Hierarchical Active Transfer Learning

David Kale* Marjan Ghazvininejad* Anil Ramakrishna* Jingrui He† Yan Liu*

Abstract

We describe a unified active transfer learning framework called *Hierarchical Active Transfer Learning* (HATL). HATL exploits cluster structure shared between different data domains to perform transfer learning by imputing labels for unlabeled target data and to generate effective label queries during active learning. The resulting framework is flexible enough to perform not only adaptive transfer learning and accelerated active learning but also unsupervised and semi-supervised transfer learning. We derive an intuitive and useful upper bound on HATL's error when used to infer labels for unlabeled target points. We also present results on synthetic data that confirm both intuition and our analysis. Finally, we demonstrate HATL's empirical effectiveness on a benchmark data set for sentiment classification.

1 Introduction

In the era of *big* data, one major challenge confronting us is that typically only a *small* portion of the vast data is labeled, since annotation is often both time consuming and costly. This can limit the performance of predictive models and their ability to generalize to new observations. In response to this challenge, researchers have explored two complementary directions: *transfer learning* and *active learning*.

The goal of *transfer learning* (and related techniques, such as *domain adaptation*), is to leverage the labeled data from a related *source* domain (or task) to compensate for lack of (labeled) data in a *target* domain. Transfer learning has seen significant success in applications, such as computer vision [16] and natural language processing [12]. However, it is also notorious for producing *negative transfer*, where transferring source knowledge hinders learning in the target domain, especially when *zero* target labels are available [14].

In some applications, we have access to a label *oracle* with limited query *budget*. In these cases, *active learning* can identify a small set of examples that, if labeled, will help to train effective predictive models. A vibrant area of machine learning research over the past decade, active learning has produced a number of practical and theoretical breakthroughs, as well as a wide spectrum of algorithms [17, 5]. Nonetheless, active learning is not a panacea; in particular, it has a circular dependency on data. In order to pose good label queries, we must have a reasonably good classifier; in order to train a good classifier, we need labeled data. When starting with no labels (i.e., a *cold start*), most active learners must resort to random sampling [5, 10].

In this paper, we develop a new hierarchical clustering-based framework, named *Hierarchical Active Transfer Learning* (HATL), that combines these two learning schemes. It takes as input a cluster tree built on source and target data and uses cluster structure and label information from both domains to impute labels on the full data set (source and target). During early learning it relies heavily on labeled source points but gradually incorporates feedback from target label inquiries to refine both its clustering and its label imputation. This accelerates the learning process and mitigates the cold start problem. We derive an intuitive and useful upper bound on HATL's error when used to infer labels for unlabeled target points and present results on synthetic data that deliver insight into our theoretical analysis. We demonstrate HATL's empirical effectiveness on a benchmark data set for sentiment classification.

The paper is organized as follows. In Section 2, we briefly review the related work. The proposed framework is introduced in Section 3, followed by theoretical analysis in Section 4. Finally, we provide experimental results in Section 5 and conclude in Section 6.

2 Related Work

There is a growing body of excellent research on combining active learning with transfer learning [15, 13], but the topic still has many open problems and gaps to bridge between theory and practice. [15] represents some of the earliest active transfer learning work and describes a simple but intuitive solution that uses a source-trained classifier as a cost-free surrogate for the target label oracle. The paper analyzes label complexity and error rates and demonstrates convincing empirical re-

*Computer Science Department, Viterbi School of Engineering, University of Southern California, USA

†School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, USA

sults. This approach uses uncertainty region sampling, which can be susceptible to bias and local optima [5].

[4] proposes an optimization-based framework (JOTAL) for adaptive transfer learning with feedback from batch target label queries. What makes this framework unique is its choice of objective function: instead of target classification error, it optimizes the similarity (measured using kernel *maximum mean discrepancy* [8]) between labeled source and target and remaining unlabeled target data points. Empirical results suggest that JOTAL is one of the most effective active transfer learning frameworks to date. The algorithm, however, lacks a rigorous theoretical treatment and uses a quadratic programming formulation that may require modification to scale up to large data sets.

[19] describes a theoretically rigorous Bayesian framework for active transfer learning, based on *prior-dependent learning*. Assuming a prior distribution over model parameters can accelerate active learning, but this requires access to the prior [18]. [19] shows that in sequential transfer learning settings, the prior is identifiable and places an upper bound on the number of samples required from each domain. However, there is no limit on the number of tasks that may be required, and the paper presents no experimental results.

A recent paper that bridges the gap between theoretical soundness and empirical effectiveness is [10], which presents a principled framework called transfer-accelerated, importance weighted consistent active learning (TIWCAL). The authors combine the agnostic active learning algorithm of [1] with transfer learning based on minimizing a convex combination of source and target errors [2]. They provide efficient algorithm and derive an intuitive upper bound on target classification error. Their empirical results show that TIWCAL can improve target task performance while reducing the number of target label queries. This approach, however, suffers from the use of static transfer learning, based on an *a priori* choice of domain weights.

3 The Proposed Framework

In this section, we introduce our *Hierarchical Active Transfer Learning* (HATL) framework. We define notation in **Section 3.1**, briefly review Hierarchical Sampling for Active Learning (HSAL) [6] in **Section 3.2**, and describe HATL in **Section 3.3**.

3.1 Notation: Let $\mathcal{X}$ denote a finite sample of data points from a distribution $\mathcal{D}$. Let f be a true labeling function, so that the label of a point x is $f(x)$. We will add subscripts to denote samples and distributions for different domains (e.g., $\mathcal{X}_{\mathrm{T}}$, $\mathcal{D}_{\mathrm{T}}$, and f_{T} for the target domain). We will view learning as searching for a hypothesis $h \in \mathcal{H}$. Now let T denote the binary tree representing a hierarchical clustering of the data points in $\mathcal{X}$. For any node (or cluster) v in T, let T_v denote the hierarchy of nodes (or subtree) rooted at v. In particular, $T = T_{\mathrm{root}}$. If we descend far enough down T_v, we reach a leaf node x, which is a data point. Denote the set of points associated with arbitrary v as $X(v)$. Each v has two child nodes u_1 and u_2, such that $X(v) = X(u_1) \cup X(u_2)$. Trivially, $X(x) = \{x\}$ for leaf nodes. A pruning P_v is a subset of non-overlapping nodes of T_v that contains all points associated with v: $X(v) = \bigcup_{u \in P_v} X(u)$. P is a pruning of the complete tree T. A labeling function $L(v)$ is a mapping of a node v to a label (e.g., $L(v) : v \mapsto \pm 1$).

