Data Mining and Knowledge Discovery
https://doi.org/10.1007/s10618-024-01065-4

Random walks with variable restarts for negative-example-informed label propagation

Sean Maxwell[1] · **Mehmet Koyutürk**[1]

Received: 17 March 2022 / Accepted: 30 July 2024

Abstract

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 nodes 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 drive the walk away from 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 network datasets commonly used in benchmarking label propagation and node classification algorithms. Our results show that CusTaRd consistently outperforms competing algorithms that learn edge weights or restart profiles, and that negatives close to positive examples are generally more informative than more distant negatives.

Keywords Random walk · Label propagation · Negative examples · Node classification

Responsible editor: Johannes Fürnkranz.

✉ Sean Maxwell
stm@case.edu

Mehmet Koyutürk
mxk331@case.edu

[1] Department of Computer Science and Data Sciences, Case Western Reserve University, 10900 Euclid Ave, Cleveland, OH 44106, USA

Published online: 13 August 2024

1 Introduction

Label propagation is a commonly encountered problem in data mining and machine learning applications on network and graph-structured data (Garza and Schaeffer 2019; Wagner et al. 2018). 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. The seed nodes have known labels, and are thus true positives, or "positively-labeled examples". 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 (Wang et al. 2021).

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 (Klicpera et al. 2019). 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 (Li et al. 2018). In addition, emerging evidence suggests that combination of label propagation with simple models often outperforms more sophisticated models, such as graph neural networks (Huang et al. 2021). Despite the ubiquity of label propagation in supervised learning, efforts on effectively utilizing negatively-labeled examples (nodes that are labeled *a priori* as true negatives in relation to the label of interest) 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 (Fu et al. 2014; Hwang and Kuang 2010; Zhou et al. 2004; Xie et al. 2021; Liao et al. 2016). 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 (Zoidi et al. 2018). Existing approaches to informing random walks with negative examples use optimization to learn edge weights (Backstrom and Leskovec 2011; Li et al. 2016) or restart probabilities (Jin et al. 2019; Berberidis et al. 2018) 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 vulnerable to over-fitting. In addition, the optimization problems are often non-convex and prone to getting stuck at local optima.

Our contributions: We make two contributions to negative-example-informed label propagation. 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 characterize how the proximity of negative examples to positives affects the accuracy of predictions. We observe that using negative examples from the close neighborhood of positive examples leads to higher classifier accuracy than when using more distant negative examples. As negative examples close to positive examples generally score higher than more distant negatives during random walk based rankings, this finding supports the notion that hard negatives are more informative to training than lower scoring negatives(Shi et al. 2023).

Organization of the paper: In the next section, we define the label propagation problem and discuss related works involving label propagation with negatively-labeled examples. 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 characterizing how negatively-labeled node proximity to positively-labeled nodes affects prediction accuracy. In Sect. 3, we start by describing the 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 negative example proximity on the performance of all algorithms considered. We conclude our discussion and outline future avenues for research in Sect. 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 (Raghavan et al. 2007; Zhou et al. 2004). 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 (Cowen et al. 2017). 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.

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*, which are our positively-labeled examples. RWR (Pan et al. 2004) 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:

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

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 (Xie et al. 2016). 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 $\mathbf{p}$:

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

Other normalization methods have been proposed for label propagation with random walks, such as the node core normalization methods explored by NetCore (Barel and Herwig 2020). These and other normalization methods can be easily incorporated into the random walk by substituting their resulting matrix for $\mathbf{A}^{(sym)}$ into Eq. 2.

Label propagation with negatively-labeled examples: In some applications, a set of negatively-labeled nodes N_i (i.e., labels that specify nodes that are true negatives, or definitely *not* labeled the same as the positive examples) 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 improve the performance of predictions by leveraging examples of both true positives and true negatives.

