Journal of Machine Learning Research 14 (2013) 1251-1284 Submitted 10/11; Revised 9/12; Published 5/13

Random Spanning Trees and the Prediction of Weighted Graphs

Nicolò Cesa-Bianchi NICOLO.CESA-BIANCHI@UNIMI.IT
Dipartimento di Informatica
Università degli Studi di Milano
via Comelico 39
20135 - Milano, Italy

Claudio Gentile CLAUDIO.GENTILE@UNINSUBRIA.IT
DiSTA
Università dell'Insubria
via Mazzini 5
21100 - Varese, Italy

Fabio Vitale FABIO.VITALE@UNIMI.IT
Dipartimento di Informatica
Università degli Studi di Milano
via Comelico 39
20135 - Milano, Italy

Giovanni Zappella GIOVANNI.ZAPPELLA@UNIMI.IT
Dipartimento di Matematica
Università degli Studi di Milano
via Saldini 50
20133 - Milano, Italy

Editor: Shie Mannor

Abstract

We investigate the problem of sequentially predicting the binary labels on the nodes of an arbitrary weighted graph. We show that, under a suitable parametrization of the problem, the optimal number of prediction mistakes can be characterized (up to logarithmic factors) by the cutsize of a random spanning tree of the graph. The cutsize is induced by the unknown adversarial labeling of the graph nodes. In deriving our characterization, we obtain a simple randomized algorithm achieving in expectation the optimal mistake bound on any polynomially connected weighted graph. Our algorithm draws a random spanning tree of the original graph and then predicts the nodes of this tree in constant expected amortized time and linear space. Experiments on real-world data sets show that our method compares well to both global (Perceptron) and local (label propagation) methods, while being generally faster in practice.

Keywords: online learning, learning on graphs, graph prediction, random spanning trees

1. Introduction

A widespread approach to the solution of classification problems is representing data sets through a weighted graph where nodes are the data items and edge weights quantify the similarity between pairs of data items. This technique for coding input data has been applied to several domains, including Web spam detection (Herbster et al., 2009b), classification of genomic data (Tsuda and

Schölkopf, 2009), face recognition (Chang and Yeung, 2006), and text categorization (Goldberg and Zhu, 2004). In many applications, edge weights are computed through a complex data-modeling process and typically convey information that is relevant to the task of classifying the nodes.

In the sequential version of this problem, nodes are presented in an arbitrary (possibly adversarial) order, and the learner must predict the binary label of each node before observing its true value. Since real-world applications typically involve large data sets (i.e., large graphs), online learning methods play an important role because of their good scaling properties. An interesting special case of the online problem is the so-called transductive setting, where the entire graph structure (including edge weights) is known in advance. The transductive setting is interesting in that the learner has the chance of reconfiguring the graph before learning starts, so as to make the problem look easier. This data preprocessing can be viewed as a kind of regularization in the context of graph prediction.

When the graph is unweighted (i.e., when all edges have the same common weight), it was found in previous works (Herbster et al., 2005; Herbster and Pontil, 2007; Herbster, 2008; Herbster and Lever, 2009) that a key parameter to control the number of online prediction mistakes is the size of the cut induced by the unknown adversarial labeling of the nodes, that is, the number of edges in the graph whose endpoints are assigned disagreeing labels. However, while the number of mistakes is obviously bounded by the number of nodes, the cutsize scales with the number of edges. This naturally led to the idea of solving the prediction problem on a spanning tree of the graph (Cesa-Bianchi et al., 2009; Herbster et al., 2009a,b), whose number of edges is exactly equal to the number of nodes minus one. Now, since the cutsize of the spanning tree is smaller than that of the original graph, the number of mistakes in predicting the nodes is more tightly controlled. In light of the previous discussion, we can also view the spanning tree as a "maximally regularized" version of the original graph.

Since a graph has up to exponentially many spanning trees, which one should be used to maximize the predictive performance? This question can be answered by recalling the adversarial nature of the online setting, where the presentation of nodes and the assignment of labels to them are both arbitrary. This suggests to pick a tree at random among all spanning trees of the graph so as to prevent the adversary from concentrating the cutsize on the chosen tree (Cesa-Bianchi et al., 2009). Kirchoff's equivalence between the effective resistance of an edge and its probability of being included in a random spanning tree allows to express the expected cutsize of a random spanning tree in a simple form. Namely, as the sum of resistances over all edges in the cut of G induced by the adversarial label assignment.

Although the results of Cesa-Bianchi et al. (2009) yield a mistake bound for arbitrary unweighted graphs in terms of the cutsize of a random spanning tree, no general lower bounds are known for online unweighted graph prediction. The scenario gets even more uncertain in the case of weighted graphs, where the only previous papers we are aware of Herbster and Pontil (2007), Herbster (2008), and Herbster and Lever (2009) essentially contain only upper bounds. In this paper we fill this gap, and show that the expected cutsize of a random spanning tree of the graph delivers a convenient parametrization[1] that captures the hardness of the graph learning problem in the general weighted case. Given any weighted graph, we prove that any online prediction algorithm must err on a number of nodes which is at least as big as the expected cutsize of the graph's random spanning tree (which is defined in terms of the graph weights). Moreover, we exhibit a simple randomized algorithm achieving in expectation the optimal mistake bound to within logarithmic

1. Different parametrizations of the node prediction problem exist that lead to bounds which are incomparable to ours—see Section 2.

factors. This bound applies to any sufficiently connected weighted graph whose weighted cutsize is not an overwhelming fraction of the total weight.

Following the ideas of Cesa-Bianchi et al. (2009), our algorithm first extracts a random spanning tree of the original graph. Then, it predicts all nodes of this tree using a generalization of the method proposed by Herbster et al. (2009a). Our tree prediction procedure is extremely efficient: it only requires *constant* amortized time per prediction and space *linear in the number of nodes*. Again, we would like to stress that computational efficiency is a central issue in practical applications where the involved data sets can be very large. In such contexts, learning algorithms whose computation time scales quadratically, or slower, in the number of data points should be considered impractical.

As in the work by Herbster et al. (2009a), our algorithm first linearizes the tree, and then operates on the resulting line graph via a nearest neighbor rule. We show that, besides running time, this linearization step brings further benefits to the overall prediction process. In particular, similar to Herbster and Pontil (2007, Theorem 4.2), the algorithm turns out to be resilient to perturbations of the labeling, a clearly desirable feature from a practical standpoint.

In order to provide convincing empirical evidence, we also present an experimental evaluation of our method compared to other algorithms recently proposed in the literature on graph prediction. In particular, we test our algorithm against the Perceptron algorithm with Laplacian kernel by Herbster and Pontil (2007); Herbster et al. (2009b), and against a version of the label propagation algorithm by Zhu et al. (2003). These two baselines can viewed as representatives of global (Perceptron) and local (label propagation) learning methods on graphs. The experiments have been carried out on five medium-sized real-world data sets. The two tree-based algorithms (ours and the Perceptron algorithm) have been tested using spanning trees generated in various ways, including committees of spanning trees aggregated by majority votes. In a nutshell, our experimental comparison shows that predictors based on our online algorithm compare well to all baselines while being very efficient in most cases.

The paper is organized as follows. Next, we recall preliminaries and introduce our basic notation. Section 2 surveys related work in the literature. In Section 3 we prove the general lower bound relating the mistakes of any prediction algorithm to the expected cutsize of a random spanning tree of the weighted graph. In the subsequent section, we present our prediction algorithm WTA (Weighted Tree Algorithm), along with a detailed mistake bound analysis restricted to weighted trees. This analysis is extended to weighted graphs in Section 5, where we provide an upper bound matching the lower bound up to log factors on any sufficiently connected graph. In Section 6, we quantify the robustness of our algorithm to label perturbation. In Section 7, we provide the constant amortized time implementation of WTA. Based on this implementation, in Section 8 we present the experimental results. Section 9 is devoted to conclusive remarks.

1.1 Preliminaries and Basic Notation

Let $G = (V, E, W)$ be an undirected, connected, and weighted graph with n nodes and positive edge weights $w_{i,j} > 0$ for $(i, j) \in E$. A labeling of G is any assignment $y = (y_1, \ldots, y_n) \in \{-1, +1\}^n$ of binary labels to its nodes. We use (G, y) to denote the resulting labeled weighted graph.

The online learning protocol for predicting (G, y) can be defined as the following game between a (possibly randomized) learner and an adversary. The game is parameterized by the graph $G = (V, E, W)$. Preliminarily, and hidden to the learner, the adversary chooses a labeling y of G. Then the nodes of G are presented to the learner one by one, according to a permutation of V, which

is adaptively selected by the adversary. More precisely, at each time step $t = 1, \ldots, n$ the adversary chooses the next node i_t in the permutation of V, and presents it to the learner for the prediction of the associated label y_{i_t}. Then y_{i_t} is revealed, disclosing whether a mistake occurred. The learner's goal is to minimize the total number of prediction mistakes. Note that while the adversarial choice of the permutation can depend on the algorithm's randomization, the choice of the labeling is oblivious to it. In other words, the learner uses randomization to fend off the adversarial choice of labels, whereas it is fully deterministic against the adversarial choice of the permutation. The requirement that the adversary is fully oblivious when choosing labels is then dictated by the fact that the randomized learners considered in this paper make all their random choices at the beginning of the prediction process (i.e., before seeing the labels).

Now, it is reasonable to expect that prediction performance degrades with the increase of "randomness" in the labeling. For this reason, our analysis of graph prediction algorithms bounds from above the number of prediction mistakes in terms of appropriate notions of graph label *regularity*. A standard notion of label regularity is the cutsize of a labeled graph, defined as follows. A ϕ-edge of a labeled graph (G,y) is any edge (i,j) such that $y_i \neq y_j$. Similarly, an edge (i,j) is ϕ-free if $y_i = y_j$. Let $E^\phi \subseteq E$ be the set of ϕ-edges in (G,y). The quantity $\Phi_G(y) = \left|E^\phi\right|$ is the *cutsize* of (G,y), that is, the number of ϕ-edges in E^ϕ (independent of the edge weights). The *weighted cutsize* of (G,y) is defined by

$$\Phi_G^W(y) = \sum_{(i,j)\in E^\phi} w_{i,j} \ .$$

For a fixed (G,y), we denote by $r_{i,j}^W$ the effective resistance between nodes i and j of G. In the interpretation of the graph as an electric network, where the weights $w_{i,j}$ are the edge conductances, the effective resistance $r_{i,j}^W$ is the voltage between i and j when a unit current flow is maintained through them. For $(i,j) \in E$, let also $p_{i,j} = w_{i,j} r_{i,j}^W$ be the probability that (i,j) belongs to a random spanning tree T—see, for example, the monograph of Lyons and Peres (2009). Then we have

$$\mathbb{E}\,\Phi_T(y) = \sum_{(i,j)\in E^\phi} p_{i,j} = \sum_{(i,j)\in E^\phi} w_{i,j} r_{i,j}^W \ , \tag{1}$$

where the expectation $\mathbb{E}$ is over the random choice of spanning tree T. Observe the natural weight-scale independence properties of (1). A uniform rescaling of the edge weights $w_{i,j}$ cannot have an influence on the probabilities $p_{i,j}$, thereby making each product $w_{i,j} r_{i,j}^W$ scale independent. In addition, since $\sum_{(i,j)\in E} p_{i,j}$ is equal to $n-1$, irrespective of the edge weighting, we have $0 \leq \mathbb{E}\,\Phi_T(y) \leq n-1$. Hence the ratio $\frac{1}{n-1}\mathbb{E}\,\Phi_T(y) \in [0,1]$ provides a *density-independent* measure of the cutsize in G, and even allows to compare labelings on different graphs.

Now contrast $\mathbb{E}\,\Phi_T(y)$ to the more standard weighted cutsize measure $\Phi_G^W(y)$. First, $\Phi_G^W(y)$ is clearly weight-scale dependent. Second, it can be much larger than n on dense graphs, even in the unweighted $w_{i,j} = 1$ case. Third, it strongly depends on the density of G, which is generally related to $\sum_{(i,j)\in E} w_{i,j}$. In fact, $\mathbb{E}\,\Phi_T(y)$ can be much smaller than $\Phi_G^W(y)$ when there are strongly connected regions in G contributing prominently to the weighted cutsize. To see this, consider the following scenario: If $(i,j) \in E^\phi$ and $w_{i,j}$ is large, then (i,j) gives a big contribution to $\Phi_G^W(y)$ (it is easy to see that in such cases $\Phi_G^W(y)$ can be much larger than n). However, this does not necessarily happen with $\mathbb{E}\,\Phi_T(y)$. In fact, if i and j are strongly connected (i.e., if there are many disjoint paths connecting them), then $r_{i,j}^W$ is very small and so are the terms $w_{i,j} r_{i,j}^W$ in (1). Therefore, the effect of the large weight $w_{i,j}$ may often be compensated by the small probability of including (i,j) in the random spanning tree. See Figure 1 for an example.

A different way of taking into account graph connectivity is provided by the covering ball approach taken by Herbster (2008) and Herbster and Lever (2009)—see the next section.

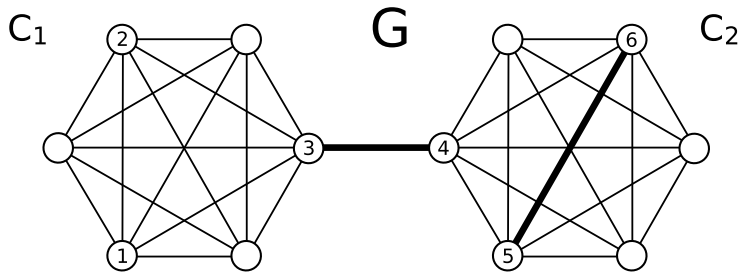

Figure 1: A barbell graph. The weight of the two thick black edges is equal to $\sqrt{V}$, all the other edges have unit weight. If the two labels y_1 and y_2 are such that $y_1 \neq y_2$, then the contribution of the edges on the left clique C_1 to the cutsizes $\Phi_G(y)$ and $\Phi_G^W(y)$ must be large. However, since the probability of including each edge of C_1 in a random spanning tree T is $O(1/|V|)$, C_1's contribution to $\mathbb{E}\,\Phi_T(y)$ is $|V|$ times smaller than $\Phi_{C_1}(y) = \Phi_{C_1}^W(y)$. If $y_3 \neq y_4$, then the contribution of edge (3,4) to $\Phi_G^W(y)$ is large. Because this edge is a bridge, the probability of including it in T is one, independent of $w_{3,4}$. Indeed, we have $p_{3,4} = w_{3,4}\, r_{3,4}^W = w_{3,4}/w_{3,4} = 1$. If $y_5 \neq y_6$, then the contribution of the right clique C_2 to $\Phi_G^W(y)$ is large. On the other hand, the probability of including edge $(5,6)$ in T is equal to $p_{5,6} = w_{5,6}\, r_{5,6}^W = O(1/\sqrt{|V|})$. Hence, the contribution of $(5,6)$ to $\mathbb{E}\,\Phi_T(y)$ is small because the large weight of $(5,6)$ is offset by the fact that nodes 5 and 6 are strongly connected (i.e., there are many different paths among them). Finally, note that $p_{i,j} = O(1/|V|)$ holds for all edges (i,j) in C_2, implying (similar to clique C_1) that C_2's contribution to $\mathbb{E}\,\Phi_T(y)$ is $|V|$ times smaller than $\Phi_{C_2}^W(y)$.

