Active Learning on Graphs via Spanning Trees

Nicolò Cesa-Bianchi
Università degli Studi di Milano, Italy
cesa-bianchi@dsi.unimi.it

Claudio Gentile
Università dell'Insubria, Italy
claudio.gentile@uninsubria.it

Fabio Vitale
Università degli Studi di Milano, Italy
fabio.vitale@unimi.it

Giovanni Zappella
Università degli Studi di Milano, Italy
giovanni.zappella@unimi.it

Abstract

Active learning algorithms for graph node classification select a subset L of nodes in a given graph. The goal is to minimize the mistakes made on the remaining nodes by a standard node classifier using L as training set. Bilmes and Guillory introduced a combinatorial quantity, $\Psi^*(L)$, and related it to the performance of the mincut classifier run on any given training set L. While no efficient algorithms for minimizing Ψ^* are known, they show that simple heuristics for (approximately) minimizing it do not work well in practice. Building on previous theoretical results about active learning on trees, we show that exact minimization of Ψ^* on suitable spanning trees of the graph yields an efficient active learner that compares well against standard baselines on real-world graphs.

1 Introduction

The avalaibility of large dataset collections extracted from social and biological networks is driving the study of fast and accurate predictive models for networked data. A simple and natural way to model such data is through a graph, where the presence of an edge between two nodes indicates similarity between the associated data elements. In this paper we consider what is perhaps the simplest learning task on networked data: node classification. Given a graph and the binary labels of a subset of nodes (the training nodes), the learner must infer the labels of the remaining nodes. As in certain domains gathering side information for each node may be very expensive, we focus on prediction techniques that rely on the network topology only.

In node classification tasks we distinguish between *passive* and *active* learning settings. While in the passive setting the training nodes are part of the learner's input, an active learner is allowed to freely select a subset of training nodes. When the amount of training nodes is severely limited (due, e.g., to the cost of acquiring labels), the choice of the training set can dramatically affect prediction performance. Active learning on networked data has been studied in [2, 10, 8] from a theoretical viewpoint and in [5, 14, 15] from a more empirical angle.

2 Active learning methods

A common and powerful assumption which a learner relies on for node classification tasks is that linked entities tend to be assigned the same class label. This is known as *homophily* in the context of social networks [6, 16]. Evidence of this bias in many real-world networked data spurred the development of several classification techniques in both passive and active settings —see, e.g., [7, 20, 18, 3, 4, 11, 12, 9, 10, 8] and references therein. Under the homophilic bias assumption, one could view the graph as a collection of weakly interconnected *clusters* induced by the unknown

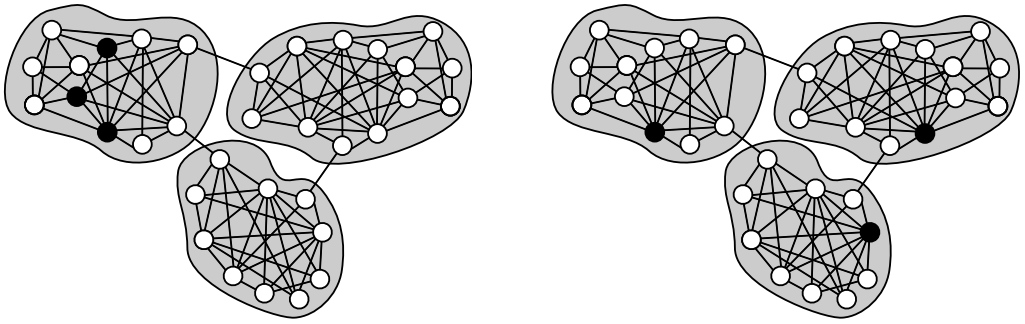

Figure 1: Two copies of a graph made up of three weakly interconnected dense clusters. In both pictures, the query set L contains the three black nodes. **Left:** A choice for L yielding a very large value of $\Psi^*(L)$. In fact, one can cut away from L the two rightmost clusters by only removing two edges. Here the value of $\Psi^*(L)$ is at least half the total number of nodes in the two rightmost clusters. A large value of $\Psi^*(L)$ corresponds to a bad choice of L. In this case the adversary can choose a single random labels for all nodes in the two rightmost clusters, thereby forcing *any* prediction algorithm to make (on average) half mistake per node in the two rightmost clusters, *irrespective* of the value of the training labels. **Right:** A choice of L yielding a small value of $\Psi^*(L)$. Since each cluster contains (at least) one of the training nodes, it is now impossible to cut away from L a large portion of the graph by removing only a few edges. The small value of $\Psi^*(L)$ corresponds to a good choice of the training nodes L. Here a prediction algorithm can label all remaining nodes in each cluster according to the label of the black node. Hence, if the algorithm makes even one prediction mistake, then the cutsize must be very large, which implies that the labeling contradicts the homophilic assumption.

labeling, where nodes in the same cluster all share a common label. Because of the homophilic bias, each dense area of the graph is likely to be part of a single cluster. So, in order to have a training set containing representative nodes from each cluster, an active learning algorithm should ideally select at least one node from each dense area.

