DNN2LR: Interpretation-inspired Feature Crossing for Real-world Tabular Data

Zhaocheng Liu, Qiang Liu, Haoli Zhang, Yuntian Chen

Abstract—For sake of reliability, it is necessary for models in real-world applications, such as financial applications, to be both powerful and globally interpretable. Simple linear classifiers, e.g., Logistic Regression (LR), are globally interpretable, but not powerful enough to model complex nonlinear interactions among features in tabular data. Fortunately, automatic feature crossing is an effective way to capture the interactions among features in tabular data, and able to promote the performances of LR without heavy handcrafted feature engineering. Meanwhile, Deep Neural Networks (DNNs) have shown great effectiveness for modeling tabular data. However, DNN can only implicitly model feature interactions in the hidden layers, and is not globally interpretable. Accordingly, it will be promising if we can propose a new automatic feature crossing method to find the feature interactions in DNN, and use them as cross features in LR. In this way, we can take advantage of the strong expressive ability of DNN and the good interpretability of LR. Recently, local piece-wise interpretability of DNN has been widely studied. The piece-wise interpretations of a specific feature are usually inconsistent in different samples, which is caused by feature interactions in the hidden layers. Inspired by this, we give a definition of the interpretation inconsistency in DNN, and accordingly propose a novel method called DNN2LR. DNN2LR can generate a compact and accurate candidate set of cross feature fields, and thus promote the efficiency of searching for useful cross feature fields. The whole process of learning feature crossing in DNN2LR can be done via simply training a DNN model and a LR model. Extensive experiments have been conducted on five public datasets, as well as two real-world datasets. The final model, a LR model empowered with cross features, generated by DNN2LR can achieve better performances compared with complex DNN models. The experimental results strongly verify the effectiveness and efficiency of DNN2LR, especially on real-world datasets with large numbers of feature fields.

Index Terms—Deep Neural Networks, Interpretation, Feature Crossing, Automated Machine Learning.

1 Introduction

In application areas such as finance and healthcare, reliability and interpretability are strongly desired. Thus, powerful and globally interpretable models are well appreciated in real-world applications. In [1], the global interpretability is defined as, *we are able to understand the whole logic of a model and follow the entire reasoning leading to all the different possible outcomes*. Commonly-used linear classifiers, e.g., Logistic Regression (LR), are simple and globally interpretable. However, LR usually has relatively poor performances, and it is hard for LR to model complex nonlinear interactions among features in tabular data. To improve the performances of LR, heavy handcrafted feature engineering is usually required. On the other hand, Deep Neural Networks (DNNs) [2] and tree ensemble models [3], [4] are able to take use of complex nonlinear feature interactions. However, as pointed in [1], these methods are not globally interpretable. Fortunately, automatic feature crossing, which takes cross-product of sparse features, is a practical direction in automated machine learning [5]. It is a promising way to capture the interactions among categorical features in tabular data in real-world applications [6], [7]. Via automatic generation of cross features, we can achieve better performances with the simple LR model without heavy handcrafted feature engineering. As pointed in [7], cross features, instead of latent embeddings or latent representations in DNN, are highly interpretable. Moreover, utilizing cross features is more flexible and suitable for large-scale online tasks [7], [8]. With the research on automatic feature crossing, we can make the simple LR model powerful and globally interpretable simultaneously, and thus reliable models can be obtained in real-world applications.

According to the definition in previous works [7], [8], in a specific sample k, we can conduct nth-order feature crossing as

$$g_{x_{k,f_1},x_{k,f_2},...,x_{k,f_n}} = x_{k,f_1} \otimes x_{k,f_2} \otimes ... \otimes x_{k,f_n}, \quad (1)$$

where $\otimes$ denotes cross-product, $g_{x_{k,f_1},x_{k,f_2},...,x_{k,f_n}}$ is the corresponding generated cross feature, and $x_{k,f}$ is a binary feature associated with categorical feature field f, e.g., feature "occupation=teacher" associated with the field "occupation". Via feature crossing, the cross feature of a female teacher can be denoted as ("gender=female" $\otimes$ "occupation=teacher"). And an example of feature crossing can be found in Fig. 1. Moreover, considering feature discretization has been proven useful to improve the capability of numerical features [6], [7], [9], [10], [11], we can conduct feature discretization on numerical feature fields to generate corresponding categorical feature fields. Thus, we are able to perform feature crossing on both categorical and numerical feature fields.

Previous works on feature crossing mostly try to search in the set of possible cross feature fields [6], [7], [12], [13], [14]. The candidate set for searching is usually inevitably

- *Z. Liu and Q. Liu contribute equally to this work.*
- *Z. Liu and H. Zhang are with RealAI.*
  E-mail: {zhaocheng.liu,haoli.zhang}@realai.ai
- *Q. Liu is with RealAI and Tsinghua University.*
  E-mail: qiang.liu@realai.ai
- *Y. Chen is with RealAI and Peng Cheng Laboratory.*
  E-mail: yuntian.chen@realai.ai

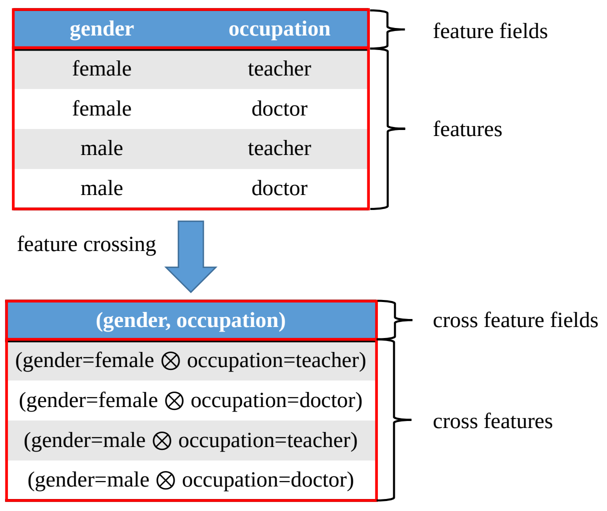

Fig. 1. An example of feature crossing.

large, which leads to low efficiency of learning feature crossing. Tree ensemble models [3], [4] have also been utilized for finding cross features [15], [16]. However, as pointed in [17], it is hard for these tree ensemble models to well handle sparse discrete features, which are important in real-world applications [6], [9]. Besides above crossing methods, some factorization-based methods [18], [19], [20], [21] seek to model feature interactions with product-based similarity. However, the product-based similarity is hard to model all kinds of feature interactions in various applications. Meanwhile, some deep learning-based methods design various crossing modules for modeling feature interactions [8], [22], [23], [24], [25], [26]. Unfortunately, due to the hidden projections in the crossing modules, which are usually incorporated in deep learning technologies, most of these deep learning-based methods are not globally interpretable and capable to explicitly generate enough cross features.

As mentioned in previous works [2], [22], [24], DNN can be a powerful method for capturing various feature interactions in its hidden layers. DNN can implicitly interact features, but can not explicitly provide interpretable cross features. Recently, the interpretability of deep models has drawn great attention in academia, and research works mostly focus on piece-wise interpretability, which means assigning a piece of local interpretation for each sample [27], [28], [29], [30], [31], [32]. That is to say, a nonlinear DNN model can be regarded as a combination of numbers of linear classifiers [33], [34], [35]. This process can be done via the gradient backpropagation from the prediction layer to the input features. We observe that, local interpretations of a specific feature are inconsistent in different samples. This is caused by feature interactions occurred in the hidden layers of DNN. Therefore, in this paper, we give definition of the interpretation inconsistency in DNN, which can help us find useful cross features. With the interpretation inconsistency in DNN, we can take advantage of the strong expressive ability of DNN and the good interpretability of LR.

