Low Rank Field-Weighted Factorization Machines for Low Latency Item Recommendation

Alex Shtoff
Michael Viderman
Naama Haramaty-Krasne
alex.shtoff@yahooinc.com
viderman@yahooinc.com
naamah@yahooinc.com
Yahoo Research

Oren Somekh
Ariel Raviv
Tularam Ban
orens@yahooinc.com
arielr@yahooinc.com
tularam@yahooinc.com
Yahoo Research

ABSTRACT

Factorization machine (FM) variants are widely used in recommendation systems that operate under strict throughput and latency requirements, such as online advertising systems. FMs have two prominent strengths. First, is their ability to model pairwise feature interactions while being resilient to data sparsity by learning factorized representations. Second, their computational graphs facilitate fast inference and training. Moreover, when items are ranked as a part of a query for each incoming user, these graphs facilitate computing the portion stemming from the user and context fields only once per query. Thus, the computational cost for each ranked item is proportional only to the number of fields that vary among the ranked items. Consequently, in terms of inference cost, the number of user or context fields is practically unlimited.

More advanced variants of FMs, such as field-aware and field-weighted FMs, provide better accuracy by learning a representation of field-wise interactions, but require computing all pairwise interaction terms explicitly. In particular, the computational cost during inference is proportional to the square of the number of fields, including user, context, and item. When the number of fields is large, this is prohibitive in systems with strict latency constraints, and imposes a limit on the number of user and context fields for a given computational budget. To mitigate this caveat, heuristic pruning of low intensity field interactions is commonly used to accelerate inference.

In this work we propose an alternative to the pruning heuristic in field-weighted FMs using a diagonal plus symmetric low-rank decomposition. Our technique reduces the computational cost of inference, by allowing it to be proportional to the number of item fields only. Using a set of experiments on real-world datasets, we show that aggressive rank reduction outperforms similarly aggressive pruning, both in terms of accuracy and item recommendation speed. Beyond computational complexity analysis, we corroborate our claim of faster inference experimentally, both via a synthetic test, and by having deployed our solution to a major online advertising system, where we observed significant ranking latency improvements. We made the code to reproduce the results on public datasets and synthetic tests available at https://anonymous.4open.science/r/pytorch-fm-0EC0.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

CCS CONCEPTS

• **Computing methodologies** → **Factorization methods**; • **Information systems** → *Online advertising*; • **Mathematics of computing** → **Computations on matrices**.

KEYWORDS

Recommender systems, Factorization machines, Low rank factorization

ACM Reference Format:
Alex Shtoff, Michael Viderman, Naama Haramaty-Krasne, Oren Somekh, Ariel Raviv, and Tularam Ban. 2018. Low Rank Field-Weighted Factorization Machines for Low Latency Item Recommendation. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Recommendation systems driven by machine-learned predictive models are widely used throughout the industry for a large variety of applications, from movie recommendation, to ad ranking in online advertising systems. In some applications, recommendation quality is the main objective, where sophisticated and computationally complex deep learning techniques are used to capture the affinity between the users and the recommended items. But other applications require striking an intricate balance between the accuracy of the predictive models, their training and inference speed. For example, real-time bidding systems in programmatic advertising are required to compute a ranking score for a large number of ads in a matter of a few milliseconds, and to train quickly in order to adapt to the ever-changing ad marketplace conditions. Such systems often deploy variants of the celebrated factorization machine (FM) models [29] in order to overcome data sparsity issues, while excelling at achieving a good balance between prediction accuracy and speed.

Rendle [29] also showed that while FMs model pairwise feature interactions, whose number is quadratic in the number of features, there is an equivalent formulation of FMs whose computational complexity is *linear* in the number features. This already facilitates fast training. Moreover, when ranking items for a given user in

a given context, the user and context features are the same for all items. The equivalent formulation allows caching the user and context computation results once, meaning that the computational cost per item is linear in the number of item features only. Therefore, factorization machines are also extremely efficient for ranking.

Many real-world applications involve large-scale multi-field data (e.g., gender and item category). However, FMs have a limited predictive accuracy, since they fail to capture the fact that the same feature can behave differently when interacting with features from different fields. To resolve this issue, many variants that incorporate field information have been proposed in recent years, including field-aware [15], field-weighted [25], and field-embedded [27, 34] factorization machines. Unfortunately, none of these variants admit an equivalent formulation whose computational complexity is linear in the number of features. This poses a challenge when building cost-effective, large-scale real-time recommendation systems, as the additional gain from incorporating field information comes at the cost of additional computing power.

The focus of this work is the *field-weighted factorization machine* (FwFM) variant [25], under the large-scale and low inference latency regime. It is similar to a regular factorization machine, with an additional symmetric matrix of parameters modeling the strength of pairwise field interactions. It is attractive in practice due to the fact that in terms of memory consumption and the number of parameters, it is on par with a regular FM. In addition, it is less prone to over-fitting, as pointed out by Juan et al. [15] and Pan et al. [25], and admits a simple heuristic for reducing the computational complexity at the inference stage by pruning low-magnitude field interactions. However, pruning has some cost in terms of accuracy. In this work, we devise a different way to significantly reduce the computational cost of field-weighted factorization machines by decomposition the matrix of pairwise field interactions. Motivated by the visualization of the field interaction matrices in [25], that resembles a block-like structure due to field groups exhibiting similar interaction behavior, we employ the well-known diagonal plus low-rank (DPLR) matrix decomposition, to produce an FwFM variant we call DPLR-FwFM. We show that our idea facilitates utilizing the additional accuracy provided by incorporating field information, while benefiting from a per-item computational cost that is linear in the number of item fields, albeit slower than regular FM by a factor proportional to the rank of the low-rank part of the decomposition.

We evaluate our technique on public datasets, and on proprietary data from a large-scale online advertising system of a major company. Our results demonstrate that, in practice, DPLR-FwFM with extremely low ranks outperform aggressively pruned FwFM models both in terms of latency and accuracy. . Finally, we deploy our solution in an online advertising system of a major company, and show that the latency incurred by a prediction module based on our DPLR-FwFM model is better than that of the module currently used in production and is based on a pruned FwFM.

To summarize, the main contributions of our paper are:

(1) Reformulate FwFM models to make their computational cost on par, or higher by a small constant factor of our choice, compared to regular FMs, and significantly cheaper than FwFMs. This, while benefiting from the higher accuracy provided by incorporating field information.
(2) Show that the accuracy of the obtained models is on par or higher than pruned FwFM models on public and proprietary data-sets.
(3) Demonstrate that, in practice, in a real-world online advertising system, our approach can significantly reduce the computational costs, thus achieving lower latency with a given computational budget, or reducing the computational power required to achieve a given latency.

2 RELATED WORK

Recommendation technologies such as *Collaborative Filtering* (CF) [11], help users discover new items based on past preferences. *Matrix Factorization* (MF) is a leading approach in CF, addressing data expansion and sparsity by using a latent factor model [18]. Such systems find applications in various domains, including movie recommendation [6], music recommendation [3], ad matching [2] and more. *Factorization machine* (FM) variants excel in benchmark tabular classification tasks, particularly when dealing with a large number of interactions and requiring fast predictions. This family includes several members as the original FM [30], *field-aware factorization machine* (FFM) [16], *field weighted factorization machine* (FwFM) [26] and the *field-matrixed factorization machine* (FmFM) [35]. For handling a wider range of features with non-linear interactions, [9] presented *Wide & Deep* learning approach, which trains a network that combines a linear model and a deep neural network. Their work was followed by a wave of deep learning techniques aimed at improving prediction accuracy, while also being sensitive to low-latency efficiency. An essential application that motivated such works, and also the focus of this work, is CTR prediction for online advertising. It has been the topic of many techniques as DeepFM [12], xDeepFM [19], DCN [37], AutoInt [33], DeepLight [10] and more. While these methods offer improved predictions, their high latency and long training time cause many applications to still prefer FM-based approaches.

