Learning Feature Interactions with Lorentzian Factorization Machine

Canran Xu,[1*] **Ming Wu**[2*†]

[1]eBay Inc.
[2]Department of Electrical Engineering, Tongji University, China
canxu@ebay.com, 1730683@tongji.edu.cn

Abstract

Learning representations for feature interactions to model user behaviors is critical for recommendation system and click-trough rate (CTR) predictions. Recent advances in this area are empowered by deep learning methods which could learn sophisticated feature interactions and achieve the state-of-the-art result in an end-to-end manner. These approaches require large number of training parameters integrated with the low-level representations, and thus are memory and computational inefficient. In this paper, we propose a new model named "LorentzFM" that can learn feature interactions embedded in a hyperbolic space in which the violation of triangle inequality for Lorentz distances is available. To this end, the learned representation is benefited by the peculiar geometric properties of hyperbolic triangles, and result in a significant reduction in the number of parameters (20% to 80%) because all the top deep learning layers are not required. With such a lightweight architecture, LorentzFM achieves comparable and even materially better results than the deep learning methods such as DeepFM, xDeepFM and Deep & Cross in both recommendation and CTR prediction tasks.

Introduction

Recommendation system and ads click-trough rate (CTR) prediction both have profound impact on web-scale big data, and have been received a lot of attention in both academia and industrial community. In these systems, the critical problem is to predict the probability of clicks, which will be used to rank the candidate items or ads.

These problems have been successfully tackled by supervised learning methods, in which user profiles and item attributes are used as input features. The challenge is that, the data for web-scale recommendation systems is mostly discrete and categorical, resulting in an extreme large and sparse feature space to optimize. Feature sparsity of this kind is commonly handled by constructing feature interactions which are modeled by inner product of their low dimensional representations. In this so-called factorization machine model (Rendle 2010), the naive inner product is not expressive enough for spurious or implicit feature interactions. Therefore, higher-order factorization machine (Blondel et al. 2016) is proposed to learn higher-order feature interactions efficiently. On the other hand, deep learning based approaches have successfully enhanced the performance significantly because sophisticated feature interactions can be captured. Besides, another merit of the deep learning methods is that they can reduce human efforts in feature engineering. To name a few, DeepFM (Guo et al. 2017) integrates the deep neural networks on top of the factorization machine, and the Deep & Cross neural network (DCN) (Wang et al. 2017) takes outer product of features at the element wise and vector wise level such that higher order feature interactions can be learned in an automatic and explicit fashion. In DCN, the order of interactions is determined by the number of multi-layer perceptron (MLP) layers. However, by using multiple layers of MLPs it is inefficient to learn multiplicative feature interactions. Therefore, in xDeepFM (Lian et al. 2018) it is proposed to use convolution neural network to replace MLP.

In this paper, we re-examine the building block for these models: the inner product between feature representations. To this end, we propose to learn the low-dimensional representations in a hyperbolic geometry equipped with Lorentz distance, such that the triangle inequality for feature vectors could be violated. Our work is inspired by the work on collaborative metric learning (CML) (Hsieh et al. 2017), where the inner product is replaced by the distance between feature vectors, but following a completely different path. As opposed to the work of CML, in which the authors argue that triangle inequality in a Euclidean space should be strictly obeyed, we propose to take advantage of the sign of the triangle inequality. Specifically, we construct our score function for feature interactions not by the inner product or distance between feature vectors, but by examining the triangle formed by them in a hyperbolic space. This reason for our approach is two-folded: (1) a hyperbolic space is intrinsically a lot more capacious than the Euclidean space; (2) the proposed score function will provide a robust objective function to learn fine-grained feature interactions. As a result, the embedding learned by this score function are therefore ready to use directly, just as the vanilla factorization ma-

---

chine. Our experiments on several benchmark datasets show that, even without using any deep learning layers on top of the embedding layer, the proposed approach will achieve the state-of-the-art performance with reducing up to 82.9% in training parameters and 69.7% in training time compared to the strong deep learning baselines such as DeepFM, DCN and xDeepFM.

Our Contributions

Overall, the primary contributions of this paper can be summarized as follows:

- We propose to learn feature interactions with a score function measuring the validity of triangle inequalities. This score function is bounded and dimension agnostic, which opens up a novel viewpoint on learning the fine-grained structure of interacting features.
- We conduct extensive experiments on a wide range of real-world datasets. Our results demonstrate that LorentzFM achieves state-of-the-art performance, compared to existing deep learning approaches.
- We thoroughly investigate the number of training parameters and training time to understand its cost in resource.

Why Learning Triangle Inequality?

In this section, we first provide some technical background about hyperbolic geometry and Lorentz distance. With these definitions, we will show that Lorentz distances for a triangle could violate triangle inequality. The motivation is to show how it is related to the metric learning in Euclidean space.

Hyperbolic Geometry and Lorentz Distance

Hyperbolic geometry aims to study non-Euclidean space with a constant negative curvature. Due to its negative curvature, hyperbolic geometry has very different properties compared to the Euclidean geometry.