A simple and natural way of measuring the hardness of a clustering induced by a labeling $\boldsymbol{y}$ is via the *cutsize* $\Phi(\boldsymbol{y})$, counting the number of edges connecting nodes with disagreeing labels. Recently, Guillory and Bilmes [10] introduced the function $\Psi(L)$, which quantifies the extent to which one can disconnect a large region of the graph from the set L of training nodes by cutting as few edges as possible. An adversary controlling the assignment of labels can then force many mistakes on the nodes of such weakly connected regions without inducing a large cutsize. Indeed, the reciprocal

$$\frac{1}{\Psi(L)} = \Psi^*(L) = \max_{\emptyset \neq V' \subseteq V \setminus L} \frac{|V'|}{\Gamma(V')}$$

where $\Gamma(V')$ is the number of edges between V' and $V' \setminus V$, corresponds to the number of prediction mistakes that can be adversarially forced per unit of cutsize (see Figure 1 for a pictorial explanation). Guillory and Bilmes showed that $\Psi^*(L)$ together with the cutsize $\Phi(\boldsymbol{y})$ bound from the above the number of mistakes made on non-training nodes by the mincut classifier [7] using training set L. This provides a theoretical justification of choosing L by minimizing $\Psi^*(L)$.

While [10] shows that it is always possible to compute $\Psi^*(L)$ in polynomial time, it leaves open how to efficiently find L minimizing $\Psi^*(L)$. In order to overcome this computational problem, in [8] we focused on the special case of trees, and introduced a fast training set selection algorithm SEL minimizing (to within constant factors) the function Ψ^* over any input tree. We proved that no other training set selection strategy for trees (irrespective of computational complexity) can do better than SEL when the assignment of labels is adversarial.

Despite these good theoretical properties on trees, heuristic minimization of Ψ^* on graphs performs worse than random selection, as the experiments in [10] demonstrate. In this work we show that using SEL to minimize Ψ^* on a spanning tree of the graph, and then running label propagation [19, 4] on the entire graph, yields a fast node classifier that, on real-world data, performs significantly better than random selection.

We ran our experiments using two types of spanning trees: random (RST) and breadth-first (BFST). Random spanning trees are good sketches of the graph from which they are drawn (see, e.g., [13]) and can be generated by a random walk in time *linear* in the number of nodes for most graphs [17]. Moreover, in [9] we showed that using a RST for node classification is optimal in the passive mistake bound setting. However, as mentioned in [8], random spanning trees are likely to hide the cluster structure of the graph, which is crucial for active learning. For example, the expected diameter of a RST drawn from an n-node clique is at least $\sqrt{n}$, while the diameter of the clique is one. Breadth-first spanning trees, instead, tend to create a high-degree node in each dense cluster, and these are precisely the nodes that SEL is most likely to pick. On the other hand, the performance of BFST trees is potentially affected by parameters like the starting node or the order of visit of children. Finally, though the use of spanning trees seems reasonable, it is difficult to provide theoretical performance guarantees.

3 Experimental setup

We ran our experiments on three real-world datasets: The first $10{,}000$ documents (in chronological order) of the RCV1 corpus, the DBLP network,[1] and the CORA network.[2] Nodes of CORA and DBLP denote scientific papers and edges denote citations or co-authorships, with no edge weights. We extracted the largest connected component obtaining $13{,}146$ nodes and $326{,}082$ edges for DBLP, and $2{,}484$ nodes and $5{,}205$ edges for CORA.

As for the RCV1 corpus, following previous experimental settings [20, 3] we built a weighted graph using k-NN with Euclidean distance $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|$ between the normalized TF-IDF vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ of documents i and j. The weight $w_{i,j}$ was set to $w_{i,j} = \exp\left(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 / \sigma^2\right)$ if j was one of the k nearest neighbors of i, and to zero otherwise. In order to choose σ^2 we first computed the average square distance between i and its k nearest neighbors, and then took a further average over i. We then generated a single graph by setting $k = 10$, resulting in $71{,}510$ weighted edges. The way edge weights enter the learning process is explained in Section 4.

