This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2903403, IEEE Transactions on Knowledge and Data Engineering

A Generalized Locally Linear Factorization Machine with Supervised Variational Encoding

Xiaoshuang Chen, Yin Zheng, Peilin Zhao, Zhuxi Jiang, Wenye Ma and Junzhou Huang

Abstract—Factorization Machines (FMs) learn weights for feature interactions, and achieve great success in many data mining tasks. Recently, Locally Linear Factorization Machines (LLFMs) have been proposed to capture the underlying structures of data for better performance. However, one obvious drawback of LLFM is that the local coding is only operated in the original feature space, which limits the model to be applied to high-dimensional and sparse data. In this work, we present a generalized LLFM (GLLFM) which overcomes this limitation by modeling the local coding procedure in a latent space. Moreover, a novel Supervised Variational Encoding (SVE) technique is proposed such that the distance can effectively describe the similarity between data points. Specifically, the proposed GLLFM-SVE trains several local FMs in the original space to model the higher order feature interactions effectively, where each FM associates to an anchor point in the latent space induced by SVE. The prediction for a data point is computed by a weighted sum of several local FMs, where the weights are determined by local coding coordinates with anchor points. Actually, GLLFM-SVE is quite flexible and other Neural Network (NN) based FMs can be easily embedded into this framework. Experimental results show that GLLFM-SVE significantly improves the performance of LLFM. By using NN-based FMs as local predictors, our model outperforms all the state-of-the-art methods on large-scale real-world benchmarks with similar number of parameters and comparable training time.

Index Terms—Factorization Machines, Locally Linear Coding, Supervised Variational Encoding

1 Introduction

Factorization machines (FMs) [1], [2] are one of the most popular and effective models to leverage the interactions between features. It models nested feature interactions via a factorized parametrization, and can mimic most factorization models such as SVD++, Pairwise Interaction Tensor Factorization (PITF) and Factorizing Personalized Markov Chain (FPMC) [1]. With the recent and impressive success of deep learning, an increasing number of neural-network-based extensions of FMs have been proposed, such as Neural FM (NFM) [3], attentional FM (AFM) [4], DeepFM [5], Wide&Deep [6], Deep Crossing [7], Deep Cross Network (DCN) [8], etc.

Although FM and its NN-based extensions [3], [4], [5] are shown successful in many prediction tasks, they only consider the high-order information of the input features, and may fail to capture the underlying structures of more complex data. In practice, high-dimensional data usually groups into clusters and lies on lower-dimensional manifolds [9], [10], in which different regions may have different characteristics. As a result, it is insufficient to train a single model over the whole feature space. Recently, Liu [11] proposed Locally Linear FM (LLFM) to handle this problem. which is based on the principle that a nonlinear manifold behaves linearly in its local neighborhood [12], [13]. LLFM trains different parameters in different subspace of the data manifold. The FM parameters at a data point can be approximated by a linear combination of the parameters associated with the nearby anchor points. The weights of the linear combination can be determined by approaches like exponential decay [14], [15] and k*-nearest-neighbor (k*NN) [16].

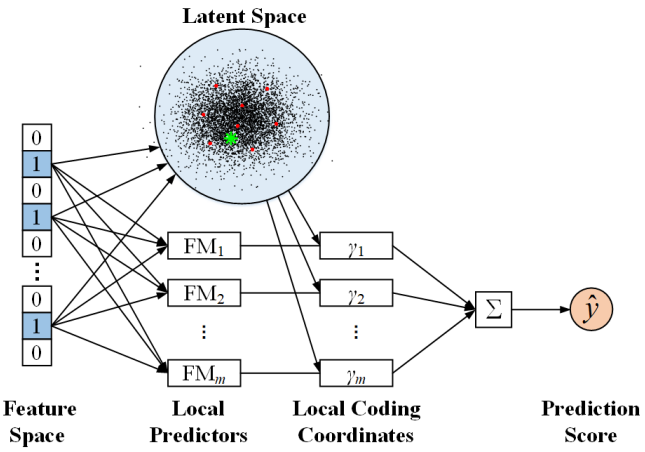

Fig. 1. Framework of GLLFM-SVE. The m local FM models are defined in the original feature space, while the anchor points (red points in the figure) are defined in a latent space. The local coordinates are obtained according to distances between the latent representation of the data point (the green point) and anchor points. The SVE technique encourages the distances in the latent space to express the similarities between data points.

Although LLFM has the potential to improve FMs, it requires the distances between samples and anchor points, i.e. how "close" a sample is to an anchor point, in order to obtain local coordinates. Unfortunately, it is challenging to measure the distances in complex feature spaces, and the Euclidean distance in the original feature space, used by LLFM, is not usually a good measure for complicated

- *Corresponding Author: Yin Zheng (yzheng3xg@gmail.com).*
- *X. Chen is with the Department of Electrical Engineering, Tsinghua University, Beijing, China.*
- *Y. Zheng, W. Ma, P. Zhao and J. Huang are with Tencent AI Lab, Shenzhen, China.*
- *Z. Jiang is with Beijing Institute of Technology, Beijing, China.*

datasets [17], [18]. Different coordinates of the original sample space may be dependent, and may have different impacts on prediction results. Specifically, when the feature vector is highly sparse, the Euclidean distances fail to measure the similarity among samples undoubtedly. This significantly limits the applications of LLFM.

In this paper, a Generalized version of LLFM (GLLFM) is proposed to address the above limitation. Similar to LLFM, the local FMs of GLLFM are defined in the original space to model the higher order feature interactions effectively. Different from LLFM, the anchor points associated with the local FMs and the local coordinates of data points are obtained in a latent space where the Euclidean distance can effectively describe the similarities between data points. Moreover, a novel Supervised Variational Encoding (SVE) technique is proposed to learn such a latent space. Hence, GLLFM can better capture the underlying structure of data and can be effectively applied to scenarios with high-dimensional and sparse data. The local FMs, SVE, anchor points and the local coordinates are optimized jointly. Actually, GLLFM-SVE is a quite general framework, i.e., the LLFM can be interpreted as a special case of GLLFM, and other neural network based FMs can also be easily embedded into this framework.

In summary, the main contributions of this paper are:

1) We propose GLLFM, which generalizes LLFM and is able to deal with high-dimensional and sparse data effectively. Moreover, GLLFM is a flexible framework which can be easily embedded with neural-network-based FMs.
2) A novel Supervised Variational Encoding (SVE) technique is proposed such that the latent space of GLLFM ensures the effectiveness of the distance computation.
3) Experimental results show that GLLFM-SVE significantly improves LLFM. By using neural-network based-FMs as local predictors, our model outperforms the state-of-the-art approaches on large-scale real-world benchmarks with similar number of parameters and comparable training time.

The rest of this paper is organized as follows. Section 2 provides some related work. Section 3 discusses the framework of GLLFM. Section 4 provides the detailed model of the SVE. Experiments are conducted in Section 5, and Section 6 concludes the paper.

2 Related Work

2.1 Factorization Machines

A standard model of the 2-way FM is:

$$f^{FM}(\boldsymbol{x};\Theta) = w_0 + \sum_{j=1}^{n} w_j x_j + \sum_{s=1}^{k} l_s, \tag{1}$$

where n is the dimensionality of the feature vector $\boldsymbol{x} \in \mathbb{R}^n$, k is the dimensionality of bi-interaction factors, and l_s is defined as

$$l_s = \sum_{1 \le j < j' \le n} v_{j,s} v_{j',s} x_j x_{j'}. \tag{2}$$

We usually have $k \ll n$ in practice. For simplicity, we define $\boldsymbol{l} = \{l_s\}_{s=1,\dots,k}$. We denote Θ as the set of model parameters to be estimated, i.e. $\Theta = \{w_0, w_1, \dots, w_n, v_{1,1}, \dots, v_{n,k}\}$.

The 2-way FMs leverage the interaction between features, which makes it efficient in very sparse problems in which other approaches like support vector machines (SVMs) might fail [2]. Recently, there are researches aiming at improving the expressiveness of FM by embedding NNs into it [3], [4], [5], [6], [8], and these approaches improve the performance of FMs.

2.2 Locally Linear Factorization Machines

The intuitive idea of the LLFM [11] is to use different parameters for different data points, i.e.

$$f^{LLFM}(\boldsymbol{x}) = f^{FM}\left(\boldsymbol{x};\Theta_{\boldsymbol{x}}\right), \tag{3}$$

where $\Theta_{\boldsymbol{x}}$ represents the FM parameters at point $\boldsymbol{x}$.

However, it is impossible to train different model parameters $\Theta_{\boldsymbol{x}}$ for each data point $\boldsymbol{x}$. In practice, people usually assume that $\Theta(\boldsymbol{x})$ satisfies the Lipschitz smooth condition [12], [16] and approximate $\Theta_{\boldsymbol{x}}$ by a linear combination of parameters at some anchor points $\tilde{\boldsymbol{x}}_i, i = 1, 2, \dots, m$. Then, $f^{LLFM}(\boldsymbol{x})$ can be approximated by

$$f^{LLFM}(\boldsymbol{x}) \approx \sum_{i=1}^{m} \gamma_i f^{FM}\left(\boldsymbol{x};\Theta_{\tilde{\boldsymbol{x}}_i}\right), \tag{4}$$

where γ_i is the local coding coordinates of each local model $f^{FM}(\boldsymbol{x};\Theta_{\tilde{\boldsymbol{x}}_i})$. Different local encoding schemes have been proposed in the literature, such as exponential decay [14], [15], local soft-assignment coding [12], inverse Euclidean distance [19], k*-NN coding [16], etc. The LLFM model proposed in [11] adopts the k*-NN approach.

While LLFM outperforms FM on some datasets, as shown in [11], an important drawback is that it computes the local coding coordinates γ_i based on the Euclidean distances between the input data $\boldsymbol{x}$ and the anchor points $\{\tilde{\boldsymbol{x}}_i\}_{i=1,\dots,m}$ in the original feature space, which is impractical for the high-dimensional and sparse cases. This paper shows that the performance of LLFM can be limited due to this problem, and proposes a novel GLLFM-SVE model which outperforms not only LLFM but also other state-of-the-art approaches.

3 The Generalized Locally Linear Factorization Machines

The description of the GLLFM-SVE model is separated into 2 sections. In this section, we describe the general framework of GLLFM and discuss its relationship with other popular FM-based approaches. In Section 4, we will present the novel SVE technique that generates the latent space ensuring the effectiveness of the local encoding.

3.1 The Framework of GLLFM

The framework of GLLFM is depicted in Figure 1. GLLFM trains m local FMs in the original feature space, each of which corresponds to an anchor point. The key difference between GLLFM and LLFM is that GLLFM defines anchor points in a latent space which is a representation of the original feature space under a certain encoder. Moreover, the local coding coordinates are also obtained in this latent space, based on the distances between the anchor points and the latent representation of the data point.

Here we provide some notations which will be used to derive the mathematical formulation of GLLFM. Denote the original feature as $\boldsymbol{x}$, the ground truth as y, and the prediction of y as $\hat{y}$. Denote the space of $\boldsymbol{x}$ as $\mathcal{X}$, and the space of y and $\hat{y}$ as $\mathcal{Y}$. Moreover, denote the latent space as $\mathcal{Z}$, and the latent representation of $\boldsymbol{x}$ as $\boldsymbol{z}$.

Based on these notations, we first define an encoder g from $\mathcal{X}$ to $\mathcal{Z}$:

$$\boldsymbol{z} = g(\boldsymbol{x}; \Theta_g), \tag{5}$$

where Θ_g denotes the parameters to be learned. Since g is learnable, it is possible to find some latent space supporting more effective local coding than the original feature space. Specifically, the original feature space $\mathcal{X}$ is usually high-dimensional, but the mapping g can be chosen so that the latent space $\mathcal{Z}$ is a lower-dimensional space, which is more likely to have an effective distance measure. In this section, we focus on the general framework of GLLFM, while the designing of the supervised variational encoder g, which is also one of the key contributions of this work, will be described in Section 4.

