Class-aware Tensor Factorization for Multi-relational Classification

Georgios Katsimpras, Georgios Paliouras

Ag. Paraskevi, Athens

NCSR Demokritos[1,1]

Abstract

In this paper, we propose a tensor factorization method, called CLASS-RESCAL, which associates the class labels of data samples with their latent representations. Specifically, we extend RESCAL to produce a semi-supervised factorization method that combines a classification error term with the standard factor optimization process. CLASS-RESCAL assimilates information from all the relations of the tensor, while also taking into account classification performance. This procedure forces the data samples within the same class to have similar latent representations. Experimental results on several real-world social network data indicate this is a promising approach for multi-relational classification tasks.

Keywords: semi-supervised tensor factorization, multi-relational networks, social network analysis

1. Introduction

Social network analysis (SNA) provides new ways of answering standard social and behavioral science research questions. Analyzing social networks typically involves tasks like node centrality prediction, node classification and link prediction. However, social networks usually comprise heterogeneous information, a variety of node types, multiple relations between them and complex

Email address: gkatsibras@iit.demokritos.gr, paliourg@iit.demokritos.gr (NCSR Demokritos)

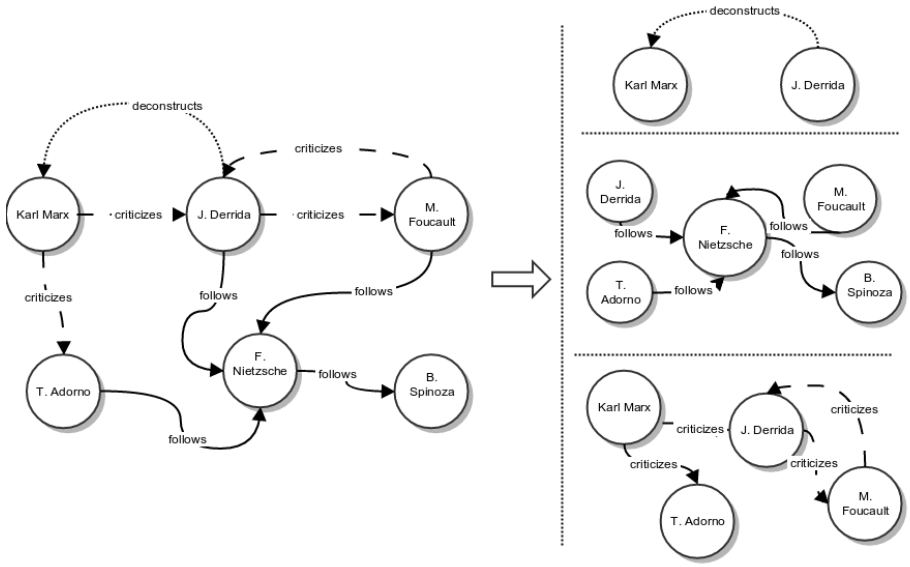


Figure 1: An example of a multi-relational network. Here, we observe three different types of interactions/relations between nodes (right).

structures. A category of such complex networks are the multi-relational ones. In multi-relational social networks, a relation represents a social tie or some type of interaction between two nodes. In this sense, a multi-relational network integrates multiple single relational networks. For a toy example of a multi-relational network see Figure 1. Nodes represent entities and edges the existing types of interactions. Here, the network displays three different kinds of relations *(deconstructs, criticizes, follows)*.

Given the proliferation of complex multi-relational social networks, it becomes imperative to design new analysis methods that will capture the diversity of information in the network. The use of simpler traditional graph analysis methods inevitably leads to partial and incomplete conclusions.

In this work, we focus on the problem of identifying the type/class of nodes in multi-relational social networks through their social interactions. The underlying assumption is that the interactions between two nodes largely depend on their positions in the social network and the roles they play [8]. Hence, find-

ing nodes of the same class can be modeled as a task of identifying nodes that have similar relationships [38]. In many cases there is additional information about nodes beyond their relationships and interactions. For example, in the scenario of Figure 1 we know that some of the philosophers are post-modernists while others are existentialists. Such information can facilitate the social modeling process, especially in classification tasks. However, in many real-world applications only a small portion of the data is accompanied with such labels.

The research problem studied in this paper can be defined as follows: we want to extract informative latent representations of nodes out of all information available in the network. Through this process, we aim to identify the type/class of a node (e.g. a user) in the graph. Solving this task raises the following research questions:

(Q1) How can we combine information from multiple relations in the network?

(Q2) How can we incorporate the label information of the data in hand to enhance the analysis?

(Q3) How can we effectively integrate the two sources of knowledge to identify the type/class of a node?

Several approaches to multi-relational network analysis have appeared in the literature, such as multilinear PageRank [11] and the multi-relational version of hubs and authorities [21]. Tensors and their factorizations have also been used widely. Due to the nature of tensors, relations expressed as tuples of the form *(object, relation, subject)*, for instance *(Derrida, criticizes, Foucault)*, can be straightforwardly mapped to a tensor. As an example, the multi-relational network in Figure 2 can be represented as a 5x5x3 tensor where an entry (i,j,r) is 1 only if the corresponding relation exists in the network. This process is depicted in Figure 2. Rows represent the nodes and each slice/matrix represents the adjacency matrix for a distinct relation. Therefore, tensors can be used to address the first question (Q1).

However, in many cases there is additional information about nodes apart from relationships and interactions. One way of making use of such information

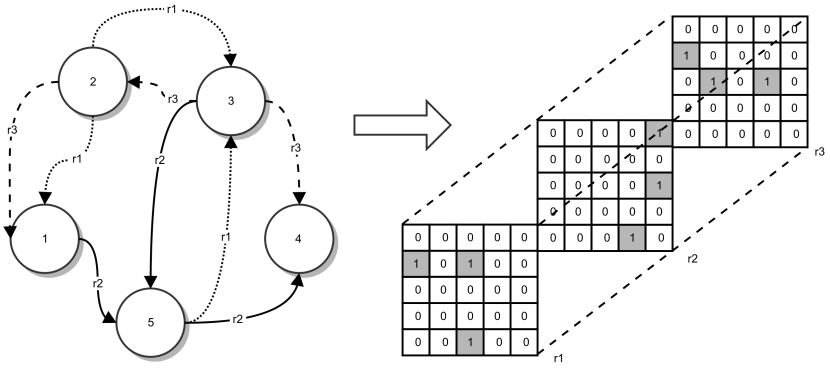


Figure 2: A multi-relational network can be easily mapped to a tensor. As we can see, the network at the left can be mapped to a 5x5x3 tensor. Each frontal slice represents the adjacency matrix for a specific relation

is to extract node vectors from the tensor (e.g. using factorization) and then append to the vectors this extra information that is available. The extended vectors could then be used to train a classifier to identify the class of a node. The problem with this approach is that the first (factorization) step remains agnostic of the final (e.g. classification) task. Therefore, the factors are not optimized for the node classification task.

In order to address this problem, we propose a new method for ranking nodes in multi-relational networks, which combines tensor factorization and classification in a joint optimization process. Similar to TripleRank [9], our idea is to produce a ranking of nodes, based on the factor scores that are computed during factorization. However, in our approach the factor matrices are a result of a new semi-supervised factorization process, which forces nodes that belong to the same class to obtain a similar factor representation. The resulting class-biased factor matrices can thus be used for node ranking/classification. In this manner, we tackle questions (Q2) and (Q3).

The method proposed in this paper (CLASS-RESCAL) is an enhanced version of RESCAL tensor factorization [31], that assimilates a-priori information about nodes. The factorization process of CLASS-RESCAL incorporates a com-

ponent that minimizes classification error. We evaluate the proposed method using three real-wold datasets that have been used in existing literature. The first was collected from Twitter and consists of a network of 4,317 users and two types of relations [35], the second was collected from Tagged.com and consists of a network of 5,317,649 users and seven types of relations [7] and the third was collected from Flickr and contains 8,536 photos and seven types of relations [20].

In summary, the main contributions of the paper are:

- Combination of tensor factorization with classification, achieving class-aware factorization of tensorial data.
- Demonstration of the benefits of using the proposed method to rank social network users according to their class, using real-world datasets.

The remainder of the paper is organized as follows. In Section 2, we provide the notation we use throughout the paper and review tensor factorization models which are commonly used in multi-relational learning. In Section 3, we formalize the task and present in detail our approach. In Section 4, we experimentally evaluate CLASS-RESCAL and compare it with competitive classification and factorization methods. Finally, we conclude with a discussion on our observations and directions for future research in Section 5.