We associated binary one-vs-all classification tasks with the categories occurring in the datasets. In particular, we used the four top categories of the RCV1 taxonomy and all the three categories appearing in the DBLP network. Finally, since some of the seven categories occurring in the CORA network are rather rare, we randomly grouped the labels so to obtain three balanced metaclasses. We did this so to avoid dealing with the problem of unbalanced classes, which is outside the scope of this work.

All the resulting $4{+}3{+}3 = 10$ binary labeled datasets we have used are available for download.[3] The following table summarizes the statistics of each dataset, including the normalized cutsize $\Phi/|E|$ and the class distribution.

	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert E\rvert/\lvert V\rvert$	$\Phi/\lvert E\rvert$	CLASSES	CLASS DISTRIBUTION
RCV1	10,000	71,510	7.15	14.02	4	{45.06, 25.54, 24.91, 14.75}
DBLP	13,146	326,082	24.80	5.26	3	{43.22, 33.74, 23.04}
CORA	2,484	5,205	2.10	10.95	3	{43.12, 36.84, 30.60}

For each dataset and binary task we used various training and test set sizes, ranging from the $0.5\% - 99.5\%$ train-test split to $25\% - 75\%$. More details are given in Section 4.

On all such splits, we first ran an active learning algorithm to select the training set L, and then ran the standard *label propagation* algorithm [20, 4] for predicting the labels of the remaining nodes using L as training set. Whenever the selection algorithm was randomized we repeated the experiment ten times and averaged the results. Note that label propagation naturally takes the edge weights into account.

We now move on to describing the active learning algorithms used in the experiments. As mentioned in Section 2, we applied the selection algorithm SEL from [8] on two types of spanning trees.

[1] Available at `www.mpi-inf.mpg.de/~angelova/`. See [1].

[2] Available at `www.cs.umd.edu/projects/linqs/projects/lbc/`.

[3] At `homes.dsi.unimi.it/~cesabian/Netdata/`.

Table 1: Test error rates (percentages) achieved by the various algorithms on the ten datasets mentioned in the main text. For brevity, percentages are macroaveraged over the binary tasks associated with each dataset. SPLIT denotes the training set size. For instance, 0.5% means that the algorithm used 0.5% train and 99.5% test. For a given dataset and training/test split, we denote in boldface the lowest errors over the algorithms (i.e., column-wise). The variance of the performance of SEL[$*$] and MSEL[$*$] over the draw of RST and BFST is quite low. On the contrary, the performance of RND suffers, as expected, from a big variance, especially when the training set is small.

DATASET	DBLP				RCV1			CORA		
SPLIT	0.25%	0.5%	1%	5%	0.5%	1%	5%	1%	5%	25%
ALG.										
RND	29.81	21.19	11.65	5.23	25.90	23.68	15.96	25.22	14.03	9.61
HD	43.97	18.77	18.23	8.50	29.50	26.14	17.32	21.63	13.77	11.34
SEL[RST]	**12.64**	16.59	9.35	4.71	**22.75**	22.43	**15.42**	19.90	14.13	9.82
SEL[BFST]	27.30	**9.16**	7.03	4.55	26.32	26.33	16.63	**17.86**	14.26	**9.46**
MSEL[RST]	17.62	13.19	9.37	5.23	24.79	**22.40**	16.07	18.57	13.08	9.88
MSEL[BFST]	13.72	9.81	**6.37**	**4.34**	29.07	27.65	17.22	18.70	**13.02**	9.66

Random Spanning Tree (RST). A spanning tree is drawn with probability proportional to the product of its edge weights —see, e.g., Ch. 4 of [13]. As shown in [17], it is possible to generate (unweighted) random spanning trees in time *linear* in the number of nodes for many and important classes of graphs.

Breadth-first spanning tree (BFST). The spanning tree is created via a randomized breadth-first visit where the root is selected at random and the child nodes at each level are also picked at random. Breadth-first spanning trees are faster to generate than random ones.

We abbreviate the combination of SEL with these trees by SEL[RST] and SEL[BFST].

In an attempt to make SEL more robust with respect to the variability in the choice of the spanning tree, we also ran SEL on *committees* of spanning trees. We generated five spanning trees, applied the SEL algorithm to each one of them, and then selected the training nodes by majority vote. The resulting combinations are denoted by MSEL[RST] and MSEL[BFST].

We compared these variants of SEL against two natural baselines: random selection (RND), which simply picks the training nodes at random among all nodes, and high-degree (HD), which ranks the nodes in decreasing order according to their degree in the graph and then selects the top-ranked nodes. This second heuristic attempts to pick training nodes from dense clusters, but unlike SEL[BFST], here there is a higher risk that some of the clusters are missed out.

