Random Walks with Variable Restarts for Negative-Example-Informed Label Propagation

by

Sean Maxwell

Submitted in partial fulfillment of the requirements

For the degree of Master of Science

Computer Science

CASE WESTERN RESERVE UNIVERSITY

May, 2023

CASE WESTERN RESERVE UNIVERSITY

SCHOOL OF GRADUATE STUDIES

We hereby approve the thesis/dissertation of

Sean Maxwell

candidate for the Master of Science degree*.

An Wang

Yinghui Wu

Mehmet Koyutürk

(date) January 3, 2023

*We also certify that written approval has been obtained for any proprietary material contained therein.

For my wife Lea and our children James and Stella.

Contents

List of Figures

Acknowledgments

I would like to thank my committee for all of their guidance: Dr. Mehmet Koyutürk, my thesis advisor and committee chair; Dr. An Wang; and Dr. Yinghui Wu. I am truly grateful to have had the opportunity to learn and work with each committee member.

This thesis would not have been possible without my family who supported me on my journey. Lea, James and Stella, you all put up with me more than I could ask for, and definitely more than I deserve.

Random Walks with Variable Restarts for

Negative-Example-Informed Label Propagation

Abstract

by

Sean Maxwell

Label propagation is frequently encountered in machine learning and data mining applications on graphs, either as a standalone problem or as part of node classification. Many label propagation algorithms utilize random walks (or network propagation), which provide limited ability to take into account negatively-labeled nodes (i.e., nodes that are known to be not associated with the label of interest). Specialized algorithms to incorporate negatively labeled samples generally focus on learning or readjusting the edge weights to drive walks away from negatively-labeled nodes and toward positively-labeled nodes. This approach has several disadvantages, as it increases the number of parameters to be learned, and does not necessarily avoid regions of the network that are rich in negatively-labeled nodes.

We reformulate random walk with restarts and network propagation to enable "variable restarts", that is the increased likelihood of restarting at a positively-labeled node when a negatively-labeled node is encountered. Based on this reformulation, we develop CusTaRd, an algorithm that effectively combines variable restart probabilities and edge re-weighting to avoid negatively-labeled nodes. To assess the performance of CusTaRd, we perform comprehensive experiments on four network datasets

commonly used in benchmarking label propagation and node classification algorithms. Our results show that CusTaRd consistently outperforms competing algorithms that learn/readjust edge weights, when negatives are available in the close neighborhood of positives.

1 Introduction

Label propagation is a commonly encountered problem in data mining and machine learning applications on network and graph-structured data [6, 20]. The problem entails assigning labels to nodes of a graph based on knowledge of the labels of a set of "seed" nodes, such that nodes that are proximate to seed nodes are assigned similar labels. Label propagation can be considered a special case of the node classification problem, in which only graph topology is used in predicting the labels of the nodes. In contrast, in the general setting for node classification, additional features are available [21].

Label propagation and machine learning on graphs: While many machine learning algorithms have been developed for semi-supervised node classification in the last few years, label propagation is often encountered as part of node classification [11]. In many cases, the set of training samples can be too small for effective learning, thus label propagation is applied prior to training more sophisticated learning algorithms [14]. In addition, emerging evidence suggests that combination of label propagation with simple models often outperforms more sophisticated models, such as graph neural networks [8]. Despite the ubiquity of label propagation in supervised learning, efforts on effectively utilizing negatively-labeled examples in label propagation have been relatively scarce.

Existing approaches to negative-example-informed label propagation: Many label propagation algorithms utilize random walks and their variants [5, 9, 24, 22, 15]. While classical random walks work with only positively-labeled examples, it has been shown that the utilization of negatively-labeled examples in training improves the accuracy of label propagation [25]. Existing approaches to informing random walks with negative examples use optimization to learn edge weights [2, 13] or restart probabilities [10, 3] that minimize flow into negatively-labeled nodes. Since the number of edges in a network is much larger than the number of nodes, the number of parameters that need to be learned is usually very large, making learning-based approaches vul-

nerable to over-fitting. In addition, the optimization problems are often non-convex and prone to getting stuck at local optima.

Our contributions: We improve negative-example-informed label propagation in two ways. Firstly, we propose a new method that *combines re-weighting of edges with variable restart probabilities during label propagation*. For this purpose, we reformulate random walks to model restarts as part of the network topology, i.e., as directed edges from any node to the positively-labeled nodes. We then use this formulation to readjust edge weights such that the flow into negatively-labeled nodes is redirected as restarts to positively-labeled nodes. The resulting algorithm, CusTaRd, utilizes negatively-labeled nodes within the random-walk/network-propagation framework and a parameter controlling the aggressiveness of redirection to reduce the flow into negatively-labeled nodes, without requiring training or optimization of a large number of parameters.

Secondly, we propose a *positive-neighborhood based approach to sampling negative examples*. This approach is motivated by two observations: 1) Propagation scores are most intense near the seed nodes, so identifying negative examples near the seeds would likely have the greatest effect, and 2) In some applications, negatively-labeled samples may be larger in number than positively-labeled examples, thus sampling of a smaller set of negative examples is usually needed. As opposed to sampling uniformly from the entire set of negative examples, we propose sampling negative examples from the close neighborhood of positive examples. We hypothesize the algorithm can better learn how to distinguish positives from negatives if it is presented negatives that are *similar* to the positives. In our experimental studies, we comprehensively investigate the merit of this approach in the context of the proposed algorithm, as well as competing algorithms.