Accordingly, in this paper, we propose a novel method called DNN2LR, which can automatically learn useful cross feature fields from the interpretation inconsistency of DNN. The process of DNN2LR can be detailed as follows: (1) We train a DNN model with the whole training set. (2) For each sample in the validation set, we calculate the gradients of the output predictions with respect to the input features, i.e., local interpretations. We also calculate the average value of all the corresponding local interpretations as the global interpretation of a specific feature. For a specific feature in a specific sample, if the corresponding local interpretation is far from the corresponding global interpretation, i.e., interpretation inconsistency is large, we regard this feature interacted with others by DNN in the sample. Thus, we obtain a set of feasible features that works for feature crossing in each sample. (3) Then, we generate a global candidate set containing both second-order and higher-order cross feature fields. (4) Based on the candidate set, we can train a simple LR model, therefore rank and select useful feature fields according to their contribution measured on the validation set. (5) Finally, we can obtain the set of useful cross feature fields, as well as a LR model empowered with cross features.

In this paper, we conduct experiments on five public datasets, as well as two real-world datasets. According to the experimental results, a simple LR model empowered with the final set of cross feature fields achieves better performances comparing with DNN, as well as some state-of-the-art feature crossing methods. The experiments also shows, to achieve this, a candidate set with only $2N$ or $3N$ cross feature fields is enough for searching, where N is the number of original feature fields in each dataset. To be noted, this is extremely small compared to the whole set of second-order and higher-order cross feature fields, especially when N is large. With such a compact candidate set, the searching for final useful cross feature fields can be efficiently done. In a word, via DNN2LR, we obtain a powerful and globally interpretable in an efficient way.

The main contributions of this paper are summarized as follows:

- We give defination of the interpretation inconsistency in DNN, and accordingly propose a novel DNN2LR method. Our proposed DNN2LR method can generate a compact and accurate candidate set of cross feature fieldsp, with relatively small amount compared to the whole set of second-order and higher-order cross feature fields.
- Useful cross feature fields can be directly ranked and selected based on our compact candidate set and corresponding contribution in a LR model. The whole process of learning feature crossing can be done via simply training a DNN model and a LR model.
- Extensive experiments have been conducted on several datasets. The final model, i.e., a LR model empowered with cross features, generated by DNN2LR can outperform the complex DNN model, as well as the state-of-the-art feature crossing methods, i.e., AutoCross [7] and AutoFM [21]. The high efficiency of DNN2LR is strongly verified, especially on real-world datasets with large numbers of feature fields.

The rest of the paper is organized as follows. In section 2, we first summarize some works on feature crossing and interpretability. Section 3 gives brief analysis about why and how we learn feature crossing from interpretations of DNN. Section 4 details our proposed DNN2LR method. In section 5p, we report and discuss our experimental results. And finally, section 6 concludes our work.

2 Related Work

In this section, we review four kinds of feature crossing methods: searching-based, tree-based, factorization-based and deep learning-based. We also review some works on interpretability of DNN.

2.1 Searching-based Feature Crossing

It is a direct way to search for useful cross feature fields in a candidate set. However, the candidate set for searching is usually inevitably large, which leads to low efficiency of learning feature crossing. Most searching-based methods focus on generating second-order features [6], [12], [13], [36], [37], [38]. Some of searching-based methods investigates interactions among numerical features [13], [14], [36], [37], [38], [39], [40], [41], which can be applied in limited scenarios. And other methods focus on discrete features [6], [7], [12], which have been proved able to be applied on all kinds of features, and promote the prediction performances [6], [9]. In [6], the authors try to generate and select second-order cross feature fields according to Conditional Mutual Information (CMI). However, once the mutual information of an original feature field is high, the generated cross feature fields containing it will also have high conditional mutual information. AutoLearn [38] selects cross feature fields by using regularized regression models, where it is hard to learn a regression model for all the cross feature fields on a wide dataset. Some works [13], [37] takes meta-learning into consideration for feature generation. However, the effectiveness of meta-learning in these methods remains a question and requires extremely large amount of data. There are also some methods incorporating genetic algorithm [40] and reinforcement learning [14], [41] for finding feature combinations. However, with genetic algorithm or reinforcement learning, we still have a large space to explore. Moreover, as introduced in [14], it also requires large amount of data for the training of reinforcement learning.

To tackle with above problems, AutoCross [7] presents a framework to search in the large candidate set more efficiently, which is a greedy and approximate alternative: (1) AutoCross iteratively searches in a set of cross feature fields, where the set is initialized as all the second-order feature fields, and the selected cross feature field is greedily used to generate new high-order cross feature fields in the next iteration. (2) AutoCross uniformly divides the dataset into at least $\sum_{i=0}^{\lceil \log_2 t \rceil -1} 2^i$ batches, where t is the size of candidate set in AutoCross, and iteratively train a field-wise LR model on part of the data to validate the contribution of a cross feature field. The authors apply the generated cross features, containing both second- and high-order cross features, in a LR model, and it achieves approximate or even better performances comparing with the complex DNN model on 10 datasets. However, AutoCross still searches for useful cross feature fields in a large candidate set, whose size shows exponential relation with the number of the original feature fields. For example, when the size of original feature fields is 10, the number of second-order cross feature fields is 45. And when the size of original feature fields becomes 100, 200, 500 and 1000, the number of second-order cross feature fields becomes 4950, 19900, 124750 and 499500 respectively. When the candidate set is large, the searching efficiency will still be low. Moreover, when the candidate set is large, data for training the field-wise LR model of a candidate cross feature field will be too little to produce reliable results. Therefore, the results of AutoCross are with some randomness, and it is dubious for AutoCross to achieve powerful performances, especially on some wide tabular datasets.

2.2 Tree-based Feature Crossing

Some works [15], [16] utilize tree ensemble model, e.g., GBDT (Gradient Boosting Decision Tree) [4] and XGBoost [3], for generating cross features. In [15], each tree in the GBDT model corresponds to a cross feature field, and the leaf node in a tree corresponds to a cross feature. As pointed in [17], it is hard for tree ensemble models to well handle sparse discrete features, which are essential in real-world applications. This constrains the effectiveness and application scenarios of tree-based feature crossing methods.

2.3 Factorization-based Feature Crossing

p As an extended method of Matrix Factorization (MF), Factorization Machine (FM) [18], [42] has been a successful way to capture second-order feature interactions. Field-aware FM (FFM) [43] incorporates field-aware interactions between different feature fields. To overcome the problem that conventional FM can only model second-order feature interactions, Higher-Order FM (HOFM) [20] proposes to model higher-order feature interactions in the FM architecture, and a linear-time algorithm is presented. Besides, FM has recently been extended via combining with DNN architectures, e.g., Neural Factorization Machines (NFM) [44] and DeepFM [24].

The calculation of feature interactions in FM is somehow product-based similarity. This may be suitable for some kinds of feature interactions, e.g., the matching and correlating in the scenario of recommendation [45]. However, it is hard for these factorization-based methods to capture all kinds of feature interactions in various real-world application scenarios. Another drawback of FM is that, it models all feature interactions of specific-order, most of which are not useful for making predictions. To deal with this problem, AutoFM [21] directly learns the weight for each cross feature field, for both second-order and higher-order. It promotes the performances of FM, but faces the problem of high computational cost, especially when the dataset is wide and the desired interaction order is high.