4 Results and comments

Table 1 gives the (average) fraction of test set prediction mistakes achieved by the various algorithms on the ten binary tasks. For brevity, we further aggregated the results by macroaveraging over the binary tasks in each dataset. Though the results are not conclusive, some interesting observations can be made.

1. SEL[RST] is never worse than RND and HD, and substantially better than both of them when the training size is small. On the contrary, in two cases out of three HD is the worst performer for small training sizes.
2. As the training set gets larger, all algorithms improve and perform similarly, except for HD which improves less than the others.
3. On RCV1, RND consistently outperforms SEL[BFST]. This could be explained by observing that RCV1 has a regular topology with very few high-degree nodes, whereas SEL[BFST] is especially effective when the graph is structured in dense clusters. On the other hand, SEL[RST] still manages to achieve a performance better than RND.
4. MSEL[$*$] does not seem to provide a consistent advantage over SEL[$*$]. An exception is DBLP, where SEL[BFST] is better than SEL[RST] (except for the smallest training size) and

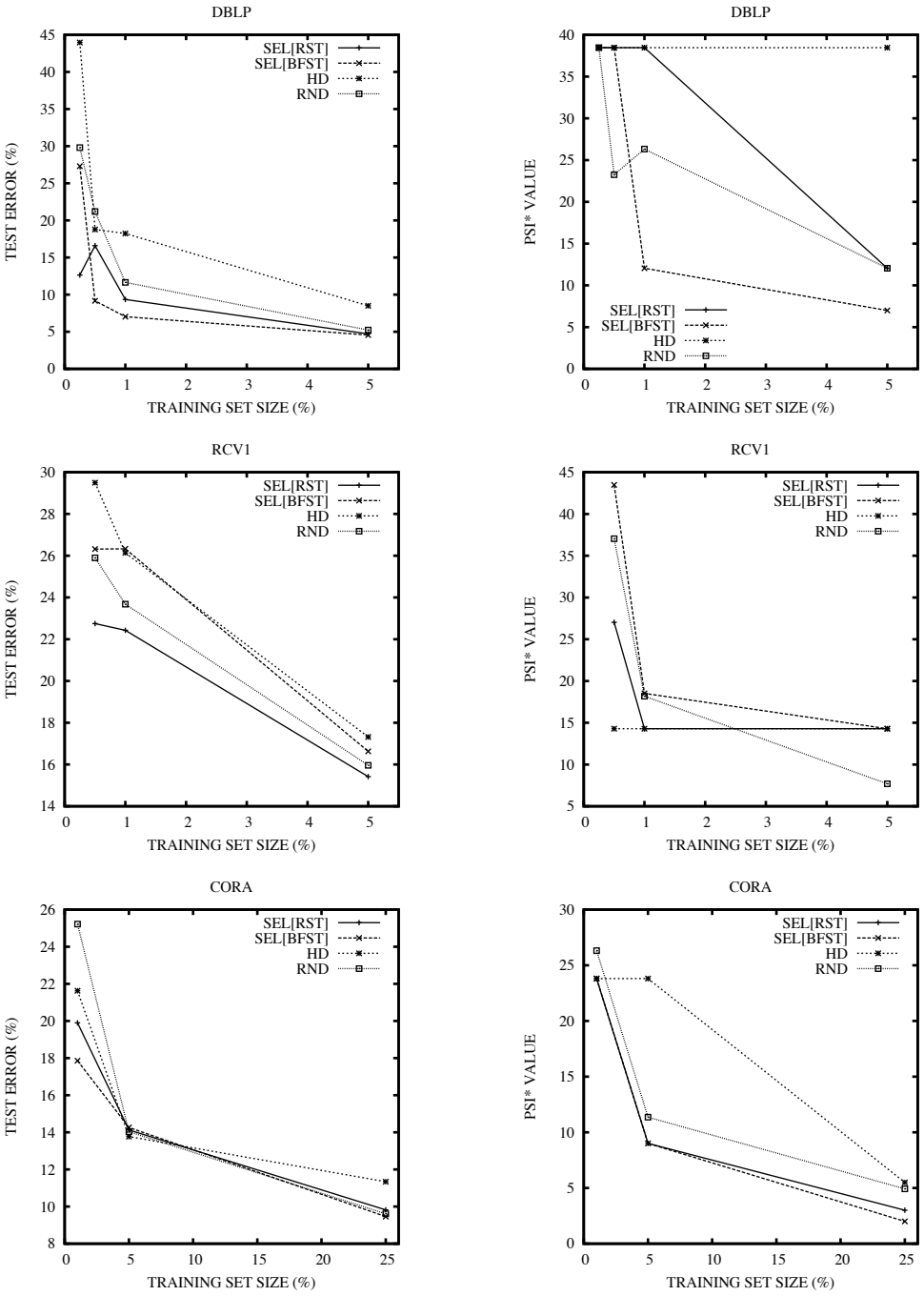

Figure 2: See main text for a description.