Organization of the paper: In the next section, we define the label propagation problem and describe random walk and network propagation based algorithms for label propagation. Subsequently, we reformulate random walks to enable variable

restarts, and show how this reformulation allows readjustment of edge weights into restart probabilities. We then describe our approach to sampling negatively-labeled nodes. In Section 3, we start by describing the four datasets we use for validation, competing algorithms, and our experimental setup. We then present the comparison of the predictive accuracy of CusTaRd and competing algorithms as well as their robustness to scarcity of training examples, characterize the effect of the redirection factor on CusTaRd's performance, and comprehensively investigate the effect of the sampling of negative examples on the performance of all algorithms considered. We conclude our discussion and outline future avenues for research in Section 4.

2 Methods

2.1 Problem Definition and Existing Approaches.

Let $G = (V, E)$ denote a graph/network with node set V and edge set E. The nodes in V are associated with categorical "labels", where the nodes in subset $S_i \subset V$ are associated with label i. There may be multiple labels available, and $\mathcal{S} = \{S_1, S_2, ..., S_k\}$ denotes the set of all available label sets. This information is usually incomplete, i.e., $\cup_{i=1}^{n} S_i \neq V$. A common problem is "label prediction" which, given the labels in $\mathcal{S}$, is the task of predicting labels for the unlabeled $v \in V$. This problem is often approached using label propagation.

In label propagation, nodes $v \in S_i$ share their label information with their neighbors, who in turn share with their neighbors etc. to "propagate" node labels across the network [17, 24]. The algorithms used to propagate labels are similar to the algorithms used for network propagation, where rather than discrete valued labels, network propagation focuses on propagating continuous values such as flow or probability across a network [4]. Random walk with restarts is a commonly utilized network propagation method that simulates a random walk across the network by making frequent restarts at the nodes labeled by S_i.

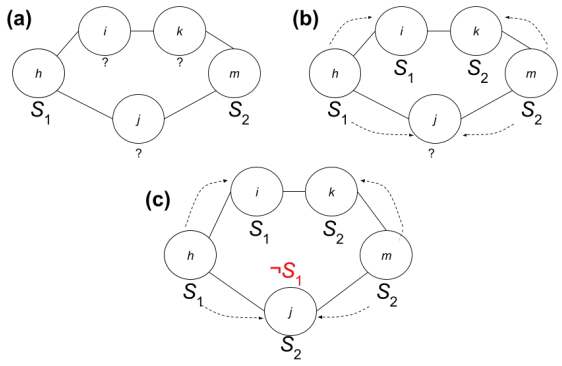

Figure 1: **The formulation of the label propagation problem.** (a) The general setting for label propagation where nodes can be labeled using multiple labels (shown under the nodes) and we want to predict labels for unlabeled nodes (b) The known labels are propagated from each node and the most likely label is assigned to unlabeled nodes. In this case it is difficult to predict a label for node j because it is equally proximal to nodes h and m. (c) Label propagation with negatively-labeled examples. The negative label for node j (shown above the node) informs the prediction that node j should not be labeled by S_1 so it is labeled by S_2.

Random walk with restarts (RWR): To formulate RWR, let $\mathbf{A}$ denote the adjacency matrix of G. We use $\mathbf{A}_{i,j}$ to denote matrix entries, $\mathbf{A}_{i,:}$ for rows and $\mathbf{A}_{:,j}$ for columns. Given S_i, were refer to the nodes $v \in S_i$ as *seed nodes*. RWR [16] propagates the labels of S_i to other nodes of G using a column stochastic transition matrix $\mathbf{A}^{(cs)}$ derived from $\mathbf{A}$ defined as $\mathbf{A}^{(cs)}_{i,j} = \mathbf{A}_{i,j} / \sum_k \mathbf{A}_{k,j}$. A restart vector $\mathbf{r}_i$ is used to localize the random walk around the seed nodes, where $\mathbf{r}_i(v) = 1/|S_i|$ for $v \in S_i$ and 0 otherwise ($\mathbf{r}_i(v)$ denotes the vector element corresponding to node v). A restart parameter, α (also called damping factor) is used to tune the frequency at which the walker "teleports" back to the seed nodes. The RWR-based proximity is defined as the steady state:

$$\mathbf{p}_i = (1-\alpha)\mathbf{A}^{(cs)}\mathbf{p}_i + \alpha\mathbf{r_i} \tag{1}$$

where $\mathbf{p}_i(v)$ denotes the probability of being at node v when the walk continues for a sufficiently long time. The steady state vector $\mathbf{p}_i$ is used to rank nodes for prediction, where higher values $\mathbf{p}_i(v)$ correspond to higher likelihood that node v is labeled the

same as nodes of S_i. This procedure can be repeated for each label set $S_i, i = 1...n$ and the most likely label (i.e. the $\mathbf{p}_i(v)$ with highest value) is predicted for node v.

Random walks with symmetric degree normalization: While the above formulation of RWR is intuitive, a different normalization technique is often used to scale the transition probabilities by the in- and out-degree of nodes [23]. This "symmetric" normalization technique uses transition matrix $\mathbf{A}^{(sym)}$, where $\mathbf{A}^{(sym)} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ and $\mathbf{D}_{i,i} = \sum_k \mathbf{A}_{i,k}$. Since $\mathbf{A}^{(sym)}$ is not a stochastic matrix, a re-normalization step is introduced to the RWR formulation to produce the probability vector $\mathbf{p}$:

$$\begin{aligned} \hat{\mathbf{p}} &= (1-\alpha)\mathbf{A}^{(sym)}\mathbf{p} + \alpha\mathbf{r} \\ \mathbf{p} &= \hat{\mathbf{p}}/|\hat{\mathbf{p}}| \end{aligned} \tag{2}$$

Label propagation with negatively-labeled examples: In some applications, a set of negatively-labeled nodes N_i (i.e., anti-labels that specify a node is *not* of a specific class) is provided. When such information is not available, it is also potentially useful to sample negatively-labeled nodes from nodes that are not positively labeled (e.g. selecting N_i as a subset of $\cup_{i \neq j} S_i$) and use them to inform label propagation. The objective of label propagation with negative examples is to predict labels for unlabeled nodes that do not contradict the anti-labels. This is illustrated in Figure 1.