2.4 Deep Learning-based Feature Crossing

Nowadays, deep learning-based prediction methods have shown their effectiveness. Among these methods, some try to generate and represent cross features via designing various deep learning-based crossing modules. For better performances, these crossing modules are usually applied

TABLE 1
The values of interpretation inconsistency in DNN of different features associated with two feature fields, where $\alpha \in \{0, 1\}$ and $\beta \in \{0, 1\}$, on four toy datasets, characterizing logical operations AND, OR, XNOR and XOR respectively. Large interpretation inconsistency values in DNN give hints about feature crossing.

sample		AND		OR		XNOR		XOR	
α	β	α	β	α	β	α	β	α	β
0	0	0.0002	0.0002	0.0001	0.0000	0.0105	0.0152	0.0137	0.0103
0	1	0.0006	0.0004	0.0003	0.0005	0.0000	0.0003	0.0411	0.0192
1	0	0.0000	0.0002	0.0002	0.0000	0.0126	0.0145	0.0189	0.0103
1	1	0.0000	0.0001	0.0002	0.0002	0.0308	0.0161	0.0190	0.0065

along with DNN architectures. The Wide & Deep model [8] directly learns parameters of manually designed cross features in the wide component. The Product-based Neural Network (PNN) [22] applies inner-product or outer-product to capture second-order features. In Deep & Cross Network (DCN) [23], the authors design an incremental crossing module, named CrossNet, to capture second-order as well as higher-order features. AutoINT [26] is a combination of residual connections [46] and a crossing module based on Self-attention [47]. With the residual connections, the DNN architecture is performed with the input of original features. In xDeepFM [25], Compressed Interaction Network (CIN) is proposed as a crossing module based on outer-product calculation of features, and it is performed together with DNN. Meanwhile, convolutional neural network is also incorporated for modeling local interactions among nearby features [48], [49], [50].

Inherit from deep learning technologies, above feature crossing modules are mostly associated with hidden projection, in each layer or between adjacent layers. This happens in most deep learning-based methods, including several state-of-the-art crossing modules, e.g., CrossNet [23], CIN [25] and Self-attention [26]. That is to say, most of deep learning-based methods are not globally interpretable, and are hard to explicitly generate enough useful cross features.

2.5 Interpretability of DNN

Recently, the interpretability of DNN has drawn great attention in academia, and research works mostly focus on local piece-wise interpretability, which means assigning a piece of local interpretation for each sample [1]. Some unified approaches are proposed to fit a linear classifier in each local space of input samples [34], [51]. Some works investigate the gradients from the final predictions to the input features in deep models, which can be applied in the visualization of deep vision models [28], [29], [31], [52], as well as the interpretation of language models [30], [32]. Perturbation on input features is also utilized to find local interpretations of both vision models [53] and language models [54]. Meanwhile, via adversarial diagnosis of neural networks, adversarial examples can also be introduced for local interpretation of DNN [55], [56]. In some views, attention in deep models can also be regarded as local interpretations [57], [58]. As discussed in some works [34], [51], the nonlinear DNN model can be regarded as a combination of numbers of linear classifiers, and the upper bound of the number in DNN with piece-wise linear activations has been given [33]. Moreover, piece-wise linear DNN has been exactly and consistently interpreted as a set of linear classifiers [35].

3 Learning from Interpretations of DNN

The widely-used DNN model has shown to be capable of capturing various feature interactions in its hidden layers [2], [22], [24]. This means, if we can explicitly find the feature interactions in DNN and use them as cross features in LR, we can take advantage of the strong expressive ability of DNN and the global interpretability of LR. Recently, extensive works have been done to study local piece-wise interpretability of DNN, which means a DNN model can be regarded as a combination of numbers of linear classifiers [33], [34], [35]. And this process can be done via the gradient backpropagation from the prediction layer to the input features [27], [28], [29], [30], [31], [32]. Therefore, we can first define the local interpretation in DNN.

Definition 1. (Local Interpretation) Given a specific feature $x_{k,f}$ associated with the feature field f in a specific sample k, the corresponding local interpretation is

$$I^{l}_{k,f} = w_{k,f}\, e^{\top}_{k,f}, \tag{2}$$

where $e_{k,f}$ denotes the corresponding feature embeddings of $x_{k,f}$ in the DNN model, and $w_{k,f}$ is the local weights computed via gradient backpropagation

$$w_{k,f} = \frac{\partial\, \hat{y}_k}{\partial\, e_{k,f}}, \tag{3}$$

where $\hat{y}_k$ denotes the prediction made by the DNN model for the sample k.

To be noted, local interpretation refers to the contribution of the corresponding feature to the final prediction in the corresponding sample. Usually, local interpretations of a specific feature are inconsistent in different samples. This is caused by feature interactions in the hidden layers of DNN. When local interpretations are consistent in different samples, it means the corresponding feature contributes to the final prediction on its own. When local interpretations are inconsistent, it means the contribution of the corresponding feature is affected by other features. To measure the degree of inconsistency among local interpretations of a specific feature in different samples, we first need to calculate its global interpretation, and then calculate the interpretation inconsistency between the local interpretation and the global interpretation.

Definition 2. (Global Interpretation) Given the feature $x_{k,f}$, the corresponding global interpretation is

$$I^{g}_{k,f} = \bar{w}_f\, e^{\top}_{k,f}, \tag{4}$$

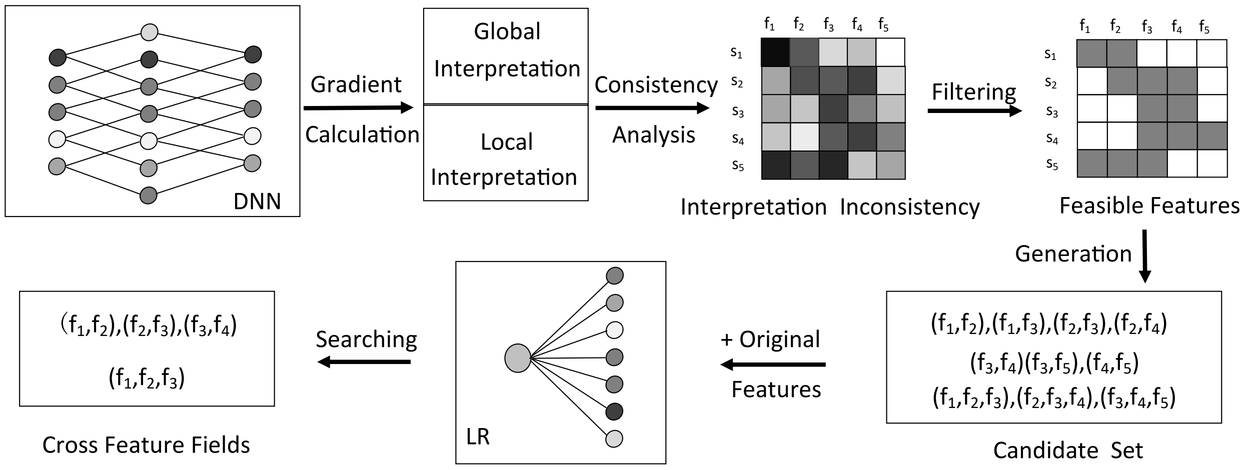

Fig. 2. Overview of our proposed DNN2LR approach, where s_i and f_j indicate the i-th sample in the dataset and the j-th original feature field respectively.

where $\bar{w}_f$ is the average local weights of all features associated with the feature field f in all samples, named as global weights and formulated as

$$\bar{w}_f = \frac{1}{\|\Omega\|} \sum_{k \in \Omega} w_{k,f}, \quad (5)$$

where Ω is the set of samples.

Definition 3. (Interpretation Inconsistency) Given a specific feature $x_{k,f}$ in a specific sample k, the corresponding interpretation inconsistency is

$$d_{k,f} = \left\| \left(w_{k,f} - \bar{w}_f \right) e_{k,f}^{\top} \right\|. \quad (6)$$

For the calculation of interpretation inconsistency, we adopt absolute difference, instead of relative difference. This is because, the values of interpretations indicate the contribution to the final predictions, and features with little contribution are hard to produce useful feature interactions.

The interpretation inconsistency in DNN is able to lead us to generate a compact and accurate candidate set of cross feature fields. We can automatically learn feature crossing based on Assum. 1.

