MultiHot Embedding: A Multiple Activation Embedding Model for Continuous Features in Deep Learning

Paper ID: 8432

Keywords: Feature representation, continuous variables, multiple activation embedding, deep learning

Abstract

The quality of representation learning highly drives deep learning (DL) model performance. To facilitate effective representation learning, most studies focus on neural network model designs for the embedded vectors. However, little attention has been paid to the embedding model itself (data representation before feeding into DL models), especially for continuous data commonly encountered in time-series learning tasks. The existing continuous data representation models may lead to poor model learning efficiency due to issues of infinite learning space by directly representing continuous data (e.g., normalization) or information loss and insufficient training by discretizing continuous data. To address these, this paper proposes a MultiHot Embedding method for continuous feature data representation in deep learning models. The MultiHot Embedding discretizes the continuous data into intervals and extends the one-hot embedding by allowing multiple activations of neighbour bits for discretized interval values. The multiple neighbour activation mechanism enables the MultiHot Embedding to use small interval widths for discretization which overcomes the infinite learning space and insufficient training issues and mitigates the information loss for discretization. The experiments on three different types of tasks (regression, time-series, and classification) validate the effectiveness and generalization capabilities of the proposed MultiHot Embedding method. Compared to the best baseline model for each task, the MultiHot Embedding model improves the prediction performance by 6% regardless of the learning task. Furthermore, the sensitivity analysis shows that the MultiHot Embedding is robust to the number of discretization intervals, facilitating easy practical uses.

1 Introduction

The performance of deep learning models is highly depend on the quality of representation learning (Bengio, Courville, and Vincent 2013). Plenty of efforts in the deep learning community have been made to help the models learn more effective representations that benefit the model performance. However, most of the efforts are made towards better module designs, e.g. Convolutional Neural Network (CNN) (Krizhevsky, Sutskever, and Hinton 2012) and Long-Short-Term-Memory (LSTM) (Hochreiter and Schmidhuber 1997)). Limited attention has been paid to the input data representation methods before feeding into deep learning models, i.e. how to transforming/processing the input data to facilitate more effective learning.

Based on the data type, the input features of deep learning models can be divided into two categories: discrete and continuous features. Discrete features are mainly in the form of categorical values, e.g. *Gender* (Female/Male). The embedding process has been commonly used to represent discrete features (Mikolov et al. 2013). The embedding process works as a look-up table, which maps each category to a distributed representation (i.e. a dense vector). It has advantages such as parameter reduction, interpretability, etc. Therefore, most popular deep learning models incorporate the embedding process to represent discrete inputs, for example, Transformer (Vaswani et al. 2017), BERT (Devlin et al. 2018), and GPT-3 (Brown et al. 2020).

Continuous features have numerical values (e.g. temperature, height, weight[1]). They are common in many typical deep learning tasks, such as CTR (Click-Through Rate) predictions (Zhang et al. 2021), stock price predictions (Khare et al. 2017), and travel predictions (Jiang, Ma, and Koutsopoulos 2022). In the literature, the continuous features are represented in three different ways:

- Directly Inputs (DI). The continuous values are directly fed into the model using basic preprocessing procedures (e.g. normalization).
- Discretization & Embedding (D&E). The continuous values are firstly discretized into intervals and then transformed to dense vectors through the embedding process.
- Projection. The continuous values are projected to a dense vector without discretization.

However, these feature representation models of continuous features could be sub-optimal given that they may fail to cover learning and trainning issues from the continuity and discretization. DI and Projection may lead to poor learning performance due to the infinite learning spaces problem of continuous values, while D&E fails since the discretization process triggers both SBD (Similar value But Dis-similar

[1]Note that some features, e.g. population, traffic flow volume, are commonly treated as continuous, although the original form is discrete.

embedding) and DBS (Dis-similar value But Same embedding) issues (Guo et al. 2021) (detailed in Section 2). To the best of our knowledge, there is no approach towards the continuous feature representation in deep learning models that covers all these issues mentioned above. More generally, there is a lack of an optimal method of continuous feature embedding in the deep learning community, which is as simplified but effective as the embedding process for discrete features. The paper fills this gap by proposing the MultiHot Embedding, which takes the advantages of the embedding process and addresses issues caused by directly representing continuous values (infinite learning space and insufficient training) and discretization (detailed in Section 2). We discuss the essence of continuous feature representation and argue that any method using the original values of continuous features as inputs is sub-optimal. In other words, discretization is a necessary processing step for continuous values in deep learning. Based on that, we develop the MultiHot Embedding encoding method for discretized continuous data, which reduces the risk of *SBD* and *DBS* in the discretization process while maintaining the learning efficiency, specifically in the context of deep learning. The main contribution of this study are summarized as follows:

- Explore the essence of continuous value processing in the context of deep learning, based on which the optimal representation method of continuous feature is developed.
- Propose a MultiHot Embedding method for the continuous value representation in deep learning, which overcomes the learning efficiency issue in the direct representation and mitigates the information loss for discretization.
- Evaluate the proposed method performance on different types of tasks including regression, time series and classification, and conducts the sensitivity analysis on embedding parameters.