2. Related Work

With the above notation and preliminaries in hand, we now briefly survey the results in the existing literature which are most closely related to this paper. Further comments are made at the end of Section 5.

Standard online linear learners, such as the Perceptron algorithm, are applied to the general (weighted) graph prediction problem by embedding the n vertices of the graph in $\mathbb{R}^n$ through a map $i \mapsto K^{-1/2} e_i$, where $e_i \in \mathbb{R}^n$ is the i-th vector in the canonical basis of $\mathbb{R}^n$, and K is a positive definite $n \times n$ matrix. The graph Perceptron algorithm (Herbster et al., 2005; Herbster and Pontil, 2007) uses $K = L_G + 1\,1^\top$, where L_G is the (weighted) Laplacian of G and $1 = (1,\ldots,1)$. The resulting mistake bound is of the form $\Phi_G^W(y) D_G^W$, where $D_G^W = \max_{i,j} r_{i,j}^W$ is the resistance diameter of G. As expected, this bound is weight-scale independent, but the interplay between the two factors in it may lead to a vacuous result. At a given scale for the weights $w_{i,j}$, if G is dense, then we may have $D_G^W = O(1)$ while $\Phi_G^W(y)$ is of the order of n^2. If G is sparse, then $\Phi_G^W(y) = O(n)$ but then D_G^W may become as large as n.

The idea of using a spanning tree to reduce the cutsize of G has been investigated by Herbster et al. (2009b), where the graph Perceptron algorithm is applied to a spanning tree T of G. The

resulting mistake bound is of the form $\Phi_T^W(y)D_T^W$, that is, the graph Perceptron bound applied to tree T. Since $\Phi_T^W(y) \le \Phi_G^W(y)$ this bound has a smaller cutsize than the previous one. On the other hand, D_T^W can be much larger than D_G^W because removing edges may increase the resistance. Hence the two bounds are generally incomparable.

Herbster et al. (2009b) suggest to apply the graph Perceptron algorithm to the spanning tree T with smallest geodesic diameter. The geodesic diameter of a weighted graph G is defined by

$$\Delta_G^W = \max_{i,j} \min_{\Pi_{i,j}} \sum_{(r,s)\in\Pi_{i,j}} \frac{1}{w_{i,j}} ,$$

where the minimum is over all paths $\Pi_{i,j}$ between i and j. The reason behind this choice of T is that, for the spanning tree T with smallest geodesic diameter, it holds that $D_T^W \le 2\Delta_G^W$. However, one the one hand $D_G^W \le \Delta_G^W$, so there is no guarantee that $D_T^W = O\big(D_G^W\big)$, and on the other hand the adversary may still concentrate all ϕ-edges on the chosen tree T, so there is no guarantee that $\Phi_T^W(y)$ remains small either.

Herbster et al. (2009a) introduce a different technique showing its application to the case of unweighted graphs. After reducing the graph to a spanning tree T, the tree is linearized via a depth-first visit. This gives a line graph S (the so-called *spine* of G) such that $\Phi_S(y) \le 2\,\Phi_T(y)$. By running a Nearest Neighbor (NN) predictor on S, Herbster et al. (2009a) prove a mistake bound of the form $\Phi_S(y)\log\big(n/\Phi_S(y)\big) + \Phi_S(y)$. As observed by Fakcharoenphol and Kijsirikul (2008), similar techniques have been developed to solve low-congestion routing problems.

Another natural parametrization for the labels of a weighted graph that takes the graph structure into account is *clusterability*, that is, the extent to which the graph nodes can be covered by a few balls of small resistance diameter. With this inductive bias in mind, Herbster (2008) developed the Pounce algorithm, which can be seen as a combination of graph Perceptron and NN prediction. The number of mistakes has a bound of the form

$$\min_{\rho>0}\big(\mathcal{N}(G,\rho) + \Phi_G^W(y)\rho\big) , \tag{2}$$

where $\mathcal{N}(G,\rho)$ is the smallest number of balls of resistance diameter ρ it takes to cover the nodes of G. Note that the graph Perceptron bound is recovered when $\rho = D_G^W$. Moreover, observe that, unlike graph Perceptron's, bound (2) is never vacuous, as it holds uniformly for all covers of G (even the one made up of singletons, corresponding to $\rho \to 0$). A further trick for the unweighted case proposed by Herbster et al. (2009a) is to take advantage of both previous approaches (graph Perceptron and NN on line graphs) by building a binary tree on G. This "support tree" helps in keeping the diameter of G as small as possible, for example, logarithmic in the number of nodes n. The resulting prediction algorithm is again a combination of a Perceptron-like algorithm and NN, and the corresponding number of mistakes is the minimum over two earlier bounds: a NN-based bound of the form $\Phi_G(y)(\log n)^2$ and an unweighted version of bound (2).

Generally speaking, clusterability and resistance-weighted cutsize $\mathbb{E}\,\Phi_T(y)$ exploit the graph structure in different ways. Consider, for instance, a barbell graph made up of two m-cliques joined by k unweighted ϕ-edges with no endpoints in common (hence $k \le m$). This is one of the examples considered by Herbster and Lever (2009). If m is much larger than k, then bound (2) scales linearly with k (the two balls in the cover correspond to the two m-cliques). On the other hand, $\mathbb{E}\,\Phi_T(y)$ tends to be constant: Because m is much larger than k, the probability of including any ϕ-edge in T tends to $1/k$, as m increases and k stays constant. On the other hand, if k gets close to m the

resistance diameter of the graph decreases, and (2) becomes a constant. In fact, one can show that when $k = m$ even $\mathbb{E}\,\Phi_T(y)$ is a constant, independent of m. In particular, the probability that a ϕ-edge is included in the random spanning tree T is upper bounded by $\frac{3m-1}{m(m+1)}$, that is, $\mathbb{E}\,\Phi_T(y) \to 3$ when m grows large. This can be shown by computing the effective resistance of ϕ-edge (i, j) as the minimum, over all unit-strength flow functions with i as source and j as sink, of the squared flow values summed over all edges, see, for example, Lyons and Peres (2009).

When the graph at hand has a large diameter, for example, an m-line graph connected to an m-clique (this is sometimes called a "lollipop" graph) the gap between the covering-based bound (2) and $\mathbb{E}\,\Phi_T(y)$ is magnified. Yet, it is fair to say that the bounds we are about to prove for our algorithm have an extra factor, beyond $\mathbb{E}\,\Phi_T(y)$, which is logarithmic in m. A similar logarithmic factor is achieved by the combined algorithm proposed by Herbster et al. (2009a).

An even more refined way of exploiting cluster structure and connectivity in graphs is contained in the paper of Herbster and Lever (2009), where the authors provide a comprehensive study of the application of dual-norm techniques to the prediction of weighted graphs, again with the goal of obtaining logarithmic performance guarantees on large diameter graphs. In order to trade-off the contribution of cutsize Φ_G^W and resistance diameter D_G^W, the authors develop a notion of p-norm resistance. The obtained bounds are dual norm versions of the covering ball bound (2). Roughly speaking, one can select the dual norm parameter of the algorithm to obtain a logarithmic contribution from the resistance diameter at the cost of squaring the contribution due to the cutsize. This quadratic term can be further reduced if the graph is well connected. For instance, in the unweighted barbell graph mentioned above, selecting the norm appropriately leads to a bound which is constant even when $k \ll m$.

Further comments on the comparison between the results presented by Herbster and Lever (2009) and the ones in our paper are postponed to the end of Section 5.

Departing from the online learning scenario, it is worth mentioning the significantly large literature on the general problem of learning the nodes of a graph in the train/test transductive setting: Many algorithms have been proposed, including the label-consistent mincut approach of Blum and Chawla (2001), Blum et al. (2004) and a number of other "energy minimization" methods—for example, the ones by Zhu et al. (2003) and Belkin et al. (2004) of which label propagation is an instance. See the work of Bengio et al. (2006) for a relatively recent survey on this subject.

Our graph prediction algorithm is based on a random spanning tree of the original graph. The problem of drawing a random spanning tree of an arbitrary graph has a long history—see, for example, the monograph by Lyons and Peres (2009). In the unweighted case, a random spanning tree can be sampled with a random walk in expected time $O(n \ln n)$ for "most" graphs, as shown by Broder (1989). Using the beautiful algorithm of Wilson (1996), the expected time reduces to $O(n)$—see also the work of Alon et al. (2008). However, all known techniques take expected time $\Theta(n^3)$ on certain pathological graphs. In the weighted case, the above methods can take longer due to the hardness of reaching, via a random walk, portions of the graph which are connected only via light-weighted edges. To sidestep this issue, in our experiments we tested a viable fast approximation where weights are disregarded when building the spanning tree, and only used at prediction time. Finally, the space complexity for generating a random spanning tree is always linear in the graph size.

To conclude this section, it is worth mentioning that, although we exploit random spanning trees to reduce the cutsize, similar approaches can also be used to approximate the cutsize of a weighted graph by sparsification—see, for example, the work of Spielman and Srivastava (2008). However,

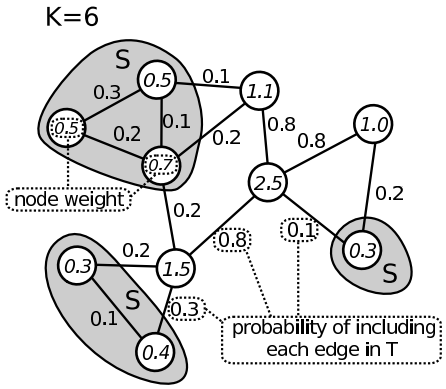

Figure 2: The adversarial strategy. Numbers on edges are the probabilities $p_{i,j}$ of those edges being included in a random spanning tree for the weighted graph under consideration. Numbers within nodes denote the weight of that node based on the $p_{i,j}$—see main text. We set the budget K to 6, hence the subset S contains the 6 nodes having smallest weight. The adversary assigns a random label to each node in S thus forcing $|S|/2$ mistakes in expectation. Then, it labels all nodes in $V \setminus S$ with a unique label, chosen in such a way as to minimize the cutsize consistent with the labels previously assigned to the nodes of S.

because the resulting graphs are not as sparse as spanning trees, we do not currently see how to use those results.

3. A General Lower Bound

This section contains our general lower bound. We show that any prediction algorithm must err at least $\frac{1}{2}\mathbb{E}\,\Phi_T(y)$ times on any weighted graph.

Theorem 1 *Let $G=(V,E,W)$ be a weighted undirected graph with n nodes and weights $w_{i,j}>0$ for $(i,j)\in E$. Then for all $K\le n$ there exists a randomized labeling y of G such that for all (deterministic or randomized) algorithms A, the expected number of prediction mistakes made by A is at least $K/2$, while $\mathbb{E}\,\Phi_T(y)<K$.*

Proof The adversary uses the weighting P induced by W and defined by $p_{i,j}=w_{i,j}r^W_{i,j}$. By (1), $p_{i,j}$ is the probability that edge (i,j) belongs to a random spanning tree T of G. Let $P_i=\sum_j p_{i,j}$ be the sum over the induced weights of all edges incident to node i. We call P_i the *weight* of node i. Let $S\subseteq V$ be the set of K nodes i in G having the smallest weight P_i. The adversary assigns a random label to each node $i\in S$. This guarantees that, no matter what, the algorithm A will make on average $K/2$ mistakes on the nodes in S. The labels of the remaining nodes in $V\setminus S$ are set either all $+1$ or all -1, depending on which one of the two choices yields the smaller $\Phi^P_G(y)$. See Figure 2 for an illustrative example. We now show that the weighted cutsize $\Phi^P_G(y)$ of this labeling y is less than K, *independent of* the labels of the nodes in S.

Since the nodes in $V\setminus S$ have all the same label, the ϕ-edges induced by this labeling can only connect either two nodes in S or one node in S and one node in $V\setminus S$. Hence $\Phi^P_G(y)$ can be written as

$$\Phi^P_G(y)=\Phi^{P,\mathrm{int}}_G(y)+\Phi^{P,\mathrm{ext}}_G(y)\ ,$$

where $\Phi_G^{P,int}(y)$ is the cutsize contribution within S, and $\Phi_G^{P,ext}(y)$ is the one from edges between S and $V \setminus S$. We can now bound these two terms by combining the definition of S with the equality $\sum_{(i,j)\in E} p_{i,j} = n-1$ as in the sequel. Let

$$P_S^{\text{int}} = \sum_{(i,j)\in E\,:\,i,j\in S} p_{i,j} \qquad \text{and} \qquad P_S^{\text{ext}} = \sum_{(i,j)\in E\,:\,i\in S,\,j\in V\setminus S} p_{i,j}\ .$$

From the very definition of P_S^{int} and $\Phi_G^{P,\text{int}}(y)$ we have $\Phi_G^{P,\text{int}}(y) \le P_S^{\text{int}}$. Moreover, from the way the labels of nodes in $V \setminus S$ are selected, it follows that $\Phi_G^{P,\text{ext}}(y) \le P_S^{\text{ext}}/2$. Finally,

$$\sum_{i\in S} P_i = 2P_S^{\text{int}} + P_S^{\text{ext}}$$

holds, since each edge connecting nodes in S is counted twice in the sum $\sum_{i\in S} P_i$. Putting everything together we obtain

$$2P_S^{\text{int}} + P_S^{\text{ext}} = \sum_{i\in S} P_i \le \frac{K}{n}\sum_{i\in V} P_i = \frac{2K}{n}\sum_{(i,j)\in E} p_{i,j} = \frac{2K(n-1)}{n}\ ,$$

the inequality following from the definition of S. Hence

$$\mathbb{E}\,\Phi_T(y) = \Phi_G^P(y) = \Phi_G^{P,\text{int}}(y) + \Phi_G^{P,\text{ext}}(y) \le P_S^{\text{int}} + \frac{P_S^{\text{ext}}}{2} \le \frac{K(n-1)}{n} < K$$

concluding the proof. ■

4. The Weighted Tree Algorithm

We now describe the Weighted Tree Algorithm (WTA) for predicting the labels of a weighted tree. In Section 5 we show how to apply WTA to the more general weighted graph prediction problem. WTA first transforms the tree into a line graph (i.e., a list), then runs a fast nearest neighbor method to predict the labels of each node in the line. Though this technique is similar to that one used by Herbster et al. (2009a), the fact that the tree is weighted makes the analysis significantly more difficult, and the practical scope of our algorithm significantly wider. Our experimental comparison in Section 8 confirms that exploiting the weight information is often beneficial in real-world graph prediction problem.