Assumption 1. Larger the values of interpretation inconsistency of a specific feature, more the corresponding feature field can work for feature crossing.

To verify Assum. 1, we conduct empirical experiments on four toy datasets. The four datasets characterize four different logical operations: AND, OR, XNOR and XOR. And we have two input feature fields, where $\alpha \in \{0, 1\}$ and $\beta \in \{0, 1\}$. Thus, for each toy dataset, we have four different samples, and the corresponding labels are in $\{0, 1\}$. As we know, the logical operations AND and OR are easy and linearly separable, therefore no cross features are needed. In contrast, the logical operations XNOR and XOR are not linearly separable, therefore second-order cross feature field consisting of α and β should be generated. We train a DNN model on each of the four toy datasets until convergence, where the Area Under Curve (AUC) evaluation becomes 1.0. On all datasets, the gradients from the prediction layer to the feature layer is computed, and the interpretation inconsistency is obtained, as shown in Tab. 1. It is clear that, interpretation inconsistency values on AND and OR are extremely small, while those on XNOR and XOR are relatively large. According to Assum. 1, α and β should be crossed on XNOR and XOR, while should not on AND and OR. Considering it is easy for linear classifiers to capture logical operations AND and OR, while hard to capture logical operations XNOR and XOR, the experimental results strongly support Assum. 1. In a word, it is proper to learn feature crossing from the interpretation inconsistency in DNN.

4 DNN2LR

In this section, we formally propose the DNN2LR approach. In general, DNN2LR consists of two steps: (1) generating a compact and accurate candidate set of cross feature fields; (2) searching in the candidate set for the final cross feature fields. Specifically, we use $c_i = (f_1, f_2, ..., f_n)$ to denote a specific cross feature field generated by crossing original fields $f_1, f_2, ..., f_n$. Fig. 2 provides an overview of the proposed DNN2LR approach.

4.1 Candidate Set Generation

As we rely on the local piece-wise interpretation of DNN to generate the compact and accurate candidate set of cross features, we first need to train a DNN model. The input of DNN is the original features, which are sparse high dimensional vectors. Thus, we use an embedding layer [2] to transform the input features into low dimensional dense representations. Then, the dense representations are passed through some linear transformation and nonlinear activation to obtain the predictions of samples, where we use ReLu as the activation for hidden layers, and sigmoid for the output layer to support binary classification tasks. The training of DNN is paralleled and accelerated based on the Parameter Server (PS) architecture [59].

Based on the trained DNN model, in each validation sample k, we compute the interpretation inconsistency $d_{k,f}$

of the feature $x_{k,f}$, as defined in Def. 3. Therefore, we obtain an interpretation inconsistency matrix D, where $D[k,f]$ is the interpretation inconsistency value of the f-th feature field in the k-th sample. Then, we conduct an element-wise filtering on matrix D with a threshold η, which can be formulated as

$$D^*[k,f] = \begin{cases} 1, & D[k,f] \geq Quantile(D, 1-\eta) \\ 0, & otherwise \end{cases}, \quad (7)$$

where D^* is a binary feasible feature matrix, and $D^*[k,f]$ indicates whether the f-th feature in the k-th sample has interacted with other features in the hidden layers. $Quantile(D, 1-\eta)$ denotes the $1-\eta$ quantile of the matrix D, which means keeping the top η elements ($0\% < \eta < 100\%$) in D with largest values of interpretation inconsistency. Then, for each feasible feature $D^*[k,f] = 1$, the corresponding feature field f can be used to generate candidate cross feature fields.

Finally, we greedily generate the candidate set of cross features fields. Considering that extremely high-order cross features are rarely useful, we construct our candidate set with only 2nd-order, 3rd-order and 4th-order cross feature fields. To construct a compact and accurate candidate set, we need to find cross feature fields which are most frequently interacted in DNN. According to the feasible feature matrix D^*, we count the occurrences of possible cross feature fields. We then rank the cross feature fields in a descending order according to the corresponding occurrence frequency, and select the top ε cross feature fields as our candidate set. According to our experiments, to make LR achieve better performances than the complex DNN model, only $\varepsilon = 2N$ or $\varepsilon = 3N$ is needed, where N is the size of the original feature fields. To be noted, in general, this is extremely small comparing with the size of the entire set of 2nd-order, 3rd-order and 4th-order cross feature fields. For example, when $N = 100$, the size of the corresponding entire set of cross feature fields will be $4,087,875$ ($4,950 + 161,700 + 3,921,225$). Accordingly, a compact candidate set of cross feature fields is generated, and efficiently searching for useful cross feature fields can be conducted.

4.2 Searching for Final Cross Feature Fields

After obtaining the candidate set of cross feature fields $S = \{c_1, c_2, ..., c_\varepsilon\}$, we can search for final useful cross feature fields. Following the idea in [7], the searching of final cross feature fields is based on their contribution measured on the validation set. Meanwhile, we do not need to involve the complex searching structure in [7], because of our compact and accurate candidate set.

To search for useful cross feature fields, we need to train a sparse LR model. The input of the LR model consists of both original features and candidate cross features. We use S as the schema to process the training set to generate corresponding candidate cross features, which are denoted as X_{train}^{cross}, with ε cross feature fields and M_{train} samples. The original features of training set are denoted as $X_{train}^{original}$, with N feature fields and M_{train} samples. Accordingly, the number of all input feature fields for the LR model is $N + \varepsilon$. As mentioned above, $\varepsilon = 2N$ or $\varepsilon = 3N$ is enough according to our experiments. Therefore, the time cost of training the LR model with both original and cross features is in the same order of magnitude with directly training a LR model with only original features. Moreover, similar with the training of DNN, the training of the LR model is also paralleled and accelerated with the PS architecture [59].

Algorithm 1 Searching for Final Cross Feature Fields.

```
Require: candidate set $S = \{c_1, c_2, ..., c_\varepsilon\}$, original features $X_{valid}^{original}$ with $N$ feature fields and $M_{valid}$ samples, candidate cross features $X_{valid}^{cross}$ with $\varepsilon$ cross feature fields and $M_{valid}$ samples, labels $Y_{valid}$ with $M_{valid}$ samples, model weights $W$ of LR trained with both original and cross features on training set, lookup function LOOKUP() for one-dimensional embeddings in the sparse LR model, sigmoid function $\gamma()$ and AUC computing function $compute\_auc()$.
Ensure: final set of cross feature fields $S^*$.
 1: $S^* = \{\}$;
 2: $E_{valid}^{original} = \mathrm{LOOKUP}\left(\mathrm{X}_{\mathrm{valid}}^{\mathrm{original}}\right)$;
 3: $E_{valid}^{cross} = \mathrm{LOOKUP}\left(\mathrm{X}_{\mathrm{valid}}^{\mathrm{cross}}\right)$;
 4: $b(-1) = E_{valid}^{original}\, W[0:N]$;
 5: $AUC(-1) = compute\_auc(Y_{valid}, \gamma(b(-1)))$;
 6: for $i$ in $[0, \varepsilon)$ do
 7:     for $j$ in $[0, \varepsilon)$ do
 8:         if $c_j$ not in $S^*$ then
 9:             $b(j) = b(-1) + E_{valid}^{cross}[:, j]\, W[N+j]$;
10:             $AUC(j) = compute\_auc(Y_{valid}, \gamma(b(j)))$;
11:         end if
12:     end for
13:     $k = \mathrm{argmax}_j\, AUC(j)$;
14:     if $AUC(k) > AUC(-1)$ then
15:         $c_k \to S^*$;
16:         $b(-1) = b(k)$;
17:         $AUC(-1) = AUC(k)$;
18:     else
19:         break;
20:     end if
21: end for
22: return $S^*$.
```

Based on the model weights W in the trained LR model, we conduct the searching procedure for useful cross feature fields according to their contribution measured on the validation set. Here, $X_{valid}^{original}$ and X_{valid}^{cross} denote original features and candidate cross features on the validation set respectively. Pseudocode of the searching procedure is presented in Alg. 1. Moreover, the steps 7-12 in Alg. 1 are paralleled with multi-threading implementation, where the measuring of each candidate cross feature field is conducted on one thread.

