Super-Graph Classification

Ting Guo[1] and Xingquan Zhu[2]

[1] Centre for Quantum Computation & Intelligent Systems, FEIT, University of Technology, Sydney, NSW 2007, Australia
[2] Dept. of Computer & Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL 33431, USA
ting.guo-1@student.uts.edu.au, xzhu3@fau.edu

Abstract. Graphs are popularly used to represent objects with dependency structures, yet all existing graph classification algorithms can only handle simple graphs where each node is a single attribute (or a set of independent attributes). In this paper, we formulate a new super-graph classification task where each node of the super-graph may contain a graph (a single-attribute graph), so a super-graph contains a set of inter-connected graphs. To support super-graph classification, we propose a *Weighted Random Walk Kernel* (WRWK) which generates a product graph between any two super-graphs, and uses the similarity (kernel value) of two single-attribute graph as the node weight. Then we calculate weighted random walks on the product graph to generate kernel value between two super-graphs as their similarity. Our method enjoys sound theoretical properties, including bounded similarity. Experiments confirm that our method significantly outperforms baseline approaches.

Keywords: Graph classification, kernel, super-graph.

1 Introduction

Many applications, such as social networks and citation networks, commonly use graph structure to represent data entries (*i.e.* nodes) and their structural relationships (*i.e.* edges). When using graphs to represent objects, all existing frameworks rely on two approaches to describe node content (1) **node as a single attribute:** each node has only one attribute (single-attribute node). A clear drawback of this representation is that a single attribute cannot precisely describe the node content [6]. This representation is commonly referred to as a *single-attribute graph* (Fig.1 (A)). (2) **node as a set of attributes:** use a set of independent attributes to describe the node content (Fig.1 (B)). This representation is commonly referred to as an *attributed graph* [2,3,8].

Indeed, in many applications, the attributes/properties used to describe the node content may be subject to dependency structures. For example, in a citation network each node represents one paper and edges denote citation relationships. It is insufficient to use one or multiple independent attributes to describe detailed information of a paper. Instead, we can represent the content of each paper as a graph with nodes denoting keywords and edges representing contextual correlations between keywords (*e.g.* co-occurrence of keywords in different sentences

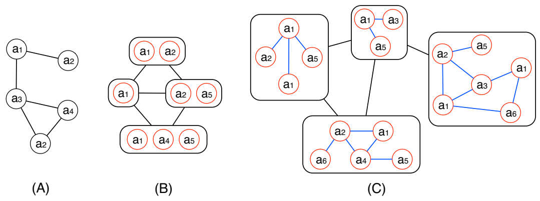


Fig. 1. (A): a single-attribute graph; (B): an attributed graph; and (C): a super-graph

or paragraphs). As a result, each paper and all references cited in this paper can form a super-graph with each edge between papers denoting their citation relationships. In this paper, we refer to this type of graph, where the content of the node can be represented as a graph, as a "*super-graph*". Likewise, we refer to the node whose content is represented as a graph, as a "*super-node*".

To build learning models for super-graphs, the mainly challenge is to properly calculate the distance between two super-graphs.

• **Similarity between two super-nodes:** Because each super-node is a graph, the overlapped/intersected graph structure between two super-nodes reveals the similarity between two super-nodes, as well as the relationship between two super-graphs. Traditional hard-node-matching mechanism is unsuitable for super-graphs which require soft-node-matching.

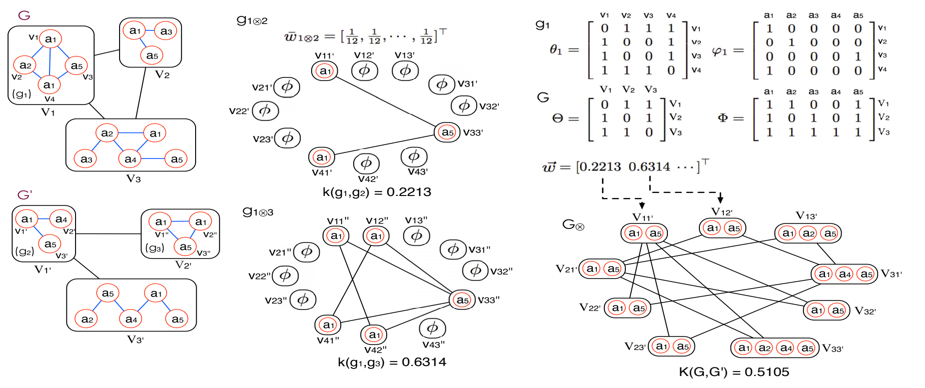
• **Similarity between two super-graphs:** The complex structure of super-graph requires that the similarity measure considers not only the structure similarity, but also the super-node similarity between two super-graphs. This cannot be achieved without combining node matching and graph matching as a whole to assess similarity between super-graphs.

The above challenges motivate the proposed Weighted Random Walk Kernel ($WRWK$) for super-graphs. In our paper, we generate a new product graph from two super-graphs and then use weighted random walks on the product graph to calculate similarity between super-graphs. A weighted random walk denotes a walk starting from a random weighted node and following succeeding weighted nodes and edges in a random manner. The weight of the node in the product graph denotes the similarity of two super-nodes. Given a set of labeled super-graphs, we can use an weighted product graph to establish walk-based relationship between two super-graphs and calculate their similarities. After that, we can obtain the kernel matrix for super-graph classification.

2 Problem Definition

Definition 1. ***(Single-attribute Graph)*** *A single-attribute graph is represented as* $g = (V, E, Att, f)$*, where* $V = \{v_1, v_2, \cdots, v_n\}$ *is a finite set of vertices,* $E \subseteq V \times V$ *denotes a finite set of edges, and* $f : V \rightarrow Att$ *is an injective function from the vertex set* V *to the attribute set* $Att = \{a_1, a_2, \cdots, a_m\}$.