Given a labeled weighted tree (T,y), the algorithm initially creates a weighted line graph L' containing some duplicates of the nodes in T. Then, each duplicate node (together with its incident edges) is replaced by a single edge with a suitably chosen weight. This results in the final weighted line graph L which is then used for prediction. In order to create L from T, WTA performs the following *tree linearization* steps:

1. An arbitrary node r of T is chosen, and a line L' containing only r is created.

2. Starting from r, a depth-first visit of T is performed. Each time an edge (i,j) is traversed (even in a backtracking step) from i to j, the edge is appended to L' with its weight $w_{i,j}$, and j becomes the current terminal node of L'. Note that backtracking steps can create in L' at most one duplicate of each edge in T, while nodes in T may be duplicated several times in L'.

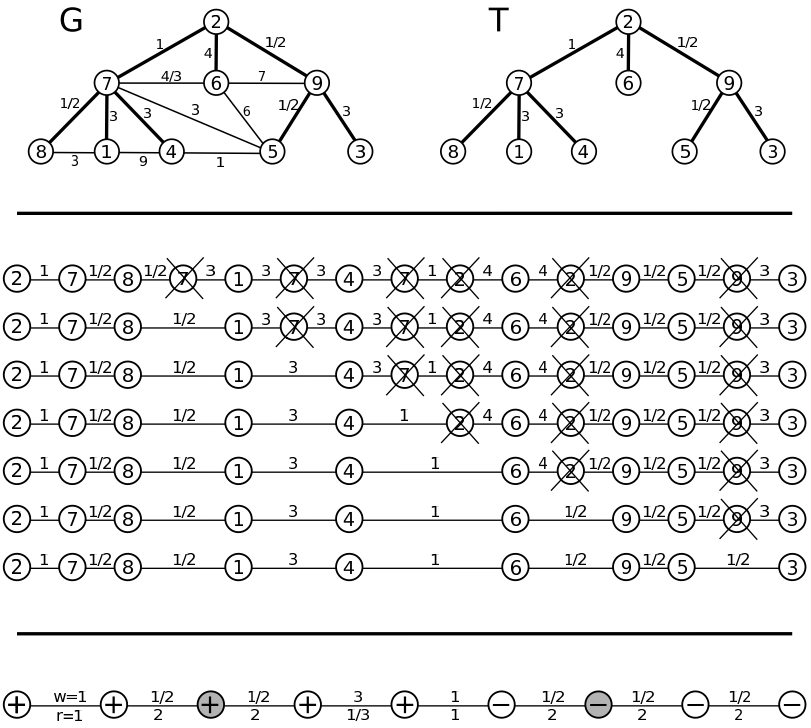

Figure 3: **Top:** A weighted graph G with 9 nodes. Initially, WTA extracts a random spanning tree T out of G. The weights on the edges in T are the same as those of G. **Middle:** The spanning tree T is linearized through a depth-first traversal starting from an arbitrary node (node 2 in this figure). For simplicity, we assume the traversal visits the siblings from left to right. As soon as a node is visited it gets stored in a line graph L' (first line graph from top). Backtracking steps produce duplicates in L' of some of the nodes in T. For instance, node 7 is the first node to be duplicated when the visit backtracks from node 8. The duplicated nodes are progressively eliminated from L' in the order of their insertion in L'. Several iterations of this node elimination process are displayed from the top to the bottom, showing how L' is progressively shrunk to the final line L (bottom line). Each line represents the elimination of a single duplicated node. The crossed nodes in each line are the nodes which are scheduled to be eliminated. Each time a new node j is eliminated, its two adjacent nodes i and k are connected by the lighter of the two edges (i, j) and (j,k). For instance: the left-most duplicated 7 is dropped by directly connecting the two adjacent nodes 8 and 1 by an edge with weight $1/2$; the right-most node 2 is eliminated by directly connecting node 6 to node 9 with an edge with weight $1/2$, and so on. Observe that this elimination procedure can be carried out *in any order* without changing the resulting list L. **Bottom:** We show WTA's prediction on the line L so obtained. In this figure, the numbers above the edges denote the edge weights, the ones below are the resistors, that is, weight reciprocals. We are at time step $t = 3$ where two labels have so far been revealed (gray nodes). WTA predicts on the remaining nodes according to a nearest neighbor rule on L, based on the resistance distance metric. All possible predictions made by WTA at this time step are shown.

3. L' is traversed once, starting from terminal r. During this traversal, duplicate nodes are eliminated as soon as they are encountered. This works as follows. Let j be a duplicate node, and (j',j) and (j,j'') be the two incident edges. The two edges are replaced by a new edge (j',j'') having weight $w_{j',j''} = \min\{w_{j',j}, w_{j,j''}\}$.[2] Let L be the resulting line.

The analysis of Section 4.1 shows that this choice of $w_{j',j''}$ guarantees that the weighted cutsize of L is smaller than twice the weighted cutsize of T.

Once L is created from T, the algorithm predicts the label of each node i_t using a nearest-neighbor rule operating on L with a *resistance distance* metric. That is, the prediction on i_t is the label of i_{s^*}, being $s^* = \operatorname{argmin}_{s<t} d(i_s, i_t)$ the previously revealed node closest to i_t, and $d(i,j) = \sum_{s=1}^{k} 1/w_{v_s,v_{s+1}}$ is the sum of the resistors (i.e., reciprocals of edge weights) along the unique path $i = v_1 \to v_2 \to \cdots \to v_{k+1} = j$ connecting node i to node j. Figure 3 gives an example of WTA at work.

4.1 Analysis of WTA

The following lemma gives a mistake bound on WTA run on any weighted line graph. Given any labeled graph (G,y), we denote by R_G^W the sum of resistors of ϕ-free edges in G,

$$R_G^W = \sum_{(i,j)\in E\setminus E^\phi} \frac{1}{w_{i,j}} \,.$$

Also, given any ϕ-free edge subset $E' \subset E \setminus E^\phi$, we define $R_G^W(\neg E')$ as the sum of the resistors of all ϕ-free edges in $E \setminus (E^\phi \cup E')$,

$$R_G^W(\neg E') = \sum_{(i,j)\in E\setminus(E^\phi\cup E')} \frac{1}{w_{i,j}} \,.$$

Note that $R_G^W(\neg E') \le R_G^W$, since we drop some edges from the sum in the defining formula.

Finally, we use $f \overset{O}{=} g$ as shorthand for $f = O(g)$. The following lemma is the starting point of our theoretical investigation—please see Appendix A for proofs.

Lemma 2 *If* WTA *is run on a labeled weighted line graph (L,y), then the total number m_L of mistakes satisfies*

$$m_L \overset{O}{=} \Phi_L(y)\left(1+\log\left(1+\frac{R_L^W(\neg E')\,\Phi_L^W(y)}{\Phi_L(y)}\right)\right) + |E'|$$

for all subsets E' of $E \setminus E^\phi$.

Note that the bound of Lemma 2 implies that, for any $K = |E'| \ge 0$, one can drop from the bound the contribution of any set of K resistors in R_L^W at the cost of adding K extra mistakes. We now provide an upper bound on the number of mistakes made by WTA on any weighted tree $T = (V,E,W)$ in terms of the number of ϕ-edges, the weighted cutsize, and R_T^W.

2. By iterating this elimination procedure, it might happen that more than two adjacent nodes get eliminated. In this case, the two surviving terminal nodes are connected in L by the lightest edge among the eliminated ones in L'.

Theorem 3 *If* WTA *is run on a labeled weighted tree (T,y), then the total number m_T of mistakes satisfies*

$$m_T \stackrel{O}{=} \Phi_T(y)\left(1+\log\left(1+\frac{R_T^W(\neg E')\,\Phi_T^W(y)}{\Phi_T(y)}\right)\right)+|E'|$$

for all subsets E' of $E\setminus E^{\phi}$.

The logarithmic factor in the above bound shows that the algorithm takes advantage of labelings such that the weights of ϕ-edges are small (thus making $\Phi_T^W(y)$ small) and the weights of ϕ-free edges are high (thus making R_T^W small). This matches the intuition behind WTA's nearest-neighbor rule according to which nodes that are close to each other are expected to have the same label. In particular, observe that the way the above quantities are combined makes the bound independent of rescaling of the edge weights. Again, this has to be expected, since WTA's prediction is scale insensitive. On the other hand, it may appear less natural that the mistake bound also depends linearly on the cutsize $\Phi_T(y)$, *independent of the edge weights*. The specialization to trees of our lower bound (Theorem 1 in Section 3) implies that this linear dependence of mistakes on the unweighted cutsize is necessary whenever the adversarial labeling is chosen from a set of labelings with bounded $\Phi_T(y)$.

5. Predicting a Weighted Graph

In order to solve the more general problem of predicting the labels of a weighted graph G, one can first generate a spanning tree T of G and then run WTA directly on T. In this case, it is possible to rephrase Theorem 3 in terms of the properties of G. Note that for each spanning tree T of G, $\Phi_T^W(y)\le\Phi_G^W(y)$ and $\Phi_T(y)\le\Phi_G(y)$. Specific choices of the spanning tree T control in different ways the quantities in the mistake bound of Theorem 3. For example, a minimum spanning tree tends to reduce the value of $\widetilde{R}_T^W$, betting on the fact that ϕ-edges are light. The next theorem relies on *random* spanning trees.

Theorem 4 *If* WTA *is run on a random spanning tree T of a labeled weighted graph (G,y), then the total number m_G of mistakes satisfies*

$$\mathbb{E}\,m_G \stackrel{O}{=} \mathbb{E}\big[\Phi_T(y)\big]\left(1+\log\left(1+w_{\max}^{\phi}\,\mathbb{E}\big[R_T^W\big]\right)\right)\ , \tag{3}$$

where $w_{\max}^{\phi}=\max_{(i,j)\in E^{\phi}} w_{i,j}$.

Note that the mistake bound in (3) is scale-invariant, since $\mathbb{E}\big[\Phi_T(y)\big]=\sum_{(i,j)\in E^{\phi}} w_{i,j}r_{i,j}^W$ cannot be affected by a uniform rescaling of the edge weights (as we said in Section 1.1), and so is the product $w_{\max}^{\phi}\mathbb{E}\big[R_T^W\big]=w_{\max}^{\phi}\sum_{(i,j)\in E\setminus E^{\phi}} r_{i,j}^W$.

We now compare the mistake bound (3) to the lower bound stated in Theorem 1. In particular, we prove that WTA is optimal (up to $\log n$ factors) on every weighted connected graph in which the ϕ-edge weights are not "superpolynomially overloaded" w.r.t. the ϕ-free edge weights. In order to rule out pathological cases, when the weighted graph is nearly disconnected, we impose the following mild assumption on the graphs being considered.

We say that a graph is *polynomially connected* if the ratio of any pair of effective resistances (even those between nonadjacent nodes) in the graph is polynomial in the total number of nodes

n. This definition essentially states that a weighted graph can be considered connected if no pair of nodes can be found which is substantially less connected than any other pair of nodes. Again, as one would naturally expect, this definition is independent of uniform weight rescaling. The following corollary shows that if WTA is not optimal on a polynomially connected graph, then the labeling must be so irregular that the total weight of ϕ-edges is an overwhelming fraction of the overall weight.

Corollary 5 *Pick any polynomially connected weighted graph G with n nodes. If the ratio of the total weight of ϕ-edges to the total weight of ϕ-free edges is bounded by a polynomial in n, then the total number of mistakes m_G made by* WTA *when run on a random spanning tree T of G satisfies*

$$\mathbb{E}\, m_G \stackrel{O}{=} \mathbb{E}\big[\Phi_T(y)\big] \log n \,.$$

Note that when the hypothesis of this corollary is not satisfied the bound of WTA is not necessarily vacuous. For example, $\mathbb{E}\big[R_T^W\big] w_{\max}^{\phi} = n^{\mathrm{polylog}(n)}$ implies an upper bound which is optimal up to $\mathrm{polylog}(n)$ factors. In particular, having a constant number of ϕ-free edges with exponentially large resistance contradicts the assumption of polynomial connectivity, but it need not lead to a vacuous bound in Theorem 4. In fact, one can use Lemma 2 to drop from the mistake bound of Theorem 4 the contribution of any set of $O(1)$ resistances in $\mathbb{E}\big[R_T^W\big] = \sum_{(i,j)\in E\setminus E^{\phi}} r_{i,j}^W$ at the cost of adding just $O(1)$ extra mistakes. This could be seen as a robustness property of WTA's bound against graphs that do not fully satisfy the connectedness assumption.

We further elaborate on the robustness properties of WTA in Section 6. In the meanwhile, note how Corollary 5 compares to the expected mistake bound of algorithms like graph Perceptron (see Section 2) on the same random spanning tree. This bound depends on the expectation of the product $\Phi_T^W(y) D_T^W$, where D_T^W is the diameter of T in the resistance distance metric. Recall from the discussion in Section 2 that these two factors are negatively correlated because $\Phi_T^W(y)$ depends linearly on the edge weights, while D_T^W depends linearly on the reciprocal of these weights. Moreover, for any given scale of the edge weights, D_T^W can be linear in the number n of nodes.

Another interesting comparison is to the covering ball bounds of Herbster (2008) and Herbster and Lever (2009). Consider the case when G is an unweighted tree with diameter D. Whereas the dual norm approach of Herbster and Lever (2009) gives a mistake bound of the form $\Phi_G(y)^2 \log D$, our approach, as well as the one by Herbster et al. (2009a), yields $\Phi_G(y) \log n$. Namely, the dependence on $\Phi_G(y)$ becomes linear rather than quadratic, but the diameter D gets replaced by n, the number of nodes in G. Replacing n by D seems to be a benefit brought by the covering ball approach.[3] More generally, one can say that the covering ball approach seems to allow to replace the extra $\log n$ term contained in Corollary 5 by more refined structural parameters of the graph (like its diameter D), but it does so at the cost of squaring the dependence on the cutsize. A typical (and unsurprising) example where the dual-norm covering ball bounds are better then the one in Corollary 5 is when the labeled graph is well-clustered. One such example we already mentioned in Section 2: On the unweighted barbell graph made up of m-cliques connected by $k \ll m$ ϕ-edges, the algorithm of Herbster and Lever (2009) has a *constant* bound on the number of mistakes (i.e., independent of both m and k), the Pounce algorithm has a *linear* bound in k, while Corollary 5 delivers a *logarithmic* bound in $m+k$. Yet, it is fair to point out that the bounds of Herbster (2008)

3. As a matter of fact, a bound of the form $\Phi_G(y) \log D$ on unweighted trees is also achieved by the direct analysis of Cesa-Bianchi et al. (2009).

and Herbster and Lever (2009) refer to computationally heavier algorithms than WTA: Pounce has a deterministic initialization step that computes the inverse Laplacian matrix of the graph (this is cubic in n, or quadratic in the case of trees), the minimum (Ψ, p)-seminorm interpolation algorithm of Herbster and Lever (2009) has no initialization, but each step requires the solution of a constrained convex optimization problem (whose time complexity was not quantified by the authors). Further comments on the time complexity of our algorithm are given in Section 7.

