Active Learning on Trees and Graphs

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

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

Fabio Vitale   Giovanni Zappella
Università degli Studi di Milano
Italy
fabio.vitale@unimi.it
giovanni.zappella@studenti.unimi.it

Abstract

We investigate the problem of active learning on a given tree whose nodes are assigned binary labels in an adversarial way. Inspired by recent results by Guillory and Bilmes, we characterize (up to constant factors) the optimal placement of queries so to minimize the mistakes made on the non-queried nodes. Our query selection algorithm is extremely efficient, and the optimal number of mistakes on the non-queried nodes is achieved by a simple and efficient mincut classifier. Through a simple modification of the query selection algorithm we also show optimality (up to constant factors) with respect to the trade-off between number of queries and number of mistakes on non-queried nodes. By using spanning trees, our algorithms can be efficiently applied to general graphs, although the problem of finding optimal and efficient active learning algorithms for general graphs remains open. Towards this end, we provide a lower bound on the number of mistakes made on arbitrary graphs by any active learning algorithm using a number of queries which is up to a constant fraction of the graph size.

1   Introduction

The abundance of networked data in various application domains (web, social networks, bioinformatics, etc.) motivates the development of scalable and accurate graph-based prediction algorithms. An important topic in this area is the graph binary classification problem: Given a graph with unknown binary labels on its nodes, the learner receives the labels on a subset of the nodes (the training set) and must predict the labels on the remaining vertices. This is typically done by relying on some notion of label regularity depending on the graph topology, such as that nearby nodes are likely to be labeled similarly. Standard approaches to this problem predict with the assignment of labels minimizing the induced cutsize (e.g., [4, 5]), or by binarizing the assignment that minimizes certain real-valued extensions of the cutsize function (e.g., [10, 2, 3] and references therein).

In the active learning version of this problem the learner is allowed to choose the subset of training nodes. Similarly to standard feature-based learning, one expects active methods to provide a significant boost of predictive ability compared to a noninformed (e.g., random) draw of the training set. The following simple example provides some intuition of why this could happen when the labels are chosen by an adversary, which is the setting considered in this paper. Consider a "binary star system" of two star-shaped graphs whose centers are connected by a bridge, where one star is a constant fraction bigger than the other. The adversary draws two random binary labels and assigns the first label to all nodes of the first star graph, and the second label to all nodes of the second star graph. Assume that the training set size is two. If we choose the centers of the two stars and predict with a mincut strategy,[1] we are guaranteed to make zero mistakes on all unseen vertices. On the other hand, if we query two nodes at random, then with constant probability both of them will belong to the bigger star, and all the unseen labels of the smaller star will be mistaken. This simple example shows that the gap between the performance of passive and active learning on graphs can be made arbitrarily big.

In general, one would like to devise a strategy for placing a certain budget of queries on the vertices of a given graph. This should be done so as to minimize the number of mistakes made on the non-queried nodes by some reasonable classifier like mincut. This question has been investigated from a theoretical viewpoint

---

[1] A mincut strategy considers all labelings consistent with the labels observed so far, and chooses among them one that minimizes the resulting cutsize over the whole graph.

by Guillory and Bilmes [6], and by Afshani et al. [1]. Our work is related to an elegant result from [6] which bounds the number of mistakes made by the mincut classifier on the worst-case assignment of labels in terms of $\Phi/\Psi(L)$. Here Φ is the cutsize induced by the unknown labeling, and $\Psi(L)$ is a function of the query (or training) set L, which depends on the structural properties of the (unlabeled) graph. For instance, in the above example of the binary system, the value of $\Psi(L)$ when the query set L includes just the two centers is 1. This implies that for the binary system graph, Guillory and Bilmes' bound on the mincut strategy is Φ mistakes in the worst case (note that in the above example $\Phi \leq 1$). Since $\Psi(L)$ can be efficiently computed on any given graph and query set L, the learner's task might be reduced to finding a query set L that maximizes $\Psi(L)$ given a certain query budget (size of L). Unfortunately, no feasible general algorithm for solving this maximization problem is known, and so one must resort to heuristic methods —see [6].

In this work we investigate the active learning problem on graphs in the important special case of trees. We exhibit a simple iterative algorithm which, combined with a mincut classifier, is optimal (up to constant factors) on any given labeled tree. This holds even if the algorithm is not given information on the actual cutsize Φ. Our method is extremely efficient, requiring $\mathcal{O}(n \ln Q)$ time for placing Q queries in an n-node tree, and space linear in n. As a byproduct of our analysis, we show that Ψ can be efficiently maximized over trees to within constant factors. Hence the bound $\min_L \Phi/\Psi(L)$ can be achieved efficiently.

Another interesting question is what kind of trade-off between queries and mistakes can be achieved if the learner is not constrained by a given query budget. We show that a simple modification of our selection algorithm is able to trade-off queries and mistakes in an optimal way up to constant factors.

Finally, we prove a general lower bound for predicting the labels of any given graph (not necessarily a tree) when the query set is up to a constant fraction of the number of vertices. Our lower bound establishes that the number of mistakes must then be at least a constant fraction of the cutsize weighted by the effective resistances. This lower bound apparently yields a contradiction to the results of Afshani et al. [1], who constructs the query set adaptively. This apparent contradiction is also obtained via a simple counterexample that we detail in Section 5.

2   Preliminaries and basic notation

A labeled tree $(T, \boldsymbol{y})$ is a tree $T = (V, E)$ whose nodes $V = \{1, \ldots, n\}$ are assigned binary labels $\boldsymbol{y} = (y_1, \ldots, y_n) \in \{-1, +1\}^n$. We measure the label regularity of $(T, \boldsymbol{y})$ by the *cutsize* $\Phi_T(\boldsymbol{y})$ induced by $\boldsymbol{y}$ on T, i.e., $\Phi_T(\boldsymbol{y}) = \big|\{(i, j) \in E : y_i \neq y_j\}\big|$. We consider the following *active* learning protocol: given a tree T with unknown labeling $\boldsymbol{y}$, the learner obtains all labels in a *query set* $L \subseteq V$, and is then required to predict the labels of the remaining nodes $V \setminus L$. Active learning algorithms work in two-phases: a *selection* phase, where a query set of given size is constructed, and a *prediction* phase, where the algorithm receives the labels of the query set and predicts the labels of the remaining nodes. Note that the only labels ever observed by the algorithm are those in the query set. In particular, no labels are revealed during the prediction phase.

We measure the ability of the algorithm by the number of prediction mistakes made on $V \setminus L$, where it is reasonable to expect this number to depend on both the uknown cutsize $\Phi_T(\boldsymbol{y})$ and the number $|L|$ of requested labels. A slightly different prediction measure is considered in Section 4.3.

Given a tree T and a query set $L \subseteq V$, a node $i \in V \setminus L$ is a **fork node generated by** L if and only if there exist three distinct nodes $i_1, i_2, i_3 \in L$ that are connected to i through edge disjoint paths. Let $\text{FORK}(L)$ be the set of all fork nodes generated by L. Then L^+ is the query set obtained by adding to L all the generated fork nodes, i.e., $L^+ \triangleq L \cup \text{FORK}(L)$. We say that $L \subseteq V$ is **0-forked** iff $L^+ \equiv L$. Note that L^+ is 0-forked. That is, $\text{FORK}(L^+) \equiv \emptyset$ for all $L \subseteq V$.

Given a node subset $S \subseteq V$, we use $T \setminus S$ to denote the forest obtained by removing from the tree T all nodes in S and all edges incident to them. Moreover, given a second tree T', we denote by $T \setminus T'$ the forest $T \setminus V'$, where V' is the set of nodes of T'. Given a query set $L \subseteq V$, a **hinge-tree** is any connected component of $T \setminus L^+$. We call **connection node** of a hinge-tree a node of L adjacent to any node of the hinge tree. We distinguish between 1-hinge and 2-hinge trees. A **1-hinge-tree** has one connection node only, whereas a **2-hinge-tree** has two (note that a hinge tree cannot have more than two connection nodes because L^+ is zero-forked, see Figure 1).

3   The active learning algorithm

We now describe the two phases of our active learning algorithm. For the sake of exposition, we call SEL the selection phase and PRED the prediction phase. SEL returns a 0-forked query set $L_{\text{SEL}}^+ \subseteq V$ of desired size. PRED takes in input the query set L_{SEL}^+ and the set of labels y_i for all $i \in L_{\text{SEL}}^+$. Then PRED returns a prediction for the labels of all remaining nodes $V \setminus L_{\text{SEL}}^+$.

In order to see the way SEL operates, we formally introduce the function Ψ^*. This is the reciprocal of the Ψ function introduced in [6] and mentioned in Section 1.
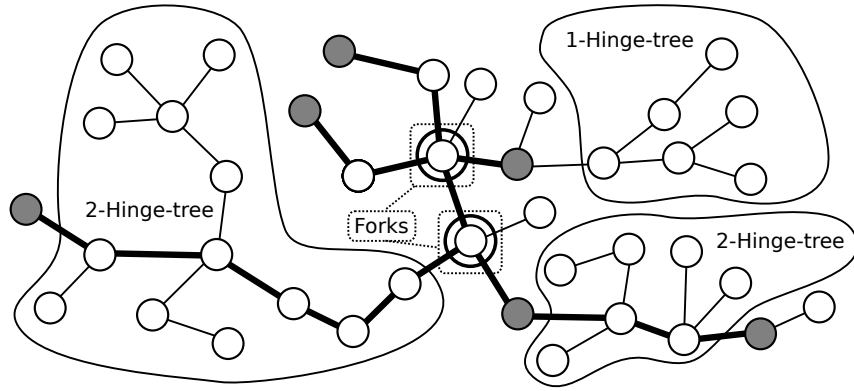
Figure 1: A tree $T = (V, E)$ whose nodes are shaded (the query set L) or white (the set $V \setminus L$). The shaded nodes are also the connection nodes of the depicted hinge trees (not all hinge trees are contoured). The fork nodes generated by L are denoted by double circles. The thick black edges connect the nodes in L.