The GLLFM can be defined as:

$$\hat{y} = f^{GLLFM}(\boldsymbol{x}) = f^{FM}(\boldsymbol{x}; \Theta_{\boldsymbol{z}}), \tag{6}$$

where $\Theta_{\boldsymbol{z}}$ denotes the parameters of the local FM as a function of the latent vector $\boldsymbol{z}$. Similar to LLFM, it is impossible to learn $\Theta_{\boldsymbol{z}}$ for each $\boldsymbol{z}$, therefore we learn the local parameters for some anchor points $\tilde{\boldsymbol{z}}$ in the latent space, and use the following approximation

$$f^{GLLFM}(\boldsymbol{x}) \approx \sum_{i=1}^{m} \gamma_i f_i^{FM}(\boldsymbol{x}; \Theta_{\tilde{\boldsymbol{z}}_i}), \tag{7}$$

where $\{\tilde{\boldsymbol{z}}_i\}_{i=1,\dots,m}$ are the m anchor points in the latent space $\mathcal{Z}$, and $f_i^{FM}(\boldsymbol{x}; \Theta_{\tilde{\boldsymbol{z}}_i})$ is the local FM corresponding to the anchor point $\tilde{\boldsymbol{z}}_{\boldsymbol{i}}$. $\{\gamma_i\}_{i=1,\dots,m}$ are the m local coding coordinates related to the m anchor points, satisfying $\sum_{i=1}^{m} \gamma_i = 1$.

For convenience, we use a bold symbol $\boldsymbol{\gamma}$ to represent the vector $\{\gamma_i\}_{i=1,\dots,m}$. Here we adopt the method in k*-NN [16], in which the local coding coordintate $\boldsymbol{\gamma}$ can be calculated by the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{\gamma}} \quad & \left| \sum_{i=1}^{m} \gamma_i f_i^{FM}(\boldsymbol{x}; \Theta_{\tilde{\boldsymbol{z}}_{\boldsymbol{i}}}) - w(\boldsymbol{z}) \right| \\ \text{s.t.} \quad & \sum_{i=1}^{m} \gamma_i = 1 \\ & \gamma_i \geq 0, i = 1, 2, \cdots, m \end{aligned} \tag{8}$$

where $w(\boldsymbol{z})$ is the "true value" of $f^{FM}(\boldsymbol{x}; \Theta_{\boldsymbol{z}})$.

It is, of course, impractical to solve Eq. (8) directly due to the fact that the value of $w(\boldsymbol{z})$ is unknown. Anava [16] proposed a k*-NN model to solve this problem. The k*-NN assumes $w(\boldsymbol{z})$ is a continuous function of $\boldsymbol{z}$, and $f^{FM}(\boldsymbol{x}; \Theta_{\boldsymbol{z}})$ is an estimation of $w(\boldsymbol{z})$, of which the upper bound of the estimation error is a certain value C. Under these conditions, the k*-NN solves an upper bound of this optimization problem, i.e.

$$\begin{aligned} \boldsymbol{\gamma} = \arg\min \; & C\|\boldsymbol{\gamma}\|_2 + L \sum_{i=1}^{m} \gamma_i d(\tilde{\boldsymbol{z}}_i, \boldsymbol{z}) \\ \text{s.t.} \quad & \sum_{i=1}^{m} \gamma_i = 1 \\ & \gamma_i \geq 0, i = 1, 2, \cdots, m \end{aligned} \tag{9}$$

and the solution can be formulated as

$$\begin{aligned} \lambda^* &= \min\{\lambda_k | \lambda_k > \beta_{k+1}\} \\ \lambda_k &= \frac{1}{k}\left(\sum_{i=1}^{k} \beta_i + \sqrt{k + \left(\sum_{i=1}^{k} \beta_i \right)^2 - k \sum_{i=1}^{k} \beta_i^2} \right) \\ \gamma_i &= \begin{cases} \dfrac{\lambda^* - \beta_i}{\sum_{\beta_j < \lambda^*} (\lambda^* - \beta_j)} & , \beta_i < \lambda^* \\ 0 & , \beta_i \geq \lambda^* \end{cases} \end{aligned} \tag{10}$$

where $\{\beta_i\}_{i=1,\dots,m}$ is the ascending-ordered normalized distance $\left\{\frac{L}{C} d(\tilde{\boldsymbol{z}}_i, \boldsymbol{z})\right\}$. L is the Lipschitz constant of $w(\boldsymbol{z})$, and $d(\cdot)$ is the Euclidean distance function. In practice, L/C is regarded as a hyperparameter. The readers can refer to [16] for details.

An interesting fact is that although $\boldsymbol{\gamma}$ is defined as the solution to an optimization problem, the gradient of $\boldsymbol{\gamma}$ over $\boldsymbol{z}$ and $\tilde{\boldsymbol{z}}_i$ can be explicitly expressed, thus the parameters can be learned via gradient-based optimization algorithms. The readers can refer to [11] for details.

3.2 The Relationship with Other FM-based Models

GLLFM is a very general framework which can be regarded as the generalization of many other FM-based models.

3.2.1 GLLFM generalizes LLFM

The LLFM proposed in [11] can be regarded as a special case of GLLFM. To see this, we simply let $\boldsymbol{z} = g(\boldsymbol{x}) = \boldsymbol{x}$, and then the latent space $\mathcal{Z}$ coincides with the original feature space $\mathcal{X}$, which is right the LLFM model.

It is worth pointing out that the GLLFM model can benefit much from training local FMs and anchor points in different spaces. In fact, a key assumption of locally linear approaches is that samples close to each other share similar model parameters. However, in the original feature space, especially in highly sparse datasets generated by one-hot encoding, it is likely that the Euclidean distances fail to measure the similarity between samples. For example, if a category feature contains 3 possible values, namely $\{A, B, C\}$, the one-hot encoding of them are $[1, 0, 0]$, $[0, 1, 0]$, and $[0, 0, 1]$ respectively. Then the Euclidean distance between each two of them is $\sqrt{2}$, which cannot show whether A is closer to B than to C.

Even in dense datasets, the Euclidean distance in the original feature space $\mathcal{X}$ may not be a good measure on how "close" a data point is to the anchor points. It is because that the concept of Euclidean distances assumes the standard orthogonality of the basis of the feature space, i.e. different coordinates should be independent, and have the same weight in distance computation. However, the feature space $\mathcal{X}$ does not usually satisfy this assumption.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2903403, IEEE Transactions on Knowledge and Data Engineering

By introducing the latent space $\mathcal{Z}$, GLLFM overcomes this drawback. By properly choosing encoding technique (see Section 4), $\mathcal{Z}$ can be a space where Euclidean distances are effective to determine the similarity between data points.

3.2.2 GLLFM is flexible to be embedded with NN-Based FM Models

Here we claim that the proposed GLLFM model is flexible, and can be easily embedded with not only FM but also its extensions such as the NN-based FM models. Take NFM [3] as an example. To embed NFM into our GLLFM, rewrite Eq. (7) as

$$f^{GLLNFM}(\boldsymbol{x}) = \sum_{i=1}^{m} \gamma_i f_i^{NFM}(\boldsymbol{x}; \Theta_{\tilde{\boldsymbol{z}}_i}), \tag{11}$$

where $f^{NFM}(\boldsymbol{x}; \Theta_{\tilde{\boldsymbol{z}}_i})$ is the local NFM related to the anchor point $\tilde{\boldsymbol{z}}_i$. We use the abbreviation GLLNFM to represent GLLFM embedded with NFM. The only difference between Eq. (11) and Eq. (7) is that the function f^{FM} is replaced by f^{NFM}, thus the structure shown in Figure 1 remains unchanged. Moreover, by letting $m = 1$ in Eq. (11), GLLNFM degenerates to NFM:

$$f^{GLLNFM}(\boldsymbol{x}) = f_1^{NFM}(\boldsymbol{x}; \Theta_{\tilde{\boldsymbol{z}}_1}). \tag{12}$$

Therefore, it is easy to embed state-of-the-art FM models into the GLLFM model, and these models can be regarded as the degenerated cases of GLLFM. Section 5 will show that GLLFM embedded with proper NN-based FMs outperforms other state-of-the-arts under similar number of parameters and comparable training time.

3.2.3 GLLFM vs. AFM

AFM [4] uses an attentional technique to provide different weights to different feature entries. Although GLLFM also computes the weights, the authors argue that the technique used in AFM is totally different from that used in this paper. Concretely, AFM uses a single FM model, and computes the weights of the elements of each pair-wise interaction, while GLLFM computes the weights of multiple FMs. Moreover, AFM uses an attentional network to compute the weights, while the proposed model utilizes the local coding technique in the latent space $\mathcal{Z}$.

3.3 Learning of GLLFM

According to Section 3.1, there are 3 parts to be learned, namely the encoder parameter Θ_g, the anchor points $\{\tilde{\boldsymbol{z}}_i\}_{i=1,\dots,m}$, and the parameters of the local FMs $\Theta_{\tilde{\boldsymbol{z}}_i}$. Now we consider the learning of these parameters based on the stochastic gradient descent (SGD) algorithm. To this end, we need to compute the gradient of the loss function with respect to these parameters.

Suppose we have sampled a data point $(\boldsymbol{x}, y)$, and obtained the prediction value $\hat{y}$ via Eq. (7). Denote by $\mathcal{L}^G(y, \hat{y})$ the sampled loss function, which can be square loss in the regression tasks or log loss in the classification tasks. Taking the log loss as an example, the objective function $\mathcal{L}^G$ is defined by

$$\mathcal{L}^G(y, \hat{y}) = -y \log \sigma(\hat{y}) - (1 - y) \log(1 - \sigma(\hat{y})) \tag{13}$$

where σ is the sigmoid function.

The objective of the learning algorithm is to minimize the expected loss, and in each step of the SGD algorithm, we compute the gradient of the sampled loss $\mathcal{L}^G(y, \hat{y})$ with respect to the parameters. Now we discuss the gradient of $\mathcal{L}^G$ with respect to these parameters, and then provide the learning algorithm.