First, the circumference and area of a circle in the hyperbolic space grow exponentially with the radius, as opposed to the linear and quadratic growth rate in Euclidean space. Therefore, the capacity of embeddings in the hyperbolic space of the same upper bound in radius is much larger than its Euclidean counterpart. Secondly, the triangle inequality Eq. (6) with Lorentz distance defined in Eq. (5) defined therein can be violated. This property enables us to characterize the pairwise relation between points in the hyperbolic space by the sign of the inequality.

There are several important models of hyperbolic space for computation purpose: the Poincaré ball model, the hyperboloid model, the Klein model and so on. These models all describe the same geometry and can be connected by transformations that preserve geometric properties of the hyperbolic space such as isometry. In the following, we only introduce the technical basics of the hyperboloid model, which will be used throughout the paper.

Hyperboloid Model   Let us define the Lorentzian inner product between $\mathbf{u}, \mathbf{v} \in \mathbb{R}^{n+1}$ as

$$\langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{L}} = -u_0 v_0 + \sum_{i=1}^{n} u_n v_n. \qquad (1)$$

The hyperboloid of dimension n, $\mathcal{H}^{n,\beta} \subseteq \mathbb{R}^{n+1}$ consists of the following set of points:

$$\mathcal{H}^{n,\beta} = \left\{ \mathbf{x} \in \mathbb{R}^{n+1} : ||x||_{\mathcal{L}}^2 = -\beta, x_0 \geq \beta \right\}, \qquad (2)$$

where $||\mathbf{x}||_{\mathcal{L}}^2 = \langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{L}}$ is the Lorentzian norm of the vector $\mathbf{x}$. With this definition, for every vector $\mathbf{x} \in \mathcal{H}^{n,\beta}$, the component x_0 is not a free parameter, and is given by

$$x_0 = \sqrt{\beta + \sum_{i=1}^{n} x_i^2}. \qquad (3)$$

The associated geodesic distance between two points is given as

$$d_\ell(\mathbf{u}, \mathbf{v}) = \text{arccosh}(-\langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{L}}). \qquad (4)$$

Note that the origin vector of the hyperboloid model $\mathcal{H}^{n,\beta}$ is $\mathbf{0} = (\beta, 0, \ldots, 0)$, and its Lorentzian inner product with vector $\mathbf{x}$ is simply $\langle \mathbf{0}, \mathbf{x} \rangle_{\mathcal{L}} = -x_0 \leq \beta$.

When $\beta = 1$, the model is called a *unit* hyberboloid model, which will be used throughout the paper. Without introducing any confusions, we will simply call it hyperboloid model and use $\mathcal{H}^n$ to denote $\mathcal{H}^{n,1}$.

Lorentz Distance   The squared Lorentz distance, or Lorentz distance in short, between $\mathbf{u}, \mathbf{v} \in \mathcal{H}^n$ is given by

$$d_{\mathcal{L}}^2(\mathbf{u}, \mathbf{v}) = ||\mathbf{u} - \mathbf{v}||_{\mathcal{L}}^2 = -2 - 2\langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{L}}. \qquad (5)$$

It satisfies almost all axioms of Euclidean geometry but the triangle inequality, which is one of the most crucial geometric property with positive definitive Riemannian metric. It states that for any three points, $\mathbf{x}, \mathbf{y}$ and $\mathbf{z}$, any two pairwise distances $d(\cdot, \cdot)$ should be greater or equal to the remaining pairwise distance:

$$d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y}). \qquad (6)$$

In the hyperbolic space, the geodesic distance defined in Eq. (4) between three points also satisfies this property. However, the inequality could be violated with Lorentz distances, because the Riemannian metric is negative. Consider the triangle formed by the origin $\mathbf{0}$ and other two points $\mathbf{u}$ and $\mathbf{v}$ in Figure 1. When the two points are far apart on the opposite side of x_1 axis, the triangle inequality is violated. On the other hand, the triangle inequality holds if the two points are on the same side of x_1 axis.

Learning Triangle Inequalities

As pointed out in Hsieh et al. (2017), learning the distance in the embedding space rather than inner product has advantages to learn a fine-grained embedding space that could capture the representation not only for item-user interactions, but also for item-item and user-user distances. Essentially, the so-called metric learning scheme is blessed by the constraint of triangle inequality.

