EAI Endorsed Transactions
on Scalable Information Systems

Research Article EAI.EU

MIED : An Improved Graph Neural Network for Node Embedding in Heterogeneous Graphs

Mingjian Ni[1], Yinghao Song[2], Gongju Wang[2], Lanxiao Feng[3], Yang Li[2], Long Yan[2], Dazhong Li[2], Yanfei Wang[4], Shikun Zhang[1], Yulun Song[2,*]

[1]School of Software and Microelectronics, Peking University, Beijing, China
[2]Data Intelligence Division, China Unicom Digital Technology Co., Ltd., Beijing, China
[3]China Unicom Network Communications Co., Ltd., Beijing, China
[4]Unicom (Beijing) Industrial Internet Co., Ltd., Beijing, China

Abstract

This paper proposes a Metapath-Infused Exponential Decay graph neural network (MIED) approach for node embedding in heterogeneous graphs. It is designed to address limitations in existing methods, which usually lose the graph information during feature alignment and ignore the different importance of nodes during metapath aggregation. Firstly, graph convolutional network (GCN) is applied on the subgraphs, which is derived from the original graph with given metapaths to transform node features. Secondly, an exponential decay encoder (EDE) is designed, in which the influence of nodes on starting point decays exponentially with a fixed parameter as they move farther away from it. Thirdly, a set of experiments is conducted on two selected datasets of heterogeneous graphs, i.e., IMDb and DBLP, for comparison purposes. The results show that MIED outperforms selected approaches, e.g., GAT, HAN, MAGNN, etc. Thus, our approach is proven to be able to take full advantage of graph information considering node weights based on distance aspects. Finally, relevant parameters are analyzed and the recommended hyperparameter setting is given.

Received on 05 September 2023; accepted on 25 September 2023; published on 27 September 2023

Keywords: Heterogeneous Graph, Node Embedding, Metapath, Graph Convolutional Network, Exponential Decay Encoder

Copyright © 2023 Mingjian Ni *et al.*, licensed to EAI. This is an open access article distributed under the terms of the Creative Commons Attribution license (http://creativecommons.org/licenses/by/4.0/), which permits unlimited use, distribution and reproduction in any medium so long as the original work is properly cited.

doi:10.4108/eetsis.3824

1. Introduction

Graph data, utilizing nodes and edges to represent entities and their relationships, is a powerful structure for modeling complex real-world data [1, 2], including social networks [3] and academic citation networks [4, 5]. Computing the embedding vectors of the nodes of the graph in the low-dimensional space is an important work. Compared to other types of data, graph data provides flexibility and expressive power for handling intricate relationships [6]. To take advantage of this, a variety of graph embedding techniques [7] have been developed, such as HOPE [8] and GraRep [9] that use matrix factorization [10], DeepWalk [11] and Node2Vec [12] that rely on random walks. Powerful models like Graph Neural Networks (GNNs) [13] can also be used to calculate embedding. These methods are carefully designed and demonstrate strong embedding capabilities and primarily focus on homogeneous graph data [14].

However, in the face of real-world data, these techniques encounter difficulties. Often, data in many cases presents not as homogeneous graphs but as heterogeneous structures [15]. These diverse entities and relationships reflect the complexity of real-world systems such as movie recommendation systems [16]. Recognizing these complexities, researchers propose many heterogeneous graph embedding algorithm and metapath-based [17] methods are widely used in this area. Metapaths are predefined sequences of node types that encapsulate high-order semantic relationships within a heterogeneous graph. By simplifying complex relationships into homogeneous subgraphs, we can

[*]Corresponding author. Email: songyl100@chinaunicom.cn

apply the homogeneous graph embedding algorithms that were previously discussed.

MAGNN [17], for instance, is one of best methods that integrate metapaths and attention mechanism to achieve efficient node representation learning for heterogeneous graph. Firstly, it uses dense layers to transform the features of different type of nodes into the same dimension. Then it uses metapaths to connect two same type nodes which are not directly connected in the original graph and aggregates the features of the metapaths. When combining results of different metapaths or their instances, it uses attention mechanism to find appropriate weights. This method has strong adaptability. Through the metapaths, the efficient aggregation of information in various heterogeneous graphs can be realized, and the attention mechanism is used to assign appropriate weights to different metapaths, so as to extract as much information as possible from the graph.

Despite these advantages, metapath-based methods still have room for improvement. Current models often neglect important adjacency relationships between nodes during feature alignment. To address these limitations, we propose a novel model called Metapath-Infused Exponential Decay graph neural network (MIED) for heterogeneous graph embedding. The model includes three parts: GCN-based node feature alignment, exponential decay encoder and metapath aggregation. Specifically, MIED first utilizes GCN to align the node features in the heterogeneous graph, enabling the aligned features to have the same dimension and reside in a unified feature space. Additionally, GCN effectively utilizes the neighborhood information based on metapaths during feature alignment, resulting in improved graph embeddings for downstream tasks. Next, when encoding features on metapath instances we introduce an exponential decay encoder to aggregate the features of nodes on the metapath with varying importance due to their distance to starting node. Finally, MIED employs the attention mechanism for metapath aggregation, fusing latent vectors obtained from multiple metapaths into the final node embeddings.

In summary, this work makes the following major contributions:

(1) We propose a novel GCN-based node feature alignment method for metapath-based heterogeneous graph node embedding.

(2) We design an effective encoder function for metapath instances called exponential decay encoder which reasonably encode the features on the metapath according to the node importance.

(3) We conduct extensive experiments on the IMDb [18] and DBLP datasets [19] for node classification and node clustering, demonstrating that the node embeddings learned by MIED consistently outperform those generated by other state-of-the-art baselines.

2. Related works

This chapter will focus on heterogeneous graph representation learning methods relevant to our research. Through a comprehensive analysis of these methods, we will uncover their characteristics in practical applications and gain valuable insights for our research.

2.1. Graph Neural Networks

GNN [13, 20] is a kind of specialized machine learning models designed for processing graph data. Graph Convolutional Networks (GCNs) extend convolution operation to GNN field. However, GCN suffers from "over-smoothing" [21–23] where its performance decreases with an increasing number of layers. Several algorithms have been proposed to address this problem or extend the GCN model. For example, Simplified Graph Convolution (SGC) [24] simplifies GCN by removing non-linear activation functions to enhance the propagation mechanism.