6. The Robustness of WTA to Label Perturbation

In this section we show that WTA is tolerant to noise, that is, the number of mistakes made by WTA on most labeled graphs (G,y) does not significantly change if a small number of labels are perturbed before running the algorithm. This is especially the case if the input graph G is polynomially connected (see Section 5 for a definition).

As in previous sections, we start off from the case when the input graph is a tree, and then we extend the result to general graphs using random spanning trees.

Suppose that the labels y in the tree (T,y) used as input to the algorithm have actually been obtained from another labeling y' of T through the perturbation (flipping) of some of its labels. As explained at the beginning of Section 4, WTA operates on a line graph L obtained through the linearization process of the input tree T. The following theorem shows that, whereas the cutsize differences $|\Phi_T^W(y) - \Phi_T^W(y')|$ and $|\Phi_T(y) - \Phi_T(y')|$ on tree T can in principle be very large, the cutsize differences $|\Phi_L^W(y) - \Phi_L^W(y')|$ and $|\Phi_L(y) - \Phi_L(y')|$ on the line graph L built by WTA are always small.

In order to quantify the above differences, we need a couple of ancillary definitions. Given a labeled tree (T,y), define $\zeta_T(K)$ to be the sum of the weights of the K heaviest edges in T,

$$\zeta_T(K) = \max_{E' \subseteq E \,:\, |E'|=K} \sum_{(i,j)\in E'} w_{i,j} \ .$$

If T is unweighted we clearly have $\zeta_T(K) = K$. Moreover, given any two labelings y and y' of T's nodes, we let $\delta(y,y')$ be the number of nodes for which the two labelings differ, that is, $\delta(y,y') = \big|\{i = 1,\ldots,n : y_i \neq y'_i\}\big|$.

Theorem 6 *On any given labeled tree (T,y) the tree linearization step of* WTA *generates a line graph L such that:*

1. $\Phi_L^W(y) \leq \min_{y' \in \{-1,+1\}^n} 2\Big(\Phi_T^W(y') + \zeta_T\big(\delta(y,y')\big)\Big)$;
2. $\Phi_L(y) \leq \min_{y' \in \{-1,+1\}^n} 2\big(\Phi_T(y') + \delta(y,y')\big)$.

In order to highlight the consequences of WTA's linearization step contained in Theorem 6, consider as a simple example an unweighted star graph (T,y) where all labels are $+1$ except for the central node c whose label is -1. We have $\Phi_T(y) = n-1$, but flipping the sign of y_c we would obtain the star graph (T,y') with $\Phi_T(y') = 0$. Using Theorem 6 (Item 2) we get $\Phi_L(y) \leq 2$. Hence, on this star graph WTA's linearization step generates a line graph with a constant number of ϕ-edges even if the input tree T has no ϕ-free edges. Because flipping the labels of a few nodes (in this case the label of c) we obtain a tree with a much more regular labeling, the labels of those nodes can naturally be seen as corrupted by noise.

The following theorem quantifies to what extent the mistake bound of WTA on trees can take advantage of the tolerance to label perturbation contained in Theorem 6. Introducing shorthands for the right-hand side expressions in Theorem 6,

$$\widetilde{\Phi}^W_T(y) = \min_{y' \in \{-1,+1\}^n} 2\Big(\Phi^W_T(y') + \zeta_T\big(\delta(y,y')\big)\Big)$$

and

$$\widetilde{\Phi}_T(y) = \min_{y' \in \{-1,+1\}^n} 2\left(\Phi_T(y') + \delta(y,y')\right) ,$$

we have the following robust version of Theorem 3.

Theorem 7 *If* WTA *is run on a weighted and labeled tree (T,y), then the total number m_T of mistakes satisfies*

$$m_T \stackrel{O}{=} \widetilde{\Phi}_T(y)\left(1+\log\left(1+\frac{R^W_T(\neg E')\,\widetilde{\Phi}^W_T(y)}{\widetilde{\Phi}_T(y)}\right)\right) + \Phi_T(y) + |E'|$$

for all subsets E' of $E \setminus E^{\phi}$.

As a simple consequence, we have the following corollary.

Corollary 8 *If* WTA *is run on a weighted and polynomially connected labeled tree (T,y), then the total number m_T of mistakes satisfies*

$$m_T \stackrel{O}{=} \widetilde{\Phi}_T(y)\log n .$$

Theorem 7 combines the result of Theorem 3 with the robustness to label perturbation of WTA's tree linearization procedure. Comparing the two theorems, we see that the main advantage of the tree linearization lies in the mistake bound dependence on the logarithmic factors occurring in the formulas: Theorem 7 shows that, when $\widetilde{\Phi}_T(y) \ll \Phi_T(y)$, then the performance of WTA can be just linear in $\Phi_T(y)$. Theorem 3 shows instead that the dependence on $\Phi_T(y)$ is in general superlinear even in cases when flipping few labels of y makes the cutsize $\Phi_T(y)$ decrease in a substantial way. In many cases, the tolerance to noise allows us to achieve even better results: Corollary 8 states that, if T is polynomially connected and there exists a labeling y' with small $\delta(y,y')$ such that $\Phi_T(y')$ is much smaller than $\Phi_T(y)$, then the performance of WTA is about the same as if the algorithm were run on (T,y'). In fact, from Lemma 2 we know that when T is polynomially connected the mistake bound of WTA mainly depends on the number of ϕ-edges in (L,y), which can often be much smaller than those in (T,y). As a simple example, let T be an unweighted star graph with a labeling y and z be the difference between the number of $+1$ and the number of -1 in y. Then the mistake bound of WTA is linear in $z\log n$ irrespective of $\Phi_T(y)$ and, specifically, irrespective of the label assigned to the central node of the star, which can greatly affect the actual value of $\Phi_T(y)$.

We are now ready to extend the above results to the case when WTA operates on a general weighted graph (G,y) via a uniformly generated random spanning tree T. As before, we need some shorthand notation. Define $\Phi^*_G(y)$ as

$$\Phi^*_G(y) = \min_{y' \in \{-1,+1\}^n}\Big(\mathbb{E}\big[\Phi_T(y')\big] + \delta(y,y')\Big) ,$$

where the expectation is over the random draw of a spanning tree T of G. The following are the robust versions of Theorem 4 and Corollary 5.

Theorem 9 *If* WTA *is run on a random spanning tree T of a labeled weighted graph (G,y), then the total number m_G of mistakes satisfies*

$$\mathbb{E}\,m_G \stackrel{O}{=} \Phi^*_G(y)\left(1+\log\left(1+w^{\phi}_{\max}\mathbb{E}\left[R^W_T\right]\right)\right)+\mathbb{E}\left[\Phi_T(y)\right]\,,$$

where $w^{\phi}_{\max} = \max_{(i,j)\in E^{\phi}} w_{i,j}$.

Corollary 10 *If* WTA *is run on a random spanning tree T of a labeled weighted graph (G,y) and the ratio of the weights of each pair of edges of G is polynomial in n, then the total number m_G of mistakes satisfies*

$$\mathbb{E}\,m_G \stackrel{O}{=} \Phi^*_G(y)\log n\,.$$

The relationship between Theorem 9 and Theorem 4 is similar to the one between Theorem 7 and Theorem 3. When there exists a labeling y' such that $\delta(y,y')$ is small and $\mathbb{E}\left[\Phi_T(y')\right] \ll \mathbb{E}\left[\Phi_T(y)\right]$, then Theorem 9 allows a linear dependence on $\mathbb{E}\left[\Phi_T(y)\right]$. Finally, Corollary 10 quantifies the advantages of WTA's noise tolerance under a similar (but stricter) assumption as the one contained in Corollary 5.

7. Implementation

As explained in Section 4, WTA runs in two phases: (i) a random spanning tree is drawn; (ii) the tree is linearized and labels are sequentially predicted. As discussed in Section 1.1, Wilson's algorithm can draw a random spanning tree of "most" unweighted graphs in expected time $O(n)$. The analysis of running times on weighted graphs is significantly more complex, and outside the scope of this paper. A naive implementation of WTA's second phase runs in time $O(n\log n)$ and requires linear memory space when operating on a tree with n nodes. We now describe how to implement the second phase to run in time $O(n)$, that is, in *constant* amortized time per prediction step.

Once the given tree T is linearized into an n-node line L, we initially traverse L from left to right. Call j_0 the left-most terminal node of L. During this traversal, the resistance distance $d(j_0,i)$ is incrementally computed for each node i in L. This makes it possible to calculate $d(i,j)$ in constant time for any pair of nodes, since $d(i,j) = |d(j_0,i)-d(j_0,j)|$ for all $i,j\in L$. On top of L, a complete binary tree T' with $2^{\lceil\log_2 n\rceil}$ leaves is constructed.[4] The k-th leftmost leaf (in the usual tree representation) of T' is the k-th node in L (numbering the nodes of L from left to right). The algorithm maintains this data-structure in such a way that at time t: (i) the subsequence of leaves whose labels are revealed at time t are connected through a (bidirectional) list B, and (ii) all the ancestors in T' of the leaves of B are marked. See Figure 4.

When WTA is required to predict the label y_{i_t}, the algorithm looks for the two closest revealed leaves i' and i'' oppositely located in L with respect to i_t. The above data structure supports this operation as follows. WTA starts from i_t and goes upwards in T' until the first marked ancestor $\mathrm{anc}(i_t)$ of i_t is reached. During this upward traversal, the algorithm marks each internal node of T' on the path connecting i_t to $\mathrm{anc}(i_t)$. Then, WTA starts from $\mathrm{anc}(i_t)$ and goes downwards in order to find the leaf $i'\in B$ closest to i_t. Note how the algorithm uses node marks for finding its way down: For instance, in Figure 4 the algorithm goes left since $\mathrm{anc}(i_t)$ was reached from below through the

4. For simplicity, this description assumes n is a power of 2. If this is not the case, we could add dummy nodes to L before building T'.

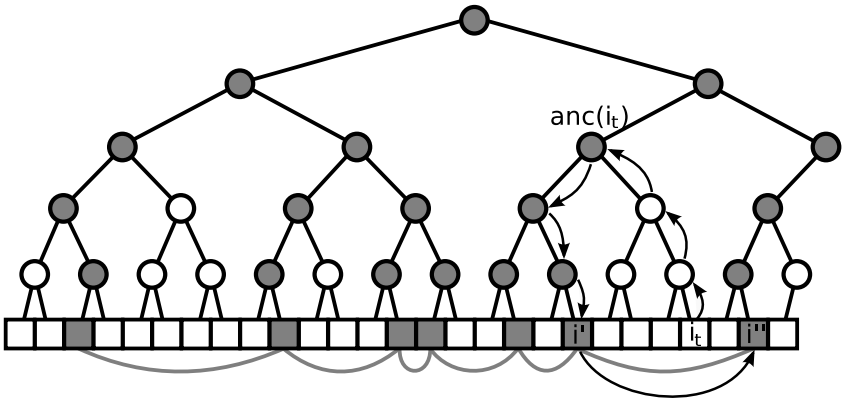

Figure 4: Constant amortized time implementation of WTA. The line L has $n = 27$ nodes (the adjacent squares at the bottom). Shaded squares are the revealed nodes, connected through a dark grey doubly-linked list B. The depicted tree T' has both unmarked (white) and marked (shaded) nodes. The arrows indicate the traversal operations performed by WTA when predicting the label of node i_t: The upward traversal stops as soon as a marked ancestor $\mathrm{anc}(i_t)$ is found, and then a downward traversal begins. Note that WTA first descends to the left, and then keeps going right all the way down. Once i' is determined, a single step within B suffices to determine i''.

right child node, and then keeps right all the way down to i'. Node i'' (if present) is then identified via the links in B. The two distances $d(i_t, i')$ and $d(i_t, i'')$ are compared, and the closest node to i_t within B is then determined. Finally, WTA updates the links of B by inserting i_t between i' and i''.

In order to quantify the amortized time per trial, the key observation is that each internal node k of T' gets visited only twice during *upward* traversals over the n trials: The first visit takes place when k gets marked for the first time, the second visit of k occurs when a subsequent upward visit also marks the other (unmarked) child of k. Once both of k's children are marked, we are guaranteed that no further upward visits to k will be performed. Since the preprocessing operations take $O(n)$, this shows that the total running time over the n trials is linear in n, as anticipated. Note, however, that the worst-case time per trial is $O(\log n)$. For instance, on the very first trial T' has to be traversed all the way up and down.

This is the way we implemented WTA on the experiments described in the next section.

8. Experiments

We now present the results of an experimental comparison on a number of real-world weighted graphs from different domains: text categorization, optical character recognition, spam detection and bioinformatics. Although our theoretical analysis is for the sequential prediction model, all experiments are carried out using a more standard train-test scenario. This makes it easy to compare WTA against popular non-sequential baselines, such as Label Propagation.

We compare our algorithm to the following other methods, intended as representatives of two different ways of coping with the graph prediction problem: global vs. local prediction.

Perceptron with Laplacian kernel: Introduced by Herbster and Pontil (2007) and here abbreviated as GPA (graph Perceptron algorithm). This algorithm sequentially predicts the nodes of a weighted graph $G = (V,E)$ after mapping V via the linear kernel based on $L_G^+ + 1\,1^\top$, where L_G is the Laplacian matrix of G. Following Herbster et al. (2009b), we run GPA on a spanning tree T of the original graph. This is because a careful computation of the Laplacian pseudoinverse of a n-node tree takes time $\Theta(n+m^2+mD)$ where m is the number of training examples plus the number of test examples (labels to predict), and D is the tree diameter—see the work of Herbster et al. (2009b) for a proof of this fact. However, in most of our experiments $m = n$, implying a running time of $\Theta(n^2)$ for GPA.

Note that GPA is a global approach, in that the graph topology affects, via the inverse Laplacian, the prediction on all nodes.

Weighted Majority Vote: Introduced here and abbreviated as WMV. Since the common underlying assumption to graph prediction algorithms is that adjacent nodes are labeled similarly, a very intuitive and fast algorithm for predicting the label of a node i is via a weighted majority vote on the available labels of the adjacent nodes. More precisely, WMV predicts using the sign of

$$\sum_{j:(i,j)\in E} y_j w_{i,j} \ ,$$

where $y_j = 0$ if node j is not available in the training set. The overall time and space requirements are both of order $\Theta(|E|)$, since we need to read (at least once) the weights of all edges. WMV is also a local approach, in the sense that prediction at each node is only affected by the labels of adjacent nodes.

Label Propagation: Introduced by Zhu et al. (2003) and here abbreviated as LABPROP. This is a batch transductive learning method based on solving a (possibly sparse) linear system of equations which requires $\Theta(|E||V|)$ time. This bad scalability prevented us from carrying out comparative experiments on larger graphs of 10^6 or more nodes. Note that WMV can be viewed as a fast approximation of LABPROP.

In our experiments, we combined WTA and GPA with spanning trees generated in different ways (note that WMV and LABPROP do not use spanning trees).