Many existing methods for label propagation utilize negative examples by formulating an optimization problem where the objective function penalizes predicting positive labels for negatively labeled nodes [2, 3, 10, 13]. Supervised Random Walk (SRW), one of the earliest algorithms that considers negative examples, learns a function to optimize edge weights such that positive examples are ranked higher than negative examples [2]. This is accomplished by embedding the restart vector $\mathbf{r}$ into the transition matrix $\mathbf{A}$ and explicitly restricting updates that would alter the matrix elements corresponding to $\mathbf{r}$. A more recent work on query-specific optimal networks (QUINT) takes a similar approach to adjusting the weight – or existence – of edges defined by $\mathbf{A}$, but it formulates the problem in terms of a single positive example (i.e.

$|S_i| = 1$) and uses an update approach that does not adjust the restart probability[13]. The teleportation tuning method of Berberidis *et al.* learns a weighted restart vector $\mathbf{r}_i$ for each label S_i that optimizes within-class predictions [3], but this results in a model where all nodes restart to a given node with the same probability. More recently, random walk with extended restarts (RWER) attempts to learn an optimal restart probability for each node $v \in V$[10] for a specific S_i. However, the method scales the strength of all edges incident to a node uniformly in relation to the restart probability, resulting in no discrepancy between positive and negative neighbors.

2.2 Proposed Approach: Combining Edge Re-weighting and Restart Tuning.

We propose to combine the ideas of edge re-weighting and restart tuning such that: (i) the walker restarts with higher probability ($> \alpha$) when it encounters an edge leading to a negatively labeled node, but (ii) continues walking with the default probability ($1 - \alpha$) when it encounters an edge leading to an unlabeled or positively labeled node. This has several benefits: 1) It does not artificially inflate the rank of nodes by redirecting the walker to a smaller group of neighbors. 2) It does not reduce the rank of unlabeled neighbor nodes by avoiding them in an effort to avoid the negatively labeled node.

Here, we develop a framework to realize this approach by reformulating RWR in an intuitive way that creates a single transition matrix composed of "restart edges" and "transition edges". We then adjust the entries of these matrices based on the given set of positive (S_i) and negative (N_i) examples.

Reformulation of random walks to unify transition and teleport: Considering the classical RWR formulation, the first term on the right-hand-side of Equation 1 captures the transition of the random walker from the current node to adjacent nodes, and the second term captures the random walker "teleporting" to seed nodes. Ob-

serving that $|\mathbf{p}| = 1$ by definition, we can express $\alpha\mathbf{r}$ as:

$$\alpha\mathbf{r} = \alpha\mathbf{r}\mathbf{1}^T\mathbf{p}$$

where $\mathbf{1}^T$ is a row vector of all 1's of compatible dimension to $\mathbf{r}$ such that $\alpha\mathbf{r}\mathbf{1}^T = \mathbf{R} \in \mathcal{R}^{|V|\times|V|}$. Noting that $\mathbf{R}\mathbf{p} = \alpha\mathbf{r}$ and setting $\mathbf{Q}^{(cs)} = (1-\alpha)\mathbf{A}^{(cs)}$, we can rearrange Equation 1 as an ordinary eigenvector equation:

$$\mathbf{p} = (\mathbf{Q}^{(cs)} + \mathbf{R})\mathbf{p}, \tag{3}$$

where $\mathbf{Q}^{(cs)}$ captures the transition of the random walker to adjacent nodes and $\mathbf{R}$ captures teleport to seed nodes. The intuition behind this formulation is illustrated in Figure 2a, where the reformulation effectively adds an edge from every $v \in V$ to every $u \in S$ with transition probability α. Similarly, for random walks with symmetric normalization, Equation 2 can be reformulated as:

$$\begin{aligned} \hat{\mathbf{p}} &= (\mathbf{Q}^{(sym)} + \mathbf{R})\mathbf{p} \\ \mathbf{p} &= \hat{\mathbf{p}}/|\hat{\mathbf{p}}| \end{aligned} \tag{4}$$

where $\mathbf{Q}^{(sym)} = (1-\alpha)\mathbf{A}^{(sym)}$. In our implementation, we use this reformulation of symmetric random walk.

Variable restarts: Consider a more flexible model where rather than the walker restarting with a fixed probability α at every node, the walker is free to restart with a unique probability depending on the current location in the network. This flexibility can be directly incorporated into the above formulation, since each entry of $\mathbf{R}$ represents a directed edge from a given node to a seed node. The immediate benefit to such a model is it allows the walker to restart to a seed whenever it encounters an edge leading to a negative example, but to continue traversing edges to positive or unlabeled nodes.