In contrast to collaborative metric learning scheme, we argue that the feature interaction between two points can be learned by the sign of the triangle inequality for Lorentz distance, instead of using the distance itself. Formally, our target score function is written as

$$\mathcal{S}(\mathbf{x}, \mathbf{y}) = \frac{d_{\mathcal{L}}^2(\mathbf{x}, \mathbf{y}) - d_{\mathcal{L}}^2(\mathbf{x}, \mathbf{0}) - d_{\mathcal{L}}^2(\mathbf{0}, \mathbf{y})}{\langle \mathbf{0}, \mathbf{x} \rangle_{\mathcal{L}} \langle \mathbf{0}, \mathbf{y} \rangle_{\mathcal{L}}}, \qquad (7)$$
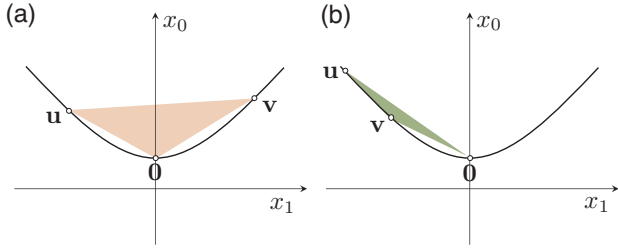
Figure 1: Visualization of the hyperboloid model $\mathcal{H}^2$. In this model, triangles could (a) violates Lorentz triangle inequality and (b) obeys triangle inequality for points on the hyperboloid.
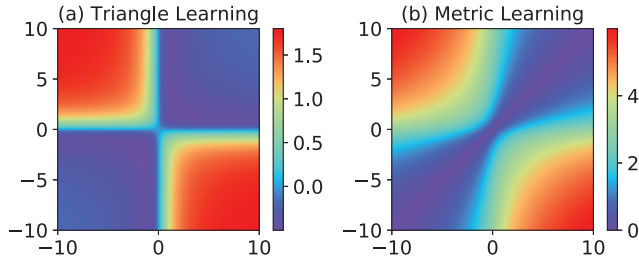

Figure 2: The 2D landscape for the score function in the (a) triangle learning scheme (*e.g.*, Eq. (7)) and (b) metric learning scheme using geodesic distance in a 2D hyperboloid model.

where $\mathbf{0} = (1, 0, \cdots, 0)$ is the origin. The nominator is simply the difference between the two sides of the triangle inequality, and the denominator is used to bound the score function.

The advantage of using Eq. (7) is that, the function is bounded in $[-0.5, 2]$ in all dimensions. Therefore, the score function is free from the curse of dimensionality, compared to the collaborative metric learning scheme. To illustrate the landscape of the function, we plot Eq. (7) for two points in 2D in Figure 2 (a). Because each point on the 2D hyperboloid has only one free parameter, we plot the landscape of the score function by setting the x-axis and y-axis corresponding to the free parameter of each point respectively in a 2D plane. We also plot the landscape for collaborative metric learning scheme using the geodesic distance defined in Eq. (4) in Figure 2 (b). By comparison, we can observe that the landscape for our approach is smooth and bounded, but the score function for collaborative metric learning scheme is unbounded. The bounded property is promising because the embedding vectors are free to be far away from the origin with a smooth growth in the score function.

Lorentzian Factorization Machine

In this section, we first give an overview of our proposed Lorentzian Factorization Machine or *LorentzFM* for short, named after learning the triangle inequality equipped with Lorentz distance. Next, we describe each component of the model in some greater details. Finally, we present a detailed description of how to optimize the model.
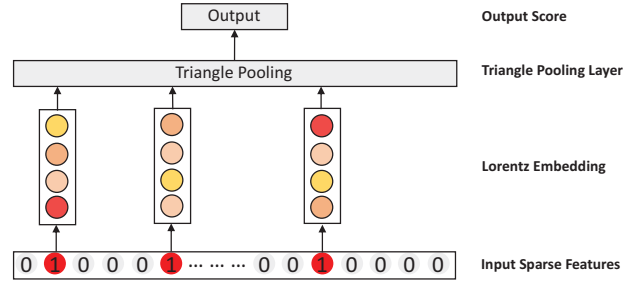

Figure 3: The architecture of LorentzFM. It comprises an input layer, a Lorentz embedding layer and a triangle pooling layer to aggregate feature interactions.

Overview

The goal of our approach is to map the original sparse features into a low-dimensional space. As shown in Figure 3, our purposed method takes the sparse feature vector $\mathcal{V}_x$ as input, followed by a Lorentz embedding layer that projects all features into the same hyperbolic space. Next, we feed the embeddings of all fields into a novel triangle pooling layer, which is coined as an aggregation function of all feature pairs to measure the soft "validness" of triangle inequality in overall. Unlike the recent state-of-the-arts neural architectures built upon Euclidean embeddings, LorentzFM does not need any extra parameters. In particular, for given sparse input $\mathcal{V}_x$, the output of the pooling layer is the model output score:

$$\hat{S}_{LFM}(\mathcal{V}_x) = \sum_{i,j=1,i\neq j}^{d} \mathcal{T}(\mathbf{v}_i, \mathbf{v}_j)x_i x_j, \qquad (8)$$

where $\mathbf{v}_i, \mathbf{v}_k \in \mathcal{H}^n$ are embedding vector for each input feature field, and $\mathcal{T}(\cdot, \cdot)$ is the feature interaction function. Although formally in Eq. (8) the linear term is missing, it actually reappears in the pooling function, $\mathcal{T}(\cdot, \cdot)$, as shown later.

