MVMNET: GRAPH CLASSIFICATION POOLING METHOD WITH MAXIMUM VARIANCE MAPPING

Lingang Wang and Lei Sun

School of Systems Science and Engineering, Sun Yat-sen University, Guangzhou, China

ABSTRACT

Graph Neural Networks (GNNs) have been shown to effectively model graph-structured data for tasks such as graph node classification, link prediction, and graph classification. The graph pooling method is an indispensable structure in the graph neural network model. The traditional graph neural network pooling methods all employ downsampling or node aggregating to reduce graph nodes. However, these methods do not fully consider spatial distribution of nodes of different classes of graphs, and making it difficult to distinguish the class of graphs with spatial locations close to each other. To solve such problems, this article proposes a Maximum Variance graph feature Multistructure graph classification method (MVM), which extracts graph information from the perspective of graph nodes feature and graph topology. From the nodes feature perspective, we enlarge the variance between different classes while maintaining the variance between the same class of data. Then the hierarchical graph convolution and pooling are performed from a topological perspective and combined with a CNN readout mechanism to preserve more graph information to obtain a graph-level representation with strong discrimination. Experiments demonstrate that our method outperforms several number of state-of-the-art graph classification methods on multiple publicly available datasets.

KEYWORDS

Double-view Graph Pooling, Constrained Maximum Variance, Hierarchical Graph Structure.

1. INTRODUCTION

Convolutional neural network (CNN),[1] has achieved satisfactory performance in computer vision[2],[3], natural language processing[4]. The data involved in these tasks are grid-structured data (e.g., texts, images, and videos), which can be described by regular Euclidean structured data. Convolution and pooling methods can be used to effectively processing such data and achieve considerable results.

However, many real-world data mostly have graph structures, which are non-Euclidean structures, such as chemical molecules, social networks, knowledge graphs, etc. These structures cannot be processed directly by the above methods. In order to solve these problems, some relevant studies [5] try to extend convolution and pooling methods to graph structured data.

Graph neural network (GNN)[6] method has received considerable attention recently. It can be competent for most graph related tasks, such as node classification [7], links prediction [8] and graph classification[9],[10]. For CNN, pooling operator is an essential part, which can decrease

David C. Wyld et al. (Eds): ICAITA,CDKP, CBW, SOFT, CMC, Signal, MLT-2023
pp. 161-176, 2023. CS & IT - CSCP 2023 DOI:10.5121/csit.2023.130613

the dimension of feature maps and number of parameters and prevent overfitting. As for graph pooling, the general idea is to employ averaging or aggregating operators to cluster the nodes of a given graph[7]. However, the simple aggregation scheme ignores the interactive information between nodes and regards all nodes as equal status, which hinders the expressive ability of the model, and also fails to produce meaningful representations of graph structure features. To solve the above problems, a large number of graph pooling schemes have been proposed, which can be divided into two parts: global pooling and hierarchical pooling. Global pooling[11] directly generates a graph-level representation, which mainly takes the average or sum of all node embedding as the graph representation. Hierarchical pooling can be further divided into node clustering pooling[12] and node drop pooling[13]. The node clustering scheme calculates the similarity or weight between nodes, and aggregates similar nodes into a new node, which is time-consuming and space-consuming[14]. The node drop is to calculate the score of each node, and then discard the low-scoring nodes proportionally, which is more efficient and suitable for large-scale graphs[15], but inevitably loses some graph information[16]. DiffPool [12], MincutPool [14], StructPool[17], and ASAP[18] generate pooled graphs through clustering nodes. gPool [13], SAGPool [15] and HGP-SL[19] collapse the graph through hierarchical drop nodes. However, these methods only simplify the topological structure of the graph, ignoring the influence of the node features of the graph on the results.

In this paper, we propose a Maximum Variance graph feature Multistructure graph classification method (MVM) to overcome above difficulties. We extract the graph information from two perspectives of graph node features and graph topology. From the graph feature perspective, general constrained maximum variance mapping[20] is introduced to enlarge the variance between different classes of graphs while maintaining the variance between the same class of data. For the matrix decomposition problem involved in the above process, we employ the subspace reconstruction[21] from local to global to avoid the large-scale matrix decomposition problem. Different from the traditional dimension reduction methods, our main purpose is not to reduce the dimension of features, but to make the node features in the projection space contribute to the downstream classification tasks to the greatest extent. For the graph topological pooling, this paper leverages the multistructure graph classification method[22] to extract and fuse the graph topology information of different layers. The main contributions of this paper are as follows:

1. We propose a scheme that extracting graph information from the perspective of graph node features and graph topology to make up for the disadvantage of only pooling from the perspective of graph topology.
2. We expand the variance between different graph classes. Subspace reconstruction is employed to solve the problem of large-scale matrix decomposition. Hierarchical multistructure graph feature extraction is used to pick up reasonable graph information.
3. A range of experiments were carried out on 7 public datasets, and the comparison with several latest methods proved the effectiveness of our method.

2. RELATED WORK

2.1. Manifolds Learning

For feature extraction, Principal Component Analysis (PCA)[23] and Linear Discriminant Analysis (LDA)[24] are both classical linear feature extraction methods. These methods assume that the data are distributed in the global Euclidean space. For data distributed on the nonlinear manifolds, the linear feature extraction method unable to get a valid embedding result, so it is necessary to introduce nonlinear manifold learning methods to process such data.

Laplace Eigenmaps (LE)[25] maintains the distance between adjacent data points before and after dimension reduction. Hessian-based Local Linear Embedding(HLLE)[26] replaces Laplacian-Beltram operator in LE with Hessian operator. Local Tangent Space Alignment (LTSA)[21] arranges the tangent space of the neighborhood where the high-dimensional sample points are located to represent the local structure of the manifold to obtain a global low dimensional representation of the data.

2.2. Graph Pooling

According to the role of graph level representation in learning, graph pooling can be roughly divided into global pooling and hierarchical pooling. The former generates a graph level representation in a single step, while the latter gradually collapses the graph into a smaller graph.