Random Spanning Tree (RST). Each spanning tree is taken with probability proportional to the product of its edge weights—see, for example, the monograph by Lyons and Peres (2009, Chapter 4). In addition, we also tested WTA combined with RST generated by ignoring the edge weights (which were then restored before running WTA). This second approach gives a prediction algorithm whose total expected running time, including the generation of the spanning tree, is $\Theta(|V|)$ on most graphs. We abbreviate this spanning tree as NWRST (non-weighted RST).

Depth-first spanning tree (DFST). This spanning tree is created via the following randomized depth-first visit: A root is selected at random, then each newly visited node is chosen with probability proportional to the weights of the edges connecting the current vertex with the adjacent nodes that have not been visited yet. This spanning tree is faster to generate than RST, and can be viewed as an approximate version of RST.

Minimum Spanning Tree (MST). The spanning tree minimizing the sum of the resistors of all edges. This is the tree whose Laplacian best approximates the Laplacian of G according to the trace norm criterion—see, for example, the paper of Herbster et al. (2009b). Note that the expected running time for the creation of a MST is $O(|E|)$, see the work by Karger et al. (1995), while the

worst-case time is $O\big(|E|\alpha(|E|,|V|)\big)$ where α is the inverse of the Ackermann function (Chazelle, 2000).

Shortest Path Spanning Tree (SPST). Herbster et al. (2009b) use the shortest path tree because it has a small diameter (at most twice the diameter of G). This allows them to better control the theoretical performance of GPA. We generated several shortest path spanning trees by choosing the root node at random, and then took the one with minimum diameter.

In order to check whether the information carried by the edge weight has predictive value for a nearest neighbor rule like WTA, we also performed a test by ignoring the edge weights during both the generation of the spanning tree and the running of WTA's nearest neighbor rule. This is essentially the algorithm analyzed by Herbster et al. (2009a), and we denote it by NWWTA (non-weighted WTA). We combined NWWTA with weighted and unweighted spanning trees. So, for instance, NWWTA+RST runs a 1-NN rule (NWWTA) that does not take edge weights into account (i.e., pretending that all weights are unitary) on a random spanning tree generated according to the actual edge weights. NWWTA+NWRST runs NWWTA on a random spanning tree that also disregards edge weights.

Finally, in order to make the classifications based on RST's more robust with respect to the variance associated with the random generation of the spanning tree, we also tested committees of RST's. For example, K*WTA+RST denotes the classifier obtained by drawing K RST's, running WTA on each one of them, and then aggregating the predictions of the K resulting classifiers via a majority vote. For our experiments we chose $K = 7, 11, 17$.

We ran our experiments on five real-world data sets.

RCV1: The first 10,000 documents (in chronological order) of Reuters Corpus Volume 1, with TF-IDF preprocessing and Euclidean normalization. This data set is available at `trec.nist.gov/data/reuters/reuters.html`.

USPS: The USPS data set with features normalized into $[0,2]$. This data set is available at `www-i6.informatik.rwth-aachen.de/~keysers/usps.html`.

KROGAN: This is a high-throughput protein-protein interaction network for budding yeast. It has been used by Krogan et al. (2006) and Pandey et al. (2007).

COMBINED: A second data set from the work of Pandey et al. (2007). It is a combination of three data sets: Gavin et al.'s (2002), Ito et al.'s (2001), and Uetz et al.'s (2000).

WEBSPAM: A large data set (110,900 nodes and 1,836,136 edges) of inter-host links created for the Web Spam Challenge 2008 (Yahoo Research and Univ. of Milan, 2007). The data set is available at `barcelona.research.yahoo.net/webspam/datasets/`. This is a weighted graph with binary labels and a pre-defined train/test split: 3,897 training nodes and 1,993 test nodes (the remaining ones being unlabeled).[5]

We created graphs from RCV1 and USPS with as many nodes as the total number of examples (x_i, y_i) in the data sets. That is, 10,000 nodes for RCV1 and 7291+2007 = 9298 for USPS. Following previous experimental settings (Zhu et al., 2003; Belkin et al., 2004), the graphs were constructed using k-NN based on the standard Euclidean distance $\|x_i - x_j\|$ between node i and node j. The weight $w_{i,j}$ was set to $w_{i,j} = \exp\big(-\|x_i - x_j\|^2 / \sigma_{i,j}^2\big)$, if j is one of the k nearest neighbors of i, and 0 otherwise. To set $\sigma_{i,j}^2$, we first computed the average square distance between i and its k nearest neighbors (call it σ_i^2), then we computed σ_j^2 in the same way, and finally set $\sigma_{i,j}^2 = \big(\sigma_i^2 + \sigma_j^2\big)/2$. We

5. We do not compare our results to those obtained in the challenge since we are only exploiting the graph (weighted) topology here, disregarding content features.

generated two graphs for each data set by running k-NN with $k = 10$ (RCV1-10 and USPS-10) and $k = 100$ (RCV1-100 and USPS-100). The labels were set using the four most frequent categories in RCV1 and all 10 categories in USPS.

In KROGAN and COMBINED we only considered the biggest connected components of both data sets, obtaining 2,169 nodes and 6,102 edges for KROGAN, and 2,871 nodes and 6,407 edges for COMBINED. In these graphs, each node belongs to one or more classes, each class representing a gene function. We selected the set of functional labels at depth one in the FunCat classification scheme of the MIPS database (Ruepp, 2004), resulting in seventeen classes per data set.

In order to associate binary classification tasks with the six non-binary data sets/graphs (RCV1-10, RCV1-100, USPS-10, USPS-100, KROGAN, COMBINED) we binarized the corresponding multiclass problems via a standard one-vs-rest scheme. We thus obtained: four binary classification tasks for RCV1-10 and RCV1-100, ten binary tasks for USPS-10 and USPS-100, seventeen binary tasks for both KROGAN and COMBINED. For a given a binary task and data set, we tried different proportions of training set and test set sizes. In particular, we used training sets of size 5%, 10%, 25% and 50%. For any given size, the training sets were randomly selected.

We report error rates and F-measures on the test set, after macro-averaging over the binary tasks. The results are contained in Tables 1–7 (Appendix B) and in Figures 5–6. Specifically, Tables 1–6 contain results for all combinations of algorithms and train/test split for the first six data sets (i.e., all but WEBSPAM).

The WEBSPAM data set is very large, and requires us a lot of computational resources in order to run experiments on this graph. Moreover, GPA has always shown inferior accuracy performance than the corresponding version of WTA (i.e., the one using the same kind of spanning tree) on all other data sets. Hence we decided not to go on any further with the refined implementation of GPA on trees we mentioned above. In Table 7 we only report test error results on the four algorithms WTA, WMV, LABPROP, and WTA with a committee of seven (nonweighted) random spanning trees.

In our experimental setup we tried to control the sources of variance in the first six data sets as follows:

1. We first generated ten random permutations of the node indices for each one of the six graphs/data sets;

2. on each permutation we generated the training/test splits;

3. we computed MST and SPST for each graph and made (for WTA, GPA, WMV, and LABPROP) one run per permutation on each of the 4+4+10+10+17+17 = 62 binary problems, averaging results over permutations and splits;

4. for each graph, we generated ten random instances for each one of RST, NWRST, DFST, and then operated as in step 2, with a further averaging over the randomness in the tree generation.

Figure 5 extracts from Tables 1–6 the error levels of the best spanning tree performers, and compared them to WMV and LABPROP. For comparison purposes, we also displayed the error levels achieved by WTA operating on a committee of seventeen random spanning trees (see below). Figure 6 (left) contains the error level on WEBSPAM reported in Table 7. Finally, Figure 6 (right) is meant to emphasize the error rate differences between RST and NWRST run with WTA.

Several interesting observations and conclusions can be drawn from our experiments.

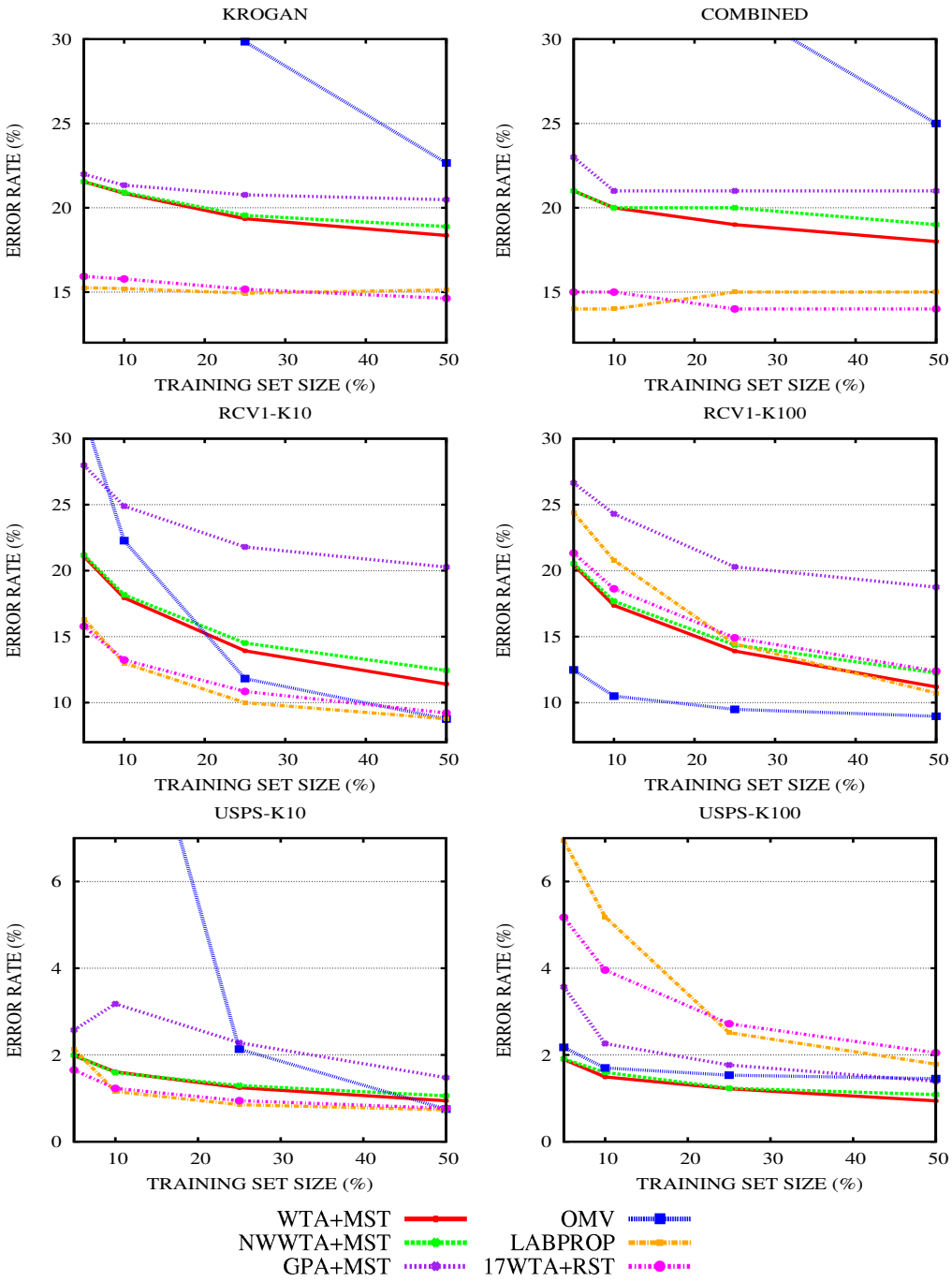

Figure 5: Macroaveraged test error rates on the first six data sets as a function of the training set size. The results are extracted from Tables 1–6 in Appendix B. Only the best performing spanning tree (i.e., MST) is shown for the algorithms that use spanning trees. These results are compared to WMV, LABPROP, and 17*WTA+RST.

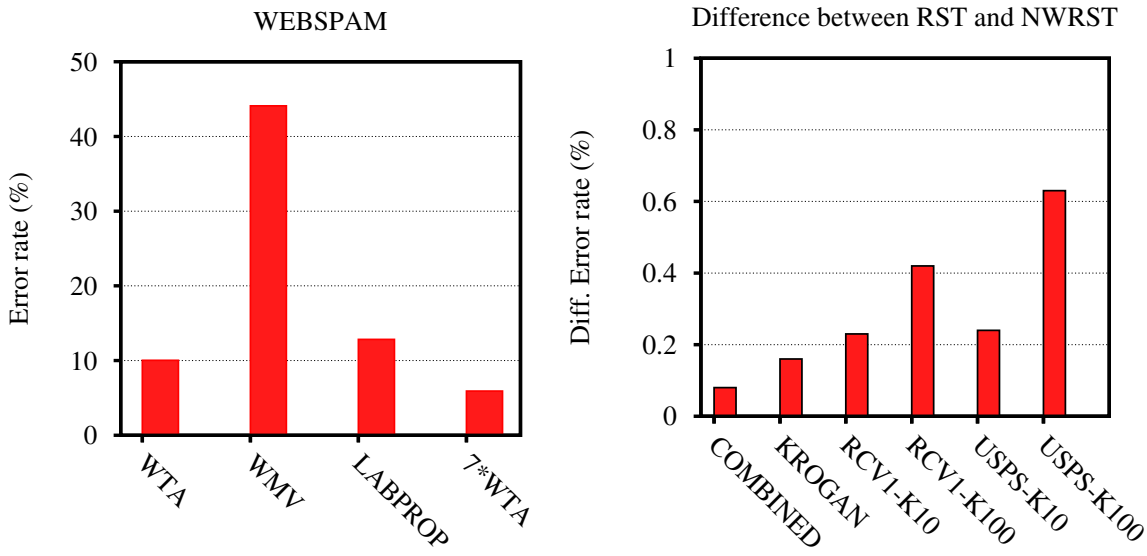

Figure 6: **Left:** Error rate levels on WEBSPAM taken from Table 7 in Appendix B. **Right:** Average error rate difference across data sets when using WTA+NWRST rather than WTA+RST.

1. WTA outperforms GPA on all data sets and with all spanning tree combinations. In particular, though we only reported aggregated results, the same relative performance pattern among the two algorithms repeats systematically over all binary classification problems. In addition, WTA runs significantly faster than GPA, requires less memory storage (linear in $|V|$, rather than quadratic), and is also fairly easy to implement.

2. By comparing NWWTA to WTA, we see that the edge weight information in the nearest neighbor rule increases accuracy, though only by a small amount.

3. WMV is a fast and accurate approximation to LABPROP when either the graph is dense (RCV1-100, and USPS-100) or the training set is comparatively large (25%–50%), although neither of the two situations often occurs in real-world applications.

4. The best performing spanning tree for both WTA and GPA is MST. This might be explained by the fact that MST tends to select light ϕ-edges of the original graph.

5. NWRST and DFST are fast approximations to RST. Though the use of NWRST and DFST does not provide theoretical performance guarantees as for RST, in our experiments they do actually perform comparably. Hence, in practice, NWRST and DFST might be viewed as fast and practical ways to generate spanning trees for WTA.