Lorentz Embedding Layer

The embedding layer is a lookup table to project sparse features to low-dimensional dense vectors in the Lorentz space. Formally, let $\mathbf{v}^k \in \mathcal{H}^n$ be the embedding vector for the k-th feature, whereas the 0-th component is given by the constraint as Eq. (3).

In some cases, categorical features can be multi-valued. For example, the genre of movie "Titanic" could be either "Drama" or "Romance". We use multiple fields for these categorical features under a universal encoding and pad them with "unknown" tags to ensure each sample is aligned in feature dimension.

Triangle Pooling Layer

We then feed the embedding vectors into the triangle pooling layer, which is an aggregation function that converts a set of

embedding vectors to one vector:

$$\mathcal{T}(\mathbf{u}, \mathbf{v}) = \frac{d_{\mathcal{L}}^2(\mathbf{u}, \mathbf{v}) - d_{\mathcal{L}}^2(\mathbf{0}, \mathbf{u}) - d_{\mathcal{L}}^2(\mathbf{0}, \mathbf{v})}{2\langle \mathbf{0}, \mathbf{u} \rangle_{\mathcal{L}} \langle \mathbf{0}, \mathbf{v} \rangle_{\mathcal{L}}}$$

$$= \frac{1 - \langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{L}} - u_0 - v_0}{u_0 v_0} \qquad (9)$$

$$= \underbrace{\frac{1 - \langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{L}}}{u_0 v_0}}_{\text{interaction term}} - \underbrace{\left(\frac{1}{u_0} + \frac{1}{v_0} \right)}_{\text{linear term}}.$$

In the second line we use the definition of Lorentz distance and Lorentz inner product. Owing to the normalization denominator, the linear term emerges, as shown in the last line.

Objective and Learning

The objective function to optimize both recommendation system and CTR prediction is the binary cross-entropy (BCE):

$$\arg \min_{\theta} \sum_i -y_i \log(p_i) - (1 - y_i) \log(1 - p_i), \qquad (10)$$

where p_is are the probabilities for the i-th sample input vector $\mathcal{V}_x^{(i)}$ and computed as $p_i = \sigma(\hat{S}_{LFM}(\mathcal{V}_x^{(i)}))$, and y_is are the true labels. Even though Bayesian Personalized Ranking (BPR) loss (Rendle et al. 2009) proves to be useful in common recommendation systems, we do not use it because the BCE loss is sign-sensitive, which is the desired property, while the BPR loss is not.

The parameters of our model are learned by using Riemannian stochastic gradient descent (RSGD) (Bonnabel 2013). As shown by Nickel and Kiela (2018), the parameters are updated by the following form

$$\theta_{t+1} = \exp_{\theta_t}(-\eta \operatorname{grad} f(\theta_t)), \qquad (11)$$

where $\operatorname{grad} f(\theta_t)$ is the gradient defined in the Riemannian manifold and η is the learning rate. The Riemannian gradient is obtained by multiplying the gradient in the Euclidean space by the Lorentz metric, and then performing an orthogonal projection onto the tangent space spanned by the current parameter set. Finally, the parameter update is given by the following exponential map

$$\exp_{\theta_t}(\mathbf{x}) = \cosh(||\mathbf{v}||_{\mathcal{L}})\mathbf{x} + \sinh(||\mathbf{v}||_{\mathcal{L}}) \frac{\mathbf{v}}{||\mathbf{v}||_{\mathcal{L}}} \qquad (12)$$

which maps a tangent vector $\mathbf{v}$ in the tangent space onto the Lorentz manifold. Details can be found in Nickel and Kiela (2018).

Since the score function Eq. (7) is bounded and dimension independent, it is not necessary to apply an L_2 regularization term over the embedding vector, because it will create a resisting gradient towards the origin of the hyperboloid model.

Experiment

In this section, we evaluate LorentzFM with four real-word datasets. We aim to answer the following research questions:

- **RQ1:** Compared with the state-of-the-art deep learning based methods, how does our method perform?

- **RQ2:** What is the resource cost for LorentzFM compared to existing deep learning based methods?

- **RQ3:** How can we interpret feature interactions learned by LorentzFM?

	Steam	MovieLens	KKBox	Avazu[†]
#Samples	1.23M	573K	2.67M	40.4M
#User Fields	1	5	7	–
#Item Fields	11	7	6	–
#Fields	12	12	12	21
#Users	26.6K	6.0K	21.1K	–
#Items	5.9K	3.1K	9.3K	–
#Features	46.2K	9.4K	38.2K	972K
Sparsity	99.22%	96.85%	98.64%	–

Table 1: Statistics of datasets after data preprocessing. [†] In Avazu dataset there are no explicit indicators for item ID and user IDs, so we leave the corresponding rows blank.

Datasets and Preprocessing

We use Steam, MovieLens and KKBox datasets to evaluate the recommendation task and Avazu dataset for the CTR task. The datasets are briefly described as follows.