Low-rank factorizations have emerged as a versatile and powerful tool in *machine learning* (ML) and *artificial intelligence* (AI), finding applications across a spectrum of domains. For instance, low-rank matrix-factorization in the context of recommendation systems [18, 30], in natural language processing for topic modeling, and dimensionality reduction [8], and more recently, for large-scale pre-training and fine-tuning of models on diverse natural language understanding tasks [14]. In cases where the involved matrices are full-ranked and low-rank factorizations are less effective, *diagonal plus low-rank factorization* (DPLR) may be considered (see the work of [31] and references therein). DPLR involves approximating a given matrix, often a large, and high-dimensional one, by decomposing it into the sum of two matrices: a diagonal matrix and a low-rank matrix. In statistics, DPLR is used to approximate high-dimensional covariance matrices of multivariate normal distributions [21, 24]. In ML, DPLR is used for enhancing the efficiency of models (see the work of [7] for an ML library that uses DPLR covariance matrices). For instance, DPLR-based methods have been employed in deep learning architectures for more efficient and accurate models [23, 36, 39]. Inspired by an observation of [20], that the regular FM field-interaction matrix equals a rank-one matrix

minus the identity matrix, we apply DPLR to FwFM and improve its training and inference efficiency.

One FM-like approach that resembles our work is of [5], where the authors apply a low-rank factorization to a generalized FM family representation (that includes FM, FFM, FwFM and FmFM as special cases). At first glance, their method seems to have much in common with our work, as both works use low-rank factorization to represent field interactions. However, there are two key differences. First, [5] takes a modeling approach to show that a family of FMs can be implemented as standard feed-forward networks. Our paper takes a different direction, that of recommendation systems: we focus on using a low-rank factorization to represent the separation into context and item fields, to improve item recommendation accuracy with FwFM models under strict latency constraints. Second, in contrast to [5], our work adds a diagonal component to the low-rank factorization, which plays a key role in the ability to be efficient for real-time systems, as shown in Section 4.1.

3 BACKGROUND AND PROBLEM FORMULATION

In this section, we provide a detailed explanation of the computational efficiency properties of FMs in recommending items, and the efficiency drop introduced by switching to field-aware variants. We then formulate the problem of reducing the gap between the two.

3.1 Efficiency of factorization machines

Given a feature vector $\boldsymbol{x} \in \mathbb{R}^n$, the FM model proposed by Rendle [29] computes

$$\Phi_{\mathrm{FM}}(\boldsymbol{x}; b_0, \boldsymbol{b}, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_n) = b_0 + \langle \boldsymbol{b}, \boldsymbol{x} \rangle + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle x_i \boldsymbol{w}_i, x_j \boldsymbol{w}_j \rangle,$$

where $b_0 \in \mathbb{R}$, $\boldsymbol{b} \in \mathbb{R}^n$, and $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n \in \mathbb{R}^k$ are trainable parameters. Overall, the model has $1 + n + nk$ trainable parameters, and the direct formula above for Φ_{FM} can be computed in $O(n^2 k)$ time. But as Rendle [29] pointed out, the pairwise interaction terms can be re-written as

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle x_i \boldsymbol{w}_i, x_j \boldsymbol{w}_j \rangle = \frac{1}{2} \left(\left\| \sum_{i=1}^{n} x_i \boldsymbol{w}_i \right\|^2 - \sum_{i=1}^{n} \left\| x_i \boldsymbol{w}_i \right\|^2 \right). \tag{1}$$

This dramatically reduces the time complexity to $O(nk)$.

In practice, the feature vector encodes a row in a tabular dataset of past interactions between users and items, whose columns, often named *fields*, contain categorical features. Some columns describe the context (including the user), whereas others describe the item. Thus, $\boldsymbol{x}$ formally consists of field-wise one-hot encodings, for example:

$$\boldsymbol{x} = (\underbrace{\underbrace{0,1,0,0,0}_{\text{field 1}}, \underbrace{1,0,0,0}_{\text{field 2}}, \ldots \underbrace{0,0,1,0,0,0}_{\text{field } m_c}}_{\text{context fields}}, \underbrace{\underbrace{0,0,1,0}_{\text{field } m_c+1}, \ldots \underbrace{0,0,0,0,1}_{\text{field } m}}_{\text{item fields}})^T,$$

Therefore, when computing Φ_{FM} we can only use the m nonzero entries of $\boldsymbol{x}$ corresponding to the m fields. We note that there may be fields having multiple values, such as a list of movie genres, but we defer their treatment to the discussion of FwFM models in the sequel.

Formally, for a given feature vector $\boldsymbol{x}$, let $\ell_1, \ldots, \ell_m$ denote the nonzero indices, and define $\boldsymbol{v}_i \equiv \boldsymbol{w}_{\ell_i}$. Hence, the formula in Equation (1) can be reformulated by summing over the fields, rather than the features. Moreover, we can split the summation over all fields to a summation over the context fields $C = \{1, \ldots, m_c\}$ and the item fields $\mathcal{I} = \{m_c + 1, \ldots, m\}$, and obtain:

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle x_i \boldsymbol{w}_i, x_j \boldsymbol{w}_j \rangle \tag{2a}$$

$$= \sum_{i=1}^{m} \sum_{j=i+1}^{m} \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle \tag{2b}$$

$$= \frac{1}{2} \left(\left\| \sum_{i=1}^{m} \boldsymbol{v}_i \right\|^2 - \sum_{i=1}^{m} \left\| \boldsymbol{v}_i \right\|^2 \right) \tag{2c}$$

$$= \frac{1}{2} \left(\left\| \sum_{i \in C} \boldsymbol{v}_i + \sum_{i \in \mathcal{I}} \boldsymbol{v}_i \right\|^2 - \sum_{i \in C} \left\| \boldsymbol{v}_i \right\|^2 - \sum_{i \in \mathcal{I}} \left\| \boldsymbol{v}_i \right\|^2 \right) \tag{2d}$$

By Equation (2d), we see that when ranking a large number of items for a given context, the sums over C can be computed only *once*. For each item, the computational complexity is $O(|\mathcal{I}|k)$, namely, and it depends largely on the number of *item fields*. Thus, in practice, we can use a model with a large number of user or context features to achieve a high degree of personalization, without incurring a significant computational cost during item ranking.