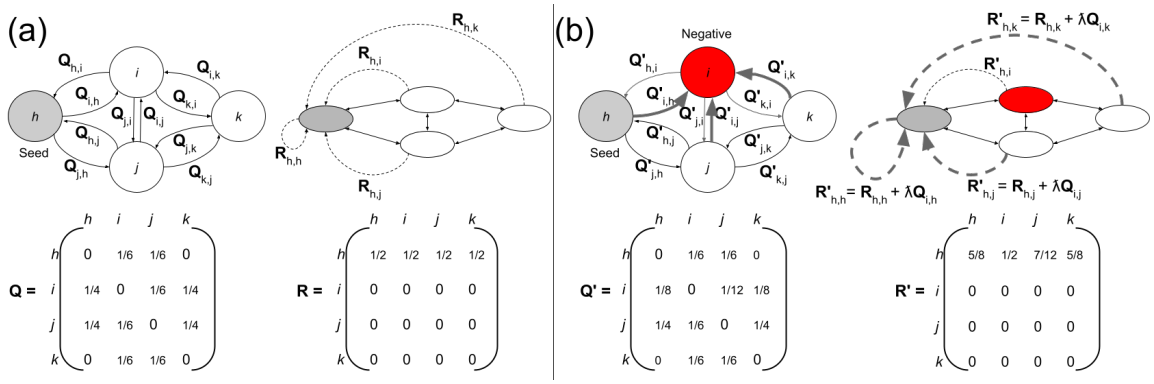

Figure 2: **Reformulation of random walks using transition and teleport matrices to allow variable restarts**. The seed node is shown in gray. The transition edges are labeled by $\mathbf{Q}$, the teleport edges are labeled by $\mathbf{R}$. (a) A reformulated random walk with restarts that is equivalent to the classical formulation with $\alpha = 0.5$ where columns of $\mathbf{Q}$ are column normalized. Note that the row $\mathbf{R}_{h,:}$ that corresponds to the seed node h contains all uniform entries. (b) The random walk modified to avoid negative node i using a redirection factor $\lambda = 0.5$, where re-weighted edges have been highlighted in bold and the updated matrices $\mathbf{Q}'$ and $\mathbf{R}'$ are shown below. The edges that lead to the negative node have been re-weighted as $\mathbf{Q}'_{i,:} = (1-\lambda)\mathbf{Q}_{i,:}$. The restart edges leaving nodes $v \in Adj(i)$ have been updated as $\mathbf{R}'_{h,v} = \mathbf{R}_{h,v} + \lambda\mathbf{Q}_{i,v}$ to direct the walker back to the seed rather than transitioning to the negative node. This formulation allows restarting with different probabilities depending on the current node visited by the walk.

Optimization problem to learn restart and transition edge weights Given query set S_i and negative example set N_i, the following quadratically constrained quadratic program (QCQP) demonstrates one approach to learning edge weights for $\mathbf{Q}$ and $\mathbf{R}$ that minimize the time the walker spends at the negative nodes:

$$f = \min \sum_{u \in N_i} \mathbf{p}(u) \tag{5}$$

Subject to:

$$
\begin{aligned}
(\mathbf{Q}+\mathbf{R})_{j,:}\mathbf{p} - \mathbf{p}(j) &= 0 \\
\textstyle\sum \mathbf{Q}_{:,j} + \sum \mathbf{R}_{:,j} &= 1 \\
\textstyle\sum \mathbf{p}(j) &= 1 \\
\forall \{u,v\} \in E \qquad \mathbf{Q}_{u,v} &\geq 0 \\
\forall \{u,v\} : u \in S, v \in V \qquad \mathbf{R}_{u,v} &\geq 0
\end{aligned}
\tag{6}
$$

Note that the variables $\mathbf{Q}_{u,v}$ and $\mathbf{R}_{u,v}$ corresponding to edges that do not exist are always equal to 0 and are excluded from the optimization problem. This QCQP suffers from a large number of parameters to learn, and from being under-constrained. For example, any cut of the graph that makes the negatives unreachable from the seeds minimizes the objective. Adding additional terms to the objective, such as minimizing the difference between the original $\mathbf{Q}+\mathbf{R}$ and the learned $\mathbf{Q}'+\mathbf{R}'$ may decrease the solution space, but does not guarantee the optimal solution is necessarily better. However, we can make a simple observation: To minimize the score at negatives nodes, with minimal impact on the reachability of the remaining nodes, we can simply reduce the capacity of inbound edges to all $u \in N_i$ using the capacity of the restart edges to absorb the difference. In the next section, we devise a direct method to modify $\mathbf{Q}$ and $\mathbf{R}$ based on the query set S_i and negative examples N_i.

Adjusting restart and transition edges based on negative examples: Let $u \in N_i$ be a negatively-labeled example for label i. For each v adjacent to u, we reduce transition probability from v to u and redistribute these probabilities to the seed vertices S_i as follows:

$$
\begin{aligned}
\mathbf{R}_{s,v} &= \mathbf{R}_{s,v} + \frac{\lambda \mathbf{Q}_{u,v}}{|S_i|} \text{ if } v \in Adj(u) \text{ and } s \in S_i \\
\mathbf{Q}_{u,v} &= (1-\lambda)\mathbf{Q}_{u,v} \text{ if } v \in Adj(u)
\end{aligned}
\tag{7}
$$

where λ is a "redirection" parameter used to tune the degree of aggressiveness in

steering the walk away from negatively-labeled nodes. In the next section, we comprehensively characterize the effect of λ on predictive accuracy. Observe that this adjustment retains the sum of the vth column of $\mathbf{Q} + \mathbf{R}$.