2 Preliminaries

The continuous feature is commonly encountered in deep learning applications. Despite its importance, limited attention has been paid to the continuous data representation in deep learning studies (Guo et al. 2021). The main reason is that deep learning could directly deal with continuous variables, thus most practices use the raw continuous data as inputs with simple preprocessing steps, such as standardization (Shanker, Hu, and Hung 1996) and normalization (Singh and Singh 2020). However, many studies reported that directly using continuous data as inputs is sub-optimal for deep learning models, which lead to unsatisfactory performance given its infinite learning space (Ke et al. 2019; Guo et al. 2021; Fernández-Delgado et al. 2014). As a continuous variable could be any value within its range, it requires the deep learning model to fit distributions at infinite number of points. It causes learning complex optimal hyperplanes with a high risk of falling into local optimums (Ke et al. 2019).

Consequently, directly using continuous data can hardly be optimal. In the context of deep learning, the infinite learning space issue is triggered when the continuous form is directly used as the model input. Recently, some continuous feature embedding approaches are proposed, which directly using continuous values (without discretization) to generate a distributed representation through certain designed embedding modules (Guo et al. 2017, 2021; Song et al. 2019). We argue that these approaches are sub-optimal since the directly inputting of continuous features will encounter the infinite learning space issue, regardless of what embedding processes are designed. To address the infinite learning space issue, the continuous nature of the input data should be eliminated before they are fed into a model.

Discretization is an effective way to deal with the infinite learning space problem (Guo et al. 2021). For a given continuous feature value c, the discretization process $D(\cdot)$ projects c into one of the K predefined intervals[2]. Transforming continuous values into discrete intervals reduces an infinite learning space to a finite one (i.e. finite intervals), thus dramatically improving the learning efficiency. Another advantage of discretization is facilitating the feature representation using the embedding process. The embedding process $E(\cdot)$ project an one-hot representation of a discrete value d to a distributed representation through a learnable look-up table (Mikolov et al. 2013). The embedding exhibits advantages upon other discrete value representation methods. For example, compared to the one-hot representation, distributed representations lie in the semantic space where features with similar semantic meanings tend to have a close distance. This dramatically increase the representation capacity of the learning model.

Despite its advantages in the feature representation, the discretization of continuous data is challenging. Theoretically, the discretization process treats values in the same interval as identical. This leads to information loss, which triggers two critical issues: *SBD* (Similar value But Dis-similar discretization) and *DBS* (Dis-similar value But Same discretization) (Guo et al. 2021):

- *SBD*. Discretization may separate similar values (boundary values) into different intervals. For example, a common discretization of the *Time of Day* is dividing a day into 24 one-hour intervals (e.g. 8:00 am -8:59 am is discretized as 8) (Feng et al. 2022; Zhang et al. 2022). In this case, 8:59 am and 9:00 am will be discretized into two different intervals (i.e. 8 and 9) although they are close and share similar context information (e.g., peak hour).
- *DBS*. Discretization may group significantly different values into the same interval. Using the same example as above, 8:00 am and 8:59 am are in the same interval although they represent different context information. That is, 8:00 am is close to the off-peak hour and 8:59 am is in the peak period.

Discretizing the continuous feature is potentially optimal if the drawbacks (i.e. *SBD* and *DBS* problems) are appropriately handled. Intuitively, using a smaller interval width in discretization can reduce the *SBD* and *DBS* problems. But the smaller interval width means a larger number of intervals. This lead to more distributed training samples among

[2]In this paper, we use the uppercase C to represent a continuous feature, and the lowercase c to represent a specific value of C.

intervals, causing the model insufficiently trained for low-frequency intervals (i.e. intervals with few training samples).

3 MultiHot Embedding

As discussed above, the discretization of continuous variables is a trade-off between the information loss and model learning efficiency. To achieve that, we propose a MultiHot Embedding method which addresses the issues of the infinite learning space aroused from the continuity and the *SBD* and *DBS* from the value discretization, as well as takes advantage of the existing embedding method for the easy integration into a wide range of deep learning structures.