3.2 Inefficiency of field-weighted factorization machines

Field-weighted factorization machines (FwFM) [25] model the varying behavior of a feature belonging to some field when interacting with features from different fields in the form of a trainable symmetric field interaction matrix $\boldsymbol{R} \in \mathbb{R}^{m \times m}$. Its components $R_{i,j}$ model the *intensity* of the interaction between field i and field j. The model's output is

$$\Phi_{\mathrm{FwFM}}(\boldsymbol{x}; b_0, \boldsymbol{b}, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_n, \boldsymbol{R}) = b_0 + \langle \boldsymbol{b}, \boldsymbol{x} \rangle + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle x_i \boldsymbol{w}_i, x_j \boldsymbol{w}_j \rangle R_{f_i, f_j},$$

where f_i is the field corresponding to feature i. Pan et al. [25] demonstrate that this model family achieves a significantly higher accuracy compared to a regular FM, and is comparable with other field-aware variants. One attractive property of this variant is its number of parameters and memory requirements - it has only $\frac{m(m-1)}{2}$ additional parameters compares to a regular FM comprising the above-diagonal entries of the field interaction matrix $\boldsymbol{R}$.

Similarly to Equation (2b), under the one-hot encoding assumption, given an input $\boldsymbol{x}$, the pairwise interaction term can be written in terms of the field vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ that are a subset of the embedding vectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n$ that correspond to the m nonzero entries of $\boldsymbol{x}$:

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle x_i \boldsymbol{w}_i, x_j \boldsymbol{w}_j \rangle R_{f_i, f_j} = \sum_{i=1}^{m} \sum_{j=i+1}^{m} \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle R_{i,j} \tag{3}$$

However, the time complexity of computing its output Φ_{FwFM} is dominated by the $O(m^2 k)$ complexity of computing the pairwise

term in Equation (3), which is quadratic in the total number of fields. Compared to the $O(|\mathcal{I}|k)$ per item complexity of a regular FM, which is linear in the number of item fields, FwFMs pose a serious challenge for applications where inference speed is critical. While the quadratic complexity property is shared by many other field-aware variants, such as Juan et al. [15], Pande [27], Sun et al. [34], in this work, we focus on addressing this challenge for FwFM models.

The one-hot encoding assumption does not cover the case when a field has multiple values, such as a list of movie genres. In this case, we assume that multiple components of a field's encoding, that correspond to multiple values, may be nonzero. We handle this case by assuming that a field does not interact with itself, i.e. $R_{f,f} = 0$ for any field f. A direct consequence is that the pairwise interaction term can be written in terms of field vectors, as in Equation (3), but each vector v_i may be a weighted sum of the corresponding feature embedding vectors. For example, a movie with 3 genres may be encoded by placing $\frac{1}{3}$ in the corresponding components of $\boldsymbol{x}$, which will result in an average of the genre embedding vectors.

3.3 Problem formulation

A typical approach used in practice to speed up inference is pruning the matrix $\boldsymbol{R}$ by zeroing-out entries whose magnitude is below a threshold, as suggested by Pan et al. [25] and Sun et al. [34]. However, as we show in Section 5, aggressive pruning reduces the accuracy of the model. In this work, we aim to devise a method to achieve inference speed that is proportional to the number of *item fields*, and is only ρ times slower than a regular FM, where $0 < \rho \ll m$ is a configurable factor. This, while retaining an accuracy that is comparable with that of a regular FwFM model, and significantly better than pruned model with a similar number of parameters.

4 LOW-RANK FIELD-WEIGHTED FACTORIZATION MACHINES

In this section we reformulate the pairwise field interaction term in Equation (3) in an equivalent, but significantly more efficient manner. Throughout this section, we assume that the field vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m \in \mathbb{R}^k$ are embedded into the *rows* of the matrix $\boldsymbol{V} \in \mathbb{R}^{m \times k}$:

$$\boldsymbol{V} = \begin{bmatrix} - \boldsymbol{v}_1 - \\ - \boldsymbol{v}_2 - \\ \vdots \\ - \boldsymbol{v}_m - \end{bmatrix} \tag{4}$$

Moreover, since Equation (3) uses only the upper triangular part of $\boldsymbol{R}$, we may choose the remaining entries arbitrarily to our convenience, and throughout this section we assume that $\boldsymbol{R}$ is symmetric with a zero diagonal. Under this assumption, Equation (3) can be re-written as

$$\sum_{i=1}^{m} \sum_{j=i+1}^{m} \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle R_{i,j} = \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle R_{i,j}. \tag{5}$$

In the sequel, we describe our method to efficiently compute the double sum in the right-hand side of Equation (5). We first review the mathematical background, and then the complete method.

4.1 Mathematical foundation

In this section we present a technical result that provides the motivation for factorizing the matrix $\boldsymbol{R}$ to obtain an efficient algorithm. Then, we present an example explaining why the widely used *low-rank factorization* lacks expressive power, and use it to motivate a *diagonal plus low-rank factorization*.

The following identity let us reformulate the pairwise interaction formula as the one in Equation (5) in a matrix form, that is inspired by the reformulation in Lin et al. [20] for regular FMs.

Identity 1. *Let $\boldsymbol{A} \in \mathbb{R}^{m \times k}$. Then, for any matrix $\boldsymbol{Q} \in \mathbb{R}^{m \times m}$ we have*

$$\sum_{i=1}^{m} \sum_{j=1}^{m} \langle \boldsymbol{A}_{i,:}, \boldsymbol{A}_{j,:} \rangle Q_{i,j} = \mathrm{Tr}(\boldsymbol{A}^T \boldsymbol{Q} \boldsymbol{A}) \tag{6}$$

Proof. Recall the definition of the Frobenius inner product of matrices:

$$\langle \boldsymbol{A}, \boldsymbol{B} \rangle = \sum_{i=1}^{m} \sum_{j=1}^{n} A_{i,j} B_{i,j} = \mathrm{Tr}(\boldsymbol{A}^T \boldsymbol{B}),$$

and the circular shift invariance property of the trace operator [28, Section 1.1]:

$$\mathrm{Tr}(\boldsymbol{ABC}) = \mathrm{Tr}(\boldsymbol{BCA})$$

For any $1 \le i, j \le m$ it holds that $\langle \boldsymbol{A}_{i,:}, \boldsymbol{A}_{j,:} \rangle = (\boldsymbol{A}\boldsymbol{A}^T)_{i,j}$, and thus

$$\sum_{i=1}^{m} \sum_{j=1}^{m} \langle \boldsymbol{A}_{i,:}, \boldsymbol{A}_{j,:} \rangle Q_{i,j} = \sum_{i=1}^{m} \sum_{j=1}^{m} \left(\boldsymbol{A}\boldsymbol{A}^T \right)_{i,j} Q_{i,j} = \langle \boldsymbol{A}\boldsymbol{A}^T, \boldsymbol{Q} \rangle = \mathrm{Tr}(\boldsymbol{A}\boldsymbol{A}^T \boldsymbol{Q}),$$

where the last two equalities follow from the definition of the Frobenius inner product, and the fact that $\boldsymbol{A}\boldsymbol{A}^T$ is symmetric. Finally, the circular shift invariance property implies that $\mathrm{Tr}(\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{Q}) = \mathrm{Tr}(\boldsymbol{A}^T\boldsymbol{Q}\boldsymbol{A})$, completing the proof. □

Since $\boldsymbol{R}$ is a real symmetrical square matrix its eigenvalue decomposition is composed of real eigenvalues. If the eigenvalues decay quickly, we can use an approximation obtained by using the ρ largest magnitude eigenvalues, for some $0 < \rho \ll m$, in the form $\boldsymbol{R} = \boldsymbol{U}^T \mathrm{diag}(\boldsymbol{e}) \boldsymbol{U}$, where $\boldsymbol{U} \in \mathbb{R}^{\rho \times m}$ and $\boldsymbol{e} \in \mathbb{R}^{\rho}$.

