GRAPE: Handling Missing Data using Graph Representation Learning

Rabin Thapa Alina Lazar

Youngstown State University

Abstract

Handling missing data has been an important pre-processing step for many machine learning tasks. However, existing imputation models tend to have strong prior assumptions and cannot learn from downstream tasks. In this project, we evaluate a graph-based framework known as GRAPE for data imputation.

To evaluate the performance of GRAPE framework, we experiment it on several benchmark datasets and show its accuracy in terms of root mean square error (*rmse*) and compare it with existing state-of-the-art methods.

The GRAPE Framework

GRAPE formulates a missing value problem using graph representation, where the observations and features are viewed as two types of nodes in a bipartite graph, and the observed feature values as edges between them. The *feature imputation* is formulated as an *edge-level prediction* task and the *label prediction* as a *node-level prediction* task (see Figure 2).

GRAPE solves both tasks via Graph Neural Networks (GNNs). Its core architecture is inspired by the GraphSAGE model from which it inherits its inductive learning capabilities across different graphs.

Datasets and Experiments

The experiments are conducted on seven datasets (naval, power, energy, kin8nm, concrete, energy, and housing) obtained from UCI Machine Learning Repository. The missing values are introduced by removing random values in the data matrix. It is done by using a *mask*, and the proportion of masked values in a dataset is called *missing ratio*.

Figure 1 shows concrete dataset which has 1029 observations and 9 features. The white spaces denote missing values (with *missing ratio* of 0.10).

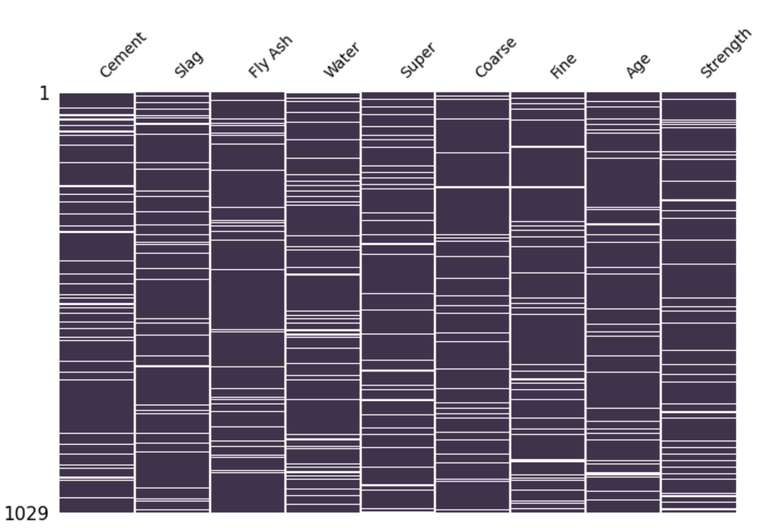

Figure 1:Concrete dataset showing missing values

Furthermore, all the experiments are conducted with following configurations:

- 20,000 training epochs using Adam optimizer with a learning rate of 0.001
- Activation function: RELU
- feature imputation tasks: 3-layer GNN with 64-hidden units
- label prediction tasks: 2-layer GNN with 16 hidden units