3.3.1 $\partial \mathcal{L}^G / \partial \Theta_{\tilde{\boldsymbol{z}}_i}$

According to (7) we have

$$\frac{\partial \mathcal{L}^G}{\partial \Theta_{\tilde{\boldsymbol{z}}_i}} = \frac{\partial \mathcal{L}^G}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial f_i^{FM}} \frac{\partial f_i^{FM}}{\partial \Theta_{\tilde{\boldsymbol{z}}_i}} = \frac{\partial \mathcal{L}^G}{\partial \hat{y}} \gamma_i \frac{\partial f_i^{FM}}{\partial \Theta_{\tilde{\boldsymbol{z}}_i}} \tag{14}$$

where $\partial \mathcal{L}^G / \partial \hat{y}$ is related to the concrete form of the loss function $\mathcal{L}^G$. For $\partial f_i^{FM} / \partial \Theta_{\tilde{\boldsymbol{z}}_i}$ we have, according to Eq. (1)(2):

$$\begin{aligned} \frac{\partial f_i^{FM}}{\partial w_{\tilde{\boldsymbol{z}}_i, j}} &= x_j \\ \frac{\partial f_i^{FM}}{\partial v_{\tilde{\boldsymbol{z}}_i, j, s}} &= \left(\sum_{j'} v_{\tilde{\boldsymbol{z}}_i, j', s} x_{j'} - v_{\tilde{\boldsymbol{z}}_i, j, s} x_j \right) x_j \end{aligned} \tag{15}$$

3.3.2 $\partial \mathcal{L}^G / \partial \tilde{\boldsymbol{z}}_i$

According to (7) we have

$$\frac{\partial \mathcal{L}^G}{\partial \tilde{\boldsymbol{z}}_i} = \frac{\partial \mathcal{L}^G}{\partial \hat{y}} \sum_{j=1}^{m} f_j^{FM} \frac{\partial \gamma_j}{\partial \tilde{\boldsymbol{z}}_i} \tag{16}$$

where the main problem is the computation of $\partial \gamma_j / \partial \tilde{\boldsymbol{z}}_i$, which, according to [11], can be derived from Eq. (10):

$$\frac{\partial \gamma_j}{\partial \tilde{\boldsymbol{z}}_i} = \frac{\partial \gamma_j}{\partial \beta_i} \frac{\partial \beta_i}{\partial \tilde{\boldsymbol{z}}_i} \tag{17}$$

Note that we omit the gradient of λ^* with respect to $\tilde{\boldsymbol{z}}_i$ to ensure numerical stability, and such a skill is also used in the training of LLFM [11]. For the right term of Eq. (17), we have

$$\frac{\partial \gamma_j}{\partial \beta_i} = \frac{\lambda^* - \beta_j - \mathbf{1}_{j=i} \left[\sum_{\beta_{j'} < \lambda^*} (\lambda^* - \beta_{j'}) \right]}{\left[\sum_{\beta_{j'} < \lambda^*} (\lambda^* - \beta_{j'}) \right]^2} \tag{18}$$

$$\frac{\partial \beta_i}{\partial \tilde{\boldsymbol{z}}_i} = \frac{L}{C} \frac{\tilde{\boldsymbol{z}}_i - \boldsymbol{z}}{d(\tilde{\boldsymbol{z}}_i, \boldsymbol{z})} \tag{19}$$

Thus, we can compute $\partial \mathcal{L}^G / \partial \tilde{\boldsymbol{z}}_i$ via Eq. (17)(18)(19).

3.3.3 $\partial \mathcal{L}^G / \partial \Theta_g$

The parameters Θ_g only exists in Eq. (5), hence we need to compute $\partial \mathcal{L}^G / \partial \boldsymbol{z}$. It is clear that

$$\frac{\partial \gamma_j}{\partial \Theta_g} = \frac{\partial \mathcal{L}^G}{\partial \hat{y}} \sum_{j=1}^{m} f_j^{FM} \frac{\partial \gamma_j}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \Theta_g} \tag{20}$$

where $\partial \boldsymbol{z} / \partial \Theta_g$ depends on the concrete form of the encoder g. Specifically, when g is the SVE proposed in Section 4, $\partial \boldsymbol{z} / \partial \Theta_g$ is easy to be obtained from backward propagation (see (48)). Therefore, we only need to discuss $\partial \gamma_j / \partial \boldsymbol{z}$, which is similar to the discussion of $\partial \gamma_j / \partial \tilde{\boldsymbol{z}}_i$:

$$\frac{\partial \gamma_j}{\partial \boldsymbol{z}} = \sum_{j'=1}^{m} \frac{\partial \gamma_j}{\partial \beta_{j'}} \frac{\partial \beta_{j'}}{\partial \boldsymbol{z}} = -\sum_{j'=1}^{m} \frac{\partial \gamma_j}{\partial \beta_{j'}} \frac{L}{C} \frac{\tilde{\boldsymbol{z}}_{j'} - \boldsymbol{z}}{d(\tilde{\boldsymbol{z}}_{j'}, \boldsymbol{z})} \tag{21}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2903403, IEEE Transactions on Knowledge and Data Engineering

3.3.4 Learning Algorithm

Based on the abovementioned discussions on the gradients, Algorithm 1 provides a SGD-based learning process of GLLFM. In Algorithm 1, the pre-training process in Line 2 depends on the specific formulation of g. The formulas of the gradients are obtained by the chain rule, and ρ_θ, ρ_z and ρ_g are the learning rates.

Algorithm 1 Learning of GLLFM

1: **procedure** PRE-TRAINING
2: $\Theta_g \leftarrow$ pre-trained value of the encoder g
3: $\mathcal{Z} \leftarrow g(\mathcal{X}; \Theta_g)$
4: $\{\tilde{\boldsymbol{z}}_i\}_{i=1,\dots,m} \leftarrow$ K-Means($\mathcal{Z}$)
5: **procedure** TRAINING
6: **while** not convergent **do**
7: Sample a data point $(\boldsymbol{x}, y)$ randomly
8: Compute $\hat{y}$ and loss function $\mathcal{L}^G(y, \hat{y})$
9: **for** $i = 1 \to m$ **do**
10: $\Theta_{\tilde{\boldsymbol{z}}_i} \leftarrow \Theta_{\tilde{\boldsymbol{z}}_i} - \rho_\theta \frac{\partial \mathcal{L}^G}{\partial \Theta_{\tilde{\boldsymbol{z}}_i}}$
11: $\tilde{\boldsymbol{z}}_i \leftarrow \tilde{\boldsymbol{z}}_i - \rho_z \frac{\partial \mathcal{L}^G}{\partial \tilde{\boldsymbol{z}}_i}$
12: $\Theta_g \leftarrow \Theta_g - \rho_g \frac{\partial \mathcal{L}^G}{\partial \Theta_g}$

4 Supervised Variational Encoding

There are some natural choices of the encoder g, such as Variational Auto-Encoder (VAE) [20], [21], Principal Component Analysis (PCA) [22], Variational Deep Embedding (VaDE) [23], etc. However, these approaches usually cannot handle highly sparse datasets. In this section, we present the Supervised Variational Encoding (SVE) technique, which learns an encoder g such that the induced latent space is suitable for GLLFM. Specifically, we first describe the conditions that the latent space should satisfy, and formulate the SVE model which can induce a latent space satisfying all these conditions. Then we establish the objective of SVE concretely. At last, we provide the details of the implementation.

4.1 Modeling

This subsection provides the SVE model. We first provide some conditions that the latent space should meet, and then propose the SVE model to meet these conditions.

4.1.1 Basic Conditions

Intuitively, the latent space $\mathcal{Z}$ should meet the following conditions.

Condition 1. *Different coordinates are independent.*

Condition 2. *Different coordinates satisfy the same distribution.*

Conditions 1 and 2 ensure the effectiveness of the distance computation. Generally, although $\mathcal{X}$ is high-dimensional, the data points in it usually lie in a lower-dimensional manifold, hence $\mathcal{Z}$ can be a low-dimensional space. Condition 1 ensures the orthogonality of the coordinates in $\mathcal{Z}$, while Condition 2 ensures that different coordinates have the same weight in distance computation.

Condition 3. *The latent space $\mathcal{Z}$ can provide some information of the label space $\mathcal{Y}$, i.e. there exists a predictor from $\mathcal{Z}$ to $\mathcal{Y}$.*

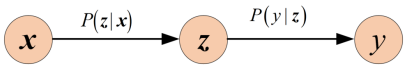

Fig. 2. Probabilistic graphical model of SVE.

This condition is to ensure the effectiveness of the local structure of the latent space $\mathcal{Z}$ in prediction, which is also important when applying the locally linear technique. We emphasize that Condition 3 does not require a strong predictor from $\mathcal{Z}$ to $\mathcal{Y}$. As will be shown in Section 5.2, even if the predictor provided by SVE is weak, GLLFM-SVE will be a strong predictor that outperforms state-of-the-art approaches. It is because that under Condition 3, the dominated features for prediction tasks are described by the relative location in $\mathcal{Z}$, then the local predictors can focus on more detailed features to form a stronger predictor.

4.1.2 The SVE Model

The SVE model can be formulated as a probabilistic graphical model, as shown in Figure 2. Let $P(\boldsymbol{z}|\boldsymbol{x})$ be the conditional probability of $\boldsymbol{z}$ given $\boldsymbol{x}$, and $P(y|\boldsymbol{z})$ be the conditional probability of y given $\boldsymbol{z}$. Therefore, $P(\boldsymbol{z}|\boldsymbol{x})$ represents the encoder from $\mathcal{X}$ to $\mathcal{Z}$, while $P(y|\boldsymbol{z})$ represents the predictor from $\mathcal{Z}$ to $\mathcal{Y}$. According to Figure 2, the joint distribution can be written as

$$P(\boldsymbol{x}, y, \boldsymbol{z}) = P(y|\boldsymbol{z})P(\boldsymbol{z}|\boldsymbol{x})P(\boldsymbol{x}). \tag{22}$$

We first claim that the proposed model, specifically the encoder $P(\boldsymbol{z}|\boldsymbol{x})$, is compatible with the general mapping $g : \mathcal{X} \to \mathcal{Z}$ defined in Eq. (5). Due to the fact that the encoder maps the input vector $\boldsymbol{x}$ into a probability distribution $P(\boldsymbol{z}|\boldsymbol{x})$, we can regard its expectation as the latent representation in GLLFM. Specifically, we have

$$g(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{z}\sim P(\boldsymbol{z}|\boldsymbol{x})}[\boldsymbol{z}], \tag{23}$$

which means that $g(\boldsymbol{x})$ is the conditional expectation of $\boldsymbol{z}$ given $\boldsymbol{x}$. We emphasize that although only the encoder $P(\boldsymbol{z}|\boldsymbol{x})$ exists in the mapping g in GLLFM, the predictor $P(y|\boldsymbol{z})$ is also an important part of SVE since it will affect the training of g. In fact, we need the predictor to ensure that the latent space $\mathcal{Z}$ satisfies Condition 3.

Now we discuss the objectives to meet the 3 conditions. In order to meet Conditions 1 and 2, we minimize the following Kullback-Leibler (KL) divergence

$$\begin{aligned} \mathcal{K} &= \mathcal{D}_{KL}\left[P(\boldsymbol{z}) \| P_0(\boldsymbol{z})\right] \\ &= \mathbb{E}_{\boldsymbol{z}\sim P(\boldsymbol{z})} \log \frac{P(\boldsymbol{z})}{P_0(\boldsymbol{z})} \end{aligned} \tag{24}$$

where $P(\boldsymbol{z})$ is the probability distribution of $\boldsymbol{z}$

$$P(\boldsymbol{z}) = \int P(\boldsymbol{z}|\boldsymbol{x})P(\boldsymbol{x})d\boldsymbol{x}, \tag{25}$$

and $P_0(\boldsymbol{z}) = \mathcal{N}(0, \boldsymbol{I})$.

Since KL divergence measures the similarity between $P(\boldsymbol{z})$ and $P_0(\boldsymbol{z})$, the minimization of $\mathcal{K}$ drives $\mathcal{Z}$ to satisfy a standard Gaussian distribution. Note that the standard-Gaussian-distribution condition is not equivalent but stronger than Conditions 1 and 2. Thus, if $\mathcal{K}$ is minimized, Conditions 1 and 2 are then satisfied.

To meet Condition 3, we use the maximum likelihood method, i.e. to maximize the conditional probability

$$\log P(y|\boldsymbol{x}) = \log \int P(y|\boldsymbol{z})P(\boldsymbol{z}|\boldsymbol{x})d\boldsymbol{z}. \tag{26}$$

In summary, the SVE finds $P(y|\boldsymbol{z})$ and $P(\boldsymbol{z}|\boldsymbol{x})$ to maximize $\log P(y|\boldsymbol{x})$ defined in Eq. (26), while minimize $\mathcal{K}$ defined in Eq. (24). After obtaining $P(y|\boldsymbol{z})$ and $P(\boldsymbol{z}|\boldsymbol{x})$, we use Eq. (23) to obtain the encoder g.