GCN requires that all nodes of the graph are present during training and do not naturally generalize to new added nodes. GraphSage [25] is designed to learn aggregators that samples and aggregates features from a node's neighbors. Therefore, it allows for efficiently generate node embeddings for previously unseen data.

Graph Attention Networks (GAT) [26] introduced attention-based neighborhood aggregation, allowing each node to attend to its neighbors and update its representation. GATv2 [27] modifies the order of internal operations in its attention function, working better in some cases.

2.2. Heterogeneous Graph Embedding

The distinct structures and properties of heterogeneous graphs make the direct application of homogeneous algorithms unsuitable [28–30]. To address these limitations, researchers have introduced heterogeneous graph embedding techniques.

In this area, early algorithms like Metapath2Vec [31] utilized metapaths to generate node sequences in heterogeneous networks, capturing complex semantic relationships. However, they primarily focused on local structures, neglecting global information. To rectify this, Heterogeneous Graph Attention Networks (HAN) [32–34] and ESim [35] were introduced. HAN incorporates node types and neighbor data through a hierarchical attention mechanism, enhancing the capture of node attributes and topological data, and ESim provides a embedding-based similarity search framework for heterogeneous information networks, enabling effective handling of large-scale networks and capturing rich semantics.

In the domain of analyzing heterogeneous graph networks, MAGNN [17] has demonstrated promising performance. This method leverages a multi-head attention mechanism to adaptively allocate weights during the learning process, enabling a more comprehensive exploration of node data. MAGNN enhances graph embedding by aggregating neighbor features via combining node content transformation, intra-metapath and inner-metapath aggregation.

Furthermore, the application of knowledge graphs in online learning frameworks has been explored, with a focus on access control decision-making [36, 37]. Another notable work is SMiLE, which presents a schema-augmented multi-level contrastive learning approach for knowledge graph link prediction [38].

3. Definition

When describing the symbolic definition of heterogeneous graph embedding, we first need to clarify the symbolic representations of concepts. To facilitate the understanding of the symbols used throughout the discussion, Table 1 provides a comprehensive list of symbols and their respective descriptions.

Heterogeneous Graph. A heterogeneous graph is represented as $G = (V, E)$, where V and E represent the sets of nodes and edges. T and R represent the sets of node types and edge types. $\psi_V : V \rightarrow T$ and $\psi_E : E \rightarrow R$ denote the node type mapping function and the edge type mapping function with $|T| + |R| > 2$.

Metapath. In the context of heterogeneous graphs, a metapath is a sequence defined by alternating nodes and edges. A metapath P is formally defined by the Equation (1), where t_i and r_i respectively represent the node types and edge types in the graph.

$$P = t_1 \xrightarrow{r_1} t_2 \xrightarrow{r_2} t_3 \xrightarrow{r3} \ldots \xrightarrow{r_{n-1}} t_n \tag{1}$$

Metapath-based Neighbor. For a given metapath P and a node $v \in V$, the metapath-based neighbors of node v, denoted as $N_P(v)$, are defined as nodes that can be reached from node v through the metapath P.

Metapath-based Subgraph. For a node type t and a given metapath P, we can derive a homogeneous subgraph $G_P^t = (V^t, E_P^t)$ from graph G. We define the adjacency matrix $A_P^t = [a_{i,j}] \in \mathbb{R}^{n_t \times n_t}$, where n_t represents the number of nodes of type t. If node v_i is adjacent to node v_j with respect to metapath P, then $a_{i,j} = 1$ else $a_{i,j} = 0$. The node feature matrix for node type t is represented as $X^t \in \mathbb{R}^{n_t \times d_t}$, where d_t represents the dimension of the original node features.

Heterogeneous Graph Embedding: For a node type t in heterogeneous graph G, $X^t \in \mathbb{R}^{n_t \times d_t}$ denotes its feature matrix, in which d_t represents the dimension of node features. Heterogeneous graph embedding aims to learn d-dimensional node representations $H^t \in \mathbb{R}^{n_t \times d}$ (usually $d \ll d_t$) which captures structural information for all nodes of type t.

4. Methodology

Our model MIED consists of three main parts: GCN-based node feature alignment, exponential decay encoder and metapath aggregation. It is outlined in Figure 1. The calculation process of heterogeneous graph embedding is illustrated in Algorithm 1.

4.1. GCN-based node feature alignment

In heterogeneous graphs, different types of nodes may have different feature dimensions. Even with the same dimension, different structures may have different feature distributions, demanding feature space alignment. In order to take into account the structural information of the graph, we propose node feature alignment using GCN.

If the adjacency matrix of a homogeneous graph is denoted as A, and I is its corresponding identity matrix, the output of $(l+1)$-th GCN layer $H^{(l+1)}$ is calculated in the form of Equation (2), where $\tilde{A} = A + I$, $H_P^{t\,(0)} = X^t$, $\tilde{D}_{i,i} = \sum_j \tilde{A}_{i,j}$ is a diagonal matrix and $W^{(l)}$ is the weights of l-th layer. ϕ is used to denote the activation function, such as $ReLU(x) = \max(0, x)$ or $sigmoid(x) = \frac{1}{1+e^x}$.

$$H^{(l+1)} = \phi(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{2}$$

For a heterogeneous graph G and a node type t, if given a metapath P, we can derive a homogeneous subgraph from G by using P and denote its adjacency matrix as A_P^t. We can apply GCN for each homogeneous subgraph and can get the transformed hidden states as in Equation (3).

$$H_P^{t\,(l+1)} = \phi(\tilde{D}_P^{t^{-\frac{1}{2}}}\tilde{A}_P^t\tilde{D}_P^{t^{-\frac{1}{2}}}H_P^{t\,(l)}W^{t\,(l)}) \tag{3}$$

If given a set of metapaths $\mathcal{P}^t = \{P_1, P_2, ..., P_n\}$ for node type t, We can apply GCN to each metapath according to the above method. Then we sum the hidden states computed by each metapath and get the final hidden states for each node of type t, as shown in Equation (4), where H_P^t represents the final output of GCN layers.

$$H^t = \sum_{P \in \mathcal{P}^t} H_P^t \tag{4}$$

During the alignment process, the features of each node type are transformed into the same dimension, and feature space and graph information from neighbor nodes is aggregated, enriching the information contained in the aligned results.

