Relational Similarity Machines (RSM): A Similarity-based Learning Framework for Graphs

Ryan A. Rossi
Adobe Research
rrossi@adobe.com

Rong Zhou
Google
rongzhou@google.com

Nesreen K. Ahmed
Intel Labs
nesreen.k.ahmed@intel.com

Hoda Eldardiry
Palo Alto Research Center
hoda.eldardiry@parc.com

Abstract—**Relational machine learning has become increasingly important due to the recent proliferation and ubiquity of network data. However, existing methods are not designed for interactive learning and have many unrealistic assumptions that greatly limit their utility in practice. For instance, most existing work has focused on graphs with high relational autocorrelation (homophily) and perform poorly otherwise. To overcome these limitations, this paper presents a similarity-based relational learning framework called *Relational Similarity Machines* (RSM) for networks with arbitrary relational autocorrelation. The RSM framework is designed to be fast, accurate, and flexible for learning on a wide variety of networks. The experiments demonstrate the effectiveness of the RSM framework.**

Keywords-**Semi-supervised learning (SSL), statistical relational learning, heterophily, collective classification, similarity-based graph learning, interactive relational learning**

I. Introduction

Networks (relational data, graphs) encode dependencies between entities (people, computers, proteins) and allow us to study phenomena across social [1], technological [2], and biological domains [3]. Recently, relational machine learning (RML) methods were developed to leverage relational dependencies [4], [5], [6] between nodes to improve predictive performance [7], [8], [9], [10], [5], [11].

Relational classifiers can sometimes outperform traditional *iid* ML techniques by exploiting dependencies between class labels (attributes) of related nodes. However, the performance of RML methods can degrade when there are few labeled instances (majority of neighboring instances are also unlabeled). Collective Classification (CC) aims to solve this problem by iteratively predicting labels and propagating them to related instances [12]. Unfortunately, the performance of CC methods may also degrade when there are very few labels available, *e.g.*, label density < 0.01 [9]. In both situations, if not careful, the performance of RML methods may degrade to a point where *iid* techniques perform better.

Despite the fundamental importance of these techniques, the vast majority of RML methods rely on a significant amount of relational autocorrelation, *i.e.*, homophily [13] existing in the data. It has been noted that RML techniques may perform worse than *iid* methods when there is low or even modest relational autocorrelation (Figure 1). One particular case that arises quite often in practice is shown in Figure 1. For instance, molecular, chemical, and protein networks often have between 2 and 20 class labels, which are highly correlated with the structural properties and behavior surrounding a given node or edge in the graph [14], [15]. Furthermore, the nodes whom share class labels are often not directly connected, or even in the same community, but share similar structural properties and behavior (or role [16]) in the network as shown in Figure 1. Current methods work well when the neighboring labels of a node are highly correlated (*i.e.*, the neighbors of a node have the same class label), but perform poorly otherwise. Instead, this work proposes a relational learning framework based on maximizing similarity called RSM, which generalizes across the spectrum of relational data characteristics, and avoids the issues, assumptions, and limitations of existing methods. In particular, RSM is able to learn and accurately predict the class labels of nodes (links, subgraphs) in large (attributed) networks that are extremely noisy and sparsely labeled (with unbalanced classes). Most importantly, RSM is designed for interactive RML and able to handle networks with arbitrary relational autocorrelation by adjusting a simple hyperparameter.

In addition, existing methods also have difficulty learning with graph data that is large, noisy, probabilistic, sparsely labeled, attributed, and are sensitive to many other issues and data characteristics that often arise in practice. Moreover, obtaining labeled data is also expensive, and thus RML methods should be robust to learning with few labeled instances. Furthermore, relational representation and transformations of the nodes, links, and/or features can dramatically affect the capabilities and results of such algorithms [6]. Unlike previous work, RSM is designed to be fast and scalable for real-time interactive RML.

To solve these problems, this paper introduces relational similarity machines (RSM) — an efficient *interactive* similarity-based graph learning framework. The framework is well-suited for classification tasks in graphs with arbitrary homophily and heterophily. It also generalizes to both graph-based supervised and semi-supervised learning (SSL) [17] and gives rise to a variety of methods for both settings. RSM is extremely fast, space-efficient, accurate, flexible with many

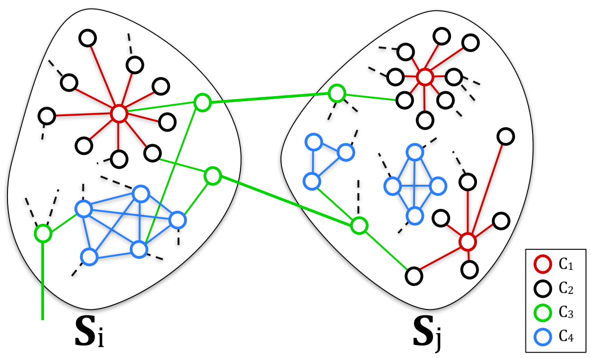


Figure 1. In practice, nodes with the same class label are often not directly connected, or even in the same community, but share similar structural properties (roles [16]). Sets of nodes of the same class are denoted by C_1, C_2, C_3, and C_4, *i.e.*, $\forall\, v_i, v_j \in C_i,\;\; \xi(v_i) = \xi(v_j)$. Current methods work well when the neighboring labels of a node are highly correlated (*i.e.*, the neighbors of a node have the same class label), but perform poorly otherwise. In contrast, RSM performs well on networks with arbitrary relational autocorrelation and therefore does not have the same issues, assumptions, and limitations of previous methods. In the above example, nodes that belong to the same class are often connected to nodes of a different class. For instance, nodes that belong to class C_1 are almost always connected to nodes of C_2, and never nodes of the same class C_1. Similarly, nodes in class C_2 mostly connect to nodes of class C_1 and less frequently nodes belonging to class $C3$. Further, nodes in class C_3 connect to nodes of all classes.

interchangeable components, and highly scalable for *real-time interactive learning*. In addition, RSM is designed to be intuitive and easy to adapt and encode application constraints. The similarity-based graph learning framework also has many other attractive properties including its robustness to noise as well as its ability to handle sparsely labeled graph data.

The main strengths of our approach include:

- A general principled similarity-based relational learning framework for supervised and semi-supervised learning (SSL) in large graphs.
- Accurate for prediction in graphs with arbitrary homophily and heterophily
- Fast and scalable for interactive relational learning and other applications requiring real-time performance
- Flexible with many interchangeable components (*e.g.*, parameterized similarity function)

II. Relational Similarity Machines

This section describes a general relational learning framework based on the notion of maximum similarity called relational similarity machines (RSM). The similarity-based relational learning framework gives rise to a large class of graph-based supervised and semi-supervised learning algorithms and serves as a unifying basis for studying them.

Let $G = (V, E, \mathcal{X})$ be an attributed graph where V is a set of nodes E is a set of edges and $\mathcal{X}$ is the set of n training objects $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_i, y_i), \ldots, (\mathbf{x}_n, y_n)\}$ where each $\mathbf{x}_i \in \mathbb{R}^d$ is a d-dimensional feature vector for node v_i and $y_i \in \{1, 2, \ldots, k\}$ is the class label of v_i, also denoted simply as $\xi(v_i)$. For convenience, let $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ and $\boldsymbol{Z} \in \mathbb{R}^{m \times d}$ be matrices consisting of n training and m test instances, respectively. Further, we define $\mathbf{x}_i \in \mathbb{R}^d$ (or $\boldsymbol{X}_{i:}$) as the i^{th} row vector of $\boldsymbol{X}$, and similarly $\mathbf{z}_j$ is the j^{th} row vector of $\boldsymbol{Z}$.

Definition 2.1 (Within-network classification): Given a (attributed) graph G, a known set of node labels $Y^\ell = \{y_i | v_i \in V^\ell\}$ for nodes $V^\ell \subset V$, the within-network classification task is to infer Y^u — the set $V^u = V \setminus V^\ell$ of remaining vertices with unknown labels.