4.2 Establishing the Objective

The KL divergence in Eq. (24) and the log-likelihood in Eq. (26) are both intractable. In this section, we describe how to derive a tractable objective of SVE via Bayesian inference.

4.2.1 $\mathcal{D}_{KL}\left[P(\boldsymbol{z})\|P_0(\boldsymbol{z})\right]$

According to Eq. (24) we have

$$\begin{aligned}\mathcal{K} &= \int P(\boldsymbol{z})\log\frac{P(\boldsymbol{z})}{P_0(\boldsymbol{z})}d\boldsymbol{z}\\ &= \iint P(\boldsymbol{z}|\boldsymbol{x})P(\boldsymbol{x})\log\frac{P(\boldsymbol{z})}{P_0(\boldsymbol{z})}d\boldsymbol{x}d\boldsymbol{z}.\end{aligned} \tag{27}$$

Since $\log P(\boldsymbol{z}) = \log P(\boldsymbol{z}|\boldsymbol{x}) + \log P(\boldsymbol{x}) - \log P(\boldsymbol{x}|\boldsymbol{z})$, Eq. (27) can be factorized as $\mathcal{K} = \mathcal{A} + \mathcal{B} + \mathcal{C}$, where

$$\begin{aligned}\mathcal{A} &= \iint P(\boldsymbol{z}|\boldsymbol{x})P(\boldsymbol{x})\log\frac{P(\boldsymbol{z}|\boldsymbol{x})}{P_0(\boldsymbol{z})}d\boldsymbol{x}d\boldsymbol{z}\\ &= \mathbb{E}_{\boldsymbol{x}\sim P(\boldsymbol{x})}\left\{\mathcal{D}_{KL}\left[P(\boldsymbol{z}|\boldsymbol{x})\|P_0(\boldsymbol{z})\right]\right\}\end{aligned} \tag{28}$$

$$\begin{aligned}\mathcal{B} &= \iint P(\boldsymbol{z}|\boldsymbol{x})P(\boldsymbol{x})\log P(\boldsymbol{x})d\boldsymbol{x}d\boldsymbol{z}\\ &= \int\left[\int P(\boldsymbol{z}|\boldsymbol{x})d\boldsymbol{z}\right]P(\boldsymbol{x})\log P(\boldsymbol{x})d\boldsymbol{x}\\ &= -H(\boldsymbol{x})\end{aligned} \tag{29}$$

$$\begin{aligned}\mathcal{C} &= -\iint P(\boldsymbol{z}|\boldsymbol{x})P(\boldsymbol{x})\log P(\boldsymbol{x}|\boldsymbol{z})d\boldsymbol{x}d\boldsymbol{z}\\ &= -\int\left[\int P(\boldsymbol{x}|\boldsymbol{z})\log P(\boldsymbol{x}|\boldsymbol{z})d\boldsymbol{x}\right]P(\boldsymbol{z})d\boldsymbol{z}\\ &= H(\boldsymbol{x}|\boldsymbol{z})\end{aligned} \tag{30}$$

where $H(\boldsymbol{x})$ is the entropy of $\boldsymbol{x}$, and $H(\boldsymbol{x}|\boldsymbol{z})$ is the conditional entropy of $\boldsymbol{x}$ given $\boldsymbol{z}$.

The expectation $\mathcal{A}$ can be approximated by emperically taking average over all samples, i.e. $\mathcal{A} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{D}_{KL}\left[P(\boldsymbol{z}|\boldsymbol{x}^{(i)})\|P_0(\boldsymbol{z})\right]$. Therefore, when considering a single data point $\boldsymbol{x}$, $\mathcal{A}$ should be $\mathcal{D}_{KL}\left[P(\boldsymbol{z}|\boldsymbol{x})\|P_0(\boldsymbol{z})\right]$. The expression of $\mathcal{B}$ only contains $P(\boldsymbol{x})$, thus has no contribution to $P(\boldsymbol{z}|\boldsymbol{x})$. However, the conditional entropy $\mathcal{C}$ is intractable because of the posterior probability $P(\boldsymbol{x}|\boldsymbol{z})$. Therefore, the sum $\mathcal{K}$ is also intractable. To solve this problem, we use the following inequality

$$\mathcal{A} = \mathcal{K} + H(\boldsymbol{x}) - H(\boldsymbol{x}|\boldsymbol{z}) \geq \mathcal{K}. \tag{31}$$

Therefore, $\mathcal{A}$ is an upper bound of $\mathcal{K}$. Moreover, it is clear that $\mathcal{A}$ and $\mathcal{K}$ has the same minima, i.e., 0 (to see this, simply let $P(\boldsymbol{z}|\boldsymbol{x}) = N(0, \boldsymbol{I})$). Thus we can minimize $\mathcal{A}$ in order to minimize $\mathcal{K}$.

4.2.2 Log-likelihood

By applying the Bayes' rule to $\log P(y|\boldsymbol{x})$, and considering Eq. (22), we have

$$\log P(y|\boldsymbol{x}) = \log P(y|\boldsymbol{z}) + \log P(\boldsymbol{z}|\boldsymbol{x}) - \log P(\boldsymbol{z}|\boldsymbol{x}, y). \tag{32}$$

Then by taking the expectation of both sides of Eq. (32) over $\boldsymbol{z}\sim P(\boldsymbol{z}|\boldsymbol{x})$, we have

$$\begin{aligned}\log P(y|\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{z}\sim P(\boldsymbol{z}|\boldsymbol{x})}\log P(y|\boldsymbol{z})\\ +\mathcal{D}_{KL}\left[P(\boldsymbol{z}|\boldsymbol{x})\|P(\boldsymbol{z}|\boldsymbol{x}, y)\right].\end{aligned} \tag{33}$$

The second term in the right of Eq. (33) is intractable due to the posterior probability $P(\boldsymbol{z}|\boldsymbol{x}, y)$. To solve this problem, define the evidence lower bound (ELBO) as

$$\begin{aligned}ELBO &= \mathbb{E}_{\boldsymbol{z}\sim P(\boldsymbol{z}|\boldsymbol{x})}\log P(y|\boldsymbol{z})\\ &= \log P(y|\boldsymbol{x}) - \mathcal{D}_{KL}\left[P(\boldsymbol{z}|\boldsymbol{x})\|P(\boldsymbol{z}|\boldsymbol{x}, y)\right].\end{aligned} \tag{34}$$

Since the KL divergence in Eq. (34) is non-negative, $\mathcal{L}^{SVE}$ is a lower bound of $\log P(y|\boldsymbol{x})$.

Intuitively, by maximizing $ELBO$, one can maximize $\log P(y|\boldsymbol{x})$ while minimize the KL divergence between $\log P(\boldsymbol{z}|\boldsymbol{x})$ and $\log P(\boldsymbol{z}|\boldsymbol{x}, y)$. Moreover, if the KL divergence is 0, then we have

$$\max\log P(y|\boldsymbol{x}) = \max ELBO. \tag{35}$$

The following proposition holds:

Proposition 1. *if y is uniquely determined by $\boldsymbol{x}$, i.e. there exists a mapping $\alpha : \mathcal{X}\to\mathcal{Y}$ so that $y = \alpha(\boldsymbol{x})$, then we have*

$$\mathcal{D}_{KL}\left[P(\boldsymbol{z}|\boldsymbol{x})\|P(\boldsymbol{z}|\boldsymbol{x}, y)\right] = 0. \tag{36}$$

Proof. We have $P(y|\boldsymbol{x}) = \delta(y-\alpha(\boldsymbol{x}))$, where $\delta(\cdot)$ is the Dirac delta function. Then we have

$$\begin{aligned}P(\boldsymbol{z}|\boldsymbol{x}) &= \int P(\boldsymbol{z}|\boldsymbol{x}, y)P(y|\boldsymbol{x})dy\\ &= \int P(\boldsymbol{z}|\boldsymbol{x}, y)\delta(y-\alpha(\boldsymbol{x}))dy\\ &= P(\boldsymbol{z}|\boldsymbol{x}, \alpha(\boldsymbol{x})) = P(\boldsymbol{z}|\boldsymbol{x}, y)\end{aligned} \tag{37}$$

Therefore the KL divergence in Eq. (36) is 0. □

Note that the condition in this proposition (approximately) holds in many practical cases. Thus, Eq. (35) approximately holds, and maximizing $\log P(y|\boldsymbol{x})$ can be replaced by maximizing $\mathcal{L}^{SVE}$. Moreover, Eq. (35) has another interpretation. By minimizing $\mathcal{D}_{KL}\left[P(\boldsymbol{z}|\boldsymbol{x})\|P(\boldsymbol{z}|\boldsymbol{x}, y)\right]$, $P(\boldsymbol{z}|\boldsymbol{x})$ will be driven close to $P(\boldsymbol{z}|\boldsymbol{x}, y)$. This implies that the information provided by y will be contained in $P(\boldsymbol{z}|\boldsymbol{x})$, which satisfies Condition 3.

4.2.3 The final objective

According to the above discussion, it needs to minimize $\mathcal{D}_{KL}\left[P(\boldsymbol{z}|\boldsymbol{x})\|P_0(\boldsymbol{z})\right]$ while maximizing $ELBO$ when training an SVE. To obtain a single objective, we use the weight-sum of the 2 objectives:

$$\min\mathcal{L}^{SVE} = \lambda\mathcal{D}_{KL}\left[P(\boldsymbol{z}|\boldsymbol{x})\|P_0(\boldsymbol{z})\right] - \mathbb{E}_{\boldsymbol{z}\sim P(\boldsymbol{z}|\boldsymbol{x})}\log P(y|\boldsymbol{z}). \tag{38}$$

where λ is a hyper-parameter which shows the trade-off between the two objectives.

In practice, $P(\boldsymbol{z}|\boldsymbol{x})$ is formulated as a parametrized conditional normal distribution,

$$P(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{\mu}(\boldsymbol{x}), \boldsymbol{\Sigma}(\boldsymbol{x})\right), \tag{39}$$

where $\boldsymbol{\mu}(\boldsymbol{x})$ and $\boldsymbol{\Sigma}(\boldsymbol{x})$ are the expectation and covariance matrix of $\boldsymbol{z}$. In practice, we model $\boldsymbol{\Sigma}(\boldsymbol{x})$ as a diagonal matrix. Note that $P_0(\boldsymbol{z}) = \mathcal{N}(0, \boldsymbol{I})$, the first term of Eq. (38) can be computed analytically

$$\begin{aligned}&\mathcal{D}_{KL}\left[P(\boldsymbol{z}|\boldsymbol{x}\|P_0(\boldsymbol{z})\right]\\ &= \frac{1}{2}\left(\mathrm{tr}(\boldsymbol{\Sigma}) + \boldsymbol{\mu}^T\boldsymbol{\mu} - \dim(\mathcal{Z}) - \log\det(\boldsymbol{\Sigma})\right).\end{aligned} \tag{40}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2903403, IEEE Transactions on Knowledge and Data Engineering

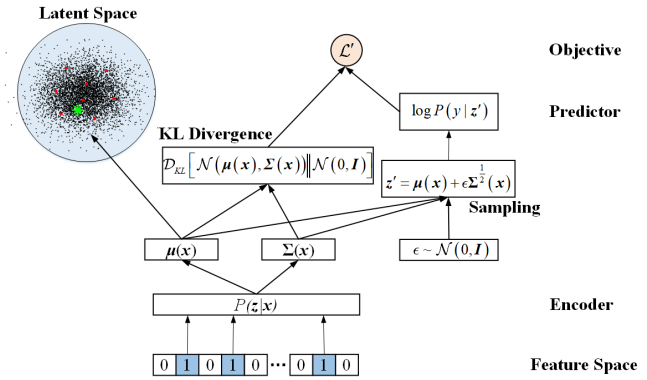

Fig. 3. Detailed structure of SVE.