2.2.1. Global Pooling

The global pooling methods typically aggregate node representations after graph convolutional layers and output graph-level representations. Common aggregation operations mainly include sum-pool, mean-pool, max-pool, and neural networks. For example, Set2Set[27] utilizes content-based attention to obtain important nodes and aggregates their information via Long-Short Term Memory(LSTM)[28]. SOPool[29] employs bilinear mapping and attention second-order pooling and second-order graph statistical information to pool graphs. However, the global pooling method ignores the correlation information between nodes and the topology information of the graph.

2.2.2. Hierarchical Pooling.

Hierarchical pooling can capture topological information of graphs by learning hierarchical representations. The hierarchical pooling can be roughly divided into node clustering pooling and node drop pooling. Node clustering pooling aggregates clusters of multiple nodes into a single node. DiffPool[12] assigns nodes to different clusters by learning a cluster assignment matrix using a GNN model. StructPool[17] utilizes conditional random fields to learn the cluster assignment matrix during the pooling process. ASAP[18] adopts a self-attention to capture nodes in clusters and computes the cluster scores through local extremum convolution(LEConv). Node drop pooling calculates the node scores and discards the low scoring nodes. gPool[13] projects all node features into the trainable projection vector, and takes the projection value of the node on the projection vector as the score of the node. SAGPool[15] leverages GCN to calculate the node's information score. HGP-SL[19] selects some representative nodes according to the predefined node information score function.

Apart from node drop and node clustering pooling methods, there also exist some other graph pooling methods. For example, EdgePool[30] and HyperDrop[31] pool graphs from the perspective of edges. Muchpool[32] combines the node clustering pooling and the node drop pooling to capture the different characteristics of the graph.

3. PRELIMINARIES

Let $G=(V,E,X,A,L)$ be a graph, which $V=\{v_1,v_2,...,v_N\}$, $E=\{e_{ij}\}_{N\times N}$ and $X=[x_1,...,x_n]\in R^{N\times d}$, $A=(a_{ij})\in R^{N\times N}$, L are sets of nodes, sets of edges, nodes features, adjacency matrix of graph G and the label of G respectively, which $N=|V|$ is numbers of nodes and d is the feature

dimension, $a_{ij} = w_{ij} \neq 0$ indicates that node v_i is connected to v_j . Let $Y = [y_1, y_2, ..., y_n]$ is corresponding low dimensional embedding of X . The main goal of manifold learning is to learn a projection V which mapping the samples from original space to feature space: $y_i = Vx_i, i = 1,2,...,n$, so that the low-dimensional data Y can better reflect the essence of the original data. And the purpose of graph classification is to learn a mapping $f : G \to L$ that maps graphs G to the label L .

3.1. Constrained Maximum Variance Mapping

CMVM[20] is a manifold learning method that utilizing local structure and dissimilarities between manifolds to construct the relationship between homogeneous and heterogeneous data respectively. The purpose of the algorithm is to enlarge the dissimilarities between different submanifolds while keeping the local structure unchanged.

3.1.1. Local Structure

The local scatter can be characterized by the Euclidean distance between any pair of the projected sample points that are within any local k nearest neighbours. The neighbourhood $N(x_i)$ of the sample point x_i forms a manifold, if $x_j \in N(x_i)$, then the locality can be expressed as:

$$d_{ij} = \| x_i - x_j \|^2$$

On this basis, CMVM consider the local relations of points in embedding and original space, the purpose is to keep the locality of the low-dimensional embedding of the sample points inside the manifold the same as in the original space. Then y_i must satisfy the constraint:

$$J_L = \sum_{ij} \left\| y_i - y_j \right\|^2 L_{ij} = \sum_{ij} \left\| x_i - x_j \right\|^2 L_{ij} = 2tr\{Y(D-L)Y^T\}$$

where $L = (L_{ij})_{n \times n}$ is the local relation adjacency and definitions as follows:

$$L_{ij} = \begin{cases} 1 & \left\| x_i - x_j \right\|^2 \leq \varepsilon \\ 0 & otherwise \end{cases}$$

3.1.2. Dissimilarities between Manifolds

The Euclidean distance is also often employed as a measure of the dissimilarity and use the sum of the squared distance to measure the variance between different manifolds. It noted that the dissimilarities exist between different manifolds. Therefore, the CMVM pulls the different manifolds apart by maximizing the total sum of distances between outputs with the different labels. The dissimilarities between manifolds can be defined as following:

$$J_D = \sum H_{ij} \left\| y_i - y_j \right\|^2 = 2tr\{Y(Q-H)Y^T\}$$

where $Q = diag\{Q_{ii}, \cdots, Q_{nn}\}$, $Q_{ii} = \sum_j H_{ij}$ and H is the label matrix constructed by H_{ij}

$$H_{ij} = \begin{cases} 0 & \text{if } x_i \text{ and } x_j \text{ have the same class} \\ 1 & \text{otherwise} \end{cases}$$

3.1.3. Projection Matrix

CMVM introduces linear transformation V to solve the problem of out-of-sample learning ability and reduce the computational complexity of generalization learning. The features after feature extraction can be obtained by linear transformation $y_i = V^T x_i$. According to the above analysis, the objective function is formulated as

$$J(V) = \max(J_D) = tr\{V^T X(Q-H)X^T V\}$$
$$s.t.\ \ tr\{V^T X(D-L)X^T V\} = tr\{X(D-L)X^T\} \quad \{1\}$$

The linear transformation V is the d eigenvectors corresponding to the first d largest eigenvalues of the generalized eigen decomposition of $\{X(Q-H)X^T, X(D-L)X^T\}$.

3.2. Graph Neural Network

Graph Neural Networks (GNN) are deep learning based models that have recently become a widely used method for graph analysis. The structure learning of the graph is reflected in the aggregation strategy by considering the adjacency information. For graph G, GNNs generally follow a message-passing architecture:

$$H^{(k)} = M(A, H^{(k-1)}; W^{(k)})$$

where $H^{(k)}$ is the node features of the k-th layer and M is the message propagation function. The trainable parameters are denoted by $W^{(k)}$ and the adjacency matrix by A, $H^{(0)}$ is initialized as $H^{(0)} = X$. The propagation function M can be implemented in various manners [33][34][35].