2. Related Work

2.1. Notation

Throughout the paper we will use the following notation. The uppercase calligraphic letters denote a tensor $\mathcal{T} \in \mathbb{R}^{n \times n \times m}$, where n is the number of nodes (e.g. users) and m is the number of relations. Matrices are represented by uppercase italic letters like A. Lowercase bold letters, like $\mathbf{v}$, denote vectors. The *(i,j)* element of a matrix A is denoted by a_{ij}. To refer to the *i-th* row of a matrix A we use $\mathbf{a_i}$. Similarly, an element *(i,j,k)* of a tensor $\mathcal{T}$ will be denoted

as $\mathcal{T}_{ijk}$. Also, $\mathcal{T}_k$ represents the k-th frontal slice of tensor $\mathcal{T}$. Additionally, $vec(A)$ is the vectorization of A and the operator $\otimes$ is the Kronecker product.

For our relational modeling of triples of the form *(object, relation, subject)*, we use a binary representation:

$$\mathcal{T}_{ijk} = \begin{cases} 1 & \text{if the k-th relation exists between nodes i and j} \\ 0 & \text{otherwise} \end{cases}$$

This representation is also known as an adjacency tensor. As an example, the relation *(u1, r2, u5)* in Figure 2, causes the corresponding entry in the tensor to be 1, i.e. $\mathcal{T}_{1,5,2} = 1$. In this sense, each frontal slice of $\mathcal{T}$ represents an adjacency matrix for a particular relation.

The *order* of a tensor, also known as *ways* or *modes*, is the number of its dimensions, therefore, $\mathcal{T}$ is called a third-order tensor. Also, a r-order tensor is of rank-one if it can be strictly decomposed into the outer product of r vectors. In addition, to simplify the notation, we follow the same notation as in [19] for Kruskal and Tucker operators[19], to express the factorization models in the next sections.

2.2. Multi-Relational Learning with Tensor Factorization

As multi-relational data can be efficiently represented by tensors, many approaches have been proposed in the literature that employ tensor factorization. Common tensor factorization approaches such as CANDECOMP/PARAFAC (CP) [13], Tucker factorization [39] and DEDICOM [3] have been utilized widely. Most of these studies use a common latent representation for both entities and relations. CP decomposes a tensor into a sum of rank-one tensors. The CP decomposition of a third-order tensor, is computed by the following least-squares loss:

$$\min_{A,B,C} \|\mathcal{T} - [\![A, B, C]\!]\|^2,$$

where A, B and C are the separate factor matrices.

On the other hand, Tucker [39] decomposes a tensor into a core tensor and separate factor matrices for each mode of the tensor. The optimization problem for Tucker takes the following form:

$$\min_{\mathcal{G},A,B,C} \|\mathcal{T} - [\![\mathcal{G}; A, B, C]\!]\|^2,$$

where $\mathcal{G}$ is the core tensor and A, B, C the separate factor matrices.

Another approach for tensor factorization, which achieves high predictive performance in the task of link prediction, is RESCAL [31]. RESCAL, which we will describe in more detail in Section 3, uses a unique latent representation for nodes. The minimization problem for RESCAL is the following:

$$\min_{A,R} \left\|\mathcal{T} - ARA^T\right\|^2.$$

ConsMRF [5] follows a similar approach, but builds a separate model for each relation. ConsMRF uses Alternate Direction Method of Multipliers (ADMM) for optimization [23, 4].

Recently, several studies have combined matrix and tensor factorization in a joint model. These techniques proved to achieve good performance when there is additional information available (e.g. the fact that Derrida and Foucault are post-modernists). Such a joint factorization method based on CP is proposed in [2]. The matrix is used to hold the available additional information about the nodes. The minimization problem is formulated as:

$$\min_{A,B,C,V} \|\mathcal{T} - [\![A, B, C]\!]\|^2 + \left\|Y - AV^T\right\|^2,$$

where Y is the additional information matrix and V is the corresponding factor matrix.

Similar approaches are presented in [29, 6, 34]. Lin et al.[24] also discussed coupled matrix and tensor factorizations using KL-divergence. They model higher-order tensors by fitting a CP model. In contrast to these studies that use alternating algorithms for optimization, Acar et al. [1] proposed an all-at-once optimization approach for coupled analysis.

The goal of our work is to incorporate classification error in the optimization process. In this direction, very few tensor factorization methods obtain the factor matrices in a supervised manner, that is, by utilizing the label information of the samples. Most of these methods, perform the decomposition in the usual unsupervised way and then build a classifier using the resulting factor matrices (e.g. [22]). Therefore, factorization remains agnostic to the labels of the samples. Differently from these methods, our approach takes into account the classification error, during the decomposition process.

2.3. Identifying Important Nodes in Multi-relational Networks

The problem of identifying important nodes has been extensively studied in the literature, albeit mostly in single-relation networks. For instance, PageRank assigns a score to each node that expresses its importance. Many studies propose centrality measures such as in-degree, betweenness or variations of PageRank-like algorithms [18, 14, 33, 26]. While most of these methods refer to single-relation networks, they can be extended to multi-relational ones. For example, the study in [11] is an extension of PageRank to multi-relational networks. In [25], the authors propose a graphical model which utilizes heterogeneous link information and the textual information associated to each node. In [41], the authors suggest a method for performing combined random walks exploiting multi-relational influence networks.

MultiRank [30] is a method to simultaneously determine the importance of both nodes and relations, based on a probability distribution computed from multi-relational data. HAR [21] calculates hub, authority and relevance scores by performing random walks in multi-relational data. Both MultiRank and HAR are unsupervised methods, that use a tensorial representation of multi-relational data and perform random walks on the tensors. On the other hand, in [15] a three-stage unsupervised process is proposed, which firstly collects features of the nodes, then performs a tensor factorization on node relations and finally applies a ranking scheme to identify the most influential nodes.

Tensor factorization has also been used in the context of linked data. TripleR-

ank [9], ranks nodes in triple stores using CP [13] to obtain two factor matrices which correspond to hub and authority scores. Another approach [36], utilizes tensor factorization over heterogeneous[1] networks to rank tags and provide tag recommendations.

Our method also aims at classifying nodes in multi-relational networks. However, it differs from existing methods as it combines the decomposition of the tensor with additional supervision, that provides information for the specific type of node.

Recent work on neural networks involves also research on methods that operate on graphs. Node2vec [12], learns a mapping of nodes to a low-dimensional space of features that maximizes the likelihood of preserving network neighborhoods of nodes. The authors define a flexible biased random walk procedure, which efficiently explores diverse neighborhoods and show that the added flexibility in exploring neighborhoods is the key to learning richer representations. On the other hand, Kipf et. al [16] propose a scalable approach for semi-supervised learning on graph-structured data which is based on an efficient variant of convolutional neural networks operating directly on graphs. In this line of work, there are also other methods that study graphs with the use of neural networks, including [17, 37, 42, 10].

The method that comes closest to our approach is the one presented in [40], which performs discriminative tensor decomposition by coupling non-negative Tucker tensor factorization and a maximum margin classifier. In contrast to that method, our approach (a) utilizes RESCAL, which does not require the construction of a core tensor and (b) produces factor matrices by employing Alternating Least Squares (ALS), which has been proven to be a more efficient optimization method than gradient descent [3]. Our approach incorporates the social network structure and class label information in a joint factorization problem with shared latent factors. Combining all available information in a unified learning process allows the method to learn the classes of nodes in a shared

[1] More than one type of nodes and edges involved in the network.

latent space. This means we not only characterize users by their interactions but also identify the connection between the class labels and the social graph.

3. Semi-Supervised RESCAL (CLASS-RESCAL)

3.1. RESCAL tensor factorization

RESCAL is a state-of-the-art relational learning method, based on a variant of the DEDICOM tensor factorization. It achieves high predictive performance in various tasks such as link prediction and collective classification [31]. A basic assumption of RESCAL's model is that a node has a unique representation over all relations in the data.

RESCAL factorization is computed for each frontal slice as follows:

$$\mathcal{T}_k = AR_kA^T, \text{ for } k = 1,..,\text{M}$$

where $A \in \mathbb{R}^{n\times r}$ is a latent factor matrix that models nodes and $R_k \in \mathbb{R}^{r\times r}$ is an asymmetric latent factor matrix that models relations (n is the number of nodes and r is the number of latent factors). Each node is represented via a unique row a_i in A and the k-th (frontal slice) relation via R_k.