Moreover, the second term of Eq. (38) can be approximated by Monte-Carlo sampling,

$$\mathbb{E}_{z\sim P(\boldsymbol{z}|\boldsymbol{x})}\left[\log P(y|\boldsymbol{z})\right] \approx \frac{1}{L}\sum_{i=1}^{L}\log P(y|\boldsymbol{z}_i'), \tag{41}$$

where $\boldsymbol{z}_i'$ the i-th sample from $P(\boldsymbol{z}|\boldsymbol{x})$ and L is the number of Monte Carlo samples, which is set to 1 in practice. To make the model can be optimized by back propagation, we use "reparameterization trick" [20], [21] to obtain $\boldsymbol{z}_i'$,

$$\boldsymbol{z}' = \boldsymbol{\mu}\left(\boldsymbol{x}\right) + \epsilon\boldsymbol{\Sigma}^{\frac{1}{2}}\left(\boldsymbol{x}\right), \tag{42}$$

where ϵ is sampled from a standard normal distribution, and $\boldsymbol{\Sigma}^{\frac{1}{2}}(\boldsymbol{x})$ outputs a vector which is the square root of the main diagonal of $\boldsymbol{\Sigma}(\boldsymbol{x})$. The detailed structure of SVE is given in Figure 3.

4.3 The Details of Implementation

This section describes the implementation details of SVE and GLLFM-SVE.

4.3.1 Encoder

The encoder $P(\boldsymbol{z}|\boldsymbol{x})$ includes $\boldsymbol{\mu}(\boldsymbol{x})$ and $\boldsymbol{\Sigma}(\boldsymbol{x})$, each of which can be implemented by a feed-forward network. However, since the input vector $\boldsymbol{x}$ may be highly sparse, thus not suitable as the input of a neural network, we add a bi-interaction layer between the input vector $\boldsymbol{x}$ and the neural network. Specifically,

$$l_s^e = \sum_{j=1}^{n}\sum_{j'=j+1}^{n} v_{j,s}^e v_{j',s}^e x_j x_{j'}, s = 1, 2, \ldots, k^e. \tag{43}$$

This equation is similar with the bi-interaction in the FM model (1)(2). The superscript "e" is used to distinguish variables of the encoder from those of FMs in (1).

It is easy to show that

$$l_s^e = \frac{1}{2}\left[\left(\sum_{j=1}^{n} v_{j,s}^e x_j\right)^2 - \sum_{j=1}^{n}\left(v_{j,s}^e x_j\right)^2\right] \tag{44}$$

Therefore, l_s^e only depends on the nonzero feature x_j. In order to make use of the sparsity of $\boldsymbol{x}$, we add an embedding layer between the input layer and the bi-interaction layer in order to extract the $v_{j,s}$ where $x_j \neq 0$, then we have

$$l_s^e = \frac{1}{2}\left[\left(\sum_{x_j\neq 0} v_{j,s}^e x_j\right)^2 - \sum_{x_j\neq 0}\left(v_{j,s}^e x_j\right)^2\right] \tag{45}$$

For simplicity, use $\boldsymbol{l}^e$ to denote the k^e bi-interaction variables. In practice, k^e can usually be much smaller than k in FM, since the predictor in SVE need not be strong.

$\boldsymbol{l}^e$ is regarded as the input of the neural network, and the output of the neural network is the mean value $\boldsymbol{\mu}(\boldsymbol{x})$ and the variance $\boldsymbol{\Sigma}(\boldsymbol{x})$ of the encoding value. Taking one-layer neural network as an example, we have

$$\begin{aligned}\boldsymbol{p} &= \sigma\left(\boldsymbol{W}_p\boldsymbol{l}^e + \boldsymbol{b}_p\right),\\ \boldsymbol{\mu} &= \boldsymbol{W}_\mu\boldsymbol{p} + \boldsymbol{b}_\mu,\\ \boldsymbol{\Sigma} &= (\boldsymbol{W}_\Sigma\boldsymbol{p} + \boldsymbol{b}_\Sigma)\boldsymbol{I},\end{aligned} \tag{46}$$

where σ is the activation function, and $\boldsymbol{p}$ is the hidden layer. $\boldsymbol{I}$ is the identity matrix, and the multiplier $\boldsymbol{I}$ represents the transformation from a vector to a diagonal matrix.

4.3.2 Predictor

After obtaining the $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, we use (42) to take a sample $\boldsymbol{z}'$ as the input of the predictor $P(y|\boldsymbol{z})$. The predictor is a standard feedforward neural network:

$$\begin{aligned}\boldsymbol{q} &= \sigma\left(\boldsymbol{W}_q\boldsymbol{z}' + \boldsymbol{b}_q\right),\\ \hat{y} &= \boldsymbol{h}^T\boldsymbol{q},\\ P(y|\boldsymbol{z}') &= \sigma(\hat{y})^y(1-\sigma(\hat{y}))^{1-y},\end{aligned} \tag{47}$$

where $\boldsymbol{q}$ is the hidden layer, and $\boldsymbol{h}$ is the output weight vector. The $P(y|\boldsymbol{z}')$ in (47) is for 2-classification problems. For other kinds of problems, the formulation of $P(y|\boldsymbol{z}')$ should be different.

4.3.3 Implementation of GLLFM-SVE

According to Section 3, SVE generates the encoder g of the GLLFM. According to Eq. (23)(39)(46), the encoder function g is

$$\boldsymbol{z} = g(\boldsymbol{x};\Theta_g) = \boldsymbol{\mu}(\boldsymbol{x}) = \boldsymbol{W}_\mu\sigma(\boldsymbol{W}_p\boldsymbol{l}^e + \boldsymbol{b}_p) + \boldsymbol{b}_\mu \tag{48}$$

where $\Theta_g = \left\{\boldsymbol{W}_\mu, \boldsymbol{b}_\mu, \boldsymbol{W}_p, \boldsymbol{b}_p, \left(v_{j,s}^e\right)_{1\leq j\leq n; 1\leq s\leq k^e}\right\}$ is the parameter set. Specifying the general encoder function in Eq. (5) by (48), we obtain the GLLFM with the encoder SVE.

Note that there are still other parameters in SVE besides Θ_g. Denote by Θ_e the other parameters in SVE, i.e.

$$\Theta_e = \{\boldsymbol{W}_\Sigma, \boldsymbol{b}_\Sigma, \boldsymbol{W}_q, \boldsymbol{b}_q, \boldsymbol{h}\} \tag{49}$$

Θ_e will not be used in the encoder, but they are also important in the training of SVE to ensure the latent space to match the 3 conditions.

Algorithm 2 provides the learning process of GLLFM-SVE. Actually, Algorithm 2 is a realization of Algorithm 1 with the encoder SVE. Specifically, we first pre-train the SVE to minimize $\mathcal{L}^{SVE}$, and obtain the initial Θ_g, Θ_e, and the initial anchor points $\tilde{\boldsymbol{z}}_i$. Then, the training process is similar to Algorithm 1 except that the final objective is the combination of $\mathcal{L}^G$ and $\mathcal{L}^{SVE}$, i.e.,

$$\mathcal{L} = \mathcal{L}^G + \eta\mathcal{L}^{SVE} \tag{50}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2903403, IEEE Transactions on Knowledge and Data Engineering

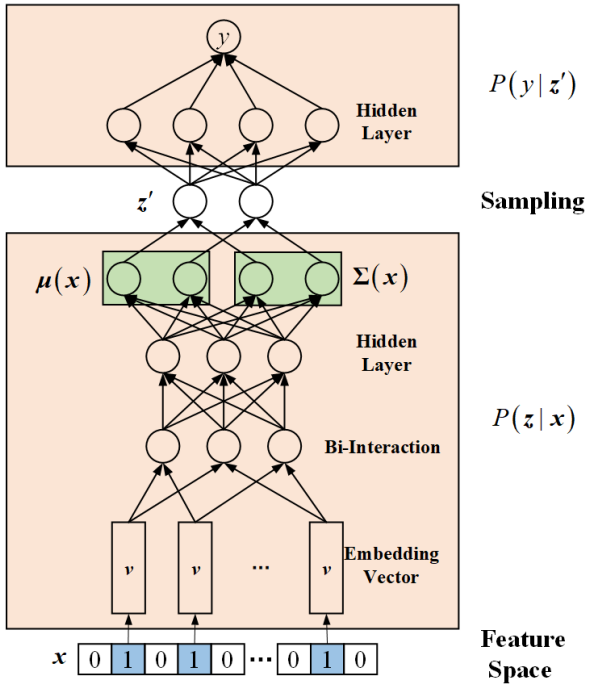

Fig. 4. Implementation of SVE via feedforward networks.

Algorithm 2 Learning of GLLFM-SVE

procedure PRE-TRAINING SVE
  Initialize Θ_g, Θ_e
  while not convergent **do**
    Sample a data point $(\boldsymbol{x}, y)$ randomly
    Compute $\hat{y}$ and loss function $\mathcal{L}^{SVE}(y, \hat{y})$
    $\Theta_g \leftarrow \Theta_g - \rho_g^e \frac{\partial \mathcal{L}^{SVE}}{\partial \Theta_g}$
    $\Theta_e \leftarrow \Theta_e - \rho_e^e \frac{\partial \mathcal{L}^{SVE}}{\partial \Theta_e}$
procedure ANCHOR POINTS
  $\mathcal{Z} \leftarrow g(\mathcal{X}; \Theta_g)$
  $\{\tilde{\boldsymbol{z}}_i\}_{i=1,\ldots,m} \leftarrow$ K-Means$(\mathcal{Z})$
procedure TRAINING GLLFM-SVE
  while not convergent **do**
    Sample a data point $(\boldsymbol{x}, y)$ randomly
    Compute $\hat{y}$ and $\mathcal{L}^G(y, \hat{y}), \mathcal{L}^{SVE}(y, \hat{y})$
    for $i = 1 \rightarrow m$ **do**
      $\Theta_{\tilde{\boldsymbol{z}}_i} \leftarrow \Theta_{\tilde{\boldsymbol{z}}_i} - \rho_\theta \frac{\partial \mathcal{L}^G}{\partial \Theta_{\tilde{\boldsymbol{z}}_i}}$
      $\tilde{\boldsymbol{z}}_i \leftarrow \tilde{\boldsymbol{z}}_i - \rho_z \frac{\partial \mathcal{L}^G}{\partial \tilde{\boldsymbol{z}}_i}$
    $\Theta_g \leftarrow \Theta_g - \rho_g \frac{\partial \mathcal{L}^G}{\partial \Theta_g} - \eta \rho_g \frac{\partial \mathcal{L}^{SVE}}{\partial \Theta_g}$
    $\Theta_e \leftarrow \Theta_e - \eta \rho_e \frac{\partial \mathcal{L}^{SVE}}{\partial \Theta_g}$

where η is the weight to control the trade-off between $\mathcal{L}^G$ and $\mathcal{L}^{SVE}$. Eq. (50) means Θ_g needs not only to support an accurate prediction, but also to generate a latent space satisfying the 3 conditions of SVE. Note that we do not provide the concrete form of $\partial\mathcal{L}/\partial\Theta_g$ and $\partial\mathcal{L}/\partial\Theta_e$ since they can be easily obtained via backward propagation.

Note that in Algorithm 2, ρ_g, ρ_z and ρ_e are usually far smaller than ρ_θ because we have obtained a good enough initial value of Θ_g and $\tilde{\boldsymbol{z}}_i$ in the pre-training process, hence only need to fine-tune them in the training process.