6. The prediction performance of WTA+MST is sometimes slightly inferior to LABPROP's. However, it should be stressed that LABPROP takes time $\Theta(|E||V|)$, whereas a single sweep of WTA+MST over the graph just takes time $O(|E|\alpha(|E|,|V|))$, where α is the inverse of the Ackermann function (Chazelle, 2000). Committees of spanning trees are a simple way to make WTA approach, and sometimes surpass, the performance of LABPROP. One can see

that on sparse graphs using committees gives a good performances improvement. In particular, committees of WTA can reach the same performances of LABPROP while adding just a constant factor to their (linear) time complexity.

9. Conclusions and Open Questions

We introduced and analyzed WTA, a randomized online prediction algorithm for weighted graph prediction. The algorithm uses random spanning trees and has nearly optimal performance guarantees in terms of expected prediction accuracy. The expected running time of WTA is optimal when the random spanning tree is drawn ignoring edge weights. Thanks to its linearization phase, the algorithm is also provably robust to label noise.

Our experimental evaluation shows that WTA outperforms other previously proposed online predictors. Moreover, when combined with an aggregation of random spanning trees, WTA also tends to beat standard batch predictors, such as label propagation. These features make WTA (and its combinations) suitable to large scale applications.

There are two main directions in which this work can improved. First, previous analyses (Cesa-Bianchi et al., 2009) reveal that WTA's analysis is loose, at least when the input graph is an unweighted tree with small diameter. This is the main source of the $\Omega(\ln|V|)$ slack between WTA upper bound and the general lower bound of Theorem 1. So we ask whether, at least in certain cases, this slack could be reduced. Second, in our analysis we express our upper and lower bounds in terms of the cutsize. One may object that a more natural quantity for our setting is the weighted cutsize, as this better reflects the assumption that ϕ-edges tend to be light, a natural notion of bias for weighted graphs. In more generality, we ask what are other criteria that make a notion of bias better than another one. For example, we may prefer a bias which is robust to small perturbations of the problem instance. In this sense Φ^*_G, the cutsize robust to label perturbation introduced in Section 6, is a better bias than $\mathbb{E}\,\Phi_T$. We thus ask whether there is a notion of bias, more natural and robust than $\mathbb{E}\,\Phi_T$, which captures as tightly as possible the optimal number of online mistakes on general weighted graphs. A partial answer to this question is provided by the recent work of Vitale et al. (2012). It would also be nice to tie this machinery with recent results in the active node classification setting on trees developed by Cesa-Bianchi et al. (2010).

Acknowledgments

We would like to thank the anonymous reviewers whose comments helped us to improve the presentation of this paper, and to better put it in the context of the existing literature. This work was supported in part by Google Inc. through a Google Research Award, and by the PASCAL2 Network of Excellence under EC grant 216886. This publication only reflects the authors views.

Appendix A.

This appendix contains the proofs of Lemma 2, Theorem 3, Theorem 4, Corollary 5, Theorem 6, Theorem 7, Corollary 8, Theorem 9, and Corollary 10. Notation and references are as in the main text. We start by proving Lemma 2.

Proof [Lemma 2] Let a *cluster* be any maximal sub-line of L whose edges are all ϕ-free. Then L contains exactly $\Phi_L(y)+1$ clusters, which we number consecutively, starting from one of the two

terminal nodes. Consider the k-th cluster c_k. Let v_0 be the first node of c_k whose label is predicted by WTA. After y_{v_0} is revealed, the cluster splits into two edge-disjoint sub-lines c'_k and c''_k, both having v_0 as terminal node.[6] Let v'_k and v''_k be the closest nodes to v_0 such that (i) $y_{v'_k} = y_{v''_k} \neq y_{v_0}$ and (ii) v'_k is adjacent to a terminal node of c'_k, and v''_k is adjacent to a terminal node of c''_k. The nearest neighbor prediction rule of WTA guarantees that the first mistake made on c'_k (respectively, c''_k) must occur on a node v_1 such that $d(v_0,v_1) \geq d(v_1,v'_k)$ (respectively, $d(v_0,v_1) \geq d(v_1,v''_k)$). By iterating this argument for the subsequent mistakes we see that the total number of mistakes made on cluster c_k is bounded by

$$1 + \left\lfloor \log_2 \frac{R'_k + (w'_k)^{-1}}{(w'_k)^{-1}} \right\rfloor + \left\lfloor \log_2 \frac{R''_k + (w''_k)^{-1}}{(w''_k)^{-1}} \right\rfloor$$

where R'_k is the resistance diameter of sub-line c'_k, and w'_k is the weight of the ϕ-edge between v'_k and the terminal node of c'_k closest to it (R''_k and w''_k are defined similarly). Hence, summing the above displayed expression over clusters $k = 1,\ldots,\Phi_L(y)+1$ we obtain

$$\begin{aligned}
m_L &\stackrel{O}{=} \Phi_L(y) + \sum_k \Big(\log\big(1 + R'_k w'_k\big) + \log\big(1 + R'_k w''_k\big) \Big) \\
&\stackrel{O}{=} \Phi_L(y) \left(1 + \log\left(1 + \frac{1}{\Phi_L(y)} \sum_k R'_k w'_k \right) + \log\left(1 + \frac{1}{\Phi_L(y)} \sum_k R''_k w''_k \right) \right) \\
&\stackrel{O}{=} \Phi_L(y) \left(1 + \log\left(1 + \frac{R^W_L \Phi^W_L(y)}{\Phi_L(y)} \right) \right) ,
\end{aligned}$$

where in the second step we used Jensen's inequality and in the last one the fact that $\sum_k (R'_k + R''_k) = R^W_L$ and $\max_k w'_k \stackrel{O}{=} \Phi^W_L(y)$, $\max_k w''_k \stackrel{O}{=} \Phi^W_L(y)$. This proves the lemma in the case $E' \equiv \emptyset$.

In order to conclude the proof, observe that if we take any semi-cluster c'_k (obtained, as before, by splitting cluster c_k, being $v_0 \in c_k$ the first node whose label is predicted by WTA), and pretend to split it into two sub-clusters connected by a ϕ-free edge, we could repeat the previous dichotomic argument almost verbatim on the two sub-clusters at the cost of adding an extra mistake. We now make this intuitive argument more precise. Let (i,j) be a ϕ-free edge belonging to semi-cluster c'_k, and suppose without loss of generality that i is closer to v_0 than to j. If we remove edge (i,j) then c'_k splits into two subclusters: $c'_k(v_0)$ and $c'_k(j)$, containing node v_0 and j, respectively (see Figure 7). Let $m_{c'_k}$, $m_{c'_k(v_0)}$ and $m_{c'_k(j)}$ be the number of mistakes made on c'_k, $c'_k(v_0)$ and $c'_k(j)$, respectively. We clearly have $m_{c'_k} = m_{c'_k(v_0)} + m_{c'_k(j)}$.

Let now γ'_k be the semi-cluster obtained from c'_k by contracting edge (i,j) so as to make i coincide with j (we sometimes write $i \equiv j$). Cluster γ'_k can be split into two parts which overlap only at node $i \equiv j$: $\gamma'_k(v_0)$, with terminal nodes v_0 and i (coinciding with node j), and $\gamma'_k(j)$. In a similar fashion, let $m_{\gamma'_k}$, $m_{\gamma'_k(v_0)}$, and $m_{\gamma'_k(j)}$ be the number of mistakes made on γ'_k, $\gamma'_k(v_0)$ and $\gamma'_k(j)$, respectively. We have $m_{\gamma'_k} = m_{\gamma'_k(v_0)} + m_{\gamma'_k(j)} - 1$, where the -1 takes into account that $\gamma'_k(v_0)$ and $\gamma'_k(j)$ overlap at node $i \equiv j$.

Observing now that, for each node v belonging to $c'_k(v_0)$ (and $\gamma'_k(v_0)$), the distance $d(v,v'_k)$ is smaller on γ_k than on c'_k, we can apply the above-mentioned dichotomic argument to bound the mistakes made on c'_k, obtaining $m_{\gamma'_k(v_0)} \leq m_{c'_k(v_0)}$. Since $m_{c'_k(j)} = m_{\gamma'_k(j)}$, we can finally write $m_{c'_k} = m_{c'_k(v_0)} + m_{c'_k(j)} \leq m_{\gamma'_k(v_0)} + m_{\gamma'_k(j)} = m_{\gamma'_k} + 1$. Iterating this argument for all edges in E' concludes the

6. With no loss of generality, we assume that neither of the two sub-lines is empty, so that v_0 is not a terminal node of c_k.

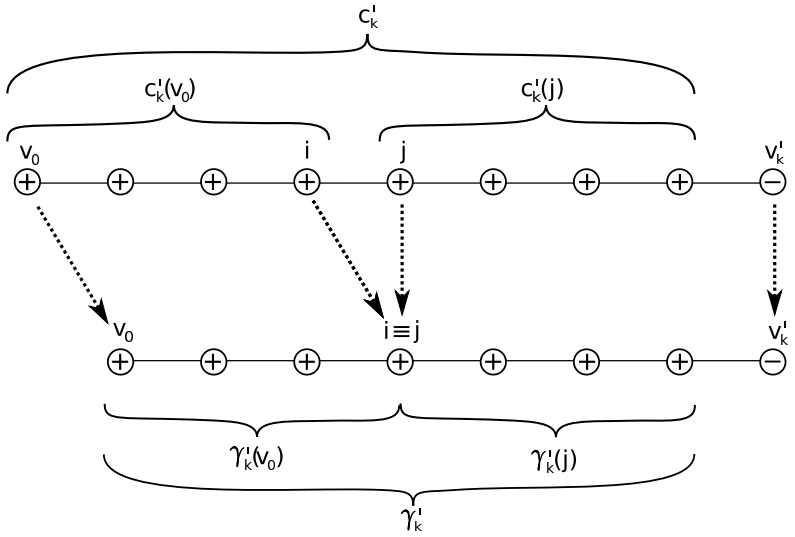

Figure 7: We illustrate the way we bound the number of mistakes on semi-cluster c'_k by dropping the resistance contribution of any (possibly very light) edge (i, j), at the cost of increasing the mistake bound on c'_k by 1. The removal of (i, j) makes c'_k split into subclusters $c'_k(v_0)$ and $c'_k(j)$. We can then drop edge (i, j) by making node i coincide with node j. The resulting semi-cluster is denoted γ'_k. This shortened version of c'_k can be viewed as split into subcluster $\gamma'_k(v_0)$ and subcluster $\gamma'_k(j)$, corresponding to $c'_k(v_0)$ and $c'_k(j)$, respectively. Now, the number of mistakes made on $c'_k(v_0)$ and $c'_k(j)$ can be bounded by those made on $\gamma'_k(v_0)$ and $\gamma'_k(j)$. Hence, we can bound the mistakes on c'_k through the ones made on γ'_k, with the addition of a single mistake, rather than two, due to the double node $i \equiv j$ of γ'_k.

proof. ■

In view of proving Theorem 3, we now prove the following two lemmas.

Lemma 11 *Given any tree T, let $E(T)$ be the edge set of T, and let $E(L')$ and $E(L)$ be the edge sets of line graphs L' and L obtained via* WTA*'s tree linearization of T. Then the following holds.*

1. *There exists a partition $\mathcal{P}_{L'}$ of $E(L')$ in pairs and a bijective mapping $\mu_{L'} : \mathcal{P}_{L'} \to E(T)$ such that the weight of both edges in each pair $S' \in \mathcal{P}_{L'}$ is equal to the weight of the edge $\mu_{L'}(S')$.*

2. *There exists a partition $\mathcal{P}_L$ of $E(L)$ in sets S such that $|S| \leq 2$, and there exists an injective mapping $\mu_L : \mathcal{P}_L \to E(T)$ such that the weight of the edges in each pair $S \in \mathcal{P}_L$ is equal to the weight of the edge $\mu_L(S)$.*

Proof We start by defining the bijective mapping $\mu_{L'} : \mathcal{P}_{L'} \to E(T)$. Since each edge (i, j) of T is traversed exactly twice in the depth-first visit that generates L',[7] once in a forward step and once in a backward step, we partition $E(L')$ in pairs S' such that $\mu_{L'}(S') = (i, j)$ if and only if S' contains the pair of distinct edges created in L' by the two traversals of (i, j). By construction, the edges in each

7. For the sake of simplicity, we are assuming here that the depth-first visit of T terminates by backtracking over all nodes on the path between the last node visited in a forward step and the root.

pair S' have weight equal to $\mu_{L'}(S')$. Moreover, this mapping is clearly bijective, since any edge of L' is created by a single traversal of an edge in T. The second mapping $\mu_L : \mathcal{P}(L) \to E(T)$ is created as follows. $\mathcal{P}_L$ is created from $\mathcal{P}_{L'}$ by removing from each $S' \in \mathcal{P}_{L'}$ the edges that are eliminated when L' is transformed into L. Note that we have $|\mathcal{P}_L| \leq |\mathcal{P}_{L'}|$ and for any $S \in \mathcal{P}_L$ there is a unique $S' \in \mathcal{P}_{L'}$ such that $S \subseteq S'$. Now, for each $S \in \mathcal{P}_L$ let $\mu_L(S) = \mu_{L'}(S')$, where S' is such that $S \subseteq S'$. Since $\mu_{L'}$ is bijective, μ_L is injective. Moreover, since the edges in S' have the same weight as the edge $\mu_{L'}(S')$, the same property holds for μ_L. ■

Lemma 12 *Let (T,y) be a labeled tree, let (L,y) be the linearization of T, and let L' be the line graph with duplicates (as described above). Then the following holds.*[8]

1. $\Phi^W_L(y) \leq \Phi^W_{L'}(y) \leq 2\Phi^W_T(y)$;
2. $\Phi_L(y) \leq \Phi_{L'}(y) \leq 2\Phi_T(y)$.

Proof From Lemma 11 (Item 1) we know that L' contains a duplicated edge for each edge of T. This immediately implies $\Phi_{L'}(y) \leq 2\Phi_T(y)$ and $\Phi^W_{L'}(y) \leq 2\Phi^W_T(y)$.

To prove the remaining inequalities, note that from the description of WTA in Section 4 (Step 3), we see that when L' is transformed into L the pair of edges (j',j) and (j,j'') of L', which are incident to a duplicate node j, gets replaced in L (together with j) by a single edge (j',j''). Now each such edge (j',j'') cannot be a ϕ-edge in L unless either (j,j') or (j,j'') is a ϕ-edge in L', and this establishes $\Phi_L(y) \leq \Phi_{L'}(y)$. Finally, if (j',j'') is a ϕ-edge in L, then its weight is not larger than the weight of the associated ϕ-edge in L' (Step 3 of WTA), and this establishes $\Phi^W_L(y) \leq \Phi^W_{L'}(y)$. ■

Recall that, given a labeled graph $G = (V,E)$ and any ϕ-free edge subset $E' \subset E \setminus E^\phi$, the quantity $R^W_G(\neg E')$ is the sum of the resistors of all ϕ-free edges in $E \setminus (E^\phi \cup E')$.