3.2 A Brief Review of HSAL: The *Hierarchical Sampling for Active Learning* (HSAL) [6] algorithm was introduced in [6] and is described in detail in **Appendix 7**.[1] HSAL begins with a cluster tree T over N points in $\mathcal{X}$ and a label query budget of B. At all times, it maintains a current P and L for T, with initial values of $P = \{\mathrm{root}\}$ and $L(\mathrm{root}) = +1$. Each iteration of HSAL consists of four main steps. First, it queries labels for a batch of b unlabeled points. Second, it estimates the label proportions in each $v \in P$. Next, it updates P by replacing any node v with its children if it has a high *label disagreement* (i.e., high likelihood that the label proportions are equal, quantified using confidence intervals). Finally, it updates L by letting $L(v)$ equal the estimated majority label for each v. Label queries are made by choosing first a cluster v from P and then an unlabeled point x from $X(v)$. [6] suggests choosing v with probability proportional to both its size and uncertainty about its majority label.

Upon termination, HSAL produces a fully labeled data set $\mathcal{Y} = \{(x, \hat{y}(x)) : \forall x \in \mathcal{X}\}$ by assigning $\hat{y}(x) = L(v)$ to each $x \in X(v)$. $\mathcal{Y}$ can be used to train any classifier. This *label imputation* step may assign some incorrect labels, but it also avoids the sample selection bias suffered by other active learning algorithms [6]. The goal of HSAL, then, is to use B label queries to search for P and L with the minimum possible *label imputation error* $\epsilon(P, L)$, where $\epsilon(P, L) = (1/N) \sum_{v \in P} \sum_{x \in X(v)} (L(v) \neq f(x))$. This is the error when each point is assigned the label $L(v)$ of its associated cluster v in P. It can be thought of as transductive classification error on the clustered data.

3.3 Hierarchical Active Transfer Learning: In active transfer learning, we have labeled data $\mathcal{X}_{\mathrm{S}}$ from

[1] Available online at `http://www-scf.usc.edu/~dkale/publications.html`

the source domain, in addition to unlabeled target data $\mathcal{X}_{\mathrm{T}}$. Our proposed HATL framework can leverage $\mathcal{X}_{\mathrm{S}}$ and a limited number of queried target labels to help impute labels for the full $\mathcal{X}_{\mathrm{T}}$, which we can then use to train an accurate classifier for the target domain. HATL is summarized in **Algorithm 1**.

Algorithm 1 Hierarchical Active Transfer Learning (HATL)

Input: Hierarchical cluster tree T of unlabeled target data $\mathcal{X}_{\mathrm{T}}$ and labeled source data $\mathcal{X}_{\mathrm{S}}$; target label budget B and batch size b; target label *oracle*

1: $P \longleftarrow \{\text{root}\}$, $L(\text{root}) \longleftarrow +1$
2: **for each** (x, y) where $x \in \mathcal{X}_{\mathrm{S}}$, $y = f_{\mathrm{S}}(x)$ **do**
3: *UpdateNodeStatistics*(x, y, root)
4: **end for**
5: $(P, L) \leftarrow$ *GetPruningAndLabeling*(T)
6: **if** $B > 0$ **then**
7: $(P, L) \leftarrow$ *HSAL*$(T, \{\mathcal{X}_{\mathrm{T}}, \mathcal{X}_{\mathrm{S}}\}, P, L, B, b, \textit{oracle})$
8: **end if**
9: **for each** $v \in P$ **do**
10: $\hat{y}(x) \longleftarrow L(v)$ **for each** $x \in \mathcal{X}(v)$
11: **end for**

Output: Labeled source and target data: $\mathcal{Y}_{\mathrm{S}} = \{(x, \hat{y}(x)) : \forall x \in \mathcal{X}_{\mathrm{S}}\}$, $\mathcal{Y}_{\mathrm{T}} = \{(x, \hat{y}(x)) : \forall x \in \mathcal{X}_{\mathrm{T}}\}$.

We begin with cluster tree T over $\mathcal{X}_{\mathrm{S}} \cup \mathcal{X}_{\mathrm{T}}$, a label budget B and batch size b, and target label *oracle*. On **line 1**, we initialize P to the root of T and L to be an arbitrary label. Then in **lines 2-4**, we update the label proportion estimates for all nodes, based on labeled source points. The *UpdateNodeStatistics*(x, y, v) subroutine performs this update for all nodes along the path from x to v in T_v. On **line 5**, we update P and L using the *GetPruningAndLabeling*(T_v) subroutine, which recursively splits nodes in T_v that have high label disagreement. If the budget $B > 0$, we run HSAL on the mixture of source and target data but using the updated P and L (**lines 6-8**). Finally, we impute labels for all source and target points in **lines 9-11** and output the fully labeled data sets. The *UpdateNodeStatistics* and *GetPruningAndLabeling* subroutines are implemented as in HSAL and are described in **Appendix 7**.[1]

Discussion. Consider P and L at **line 5**, following the initial update and pruning but before any target label queries. Every cluster v in P must have *either* a clear source majority label $L(v)$ *or* a proportionally small number of labeled source points (so that $L(v)$ could not be estimated with high confidence). Otherwise, v would have been replaced with its children. In the latter case, the source data in v will play a limited role during subsequent active learning (**line 7**), and the number of target queries required to choose $L(v)$ will be similar to that in plain HSAL. In the first case, the source-based $L(v)$ provides a strong bias about the majority label in v. If most target labels in v agree with $L(v)$, then the bias is beneficial, and we can achieve a low label imputation error with few, if any, target label queries. If, however, most target labels in v disagree with $L(v)$, then we must query target labels until we confirm this and split v.

A natural question to ask is under what conditions HATL might perform poorly or even worse than plain HSAL? First, suppose that source and target distributions are quite similar, forming several clear large clusters, but that source and target points within each cluster have the opposite labels. The initial pruning P (**line 5**) will include large clusters with relatively pure source labels but high target label imputation error. HATL may need to query a large number of target labels in order to find separate source-only and target-only clusters in order to improve its labeling. Meanwhile, HSAL will discover the large, pure target clusters with a small number of queries. *Thus, HATL requires that the source and target tasks share reasonably similar labeling functions.* We will formalize this intuition in **Section 4**.

Now suppose that the source and target data share similar labeling functions but have very different cluster structures with few overlapping regions. The initial pruning (**line 5**) will be pure with respect to source labels but may not reflect the natural cluster structure of the target data and may group together source and target points with different labels. HATL's initial label imputation may be superior to that of HSAL, but HSAL will improve rapidly as it finds the large, pure target clusters with a small number of queries. HATL, in contrast, will likely plateau until it queries enough target labels to discover a pruning that matches the target cluster structure. *Thus, the more similar the target and source distributions and cluster structure*, the better HATL's performance. In **Section 4** we will derive an upper bound on the target label imputation error that incorporates domain similarity.