The main idea in Alg. 1 is that, based on parameters in the trained LR model, we measure the contribution in AUC of each candidate cross feature field on the validation set, and select those have positive contribution as the final set S^* of cross feature fields. This searching procedure is easy and effective. The reason why we can use such an easy searching strategy is that, the DNN2LR approach can generate a compact and accurate candidate set of cross feature fields according to the interpretation inconsistency

TABLE 2
Summarization of the datasets.

dataset	#samples		#feature fields		width
	training	testing	#Num.	#Cate.	
Employee	29,494	3,277	0	9	narrow
Adult	32,562	16,282	6	8	narrow
Criteo	41.256M	4.584M	13	26	medium
Movielens	517,310	221,704	109	20	wide
Allstate	131,823	56,497	15	115	wide
RW1	233,123	58,282	178	15	wide
RW2	151,236	52,169	302	56	wide

in DNN, and the searching procedure can greatly benefit from this.

5 Experiments

In this section, we empirically evaluate our proposed DNN2LR approach. We first describe settings of the experiments, then report and analyze the experimental results. Thorough evaluations are conducted to answer the following research questions:

- **RQ1** Can DNN2LR empower the simple LR model achieving good performances?
- **RQ2** How is the efficiency of DNN2LR for feature crossing, especially on datasets with large numbers of feature fields?
- **RQ3** How many cross feature fields can be automatically generated?
- **RQ4** How is the stability of the DNN2LR method?
- **RQ5** How is the interpretability of the final model, i.e., a LR model empowered with cross features, generated by DNN2LR?

5.1 Experimental Datasets

We evaluate the proposed DNN2LR method on 5 public datasets, i.e., Employee[1], Adult[2], Movielens[3], Criteo[4] and Allstate[5]. Meanwhile, we also conduct experiments on 2 real-world datasets, i.e., RW1 and RW2, after anonymization and sanitization. More details about these datasets can be found in Tab. 2. These datasets have one thing in common: DNN can outperform LR on these datasets, which means there are cross features to find. According to the numbers of feature fields in Tab. 2, we can conclude three kinds of datasets: narrow datasets, medium datasets and wide datasets. For the splitting of Employee, Adult and Criteo, we follow [7]. For Movielens and Allstate, we use the first 70% data as the training set, and the last 30% data as the testing set. Moreover, on each dataset, we use 20% of the training samples for validation.

The tasks in Employee, Adult and Criteo are binary classification. Meanwhile, we transform the regression tasks in Allstate and Movielens as binary classification tasks with settled thresholds. On Allstate, labels less than 2550 are set as negative, and positive otherwise. On Movielens, labels equal to 4 or 5 are set as positive, and labels equal to 1 or 2 are set as negative. The task in Movielens becomes the classification of whether a user likes a target movie according to the user's rating history on movies. Moreover, we conduct some feature engineering, and design more than 100 handcrafted feature based on the original Movielens datasets. On Movielens, we regard reviews as behaviors, and sort the reviews from one user in order of time. We extract a series of statistical features based on each user's historical behavior. Features consist of three types: user meta features, movie meta features and user history features. User meta features include *user_age*, *user_gender* and *user_occupation*. Movie meta feature include *is_cate*, where cate $\in$ {action, adventure, animation, children's, comedy, crime, documentary, drama, fantasy, film-noir, horror, musical, mystery, romance, sci-fi, thriller, war, western}. And the feature *is_cate* indicates the category of the target movie. The user history features include *recent_k_cate_mean_score* and *recent_k_cate_cnt*, where $k \in \{10, 30, 50\}$. And the feature *recent_k_cate_cnt* is the count of the corresponding category that the user has reviewed in recent k times, while the feature *recent_k_cate_mean_score* is the mean score of the corresponding category that the user has reviewed in recent k times.

1. https://www.kaggle.com/c/amazon-employee-access-challenge/data
2. https://archive.ics.uci.edu/ml/datasets/adult
3. https://grouplens.org/datasets/movielens/1m/
4. https://www.kaggle.com/c/criteo-display-ad-challenge/data
5. https://www.kaggle.com/c/allstate-claims-severity/data

5.2 Experimental Settings

In our experiments, we are going to verify whether we can obtain powerful and globally interpretable models. Specifically, we are going to verify whether DNN2LR can empower the simple LR model to achieve better performances comparing with the complex DNN model, as well as other competitive feature crossing methods. Accordingly, we give comparison among several types of methods: baselines, factorization-based methods, deep learning-based methods, tree-based methods, searching-based methods and interpretation-inspired methods. We select some representative methods in each type and introduce them as follows.

Baselines: We incorporate sparse **LR** and **DNN** as baselines. LR is a simple linear model, while DNN is a complex nonlinear model. For LR, we tune the learning rate in the range of $\{0.005, 0.01, 0.05, 0.1, 0.5, 1.0\}$, and the l2 regularization in the range of $\{0.0001, 0.001, 0.01, 0.1, 1.0\}$. For DNN, we set the dimensionality of feature embeddings as 10, the learning rate as 0.001, the l2 regularization as 0.0001, and use Adam [60] for optimization. We use the commonly-applied ReLu as activation function in hidden layers of DNN. The hidden components in deep layers of DNN are set as $[400, 200]$ on Criteo, and $[400, 100]$ on other datasets.

Factorization-based: We include **FM** [18], **HOFM** [20] and **AutoFM** [21]. FM is a successive method for modeling second-order feature interactions, and HOFM extends FM with higher-order feature interactions. AutoFM directly learns the weight for each cross feature field, for both 2nd-order and higher-order. We set the dimensionality of embeddings as 10, the learning rate as 0.001, the l2 regularization as 0.0001. We generate up to 4th-order cross features with HOFM. The efficiency of AutoFM is low when datasets are wide and desired interaction order is high. Thus, for

TABLE 3
Performance comparison among different baselines and feature crossing methods in terms of AUC (%). We highlight methods those can outperform DNN on each dataset. Clearly, DNN2LR is the only method that can constantly outperform DNN on all datasets. $*$ and $**$ denote statistically significant improvement, measured by t-test with p-value< 0.05 and p-value< 0.01 respectively, over the most competitive compared method on each dataset.

type	method	Employee	Adult	Criteo	Movielens	Allstate	RW1	RW2
baselines	LR	86.75	92.10	78.51	81.36	86.10	72.36	78.16
	DNN	87.85	92.68	79.95	85.86	86.60	74.60	79.68
factorization-based	FM	86.89	91.71	79.27	85.12	86.41	72.54	78.92
	HOFM	86.96	91.67	79.72	85.51	86.48	72.78	79.21
	AutoFM	87.32	92.18	**80.08**	**86.02**	86.46	73.08	79.32
deep learning-based	CrossNet	87.21	91.97	79.57	85.10	86.38	73.16	79.07
	CIN	**87.89**	92.21	**80.05**	**85.92**	86.49	73.49	79.38
	Self-attention	**88.33**	92.35	79.89	85.71	86.53	73.74	79.51
tree-based	GBDT	87.21	92.57	79.76	85.67	86.19	**74.77**	79.46
	GBDT+LR	87.29	92.51	79.81	85.78	86.32	**75.06**	79.52
serching-based	CMI+LR	**89.12**	91.61	78.52	82.16	86.35	73.11	78.81
	AutoCross+LR	**89.42**	**92.80**	**80.31**	84.33	86.41	72.67	78.29
interpretation-inspired	DNN2LR	**89.66***	**92.96****	**80.48****	**86.25****	**86.72****	**76.23****	**79.91****

efficiency, with AutoFM, we generate up to 4th-order cross features on narrow datasets, 3rd-order cross features on medium datasets and 2nd-order cross features on wide datasets.

