numbers of participants are enough to correctly label the data. Under these considerations, finding the optimal size of the subgroups seems like a hyper-parameter fine-tuning task as the optimal value can change depending on the task and

dataset. The loss of accuracy that can occur without this step can be a direct eliminatory defect for accuracy-critical modes.

2.2.3. Normalization

KNN is an algorithm that very often goes hand-in-hand with normalization. However, in this vertical structure, each partial distance is computed local on each participant without visibility to others. This could lead to unbalance of the distance weights and overall decreased accuracy. It would be interesting to think about a normalization method as a pre-processing via the aggregation server.

2.2.4. Devil is in the details

Some of the content of the algorithms are implied and very briefly explained. For example, the shuffle performed before sending the Fagin sub-ranking from participants to the aggregator server is mentioned but never justified. Although the first read gives a good overview of the paper mechanisms, further scrutiny brings subtle questions that require multiple attentive read to figure out. The details of the algorithm could be more fleshed out.

2.2.5. Dataset assumption

The paper assumes that the whole dataset is represented on each participant. However, some examples could be present in only a partial subgroup of the participants. The paper never addresses remedies to this problem, although they could bring difficult questions:

- would samples present in only a part of the dataset represent a problem with little partial distances computable ?
- Would they be excluded from the process altogether ?
- What would happen if every participant with information about a class are all excluded, meaning that the model has no access to a particular label features ?

3. Conclusion

The conclusion presents some avenues of improvement following the previously established strengths and weaknesses of VF-PS and finishes on a wrap-up and personal notes over the paper.

3.1. Avenues of improvement

This section proposes some improvements or interesting discussions about changes that could be beneficial to VF-PS.

3.1.1. Distance metrics

The KNN used in the MI computation and optimized by the Fagin's sub ranking is implemented with the Euclidian distance. The important property about the distance in this setup is that it must be computable in multiple places via partial distances and aggregated once at the end. There are multiple distances that correspond to this requirement and it would be interesting to analyze the effect of distances on training time, accuracy, and its combinatory effect with model / dataset. Distances is an interesting field studied in KNNs that could be ported into VF-PS.

3.1.2. Sub-ranking and aggregation

The sub-ranking merge into top-k selection is done by the Fagin algorithm with an additive aggregation function. However, top-k selection is a mechanism studied extensively in the scope of group-based recommender system, where each user has a personal ranking that has to be aggregated into a final top-

k list. It could be interesting to port that into VF-PS – for example, via trying other aggregation functions. Multiplication would be an obvious first choice, but min / max or a mix of them such as least misery could be interesting as long as they respect the monotonous requirement.

3.1.3. One-shot selection

Lastly, an interesting opportunity of extension that is acknowledged by the research team is the one-shot process of the participant selection. The premise of VF-PS is that some participants are more relevant / important for the classification task and others are not bringing as much information and can be ignored. However, the training of a model is not necessarily homogeneous: this importance could be dynamic and the participant selection could be done at multiple points of the training.

3.2. Final word

VF-PS is a paper that is nice to read as it involves the reader and explains the core of its components in an easy manner. It has elegant solutions that could be summarized as offering a known point of view, data reduction, to a field that did not consider it as is before, federated vertical learning. This brings a pool of useful studies and algorithms that can be carried out and adapted to the task. For example, the Fagin sub-ranking and the MI computations are not re-invented but simply applied to the problem, with the participants being selected instead of the data itself.

A lot of aspects such as privacy and security have been thoroughly planned and seem to hold under the selected thread model. Multiple optimizations for the sake of being more efficient are detailed over different steps of the algorithm. The overall process does not require any extravagant requirement and seem overall feasible. The experiments yield very encouraging results with little concerns overall. This gives the impression of a mature paper that has been well thought out.

4. References

[1] S. Kulhari, "Combining Fuzzy Information - Top-k Query Algorithms," [Online]. Available: http://alumni.cs.ucr.edu/~skulhari/Top-k-Query.pdf.