4. METHODOLOGY

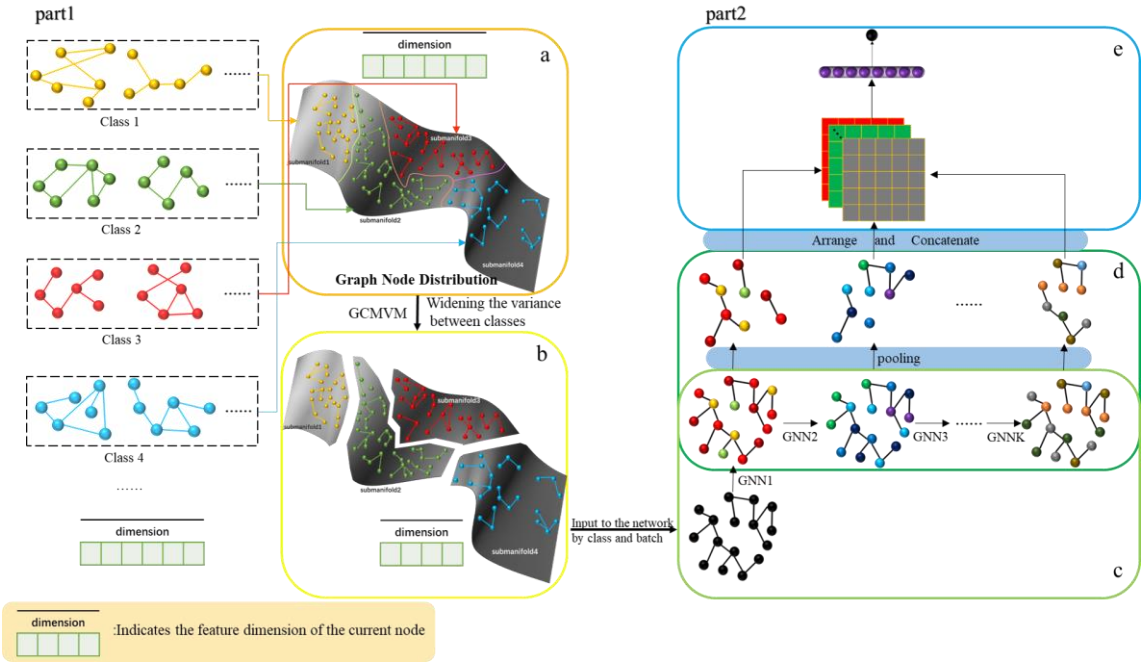

Figure 1. The structure of model. Part a represents the distribution of data over the original manifold. Part b indicates the distribution of the separated data on each sub-manifold. Part c represents the hierarchical graph feature extraction block. Part d indicates the hierarchical graph pooling module. Part e represents the readout layer and the classification layer.

4.1. Model Structure

The overall model structure is shown in
Figure 1, which consists of two parts.
1) Graph node feature embedding (GNFE): this part mainly enlarges the distance between different classes and the variance between different sub-manifolds by means of manifold learning.
2) Graph topology pooling (GTP): which consists of four modules, a hierarchical graph convolution module, a graph pooling module, a structure readout module and an output module. The main purpose of this part is to extract the feature information of the graph hierarchically by means of GNNs.

4.2. Graph Node Feature Embedding(GNFE)

Above graph pooling models [12][27][29][32]reduce the number of graph nodes through node clustering pooling and node drop pooling. Although such methods can obtain relatively reliable graph representations, their classification performance is limited by the distribution of graph nodes. As shown in
Figure 2 a, different classes of node with low discrimination will affect the performance of graph classification methods. Therefore, we need to enhance the distinguishability of different classes of graph nodes. We design a variance maximization model based on the CMVM module, as depicted in
Figure 2, which can enlarge the distance between different classes of graph data to benefit downstream classification tasks.

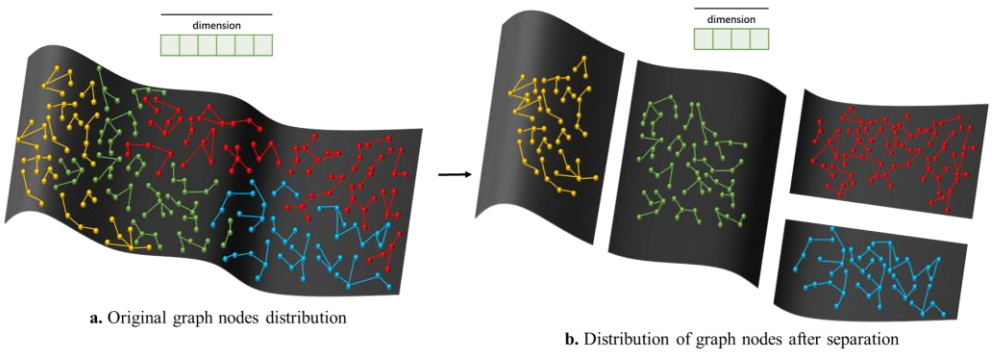

Figure 2. Distribution of graph nodes before and after separation

In this paper, for the following reason, it is not feasible to directly utilize the CMVM algorithm to graph data. First, the large number of graph samples leads to higher computational costs; Second, the graph relational structure of the samples is different from the data structure processed by the original algorithm. Therefore, we need to further generalize CMVM to handle graph data.

Let $X=[x_{g_{11}},\cdots,x_{g_{1N_1}},\cdots,x_{g_{M1}},\cdots,x_{g_{MN_M}}]$ be the graph data, M represents the number of all graphs, and $N_i(i=1,2,\cdots,M)$ represents the number of nodes of the i th graph. We batch the data as $X=[X_{b_1},\cdots,X_{b_Z}]$,where $X_{b_i}=[x_1,...,x_{N_i'}]$ consist of a set of graphs and Z is the number of

batches, N_i' is the number of the graph of the batch b_i. For the convenience of description, we take the first batch of data as an example and denote the first batch of data as $X = X_{b_1} = [x_1, ..., x_{N_1'}] = [x_1, x_2, \cdots, x_n]$ and the data after dimension reduction is $Y = [y_1, y_2, ..., y_n]$. It is a disconnected graph $G_B = [X, A]$ composed of multiple graphs in a batch, A is the adjacent matrix of G_B.