Definition 1 *Given a tree $T = (V, E)$ and a set of nodes $L \subseteq V$,*

$$\Psi^*(L) \triangleq \max_{\emptyset \neq V' \subseteq V \setminus L} \frac{|V'|}{\left|\{(i,j) \in E : i \in V', j \in V \setminus V'\}\right|} .$$

In words, $\Psi^*(L)$ measures the largest set of nodes not in L that share the least number of edges with nodes in L. From the adversary's viewpoint, $\Psi^*(L)$ can be described as the largest return in mistakes per unit of cutsize invested. We now move on to the description of the algorithms SEL and PRED.

The **selection algoritm** SEL greedily computes a query set that minimizes Ψ^* to within constant factors. To this end, SEL exploits Lemma 10 (a) (see Section 4.2) stating that, for any fixed query set L, the subset $V' \subseteq V$ maximizing $\frac{|V'|}{\left|\{(i,j) \in E : i \in V', j \in V \setminus V'\}\right|}$ is always included in a connected component of $T \setminus L$. Thus SEL places its queries in order to end up with a query set L_{SEL}^+ such that the largest component of $T \setminus L_{\text{SEL}}^+$ is as small as possible.

SEL operates as follows. Let $L_t \subseteq L$ be the set including the first t nodes chosen by SEL, $T_{\max}^t$ be the largest connected component of $T \setminus L_{t-1}$, and $\sigma(T', i)$ be the size (number of nodes) of the largest component of the forest $T' \setminus \{i\}$, where T' is any tree. At each step $t = 1, 2, \ldots$, SEL simply picks the node $i_t \in T_{\max}^t$ that minimizes $\sigma(T_{\max}^t, i)$ over i and sets $L_t = L_{t-1} \cup \{i_t\}$. During this iterative construction, SEL also maintains a set containing all fork nodes generated in each step by adding nodes i_t to the sets L_{t-1}.[2] After the desired number of queries is reached (also counting the queries that would be caused by the stored fork nodes), SEL has terminated the construction of the query set L_{SEL}. The final query set L_{SEL}^+, obtained by adding all stored fork nodes to L_{SEL}, is then returned.

The **Prediction Algorithm** PRED receives in input the labeled nodes of the 0-forked query set L_{SEL}^+ and computes a mincut assignment. Since each component of $T \setminus L_{\text{SEL}}^+$ is either a 1-hinge-tree or a 2-hinge-tree, PRED is simple to describe and is also very efficient. The algorithm predicts all the nodes of hinge-tree $\mathcal{T}$ using the same label $\widehat{y}_{\mathcal{T}}$. This label is chosen according to the following two cases:

1. If $\mathcal{T}$ is a 1-hinge-tree, then $\widehat{y}_{\mathcal{T}}$ is set to the label of its unique connection node;

2. If $\mathcal{T}$ is a 2-hinge-tree and the labels of its two connection nodes are equal, then $\widehat{y}_{\mathcal{T}}$ is set to the label of its connection nodes, otherwise $\widehat{y}_{\mathcal{T}}$ is set as the label of the closer connection node (ties are broken arbitrarily).

In Section 6 we show that SEL requires overall $\mathcal{O}(|V| \log Q)$ time and $\mathcal{O}(|V|)$ memory space for selecting Q query nodes. Also, we will see that the total running time taken by PRED for predicting all nodes in $V \setminus L$ is linear in $|V|$.

4   Analysis

For a given tree T, we denote by $m_A(L, \boldsymbol{y})$ the number of prediction mistakes that algorithm A makes on the labeled tree $(T, \boldsymbol{y})$ when given the query set L. Introduce the function

$$m_A(L, K) = \max_{\boldsymbol{y} : \Phi_T(\boldsymbol{y}) \leq K} m_A(L, \boldsymbol{y})$$

---

[2]In Section 6 we will see that during each step $L_{t-1} \to L_t$ at most a single new fork node may be generated.
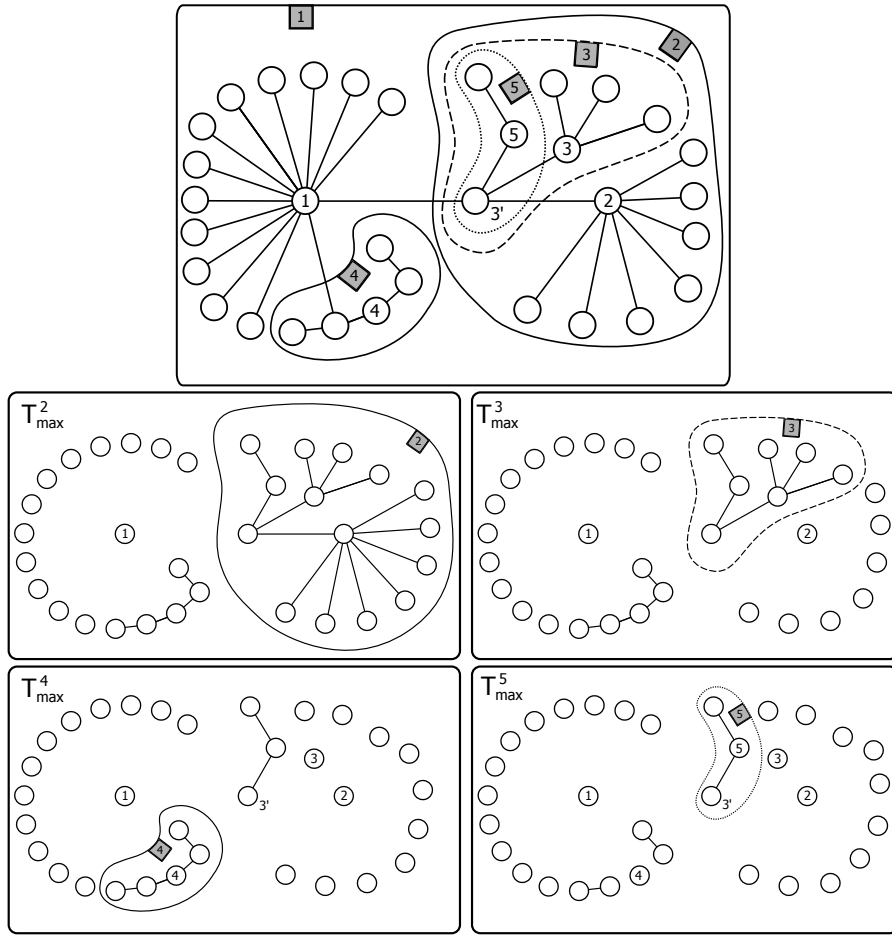
Figure 2: The SEL algorithm at work. The upper pane shows the initial tree $T = T_{\max}^1$ (in the box tagged with "1"), and the subsequent subtrees $T_{\max}^2$, $T_{\max}^3$, $T_{\max}^4$, and $T_{\max}^5$. The left pane also shows the nodes selected by SEL in chronological order. The four lower panes show the connected components of $T \setminus L_t$ resulting from this selection. Observe that at the end of round 3, SEL detects the generation of fork node $3'$. This node gets stored, and is added to L_{SEL} at the end of the selection process.

denoting the number of prediction mistakes made by A with query set L on all labeled trees with cutsize bounded by K. We will also find it useful to deal with the "lower bound" function $\text{LB}(L, K)$. This is the maximum expected number of mistakes that any prediction algorithm A can be forced to make on the labeled tree $(T, \boldsymbol{y})$ when the query set is L and the cutsize is not larger than K.

We show that the number of mistakes made by PRED on any labeled tree when using the query set L_{SEL}^+ satisfies

$$m_{\text{PRED}}(L_{\text{SEL}}^+, K) \leq 10 \, \text{LB}(L, K)$$

for all query sets $L \subseteq V$ of size up to $\frac{1}{8}|L_{\text{SEL}}^+|$. Though neither SEL nor PRED do know the actual cutsize of the labeled tree $(T, \boldsymbol{y})$, the combined use of these procedures is competitive against any algorithm that knows the cutsize budget K beforehand.

While this result implies the optimality (up to constant factors) of our algorithm, it does not relate the mistake bound to the cutsize, which is a clearly interpretable measure of the label regularity. In order to address this issue, we show that our algorithm also satisfies the bound

$$m_{\text{PRED}}(L_{\text{SEL}}^+, \boldsymbol{y}) \leq 4 \, \Psi^*(L) \, \Phi_T(\boldsymbol{y})$$

for all query sets $L \subseteq V$ of size up to $\frac{1}{8}|L_{\text{SEL}}^+|$. The proof of these results needs a number of preliminary lemmas.

