# Approximating a Distribution Using Weight Queries

**Nadav Barak** [1] **Sivan Sabato** [1]

## Abstract

We consider a novel challenge: approximating a distribution without the ability to randomly sample from that distribution. We study how such an approximation can be obtained using *weight queries*. Given some data set of examples, a weight query presents one of the examples to an oracle, which returns the probability, according to the target distribution, of observing examples similar to the presented example. This oracle can represent, for instance, counting queries to a database of the target population, or an interface to a search engine which returns the number of results that match a given search.

We propose an interactive algorithm that iteratively selects data set examples and performs corresponding weight queries. The algorithm finds a reweighting of the data set that approximates the weights according to the target distribution, using a limited number of weight queries. We derive an approximation bound on the total variation distance between the reweighting found by the algorithm and the best achievable reweighting. Our algorithm takes inspiration from the UCB approach common in multi-armed bandits problems, and combines it with a new discrepancy estimator and a greedy iterative procedure. In addition to our theoretical guarantees, we demonstrate in experiments the advantages of the proposed algorithm over several baselines. A python implementation of the proposed algorithm and of all the experiments can be found at `https://github.com/Nadav-Barak/AWP`

## 1. Introduction

A basic assumption in learning and estimation tasks is the availability of a random sample from the distribution of interest. However, in many cases, obtaining such a random sample is difficult or impossible. In this work, we study a novel challenge: approximating a distribution without the ability to randomly sample from it. We consider a scenario in which the only access to the distribution is via *weight queries*. Given some data set of examples, a weight query presents one of these examples to an oracle, which returns the probability, according to the target distribution, of observing examples which are similar to the presented example. For instance, the available data set may list patients in a clinical trial, and the target distribution may represent the population of patients in a specific hospital, which is only accessible through certain database queries. In this case, the weight query for a specific data set example can be answered using a database counting query, which indicates how many records with the same demographic properties as the presented example exist in the database. A different example of a relevant oracle is that of a search engine for images or documents, which returns the number of objects in its database that are similar to the searched object.

We study the possibility of using weight queries to find a reweighting of the input data set that approximates the target distribution. Importantly, we make no assumptions on the relationship between the data set and the target distribution. For instance, the data set could be sampled from a different distribution, or be collected via a non-random process. Reweighting the data set to match the true target weights would be easy if one simply queried the target weight of all the data set examples. In contrast, our goal in this work is to study whether a good approximated weighting can be found using a number of weight queries that is independent of the data set size. A data set reweighted to closely match a distribution is often used as a proxy to a random sample in learning and statistical estimation tasks, as done, for instance, in domain adaptation settings (e.g., Bickel et al. 2007; Sugiyama et al. 2008; Bickel et al. 2009).

We consider a reweighting scheme in which the weights are fully defined by some partition of the data set to a small number of subsets. Given the partition, the weights of examples in each subset of the partition are uniform, and are set so that the total weight of the subset is equal to its true target weight. An algorithm for finding a good reweighting should thus search for a partition such that within each of its subsets, the true example weights are as close to uniform as

[1]Department of Computer Science, Ben Gurion University, Israel. Correspondence to: Nadav Barak <barakna@post.bgu.ac.il>, Sivan Sabato <sabatos@cs.bgu.ac.il>.

*Proceedings of the 38th International Conference on Machine Learning*, PMLR 139, 2021. Copyright 2021 by the author(s).

possible. For instance, in the hospital example, consider a case in which the hospital of interest has more older patients than their proportion in the clinical trial data set, but within each of the old and young populations, the makeup of the patients in the hospital is similar to that in the clinical trial. In this case, a partition based on patient ages will lead to an accurate reweighting of the data set. The goal of the algorithm is thus to find a partition which leads to an accurate reweighting, using a small number of weight queries.

We show that identifying a good partition using only weight queries of individual examples is impossible, unless almost all examples are queried for their weight. We thus consider a setting in which the algorithm has a limited access to higher order queries, which return the total weight of a subset of the data set. These higher-order queries are limited to certain types of subsets, and the algorithm can only use a limited number of such queries. For instance, in the hospital example, higher-order queries may correspond to database counting queries for some less specific demographic criteria.

To represent the available higher order queries for a given problem, we assume that a hierarchical organization of the input data set is provided to the algorithm in the form of a tree, whose leaves are the data set examples. Each internal node in the tree represents the subset of leaves that are its descendants, and the only allowed higher-order queries are those that correspond to one of the internal nodes. Given such a tree, we consider only partitions that are represented by some *pruning* of the tree, where a pruning is a set of internal nodes such that each leaf has exactly one ancestor in the set. A useful tree is one that includes a small pruning that induces a partition of the data set into near-uniform subsets, as described above.

**Main results:** We give an algorithm that for any given tree, finds a near-optimal pruning of size $K$ using only $K-1$ higher order queries and $O(K^3/\epsilon^2)$ weight queries of individual examples, where $\epsilon$ measures the total variation distance of the obtained reweighting. The algorithm greedily splits internal nodes until reaching a pruning of the requested size. To decide which nodes to split, it iteratively makes weight queries of examples based on an Upper Confidence Bound (UCB) approach. Our UCB scheme employs a new estimator that we propose for the quality of an internal node. We show that this is necessary, since a naive estimator would require querying the weight of almost all data set examples. Our guarantees depend on a property of the input hierarchical tree that we term the *split quality*, which essentially requires that local node splits of the input tree are not too harmful. We show that any algorithm that is based on iteratively splitting the tree and obtains a non-trivial approximation guarantee requires some assumption on the quality of the tree.

To supplement our theoretical analysis, we also implement the proposed algorithm and report several experiments, which demonstrate its advantage over several natural baselines. A python implementation of the proposed algorithm and of all the experiments can be found at `https://github.com/Nadav-Barak/AWP`.