However, a low-rank decomposition lacks the expressive power we need. To see why, consider a regular FM, which is equivalent to an FwFM where all field interactions are 1, meaning that the field-interaction matrix is:

$$\boldsymbol{R}_{\mathrm{FM}} = \begin{pmatrix} 0 & 1 & \ldots & 1 \\ 1 & 0 & \ldots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \ldots & 0 \end{pmatrix}$$

We have $\mathrm{rank}(\boldsymbol{R}_{\mathrm{FM}}) = m$, and its eigenvalues are $\{m-1, -1, \ldots, -1\}$, and hence a low-rank approximation within a reasonable accuracy is impossible. Since a low-rank decomposition does not even have enough expressive to model a regular FM, an FwFM is clearly out of reach. The obstacle stems from the zero diagonal of $\boldsymbol{R}$, which is

an "anomaly" in the matrix. But as Lin et al. [20] pointed out, this anomaly can be easily handled using a diagonal matrix:

$$\boldsymbol{R}_{\text{FM}} = \mathbf{1}\mathbf{1}^T - \boldsymbol{I}, \tag{7}$$

where $\mathbf{1}$ is a column vector whose components are all 1. In other words, the matrix $\boldsymbol{R}_{\text{FM}}$ can be decomposed into a sum of a diagonal matrix and a low-rank matrix. Motivated by the above, we use a diagonal plus low-rank (DPLR) decomposition to facilitate fast inference, as is shown in the proposition below. The low-rank part resembles the eigenvalue decomposition, to be able to model arbitrary symmetric matrices, including indefinite ones.

Proposition 1. *Let $\boldsymbol{V}$ be as defined in Equation (4), let $\boldsymbol{R} \in \mathbb{R}^{m\times m}$ be a symmetric matrix, and suppose that $\boldsymbol{U} \in \mathbb{R}^{\rho\times m}$, $\boldsymbol{e} \in \mathbb{R}^{\rho}$, and $\boldsymbol{d} \in \mathbb{R}^{m}$ are such that*

$$\boldsymbol{R} = \boldsymbol{U}^T \operatorname{diag}(\boldsymbol{e})\boldsymbol{U} + \operatorname{diag}(\boldsymbol{d}). \tag{8}$$

Define $\boldsymbol{P} = \boldsymbol{U}\boldsymbol{V}$. Then,

$$\sum_{i=1}^{m}\sum_{j=1}^{m}\langle \boldsymbol{v}_i, \boldsymbol{v}_j\rangle R_{i,j} = \sum_{i=1}^{m} d_i\|\boldsymbol{v}_i\|^2 + \sum_{i=1}^{\rho} e_i\|\boldsymbol{P}_{i,:}\|^2 \tag{9}$$

Proof. Invoking Identity 1 with $\boldsymbol{A} = \boldsymbol{V}$ and $\boldsymbol{Q} = \boldsymbol{R}$, and then using the decomposition in Equation (8) we have

$$\begin{aligned}\sum_{i=1}^{m}\sum_{j=1}^{m}\langle \boldsymbol{v}_i, \boldsymbol{v}_j\rangle R_{i,j} &= \operatorname{Tr}(\boldsymbol{V}^T\boldsymbol{R}\boldsymbol{V})\\ &= \operatorname{Tr}\left(\boldsymbol{V}^T(\operatorname{diag}(\boldsymbol{d}) + \boldsymbol{U}^T\cdot\operatorname{diag}(\boldsymbol{e})\cdot\boldsymbol{U})\boldsymbol{V}\right)\\ &= \operatorname{Tr}(\boldsymbol{V}^T\operatorname{diag}(\boldsymbol{d})\boldsymbol{V}) + \operatorname{Tr}(\underbrace{(\boldsymbol{U}\boldsymbol{V})^T}_{\boldsymbol{P}^T}\operatorname{diag}(\boldsymbol{e})\underbrace{(\boldsymbol{U}\boldsymbol{V})}_{\boldsymbol{P}}).\end{aligned}$$

Invoking Identity 1 with $\boldsymbol{A} = \boldsymbol{V}$ and $\boldsymbol{Q} = \operatorname{diag}(\boldsymbol{d})$, we obtain

$$\operatorname{Tr}(\boldsymbol{V}^T\operatorname{diag}(\boldsymbol{d})\boldsymbol{V}) = \sum_{i=1}^{m} d_i\|\boldsymbol{v}_i\|^2,$$

and again with $\boldsymbol{A} = \boldsymbol{P}$ and $\boldsymbol{Q} = \operatorname{diag}(\boldsymbol{e})$ we obtain

$$\operatorname{Tr}(\boldsymbol{P}^T\operatorname{diag}(\boldsymbol{e})\boldsymbol{P}) = \sum_{i=1}^{\rho} e_i\|\boldsymbol{P}_{i,:}\|^2.$$

Therefore,

$$\sum_{i=1}^{m}\sum_{j=1}^{m}\langle \boldsymbol{v}_i, \boldsymbol{v}_j\rangle R_{i,j} = \sum_{i=1}^{m} d_i\|\boldsymbol{v}_i\|^2 + \sum_{i=1}^{\rho} e_i\|\boldsymbol{P}_{i,:}\|^2$$

□

Now we are ready to present our solution based on Proposition 1.

4.2 The modified model and fast inference algorithm

Our solution consists of a modification of the FwFM model, and an algorithm that achieves inference during ranking in $O(\rho|\mathcal{I}|k)$ per item.

4.2.1 The diagonal plus low-rank FwFM model. Instead of learning the field interaction matrix $\boldsymbol{R}$ directly, we learn its decomposition in the diagonal plus low-rank form. Since $\operatorname{diag}(\boldsymbol{R}) = \mathbf{0}$, the diagonal component is fully determined by the low-rank decomposition. Formally, we replace the learned parameter $\boldsymbol{R}$, with the learned parameters $\boldsymbol{U} \in \mathbb{R}^{\rho\times m}$ and $\boldsymbol{e} \in \mathbb{R}^{\rho\times m}$, where ρ is a hyper-parameter. The matrix $\boldsymbol{R}$ is formally defined as

$$\boldsymbol{R} = \boldsymbol{U}^T\operatorname{diag}(\boldsymbol{e})\boldsymbol{U} + \operatorname{diag}(\boldsymbol{d}), \tag{10}$$

where

$$\boldsymbol{d} \equiv -\operatorname{diag\_of}(\boldsymbol{U}^T\operatorname{diag}(\boldsymbol{e})\boldsymbol{U}).$$

In other words, we ensure that $\boldsymbol{R}$ is defined to be a symmetric matrix with a zero diagonal by forming a symmetric eigenvalue-like decomposition, and subtracting the diagonal of the resulting matrix. The formal definition of $\boldsymbol{R}$ allows us to apply Proposition 1. This means that, we do not need to compute the matrix $\boldsymbol{R}$ itself at any stage. Instead, we replace the FwFM pairwise interaction term in Equation (3), with the outcome of the proposition in Equation (9). Namely, we compute $\boldsymbol{P} = \boldsymbol{U}\boldsymbol{V}$ in $O(\rho mk)$ time, and then compute $\frac{1}{2}\left(\sum_{i=1}^{m} d_i\|\boldsymbol{v}_i\|^2 + \sum_{i=1}^{\rho} e_i\|\boldsymbol{P}_{i,:}\|^2\right)$ in $O(\rho k + mk)$ time.