Definition 2. ***(Super-graph and Super-node)*** *A super-graph is represented as* $G = (\mathcal{V}, \mathcal{E}, \mathcal{G}, \mathcal{F})$*, where* $\mathcal{V} = \{V_1, V_2, \cdots, V_N\}$ *is a finite set of graph-structured*


G
g1⊗2
$\bar{w}_{1\otimes 2} = [\frac{1}{12}, \frac{1}{12}, \cdots, \frac{1}{12}]^\top$
k(g1,g2) = 0.2213
g1⊗3
k(g1,g3) = 0.6314
G'
$\boldsymbol{w}$ = [0.2213 0.6314 ⋯]ᵀ
G⊗
K(G,G') = 0.5105

$$\theta_1 = \begin{array}{c|cccc|c} & v_1 & v_2 & v_3 & v_4 & \\ \hline & 0 & 1 & 1 & 1 & v_1 \\ & 1 & 0 & 0 & 1 & v_2 \\ & 1 & 0 & 0 & 1 & v_3 \\ & 1 & 1 & 1 & 0 & v_4 \end{array} \quad \varphi_1 = \begin{array}{ccccc|c} a_1 & a_2 & a_3 & a_4 & a_5 & \\ \hline 1 & 0 & 0 & 0 & 0 & v_1 \\ 0 & 1 & 0 & 0 & 0 & v_2 \\ 0 & 0 & 0 & 0 & 1 & v_3 \\ 1 & 0 & 0 & 0 & 0 & v_4 \end{array}$$

$$\Theta = \begin{array}{ccc|c} V_1 & V_2 & V_3 & \\ \hline 0 & 1 & 1 & V_1 \\ 1 & 0 & 1 & V_2 \\ 1 & 1 & 0 & V_3 \end{array} \quad \Phi = \begin{array}{ccccc|c} a_1 & a_2 & a_3 & a_4 & a_5 & \\ \hline 1 & 1 & 0 & 0 & 1 & V_1 \\ 1 & 0 & 1 & 0 & 1 & V_2 \\ 1 & 1 & 1 & 1 & 1 & V_3 \end{array}$$

Fig. 2. $WRWK$ on the super-graphs (G, G') and the single-attribute graphs (g_1, g_2, g_3), where $g_{1\otimes 2}$ and $g_{1\otimes 3}$ are the single-attribute product graphs for (g_1, g_2) and (g_1, g_3), respectively, and $G_\otimes$ is the super product graph for (G, G'). (θ_1, φ_1) and (Θ, Φ) are the adjacency and attribute matrices for g_1 and G, respectively. $\bar{w}_{1\otimes 2}$ and $\boldsymbol{w}$ are the weight vectors for $g_{1\otimes 2}$ and $G_\otimes$, respectively. Each element of $\boldsymbol{w}$ is equal to the $WRWK$ of two super-nodes (as dotted arrows show). A random walk on single-attribute product graph, say $g_{1\otimes 3}$, is equivalent to performing simultaneous random walks on graphs g_1 and g_3, respectively. Assume that $v_1(a_1)$ means node v_1 with attribute a_1 in g_1. The walk $v_1(a_1) \to v_3(a_5) \to v_4(a_1)$ in g_1 can match the walk $v_{1''}(a_1) \to v_{3''}(a_5) \to v_{2''}(a_1)$ in g_3. The corresponding walk on the single-attribute product graph $g_{1\otimes 3}$ is: $v_{11''}(a_1) \to v_{33''}(a_5) \to v_{42''}(a_1)$. $k(g_1, g_2)$ and $k(g_1, g_3)$ are the kernels. $k(g_1, g_2) < k(g_1, g_3)$ means g_3 is more similar to g_1 than g_2. Likewise, $K(G, G')$ is the $WRWK$ of G and G'.

nodes. $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes a finite set of edges, and $\mathcal{F} : \mathcal{E} \to \mathcal{G}$ is an injective function from $\mathcal{E}$ to $\mathcal{G}$, where $\mathcal{G} = \{g_1, g_2, \cdots, g_M\}$ is the set of single-attribute graphs. A node in the super-graph, which is represented by a single-attribute graph, is called a Super-node.

Formally, a single-attribute graph $g = (V, E, Att, f)$ can be uniquely described by its attribute and adjacency matrices. The attribute matrix φ is defined by $\varphi_{ri} = 1 \Leftrightarrow a_i \in f(v_r)$, otherwise $\varphi_{ri} = 0$. The adjacency matrix θ is defined by $\theta_{ij} = 1 \Leftrightarrow (v_i, v_j) \in E$, otherwise $\theta_{ij} = 0$. Similarly, the adjacency matrix Θ of an super-graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{G}, \mathcal{F})$ is defined by $\Theta_{ij} = 1 \Leftrightarrow (V_i, V_j) \in \mathcal{E}$, otherwise $\Theta_{ij} = 0$. However, because of the complicated structure of super-graph, we cannot use an unique matrix to describe its super-node information. To calculate conveniently as this article shows later, an super attribute matrix $\Phi \in \mathbb{R}^{N\times S}$ is defined for super-graph G, where $N = |\mathcal{V}|$ and $S = |Att_{g_1} \cup Att_{g_2} \cup \cdots \cup Att_{g_M}|$. $\Phi_{ri} = 1 \Leftrightarrow a_i \in Att_{g_r}$, otherwise $\Phi_{ri} = 0$. Examples of the attribute and adjacency matrices for a single-attribute graph and a super-graph are shown in the top right corner of Fig. 2.

A super-graph G_i is a labeled graph if a class label $L(G_i) \in \mathcal{Y} = \{y_1, y_2, \cdots, y_x\}$ is assigned to G_i. A super-graph G_i is either labeled (G_i^L) or unlabeled (G_i^U).