Since the sample data itself has a graph structure, the construction of the graph structure can be omitted. We can directly leverage the label information of the disconnected graph G_B to construct the local relation matrix and dissimilarities matrix:

$$L_{ij} = \begin{cases} 1 & x_i \text{ and } x_j \text{ are graph nodes with the same class} \\ 0 & \text{otherwise} \end{cases}$$

$$H_{ij} = \begin{cases} 0 & \text{if } x_i \text{ and } x_j \text{ are graph nodes with same class} \\ 1 & \text{otherwise} \end{cases}$$

By solving the optimization problem {1}, we can obtain the projection matrix V mapping the original space to the feature space.

The projection matrix $V_i (i = 1, 2, ..., Z)$ is calculated for each batch. Assuming that the local embedding feature in the *ith* projection space is $\Theta_i = [\theta_1^{(i)}, \theta_2^{(i)}, ..., \theta_n^{(i)}]$, according to LTSA, the global embedding y_j can be obtained by the linear combination between the local embedding:

$$y_j = \sum_i \alpha_i \theta_j^{(i)}$$

where α_i is the weight coefficient.
Therefore, inspired by LTSA, we can similarly define the relationship between global projection and local mapping as:

$$V = \beta_1 V_1 + \beta_2 V_2 + \cdots + \beta_Z V_Z$$

The weight parameter β_i can be obtained by training. Through experiments, it is found that there is almost no performance difference between directly averaging all projection matrices and using the attention mechanism, so for simplicity we can directly average all projection matrices to get the global projection matrix:

$$V = \frac{1}{Z}(V_1 + V_2 + \cdots + V_Z)$$

4.3. Graph Topology Pooling (GTP)

GNFE pools the graph from the perspective of node feature dimension without changing the structure of the graph and makes different classes of graph data have obvious distribution variance, which is more conducive to downstream classification tasks. For the task of graph node classification, we are required to design a reasonable graph feature aggregation scheme. The traditional GNN method [12][27][29][32]only leverage a single strategy to extract graph features, which is more prone to lose graph information. In order to pool the topological structure of the graph from multi-routine, preserving the most representative nodes and extract different

substructures of the graph, we employ a hierarchical structure and cross-information convolution operator[22] for graphs composed of a batch of small graphs to extract graph information.
Based on the GNFE module, we can obtain the global projection matrix V , then the feature matrix after projecting is

$$X' = V^T X \in R^{N \times d}$$

which N is the number of nodes of the batch and d is the dimension of the feature after dimension reduction. We still record the feature after dimension reduction as X .

4.3.1. Multistructure Subgraphs

The traditional graph network[27][29] simply stacks GNN modules and only use the graph features of the last layer as the input of downstream tasks, which will lose more graph information. Inspired by ResNet[36], we pool the graph information of each layer to retain as much information as possible. Therefore, the module can be divided into two parts: hierarchical subgraph feature extraction and hierarchical graph pooling. Then we will introduce these two modules respectively.

4.3.1.1. Hierarchical Subgraph Feature Extraction(HSFE)

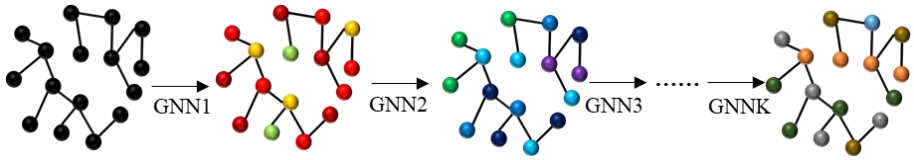

Figure 3. Hierarchical feature extraction

In order to preserve the structural information of different layers, as is shown in
Figure 3, we use K GNN models to update the nodes of the original graph. Let $G_1, G_2, \ldots, G_K$ represent the graphs extracted by the K graph networks respectively

$$G_i = GNN_i(G_{i-1}), i = 1, \ldots, K$$

where $G_0 = G$ and GNN models G_i have the following options: GCN, GAT, GraphSAGE. If the graph data have a heterogeneous graph, the heterogeneous graph GNN models can also be used to process the graph data.

4.3.1.2. Hierarchical Graph Pooling(HGP)

Since our model uses all layers of information, it inevitably brings information redundancy. Therefore, we need to process the graph data of each layer of the model by pooling to reduce the impact of redundancy.

Generally, graph pooling can be accomplished through two strategies, one is the discarding of nodes and the other is the aggregation of nodes. The latter will aggravate information redundancy, it is not suitable for our model, so we leverage nodes dropping to pool the graph of each layer. The goal of node discard pooling is to define a standard for sampling nodes and reconstructing graphs. Therefore, it is necessary to evaluate the importance of each node when performing node sampling. We employ GCN and MLP to calculate the score of nodes in each

layer of graph to pool the graph from both local and global perspectives. And the attention mechanism is employed to update the representation of graph nodes.

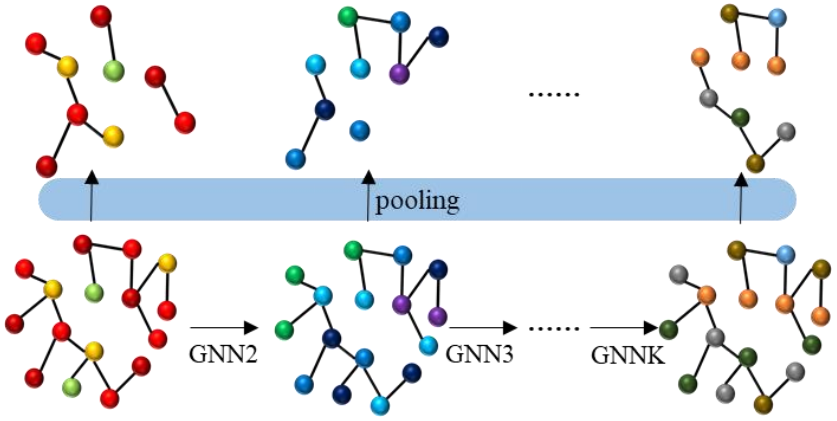

Figure 4. Hierarchical graph pooling

As shown in
Figure 4, let $G_l :< H_l, A_l >$ be the graph obtained by the l th graph convolution layer in HSFE, where H_l is the node feature matrix of the graph and A_l is the adjacency matrix of G_l. The calculation formula of node importance is given as follows:

$$S = \max(\sigma(\tilde{D}_l^{-\frac{1}{2}}\tilde{A}_l\tilde{D}_l^{-\frac{1}{2}}H_l W_g), \sigma(H_l W_m))$$

where σ is a nonlinear activation function, $\tilde{A}_l = A_l + I$ is the normalized adjacency matrix of graph G_l, $\tilde{D}_l$ is the degree normalized matrix of adjacency matrix $\tilde{A}_l$, W_g and W_m are learnable parameters.