We define $\mathcal{C}$ to be the set of class labels, and thus, $k = |\mathcal{C}|$ is the number of unique labels. As shown later, our approach naturally generalizes to both binary classification problems where $y_i \in \{1, 2\}$ as well as large multiclass classification problems where $|\mathcal{C}| > 2$, and thus, $y_i \in \{1, 2, \ldots, k\}$. Further, let $\boldsymbol{A} = \big[A_{ij}\big]$ be the adjacency matrix of G where $A_{ij} = 1$ if there exists $(v_i, v_j) \in E$ and $A_{ij} = 0$ otherwise. Note $\boldsymbol{A}$ may also be used to encode edge attributes; given w for $(v_i, v_j) \in E$, $A_{ij} = w$. WLOG sets are ordered, thus $V = \{v_1, ..., v_i, ..., v_j, ..., v_n\}$ is an ordering of the nodes in G *s.t.* $v_i < v_j$ iff $f(v_i) \leq f(v_j)$ where $f(\cdot)$ is an arbitrary function. Vertices u and v are adjacent if $(v, u) \in E$. Given a vertex $v \in V$, let $\Gamma(v) = \{w | (v, w) \in E\}$ be the set of vertices adjacent to v in G. For a vertex $v \in V$, let d_v be the degree of v. The maximum degree is denoted by Δ.

Assume *w.l.o.g.* that rows of $\boldsymbol{X}$ and $\boldsymbol{Z}$ have been normalized to length 1 using the ℓ^2-norm. Nevertheless, given a radius of sufficient magnitude it is conceptually simple and trivial algebraically to approximate a hyperplane by a spherical area, *i.e.*, simply add the radius of the sphere to the coefficients of each point and normalize. Thus, general classification of points in $\mathbb{R}^d$ are easily addressed in RSM.

A general computational framework for RSM is given in Alg. 1. The RSM framework gives rise to a large space of potential relational learning methods due to RSM's powerful representation and flexibility. Furthermore, many of the learning components in Alg. 1 are naturally interchangeable. The subsequent sections discuss the main components of RSM. In particular, Section II-A presents the supervised learning component whereas Section II-B introduces the semi-supervised learning (SSL) component of RSM.

A. Supervised Component

Given an unlabeled test node $v_i \in V^u$ and its feature vector $\mathbf{z}_i \in \mathbb{R}^d$, we demonstrate how to estimate the weights for v_i from the set of *labeled nodes* V^ℓ (Line 11-17). First, the set of labeled nodes V^ℓ is decomposed into two key subsets:

$$N = \{v_j \in \Gamma_h(v_i) \mid v_j \in V^\ell\} \tag{1}$$

$$Q = \{v_j \notin \Gamma_h(v_i) \mid v_j \in V^\ell\} \tag{2}$$

where N (Eq. 1) is the set of labeled neighbors within h-hops of $v_i \in V^u$ and Q (Eq. 2) is the set of labeled non-neighbors.

These sets are then used to decompose the similarity scores of the labeled nodes V^ℓ into k-dimensional weight vectors $\mathbf{w}_i^R$ and $\mathbf{w}_i^I$ where $\mathbf{w}_i^R \in \mathbb{R}^k$ is the weight vector estimated using N (Eq. 1) and $\mathbf{w}_i^I \in \mathbb{R}^k$ is the weight vector estimated using Q (Eq. 2). The similarity score S_{ij} in Line 12 of Alg. 1 represents the similarity between the feature vector $\mathbf{z}_i$ of the test node $v_i \in V^u$ and the feature vector $\mathbf{x}_j$ of $v_j \in V^\ell$. Note S_{ij} is implicitly computed and the full similarity matrix $\boldsymbol{S}$ is never stored as S_{ij} is immediately combined and discarded.

The similarity scores accumulated from each set of labeled nodes are then combined as follows:

$$\mathbf{w}_i = \overbrace{\underbrace{\alpha \mathbf{w}_i^R}_{\text{Relational}} + \underbrace{(1-\alpha)\mathbf{w}_i^I}_{\text{IID}}}^{\text{Supervised}} \tag{3}$$

where α is a hyperparameter which satisfies $0 \leq \alpha \leq 1$, and $\mathbf{w}_i \in \mathbb{R}^k$ is the non-negative vector with $W_{ik} \geq 0$. For convenience, assume *w.l.o.g.* that $\mathbf{w}_i$ is also a stochastic row vector with $\sum_k W_{ik} = \mathbf{w}_i\mathbf{e} = 1$. For a completely supervised variant of RSM, we simply skip Lines 18-25 in Alg. 1 and set $\tau = 1$. The class of a test node $v_i \in V^u$ is then predicted as:

$$\xi(\mathbf{z}_i) = \arg\max_{k \in \mathcal{C}} \left[\alpha w_{ik}^R + (1-\alpha) w_{ik}^I \right] \tag{4}$$

where $\xi(\mathbf{z}_i)$ is the class with maximum similarity taken over all other class labels.

B. Semi-Supervised Learning Components

In this section we discuss the semi-supervised learning (SSL) components of RSM that make use of unlabeled data to improve predictions. In this work, we consider the similarity between the set of unlabeled nodes V^u (Line 19-25) and use this similarity to estimate class probabilities for the test node v_i being predicted. Given an unlabeled test node $v_i \in V^u$ (to predict), we decompose V^u into the sets:

$$N' = \{v_j \in \Gamma_h(v_i) \mid v_j \in V^u\} \tag{5}$$

$$Q' = \{v_j \notin \Gamma_h(v_i) \mid v_j \in V^u\} \tag{6}$$

where N' is the set of *unlabeled neighbors* of v_i and Q' is the set of *unlabeled non-neighbors*, *i.e.*, nodes that are not neighbors of v_i. The weights are decomposed into $\mathbf{m}_i^R \in \mathbb{R}^k$ and $\mathbf{m}_i^I \in \mathbb{R}^k$ where $\mathbf{m}_i^R$ is the weight vector estimated using N' and $\mathbf{m}_i^I$ is the weight vector estimated using Q'.

Given $v_i, v_j \in V^u$ and their d-dimensional feature vectors $\mathbf{z}_i$ and $\mathbf{z}_j$, we first derive a measure of similarity $S_{ij} = \Phi\langle \mathbf{z}_i, \mathbf{z}_j \rangle$ where $\Phi\langle \cdot, \cdot \rangle$ is an arbitrary similarity function. Recall the *supervised* component (Line 11-17) of RSM used $S_{ij} = \Phi\langle \mathbf{z}_i, \mathbf{x}_j \rangle$ to update the weight corresponding to the class of the training node $v_j \in V^\ell$. However, in this case both the class label of v_i and v_j are unknown. Thus, instead of using the similarity S_{ij} to update the estimate of a particular class (Section II-A), we instead use S_{ij} to update all class