Table 1. Symbols and Descriptions.

Symbol	Description
G	Graph
V	Set of nodes in the graph G
E	Set of edges in the graph G
T	Set of node types
R	Set of edge types
ψ_V	Function mapping nodes to their types
ψ_E	Function mapping edges to their types
P	Metapath, a sequence defined by alternating nodes and edges
$N_P(v)$	Set of metapath-based neighbors of node v
G_P^t	Homogeneous subgraph derived from G using metapath P for node type t
A_P^t	Adjacency matrix for node type t derived using metapath P
X^t	Node feature matrix for node type t
$h_{\mathcal{P}^t}^v$	Aggregated information representation of node v for node type t
$\tilde{A}$	Adjusted adjacency matrix
$\tilde{D}$	Diagonal matrix corresponding to $\tilde{A}$
$H^{(l+1)}$	Output of the $l+1$-th GCN layer
$W^{(l)}$	Weights of the l-th GCN layer
α	Decay parameter in the exponential decay encoder

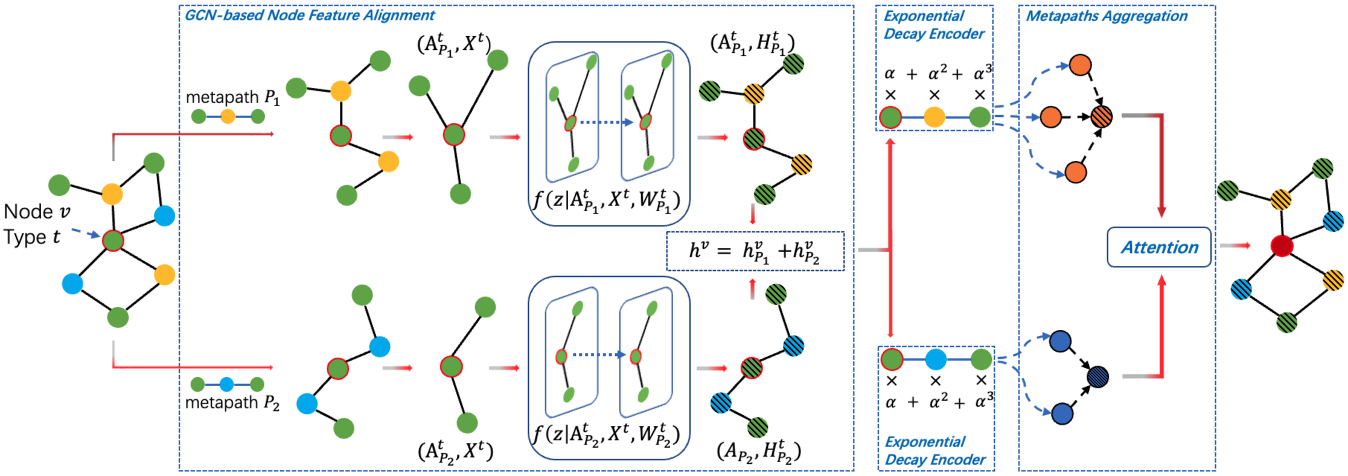
GCN-based Node Feature Alignment
Exponential Decay Encoder
Metapaths Aggregation
Node v
Type t
metapath P_1
metapath P_2
$(A_{P_1}^t, X^t)$
$(A_{P_1}^t, H_{P_1}^t)$
$f(z|A_{P_1}^t, X^t, W_{P_1}^t)$
$h^v = h_{P_1}^v + h_{P_2}^v$
$(A_{P_2}^t, X^t)$
$f(z|A_{P_2}^t, X^t, W_{P_2}^t)$
$(A_{P_2}, H_{P_2}^t)$
$\alpha + \alpha^2 + \alpha^3$
× × ×
× × ×
$\alpha + \alpha^2 + \alpha^3$
Exponential Decay Encoder
Attention

Figure 1. Framework of MIED (GCN feature transformations for yellow and blue nodes are executed but not illustrated for clarity).

Regarding computational complexity, in comparison to the baseline MAGNN, the addition of GCN-based node feature alignment involves GCN aggregation for each node type. Consequently, the increased computational complexity can be expressed as $\sum_{i=1}^{|T|}(N_i \cdot O(M_i \cdot K_i))$, where $|T|$ represents the number of node types, N_i denotes the number of nodes of the i-th type, M_i signifies the mean number of neighbors for the i-th type, and K_i represents the number of features for the i-th type. In many real-world graph data, the mean number of neighbors and the number of features for different node types may be constrained by a constant. Therefore, if we denote the maximum number of neighbors and features by M and K respectively, the total computational complexity increased by GCN-based node feature alignment compared to MAGNN is $O(M \cdot K \cdot N)$, where N corresponds to the total number of nodes in the graph.

4.2. Exponential Decay Encoder

After feature alignment, feature aggregation on the metapath must consider that nodes farther from the starting point have less relevance. Some proposed methods use average encoder, linear encoder or RNN-based encoder, but they lack effectiveness in capturing node weight decay on a metapath because of a long path of gradient backpropagation. MAGNN's relational rotation encoder [39] mitigates this to some extent but increases computational complexity and has potential for improvement.