3.1 Methodology

As discussed, using a small discretizing interval width could reduce the *SBD* and *DBS* issues. Therefore, the MultiHot Embedding utilizes relatively a small interval width compared to the conventional discretization method. For example, in the case of the *Time of Day* discretization, the interval width could be set as 10 minutes (compared to the 1 hour interval in conventional methods). To clarify the illustration, we denote the discretization using a normal interval width l_N as $D_N(\cdot)$, and $D_S(\cdot)$ to represent the one with a small interval width $l_S \ll l_N$. After discretization, the range of feature C is divided into K intervals $\{I_1, I_2, \ldots, I_K\}$. For interval $I_i \in \{I_1, I_2, \ldots, I_K\}$, we denote its range as $[c_i, c_{i+1}]$, where c_i and c_{i+1} are the upper and lower bounds of I_i, respectively. Note that the intervals are arranged in an ordinal sequence, i.e. $c_1 < c_2 < \cdots < c_K$. For a specific continuous value c within an interval I_i, we denote its one-hot representation under $D_S(\cdot)$ as:

$$\mathbf{c}_o^S = onehot(D_S(c)) = [o_1, o_2, \ldots, o_K] \tag{1}$$

$$o_t = \begin{cases} 1, & t = i \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$$\mathbf{E}_{\mathbf{c}_o^S} = \mathbf{c}_o^S \mathbf{M} \tag{3}$$

where $\mathbf{c}_o^S \in R^{1\times K}$ is the one-hot representation of c under $D_S(\cdot)$, o_t the t^{th} ($t \in [1, K]$) element of $\mathbf{c}_o^S$, and i the index of the discretization interval. $\mathbf{E}_{\mathbf{c}_o^S} \in R^{1\times h}$ is the corresponding embedding of $\mathbf{c}_o^S$, $\mathbf{M} \in R^{K\times h}$ the learnable look-up matrix, and h the embedding size.

As we use a small interval width l_S, $\mathbf{E}_{\mathbf{c}_o^S}$ is sub-optimal due to the insufficient training issue. To address that, we propose the MultiHot representation of c as:

$$\mathbf{c}_m^S = MultiHot(D_S(c)) = [m_1, m_2, \ldots, m_K] \tag{4}$$

$$m_t = \begin{cases} 1, & max(i-m, 1) \le t \le min(i+m, K) \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

$$\mathbf{E}_{\mathbf{c}_m^S} = \mathbf{c}_m^S \mathbf{M} \tag{6}$$

where m is a non-negative integer predefined specifying the degree of neighbourhood information sharing and m_t is the t^{th} ($t \in [1, K]$) element of $\mathbf{c}_m^S$. $\mathbf{E}_{\mathbf{c}_m^S} \in R^{1\times h}$ is the MultiHot embedding of the continuous value c

The MultiHot representation is transformed from one-hot by additionally activating the m-neighbors of i (i.e. set m-neighbors of i as 1). For example, suppose an one-hot representation of $D_S(c)$ is $[0, 0, 0, 0, 1, 0, 0, 0]$ where $i = 5$, then the corresponding MultiHot representation is $[0, 0, 1, 1, 1, 1, 1, 0]$ with $m = 2$. Note that when $m = 0$, the MultiHot representation is identical to the one-hot.

3.2 Properties

We discuss the MultiHot Embedding properties and illustrate them using an example.

Property 1: *The MultiHot Embedding retains the numerical relationship under the discrete context leveraging the "overlapping" concept, which addresses the root causes of SBD and DBS issues in discretizing and representing continuous features.*

Using the discretization of the *Time of Day* as an example, as discussed before, the embeddings of 8:59 am and 9:00 am are different under an one-hour discretization. The MultiHot Embedding uses a small discretization interval width (e.g. 10 minutes) and the m-neighbor activation mechanism. Therefore, most of the "1" in their MultiHot representations are overlapped resulting in similar embeddings (Figure 1), which addresses the *SBD* issue. Following the same analysis logic, given the small discretization length, the MultiHot Embeddings of 8:00 am and 8:59 am are rather different as what they are supposed to be in representing different context information. Therefore, the *DBS* problem is overcame to a certain level. In all, the MultiHot Embedding captures numerical relationships between continuous values in representing their context difference by overlapping their one-hot representations to m-neighbors. The traditional discretization & embedding models basically lose the numerical feature of continuous variable values.

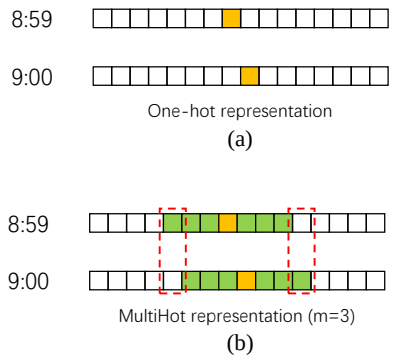

Figure 1: One-hot and MultiHot representations for 8:59 and 9:00, respectively. One-hot representation of 8:59 and 9:00 are totally different although they are very similar in the context (Figure 1 (a)), while in MultiHot, most of the activated elements (colored boxes) are overlapped, resulting in similar context values after embedding.