**Related work**

In classical density estimation, the goal is to estimate the density function of a random variable given observed data (Silverman, 1986). Commonly used methods are based on Parzen or Kernel estimators (Wand and Jones, 1994; Goldberger and Roweis, 2005), expectation maximization (EM) algorithms (McLachlan and Krishnan, 2007; Figueiredo and Jain, 2002) or variational estimation (Corduneanu and Bishop, 2001; McGrory and Titterington, 2007). Some works have studied active variants of density estimation. In Ghasemi et al. (2011), examples are selected for kernel density estimation. In Kristan et al. (2010), density estimation in an online and interactive setting is studied. We are not aware of previous works that consider estimating a distribution using weight queries. The domain adaptation framework (Kifer et al., 2004; Ben-David et al., 2007; Blitzer et al., 2008; Mansour et al., 2009; Ben-David et al., 2010) assumes a target distribution with scarce or unlabeled random examples. In Bickel et al. (2007); Sugiyama et al. (2008); Bickel et al. (2009), reweighting the labeled source sample based on unlabeled target examples is studied. Trade-offs between source and target labels are studied in Kpotufe and Martinet (2018). In Berlind and Urner (2015), a labeled source sample guides target label requests.

In this work, we assume that the input data set is organized in a hierarchical tree, which represents relevant structures in the data set. This type of input is common to many algorithms that require structure. For instance, such an input tree is used for active learning in Dasgupta and Hsu (2008); Urner et al. (2013). In Slivkins (2011); Bubeck et al. (2011), a hierarchical tree is used to organize different arms in a multi-armed-bandits problem, and in Munos (2011) such a structure is used to adaptively estimate the maximum of an unknown function. In Kpotufe et al. (2015); Cortes et al. (2020), an iterative partition of the domain is used for active learning.

## 2. Setting and Notations

Denote by $\mathbf{1}$ the all-1 vector; its size will always be clear from context. For an integer $n$, let $[n] := \{1, \ldots, n\}$. For a vector or a sequence $\mathbf{x} = (x(1), \ldots, x(n))$, let $\|\mathbf{x}\|_1 := \sum_{i\in[n]} |x(i)|$ be the $\ell_1$ norm of $\mathbf{x}$. For a function $f : \mathcal{X} \to \mathbb{R}$ on a discrete domain $\mathcal{X}$, denote $\|f\|_1 := \sum_{x\in\mathcal{X}} |f(x)|$.

The input data set is some finite set $S \subseteq \mathcal{X}$. We assume that the examples in $S$ induce a partition on the domain $\mathcal{X}$, where the part represented by $x \in S$ is the set of target examples that are similar to it (in some application-specific sense). The target weighting of $S$ is denoted $w^* : S \to [0, 1]$, where $\|w^*\|_1 = 1$. $w^*(x)$ is the probability mass, according to the unknown target distribution, of the examples in the domain that are represented by $x$. For a set $V \subseteq S$, denote $w^*(V) := \sum_{x \in V} w^*(x)$. The goal of the algorithm is to approximate the target weighting $w^*$ using weight queries, via a partition of $S$ into at most $K$ parts, where $K$ is provided as input to the algorithm.

A *basic weight query* presents some $x \in S$ to the oracle and receives its weight $w^*(x)$ as an answer. To define the available higher-order queries, the input to the algorithm includes a binary hierarchical tree $T$ whose leaves are the elements in $S$. For an internal node $v$, denote by $\mathcal{L}_v \subseteq S$ the set of examples in the leaves descending from $v$. A *higher-order query* presents some internal node $v$ to the oracle, and receives its weight $w^*(\mathcal{L}_v)$ as an answer. We note that the algorithm that we propose below uses higher-order queries only for nodes of depth at most $K$.

A reweighting algorithm attempts to approximate the target weighting by finding a small *pruning* of the tree that induces a weighting on $S$ which is as similar as possible to $w^*$. Formally, for a given tree, a pruning of the tree is a set of internal nodes such that each tree leaf is a descendant of exactly one of these nodes. Thus, a pruning of the input tree $T$ induces a partition on $S$. The weighting induced by the pruning is defined so that the weights assigned to all the leaves (examples) descending from the same pruning node $v$ are the same, and their total weight is equal to the true total weight $w^*(\mathcal{L}_v)$. Formally, let $N_v := |\mathcal{L}_v|$. For a pruning $P$ and an example $x \in S$, let $\mathsf{A}(P, x)$ be the node in $P$ which is the ancestor of $x$. The weighting $w_P : S \to \mathbb{R}^+$ induced by the pruning $P$ is defined as $w_P(x) := w^*_{\mathsf{A}(P,x)}/N_{\mathsf{A}(P,x)}$.

The quality of a weighting $w_P$ is measured by the total variation distance between the distribution induced by $w_P$ and the one induced by $w^*$. This is equivalent to the $\ell_1$ norm between the weight functions (see, e.g., Wilmer et al., 2009). Formally, the total variation distance between two weight functions $w_1, w_2 : S \to \mathbb{R}^+$ is

$$\begin{aligned}\mathrm{dist}(w_1, w_2) &:= \max_{S' \subseteq S} |w_1(S') - w_2(S')| \\ &\equiv \frac{1}{2} \sum_{x \in S} |w_1(x) - w_2(x)| \equiv \frac{1}{2} \|w_1 - w_2\|_1.\end{aligned}$$

For a node $v$, define $\mathbb{D}_v := \sum_{x \in \mathcal{L}_v} |w^*_v/N_v - w^*(x)|$. The *discrepancy* of $w_P$ (with respect to $w^*$) is

$$\mathbb{D}_P := 2\mathrm{dist}(w_P, w^*) = \|w_P - w^*\|_1 = \sum_{v \in P} \mathbb{D}_v.$$

Intuitively, this measures the distance of the weights in each pruning node from uniform weights. More generally, for any subset $G$ of a pruning, define $\mathbb{D}_G := \sum_{v \in G} \mathbb{D}_v$. We also call $\mathbb{D}_G, \mathbb{D}_v$ the discrepancy of $G$, $v$, respectively. The goal of the algorithm is thus to find a pruning $P$ with a low discrepancy $w_P$, using a small number of weight queries.

## 3. Estimating the Discrepancy

As stated above, the goal of the algorithm is to find a pruning with a low discrepancy. A necessary tool for such an algorithm is the ability to estimate the discrepancy $\mathbb{D}_v$ of a given internal node using a small number of weight queries. In this section, we discuss some challenges in estimating the discrepancy and present an estimator that overcomes them.