Related Work: 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 (Backstrom and Leskovec 2011; Berberidis et al. 2018; Jin et al. 2019; Li et al. 2016). 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 (Backstrom and Leskovec 2011). 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 does not modify the restart profile of the walker(Li et al. 2016). 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 (Berberidis et al. 2018), 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 $v \in V$(Jin et al. 2019) 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: (1) the walker restarts with higher probability ($> \alpha$) when it encounters an edge leading to a negatively labeled node, but (2) 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 Eq. 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. Observing 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 Eq. 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 the random walker teleporting to seed nodes. The intuition behind this formulation is illustrated in Fig. 1, 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, Eq. 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. i.e, given a node $u \in N_i$, the values $\mathbf{Q}_{u,v}$ for all $v \in Adj(u)$ can be reduced, and the difference distributed among the restart edges $\mathbf{R}_{w,v}$ for $w \in S_i$.

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:

$$\begin{aligned} \mathbf{R}_{s,v} &= \mathbf{R}_{s,v} + \frac{\lambda \mathbf{Q}_{u,v}}{|S_i|} \text{ if } v \in Adj(u) \text{ and } s \in S_i \\ \mathbf{Q}_{u,v} &= (1-\lambda)\mathbf{Q}_{u,v} \text{ if } v \in Adj(u) \end{aligned} \tag{5}$$

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 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 $S_{j \neq i}$ (i.e. we sample from within the nodes that we know have another label) 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 Eq. 5. We then compute $\mathbf{p}_i$ using Eq. 4 and rank the nodes in $V \setminus (S_i \cup N_i)$ according to this vector to prioritize the assignment of label i.