Property 2: *The MultiHot Embedding leads to more efficient training than the one-hot, thus allowing for using a small discretization interval width.*

As mentioned, the *SBD* and *DBS* problems can be reduced by using a small interval width in discretization. However, conventional discretization approaches cannot handle small

interval widths due to the consequent insufficient training issue. For example, if the 10-minutes interval width is used in the *Time of Day* discretization, the one-hot embedding of a large number of intervals may not be well-trained due to inadequate training sample sizes. In the MultiHot Embedding representation, the training of a specific discretization interval will activate the bit itself and its m neighbors, thus the embedding of each discretization interval gets more chance to be trained (explicitly $2m$ times more than the one-hot). In other words, the MultiHot Embedding functions equivalent to increasing the number of training samples for each discretization interval.

In summary, the MultiHot Embedding model would conceptually overcome the main concerns of representation of continuous features in deep learning, including the infinite learning space problem with continuous data, the *SBD* and *DBS* problems from continuous data discretization, and insufficient training with small discretization interval width. We will further validate these in the case study using three typical types of learning tasks.

3.3 Interpretation

In the one-hot representation, each element of the vector can be explained as whether the value belongs to the corresponding interval (1 for belonging and 0 otherwise). To better illustrate the MultiHot representation, we also provide a corresponding semantic explanation. Let $D_S(C) = \{[c_1, c_2], [c_2, c_3], \ldots, [c_{K-1}, c_K]\}$ be the discretization of continuous feature C. The t^{th} element in $\mathbf{c}_m^S$ represents whether the value c belongs to interval $[c_{t-m}, c_{t+1+m}]$ (1 for belonging and 0 otherwise). For comparison, we also give the corresponding definition of one-hot representation: the t^{th} element in $\mathbf{c}_o^S$ represents whether the value c belongs to interval $[c_t, c_{t+1}]$. Figure 2 show the difference between two representations. From this perspective, the main difference between one-hot and MultiHot representations is that the adjacent intervals in one-hot are disjoint, while in MultiHot, they are overlapped.

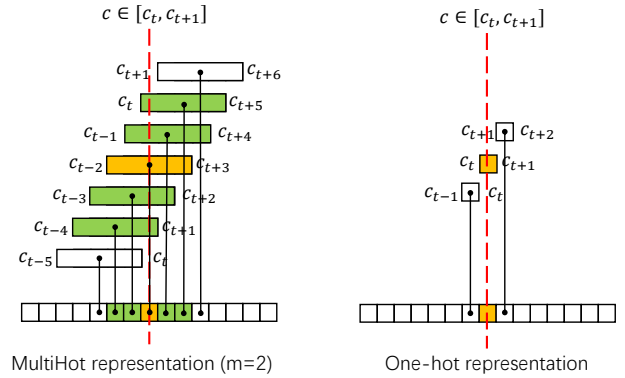

Figure 2: Explanation of MultiHot and one-hot representations. The values specify the upper and lower bounds of each interval. The black lines connect the vector elements and the corresponding interval they represent. The colored intervals contains the given value c, of which their corresponding values are set as 1 in the vector. For one-hot, as the intervals are non-overlapped, only one element is 1 in the vector.

4 Empirical Evaluation

4.1 Experimental Setup

To evaluate the proposed MultiHot Embedding, we conduct experiments on three tasks using three different datasets.

Task 1: *California housing price prediction*. Given eight attributes of building blocks in California, USA (e.g. income and house age in the block group), predicting the corresponding average housing price. All the eight attributes are continuous features. The dataset is open-source in the sklearn[3] package. The ratio of the training and testing is set as 80:20.

Task 2: *Metro passenger demand prediction*. Based on historical passenger demand using metro systems, predicting the passenger demand of the next time period. The dataset is collected from January 1 to November 30, 2018 in Hong Kong, China. The passenger demand is aggregated in 15-minute periods. In the experiment, we use previous 4 steps (i.e passenger demand at previous 4 15-minutes periods) to predict the passenger demand of the next 15 minutes. We randomly select 200 days and construct the training set, and the other 50 days as the testing set.

Task 3: *Floor type prediction*. Based on the motion state of a robot, predicting which one of the nine floor types (e.g. carpet, tile, concrete) it moves on. This task is from the Kaggle Competition[4]. The data is collected from inertial measurement units sensors monitoring the robot's motion state for 128 time steps while it is moving on one specific floor. At each step, the sensor collects the orientation (as a quaternion), angular velocity, and linear acceleration of the robot.

We choose these tasks as they are representative for different types of prediction tasks. Task 1 is a regression task with multi-dimensional inputs, tasks 2 and 3 are both time-series prediction tasks with single- and multi-dimensional inputs at each step, respectively. Table 1 summarizes the statistics of these datasets.

Metrics Tasks 1 and 2 are regression problems, we use *RMSE* and *sMAPE* to quantify the performance:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \tag{7}$$

$$sMAPE = \frac{100\%}{n} \sum_{i=1}^{n} \frac{|\hat{y}_i - y_i|}{(|\hat{y}_i| + |y_i|)/2} \tag{8}$$

where n is the number of observations, y_i and $\hat{y}_i$ are the i^{th} observed and predicted values, respectively.