We call the resulting model a diagonal plus low-rank field weighted factorization machine, or DPLR-FwFM. Although this is not our main objective, a nice byproduct is that training also becomes slightly faster, since the above two steps cost $O(\rho mk)$ time instead the naïve $O(m^2k)$. This is important for some online advertising systems that require deploying a "fresh" model as soon as possible, to adapt to the quickly evolving marketplace conditions [22, 32, 38]. However, for inference during *ranking* we need to take the result of the proposition one step further.

4.2.2 Fast inference for item ranking. Observe that the field vectors embedded into the rows of $\boldsymbol{V}$ can be decomposed into context and item vectors:

$$\boldsymbol{V} = \begin{bmatrix}\boldsymbol{V}_C\\ \\ \boldsymbol{V}_{\mathcal{I}}\end{bmatrix}$$

The corresponding columns of $\boldsymbol{U}$ can be decomposed similarly:

$$\boldsymbol{U} = \begin{bmatrix}\boldsymbol{U}_C & \boldsymbol{U}_{\mathcal{I}}\end{bmatrix}$$

Consequently, matrix $\boldsymbol{P}$ can be written as:

$$\boldsymbol{P} = \boldsymbol{U}_C\boldsymbol{V}_C + \boldsymbol{U}_{\mathcal{I}}\boldsymbol{V}_{\mathcal{I}},$$

The product $\boldsymbol{U}_C\boldsymbol{V}_C$ can be computed only once when ranking with a given context. The sum $\sum_{i=1}^{m} d_i\|\boldsymbol{v}_i\|^2$ can also be similarly decomposed as

$$\sum_{i=1}^{m} d_i\|\boldsymbol{v}_i\|^2 = \sum_{i\in C} d_i\|\boldsymbol{v}_i\|^2 + \sum_{i\in\mathcal{I}} d_i\|\boldsymbol{v}_i\|^2,$$

where $\sum_{i\in C} d_i\|\boldsymbol{v}_i\|^2$ can be computed once when ranking with a given context. To summarize, during ranking, the pairwise interaction term of each item during ranking using DPLR-FwFM can be

computed by the following algorithm:

Algorithm 1

DPLR FwFM pairwise interaction inference with cached context

Input:

- The field matrix $\boldsymbol{V}$, partitioned into $\boldsymbol{V}_C, \boldsymbol{V}_I$
- $\boldsymbol{U} \in \mathbb{R}^{\rho \times m}, \boldsymbol{d} \in \mathbb{R}^m, \boldsymbol{e} \in \mathbb{R}^\rho$, that form the decomposition $\boldsymbol{R} = \boldsymbol{U}^T \operatorname{diag}(\boldsymbol{e})\boldsymbol{U} + \operatorname{diag}(\boldsymbol{d})$, with $\boldsymbol{U}$ partitioned into $\boldsymbol{U}_C$, $\boldsymbol{U}_I$

Output: The pairwise interactions $\sum_{i=1}^m \sum_{j=1}^m \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle R_{i,j}$

Steps:

(1) Once per context, compute:
  (a) Compute $\boldsymbol{P}_C = \boldsymbol{V}_C \boldsymbol{U}_C$
  (b) Compute $s_C = \sum_{i \in C} d_i \|\boldsymbol{v}_i\|^2$
(2) Compute $\boldsymbol{P} = \boldsymbol{P}_c + \boldsymbol{U}_I \boldsymbol{V}_I$
(3) Return $s_C + \sum_{i \in I} d_i \|\boldsymbol{v}_i\|^2 + \sum_{i=1}^\rho e_i \|\boldsymbol{P}_{i,:}\|^2$

The first step is computed only once per context in $O(\rho|C|k)$ time, and its results are cached. The last two steps are computed for every item, and their complexity is $O(\rho|I|k)$ per item, as we desire.

5 EXPERIMENTS

In this section we present experimental results demonstrating that our approach significantly reduces the cost of item recommendation serving without compromising accuracy, in comparison to the pruning approach. Finally, we demonstrate that finding a DPLR factorization of the field interaction matrix of a regular FwFM, instead of training a DPLR representation, may not be a good approach in practice.

5.1 Accuracy on public data-sets

We compare our approach to an FwFM, an FM, and a pruned FwFM using the Criteo Display Advertising Challenge[1] data-set, comprising of 39 fields and ~ 45M samples, the Avazu challenge dataset[2] comprising of 33 fields and ~ 40M samples, and the Movielens 1M [13] data-set, with 8 fields and ~ 1M samples. Out of the available MovieLens data-sets, we chose the one with the most number of informative context and item fields. The timestamp is converted into year, month, day of week, and hour of day, creating a 11 field data-set. The "genres" field in the MovieLens data-set has multiple values, and as described in Section 3.2, we average the genre embedding vectors to produce one genre vector.

All data-sets were randomly split into 80% training, 10% validation, and 10% test sets. The validation sets were used for tuning the learning rate using Optuna [4]. Features with less than 10 occurrences in the training set, and those appearing in the test and validation set that did not appear in the training set, are replaced with a special "rare feature". Numerical features are binned similarly to the strategy used by the Criteo Challenge winners [1], via the $x \to \lfloor \ln^2(x) \rfloor$ function. We tested embedding dimensions of 8 and 16.

We compare DPLR with pruning by matching the number of entries we retain in the pruned field interaction matrix to the number of parameters in the DPLR model. For a rank ρ, we use $\rho(m+1)$ parameters, and the corresponding pruned model is left with the largest magnitude $\rho(m+1)$ field interaction coefficients. Equivalently, we keep $100 \times \frac{2\rho(m+1)}{m(m-1)}$ percent of the field interactions.

The results are summarized in Table 1. We see that for aggressive pruning, retaining less than 20% of the interactions, the DPLR models out-perform, or perform on par, compared the pruned models. For MovieLens, due to a low number of fields, a DPLR model is equivalent to mild pruning, and still outperforms a pruned model.

5.2 Synthetic latency measurements

We use a simulation to compare inference time for a large batch of items for various amounts of context fields. We use the Criteo dataset as our example for the number of fields, and simulate vectors from 40 fields. We designate $k \in \{10, 15, 20, 25, 30\}$ of them as "context" fields, whereas the rest are item fields. Note, that since Criteo is anonymized, we have no way of knowing which fields are contextual. We measure the scoring time for DPLR models of ranks $\rho \in \{1, 2, 3\}$, and equivalently pruned models that have identical number of parameters by retaining $\rho(m+1)$ field interactions. The experimental code is implemented in Python, with the score computation parts implemented in Cython to eliminate Python runtime overhead. Moreover, to ensure CPU cache-friendliness of scoring using a pruned model, we fit the 3D array of latent vectors of an entire auction in an appropriate memory format.

The scoring times for various auction (batch) sizes are reported in Figure 1, together with scoring times using a regular FwFM. To measure standard error, the average time per auction for each configuration is measured by repeating every scoring experiment 50 times. To ensure precise measurement of each experiment due to timer resolution, each experiment comprises of 10 scored auctions. The experiments are conducted on a 2019 MacBook Pro laptop with a 2.4 GHz 8-Core Intel Core i9 CPU.