- **Steam**[1] is a dataset crawled from Steam database, which includes rich information such as users' playing hours, games' price, category and publisher *etc*. We keep positive samples by setting the threshold of playing hours to 100.

- **MovieLens**[2] is a set of benchmark datasets for evaluating recommendation algorithms. We use the MovieLens 1M version and keep samples with ratings above 3.

- **KKBox**[3] dataset is adopted from the WSDM cup 2018 Challenge provided by the music streaming service KKBox. This dataset includes demographic information about the users and structured meta data about the songs. The original dataset has a binary label and we only keep the positive samples.

- **Avazu**[4] is an ads click-through dataset with more than 40 millions instances. It consists of 10 days of ad click-through data with only sparse features.

For the three recommendation datasets we process the dataset as follows: (1) keeping only positive ratings as described above and filter the {KKBox, Steam, MovieLens} dataset to {20, 20, 5}-core setting; (2) randomly splitting {10K, 10K} samples for {validation, testing}, and the rest as training set. For the Avazu dataset, we remove the timestamp field and then replace features less than 5 times by a universal "unknown" tag. We randomly select 80% samples for

---

[1]http://cseweb.ucsd.edu/~jmcauley/datasets.html#steam_data
[2]https://grouplens.org/datasets/movielens/1m/
[3]https://www.kaggle.com/c/kkbox-music-recommendation-challenge
[4]https://www.kaggle.com/c/avazu-ctr-prediction

	Steam			MovieLens			KKBox			Avazu	
	MRR	HR@10	NDCG	MRR	HR@10	NDCG	MRR	HR@10	NDCG	AUC	Logloss
FM	0.1042	0.1898	0.2376	0.0638	0.1258	0.1966	0.0578	0.0998	0.1735	0.7582	0.3920
NFM	0.1213	0.2286	0.2591	0.0723	0.1446	0.2082	0.0586	0.1086	0.1779	0.7641	0.3884
DeepFM	0.1284	0.2408	0.2664	0.0780	0.1548	0.2143	0.0637	0.1140	0.1843	0.7674	0.3864
xDeepFM	0.1228	0.2316	0.2615	**0.0791**	**0.1570**	**0.2156**	0.0639	0.1148	0.1846	0.7677	0.3865
DCN	0.1213	0.2261	0.2598	0.0749	0.1512	0.2119	0.0644	0.1136	0.1848	0.7677	0.3863
LorentzFM	**0.1362**	**0.2592**	**0.2746**	0.0780	0.1533	0.2129	**0.0733**	**0.1295**	**0.1938**	**0.7775**	**0.3828**

Table 2: Best performance of each model with fixed embedding size 10 on the test set.

training and, 10% for validation and 10% for testing. Data statistics are shown in Table 1.

Experiment Settings

Evaluation Metrics   For each positive user-item pair in the recommendation task, we calculate the following three ranking metrics to evaluate the performance over *all* unobserved samples:

- **MRR** is the average of the reciprocal ranks of the test items by their scores over the entire ranking list.

- **HR@10** measures the relative orders among positive and negatives within the top 10 of the ranking list.

- **NDCG** accounts for the position of the test items by assigning higher scores to top ranks over the entire ranking list.

For the CTR prediction task, we use AUC (Area Under the ROC curve) and logloss (binary cross-entropy) for evaluation. AUC measures the probability that an instance prediction will be ranked a higher score to a randomly chosen negative item, and in contrast logloss measures the distance between the predicted score and the ground truth label for each sample.

Compared Baselines   We consider the following models are our baselines:

- **FM** (Rendle 2010) combines the second-order feature interaction as well as a first order term.

- **NFM** (He and Chua 2017) can learn implicit high-order features by stacking MLP on top of the FM model.

- **DCN** (Wang et al. 2017) concatenates MLP and Cross Network, which takes outer product of concatenated feature vector at bit-wise level. It can extract both the implicit and explicit high-order features.

- **DeepFM** (Guo et al. 2017) combines the FM model and deep MLP to get the low and high-order features.

- **xDeepFM** (Lian et al. 2018) takes out product at the vector-wise level and applies the convolution layers to get the high-order explicit and implicit features.

Training Details

We implement LorentzFM and all baselines in PyTorch[5] (Paszke et al. 2017), and for each model the reported performance is given by performing grid search over the hyperparameters on the validation set, with early stopping at 20 epochs. The embedding size is set to 10 for all models for fair comparisons. For the recommendation task, for each item pair in the training set, we randomly sample 10 unobserved items together with the given user as negative samples. MRR is the metric for early stopping. For the CTR task, we use the original binary labels for training, and logloss is monitored for early stopping .

LorentzFM Details Burn-in period is set to 25 epochs. For the recommendation task, we tune the learning rate $\in$ [0.05, 0.1, 0.2, 0.3] with RSGD and batch size $\in$ [64, 128, 256, 512]. For the CTR prediction task, the learning rate is tuned $\in$ [0.1, 0.2, 0.3] and batch size is set to 4096.