Lemma 2 *For any tree $T = (V, E)$ it holds that $\min_{v \in V} \sigma(T, v) \leq \frac{1}{2}|V|$.*

Proof: Let $i \in \operatorname{argmin}_{v \in V} \sigma(T, v)$. For the sake of contradiction, assume there exists a component $T_i = (V_i, E_i)$ of $T \setminus \{i\}$ such that $|V_i| > |V|/2$. Let s be the sum of the sizes all other components. Since

$|V_i| + s = |V| - 1$, we know that $s \leq |V|/2 - 1$. Now let j be the node adjacent to i which belongs to V_i and $T_j = (V_j, E_j)$ be the largest component of $T \setminus \{j\}$. There are only two cases to consider: either $V_j \subset V_i$ or $V_j \cap V_i \equiv \emptyset$. In the first case, $|V_j| < |V_i|$. In the second case, $V_j \subseteq \{i\} \cup (T \setminus V_i)$, which implies $|V_j| \leq 1 + s \leq |V|/2 < |V_i|$. In both cases, $i \notin \operatorname{argmin}_{v \in V} \sigma(T, v)$, which provides the desired contradiction. ∎

Lemma 3 *For all subsets $L \subset V$ of the nodes of a tree $T = (V, E)$ we have $|L^+| \leq 2|L|$.*

Proof: Pick an arbitrary node of T and perform a depth-first visit of all nodes in T. This visit induces an ordering $\mathcal{T}_1, \mathcal{T}_2, \ldots$ of the connected components in $T \setminus L$ based on the order of the nodes visited first in each component. Now let $\mathcal{T}_1', \mathcal{T}_2', \ldots$ be such that each $\mathcal{T}_i'$ is a component of $\mathcal{T}_i$ extended to include all nodes of L adjacent to nodes in $\mathcal{T}_i$. Then the ordering implies that, for $i \geq 2$, $\mathcal{T}_i'$ shares exactly one node (which must be a leaf) with all previously visited trees. Since in any tree the number of nodes of degree larger than two must be strictly smaller than the number of leaves, we have $|\text{FORK}(\mathcal{T}_i')| < |\Lambda_i|$ where, with slight abuse of notation, we denote by $\text{FORK}(\mathcal{T}_i')$ the set of all fork nodes in subtree $\mathcal{T}_i'$. Also, we let Λ_i be the set of leaves of $\mathcal{T}_i'$. This implies that, for $i = 1, 2, \ldots$, each fork node in $\text{FORK}(\mathcal{T}_i')$ can be injectively associated with one of the $|\Lambda_i| - 1$ leaves of $\mathcal{T}_i'$ that are not shared with any of the previously visited trees. Since $|\text{FORK}(L)|$ is equal to the sum of $|\text{FORK}(\mathcal{T}_i)|$ over all indices i, this implies that $|\text{FORK}(L)| \leq |L|$. ∎

Lemma 4 *Let $L_{t-1} \subseteq L_{\text{SEL}}$ be the set of the first $t - 1$ nodes chosen by SEL. Given any tree $T = (V, E)$, the largest subtree of $T \setminus L_{t-1}$ contains no more than $\frac{2}{t}|V|$ nodes.*

Proof: Recall that i_s denotes the s-th node selected by SEL during the incremental construction of the query set L_{SEL}, and that $T_{\max}^s$ is the largest component of $T \setminus L_{s-1}$. The first t steps of the recursive splitting procedure performed by SEL can be associated with a splitting tree T' defined in the following way. The internal nodes of T' are $T_{\max}^s$, for $s \geq 1$. The children of $T_{\max}^s$ are the connected components of $T_{\max}^s \setminus \{i_s\}$, i.e., the subtrees of $T_{\max}^s$ created by the selection of i_s. Hence, each leaf of T' is bijectively associated with a tree in $T \setminus L_t$.

Let T_{nol}' be the tree obtained from T' by deleting all leaves. Each node of T_{nol}' is one of the t subtrees split by SEL during the construction of L_t. As $T_{\max}^t$ is split by i_t, it is a leaf in T_{nol}'. We now add a second child to each internal node s of T_{nol}' having a single child. This second child of s is obtained by merging all the subtrees belonging to leaves of T' that are also children of s. Let T'' be the resulting tree.

We now compare the cardinality of $T_{\max}^t$ to that of the subtrees associated with the leaves of T''. Let Λ be the set of all leaves of T'' and $\Lambda_{\text{add}} = T'' \setminus T_{\text{nol}}' \subset \Lambda$ be the set of all leaves added to T_{nol}' to obtain T''. First of all, note that $|T_{\max}^t|$ is not larger than the number of nodes in any leaf of T_{nol}'. This is because the selection rule of SEL ensures that $T_{\max}^t$ cannot be larger than any subtree associated with a leaf in T_{nol}', since it contains no node selected before time t. In what follows, we write $|s|$ to denote the size of the forest or subtree associated with a node s of T''. We now prove the following claim:

Claim. For all $\ell \in \Lambda$, $|T_{\max}^t| \leq |\ell|$, and for all $\ell \in \Lambda_{\text{add}}$, $|T_{\max}^t| - 1 \leq |\ell|$.

Proof of Claim. The first part just follows from the observation that any $\ell \in \Lambda$ was split by SEL before time t. In order to prove the second part, pick a leaf $\ell \in \Lambda_{\text{add}}$. Let ℓ' be its unique sibling in T'' and let p be the parent of ℓ and ℓ', also in T''. Lemma 2 applied to the subtree p implies $|\ell'| \leq \frac{1}{2}|p|$. Moreover, since $|\ell| + |\ell'| = |p| - 1$, we obtain $|\ell| + 1 \geq \frac{1}{2}|p| \geq |\ell'| \geq |T_{\max}^t|$, the last inequality using the first part of the claim. This implies $|T_{\max}^t| - 1 \leq |\ell|$, and the claim is proven.

Let now $N(\Lambda)$ be the number of nodes in subtrees and forests associated with the leaves of T''. With each internal node of T'' we can associate a node of L_{SEL} which does not belong to any leaf in Λ. Moreover, the number $|T'' \setminus \Lambda|$ of internal nodes in T'' is bigger than the number $|\Lambda_{\text{add}}|$ of internal nodes of T_{nol}' to which a child has been added. Since these subtrees and forests are all distinct, we obtain $N(\Lambda) + |T'' \setminus \Lambda| < N(\Lambda) + |\Lambda_{\text{add}}| \leq |V|$. Hence, using the above claim we can write $N(\Lambda) \geq (|\Lambda| - |\Lambda_{\text{add}}|)|T_{\max}^t| + |\Lambda_{\text{add}}|(|T_{\max}^t| - 1)$, which implies $|T_{\max}^t| \leq (N(\Lambda) + |\Lambda_{\text{add}}|)/|\Lambda| \leq |V|/|\Lambda|$. Since each internal node of T'' has at least two children, we have that $|\Lambda| \geq |T''|/2 \geq |T_{\text{nol}}'|/2 = t/2$. Hence, we can conclude that $|T_{\max}^t| \leq 2|V|/t$. ∎

4.1 Lower bounds

We now state and prove a lower bound on the number of mistakes that any prediction algorithm (even knowing the cutsize budget K) makes on any given tree, when the query set L is 0-forked. The bound depends on the following quantity: Given a tree $T(V, E)$, a node subset $L \subseteq V$ and an integer K, the **component function** $\Upsilon(L, K)$ is the sum of the sizes of the K largest components of $T \setminus L$, or $|V \setminus L|$ if $T \setminus L$ has less than K components.

Theorem 5 *For all trees $T = (V, E)$, for all 0-forked subsets $L^+ \subseteq V$, and for all cutsize budgets $K = 0, 1, \ldots, |V| - 1$, we have that $\mathrm{LB}(L^+, K) \geq \frac{1}{2}\Upsilon(L^+, K)$.*

Proof: We describe an adversarial strategy causing any algorithm to make at least $\Upsilon(L^+, K)/2$ mistakes even when the cutsize budget K is known beforehand. Since L^+ is 0-forked, each component of $T \setminus L^+$ is a hinge-tree. Let $F_{\max}$ be the set of the K largest hinge-trees of $T \setminus L^+$, and $E(\mathcal{T})$ be the set of all edges in E incident to at least one node of a hinge-tree $\mathcal{T}$. The adversary creates at most one ϕ-edge[3] in each edge set $E(\mathcal{T}_1)$ for all 1-hinge-trees $\mathcal{T}_1 \in F_{\max}$, exactly one ϕ-edge in each edge set $E(\mathcal{T}_2)$ for all 2-hinge-trees $\mathcal{T}_2 \in F_{\max}$, and no ϕ-edges in the edge set $E(\mathcal{T})$ of any remaining hinge-tree $\mathcal{T} \notin F_{\max}$. This is done as follows. By performing a depth-first visit of T, the adversary can always assign disagreeing labels to the two connection nodes of each 2-hinge-tree in $F_{\max}$, and agreeing labels to the two connection nodes of each 2-hinge-tree not in $F_{\max}$. Then, for each hinge-tree $\mathcal{T} \in F_{\max}$, the adversary assigns a unique random label to all nodes of $\mathcal{T}$, forcing $|\mathcal{T}|/2$ mistakes in expectation. The labels of the remaining hinge-trees not in $F_{\max}$ are chosen in agreement with their connection nodes. ∎