2.3 Label Propagation via CusTaRd.

The matrix $\mathbf{Q}^{(sym)}$ is independent of the label that is to be propagated, thus we first construct $\mathbf{Q}^{(sym)}$ based on the input graph $G(V, E)$. Then, for each label i with seed set S_i of positively-labeled nodes, we first construct the matrix $\mathbf{R}$. If negatively-labeled nodes are not available, we sample negatively-labeled nodes from $\cup_{i \neq j} S_i$ to obtain N_i, using the methodology described in the next subsection. Subsequently, we adjust $\mathbf{R}$ and $\mathbf{Q}^{(sym)}$ based on N_i, using Equation 7. We then compute $\mathbf{p}_i$ using Equation 4 and rank the nodes in $V \setminus (S_i \cup N_i)$ according to this vector to prioritize the assignment of label i.

2.4 Sampling Negatively Labeled Nodes.

If a set of negatively-labeled nodes is not available, it is necessary to sample negatively-labeled nodes from the set of nodes that are not positively-labeled (i.e. from $\cup_{i \neq j} S_i$ or using alternate heuristics or selection criteria). In the literature, negative sampling methods have been proposed based on prioritizing confident false predictions [25]. It follows that false negatives are nodes that are close to one or more seed nodes. For this reason, it can be a good strategy to select negatively-labeled examples from the set of nodes that are in the neighborhood of positively-labeled nodes. The pre-processing step of OBOE[12] that predicts positive/negative edge signs from toplogical node features is directly applicable here for expanding the available negatives and/or predicting negatives closer to the positive examples.

To investigate how the proximity of the sampled negatively-labeled nodes to seeds affects predictive performance,we sample negatives from the nodes uniquely reachable in exactly k-hops from each seed node. For this purpose, to generate a pool of

candidate negatively-labeled nodes, we use breadth-first search and identify nodes that (i) are at depth of k hops from the seeds, and (ii) do not have the same label as the seed. From this pool, we draw uniformly at random a sample that is of size at most (if possible, equal to) the number of seeds (positively-labeled nodes). This ensures that the sets of positively and negatively labeled nodes are as balanced as possible.

2.5 Complexity.

The space complexity to store the transition matrix scales as a factor of the seed set size $|S|$, where the initial edge set E is expanded by the addition of restart edges from each node $v \in V$ of the network to every seed $s \in S$. Thus, the space complexity of the transition matrix is $\Theta(|E| + |S||V|)$ with worst case $O(|V|^2)$ when the full vertex set is used as a seed set.

The time complexity involves the construction of the transition matrix and the convergence of the random walk. Adding the restart edges to a single seed requires $|V|$ operations, so the full operation requires $\Theta(|S||V|)$ time with worst case $O(|V|^2)$ when the full vertex set is used as the seed set. The random walk converges after a finite number of steps c, where each step requires multiplying a vector of size $|V|$ by the sparse transition matrix with runtime $\Theta(c|V|(|E| + |S||V|))$. The overall time complexity is dominated by the propagation which has asymptotic runtime $O(|V|^3)$ when $S = V$. However, in practice $|S| \ll |V|$ so the expected runtime would be much lower than the worst case.

Table 1: **Network datasets with node labels used to evaluate label prediction performance.**

Name	# Nodes	# Edges	# Labels
CiteSeer	3312	4660	6
CORA	2,708	10,556	7
Polblogs	1,224	16,718	2
Facebook	22,470	171,002	4

3 Results

3.1 Experimental Setup.

We evaluate the predictive performance of CusTaRd against existing methods using multiple network datasets that are often used to benchmark label propagation and node classification algorithms. These datasets include the CORA dataset [19], a CiteSeer dataset [7], the Political Blogosphere dataset [1] and a Facebook dataset [18]. The characteristics are summarized in Table 1. For consistency, we convert networks with directed edges to undirected networks, and remove nodes that are isolated from the rest of the network.

Sampling of training and validation sets: In our experiments, we consider the case where positive examples are scarce, i.e., most of the seed labels in the network are unknown, but negatives are readily available. Namely, from each set of labeled nodes $S_i \in \mathcal{S}$ for a given network, we sample, uniformly at random, 50 positive training (seed) sets $s_1, s_2, ..., s_{50}$ of fraction γ of the nodes in S_i, e.g. $s_j \subset S_i$ and $|s_j| = \gamma * |S_i|$. For each seed set s_j, we draw up-to the same number of negative training sets n_j at distances $k = [1, 2, 3]$ from the seeds using the strategies outlined in Section 2.4. Due to network topology and the location of the nodes in s_j, there are cases where $|n_j| < |s_j|$, we perform the experiment as long as $|n_j| > 0$. If $|n_j| = 0$, we sample a new seed set s_j until at least one negatively labeled node at distance k can be found. We use the set $T_j = \{s_j \cup\ n_j\}$ for training, leaving $V \backslash T_j$ for validation.

Parameter settings: We determine through an initial parameter sweep that the RWR based methods perform optimally with a restart probability $\alpha = 0.05$, thus we use this value in all experiments. During our baseline accuracy assessment, we set CusTaRd's redirection parameter to $\lambda = 0.9$ based on initial experiments that showed higher values of this parameter provide better predictive performance. We perform two additional experiments to characterize the effects of the redirection factor λ and the training set size γ. We varied λ over the values $[0.2, 0.4, 0.6, 0.8, 1.0]$ and γ over the values $[0.02, 0.05, 0.1]$.

Competing methods: We compare the predictive performance of CusTaRd against classical RWR with symmetric normalization [16], QUINT [13], and RWER [10], where the latter two methods learn optimal transition strategies using gradient descent. For QUINT, the authors provide several variations and we select their first order Taylor polynomial approximation as all three variations show equivalent performance in the benchmark experiments reported by the authors [13].