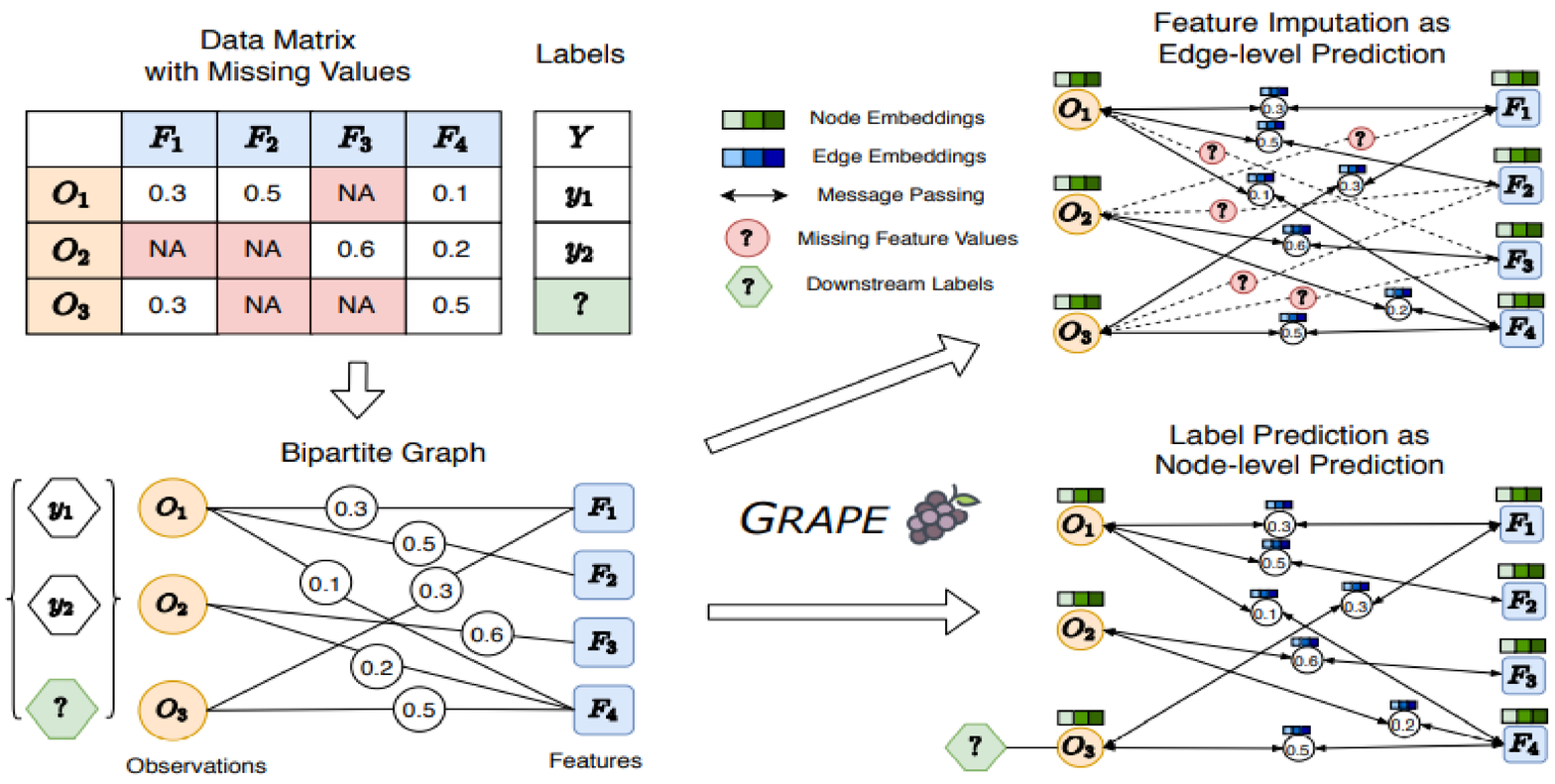

Figure 2:Overview of the proposed GRAPE framework

Model Evaluation

Feature imputation: Figure 3 shows how the values of different test metrics change as the number of epochs increases.

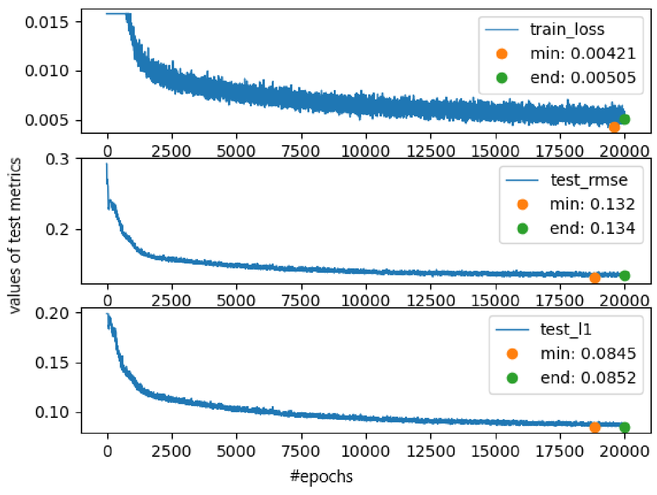

Figure 3:Values of test metrics across different epochs

Figure 4 compares *rmse* values of feature imputation for GRAPE against other commonly known imputation methods. The test metric is chosen to be the root mean square error (*rmse*) between predicted and actual values (wherever the actual values are masked for testing).