Definition 3. ***(Super-graph Classification)*** *Given a set of labeled super-graphs $\mathcal{D}^L = \{G_1^L, G_2^L, \cdots\}$, the goal of super-graph classification is to learn* *a discriminative model from $\mathcal{D}^L$ to predict some previously unseen super-graphs $\mathcal{D}^U = \{G_1^U, G_2^U, \cdots\}$ with maximum accuracy.*

3 Weighted Random Walk Kernel

Defining a kernel on a space $\mathcal{X}$ paves the way for using kernel methods [7] for classification, regression, and clustering. To define a kernel function for super-graphs, we employ a random walk kernel principle [4].

Our Weighted Random Walk Kernel ($WRWK$) is based on a simple idea: Given a pair of super-graphs (G_1 and G_2), we can use them to build a product graph, where each node of the product graph contains attributes shared by super-nodes in G_1 and G_2. For each node in the product graph, we can assign a weight value (which is based on the similarity of the two super-nodes generating the current node). Because a weighted product node means the common weight of the node appeared in both G_1 and G_2, we can perform random walks through these nodes to measure the similarity by counting the number of matching walks (the walks through nodes containing intersected attribute sets) and combining weights of the nodes. The larger the number, the more similar the two super-graphs are. Adding weight value to the random walk is meaningful. It provides a solution to take graph matching of super-nodes into consideration to calculate the similarity between super-graphs. We consider the similarity between any two super-nodes as the weight value instead of just using 1/0 hard-matching. The node similarities between different super-nodes represent the relationship between two graphs in a more precise way, which, in turn, helps improve the classification accuracy. Accordingly, we divide our kernel design into two parts based on the similarity of super-nodes and super-graphs.

3.1 Kernel on Single-Attribute Graphs

We firstly introduce the weighted random walk kernel on single-attribute graphs.

Definition 4. ***(Single-Attribute Product Graph)*** *Given two single-attribute graphs $g_1 = (V_1, E_1, Att_1, f_1)$ and $g_2 = (V_2, E_2, Att_2, f_2)$, their single-attribute product graph is denoted by $g_{1\otimes 2} = (V^*, E^*, Att^*, f^*)$ ($g_\otimes$ for short), where*

- $V^* = \{v | v =< v_1, v_2 >, v_1 \in V_1, v_2 \in V_2\}$;
- $E^* = \{e | e = (u', v'), u' \in V^*, v' \in V^*, f^*(u') \neq \phi, f^*(v') \neq \phi,$
  $u' =< u_1, u_2 >, v' =< v_1, v_2 >, (u_1, v_1) \in E_1, (u_2, v_2) \in E_2\}$;
- $Att^* = Att_1 \cup Att_2$;
- $f^* = \{f^*(v) | f^*(v) = f(v_1) \cap f(v_2), v =< v_1, v_2 >, v_1 \in V_1, v_2 \in V_2\}$.

In other words, $g_\otimes$ is a single-attribute graph where a vertex v is the intersection between a pair of nodes in g_1 and g_2. There is an edge between a pair of vertices in $g_\otimes$, if and only if an edge exists in corresponding vertices in g_1 and g_2, respectively. An example is shown in Fig. 2 ($g_{1\otimes 2}$). In the following, we show that an inherent property of the product graph is that performing a weighted random walk on the product graph is equivalent to performing simultaneous random walks on g_1 and g_2, respectively. So the single-attribute product graph provides an effective way to count the number of walks combining weight values on nodes between graphs without expensive graph matching.

To generate $g_{\otimes}$'s adjacency matrix $\theta_{\otimes}$ from g_1 and g_2 by using matrix operations, we define the *Attributed Product* as follow:

Definition 5. ***(Attributed Product)*** *Given matrices $B \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{m\times m}$ and $H \in \mathbb{R}^{n'\times m'}$, the attributed product $B \boxtimes C \in \mathbb{R}^{nm\times nm}$ and the column-stacking operator $vec(H) \in \mathbb{R}^{n'm'}$ are defined as*

$$B \boxtimes C = [vec(B_{*1}C_{1*})\ vec(B_{*1}C_{2*})\ \cdots\ vec(B_{*n}C_{m*})],$$

$$vec(H) = [H_{*1}^{\top}\ H_{*2}^{\top}\ \cdots\ H_{*n}^{\top}]^{\top},$$

where H_{*i} and H_{j*} denote i^{th} column and j^{th} row of H, respectively.

Based on Def. 5, the adjacency matrix $\theta_{\otimes}$ of the single-attribute product graph $g_{\otimes}$ can be directly derived from $g_1(\theta_1, \varphi_1)$ and $g_2(\theta_2, \varphi_2)$ as follow:

$$\theta_{\otimes} = (\theta_1 \boxtimes \theta_2)\, \bar{\wedge}\, vec(\varphi_1 \varphi_2^{\top}) \tag{1}$$

where

$$B\, \bar{\wedge} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} B_{11} \wedge x_1 & \cdots & B_{1n} \wedge x_1 \\ \vdots & \ddots & \vdots \\ B_{n1} \wedge x_n & \cdots & B_{nn} \wedge x_n \end{bmatrix},$$

"$\wedge$" is a conjunction operation ($a \wedge b = 1$ *iff* $a = 1$ and $b = 1$).

