ESANN 2023 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 4-6 October 2023, i6doc.com publ., ISBN 978-2-87587-088-9. Available from http://www.i6doc.com/en/.

Graph-based Categorical Embedding

Weiwei Wang[1] and Stefano Bromuri [2] and Michel Dumontier [1]

1- Maastricht University - Institute of Data Science
Paul-Henri Spaaklaan 1, 6229 EN Maastricht -The Netherlands

2- Open University - Computer Science Department
Valkenburgerweg 177, 6419 AT Heerlen -The Netherlands

Abstract. Categorical features are a challenge for most machine learning algorithms that only accept numerical vectors in input. However, the emergence of graph neural networks is revolutionizing the application of machine learning models to traditional data sets. This is thanks to the possibility of introducing graph relationships amongst features and samples. In this contribution, we describe an algorithm leveraging the assignment matrix of a DiffPool graph neural network to calculate embeddings for categorical features, using as an adjacency matrix the co-occurrence matrix between the categorical values and as nodes feature the one hot encoded categorical values. We show that the algorithm proposed is scalable and presents a competitive performance in three publicly available data sets presenting both numerical and categorical values.

1 Introduction

Graph neural networks (GNNs), introduced in [1, 2] are quickly finding applications in standard machine learning tasks, as researchers finds new ways to represent tabular data in graphical structures. It is common to see categorical features in data sets. Categorical features represent finite discrete values, which can be textual or numeric, for example a country code ("31", "86", "61") or an economic status ("low income","middle income", "high income"), with a further possible subdivision of the features into ordinal and nominal. Most ML algorithms require numerical features in input. Therefore, it is essential to transform the categorical features into numerical features so that algorithms can leverage the information included in these features. There are many approaches being used to convert the categories numerically and extract the relationship and semantic amongst categories [3]. There are several terms to refer to these techniques, such as "embeddings" [4] ,"distribution representation" [5] or simply "encoding" approaches [6]. A recent survey [7] has discussed the current existing techniques to deal with categorical features in a ML context, including solutions involving Word2Vec [8] and transfer learning with transformer models [9].

The main contribution of this paper is to present a new algorithm to extract categorical features embeddings by means of the Differential Pooling (DIFFPool) GNN clustering algorithm [10] leveraging on the co-occurrence matrix of the categorical features values. DIFFPool identifies an assignment matrix that effectively reshapes the adjacency matrix of the original problem, by grouping nodes in super concepts. Other works have used DIFFPool features to perform classification tasks: the contribution in [11] proposes an approach called FPool,

ESANN 2023 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 4-6 October 2023, i6doc.com publ., ISBN 978-2-87587-088-9. Available from http://www.i6doc.com/en/.

which is an improvement on the basic method adopted in DIFFPool to extract the powerful features for the downwards classification task; the contribution presented in [12] uses graph embeddings calculated with DIFFPool to detect controversy in text; the contribution presented in [13] uses multiple channels of DIFFPool to extract multiple node clusters to increase the classification performance of the GNN. With respect to such works, this contribution differs as it focuses on using the assignment matrix of DIFFPool to extract embeddings from categorical data, rather than focusing on optimizing the classification task.

The rest of this paper is organized as follows: Section 2 discusses the data used for the experimentation; Section 3 discusses the techniques and main algorithm sued in this work; Section 4 evaluates the results of the experimentation; Section 5 concludes this contribution discussing relevant future research directions.