Deep learning-based: We select **CrossNet** [23], **CIN** [25] and **Self-attention** [26], [47]. CrossNet, CIN and Self-attention are the crossing modules designed and applied in DCN [23], xDeepFM [25] and AutoINT [26] respectively. For those overlapping settings, we stay the same with above settings of DNN. Moreover, for CrossNet, we have 3 layers of feature crossing. For CIN, we have 3 layers of feature crossing, and the number of feature maps in the crossing module stays the same with the number of original feature fields. For Self-attention, we have 3 layers of feature crossing, where the number of hidden units in self-attention and the number of attention heads are set as 32 and 2 respectively. These settings stay the same as in corresponding previous works, and 3 layers of feature crossing means conducting up to 4th-order crossing.

Tree-based: We include the **GBDT** classifier, as well as the corresponding feature crossing method **GBDT+LR** [15]. Each tree in the GBDT model corresponds to a cross feature field in GBDT+LR. For fair comparison, we set the number of trees in GBDT as the same as the number of cross feature fields generated by DNN2LR on each dataset.

Searching-based: We incorporate **CMI** [6] and **AutoCross** [7]. CMI searches for second-order cross feature fields based on conditional mutual information. Following [7], we estimate CMI with all samples in the training set. And on the Criteo dataset, to conduct computation in a reasonable time, we sub-sample 10% of the data for estimating CMI. AutoCross is the state-of-the-art method for feature crossing, which can generate both second-order and higher-order cross feature fields. The major setting of AutoCross is how many batches of data are divided for learning the contribution of each candidate cross feature field. As in [7], on Employee and Adult, there are $2\sum_{i=0}^{\lceil \log_2 t\rceil-1} 2^i$ batches of data, where t is the size of candidate set in AutoCross. Meanwhile, on Criteo, there are $5\sum_{i=0}^{\lceil \log_2 t\rceil-1} 2^i$ batches of data. For other datasets, i.e., Allstate, Movielens, RW1 and RW2, considering these datasets are relatively wide, we have $\sum_{i=0}^{\lceil \log_2 t\rceil-1} 2^i$ batches of data. The terminal condition of searching in AutoCross is set as, when newly added cross feature field leads to a performance degradation, the searching procedure stops. Moreover, the cross features generated by CMI and AutoCross are fed into LR, and the corresponding performances are reported as **CMI+LR** and **AutoCross+LR** respectively. The LR models used in these methods are set as above settings.

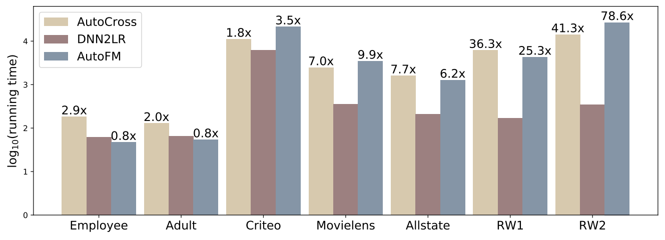

Fig. 3. The running time (seconds) of AutoCross, AutoFM and DNN2LR after the log_{10} calculation on each dataset. The ratios of running time of AutoCross and AutoFM to that of DNN2LR are respectively shown above the corresponding bars.

Interpretation-inspired: We include our proposed **DNN2LR** approach. The quantile threshold η for filtering feasible features in the interpretation inconsistency matrix is tuned in the range of $\{10\%, 3.3\%, 1\%, 0.33\%, 0.1\%\}$, and the size ε of candidate set is tuned in the range of $\{N, 2N, 3N, 4N, 5N\}$, where N is the size of the original feature fields. To verify the stability, we run DNN2LR 10 times with different parameter initialization. The LR models and DNN models used in DNN2LR are set as above settings.

Our experiments are conducted on a machine with Intel(R) Xeon(R) CPU (E5-26p30 v4 @ 2.20GHz, 24 cores) and 256G memory. Moreover, we apply multi-granularity discretization [7] on numerical features. The granularities in this method are set as $\{10^p\}_{p=1}^3$. The resulting features are utilized in all our compared methods on all datasets.

5.3 Performance comparison

Performance comparison is shown in Tab. 3, where we have $\eta = 3.3\%$ and $\varepsilon = 3N$ in DNN2LR as the optimal hyper-parameters. On most datasets, different kinds of feature crossing methods can achieve somehow improvements on the performances of LR. AutoFM performs better than FM and HOFM, for it can select partial useful cross features instead of using all possible cross features. As shown in the experimental results, AutoFM outperforms DNN on Criteo

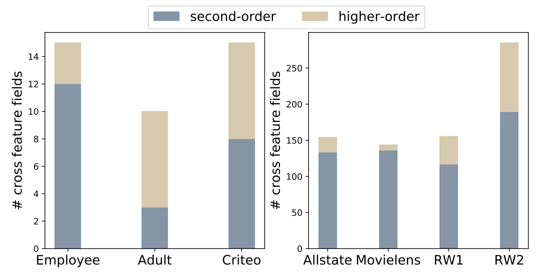

Fig. 4. The count of second-order and higher-order cross feature fields generated by DNN2LR on each dataset.

and Movielens, but can not achieve good performances on other datasets. This may indicate that, the product-based similarity calculation in factorization-based methods is not suitable for all kinds of feature interactions. Among the three deep learning-based methods, CrossNet performs relatively poor. CIN and Self-attention achieve relatively good performances on Criteo and Movielens, but can not constantly performs well on all datasets. Nevertheless, as discussed in Sec. 2.4, these deep learning-based methods are not globally interpretable, and can not explicitly generate cross features. Similarly, GBDT+LR performs well on Employee and RW1, but fails on other datasets. CMI+LR, as a searching-based method, has relatively poor overall performances. This may caused by the fact that the CMI metric can not guarantee the effectiveness of generated cross features. Meanwhile, AutoCross+LR outperforms DNN on Employee, Adult and Criteo, but performs poor on wide datasets. This is because that, on wide datasets with too many feature fields, data for training the field-wise LR model for evaluating a candidate cross feature field will be too little to produce reliable results. This causes the failure of AutoCross on wide datasets. Clearly, DNN2LR is the only method that can constantly achieve powerful performances on all datasets. Moreover, we have conducted significance test, which indicates that DNN2LR stably outperforms DNN, as well as other feature crossing methods, on both public and real-world datasets. As pointed in previous works [7], [8], [24], improvement in AUC can lead to great commercial benefits. In a word, the effectiveness of DNN2LR is strongly verified by results on 5 public datasets, as well as 2 real-world datasets.

5.4 Running time for feature crossing

For the efficiency comparison, we involve the two most competitive compared methods, AutoCross and AutoFM. Fig. 3 shows the running time of AutoCross, AutoFM and DNN2LR for generating cross feature fields, which is illustrated after the log_{10} calculation. We can clearly observe that, DNN2LR can perform faster than AutoCross and AutoFM, especially on wide datasets. Compared with AutoCross, on Employee, Adult and Criteo, DNN2LR is about $1.0\times$ to $2.0\times$ faster. On wide datasets, DNN2LR performs much faster than AutoCross. Specifically, on Movielens, Allstate, RW1 and RW2, DNN2LR's ratios of speedup on AutoCross are about $6.0\times$, $6.7\times$, $35.3\times$ and $40.3\times$ respectively. Meanwhile, the speed of AutoFM and that of DNN2LR are similar on Employee and Adult. On Criteo,

TABLE 4
Standard deviation in terms of AUC (%) of DNN2LR on each dataset. We run DNN2LR 10 times with different random parameter initialization.