First, it can be observed that the discrepancy of a node cannot be reliably estimated from basic weight queries alone, unless almost all leaf weights are queried. To see this, consider two cases: one in which all the leaves in $\mathcal{L}_v$ have an equally small weight, and one in which this holds for all but one leaf, which has a large weight. The discrepancy $\mathbb{D}_v$ in the first case is zero, while it is large in the second case. However, it is impossible to distinguish between the cases using basic weight queries, unless they happen to include the heavy leaf. A detailed example of this issue is provided in Appendix B.

To overcome this issue, the algorithm uses a higher order query to obtain the total weight of the internal node $w^*_v$, in addition to a random sample of basic weight queries of examples in $\mathcal{L}_v$. However, even then, a standard empirical estimator of the discrepancy, obtained by aggregating $|w^*_v/N_v - w^*(x)|$ over sampled examples, can have a large estimation error due to the wide range of possible values (see example in Appendix B). We thus propose a different estimator, which circumvents this issue. The lemma below gives this estimator and proves a concentration bound for it. In this lemma, the leaf weights are represented by $\mathbf{w} = (w_1, \ldots, w_n)$ and the discrepancy of the node is $\mathbb{D}(\mathbf{w})$. In more general terms, this lemma gives an estimator for the uniformity of a set of values.

**Lemma 3.1.** *Let $\mathbf{w} = (w_1, \ldots, w_n)$ be a sequence of non-negative real values with a known $\|\mathbf{w}\|_1$. Let $W := \|\mathbf{w}\|/n$, and $\mathbb{D}(\mathbf{w}) := \|\mathbf{w} - W \cdot \mathbf{1}\|_1$. Let $U$ be a uniform distribution over the indices in $[n]$, and suppose that $m$ i.i.d. samples $\{I_i\}_{i \in [m]}$ are drawn from $U$. Denote $Z_i := w_{I_i}$ for $i \in [m]$, and $\mathbf{Z} := (Z_1, \ldots, Z_m)$. Let the estimator for $\mathbb{D}(\mathbf{w})$ be:*

$$\hat{\mathbb{D}}(\mathbf{w}) := \|\mathbf{w}\|_1 + \frac{n}{m}(\|\mathbf{Z} - W \cdot \mathbf{1}\|_1 - \|\mathbf{Z}\|_1).$$

*Then, with a probability at least $1 - \delta$,*

$$|\mathbb{D}(\mathbf{w}) - \hat{\mathbb{D}}(\mathbf{w})| \le \|\mathbf{w}\|_1 \sqrt{2 \ln(2/\delta)/m}.$$

*Proof.* Let $Z'_i = |Z_i - W| - Z_i$. If $Z_i \ge W$, we have

$Z'_i = -W$. Otherwise, $0 \leq Z_i < W$, and therefore $Z'_i = W - 2Z_i \in [-W, W]$.

Combining the two cases, we get $\mathbb{P}[Z'_i \in [-W, W]] = 1$. Thus, by Hoeffding's inequality, with a probability at least $1 - \delta$, we have

$$|\mathbb{E}[Z'_1] - \frac{1}{m} \sum_{i \in [m]} Z'_i| \leq W \cdot \sqrt{2 \ln(2/\delta)/m}, \text{ and}$$

$$\mathbb{E}[Z'_1] = \frac{1}{n}(\|\mathbf{w} - W \cdot \mathbf{1}\|_1 - \|\mathbf{w}\|_1) = \frac{1}{n}(\mathbb{D}(\mathbf{w}) - \|\mathbf{w}\|_1).$$

In addition,

$$\frac{1}{m} \sum_{i \in [m]} Z'_i = \frac{1}{m}(\|\mathbf{Z} - W \cdot \mathbf{1}\|_1 - \|\mathbf{Z}\|_1) = \frac{1}{n}(\hat{\mathbb{D}}(\mathbf{w}) - \|\mathbf{w}\|_1).$$

Therefore, with a probability at least $1 - \delta$,

$$\begin{aligned}\big|\mathbb{D}(\mathbf{w}) - \|\mathbf{w}\|_1 - (\hat{\mathbb{D}}(\mathbf{w}) - \|\mathbf{w}\|_1)\big| &\leq nW\sqrt{2\ln(2/\delta)/m} \\ &= \|\mathbf{w}\|_1 \sqrt{2\ln(2/\delta)/m},\end{aligned}$$

as claimed. □

## 4. Main result: the AWP algorithm

We propose the AWP (Approximated Weights via Pruning) algorithm, listed in Alg. 1. AWP uses weight queries to find a pruning $P$, which induces a weight function $w_P$ as defined in Section 2. AWP gets the following inputs: a binary tree $T$ whose leaves are the data set elements, the requested pruning size $K \geq 2$, a confidence parameter $\delta \in (0, 1)$, and a constant $\beta > 1$, which controls the trade-off between the number of weight queries requested by AWP and the approximation factor that it obtains. Finding a pruning with a small discrepancy while limiting the number of weight queries involves several challenges, since the discrepancy of any given pruning is unknown in advance, and the number of possibilities is exponential in $K$. AWP starts with the trivial pruning, which includes only the root node. It iteratively samples weight queries of leaves to estimate the discrepancy of nodes in the current pruning. AWP decides in a greedy manner when to *split* a node in the current pruning, that is, to replace it in the pruning with its two child nodes. It stops after reaching a pruning of size $K$.

We first provide the notation for Alg. 1. For a node $v$ in $T$, $M_v$ is the number of weight queries of examples in $\mathcal{L}_v$ requested so far by the algorithm for estimating $\mathbb{D}_v$. The sequence of $M_v$ weights returned by the oracle for these queries is denoted $\mathbf{z}_v$. Note that although an example in $\mathcal{L}_u$ is also in $\mathcal{L}_v$ for any ancestor $v$ of $u$, weight queries used for estimating $\mathbb{D}_v$ are not reused for estimating $\mathbb{D}_u$, since this would bias the estimate. AWP uses the estimator for $\mathbb{D}_v$ provided in Lemma 3.1. In AWP notation, the estimator is:

$$\hat{\mathbb{D}}_v := w^*_v + \frac{N_v}{M_v}(\|\mathbf{z}_v - \frac{w^*_v}{N_v} \cdot \mathbf{1}\|_1 - \|\mathbf{z}_v\|_1). \quad (1)$$