2 Data

This Section discusses the data sets used in this contribution and the number of categorical variables available in such data sets. Three UCI data sets [14] were used for the experimentation.
Banking Data Set: we chose the Portuguese banking institution relational data from the the UCI repository, denoted as D_1 in this study. The data was collected from $41,188$ bank clients from May 2008 to November 2010 and published in February 2012. The original authors of the data set performed pre-processing, resulting in a data set comprised of 10 categorical features that exhibit both ordinal and nominal properties, as well as 10 numerical features. The goal is to classify whether a client will subscribe to a bank term deposit or not.
Adult Income Data Set: this data set was extracted from the census bureau database in the United States from the UCI repository, denoted as D_2 in this study. It contains both continuous and categorical features. There are, in total, 48842 instances and 14 input features, including 6 continuous features and 8 ordinal and nominal categorical features. The target variable indicates whether a person earns over $50,000$ USD per year.
Census Income Data Set: this data set comprises census data that has been weighted and extracted from the Current Population Surveys conducted by the U.S. Census Bureau in 1994 and 1995, denoted as D_3 in this study. It consists of 199,533 instances and includes 41 demographic and employment-related features, of which 9 are continuous, 1 is a date-time feature, and 31 are categorical features. Notably, the target variable in D_3 is used to indicate total person income, which differs from the target variable in D_2 that indicates adjusted gross income. Thus, D_3 behaves differently from the original Adult dataset with respect to the target variable.

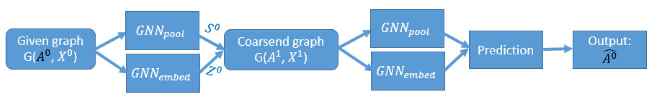

Fig. 1: Forward Pass of the embedding network based on DIFFPool model, using an autoencoder approach where we learn to reconstruct the adjacency matrix.

3 Methods

This Section discusses the main methods used in this contribution, focusing on the main algorithm to extract embeddings from categorical data, and discussing the baseline algorithms for the comparison performed in Section 4.

3.1 Graph Neural Networks and Differential Pooling

A graph convolutional neural network layer performs the following operation:

$$X^{(l+1)} = \sigma\left(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}X^{(l)}W^{(l)}\right)$$

Assume the given graph has n nodes at layer l and each node is represented by a d dimensional vector. Then $X^{(l)} \in R^{n\times d}$ is a node feature matrix (or *embeddings*). $W^{(l)} \in R^{d\times d}$ is the weight parameters with which we transform the input features into messages ($X^{(l)}W^{(l)}$). To the adjacency matrix $A \in R^{n\times n}$, we add the identity matrix so that each node sends its own message also to itself: $\hat{A} = A+I \in R^{n\times n}$. Finally, to take the average instead of summing, we calculate the matrix $\hat{D} \in R^{n\times n}$ which is a diagonal matrix with $d_{i,i}$ denoting the number of neighbors node i has, σ represents an arbitrary activation function.

Differential pooling is a graph pooling algorithm that, given a graph in input, in terms of an X matrix of node embeddings (one vector per node), and an adjacency matrix A, can calculate a soft assignment $S \in R^{n\times m}$ and the new node embeddings $Z \in R^{n\times d}$ that aggregates the nodes of the graph to super-nodes according to the following mappings where $m < n$:

$$X^1 = {S^0}^\top Z^0 \in R^{m\times d} \tag{1}$$
$$A^1 = {S^0}^\top A^0 S^0 \in R^{m\times m} \tag{2}$$

Where the coefficients of the matrix S are learned by backpropagation, with S having precisely one row per node in input and a number of columns defined as a design parameter. Algorithm 1 illustrates the pseudocode to calculate embeddings for the categorical values of a data set, by means of an AutoDIFFPool network as illustrated in Fig. 1. The A_c matrix is a square matrix $A_c \in R^{n\times n}$ where N is the amount of categorical values in the data set. Such a matrix is calculated by counting the co-occurrence of the pairs of categorical values and then normalizing the final result. We then use this matrix as an adjacency matrix between the categorical feature values, treated as node in a graph.