Algorithm 1 Relational Similarity Machines (RSM) Framework

```
1  procedure RSM( $\boldsymbol{G}, \boldsymbol{X}, \boldsymbol{Z}, \tau_{\max},$ )
2      Normalize all data; Set $\tau \leftarrow 1$ and $U = \emptyset$
3      Estimate the class priors $\boldsymbol{P} = [\cdots\ \mathbf{p}_i\ \cdots]^T$ where $\mathbf{p}_i \in \mathbb{R}^k$ is the
       estimated prior for $v_i$
4      repeat                                  ▷ outer iterations $\tau = 1, 2, \ldots, \tau_{\max}$
5          Compute relational features based on neighbor classes
6          Compute relational features based on neighbor attributes
7          Append the relational features from Line 5-6 to the current set of
           features and renormalize.
8          for each $v_i \in V^u$ in order                       ▷ next test instance
9              Set $\mathbf{w}_i^R, \mathbf{w}_i^I, \mathbf{m}_i^R, \mathbf{m}_i^I$ to $\mathbf{0} = [0\ \cdots\ 0] \in \mathbb{R}^k$
10             Obtain a set $J$ by sampling $(V^\ell \cup U)$ via an arbitrary (weighted/u-
               niform) distribution F (if needed, otherwise $J = V^\ell \cup U$)
11             parallel for each $v_j \in J$                  ▷ Supervised component
12                 Set $s_{ij}$ to be $\Phi\langle \mathbf{z}_i, \mathbf{x}_j \rangle$
13                 Let $k \in \{1, \ldots, |\mathcal{C}|\}$ be the class label of $v_j \in V^\ell$
14                 if $v_j \in \Gamma_h(v_i)$ then
15                     Update $w_{ik}^R \leftarrow w_{ik}^R + p_{ik} \cdot s_{ij}$       ▷ labeled neighbor
16                 else Update $w_{ik}^I \leftarrow w_{ik}^I + p_{ik} \cdot s_{ij}$     ▷ labeled non-neigh.
17             end parallel
18             Obtain a set $J$ by sampling $(V^u \setminus U)$ via an arbitrary (weighted/u-
               niform) distribution F (if needed, otherwise $J = V^u \setminus U$)
19             parallel for each $v_j \in J$             ▷ Semi-supervised component
20                 Set $s_{ij}$ to be $\Phi\langle \mathbf{z}_i, \mathbf{z}_j \rangle$
21                 for each class $k \in \mathcal{C}$ do
22                     if $v_j \in \Gamma_h(v_i)$ then
23                         $m_{ik}^R \leftarrow m_{ik}^R + s_{ij}(p_{ik}p_{jk})$    ▷ unlabeled neighbor
24                     else $m_{ik}^I \leftarrow m_{ik}^I + s_{ij}(p_{ik}p_{jk})$   ▷ unlabel. non-neigh.
25             end parallel
26             Normalize $\mathbf{w}_i^R, \mathbf{w}_i^I, \mathbf{m}_i^R, \mathbf{m}_i^I$ (see text for discussion).
27             Update $\mathbf{p}_i$ via Eq.(3)–(8)
28             Solve $y_i \leftarrow \arg\max_{k \in \mathcal{C}} P_{ik}$                 ▷ k is the class label
29         end for
30         Given $\boldsymbol{P}$, estimate $\mathbf{c} = [c_i\ c_2\ \cdots\ c_i\ \cdots]$ for each $v_i \in (V^u \setminus U)$
           where $c_i$ is a measure of "confidence" for $v_i$
31         Find $V' \subset V^u$ s.t. $V'$ is the top-$\kappa$ nodes with largest confidence and
           $\kappa = \lceil \psi |V^u \setminus U| \rceil$. Set $U \leftarrow U \cup V'$ and update $\psi$ if needed.
32         Include $\boldsymbol{P}, \boldsymbol{W}^R, \boldsymbol{W}^I, \boldsymbol{M}^R, \boldsymbol{M}^I$, and/or $\mathbf{c}$ as features (see text)
33         Set $\tau \leftarrow \tau + 1$ and renormalize data
34     until stopping criterion is reached or $\tau > \tau_{\max}$
```

estimates (Lines 21-24). More specifically, given a class $k \in \mathcal{C}$, the update is $S_{ij}(P_{ik}P_{jk})$ where P_{ik} and P_{jk} represent the current probability that v_i and v_j belong to class k. Thus, S_{ij} is essentially a weighted similarity using the current class probabilities P_{ik} and P_{jk}. The weighted similarities are added to either $\mathbf{m}_i^R$ or $\mathbf{m}_i^I$ (Lines 21-24). The semi-supervised weight vectors $\mathbf{m}_i^R$ and $\mathbf{m}_i^I$ accumulated from the sets of unlabeled neighbors (Eq. (5)) and unlabeled non-neighbors (Eq. (6)) of a given test node $v_i \in V^u$ are then used to derive $\mathbf{m}_i \in \mathbb{R}^k$ as follows:

$$\mathbf{m}_i = \overbrace{\underbrace{\beta \mathbf{m}_i^R}_{\text{Relational}} + \underbrace{(1-\beta)\mathbf{m}_i^I}_{\text{IID}}}^{\text{Semi-Supervised}} \tag{7}$$

where β is a hyperparameter which satisfies $0 \leq \beta \leq 1$ and effectively determines the weight given to the unlabeled disconnected nodes relative to the importance of the unlabeled connected nodes. Hence, as $\beta \to 1$ the weights estimated via the connected unlabeled nodes are given more influence in the learning and conversely for $\beta \to 0$.

In addition to the above SSL strategy, we also estimate the confidence of each unlabeled test node after every iteration τ in Alg 1 and predict the labels of the top-κ nodes for which we are most certain (min uncertainty) where $\kappa = \lceil \psi \cdot |V^u \setminus U| \rceil$ and ψ is the fraction of nodes to predict labels for at each iteration. These nodes are added to U and used in subsequent iterations (Alg. 1 Line 31).

C. Collective Update

Now we jointly estimate $\mathbf{p}_i^{(\tau+1)}$ for test node $v_i \in V^u$ using the k-dimensional weight vectors $\mathbf{w}_i$ and $\mathbf{m}_i$ as follows:

$$\mathbf{p}_i^{(\tau+1)} = \overbrace{\Big(\underbrace{\mathbf{w}_i}_{\text{Supervised}} + \underbrace{\mathbf{m}_i}_{\text{SSL}}\Big)}^{\text{current estimate}} + \overbrace{\mathbf{p}_i^{(\tau)}}^{\text{previous}} \tag{8}$$

The update equations in Eq. (3)-(8) combine the learned weights from the different learning components that leverage labeled (supervised) and unlabeled nodes (semi-supervised), as well as connected (neighbors) and disconnected (non-neighbor) nodes. Thus, $\mathbf{p}_i^{(\tau+1)}$ is the new estimate and derived by combing the estimated weights from the four different learning settings. It is straightforward to leverage other update equations as well as add regularization for different applications and data characteristics. This is especially useful for interactive relational learning [18]. A fundamental advantage over other approaches is the accuracy, efficiency, and flexibility of RSM for graphs with homophily *and* heterophily. To ensure RSM is fast with real-time response rates for interactive learning, we sample from the various *types* of nodes (Alg. 1: Line 10 and 18).

D. Maximizing Relational Similarity

At the heart of relational similarity machines (RSM) lies the notion of maximum similarity. Let $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ be a matrix where the rows represent training nodes and the columns represent features. Further, let $\boldsymbol{Z}$ be a matrix of test node feature vectors, then the class of a single test node $\mathbf{z}_i \in V^u$ is predicted as follows. First, the similarity of $\mathbf{z}_i$ with respect to each training example in $\boldsymbol{X}$ is computed. For instance, suppose $\mathbf{x}_j$ belongs to class $k \in \mathcal{C}$ then $S_{ij} = \Phi(\mathbf{z}_i, \mathbf{x}_j)$ is added to the kth element w_k of the weight vector $\mathbf{w}$. Note we use $\mathbf{w}$ in this section to denote an arbitrary weight vector which is not to be confused with $\mathbf{w}$ in Section II-A. The similarity of the instances in $\boldsymbol{X}$ of class k with respect to the test node feature vector $\mathbf{z}_i$ is formalized as,

$$w_k = \sum_{\mathbf{x}_j \in \mathcal{X}_k} \Phi \left\langle \mathbf{z}_i, \mathbf{x}_j \right\rangle \tag{9}$$

where $\mathcal{X}_k$ is the set of training node feature vectors from $\boldsymbol{X}$ of class $k \in \mathcal{C}$. Thus $\mathbf{w}$ is simply

$$\mathbf{w} = \begin{bmatrix} \sum\limits_{\mathbf{x}_j \in \mathcal{X}_1} \Phi \left\langle \mathbf{z}_i, \mathbf{x}_j \right\rangle \\ \vdots \\ \sum\limits_{\mathbf{x}_j \in \mathcal{X}_c} \Phi \left\langle \mathbf{z}_i, \mathbf{x}_j \right\rangle \end{bmatrix} \tag{10}$$

After computing $\mathbf{w}$, then $\mathbf{z}_i$ is assigned to the class with maximum similarity. Therefore, we predict the class of $\mathbf{z}_i$ using the following decision function:

$$\xi(\mathbf{z}_i) = \arg\max_{k \in \mathcal{C}} \quad w_k \tag{11}$$

where $\xi(\cdot)$ is the predicted class. The above requires $\mathbf{z}_i$ to be more similar to class k than to any other class.