To better assess the similarity between two single-attribute graphs, we assign each node a weight value, with weight indicating the importance of a node in the graph. So a weighted random walk can be calculated by multiply all the weight values of the nodes along the walk. Then we can calculate the weighted random walk counting by using matrix operation. More specifically, for the adjacency matrix θ_x of a graph g_x, each element $[\theta_x^z]_{ij}$ of this n^{th} power matrix gives the number of walks of length z from v_i to v_j in g_x. If we have the weight vector of g_x as $\bar{w}_x = [w_1, w_2, ...w_n]^{\top}$. Each element $[(\bar{w}_x\ \bar{w}_x^{\top} \odot \theta_x)^z]_{ij}$ corresponds to the total weight value of all random walks with length z from v_i to v_j in g_x, where "$\odot$" denotes element-wise multiplication (the same as ".*" operator in Matlab). For simplicity, we set an uniform distributions for all the statistical weights of nodes as $w_i = \frac{1}{n}$. So the weight vector $\bar{w}$ denote the weights over the vertices of $g_{\otimes}$ generated from g_1 and g_2 with the node sizes of n_1 and n_2, respectively, where

$$\bar{w} = [\frac{1}{n_1 n_2}, \frac{1}{n_1 n_2}, \cdots, \frac{1}{n_1 n_2}]^{\top} \in \mathbb{R}^{n_1 n_2}$$

According to Eq. (1), performing a weighted random walk on the single-attribute product graph $g_{\otimes}$ is equivalent to performing random walks on graphs g_1 and g_2 simultaneously. After $g_{\otimes}$ is generated, *Weighted Random Walk Kernel*

(WRWK), which computes the similarity between g_1 and g_2, can be defined with a sequence of weights $\delta = \delta_0, \delta_1, \cdots$ ($\delta_i \in \mathbb{R}$ and $\delta_i \geq 0$ for all $i \in \mathbb{N}$):

$$k(g_1, g_2) = \rho \sum_{i,j=1}^{n_1 n_2} \left[\sum_{z=0}^{\infty} \delta_z \ (\bar{w}\ \bar{w}^\top \odot \theta_\otimes)^z \right]_{ij} \tag{2}$$

where n_1 and n_2 are the node sizes of g_1 and g_2, respectively. ρ is the control parameter used to make the kernel function convergence. As a result, kernel values are upper bounded (the proof is given in Section 5). For simplicity, we set $\rho = \frac{1}{n_1 n_2}$. We call $\varGamma_\otimes = \bar{w}\ \bar{w}^\top \odot \theta_\otimes$ the weighted adjacency matrix of single-attribute product graph.

To compute the *WRWK* for single-attribute graph, as defined in Eq. (2), a diagonalization decomposition method [4] can be used. Because $\varGamma_\otimes$ is a symmetric matrix, the diagonalization decomposition of $\varGamma_\otimes$ exists: $\varGamma_\otimes = \mathcal{T}\mathcal{H}\mathcal{T}^{-1}$, where the columns of $\mathcal{T}$ are its eigenvectors, and $\mathcal{H}$ is a diagonal matrix of corresponding eigenvalues. The kernel defined in Eq. (2) can then be rewritten as:

$$k(g_1, g_2) = \rho \sum_{i,j=1}^{n_1 n_2} \left[\sum_{z=0}^{\infty} \delta_z \ (\mathcal{T}\mathcal{H}^z\mathcal{T}^{-1}) \right]_{ij} = \rho \sum_{i,j=1}^{n_1 n_2} \left[\mathcal{T}(\sum_{z=0}^{\infty} \delta_z \ \mathcal{H}^z)\ \mathcal{T}^{-1} \right]_{ij} \tag{3}$$

By setting $\delta_z = \lambda^z/z!$ in Eq. (3), and use $e^x = \sum_{z=0}^{\infty} x^z/z!$, we have,

$$k(g_1, g_2) = \rho \sum_{i,j=1}^{n_1 n_2} \left[\mathcal{T} e^{\lambda\mathcal{H}}\ \mathcal{T}^{-1} \right]_{ij} \tag{4}$$

The diagonalization decomposition can greatly expedite *WRWK* kernel computation. An example of *WRWK* kernel is shown in Fig. 2.

3.2 Kernel on Super-Graphs

The *WRWK* of single-attribute graph helps calculate the similarity between two graphs, so it can be used to calculate the similarity between two super-nodes. Given a pair of super-graphs G_1 and G_2, assume we can generate a new product graph $G_\otimes$ whose nodes are generated by super-nodes in G_1 and G_2, and weight value of each node is the similarity between the super-nodes which generate this node, then the same process shown in Section 3.1 can be used to calculate the *WRWK* for super-graphs G_1 and G_2 to denote their similarity.

Definition 6. ***(Super Product Graph)*** *Given two super-graphs* $G_1 = (\mathcal{V}_1, \mathcal{E}_1, \mathcal{G}_1, \mathcal{F}_1)$ *and* $G_2 = (\mathcal{V}_2, \mathcal{E}_2, \mathcal{G}_2, \mathcal{F}_2)$, *their Super Product Graph is denoted by* $G_{1\otimes 2} = (\mathcal{V}^*, \mathcal{E}^*, \mathcal{G}^*, \mathcal{F}^*)$ *(*$G_\otimes$ *for short), where*

- $\mathcal{V}^* = \{V | V =< V_1, V_2 >, V_1 \in \mathcal{V}_1, V_2 \in \mathcal{V}_2\}$;
- $\mathcal{E}^* = \{e | e = (V, V'), V \in \mathcal{V}^*, V' \in \mathcal{V}^*, \mathcal{F}^*(V) \neq \phi, \mathcal{F}^*(V') \neq \phi,$ $V =< V_1, V_2 >, V' =< V_1', V_2' >, (V_1, V_1') \in \mathcal{E}_1, (V_2, V_2') \in \mathcal{E}_2\}$;
- $\mathcal{G}^* = \mathcal{G}_1 \cup \mathcal{G}_2$;
- $\mathcal{F}^* = \{\mathcal{F}^*(V) | \mathcal{F}^*(V) = \mathcal{F}(V_1) \cap \mathcal{F}(V_2), V =< V_1, V_2 >, V_1 \in \mathcal{V}_1, V_2 \in \mathcal{V}_2\}$.