To compute this factorization, a regularized least squares optimization problem must be solved:

$$\min_{A,R} \quad \sum_k \left\|\mathcal{T}_k - AR_kA^T\right\|^2 \quad +\lambda_A \left\|A\right\|^2 + \lambda_R \sum_k \left\|R_k\right\|^2 \tag{1}$$

where λ_A and λ_R are regularization hyperparameters. This function can be minimized via Alternating Least Squares (ALS) [3], where the updates of A and R_k can be calculated as follows:

$$A \leftarrow [\sum_{k=1}^{M} \mathcal{T}_k AR_k^T + \mathcal{T}_k^T AR_k][\sum_{k=1}^{M} R_k A^T AR_k^T + R_k^T A^T AR_k + \lambda I]^{-1} \tag{2}$$

and

$$R_k \leftarrow (Z^T Z + \lambda I)^{-1} Z vec(\mathcal{T}_k), \tag{3}$$

where $Z = A \otimes A$. It is important to recall that during the optimization process, a unique latent representation for each node is shared over all relations. This means that the same A is used with all relations.

3.2. The CLASS-RESCAL approach

Our approach combines two basic mechanisms: the collective multi-relational learning achieved through factorization and the class-driven optimization in relation to the latent factors. Similar to RESCAL, input data is decomposed into the matrices $A \in \mathbb{R}^{n \times r}$ and $R \in \mathbb{R}^{r \times r}$. To capture class-aware information we add a classification error term, based on the latent factor A. Key points of this approach are the following:

- Similar to RESCAL, the tensor factorization uses a unique latent representation over all relations for each node, regardless of their occurrence as subjects or objects in a relation. Consequently, all relations have a determining influence on the calculation of factors in A.
- Node vectors a_i in A are linked to class labels. The similarity of nodes is assumed to be reflected in their latent-component representation in A. Entities of the same class are expected to be represented by similar latent vectors. In order to achieve this, a classifier is trained on the matrix A, and A is updated with respect to the classification error. In this manner, the similarity between nodes in A captures their similarity across multiple relations.
- A folding-in technique is employed, in order to project a new test instance to the existing latent space of A and R. This representation is then used to rank/classify the test instance.

The conceptual diagram of the proposed approach is depicted in Figure 3. The first step in the approach is to build a multi-relational graph from the raw data. In this paper, we focus on social media data, but our method can be readily applied to other domains (e.g. biology and medicine). Next step is to construct a tensor using the adjacency matrices, induced by the multi-relational graph. The result is a 3rd-order tensor. CLASS-RESCAL extends the RESCAL factorization model using supervision. That is, the 3rd-order

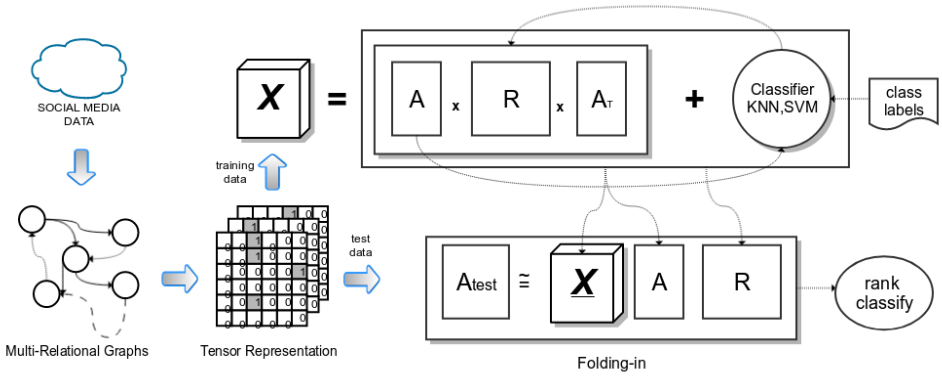


Figure 3: Conceptual diagram of CLASS-RESCAL.

tensor is approximated by taking into account the available labeled data. In particular, the node vectors in A are linked to class labels.

Regarding new nodes (test instances to be classified), CLASS-RESCAL computes new vectors by projecting their observed data onto the latent space of existing nodes (Folding-in). Given a new node's data $\underline{X}$, and using the node factor A and the relation factor R from the learned model, the method estimates the new node's vector $\mathbf{a_{test}}$. The produced vector in this way can be used for ranking or classification.

All the steps of the CLASS-RESCAL method, except tensor construction which is the same as in RESCAL (see 2.1) are described in the following subsections.

3.3. Joint optimization in CLASS-RESCAL

Suppose we have a set of labels for some of the nodes in our tensorial data. For example, we may have a label for each author in Figure 1, which indicates whether the author is influential or not. We could assume that influential authors will have similarities in the way they interact and relate with each other [27]. In other words, we expect similarly labeled nodes to share similar factors. Based on this assumption, we introduce class-label information into the tensor factorization, in order to move nodes of the same class closer in the latent space.

CLASS-RESCAL models this problem as a joint optimization of tensor factors and classification.

Given an adjacency tensor $\mathcal{T} \in \mathbb{R}^{\mathrm{n}\times\mathrm{n}\times\mathrm{m}}$, as presented in Section 2, and a set of labels $Y_i \in \{-1, 1\}$, where i = 1,..,l, with l<n, we solve the optimization problem presented in Eq. 4.

$$\underset{A,R}{\text{minimize}} \quad f(A,R) + g(A) + h(A,R), \tag{4}$$

where

$$f(A,R) = \sum_k \left\| \mathcal{T}_k - AR_kA^T \right\|^2$$

is the tensor factorization least squares problem,

$$g(A) = \lambda_g \left\| Y - y(A) \right\|^2 \tag{5}$$

is the prediction error of the classifier *y(a)* and

$$h(A,R) = \lambda_A \left\| A \right\|^2 + \lambda_R \sum_k \left\| R_k \right\|^2$$

is the regularization term. λ_g is a hyperparameter to control the influence of the classification error in the optimization. Note that the classifier $y(A)$ is produced with respect to the current values of the latent factor matrix A. In the next subsection we provide more information regarding the calculation of $y(A)$.

3.4. Classification in CLASS-RESCAL

Each row $\mathbf{a_i}$ of the factor matrix A represents a unique latent-factor representation of the i-th node. The similarity of nodes is assumed to be reflected in their latent-factor representation. Namely, nodes of the same class will also share similar latent representations. So, if the i-th node is similar to the j-th node then their rows in the factor matrix A should also be similar, i.e. $\mathbf{a_i} \approx \mathbf{a_j}$. In order to force nodes of the same class to have similar latent representation we minimize the prediction error of a classifier trained on A (Eq. 5). To further

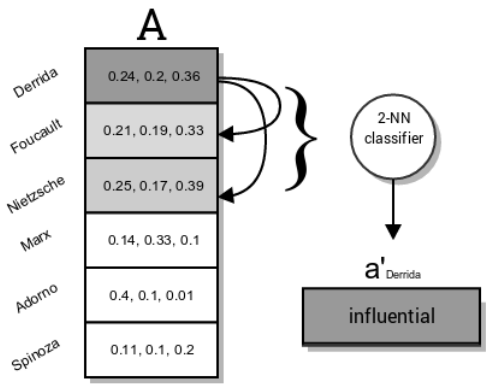


Figure 4: The process of k-NN classification. The top-k similar nodes are used for labeling, based on majority voting. Here, the top-2 neighbours of Derrida are Foucault and Nietzsche, who are labeled as influential (shaded). As a result, Derrida will also be classified as influential.

explore the role of a classifier in our method, we show how two different classifiers can be used: a lazy k-nearest neighbor (k-NN) classifier and a support vector machine (SVM).

k-NN Classifier. Nearest neighbor techniques use the similarity among nodes to construct a neighborhood and make a prediction, assuming that the class of a node is close to the class of its neighbors. For instance, an influential individual is likely to interact and share social connections with other influential individuals. In this context, we assign a label to a node in A, based on its top k neighbors in A (see Figure 4).

For example, the latent representation of Derrida is similar to the latent representations of Foucault and Nietzsche. Assuming that both Foucault and Nietzsche are influential according to supervision, the majority rule[2] will classify Derrida as influential.

SVM Classifier. We build a SVM model out of the labeled data which uses as features the latent values of each node, i.e. the factor values of each row

[2]In case of ties, the classification result will be the class that happens to appear first in the set of neighbors.

$\mathbf{a_i}$. As we want similarly labeled nodes to share similar latent representation, the discrimination of classes should rely on the factor values produced during the factorization process.

The resulting classifier, be it k-NN, SVM or a different one, is used in the optimization process (see Eq. 5) to update A and R.