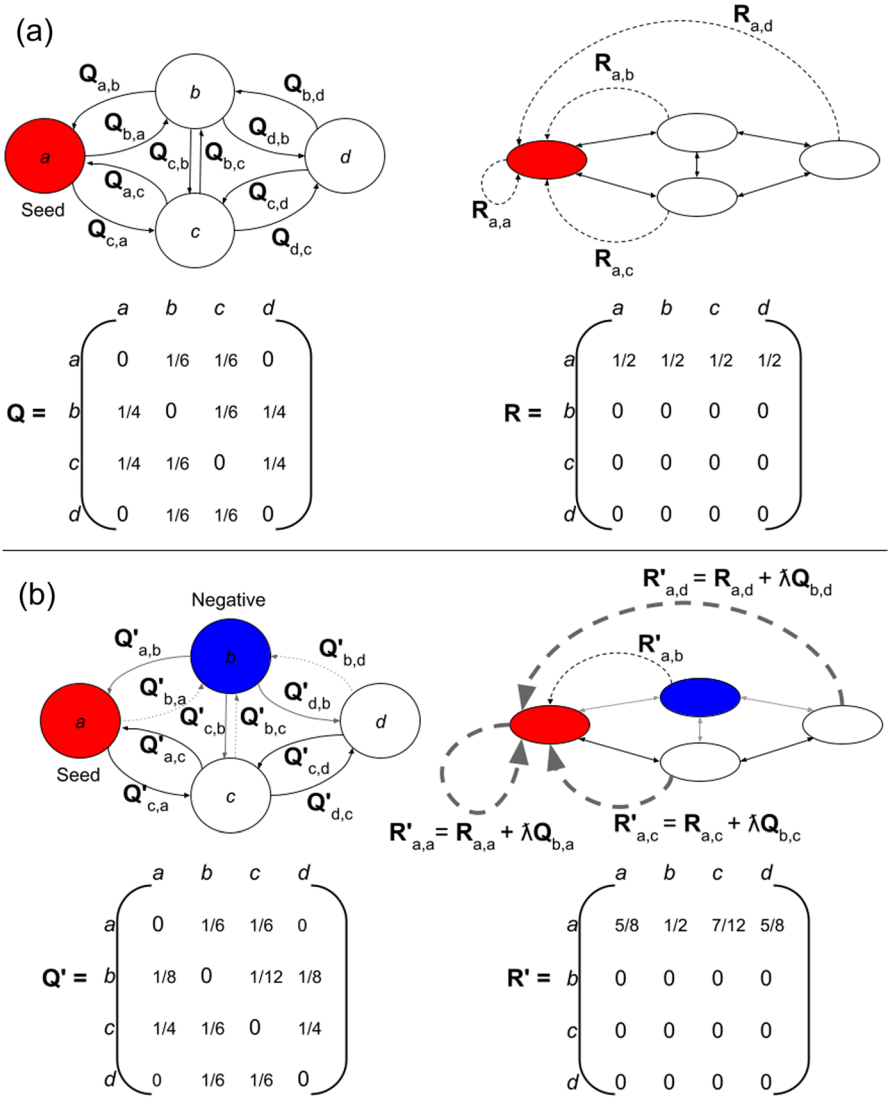

Fig. 1 Reformulation of random walks using transition and teleport matrices to allow variable restarts. The seed node (positive) is shown in red and the negative example node in blue. The transition edges are labeled by $\mathbf{Q}$, the teleport edges are labeled by $\mathbf{R}$. **a** A reformulated 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 $\mathbf{R}_{a,:}$ that corresponds to the seed node a contains all uniform entries. **b** The random walk modified to avoid negative node b 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 $\mathbf{Q}'_{b,:} = (1 - \lambda)\mathbf{Q}_{b,:}$. The restart edges leaving nodes $v \in Adj(b)$ have been updated as $\mathbf{R}'_{a,v} = \mathbf{R}_{a,v} + \lambda\mathbf{Q}_{b,v}$ 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

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. In the literature, negative sampling methods have been proposed based on prioritizing confident false predictions (Zoidi et al. 2018), also known as hard sampling(Shi et al. 2023). It follows that confident false positives in random walk based rankings 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. 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 (1) are at depth of k hops from the seeds, and (2) 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.

3 Results

3.1 Experimental setup

Data Sets: 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 (Sen et al. 2008), a CiteSeer dataset (Getoor 2005), the Political Blogosphere dataset (Adamic and Glance 2005), a Facebook dataset (Rozemberczki et al. 2019) and the arXiv dataset from OGB (Wang et al. 2020). 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.

Competing methods: We compare the predictive performance of CusTaRd against classical RWR with symmetric normalization (Pan et al. 2004), QUINT (Li et al. 2016), and RWER (Jin et al. 2019). QUINT learns an optimal transition matrix, and RWER learns an optimal restart profile, and both methods use gradient based optimization for learning. 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 (Li et al. 2016).

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.

Sampling of training and validation sets: In our experiments, we consider the case where training data is scarce, i.e., most of the labels in the network are unknown. 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 Sect. 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

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

Name	# Nodes	# Edges	# Labels
CiteSeer	3312	4660	6
CORA	2708	10,556	7
Polblogs	1224	16,718	2
Facebook	22,470	171,002	4
OGB arXiv	169,343	1,166,243	40

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 determined through an initial parameter sweep that the RWR based methods performed 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].

Evaluation of predictive performance. We use each method to propagate the 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 (AUC) and Precision@100 are computed by combining the TP/FP counts at each rank 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.

3.2 Predictive performance

The predictive performance of all algorithms on all datasets are shown in Fig. 2 as a function of training set size. 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@100, and for AUC the best performance is achieved by either CusTaRd or CusTaRd $_{sq}$. We observe that the CITESEER and OGBNARXIV networks are the most difficult datasets for all methods to deliver accurate predictions. For CITESEER, the Precision@100 shows significant improvement for most methods as the training set size increases, but for OGBNARXIV the change is less pronounced. The minimum variance in prediction accuracy for the smallest training set size is displayed by CusTaRd for most datasets and metrics, with the exception of the CITESEER and OGBNARXIV networks, where the conventional RWR and QUINT display lower variance than CusTaRd.