The above formalization corresponds to the supervised learning component presented in Section II-A where $\mathcal{X}_k$ can be defined as the feature vectors for the train nodes in N, Q, or both $V^\ell = N \cup Q$. It is straightforward to extend the above to the SSL components described formally in Section II-B. We now state the complete decision function that uses both the supervised and semi-supervised weight vectors. More formally, the class label of a test node $\mathbf{z}_i$ is:

$$\arg\max_{k \in \mathcal{C}} \left[\Big(w_{ik} + m_{ik} \Big) + p_{ik} \right] \tag{12}$$

The RSM framework uses parameterized similarity functions and does not require mappings in high-dimensional Hilbert spaces. A few of the similarity functions investigated in this work are given below. Recall that $\boldsymbol{X}$ and $\boldsymbol{Z}$ are matrices consisting of training and test node features, respectively. We can interpret $\boldsymbol{Z}\boldsymbol{X}^T$ as a similarity matrix $\boldsymbol{S}$ with the dot product as the similarity measure.

Radial Basis Functions (RBF): Given d-dimensional vectors $\mathbf{z}_i$ and $\mathbf{x}_j$, the RBF similarity function is:

$$\Phi(\mathbf{z}_i, \mathbf{x}_j) = \exp\left(- \frac{\|\mathbf{z}_i - \mathbf{x}_j\|^2}{2\sigma^2} \right) \tag{13}$$

where the radius of the RBF function is controlled by choice of σ, *i.e.*, tightness of the similarity measure.

Polynomial Functions: A common class of similarity measures are polynomial functions of the form:

$$\Phi(\mathbf{z}_i, \mathbf{x}_j) = \Big(\left\langle \mathbf{z}_i, \mathbf{x}_j \right\rangle + c \Big)^q \tag{14}$$

where q is the degree of the polynomial and c is a regularization term trading off higher-order terms for lower-order ones in the polynomial. A generalization of the above is $\Phi(\mathbf{z}_i, \mathbf{x}_j) = \big(a \left\langle \mathbf{z}_i, \mathbf{x}_j \right\rangle + c \big)^q$. Linear-RSM and quadratic-RSM are special cases of Eq. (14) where $q = 1$ and $q = 2$, respectively. Polynomial kernels are important for NLP and other applications [19].

Sigmoid Kernel: The sigmoid kernel is defined as

$$\Phi(\mathbf{z}_i, \mathbf{x}_j) = \tanh\big(a\,\langle \mathbf{z}_i, \mathbf{x}_j\rangle + c\big), \tag{15}$$

where a and c are constants representing the slope and intercept, respectively. The sigmoid kernel is important in neural networks and deep learning [20], [21].

Cauchy Similarity Function: The Cauchy "long-tailed" similarity function is defined as:

$$\Phi(\mathbf{z}_i, \mathbf{x}_j) = \left(\frac{1}{1 + \frac{\|\mathbf{z}_i - \mathbf{x}_j\|^2}{\sigma^2}}\right)$$

E. Over and Under Fitting Conditions

Parameterized similarity measures such as RBF's σ and the degree-q of the polynomial kernel control the tightness of such measures. Extreme values of these parameters correspond to *total overfitting* (extreme tightness) where $\sigma \to \infty$, or *total underfitting* (lack of tightness) at the other extreme where $\sigma \to 0$. Let $\boldsymbol{S} = \langle \boldsymbol{Z}, \boldsymbol{X}\rangle$ be defined as the similarity matrix $\forall\, v_i \in V^u$, $v_j \in V^\ell : \Phi(\mathbf{z}_i, \mathbf{x}_j)$. As $\sigma \to 0$ (or $q \to 0$ in polynomial functions) then $\boldsymbol{S} \to \mathbf{e}\mathbf{e}^T$ where $\mathbf{e} = [1\ 1\ \cdots\ 1]^T$ and thus $\boldsymbol{S}$ is close to low-rank. Consequently, all objects appear equally similar, and results in a decision function that assigns each test instance to the class with the largest number of instances, thus, this extreme case is called extreme underfitting. Similarly, $\sigma \to \infty$ then $\boldsymbol{S} \to \boldsymbol{I}$ where $\boldsymbol{I}$ is the identity matrix and thus is full rank with all eigenvalues equal to 1. As the similarity matrix $\boldsymbol{S}$ converges towards $\boldsymbol{I}$, the decision function converges towards extreme overfitting.

F. Discussion

To ensure RSM is fast for large graph data, Alg. 1 samples nodes in the training set from each class (Line 10 and 18). Recall that n denotes the number of train instances, m denotes the number of test instances, and d is the number of features. Sampling has three important advantages. First, it serves as a bound on the time complexity per node and reduces it from $\mathcal{O}(nd)$ to $\mathcal{O}(|J| \cdot d)$ where $|J| \ll n$. Second, it reduces the impact when one or more classes are significantly skewed or underrepresented (*e.g.*, one can ensure that approximately the same number of unlabeled and labeled nodes are used in the prediction). Finally, a stratified sampling approach can be used to select nodes proportionally from the different classes (or for the unlabeled nodes, one can use the probability distributions estimated for each node).

III. Complexity Analysis

Time Complexity: Given n training instances, m test instances, and d features. RSM takes $\mathcal{O}(nd)$ time on a single test instance (in the stream), and therefore $\mathcal{O}(nmd)$ for all m test examples. The complexity for both sparse and dense training sets $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ are given below. Consider the case where $\boldsymbol{X}$ is sparse and stored as a sparse matrix using compressed sparse column/row format. The cost of a single test example is $\mathcal{O}(|\boldsymbol{X}|)$ linear in the number of nonzeros in $\boldsymbol{X}$ denoted by $|\boldsymbol{X}|$. Further, assuming p processing units, then $\mathcal{O}(|\boldsymbol{X}|/p)$, and hence is very scalable for real-time systems. Now suppose $\boldsymbol{X}$ is a dense matrix. Given a dense training set $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ (few zeros), it takes $\mathcal{O}(nd)$ time for each test object, and $\mathcal{O}(\frac{nd}{p})$ assuming p processing units. The time is reduced by sampling from the various types of instances (Alg. 1: Line 10 & 18). In particular, the time to infer the label of a single test instance is reduced from $\mathcal{O}(nd)$ *to* $\mathcal{O}(|J| \cdot d)$ where $|J| \ll n$.

Space Complexity: Finally, we discuss the space complexity of the proposed relational learning framework. Given a single test node to predict, RSM takes only $\mathcal{O}(k)$ space where k is the number of classes. This is of course in addition to the graph and features. For parallel learning and inference using p workers, the lock-free version of RSM takes $\mathcal{O}(pk)$ space. To avoid locks, each worker maintains a k-dimensional vector of similarity scores, which are then combined upon completion. Now, suppose collective inference is used, then RSM takes only $\mathcal{O}(nk)$ space where n is the number of test nodes with unknown class labels.

IV. Experiments

The experiments are designed to investigate the effectiveness and scalability of the RSM learning framework.