Remark 1 *Note that Theorem 5 holds for all query sets, not only those that are 0-forked, since any adversarial strategy for a query set L^+ can force at least the same mistakes on the subset $L \subseteq L^+$. Note also that it is not difficult to modify the adversarial strategy described in the proof of Theorem 5 in order to deal with algorithms that are allowed to adaptively choose the query nodes in L depending on the labels of the previously selected nodes. The adversary simply assigns the same label to each node in the query set and then forces, with the same method described in the proof, $\frac{1}{2}\Upsilon\left(L^+, \frac{K}{2}\right)$ mistakes in expectation on the $\frac{K}{2}$ largest hinge-trees. Thus there are at most two ϕ-edges in each edge set $E(\mathcal{T})$ for all hinge-trees $\mathcal{T}$, yielding at most K ϕ-edges in total. The resulting (slightly weaker) bound is $\mathrm{LB}(L^+, K) \geq \frac{1}{2}\Upsilon\left(L^+, \frac{K}{2}\right)$. Theorem 8 and Corollary 9 can also be easily rewritten in order to extend the results in this direction.*

4.2 Upper bounds

We now bound the total number of mistakes that PRED makes on any labeled tree when the queries are decided by SEL. We use Lemma 2 and 3, together with the two lemmas below, to prove that $m_{\mathrm{PRED}}(L^+_{\mathrm{SEL}}, K) \leq 10\,\mathrm{LB}(L, K)$ for all cutsize budgets K and for all node subset $L \subseteq V$ such that $|L| \leq \frac{1}{8}|L^+_{\mathrm{SEL}}|$.

Lemma 6 *For all labeled trees $(T, \boldsymbol{y})$ and for all 0-forked query sets $L^+ \subseteq V$, the number of mistakes made by PRED satisfies $m_{\mathrm{PRED}}(L^+, \boldsymbol{y}) \leq \Upsilon\left(L^+, \Phi_T(\boldsymbol{y})\right)$.*

Proof: As in the proof of Theorem 5, we first observe that each component of $T \setminus L^+$ is a hinge-tree. Let $E(\mathcal{T})$ be the set of all edges in E incident to nodes of a hinge-tree $\mathcal{T}$, and F_ϕ be the set of hinge-trees such that, for all $\mathcal{T} \in F_\phi$, at least one edge of $E(\mathcal{T})$ is a ϕ-edge. Since $E(\mathcal{T}) \cap E(\mathcal{T}') \equiv \emptyset$ for all $\mathcal{T}, \mathcal{T}' \in T \setminus L^+$, we have that $|F_\phi| \leq \Phi_T(\boldsymbol{y})$. Moreover, since for any $\mathcal{T} \notin F_\phi$ there are no ϕ-edges in $E(\mathcal{T})$, the nodes of $\mathcal{T}$ must be labeled as its connections nodes. This, together with the prediction rule of PRED, implies that PRED makes no mistakes over any of the hinge-trees $\mathcal{T} \notin F_\phi$. Hence, the number of mistakes made by PRED is bounded by the sum of the sizes of all hinge-trees $\mathcal{T} \in F_\phi$, which (by definition of Υ) is bounded by $\Upsilon\left(L^+, \Phi_T(\boldsymbol{y})\right)$. ∎

The next lemma, whose proof is a bit involved, provides the relevant properties of the component function $\Upsilon(\cdot, \cdot)$. Figure 3 helps visualizing the main ingredients of the proof.

Lemma 7 *Given a tree $T = (V, E)$, for all node subsets $L \subseteq V$ such that $|L| \leq \frac{1}{2}|L_{\mathrm{SEL}}|$ and for all integers k, we have: (a) $\Upsilon(L_{\mathrm{SEL}}, k) \leq 5\Upsilon(L, k)$; (b) $\Upsilon(L_{\mathrm{SEL}}, 1) \leq \Upsilon(L, 1)$.*

Proof: We prove part (a) by constructing, via SEL, three bijective mappings $\mu_1, \mu_2, \mu_3 : \mathcal{P}_{\mathrm{SEL}} \to \mathcal{P}_L$, where $\mathcal{P}_{\mathrm{SEL}}$ is a suitable partition of $T \setminus L_{\mathrm{SEL}}$, $\mathcal{P}_L$ a subset of 2^V such that any $S \in \mathcal{P}_L$ is all contained in a single connected component of $T \setminus L$, and the union of the domains of the three mappings covers the whole set $T \setminus L_{\mathrm{SEL}}$. The mappings μ_1, μ_2 and μ_3 are shown to satisfy, for all forests[4] $F \in \mathcal{P}_{\mathrm{SEL}}$,

$$|F| \leq |\mu_1(F)|, \qquad |F| \leq 2|\mu_2(F)|, \qquad |F| \leq 2|\mu_3(F)| \ .$$

Since each $S \in \mathcal{P}_L$ is all contained in a connected component of $T \setminus L$, this we will enable us to conclude that, for each tree $T' \in T \setminus L$, the forest of all trees $T \setminus L_{\mathrm{SEL}}$ mapped (via any of these mappings) to any node subset of T' has at most five times the number of nodes of T'. This would prove the statement in (a).

---

[3]A ϕ-edge (i, j) is one where $y_i \neq y_j$.

[4]In this proof, $|\mu(A)|$ denotes the number of nodes in the set (of nodes) $\mu(A)$. Also, with a slight abuse of notation, for all forests $F \in \mathcal{P}_{\mathrm{SEL}}$, we denote by $|F|$ the sum of the number of nodes in all trees of F. Finally, whenever $F \in \mathcal{P}_{\mathrm{SEL}}$ contains a single tree, we refer to F as it were a tree, rather than a (singleton) forest containing only one tree.
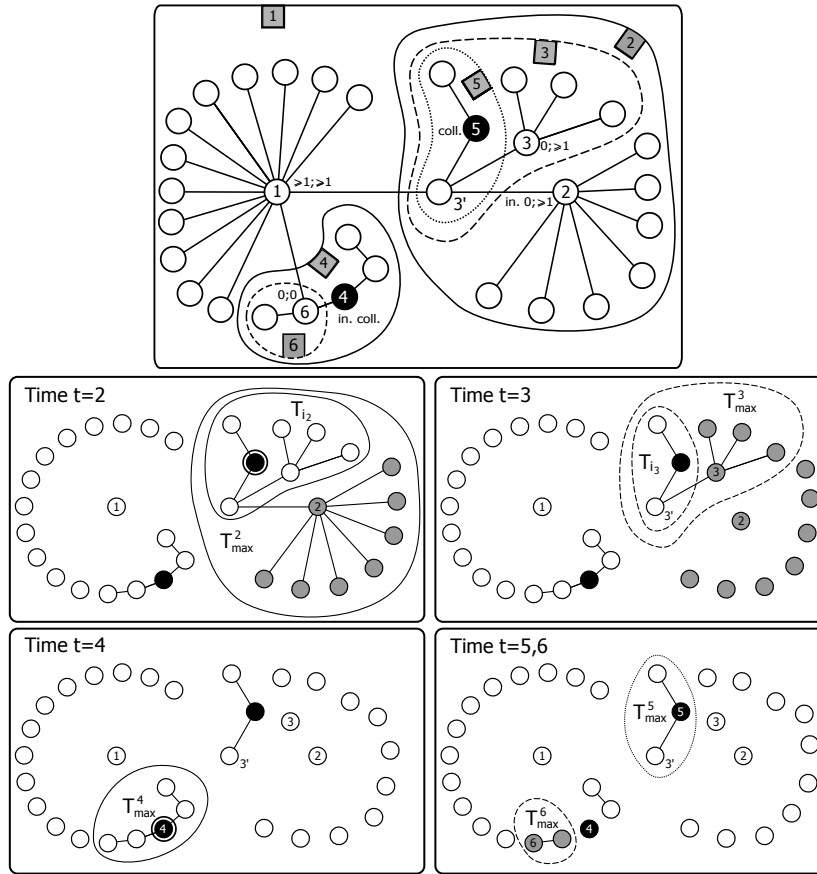
Figure 3: The upper pane illustrates the different kinds of nodes chosen by SEL. Numbers in the square tags indicate the first six subtrees $T_{\max}^t$, and their associated nodes i_t, selected by SEL. Node i_1 is a $[\geq 1; \geq 1]$-node, i_2 is an initial $[0; \geq 1]$-node, i_3 is a (noninitial) $[0; \geq 1]$-node, i_4 is an initial collision node, i_5 is a (noninitial) collision node, and i_6 is a $[0; 0]$-node. As in Figure 2, we denote by $3'$ the fork node generated by the inclusion of i_3 into L_{SEL}. Note that node i_6 may be chosen arbitrarily among the four nodes in $T_{\max}^4 \setminus i_4$. The two black nodes are the set of nodes we are competing against, i.e., the nodes in the query set L. Forest $T \setminus L$ is made up of one large subtree and two small subtrees. In the lower panes we illustrate some steps of the proof of Lemma 7, with reference to the upper pane. Time $t = 2$: Trees $T_{\max}^2$ and T_{i_2} are shown. As explained in the proof, $|T_{i_2}| \leq |T_{\max}^2 \setminus T_{i_2}|$. The circled black node is captured by i_2. The nodes of tree $T_{\max}^2 \setminus T_{i_2}$ are shaded, and can be used for mapping any ζ-component through μ_2. Time $t = 3$: Trees $T_{\max}^3$ and T_{i_3} are shown. Again, one can easily verify that $|T_{i_3}| \leq |T_{\max}^3 \setminus T_{i_3}|$. As before, the nodes of $T_{\max}^3 \setminus T_{i_3}$ are shaded, and can be used for mapping any ζ-component via μ_2. The reader can see that, according to the injectivity of μ_2, these grey nodes are well separated from the ones in $T_{\max}^2 \setminus T_{i_2}$. Time $t = 4$: $T_{\max}^4$ and the initial collision node i_4 are depicted. The latter is enclosed in a circled black node since it captures itself. Time $t = 5, 6$: We depicted trees $T_{\max}^5$ and $T_{\max}^6$, together with nodes i_5 and i_6. Node i_5 is a collision node, which is not initial since it was already captured by the $[0; \geq 1]$-node i_2. Node i_6 is a $[0; 0]$ node, so that the whole tree $T_{\max}^6$ is completely included in a component (the largest, in this case) of $T \setminus L$. Tree $T_{\max}^6$ can be used for mapping via μ_3 any ζ-component. The resulting forest $T \setminus L_6$ includes several single-node trees and one two-node tree. If i_6 is the last node selected by L_{SEL}, then each component of $T \setminus L_6$ can be exploited by mapping μ_1, since in this specific case none of these components contains nodes of L, i.e., there are no ζ-components left.