AWP iteratively samples weight queries of examples for nodes in the current pruning, until it can identify a node which has a relatively large discrepancy. The iterative sampling procedure takes inspiration from the upper-confidence-bound (UCB) approach, common in best-arm-identification problems (Audibert et al., 2010); In our case, the goal is to find the best node up to a multiplicative factor. In each iteration, the node with the maximal known upper bound on its discrepancy is selected, and the weight of a random example from its leaves is queried. To calculate the upper bound, we define

$$\Delta_v := w^*_v \cdot \sqrt{2\ln(2K\pi^2 M_v^2/(3\delta))/M_v}. \quad (2)$$

We show in Section 5 that $|\mathbb{D}_v - \hat{\mathbb{D}}_v| \leq \Delta_v$ with a high probability. Hence, the upper bound for $\mathbb{D}_v$ is set to $\hat{\mathbb{D}}_v + \Delta_v$. Whenever a node from the pruning is identified as having a large discrepancy in comparison with the other nodes, it is replaced by its child nodes, thus increasing the pruning size by one. Formally, AWP splits a node if with a high probability, $\forall v' \in P \setminus \{v\}, \mathbb{D}_v \geq \mathbb{D}_{v'}/\beta$.

The factor of $\beta$ trades off the optimality of the selected node in terms of its discrepancy with the number of queries needed to identify such a node and perform a split. In addition, it makes sure that a split can be performed even if all nodes have a similar discrepancy. The formal splitting criterion is defined via the following Boolean function:

$$\mathrm{SC}(v, P) = \left\{\beta(\hat{\mathbb{D}}_v - \Delta_v) \geq \max_{v' \in P \setminus \{v\}} \hat{\mathbb{D}}_{v'} + \Delta_{v'}\right\}. \quad (3)$$

To summarize, AWP iteratively selects a node using the UCB criterion, and queries the weight of a random leaf of that node. Whenever the splitting criterion holds, AWP splits a node that satisfies it. This is repeated until reaching a pruning of size $K$. In addition, when a node is added to the pruning, its weight is queried for use in Eq. (1).

We now provide our guarantees for AWP. The properties of the tree $T$ affect the quality of the output weighting. First, the pruning found by AWP cannot be better than the best pruning of size $K$ in $T$. Thus, we guarantee an approximation factor relative to that pruning. In addition, we require the tree to be sufficiently nice, in that a child node should have a somewhat lower discrepancy than its parent. Formally, we define the notion of *split quality*.

**Definition 4.1** (Split quality)**.** *Let $T$ be a hierarchical tree for $S$, and $q \in (0, 1)$. $T$ has a* split quality *$q$ if for any two nodes $v, u$ in $T$ where $u$ is a child of $v$, we have $\mathbb{D}_u \leq q\mathbb{D}_v$.*

This definition is similar in nature to other tree quality notions, such as the taxonomy quality of Slivkins (2011), though the latter restricts weights and not the discrepancy. We note that Def. 4.1 could be relaxed, for instance by allowing different values of $q$ in different tree levels. Nonetheless,

**Algorithm 1** AWP: Approximated weighting of a data set via a low-discrepancy pruning

```
1: Input: A binary tree $T$ with examples as leaves; Maximum pruning size $K \geq 2$; $\delta \in (0,1)$; $\beta > 1$.
2: Output: $P_o$: A pruning of $T$ with a small $\mathbb{D}_{P_o}$
3: Initializations: $P \leftarrow \{$The root node of $T\}$;
4: For all nodes $v$ in $T$: $\mathbf{z}_v \leftarrow ()$, $M_v \leftarrow 0$.
5: while $|P| < K$ do
6:    ▷ Select a node to sample from
7:    Set $v_s \leftarrow \mathrm{argmax}_{v \in P}(\hat{\mathbb{D}}_v + \Delta_v)$.
8:    Draw uniformly random example $x$ from $\mathcal{L}_{v_s}$
9:    Query the true weight of $x$, $w^*(x)$.
10:   $M_{v_s} \leftarrow M_{v_s} + 1$, $\mathbf{z}_{v_s} \leftarrow \mathbf{z}_{v_s} \circ w^*(x)$.
11:   ▷ Decide whether to split a node in the pruning:
12:   while $\exists v \in P$ s.t. $\mathrm{SC}(v, P)$ holds (Eq. (3)) do
13:      Let $v_R, v_L$ be children of such a $v$.
14:      Set $P \leftarrow P \setminus \{v\} \cup \{v_R, v_L\}$.
15:      Query $w^*_{v_R}$; set $w^*_{v_L} \leftarrow w^*_v - w^*_{v_R}$.
16:   end while
17: end while
18: return $P_o := P$.
```

we prove in Appendix A that greedily splitting the node with the largest discrepancy cannot achieve a reasonable approximation factor without some restriction on the input tree, and that this also holds for other types of greedy algorithms. It is an open problem whether this limitation applies to all greedy algorithms. Note also that even in trees with a split quality less than 1, splitting a node might increase the total discrepancy; see Section 5 for further discussion. Our main result is the following theorem.

**Theorem 4.2.** *Suppose that* AWP *gets the inputs $T$, $K \geq 2$, $\delta \in (0,1)$, $\beta > 1$ and let $P_o$ be its output. Let $Q$ be a pruning of $T$ of size $K$ with a minimal $\mathbb{D}_Q$. With a probability at least $1 - \delta$, we have:*

- *If $T$ has split quality $q$ for some $q < 1$, then*
$$\mathbb{D}_{P_o} \leq 2\beta(\frac{\log(K)}{\log(1/q)} + 1)\mathbb{D}_Q.$$
- AWP *requests $K - 1$ weight queries of internal nodes.*
- AWP *requests $\tilde{O}((1 + \frac{1}{\beta-1})^2 K^3 \ln(1/\delta)/\mathbb{D}^2_{P_o})$ weight queries of examples.*[1]

Thus, an approximation factor with respect to the best achievable discrepancy is obtained, while keeping the number of higher-order weight queries minimal and bounding the number of weight queries of examples requested by the algorithm. The theorem is proved in the next section.

[1]The $\tilde{O}$ notation hides constants and logarithmic factors; These are explicit in the proof of the theorem.

## 5. Analysis

In this section, we prove the main result, Theorem 4.2. First, we prove the correctness of the definition of $\Delta_v$ given in Section 4, using the concentration bound given in Lemma 3.1 for the estimator $\hat{\mathbb{D}}_v$. The proof is provided in Appendix C.