Lemma 13 *If* WTA *is run on a weighted line graph (L,y) obtained through the linearization of a given labeled tree (T,y) with edge set E, then the total number m_T of mistakes satisfies*

$$m_T \stackrel{O}{=} \Phi_L(y)\left(1+\log_2\left(1+\frac{R^W_T(\neg E')\,\Phi^W_L(y)}{\Phi_L(y)}\right)\right)+\Phi_T(y)+|E'|\,,$$

where E' is an arbitrary subset of $E \setminus E^\phi$.

Proof Lemma 11 (Item 2), exhibits an injective mapping $\mu_L : \mathcal{P} \to E$, where $\mathcal{P}$ is a partition of the edge set $E(L)$ of L, such that every $S \in \mathcal{P}$ satisfies $|S| \leq 2$. Hence, we have $|E'(L)| \leq 2|E'|$, where $E'(L)$ is the union of the pre-images of edges in E' according to μ_L—note that some edge in E' might not have a pre-image in $E(L)$. By the same argument, we also establish $|E_0(L)| \leq 2\Phi_T$, where $E_0(L)$ is the set of ϕ-free edges of L that belong to elements S of the partition $\mathcal{P}_L$ such that $\mu_L(S) \in E^\phi$.

Since the edges of L that are neither in $E_0(L)$ nor in $E'(L)$ are partitioned by $\mathcal{P}_L$ in edge sets having cardinality at most two, which in turn can be injectively mapped via μ_L to $E \setminus (E^\phi \cup E')$, we have $R^W_L\Big(\neg\big(E'(L) \cup E_0(L)\big)\Big) \leq 2R^W_T(\neg E')$. Finally, we use $|E'(L)| \leq 2|E'|$ and $|E_0(L)| \leq 2\Phi_T(y)$ (which we just established) and apply Lemma 2 with $E' \equiv E'(L) \cup E_0(L)$. This concludes the proof.

8. Item 2 in this lemma is essentially contained in the paper by Herbster et al. (2009a).

■

Proof [of Theorem 3] We use Lemma 12 to establish $\Phi_L(y) \leq 2\Phi_T(y)$ and $\Phi_L^W(y) \leq 2\Phi_T^W(y)$. We then conclude with an application of Lemma 13. ■

Lemma 14 *If* WTA *is run on a weighted line graph (L,y) obtained through the linearization of random spanning tree T of a labeled weighted graph (G,y), then the total number m_G of mistakes satisfies*

$$\mathbb{E}\, m_G \overset{O}{=} \mathbb{E}\big[\Phi_L(y)\big]\left(1+\log\left(1+w_{\max}^{\phi}\mathbb{E}\big[R_T^W\big]\right)+\mathbb{E}\big[\Phi_T(y)\big]\right) ,$$

where $w_{\max}^{\phi} = \max_{(i,j)\in E^{\phi}} w_{i,j}$.

Proof Using Lemma 13 with $E' \equiv \emptyset$ we can write

$$\begin{aligned}
\mathbb{E}\, m_G &\overset{O}{=} \mathbb{E}\left[\Phi_L(y)\left(1+\log\left(1+\frac{R_T^W\Phi_L^W(y)}{\Phi_L(y)}\right)\right)+\Phi_T\right] \\
&\overset{O}{=} \mathbb{E}\Big[\Phi_L(y)\Big(1+\log\Big(1+R_T^W w_{\max}^{\phi}\Big)\Big)+\Phi_T\Big] \\
&\overset{O}{=} \mathbb{E}\big[\Phi_L(y)\big]\Big(1+\log\Big(1+\mathbb{E}\big[R_T^W\big]w_{\max}^{\phi}\Big)\Big)+\mathbb{E}\big[\Phi_T(y)\big] ,
\end{aligned}$$

where the second equality follows from the fact that $\Phi_L^W(y) \leq \Phi_L(y) w_{\max}^{\phi}$, which in turn follows from Lemma 11, and the third one follows from Jensen's inequality applied to the concave function $(x,y) \mapsto x\Big(1+\log\Big(1+y w_{\max}^{\phi}\Big)\Big)$ for $x,y \geq 0$. ■

Proof [Theorem 4] We apply Lemma 14 and then Lemma 12 to get $\Phi_L(y) \leq 2\Phi_T(y)$. ■

Proof [Corollary 5] Let $f > \text{poly(n)}$ denote a function growing faster than any polynomial in n. Choose a polynomially connected graph G and a labeling y. For the sake of contradiction, assume that WTA makes more than $O(\mathbb{E}\big[\Phi_T(y)\big]\log n)$ mistakes on (G,y). Then Theorem 4 implies $w_{\max}^{\phi}\mathbb{E}\big[R_T^W\big] > \text{poly}(n)$. Since $\mathbb{E}\big[R_T^W\big] = \sum_{(i,j)\in E\setminus E^{\phi}} r_{i,j}^W$, we have that $w_{\max}^{\phi}\max_{(i,j)\in E\setminus E^{\phi}} r_{i,j}^W > \text{poly}(n)$. Together with the assumption of polynomial connectivity for G, this implies $w_{\max}^{\phi} r_{i,j}^W > \text{poly}(n)$ for all ϕ-free edges (i,j). By definition of effective resistance, $w_{i,j} r_{i,j}^W \leq 1$ for all $(i,j) \in E$. This gives $w_{\max}^{\phi}/w_{i,j} > \text{poly}(n)$ for all ϕ-free edges (i,j), which in turn implies

$$\frac{\sum_{(i,j)\in E^{\phi}} w_{i,j}}{\sum_{(i,j)\in E\setminus E^{\phi}} w_{i,j}} > \text{poly}(n) .$$

As this contradicts our hypothesis, the proof is concluded. ■

Proof [Theorem 6] We only prove the first part of the theorem. The proof of the second part corresponds to the special case when all weights are equal to 1.

Let $\Delta(y,y') \subseteq V$ be the set of nodes i such that $y_i \neq y'_i$. We therefore have $\delta(y,y') = |\Delta(y,y')|$. Since in a line graph each node is adjacent to at most two other nodes, the label flip of any node $j \in \Delta(y,y')$ can cause an increase of the weighted cutsize of L by at most $w_{i',j} + w_{j,i''}$, where i' and i'' are the two nodes adjacent to j in L (in the special case when j is terminal node we can set $w_{j,i''} = 0$). Hence, flipping the labels of all nodes in $\Delta(y,y')$, we have that the total cutsize increase is bounded by the sum of the weights of the $2\delta(y,y')$ heaviest edges in L, which implies

$$\Phi_L^W(y) \leq \Phi_L^W(y') + \zeta_L\big(2\delta(y,y')\big) \ .$$

By Lemma 12, $\Phi_L^W(u) \leq 2\Phi_T^W(u)$. Moreover, Lemma 11 gives an injective mapping $\mu_L : \mathcal{P}_L \to E$ (E is the edge set of T) such that the elements of $\mathcal{P}$ have cardinality at most two, and the weight of each edge $\mu_L(S)$ is the same as the weights of the edges in S. Hence, the total weight of the $2\delta(y,y')$ heaviest edges in L is at most twice the total weight of the $\delta(y,y')$ heaviest edges in T. Therefore $\zeta_L\big(2\delta(y,y')\big) \leq 2\zeta_T\big(\delta(y,y')\big)$. Hence, we have obtained

$$\Phi_L^W(y) \leq 2\Phi_T^W(y') + 2\zeta_T\big(\delta(y,y')\big) \ ,$$

concluding the proof. ■

Proof [Theorem 7] We use Theorem 6 to bound $\Phi_L(y)$ and $\Phi_L^W(y)$ in the mistake bound of Lemma 13. ■

Proof [Corollary 8] Recall that the resistance between two nodes i and j of any tree is simply the sum of the inverse weights over all edges on the path connecting the two nodes. Since T is polynomially connected, we know that the ratio of any pair of edge weights is polynomial in n. This implies that $R_L^W \Phi_L^W(y)$ is polynomial in n, too. We apply Theorem 6 to bound $\Phi_L(y)$ in the mistake bound of Lemma 2 with $E' = \emptyset$. This concludes the proof. ■

Lemma 15 *If* WTA *is run on a line graph L obtained by linearizing a random spanning tree T of a labeled and weighted graph (G,y), then we have*

$$\mathbb{E}\big[\Phi_L(y)\big] \stackrel{O}{=} \Phi_G^*(y) \ .$$

Proof Recall that Theorem 6 holds for any spanning tree T of G. Thus it suffices to apply part 2 of Theorem 6 and use $\mathbb{E}\big[\min X\big] \leq \min \mathbb{E}[X]$. ■

Proof [Theorem 9] We apply Lemma 15 to bound $\mathbb{E}\big[\Phi_L(y)\big]$ in Lemma 14. ■

Proof [Corollary 10] We can use Lemma 2 with the setting $E' \equiv \emptyset$, and bound $\mathbb{E}\big[\Phi_L(y)\big]$ via Lemma 15. To conclude, observe that since the ratio of the weights of any pair of edges in G is polynomial in n, then $R_L^W \Phi_L^W(y)$ is polynomial in n, too. ■

Appendix B.

This appendix summarizes all our experimental results. For each combination of data set, algorithm, and train/test split, we provide macro-averaged error rates and F-measures on the test set. The algorithms are WTA, NWWTA, and GPA (all combined with various spanning trees), WMV, LABPROP, and WTA run with committees of random spanning trees. WEBSPAM was too large a data set to perform as thorough an investigation. Hence we only report test error results on the four algorithms WTA, WMV, LABPROP, and WTA with a committee of 7 (nonweighted) random spanning trees.

Train/test split	5%		10%		25%		50%	
Predictors	Error	F	Error	F	Error	F	Error	F
WTA+RST	25.54	0.81	22.67	0.84	19.06	0.86	16.57	0.88
WTA+NWRST	25.81	0.81	22.70	0.83	19.24	0.86	17.00	0.87
WTA+MST	21.09	0.84	17.94	0.87	13.93	0.90	11.40	0.91
WTA+SPST	25.47	0.81	22.65	0.83	19.31	0.86	17.24	0.87
WTA+DFST	26.02	0.81	22.34	0.84	17.73	0.87	14.89	0.89
NWWTA+RST	25.28	0.81	22.45	0.84	19.12	0.86	17.16	0.87
NWWTA+NWRST	25.97	0.81	23.14	0.83	19.54	0.86	17.84	0.87
NWWTA+MST	21.18	0.84	18.17	0.87	14.51	0.89	12.44	0.91
NWWTA+SPST	25.49	0.81	22.81	0.83	19.64	0.86	17.55	0.87
NWWTA+DFST	26.08	0.81	22.82	0.83	17.93	0.87	15.64	0.88
GPA+RST	32.75	0.75	29.85	0.78	27.67	0.80	24.44	0.82
GPA+NWRST	34.27	0.74	30.36	0.78	28.90	0.79	25.99	0.81
GPA+MST	27.98	0.79	24.89	0.82	21.80	0.84	20.27	0.85
GPA+SPST	27.18	0.79	25.13	0.82	22.20	0.84	20.27	0.85
GPA+DFST	47.11	0.61	45.65	0.64	43.08	0.66	38.20	0.71
7*WTA+RST	17.40	0.87	14.85	0.90	12.15	0.91	10.39	0.92
7*WTA+NWRST	17.81	0.87	15.15	0.89	12.51	0.91	10.92	0.92
11*WTA+RST	16.40	0.88	13.86	0.90	11.38	0.92	9.71	0.93
11*WTA+NWRST	16.78	0.88	14.22	0.90	11.73	0.92	10.20	0.93
17*WTA+RST	15.78	0.89	13.23	0.91	10.85	0.92	9.22	0.94
17*WTA+NWRST	16.07	0.89	13.55	0.90	11.18	0.92	9.65	0.93
WMV	31.82	0.76	22.27	0.84	11.82	0.91	8.76	0.93
LABPROP	16.33	0.89	13.00	0.91	10.00	0.93	8.77	0.94

Table 1: RCV1-10. Average error rate and F-measure on 4 classes.

References

N. Alon, C. Avin, M. Koucký, G. Kozma, Z. Lotker, and M.R. Tuttle. Many random walks are faster than one. In *Proc. of the 20th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 119–128. Springer, 2008.

M. Belkin, I. Matveeva, and P. Niyogi. Regularization and semi-supervised learning on large graphs. In *Proc. of the 17th Annual Conference on Learning Theory*, pages 624–638. Springer, 2004.

Y. Bengio, O. Delalleau, and N. Le Roux. Label propagation and quadratic criterion. In *Semi-Supervised Learning*, pages 193–216. MIT Press, 2006.

Train/test split	5%		10%		25%		50%	
Predictors	Error	F	Error	F	Error	F	Error	F
WTA+RST	32.03	0.77	29.36	0.79	26.09	0.81	23.25	0.83
WTA+NWRST	32.05	0.77	29.89	0.78	26.65	0.80	23.82	0.83
WTA+MST	20.45	0.85	17.36	0.87	13.91	0.90	11.19	0.92
WTA+SPST	29.26	0.79	27.06	0.80	24.96	0.82	23.17	0.83
WTA+DFST	32.03	0.77	28.89	0.79	24.18	0.82	20.57	0.85
NWWTA+RST	31.72	0.77	29.46	0.78	26.20	0.81	24.04	0.82
NWWTA+NWRST	32.52	0.76	29.95	0.78	26.88	0.80	24.84	0.82
NWWTA+MST	20.54	0.85	17.68	0.87	14.37	0.89	12.25	0.91
NWWTA+SPST	29.28	0.79	27.13	0.80	25.16	0.82	23.72	0.83
NWWTA+DFST	32.05	0.77	28.81	0.79	24.14	0.82	21.28	0.84
GPA+RST	36.47	0.73	35.33	0.74	33.81	0.75	32.32	0.76
GPA+NWRST	38.26	0.72	35.91	0.73	35.20	0.74	32.73	0.76
GPA+MST	26.65	0.81	24.30	0.82	20.29	0.85	18.75	0.86
GPA+SPST	32.43	0.74	28.00	0.78	26.61	0.79	25.77	0.80
GPA+DFST	48.35	0.61	47.85	0.61	44.78	0.65	41.12	0.68
7*WTA+RST	23.30	0.84	20.55	0.86	16.87	0.88	14.34	0.90
7*WTA+NWRST	23.64	0.84	20.77	0.86	17.27	0.88	14.81	0.90
11*WTA+RST	22.06	0.85	19.39	0.87	15.63	0.89	13.20	0.91
11*WTA+NWRST	22.29	0.85	19.54	0.87	16.09	0.89	13.61	0.91
17*WTA+RST	21.33	0.86	18.62	0.88	14.91	0.90	12.39	0.92
17*WTA+NWRST	21.49	0.86	18.86	0.87	15.29	0.89	12.78	0.91
WMV	12.48	0.91	10.50	0.93	9.49	0.93	8.96	0.94
LABPROP	24.39	0.85	20.78	0.87	14.45	0.91	10.73	0.93

Table 2: RCV1-100. Average error rate and F-measure on 4 classes.