For the classification task 3, the *Accuracy* (Accu) is used:

$$Accuracy = \frac{Y_C}{Y_N} \tag{9}$$

[3]https://scikit-learn.org/stable/datasets/real_world.html\#california-housing-dataset

[4]https://www.kaggle.com/competitions/career-con-2019/overview

	Data Form	#Time Steps	#Features	#Instances	Problem Type
Task 1	Multi-dimensional data	NaN	8	20640	Regression
Task 2	Time-series	4	1	23000	Regression
Task 3	Multi-dimensional time-series	128	10	3809	Classification

Table 1: Datasets details

where Y_C and Y_N are the number of correct and total classifications, respectively.

Baselines For each task, we construct a neural network model containing two parts: Feature Extraction and Prediction modules. The Feature Extraction module learns input representations through different continuous feature representation methods, while the Prediction module takes inputs of extracted representations and generates the desired prediction. Figure 3 shows the overall structure of the tested tasks.

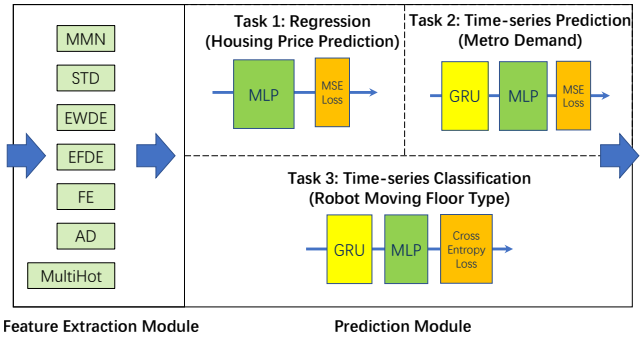

Figure 3: Model structure for each task

Feature Extraction Module. To demonstrate the effectiveness of the proposed MultiHot Embedding, we compare it with 6 other continuous representation methods:

- Min-Max Normalization (MMN). Rescaling the range of continuous feature C to $[0, 1]$:

$$c' = \frac{c - c_{min}}{c_{max} - c_{min}} \tag{10}$$

- Standardization (STD). Making the values of continuous feature C have zero-mean and unit-variance:

$$c' = \frac{c - \bar{c}}{\sigma} \tag{11}$$

  where $\bar{c}$ and σ denote the mean and variance of feature C, respectively.
- Equal Width Discretization & Embedding (EWDE). The range of the continuous feature C is divided into K intervals with the same width. The values are then transformed to the corresponding embedding vectors.
- Equal Frequency Discretization & Embedding (EFDE). The range of the continuous feature C is divided into K intervals with the same number of samples. The values are then transformed to the corresponding embedding vectors.
- Field Embedding (FE) (Guo et al. 2017). The continuous value c is directly embedded to a dense vector without discretization:

$$\mathbf{c}' = c\mathbf{M}_F \tag{12}$$

  where $\mathbf{M}_F \in R^{1\times h}$ is the look-up vector, $\mathbf{c}'$ is the Field Embedding of c.
- AutoDis (AD) (Guo et al. 2021). The state-of-the-art continuous feature representation method which combines a Meta-Embedding and Automatic Discretization mechanisms. Details of this method refers to (Guo et al. 2021).

These methods belong to three categories discussed in Section 1. The MMN and STD are DI models, i.e. only basic prepossessing are conducted without the embedding process. The EWDE and EFDE belong to the D&E model which follows the discretization and embedding pipeline. The FE and AD are Projection models which use the continuous input values and the embedding process.

Prediction Module. For task 1, we use Multilayer Perceptron (MLP) as the backbone of the prediction module. As tasks 2 and 3 are time-series prediction problems, the Gated Recurrent Unit (GRU) is added. Given the main purpose is to evaluate feature representations, we choose those models considering they are representative and simple. Complex models (e.g. Transformer) are more likely to be impacted by the parameter tuning which may entangle the influence from different representations.

Training Details The models for regression problems (task 1 and 2) are trained with MSELoss[5], while CrossEntropy[6] Loss is utilized for classification (task 3). The learning rate is searched from {10e-6, 10e-5,...,10e-1} with the Adam optimizer. We conduct multiple experiments to find the best parameter setting of the representation methods. The parameter searching for AD is based on the settings specified in the original paper (Guo et al. 2021). Table 2 summarizes the model parameter settings that achieve the best performance in each task. The experiments are running on Ubuntu 18.03 with devices including a GTX 1080 Ti GPU and an Xeon E5-2650V3 CPU.

