2022 21st International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)

A Comparison Study of Graph Neural Network and Support Vector Machine

Siying Lin, José Alves, Francesca Bugiotti, Frédéric Magoulès
Université Paris-Saclay, CentraleSupélec, CNRS/LISN and MICS, 9 rue Joliot Curie, 91192 Gif-sur-Yvette, France and
Transvalor SA, E-Golf Park, 950 Avenue Roumanille, 06410 Biot, France
Email: frederic.magoules@hotmail.com

***Abstract*—A variety of issues, including classification, link prediction, and graph clustering, have been solved using graph neural network (GNN), an efficient method for handling non-Euclidean structural data. Another effective and reliable mathematical tool for classification and regression applications is support vector machine (SVM). We hope that this paper will help readers gain a better knowledge of the latest developments in graph neural networks and how they are used in a variety of fields. We also describe current research on using support vector machines for prediction and classification problems. Following that, a comparison between SVM and GNN is made, and the results are discussed.**

I. Introduction

Significant advances in speech recognition, image processing, and text translation have been made thanks to the development of machine learning and deep learning. These tasks represent data as simple sequences or regular grids in Euclidean space. However, not all data can be represented as sequences or grids, such as biological networks, social networks, complex file systems, and a majority of data are unstructured. Graph neural networks proposed in [12] use neural networks on graph-structured data. A graph model is used to depict a group of items and their relationships. Molecular structures can be modeled as graphs in which the atoms can be represented as nodes and the bonds can be represented as edges. Numerous graph issues, such as traffic forecasting, object detection, molecular fingerprints, and human-object interaction are tackled by GNN due to its ability to handle non-Euclidean data structures.

An overview of graph neural network model is available in several publications. In [14], the authors introduce a new method to divide GNN models into four categories and they provide the most detailed analysis of deep learning techniques for graphs that have been done so far. The authors in [22] give a general design step of GNNs and an overview of their computational modules. The application scenarios are further divided into structural and non-structural scenarios using systematic categorization. In addition to examining the most common GNN models, we also compare them with SVM models.

Support vector machine is a supervised algorithm based on statistical learning theory, which is extensively used in regression and classification analysis. It aims to determine a decision function, called a hyperplane, in which the data points are classified distinctly with the largest margin. The nearest points to the hyperplane, which are dispersed on both sides of the hyperplane, are referred to as support vectors. The strength of SVM in solving nonlinear problems makes it widely applicable to analyze complex systems. The work [5] summarizes the application of SVM in building energy consumption, including the complete theoretical background and parallel computing methods to speed up the training process.

There are numerous studies combining artificial neural network (ANN) and SVM for different tasks, such as face image recognition, spam detection, and energy production forecast. While other studies have compared ANN and SVM, no work has been done to compare GNN and SVM. In this paper (i) we present a brief overview of several GNNs and how they can be applied in various fields, (ii) we summarize a list of classification and regression tasks to which SVMs have been applied, and (iii) we compare the similarities and differences between GNN and SVM.

This paper is structured as follows: Section II introduces several GNN models. Section III presents the general formulation of SVM and previous research works over applications. Section IV discusses the similarities and differences between GNN and SVM. Section V concludes this article.

II. Graph neural networks

Graph models are used to illustrate the relationships between items in a group, as well as the representations of items themselves. A graph is defined as $G = (V, E)$, where V corresponds to the set of nodes, and E corresponds to the set of edges. Let $v_i \in V$ stands for a node and $e_{ij} = (v_i, v_j) \in E$ stands for an edge. The neighborhood of a node v can be defined as $\mathcal{N}(v) = \{u \in V | (v, u) \in E\}$. The adjacency matrix of a graph can be represented by $\boldsymbol{A}$. The feature vector of the node v is denoted by $\boldsymbol{x}_v \in \mathbb{R}^d$, where d is the dimension of a node feature vector. The node feature matrix of a graph is denoted by $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, where n is the number of nodes. The feature vector of the edge (v, u) is denoted by $\boldsymbol{x}^e_{v,u} \in \mathbb{R}^c$, where c is the dimension of an edge feature vector. The edge feature matrix of a graph is expressed as $\boldsymbol{X}^e \in \mathbb{R}^{m \times c}$, where m is the number of edges.

A. Recurrent Graph Neural Network

GNN proposed in [12] details the structure, calculation method, optimization algorithm, and process implementation.