3.5. Updating matrices A and R

We solve the minimization problem in Eq. 4, using the efficient alternating least squares method (ALS), as in [31]. This approach alternately fixes and solves A and R following update rules. The update rules for A and R_k are derived by setting the gradient of Eq. 4 with respect to each matrix to zero. The update rule for R_k is the same as in the original RESCAL, but the update rule for A becomes:

$$\begin{aligned} A \leftarrow & [\sum_{k=1}^{m} \mathcal{T}_k A R_k^T + \mathcal{T}_k^T A R_k + 2(Y - y(A))] \\ & [\sum_{k=1}^{m} R_k A^T A R_k^T + R_k^T A^T A R_k + \lambda I]^{-1} \end{aligned} \tag{6}$$

Note that the update of A is directly coupled to the classification result of *y(A)*. When $\frac{f(A,R)+g(A)+h(A,R)}{\|\mathcal{T}\|^2}$ converges to a predefined threshold ϵ or a maximum number of iterations is exceeded, the procedure stops. The details of the proposed method are summarized in Algorithm 1.

3.6. Folding-in a new instance

Given a new instance, we want to project it to the latent space of A and R, learned from the training data. Figure 5, illustrates the folding-in process for CLASS-RESCAL. The shaded part of $\mathcal{T}$ is the new instance which represents a new node (e.g user), $\mathcal{T}_{(new)} \in \mathbb{R}^{1 \times n \times m}$, where n is the number of nodes and m is the number of relations. Matrix R remains unchanged while A is updated in an alternating setting, exactly as in the optimization process described in Section

Algorithm 1 CLASS-RESCAL: Given a tensor $\mathcal{T}$ and a set of labels Y, approximate A and R

Require: adjacency tensor $\mathcal{T}$, labels Y

Ensure: latent matrix A, latent matrices R_k

```
Initialize A, R and hyperparameters λ_g, λ_A
repeat
    procedure UPDATE(A)
        y(A) ← (CLASSIFIER(A))
        update A using Eq. 6
    end procedure
    procedure UPDATE(R)
        for each k in k-relations do
            update R_k using Eq. 3
        end for
    end procedure
until convergence
```

3.5. Hence, the projection can be computed by the following equation:

$$A_{(new)} = [\sum_{k=1}^{m} \mathcal{T}_{k(new)} A_{(old)} R_{k(old)}^T] \\ [\sum_{k=1}^{m} R_{k(old)}^T A_{(old)}^T A_{(old)} R_{k(old)} + \lambda I]^{-1} \tag{7}$$

This procedure will produce a matrix $A_{new} \in \mathbb{R}^{1 \times r}$ where r is the number of factors[3]. As it requires just a few simple matrix operations, it is fast and hence it can be used for incorporating unseen nodes in A.

3.7. Classifying nodes

In RESCAL, the factor matrix A is a latent representation of nodes (e.g. users). Each row $\mathbf{a_i}$ contains r values, one for each latent factor. Each factor

[3]note that $\mathcal{T}_{k(new)} \in \mathbb{R}^{1 \times n}$

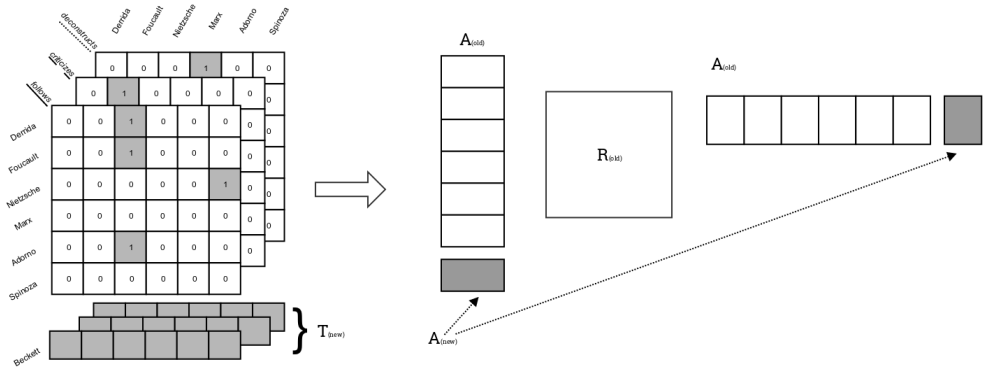


Figure 5: The *folding-in* process of CLASS-RESCAL. A new test instance T_{new}, consists of the interactions with other nodes for each relation. The result of the *folding-in* process is a new vector (A_{new}) which is the projection of T_{new} in the latent space of A and R. Note that R remains unchanged

value represents the placement of the i-th node with respect to the factor. In CLASS-RESCAL the factor matrix A is influenced also by the classifier, which forces users of the same class to obtain a similar factor representation. Consequently, in order to classify nodes (e.g. identify influential users) we can rank the rows of the factor matrix A, using a ranking score for each row $\mathbf{a_i}$. Giving an extra weight to the "positive" class by adjusting the parameter λ_g will move positive instances higher in the ranked list. The score is based on the factor values and is defined as follows

$$\mathbf{a_i^{score}} = \frac{\sum_{j=1}^{r} |a_{ij}|}{r}.$$

A new test instance is placed in the total ranking, according to the ranking score computed from its folded vector. The assumption here is that the values of $\mathbf{a_i}$ will be higher for $i \in P$, where P is the set of indexes of the positive labeled instances. This is achieved by the integration of the classification error in the update of A, which penalizes the values of negative instances. Therefore, we expect that $\mathbf{a_i^{score}}$ will be higher for the positive labeled instances.

3.8. Complexity

Following the analysis in [32], the complexity of updating A and R in RESCAL is $O(pnr + nr^2)$ and $O(pnr^2 + nr^2)$ respectively, where p are the non-zero entries, n is the number of nodes and r is the number of factors. The time complexity of adding a classifier in the update process of A, depends on the number of nodes, as well as the number and size (e.g. the number of factors) of training instances. Specifically, for the purposes of the study, adding a k-NN classifier[4] will only change the complexity of updating A to $O(pnr + nr^2 + rlog(n))$. On the other hand, adding a SVM classifier will change the complexity in the range between $O(pnr+nr^2+r(l^2))$ and $O(pnr+nl^2+r(l^3))$, depending on the implementation, where $l \ll n$ is the number of labeled nodes.

4. Experiments

In this section we perform an analysis of the proposed method and assess its performance, in terms of prediction quality and run time, compared to other related algorithms. In the following subsections we describe the datasets used in the study, the experimental settings, the evaluation protocol and the results.

4.1. Datasets

We conduct experiments in three real-world datasets. The first dataset [35] was collected from Twitter using the keyword "Amazon Kindle". It consists of 4,317 nodes (users) out of which 248 are labeled as influential. The nodes in this dataset interact with each other in two different ways representing different relations in the graph. The second dataset [7] was collected from Tagged.com and represents a sample of its social network. It consists of 5,617,345 nodes (users), who interact with each other in seven ways and 221,305 of them are labeled as spammers. The third dataset [20] was collected from Flickr and contains 8,536 nodes (images), out of which 1,877 are labeled as popular. The nodes in this dataset are linked in seven different ways.

[4] Using a kd-tree implementation (https://github.com/scikit-learn/scikit-learn)

Table 1: Overview of the datasets. E is the number of nodes, R is the number of relations, the fourth column presents the class balance, the fifth column shows the size of the dataset and the last column is the fraction of the number of interactions between entities, relative to the number of all possible interactions.

	$\|E\|$	$\|R\|$	% labels	tensor	density
Twitter	4,317	2	5.7% influentials	4317x4317x2	0.029
Tagged.com	6,733	7	43% spammers	6733x6733x7	0.002
Flickr	8,536	7	22% popular	8536x8536x7	0.039

All datasets present high sparsity. The average density of the Twitter dataset is 0.029, that of the Flickr dataset is 0.039, while that of the Tagged.com dataset is lower than 0.002. For the Tagged.com dataset we perform random subsampling, in order to achieve a better representation of the classes and resolve memory issues[5]. We randomly sample N nodes labeled as spammers and N nodes labeled as non-spammers. For these nodes we find all interactions among them that are present in the dataset. Users without interactions are discarded. Thus, we obtain a MxMx7 tensor, where M is the number of nodes left in the dataset. Table 1 summarizes the three datasets in numbers. For further details of the datasets used in this study the reader can refer to the corresponding studies, where they were first used [35, 7, 20].

4.2. Analysis of CLASS-RESCAL

Next, we examine the role of latent factors and relations in the performance of CLASS-RESCAL. For all experiments we follow a 10-fold cross-validation evaluation methodology. To assess the performance, we employ the average precision-recall curve, which is generated by varying a selection threshold and computing precision and recall successively at each point of the ranked list (as produced by CLASS-RESCAL) starting from the top of the list.