The construction of these mappings requires some auxiliary definitions. We call ζ-component each connected component of $T \setminus L_{\text{SEL}}$ containing at least one node of L. Let i_t be the t-th node selected by SEL during the incremental construction of the query set L_{SEL}. We distinguish between four kinds of nodes chosen by SEL—see Figure 3 for an example.

Node i_t is:

1. A *collision node* if it belongs to $L_{\text{SEL}} \cap L$;

2. a $[0; 0]$-*node* if, at time t, the tree $T_{\max}^t$ does not contain any node of L;

3. a $[0; \geq 1]$-*node* if, at time t, the tree $T_{\max}^t$ contains $k \geq 1$ nodes $j_1, \dots, j_k \in L$ all belonging to the same connected component of $T_{\max}^t \setminus \{i_t\}$;

4. a $[\geq 1; \geq 1]$-*node* if $i_t \notin L$ and, at time t, the tree $T_{\max}^t$ contains $k \geq 2$ nodes $j_1, \dots, j_k \in L$, which do not belong to the same connected component of $T_{\max}^t \setminus \{i_t\}$.

We now turn to building the three mappings.

μ_1 simply maps each tree $T' \in T \setminus L_{\text{SEL}}$ that is *not* a ζ-component to the node set of T' itself. This immediately implies $|F| \leq |\mu_1(F)|$ for all forests F (which are actually single trees) in the domain of μ_1. Mappings μ_2 and μ_3 deal with the ζ-components of $T \setminus L_{\text{SEL}}$. Let Z be the set of all such ζ-components, and denote by $V_{0;0}$, $V_{0;1}$, and $V_{1;1}$ the set of all $[0; 0]$-nodes, $[0; \geq 1]$-nodes, and $[\geq 1; \geq 1]$-nodes, respectively. Observe that $|V_{1;1}| < |L|$. Combined with the assumption $|L_{\text{SEL}}| \geq 2|L|$, this implies that $|V_{0;0}| + |V_{0;1}|$ plus the total number of collision nodes must be larger than $|L|$; as a consequence, $|V_{0;0}| + |V_{0;1}| > |Z|$. Each node $i_t \in V_{0;1}$ chosen by SEL splits the tree $T_{\max}^t$ into one component T_{i_t} containing at least one node of L and one or more components all contained in a single tree T'_{i_t} of $T \setminus L$. Now mapping μ_2 can be constructed incrementally in the following way. For each $[0; \geq 1]$-node selected by SEL at time t, μ_2 sequentially maps any ζ-component generated to the set of nodes in $T_{\max}^t \setminus T_{i_t}$, the latter being just a subset of a component of $T \setminus L$. A future time step $t' > t$ might feature the selection of a new $[0; \geq 1]$-node within T_{i_t}, but mapping μ_2 would cover a different subset of such component of $T \setminus L$. Now, applying Lemma 2 to tree $T_{\max}^t$, we can see that $|T_{\max}^t \setminus T_{i_t}| \geq |T_{\max}^t|/2$. Since the selection rule of SEL guarantees that the number of nodes in $T_{\max}^t$ is larger than the number of nodes of any ζ-component, we have $|F| \leq 2|\mu_2(F)|$, for any ζ-component F considered in the construction of μ_2.

Mapping μ_3 maps all the remaining ζ-components that are not mapped through μ_2. Let $\sim$ be an equivalence relation over $V_{0;0}$ defined as follows: $i \sim j$ iff i is connected to j by a path containing only $[0; 0]$-nodes and nodes in $V \setminus (L_{\text{SEL}} \cup L)$. Let $i_{t_1}, i_{t_2}, \dots, i_{t_k}$ be the sequence of nodes of any given equivalence class $[C]_\sim$, sorted according to SEL's chronological selection. Lemma 4 applied to tree $T_{\max}^{t_1}$ shows that $|T_{\max}^{t_k}| \leq 2|T_{\max}^{t_1}|/k$. Moreover, the selection rule of SEL guarantees that the number of nodes of $T_{\max}^{t_k}$ cannot be smaller than the number of nodes of any ζ-component. Hence, for each equivalence class $[C]_\sim$ containing k nodes of type $[0; 0]$, we map through μ_3 a set F_ζ of k arbitrarily chosen ζ-components to $T_{\max}^{t_1}$. Since the size of each ζ-component is $\leq |T_{\max}^{t_k}|$, we can write $|F_\zeta| \leq k|T_{\max}^{t_k}| \leq 2|T_{\max}^{t_1}|$, which implies $|F_\zeta| \leq 2|\mu_3(F_\zeta)|$ for all F_ζ in the domain of μ_3. Finally, observe that the number of ζ-components that are not mapped through μ_2 cannot be larger than $|V_{0;0}|$, thus the union of mappings μ_2 and μ_3 do actually map all ζ-components. This, in turn, implies that the union of the domains of the three mappings covers the whole set $T \setminus L_{\text{SEL}}$, thereby concluding the proof of part (a).

The proof of (b) is built on the definition of collision nodes, $[0; 0]$-nodes, $[0; \geq 1]$-nodes and $[\geq 1; \geq 1]$-nodes given in part (a). Let $L_t \subseteq L_{\text{SEL}}$ be the set of the first t nodes chosed by SEL. Here, we make a further distinction within the collision and $[0; \geq 1]$-nodes. We say that during the selection of node $i_t \in V_{0;1}$, the nodes in $L \cap T_{\max}^t$ are *captured* by i_t. This notion of capture extends to collision nodes by saying that a collision node $i_t \in L \cap L_{\text{SEL}}$ just *captures itself*. We say that i_t is an *initial* $[0; \geq 1]$-node (resp., *initial* collision node) if i_t is a $[0; \geq 1]$-node (resp., collision node) such that the whole set of nodes in L captured by i_t contains no nodes captured so far. See Figure 3 for reference. The simple observation leading to the proof of part (b) is the following. If i_t is a $[0; 0]$-node, then $T_{\max}^t$ cannot be larger than the component of $T \setminus L$ that contains $T_{\max}^t$, which in turn cannot be larger than $\Upsilon(L, 1)$. This would already imply $\Upsilon(L_{t-1}, 1) \leq \Upsilon(L, 1)$. Let now i_t be an initial $[0; \geq 1]$-node and T_{i_t} be the unique component of $T_{\max}^t \setminus \{i_t\}$ containing one or more nodes of L. Applying Lemma 2 to tree $T_{\max}^t$ we can see that $|T_{i_t}|$ cannot be larger than $|T_{\max}^t \setminus T_{i_t}|$, which in turn cannot be larger than $\Upsilon(L, 1)$. If at time $t' > t$ the procedure SEL selects $i_{t'} \in T_{i_t}$ then $|T_{\max}^{t'}| \leq |T_{i_t}| \leq \Upsilon(L, 1)$. Hence, the maximum integer q such that $\Upsilon(L_q, 1) > \Upsilon(L, 1)$ is bounded by the number of $[\geq 1; \geq 1]$-nodes plus the number of initial $[0; \geq 1]$-nodes plus the number of initial collision nodes. We now bound this sum as follows. The number of $[\geq 1; \geq 1]$-nodes is clearly bounded by $|L| - 1$. Also, any initial $[0; \geq 1]$-node or initial collision node selected by SEL captures at least a new node in L, thereby implying that the total number of initial $[0; \geq 1]$-node or initial collision node must be $\leq |L|$. After $q = 2|L| - 1$ rounds, we are sure that the size of the largest tree of $T_{\max}^q$ is not larger than the size of the largest component of $T \setminus L$, i.e., $\Upsilon(L, 1)$. ∎

We now put the above lemmas together to prove our main result concerning the number of mistakes made by PRED on the query set chosen by SEL.