Algorithm 1 Categorical Graph Embeddings

```
Require: Data set $D$, with $D_c$ being the subset of categorical columns in $D$
Require: AutoDIFFPool an autoencoder with DIFFPool operators
Ensure: $C$ as the set of categorical values in $D_c$ of cardinality $n$
Ensure: $S$ a randomly initialized matrix in $\mathbb{R}^{n \times m}$, with $m < n$
  $A_c \leftarrow$ Compute count of each pair $(c_i, c_j), i \neq j, c_i, c_j \in C$, appearing in $D_c$
  $A_c \leftarrow$ normalize($A_c$)
  for $i \in [0, Epochs]$ do
      $\hat{X} \leftarrow$ One-hot encode values in $C$
      $S, \hat{A} \leftarrow AutoDIFFPool(A_c, \hat{X})$ with $A_c$ as the target
      $error \leftarrow \|A - A_c\|_2$
      $backpropagate(AutoDIFFPool, error)$
  end for

  return $S$
```

AutoDIFFPool neural network calculates the matrix of soft assignments S that assigns each of the categorical values to a set of super nodes. In this case we treat the super nodes as being the effective embeddings representing each of the categories. Each of the items in a data set may still have multiple categorical values. As a consequence we concatenate each of the embeddings in vectorial format to be able to apply machine learning methods. The network is trained to reconstruct the co-occurrence adjacency matrix in input, following an autoencoder schema, with a loss specified in terms of the l_2 norm between the adjacency matrix A_c in input and the predicted matrix $\hat{A}$.

3.2 Baseline methods

For the purpose of performing a comparison with available methods to deal with categorical variables, we selected a subset of techniques from [7] and included the Node2Vec technique [15] based on random walks to perform a comparison with the embeddings calculated with the AutoDIFFPool network. The following techniques were therefore selected: label encoding, one hot encoding, Word2vec using a concatenation of the embeddings, Node2Vec technique applied on the normalized co-occurrence matrix and then concatenating the embeddings.

4 Results

This Section discusses the results concerning the comparison of the embeddings calculated by means of the soft assignment proposed by the DIFFPool autoencoding network discussed in the previous Section and the baseline methods. The DIFFPool network has hidden dimension fixed to 64 neurons in each of the hidden layers. For each of the methods that produce categorical embeddings, we performed a linear parameters search with step 10 in the range between dimension 10 and 50, discovering that in D1, D2, D3, the best results are obtained with embeddings of dimension 30. The Node2Vec algorithm has also parameters concerning the number of walks, for which we used a step of 50 between 50 and 300 walks, discovering that Node2Vec performance saturates at 200 walks. We kept fixed to default values the parameters of Node2Vec that regulate the random walks. Table 1 shows a summary in terms of F1, Precision Recall and AUC of each of the algorithm employed in this paper in the selected data sets, using logistic regression as the classifier.

D1 Data Set					
Classifier	**Accuracy**	**Precision**	**Recall**	**F1_score**	**ROC_AUC**
OHE	0.90	0.79	0.66	0.70	0.917
LE	0.90	0.79	0.65	0.75	0.892
W2V	0.91	0.79	0.67	0.71	0.917
N2V	0.91	0.79	0.67	0.71	0.917
GCE	0.91	0.81	0.70	0.74	**0.937**
D2 Data Set					
Classifier	**Accuracy**	**Precision**	**Recall**	**F1_score**	**ROC_AUC**
OHE	0.84	0.79	0.76	0.78	0.899
LE	0.81	0.77	0.69	0.55	0.849
W2V	0.84	0.80	0.75	0.77	0.895
N2V	0.83	0.78	0.71	0.74	0.887
GCE	0.85	0.81	0.76	0.78	**0.902**
D3 Data Set					
Classifier	**Accuracy**	**Precision**	**Recall**	**F1_score**	**ROC_AUC**
OHE	0.95	0.82	0.65	0.70	0.933
LE	0.95	0.84	0.59	0.63	0.909
W2V	0.95	0.82	0.64	0.69	**0.934**
N2V	0.95	0.83	0.61	0.66	0.927
GCE	0.95	0.82	0.64	0.69	**0.934**

Table 1: Macro average performance metrics of binary classifiers Using the proposed categorical embeddings in data sets D1, D2 and D3. The algorithm used to classify the data is logistic regression. The best area under the roc curve (ROC_AUC) results are highlighted in bold and are underlined if they are statistically significant ($pvalue < 0.05$), with respect to a 5 fold paired t-test.