While there are caveats in synthetic timing benchmarks that stem from the implementation and the hardware used, the results allow appreciating the potential of our approach: indeed, scoring using a DPLR model is significantly faster than scoring by a pruned model. Since we made the code available, readers can also appreciate the implementation simplicity of the DPLR scoring algorithm - it relies on batched matrix-matrix products, whose efficient implementations are provided in a variety of numerical computing packages.

5.3 Proprietary online advertising system

We validate our approach on an online advertising system of a large commercial company, by comparing the trained model's accuracy, and the ad ranking latency. We perform our tests on FwFM models for *click-through rate* (CTR) prediction having 82 fields. To conform to the strict latency requirements, the FwFM's field interaction matrix is pruned to retain only 10% of the largest magnitude entries.

5.3.1 Accuracy experiment. The model trains periodically in a "sliding window" mode, meaning that in each period associated with time T, the model is trained on logged data from the month before T, and is evaluated on logged data from the day before T. After subsampling to allow training within a reasonable amount of time, the

[1] https://www.kaggle.com/c/criteo-display-ad-challenge
[2] https://www.kaggle.com/c/avazu-ctr-prediction

Table 1: Results for public data-sets. A lower LogLoss or MSE, and a higher AUC are better. The pruned sparsity column shows the percentage of entries in the pruned model equivalent to a DPLR model of the given rank, in terms of the number of parameters. The FM/FwFM columns show results without pruning or rank reduction, for reference, and are replicated across ranks. The next columns show results with pruning and rank reduction. Finally, the last column shows the improvement percentage of the DPLR over the equivalently pruned model.

Dataset (Metric)	Rank	Pruned sparsity	$k = 8$					$k = 16$				
			FM	FwFM	DPLR	Pruned	DPLR vs Pruned (%)	FM	FwFM	DPLR	Pruned	DPLR vs Pruned (%)
Criteo (AUC)	1	5.4%	0.8044	0.8088	0.8050	0.8008	0.52%	0.8069	0.8100	0.8069	0.8020	0.61%
	2	10.8%	0.8044	0.8088	0.8066	0.8049	0.21%	0.8069	0.8100	0.8080	0.8057	0.29%
	3	16.2%	0.8044	0.8088	0.8067	0.8065	0.02%	0.8069	0.8100	0.8083	0.8076	0.09%
	4	21.6%	0.8044	0.8088	0.8069	0.8074	-0.05%	0.8069	0.8100	0.8085	0.8086	-0.01%
	5	27%	0.8044	0.8088	0.8070	0.8078	-0.10%	0.8069	0.8100	0.8085	0.8091	-0.07%
Criteo (LogLoss)	1	5.4%	0.4470	0.4429	0.4467	0.4510	0.97%	0.4449	0.4417	0.4449	0.4508	1.30%
	2	10.8%	0.4470	0.4429	0.4454	0.4469	0.34%	0.4449	0.4417	0.4437	0.4464	0.60%
	3	16.2%	0.4470	0.4429	0.4450	0.4451	0.01%	0.4449	0.4417	0.4435	0.4444	0.21%
	4	21.6%	0.4470	0.4429	0.4443	0.4443	0.00%	0.4449	0.4417	0.4431	0.4431	-0.00%
	5	27%	0.4470	0.4429	0.4443	0.4439	-0.10%	0.4449	0.4417	0.4430	0.4426	-0.08%
Avazu (AUC)	1	6.4%	0.7768	0.7777	0.7764	0.7727	0.49%	0.7787	0.7796	0.7776	0.7738	0.49%
	2	12.9%	0.7768	0.7777	0.7769	0.7756	0.16%	0.7787	0.7796	0.7784	0.7779	0.05%
	3	19.3%	0.7768	0.7777	0.7772	0.7764	0.10%	0.7787	0.7796	0.7789	0.7789	0.00%
	4	25.8%	0.7768	0.7777	0.7772	0.7770	0.03%	0.7787	0.7796	0.7789	0.7794	-0.06%
	5	32.2%	0.7768	0.7777	0.7774	0.7774	-0.00%	0.7787	0.7796	0.7788	0.7796	-0.10%
Avazu (LogLoss)	1	6.4%	0.3811	0.3808	0.3817	0.3841	0.62%	0.3800	0.3795	0.3810	0.3830	0.52%
	2	12.9%	0.3811	0.3808	0.3812	0.3817	0.11%	0.3800	0.3795	0.3801	0.3802	0.02%
	3	19.3%	0.3811	0.3808	0.3812	0.3814	0.07%	0.3800	0.3795	0.3801	0.3800	-0.02%
	4	25.8%	0.3811	0.3808	0.3810	0.3811	0.01%	0.3800	0.3795	0.3800	0.3796	-0.12%
	5	32.2%	0.3811	0.3808	0.3811	0.3810	-0.05%	0.3800	0.3795	0.3800	0.3796	-0.11%
Movielens (MSE)	1	33.1%	0.7431	0.7407	0.7449	0.7572	1.61%	0.7376	0.7394	0.7445	0.7630	2.43%
	2	64.3%	0.7431	0.7407	0.7418	0.7464	0.62%	0.7376	0.7394	0.7398	0.7484	1.15%

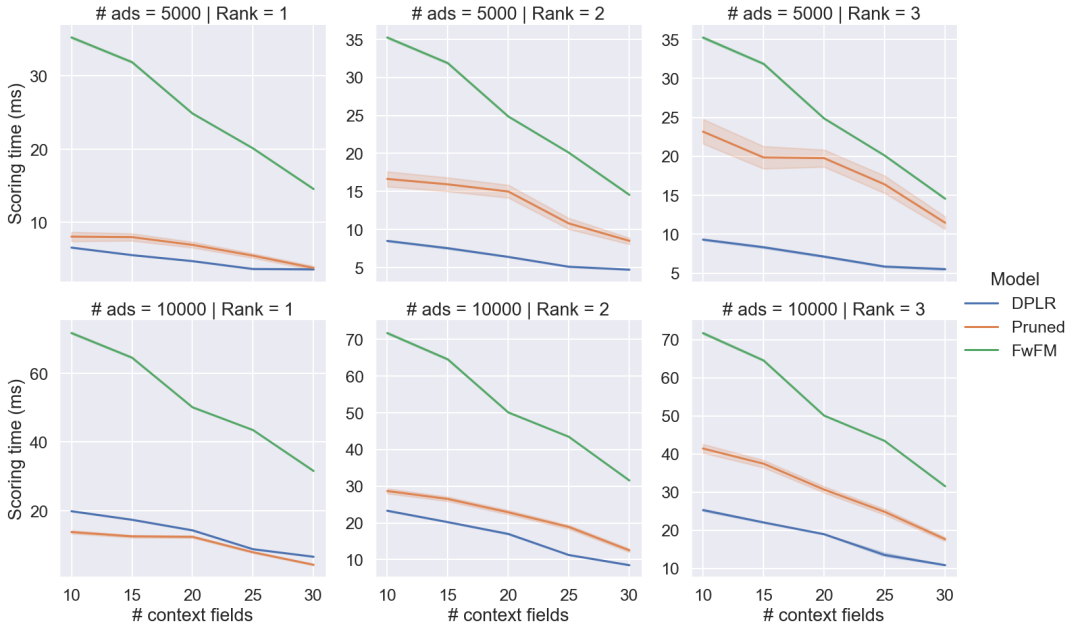


Figure 1: Synthetic timing measurement of a single auction for various auction sizes, DPLR model ranks, and amounts of context fields. Standard error is plotted as a narrow band around each line.

training set comprises tens of millions logged user-ad interactions. We train and evaluate the model on 7 consecutive intervals using the LogLoss and AUC metrics, and average the results by weighting them according to the number of evaluation samples. The trained DPLR model is compared to an FwFM pruned to 10% of the entries,

due to various restrictions of the production code-base. The performance of the models, and the improvement of DPLR over pruned models, are summarized in Table 2 Surprisingly, the DPLR models perform *better* on the evaluation set than a regular FwFM models. We believe it means that a low-rank field interaction matrix is a good regularization prior for this specific domain. Nevertheless, as our results on public data-sets show, this is not the case in general.