Baselines Details The loss function is BPR loss for the recommendation task and BCE loss for the CTR task. We tune the dropout rate $p \in \{0, 0.1, \ldots, 0.5\}$, the MLP layers $\in$ [[512, 256], [400, 400], [100, 100]], and the batch size $\in$ [64, 128, 256, 512]. Learning rate is set to 0.001 with Adam optimizer (Kingma and Ba 2015). We use an L_2 regularization with $\lambda = 10^{-5}$. For the CTR prediction task, all hyperparameters except for the embedding size are set identical to the original paper.

Performance Comparison (RQ1)

Table 2 reports our experiment results of each individual model on the four datasets with fixed embedding size 10. For each dataset, all the deep learning methods (NFM, DeepFM, xDeepFM and DCN) substantially outperform FM. It is expected because additional neural net layers built on top of the second-order feature interaction layer will incorporate higher-order feature interactions. The performance between these deep learning models are close upon fine-tuning over the hyper-parameters. Surprisingly, LorentzFM achieves even better results compared to these deep learning models significantly except for the MovieLens dataset. We observe that in MovieLens the number of sparse features is 5-100 times smaller than the other three datasets, which reflects that LorentzFM is especially powerful when the data is extremely sparse.
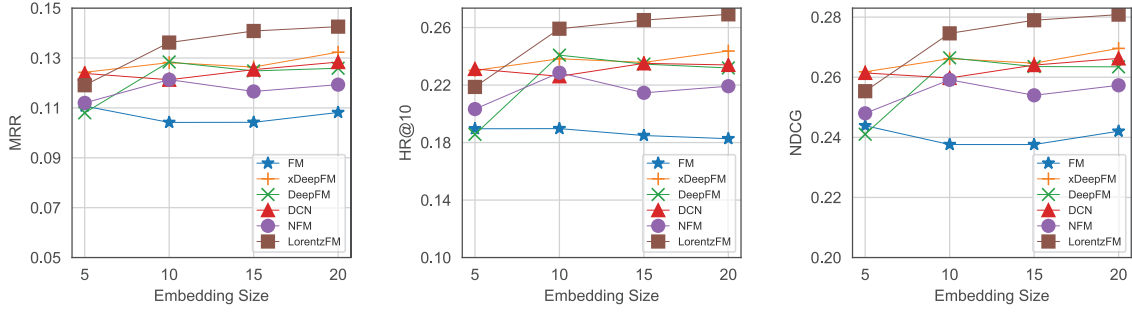
---

[5]https://pytorch.org/

Figure 4: Performance comparison on the test set *w.r.t.* different embedding sizes for the Steam dataset.

	# Training Parameters				Training Time per Epoch (seconds)			
	Steam	MovieLens	KKBox	Avazu	Steam	MovieLens	KKBox	Avazu
FM	507k	103k	420k	10.68M	124.0	82.4	648.3	312.1
NFM	519k	115k	432k	11.36M	158.7	87.7	746.6	478.2
DeepFM	717k	313k	634k	11.10M	159.2	87.1	693.3	450.7
xDeepFM	865k	498k	808k	11.46M	391.0	223.4	1340.2	1459.8
DCN	530k	126k	444k	11.24M	152.6	62.16	666.8	547.2
LorentzFM	415k	85k	344k	8.75M	160.7	67.6	664.2	522.2
$-\Delta\%$	21.8%	82.9%	57.4%	22.1%	-8.1%	69.7%	50.4%	4.6%

Table 3: Number of training parameters and training time per epoch according to the best performance in Table 2. The last row corresponds to the reduced percentage in training parameters or training time of LorentzFM, compared to the best performed baseline.

In order to understand the impact of embedding size on LorentzFM as well as the compared baselines, we conduct control experiments on the Steam dataset by holding the best hyper-parameter settings for each model in Table 2, while varying the embedding size $\in [5, 10, 15, 20]$. Figure 4 demonstrates that in general the performance of all models increases with embedding size except for FM. Note that the performance of LorentzFM surpasses over other deep learning methods at embedding size $\in [10, 15, 20]$.

Resource Cost Comparison (RQ2)

Table 3 compares the amount of training parameters and the training time per epoch between LorentzFM and the best tuned baselines. In order to compare the training time fairly, we use the same batch size for each model. The number of training parameters for LorentzFM is 9 times feature size, because the zeroth component of each embedding vector is not a free parameter. In the last row of Table 3, we show the reduced percentage of the corresponding training parameters and training time when LorentzFM is compared with the best performing baseline. Because LorentzFM does not need any deep learning layers on top of the triangle pooling layer, it could reduce up to 82.9% training parameters and 69.7% in training time compared to xDeepFM.