4 Theoretical Analysis

Here we provide a theoretical analysis of the performance of HATL. We begin by stating our main theoretical result, which places an upper bound on the target label imputation error, given that HATL has queried a minimum number of labels required to find a relatively pure pruning of the cluster tree.

THEOREM 4.1. *Suppose we have constructed a cluster tree T on over N_{S} source points drawn from $\mathcal{D}_{\mathrm{S}}$ and N_{T} target points drawn $\mathcal{D}_{\mathrm{T}}$. Choose $\delta, \eta > 0$ and any pruning P^* of T with label imputation error $\epsilon(P^*) \leq \eta$.*

Now assume that HATL has queried bt *labels to discover pruning P. Then with probability at least $1-\delta$ our target label imputation error $\epsilon_{\mathrm{T}}(P)$ is*

$$\epsilon_{\mathrm{T}}(P) \leq \widetilde{O}(\epsilon(P^*) + \eta) + (1-\alpha)\left(\frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{\mathrm{S}}, \mathcal{D}_{\mathrm{T}}) + \varepsilon^*_{\mathrm{ST}}\right) + 2\sqrt{\frac{D\log(2N_{\mathrm{T}}) - \log\delta}{2N_{\mathrm{T}}}}$$

where $bt = \widetilde{O}\left(\frac{|P^*|}{\eta}\log(\frac{2^{d^*}b|P^*|}{\eta\delta})\right)$ (Theorem 1 from [6]) , $\alpha = N_{\mathrm{T}}/(N_{\mathrm{S}}+N_{\mathrm{T}})$, $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{\mathrm{S}}, \mathcal{D}_{\mathrm{T}})$ is the distance between the source and target distributions $\mathcal{D}_{\mathrm{S}}$ and $\mathcal{D}_{\mathrm{T}}$ [2], $\varepsilon^*_{\mathrm{ST}}$ is the combined risk of the ideal hypothesis [2], t is the number of iterations, b is the batch size (queries per iteration), D is the VC dimension of the hypothesis class, and d^* and $|P^*|$ are the depth and size of P^*.

This theorem places an upper bound on the target label imputation error with three intuitive terms. The first is related to the overall (source and target) label imputation error for optimal P^*. The second is a distance measure between source and target data distributions, weighted by the proportion $1-\alpha$ of the data that comes from the source domain. This term is small when our tasks are similar or when $N_{\mathrm{T}} \gg N_{\mathrm{S}}$. The third comes from an application of Hoeffding's inequality for bounding the deviation between empirical and true errors and will be relatively small when N_{T} is large.

The proof, given below, builds upon *Theorem 1* from [6] , which provides a guarantee about the number of label queries bt needed by HSAL to find a P with label imputation error no worse than $\epsilon(P^*) + \eta$, where $\epsilon(P^*) \leq \eta$. The total number of queries, bt, depends upon the number of clusters in P^*, the maximum depth d^* of a cluster in P^*, and choices of δ and η.

4.1 Proof of *Theorem 4.1*: To prove our theorem, we begin by introducing several definitions and lemmas. First, we will use $d_{\mathcal{H}}$ distance [11], a hypothesis class-dependent distance between data distributions:

Definition 4.1. ($d_{\mathcal{H}}$ distance [11]) *Suppose we have source and target data distributions $\mathcal{D}_{\mathrm{S}}$ and $\mathcal{D}_{\mathrm{T}}$ and hypothesis class $\mathcal{H}$. We can define a $d_{\mathcal{H}}$ distance [11] between $\mathcal{D}_{\mathrm{S}}$ and $\mathcal{D}_{\mathrm{T}}$ as:*

$$d_{\mathcal{H}}(\mathcal{D}_{\mathrm{S}}, \mathcal{D}_{\mathrm{T}}) \triangleq 2 \sup_{h\in\mathcal{H}} |P_{\mathcal{D}_{\mathrm{S}}}\{\mathcal{A}_h\} - P_{\mathcal{D}_{\mathrm{T}}}\{\mathcal{A}_h\}|$$

where $\mathcal{A}_h = \{x : x \in \mathcal{X} \text{ where } h(x) = +1\}$, the set of points classified as positive by h.

The $d_{\mathcal{H}}$ distance is the difference between probability masses assigned by the source and target domains to $\mathcal{A}_h$, maximized over hypotheses $h \in \mathcal{H}$. Throughout this paper, we will use $d_{\mathcal{H}}$ with a specific hypothesis class $\mathcal{H}$ that consists of one nearest neighbor (1NN) classifiers that can be constructed from points in cluster tree T that have been relabeled using (P, L).

We actually use the more specialized $d_{\mathcal{H}\Delta\mathcal{H}}$ distance [2], based on the *symmetric difference hypothesis class*:

Definition 4.2. ($\mathcal{H}\Delta\mathcal{H}$ hypothesis class [2]) *Define the symmetric difference hypothesis class as*

$$\mathcal{H}\Delta\mathcal{H} = \{g : g(x) = +1 \iff h(x) \neq h'(x), h, h' \in \mathcal{H}\}$$

For each pair of hypotheses $h, h' \in \mathcal{H}$, there exists a $g \in \mathcal{H}\Delta\mathcal{H}$ such that $g(x) = +1$ if and only if $h(x) \neq h'(x)$.

Using $\mathcal{H}\Delta\mathcal{H}$ we can define the $d_{\mathcal{H}\Delta\mathcal{H}}$ distance, for which the following inequality holds for all $h, h' \in \mathcal{H}$ [2]:

$$|\varepsilon_{\mathrm{S}}(h, h') - \varepsilon_{\mathrm{T}}(h, h')| \leq \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{\mathrm{S}}, \mathcal{D}_{\mathrm{T}})$$

where $\varepsilon_{\mathrm{S}}(h, h') = P\{h(x) \neq h'(x) \text{ for } x \sim \mathcal{D}_{\mathrm{S}}\}$ is the label disagreement between $h, h' \in \mathcal{H}$ on points from $\mathcal{D}_{\mathrm{S}}$. This places an upper bound on the difference between source and target label disagreement for any pair of hypotheses in $\mathcal{H}$. $\varepsilon_{\mathrm{S}}(h)$ refers to $\varepsilon_{\mathrm{S}}(h, f_{\mathrm{S}})$, label disagreement with the true labeling function f_{S}.