An example of super product graph is shown in Fig. 2 ($G_\otimes$). Similar to Eq. (1), the adjacency matrix $\Theta_\otimes$ of the super product graph $G_\otimes$ can be directly derived from $G_1(\Theta_1, \Phi_1)$ and $G_2(\Theta_2, \Phi_2)$ as follows:

$$\Theta_\otimes = (\Theta_1 \boxtimes \Theta_2) \barwedge vec(\Phi_1 \Phi_2^\top) \tag{5}$$

Because we use the similarity between two super-nodes as the weight value of the node in the super product graph, the kernel value may increase infinitely with the increasing size of the super-graph. So for super-graph kernel, we add a control variable η to limit the range of the super-graph kernel value. Then the Weighted Random Walk Kernel, which computes the similarity between G_1 and G_2, can be defined with a sequence of weights $\sigma = \sigma_0, \sigma_1, \cdots$ ($\sigma_i \in \mathbb{R}$ and $\sigma_i \geq 0$ for all $i \in \mathbb{N}$):

$$K(G_1, G_2) = \eta \sum_{i,j=1}^{N_1 N_2} \left[\sum_{z=0}^{\infty} \sigma_z \; (\boldsymbol{w}\;\boldsymbol{w}^\top \odot \Theta_\otimes)^z \right]_{ij} \tag{6}$$

where N_1 and N_2 are the node sizes of G_1 and G_2, respectively, and

$$\boldsymbol{w} = [k(g_1, g_1)\; k(g_1, g_2)\; \cdots\; k(g_{N_1}, g_{N_2})]^\top$$

Similar to WRWK of single-attribute graphs in Eq. (4), the WRWK on super-graphs can be calculated by setting $\sigma_z = \gamma^z / z!$, we have,

$$K(G_1, G_2) = \eta \sum_{i,j=1}^{N_1 N_2} \left[\widetilde{\mathcal{T}} e^{\gamma \widetilde{\mathcal{H}}}\; \widetilde{\mathcal{T}}^{-1} \right]_{ij} \tag{7}$$

where $\boldsymbol{w}\;\boldsymbol{w}^\top \odot \Theta_\otimes = \widetilde{\mathcal{T}} \widetilde{\mathcal{H}} \widetilde{\mathcal{T}}^{-1}$. To ensure kernel function convergence, we set

$$\eta = \frac{1}{N_1 N_2} e^{\frac{1 - N_1^2 N_2^2}{N_1 N_2} \gamma e^{2\lambda}} \tag{8}$$

where γ is the parameter of $WRWK$ on super-graphs, and λ is the parameter of $WRWK$ on single-attribute graphs which is given in previous sub-section. The $WRWK$ of super-graphs is also upper bounded with proof showing in Section 5.

4 Super-Graph Classification

The $WRWK$ provides an effective way to measure the similarity between super-graphs. Given a number of labeled super-graphs, we can use their pair-wise similarity to form an kernel matrix. Then generic classifiers, such as Support Vector Machine (SVM), Decision Tree (DT), Naive Bayes (NB) and Nearest Neighbour (NN), can be applied to the kernel matrix for super-graph classification.

Algorithm 1 shows the framework of using $WRWK$ to train a classifier.

5 Theoretical Study

Theorem 1. *The Weighted Random Walk Kernel function is positive definite.*

Algorithm 1. WRWK Classifier Generation

```
Input: Labeled super-graph set $\mathcal{D}^L$, Kernel parameters $\lambda$ and $\gamma$
Output: Classifier $\zeta$
Initialize: Kernel matrix $M \leftarrow [\,]$
 1: for any two super-graphs $G_a$ and $G_b$ in $\mathcal{D}^L$ do
 2:   $\boldsymbol{w} \leftarrow [\,]$
 3:   for any $g_x$ in $G_a$ and $g_y$ in $G_b$ do
 4:     $q \leftarrow (x-1)N_b + y$        // $q$ is the element index of $\boldsymbol{w}$
 5:     if $Att_x \cap Att_y = \phi$ then
 6:       $\boldsymbol{w}_q \leftarrow 0$
 7:     else
 8:       $\rho \leftarrow \frac{1}{n_x n_y}$, $\bar{w} \leftarrow [\frac{1}{n_x n_y}\ \frac{1}{n_x n_y}\ \cdots\ \frac{1}{n_x n_y}]$, $\theta_\otimes \leftarrow (\theta_x \boxtimes \theta_y) \barwedge vec(\varphi_x \varphi_y^\top)$
 9:       $\boldsymbol{w}_q \leftarrow k(g_x, g_y)$        // $k(g_x, g_y)$ is calculated by eq. (4)
10:     end if
11:   end for
12:   $\eta \leftarrow \frac{1}{N_a N_b} e^{\frac{1-N_a^2 N_b^2}{N_a N_b}\gamma e^{2\lambda}}$, $\Theta_\otimes \leftarrow (\Theta_a \boxtimes \Theta_b) \barwedge vec(\Phi_a \Phi_b^\top)$
13:   $M_{ab} \leftarrow K(G_a, G_b)$        // $K(G_a, G_b)$ is calculated by eq. (7)
14: end for
15: $\zeta \leftarrow TrainClassifier(\mathcal{C}, M, \bar{L})$
    /* $\mathcal{C}$ is the learning algorithm. $\bar{L}$ is the class label vector of $\mathcal{D}^L$ */
```

Proof. As the random walk-based kernel is closed under products [4] and the *WRWK* can be written as the limit of a polynomial series with positive coefficients (as Eqs. (4) and (7)), the *WRWK* function is positive definite.