A. Experimental Setup

The data used in the experiments is available at Network Repository [22]. Unless otherwise mentioned, we perform a grid search over $\alpha, \beta \in \{0, 0.01, 0.1, 0.25, 0.5, 0.75, 0.99, 1\}$ and $\sigma \in \{0.001, 0.01, 0.05, 0.1\}$. We use 10% of the labeled nodes as training unless otherwise mentioned. The model is selected using 5-fold cross-validation and results are averaged over 20 trials. The initial class prior is set to the overall class label distribution and a fast belief-propagation approach is used to obtain an initial estimate. Unfortunately, there are no existing interactive RML methods to use as baselines for evaluating RSM. Nevertheless, we extend WVRN [7], RPT [23], SVM-G (with Graphlet features) [24] and RSM-*iid* for interactive RML and use these as baselines for comparison. These baseline models were chosen since they have similar time *and* space complexity and are straightforward to extend for interactive RML.

B. Experiments on Graphs with Homophily

This section investigates the classification performance of RSM for graphs with homophily. Classification results comparing RSM to the baseline methods are shown in Table I. Note $\mathcal{L}$ in Table I is an intuitive measure of homophily called *label consistency* [7] defined as $\mathcal{L}(G) = {}^1\!/_{|E|} \sum_{(v_i,v_j)\in E} \mathcal{L}(v_i, v_j)$ where $\mathcal{L}(v_i, v_j) = 1$ if $\xi(v_i) =$

Table I
CLASSIFICATION RESULTS FOR GRAPHS WITH HOMOPHILY

			Accuracy				
Graph	$\lvert\mathcal{C}\rvert$	$\mathcal{L}$	RSM	RSM-IID	WVRN	SVM-G	RPT
aff–polbooks	3	66.6%	**89.25**	71.16	74.41	69.89	70.97
bio–Gene	2	79.7%	**85.27**	64.88	72.41	61.85	60.24
Enzymes349	2	50.0%	**77.81**	67.93	62.50	55.36	66.07
musicGenre	8	84.9%	**82.35**	51.57	60.59	48.63	55.56

$\xi(v_j)$ the class labels of node v_i and v_j match and 0 otherwise. Similar results were observed with other "homophily measures" such as assortativity/relational autocorrelation. In all cases, RSM outperforms the other methods and the results are significant at p-val $<$0.05. These results demonstrate the effectiveness of RSM for classification on graphs with high degrees of homophily (Table I). Notably, the label consistency ranges from 50% observed in Enzymes349 all the way up to 84.9% in musicGenre.

C. Experiments on Graphs with Heterophily

We also investigate the effectiveness of RSM on graphs with heterophily. Results are provided in Table II. F1 score is used for evaluation due to the unbalanced nature of these graph problems and the large number of classes [25]. Notice the label consistency $\mathcal{L}$ of these graphs is small as shown in Table II which indicates heterophily. In all cases, RSM outperforms WVRN, SVM-G, and RPT across all networks with an average gain of 24%, 52%, and 17% respectively. More strikingly, we observe that RSM-IID outperforms WVRN, SVM-G, and RPT across all graphs with heterophily. Notably, RSM-IID uses only relational and graph topology features such as 3 and 4-vertex graphlet counts. This indicates the importance of appropriately moderating the heterophily and homophily dependencies. Further, existing relational learning methods are unable to handle graphs with low levels of label consistency (heterophily). It is evident from the results that RSM is flexible for prediction in networks with both homophily and heterophily.

Table II
HETEROPHILY CLASSIFICATION RESULTS

			F1 score				
Graph	$\lvert\mathcal{C}\rvert$	$\mathcal{L}$	RSM	RSM-IID	WVRN	SVM-G	RPT
DD645	20	2.0%	**0.769**	0.701	0.607	0.634	0.680
DD411	20	12.5%	**0.712**	0.695	0.605	0.239	0.615
DD5	19	6.7%	**0.561**	0.510	0.303	0.426	0.465
DD244	20	7.1%	**0.742**	0.708	0.558	0.175	0.646
DD159	20	2.70%	**0.734**	0.702	0.605	0.280	0.652
DD185	18	12.5%	**0.738**	0.644	0.585	0.262	0.489

D. Real-time Performance

We investigate the real-time performance capabilities of RSM for the interactive RML problem (Figure 3). Overall, RSM is shown in Table III to be extremely fast with real-time response times in the range of a few ms or less. Notably, the results indicate that RSM is fast and naturally able to support real-time interactive learning and inference. In particular, RSM is able to provide rapid immediate (and visual) feedback to the user at real-time interactive response times (Section IV-F).

Table III
AVERAGE TIME PER TEST INSTANCE

Graph	$\lvert V^{\ell}\rvert$	$\lvert\mathcal{C}\rvert$	**avg. time**
soc–TerroristRel	90	2	0.18 ms
aff–polbooks	12	3	0.12 ms
cora	272	7	0.97 ms
DD6	417	20	3.10 ms
political-retweet	1848	2	0.15 ms

E. Parallel Scalability

This section demonstrates the parallel scalability of the proposed learning framework in Figure 2. Given p parallel workers (cores, processors), speedup is defined as $S_p = \frac{T_{\text{seq}}}{T_p}$ where T_{seq} is the runtime of the sequential algorithm and T_p is the runtime of the parallel algorithm on p workers. In particular, we observe strong scaling results in Figure 2. Note a machine with two Intel Xeon E5-2687 CPUs @3.10GHz were used with 8 cores each.

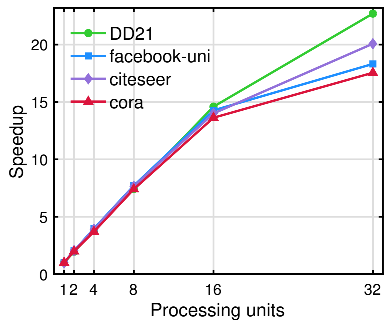
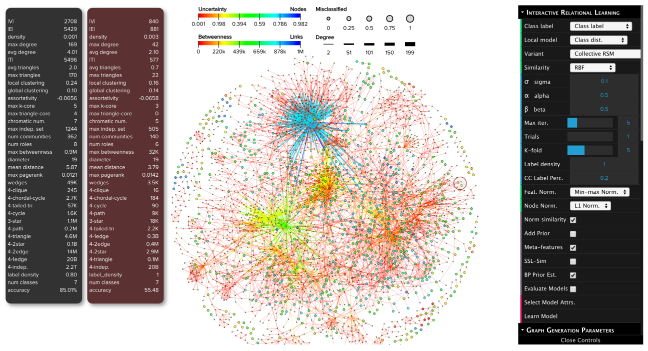


Figure 2. Parallel scaling. Strong scaling results are observed across a variety of networks including social and information networks.

F. Effectiveness for Interactive RML

Relational Machine Learning (RML) [4] methods exploit the relational dependencies between nodes to improve predictive performance [7]. However, these approaches often fail in practice due to low relational autocorrelation, noisy links, sparsely labeled graphs, and data representation [6]. To overcome these problems, we designed the class of RSM models to be fast for interactive RML with real-time response rates. This allows users to interactively search the space of RSM models in an interactive and visual fashion. For instance,



|V|	2708
|E|	5429
density	0.001
max degree	169
avg degree	4.01
|T|	5496
avg triangles	2.0
max triangles	170
local clustering	0.24
global clustering	0.10
assortativity	-0.0656
max k-core	5
max triangle-core	4
chromatic num.	7
max indep. set	1244
num communities	362
num roles	8
max betweenness	0.9M
diameter	19
mean distance	5.87
max pagerank	0.0121
wedges	49K
4-clique	245
4-chordal-cycle	2.7K
4-tailed-tri	57K
4-cycle	1.6K
3-star	1.1M
4-path	0.2M
4-triangle	4.6M
4-2star	0.1B
4-2edge	14M
4-1edge	20B
4-indep.	2.2T
label density	0.80
num classes	7
accuracy	85.01%

|V|	840
|E|	881
density	0.003
max degree	42
avg degree	2.10
|T|	577
avg triangles	0.7
max triangles	22
local clustering	0.16
global clustering	0.14
assortativity	-0.0658
max k-core	3
max triangle-core	0
chromatic num.	5
max indep. set	505
num communities	140
num roles	6
max betweenness	32K
diameter	19
mean distance	3.79
max pagerank	0.0142
wedges	3.5K
4-clique	16
4-chordal-cycle	184
4-cycle	90
4-path	9K
3-star	18K
4-tailed-tri	2.2K
4-1edge	0.3B
4-2edge	0.4M
4-2star	2.9M
4-triangle	0.1M
4-indep.	20B
label_density	1
num classes	7
accuracy	55.48