We propose an exponential decay encoder (EDE) for better feature aggregation on a metapath. Given a metapath P for node type t, the encoder function is $f_\theta(P(v,u))$, where $P(v,u)$ denotes the metapath from target node v to a neighbor instance $u \in N_P(v)$. The aggregation process is in Equation (5), with $h_P^{(i)} \in \mathbb{R}^d$ as the aligned feature vector of the i-th node on $P(v,u)$ from v to u and α as the decay parameter. The length of $P(v,u)$ is denoted $|P(v,u)|$.

$$\begin{aligned} h_{P(v,u)} &= f_\theta(P(v,u)) \\ &= f_\theta(h_v, ..., h_u) \\ &= \sum_{i=1}^{|P(v,u)|} \alpha^i * h_P^{(i)} \end{aligned} \tag{5}$$

Algorithm 1 : MIED forward propagation.

```
Input: The heterogeneous graph $G = (V, E)$ ,
        node types $T = \{t_1, t_2, ..., t_{|T|}\}$,
        metapaths $\mathcal{P}^t = \{P_1, P_2, ..., P_{|\mathcal{P}^t|}\}$ for node type $t$ ,
        node features $\{X^t, \forall t \in T\}$,
        the number of attention heads $K$,
        the number of layers $L$,
        the number of GCN layers $I$,
        the exponential decay weight $\alpha$
Output: The node embeddings $\{h^v, \forall v \in V\}$
for node type $t \in T$ do
    for metapath $P \in \mathcal{P}^t$ do
        Set ${H_P^t}^{(0)} = X^t$;
        for $i = 1, 2, .., I$ do
            ${H_P^t}^{(i)} = \phi({\tilde{D}_P^t}^{-\frac{1}{2}} \tilde{A}_P^t {\tilde{D}_P^t}^{-\frac{1}{2}} {H_P^t}^{(i-1)} W^{t(i-1)})$;
        end for
    end for
    Aligned node feature matrix $H^t = \sum_{P \in \mathcal{P}^t} {H_P^t}^{(I)}$;
end for
for $l = 1, 2, ..., L$ do
    for node type $t \in T$ do
        for metapath $P \in \mathcal{P}^t$ do
            for $v \in V^t$ do
                Calculate $h_{P(v,u)}^l$ for all $u \in N_P(v)$ using EDE refer to Equation (5);
                Calculate $\beta_{P(v,u)}$ refer to Equation (7);
                Combine extracted metapath instances of multi-head attention:
                $[h_P^v]^l = \|_{k=1}^K \sigma(\sum_{u \in N_P(v)} \beta_{P(v,u)}^k \cdot h_{P(v,u)}^l)$;
            end for
        end for
        Calculate $\gamma_P$ for each metapath $P \in \mathcal{P}^t$ refer to Equation (12) and get :
        $[h_{\mathcal{P}^t}^v]^l = \sum_{P \in \mathcal{P}^t} \gamma_P \cdot [h_P^v]^l$ ;
    end for
    Layer output projection: $[h^v]^l = \sigma(W_o \cdot [h_{\mathcal{P}^t}^v]^l)$
end for
```

In order to reduce the impact of different magnitudes of features on EDE, techniques such as normalization can also be used to normalize the aligned features. Many normalization methods can be used, such as Min-Max normalization and Z-score normalization [40]. During actual training, these methods can be set as a hyperparameter to let the model choose the most suitable normalization method.

The computational complexity of EDE on a metapath instance with n nodes is $\sum_{i=1}^{n}(i - 1 + K) = O(n^2) + O(K \cdot n)$, where K represents the number of aligned features. In contrast, for a similar method, the relational rotation encoder of the baseline model has a computational complexity of $K + \sum_{i=1}^{n}(2 \cdot K) = O(K \cdot n)$. EDE has an increase of $O(n^2)$, but n represents the length of a metapath, which is usually not large and much smaller than K. Additionally, EDE reduces data storage space from $O(K \cdot n)$ to $O(1)$ compared with relational rotation encoder.

4.3. Metapath Aggregation

Upon aggregating features of nodes on each metapath instance to $h_{P(v,u)}$, we seek to aggregate all metapath instances linking node v. Appropriate weights must be assigned to each metapath instance's vector representation, as they influence the target node v differently. A graph attention layer is applied to metapath instances related to v, allowing the model to find optimal weights. The process uses $\|$ for vector concatenation.

$$e_{P(v,u)} = LeakReLU(a_P^T \cdot [h_v \| h_{P(v,u)}]) \tag{6}$$

$$\beta_{P(v,u)} = \frac{exp(e_{P(v,u)})}{\sum_{s \in N_p(v)} exp(e_{P(v,s)})} \tag{7}$$

$$h_P^v = \sigma(\sum_{u \in N_P(v)} \beta_{P(v,u)} \cdot h_{P(v,u)}) \tag{8}$$

Similarly, this method can be extended to incorporate a multi-head attention mechanism, as shown in Equation (9). Here, K represents the number of attention heads and $\beta_{P(v,u)}^k$ represents the relative contribution value of the k-th attention head.

$$h_P^v = \|_{k=1}^K \sigma(\sum_{u \in N_P(v)} \beta_{P(v,u)}^k \cdot h_{P(v,u)}) \tag{9}$$

After aggregating the instances of the metapath P for node v, we need to further aggregate the information about the metapath set $\mathcal{P}^t = \{P_1, P_2, ..., P_n\}$ for node type t. For a node v, we denote $\{h_{P_1}^v, h_{P_2}^v, ..., h_{P_m}^v\}$ as the aggregated representation of each metapath, where m represents the number of metapaths corresponding to its node type t. Therefore, considering the different contributions of metapaths, we can apply an attention mechanism to find the weights of these metapaths.

Firstly, we average the representations of all nodes of type t with each metapath $P \in \mathcal{P}^t$, as shown in Equation (10). Here, $|V^t|$ represents the number of nodes of type t, $M^t \in \mathbb{R}^{d_m \times d}$ and $b^t \in \mathbb{R}^{d_m}$ are the weight parameters and bias vector. We denote the dimension of parameterized attention vector as d_m.

$$s_P = \frac{1}{|V^t|} \sum_{v \in V^t} tanh(M^t \cdot h_P^v + b^t) \tag{10}$$

Then the attention mechanism can be represented as follows, where $q^t \in \mathbb{R}^{d_m}$ is the parameterized attention vector. By summing all the weighted vectors, we calculate the aggregation of information for node v.

$$e_P = (q^t)^T \cdot s_P \tag{11}$$

$$\gamma_P = \frac{exp(e_P)}{\sum_{P_i \in \mathcal{P}^t} exp(e_{P_i})} \tag{12}$$

$$h^v_{\mathcal{P}^t} = \sum_{P \in \mathcal{P}^t} \gamma_P \cdot h^v_P \tag{13}$$