[5]We used a machine equipped with a Core i7 and 16GB Ram. That is, we can fit matrix sizes of around $1.4 * 10^{10}$ bytes.

4.2.1. Number of factors

In order to measure the effect of the number of factors on the performance of CLASS-RESCAL, we ran the experiment for various values of r = 2, 3, 5, 10, 15, 20, 30. For these experiments we used an SVM classifier and the results of this

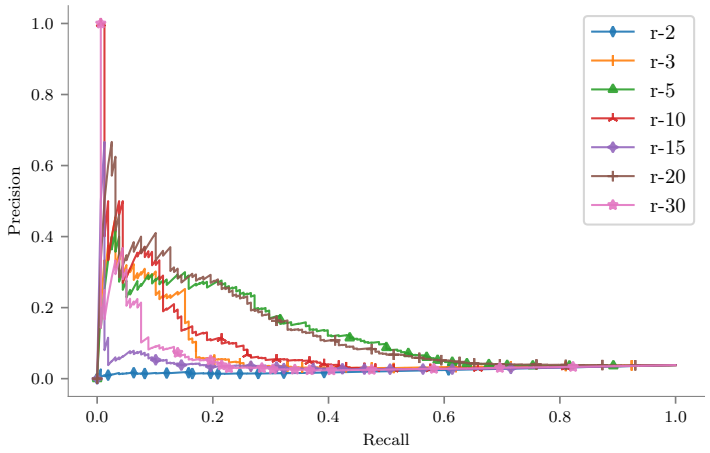


Figure 6: Behavior of CLASS-RESCAL, using different number of factors on Twitter.

experiment are displayed in Figures 6, 7 and 8. As we can see, CLASS-RESCAL

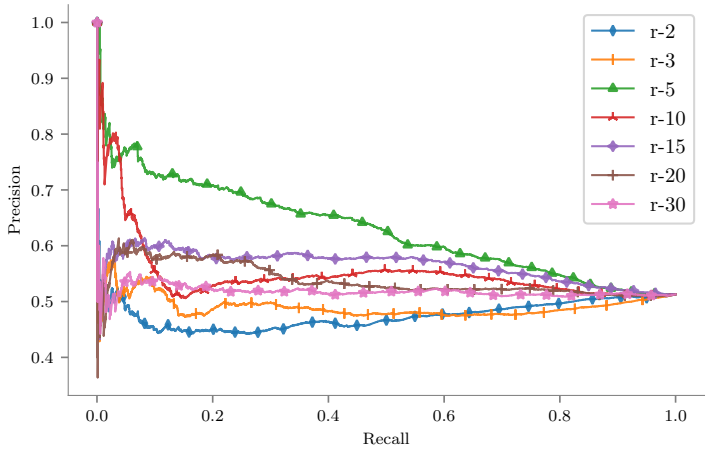


Figure 7: Behavior of CLASS-RESCAL, using different number of factors on Tagged.

performs better for r = 5, while additional factors (r > 5) do not seem to lead to significant gains and in some cases they lead to lower performance. Moreover, as

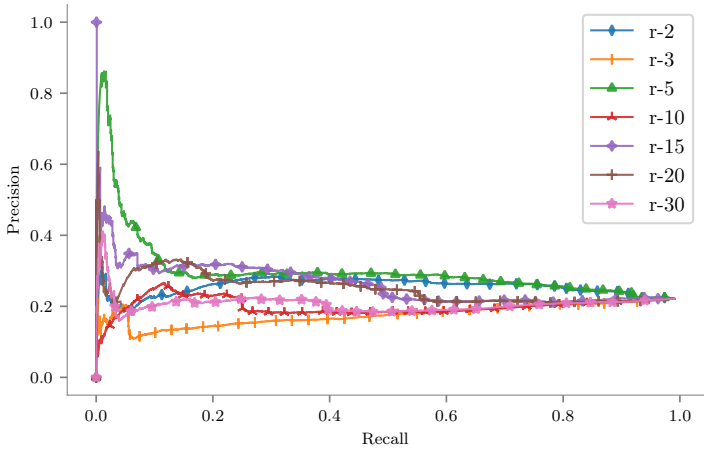


Figure 8: Behavior of CLASS-RESCAL, using different number of factors on Flickr.

the number of factors increases, the computation time of the factorization also increases. As a result, in the rest of the experiments we use r = 5 (green line), which seems to be a good compromise between performance and computational cost.

4.2.2. Number of relations.

Next, we examine the role of the number of relations (frontal slices of the tensor) in the performance of the method. In Figures 9 and 10, we show the performance of CLASS-RESCAL on the Tagged and Flickr datasets under various settings.

We omit the Twitter dataset as it has only two relations. As can be seen, the best performance is obtained when all available information (all seven relations) are used. This is an indication that the richness of multi-relational networks can lead to better results and thus, a method which exploit them should be considered.

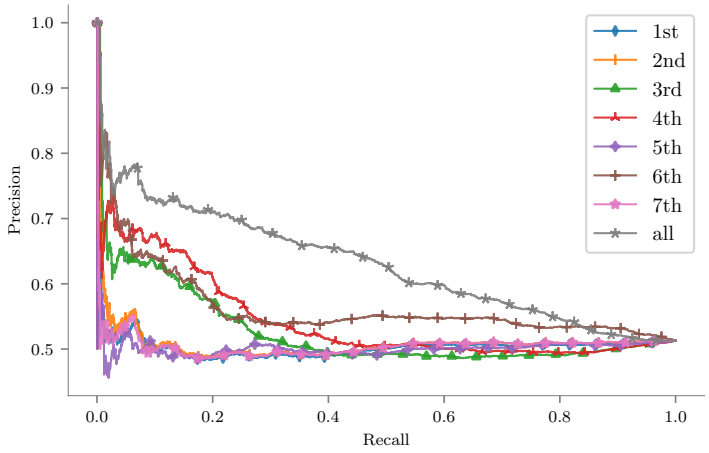


Figure 9: Performance of CLASS-RESCAL for various values of the number of relations (using 5 factors) on Tagged.

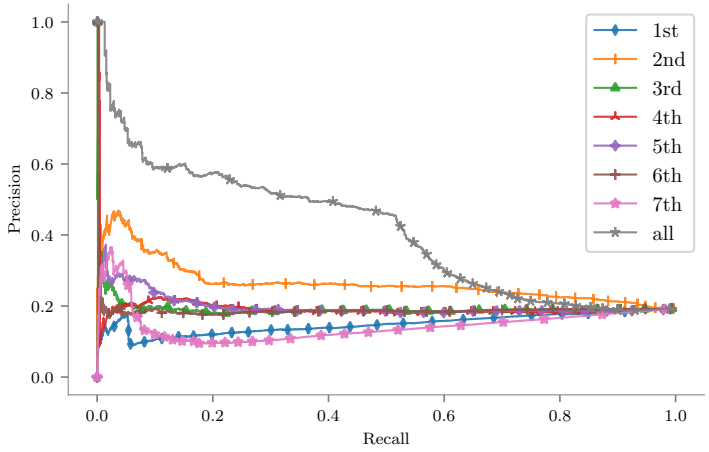


Figure 10: Performance of CLASS-RESCAL for various values of the number of relations (using 5 factors) on Flickr.

4.2.3. CLASS-RESCAL variations

As part of the analysis, we also compared the following variations of the proposed method:

- **CLASS-RESCAL-knn**: The proposed method using a k-NN classifier.

- **CLASS-RESCAL-svm**: The proposed method using a SVM classifier.
- **NN-RESCAL-svm**: A non-negative version of RESCAL, adapted to use supervision as in CLASS-RESCAL-svm.

Additionally, we include in the experiments a baseline usnupervised RESCAL and a SVM classifier (SVM-baseline), using as features the degree of a node in each relation.

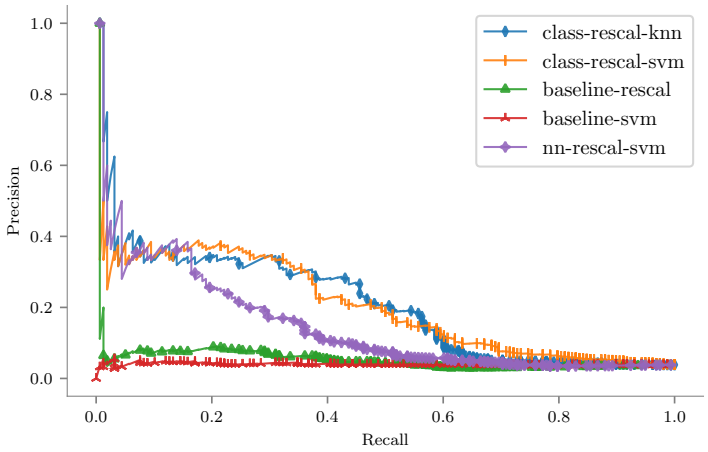


Figure 11: Performance of CLASS-RESCAL variants and a baseline classifier (using 5 factors) on Twitter.