2473-3636/22/$31.00 ©2022 IEEE
DOI 10.1109/DCABES57229.2022.00009

With GNN, what is learned is the state embedding of each node's neighbors, which can be utilized to produce the output like a node's label. The state embedding $\boldsymbol{h}_v$ and the output $\boldsymbol{o}_v$ take the form

$$\boldsymbol{h}_v = f(\boldsymbol{x}_v, \boldsymbol{x}^e_{(v,u)}, \boldsymbol{x}_{\mathcal{N}(v)}, \boldsymbol{h}_{\mathcal{N}(v)}),$$
$$\boldsymbol{o}_v = g(\boldsymbol{h}_v, \boldsymbol{x}_v),$$

where $\boldsymbol{x}_{\mathcal{N}(v)}$ and $\boldsymbol{h}_{\mathcal{N}(v)}$ represent the states and the features of v's neighbors, respectively. $f(\cdot)$ is a parametric function that expresses a node's dependence on its neighborhood. The local output function, g, provides information on how the output is generated. When it comes to training a model, gradient descent is commonly used as the algorithm to do so.

Gated graph neural network (GGNN) [10] is characterized by the use of Gated Recurrent Units (GRU) and unrolled recursion for a fixed number of steps T. Hence, f no longer needs to be a contraction map, and iteration does not need to converge to the output. The model can output a value for each node to solve node classification, or it can output a value for the entire graph to solve graph classification.

B. Convolutional Graph Neural Network

Convolutional graph neural networks are classified into two primary subcategories: one that uses spectral information and one that uses spatial information.

Spectral convolutional GNN: Inspired by ChebNet, the authors in [7] propose graph convolutional neural network (GCN), which is equivalent to an approximation of first-order ChebNet. According to layer-wise operation, the propagation rule for different layers of GCN takes the form:

$$\boldsymbol{H}^{(l+1)} = \sigma(\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}),$$

where $\widetilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}_N$ refers to the adjacency matrix plus self-connections. $\boldsymbol{I}_N$ represents the identity matrix. $\widetilde{\boldsymbol{D}}_{ii} = \sum_j \widetilde{\boldsymbol{A}}_{ij}$. $\boldsymbol{W}^{(l)}$ refers to a trainable weight matrix for a particular layer and $\sigma(\cdot)$ is the activation function.

Adaptive Graph Convolution Network (AGCN) [9] takes data of arbitrary graph structure and size as input. It constructs a unique residual Laplacian matrix for each sample in batch and learns distance metric for graph update.

Diffusion-Convolutional Neural Network (DCNN) [1] extends convolution operation to generate graph-structured data through a diffusion process. It defines the probability of jumping from a node to one of its neighbors. Node classification and graph classification are both applicable to this model.

Spatial convolutional GNN: GraphSAGE [6] aggregates feature information of neighboring nodes $\mathcal{N}(v)$ and directly generates embeddings for unseen nodes. Each node v aggregates the representations of neighbor nodes $\boldsymbol{h}^{k-1}_{\mathcal{N}(v)}$ at $k-1$ step into a vector $\boldsymbol{h}^k_{\mathcal{N}(v)}$,

$$\boldsymbol{h}^k_{\mathcal{N}(v)} \leftarrow AGGREGATE_k(\{\boldsymbol{h}^{k-1}_u, \forall u \in \mathcal{N}(v)\}).$$

The node's representation $\boldsymbol{h}^{k-1}_v$ is then concatenated with the aggregated neighborhood vector $\boldsymbol{h}^k_{\mathcal{N}(v)}$ to get the representations $\boldsymbol{h}^k_v$,

$$\boldsymbol{h}^k_v \leftarrow \sigma\left(\boldsymbol{W}^k \cdot CONCAT(\boldsymbol{h}^{k-1}_v, \boldsymbol{h}^k_{\mathcal{N}(v)})\right).$$

The output is the aggregation of the neighbor representations $\boldsymbol{z}_v$ at depth K. The embedding $\boldsymbol{z}_v$ of node v is obtained through forward propagation. Then standard stochastic gradient descent and back propagation algorithms are performed to optimize parameters.

C. Graph Attention Network

Graph Attention Network (GAT) [13] proposes an attention mechanism to learn the weights of the node's neighbors. Self-attention is performed over all nodes

$$e_{vu} = a(\boldsymbol{W}\boldsymbol{h}_v, \boldsymbol{W}\boldsymbol{h}_u),$$

where $\boldsymbol{W} \in \mathbb{R}^{F' \times F}$ is a weight matrix that applies to every node, with F and F' indicating the number of input and output features in each node. Generally, self-attention assigns importances to all nodes in the graph, which loses structural information. Masked attention can be used to solve this problem, considering only $u \in \mathcal{N}_v$.