Finally, we use a linear layer followed by a non-linear activation function to map the aggregation of information for node v into the desired dimension of the vector feature space. This can be represented by Equation (14), where σ is the activation function and $W_o \in \mathbb{R}^{d_o \times d}$ is the weight matrix. The entire equation can be seen as the output layer that connects to downstream tasks.

$$h^v = \sigma(W_o \cdot h^v_{\mathcal{P}^t}) \tag{14}$$

4.4. Training

After obtaining the embedding representation h^v for each node, using the node labels, we perform backpropagation and gradient descent to minimize the cross-entropy and optimize the model weights. The loss function can be expressed by Equation (15), where C represents the number of classes, $y^v[c]$ represents the one-hot vector of the node label c, and $log(h^v[c])$ represents the predicted vector for the node label c.

$$\mathcal{L} = -\sum_{v \in V_L} \sum_{c=1}^{C} y^v[c] \cdot log(h^v[c]) \tag{15}$$

The MIED forward propagation algorithm, as delineated in Algorithm 1, is meticulously designed to produce node embeddings from a heterogeneous graph G with its nodes and edges.

Starting with the inputs, the algorithm receives the heterogeneous graph G, node types T, metapaths $\mathcal{P}^t$ for each node type t, initial node features X^t, the number of attention heads K, the number of layers L, the number of GCN layers I, and the exponential decay weight α. The end goal is to compute the node embeddings h^v for every node v in G.

For every node type t, the algorithm, in lines 1-9 of Algorithm 1, initializes the feature matrix for each related metapath P using X^t and subsequently refines it over I GCN layers. The formula $H_P^{t\,(i)}$ (line 5 of Algorithm 1) depicts the GCN transformation, which incorporates adjacency and degree matrices with the node features to capture the graph structure. After iterating over all metapaths, the aligned node feature matrix is produced in line 8 by summing the outputs.

Delving deeper, lines 10-24 of Algorithm 1 iterate over L layers. Within each layer, for every node v of type t, the algorithm calculates features for all its neighbors using the metapath and the Exponential Decay Encoder (EDE) as detailed in line 14 of Algorithm 1. The attention mechanism is then invoked in lines 15-17 to aggregate these features into a single representation for the node. This ensures that more significant neighboring nodes, as described by the metapath, have a stronger influence on the node's representation.

Subsequent to aggregating features for each metapath, the algorithm, in line 20 of Algorithm 1, computes weights γ_P for every metapath. This information is then employed in line 21 to combine embeddings across metapaths, forming a complete representation for each node type. Lastly, line 26 of Algorithm 1 applies a dense layer to refine these embeddings, making them suitable for downstream tasks.

By intertwining information from various metapaths in a heterogeneous graph, and leveraging the power of GCN and attention mechanisms, this algorithm efficiently derives rich and informative node embeddings.

5. Experiment

In this section, we apply MIED on two datasets to compare MIED with baselines on node classification task and node clustering task. We also try different normalization strategies and different hyperparameters α for exponential decay encoder to further explore their impact.

5.1. Experimental Settings

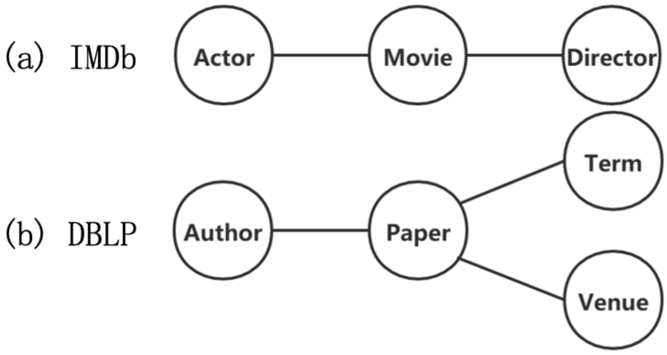

Figure 2. Schemas of the two heterogeneous graph datasets used in experiment.

Dataset. We experiment with two heterogeneous graph datasets, similar to the base model MAGNN. The detailed information of the two datasets is shown in Table 2 and the schemas of the datasets are shown in Figure 2. IMDb is a database about movies and television programs and has three kinds of labels:

Table 2. Dataset details.

Dataset	Node	Edge	Metapath
IMDb	movie(M):4278 director(D):2081 actor(A):5257	M-D:4278 M-A:12828	MDM MAM DMD DMAMD AMA AMDMA
DBLP	author(A):4057 paper(P):14328 term(T):7723 venue(V):20	A-P:19645 P-T:85810 P-V:14328	APA APTPA APVPA

Action, Comedy and Drama. We divide the IMDb dataset into training, validation, and testing sets with 400, 400 and 3478 nodes, respectively. DBLP dataset is about computer science bibliography [41] and we use a subset of the dataset extracted by [17]. The authors in DBLP have three kinds of labels: Database, Data Mining, Artificial Intelligence and Information Retrieval. We divide the DBLP dataset into training, validation, and testing sets with 400, 400 and 3257 nodes, respectively. We use one-hot id vectors as input features for nodes with no features in these datasets. We conduct node classification and node clustering tasks on these two datasets to evaluate the performance of our model.

Baselines and Hyperparameters. We compare MIED with various types of graph embedding models including MAGNN. These models include homogeneous models such as node2vec, GCN and GAT, as well as heterogeneous models such as ESim, metapath2vec, HERec and HAN. For MIED, we use the same settings and metapaths with MAGNN, if applicable. We set dropout rate to 0.5 and learning rate to 0.005. The number of attention heads is set to 8 and the dimension of attention vectors is set to 128. We set the dimension of the aligned features to 64 for MAGNN and MIED. For exponential decay encoder, we conduct a grid search on the weight decay parameter, using both original and Z-score normalized inputs to find the optimal model.

5.2. Node Clustering

We use the same split method of training, validation and testing sets and use K-Means [42] algorithm to cluster embeddings of labeled nodes into the number of classes for each dataset. Normalized mutual information (NMI) [43] and adjusted rand index (ARI) [44] are used as the evaluation metrics. MIED is tested over 10 runs, and we report the average results in Table 3. We can see that MIED outperforms MAGNN and other baselines in all metrics. When compared with the base model, MIED has improved by a maximum of 7.23% and a minimum of 1.27%. For the IMDb dataset, the best result is attained when using a decay value of 2/3 on original input. For the DBLP dataset, the best result is attained when using a decay value of 2/3 on Z-score normalized input.