Theorem 2. *Given any two single-attribute graphs g_1 and g_2, the weighted random walk kernel of these two graphs is bounded by $0 < k(g_1, g_2) < e^\lambda$, where λ is the parameter of the weighted random walk kernel.*

Proof. Because $k(g_1, g_2)$ is a positive definite kernel by Theorem 1, so $k(g_1, g_2) > 0$. Then we only need to show that the upper bound of $k(g_1, g_2)$ is e^λ.

Based on the definition of WRWK, assume the node sizes of two single-attribute graphs g_1 and g_2 are n_1 and n_2, respectively, the number of random walks on $g_1 \otimes g_2$ must be not greater than that of the complete connect graph g^c which has $n_1 \times n_2$ nodes. We assume that $g^c = g_1' \otimes g_2'$, where g_1' and g_2' are with n_1 and n_2 nodes respectively. So we have $k(g_1, g_2) < k(g_1', g_2')$. More specifically,

$$k(g_1', g_2') = \rho \sum_{i,j=1}^{n_1 n_2} \left[\sum_{z=0}^{\infty} \delta_z \ (\bar{w}' \ \bar{w}'^\top \odot \theta_\otimes')^z \right]_{ij} = \rho \sum_{z=0}^{\infty} \{ \delta_z \sum_{i,j=1}^{n_1 n_2} \left[(\bar{w}' \ \bar{w}'^\top \odot \theta_\otimes')^z \right]_{ij} \}$$

Because $\rho = \frac{1}{n_1 n_2}$, $\bar{w}' = [\frac{1}{n_1 n_2}, \frac{1}{n_1 n_2}, \cdots, \frac{1}{n_1 n_2}]^\top$, and g^c is a complete connect graph, where the diagonal elements are equal to 0 and other element in the adjacency matrix are equal to 1.

So
$$\sum_{i,j=1}^{n_1 n_2}\left[(\bar{w}'\ \bar{w}'^{\top}\odot\theta'_{\otimes})^z\right]_{ij}=(\frac{n_1n_2-1}{n_1^2n_2^2})^{(z-1)}(1-\frac{1}{n_1n_2})$$

Then
$$k(g'_1,g'_2)=\frac{1}{n_1n_2}\sum_{z=0}^{\infty}[\delta_z(\frac{n_1n_2-1}{n_1^2n_2^2})^{(z-1)}(1-\frac{1}{n_1n_2})]$$
$$=\frac{1}{n_1n_2}(1-\frac{1}{n_1n_2})\sum_{z=0}^{\infty}[\delta_z(\frac{n_1n_2-1}{n_1^2n_2^2})^{(z-1)}]$$

Because $\delta_z=\frac{\lambda^z}{z!}$, we have
$$k(g'_1,g'_2)=\frac{1}{n_1n_2}(1-\frac{1}{n_1n_2})\sum_{z=0}^{\infty}[\frac{\lambda^z}{z!}(\frac{n_1n_2-1}{n_1^2n_2^2})^{(z-1)}]$$
$$=\frac{1}{n_1n_2}(1-\frac{1}{n_1n_2})(\frac{n_1^2n_2^2}{n_1n_2-1})\sum_{z=0}^{\infty}[\frac{\lambda^z}{z!}(\frac{n_1n_2-1}{n_1^2n_2^2})^z]$$

Because $e^x=\sum_{z=0}^{\infty}x^z/z!$, we have
$$k(g_1,g_2)<k(g'_1,g'_2)=e^{\frac{\lambda(n_1n_2-1)}{n_1^2n_2^2}}<e^{\lambda}$$

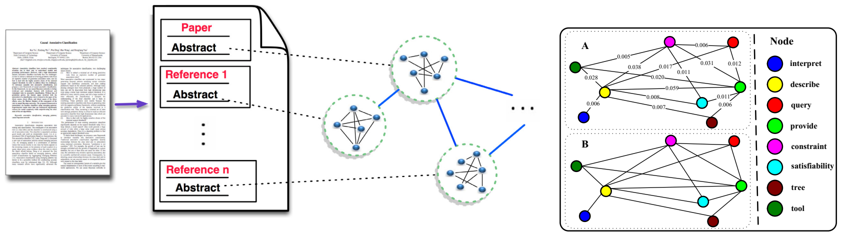
Theorem 3. *The weighted random walk kernel between super-graphs G_1 and G_2 is bounded by $0<K(G_1,G_2)<e^{\gamma e^{2\lambda}-2\lambda}$, where λ and γ are weighted random walk kernel parameters for single-attribute graph and super-graph, respectively.*

Similar to Theorem 2, Theorem 3 can be derived.

6 Experiments and Analysis

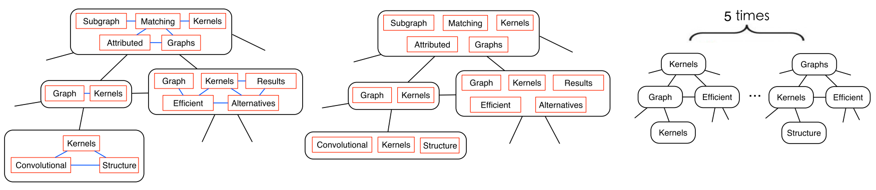
6.1 Benchmark Data

DBLP Dataset: DBLP dataset consists of bibliography data in computer science (http://arnetminer.org/citation/). Each record in DBLP is a scientific publication with a number of attributes such as abstract, authors, year, venue, title, and references. To build super-graphs, we select papers published in Artificial Intelligence (AI: IJCAI, AAAI, NIPS, UAI, COLT, ACL, KR, ICML, ECML and IJCNN) and Computer Vision (CV: ICCV, CVPR, ECCV, ICPR, ICIP, ACM Multimedia and ICME) fields to form a classification task. The goal is to predict which field (AI or CV) a paper (*i.e.* a super-graph) belongs to by using the abstract of each paper (*i.e.* a super-node) and abstracts of references (*i.e.* other super-nodes), as shown in Fig. 3. An edge (undirected) between two super-nodes indicates a citation relationship between two papers. For each paper, we use fuzzy cognitive map (E-FCM) [5] to convert paper abstract into a graph (which represents relations between keywords with weights over a threshold (*i.e.* edge-cutting threshold)). This graph representation has shown better performance than simple bag-of-words representation [1]. We select 1000 papers