Employee	Adult	Criteo	Movielens	Allstate	RW1	RW2
0.09	0.03	0.03	0.04	0.05	0.09	0.03

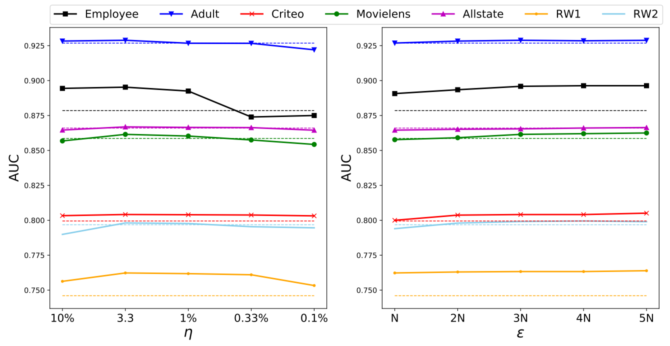

Fig. 5. Performances of DNN2LR with varying hyper-parameters: (1) the left part shows the impact of the quantile threshold η for filtering feasible features in the interpretation inconsistency matrix; (2) the right part shows the impact of the size ε of candidate set. The dashed lines illustrate the performances of DNN on different datasets.

DNN2LR is about $2.5\times$ faster than AutoFM. Similar with AutoCross, AutoFM runs much slower than DNN2LR on wide datasets. Specifically, on Movielens, Allstate, RW1 and RW2, DNN2LR's ratios of speedup on AutoFM are about $8.9\times$, $5.2\times$, $24.3\times$ and $77.6\times$ respectively. Nevertheless, for efficiency, we only generate second-order cross features with AutoFM. It is obvious that, compared with AutoCross and AutoFM, wider the dataset, larger DNN2LR's ratio of speedup. That is to say, AutoCross and AutoFM have problem in efficiency on wide datasets. As for DNN2LR, the whole procedure can be done via simply training a DNN model and a LR model, and the size of candidate set is only $3N$. These results strongly illustrate the high efficiency of DNN2LR for feature crossing, especially on datasets with large numbers of feature fields.

5.5 Count of cross feature fields

As shown in Fig. 4, we illustrate the count of both second-order and higher-order cross feature fields generated by DNN2LR. Overall, on most datasets, we have more second-order cross feature fields than higher-order cross feature fields. The only exception is the Adult dataset. This may indicates that, second-order cross features are more important than higher-order cross features in most cases. Moreover, we can conclude that, more original feature fields we have, more cross feature fields in the final LR model we need.

5.6 Stability analysis

Considering the learned parameters in DNN are with some randomness, we need to verify the stability of the performances of DNN2LR. Accordingly, we run DNN2LR 10 times with different parameter initialization, and illustrate the standard deviation in terms of AUC on each dataset, as shown in Tab. 4. It is clear that, the values of standard deviation are relatively small, and the performances of DNN2LR are relatively stable.

TABLE 5
The five most important cross feature fields generated by DNN2LR on the Movielens dataset.

rank	cross feature field
1	(is_action, recent_50_crime_mean_score)
2	(is_horror, recent_50_action_count)
3	(is_horror, recent_50_thriller_mean_score)
4	(is_children's, recent_30_fantasy_count)
5	(is_action, recent_30_crime_mean_score)

As shown in Fig. 5, we illustrate the sensitivity of hyper-parameters in DNN2LR. First, we set $\varepsilon = 3N$, and investigate the performances of DNN2LR with varying quantile thresholds η for filtering feasible features in the interpretation inconsistency matrix. According to the curves in Fig. 5, the performances of DNN2LR are relatively stable, especially in the range of $\{3.3\%, 1\%, 0.33\%\}$. The only exception is the Employee dataset, for it has only 9 feature fields. If we set η too small, we will have too little feasible features for generating cross features on Employee. Then, we set $\eta = 3.3\%$, and investigate the performances of DNN2LR with varying sizes ε of candidate set. We can observe that, the curves are stable, and when we have larger ε, the performances of DNN2LR slightly increase. On each dataset, with only $\varepsilon = 2N$ or $\varepsilon = 3N$, DNN2LR can outperform the complex DNN model, where N is the size of the original feature fields. This makes the candidate size extremely small, comparing to the while set of possible cross feature fields. According to our observations, we set $\eta = 3.3\%$ and $\varepsilon = 3N$ as the optimal hyper-parameters in other experiments.

5.7 Interpretability

Considering features in Movielens have rich meanings, we conduct experiments to demonstrate the interpretability of DNN2LR on Movielens. The five most important cross feature fields generated by DNN2LR on Movielens are shown in Tab. 5. The importance of a cross feature field is calculated according to the corresponding contribution in the LR model measured on the validation set as in Alg. 1. As we can observe in Tab. 5, these cross feature fields are reasonable and congenial with common sense. For example, the cross feature field (is_action, recent_50_crime_mean_score) indicates crossing between whether the target movie belongs to the category action and the mean score on crime movies the user has rated in the recent 50 behaviours. This cross feature fields is obviously useful for predicting the user's preference, because of the strong connections between action movies and crime movies. Similarly, horror movies and action movies, horror movies and thriller movies, children's movies and fantasy movies, are also respectively related.

6 Conclusion

In this paper, we observe that the local interpretations of DNN are usually inconsistent in different samples, and accordingly define the interpretation inconsistency in DNN. According to the interpretation inconsistency, we propose a novel DNN2LR method. DNN2LR can generate a compact and accurate candidate set of cross feature fields, with extremely small amount compared to the whole set of second-order and higher-order cross feature fields in a dataset. Based on the corresponding contribution in a LR model, useful cross feature fields can be directly ranked and selected from our compact candidate set. The whole process of learning feature crossing can be done via simply training a DNN model and a LR model. Extensive experiments have been conducted on five public datasets, as well as two real-world datasets. Cross features generated by DNN2LR can empower a simple LR model achieving better performances comparing with the complex DNN model, as well as several state-of-the-art feature crossing methods, i.e., AutoCross and AutoFM. The experiments also strongly verify the high efficiency of DNN2LR, especially on real-world datasets with large numbers of feature fields. In a word, with our proposed DNN2LR method, we can obtain powerful and globally interpretable models.

References

[1] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 1–42, 2018.

[2] W. Zhang, T. Du, and J. Wang, "Deep learning over multi-field categorical data," in *European conference on information retrieval*. Springer, 2016, pp. 45–57.

[3] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[4] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in neural information processing systems*, 2017, pp. 3146–3154.

[5] Q. Yao, M. Wang, J. E. Hugo, G. Isabelle, Y.-Q. Hu, Y.-F. Li, W.-W. Tu, Q. Yang, and Y. Yu, "Taking human out of learning applications: A survey on automated machine learning," *arXiv preprint arXiv:1810.13306*, 2018.

[6] O. Chapelle, E. Manavoglu, and R. Rosales, "Simple and scalable response prediction for display advertising," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 4, p. 61, 2015.

[7] Y. Luo, M. Wang, H. Zhou, Q. Yao, W. Tu, Y. Chen, Q. Yang, and W. Dai, "Autocross: Automatic feature crossing for tabular data in real-world applications," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2019, pp. 1936–1945.

[8] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir *et al.*, "Wide & deep learning for recommender systems," in *Proceedings of the 1st workshop on deep learning for recommender systems*. ACM, 2016, pp. 7–10.

[9] H. Liu, F. Hussain, C. L. Tan, and M. Dash, "Discretization: An enabling technique," *Data mining and knowledge discovery*, vol. 6, no. 4, pp. 393–423, 2002.