3.3 Effect of sampling of negative examples

Figure 3 plots the different performance metrics versus the k-hop proximity of negative examples. For RWR, *CusTaRd* and *CusTaRd*$_{sq}$ the AUC appears either unchanged or inversely correlated to k, but optimal precision was always achieved at $k = 1$. QUINT shows a less consistent correlation to k with AUC, but with the exception of the CITESEER network it always achieved highest precision at $k = 1$ (data point for OGBNARXIV not shown because it was less than 0.5). The AUC for RWER achieves the maximum value for $k = 3$ on all datasets except POLBLOGS, while optimal precision was achieved at k-hop distance 2 or 3. Based on the results,

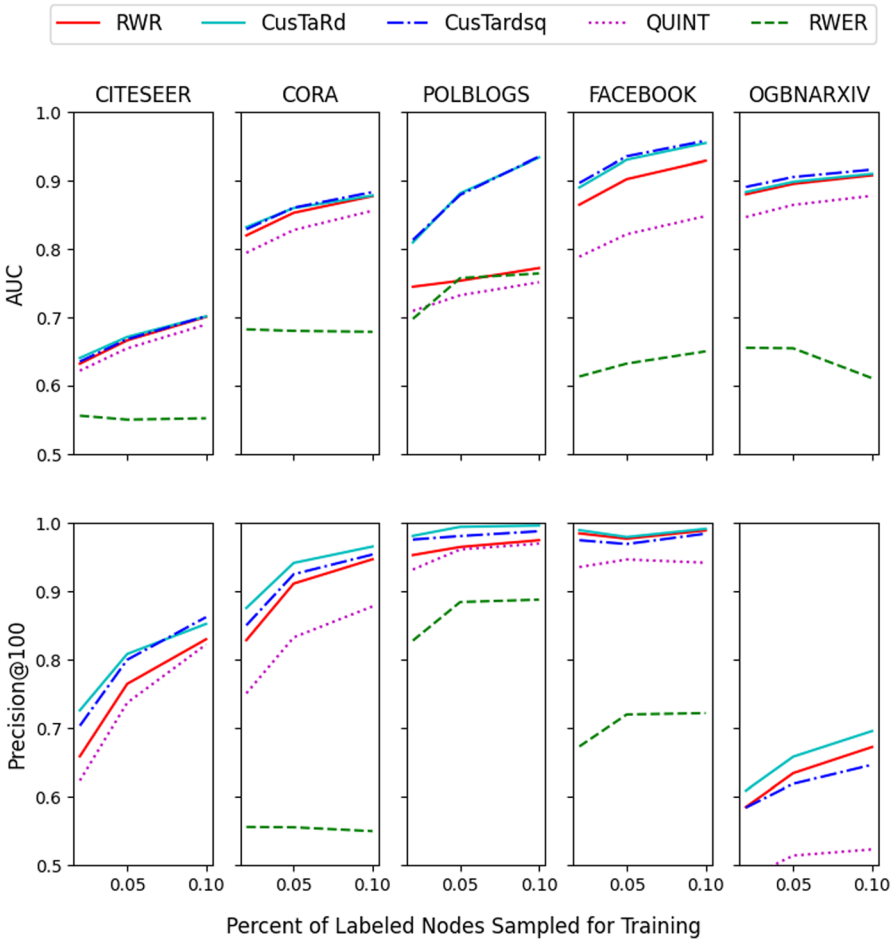

Fig. 2 Predictive performance of label propagation algorithms as a function of training set size. Positive training sets are sampled of sizes 2%, 5% and 10% of available positive examples (seed nodes) for each label. Negative examples are sampled to be of equal size as positives at a k-hop distance of 1 to positive examples. The reported values are averages across 50 validation instances

it would be reasonable to sample negatives as close to the seeds as possible because doing so results in superior precision across most methods and datasets. 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 most networks show slight