5 EXPERIMENTS

This section provides experimental studies. As discussed in Section 3.2.1, LLFM fails in high-dimensional and sparse real-world datasets. Thus, in order to make a complete test on the proposed approach, we provide 2 groups of experiments. The first group tests the advantage of GLLFM-SVE over LLFM [11], where the same datasets are used as [11]. In the second group of experiments, the proposed approach is tested on 2 real-world datasets in comparison with state-of-the-art models rather than LLFMs.

5.1 Experiment 1: GLLFM-SVE vs LLFM

5.1.1 Experiment Setup

This group of experiments aims to verify the advantages of GLLFM-SVE over LLFM [11]. To be fair, the datasets used here is the same as the datasets used in [11] except that we do not use the Banana dataset, which is a two-dimensional dataset where there is no need to introduce a latent space for local coding. Specifically, these datasets are

1) **Major Atmospheric Gamma Imaging Cherenkov Telescope 2004 (MAGIC04)**[1]: The dataset generated to simulate registration of high energy gamma particles in a ground-based atmospheric Cherenkov gamma telescope using the imaging technique, with 19020 instances, each with 10 numeric features.
2) **IJCNN**[2]: The dataset of the generalization ability challenge of IJCNN 2001 competition. This dataset is preprocessed by Chang [24], and contains 22 features.
3) **Covtype**[3]: The dataset to predict forest cover type from cartographic variables. There are 55 features including numerical and categorical features, such as wilderness areas and soil types.
4) **LETTER**[4]: A letter recognition dataset containing 20000 instances. The features include 16 numerical attributes, i.e., statistical moments and edge counts obtained from a black-and-white rectangular pixel displays.
5) **MNIST**[5]: A famous handwritten digit recognition dataset, containing 70000 examples.

The information of these datasets is shown in Table 1. All datasets are split into training (80%), validation (10%) and test (10%) sets.

The following approaches are compared:
LR and **SVM**: benchmarks of linear classifiers.
FM[6]: the original factorization machines, proposed in [1].
LLFM: the LLFM model introduced in [11].

1. https://archive.ics.uci.edu/ml/datasets/magic+gamma+telescope
2. https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html
3. https://archive.ics.uci.edu/ml/datasets/Covertype
4. https://archive.ics.uci.edu/ml/datasets/letter+recognition
5. http://yann.lecun.com/exdb/mnist/
6. http://www.libfm.org

TABLE 1
Datasets of Experiment 1

Dataset	#Feature	#Example	#Class
MAGIC04	10	19020	2
IJCNN	22	141691	2
Covtype	55	581012	2
LETTER	16	20000	26
MNIST	784	70000	10

SVE: the predictor $P(y|\boldsymbol{z})$ in SVE, which is used to show that SVE satisfies Condition 3.

SVE-LLFM: the LLFM model applied to the latent space obtained by SVE rather than the original space. We use this baseline to show that GLLFM benefits from using local FMs in the original space and only local coding in the latent space.

GLLFM: To show the effectiveness of the proposed model, we use GLLFM with different types of encoders as benchmarks:

1) **GLLFM-VAE**: $\mathcal{Z}$ is the latent space of a variational autoencoder (VAE) [20]. The VAE model maps the feature space into a standard-Gaussian-distributed latent space, but it is an unsupervised model. We do not use supervised VAE models (e.g. Conditional VAE [25]) since in these models the latent representation $\boldsymbol{z}$ is a function of both the input vector $\boldsymbol{x}$ and the output value y, of which the latter is unknown in the prediction process.
2) **GLLFM-AE**, **GLLFM-AAE**: Similar to GLLFM-VAE, but $\mathcal{Z}$ is the latent space of an autoencoder (AE) or an adversial autoencoder (AAE).
3) **GLLFM-MLP**: The encoder is the last hidden layer of a multi-layer perceptron (MLP).
4) **GLLFM-SVE(0)**: The GLLFM-SVE with $\lambda = 0$ in (38). This model is purely supervised, which means that $\mathcal{Z}$ may not satisfy Condition 1 and 2.
5) **GLLFM-SVE**: The proposed GLLFM-SVE model. We choose λ and η from $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}\}$ by cross validation.

For 2-class datasets, we use the logarithm loss function, the prediction accuracy and the area under ROC curve (AUC) as the performance criteria. While for multi-class datasets, we use the cross entropy and the prediction accuracy.

5.1.2 Results and Discussion

Results are shown in Tables 2 and 3, of which each result is with 5 independent runs. We can observe that the proposed GLLFM-SVE approach consistently achieves better performance than the original LLFM in all datasets and criteria. This validates the efficacy of extracting the local coding coordinates in the latent space generated by SVE.

It is also shown that the choice of the encoding technique influences the performance of the GLLFM. To illustrate this, the t-SNE figures of the original feature space and the latent spaces generated by these encoding approaches are drawn in Figure 5. The datasets IJCNN and MNIST are used as examples of 2-class and multi-class datasets. Figure 5 and Tables 2,3 show that

1) GLLFM with unsupervised encoders (GLLFM-VAE/GLLFM-AE/GLLFM-AAE) cannot guarantee a better performance than LLFM. The possible reason is that unsupervised encoders usually cannot extract enough information useful for prediction in the latent space. In these models, GLLFM-VAE slightly outperforms LLFM in most cases. It is because VAE generates a standard-Gaussian-distributed latent space, and can extract some structural information in the latent space. However, since VAE is an unsupervised approach, it cannot guarantee the label information to be extracted. In some cases like MNIST(see Figure 5(f)), VAE can extract partial label information, thus GLLFM-VAE performs almost as well as GLLFM-SVE. But in cases like the IJCNN (see Figure 5(b)), GLLFM-VAE can only be slightly better than the LLFM.
2) GLLFM-SVE(0) and GLLFM-SVE perform better than LLFM and GLLFM-VAE. It is because SVE extracts the dominated features for the label prediction. It is also shown in Figure 5 (c)(d)(g)(h) that samples with different labels lie in different regions, thus GLLFM will be able to extract more detailed information to make a better prediction.
3) SVE also has the ability to make prediction. Sometimes it performs better than LLFM since it contains neural-network parts, but the authors argue that the major reason why GLLFM-SVE outperforms LLFM is not the neural-network part. On one hand, the local predictors in GLLFM-SVE is pure FMs without neural networks, i.e., neural networks is only used to obtain the latent space. On the other hand, GLLFM-SVE also significantly outperforms SVE, which means GLLFM-SVE extracts more detailed information based on the local structure provided by SVE, furthermore, these detailed information are extracted by local FMs rather than neural-betwork-based models.
4) GLLFM-SVE performs better than GLLFM-SVE(0) in most cases, due to the fact that the latent space generated by SVE(0) may not support effective distance computation. For example, GLLFM-SVE performs better than GLLFM-SVE(0) on IJCNN, which can be explained by the fact that the latent space in Figure 5(c) is not as good as that in Figure 5(d): in Figure 5(c), red (positive) points in Area A and green (negative) points in Area B has been well-classified by the SVE(0), but they are close to each other in the latent space, which means the local classifiers of both areas will be considered in the prediction of nearby points. However, the classifier of Area B may be harmful when used to predict points in Area A. In contrast, in Figure 5(d), the well-classified points can be distinguished simply by distance, thus the local FMs only need to extract more detailed information for prediction of the points that SVE cannot distinguish (e.g. Area C). It is also shown in Table 3 that the difference between GLLFM-SVE and GLLFM-SVE(0) is not significant on MNIST, which can be explained by the fact that the latent space in Figure 5(h) is not significantly better than that in

TABLE 2
Results of Experiment 1: MAGIC04, IJCNN, and Covtype

	MAGIC04			IJCNN			Covtype		
	logloss	Acc(%)	AUC	logloss	Acc(%)	AUC	logloss	Acc(%)	AUC
LR	0.4659 ±0.0012	78.88 ±0.15	0.8324 ±0.0023	0.2109 ±0.0005	92.09 ±0.05	0.9120 ±0.0001	0.5503 ±0.0013	70.76 ±0.05	0.7936 ±0.0004
SVM	0.4430 ±0.0023	79.02 ±0.15	0.8385 ±0.0046	0.2253 ±0.0010	91.42 ±0.06	0.9080 ±0.0003	0.5825 ±0.0026	67.66 ±0.09	0.7324 ±0.0012
FM	0.3800 ±0.0016	82.33 ±0.18	0.8858 ±0.0039	0.0924 ±0.0007	96.79 ±0.03	0.9786 ±0.0001	0.4568 ±0.0019	78.66 ±0.07	0.8652 ±0.0007
LLFM	0.3358 ±0.0008	85.87 ±0.12	0.9113 ±0.0032	0.0519 ±0.0010	98.51 ±0.03	0.9935 ±0.0002	0.4130 ±0.0037	80.78 ±0.32	0.8886 ±0.0033
SVE	0.3353 ±0.0006	86.21 ±0.13	0.9139 ±0.0025	0.0562 ±0.0013	98.01 ±0.05	0.9913 ±0.0003	0.2950 ±0.0035	87.34 ±0.27	0.9460 ±0.0029
SVE-LLFM	0.3325 ±0.0012	85.98 ±0.18	0.9135 ±0.0030	0.0502 ±0.0011	98.26 ±0.05	0.9935 ±0.0005	0.2710 ±0.0028	88.48 ±0.21	0.9553 ±0.0023
GLLFM-VAE	0.3546 ±0.0020	85.08 ±0.25	0.9070 ±0.0032	0.0484 ±0.0018	98.35 ±0.09	0.9943 ±0.0003	0.3713 ±0.0042	82.87 ±0.23	0.9132 ±0.0032
GLLFM-AE	0.3582 ±0.0025	84.95 ±0.23	0.9026 ±0.0033	0.0492 ±0.0019	98.21 ±0.09	0.9938 ±0.0006	0.3826 ±0.0022	81.62 ±0.16	0.9027 ±0.0015
GLLFM-AAE	0.3523 ±0.0018	85.32 ±0.20	0.9102 ±0.0026	0.0501 ±0.0015	98.17 ±0.07	0.9931 ±0.0004	0.3654 ±0.0021	84.06 ±0.15	0.9201 ±0.0015
GLLFM-MLP	0.3325 ±0.0023	86.44 ±0.20	0.9151 ±0.0020	0.0471 ±0.0018	98.58 ±0.06	0.9943 ±0.0005	0.2842 ±0.0028	87.96 ±0.23	0.9513 ±0.0026
GLLFM-SVE(0)	0.3261 ±0.0033	86.28 ±0.25	0.9179 ±0.0011	0.0467 ±0.0010	98.57 ±0.05	0.9942 ±0.0001	0.2650 ±0.0052	88.83 ±0.39	0.9572 ±0.0038
GLLFM-SVE	**0.3159** ±0.0019	**86.93** ±0.14	**0.9239** ±0.0021	**0.0420** ±0.0012	**98.72** ±0.05	**0.9948** ±0.0002	**0.2310** ±0.0039	**90.28** ±0.31	**0.9647** ±0.0030

Figure 5(g).

5) Briefly speaking, the performance of SVE-LLFM is not as good as GLLFM-SVE. It is because that SVE is only used to help extract the local information, and the detailed information may be lost during the encoding process. Moreover, the latent space is a low-dimensional dense space, where FM cannot usually perform well. Therefore in GLLFM-SVE, we only compute the distances and local coding coordinates in the latent space, while the local FMs are still used in the original feature space.