In CusTaRd, the positively labeled training nodes and the seed set are identical. This is not the case for QUINT and RWER. For both algorithms, the setting involves sets of positive and negative example nodes, as well as a single query (seed) node (i.e. $|s_i| = 1$). The methods then learn optimal networks or restart profiles that rank the positive nodes higher than the negative nodes while propagating the label only from the single query node. This makes direct comparison to our set-based method problematic, so we create a modified version of our method that also works with a single query node. The modified CusTaRd$_{sq}$ accepts the same inputs as QUINT and RWER, but adds edges to G between the query node and the positively-labeled training nodes before applying the edge-weight redistribution for negatively-labeled training nodes. This allows us to propagate the label from a single query node, but leverage the positive nodes in a way that is similar to treating them as additional seed nodes.

Evaluation of predictive performance. We use each method to propagate the

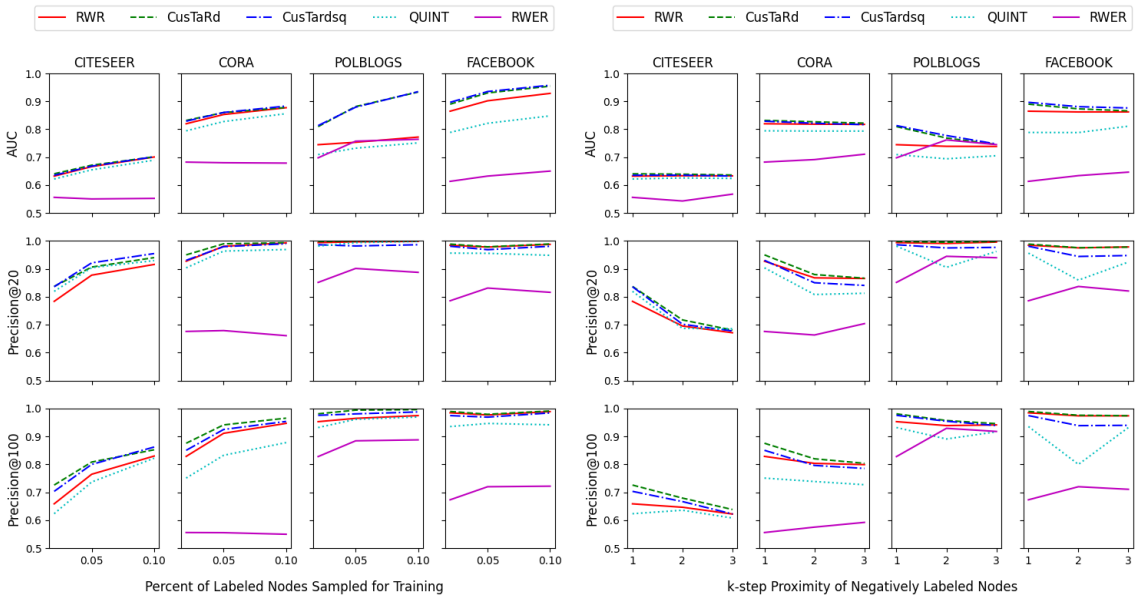

Figure 3: **(Left) Predictive performance of label propagation algorithms as a function of training set size.** Positive training sets are sampled of sizes 2%, 5% and 10% of available positive examples (seed nodes) for each label. Negative examples are sampled to be of equal size as positives at a k-hop distance of 1 to positive examples. The reported values are averages across 50 validation instances. **(Right) The effects of negative example proximity to seed nodes on predictive performance.** As discussed in Section 2.4, we sample negatively-labeled training nodes from the set of nodes that are not positively labeled, by constructing pools of candidate nodes based on their distance to positively-labeled nodes. The curves show the effect of this distance on predictive performance for k-hop distances 1, 2 and 3 using positive node sample size of 2%.

labels of sample s_j and then rank the nodes by confidence of predictions. The node rankings are then evaluated from most confident to least confident, assigning "true positive" or "false positive" to each prediction. The Area Under ROC Curve (AUCs), Precision@20, and Precision@100 are computed by combining the TP/FP counts at each rank position for all s_j across all labeled sets S_i to generalize the performance for each dataset. We report the mean and standard deviation of these values across the 50 validation instances.

Table 2: **Predictive performance of CusTaRd and competing methods on four benchmarking datasets according to three different performance criteria**. For each algorithm, dataset, and performance metric, the mean performance metrics ± standard deviation is shown across 50 randomly generated validation instances, with 2% of positively-labeled nodes selected for training, and negatives sampled at k-hop distance 1.

Network	*RWR*	CusTaRd	CusTaRd$_{sq}$	*QUINT*	*RWER*
			AUC		
CiteSeer	0.632±0.129	**0.641±0.130**	0.635±0.131	0.622±0.130	0.556±0.122
Cora	0.820±0.089	**0.832±0.084**	0.829±0.084	0.794±0.093	0.682±0.188
Polblogs	0.745±0.051	0.810±0.050	**0.813±0.057**	0.709±0.049	0.698±0.190
Facebook	0.865±0.037	0.890±0.034	**0.897±0.035**	0.789±0.063	0.613±0.142
			Precision@20		
CiteSeer	0.784±0.236	**0.837±0.243**	0.836±0.247	0.820±0.250	0.448±0.502
Cora	0.927±0.112	**0.950±0.093**	0.931±0.123	0.904±0.120	0.676±0.398
Polblogs	0.994±0.026	**0.998±0.012**	0.987±0.043	0.982±0.043	0.852±0.320
Facebook	0.985±0.039	**0.989±0.026**	0.981±0.038	0.957±0.083	0.786±0.353
			Precision@100		
CiteSeer	0.658±0.250	**0.726±0.272**	0.703±0.266	0.623±0.252	0.367±0.392
Cora	0.828±0.145	**0.875±0.133**	0.850±0.146	0.751±0.149	0.556±0.363
Polblogs	0.953±0.029	**0.981±0.018**	0.976±0.029	0.932±0.051	0.828±0.306
Facebook	0.984±0.020	**0.989±0.013**	0.975±0.033	0.935±0.082	0.673±0.401

3.2 Predictive Performance.

The predictive performance of all algorithms on all four datasets are shown in the left panel of Figure 3 as a function of training set size, using three different performance criteria. The average and standard deviation of the performance metrics for training size 2% are also shown in Table 2.