▾ INTERACTIVE RELATIONAL LEARNING

Class label	Class label
Local model	Class dist.
Variant	Collective RSM
Similarity	RBF
σ sigma	0.1
α alpha	0.5
β beta	0.5
Max iter.	5
Trials	1
K-fold	5
Label density	1
CC Label Perc.	0.2
Feat. Norm.	Min-max Norm.
Node Norm.	L1 Norm.
Norm similarity	✓
Add Prior	☐
Meta-features	✓
SSL-Sim	☐
BP Prior Est.	✓
Evaluate Models	☐

Figure 3. Overview and effectiveness of RSM for interactive RML. The visualization above is from `cora`.

a user can specify the RSM model in real-time using a visual interface as well as perform evaluation, analyze errors, and make adjustments and refinements in a closed-loop [18].

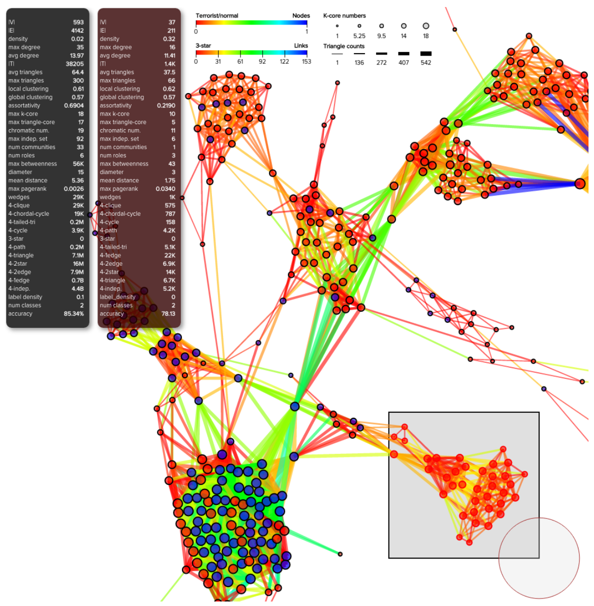
An overview is provided in Figure 3. We first interactively learn a model in Figure 3, then select the misclassified nodes for further analysis. The global statistics of the selected subgraph are shown in the right-most panel. Node color represents the model's uncertainty using an entropy-based measure, whereas the size of the node indicates whether it was correctly classified or not. In Figure 3, misclassified nodes are given a larger size so that they can easily be identified for further exploration. Localized updates are also used in RSM to enable real-time inference and exploration capabilities by leveraging fast exact or approximate solutions to support real-time interactive queries seamlessly; see Figure 4. Many components in RSM may be explored in real-time using interactive visualization and analytic techniques (Figure 3), including the attribute to predict, initial features to use (non-relational and graph-based features), local model for estimation, kernel function (RBF, linear, polynomial), hyper-parameters (for selected kernel), node- and feature-wise normalization scheme (L1, min-max), as well as whether to use both supervised and SSL components, among many others. See Figure 3 (right panel).

G. Varying Inference Types

Previous work defined different "relational models" based on the class of relational features used. The simplest models leverage only one class of features. The most accurate models typically combine other classes of features including:

- **Collective Inference (CI)**: Model using *neighbor labels*.
- **Relational Inference (RI)**: The RI model uses *neighbor attributes*.
- **Relational-Collective Inference (RCI)**: The RCI model uses both *neighbor labels* and *neighbor attributes*.

Collective inference uses "neighbor labels" to estimate the nodes with unknown labels, whereas *relational inference* uses "neighbor attributes" [26]. Our approach naturally leverages both collective and relational inference.



|V|	593
|E|	4142
density	0.02
max degree	35
avg degree	13.97
|T|	38205
avg triangles	64.4
max triangles	300
local clustering	0.61
global clustering	0.57
assortativity	0.6904
max k-core	18
max triangle-core	17
chromatic num.	19
max indep. set	92
num communities	33
num roles	6
max betweenness	56K
diameter	15
mean distance	5.36
max pagerank	0.0026
wedges	29K
4-clique	29K
4-chordal-cycle	19K
4-tailed-tri	0.2M
4-cycle	3.9K
3-star	0
4-path	0.2M
4-triangle	7.1M
4-2star	16M
4-2edge	7.9M
4-1edge	0.7B
4-indep.	4.4B
label density	0.1
num classes	2
accuracy	85.34%

|V|	37
|E|	211
density	0.32
max degree	16
avg degree	11.41
|T|	1.4K
avg triangles	37.5
max triangles	66
local clustering	0.62
global clustering	0.57
assortativity	0.2190
max k-core	10
max triangle-core	5
chromatic num.	11
max indep. set	6
num communities	1
num roles	3
max betweenness	43
diameter	3
mean distance	1.75
max pagerank	0.0340
wedges	1K
4-clique	575
4-chordal-cycle	787
4-cycle	158
4-path	4.2K
3-star	0
4-tailed-tri	5.1K
4-1edge	22K
4-2edge	6.9K
4-2star	14K
4-triangle	6.7K
4-indep.	5.2K
label_density	0
num classes	2
accuracy	78.13

Figure 4. At the heart of RSM is the ability to interactively learn models in real-time and adjust them accordingly, as well as tune parameters, explore/construct features, among many other possibilities. In the screenshot above (from soc–terror), the user interactively learns a global model (see the leftmost black side panel for stats and accuracy), and then selects a subgraph H by visually selecting nodes and edges with a simple and intuitive drag-of-the-mouse/gesture. From this selected subgraph (in real-time), a local RSM model is learned for H; and the subgraph statistics and accuracy are reported in the red side panel on the left side. Node color represents the class label (terrorist/normal), whereas the link color encodes the number of 3-star graphlet patterns centered at each edge (both can be adapted in real-time by the user). The weight (or strength) of the nodes and links represent the local max k-core number and triangle counts, respectively.

In Figure 5, we investigate the impact of RSM using the different classes of features. The default hyperparameters were used in Figure 5. In particular, $\sigma = 0.01$, and α, β are set to 0.5. We make three important observations from Figure 5. First, RSM-RCI that uses both collective inference (CI) and relational inference (RI) always improves over RSM-CI and RSM-RI. Second, RSM-CI, RSM-RI, and RSM-RCI are most useful for graphs with only a few known

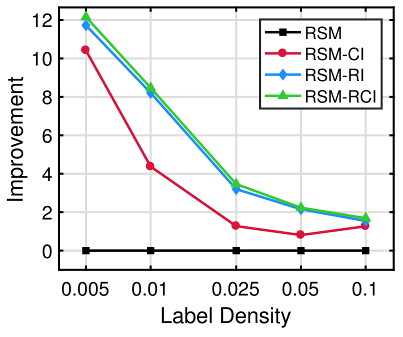


Figure 5. Impact of different classes of features on classification using soc–TerrorRel. Accuracy (percent) improvement of RSM-CI, RSM-RI, and RSM-RCI over a basic variant of RSM. See text for discussion.

labels (sparsely labeled graphs). In particular, we see that the improvement over RSM is largest when label density of the graph is the smallest (label density=0.005). Third, we observe in Figure 5 that RSM-RI generally leads to the largest improvement over RSM compared to RSM-CI.

Table IV
IMPACT OF NODE AND FEATURE-WISE NORMALIZATION.