(a) (b)

Fig. 3. An example of using super-graph representation for scientific publications. The left figure shows that each paper cites a number of references. For each paper (or each reference), its abstract can be converted as a graph. So each paper denotes a super-node, and the citation relationships between papers form a super-graph. The figure to the right shows a graph representation of paper abstract with each node denoting a keyword. A: The weight values between nodes indicate correlations between keywords. B: by using proper threshold, we can convert each abstract as an undirected unweighted graph.

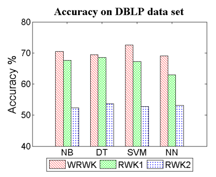


(A) (B) (C)

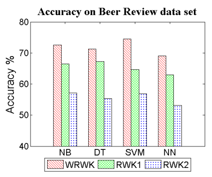
Fig. 4. Super-graph and comparison graph representations. A super-graph (A) can be represented as an attributed graph by discarding edges in each super-super as shown in (B). We also simplify a super-graph as a set of single-attribute graphs by retaining only one attribute in each super-node, as shown in (C).

(500 in each class), each of which contains 1 to 10 references, to form 1000 super-graphs.

Beer Review Dataset: The online beer review dataset consists of review data for beers (http://beeradvocate.com/). Each review in the dataset is associated with some attributes such as appearance score, aroma score, palate score, and taste score (rating of the product varies from 1 to 5), and detailed review texts. Our goal is to classify each beer to the right style (*Ale* vs. *Not Ale*) by using customer reviews. The graph representation for reviews is similar to the sub-graphs in DBLP dataset. Each review is represented as a super-node. The edge between super-nodes are built using following method: Because a review has four rating scores in appearance, aroma, palate, and taste, we use these four scores as a feature vector for each review. If two reviews's distance in the feature space (Euclidean distance) is less than 2, an edge is used to link two reviews (*i.e* super-nodes). We choose 1000 beer products, half from Ale and the rest is from Large and Hybrid Style, to form 1000 super-graphs for classification.



(a)



(b)

Fig. 5. Classification accuracy on DBLP and Beer review datasets *w.r.t.* different classification methods (NB, DT, SVM, and NN). For DBLP dataset, the number of super-nodes in each super-graph varies within 1-10 and the edge-cutting threshold for single-attribute graph is 0.001. For Beer Review dataset, the number of super-nodes in each super-graph varies within 30-60 and the edge-cutting threshold is 0.001. $WRWK$ uses super-graph representation (Fig.4(A)), $RWK1$ and $RWK2$ use traditional random walk kernel method with attributed graph representation (Fig. 4(B)) and single-attribute graph representation (Fig. 4(C)), respectively.

6.2 Experimental Settings

Baseline Methods: Because no existing method can handle super-graph classification, for comparison purposes, we use two approaches to generate traditional graph representations from each super-graph: (1) randomly selecting one attribute (*i.e* one word) in each super-node (and ignoring all other words) to form a single-attribute graph, as shown in Fig. 4 (C). We repeat random selection for five times with each time generating a set of single-attribute graphs from super-graphs. Each single-attribute graph set is used to train a classifier and their majority *voted accuracy* on test super-graphs is reported in the experiments. (2) We use the attribute set of the single-attribute graph in each super-node as multi-attributes of the super-node to generate an attributed graph (which is equal to removing all the edges from the super-graph as shown in Fig. 4 (B)). For all the two graph representations, we use the traditional random walk kernel (RWK) to measure the similarity between any two graphs [4].

We use 10 times 10-fold cross-validation classification accuracy to measure and compare the algorithm performance. To train classifiers from graph data, we use Naive Bayes (NB), Decision Tree (DT), Support Vector Machines (SVM) and Nearest Neighbor algorithm (NN). Majority examples are based on the kernel parameter settings: $\lambda = 100$ and $\gamma = 100$.

6.3 Results and Analysis

Performance on Standard Benchmarks: In Fig. 5, we report the classification accuracy on the benchmark datasets. The experimental results show that $WRWK$ constantly outperforms traditional RWK method, regardless of the type of graph representations used by RWK. This is mainly because in traditional graph representations each node only has one attribute or a set of independent attributes, whereas single-attribute nodes (or multiple independent attributes) cannot precisely describe the node content. For super-graphs, the

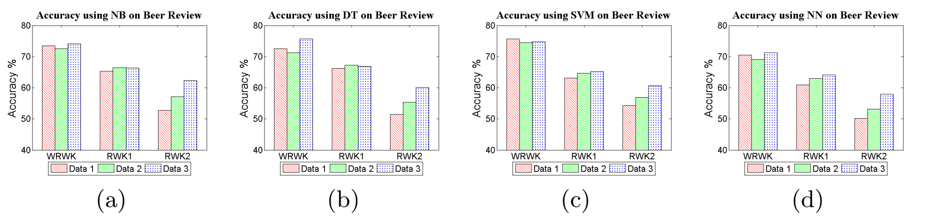


Fig. 6. Classification accuracy on Beer review dataset *w.r.t.* different datasets and classification methods (NB, DT, SVM, and NN). Figures correspond to three types of super-graph structure on Beer review. *Data 1*: the number of super-nodes in each super-graph is 2-30 and the edge-cutting threshold is 0.00001. *Data 2*: the number of super-nodes in each super-graph is 30-60 and the edge-cutting threshold is 0.001. *Data 3*: the number of super-nodes in each super-graph is > 60 and the threshold is 0.1.