The results in Figures 11, 12 and 13, show that CLASS-RESCAL outperforms the baselines while, CLASS-RESCAL-svm performs slightly better than the ohter variations of CLASS-RESCAL. The use of SVM in the classification procedure seems to lead to models that associate effectively the latent representation of the data samples with their class label. Hence, we use CLASS-RESCAL-svm in the remaining experiments.

4.3. Learning curve

Additionally, we investigate the behavior of CLASS-RESCAL with respect to the size of the labeled data. In Figure 14, we show the F1-score learning curves in relation to the labeled data ratio on all datasets. As we can see, the

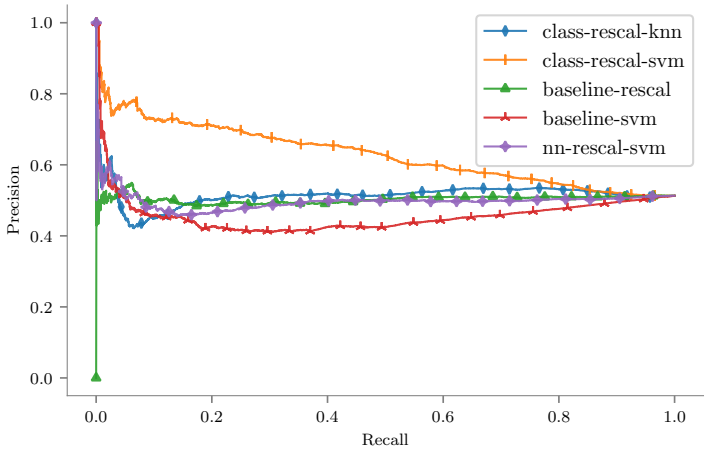


Figure 12: Performance of CLASS-RESCAL variants and a baseline classifier (using 5 factors) on Tagged.

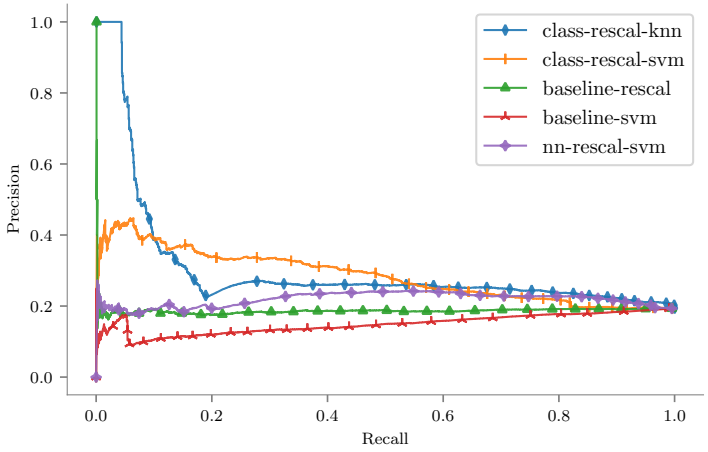


Figure 13: Performance of CLASS-RESCAL variants and a baseline classifier (using 5 factors) on Flickr.

more labeled instances are used in the training stage, the better performance is achieved.

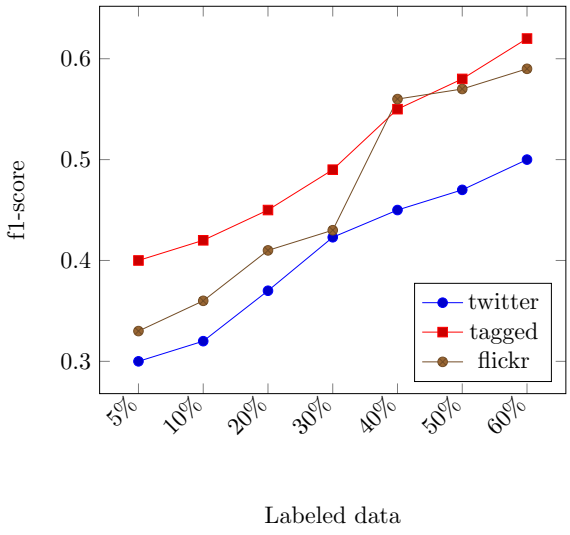


Figure 14: The learning curve of CLASS-RESCAL with respect to the number of labeled data

4.4. Comparison with other methods

In this experiment, we compare the performance of CLASS-RESCAL against other related approaches.

4.4.1. Evaluation Methodology

All datasets, used in this study, incorporate a portion of labeled data: in the Twitter dataset we have influential and non-influential users, in Tagged.com we have spammers and non-spammers, while in Flickr we have popular and non-popular items. As CLASS-RESCAL is a semi-supervised method we evaluate it over various sizes of the labeled training set, $s \in [10\%, 50\%]$. For each dataset size, we average the results over 10 random runs. In each run, we randomly sample a percentage of instances from each label to use as the training set, and consider the remaining as the test set. The training set is used to obtain the semi-supervised tensor decomposition. Then, we compute the latent representation of each test instance, using the *folding-in* technique presented in 3.6. Thereupon, we compute a ranking with respect to the ranking scores of entities, $\mathbf{a_i^{score}}$, as described in 3.7. The results reported correspond to the performance of the methods on the test set only.

4.4.2. Competing methods

We compare CLASS-RESCAL against the following methods from the literature:

- **MultiRank**[30]: MultiRank is a co-ranking method, performing a random walk over multi-relational graphs. It constructs two transition probability tensors, one for nodes and the other for relations. Based on this tensorial representation of multi-relational data, it performs a Pagerank-like random walk and computes one score for nodes and another one for relations.

- **HARrank**[21]: A multi-relational analog of the HITS algorithm. It constructs three different transition probability tensors, one for hubs, one for authorities and the last one for relations. Similar to MultiRank, it performs a random walk in the corresponding tensors and computes three different scores (hub, authority, relation).

- **TripleRank**[9]: A tensor factorization approach based on CP. It uses the factor scores produced from the tensor decomposition to rank nodes in linked data.

- **Node2vec**[12]: A node embedding method, which employs biased-random walks that preserve the structure of neighborhoods. It uses the skip-gram architecture [28] to learn the node representations.

- **GCN-AE**[17]: An unsupervised method for learning node embeddings. It uses Graph Convolutional Networks(GCN) [16] as an encoder for capturing network structural properties and a simple inner product as a decoder in the framework. It presents state-of-the-art performance in classification and link prediction tasks.

For the implementation of CLASS-RESCAL and TripleRank we used the python scikit-learn library[6]. MultiRank and HARrank were also implemented

[6] http://scikit-learn.org/stable/

in Python. We used a tensorflow implementation of Node2vec[7] and the python implementation of GCN-AE[8] provided by their authors.

4.4.3. Parameter configuration

For CLASS-RESCAL, we set the number of factors to $k = 5$ as explained in subsection 4.2.1 and hyperparameters $\lambda_g = 0.1$ and $\lambda_A, \lambda_R = 0.5$, based on the results of a grid search performed on dataset samples as in subsection 4.2. We also use $k = 5$ for TripleRank. For the other methods we use the parameter values proposed by their authors.

Furthermore, for the Node2vec and GCN-AE methods, which cannot be directly applied to multirelational data, we transform the multirelational network into a single-relational network. This is achieved by summing all slices of the third-order tensor. This results in a weighted adjacency matrix (second-order tensor).