We now consider a *convex combination of source and target errors* [2], e.g., $\varepsilon^{\alpha}_{\mathrm{ST}}(h) = (1-\alpha)\varepsilon_{\mathrm{S}}(h) + \alpha\varepsilon_{\mathrm{T}}(h)$ for $0 < \alpha < 1$. *Lemma 1* from [2] upper bounds the difference between combined and target risk for hypothesis h:

$$|\varepsilon^{\alpha}_{\mathrm{ST}}(h) - \varepsilon_{\mathrm{T}}(h)| \leq (1-\alpha)\left(\frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{\mathrm{S}}, \mathcal{D}_{\mathrm{T}}) + \varepsilon^*_{\mathrm{ST}}\right)$$

where $\varepsilon^*_{\mathrm{ST}} = \min_{h\in\mathcal{H}} \varepsilon_{\mathrm{S}}(h) + \varepsilon_{\mathrm{T}}(h)$ is the *combined risk* of the hypothesis in $\mathcal{H}$ that simultaneously minimizes source and target error [2]. This quantifies the excess risk that is due to the difficulty of the task and the differences between the labeling functions f_{S} and f_{T}. Most domain adaptation and transfer learning frameworks require that $\varepsilon^*_{\mathrm{ST}}$ be small in order to succeed [2].

Next we prove a lemma that shows that we can rewrite the label imputation error as a convex combination of source and target empirical errors:

Lemma 4.1. *Suppose we have run HATL on source and target samples $\mathcal{X}_{\mathrm{S}}$ and $\mathcal{X}_{\mathrm{T}}$ and discovered pruning P. The label imputation error $\epsilon(P)$ can be rewritten as*

$$\epsilon(P) = \varepsilon^{\alpha}_{\mathcal{X}_{\mathrm{ST}}}(h_P) = (1-\alpha)\varepsilon_{\mathcal{X}_{\mathrm{S}}}(h_P) + \alpha\varepsilon_{\mathcal{X}_{\mathrm{T}}}(h_P)$$

where h_P is a 1NN classifier induced by P; $\varepsilon_{\mathcal{X}_{\mathrm{S}}}$, $\varepsilon_{\mathcal{X}_{\mathrm{T}}}$, and $\varepsilon^{\alpha}_{\mathcal{X}_{\mathrm{ST}}}$ are the source, target, and combined empirical errors using $\mathcal{X}_{\mathrm{S}}$, $\mathcal{X}_{\mathrm{T}}$, and $\mathcal{X}_{\mathrm{ST}} = \mathcal{X}_{\mathrm{S}} \cup \mathcal{X}_{\mathrm{T}}$; and $\alpha = \frac{N_{\mathrm{T}}}{N_{\mathrm{S}}+N_{\mathrm{T}}}$.

Proof.

$$\begin{aligned}
\epsilon(P) &= \frac{1}{N_\mathrm{S}+N_\mathrm{T}}\left(\sum_{x\in\mathcal{X}_\mathrm{S}} \mathbb{1}\{h_P(x)\neq f_\mathrm{S}(x)\}\right.\\
&\qquad\left.+\sum_{x\in\mathcal{X}_\mathrm{T}} \mathbb{1}\{h_P(x)\neq f_\mathrm{T}(x)\}\right)\\
&= \frac{N_\mathrm{S}}{N_\mathrm{S}+N_\mathrm{T}}\frac{1}{N_\mathrm{S}}\sum_{x\in\mathcal{X}_\mathrm{S}} \mathbb{1}\{h_P(x)\neq f_\mathrm{S}(x)\}\\
&\quad+\frac{N_\mathrm{T}}{N_\mathrm{S}+N_\mathrm{T}}\frac{1}{N_\mathrm{T}}\sum_{x\in\mathcal{X}_\mathrm{T}} \mathbb{1}\{h_P(x)\neq f_\mathrm{T}(x)\}\\
&= (1-\alpha)\varepsilon_{\mathcal{X}_\mathrm{S}}(h_P)+\alpha\varepsilon_{\mathcal{X}_\mathrm{T}}(h_P),\quad \alpha=\frac{N_\mathrm{T}}{N_\mathrm{S}+N_\mathrm{T}}\\
&= \varepsilon^\alpha_{\mathcal{X}_{\mathrm{ST}}}(h_P)
\end{aligned}$$

The next lemma, adapted from *Lemma 2* of [2] , bounds the deviation between empirical and true combined risk:

LEMMA 4.2. *Suppose $\mathcal{H}$ has finite VC dimension D and that we have finite samples $\mathcal{X}_\mathrm{S}$ and $\mathcal{X}_\mathrm{T}$ of N_S and N_T source and target points, respectively. Then with probability $1-\delta$ we have*

$$|\varepsilon^\alpha_{\mathcal{X}_{\mathrm{ST}}}(h)-\varepsilon^\alpha_{\mathrm{ST}}(h)|\le\sqrt{\frac{D\log(\frac{2N}{\delta})}{2N}}$$

where $\alpha=\frac{N_\mathrm{T}}{N_\mathrm{S}+N_\mathrm{T}}$, $N=N_\mathrm{S}+N_\mathrm{T}$, and the expectation is over finite samples from the source and target distributions $\mathcal{D}_\mathrm{S}$ and $\mathcal{D}_\mathrm{T}$, respectively.

Proof. This is a straightforward application of *Lemma 2* from [2] , which says that

$$|\varepsilon^\alpha_{\mathcal{X}_{\mathrm{ST}}}(h)-\varepsilon^\alpha_{\mathrm{ST}}(h)|\le\sqrt{\frac{\alpha^2}{\gamma}+\frac{(1-\alpha)^2}{1-\gamma}}\sqrt{\frac{D\log(\frac{2N}{\delta})}{2N}}$$

where $\gamma=N_\mathrm{T}/N$. In our case, $\gamma=\alpha$.

This is a standard Hoeffding inequality bound, assuming finite VC dimension D. Ordinarily, 1NN classifiers have infinite VC dimension, but our hypothesis class is a subset of 1NN classifiers that can be created using prunings of our cluster tree T. There is only a finite number of such prunings. Thus, $\mathcal{H}$ is a finite hypothesis space with VC dimension $D\le\log_2|\mathcal{H}|$ [9].

Finally, we combine **Lemmas 4.1 and 4.2** with *Theorem 1* from [6] to prove **Theorem 4.1**. In the following proof, Steps 1 and 5 are by definition. Steps 2 and 4 apply **Lemma 4.2**, first to the target-only error (with N_T target points), then to the combined error. Because $N_\mathrm{T}\le N$, $\log(2N_\mathrm{T})/N_\mathrm{T}\ge\log(2N)/N$, so we use N_T instead of N so that we have only one square root term. The actual bound (using N) is tighter. Step 3 applies *Lemma 1* from [2] to upper bound the combined error. Step 6 applies *Theorem 1* from [6] to place an upper bound on the label imputation error of P, assuming that we have queried bt labels.