Table 2: AUC and LogLoss improvements versus an FwFM on proprietary data. Higher is better.

Metric	Rank					
	1	2	3	4	5	6
LogLoss lift	-0.48%	0.23%	0.36%	0.41%	0.44%	0.404%
AUC lift	-0.13%	0.07%	0.10%	0.11%	0.12%	0.115%

5.3.2 Latency experiment. We deploy the model with rank 3 to a test environment that serves a small portion of the traffic, a few thousand ad ranking queries per minute. The rank was chosen to correspond to the pruning of 90% of the field interaction entries in terms of the number of parameters. Out of the model's 63 fields, 38 are item fields, and therefore we can theoretically expect approximately 40% latency inference speed improvement in the best case.

We measure the latency incurred by the inference for ad CTR prediction, and the total latency incurred by ranking all eligible ads for a given query. Table 3 summarizes the results, and appendix A presents full time-series plots. Evidently, the inference latency is improved by 20% – 30%, whereas the query latency by 5%, as CTR prediction is only one component of the ad query serving algorithm.

Table 3: Average and high percentile latency of a single inference or ranking operation in our production system. In each minute, the average, 95th percentile, and the 99th percentile latency for inference, and the 95th percentile latency for ranking. The measurements were averaged over a 10 hour period. The lifts represent improvements - higher is better.

	Inference per ad			Ranking
	Average	P95	P99	P95
Lift (%)	34.27%	29.11%	25.57%	5.45%

5.4 Post-hoc factorization of the field interaction matrix

An alternative approach to training a DPLR representation of the field interaction matrix can be training a regular FwFM model, and computing the best DPLR representation we can afterwards. We call this the "post-hoc" approach.

Unfortunately, this may not be a good idea. Suppose we obtained an approximation $\tilde{\boldsymbol{R}}$ of the model's field interaction matrix $\boldsymbol{R}$. Let $\tilde{\boldsymbol{R}} = \boldsymbol{R} + \boldsymbol{E}$, where $\boldsymbol{E}$ is the approximation error. Denote by $\lambda_i(\cdot)$ and $\sigma_i(\cdot)$ the i^{th} largest eigenvalue and singular value of a given matrix, respectively. By Identity 1 we obtain

$$\begin{aligned}\sum_{i=1}^{m}\sum_{j=1}^{m}\langle \boldsymbol{v}_i, \boldsymbol{v}_j\rangle \tilde{R}_{i,j} &= \mathrm{Tr}(\boldsymbol{V}^T\tilde{\boldsymbol{R}}\boldsymbol{V}) \\ &= \mathrm{Tr}(\boldsymbol{V}^T\boldsymbol{R}\boldsymbol{V}) + \mathrm{Tr}(\boldsymbol{V}^T\boldsymbol{E}\boldsymbol{V}) \\ &\le \mathrm{Tr}(\boldsymbol{V}^T\boldsymbol{R}\boldsymbol{V}) + \sum_{i=1}^{m}\lambda_i(\boldsymbol{V}\boldsymbol{V}^T)\sigma_i(\boldsymbol{E}),\end{aligned}$$

where the last inequality stems from the Von Neumann trace inequality. Thus, the approximation error boils down to the singular value spectrum of the error matrix. We took the field interaction weights obtained from an FwFM trained on the Criteo data-set, and computed the singular value spectrum of the approximation errors obtained from two approximations: (a) a DPLR approximation of rank 5, computed by minimizing the nuclear norm of the error, and (b) a pruned field interaction matrix where the top 200 entries (same number of parameters as the rank-5 DPLR approximation) were left. The error singular-value spectrum of both approximations is plotted in Figure 2. We observe that the large eigenvalues of the post-hoc DPLR approximation error are *much* larger than the ones of the pruned approximation error. Of course, an upper bound cannot replace a true experimental result on the given dataset, but it gives a rough idea of weather this direction is worth pursuing.

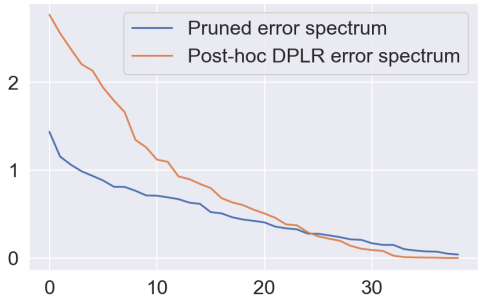


Figure 2: Singular value spectrum of two approximations of the true field interaction matrix. The singular value index is in the horizontal axis, whereas its value is in the verical axis.

6 CONCLUSIONS AND FUTURE WORK

In this work we proposed learning a diagonal plus low-rank decomposition of the field interaction matrix of FwFM models as an alternative to the commonly used pruning heuristic in large scale low-latency recommendation systems. We demonstrated that our approach has the potential to outperform pruned models in in terms of both item recommendation speed and model accuracy.

Looking at Equation (9), we observe that the columns of $\boldsymbol{U}$ have a notion of "field importance" - if all the entries of $\boldsymbol{U}_{:,i}$ are negligible, the effect of the i-th field is negligible, and it can be discarded. Hence, another future direction is mathematically or experimentally quantifying this notion of field importance, as an alternative to the approach in [17]. This is especially important for real-time systems, where just reducing the number of fields may have a tremendous effect on latency and recommendation costs.

REFERENCES

[1] 2014. 3 Idiots' Approach for Display Advertising Challenge. https://github.com/ycjuan/kaggle-2014-criteo.git. Accessed: 2024-04-01.

[2] Michal Aharon, Natalie Aizenberg, Edward Bortnikov, Ronny Lempel, Roi Adadi, Tomer Benyamini, Liron Levin, Ran Roth, and Ohad Serfaty. 2013. OFF-set: one-pass factorization of feature sets for online recommendation in persistent cold start settings. In *RecSys'2013*. 375–378.

[3] Natalie Aizenberg, Yehuda Koren, and Oren Somekh. 2012. Build your own music recommender by modeling internet radio streams. In *Proceedings of the 21st international conference on World Wide Web*. ACM, 1–10.

[4] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[5] Chen Almagor and Yedid Hoshen. 2022. You Say Factorization Machine, I Say Neural Network-It's All in the Activation. In *Proceedings of the 16th ACM Conference on Recommender Systems*. 389–398.

[6] Robert M Bell and Yehuda Koren. 2007. Lessons from the Netflix prize challenge. *Acm Sigkdd Explorations Newsletter* 9, 2 (2007), 75–79.

[7] Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul A. Szerlip, Paul Horsfall, and Noah D. Goodman. 2019. Pyro: Deep Universal Probabilistic Programming. *J. Mach. Learn. Res.* 20 (2019), 28:1–28:6. http://jmlr.org/papers/v20/18-403.html

[8] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3, Jan (2003), 993–1022.