Case Study (RQ3)

The fine-grained feature interactions are the essence for LorentzFM to outperform deep learning models. Therefore,

it is expected that the structure of the feature interactions is capable to explain the human behavior. To demonstrate this, we sample a typical user (user ID "76561198071045315") whose almost *all* positive ratings are free in price from the Steam dataset. We then plot the heatmap of feature interaction score of a positive item and a negative item from the test set for both LorentzFM and FM. The result is shown in Figure 5, in which we mask the diagonal elements. From the heatmap we observe the following:

- For the positive sample, the game is free of charge. The feature interaction score between "UserID" and "Price" learned by LorentzFM is highlighted with score 0.4, indicating that price free is strongly preferred by the user, see Figure 5 (a). However, as shown in Figure 5 (c), the information about this fine structure is missing in the heatmap for FM, because the inner product between the embedding of this "UserID" and "Price" is -0.14.

- For the negative sample, the price of the game is $14.99. In this case, the score learned by LorentzFM between "UserID' and "Price" is the most negative among others, as shown in Figure 5 (b), meaning that the negative sample is recognized mostly due to its price. On the other hand, the inner product between "UserID" and "Price" is not dominating for FM.

This case study shows that compared to FM, the interactions between feature pairs learned from LorentzFM are more informatively related to the underlying facts according to the
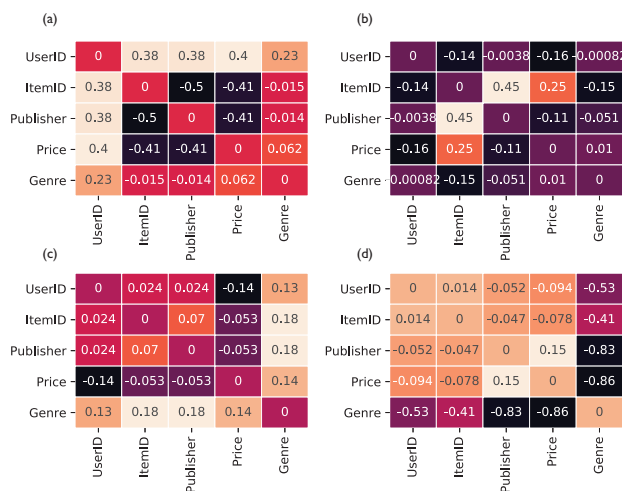
Figure 5: Visualization of the heatmap of score functions from LorentzFM for (a) a positive sample and (b) a negative sample, and from FM for the same (c) positive sample and (d) negative sample.

sign and the magnitude of the pairwise score.

Related Work

Factorization machine (FM) (Rendle 2010) and field-aware factorization machine (FFM) (Juan et al. 2016) are two of the most well-known models for both recommendation systems and CTR predictions. FM models the sparse feature interactions by learning latent factors, and FFM introduced field aware factors to increase the model capacity and expressiveness. However, FFM consumes huge memory due to its field awareness and henceforth its scalability is restricted. Therefore, higher-order factorization machine (Blondel et al. 2016) is proposed to model feature interactions beyond quadratic level.

The research community of CTR prediction recently pays more attention to the deep learning models, due to their strong capabilities in feature extraction. Wide & Deep model was initially introduced for jointly training wide linear models and deep neural networks to capture both shallow and deep representations. However, these advances still require manual efforts for feature engineering in advance of model training stage. DeepFM (Guo et al. 2017), Neural FM (He and Chua 2017) and Deep & Cross Network (Wang et al. 2017) are representative models to combine low-order and high-order feature interactions to improve the performance in CTR in an automatic manner. Product-based Neural Networks (Qu et al. 2016) tries to capture high-order feature interactions by involving a product layer on top of the embedding layer. Apart from the advances in model architectures, attention mechanism, which first appears in neural machine translation models to learn a weighted sum of model outputs, also finds its impact for CTR and recommendation systems. Works that incorporate attention mechanism include Deep Interest Network (Zhou et al. 2018), Attentional Factoriza-

tion Machine (Xiao et al. 2017) and AutoInt (Song et al. 2019).

Traditional research on recommendation system tries to learn latent factors for user and items based on their preferences (Salakhutdinov and Mnih 2007; Wang, Wang, and Yeung 2015). In these works, learning latent factors by factorizing the interaction matrix is equivalent to using Euclidean inner product of embeddings of user, item and their side information. Our work is inspired by the seminal paper collaborative metric learning (Hsieh et al. 2017) which argues that inner product formulation will result in violation of triangle inequality in Euclidean space. The authors propose to learn not only user-item, but also user-user and item-item distances. This idea is later on generalized to use hyperbolic distance (Vinh et al. 2018; Chamberlain et al. 2019) for recommendation systems, accompanied by the recent advances in hyperbolic representation learning and hyperbolic neural networks (Nickel and Kiela 2017; 2018; Ganea, Bécigneul, and Hofmann 2018; Sala et al. 2018; Law et al. 2019; Gülçehre et al. 2019; Bécigneul and Ganea 2019). While hyperbolic representation seems eminent over Euclidean representations in many applications, especially for problems with underlying hierarchical structures, previous study focus on learning the distances rather than the violation of triangle inequalities in terms of Lorentz distance. Besides, another type of approach to model feature interactions is to replace the inner product in the FM by circular correlation or circular convolution, as shown in Tay et al. (2019).