Proof.

$$\begin{aligned}
\epsilon_\mathrm{T}(P) &= \varepsilon_{\mathcal{X}_\mathrm{T}}(h_P)\\
&\le \varepsilon_\mathrm{T}(h_P)+\sqrt{\frac{D\log(2N_\mathrm{T})-\log\delta}{2N_\mathrm{T}}}\\
&\le \varepsilon^\alpha_{\mathrm{ST}}(h_P)+(1-\alpha)\left(\frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_\mathrm{S},\mathcal{D}_\mathrm{T})+\varepsilon^*_{\mathrm{ST}}\right)\\
&\qquad+\sqrt{\frac{D\log(2N_\mathrm{T})-\log\delta}{2N_\mathrm{T}}}\\
&\le \varepsilon^\alpha_{\mathcal{X}_{\mathrm{ST}}}(h_P)+(1-\alpha)\left(\frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_\mathrm{S},\mathcal{D}_\mathrm{T})+\varepsilon^*_{\mathrm{ST}}\right)\\
&\qquad+2\sqrt{\frac{D\log(2N_\mathrm{T})-\log\delta}{2N_\mathrm{T}}}\\
&= \epsilon(P)+(1-\alpha)\left(\frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_\mathrm{S},\mathcal{D}_\mathrm{T})+\varepsilon^*_{\mathrm{ST}}\right)\\
&\qquad+2\sqrt{\frac{D\log(2N_\mathrm{T})-\log\delta}{2N_\mathrm{T}}}\\
&\le \widetilde{O}(\epsilon(P^*)+\eta)\\
&\qquad+(1-\alpha)\left(\frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_\mathrm{S},\mathcal{D}_\mathrm{T})+\varepsilon^*_{\mathrm{ST}}\right)\\
&\qquad+2\sqrt{\frac{D\log(2N_\mathrm{T})-\log\delta}{2N_\mathrm{T}}}
\end{aligned}$$

4.2 Discussion of Theorem 4.1: This theorem represents a useful guide to understanding and applying HATL. We can expect it to perform best when the source and target distributions are similar and when we have relatively large data sets. However, we should be careful when interpreting its guarantees, as it makes strong assumptions. In particular, it assumes that clusters are selected for querying with probability proportional to size (similar to *Theorem 1* from [6]). In transfer learning, we typically begin with m "free" labeled source points. To apply **Theorem 4.1**, we must assume that these, too, were sampled at random from a larger pool of $N_\mathrm{S}\propto m/(bt-m)$ source points. This value will change with each iteration of HATL, but the source data set is of fixed size. In practice, however, these assumptions should have minimal impact.

5 Experimental Results

Now we present experimental results that confirm our intuition and analysis and deepen our understanding of HATL. All code and data for these experiments (including implementations of HSAL [6], TIWCAL [10], and JOTAL [4]), may be found at `http://www-bcf.usc.edu/~liu32/code.html`.

5.1 Synthetic data: We design a series of one-dimensional synthetic data sets to examine the behavior of HATL and compare its performance with HSAL under different conditions. We create a single target domain and five source domains ($S1 - S5$) of varying similarity to the target domain, as visualized in **Figure 1a**. The target data fall into four distinct clusters of equal size and extent, while each source domain has between two and four clusters of different sizes and at different locations. All domains share the same labeling function: we divide the real line into four regions with alternating labels. For our experiments, we sample 25 data sets with 200 points each from each domain.

Measuring domain similarity: We approximate the $d_{\mathcal{H}\Delta\mathcal{H}}$ distance (denoted $\hat{d}_{\mathcal{H}\Delta\mathcal{H}}$) using a *domain separator hypothesis* [15], i.e., a classifier trained to distinguish source from target points. The accuracy of this classifier is proportional to the distance between distributions: the more separable the classes, the more different their distributions. We use a 1NN classifier (the type of classifier induced by HATL) as our domain separator and its mean accuracy (estimated via 10-fold cross validation) as the $\hat{d}_{\mathcal{H}\Delta\mathcal{H}}$ distance.

Label imputation error: Figure 1b shows target label imputation error versus target label queries for HSAL and HATL using $S1$, $S3$, and $S5$. We report mean performance (with 95% confidence intervals) across all 25 data samples from each domain. HATL provides substantial benefit over HSAL, particularly with few queries. It takes HSAL over 20 queries to reach a mean error of 0.1. Using $S1$, HATL reaches 0.1 with 15 queries (a 25% reduction) and dominates HSAL through all 50 queries. Using $S2$ HATL outperforms HSAL through the first 15 queries and is competitive thereafter. $S5$ provides an early benefit but increases HATL's error in the long run. This is not surprising: $S5$ has the highest $\hat{d}_{\mathcal{H}\Delta\mathcal{H}}$ distance and minimal cluster overlap. This is an example of a persistent negative bias leading to a plateau, as discussed in **Section 3.3**.

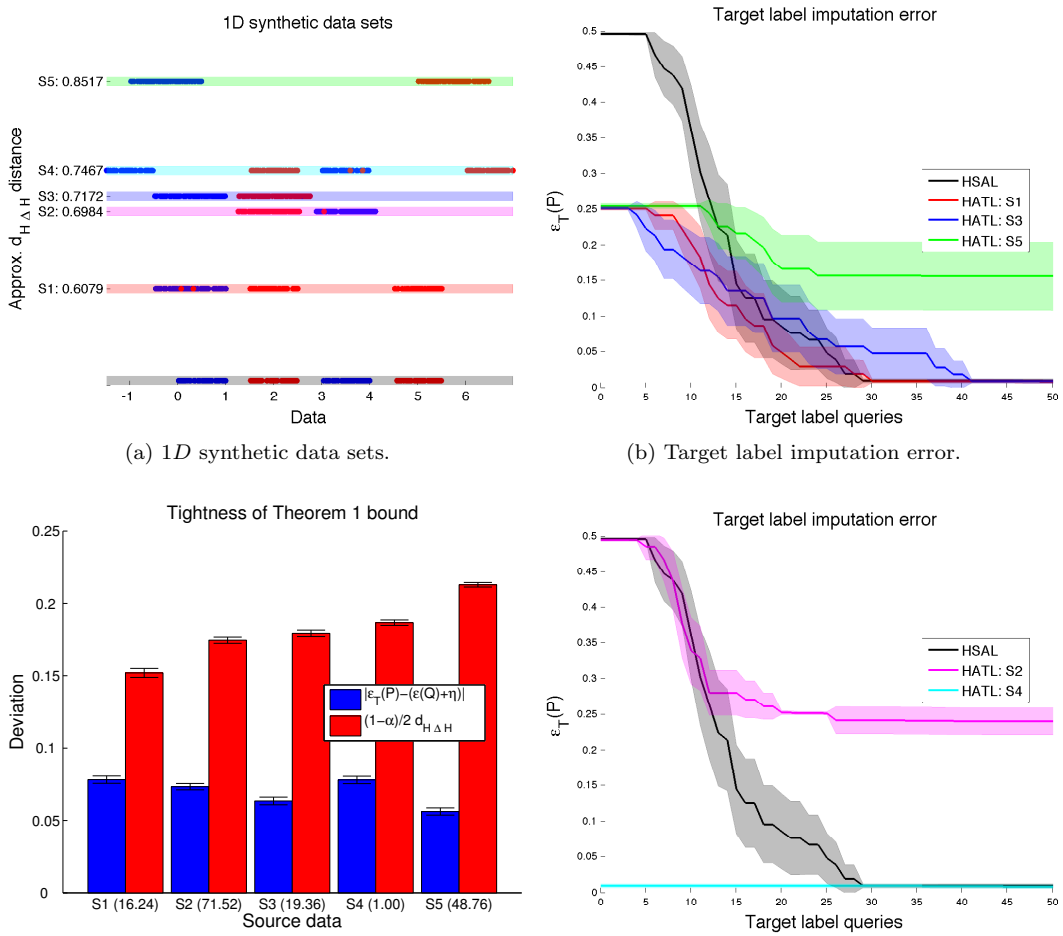
Tightness of Theorem 4.1: We are also interested in exploring the tightness and utility of the bound in **Theorem 4.1**. While it is intuitive and delivers insight into the strengths and limitations of HATL, such bounds are known to be loose in practice. We can rewrite it as bounding the deviation between the target and optimal overall label imputation errors:

$$\left|\epsilon_{\mathrm{T}}(P) - \widetilde{O}(\epsilon(P^*) + \eta)\right| \leq \frac{1-\alpha}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{\mathrm{S}}, \mathcal{D}_{\mathrm{T}})$$

We have dropped the $\varepsilon^*_{\mathrm{ST}}$ and square root terms from the theoretical bound. The former will be negligibly small since we labeled all synthetic data sets using the same labeling function. The latter is large because the synthetic data sets are very small (for $N_{\mathrm{T}} = 200$, $2\sqrt{\log(2N_{\mathrm{T}})/(2N_{\mathrm{T}})} > 0.25$) and will overwhelm the domain similarity term. The remaining righthand term is the weighted similarity between source and target domains, which we can again approximate using the 1NN domain separator classifier. We will subsequently denote this as $\hat{d}_{\mathcal{H}\Delta\mathcal{H}}$. We estimate the lefthand term (the deviation) as follows: first, we set the label imputation error threshold $\eta = 0.05$ and perform a greedy breadth-first search for a pruning P^* with error $\epsilon(P^*) \leq \eta$. Then we run HATL and identify the first iteration where $\epsilon(P) \leq \epsilon(P^*) + \eta$. We assume that we have enough queries to satisfy the assumptions of **Theorem 4.1**. Finally, we compute the target-only error $\epsilon_{\mathrm{T}}(P)$ and the deviation term. **Figure 1c** compares the average deviation with the average $\hat{d}_{\mathcal{H}\Delta\mathcal{H}}$ distance for each source data set. The actual deviation is nearly constant across data sets, between 0.06 and 0.07. The deviation predicted by $\hat{d}_{\mathcal{H}\Delta\mathcal{H}}$ grows faster than the actual deviation.

This is surprising because **Theorem 4.1** suggest that HATL's performance should improve as $d_{\mathcal{H}\Delta\mathcal{H}}$ distance decreases. We can derive some insight into why this might not be the case if we consider $S2$ vs. $S4$ (**Figure 1d**). $S2$ clearly hurts the performance of HATL; its initial error is as bad as HSAL, and it plateaus very early. $S4$ is nearly perfect from the beginning. This is because $S4$ has points lying in all four regions of the true labeling function and provides a useful initial bias for HATL. $S2$ has points concentrated only in the center two regions. However, $S2$ has a lower $\hat{d}_{\mathcal{H}\Delta\mathcal{H}}$ distance than $S4$. This suggests that while $d_{\mathcal{H}\Delta\mathcal{H}}$ is a principled and often useful measure of domain similarity, it has important limitations. In particular, it ignores the sometimes complex interaction between the marginal data and conditional label distributions.

5.2 Sentiment classification: Next, we perform experiments using version 2.0 of the Multi-Domain Sentiment Classification (`sentiment`) data set [3] of Amazon product reviews. Each product category is treated as a domain, and the task is to predict the sentiment (positive or negative) of an individual review based on its contents. In version 2.0, each category contains 1000 positive and 1000 negative documents, represented as sparse vector of word and bigram counts from a vocabulary of well over a million unique entries. We reduced



(a) $1D$ synthetic data sets.

(b) Target label imputation error.

(c) Tightness of the **Theorem 4.1** bound.

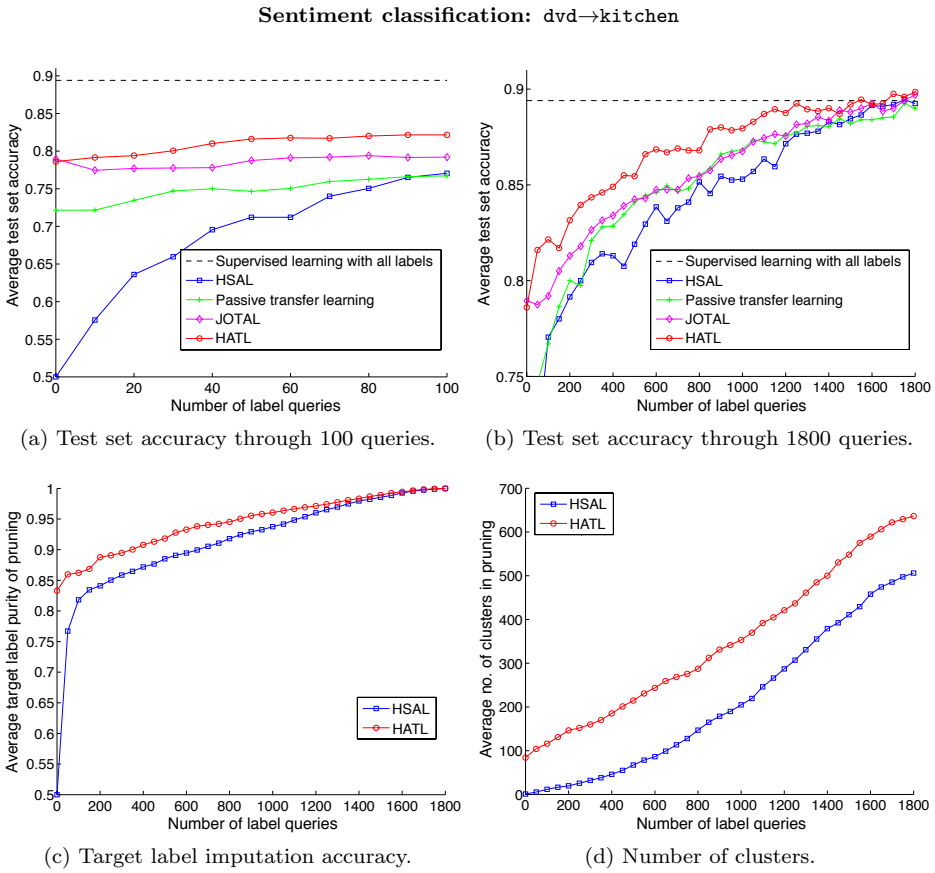
(d) A higher $\hat{d}_{\mathcal{H}\Delta\mathcal{H}}$ outperforms a lower $\hat{d}_{\mathcal{H}\Delta\mathcal{H}}$.