5.2 Experiment 2: GLLFM-SVE on Real-World Datasets

5.2.1 Experiment Setup

This group of experiments verifies the performance of the proposed GLLFM-SVE on real-world datasets. Since all the baselines used NN-based FMs, we embed NFM into GLLFM-SVE for a fair comparison. This can be easily achieved by Eq. (11), which is referred as GLLNFM-SVE.

The datasets used here are

1) **Frappe**: This dataset, introduced in [26], contains 96203 app usage logs, each with user ID, app ID and 8 context variables. He [3] preprocessed this dataset by adding negative instances and applying one-hot encoding. We directly adopt the data released with their code[7].

2) **Criteo**: Criteo dataset[8] is a click-through rate dataset including 45 million users' click records. **For this dataset, an improvement of 0.001 in logloss is considered as practically significant [8]**. There are 13 continuous features and 26 categorical ones. We use the one-hot encoding to handle the features.

The information of the two datasets are shown in Table. All datasets are split into training (80%), validation (10%) and test (10%) sets. 4. Furthermore, the following state-of-the-art approaches are compared:

LR and **SVM**: benchmarks of linear classifiers.

FM: The original factorization machines, proposed in [1].

NFM [3][9]: It uses the bi-interaction terms of the factorization machines as the input of a feedforward neural network..

AFM [4][10]: It applies the attentional technique in the feature embedding part, and uses a feedforward neural network to generate the attentional coefficients.

DCN [8]: The DCN introduces a cross network to model the interaction between features, and combines it with a deep neural network.

DeepFM [5]: The deepFM model is a summation of the FM model and a feedforward network sharing the embedding vectors.

7. https://github.com/hexiangnan/neural_factorization_machine

8. http://labs.criteo.com/2014/02/download-kaggle-display - advertising-challenge-dataset/

9. https://github.com/hexiangnan/neural_factorization_machine

10. https://github.com/hexiangnan/attentional_factorization_machine

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2903403, IEEE Transactions on Knowledge and Data Engineering

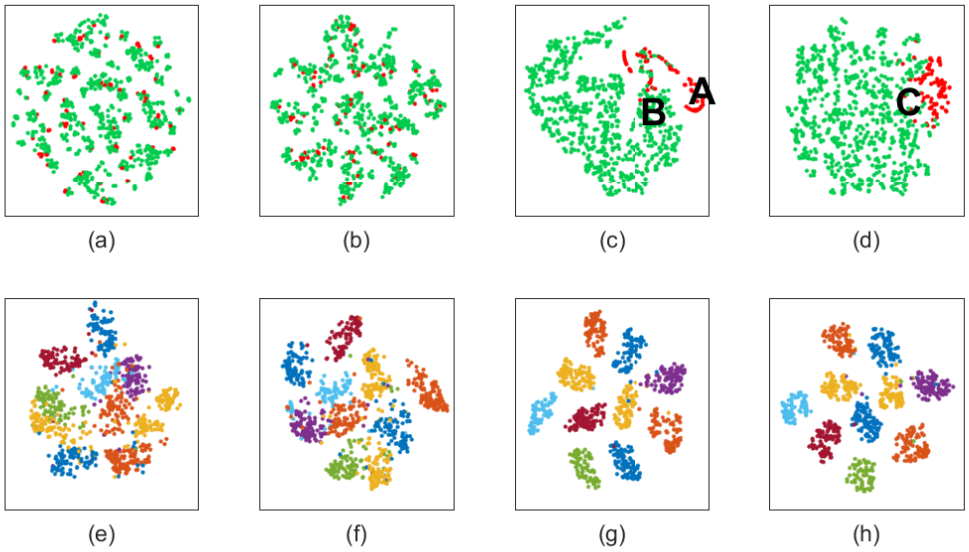

Fig. 5. t-SNE of IJCNN(the top row) and MNIST (the bottom row). The 1st column is the t-SNE of the original feature space $\mathcal{X}$. The 2nd $\sim$ 4th columns are the latent spaces obtained by VAE, SVE(0), and SVE.

TABLE 3
Results of Experiment 1: LETTER and MNIST

	LETTER		MNIST	
	cross entropy	Acc(%)	cross entropy	Acc(%)
LR	0.9075 ±0.0032	76.62 ±0.32	0.3334 ±0.0053	90.78 ±0.20
SVM	0.8836 ±0.0084	78.42 ±0.46	0.3692 ±0.0049	89.32 ±0.17
FM	0.5567 ±0.0138	83.32 ±0.61	0.1461 ±0.0062	95.61 ±0.15
LLFM	0.1738 ±0.0094	95.45 ±0.15	0.1133 ±0.0057	96.26 ±0.19
SVE	0.1152 ±0.0028	96.92 ±0.13	0.1317 ±0.0026	95.93 ±0.10
SVE-LLFM	0.1152 ±0.0031	96.92 ±0.14	0.1206 ±0.0022	96.74 ±0.15
GLLFM-VAE	0.1606 ±0.0030	95.30 ±0.20	0.0865 ±0.0028	98.12 ±0.04
GLLFM-AE	0.1606 ±0.0026	95.30 ±0.15	0.0924 ±0.0031	97.82 ±0.06
GLLFM-AAE	0.1606 ±0.0032	95.30 ±0.18	0.0891 ±0.0041	98.01 ±0.05
GLLFM-MLP	0.1606 ±0.0024	95.30 ±0.13	0.0809 ±0.0014	98.48 ±0.03
GLLFM-SVE(0)	0.0846 ±0.0021	97.44 ±0.12	0.0810 ±0.0021	98.45 ±0.06
GLLFM-SVE	**0.0733** ±0.0024	**97.68** ±0.15	**0.0806** ±0.0018	**98.52** ±0.09

TABLE 4
Datasets of Experiment 2

Dataset	#Feature	#Example	#Non-Zero Entry
Frappe	5382	2.9×10^5	10
Criteo	3.3×10^7	4.6×10^7	39

There are also other models such as Deep Crossing, Wide&Deep, etc. We choose the abovementioned models since they are the most recent ones, and are reported to outperform other models [3], [4], [5].

We also consider GLLFM-VAE/GLLFM-AE/GLLFM-AAE/SVE/SVE-LLFM as benchmarks. For GLLFM-VAE/GLLFM-AE/GLLFM-AAE, we use the techniques in [27], [28] to accelerate the training of the autoencoders for sparse data. we do not report the results of LLFM, which fails for high-dimensional and sparse datasets.

For DCN and DeepFM, we adopt the suggested parameters in [8] and [5] respectively. For the other models, we use grid search to tune the hyperparameters. The logarithm loss function and the area under curve (AUC) are used as performance criteria. The prediction accuracy is not used since the labels in these datasets are imbalanced.

5.2.2 *Results*

The results are shown in Table 5. From Table 5 we can conclude that

1) The proposed approach achieves the best performance on both datasets. To further examine this, Figure 6 shows the loss function and AUC of some state-of-the-art approaches on the test sets at each epoch. It is clearly shown that the proposed GLLNFM-SVE outperforms the other approaches.
2) The GLLNFM-SVE outperforms the GLLNFM-SVE(0) especially in the log loss. This shows that a standard-Gaussian-distribution latent space is helpful when applying local encoding techniques. It also shows the necessity of Conditions 1 and 2.
3) Although the performance of the predictor in the SVE is not as good as state-of-the-art approaches, the information it extracts in the latent space helps the GLLNFM-SVE to outperform these approaches. This shows the necessity of Condition 3.

11. The AFM fails in Criteo dataset due to the large computation burden.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2903403, IEEE Transactions on Knowledge and Data Engineering

TABLE 5
Results of Experiment 2 (each with 5 independent runs)

	Frappe		Criteo	
	logloss	AUC	logloss	AUC
LR	0.2893 ± 0.0002	0.9329 ± 0.0006	0.4659 ± 0.0002	0.7851 ± 0.0001
SVM	0.3024 ± 0.0004	0.9268 ± 0.0008	0.4704 ± 0.0002	0.7786 ± 0.0002
FM	0.1520 ± 0.0003	0.9808 ± 0.0006	0.4536 ± 0.0001	0.7968 ± 0.0002
NFM	0.1410 ± 0.0004	0.9842 ± 0.0004	0.4494 ± 0.0002	0.8020 ± 0.0003
AFM	0.1513 ± 0.0013	0.9752 ± 0.0004	N/A[11]	N/A
DCN	0.1476 ± 0.0017	0.9839 ± 0.0002	0.4509[12] ± 0.0002	0.8012 ± 0.0002
DeepFM	0.1489 ± 0.0004	0.9832 ± 0.0003	0.4499 ± 0.0002	0.8018 ± 0.0003
SVE	0.1604 ± 0.0023	0.9811 ± 0.0012	0.4550 ± 0.0004	0.7953 ± 0.0005
SVE-LLFM	0.1404 ± 0.0026	0.9827 ± 0.0015	0.4521 ± 0.0006	0.7981 ± 0.0006
GLLNFM-VAE	0.1419 ± 0.0003	0.9828 ± 0.0007	0.4499 ± 0.0004	0.8015 ± 0.0003
GLLNFM-AE	0.1440 ± 0.0003	0.9831 ± 0.0002	0.4508 ± 0.0003	0.8008 ± 0.0002
GLLNFM-AAE	0.1435 ± 0.0004	0.9826 ± 0.0003	0.4501 ± 0.0002	0.8011 ± 0.0003
GLLNFM-MLP	0.1732 ± 0.0004	0.9817 ± 0.0004	0.4512 ± 0.0004	0.7999 ± 0.0005
GLLNFM-SVE(0)	0.1377 ± 0.0004	0.9837 ± 0.0005	0.4491 ± 0.0003	0.8044 ± 0.0004
GLLNFM-SVE	**0.1284** ± 0.0003	**0.9858** ± 0.0006	**0.4475** ± 0.0003	**0.8048** ± 0.0003

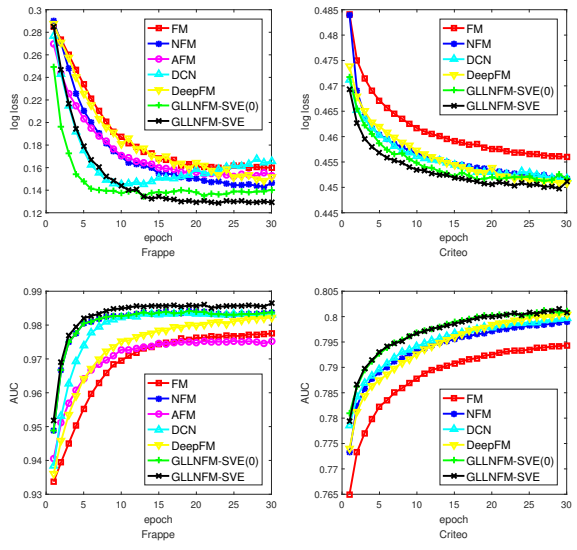

Fig. 6. Epoch-wise demonstration of different algorithms with logloss and AUC on test data.

5.2.3 Discussion

Here we provide some discussions upon the proposed approach.

The number of parameters: A GLLNFM-SVE model with k bi-interaction factors and m anchor points has a similar number of parameters to an FM model with $m \times k$ bi-interaction factors. Therefore, when tuning the two hyper-parameters, a natural concern is the difference between increasing k and increasing m. Figure 7 shows the performance of different models w.r.t. the number of parameters. We use $m \times k$ to describe the number of parameters (for the baseline approaches we let $m = 1$). We observe that

1) By simply increasing the number of parameters k in baseline approaches, the performance of FM/NFM/AFM saturates, and that of DCN/DeepFM becomes even worse due to overfitting. But for GLLNFM-SVE, increasing anchor points improves the performance more significantly. This phenomenon is stronger when k is large. For example, in Frappe, $k = 64, m = 2$ is better than $k = 128, m = 1$.
2) If k is small, the performance of GLLNFM-SVE will also saturate when m is large. For example, in our experiments, $k = 64, m = 8$ is the best on Frappe, while on Criteo the $k = 32, m = 8$ is the best. This can be explained by the fact that a certain number of bi-interaction factors is necessary in each local FM, thus fewer bi-interaction factors will decrease the performance even if the number of anchor points is large. When tuning an GLLNFM-SVE, one can first tune k to a critical value so that the NFM does not saturate, and then tune m to reach the best performance.