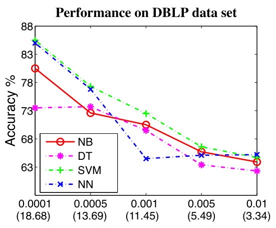
graph associated to each super-node provides an effective way to describe the node content. In $WRWK$ method, we consider the similarity between any two super-nodes as the weight value instead of just using 1/0 hard matching to represent whether there is an intersection of attribute sets between two nodes. The soft matching node similarities between super-nodes captures the relationship between two graphs in a more precise way. This, in turn, helps improve the classification accuracy.

Performance under different Super-Graph Structures: To demonstrate the performance of our $WRWK$ method on super-graphs with different characteristics, we construct super-graphs on Beer review dataset by using different super-node sizes and different structures of single-attribute graph (the structure of the single-attribute node is controlled by the edge-cutting threshold) as shown in Fig. 6 (a)-(d). The result shows that our $WRWK$ method is stable on super-graphs with different structures.

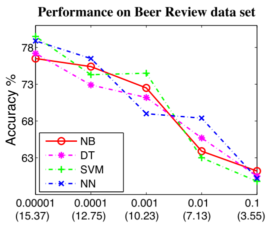
Performance *w.r.t.* Changes in Super-Nodes and Walks: The proposed weighted random walk kernel relies on the similarity between super-nodes and the common walks between two super-graphs to calculate the graph similarity. This raises a concern on whether super-node similarity or walk similarity (or both) plays a more important role in assessing the super-graph similarity.

In order to resolve this concern, we design the following experiments. In the first set of experiments (Fig. 7 (a) and (b)), we fix super-graph edges, and change edges inside super-nodes, which will impact on the super-node similarities. If this results in significant changes, it means that super-node similarity plays a more important role. In the second set of experiments (Fig. 7 (c)), we fix super-nodes but vary the edges in super-graphs (by randomly removing edges), which will impact on the common walks between super-graphs. If this results in significant changes, it means that walk plays a more important role than super-nodes.

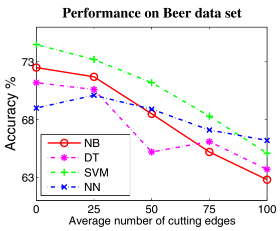
Fig. 7 (a) and (b) report the algorithm performance with respect to the edge-cutting threshold. The accuracy decreases dramatically with the increase of the edge-cutting threshold for both datasets. When the edge-cutting threshold is set to 0.0001 on DBLP and 0.00001 on Beer Review, the single-attribute graph



(a)



(b)



(c)

Fig. 7. The performance *w.r.t.* different edge-cutting thresholds on DBLP and Beer Review datasets by using $WRWK$ method. In (a) and (b), the node size of super-graph varies from 0-10 on DBLP dataset and 30-60 on Beer Review dataset. The x-axis shows the value of the edge-cutting threshold and the average node degree in its corresponding single-attribute graph is reported in the parentheses. In (c), the average number of edges cut from each super-graph varies from 0 to 100.

in each super-node is almost a complete graph and four methods achieve the highest classification accuracy. As the threshold is set to 0.01 on DBLP and 0.1 on Beer review, the single-attribute graph in each super-node is very small and contains very few edges. As a result, the accuracies are just around 65%. This demonstrates that $WRWK$ heavily relies on the structure information of each super-node to assess the super-graph similarities.

Fig. 7 (c) reports the algorithm performance by fixing the super-nodes but gradually removing edges in the super-graph of Beer review dataset (as Fig. 5 (b)). Similar to the result in Fig. 7 (a) and (b), the classification accuracy decreases when edges are continuously removed from the super-graph (even if the super-nodes are fixed). From Figs. 7, we find that graph structure of super-graph is as important as that of super-nodes. This is mainly attributed to the fact that our weighted random walk kernel relies on both super-node similarity and walks in the super-graph to calculate graph similarities.

7 Conclusion

In this paper, we formulated a new super-graph classification problem. Due to the inherent complex structure representation, all existing graph classification methods cannot be applied for super-graph classification. In the paper, we proposed a weighted random walk kernel which calculates the similarity between two super-graphs by assessing (a) the similarity between super-nodes of the super-graphs, and (b) the common walks of the super-graphs. Our key contribution is twofold: (1) a weighted random walk kernel considering node and structure similarities between graphs; and (2) an effective kernel-based super-graph classification method with sound theoretical basis.

References

1. Angelova, R., Weikum, G.: Graph-based text classification: learn from your neighbors. In: ACM SIGIR, pp. 485–492 (2006)
2. Cai, Y., Cercone, N., Han, J.: An attribute-oriented approach for learning classification rules from relational databases. In: ICDE, pp. 281–288 (1990)
3. Cheng, H., Zhou, Y., Yu, J.X.: Clustering large attributed graphs: A balance between structural and attribute similarities. ACM TKDD 5(2) (2011)
4. Gärtner, T., Flach, P.A., Wrobel, S.: On graph kernels: Hardness results and efficient alternatives. In: COLT, pp. 129–143 (2003)
5. Luo, X., Xu, Z., Yu, J.: Building association link network for semantic link on web resources. IEEE Trans. Autom. Sci. Eng. 8(3), 482–494 (2011)
6. Riesen, K., Bunke, H.: Graph classification and clustering based on vector space embedding. World Scientific Publishing Co., Inc. (2010)
7. Schölkopf, B., Smola, A.J.: Learning with kernels. MIT Press (2002)
8. Xu, Z., Ke, Y., Wang, Y., Cheng, H., Cheng, J.: A model-based approach to attributed graph clustering. In: ACM SIGMOD, pp. 505–516 (2012)