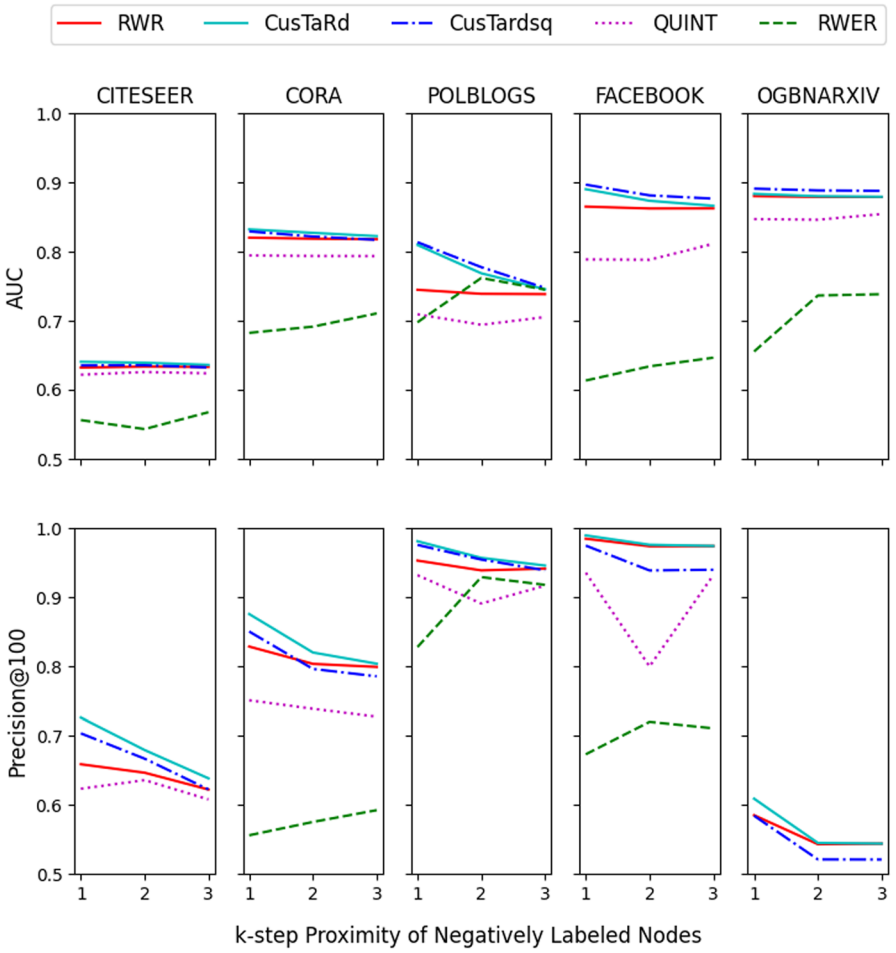

Fig. 3 The effects of negative example proximity to seed nodes on predictive performance. As discussed in Sect. 2.4, we sample negatively-labeled training nodes from the set of nodes that are not positively labeled, by constructing pools of candidate nodes based on their distance to positively-labeled 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%

increases in performance as λ increases, while the POLBLOGS result shows significant jumps as λ increases through certain ranges. The gain in performance for Precision@100 increases with higher values of λ except at the highest values on CORA where it interestingly showed lower accuracy than at slightly lower values.

3.5 Runtime

Our implementation and testing code were written in Python, while competing methods used MATLAB, making a direct and systematic runtime comparison somewhat

Table 2 Predictive performance of CusTaRd and competing methods

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
OGBNARXIV	0.880±0.145	0.883±0.150	**0.891±0.147**	0.847±0.154	0.656±0.209
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
OGBNARXIV	0.585±0.274	**0.609±0.284**	0.584±0.273	0.480±0.253	0.287±0.363