Conclusion

In this paper, we present a lightweight yet effective model named LorentzFM for recommendation and CTR prediction tasks. We propose a new score function by characterizing if the triangle inequality for Lorentz distance is violated or not in the hyperboloid model. Our model is advantageous with its capability in modeling feature interactions with the underlying geometry, without using any deep learning layers. Empirical experiment results on four real-world benchmark dataset show that LorentzFM achieves the state-of-the-art performance, outperforming multiple existing deep learning baselines, while using fewer training parameters and less training time.

Acknowledgements

References

Bécigneul, G., and Ganea, O. 2019. Riemannian adaptive optimization methods. In *7th International Conference on Learning Representations, ICLR 2019*.

Blondel, M.; Fujino, A.; Ueda, N.; and Ishihata, M. 2016. Higher-order factorization machines. In *Advances in Neural Information Processing Systems 29, 2016*, 3351–3359.

Bonnabel, S. 2013. Stochastic gradient descent on riemannian manifolds. *IEEE Trans. Automat. Contr.* 58(9):2217–2229.

Chamberlain, B. P.; Hardwick, S. R.; Wardrope, D. R.; Dzogang, F.; Daolio, F.; and Vargas, S. 2019. Scalable hyperbolic recommender systems. *CoRR* abs/1902.08648.

Ganea, O.; Bécigneul, G.; and Hofmann, T. 2018. Hyperbolic entailment cones for learning hierarchical embeddings. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, 1632–1641.

Gülçehre, Ç.; Denil, M.; Malinowski, M.; Razavi, A.; Pascanu, R.; Hermann, K. M.; Battaglia, P. W.; Bapst, V.; Raposo, D.; Santoro, A.; and de Freitas, N. 2019. Hyperbolic attention networks. In *7th International Conference on Learning Representations, ICLR 2019*.

Guo, H.; Tang, R.; Ye, Y.; Li, Z.; and He, X. 2017. Deepfm: A factorization-machine based neural network for CTR prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI 2017*, 1725–1731.

He, X., and Chua, T. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2017*, 355–364.

Hsieh, C.; Yang, L.; Cui, Y.; Lin, T.; Belongie, S. J.; and Estrin, D. 2017. Collaborative metric learning. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017*, 193–201.

Juan, Y.; Zhuang, Y.; Chin, W.; and Lin, C. 2016. Field-aware factorization machines for CTR prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems, RecSys 2016*, 43–50.

Kingma, D. P., and Ba, J. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015*.

Law, M. T.; Liao, R.; Snell, J.; and Zemel, R. S. 2019. Lorentzian distance learning for hyperbolic representations. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, 3672–3681.

Lian, J.; Zhou, X.; Zhang, F.; Chen, Z.; Xie, X.; and Sun, G. 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, KDD 2018*, 1754–1763.

Nickel, M., and Kiela, D. 2017. Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems 30, 2017*, 6338–6347.

Nickel, M., and Kiela, D. 2018. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, 3776–3785.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.

Qu, Y.; Cai, H.; Ren, K.; Zhang, W.; Yu, Y.; Wen, Y.; and Wang, J. 2016. Product-based neural networks for user response prediction. In *IEEE 16th International Conference on Data Mining, ICDM 2016*, 1149–1154.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2009. BPR: bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence, UAI 2009*, 452–461.

Rendle, S. 2010. Factorization machines. In *ICDM 2010, The 10th IEEE International Conference on Data Mining, 2010*, 995–1000.

Sala, F.; Sa, C. D.; Gu, A.; and Ré, C. 2018. Representation tradeoffs for hyperbolic embeddings. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, 4457–4466.

Salakhutdinov, R., and Mnih, A. 2007. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems 20, 2007*, 1257–1264.

Song, W.; Shi, C.; Xiao, Z.; Duan, Z.; Xu, Y.; Zhang, M.; and Tang, J. 2019. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019*, 1161–1170.

Tay, Y.; Zhang, S.; Luu, A. T.; Hui, S. C.; Yao, L.; and Vinh, T. D. Q. 2019. Holographic factorization machines for recommendation. In *The 33rd AAAI Conference on Artificial Intelligence, AAAI 2019*, 5143–5150.

Vinh, T. D. Q.; Tay, Y.; Zhang, S.; Cong, G.; and Li, X. 2018. Hyperbolic recommender systems. *CoRR* abs/1809.01703.

Wang, R.; Fu, B.; Fu, G.; and Wang, M. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*, 12:1–12:7.

Wang, H.; Wang, N.; and Yeung, D. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, SIGKDD 2015*, 1235–1244.

Xiao, J.; Ye, H.; He, X.; Zhang, H.; Wu, F.; and Chua, T. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI 2017*, 3119–3125.

Zhou, G.; Zhu, X.; Song, C.; Fan, Y.; Zhu, H.; Ma, X.; Yan, Y.; Jin, J.; Li, H.; and Gai, K. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018*, 1059–1068.