4.2 Overall Performance

Table 3 shows the performance comparison results of different representation models in the three prediction tasks. The proposed MultiHot Embedding outperforms all the baselines

[5]API from: https://pytorch.org/docs/stable/generated/torch.nn.MSELoss.html?highlight=mseloss#torch.nn.MSELoss

[6]API from: https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html

	GRU	MLP	emb_siz	Representation Settings			
				EFDE	EWDE	AD	MultiHot
Task 1	NaN	[1600,100,50,25,1]	200	K=50	K=300	K=200	K=200,m=5,dis=EFD
Task 2	[300]	[300,50,50,1]	200	K=200	K=140	K=175	K=510,m=16,dis=EFD
Task 3	[128,128]	[128,9]	100	K=20	K=45	K=40	K=200,m=5,dis=EWD

Table 2: Parameter settings for the models in each task that achieve the best performance. emb_siz denotes the length of the embedding vectors, K denotes the number of intervals used in the discretization. Note that in AD, K denotes the number of Meta-Embeddings. m is the number of activated neighbors in MultiHot representation and dis denotes the discretization method, in which EFD and EWD represents equal frequency and equal width discretization, respectively.

Category	Method	Task 1		Task 2		Task 3
		RMSE	sMAPE(%)	RMSE	sMAPE(%)	Accu(%)
DI	MMN	0.6569	23.05	100.09	51.55	51.47
	STD	0.5433	18.28	99.31	52.01	70.06
D&E	EWDE	0.5313	17.21	<u>96.72</u>	51.93	84.76
	EFDE	<u>0.4858</u>	<u>16.40</u>	101.63	<u>50.75</u>	83.71
Projection	FE	0.5352	18.21	97.48	51.13	77.83
	AD	0.5225	17.20	97.21	51.22	<u>85.40</u>
MultiHot		**0.4536**	**15.18**	**90.54**	**50.05**	**90.65**
Improve (%)		6.62	7.44	6.39	1.38	6.15

Table 3: Performance comparison results for different continuous representation methods in each task. The underline values indicate the performance of the best baseline model in different tasks. The 'Improve' specifies the performance improvement (in percentage) of MultiHot Embedding compared to the best baseline performance.

in all the tested tasks with significant improvements. It consistently improves the prediction performance 6% over the base baseline models in different tasks, which indicates a strong generalization ability and robustness of the proposed continuous feature representation method.

Among those baselines, the DI approach, i.e. MMN and STD, performs the worst among the evaluated continuous representation models. It indicates the weakness of directly using continuous values as inputs in deep learning models. It implicates that the preprocessing of continuous variables is a necessity in the deep learning context (instead of using raw values). The projection and D&E methods performs generally better than the DI models. Although the discretization may lose certain information, the D&E model still achieves the best performance in tasks 1 and 2. This indicates that the benefit of reducing the learning complexity overwhelms the value of having precise information. Compared to the D&E methods, the MultiHot Embedding model boosts the prediction performance. It is worth to note that MultiHot Embedding achieves its best performance with a larger number of discretization intervals K than D&E models (e.g. 50 vs 200 in task 1, 200 vs 510 in task 2 in Table 2). This shed lights on the MultiHot Embedding's ability on handling the small interval width discretization in mitigating the *SBD* and *DBS* problems.

The AD performs the best in task 3 among baseline models, which is a classification problem with multi-dimensional time-series inputs. Compared to other baselines, the AD has a more complex structure with the Auto-discretization and Meta-Embedding mechanisms facilitating more efficient feature extractions in complicated tasks. However, its performance improvement over other baseline models is not that significant (85.40% compared to 84.76% from EWDE). The main reason is that the AD still uses the continuous value as inputs, thus suffering from the infinite learning space issue even with well-designed and complex model structures. The proposed MultiHot Embedding, on the other hand, has a very simple structure but exhibits a highly significant improvement on the classification accuracy compared to AD (6.15%).

In all, the proposed MultiHot Embedding performs well for feature representation of continuous features in deep learning models. Also, it is simple for the representation construction and easy to be incorporated in existing deep learning frameworks.

4.3 Sensitivity Analysis

We evaluate the impact of different discretization settings on the model performance. Figure 4 summarizes the corresponding model performance under different number of discretization intervals.

The performance of traditional D&E methods (i.e. EFDE or EWDE) firstly increases and then decreases as the number of intervals gradually increases regardless of the tested task. It indicates that they are not capable of handling small discretization intervals for a sufficient model training. Their performance are sub-optimal with either a small or large number of discretization intervals. That is, the information loss issue is serious with few intervals leading to *SBD* and *DBS* problems, while too many intervals leads to the insufficient training issue, both resulting in unsatisfied performance. This indicates that the performance of D&E method is more likely to be fluctuated since they are sensitive to the parameter setting. In other words, searching for an optimal

setting of the traditional D&E methods is exhausted. The performance of MultiHot Embedding keeps increasing even when the number of intervals reaches a relatively large value (e.g. $K = 500$ for Task 2). This highlights the advantage of MultiHot Embedding over discretizating interval lengths, i.e. mitigating *SBD* and *DBS* problems given its capability in handling a small interval width in discretization.