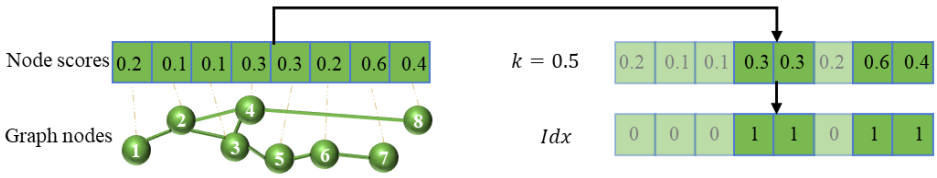

Figure 5. Illustration of the Idx vector

According to the score vector S, the k nodes with the largest score and the edges connecting these nodes in G_l form a pooled graph. The feature matrix of the pooled graph can be obtained as follow:

$$H_{ls} = H^l_{Idx,:} \odot S_{Idx,:}$$

where, $Idx = topk(S, k)$ as is depicted in
Figure 5, $H^l_{Idx,:} \in R^{k \times d}$ and $S_{Idx,:} \in R^{k \times d}$ represent the subset of rows of H^l and S respectively. The subset consists of indexed rows corresponding to elements in Idx that are not equal 0. $\odot$ refers to Hadmard product and $topk$ is a sort function, which returns the node index corresponding to the top k highest scores in the score vector S.

The multi-head attention mechanism is exploited to update the feature of nodes in the pool graph

$$H_{l'} = \Big\|_{i=1}^{T} Att(H_{ls}W_i^q, H_{ls}W_i^k, H_{ls}W_i^v)$$
$$= \Big\|_{i=1}^{T} soft\max\left(\frac{H_{ls}W_i^q(H_{ls}W_i^k)^T}{\sqrt{d_k}}\right)H_{ls}W_i^v$$

where $W_i^k \in R^{d\times d_k}$, $W_i^q \in R^{d\times d_k}$ and $W_i^v \in R^{d\times d_v}$ are all trainable parameters. T indicates the number of heads of attention. $\|$ indicates splicing calculation.

4.3.2. Readout Function

In this section, we discuss how to aggregate the pooled graphs at each layer to learn reasonable graph representation. One idea is directly weight and sum the graph node features of each layer, but it ignores the relationship between layers and loses the advantages of hierarchical feature extraction models. Therefore, in order to preserve the relational information between different layers, as is shown in
Figure 6, we use convolutional aggregation to extract graph-level representations. The feature extraction model proposed in GTP will output different pooling feature graphs. Therefore, the representation of feature tensor learning graph level can be constructed by merging node feature in these pooled graphs:

$$\bar{H} = \Big\|_{i=1}^{C} H_{i'}$$

where C represents the number of pooling graphs, and $H_{i'}$ represents the feature matrix of the i th pooling graph.

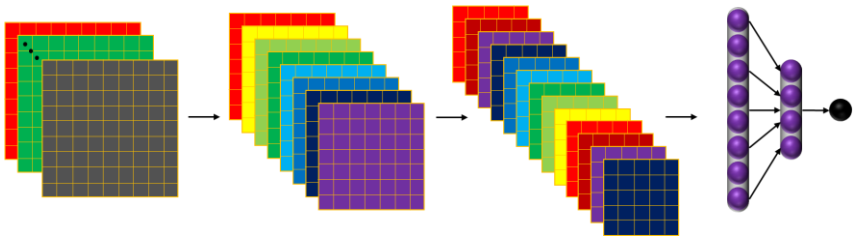

Figure 6. Illustration of CNN readout mechanism

Since $\bar{H}$ is a tensor, a convolutional neural network can be adopted to extract the hierarchical graph-level representation and C can be regarded as the number of channels of a tensor. Finally, we obtain graph-level representations F_g of different subgraphs through a two-layer CNN.

4.3.3. Loss Function

Feeding F_G into the MLP classification layer to get the prediction of the graph

$$\hat{y} = \text{softmax}(\sigma(F_g W_{m1})W_{m2})$$

The loss function is defined as follows

$$L = -\frac{1}{n}\sum_{i,j=1}^{n} y_{ij} \log \hat{y}_{ij}$$

where σ represents the activation function, and W_{m1}, W_{m2} are the trainable parameter matrix of MLP. y_{ij} and $\hat{y}_{ij}$ respectively represent the label and model prediction probability that graph G_i belongs to class j.

5. EXPERIMENTS AND ANALYSIS

In this section, we evaluate the method MVM proposed in the previous section. This experiment is applied to seven graph classification datasets. Then, we will give the elaborate description and analysis of datasets, baseline methods, experimental settings and results.