5.3. Node Classification

To evaluate the embeddings of labeled nodes generated by each model, we use a linear support vector machine (SVM) [45] to classify them with varying training proportions. We test MIED over 10 runs and report the averaged Macro-F1 and Micro-F1, as shown in Table 4. We can see that MIED performs best across all metrics. When compared with base model, MIED has improved by 1.97% at most when the dataset is IMDb and the training proportion is 20%. For the IMDb dataset, the best result is attained when using a decay value of 1/2 on original input. For the DBLP dataset, the best result is attained when using a decay value of 1/3 on original input.

5.4. Module Analysis

To further explore the effect of two modules, we train a model that only applies the GCN module to MAGNN on the same data in 5.2 and 5.3, and collect results for comparison and analysis. We average the scores of node classification in different training proportions and show them in Table 5.

As shown in the table, after incorporating GCN method in MAGNN, the model performs better across all metrics. This shows that replacing the dense layer with GCN is effective. After implementing the EDE, the performance of the model is further improved. Among the 8 sets of comparative data, 7 sets of data have shown further improvement, which shows EDE is also effective. This table shows that our methods enhance metapath based heterogeneous graph embedding.

5.5. Parameter Analysis

We also try to inspect the influence of hyperparameters in the two modules. For GCN, we use one layer in our best result. We also experiment with using two layers of GCN, but the performance of the model get worse. We think the reason is when using two (or more) layers of GCN, the model will consider neighbors that are two (or more) hops away in the homogeneous graphs derived from metapaths, which are actually much far away in the original heterogeneous graph.

For the exponential decay encoder, an important parameter is α, and data normalization is also an important factor. So more experiments are done to inspect the influence of these two factors. We search best α in a set of $\{1/3, 1/2, 2/3, 1\}$ on both original input and Z-score normalized input when

Table 3. Experimental results (%) on the datasets for the node clustering task (n2vec is short for node2vec and m2vec is short for metapath2vec in the table).

Dataset	Metrics	Unsupervised				Semi-supervised				
		n2vec	ESsim	m2vec	HERec	GCN	GAT	HAN	MAGNN	MIED
IMDb	NMI	5.22	1.07	0.89	0.39	7.46	7.84	10.79	15.58	**15.84**
	ARI	6.02	1.01	0.22	0.11	7.69	8.87	11.11	16.74	**17.95**
DBLP	NMI	77.01	68.33	74.18	69.03	73.45	70.73	77.49	80.81	**81.84**
	ARI	81.37	72.22	78.11	72.45	77.50	76.04	82.95	85.54	**86.66**

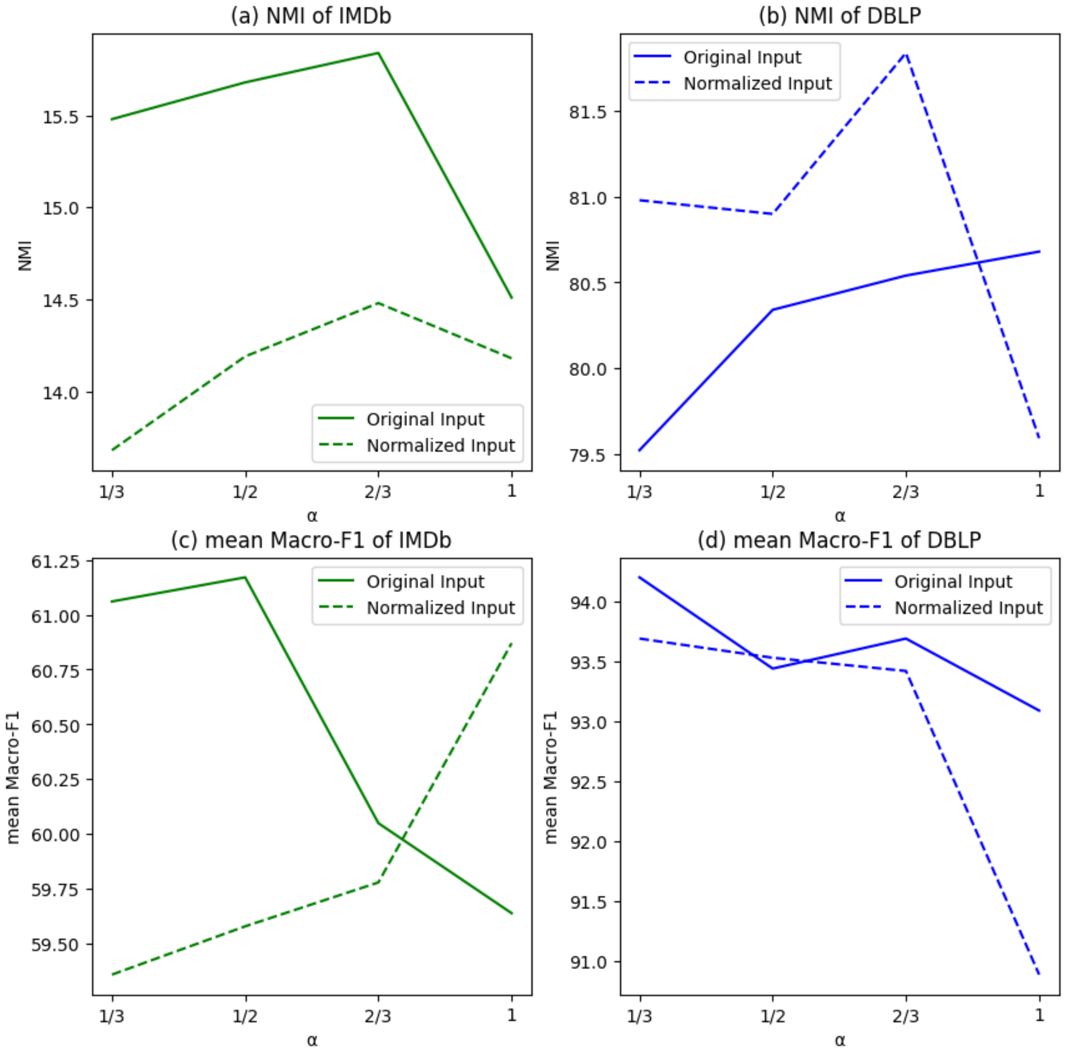

Figure 3. NMI and mean Macro-F1 of IMDb and DBLP with different settings.

implementing exponential decay encoder. We utilize NMI of node clustering and mean Macro-F1 of node classification as metrics to analyze the influence of different hyperparameters on the model. The result is shown in Figure 3.