MSEL[BFST] improves on SEL[BFST]. This might be due to the fact that DBLP is irregular with many high-degree nodes. This is the ideal scenario for breadth-first trees, and MSEL here has chance to effectively reduce the variance in the tree construction.

In order to gain a better understanding of the comparative behavior of the algorithms, in Figure 2 we plotted back to back the test error rates of RND, HD, and SEL[$*$], and the corresponding average value of $\Psi^*(L)$ for the chosen training sets L. As evinced by these plots, there is a positive correlation between small test set accuracy and small value of Ψ^*.

Acknowledgments

We warmly thank Andrew Guillory for providing us with his code for computing $\Psi^*(L)$. This work was supported in part by Google Inc. through a Google Research Award, and by the PASCAL2 Network of Excellence (EC grant no. 216886). This publication only reflects the authors views.

References

[1] R. Angelova and G. Weikum. Graph-based text classification: learn from your neighbors In *Proc. 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 485–492, 2006.

[2] P. Afshani, E. Chiniforooshan, R. Dorrigiv, A. Farzan, M. Mirzazadeh, N. Simjour, H. Zarrabi-Zadeh. On the complexity of finding an unknown cut via vertex queries. In *COCOON 2007*, pages 459–469, 2007.

[3] M. Belkin, I. Matveeva, and P. Niyogi. Regularization and semi-supervised learning on large graphs. In *Proc. 17th COLT*, 2004.

[4] Y. Bengio , O. Delalleau, N. Le Roux. Label propagation and quadratic criterion. In O. Chapelle, B. Schlkopf and A. Zien Eds. *Semi-Supervised Learning*, pages 193–216. MIT Press, 2006.

[5] Mustafa Bilgic, Lilyana Mihalkova, and Lise Getoor. Active Learning for Networked Data. In *Proc. of ICML 2010*.

[6] P.M. Blau. Inequality and Heterogeneity: A Primitive Theory of Social Structure. New York: Free Press, 1977.

[7] A. Blum, and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proc. of ICML 2001*, pages 19–26, 2001.

[8] N. Cesa-Bianchi, C. Gentile, F. Vitale, G. Zappella. Active learning on trees and graphs. In *Proc. of COLT 2010*.

[9] N. Cesa-Bianchi, C. Gentile, F. Vitale, G. Zappella. Random spanning trees and the prediction of weighted graphs. In *Proc. of ICML 2010*.

[10] A. Guillory and J. Bilmes. Label Selection on Graphs. In *Advances in Neural Information Processing Systems 23*, MIT Press, 2010.

[11] M. Herbster, M. Pontil, S. Rojas-Galeano. Fast prediction on a tree. In *Advances in Neural Information Processing Systems 22*, MIT Press, 2009.

[12] M. Herbster, G. Lever, M. Pontil. Online prediction on large diameter graphs. In *Advances in Neural Information Processing Systems 22*, MIT Press, 2009.

[13] R. Lyons and Y. Peres. Probability on Trees and Networks. Manuscript, 2009.

[14] J. Long, J. Yin, W. Zhao, and E. Zhu. Graph-Based Active Learning Based on Label Propagation. In *Modeling Decisions for Artificial Intelligence*, Lecture Notes in Computer Science 5285/2008, pages 179–190, 2008.

[15] W. Zhao, J. Long, E. Zhu, and Y. Liu. A Scalable Algorithm for Graph-Based Active Learning. In *Frontiers in Algorithmics*, Lecture Notes in Computer Science 5059/2008, pages 311–322, 2008.

[16] M. McPherson, L. Smith-Lovin, J. M. Cook. Birds of a Feather: Homophily in Social Networks. Annual Review of Sociology, 27: pages 415–444, 2001.

[17] Wilson, D.B. Generating random spanning trees more quickly than the cover time. In *Proc. 28th ACM STOC*, pp. 296–303. ACM Press, 1996.

[18] D. Zhou, O. Bousquet, T. Lal, J. Weston, B. Schlkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*, MIT Press, 2004.

[19] X. Zhu. and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, 2002.

[20] X. Zhu, Z. Ghahramani, J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In ICML Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining, 2003.