A Value-Preserving and Efficiency-Aware Split Learning Framework for Recommender System: Rebuttal

To Reviewer SKFD

Thanks very much for your precious time and careful review. Here we provide answers for some of the questions mentioned hoping to clarify your confusion.

Q1 Figure 2 is not well-presented, primarily due to the absence of a legend, which makes it difficult for readers to interpret the meaning of the individual shapes. For instance, what do the circles and rectangles signify? Do they represent users and items, respectively? Including a legend or additional annotations would greatly enhance the clarity and comprehensibility of the figure.

R1 The circles are the original input, and the rectangles are intermediate features processed by the DeepFM. Red inputs are continuous inputs, while the blue ones are categorical inputs. According to the original DeepFM, only the categorical inputs are involved in the FM part calculation. We will include a legend to better clarify our system.

Q2 Although the calculations for hp1, hp2 and hp3 in Figure 2 are described, their specific meanings remain unclear due to the lack of information about the input data for both the leader and follower. Providing concrete examples or case studies to illustrate the workflow of the entire model could significantly help readers better understand these representations and their roles within the framework.

R2 The final result of DeepFM can be split into a deep part and an FM part. Under our split learning multi-party paradigm, the output of the FM part (Eq.2) can be split into the sum of $y_1 = \sum_{p=1}^{q} w_{p0} + \sum_{p=1}^{q}\sum_{i=1}^{n_p} w_{pi}x_{pi} - \frac{1}{2}[\sum_{p=1}^{q}\sum_{j=1}^{k}\sum_{i=1}^{n_p} v_{pi,j}^2 x_{pi}^2]$and $y_2 = \frac{1}{2}\sum_{j=1}^{k}(\sum_{p=1}^{q}\sum_{i=1}^{n_p} v_{pi,j}x_{pi})^2$. Here $y_1$can be separately calculated by local input features by participants. However, $y_2$requires the early interaction of different parties. In a centralized setting, the input features are put together, here, each follower only sends its local sum$h_{p_2} = \sum_{i=1}^{n_p} v_{pi,j}x_{pi}$ to the leader. The leader aggregates all $h_{p_2}$and calculates the finaly_2 at the leader's end. The leader cannot infer the concrete value of input features from followers in h_{p_2}, which guarantees the secure split DeepFM modeling. $h_{p3}$is the intermediate feature of the deep part and can be jointly processed by the leader's top model (data protection of $h_{p_3}$is supported by VIB proposed in section 3.3).

Q3 In section 3.3.2, the authors use the VIB technique to prevent data leakage, from Fig. 3 and Eqs. 8 10, I understand that the authors use the encoder part of VAE to learn the variational posterior to make it approximate the prior, where this prior obeys the normal distribution and the KL dispersion is used as the regular term. What I don't understand is (1) why KL divergence prevents data leakage, it is well known that KL divergence measures the similarity between two distributions. (2) Also, why did the authors use only an encoder and not a decoder in the training process? (3) What is the difference between VIB and VAE?

R3 (1) The KL divergence is utilized in our framework to optimize the relationship between the learned variational posterior $q_\phi(h'_{p3}|x_F)$and the prior $p(h'_{p3})$. Data leakage refers to the scenario where sensitive information about the input x_F is encoded unintentionally in the intermediate feature representation h'_{p_3}. High mutual information $I(h'_{p_3}; x_F)$signifies that the representation carries significant information about the input data, potentially compromising privacy. By minimizing the KL divergence $D_{\mathrm{KL}}(q_\phi(h'_{p3}|x_F)||p(h'_{p3}))$, we encourage the learned distribution (the encoding of x_F through h'_{p_3}) to approximate the prior, which is assumed to be a standard normal distribution. This regularization means that rather than directly capturing sensitive information from data, the model (through h'_{p_3}) focuses on learning a distribution that is less dependent on the specific inputs, thereby reducing potential leakage of information about x_F.

(2) In the context of our variational information bottleneck framework, we focus primarily on the encoding process because our goal is to extract features that are maximally informative relevant to the labels while minimizing leakage from the input. The encoder maps the input x_F to a feature space h'_{p_3} where we can control the trade-off between task-relevance (through mutual information with labels) and privacy (through KL divergence with the prior). While VAE typically employs an encoder-decoder architecture for reconstructing data, our method simplifies to focus solely on the encoding process, as we are primarily interested in generating embeddings h'_{p_3} that fulfill our dual goals rather than reconstructing original input data. This design choice streamlines our approach while still preserving essential characteristics of the input, thus adhering to our privacy requirements.

(3) Variational Information Bottleneck (VIB) and Variational Autoencoder (VAE) share certain foundational principles, yet they are distinct in their goals and methodologies. The primary objective of VIB is to optimize the representation of data to retain relevant information for a specific task (e.g., predicting y_{label}) while constraining the information leakage about the input x_F. The mutual information terms guide this process. VAE, on the other hand, focuses on reconstruction. Its main goal is to learn a generative model of the data, where the embeddings are designed to allow for data reconstruction rather than concentrating on the embedding's relevance to a task or privacy constraints. While both frameworks utilize variational inference and KL divergence, VIB emphasizes controlling information flow for predictive tasks while VAE seeks to generate data points that resemble the training data.

In summary, we design our approach around utilizing KL divergence as a mechanism to enforce privacy in our representation while focusing on encoding with the aim of preserving task-relevant information. We chose to omit the decoder to streamline our process towards the desired outcomes of retaining useful embeddings without reconstructing input data, contrasting with the broader goals of VAE, which emphasize reconstruction and generative capacities.

Q4 In Table 1, the 'Split DeepFM' method did not achieve sota result, can the authors analyse and explain it?

R4 The initial and first purpose of our paper is to propose a cohesive split learning framework to enable the usage of more user or item-related features (not available or have serious safety concerns in a centralized setting) from different participants

for a potentially better overall recommendation performance (both training & inference).

FedAds [1] is a previous SOTA (2023) that pioneers the usage of features from multiple parts in a vertically split scenario. However, the model structure used in FedAds is naive with only fully connected layers. In this paper, we take a step forward and adapt DeepFM to such a multiple-party-involved, vertical federated setting.

The results in Table 1 of our paper are actually meant to support two conclusions:

(1) Using more input features (extra item-related features here) can indeed improve the performance of the recommendation model. Both results under split @ 2 parties are generally higher than centralized methods with fewer input features.

(2) Compared with FedAds's [1] naive structure, our split DeepFM can achieve better performance.

The reason we compare our split DeepFM with benchmarks like Wide&Deep, DeepFM, NFM, DCN, etc. is that the purpose of this comparison is to demonstrate that using more input features does help the RS performance. The comparison is only meaningful when the base model under the split paradigm (DNN in FedAds) and (DeepFM in our paper) is of a similar age or similar complexity. The author does believe that with a more complex model structure, the recommendation performance may be significantly better (AFM already excels in the 'Share' task with fewer features) than aged models like Wide&Deep or DeepFM, or even higher than FedAds or our split DeepFM with even more input features. However, this is not the core of our part of this experiment.

[1] FedAds: A Benchmark for Privacy-Preserving CVR Estimation with Vertical Federated Learning, https://arxiv.org/pdf/2305.08328