4.4.4. Experimental Results

Figures 15, 16 and 17, show the performance of all methods on all three datasets. CLASS-RESCAL almost consistently outperforms the other methods in all datasets. The SVM-baseline yields the lowest performance as it fails to represent effectively the class labels. Triplerank and RESCAL perform better than SVM-baseline, which indicates the importance of graph structure and tensor representation. Node2vec performs slightly better than the previous tensor factorization methods, except in the Tagged.com dataset. The curves for HARrank and MultiRank are comparable across all datasets, with the former performing slightly better in the Flickr dataset and the latter being superior in Twitter. GCN-AE comes closest to the performance of CLASS-RESCAL, indicating the effectiveness of graph convolutional networks in capturing latent relations. However, the use of class-label information in the decomposition and the use of the rich multi-relational data, allows CLASS-RESCAL to outperform

[7] https://github.com/thunlp/OpenNE
[8] https://github.com/tkipf/gae

GCN-AE in all datasets. This is evident also when measuring the area under the precision-recall curve (see Table 2). A general observation is that all methods

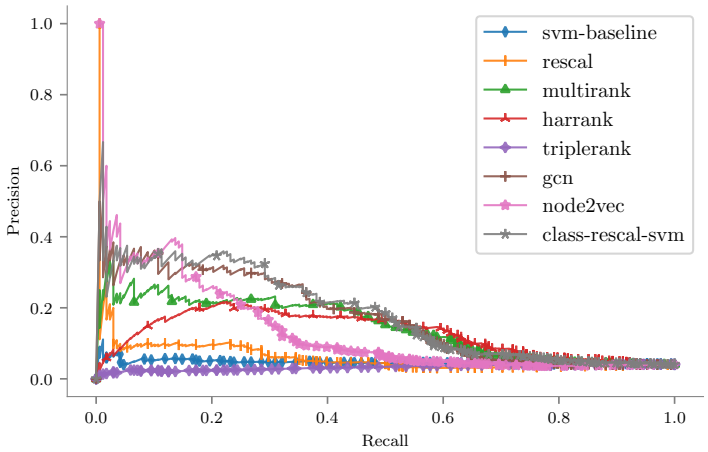


Figure 15: Performance comparison of CLASS-RESCAL against related methods on Twitter.

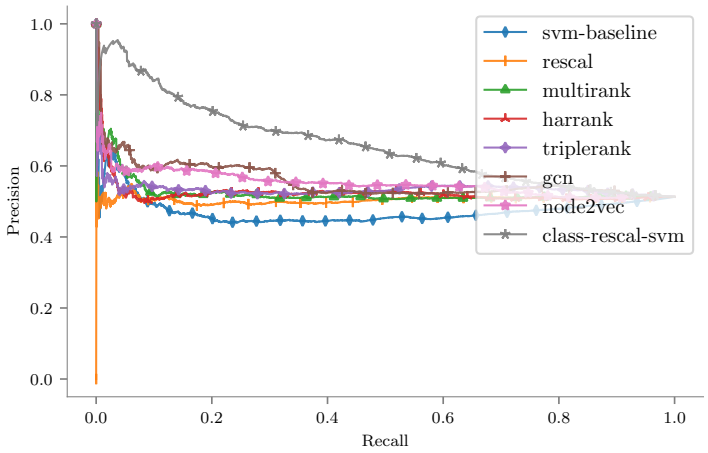


Figure 16: Performance comparison of CLASS-RESCAL against related methods on Tagged.

perform better in Tagged.com dataset. We suspect that the increasing extent of class imbalance in the data has a significant effect on the performance. In

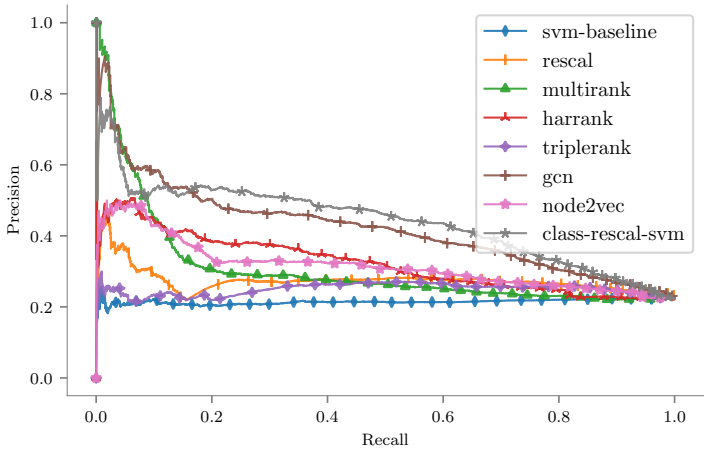


Figure 17: Performance comparison of CLASS-RESCAL against related methods on Flickr.

fact, for Flickr dataset the positive class prevalence[9] is 0.22, we observe a large drop in performance. Class imbalance seems to have a greater effect on performance in the Twitter dataset in which the positive class prevalence is 0.057. On the contrary, Tagged.com dataset presents a positive class prevalence of 0.43.

Table 2: The area under the precision-recall curve scores for all methods.

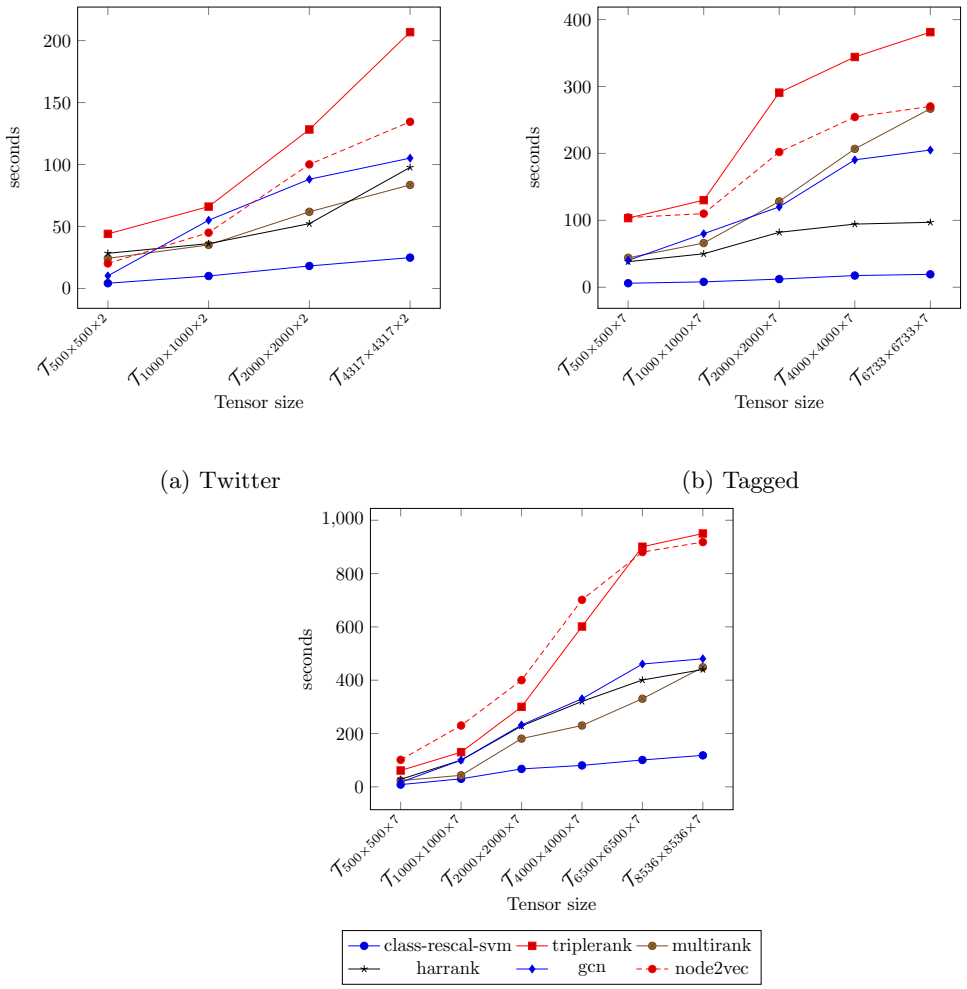
AUPR	Twitter	Tagged.com	Flickr
SVM-baseline	0.046	0.473	0.214
RESCAL	0.056	0.505	0.274
MultiRank	0.144	0.522	0.315
HARrank	0.126	0.523	0.321
TripleRank	0.032	0.533	0.249
GCN-AE	0.171	0.556	0.421
Node2vec	0.131	0.512	0.305
CLASS-RESCAL	**0.182**	**0.656**	**0.438**

[9] prevalence is defined as $\frac{numberofpositives}{numberoftotal}$

As expected the increasing positive class prevalence lead to a notable gain in performance.