Figure 1: $1D$ synthetic data experiments. (a) Sources organized vertically by $\hat{d}_{\mathcal{H}\Delta\mathcal{H}}$. (b,d) Mean target label imputation error for (b) $S1$, $S3$, and $S5$ and (d) $S2$ and $S4$. Shading indicates 95% confidence intervals. (c) Actual vs. predicted absolute deviation between P's target label error and P^*'s overall label error (average number of queries in parentheses).

the number of features to just over 2000 unigrams and converted raw counts to log counts. We used 10-fold cross-validation to estimate the prediction error on the test set. For each experiment, we choose one category as the target domain. We start with zero target data labels and run active learning until all labels have been queried. For the active transfer learning algorithms, we use a second category as our source data, choosing 200 positive and 200 negative examples at random to use as the labeled sample. HATL also uses the remaining 1600 unlabeled source data points in its clustering but does not have access to the true labels.

We compare HATL's performance against standard baselines and competing state-of-the-art algorithms, including HSAL, *passive transfer learning* (train on source plus labeled target points chosen at random), and *joint transfer and batch-mode active learning* (JOTAL)[4]. The reported accuracies use a linear classifier with squared hinge loss and L_2 regularization (from `LIBLINEAR` [7]). For HSAL and HATL, the classifier is trained on all training data (including source for HATL) with imputed labels. Other algorithms are trained only on data points for which labels are available or queried.

Figure 2 shows results using `kitchen` as target and `dvd` as source. These are among the least similar categories [3]. The *baseline supervised* (train on labeled target data) accuracy for `kitchen` is 0.8925 ± 0.0121. The baseline *unsupervised transfer learning* (train on *all* `dvd` data) accuracy is 0.7850 ± 0.0283. HATL outperforms the competing approaches throughout the entire



(a) Test set accuracy through 100 queries.

(b) Test set accuracy through 1800 queries.

(c) Target label imputation accuracy.

(d) Number of clusters.

Figure 2: `dvd`-to-`kitchen` sentiment classification results. Mean test set accuracy through (a) 100 and (b) the 1800 target label queries. (c) Mean target label imputation accuracy of P. (d) Mean size of P.

label query budget (**Figure 2a** and **Figure 2b**). The initial test set accuracy of both HATL and JOTAL is equivalent to that of the unsupervised transfer learning baseline and 7 points better than the passive transfer learner. This suggests that HATL and JOTAL can both perform unsupervised transfer learning effectively, using very different approaches. **Figure 2c** shows that HATL can also perform semi-supervised learning; with even a small number of queries, the accuracy for target points in the clustering is higher than for the test set.

HATL significantly outperforms HSAL. With zero target labels, HATL is 30 points better than HSAL in both label imputation accuracy and test set accuracy. HSAL improves more rapidly over the first few hundred label queries, but HATL continues to outperform HSAL by at least a few points in accuracy until the very end. Further, HATL converges to the fully labeled supervised learning performance, just as HSAL does. This indicates that it avoids negative transfer, successfully adapting the initial source-based transfer bias based on the target label feedback that it receives.

6 Discussion and Conclusion

We have described a unified framework for active transfer learning, called *Hierarchical Active Transfer Learning* (HATL), which uses the machinery of the well-known, clustering-based *Hierarchical Sampling for Active Learning* (HSAL) [6]. HATL exploits source and target domain similarity in the form of shared cluster structure to impute labels for unlabeled target data and gradually adapts this transfer as it active queries target labels. We formalized the intuition behind HATL by deriving an upper bound on the target label imputation error that includes terms related to the imputation error of an optimal pruning and the similarity and respective sizes of the source and target data sets. We used synthetic data experiments to deepen our understanding of the error bound and of when and how HATL works. On a benchmark domain adaptation data set [3], HATL outperformed its progenitor algorithm, HSAL, and competed with a state-of-the-art active transfer learning framework [4]. What is more, our results suggest that this flexible framework can also be used to perform un-

supervised transfer learning and semi-supervised learning. For future work, we are exploring ways to enable HATL to adapt more rapidly to target label queries. One possibility is to reweight source data (based on cluster proportions) whenever the pruning changes. We are also experimenting with alternative measures (e.g., two-sample tests) of the within-cluster label agreement between source and target points.

Acknowledgements

The research was partially supported by NSF research grants IIS-1134990 and IIS-1254206; by the U.S. Defense Advanced Research Projects Agency (DARPA) under the Social Media in Strategic Communication (SMISC) program, Agreement Number W911NF-12-1-0034; and by an IBM Faculty Award. David Kale is supported by the Alfred E. Mann Innovation in Engineering Doctoral Fellowship. The views and conclusions are those of the authors and should not be interpreted as representing the official policies of the funding agencies or the U.S. Government. We also would like to thank Sanjoy Dasgupta and our anonymous reviewers for their helpful feedback.

References

[1] A. Beygelzimer, J. Langford, D. Hsu, and T. Zhang. Agnostic Active Learning Without Constraints. In *Advances in Neural Information Processing Systems (NIPS) 23*, pages 199–207. 2011.

[2] J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Wortman. Learning bounds for domain adaptation. In *Advances in Neural Information Processing Systems (NIPS) 20*, pages 129–136, 2007.

[3] J. Blitzer, M. Dredze, and F. Pereira. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2007.

[4] R. Chattopadhyay, W. Fan, I. Davidson, S. Panchanathan, and J. Ye. Joint transfer and batch-mode active learning. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 253–261, 2013.

[5] S. Dasgupta. Two Faces of Active Learning. *Theoretical Computer Science*, 412(19):1767–1781, 2011.

[6] S. Dasgupta and D. Hsu. Hierarchical sampling for active learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 208–215, 2008.

[7] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[8] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola. A kernel method for the two-sample-problem. In *Advances in Neural Information Processing Systems (NIPS) 19*, pages 513–520. 2007.

[9] D. Haussler. Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence*, 36(2):177–221, September 1988.

[10] D. Kale and Y. Liu. Accelerating active learning with transfer learning. In *Proceedings of the IEEE 13th International Conference on Data Mining (ICDM)*, 2013.