5 Related Work

5.1 Representation Learning

The essence of deep learning models is automatically learning latent representations of input features. The representation learning is important since it can extract general priors about the physical world and data generation process, i.e., priors that are not task-specific but would be likely to be useful (Bengio, Courville, and Vincent 2013). For example, CNN (Krizhevsky, Sutskever, and Hinton 2012), the most widely used deep leaning model in the image processing, learns gradually the abstracted image representations layer by layer. Explicitly, the basic features, such as colors, edges, etc., are learned by shallow layers and synthesized to more specific representations as layers go deeply (Krizhevsky, Sutskever, and Hinton 2012). For the NLP models, such as Word2vec (Mikolov et al. 2013), ELMo (Peters et al. 2018), Transformer (Vaswani et al. 2017), endeavor to learn distributed representations of words and sentences in the semantic space, in which similar semantic meaning exhibits a close distance. However, limited attention has been paid to the input feature representation design, especially feature representation for continuous data.

5.2 Discretization

Discretization is a commonly used pre-process method for continuous input features (Garcia et al. 2013). Many discretization methods were proposed in the literature, including the equal width and frequency discretization schemes (tested in Section 4), and approaches based on theories such as information entropy (Dougherty, Kohavi, and Sahami 1995), statistical $\mathcal{X}^2$ test (Kerber 1992; Liu and Setiono 1997), likelihood (Wu 1996; Boullé 2006), rough set (Nguyen and Skowron 1995; Zhang, Hu, and Jin 2004), etc. However, these approaches suffer from the *SBD* and *DBS* problems since they cannot handle small interval widths due to the insufficient training issue. It is worth to mention that the fuzzy discretization (Shanmugapriya et al. 2017) also considers a value's membership of multiple intervals. However, the fuzzy logic only fits the dataset with a vagueness nature, thus it is not generic to a wide range of deep learning tasks.

6 Conclusion

In this paper, we propose a MultiHot Embedding model for the representation of continuous features in deep learning models. The MultiHot Embedding use a simple but effective mechanism to add a m-neighbour activation mechanism upon the OneHot representation. It addresses three main issues of continuous feature representations, including the infinite training space from continuous data, the *SBD* & *DBS* problems from discretization, and insufficient training with small discretization interval widths. The multiple activation mechanism enables the MultiHot Embedding to utilize relatively a small interval width during discretization which overcomes the insufficient training problem. The experiments on three different type of tasks (regression, time series and classification) in areas of housing, transport and robotics validates the effectiveness and generalization capabilities of the proposed continuous feature representation model. Compared to the best baseline models, the MultiHot Embedding model significantly improves the prediction performance by 6% regardless of the learning task.

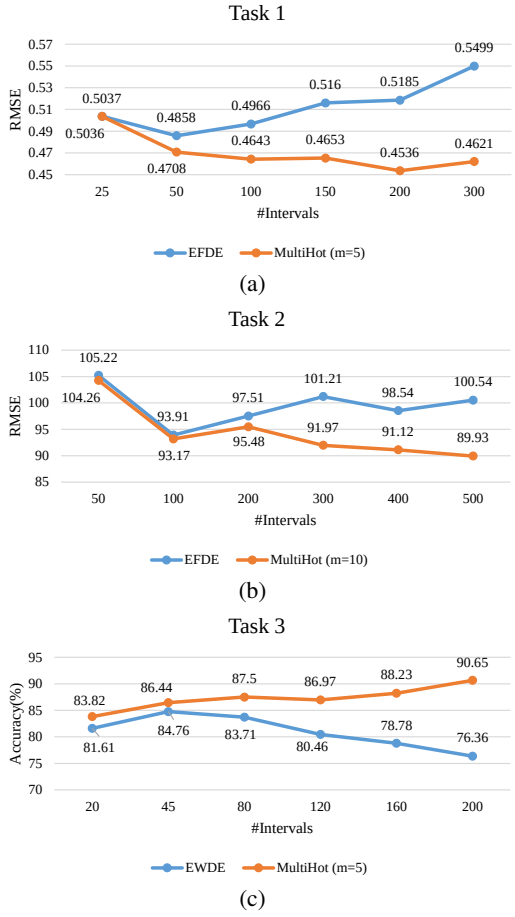

Figure 4: Sensitivity analysis of the number of discretization intervals. The #Intervals denotes the number of intervals used in the model. The traditional D&E methods achieve the best performance with a small number of intervals and its performance degrades due to the lack of enough training samples at each interval (i.e. insufficient training issue). For the MultiHot Embedding, its performance consistently improves even it reaches a relatively large number of discretization intervals.