D. Graphs

Heterogeneous Graph: There are few studies applying GNN to heterogeneous graph, which has many different kinds of nodes and links. Neighbor Averaging over Relation Subgraphs (NARS) [15] are extended to heterogeneous graphs for node classification. NARS samples a certain number of unique subsets from a certain edge relationship type of a heterogeneous graph, and for each sampled subset, retains the edge relationship belonging to the subset, and establishes a subgraph.

Large Graph: Several studies have taken advantage of subgraph sampling as a method of training when dealing with large graphs. For instance, we can handle large graphs by sampling a subgraph and performing graph convolutions on nodes in the sampled subgraph. To train the model, there are two methods that can be used. One is to treat each subgraph as a minibatch, and the second is to combine several subgraphs into a minibatch.

III. Support Vector Machines

Support vector machine can handle linear/nonlinear classification and regression. When the data are linearly separable, SVM can divide the data by creating a hyperplane that maximizes the margin between the classes. In the case that the data cannot be linearly separated, then the data can be mapped to a higher dimensional feature space by applying a proper kernel function. Once the data have been mapped to a higher dimensional feature space, the algorithm determines the maximal margin hyperplane within the feature space.

A. Theoretical Framework

Suppose there are n samples in total. Let vector $x_i \in \mathbb{R}^n$ denotes the i-th features sample, and $y_i \in \mathbb{R}$ indicates the value of the corresponding x_i. All of the samples are, therefore, expressed as $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. The hyperplane takes the form

$$f(x) = \omega x + b,$$

where $f(x_i)$ is an estimate of y_i. In this equation, ω and b are coefficients. In the case where y is a continuous variable, $f(x)$ can be used to predict the value of y_i corresponding to the new feature x_i. If $y_i \in \{-1, +1\}$, this method can be used for classification problems.

Model training refers to the process of finding function $f(x)$, determining the values of ω and b. The goal is to make the margin as large as possible, then SVM can be formulated as an optimization problem. Besides, nonlinear decision functions can be implemented by utilizing kernel functions

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(x_i \cdot x_j) - \sum_{i=1}^{n}\alpha_i \\ \text{s.t.} \quad & \sum_{i=1}^{n}\alpha_i y_i = 0 \\ & 0 \geq \alpha_i \leq C, i = 1, ..., n, \end{aligned} \tag{1}$$

where α is Lagrange multiplier and K is kernel function. By breaking the optimization problem down into sub-problems and tackling each one separately, the sequential minimal optimization (SMO) method can be utilized to solve the quadratic function mentioned above.

B. Applications

SVM has been extensively applied to regression tasks, such as the prediction of energy consumption [11, 21]. The authors in [3] discuss the viability and effectiveness of SVM in the field of forecasting building load and apply it to predict building energy consumption in the torrid zone. In [4], the authors show the theoretical basis of SVM and apply it to predict electricity consumption. Considering one year's measured data, this was done by training with the first six months of the dataset, validating with the following three months, and testing with the final three months. Based on the results, it appears that SVM has good performance. In [8], the authors not only predict the electricity consumption, but also re-predict the daily electricity consumption within a year with one day lag to determine the influencing factors. To improve the accuracy of the predicted results, the authors in [16, 17] present a simulation method that can be used to collect sufficient historical data on the energy consumption of various buildings. Additionally, they adopt a parallel method of SVM on such large data to accelerate the training process. Training SVM is a quadratic optimization problem that requires a lot of time and memory, making it difficult to apply SVR to large-scale problems. A new parallel implementation based on decomposition method called Map-Reduce parallel SVM (MRPsvm) is proposed to speed up the training process on large scale datasets [19]. Since there are lots of factors that influence energy consumption, choosing the proper features is also important. The authors in [18, 20] focus on reducing model complexity without compromising performance by selecting features. The proposed method has been shown to be effective in selecting valid feature subsets in numerous experiments. Minimal redundancy maximal relevance is adopted as an algorithm for the selection of features in short-period load forecasting [2].

In addition to its application in regression, SVM has numerous other applications in classification as well, such as face detection, text categorization, and cancer classification. For instance, the face recognition problem is classifying the feature vector at each pixel into a face or a non-face.

IV. Comparison

A. Multi-class Classification

GNN and SVM are the most popular algorithms for classification in machine learning algorithms. GNN supports multi-class classification, the number of neurons in the output layer of GNN depends on how many classes are to be classified. SVM is a binary classifier and doesn't support multi-class classification natively. However, multi-class classification problems can be decomposed into multiple binary classification problems. Nevertheless, it is possible to decompose multi-class classification problems into several binary classification problems to solve them. This approach consists of mapping the data into a higher dimensional space to obtain a mutual linear separation between each of the two classes that are being analyzed.