[9] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.

[10] Wei Deng, Junwei Pan, Tian Zhou, Deguang Kong, Aaron Flores, and Guang Lin. 2021. Deeplight: Deep lightweight feature interactions for accelerating ctr predictions in ad serving. In *Proceedings of the 14th ACM international conference on Web search and data mining*. 922–930.

[11] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. 1992. Using collaborative filtering to weave an information tapestry. *Commun. ACM* 35, 12 (1992), 61–70.

[12] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

[13] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.

[14] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021).

[15] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-Aware Factorization Machines for CTR Prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems* (Boston, Massachusetts, USA) *(RecSys '16)*. Association for Computing Machinery, New York, NY, USA, 43–50. https://doi.org/10.1145/2959100.2959134

[16] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware factorization machines for CTR prediction. In *Proceedings of the 10th ACM conference on recommender systems*. 43–50.

[17] Yohay Kaplan, Yair Koren, Rina Leibovits, and Oren Somekh. 2021. Dynamic length factorization machines for CTR prediction. In *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 1950–1959.

[18] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[19] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1754–1763.

[20] Xiao Lin, Wenpeng Zhang, Min Zhang, Wenwu Zhu, Jian Pei, Peilin Zhao, and Junzhou Huang. 2018. Online Compact Convexified Factorization Machine. In *Proceedings of the 2018 World Wide Web Conference* (Lyon, France) *(WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1633–1642. https://doi.org/10.1145/3178876.3186075

[21] Antoine Liutkus and Kazuyoshi Yoshii. 2017. A diagonal plus low-rank covariance model for computationally efficient source separation. In *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 1–6.

[22] Andreas Lommatzsch, Benjamin Kille, and Sahin Albayrak. 2017. Incorporating context and trends in news recommender systems. In *Proceedings of the international conference on web intelligence*. 1062–1068.

[23] Aaron Mishkin, Frederik Kunstner, Didrik Nielsen, Mark Schmidt, and Mohammad Emtiyaz Khan. 2018. Slang: Fast structured covariance approximations for bayesian deep learning with natural gradient. *Advances in Neural Information Processing Systems* 31 (2018).

[24] Victor M-H Ong, David J Nott, and Michael S Smith. 2018. Gaussian variational approximation with a factor covariance structure. *Journal of Computational and Graphical Statistics* 27, 3 (2018), 465–478.

[25] Junwei Pan, Jian Xu, Alfonso Lobos Ruiz, Wenliang Zhao, Shengjun Pan, Yu Sun, and Quan Lu. 2018. Field-Weighted Factorization Machines for Click-Through Rate Prediction in Display Advertising. In *Proceedings of the 2018 World Wide Web Conference* (Lyon, France) *(WWW '18)*. Association for Computing Machinery, New York, NY, USA, 1349–1357. https://doi.org/10.1145/3178876.3186040

[26] Junwei Pan, Jian Xu, Alfonso Lobos Ruiz, Wenliang Zhao, Shengjun Pan, Yu Sun, and Quan Lu. 2018. Field-weighted factorization machines for click-through rate prediction in display advertising. In *Proceedings of the 2018 World Wide Web Conference*. 1349–1357.

[27] Harshit Pande. 2021. Field-Embedded Factorization Machines for Click-through rate prediction. arXiv:2009.09931 [cs.IR]

[28] Kaare Brandt Petersen and Michael Syskind Pedersen. 2012. The Matrix Cookbook.

[29] Steffen Rendle. 2010. Factorization Machines. In *2010 IEEE International Conference on Data Mining*. 995–1000. https://doi.org/10.1109/ICDM.2010.127

[30] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International conference on data mining*. IEEE, 995–1000.

[31] James Saunderson, Venkat Chandrasekaran, Pablo A Parrilo, and Alan S Willsky. 2012. Diagonal and low-rank matrix decompositions, correlation matrices, and ellipsoid fitting. *SIAM J. Matrix Anal. Appl.* 33, 4 (2012), 1395–1416.

[32] David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. 2015. Hidden technical debt in machine learning systems. *Advances in neural information processing systems* 28 (2015), 2503–2511.

[33] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 1161–1170.

[34] Yang Sun, Junwei Pan, Alex Zhang, and Aaron Flores. 2021. FM2: Field-Matrixed Factorization Machines for Recommender Systems. In *Proceedings of the Web Conference 2021* (Ljubljana, Slovenia) *(WWW '21)*. Association for Computing Machinery, New York, NY, USA, 2828–2837. https://doi.org/10.1145/3442381.3449930

[35] Yang Sun, Junwei Pan, Alex Zhang, and Aaron Flores. 2021. FM2: Field-matrixed factorization machines for recommender systems. In *Proceedings of the Web Conference 2021*. 2828–2837.

[36] Marcin Tomczak, Siddharth Swaroop, and Richard Turner. 2020. Efficient low rank gaussian variational inference for neural networks. *Advances in Neural Information Processing Systems* 33 (2020), 4610–4622.

[37] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*. 1–7.

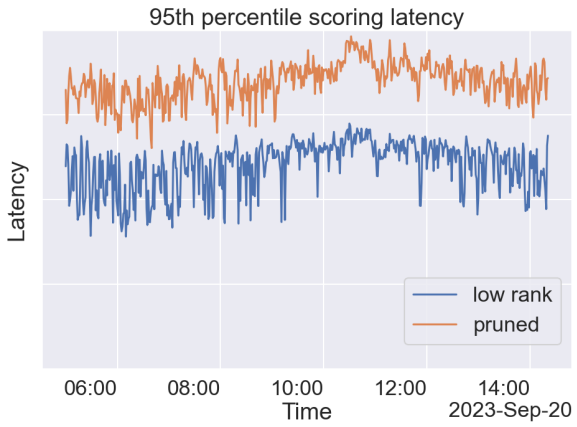
[38] Yang Zhang, Fuli Feng, Chenxu Wang, Xiangnan He, Meng Wang, Yan Li, and Yongdong Zhang. 2020. How to retrain recommender system? A sequential meta-learning method. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1479–1488.

[39] Yong Zhao, Jinyu Li, and Yifan Gong. 2016. Low-rank plus diagonal adaptation for deep neural networks. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 5005–5009.

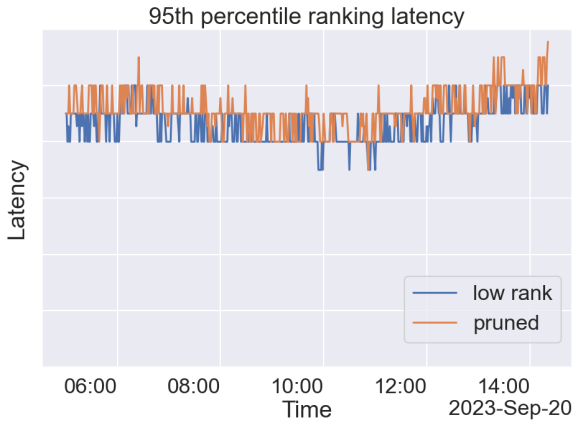
A LATENCY CHARTS

In Figure 3 we observe that the improvement in the latency of the low-rank solution is consistent over time.

(a)



(b)



(c)

Figure 3: Latency graphs over a 10 hour period.