CusTaRd consistently achieves highest scores for Precision@20 and Precision@100, and for AUC the best performance is achieved by either CusTaRd or CusTaRd$_{sq}$.

We observe that the CiteSeer network is the most difficult dataset for all methods to deliver accurate predictions, where both QUINT and CusTaRd achieve better early precision than the conventional RWR. For this network, Precision@20 is in the low 80 percent range even for the best-performing algorithms. The minimum variance in prediction accuracy is displayed by CusTaRd for most datasets and metrics, with the exception of the CiteSeer network where the conventional RWR has the lowest

variance.

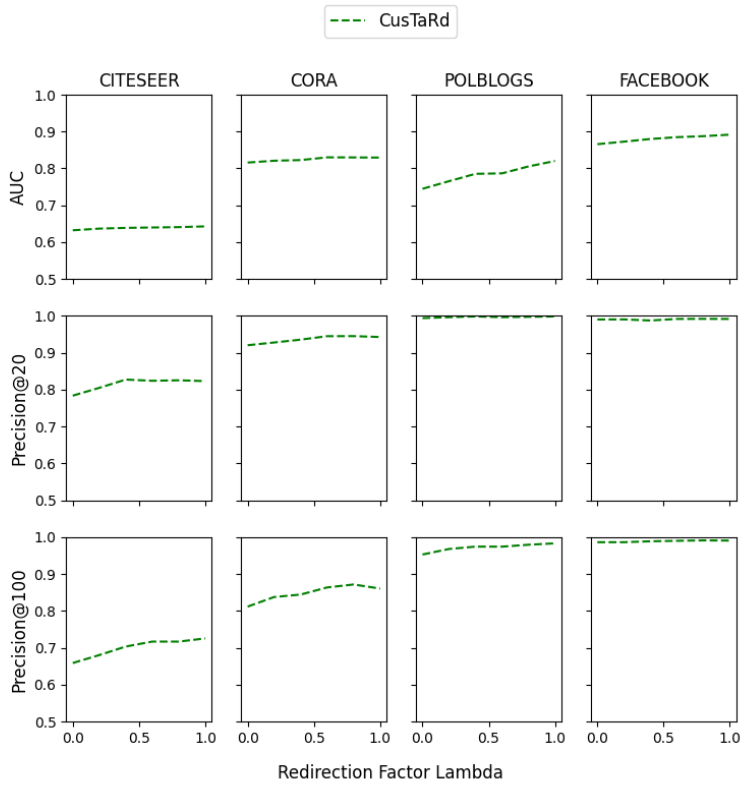

Figure 4: **Effects of redirection factor λ on predictive performance.** Our reformulated random walk depends on the redirection factor λ as defined in Equation 7. The plot shows the effects of varying the redirection factor using training sets of size 2% for each label and negative nodes sampled at a k-hop distance of 1. The value of λ was varied over $[0.0, 0.2, 0.4, 0.6, 0.8, 1.0]$.

3.3 Effect of Sampling of Negative Examples.

The right panel of Figure 3 plots the different performance metrics versus the k-hop proximity of negative examples for all four networks. It shows for all methods except RWER that negatives at k-hop proximity 1 to the seeds result in optimal performance. For RWR, $CusTaRd$, $CusTaRd_{sq}$ and some QUINT results, performance was inversely correlated with k (i.e. performance decreased as k-hop distance increased). However, some QUINT results exhibited lowest performance at k-hop distance 2 rather than 3, making them less correlated but still consistent with the

observation that the negatives sampled at distance 1 are most informative. RWER achieved optimal performance at k-hop distance 2, though the performance was still lower than the optimal performance of $CusTaRd$ and $CusTaRd_{sq}$. Based on the results, it would be reasonable to sample negatives as close to the seeds as possible. This behavior has the nice property of limiting the neighborhood of nodes that must be evaluated in the search to manually annotate negatives.

3.4 Effect of Redirection Factor.

Figure 4 plots the performance metrics for $CusTaRd$ versus the redirection factor λ for sample sizes 2% and fixed negative node k-hop distance 1. The curves are quite different between networks. The AUC curves for CORA and CITESEER show slight decreases in performance at the highest values of λ, while the POLBLOGS result shows increasing performance all the way to $\lambda = 1.0$. The gain in performance is more pronounced for Precision@100 than for Precision@20 showing that increasing λ helps to increase the ranking of more distant nodes.

3.5 Runtime

We did not perform a rigorous runtime comparison of the different methods as the optimization methods in general were significantly slower than conventional RWR or our two variations. RWR and our two variations were implemented in Python and completed in a few seconds, with RWR being the fastest due to the lowest sparsity and lack of modifications to the transition matrix. The optimization methods took multiple minutes per run, but this may be partially related to their implementations using a different language and inefficiency in how we were loading the adjacency matrix.