Theorem 8 *For all trees T and all cutsize budgets K, the number of mistakes made by* PRED *on the query set L_{SEL}^+ satisfies*

$$m_{\mathrm{PRED}}(L_{\mathrm{SEL}}^+, K) \leq \min_{L \subseteq V \,:\, |L| \leq \frac{1}{8}|L_{\mathrm{SEL}}^+|} 10\,\mathrm{LB}(L, K) .$$

Proof: Pick any $L \subseteq V$ such that $|L| \leq \frac{1}{8}|L_{\mathrm{SEL}}^+|$. Then

$$m_{\mathrm{PRED}}(L_{\mathrm{SEL}}^+, K) \overset{\text{(Lem. 6)}}{\leq} \Upsilon(L_{\mathrm{SEL}}^+, K) \overset{\text{(A)}}{\leq} \Upsilon(L_{\mathrm{SEL}}, K) \overset{\text{(Lem. 7 (a))}}{\leq} 5\Upsilon(L^+, K) \overset{\text{(Thm. 5)}}{\leq} 10\,\mathrm{LB}(L^+, K) \overset{\text{(B)}}{\leq} 10\,\mathrm{LB}(L, K) .$$

Inequality (A) holds because $L_{\mathrm{SEL}} \subseteq L_{\mathrm{SEL}}^+$, and thus $T \setminus L_{\mathrm{SEL}}^+$ has connected components of smaller size than L_{SEL}. In order to apply Lemma 7 (a), we need the condition $|L^+| \leq \frac{1}{2}|L_{\mathrm{SEL}}|$. This condition is seen to hold after combining Lemma 3 with our assumptions: $|L^+| \leq 2|L| \leq \frac{1}{4}|L_{\mathrm{SEL}}^+| \leq \frac{1}{2}|L_{\mathrm{SEL}}|$. Finally, inequality (B) holds because any adversarial strategy using query set L can also be used with the larger query set $L^+ \supseteq L$. ■

Note also that Theorem 5 and Lemma 6 imply the following statement about the optimality of PRED over 0-forked query sets.

Corollary 9 *For all trees T, for all cutsize budgets K, and for all 0-forked query sets $L^+ \subseteq V$, the number of mistakes made by* PRED *satisfies $m_{\mathrm{PRED}}(L^+, K) \leq 2\mathrm{LB}(L^+, K)$.*

In the rest of this section we derive a more intepretable bound on $m_{\mathrm{PRED}}(L^+, \boldsymbol{y})$ based on the function Ψ^* introduced in [6]. To this end, we prove that L_{SEL} minimizes Ψ^* up to constant factors, and thus is an optimal query set according to the analysis of [6].

For any subset $V' \subseteq V$, let $\Gamma(V', V \setminus V')$ be the number of edges between nodes of V' and nodes of $V \setminus V'$. Using this notation, we can write

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

Lemma 10 *For any tree $T = (V, E)$ and any $L \subseteq V$ the following holds.*

(a) A maximizer of $\frac{|V'|}{\Gamma(V', V \setminus V')}$ exists which is included in the node set of a single component of $T \setminus L$;

(b) $\Psi^*(L) \leq \Upsilon(L, 1)$.

Proof: Let $V'_{\max}$ be any maximizer of $\frac{|V'|}{\Gamma(V', V \setminus V')}$. For the sake of contradiction, assume that the nodes of $V'_{\max}$ belong to $k \geq 2$ components $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k \in T \setminus L$. Let $V'_i \subset V'_{\max}$ be the subset of nodes included in the node set of $\mathcal{T}_i$, for $i = 1, \ldots, k$. Then $|V'| = \sum_{i \leq k} |V'_i|$ and $\Gamma(V', V \setminus V') = \sum_{i \leq k} \Gamma(V'_i, V \setminus V'_i)$. Now let $i^* = \mathrm{argmax}_{i \leq k} |V'_i|/\Gamma(V'_i, V \setminus V'_i)$. Since $\left(\sum_i a_i\right)/\left(\sum_i b_i\right) \leq \max_i a_i/b_i$ for all $a_i, b_i \geq 0$, we immediately obtain $\Psi(V'_{i^*}) \geq \Psi(V'_{\max})$, contradicting our assumption. This proves (a). Part (b) is an immediate consequence of (a). ■

Lemma 11 *For any tree $T = (V, E)$ and any 0-forked subset $L^+ \subseteq V$ we have $\Upsilon(L^+, 1) \leq 2\Psi^*(L^+)$.*

Proof: Let $\mathcal{T}_{\max}$ be the largest component of $T \setminus L^+$ and $V_{\max}$ be its node set. Since L^+ is a 0-forked query set, $\mathcal{T}_{\max}$ must be either a 1-hinge-tree or a 2-hinge-tree. Since the only edges that connect a hinge-tree to external nodes are the edges leading to connection nodes, we find that $\Gamma(V_{\max}, V \setminus V_{\max}) \leq 2$. We can now write

$$\Psi^*(L^+) = \max_{\emptyset \neq V' \subseteq V \setminus L^+} \frac{|V'|}{\Gamma(V', V \setminus V')} \geq \frac{|V_{\max}|}{\Gamma(V_{\max}, V \setminus V_{\max})} \geq \frac{|V_{\max}|}{2} = \frac{\Upsilon(L^+, 1)}{2}$$

thereby concluding the proof. ■

Lemma 12 *For any tree $T = (V, E)$ and any subset $L \subseteq V$ we have $\Psi^*(L^+) \leq \Psi^*(L)$.*

Proof: Let $V'_{\max}$ be any set maximizing $\Psi^*(L^+)$. Since $V'_{\max} \in V \setminus L^+$, $V'_{\max}$ cannot contain any node of $L \subseteq L^+$. Hence

$$\Psi^*(L) = \max_{\emptyset \neq V' \subseteq V \setminus L} \frac{|V'|}{\Gamma(V', V \setminus V')} \geq \frac{|V'_{\max}|}{\Gamma(V'_{\max}, V \setminus V'_{\max})} = \Psi^*(L^+)$$

which concludes the proof. ■

We now put together the previous lemmas to show that the query set L_{SEL} minimizes Ψ^* up to constant factors.

Theorem 13 *For any tree $T = (V, E)$ we have* $\quad \Psi^*(L_{\text{SEL}}) \leq \min_{L \subseteq V \,:\, |L| \leq \frac{1}{4}|L_{\text{SEL}}|} 2\Psi^*(L).$

Proof: Let L be a query set such that $|L| \leq |L_{\text{SEL}}|/4$. Then we have the following chain of inequalities:

$$\Psi^*(L_{\text{SEL}}) \overset{\text{(Lemma 10 (b))}}{\leq} \Upsilon(L_{\text{SEL}}, 1) \overset{\text{(Lemma 7 (b))}}{\leq} \Upsilon(L^+, 1) \overset{\text{(Lemma 11)}}{\leq} 2\Psi^*(L^+) \overset{\text{(Lemma 12)}}{\leq} 2\Psi^*(L) \,.$$

In order to apply Lemma 7 (b), we need the condition $|L^+| \leq \frac{1}{2}|L_{\text{SEL}}|$. This condition holds because, by Lemma 3, $|L^+| \leq 2|L| \leq \frac{1}{2}|L_{\text{SEL}}|$. ∎

Finally, as promised, the following corollary contains an interpretable mistake bound for PRED run with a query set returned by SEL.

Corollary 14 *For any labeled tree $(T, \boldsymbol{y})$, the number of mistakes made by PRED when run with query set L_{SEL}^+ satisfies*

$$m_{\text{PRED}}(L_{\text{SEL}}^+, \boldsymbol{y}) \leq 4 \min_{L \subseteq V \,:\, |L| \leq \frac{1}{8}|L_{\text{SEL}}^+|} \Psi^*(L) \, \Phi_T(\boldsymbol{y}) \,.$$

Proof: Observe that PRED assigns labels to nodes in $V \setminus L_{\text{SEL}}^+$ so as to minimize the resulting cutsize given the labels in the query set L_{SEL}^+. We can then invoke [6, Lemma 1], which bounds the number of mistakes made by the mincut strategy in terms of the functions Ψ^* and the cutsize. This yields

$$m_{\text{PRED}}(L_{\text{SEL}}^+, \boldsymbol{y}) \overset{\text{[6, Lemma 1]}}{\leq} 2\,\Psi^*(L_{\text{SEL}}^+)\,\Phi_T(\boldsymbol{y}) \overset{\text{(A)}}{\leq} 2\,\Psi^*(L_{\text{SEL}})\,\Phi_T(\boldsymbol{y}) \overset{\text{(Theorem 13)}}{\leq} 4\,\Psi^*(L)\,\Phi_T(\boldsymbol{y}) \,.$$

Inequality (A) holds because $L_{\text{SEL}} \subseteq L_{\text{SEL}}^+$, and thus $T \setminus L_{\text{SEL}}^+$ has connected components of smaller size than L_{SEL}. In order to apply Theorem 13, we need the conditon $|L| \leq \frac{1}{4}|L_{\text{SEL}}|$, which follows from a simple combination of Lemma 3 and our assumptions: $|L| \leq \frac{1}{8}|L_{\text{SEL}}^+| \leq \frac{1}{4}|L_{\text{SEL}}|$. ∎

Remark 2 *A mincut algorithm exists which efficiently predicts even when the query set L is not 0-forked (thereby gaining a factor of 2 in the cardinality of the competing query sets L – see Theorem 8 and Corollary 14). This algorithm is a "batch" variant of the TreeOpt algorithm analyzed in [7]. The algorithm can be implemented in such a way that the total time for predicting $|V| - |L|$ labels is $\mathcal{O}(|V|)$.*

4.3 Automatic calibration of the number of queries

A key aspect to the query selection task is deciding when to stop asking queries. Since the more queries are asked the less mistakes are made afterwards, a reasonable way to deal with this trade-off is to minimize the number of queries issued during the selection phase plus the number of mistakes made during the prediction phase. For a given pair $A = \langle S, P \rangle$ of prediction and selection algorithms, we denote by $[q + m]_A$ the sum of queries made by S and prediction mistakes made by P. Similarly to m_A introduced in Section 4, $[q + m]_A$ has to scale with the cutsize $\Phi_T(\boldsymbol{y})$ of the labeled tree $(T, \boldsymbol{y})$ under consideration.