**Lemma 5.1.** *Fix inputs $T, K, \delta, \beta$ to* AWP. *Recall that $P$ is the pruning updated by* AWP *during its run. The following event holds with a probability at least $1 - \delta$ on the randomness of* AWP*:*

$$E_0 := \{\text{At all times during the run of AWP}, \forall v \in P, |\mathbb{D}_v - \hat{\mathbb{D}}_v| \leq \Delta_v\}. \quad (4)$$

Next, we bound the increase in discrepancy that could be caused by a node split. Even in trees with a split quality less than 1, a split could increase the discrepancy of the pruning. The next lemma bounds this increase, and shows that this bound is tight. The proof is provided in Appendix D.

**Lemma 5.2.** *Let $r$ be the root of a hierarchical tree and let $P$ be a pruning of this tree. Then $\mathbb{D}_P \leq 2\mathbb{D}_r$. Moreover, for any $\epsilon > 0$, there exists a tree with a split quality $q < 1$ and a pruning $P$ of size 2 such that $\mathbb{D}_P \geq (2 - \epsilon)\mathbb{D}_r$.*

We now prove the two main parts of Theorem 4.2, starting with the approximation factor of AWP. In the proof of the following lemma, the proof of some claims is omitted. The full proof is provided in Appendix E.

**Lemma 5.3.** *Fix inputs $T, K, \delta, \beta$ to* AWP, *and suppose that $T$ has a split quality $q \in (0,1)$. Let $P_o$ be the output of* AWP. *Let $E_0$ be the event defined in Eq. (4). In any run of* AWP *in which $E_0$ holds, for any pruning $Q$ of $T$ such that $|Q| = K$, we have $\mathbb{D}_{P_o} \leq 2\beta(\frac{\log(K)}{\log(1/q)} + 1)\mathbb{D}_Q$.*

*Proof.* Let $Q$ be some pruning such that $|Q| = K$. Partition $P_o$ into $R, P_a$ and $P_d$, where $R := P_o \cap Q$, $P_a \subseteq P_o$ is the set of strict ancestors of nodes in $Q$, and $P_d \subseteq P_o$ is the set of strict descendants of nodes in $Q$. Let $Q_a \subseteq Q$ be the ancestors of the nodes in $P_d$ and let $Q_d \subseteq Q$ be the descendants of the nodes in $P_a$, so that $R, Q_d$ and $Q_a$ form a partition of $Q$. First, we prove that we may assume without loss of generality that $P_a, P_d, Q_a, Q_d$ sets are non-empty.

**Claim 1**: If any of the sets $P_a, P_d, Q_a, Q_d$ is empty then the statement of the lemma holds.

The proof of Claim 1 is deferred to Appendix E. Assume henceforth that $P_a, P_d, Q_a, Q_d$ are non-empty. Let $r$ be the node with the smallest discrepancy out of the nodes that were split by AWP during the entire run. Define $\theta := |P_a| \cdot \mathbb{D}_r/\mathbb{D}_{Q_a}$ if $\mathbb{D}_{Q_a} > 0$ and $\theta := 0$ otherwise.

**Claim 2**: $\mathbb{D}_{P_o} \leq \max(2, \beta\theta)\mathbb{D}_Q$.

**Proof of Claim 2**: We bound the discrepancies of $P_d$ and of $P_a$ separately. For each node $u \in Q_a$, denote by $P(u)$

the descendants of $u$ in $P_d$. These form a pruning of the sub-tree rooted at $u$. In addition, the sets $\{P(u)\}_{u\in Q_a}$ form a partition of $P_d$. Thus, by the definition of discrepancy and Lemma 5.2,

$$\mathbb{D}_{P_d} = \sum_{u\in Q_a} \mathbb{D}_{P(u)} \leq \sum_{u\in Q_a} 2\mathbb{D}_u = 2\mathbb{D}_{Q_a}. \qquad (5)$$

Let $P$ be the pruning when AWP decided to split node $r$. By the definition of the splitting criterion SC (Eq. (3)), for all $v \in P\backslash\{r\}$, at that time it held that $\beta(\hat{\mathbb{D}}_r - \Delta_r) \geq \hat{\mathbb{D}}_v + \Delta_v$. Since $E_0$ holds, we have $\mathbb{D}_r \geq \hat{\mathbb{D}}_r - \Delta_r$ and $\hat{\mathbb{D}}_v + \Delta_v \geq \mathbb{D}_v$. Therefore, $\forall v \in P \setminus \{r\}, \beta\mathbb{D}_r \geq \mathbb{D}_v$.

Now, any node $v' \in P_o \setminus P$ is a descendant of some node $v \in P$. Since $T$ has split quality $q$ for $q < 1$, we have $\mathbb{D}_{v'} \leq \mathbb{D}_v$. Therefore, for all $v' \in P_o$, $\mathbb{D}_{v'} \leq \beta\mathbb{D}_r$. In particular, $\mathbb{D}_{P_a} \equiv \sum_{v\in P_a} \mathbb{D}_v \leq \beta|P_a|\mathbb{D}_r$. Since all nodes in $Q_a$ were split by AWP $\mathbb{D}_{Q_a} = 0$ implies $\mathbb{D}_r = 0$, therefore in all cases $\mathbb{D}_{P_a} \leq \beta\theta\mathbb{D}_{Q_a}$. Combining this with Eq. (7), we get that

$$\begin{aligned}\mathbb{D}_{P_o} &= \mathbb{D}_R + \mathbb{D}_{P_d} + \mathbb{D}_{P_a} \leq \mathbb{D}_R + 2\mathbb{D}_{Q_a} + \beta\theta\mathbb{D}_{Q_a} \\ &\leq \max(2, \beta\theta)\mathbb{D}_Q,\end{aligned}$$

which completes the proof of Claim 2.

It follows from Claim 2 that to bound the approximation factor, it suffices to bound $\theta$. Let $P'_d$ be the set of nodes both of whose child nodes are in $P_d$ and denote $n := |P'_d|$. In addition, define

$$\alpha := \frac{\log(1/q)}{\log(|P_a|) + \log(1/q)} \leq 1.$$

We now prove that $\theta \leq 2/\alpha$ by considering two complementary cases, $n \geq \alpha|P_a|$ and $n < \alpha|P_a|$. The following claim handles the first case.