Table 1 Graph statistical information

Dataset	Numbers of Graph	classes	dimension
D&D	1178	2	89
NCI1	4110	2	37
NCI109	4127	2	38
FRANKENSTEIN	4337	2	780
P388	41472	2	72
UACC257	39988	2	64
TRIANGLES	45000	10	1

5.1. Datasets

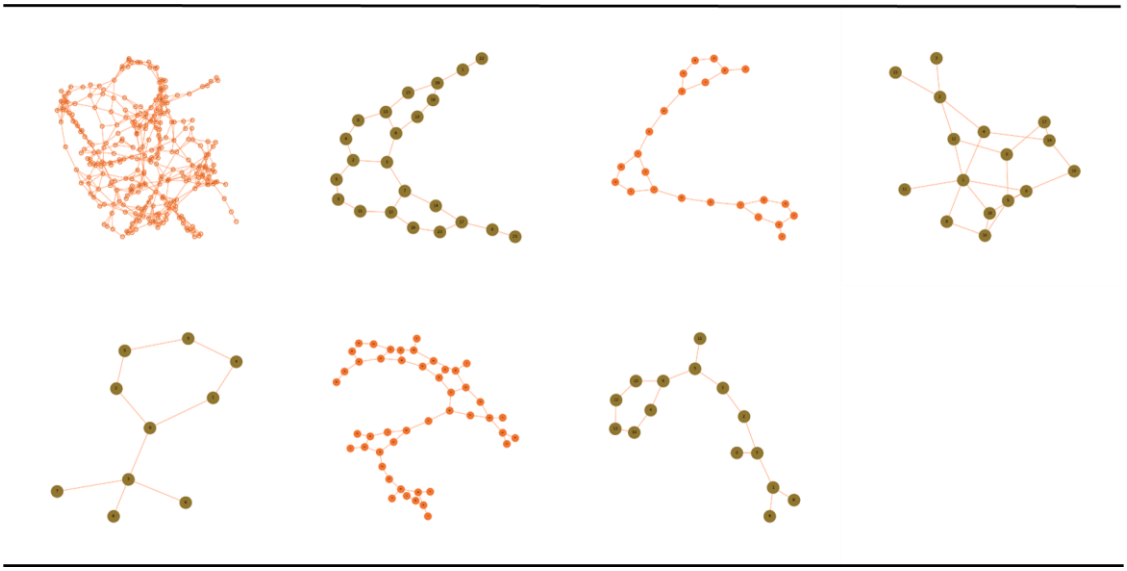

Figure 7. Visualization of graph data

In this experiment, in order to prove the effectiveness of our proposed method, we selected seven graph classification datasets. The statistical information of these datasets is described in detail in Table 1. DD is a protein graph dataset and labels indicates whether the protein is an enzyme.

NCI1 and NCI109 are two chemical structure graph datasets and labels are divided into two categories, indicating whether the chemical molecular structure has anticancer effect and whether it can inhibit the growth of cancer cells respectively. FRANKENSTEIN contains molecules as graph for mutagen classification. P388 and UACC257 belongs to a certain type of cancer screen with the outcome active or inactive. Triangles is a graph dataset that its label represent how many triangles clusters are in each graph.
Figure 7 shows the structure of each dataset.

5.2. Baseline and Experiments Setting

Baseline: We choose several graph pooling models as baseline to compare with our method: set2set, Sortpool, gPool, SAGpool, HGP-SL, EdgePool, DiffPool, ASAP, GMT and MAC. These methods all consider the graph level classification task, and extract the graph structure information in the way of global or hierarchical pooling.

Table 2 Experiment Setting

Parameters	**Scope**	**Parameters**	**Scope**
Learning rate	{0.001,0.0005,0.0001}	Regularization	{0.001,0.0001}
Batchsize	{16,32,64,128,256}	pooling	{0.2,0.3,0.4,0.5,0.6,0.7,0.8}
Hidden layer	{32,64,128}	Dropout	{0.2,0.3,0.4,0.5,0.6,0.7,0.8}
C1	{16,32,64}	Epochs	{300,500,1000}
C2	{32,64,128}	Patience	{50,80}
Network layers K	{2,3,4,5}	dimensionality	$range(d)=\begin{cases}\{d_{L10}, d_{L10}*2,..., d_{L10} & D>10\\ \{1,2,...,D\} & 1<D\le 10\\ \{1\} & D=1\end{cases}$

Experimental Setting: For the fairness of the experiment, we randomly divide all datasets into 10 parts, of which 80% is the training set, 10% is the validation set, and 10% is the test set. We employ tenfold cross validation to evaluate the effectiveness of all methods mentioned above. For baseline methods, we obtain the best result according to the parameter settings on the source code provided by the author. We use pyTorch deep learning framework to implement our method and train the model on a calculating server containing four NVIDIA TITAN X graphics processing unit. The range of all Hyperparameters of our method is shown in Table 2.

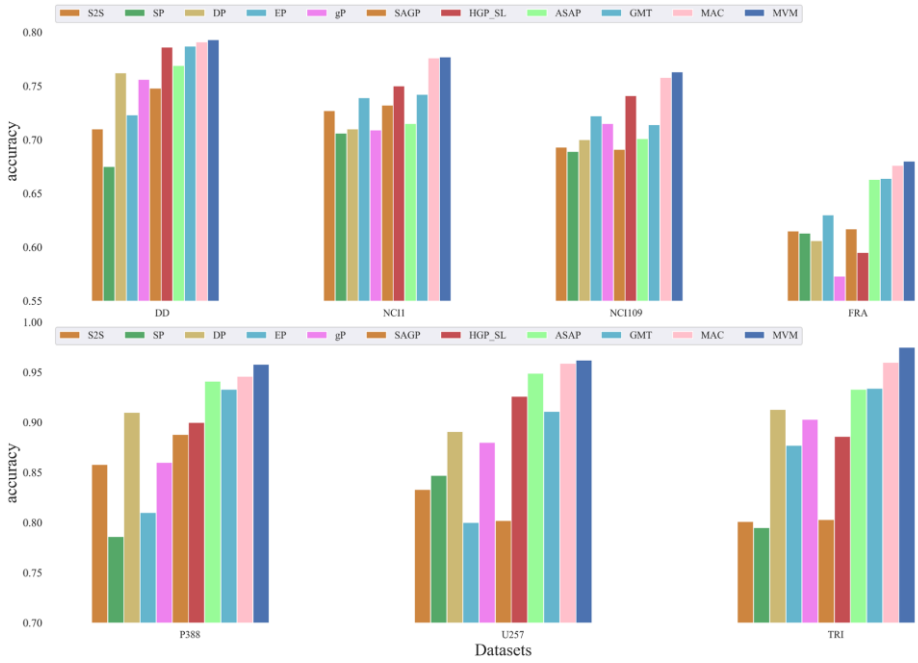

Figure 8. Experimental results

5.3. Performance Results and Analysis

In this section, we will evaluate our module against the baseline methods mentioned above on all datasets and the results are shown in the Figure 8. It can be seen from the table that our method has achieved the best results in all datasets compared with the baseline methods. Since the Maximum Variance Mapping Embedding widens the gap between graphs of different classes, the node features of graphs after embedding are more suitable for graph classification. And benefit from GTP, it can capture the substructure of multi-layer GNN layers and aggregate the relationship between different layers from the perspective of CNN to improve the performance of graph classification.