It can be seen from the figure that the results vary greatly with different settings. Original input outperforms normalized input in most cases except in the node clustering task of DBLP. When α changes, the trends of the two lines in the (a) and (d) subgraph are similar, but in the other two images, one line shows a downward trend and the other line shows an upward trend. So we can not identify an α that performs best in all cases because different datasets exhibit distinct distributions and vary significantly in terms of magnitudes. However, we can still figure out

Table 4. Experimental results (%) on the datasets for the node classification task (n2vec is short for node2vec and m2vec is short for metapath2vec in the table).

Dataset	Metrics	Train	Unsupervised				Semi-supervised				
			n2vec	ESsim	m2vec	HERec	GCN	GAT	HAN	MAGNN	MIED
IMDb	Macro-F1	20%	49.00	48.37	46.05	45.61	52.73	53.64	56.19	59.35	**60.52**
		40%	50.63	50.09	47.57	46.80	53.67	55.5	56.15	60.27	**61.08**
		60%	51.65	51.45	48.17	46.84	54.24	56.46	57.29	60.66	**61.20**
		80%	51.49	51.37	49.99	47.73	54.77	57.43	58.51	61.44	**61.86**
	Micro-F1	20%	49.94	49.32	47.22	46.23	52.80	53.64	56.32	59.60	**60.68**
		40%	51.77	51.21	48.17	47.89	53.76	55.56	57.32	60.50	**61.30**
		60%	52.79	52.53	49.87	48.19	54.23	56.47	58.42	60.88	**61.42**
		80%	52.72	52.54	50.50	49.11	54.63	57.40	59.24	61.53	**62.13**
DBLP	Macro-F1	20%	86.70	90.68	88.47	90.82	88.00	91.05	91.69	93.13	**93.95**
		40%	88.07	91.61	89.91	91.44	89.00	91.24	91.96	93.23	**94.18**
		60%	88.69	91.84	90.50	92.08	89.43	91.42	92.14	93.57	**94.22**
		80%	88.93	92.27	90.86	92.25	89.98	91.73	92.50	94.10	**94.43**
	Micro-F1	20%	87.21	91.21	89.02	91.49	88.51	91.61	92.33	93.61	**94.38**
		40%	88.51	92.05	90.36	92.05	89.22	91.77	92.57	93.68	**94.59**
		60%	89.09	92.28	90.94	92.66	89.57	91.97	92.72	93.99	**94.64**
		80%	89.37	92.68	91.31	92.78	90.33	92.24	93.23	94.47	**94.80**

Table 5. The effect of the two modules on the model.

model	IMDb				DBLP			
	Macro-F1	Micro-F1	NMI	ARI	Macro-F1	Micro-F1	NMI	ARI
MAGNN	60.43	60.63	15.58	16.74	93.51	93.94	80.81	85.54
only applies GCN	60.85	61.05	**16.14**	17.81	93.59	94.06	80.86	86.16
MIED	**61.17**	**61.38**	15.84	**17.95**	**94.20**	**94.60**	**81.84**	**86.66**

that the model works relatively well in the middle part of the curves. Therefore, when selecting α we recommend selecting an α in $[1/2, 2/3]$ or performing a grid search on $(0, 1)$.

6. Conclusions and future work

In this paper, we propose MIED which contains two modules to enhance the node embeddings of heterogeneous graph. These two modules are: (1) using metapath-based GCN in the feature alignment to include graph information; (2) proposing EDE to distinguish the importance of different nodes when aggregating features on the metapath. Our comparative experiments demonstrate the effectiveness of our methods.

We also analyze the computational complexity of two proposed methods. They increase in computational complexity, but are controllable compared to their effects. Meta-path based methods have some scalability challenges when applied to large-scale graphs. But this can be effectively controlled by controlling the number and length of metapaths. After all, the value of information along very long metapaths is rapidly diminishing.

As heterogeneous graphs play an important role in fields such as multi-mode heterogeneous graph recommendation, we will apply our methods to graphs in these fields in the future, with the aim of further improving the performance of these models.

References

[1] Singh, R., Subramani, S., Du, J., Zhang, Y., Wang, H., Miao, Y. and Ahmed, K. (2023) Antisocial behavior identification from twitter feeds using traditional machine learning algorithms and deep learning. *EAI Endorsed Transactions on Scalable Information Systems* **10**(4): e17–e17.

[2] Zhou, Y., Lin, Z., La, Y., Huang, J. and Wang, X. (2022) Analysis and design of power system transformer standard based on knowledge graph. *EAI Endorsed Transactions on Scalable Information Systems* **10**(2).

[3] Kumar, S., Mallik, A., Khetarpal, A. and Panda, B. (2022) Influence maximization in social networks using graph embedding and graph neural network. *Information Sciences* **607**: 1617–1636.

[4] Liu, J., Xia, F., Feng, X., Ren, J. and Liu, H. (2022) Deep graph learning for anomalous citation detection. *IEEE Transactions on Neural Networks and Learning Systems* **33**(6): 2543–2557.

[5] Zhou, Y., Lin, Z., Tu, L., Huang, J. and Zhang, Z. (2022) Analysis and design of standard knowledge service system based on deep learning. *EAI Endorsed Transactions on Scalable Information Systems* **10**(2).

[6] Le, T., Le, N. and Le, B. (2023) Knowledge graph embedding by relational rotation and complex convolution for link prediction. *Expert Systems with Applications* **214**: 119122.

[7] Zhou, J., Liu, L., Wei, W. and Fan, J. (2022) Network representation learning: from preprocessing, feature extraction to node embedding. *ACM Computing Surveys (CSUR)* **55**(2): 1–35.

[8] Ou, M., Cui, P., Pei, J., Zhang, Z. and Zhu, W. (2016) Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*: 1105–1114.