**Claim 3:** if $n \geq \alpha|P_a|$, then $\theta \leq 2/\alpha$.

**Proof of Claim 3**: Each node in $P'_d$ has an ancestor in $Q_a$, and no ancestor in $P'_d$. Therefore, $P'_d$ can be partitioned to subsets according to their ancestor in $Q_a$, and each such subset is a part of some pruning of that ancestor. Thus, by Lemma 5.2, $\mathbb{D}_{P'_d} \leq 2\mathbb{D}_{Q_a}$. Hence, for some node $v \in P'_d$, $\mathbb{D}_v \leq 2\mathbb{D}_{Q_a}/n$. It follows from the definition of $r$ that $\mathbb{D}_r \leq 2\mathbb{D}_{Q_a}/n$. Hence, $\theta \leq 2|P_a|/n$. Since $n \geq \alpha|P_a|$, we have $\theta \leq 2/\alpha$ as claimed.

We now prove this bound hold for the case $n < \alpha|P_a|$. For a node $v$ with an ancestor in $Q_a$, let $l_v$ be the path length from this ancestor to $v$, and define $L := \sum_{v\in P'_d} l_v$. We start with an auxiliary Claim 4.

**Claim 4**: $L \geq |P_a| - n$.

The proof of Claim 4 is deferred to Appendix E. We use Claim 4 to prove the required upper bound on $\theta$ in Claim 5.

**Claim 5**: if $n < \alpha|P_a|$, then $\theta \leq 2/\alpha$.

**Proof of Claim 5**: It follows from Claim 4 that for some node $v \in P'_d$, $l_v \geq (|P_a| - n)/n = |P_a|/n - 1 > 0$, where the last inequality follows since $n < \alpha|P_a| < |P_a|$. Letting $u \in Q_a$ be the ancestor of $v$ in $Q_a$, we have by the split quality $q$ of $T$ that $\mathbb{D}_v \leq \mathbb{D}_u \cdot q^{\frac{|P_a|}{n}-1}$. Since $u \in Q_a$, we have $\mathbb{D}_u \leq \mathbb{D}_{Q_a}$. In addition, $\mathbb{D}_r \leq \mathbb{D}_v$ by the definition of $r$. Therefore, $\mathbb{D}_r \leq \mathbb{D}_{Q_a} \cdot q^{\frac{|P_a|}{n}-1}$. Since $n < |P_a|\alpha$ and $q < 1$, from the definition of $\alpha, \theta$ we have $\theta \leq |P_a|q^{\frac{|P_a|}{n}-1} \leq 1 \leq 2/\alpha$. This proves Claim 5.

Claims 3 and 5 imply that in all cases, $\theta \leq 2/\alpha$. By Substituting $\alpha$, we have that

$$\theta \leq 2(\frac{\log(|P_a|)}{\log(1/q)} + 1) \leq 2(\frac{\log(K)}{\log(1/q)} + 1).$$

Placing this upper bound in the statement of Claim 2 concludes of the lemma. $\square$

Next, we prove an upper bound on the number of weight queries requested by AWP.

**Lemma 5.4.** *Let $\beta > 1$. Consider a run of* AWP *in which $E_0$ holds, and fix some iteration. Let $P$ be the current pruning, and for a node $v$ in $T$, denote*

$$\alpha_v := \frac{\beta}{\beta - 1} \cdot \frac{w^*_v}{\max_{u\in P} \mathbb{D}_u}.$$

*Then in this iteration, the node $v_s$ selected by* AWP *satisfies $M_{v_s} < 22\alpha^2_{v_s}(\ln(\alpha^4_{v_s}K/\delta) + 10)$.*

*Proof.* Recall that the next weight query to be sampled by AWP is set to $v_s \in \text{argmax}_{v\in P}(\hat{\mathbb{D}}_v + \Delta_v)$. First, if some nodes $v \in P$ have $M_v = 0$, they will have $\Delta_v = \infty$, thus one of them will be set as $v_s$, in which case the bound on $M_{v_s}$ trivially holds. Hence, we assume below that for all $v \in P$, $M_v \geq 1$. Denote $\mathbb{D}_{\max} := \max_{v\in P} \mathbb{D}_v$. Let $u \in \text{argmax}_{v\in P} \hat{\mathbb{D}}_v$ be a node with a maximal estimated discrepancy. Since $E_0$ holds, for all $v \in P$ we have $\hat{\mathbb{D}}_v + \Delta_v \geq \mathbb{D}_v$. Thus, by definition of $v_s$, $\hat{\mathbb{D}}_{v_s} + \Delta_{v_s} \geq \mathbb{D}_{\max}$ which implies $\hat{\mathbb{D}}_u \geq \hat{\mathbb{D}}_{v_s} \geq \mathbb{D}_{\max} - \Delta_{v_s}$. Therefore,

$$\begin{aligned}\beta(\hat{\mathbb{D}}_u - \Delta_u) &= \hat{\mathbb{D}}_u + (\beta - 1)\hat{\mathbb{D}}_u - \beta\Delta_u \\ &\geq \hat{\mathbb{D}}_u + (\beta - 1)(\mathbb{D}_{\max} - \Delta_{v_s}) - \beta\Delta_u.\end{aligned}$$

Denote $\theta := (\beta - 1)\mathbb{D}_{\max}/(2\beta)$, and assume for contradiction that $\Delta_{v_s} \leq \theta$. By the definitions of $u$ and $v_s$, we have $\Delta_u \leq \Delta_{v_s} \leq \theta$. Thus, from the inequality above and the definitions of $\theta, u, v_s$,

$$\begin{aligned}\beta(\hat{\mathbb{D}}_u - \Delta_u) &\geq \hat{\mathbb{D}}_u + (\beta - 1)(\mathbb{D}_{\max} - \theta) - \beta\theta \qquad (6) \\ &= \hat{\mathbb{D}}_u + \theta \geq \hat{\mathbb{D}}_{v_s} + \Delta_{v_s} \geq \max_{z\in P\backslash\{u\}}(\hat{\mathbb{D}}_z + \Delta_z),\end{aligned}$$

implying that $\mathrm{SC}(u, P)$ holds. But this means that the previous iteration should have split this node in the pruning, a contradiction. Therefore, $\Delta_{v_s} > \theta$. Since by Eq. (2), $\Delta_{v_s} \equiv w^*_{v_s}\sqrt{2\ln(2K\pi^2 M^2_{v_s}/(3\delta))/M_{v_s}}$, it follows that

$$M_{v_s} < \frac{2{w^*_{v_s}}^2}{\theta^2} \cdot \ln(\frac{2K\pi^2 M^2_{v_s}}{3\delta}).$$

Denoting $\phi = 4{w^*_{v_s}}^2/\theta^2$, $\mu = \sqrt{2K\pi^2/(3\delta)}$, this is equivalent to $M_{v_s} < \phi\ln(\mu M_{v_s})$. By Lemma F.1 in Appendix F, this implies $M_{v_s} < e\phi\ln(e\mu\phi)$. Noting that $\phi = 16\alpha_v^2$ and bounding constants, this gives the required bound. □

Lastly, we combine the lemmas to prove the main theorem.