As a simple example of computing $[q + m]_A$, consider a line graph $T = (V, E)$. Since each query set on T is 0-forked, Theorem 5 and Corollary 9 ensure that an optimal strategy for selecting the queries in T is choosing a sequence of nodes such that the distance between any pair of neighbor nodes in L is equal. The total number of mistakes that can be forced on $V \setminus L$ is, up to a constant factor, $\big(|V|/|L|\big)\Phi_T(\boldsymbol{y})$. Hence, the optimal value of $[q + m]_A$ is about

$$|L| + \frac{|V|}{|L|}\Phi_T(\boldsymbol{y}) \,. \tag{1}$$

Minimizing the above expression over $|L|$ clearly requires knowledge of $\Phi_T(\boldsymbol{y})$, which is typically unavailable. In this section we investigate a method for choosing the number of queries when the labeling is known to be sufficiently regular, that is when a bound K is known on the cutsize $\Phi_T(\boldsymbol{y})$ induced by the adversarial labeling.[5]

We now show that when a bound K on the cutsize is known, a simple modification of SEL(we call it SEL$\star$) exists which optimizes the $[q + m]_A$ criterion. This means that the combination of SEL$\star$ and PRED can trade-off optimally (up to constant factors) queries against mistakes.

---

[5]In [1] a labeling $\boldsymbol{y}$ of a graph G is said to be α-balanced if, after the elimination of all ϕ-edges, each connected component of G is not smaller than $\alpha|V|$ for some known constant $\alpha \in (0, 1)$. In the case of labeled trees, the α-balancing condition is stronger than our regularity assumption. This is because any α-balanced labeling $\boldsymbol{y}$ implies $\Phi_T(\boldsymbol{y}) \leq 1/\alpha - 1$. In fact, getting back to the line graph example, we immediately see that, if $\boldsymbol{y}$ is α-balanced, then the optimal number of queries $|L|$ is order of $\sqrt{|V|(1/\alpha - 1)}$, which is also $\inf_A [q + m]_A$.

Given a selection algorithm S and a prediction algorithm P, define $[q + m]_{\langle S,P \rangle}$ by

$$[q + m]_{\langle S,P \rangle} = \min_{Q \geq 1} \big(Q + m_P(L_{S(Q)}, K) \big)$$

where $L_{S(Q)}$ is the query set output by S given query budget Q, and $m_P(L_{S(Q)}, K)$ is the maximum number of mistakes made by P with query set $L_{S(Q)}$ on any labeling $\boldsymbol{y}$ with $\Phi_T(\boldsymbol{y}) \leq K$ —see definition in Section 4. Define also $[q + m]_{\text{OPT}} = \inf_{S,P} [q + m]_{\langle S,P \rangle}$, where $\text{OPT} = \langle S^*, P^* \rangle$ is an optimal pair of selection and prediction algorithms. If SEL knows the size of the query set L^* selected by S^*, so that SEL can choose a query budget $Q = 8|L^*|$, then a direct application of Theorem 8 guarantees that $|L_{\text{SEL}}^+| + m_{\text{PRED}}(L_{\text{SEL}}^+, K) \leq 10\,[q + m]_{\text{OPT}}$. We now show that SEL$\star$, the announced modification of SEL, can efficiently search for a query set size Q such that $Q + m_{\text{PRED}}(L_{\text{SEL}(Q)}^+, K) = \mathcal{O}\big([q + m]_{\text{OPT}} \big)$ when only K, rather than $|L^*|$, is known. In fact, Theorem 5 and Corollary 9 ensure that $m_{\text{PRED}}(L_{\text{SEL}}^+, K) = \Theta\big(\Upsilon(L_{\text{SEL}}^+, K) \big)$. When K is given as side information, SEL$\star$ can operate as follows. For each $t \leq |V|$, the algorithm builds the query set L_t^+ and computes $\Upsilon(L_t^+, K)$. Then it finds the smallest value t^* minimizing $t + \Upsilon(L_t^+, K)$ over all $t \leq |V|$, and selects $L_{\text{SEL}\star} \equiv L_{t^*}$. We stress that the above is only possible because the algorithm can estimate within constant factors its own future mistake bound (Theorem 5 and Corollary 9), and because the combination of SEL and PRED is competitive against all query sets whose size is a constant fraction of $|L_{\text{SEL}}^+|$ —see Theorem 8. Putting together, we have shown the following result.

Theorem 15 *For all trees $(T, \boldsymbol{y})$, for all cutsize budgets K, and for all labelings $\boldsymbol{y}$ such that $\Phi_T(\boldsymbol{y}) \leq K$, the combination of* SEL$\star$ *and* PRED *achieves $|L_{\text{SEL}\star}| + m_{\text{PRED}}(L_{\text{SEL}\star}^+, K) = \mathcal{O}\big([q + m]_{\text{OPT}} \big)$ when K is given to* SEL$\star$ *as input.*

Just to give a few simple examples of how SEL$\star$ works, consider a star graph. It is not difficult to see that in this case $t^* = 1$ *independent* of K, i.e., SEL$\star$ always selects the center of the star, which is intuitively the optimal choice. If T is the "binary system" mentioned in the introduction, then $t^* = 2$ and SEL$\star$ always selects the centers of the two stars, again independent of K. At the other extreme, if T is a line graph, then SEL$\star$ picks the query nodes in such a way that the distance between two consecutive nodes of L in T is (up to a constant factor) equal to $\sqrt{|V|/K}$. Hence $|L| = \Theta(\sqrt{|V|K})$, which is the minimum of (1) over $|L|$ when $\Phi_T(\boldsymbol{y}) \leq K$.

5   On the prediction of general graphs

In this section we provide a general lower bound for prediction on arbitrary labeled graphs $(G, \boldsymbol{y})$. We then contrast this lower bound to some results contained in Afshani et al. [1].

Let $\Phi_G^R(\boldsymbol{y})$ be the sum of the effective resistances (see, e.g., [9]) on the ϕ-edges of $G = (V, E)$. The theorem below shows that any prediction algorithm using any query set L such that $|L| \leq \frac{1}{4}|V|$ makes at least order of $\Phi_G^R(\boldsymbol{y})$ mistakes. This lower bound holds even if the algorithm is allowed to use a randomized adaptive strategy for choosing the query set L, that is, a randomized strategy where the next node of the query set is chosen after receiving the labels of all previously chosen nodes.

Theorem 16 *Given a labeled graph $(G, \boldsymbol{y})$, for all $K \leq |V|/2$, there exists a randomized labeling strategy such that for all prediction algorithms A choosing a query set of size $|L| \leq \frac{1}{4}|V|$ via a possibly randomized adaptive strategy, the expected number of mistakes made by A on the remaining nodes $V \setminus L$ is at least $K/4$, while $\Phi_G^R(\boldsymbol{y}) \leq K$.*

The above lower bound (whose proof is omitted) appears to contradict an argument by Afshani et al. [1, Section 5]. This argument establishes that for any $\varepsilon > 0$ there exists a randomized algorithm using at most $K \ln(3/\varepsilon) + K \ln(|V|/K) + \mathcal{O}(K)$ queries on any given graph $G = (V, E)$ with cutsize K, and making at most $\varepsilon|V|$ mistakes on the remaining vertices. This contradiction is easily obtained through the following simple counterexample: assume G is a line graph where all node labels are $+1$ but for $K = o(|V|/\ln|V|)$ randomly chosen nodes, which are also given random labels. For all $\varepsilon = o\big(\frac{K}{|V|}\big)$, the above argument implies that order of $K \ln|V| = o(|V|)$ queries are sufficient to make at most $\varepsilon|V| = o(K)$ mistakes on the remaining nodes, among which $\Omega(K)$ have random labels —which is clearly impossible.

6   Efficient Implementation

In this section we describe an efficient implementation of SEL and PRED. We will show that the total time needed for selecting Q queries is $\mathcal{O}(|V| \log Q)$, the total time for predicting $|V| - Q$ nodes is $\mathcal{O}(|V|)$, and that the overall memory space is again $\mathcal{O}(|V|)$.

In order to locate the largest subtree of $T \setminus L_{t-1}$, the algorithm maintains a priority deque [8] D containing at most Q items. This data-structure enables to find and eliminate the item with the smallest (resp., largest) key in time $\mathcal{O}(1)$ (resp., time $\mathcal{O}(\log Q)$). In addition, the insertion of a new element takes time $\mathcal{O}(\log Q)$.

Each item in D has two records: a reference to a node in T and the priority key associated with that node. Just before the selection of the[6] t-th query node i_t, the Q references point to nodes contained in the Q largest subtrees in $T \setminus L_{t-1}$, while the corresponding keys are the sizes of such subtrees. Hence at time t the item top of D having the largest key points to a node in $T^t_{\max}$.