4 Conclusion

In this study, we reformulated random walks to enable variable restarts, which in turn gave rise to CusTaRd, an algorithm for effectively utilizing negatively-labeled nodes in label propagation. CusTaRd does not "learn" parameters or solve an optimization problem, it uses a single parameter to directly modify the entries of the transition matrix to redirect flow from negatively-labeled nodes to positively-label nodes. In addition to reformulation of random walks, CusTaRd samples negatively-labeled nodes from the neighborhood of positively-labeled nodes, thereby learning to discriminate between positively and negatively labeled nodes. Our experiments on four benchmark networks showed that CusTaRd consistently outperforms competing optimization/learning-based algorithms, and its predictions are robust to scarce training samples. Finally, our experimental results showed that sampling negative examples in the neighborhood of positive examples improves prediction accuracy for all algorithms.

These results lay the foundations for more effective incorporation of label propagation into machine learning frameworks. Integration of the algorithm described here with machine learning models that use node features can further improve the accuracy and robustness of such models.

References

[1] Lada A Adamic and Natalie Glance. The political blogosphere and the 2004 us election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*, pages 36–43, 2005.

[2] Lars Backstrom and Jure Leskovec. Supervised random walks: Predicting and recommending links in social networks. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, page 635–644, New York, NY, USA, 2011. Association for Computing Machinery.

[3] Dimitris Berberidis, Athanasios N. Nikolakopoulos, and Georgios B Giannakis. Random walks with restarts for graph-based classification: Teleportation tuning and sampling design. In *2018 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2018 - Proceedings*, volume 2018-April of *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pages 2811–2815. Institute of Electrical and Electronics Engineers Inc., September 2018.

[4] Lenore Cowen, Trey Ideker, Benjamin J Raphael, and Roded Sharan. Network propagation: a universal amplifier of genetic associations. *Nature Reviews Genetics*, 18(9):551–562, 2017.

[5] Bin Fu, Zhihai Wang, Guandong Xu, and Longbing Cao. Multi-label learning based on iterative label propagation over graph. *Pattern Recognition Letters*, 42:85–90, 2014.

[6] Sara E Garza and Satu Elisa Schaeffer. Community detection with the label propagation algorithm: A survey. *Physica A: Statistical Mechanics and its Applications*, 534:122058, 2019.

[7] Lise Getoor. *Link-based Classification*, pages 189–207. Springer London, London, 2005.

[8] Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R Benson. Combining label propagation and simple models out-performs graph neural networks. *ICLR*, 2021.

[9] Taehyun Hwang and Rui Kuang. A heterogeneous label propagation algorithm for disease gene discovery. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 583–594. SIAM, 2010.

[10] Woojeong Jin, Jinhong Jung, and U. Kang. Supervised and extended restart in random walks for ranking and link prediction in networks. *PLOS ONE*, 14(3):1–23, 03 2019.

[11] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR*, 2019.

[12] Wonchang Lee, Yeon-Chang Lee, Dongwon Lee, and Sang-Wook Kim. Look before you leap: Confirming edge signs in random walk with restart for personalized node ranking in signed networks. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 143–152, New York, NY, USA, 2021. Association for Computing Machinery.

[13] Liangyue Li, Yuan Yao, Jie Tang, Wei Fan, and Hanghang Tong. Quint: On query-specific optimal networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 985–994, New York, NY, USA, 2016. Association for Computing Machinery.

[14] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolu-

tional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*, 2018.

[15] Yongxin Liao, Shenxi Yuan, Jian Chen, Qingyao Wu, and Bin Li. Joint classification with heterogeneous labels using random walk with dynamic label propagation. In James Bailey, Latifur Khan, Takashi Washio, Gill Dobbie, Joshua Zhexue Huang, and Ruili Wang, editors, *Advances in Knowledge Discovery and Data Mining*, pages 3–13, Cham, 2016. Springer International Publishing.

[16] Jia-Yu Pan, Hyung-Jeong Yang, Christos Faloutsos, and Pinar Duygulu. Automatic multimedia cross-modal correlation discovery. KDD '04, page 653–658, New York, NY, USA, 2004. Association for Computing Machinery.

[17] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Phys. Rev. E*, 76:036106, Sep 2007.

[18] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding, 2019.

[19] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. Technical report, University of Maryland, College Park and Lawrence Livermore National Laboratory, 2008.

[20] Tal Wagner, Sudipto Guha, Shiva Kasiviswanathan, and Nina Mishra. Semi-supervised learning on data streams via temporal label propagation. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5095–5104. PMLR, 10–15 Jul 2018.

[21] Binghui Wang, Jinyuan Jia, and Neil Zhenqiang Gong. Semi-supervised node classification on graphs: Markov random fields vs. graph neural networks. In

Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pages 10093–10101, 2021.

[22] Guobo Xie, Bin Huang, Yuping Sun, Changhai Wu, and Yuqiong Han. Rwsf-blp: a novel lncrna-disease association prediction model using random walk-based multi-similarity fusion and bidirectional label propagation. *Molecular Genetics and Genomics*, pages 1–11, 2021.

[23] Pinchen Xie, Zhongzhi Zhang, and Francesc Comellas. On the spectrum of the normalized laplacian of iterated triangulations of graphs. *Applied Mathematics and Computation*, 273:1123–1129, 2016.

[24] Dengyong Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Advances in neural information processing systems*, pages 321–328, 2004.

[25] Olga Zoidi, Anastasios Tefas, Nikos Nikolaidis, and Ioannis Pitas. Positive and negative label propagations. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(2):342–355, 2018.