*Proof of Theorem 4.2.* Consider a run in which $E_0$ holds. By Lemma 5.1, this occurs with a probability of $1-\delta$. If $T$ has a split quality $q < 1$, the approximation factor follows from Lemma 5.3. Next, note that AWP makes a higher-order query of a node only if it is the right child of a node that was split in line 1 of Alg. 1. Thus, it makes $K-1$ higher-order weight queries. This proves the first two claims.

We now use Lemma 5.4 to bound the total number of weight queries of examples under $E_0$. First, we lower bound $\max_{v\in P}\mathbb{D}_v$ for any pruning $P$ during the run of AWP. Let $u \in \mathrm{argmax}_{v\in P_o}\mathbb{D}_v$. By the definition of $u$, we have $\mathbb{D}_u \geq \mathbb{D}_{P_o}/K$. Now, at any iteration during the run of AWP, some ancestor $v$ of $u$ is in $P$. By Lemma 5.2, we have $\mathbb{D}_v \geq \mathbb{D}_u/2$. Therefore, $\max_{v\in P}\mathbb{D}_v \geq \mathbb{D}_{P_o}/(2K)$. Thus, by Lemma 5.4, at any iteration of AWP, $v_s$ is set to some node in $P$ that satisfies $M_{v_s} < 22\alpha^2_{v_s}(\ln(\alpha^4_{v_s}K/\delta)+10)$, where $\alpha_{v_s} := \frac{\beta}{\beta-1}\cdot\frac{2Kw^*_v}{\mathbb{D}_{P_o}}$. Hence, AWP takes at most $T_v := 22\alpha^2_{v_s}(\ln(\alpha^4_{v_s}K/\delta)+10)+1$ samples for node $v$, and so the the total number of example weight queries taken by AWP is at most $\sum_{v\in V}T_v$, where $V$ is the set of nodes that participate in $P$ at any time during the run. To bound this sum, note that for any pruning $P$ during the run of AWP, we have $\sum_{v\in P}w^*_v = 1$. Hence, $\sum_{v\in P}{w^*_v}^2 \leq 1$. Since there are $K$ different prunings during the run, we have $\sum_{v\in V}{w^*_v}^2 \leq K$. Substituting $\alpha_v$ by its definition and rearranging, we get that the total number of example weight queries by AWP is $\tilde{O}((1+\frac{1}{\beta-1})^2K^3\ln(1/\delta)/\mathbb{D}^2_{P_o})$. □

## 6. Experiments

We report experiments that compare AWP to several natural baselines. The full results of the experiments described below, as well as results for additional experiments, are reported in Appendix H. A python implementation of the proposed algorithm and of all the experiments can be found at `https://github.com/Nadav-Barak/AWP`. The implementation can be easily run on a standard personal computer, with the longest runs taking a few hours.

The implementation of AWP includes two practical improvements: First, we use an empirical Bernstein concentration bound (Maurer and Pontil, 2009) to reduce the size of $\Delta_v$ when possible; This does not affect the correctness of the analysis. See Appendix G for details. Second, for all algorithms, we take into account the known weight values of single examples in the output weighting, as follows. For $v \in P_o$, let $S_v$ be the examples in $\mathcal{L}_v$ whose weight was queried. Given the output pruning $P$, we define the weighting $w'_P$ as follows. For $x \in S_v$, $w'_P(x) := w^*(x)$; for $x \in \mathcal{L}_v \setminus S_v$, we set $w'_P(x) := (w^*_v - \sum_{x\in S_v}w^*(x))/(N_v - |S_v|)$. In all the plots, we report the normalized output distance $\mathrm{dist}(w'_P, w^*) = \|w'_P - w^*\|_1/2 \in [0,1]$, which is equal to half the discrepancy of $w'_P$.

To fairly compare AWP to the baselines, they were allowed the same number of higher-order weight queries and basic weight queries as requested by AWP for the same inputs. The baselines are non-adaptive, thus the basic weight queries were drawn uniformly from the data set at the start of their run. We tested the following baselines: (1) WEIGHT: Iteratively split the node with the largest weight. Use basic weight queries only for the final calculation of $w'_P$. The rationale here is to get a finer resolution pruning in more important parts of the data set. However, this can lead to wasted resources on parts of the tree which are both high weight and low-discrepancy and an unbounded approximation factor, as proved in Appendix A. (2) UNIFORM: Iteratively split the node $v$ with the largest $\hat{\mathbb{D}}_v$, the estimator used by AWP, but without adaptive queries. (3) EMPIRICAL: Same as UNIFORM, except that instead of $\hat{\mathbb{D}}_v$, it uses the naive empirical estimator, $\frac{n}{|S_v|}\sum_{x\in S_v}|w^*_v(x)/N_v - w^*(x)|$ (See Section 3).

We ran several types of experiments, all with inputs $\delta = 0.05$ and $\beta = 4$. Note that $\beta$ defines a trade-off between the number of queries and the quality of the solution. Therefore, its value represents user preferences and not a hyperparameter to be optimized. For each experiment, we report the average normalized output distance over 10 runs, as a function of the pruning size. Error bars, displayed as shaded regions, show the maximal and minimal normalized distances obtained in these runs. For each input data set, we define an input hierarchical tree $T$ and test several target distributions, which determine the true weight function $w^*$.