		NODE-WISE NORMALIZATION			
		None	**L1**	**L2**	**min-max**
FEATURE-WISE NORMALIZATION	**None**	0.102	0.754	0.688	0.684
	L1	0.526	**0.871**	0.848	0.841
	L2	0.747	0.846	0.845	0.853
	min-max	0.859	0.855	0.864	0.859

H. *Impact of Node and Feature-wise Normalization*

In this section, we investigate the impact of normalization on multiclass classification by varying both the feature-wise and node-wise normalization technique used in RSM. Table IV reports the F1 scores for DD68 with $|\mathcal{C}| = 20$ class labels. We observe in Table IV that the normalization technique may significantly impact predictive performance. Overall, the best F1 score is obtained when L1 is used for both feature-wise and node-wise normalization. Note that L1 performs reasonably well on other graphs as well. However, there are some graphs where other normalization schemes may result in significant improvement over L1. Furthermore, we have also observed graphs where different feature-wise and node-wise normalization techniques give the best classification performance.

I. *Varying Outer (SSL) Iterations*

We also investigate the impact of varying the number of outer/SSL iterations of RSM for two different networks. Results are provided in Figure 6. Note the same hyperparameters, normalization schemes, and similarity functions were used for consistency and comparison. Both experiments use RSM-RBF, though similar behavior was also observed with RSM-LINEAR and other polynomial similarity machines. In particular, we observe that the improvement in accuracy generally increases rapidly as a function of the number of outer iterations, then begins to stabilize.

V. Related Work

Statistical Relational Learning: Relational and graph-based machine learning has become increasingly important due to the recent proliferation and ubiquity of network data [4]. However, existing SRL methods such as Relational Dependency Networks (RDNs) [27] and Probabilistic Soft Logic (PSL) [28] are inefficient, perform poorly, and/or do not scale to graphs that are either large or streaming [4]. In contrast, RSM is efficient in terms of time and space, while

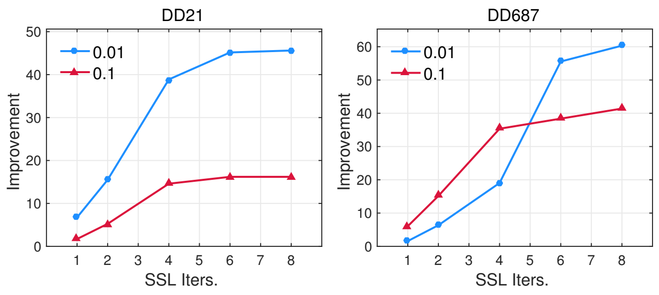


Figure 6. Accuracy (percent) improvement over label propagation as the number of outer/SSL iterations increases.

also providing high accuracy for a variety of classification tasks. Another key difference is that RSM avoids many of the unrealistic assumptions made by existing methods which have greatly limited the use of these techniques in practice. For instance, past work has focused primarily on network data that has significant levels of relational autocorrelation (homophily) [4], [6], [7], [23]. These methods perform well only when such high levels of relational autocorrelation are present and fail otherwise. This assumption is often violated in the noisy data found in many real-world settings. Moreover, in cases where there does not exist high autocorrelation in the data, then relational learning methods may actually be less accurate than traditional *iid* machine learning methods. In contrast, RSM is flexible and general enough for prediction in networks with both homophily and heterophily. Additionally, one may also adapt RSM for use with relational active learning [29] and active search methods [30]. Moreover, RSM is flexible with many interchangeable components, and thus may use methods from collective classification [31], [9], relational classification [23], [32], and relational representation learning [6] for improving performance in some situations.

Semi-Supervised Learning (SSL): While pure supervised learning (SL) techniques use only labeled data for training, semi-supervised learning (SSL) techniques also use unlabeled data for training [17], [33]. The MultiRankWalk (MRW) proposed by Li *et al.* uses random walks to estimate the class labels of the unlabeled nodes [34]. An approach similar to WVRN and MRW was proposed by Wang *et al.* which estimates the node class using a probabilistic approach [35]. Most of these techniques are based on simple belief propagation [36], are inefficient, lack support for attributes, do not leverage a parameterized similarity function, and have many other differences and disadvantages. In contrast, RSM is efficient, parallel, flexible, leverages a parameterized similarity function, naturally handles large attributed graphs, and supports the full spectrum of relational dependencies including homophily, heterophily, and dependencies between these two extremes. Unlike other graph-based SSL methods, RSM also supports supervised learning only without semisupervision (*i.e.*, turning off Lines 19-25 in Alg. 1 and setting $\tau_{\max} = 1$).

Learning in Graph Streams: Most existing work on graph streams has focused on estimating structural properties such as triangles or more generally graphlets [37]. There has also been some work on graph classification [38] and ranking nodes in graph streams [39]. In contrast, RSM naturally supports classification in graph stream data. It is also fast and efficient for classifying new nodes that arrive or become active at a particular time. For a single edge insertion or deletion, observe that the updates required by RSM are localized. The localized updates are efficient to compute and can be viewed as a local approximation. In the simplest case, RSM can classify a new node by considering only localized information such as the nodes adjacent to it; however in the general case RSM also considers a sample of unrelated nodes from the various classes. Moreover, RSM can also naturally leverage both the graph structure and any existing attributes.

Interactive ML: Existing work has focused mainly on traditional machine learning problems that are limited to independent and identically distributed (IID) data [40]. In addition, Oglic *et al.* propose interactive kernel PCA [41]. Kapoor *et al.* [42] use a human-assisted optimization strategy in the design of multiclass classifiers for *iid* data. In contrast, this work demonstrates the utility of RSM for the interactive relational learning task; RSM is integrated into a visual graph mining and learning platform and shown to be effective for a variety of interactive relational learning tasks. Visual analytic methods are becoming increasingly important [43], [44], [45], [46] and have been deployed for numerous real-world applications, including maritime risk assessment [47], astrophysics [48], financial planning [49], law enforcement [50], and many others [51], [52], [53]. Majority of existing work in visual analytics has largely centered around visualization and human computer interaction (HCI) techniques. However, some work has investigated combining traditional ML methods with visual analytics (e.g., interactive PCA [54], classification via supervised dimension reduction [55]). In contrast, we demonstrate the effectiveness of RSM by combining it with visual analytic techniques for real-time interactive relational learning with visualization and easy-to-use interaction techniques.

VI. Conclusion

We have described a general similarity-based relational learning framework called *relational similarity machines* (RSM) for graph data based on the notion of maximum similarity. Unlike previous work, RSM is designed to be fast and scalable for real-time interactive RML. The experiments demonstrated the effectiveness and efficiency of RSM on a wide variety of graph data. The RSM framework has all the following desired properties: (a) it naturally supports interactive RML, (b) it is space- and time-efficient, (c) has excellent predictive accuracy, (d) generalizes to graphs with homophily and heterophily, and (e) flexible with many interchangeable components.

References

[1] N. K. Ahmed, J. Neville, and R. Kompella, "Space-efficient sampling from social activity streams," in *SIGKDD BigMine*, 2012, pp. 53–60.

[2] P. Mahadevan, D. Krioukov, M. Fomenkov, X. Dimitropoulos, A. Vahdat *et al.*, "The Internet AS-level Topology: three data sources and one definitive metric," *SIGCOMM*, vol. 36, no. 1, pp. 17–26, 2006.

[3] D. S. Bassett and E. Bullmore, "Small-world brain networks," *The Neurosci.*, vol. 12, no. 6, pp. 512–523, 2006.

[4] L. Getoor and B. Taskar, Eds., *Intro. to SRL.* MIT Press, 2007.

[5] L. De Raedt and K. Kersting, *Prob. Induc. Logic Prog.* Springer, 2008.

[6] R. A. Rossi, L. K. McDowell, D. W. Aha, and J. Neville, "Transforming graph data for statistical relational learning," *JAIR*, vol. 45, 2012.