B. Classification Accuracy

Training sample size can influence the performance. The model's accuracy can be increased by providing more training data. The use of a normalization method before feeding the training data to different models can also improve accuracy. When solving classification problems, kernel functions are crucial since they convert data into the desired form, and choosing different SVM kernel functions will result in different prediction accuracy.

C. Optimization of the Parameters

GNN optimizes the parameters via stochastic gradient descent optimization algorithm. The most widely used optimization approach in machine learning, particularly deep learning, is stochastic gradient descent. SVM can be modified as a quadratic programming problem. The function is optimized subject to linear constraints on its variables. SVM is solved by sequential minimal optimization, a quadratic programming algorithm that identifies possible solutions by iteratively computing analytical solutions to a series of smallest possible subproblems.

V. Conclusion

A lot of interest has been drawn to graph neural networks as a potential solution to graph-related problems. With the help of SVM, we can perform regression and classification functions on linear and non-linear data. In this paper, we draw a comparison of these two machine learning algorithms and give a brief review of the most recent GNN models. Then we introduce the general formulation of nonlinear SVM and broad applications. Finally, we compare GNN and SVM for classification tasks.

References

[1] J. Atwood and D. Towsley. Diffusion-convolutional neural networks. *Advances in neural information processing systems*, 29, 2016.

[2] Y. Dai and P. Zhao. A hybrid load forecasting model based on support vector machine with intelligent methods for feature selection and parameter optimization. *Applied Energy*, 279:115332, 2020.

[3] B. Dong, C. Cao, and S. E. Lee. Applying support vector machines to predict building energy consumption in tropical region. *Energy and Buildings*, 37(5):545–553, 2005.

[4] F. Magoulès, M. Piliougine, and D. Elizondo. Support vector regression for electricity consumption prediction in a building in japan. In *Proceedings of the 2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES), Paris, France, August 24-26, 2016*, pages 189–196. IEEE Conference Publications, 2016.

[5] F. Magoulès and H.-X. Zhao. *Data Mining and Machine Learning in Building Energy Analysis*. Computer Engineering Series. Wiley-ISTE, London, UK, 2016. Hardcover 186 pages.

[6] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[7] T. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *ArXiv*, abs/1609.02907, 2017.

[8] F. Lai, F. Magoulès, and F. Lherminier. Vapnik's learning theory applied to energy consumption forecasts in residential buildings. *International Journal of Computer Mathematics*, 85(10):1563–1588, 2008.

[9] R. Li, S. Wang, F. Zhu, and J. Huang. Adaptive graph convolutional neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32(1), 2018.

[10] Y. Li, D. Tarlow, M. Brockschmidt, and R. S. Zemel. Gated graph sequence neural networks. *CoRR*, abs/1511.05493, 2016.

[11] Z. Ma, C. Ye, H. Li, and W. Ma. Applying support vector machines to predict building energy consumption in china. *Energy Procedia*, 152:780–786, 2018.

[12] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[13] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio', and Y. Bengio. Graph attention networks. *ArXiv*, abs/1710.10903, 2018.

[14] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[15] L. Yu, J. Shen, J. Li, and A. Lerer. Scalable graph neural networks for heterogeneous graphs. *arXiv preprint arXiv:2011.09679*, 2020.

[16] H.-X. Zhao and F. Magoulès. A parallel statistical learning approach to the prediction of building energy consumption based on large datasets. In Q. Guo and Y. Guo, editors, *Proceedings of the 6th International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES), Wuhan, Hubei, China, October 16–19, 2009*. Publishing House of Electronics Industry, 2009.

[17] H.-X. Zhao and F. Magoulès. Parallel support vector machines applied to the prediction of multiple buildings energy consumption. *Journal of Algorithms and Computational Technology*, 4(2):231–250, 2010.

[18] H.-X. Zhao and F. Magoulès. Feature selection for support vector regression in the application of building energy prediction. In *Proceedings of the 9th IEEE International Symposium on Applied Machine Intelligence and Informatics (SAMI 2011), Smolenice, Slovakia, January 27–29, 2011*. IEEE Computer Society, 2011.

[19] H.-X. Zhao and F. Magoulès. New parallel support vector regression for predicting building energy consumption. In *Proceedings of the IEEE Symposium Series on Computational Intelligence in Multicriteria Decision Making (MDCM 2011), Paris, France, April 11–15, 2011*. IEEE Computer Society, 2011.

[20] H.-X. Zhao and F. Magoulès. Feature selection for predicting building energy consumption based on statistical learning method. *Journal of Algorithms and Computational Technology*, 6(1):59–78, 2012.

[21] H.-X. Zhao and F. Magoulès. A review on the prediction of building energy consumption. *Renewable and Sustainable Energy Reviews*, 16(6):3586–3592, 2012.

[22] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.