Table 1 The results of extend experiment

	D&D	NCI1	NCI109	FRANKENSTEIN	TRIANGLES
SAGPool	0.7478	0.7323	0.6908	0.6230	0.8032
MVM-SAGPool	0.7899	0.7615	0.7617	0.6630	0.8577
HGP-SL	0.7861	0.7540	0.7405	0.5951	0.8860
MVM-HGP-SL	0.7903	0.7760	0.7588	0.6237	0.9001
MAC	0.7913	0.7762	0.7584	0.6762	0.9605
MVM-MAC	0.7922	0.7769	0.7607	0.6769	0.9697

5.4. Extend Experiment

In order to prove the effectiveness of the Maximum Variance Mapping Embedding, we combine it with the baseline methods SAGPool, HGP-SL, MAC respectively for comparison. We preserve all the original experimental settings of the baselines and only prepend the GNFE module to the baselines for comparison. From the Table 1, We can observe that all the baseline methods have a significant improvement after incorporating the GNFE module, which proves the effectiveness of the module.

5.5. Effect of Dimensionality Reduction on Results

Table 4 Dimension Sampling Results

Dataset	Original Dimension	Sampling results
D&D	89	{5,10,15,20,25,30,35,40,45,50,55,60,65,70,75,80,85}
NCI109	38	{3,6,9,12,15,18,21,24,27,30,33,36}
FRANKENSTEIN	780	{50,100,150,200,250,300,350,400,450,500,550,600}

In this section, we evaluate the impact of the embedded dimensions on the results. We choose three datasets DD, NCI109, FRANKENSTEIN, which have different feature dimensions. Since the amount of computation is too large for us to calculate the final results for each dimension, we sample the dimensions and calculate the corresponding model accuracy. The results of dimension sampling for the three datasets are shown in Table 4 and the evaluation results are shown in the Figure 9. From the results, we can conclude that the dimensions that achieve better results are generally between 50% and 90% of the original dimension. However, for datasets with low feature dimensionality, the conclusions may differ from the actual situation.

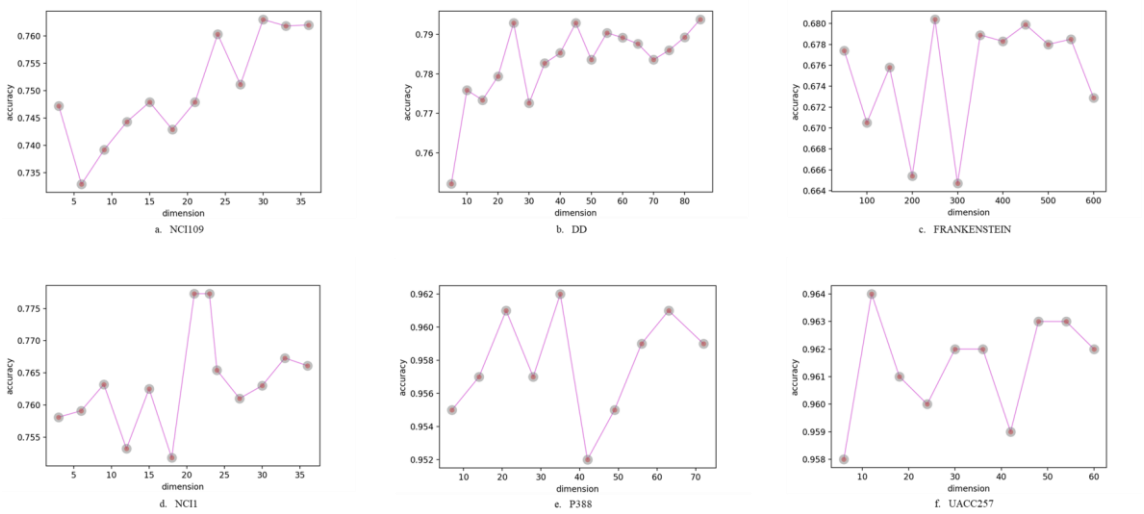

Figure 9. Illustration of dimensionality on results

6. CONCLUSION

This paper proposes a graph pooling method MVM from the perspectives of graph topology and graph features. The GNFE module is used to widen the dissimilarities between the different classes, while reducing the dimensionality of nodes feature. For the difficulty of excessive computation caused by large-scale matrix decomposition, we perform local matrix decomposition by batch, and then establish the relationship between local and global mappings to obtain the global embedding features. Then, the GTP module extracts graph node information and pooling hierarchically, preserving the structural information of each layer. The features of different layers are concatenated together and then aggregated using CNN convolution to conserve more graph information. It can be seen from the experiments that the MVM method outperforms most existing algorithms, and the GNFE module can be easily embedded into other methods and achieve better results. In the future, we will continue to study the influence of the dimension of

dimensionality reduction on the results and explain the dimension range of the optimal result theoretically to reduce unnecessary parameter settings.

ACKNOWLEDGEMENTS

This work was supported by the Open Fund of Science and Technology on Integrated Information System Laboratory under Grant HLJGXQ20210701001.

REFERENCES

[1] LeCun, Y., Bottou, L., Orr, G. B., & Müller, K. R. (2002). "Efficient back-prop". In Neural networks: Tricks of the trade. pp. 9-50. Berlin, Heidelberg: Springer Berlin Heidelberg

[2] Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., & Xiao, B. (2020). Deep high-resolution representation learning for visual recognition. IEEE transactions on pattern analysis and machine intelligence. Vol. 43, No 10, pp. 3349-3364

[3] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... & Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF international conference on computer vision. pp. 10012-10022

[4] Carlini, N., Tramer, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., & Raffel, C. (2021). Extracting Training Data from Large Language Models. In USENIX Security Symposium. Vol. 6.