[9] Cao, S., Lu, W. and Xu, Q. (2015) Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*: 891–900.

[10] Lee, D.D. and Seung, H.S. (1999) Learning the parts of objects by non-negative matrix factorization. *Nature* **401**(6755): 788–791.

[11] Perozzi, B., Al-Rfou, R. and Skiena, S. (2014) Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*: 701–710.

[12] Grover, A. and Leskovec, J. (2016) node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*: 855–864.

[13] Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L. *et al.* (2020) Graph neural networks: A review of methods and applications. *AI open* **1**: 57–81.

[14] Di Natale, R., Ferro, A., Giugno, R., Mongiovì, M., Pulvirenti, A. and Shasha, D. (2010) Sing: Subgraph search in non-homogeneous graphs. *BMC bioinformatics* **11**: 1–15.

[15] Hu, Z., Dong, Y., Wang, K. and Sun, Y. (2020) Heterogeneous graph transformer. In *Proceedings of the web conference 2020*: 2704–2710.

[16] Darban, Z.Z. and Valipour, M.H. (2022) Ghrs: Graph-based hybrid recommendation system with application to movie recommendation. *Expert Systems with Applications* **200**: 116850.

[17] Fu, X., Zhang, J., Meng, Z. and King, I. (2020) Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of The Web Conference 2020*: 2331–2341.

[18] Oghina, A., Breuss, M., Tsagkias, M. and De Rijke, M. (2012) Predicting imdb movie ratings using social media. In *ECIR* (Springer): 503–507.

[19] Deng, H., King, I. and Lyu, M.R. (2008) Formal models for expert finding on dblp bibliography data. In *2008 Eighth IEEE International Conference on Data Mining* (IEEE): 163–172.

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

[21] Chen, D., Lin, Y., Li, W., Li, P., Zhou, J. and Sun, X. (2020) Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI conference on artificial intelligence*, **34**: 3438–3445.

[22] Chen, M., Wei, Z., Huang, Z., Ding, B. and Li, Y. (2020) Simple and deep graph convolutional networks. In *International conference on machine learning* (PMLR): 1725–1735.

[23] Sarki, R., Ahmed, K., Wang, H., Zhang, Y. and Wang, K. (2022) Convolutional neural network for multi-class classification of diabetic eye disease. *EAI Endorsed Transactions on Scalable Information Systems* **9**(4): e5–e5.

[24] Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T. and Weinberger, K. (2019) Simplifying graph convolutional networks. In *International conference on machine learning* (PMLR): 6861–6871.

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

[26] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P. and Bengio, Y. (2017) Graph attention networks. *arXiv preprint arXiv:1710.10903* .

[27] Brody, S., Alon, U. and Yahav, E. (2021) How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491* .

[28] Wang, X., Bo, D., Shi, C., Fan, S., Ye, Y. and Philip, S.Y. (2022) A survey on heterogeneous graph embedding: methods, techniques, applications and sources. *IEEE Transactions on Big Data* .

[29] Yin, J., Tang, M., Cao, J., Wang, H., You, M. and Lin, Y. (2022) Vulnerability exploitation time prediction: an integrated framework for dynamic imbalanced learning. *World Wide Web* : 1–23.

[30] Pandey, D., Wang, H., Yin, X., Wang, K., Zhang, Y. and Shen, J. (2022) Automatic breast lesion segmentation in phase preserved dce-mris. *Health Information Science and Systems* **10**(1): 9.

[31] Dong, Y., Chawla, N.V. and Swami, A. (2017) metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*: 135–144.

[32] Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P. and Yu, P.S. (2019) Heterogeneous graph attention network. In *The world wide web conference*: 2022–2032.

[33] Linmei, H., Yang, T., Shi, C., Ji, H. and Li, X. (2019) Heterogeneous graph attention networks for semi-supervised short text classification. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*: 4821–4830.

[34] Bi, S., Li, Z., Brown, M., Wang, L. and Xu, Y. (2022) Dynamic weighted and heat-map integrated scalable information path-planning algorithm. *EAI Endorsed Transactions on Scalable Information Systems* **10**(2).

[35] Shang, J., Qu, M., Liu, J., Kaplan, L.M., Han, J. and Peng, J. (2016) Meta-path guided embedding for similarity search in large-scale heterogeneous information networks. *arXiv preprint arXiv:1610.09769* .

[36] Unknown (2022) A knowledge graph empowered online learning framework for access control decision-making. *World Wide Web* : 1–22URL https://link.springer.com/article/10.1007/s11280-022-01076-5.

[37] Pang, X., Ge, Y.F., Wang, K., Traina, A.J. and Wang, H. (2023) Patient assignment optimization in cloud healthcare systems: a distributed genetic algorithm. *Health Information Science and Systems* **11**(1): 30.

[38] Unknown (2022) Smile: Schema-augmented multi-level contrastive learning for knowledge graph link prediction. In *EMNLP (Findings)*: 4165–4177. URL https://arxiv.org/abs/2210.04870.

[39] Sun, Z., Deng, Z.H., Nie, J.Y. and Tang, J. (2019) Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197* .

[40] Cheadle, C., Vawter, M.P., Freed, W.J. and Becker, K.G. (2003) Analysis of microarray data using z score transformation. *The Journal of molecular diagnostics* **5**(2): 73–81.

[41] Ley, M. (2002) The dblp computer science bibliography: Evolution, research issues, perspectives. In *String Processing and Information Retrieval: 9th International Symposium, SPIRE 2002 Lisbon, Portugal, September 11–13, 2002 Proceedings 9* (Springer): 1–10.

[42] Hamerly, G. and Elkan, C. (2003) Learning the k in k-means. *Advances in neural information processing systems* **16**.

[43] Estévez, P.A., Tesmer, M., Perez, C.A. and Zurada, J.M. (2009) Normalized mutual information feature selection. *IEEE Transactions on neural networks* **20**(2): 189–201.

[44] Steinley, D. (2004) Properties of the hubert-arable adjusted rand index. *Psychological methods* **9**(3): 386.

[45] Hearst, M.A., Dumais, S.T., Osuna, E., Platt, J. and Scholkopf, B. (1998) Support vector machines. *IEEE Intelligent Systems and their applications* **13**(4): 18–28.