The dimensionality of latent space: We test the performance of the proposed model under different dimensionality of the latent space, as shown in Figure 8. It is shown that there exists an optimal setting of the dimensionality of the latent space. An intuitive explanation is that the high-dimensional data usually lies in a low-dimensional manifold, of which the dimensionality is a fixed (but unknown) value. If the dimensionality of the latent space is close to this value, the performance will be better. A lower dimensionality usually cannot describe the features of the data, while a higher dimensionality will lead to the dependency between some axes.

Training time: The proposed model is natural for parallization due to the fact that the local predictors are independent of each others. Therefore, the computational time is comparable to a single local predictor. We test the per-epoch training time of these approaches on Criteo with a GPU environment, as shown in Figure 9. The baseline used here is LR, which consumes about 30 min on a Tesla M40 GPU. It is shown that the training time of GLLNFM-SVE is a bit longer than NFM and FM, but less than other models. Therefore, the proposed approach achieves a good trade-off between computational burden and performance.

12. This result is different from the log loss reported in [8], which is based on a more complicated data preprocessing method. We use the same preprocessing for all models for a fair comparison in our experiments.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2903403, IEEE Transactions on Knowledge and Data Engineering

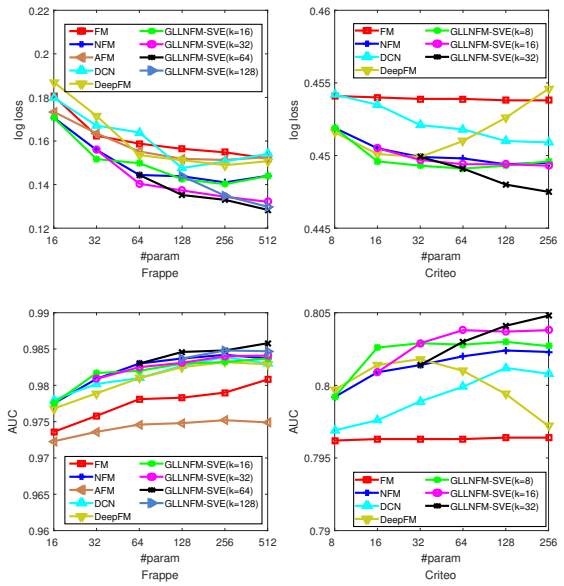

Fig. 7. Performance w.r.t. number of parameters. The number of parameters is described by $m \times k$. Points in the same line of the GLLNFM-SVE model has the same k but different m. For example, in Line "GLLNFM-SVE($k = 16$)" in (a), points at "#param= $16, 32, 64, \ldots$" are with $m = 1, 2, 4, \ldots$.

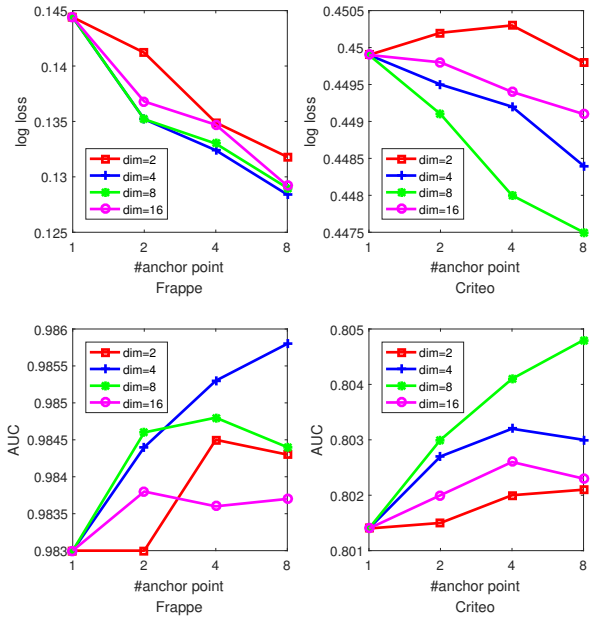

Fig. 8. Performance w.r.t. the dimensionality of latent space. The number k is 64 on Frappe and 32 on Criteo.

6 Conclusion

This paper presents the GLLFM-SVE model, which can effectively capture the local structure of complex datasets to improve the performance of FM. The proposed approach generalizes LLFM provided in [11], and is flexible to be embedded with state-of-the-art extensions of FM. The GLLFM-SVE model trains local FMs in the original feature space, while the anchor points and local coordinates in a latent space supporting effective distance computation. We also provide the SVE technique to train the encoder to obtain

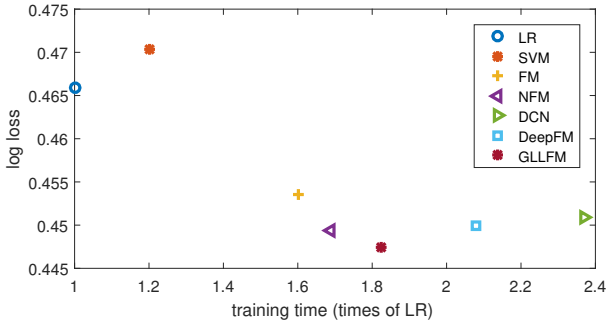

Fig. 9. Training time and log loss on Criteo.

a latent space which is standard-Gaussian-distributed and contains the dominated information of the label space. Our experiment results demonstrate that 1) The SVE can extract the structural information of different datasets effectively, and this information is crucial to improve the performance of GLLFM. 2) GLLFM-SVE outperforms not only LLFM, but also other state-of-the-art approaches in real-world benchmarks.

There are two interesting directions for future study. On one hand, we will explore the semantics of different anchor points. On the other hand, it is attractive to design distributed algorithms to train GLLFM-SVE model more efficiently, since the locally linear structure makes it natural for distributed learning.

References

[1] S. Rendle, "Factorization machines," in *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pp. 995–1000, IEEE, 2010.

[2] S. Rendle, "Factorization machines with libfm," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 3, no. 3, p. 57, 2012.

[3] X. He and T. Chua, "Neural factorization machines for sparse predictive analytics," *CoRR*, vol. abs/1708.05027, 2017.

[4] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, and T. Chua, "Attentional factorization machines: Learning the weight of feature interactions via attention networks," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)*, 2017.

[5] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "Deepfm: A factorizaion-machine based neural network for ctr prediction," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)*, 2017.

[6] H. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah, "Wide & deep learning for recommender systems," *CoRR*, vol. abs/1606.07792, 2016.

[7] Y. Shan, T. R. Hoens, J. Jiao, H. Wang, D. Yu, and J. Mao, "Deep crossing: Web-scale modeling without manually crafted combinatorial features," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 255–262, ACM, 2016.

[8] R. Wang, B. Fu, G. Fu, and M. Wang, "Deep & cross network for ad click predictions," *CoRR*, vol. abs/1708.05123, 2017.

[9] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[10] T.-K. Kim and J. Kittler, "Locally linear discriminant analysis for multimodally distributed classes for face recognition with a single model image," *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 3, pp. 318–327, 2005.

[11] C. Liu, T. Zhang, P. Zhao, J. Zhou, and J. Sun, "Locally linear factorization machines," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)*, 2017.

[12] K. Yu, T. Zhang, and Y. Gong, "Nonlinear learning using local coordinate coding," in *Advances in neural information processing systems*, pp. 2223–2231, 2009.

[13] X. Mao, Z. Fu, O. Wu, and W. Hu, "Optimizing locally linear classifiers with supervised anchor point learning.," in *IJCAI*, pp. 3699–3706, 2015.
[14] J. C. Van Gemert, J.-M. Geusebroek, C. J. Veenman, and A. W. Smeulders, "Kernel codebooks for scene categorization," in *European conference on computer vision*, pp. 696–709, Springer, 2008.
[15] L. Liu, L. Wang, and X. Liu, "In defense of soft-assignment coding," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2486–2493, IEEE, 2011.
[16] O. Anava and K. Levy, "k*-nearest neighbors: From global to local," in *Advances in Neural Information Processing Systems*, pp. 4916–4924, 2016.
[17] E. P. Xing, M. I. Jordan, S. J. Russell, and A. Y. Ng, "Distance metric learning with application to clustering with side-information," in *Advances in neural information processing systems*, pp. 521–528, 2003.
[18] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Advances in neural information processing systems*, pp. 1473–1480, 2006.
[19] L. Ladicky and P. Torr, "Locally linear support vector machines," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 985–992, 2011.
[20] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.
[21] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and variational inference in deep latent gaussian models," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2014.
[22] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.
[23] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou, "Variational deep embedding: An unsupervised and generative approach to clustering," in *International Joint Conferences on Artificial Intelligence*, 2017.
[24] C.-C. Chang and C.-J. Lin, "Ijcnn 2001 challenge: generalization ability and text decoding," in *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222)*, vol. 2, pp. 1031–1036 vol.2, July 2001.
[25] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in Neural Information Processing Systems 28* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 3483–3491, Curran Associates, Inc., 2015.
[26] L. Baltrunas, K. Church, A. Karatzoglou, and N. Oliver, "Frappe: Understanding the usage and perception of mobile app recommendations in-the-wild," *CoRR*, vol. abs/1505.03014, 2015.
[27] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-n recommender systems," in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pp. 153–162, ACM, 2016.
[28] Y. Dauphin, X. Glorot, and Y. Bengio, "Large-scale learning of embeddings with reconstruction sampling.," in *ICML*, pp. 945–952, 2011.

Xiaoshuang Chen received his Bachelor degree from Tsinghua University in 2014. He is currently pursuing the Ph.D. degree in the Department of Electrical Engineering, Tsinghua University, Beijing, China. His research interests include recommender system techniques and optimization methods.

Yin Zheng is currently a senior researcher with Tencent AI Lab. He received his Ph.D degree from Tsinghua University in 2015. His research mainly focuses on recommender systems, AutoML and deep generative models. He serves as a PC member or reviewer for many journals and conferences in his area.

Peilin Zhao is currently a Principal Researcher at Tencent AI Lab, China. Previously, he has worked at Rutgers University, Institute for Infocomm Research (I2R), Ant Financial Services Group. His research interestes include: Online Learning, Deep Learning, Recommendation System, Automatic Machine Learning, etc. He has published over 90 papers in top venues, including JMLR, ICML, KDD, etc. He has been invited as a PC member, reviewer or editor for many international conferences and journals, such as ICML, JMLR, etc. He received his bachelor degree from Zhejiang University, and his PHD degree from Nanyang Technological University.

Zhuxi Jiang received the Bachelor's degree and the Master's degree from Beijing Institute of Technology, Beijing, China, in 2015 and 2018 respectively. His research interests include machine learning, computer vision and intelligent transportation systems.

Wenye Ma received his PhD degree in Mathematics from University California, Los Angeles in 2011. He is currently an senior researcher at Tencent AI Lab. His research mainly focuses on online learning, optimization, and recommender system techniques.

Junzhou Huang received the B.E. degree from Huazhong University of Science and Technology, Wuhan, China, the M.S. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, and the Ph.D. degree in Computer Science at Rutgers, The State University of New Jersey. His major research interests include machine learning, computer vision and biomedical informatics, with focus on the development of sparse modeling, imaging, and learning for big data analytics.