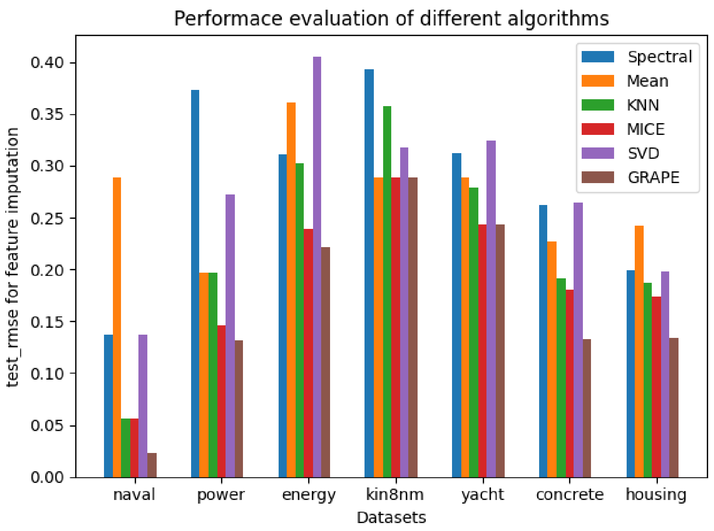

Figure 4:Comparision of different methods for imputation

Label prediction: The labels are randomly split into 70/30% training and test sets. Figure 5 shows actual data from concrete dataset and values predicted by GRAPE on the dataset. Further analysis on label prediction is yet to be done.

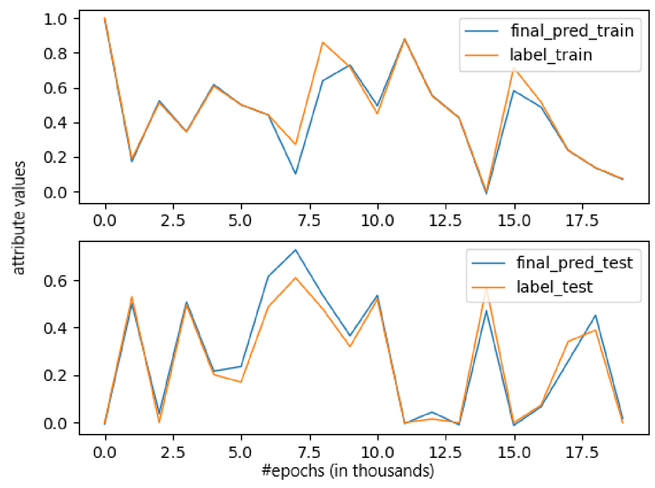

Figure 5:Label prediction for train and test sets

Results

As shown in Figure 4, GRAPE has the lowest *rmse* among all methods on all datasets. For feature imputation, GRAPE gives 12% lower average *rmse* than the best bestline model (MICE).

Conclusion

In this project, GRAPE, a graph-based approach to handle missing data is introduced. It represents feature imputation as an edge-level prediction task and the label prediction as a node-level prediction task in a bipartite graph. Using these representations, it trains GNN to solve the tasks.

From the experiments on seven UCI datasets, the performance (*feature imputation*) of GRAPE is seen to be significantly better (lower *rmse*) than existing state-of-the-art methods.

Future Works

- Analysis of label prediction by GRAPE
- Imputation of missing values in timeseries data

Acknowledgments

We are thankful to UCI Machine Learning Repository for the datasets and to Ohio Supercomputer Center (OSC) for GPU computing services.

References

Handling Missing Data with Graph Representation Learning. Jiaxuan You, Xiaobai Ma, et. al. Neural Information Processing Systems (NeurIPS), 2020.