References

Bengio, Y.; Courville, A.; and Vincent, P. 2013. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35: 1798–1828.

Boullé, M. 2006. MODL: A Bayes optimal discretization method for continuous attributes. *Machine Learning*, 65: 131–165.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. arXiv:2005.14165.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805.

Dougherty, J.; Kohavi, R.; and Sahami, M. 1995. Supervised and Unsupervised Discretization of Continuous Features. *Machine Learning Proceedings 1995*, 194–202.

Feng, J.; Li, Y.; Yang, Z.; Qiu, Q.; and Jin, D. 2022. Predicting Human Mobility With Semantic Motivation via Multi-Task Attentional Recurrent Networks. *IEEE Transactions on Knowledge and Data Engineering*, 34: 2360–2374.

Fernández-Delgado, M.; Cernadas, E.; Barro, S.; and Amorim, D. 2014. Do we need hundreds of classifiers to solve real world classification problems? *The journal of machine learning research*, 15(1): 3133–3181.

Garcia, S.; Luengo, J.; Sáez, J. A.; López, V.; and Herrera, F. 2013. A Survey of Discretization Techniques: Taxonomy and Empirical Analysis in Supervised Learning. *IEEE Transactions on Knowledge and Data Engineering*, 25: 734–750.

Guo, H.; Chen, B.; Tang, R.; Zhang, W.; Li, Z.; and He, X. 2021. An Embedding Learning Framework for Numerical Features in CTR Prediction. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery amp; Data Mining*, KDD '21, 2910–2918. New York, NY, USA: Association for Computing Machinery.

Guo, H.; Tang, R.; Ye, Y.; Li, Z.; and He, X. 2017. DeepFM: A Factorization-Machine Based Neural Network for CTR Prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, 1725–1731. AAAI Press.

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation*, 9: 1735–1780.

Jiang, W.; Ma, Z.; and Koutsopoulos, H. N. 2022. Deep learning for short-term origin–destination passenger flow prediction under partial observability in urban railway systems. *Neural Computing and Applications*, 34: 4813–4830.

Ke, G.; Xu, Z.; Zhang, J.; Bian, J.; and Liu, T.-Y. 2019. 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*, 384–394. New York, NY, USA: ACM.

Kerber, R. 1992. Chimerge: Discretization of numeric attributes. In *Proceedings of the tenth national conference on Artificial intelligence*, 123–128. San Jose California: NCAI.

Khare, K.; Darekar, O.; Gupta, P.; and Attar, V. Z. 2017. Short term stock price prediction using deep learning. In *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, 482–486. Bangalore, India: IEEE.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

Liu, H.; and Setiono, R. 1997. Feature selection via discretization. *IEEE Transactions on Knowledge and Data Engineering*, 9: 642–645.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781.

Nguyen, S. H.; and Skowron, A. 1995. Quantization Of Real Value Attributes - Rough Set and Boolean Reasoning Approach. In *Proc. of the Second Joint Annual Conference on Information Sciences*, 34–37. Wrightsville Beach, North Carolina: JCIS.

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. arXiv:1802.05365.

Shanker, M.; Hu, M.; and Hung, M. 1996. Effect of data standardization on neural network training. *Omega*, 24: 385–397.

Shanmugapriya, M.; Nehemiah, H. K.; Bhuvaneswaran, R.; Arputharaj, K.; and Sweetlin, J. D. 2017. Fuzzy Discretization based Classification of Medical Data. *Research Journal of Applied Sciences, Engineering and Technology*, 14: 291–298.

Singh, D.; and Singh, B. 2020. Investigating the impact of data normalization on classification performance. *Applied Soft Computing*, 97: 105524.

Song, W.; Shi, C.; Xiao, Z.; Duan, Z.; Xu, Y.; Zhang, M.; and Tang, J. 2019. AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 1161–1170. New York, NY, USA: ACM.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, 6000–6010. Red Hook, NY, USA: Curran Associates Inc.

Wu, X. 1996. A Bayesian Discretizer for Real-Valued Attributes. *The Computer Journal*, 39: 688–691.

Zhang, G.; Hu, L.; and Jin, W. 2004. Discretization of Continuous Attributes in Rough Set Theory and Its Application. In *Proceedings of the First International Conference on Computational and Information Science*, CIS'04, 1020–1026. Berlin, Heidelberg: Springer-Verlag.

Zhang, P.; Ma, Z.; Weng, X.; and Koutsopoulos, H. N. 2022. Recovering the Association Between Unlinked Fare Machines and Stations Using Automated Fare Collection Data in Metro Systems. *Transportation Research Record: Journal of the Transportation Research Board*, 2676: 718–731.

Zhang, W.; Qin, J.; Guo, W.; Tang, R.; and He, X. 2021. Deep Learning for Click-Through Rate Estimation. arXiv:2104.10584.