A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proc. of the 18th International Conference on Machine Learning*, pages 19–26. Morgan Kaufmann, 2001.

A. Blum, J. Lafferty, M. Rwebangira, and R. Reddy. Semi-supervised learning using randomized mincuts. In *Proc. of the 21st International Conference on Machine Learning*, pages 97–104, 2004.

A. Broder. Generating random spanning trees. In *Proc. of the 30th Annual Symposium on Foundations of Computer Science*, pages 442–447. IEEE Press, 1989.

N. Cesa-Bianchi, C. Gentile, and F. Vitale. Fast and optimal prediction of a labeled tree. In *Proceedings of the 22nd Annual Conference on Learning Theory*. Omnipress, 2009.

N. Cesa-Bianchi, C. Gentile, F. Vitale, and G. Zappella. Active learning on trees and graphs. In *Proceedings of the 23rd Annual Conference on Learning Theory*. Omnipress, 2010.

H. Chang and D.Y. Yeung. Graph Laplacian kernels for object classification from a single example. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2011–2016. IEEE Press, 2006.

Train/test split	5%		10%		25%		50%	
Predictors	Error	F	Error	F	Error	F	Error	F
WTA+RST	5.32	0.97	4.28	0.98	3.08	0.98	2.36	0.99
WTA+NWRST	5.65	0.97	4.51	0.97	3.29	0.98	2.56	0.98
WTA+MST	1.98	0.99	1.61	0.99	1.24	0.99	0.94	0.99
WTA+SPST	6.25	0.97	4.72	0.97	3.37	0.98	2.60	0.99
WTA+DFST	6.43	0.96	4.60	0.97	2.92	0.98	2.04	0.99
NWWTA+RST	5.31	0.97	4.25	0.98	3.19	0.98	2.70	0.99
NWWTA+NWRST	5.95	0.97	4.65	0.97	3.45	0.98	2.92	0.98
NWWTA+MST	1.99	0.99	1.59	0.99	1.29	0.99	1.06	0.99
NWWTA+SPST	6.30	0.96	4.83	0.97	3.50	0.98	2.84	0.98
NWWTA+DFST	6.49	0.96	4.59	0.97	3.09	0.98	2.35	0.99
GPA+RST	12.64	0.93	8.53	0.95	6.65	0.96	5.65	0.97
GPA+NWRST	12.53	0.93	9.05	0.95	6.90	0.96	5.19	0.97
GPA+MST	2.58	0.99	3.18	0.98	2.28	0.99	1.48	0.99
GPA+SPST	7.64	0.96	6.26	0.96	4.13	0.98	3.55	0.98
GPA+DFST	42.77	0.70	39.39	0.73	32.38	0.79	20.53	0.87
7*WTA+RST	2.09	0.99	1.56	0.99	1.14	0.99	0.90	0.99
7*WTA+NWRST	2.35	0.99	1.75	0.99	1.26	0.99	1.02	0.99
11*WTA+RST	1.84	0.99	1.35	0.99	1.01	0.99	0.82	1.00
11*WTA+NWRST	2.05	0.99	1.53	0.99	1.14	0.99	0.91	0.99
17*WTA+RST	1.65	0.99	1.23	0.99	0.95	0.99	0.77	1.00
17*WTA+NWRST	1.87	0.99	1.39	0.99	1.06	0.99	0.85	1.00
WMV	24.84	0.85	12.28	0.93	2.13	0.99	0.75	1.00
LABPROP	2.14	0.99	1.16	0.99	0.85	0.99	0.73	1.00

Table 3: USPS-10. Average error rate and F-measure on 10 classes.

B. Chazelle. A minimum spanning tree algorithm with inverse-Ackermann type complexity. In *Journal of the ACM*, 47(6):1028–1047, 2000.

J. Fakcharoenphol and B. Kijsirikul. Low congestion online routing and an improved mistake bound for online prediction of graph labeling. *CoRR*, abs/0809.2075, 2008.

A.-C. Gavin et al. Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature*, 415(6868):141–147, 2002.

A. Goldberg and X. Zhu. Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In *HLT-NAACL 2006 Workshop on Textgraphs: Graph-based algorithms for Natural Language Processing*, 2004.

M. Herbster. Exploiting cluster-structure to predict the labeling of a graph. In *Proc. of the 19th International Conference on Algorithmic Learning Theory*, pages 54–69. Springer, 2008.

M. Herbster and G. Lever. Predicting the labelling of a graph via minimum p-seminorm interpolation. In *Proc. of the 22nd Annual Conference on Learning Theory*. Omnipress, 2009.

M. Herbster and M. Pontil. Prediction on a graph with the Perceptron. In *Advances in Neural Information Processing Systems 21*, pages 577–584. MIT Press, 2007.

Train/test split	5%		10%		25%		50%	
Predictors	Error	F	Error	F	Error	F	Error	F
WTA+RST	9.62	0.95	8.29	0.95	6.55	0.96	5.36	0.97
WTA+NWRST	10.32	0.94	9.00	0.95	7.17	0.96	5.83	0.97
WTA+MST	1.90	0.99	1.49	0.99	1.22	0.99	0.94	0.99
WTA+SPST	8.68	0.95	7.27	0.96	5.78	0.97	4.88	0.97
WTA+DFST	10.36	0.94	8.13	0.96	5.62	0.97	4.21	0.98
NWWTA+RST	9.71	0.95	8.38	0.95	6.78	0.96	5.89	0.97
NWWTA+NWRST	10.39	0.94	9.08	0.95	7.46	0.96	6.45	0.96
NWWTA+MST	1.91	0.99	1.60	0.99	1.23	0.99	1.09	0.99
NWWTA+SPST	8.76	0.95	7.46	0.96	5.94	0.97	5.28	0.97
NWWTA+DFST	10.46	0.94	8.30	0.95	6.00	0.97	4.65	0.97
GPA+RST	14.81	0.91	13.38	0.92	11.94	0.93	9.81	0.94
GPA+NWRST	17.34	0.90	13.68	0.92	11.39	0.94	11.46	0.94
GPA+MST	3.57	0.98	2.26	0.99	1.77	0.99	1.39	0.99
GPA+SPST	8.42	0.95	7.94	0.95	7.20	0.96	5.71	0.97
GPA+DFST	46.09	0.67	42.59	0.71	37.66	0.75	28.45	0.82
7*WTA+RST	5.28	0.97	4.24	0.98	3.05	0.98	2.37	0.99
7*WTA+NWRST	5.82	0.97	4.73	0.97	3.48	0.98	2.69	0.98
11*WTA+RST	5.07	0.97	3.96	0.98	2.76	0.99	2.11	0.99
11*WTA+NWRST	5.55	0.97	4.38	0.98	3.14	0.98	2.40	0.99
17*WTA+RST	5.17	0.97	3.96	0.98	2.72	0.99	2.05	0.99
17*WTA+NWRST	7.60	0.96	6.38	0.97	4.68	0.97	3.32	0.98
WMV	2.17	0.99	1.70	0.99	1.53	0.99	1.45	0.99
LABPROP	6.94	0.96	5.19	0.97	2.51	0.99	1.79	0.99

Table 4: USPS-100. Average error rate and F-measure on 10 classes.

M. Herbster, M. Pontil, and L. Wainer. Online learning over graphs. In *Proc. of the 22nd International Conference on Machine Learning*, pages 305–312. ACM Press, 2005.

M. Herbster, G. Lever, and M. Pontil. Online prediction on large diameter graphs. In *Advances in Neural Information Processing Systems 22*, pages 649–656. MIT Press, 2009a.

M. Herbster, M. Pontil, and S. Rojas-Galeano. Fast prediction on a tree. In *Advances in Neural Information Processing Systems 22*, pages 657–664. MIT Press, 2009b.

T. Ito, T. Chiba, R. Ozawa, M. Yoshida, M. Hattori, and Y. Sakaki. A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proceedings of the National Academy of Sciences of the United States of America*, 98(8):4569–4574, 2001.

D. Karger, P. Klein, and R. Tarjan. A randomized linear-time algorithm to find minimum spanning trees. In *Journal of the ACM*, 42: 321–328. ACM, 1995b.

N.J. Krogan et al. Global landscape of protein complexes in the yeast Saccharomyces cerevisiae. *Nature*, 440:637–643, 2006.

R. Lyons and Y. Peres. Probability on trees and networks. Manuscript, 2009.

Train/test split	5%		10%		25%		50%	
Predictors	Error	F	Error	F	Error	F	Error	F
WTA+RST	21.73	0.86	21.37	0.86	19.89	0.87	19.09	0.88
WTA+NWRST	21.86	0.86	21.50	0.86	20.03	0.87	19.33	0.88
WTA+MST	21.55	0.86	20.86	0.87	19.35	0.88	18.36	0.88
WTA+SPST	21.86	0.86	21.58	0.86	20.38	0.87	19.40	0.88
WTA+DFST	21.78	0.86	21.22	0.86	19.88	0.87	18.60	0.88
NWWTA+RST	21.83	0.86	21.43	0.86	20.08	0.87	19.64	0.88
NWWTA+NWRST	21.98	0.86	21.55	0.86	20.26	0.87	19.75	0.87
NWWTA+MST	21.55	0.86	20.91	0.87	19.55	0.88	18.89	0.88
NWWTA+SPST	21.86	0.86	21.57	0.86	20.50	0.87	19.81	0.87
NWWTA+DFST	21.79	0.86	21.33	0.86	20.00	0.87	19.09	0.88
GPA+RST	22.70	0.85	22.75	0.85	22.14	0.86	21.28	0.86
GPA+NWRST	23.83	0.84	23.28	0.85	22.48	0.85	21.53	0.86
GPA+MST	21.99	0.86	21.34	0.86	20.77	0.86	20.48	0.87
GPA+SPST	22.33	0.84	21.34	0.86	20.71	0.86	20.74	0.86
GPA+DFST	39.77	0.72	31.93	0.78	25.70	0.83	24.09	0.84
7*WTA+RST	16.83	0.90	16.63	0.90	15.78	0.90	15.29	0.90
7*WTA+NWRST	16.85	0.90	16.60	0.90	15.89	0.90	15.41	0.90
11*WTA+RST	16.28	0.90	16.11	0.90	15.36	0.91	14.92	0.91
11*WTA+NWRST	16.28	0.90	16.08	0.90	15.55	0.90	14.99	0.91
17*WTA+RST	15.93	0.90	15.78	0.90	15.17	0.91	14.63	0.91
17*WTA+NWRST	15.98	0.90	15.69	0.91	15.23	0.91	14.68	0.91
WMV	42.98	0.70	38.88	0.73	29.85	0.80	22.66	0.85
LABPROP	15.26	0.91	15.21	0.91	14.94	0.91	15.13	0.91

Table 5: KROGAN. Average error rate and F-measure on 17 classes.

G. Pandey, M. Steinbach, R. Gupta, T. Garg, and V. Kumar. Association analysis-based transformations for protein interaction networks: a function prediction case study. In *Proc. of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 540–549. ACM Press, 2007.

Yahoo! Research (Barcelona) and Laboratory of Web Algorithmics (Univ. of Milan). Web Spam Collection. URL `barcelona.research.yahoo.net/webspam/datasets/`.

A. Ruepp. The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic Acids Research*, 32(18):5539–5545, 2004.

D.A. Spielman and N. Srivastava. Graph sparsification by effective resistances. In *Proc. of the 40th annual ACM symposium on Theory of computing*, pages 563–568. ACM Press, 2008.

H. Shin K. Tsuda and B. Schölkopf. Protein functional class prediction with a combined graph. *Expert Systems with Applications*, 36:3284–3292, 2009.

P. Uetz et al. A comprehensive analysis of protein-protein interactions in Saccharomyces cerevisiae. *Nature*, 6770(403):623–627, 2000.

Train/test split	5%		10%		25%		50%	
Predictors	Error	F	Error	F	Error	F	Error	F
WTA+RST	21.68	0.86	21.05	0.87	20.08	0.87	18.99	0.88
WTA+NWRST	21.47	0.87	21.29	0.86	20.18	0.87	19.17	0.88
WTA+MST	21.57	0.86	20.63	0.87	19.61	0.88	18.37	0.88
WTA+SPST	21.39	0.87	21.34	0.86	20.52	0.87	19.57	0.88
WTA+DFST	21.88	0.86	21.09	0.87	19.82	0.87	18.83	0.88
NWWTA+RST	21.50	0.87	21.15	0.87	20.43	0.87	19.95	0.87
NWWTA+NWRST	21.61	0.86	21.26	0.87	20.52	0.87	20.09	0.87
NWWTA+MST	21.53	0.86	20.95	0.87	20.35	0.87	19.81	0.88
NWWTA+SPST	21.37	0.87	21.06	0.87	20.55	0.87	20.06	0.87
NWWTA+DFST	21.88	0.86	21.05	0.87	20.50	0.87	19.74	0.88
GPA+RST	23.56	0.85	22.27	0.86	21.86	0.86	21.68	0.86
GPA+NWRST	23.91	0.85	23.11	0.85	22.47	0.86	21.30	0.86
GPA+MST	23.32	0.85	21.60	0.86	21.77	0.86	21.67	0.86
GPA+SPST	22.55	0.85	21.89	0.85	21.64	0.85	21.70	0.85
GPA+DFST	41.69	0.71	30.82	0.79	26.75	0.82	23.56	0.84
7*WTA+RST	16.39	0.90	16.09	0.90	15.77	0.91	15.29	0.91
7*WTA+NWRST	16.35	0.90	16.10	0.90	15.77	0.90	15.47	0.91
11*WTA+RST	15.89	0.91	15.61	0.91	15.32	0.91	14.84	0.91
11*WTA+NWRST	15.82	0.91	15.57	0.91	15.34	0.91	14.98	0.91
17*WTA+RST	15.54	0.91	15.31	0.91	14.97	0.91	14.55	0.91
17*WTA+NWRST	15.45	0.91	15.29	0.91	15.05	0.91	14.66	0.91
WMV	44.74	0.68	40.75	0.72	32.97	0.78	25.28	0.84
LABPROP	14.93	0.91	14.98	0.91	15.23	0.91	15.31	0.90

Table 6: COMBINED. Average error rate and F-measure on 17 classes.

Predictors	Error	F
WTA+NWRST	10.03	0.95
3*WTA+NWRST	6.44	0.97
7*WTA+NWRST	5.91	0.97
WMV	44.1	0.71
LABPROP	12.84	0.93

Table 7: WEBSPAM. Test set error rate and F-measure. WTA operates only on NWRST.

F. Vitale, N. Cesa-Bianchi, C. Gentile, and G. Zappella. See the tree through the lines: the Shazoo algorithm. In *Proc. of the 25th Annual Conference on Neural Information Processing Systems*, pages 1584-1592. Curran Associates, 2012.

D.B. Wilson. Generating random spanning trees more quickly than the cover time. In *Proc. of the 28th ACM Symposium on the Theory of Computing*, pages 296–303. ACM Press, 1996.

X. Zhu, Z. Ghahramani, and 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.