First, during an initialization step, SEL creates, for each edge $(i, j) \in E$, a directed edge $[i, j]$ from i to j and the twin directed edge $[j, i]$ from j to i. During the construction of L_{SEL} the algorithm also stores and maintains the current size $\sigma(D)$ of D, i.e., the total number of items contained in D. We first describe the way SEL finds node i_t in $T^t_{\max}$. Then we will see how SEL can efficiently augment the query set L_{SEL} to obtain L^+_{SEL}.

Starting from the node r of $T^t_{\max}$ referred to by[7] D, SEL performs a depth-first visit of $T^t_{\max}$, followed by the elimination of the item with the largest key in D. For the sake of simplicity, consider $T^t_{\max}$ as rooted at node r. Given any edge (i, j), we let T_i and T_j be the two subtrees obtained from $T^t_{\max}$ after removing edge (i, j), where T_i contains node i, and T_j contains node j. During each backtracking step of the depth-first visit from a node i to a node j, SEL stores the number of nodes $|T_i|$ contained in T_i. This number gets associated with $[j, i]$. Observe that this task can be accomplished very efficiently, since $|T_i|$ is equal to 1 plus the number of nodes of the union of $T_{c(i)}$ over all children $c(i)$ of i. These numbers can be recursively calculated by summing the size values that SEL associates with all direct edges $[i, c(i)]$ in the previous backtracking steps. Just after storing the value $|T_i|$, the algorithm also stores $|T_j| = |T^t_{\max}| - |T_i|$ and associates this value with the twin directed edge $[i, j]$. The size of $T^t_{\max}$ is then stored in D as the key record of the pointer to node r.

It is now important to observe that the quantity $\sigma(T^t_{\max}, i)$ used by SEL (see Section 3) is simply the largest key associated with the directed edges $[i, j]$ over all j such that (i, j) is an edge of $T^t_{\max}$. Hence, a new depth-first visit is enough to find in time $\mathcal{O}(|T^t_{\max}|)$ the t-th node $i_t = \arg\min_{i \in T^t_{\max}} \sigma(T^t_{max}, i)$ selected by SEL. Let $N(i_t)$ be the set of all nodes adjacent to node i_t in $T^t_{\max}$. For all nodes $i' \in N(i_t)$, SEL compares $|T_{i'}|$ to the smallest key $bottom$ stored in D. We have three cases:

1. If $|T_{i'}| \le bottom$ and $\sigma(D) \ge Q - t$ then the algorithm does nothing, since $T_{i'}$ (or subtrees thereof) will never be largest in the subsequent steps of the construction of L_{SEL}, i.e., there will not exist any node $i_{t'}$ with $t' > t$ such that $i_{t'} \in T_{i'}$.

2. If $|T_{i'}| \le bottom$ and $\sigma(D) < Q - t$, or if $|T_{i'}| > bottom$ and $\sigma(D) < Q$ then SEL inserts a pointer to i' together with the associated key $|T_{i'}|$. Note that, since D is not full (i.e., $\sigma(D) < Q$), the algorithm need not eliminate any item in D.

3. If $|T_{i'}| > bottom$ and $\sigma(D) = Q$ then SEL eliminates from D the item having the smallest key, and inserts a pointer to i', together with the associated key $|T_{i'}|$.

Finally, SEL eliminates node i_t and all edges (both undirected and directed) incident to it. Note that this elimination implies that we easily perform a depth-first visit within $T^s_{\max}$ for each $s \le Q$, since $T^s_{\max}$ is always completely disconnected from the rest of the tree T.

In order to turn L_{SEL} into L^+_{SEL}, the algorithm proceeds incrementally, using a technique borrowed from [7]. Just after the selection of the first node i_1, a depth-first visit starting from i_1 is performed. During each backtracking step of this visit, the algorithm associates with each edge (i, j), the closer node to i_1 between the two nodes i and j. In other words, SEL assigns a direction to each undirected edge (i, j) so as to be able to efficiently find the path connecting each given node i to i_1. When the t-th node i_t is selected, SEL follows these edge directions from i_t towards i_1. Let us denote by $\pi(i, j)$ the path connecting node i to node j. During the traversal of $\pi(i_1, i_t)$, the algorithm assigns a special mark to each visited node, until the algorithm reaches the first node $j \in \pi(i_1, i_t)$ which has already been marked. Let $\eta(i, L)$ be the maximum number of edge disjoint paths connecting i to nodes in the query set L. Observe that all nodes i for which $\eta(i, L_t) > \eta(i, L_{t-1})$ must necessarily belong to $\pi(i_t, j)$. We have $\eta(i_t, L_t) = 1$, and $\eta(i, L_t) = 2$, for all internal nodes i in the path $\pi(i_t, j)$. Hence, j is the unique node that we may need to add as a new fork node (if $j \notin \text{FORK}(L_{t-1})$). In fact, j is the unique node such that the number of edge-disjoint paths connecting it to query nodes may increase, and be actually larger than 2.

Therefore if $j \in L^+_{t-1}$ we need not add any fork node during the incremental construction of L^+_{SEL}. On the other hand, if $j \notin L^+_{t-1}$ then $\eta(i, L_{t-1}) = 2$, which implies $\eta(i, L_t) = 3$. This is the case when SEL views j as new fork node to be added to the query set L_{SEL} under consideration.

In order to bound the total time required by SEL for selecting Q nodes, we rely on Lemma 4, showing that $|T^t_{\max}| \le 2|V|/t$. The two depth-first visits performed for each node i_t take $\mathcal{O}(|T^t_{\max}|)$ steps. Hence the overall running time spent on the depth-first visits is $\mathcal{O}(\sum_{t \le Q} 2|V|/t) = \mathcal{O}(|V| \log Q)$. The total time

---

[6]If $t = 1$ the priority deque D is empty.

[7]In the initial step $t = 1$ (i.e., when $T^t_{\max} \equiv T$) node r can be chosen arbitrarily .

spent for incrementally finding the fork nodes of L_{SEL} is linear in the number of nodes marked by the algorithm, which is equal to $|V|$. Finally, handling the priority deque D takes $|V|$ times the worst-case time for eliminating an item with the smallest (or largest) key or adding a new item. This is again $\mathcal{O}(|V| \log Q)$.

We now turn to the implementation of the prediction phase. PRED operates in two phases. In the first phase, the algorithm performs a depth-first visit of each hinge-tree $\mathcal{T}$, starting from each connection node (thereby visiting the nodes of all 1-hinge-tree once, and the nodes of all 2-hinge-tree twice). During these visits, we add to the nodes a tag containing (i) the label of node $i_{\mathcal{T}}$ from which the depth-first visit started, and (ii) the distance between $i_{\mathcal{T}}$ and the currently visited node. In the second phase, we perform a second depth-first visit, this time on the whole tree T. During this visit, we predict each node $i \in V \setminus L$ with the label coupled with smaller distance stored in the tags of[8] i. The total time of these visits is linear in $|V|$ since each node of T gets visited at most 3 times.

7 Conclusions and ongoing work

The results proven in this paper characterize, up to constant factors, the optimal algorithms for adversarial active learning on trees in two main settings. In the first setting the goal is to minimize the number of mistakes on the non-queried vertices under a certain query budget. In the second setting the goal is to minimize the sum of queries and mistakes under no restriction on the number of queries.

An important open question is the extension of our results to the general case of active learning on graphs. While a direct characterization of optimality on general graphs is likely to require new analytical tools, an alternative line of attack is reducing the graph learning problem to the tree learning problem via the use of spanning trees. Certain types of spanning trees, such as random spanning trees, are known to summarize well the graph structure relevant to passive learning —see, e.g., [7]. In the case of active learning, however, we want good query sets on the graph to correspond to good query sets on the spanning tree, and random spanning trees may fail to do so in simple cases. For example, consider a set of m cliques connected through bridges, so that each clique is connected to, say, k other cliques. The breadth-first spanning tree of this graph is a set of connected stars. This tree clearly reveals a query set (the star centers) which is good for regular labelings (cfr., the binary system example of Section 1). On the other hand, for certain choices of m and k a random spanning tree has a good probability of hiding the clustered nature of the original graph, thus leading to the selection of bad query sets.

In order to gain intuition about this phenomenon, we are currently running experiments on various real-world graphs using different types of spanning trees, where we measure the number of mistakes made by our algorithm (for various choices of the budget size) against common baselines.

We also believe that an extension to general graphs of our algorithm does actually exist. However, the complexity of the methods employed in [6] suggests that techniques based on minimizing Ψ^* on general graphs are computationally very expensive.

References

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

[2] Belkin, M., Matveeva, I., and Niyogi, P. Regularization and semi-supervised learning on large graphs. *COLT 2004*, pages 624–638.

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

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

[5] A. Blum, J. Lafferty, R. Reddy, and M.R. Rwebangira. Semi-supervised learning using randomized mincuts. *ICML 2004*.

[6] A. Guillory and J. Bilmes. Label Selection on Graphs. *NIPS 2009*.

[7] N. Cesa-Bianchi, C. Gentile, F. Vitale. Fast and optimal prediction of a labeled tree. *COLT 2009*.

[8] J. Katajainen, F. Vitale. Navigation piles with applications to sorting, priority queues, and priority deques. *Nordic Journal of Computing*, 10(3):238–262, 2003.

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

[10] Zhu, X., Ghahramani, Z., and Lafferty, J. 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.

---

[8]If i belongs to a 1-hinge-tree, we simply predict y_i with the unique label stored in the tag.