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

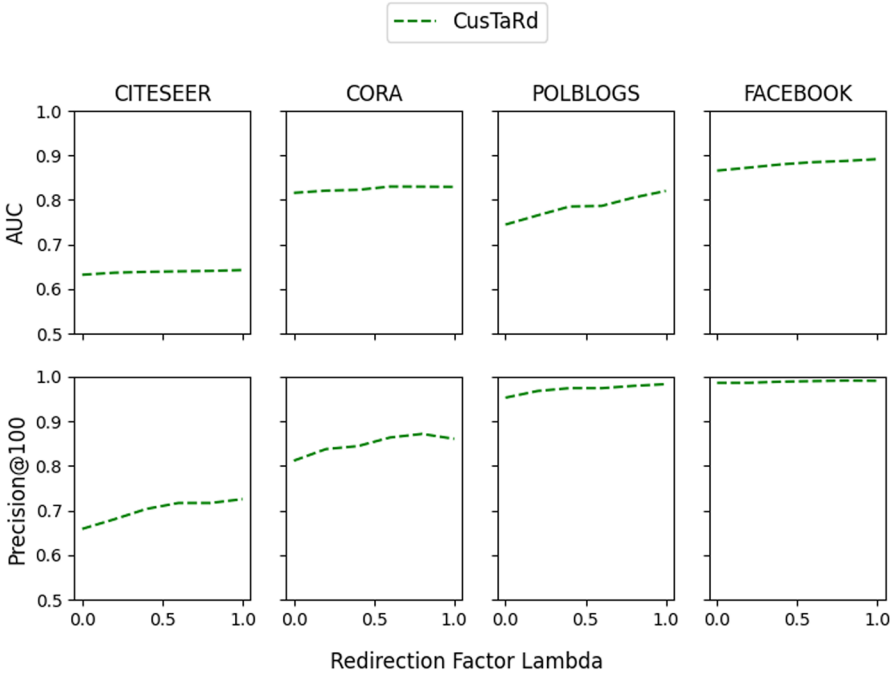

Fig. 4 Effects of redirection factor λ **on predictive performance.** Our reformulated random walk depends on the redirection factor λ as defined in Eq. 5. The plot shows the effects of varying the redirection factor using training sets of size 2% for each label and negative nodes sampled at a k-hop distance of 1. The value of λ was varied over [0.0, 0.2, 0.4, 0.6, 0.8, 1.0]

difficult and potentially misleading. For this reason, to get a general idea on the runtime of each method, we perform a more generic comparison on the OGB dataset. Our runtime measurements include the time to complete all steps of each method (loading the adjacency matrix, training, and output) on the OGB dataset. Our results show that for the smallest seed set size, the methods all require around 1 min on average on this dataset. As the seed set size increases, CusTaRd and the gradient based methods require more time, but the gradient based methods' runtimes increase more quickly. The increase for RWER is not as pronounced as QUINT, but for the largest seed set size, both methods on average require more than 1.5x the runtime of CusTaRd to complete.

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 stochastic matrix to redirect flow from negatively-labeled nodes to positively-label nodes. In addition to reformulation of random walks, CusTaRd we characterized how the proximity of negatively-labeled nodes to positively-labeled nodes affects the accuracy of predictions. Our experiments on benchmark networks showed that CusTaRd consistently outperforms competing optimization/learning-based algorithms, and its predictions are consistently robust to scarce training samples.

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.

Acknowledgements This work made use of the High Performance Computing Resource in the Core Facility for Advanced Research Computing at Case Western Reserve University.

Declarations

Conflict of interest The authors have no conflict of interest to declare that are relevant to the content of this article.

References

Adamic LA, Glance N (2005) The political blogosphere and the 2004 us election: divided they blog. In: Proceedings of the 3rd international workshop on Link discovery, pp 36–43

Backstrom L, Leskovec J (2011) 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, New York. Association for Computing Machinery, pp 635–644. ISBN 9781450304931. https://doi.org/10.1145/1935826.1935914

Barel G, Herwig R (2020) Netcore: a network propagation approach using node coreness. Nucl Acids Res 48(17):e98–e98

Berberidis D, Nikolakopoulos A, Giannakis G (2018) 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. Institute of Electrical and Electronics Engineers Inc., pp 2811–2815. ISBN 9781538646588. https://doi.org/10.1109/ICASSP.2018.8461548

Cowen L, Ideker T, Raphael BJ, Sharan R (2017) Network propagation: a universal amplifier of genetic associations. Nat Rev Genet 18(9):551–562

Fu B, Wang Z, Xu G, Cao L (2014) Multi-label learning based on iterative label propagation over graph. Pattern Recogn Lett 42:85–90. https://doi.org/10.1016/j.patrec.2014.01.001.