[7] S. A. Macskassy and F. Provost, "Classification in networked data: A toolkit and a univariate case study," *JMLR*, vol. 8, pp. 935–983, 2007.

[8] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer, "Learning probabilistic relational models," in *IJCAI.* Springer-Verlag, 1999, pp. 1300–1309.

[9] L. K. McDowell, K. M. Gupta, and D. W. Aha, "Cautious collective classification," *JMLR*, vol. 10, pp. 2777–2836, 2009.

[10] L. McDowell, K. Gupta, and D. Aha, "Meta-prediction for coll. classif." in *FLAIRS*, 2010.

[11] J. Neville, D. Jensen, and B. Gallagher, "Simple estimators for relational Bayesian classifers," in *ICDM*, 2003.

[12] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Mag.*, vol. 29, no. 3, 2008.

[13] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Ann. Rev. of Soc.*, vol. 27, no. 1, 2001.

[14] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, "Graph kernels," *JMLR*, vol. 11, pp. 1201–1242, 2010.

[15] E. J. Gardiner, P. Willett, and P. J. Artymiuk, "Graph-theoretic techniques for macromolecular docking," *Journal of Chemical Information and Computer Sciences*, vol. 40, no. 2, pp. 273–279, 2000.

[16] R. A. Rossi and N. K. Ahmed, "Role discovery in networks," *Transactions on Knowledge and Data Engineering*, vol. 27, no. 4, pp. 1112–1131, April 2015.

[17] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synthesis Lectures on AI and ML*, vol. 3, no. 1, pp. 1–130, 2009.

[18] R. Rossi and R. Zhou, "Toward interactive relational learning," in *AAAI*, 2016, pp. 4383–4384.

[19] Y.-W. Chang, C.-J. Hsieh, K.-W. Chang, M. Ringgaard, and C.-J. Lin, "Training and testing low-degree polynomial data mappings via linear svm," *Journal of Machine Learning Research*, vol. 11, no. Apr, pp. 1471–1490, 2010.

[20] Y. Bengio, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[21] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[22] R. A. Rossi and N. K. Ahmed, "An interactive data repository with visual analytics," *SIGKDD Explor.*, vol. 17, no. 2, pp. 37–41, 2016. [Online]. Available: http://networkrepository.com

[23] J. Neville, D. Jensen, L. Friedland, and M. Hay, "Learning relational probability trees," in *SIGKDD*, 2003.

[24] N. Shervashidze, S. Vishwanathan, T. Petri *et al.*, "Efficient graphlet kernels for large graph comparison," in *AISTATS*, 2009, pp. 488–495.

[25] L. Liu, "Hierarchical learning for large multi-class network classification," in *ICPR*, 2016, pp. 2307–2312.

[26] L. K. McDowell and D. W. Aha, "Labels or attributes? Rethinking the neighbors for collective classification in sparsely-labeled networks," in *CIKM*, 2013, pp. 847–852.

[27] J. Neville and D. Jensen, "Relational dependency networks," *Journal of Machine Learning Research*, vol. 8, no. Mar, pp. 653–692, 2007.

[28] A. Kimmig, S. Bach, M. Broecheler, B. Huang, and L. Getoor, "A short introduction to probabilistic soft logic," in *NIPS Workshop*, 2012.

[29] M. Bilgic, L. Mihalkova, and L. Getoor, "Active learning for networked data," in *ICML*, 2010.

[30] X. Wang, R. Garnett, and J. Schneider, "Active search on graphs," in *SIGKDD*, 2013, pp. 731–738.

[31] M. Bilgic, G. M. Namata, and L. Getoor, "Combining collective classification and link prediction," in *ICDM Workshops*, 2007.

[32] R. A. Rossi and J. Neville, “Modeling the evolution of discussion topics and communication to improve relational classification,” in *SOMA*, 2010.

[33] X. Zhu, Z. Ghahramani, J. Lafferty *et al.*, “Semi-supervised learning using gaussian fields and harmonic functions,” in *ICML*, vol. 3, 2003.

[34] F. Lin and W. W. Cohen, “Semi-supervised classification of network data using very few labels,” in *ASONAM*, 2010, pp. 192–199.

[35] Z. Wang, F. Yin, W. Tan, and W. Xiao, “Classification in networked data with heterophily,” *The Scientific World Journal*, 2013.

[36] J. Pearl, “Reverend bayes on inference engines: A distributed hierarchical approach,” in *AAAI*, 1982, pp. 133–136.

[37] N. K. Ahmed, J. Neville, and R. Kompella, “Network sampling: From static to streaming graphs,” *TKDD*, vol. 8, no. 2, pp. 1–56, 2014.

[38] C. C. Aggarwal, “On classification of graph streams,” in *SDM*, 2011.

[39] J. O’Madadhain and P. Smyth, “EventRank: A framework for ranking time-varying networks,” in *LinkKDD Workshop*, 2005, pp. 9–16.

[40] J. A. Fails and D. R. Olsen Jr, “Interactive machine learning,” in *IUI*, 2003, pp. 39–45.

[41] D. Oglic, D. Paurat, and T. Gärtner, “Interactive knowledge-based kernel pca,” in *PKDD*, 2014.

[42] A. Kapoor, B. Lee, D. S. Tan, and E. Horvitz, “Performance and preferences: Interactive refinement of machine learning procedures.” in *AAAI*, 2012.

[43] J. Kielman, J. Thomas, and R. May, “Foundations and frontiers in visual analytics,” *Info. Vis.*, vol. 8, no. 4, p. 239, 2009.

[44] D. A. Keim, J. Kohlhammer, G. Ellis, and F. Mansmann, *Mastering the information age-solving problems with visual analytics*, 2010.

[45] R. Arias-Hernández, J. Dill, B. Fisher, and T. M. Green, “Visual analytics and human-computer interaction,” *Interact.*, vol. 18, no. 1, pp. 51–55, 2011.

[46] G. Robertson, D. Ebert, S. Eick, D. Keim, and K. Joy, “Scale and complexity in visual analytics,” *Info. Vis.*, vol. 8, no. 4, pp. 247–253, 2009.

[47] A. Malik, R. Maciejewski, B. Maule, and D. S. Ebert, “A visual analytics process for maritime resource allocation and risk assessment,” in *VAST*, 2011.

[48] C. R. Aragon, S. S. Poon, G. S. Aldering, R. C. Thomas, and R. Quimby, “Using visual analytics to develop situation awareness in astrophysics,” *Info. Vis.*, vol. 8, no. 1, pp. 30–41, 2009.

[49] A. Savikhin, H. C. Lam, B. Fisher, and D. S. Ebert, “An experimental study of financial portfolio selection with visual analytics for decision support,” in *HICSS*, 2011.

[50] A. Malik, R. Maciejewski, T. F. Collins, and D. S. Ebert, “Visual analytics law enforcement toolkit,” in *HST*, 2010, pp. 222–228.

[51] S. Kim, Y. Jang, A. Mellema, D. S. Ebert, and T. Collins, “Visual analytics on mobile devices for emergency response,” in *VAST*, 2007, pp. 35–42.

[52] N. Andrienko and G. Andrienko, “A visual analytics framework for spatio-temporal analysis and modelling,” *DMKD*, vol. 27, no. 1, pp. 55–83, 2013.

[53] M. Sedlmair, P. Isenberg, D. Baur, M. Mauerer, C. Pigorsch, and A. Butz, “Cardiogram: visual analytics for automotive engineers,” in *SIGCHI*, 2011, pp. 1727–1736.

[54] D. H. Jeong, C. Ziemkiewicz, B. Fisher, W. Ribarsky, and R. Chang, “iPCA: An Interactive System for PCA-based Visual Analytics,” in *C. Grap.*, vol. 28, no. 3, 2009, pp. 767–774.

[55] J. Choo, H. Lee, J. Kihm, and H. Park, “ivisclassifier: An interactive visual analytics system for classification based on supervised dimension reduction,” in *VAST*, 2010.