4.4.5. Running Times

An empirical run-time comparison of the different multi-relational methods in producing a ranked list of nodes is depicted in Figure 18. The run-times of



(a) Twitter

(b) Tagged

(c) Flickr

Figure 18: The running times of all methods for different tensor sizes of each dataset.

all methods depend on the dimensions of the tensor and hence the size of the

dataset. As the dimensionality increases, the run-time of all methods increases. GCN-AE, MultiRank and HARrank present comparable run-times with the latter being marginally better in all datasets. The run-times of MultiRank and HARrank are influenced by the number of random walks they perform. In particular, random walks become fewer when the sparsity of the dataset is high. Hence, MultiRank and HARrank perform slightly better in the Tagged dataset which is sparser than Twitter and Flickr. However, MultiRank and HARrank require the construction of two and three tensors respectively, in contrast to CLASS-RESCAL which uses only one. This makes CLASS-RESCAL more efficient. Interestingly, the run-times of TripleRank and Node2vec are inferior to the rest of the methods. Specifically, TripleRank performance is problematic even for small datasets and becomes unfeasible as the size of the tensor increases.

The results show that CLASS-RESCAL outperforms the other methods not only in terms of prediction quality but also in run-time comparison. Therefore, our approach can be regarded as accurate and efficient.

5. Conclusion - Discussion

In this paper, we propose a semi-supervised extension of the tensor factorization method RESCAL for classifying nodes in multi-relational networks. The factorization that we propose incorporates a classification error term in the optimization process achieving class-aware modeling of the tensorial data. Therefore, our method models the tensorial data while minimizing classification error.

In order to assess the value of the new method, we conducted experiments with real-life social network data. CLASS-RESCAL outperforms standard multi-relational factorization methods by a notable margin. TripleRank, MultiRank and HARrank are limited due to their incapacity of supervision. Based solely on the structure of the networks, these methods ignore the class labels that are available. On the other hand, state-of-the-art methods Node2vec and GCN-

AE, are limited also by the fact that they ignore the richness of multi-relational networks. CLASS-RESCAL can deal with both limitations and present better performance both in terms of prediction quality, as well as in prediction runtime. As a more general conclusion, our detailed experiments demonstrate that incorporating supervision, when available, leads to more accurate models.

Several interesting future extensions of this work could be considered. As shown in this study, taking into account the rich multi-relational data results to more effective models. However, the dimensionality of such data can easily become restrictive in terms of time and memory complexity. To this end, a future direction is to investigate an efficient and parallel multi-relational factorization approach to tackle these issues. Currently, our approach does not support side information such as node or edge features. CLASS-RESCAL's learning process could be further enhanced by incorporating such data in the optimization. More than that, the current setup of CLASS-RESCAL can handle single or multi-relational networks. A possible extension of the method could consider a more complex network representation, which integrates multiple types of nodes and relations. Finally, a future plan is to apply CLASS-RESCAL to other tasks in Social Network Analysis, such as link prediction and recommendation.

References

[1] Acar, E., Kolda, T.G., Dunlavy, D.M.: All-at-once optimization for coupled matrix and tensor factorizations. CoRR (2011)

[2] Acar, E., Rasmussen, M.A., Savorani, F., Næs, T., Bro, R.: Understanding data fusion within the framework of coupled matrix and tensor factorizations. Chemometrics and Intelligent Laboratory Systems (2013)

[3] Bader, B.W., Harshman, R.A., Kolda, T.G.: Temporal analysis of semantic graphs using asalsan. IEEE (2007)

[4] Cai, X., Chen, Y., Han, D.: Nonnegative tensor factorizations using an alternating direction method. Frontiers of Mathematics in China (2013)

[5] Drumond, L., Diaz-Aviles, E., Schmidt-Thieme, L.: Multi-relational learning at scale with ADMM. CoRR (2016)

[6] Erdos, D., Miettinen, P.: Discovering facts with boolean tensor tucker decomposition. In: Proceedings of the 22nd ACM international conference on Conference on information & knowledge management, pp. 1569–1572. ACM (2013)

[7] Fakhraei, S., Foulds, J., Shashanka, M., Getoor, L.: Collective spammer detection in evolving multi-relational social networks. ACM (2015)

[8] Ferrante, J.: Sociology: A global perspective. Nelson Education (2012)

[9] Franz, T., Schultz, A., Sizov, S., Staab, S.: Triplerank: Ranking semantic web data by tensor decomposition. Springer (2009)

[10] Gao, H., Wang, Z., Ji, S.: Large-scale learnable graph convolutional networks. KDD '18 (2018)

[11] Gleich, D.F., Lim, L.H., Yu, Y.: Multilinear pagerank. SIAM Journal on Matrix Analysis and Applications (2015)

[12] Grover, A., Leskovec, J.: Node2vec: Scalable feature learning for networks. KDD '16 (2016)

[13] Harshman, R.A., Lundy, M.E.: Parafac: Parallel factor analysis. Computational Statistics & Data Analysis (1994)

[14] Hwang, H., Hristidis, V., Papakonstantinou, Y.: Objectrank: A system for authority-based search on databases. SIGMOD '06 (2006)

[15] Jingjing, W., Changhong, T., Xiangwen, L., Guolong, C.: Mining social influence in microblogging via tensor factorization approach. IEEE (2013)

[16] Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. CoRR (2016)

[17] Kipf, T.N., Welling, M.: Variational graph auto-encoders (2016)

[18] Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. J. ACM (1999)

[19] Kolda, T.G.: Multilinear operators for higher-order decompositions, vol. 2. United States. Department of Energy (2006)

[20] Leskovec, J., Krevl, A.: SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data` (2014)

[21] Li, X., Ng, M.K., Ye, Y.: Har: hub, authority and relevance scores in multi-relational data for query search. SIAM (2012)

[22] Li, Y., Ngom, A.: Non-negative matrix and tensor factorization based classification of clinical microarray gene expression data. 2010 IEEE International Conference on Bioinformatics and Biomedicine (BIBM) (2010)

[23] Liavas, A.P., Sidiropoulos, N.D.: Parallel algorithms for constrained tensor factorization via alternating direction method of multipliers. IEEE Transactions on Signal Processing (2015)

[24] Lin, Y.R., Sun, J., Castro, P., Konuru, R., Sundaram, H., Kelliher, A.: Metafac: community discovery via relational hypergraph factorization. ACM (2009)

[25] Liu, L., Tang, J., Han, J., Jiang, M., Yang, S.: Mining topic-level influence in heterogeneous networks. CIKM '10 (2010)

[26] Liu, X., Bollen, J., Nelson, M.L., Van de Sompel, H.: Co-authorship networks in the digital library research community. Inf. Process. Manage. (2005)

[27] McPherson, M., Smith-Lovin, L., Cook, J.M.: Birds of a feather: Homophily in social networks. Annual Review of Sociology (2001)

[28] Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR (2013)

[29] Narita, A., Hayashi, K., Tomioka, R., Kashima, H.: Tensor factorization using auxiliary information. Springer (2011)

[30] Ng, M.K.P., Li, X., Ye, Y.: Multirank: co-ranking for objects and relations in multi-relational data. ACM (2011)

[31] Nickel, M., Tresp, V., Kriegel, H.P.: A three-way model for collective learning on multi-relational data (2011)

[32] Nickel, M., Tresp, V., Kriegel, H.P.: Factorizing yago: Scalable machine learning for linked data. In: Proceedings of the 21st International Conference on World Wide Web, WWW '12 (2012)

[33] Nie, Z., Zhang, Y., Wen, J.R., Ma, W.Y.: Object-level ranking: Bringing order to web objects. WWW '05 (2005)

[34] Nimishakavi, M., Saini, U.S., Talukdar, P.: Relation schema induction using tensor factorization with side information. arXiv preprint arXiv:1605.04227 (2016)

[35] Rabiger, S., Spiliopoulou, M.: A framework for validating the merit of properties that predict the influence of a twitter user. Expert Systems with Applications (2015)

[36] Rendle, S., Balby Marinho, L., Nanopoulos, A., Schmidt-Thieme, L.: Learning optimal ranking with tensor factorization for tag recommendation. ACM (2009)

[37] Schlichtkrull, M., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: A. Gangemi, R. Navigli, M.E. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai, M. Alam (eds.) The Semantic Web. Springer International Publishing (2018)

[38] Tian Dai, B., Chong Tat Chua, F., Lim, E.P.: Structural analysis in multi-relational social networks. SIAM (2012)

[39] Tucker, L.R.: Some mathematical notes on three-mode factor analysis. Psychometrika (1966)

[40] Wu, F., Tan, X., Yang, Y., Tao, D., Tang, S., Zhuang, Y.: Supervised non-negative tensor factorization with maximum-margin constraint. In: AAAI (2013)

[41] Zhaoyun, D., Yan, J., Bin, Z., Yi, H.: Mining topical influencers based on the multi-relational network in micro-blogging sites. China Communications (2013)

[42] Zhuang, C., Ma, Q.: Dual graph convolutional networks for graph-based semi-supervised classification. WWW '18 (2018)