Garza SE, Schaeffer SE (2019) Community detection with the label propagation algorithm: a survey. Physica A Stat Mech Appl 534:122058

Getoor L (2005) Link-based Classification. Springer, London, pp 189–207. ISBN: 978-1-84628-284-3, https://doi.org/10.1007/1-84628-284-5_7

Huang Q, He H, Singh A, Lim S-N, Benson AR (2021) Combining label propagation and simple models out-performs graph neural networks. In: ICLR

Hwang T, Kuang R (2010) A heterogeneous label propagation algorithm for disease gene discovery. In: Proceedings of the 2010 SIAM international conference on data mining. SIAM, pp 583–594

Jin W, Jung J, Kang U (2019) Supervised and extended restart in random walks for ranking and link prediction in networks. PLOS ONE 14(3):1–23. https://doi.org/10.1371/journal.pone.0213857

Klicpera J, Bojchevski A, Günnemann S (2019) Predict then propagate: graph neural networks meet personalized pagerank. In: ICLR

Li L, Yao Y, Tang J, Fan W, Tong H (2016) Quint: on query-specific optimal networks. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, KDD '16, New York. Association for Computing Machinery, pp 985–994. ISBN 9781450342322. https://doi.org/10.1145/2939672.2939768

Li Q, Han Z, Wu X-M (2018) Deeper insights into graph convolutional networks for semi-supervised learning. In: Thirty-second AAAI conference on artificial intelligence

Liao Y, Yuan S, Chen J, Wu Q, Li B (2016) Joint classification with heterogeneous labels using random walk with dynamic label propagation. In: Bailey J, Khan L, Washio T, Dobbie G, Huang JZ, Wang R (eds) Advances in knowledge discovery and data mining. Springer, Cham, pp 3–13. ISBN: 978-3-319-31753-3

Pan J-Y, Yang H-J, Faloutsos C, Duygulu P, Automatic multimedia cross-modal correlation discovery. KDD '04. Association for Computing Machinery, New York, pp 653–658. ISBN: 1581138881

Raghavan UN, Albert R, Kumara S (2007) Near linear time algorithm to detect community structures in large-scale networks. Phys Rev E 76:036106

Rozemberczki B, Allen C, Sarkar R (2019) Multi-scale attributed node embedding

Sen P, Namata G, Bilgic M, Getoor L, Gallagher B, Eliassi-Rad T (2008) Collective classification in network data. Technical report, University of Maryland, College Park and Lawrence Livermore National Laboratory. https://drum.lib.umd.edu/handle/1903/7546

Shi W, Chen J, Feng F, Zhang J, Wu J, Gao C, He X (2023) On the theories behind hard negative sampling for recommendation

Wagner T, Guha S, Kasiviswanathan S, Mishra N (2018) Semi-supervised learning on data streams via temporal label propagation. In: Dy J, Krause A (eds) Proceedings of the 35th international conference on machine learning, volume 80 of proceedings of machine learning research. PMLR, pp 5095–5104. https://proceedings.mlr.press/v80/wagner18a.html

Wang B, Jia J, Gong NZ (2021) Semi-supervised node classification on graphs: Markov random fields versus graph neural networks. In: Proceedings of the AAAI conference on artificial intelligence, vol 35, pp 10093–10101

Wang K, Shen Z, Huang C, Wu C-H, Dong Y, Kanakia A (2020) Microsoft academic graph: when experts are not enough. Quant Sci Stud 1(1):396–413. https://doi.org/10.1162/qss_a_00021

Xie G, Huang B, Sun Y, Wu C, Han Y (2021) Rwsf-blp: a novel lncrna-disease association prediction model using random walk-based multi-similarity fusion and bidirectional label propagation. Mol Genet Genom 1–11

Xie P, Zhang Z, Comellas F (2016) On the spectrum of the normalized Laplacian of iterated triangulations of graphs. Appl Math Comput 273:1123–1129. https://doi.org/10.1016/j.amc.2015.09.057

Zhou D, Bousquet O, Lal TN, Weston J, Schölkopf B (2004) Learning with local and global consistency. In: Advances in neural information processing systems, pp 321–328

Zoidi O, Tefas A, Nikolaidis N, Pitas I (2018) Positive and negative label propagations. IEEE Trans Circuits Syst Video Technol 28(2):342–355. https://doi.org/10.1109/TCSVT.2016.2598671