[10] V. Franc, O. Fikar, K. Bartos, and M. Sofka, "Learning data discretization via convex optimization," *Machine Learning*, vol. 107, no. 2, pp. 333–355, 2018.

[11] Q. Liu, Z. Liu, and H. Zhang, "An empirical study on feature discretization," *arXiv preprint arXiv:2004.12602*, 2020.

[12] R. Rosales, H. Cheng, and E. Manavoglu, "Post-click conversion modeling and analysis for non-guaranteed delivery display advertising," in *Proceedings of the fifth ACM international conference on Web search and data mining*. ACM, 2012, pp. 293–302.

[13] G. Katz, E. C. R. Shin, and D. Song, "Explorekit: Automatic feature generation and selection," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 2016, pp. 979–984.

[14] U. Khurana, H. Samulowitz, and D. Turaga, "Feature engineering for predictive modeling using reinforcement learning," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[15] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers *et al.*, "Practical lessons from predicting clicks on ads at facebook," in *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, 2014, pp. 1–9.

[16] Q. Shi, Y.-L. Zhang, L. Li, X. Yang, M. Li, and J. Zhou, "Safe: Scalable automatic feature engineering framework for industrial tasks," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 2020, pp. 1645–1656.

[17] G. Ke, Z. Xu, J. Zhang, J. Bian, and T.-Y. Liu, "Deepgbm: A deep learning framework distilled by gbdt for online prediction tasks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 384–394.

[18] S. Rendle, "Factorization machines," in *2010 IEEE International Conference on Data Mining*. IEEE, 2010, pp. 995–1000.

[19] Q. Liu, S. Wu, L. Wang *et al.*, "Cot: Contextual operating tensor for context-aware recommender systems." in *AAAI*, 2015, pp. 203–209.

[20] M. Blondel, A. Fujino, N. Ueda, and M. Ishihata, "Higher-order factorization machines," in *Advances in Neural Information Processing Systems*, 2016, pp. 3351–3359.

[21] B. Liu, C. Zhu, G. Li, W. Zhang, J. Lai, R. Tang, X. He, Z. Li, and Y. Yu, "Autofis: Automatic feature interaction selection in factorization models for click-through rate prediction," *arXiv preprint arXiv:2003.11235*, 2020.

[22] Y. Qu, H. Cai, K. Ren, W. Zhang, Y. Yu, Y. Wen, and J. Wang, "Product-based neural networks for user response prediction," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 2016, pp. 1149–1154.

[23] R. Wang, B. Fu, G. Fu, and M. Wang, "Deep & cross network for ad click predictions," in *Proceedings of the ADKDD'17*. ACM, 2017, p. 12.

[24] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "Deepfm: a factorization-machine based neural network for ctr prediction," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017, pp. 1725–1731.

[25] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun, "xdeepfm: Combining explicit and implicit feature interactions for recommender systems," in *Proceedings of the 24th ACM International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 1754–1763.

[26] W. Song, C. Shi, Z. Xiao, Z. Duan, Y. Xu, M. Zhang, and J. Tang, "Autoint: Automatic feature interaction learning via self-attentive neural networks," *arXiv preprint arXiv:1810.11921*, 2018.

[27] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[28] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 618–626.

[29] D. Alvarez-Melis and T. S. Jaakkola, "Towards robust interpretability with self-explaining neural networks," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 7786–7795.

[30] J. Li, X. Chen, E. Hovy, and D. Jurafsky, "Visualizing and understanding neural models in nlp," *arXiv preprint arXiv:1506.01066*, 2015.

[31] D. Smilkov, N. Thorat, B. Kim, F. Viegas, and M. Wattenberg, "Smoothgrad: removing noise by adding noise," in *ICML*, 2017.

[32] H. Yuan, Y. Chen, X. Hu, and S. Ji, "Interpreting deep models for text analysis via optimization and regularization methods," in *AAAI*, 2019.

[33] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Advances in neural information processing systems*, 2014, pp. 2924–2932.

[34] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[35] L. Chu, X. Hu, J. Hu, L. Wang, and J. Pei, "Exact and consistent interpretation for piecewise linear neural networks: A closed form solution," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 1244–1253.

[36] J. M. Kanter and K. Veeramachaneni, "Deep feature synthesis: Towards automating data science endeavors," in *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2015, pp. 1–10.

[37] F. Nargesian, H. Samulowitz, U. Khurana, E. B. Khalil, and D. S. Turaga, "Learning feature engineering for classification," in *IJCAI*, 2017.

[38] A. Kaul, S. Maheshwary, and V. Pudi, "Autolearnłautomated feature generation and selection," in *2017 IEEE International Conference on data mining (ICDM)*. IEEE, 2017, pp. 217–226.

[39] W. Fan, E. Zhong, J. Peng, O. Verscheure, K. Zhang, J. Ren, R. Yan, and Q. Yang, "Generalized and heuristic-free feature construction for improved accuracy," in *Proceedings of the 2010 SIAM International Conference on Data Mining*. SIAM, 2010, pp. 629–640.

[40] B. Tran, B. Xue, and M. Zhang, "Genetic programming for feature construction and selection in classification on high-dimensional data," *Memetic Computing*, vol. 8, no. 1, pp. 3–15, 2016.

[41] X. Chen, Q. Lin, C. Luo, X. Li, H. Zhang, Y. Xu, Y. Dang, K. Sui, X. Zhang, B. Qiao *et al.*, "Neural feature search: A neural architecture for automated feature engineering," in *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2019, pp. 71–80.

[42] S. Rendle, "Factorization machines with libfm," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 3, no. 3, pp. 1–22, 2012.

[43] Y. Juan, Y. Zhuang, W.-S. Chin, and C.-J. Lin, "Field-aware factorization machines for ctr prediction," in *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016, pp. 43–50.

[44] X. He and T.-S. Chua, "Neural factorization machines for sparse predictive analytics," in *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2017, pp. 355–364.

[45] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme, "Fast context-aware recommendations with factorization machines," in *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 2011, pp. 635–644.

[46] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[47] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[48] Q. Liu, F. Yu, S. Wu, and L. Wang, "A convolutional click prediction model," in *Proceedings of the 24th ACM international on conference on information and knowledge management*. ACM, 2015, pp. 1743–1746.

[49] P. P. Chan, X. Hu, L. Zhao, D. S. Yeung, D. Liu, and L. Xiao, "Convolutional neural networks based click-through rate prediction with multiple feature sequences." in *IJCAI*, 2018, pp. 2007–2013.

[50] B. Liu, R. Tang, Y. Chen, J. Yu, H. Guo, and Y. Zhang, "Feature generation by convolutional neural network for click-through rate prediction," in *The World Wide Web Conference*. ACM, 2019, pp. 1119–1129.

[51] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in neural information processing systems*, 2017, pp. 4765–4774.

[52] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929.

[53] R. C. Fong and A. Vedaldi, "Interpretable explanations of black boxes by meaningful perturbation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3429–3437.

[54] C. Guan, X. Wang, Q. Zhang, R. Chen, D. He, and X. Xie, "Towards a deep and unified understanding of deep neural models in nlp," in *International Conference on Machine Learning*, 2019, pp. 2454–2463.

[55] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *ICML*, 2017.

[56] Y. Dong, H. Su, J. Zhu, and F. Bao, "Towards interpretable deep neural networks by leveraging adversarial examples," in *CVPR*, 2018.

[57] R. Ghaeini, X. Z. Fern, and P. Tadepalli, "Interpreting recurrent and attention-based neural models: a case study on natural language inference," *arXiv preprint arXiv:1808.03894*, 2018.

[58] J. Wang, Q. Liu, Z. Liu, and S. Wu, "Towards accurate and interpretable sequential prediction: A cnn & attention-based feature extractor," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 1703–1712.

[59] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*, 2014, pp. 583–598.

[60] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.