The evaluation above shows both the advantages and limits of graph categorical embeddings (GCE). GCE performs better than Word2Vec in $D1$ and $D2$, while in $D3$ it performs like Word2Vec. The Word2Vec algorithm implicitly learns the context of the categorical features and their co-occurrence by means of the CBOW (or Skipgram) procedure. What GCE can manage to do more than Word2Vec is to consider neighbours at a deeper distance than one hop in the co-occurrence matrix. The reason why the performance of GCE converges to that of Word2Vec in $D3$ is due to the fact that such a data set has many more categorical features and values than $D1$ and $D2$ with a sparser co-occurrence matrix that reduces the effectiveness of GCE. The use of a directed graph could improve the behaviour of GCE in cases in which the co-occurrence matrix is insufficient to model the relationship between categorical attributes. It is necessary to mention that Node2Vec does not perform as well as Word2Vec or GCE in the three selected data sets, despite being based on the same co-occurrence matrix used with GCE. This may happen because Node2Vec embeddings try to approximate the neighbourhood at many multiple hops, that, with sparse co-occurrence matrices may introduce noise in the final calculated embeddings. Fine tuning the parameters of Node2Vec walks may allow to achieve better results, but the required search of parameters is not pratical if comparared with Word2Vec or GCE.

5 Conclusion and Future Work

This paper presented an approach to leverage the strong inductive bias of GNNs towards the definition of categorical embeddings, calculated through the soft assignment of the DIFFPool algorithm applied to the normalized co-occurence matrix of the categorical features, used as an adjacency matrix.

Future works imply the definition of different approaches towards calculating the adjacency matrix, as certain categorical features may be reduntant and not

ESANN 2023 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 4-6 October 2023, i6doc.com publ., ISBN 978-2-87587-088-9. Available from http://www.i6doc.com/en/.

define an interesting neighbourhood for DIFFPool. In addition, further information concerning the categories could be inputed to the autoencoder architecture to further bias the embeddings, such as for example information concerning the relationships between the categorical values and the target.

References

[1] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[2] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE, 2005.

[3] Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. Tem: Tree-enhanced embedding model for explainable recommendation. In *Proceedings of the 2018 World Wide Web Conference*, pages 1543–1552, 2018.

[4] Cheng Guo and Felix Berkhahn. Entity Embeddings of Categorical Variables. (1):1–9, 2016.

[5] Geoffrey E Hinton et al. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA, 1986.

[6] Kedar Potdar, Taher S., and Chinmay D. A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers. *Int. J. Comput. Appl.*, 175(4):7–9, 2017.

[7] John T. Hancock and Taghi M. Khoshgoftaar. Survey on categorical data for neural networks. *J. Big Data*, 7(1), 2020.

[8] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[10] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018.

[11] Hai Van Pham, Dat Hoang Thanh, and Philip Moore. Hierarchical pooling in graph neural networks to enhance classification performance in large datasets. *Sensors*, 21(18):6070, 2021.

[12] Samy Benslimane, Jérome Azé, Sandra Bringay, Maximilien Servajean, and Caroline Mollevi. Controversy detection: A text and graph neural network based approach. page 339â354, Berlin, Heidelberg, 2021. Springer-Verlag.

[13] Kaixiong Zhou, Qingquan Song, Xiao Huang, Daochen Zha, Na Zou, and Xia Hu. Multi-channel graph neural networks. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 1352–1358, 2021.

[14] Stephen D Bay, Dennis Kibler, Michael J Pazzani, and Padhraic Smyth. The uci kdd archive of large data sets for data mining research and experimentation. *ACM SIGKDD explorations newsletter*, 2(2):81–85, 2000.

[15] Aditya Grover and Jure Leskovec. Node2vec: Scalable feature learning for networks. KDD '16, page 855â864, New York, NY, USA, 2016. Association for Computing Machinery.