[11] D. Kifer, S. Ben-David, and J. Gehrke. Detecting change in data streams. In *Proceedings of the 30th International Conference on Very Large Databases (VLDB)*, pages 180–191. VLDB Endowment, 2004.

[12] A. Kumar, A. Saha, and H. Daumé III. A co-regularization based semi-supervised domain adaptation. In *Advances in Neural Information Processing Systems (NIPS) 23*, 2010.

[13] C. Luo, Y. Ji, X. Dai, and J. Chen. Active learning with transfer learning. In *Proceedings of ACL 2012 Student Research Workshop*, pages 13–18, 2012.

[14] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, Oct 2010.

[15] P. Rai, A. Saha, H. Daumé, and S. Venkatasubramanian. Domain adaptation meets active learning. In *Proceedings of the NAACL HLT 2010 Workshop on Active Learning for Natural Language Processing*, pages 27–32, 2010.

[16] R. Raina, A. Battle, H. Lee, B. Packer, and A. Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 759–766, 2007.

[17] B. Settles. *Active Learning*, volume 6 of *Synthesis Lectures on Artificial Intelligence and Machine Learning Series*. Morgan & Claypool Publishers, June 2012.

[18] L. Yang, S. Hanneke, and J. Carbonell. Identifiability of priors from bounded sample sizes with applications to transfer learning. pages 791–808, 2011.

[19] L. Yang, S. Hanneke, and J. Carbonell. A Theory of Transfer Learning with Applications to Active Learning. *Machine Learning*, 90(2):1–28, 2012.

7 Appendix

7.1 HSAL: Algorithm 2 shows original HSAL algorithm, adapted from [6, Algorithm 1], which we use as a subroutine in HATL. Below we provide a high-level description of how it works and of notation. We direct readers interested in further detail to [6].

Algorithm 2 Hierarchical Sampling for Active Learning (HSAL) [6]

1: **Input:** Hierarchical cluster tree T of labeled and unlabeled data $\mathcal{X}$, including optional instance weights; initial pruning P and labeling L; active learning label budget B, batch size b, and *oracle*
2: $q \longleftarrow 0$ // number of label queries made so far
3: **repeat**
4: $Q \leftarrow \{\}$ // list of queried nodes
5: **for** $i = 1$ **to** b **do**
6: $(v, x) \longleftarrow$ *ChooseNextQuery*(P)
7: $y \longleftarrow$ *oracle*(x), $Q \longleftarrow Q \cup \{v\}$
8: **for** $u \in T$ where $x \in X(u)$ **do**
9: *UpdateNodeStatistics*(x, y, u) // counts, estimates and bounds, admissibility, score
10: **end for**
11: **end for**
12: **for each** $v \in Q$ **do**
13: $(P_v, L_v) \leftarrow$ *GetPruningAndLabeling*(T_v)
14: $P \leftarrow (P \setminus \{v\}) \cup P_v$ // replace v with $u \in P_v$
15: $L(u) \longleftarrow L_v(u)$ **for each** $u \in P_v$
16: **end for**
17: **until** $q \geq B$ or no unlabeled data remain
18: **for each** $v \in P$ **do**
19: $\hat{y}(x) \longleftarrow L(v)$ **for each** $x \in X(v)$
20: **end for**

The algorithm takes as its input a hierarchical clustering (in the form of a tree T) over all data points, optionally including some labels (e.g., when called from HATL). A tree T_v contains a hierarchy of nodes rooted at node v (where $T = T_{\text{root}}$). If we descend far enough down T_v, we reach a leaf node x, which is in fact a data point. Denote the set of data points associated with an arbitrary node v as $X(v) = \bigcup_{u:\text{ch}(v)} X(u)$, i.e., v is associated with all of its children's points. Trivially, $X(x) = x$. A pruning P_v of tree T_v is a subset of v's descendant nodes containing all points associated with v: $X(v) = \cup_{u:\text{desc}(v)} X(u)$. P is the pruning of the complete tree T. A labeling $L(v)$ is a mapping of a node v to a label. We typically work with a pruning/labeling pair (P, L). The algorithm begins with an initial pruning $P = \{\text{root}\}$ and labeling $L(\text{root}) = +1$, i.e., our pruning is just the root of T and we assign all data points the same arbitrary label.

The active learning loop is performed in **lines 3-17**, where we iteratively query a batch of b labels and update our node statistics, pruning P, and labeling L, until our budget B is spent or no unlabeled target points remain. We choose query t using *ChooseNextQuery* (**line 11**). Typically, we first select a node v from current pruning P and then choose a unlabeled point $x \in X(v)$ at random. The node selection procedure determines the nature of our active learning strategy. [6] recommends choosing v with probability proportional to the product of its size and estimated label purity ($p(v) \propto w_v(1 - p^{\text{LB}}_{v,L(y)})$ using notation defined below). We maintain a list Q of nodes from which we have queried labels.

Now suppose for our query t, we receive label y for point $x \in X(v)$ and node $v \in P$. In **lines 8-10**, we perform a bottom-up pass of updates to all nodes u such that $x \in X(u)$ (i.e., nodes along the path from x to v). We first add one to the number of observations of label y for u and compute an estimate of and lower and upper bounds on the expected fraction of points with label y in $X(u)$ (denoted $p_{u,y}(t)$, $p^{\text{LB}}_{u,y}(t)$, and $p^{\text{UB}}_{u,y}(t)$). We then determine which labels are *admissible* for u; an admissible label is one that we are willing to assign to all points $x \in X(u)$ because its label fraction bounds meet some criterion. For example, with two labels, we might require that $p^{l}_{u,y}(t) > 1/3$ for y to be admissible. Note that admissibility persists (once y is admissible for u, it remains so). Next we choose u's best label $y_u(t)$, if any, from among all admissible labels and then determine its score $s_u(t) = \min\{s'_u(t), w_\ell/w_u s_l(t) + w_r/w_u s_r(t)\}$, i.e., the minimum of $s'_u(t)$ and a weighted sum of the scores of its left and right children ℓ, r (a node v's weight is proportional to its size $w_v \propto |X(v)|$). Here $s_u(t) = 1 - p^{\text{LB}}_{u,y}$ if $y_u(t)$ exists and $s = 1$ otherwise (i.e., the expected error if we assign u its best label, if any). This means that $s_u(t)$ is determined by whether we expect better labeling from assigning u its best label or from splitting u into its children; in the latter case, we mark this node for splitting.

After a batch of b label queries and updates, we update our pruning P and labeling function L (**lines 12-16**). For each queried node v, we examine whether it should be split. If not, we keep it; if so, we replace it with its children and then recurse down the tree. In this way we determine the optimal pruning P_v and labeling L_v of T_v, which we then use to update P and L. Once we have exhausted our query budget, we iterate over nodes v in our pruning and assign label $L(v)$ as the label to all points $x \in X(v)$ (**lines 18-20**).