[5] Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., & Wang, L. (2021). Graph contrastive learning with adaptive augmentation. In Proceedings of the Web Conference 2021. pp. 2069-2080.

[6] Zheng, X., Liu, Y., Pan, S., Zhang, M., Jin, D., & Yu, P. S. (2022). Graph neural networks for graphs with heterophily: A survey. arXiv preprint arXiv:2202.07082.

[7] Zhao, T., Zhang, X., & Wang, S. (2021). Graphsmote: Imbalanced node classification on graphs with graph neural networks. In Proceedings of the 14th ACM international conference on web search and data mining. pp. 833-841.

[8] Cai, L., & Ji, S. (2020, April). A multi-scale approach for graph link prediction. In Proceedings of the AAAI conference on artificial intelligence. Vol. 34, No. 04, pp. 3308-3315.

[9] Sun, X., Yin, H., Liu, B., Chen, H., Cao, J., Shao, Y., & Viet Hung, N. Q. (2021, March). Heterogeneous hypergraph embedding for graph classification. In Proceedings of the 14th ACM international conference on web search and data mining. pp. 725-733.

[10] Ma, T., Wang, H., Zhang, L., Tian, Y., & Al-Nabhan, N. (2021). Graph classification based on structural features of significant nodes and spatial convolutional neural networks. Neurocomputing. Vol. 423, pp. 639-650

[11] Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., & Adams, R. P. (2015). Convolutional networks on graphs for learning molecular fingerprints. Advances in neural information processing systems. Vol. 28.

[12] Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., & Leskovec, J. (2018). Hierarchical graph representation learning with differentiable pooling. Advances in neural information processing systems. Vol. 31.

[13] Lee, J., Lee, I., & Kang, J. (2019). Self-attention graph pooling. In International conference on machine learning. pp. 3734-3743.

[14] Bianchi, F. M., Grattarola, D., & Alippi, C. (2020). Spectral clustering with graph neural networks for graph pooling. In International conference on machine learning. pp. 874-883.

[15] Lee, J., Lee, I., & Kang, J. (2019). Self-attention graph pooling. In International conference on machine learning. pp. 3734-3743.

[16] Gao, X., Dai, W., Li, C., Xiong, H., & Frossard, P. (2021). ipool—information-based pooling in hierarchical graph neural networks. IEEE Transactions on Neural Networks and Learning Systems. Vol. 33, No.9, pp. 5032-5044.

[17] Yuan, H., & Ji, S. (2020). Structpool: Structured graph pooling via conditional random fields. In Proceedings of the 8th International Conference on Learning Representations.

[18] Ranjan, E., Sanyal, S., & Talukdar, P. (2020). Asap: Adaptive structure aware pooling for learning hierarchical graph representations. In Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 34, No. 04, pp. 5470-5477.

[19] Zhang, Z., Bu, J., Ester, M., Zhang, J., Yao, C., Yu, Z., & Wang, C. (2019). Hierarchical graph pooling with structure learning. arXiv preprint arXiv:1911.05954.
[20] Li, B., Huang, D. S., Wang, C., & Liu, K. H. (2008). Feature extraction using constrained maximum variance mapping. Pattern Recognition. Vol. 41, No. 11, pp. 3287-3294.
[21] Zhang, Z., & Zha, H. (2003). Nonlinear dimension reduction via local tangent space alignment. In Intelligent Data Engineering and Automated Learning: 4th International Conference. pp. 477-481.
[22] Xu, Y., Wang, J., Guang, M., Yan, C., & Jiang, C. (2022). Multistructure Graph Classification Method with Attention-Based Pooling. IEEE Transactions on Computational Social Systems.
[23] Pearson, K. (1901). LIII. On lines and planes of closest fit to systems of points in space. The London, Edinburgh, and Dublin philosophical magazine and journal of science. Vol. 2, No. 11, pp. 559-572.
[24] Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. Annals of eugenics. Vol. 7, No. 02, pp. 179-188.
[25] Belkin, M., & Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. Neural computation. Vol. 15, No. 6, pp. 1373-1396.
[26] Donoho, D. L., & Grimes, C. (2003). Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. Proceedings of the National Academy of Sciences. Vol. 100, No. 10, pp. 5591-5596.
[27] Vinyals, O., Bengio, S., & Kudlur, M. (2015). Order matters: Sequence to sequence for sets. arXiv preprint arXiv:1511.06391.
[28] Graves, A., & Graves, A. (2012). Long short-term memory. Supervised sequence labelling with recurrent neural networks. pp. 37-45.
[29] Wang, Z., & Ji, S. (2020). Second-order pooling for graph neural networks. IEEE Transactions on Pattern Analysis and Machine Intelligence.
[30] Diehl, F. (2019). Edge contraction pooling for graph neural networks. arXiv preprint arXiv:1905.10990.
[31] Jo, J., Baek, J., Lee, S., Kim, D., Kang, M., & Hwang, S. J. (2021). Edge representation learning with hypergraphs. Advances in Neural Information Processing Systems. Vol. 34, pp. 7534-7546.
[32] Du, J., Wang, S., Miao, H., & Zhang, J. (2021). Multi-Channel Pooling Graph Neural Networks. In IJCAI. pp. 1442-1448.
[33] Yang, Y., Feng, Z., Song, M., & Wang, X. (2020). Factorizable graph convolutional networks. Advances in Neural Information Processing Systems. Vol. 33, pp. 20286-20296.
[34] Leskovec, Keyulu Xu Weihua Hu Jure, and Stefanie Jegelka. (2019)."How powerful are graph neural networks." ICLR.
[35] Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.
[36] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770-778.

AUTHORS

Lingang Wang received Bachelor degree in Information and Computing Sciences from SDTBU. Currently, he is pursuing his M.Tech in Computational Mathematics from the SYSU. His research interests include manifolds learning and graph neural networks.

Lei Sun received his PhD degree in computational mathematics from the NUDT, Changsha, China, in 2010. She is currently an associate professor in the School of Systems Science and Engineering at SYSU in Guangzhou, China.