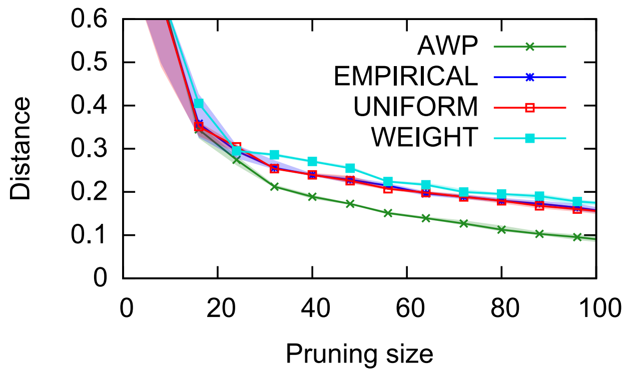
In the first set of experiments, the input data set was Adult (Dua and Graff, 2017), which contains $\sim 45{,}000$ population census records (after excluding those with missing values). We created the hierarchical tree via a top-down procedure, in which each node was split by one of the data set attributes, using a balanced splitting criterion. Similarly to the hospital example given in Section 1, the higher-order queries (internal nodes) in this tree require finding the proportion of the population with a specific set of demographic characteristics, which could be obtained via a database counting query. For instance, a higher-order query could correspond to the set of single government employees aged 45 or higher. To generate various target distributions, we partitioned the data set into ordered bins, where in each experiment the partition was based on the value of a different attribute. The target weights were set so that each example in a given bin was $N$ times heavier than each example in the next bin. We ran this experiment with values $N = 2, 4$. Some plots are given in Figure 1. See Appendix H for full details and results.

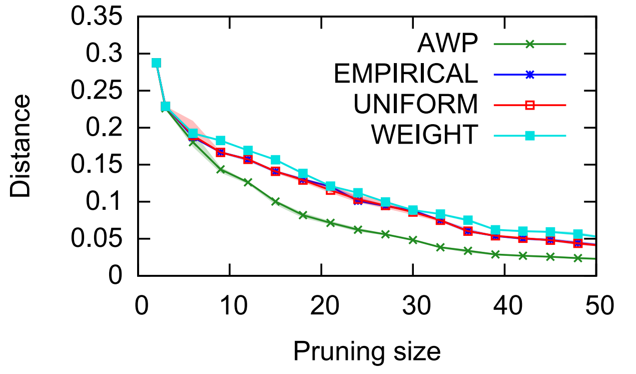
The rest of the experiments were done on visual data sets, with a tree that was generated synthetically from the input data set using Ward's method (Müllner, 2011), as implemented in the `scipy` python package.

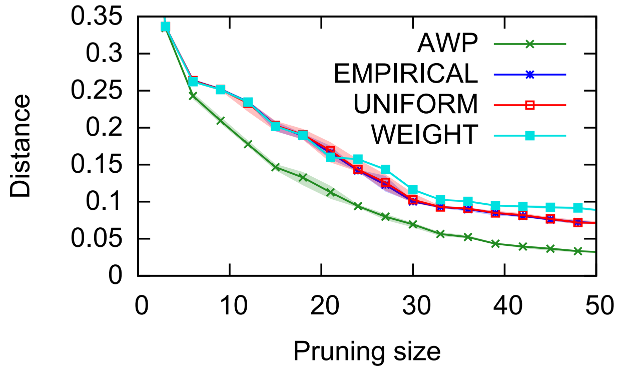
First, we tested the MNIST (LeCun and Cortes, 2010) training set, which contains 60,000 images of hand-written digits, classified into 10 classes (digits). The target distributions were generated by allocating the examples into ordered bins based on the class labels, and allocating the weights as in the Adult data set experiment above, again with $N = 2, 4$. We tested three random bin orders. See Figure 1 for some of the results. See Appendix H for full details and results.

Lastly, we ran experiments using an input data set and target distribution based on data set pairs commonly studied in domain adaptation settings (e.g., Gong et al. 2012; Hoffman et al. 2012; Ding et al. 2015). The input data set was Caltech256 (Griffin et al., 2007), with 29,780 images of various objects, classified into 256 classes (not including the singular "clutter" class). In each experiment, the target distribution was determined by a different data set as follows: The target weight $w^*$ of each image in the Caltech256 data set was set to the fraction of images from the target data set that have this image as their nearest neighbor. We used two target data sets: (1) The Office data set (Saenko et al., 2010), of which we used the 10 classes that also exist in Caltech256 (1410 images); (2) The Bing data set (Alessandro Bergamo, 2010a;b), which includes 300 images in each Caltech256 class. For Bing, we also ran experiments where images from a single super-class from the taxonomy in Griffin et al. (2007) were used as the target data set. See Figure 1 for some of the results, and Appendix H for full results.

In all experiments, except for a single configuration, `AWP` performed better than the other algorithms. In addition, `UNIFORM` and `EMPIRICAL` behave similarly in most experiments, with `UNIFORM` sometimes being slightly better. This shows that our new estimator is empirically adequate, on top of its crucial advantage in getting a small $\Delta_v$. We note that in our experiments, the split quality $q$ was usually close to one. This shows that `AWP` can be successful even in cases not covered by Theorem 4.2. We did find that the average splitting values were usually lower, see Appendix I.







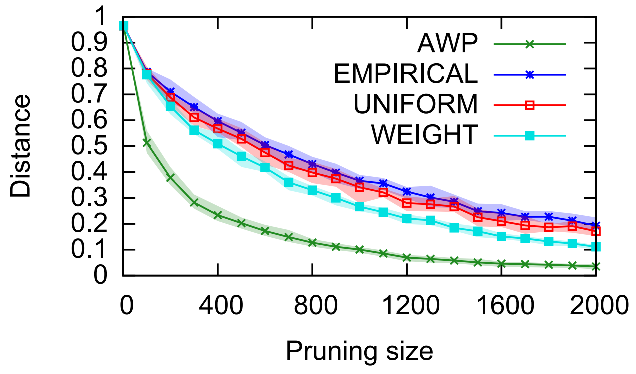


*Figure 1.* Some of the experiment results. Full results are provided in Appendix H. Top to Bottom: Adult data set, bins assigned by marital status, $N = 4$; MNIST: $N = 2$, $N = 4$; The Caltech$\rightarrow$Office experiment.

## 7. Conclusions

In this work, we studied a novel problem of approximating a distribution via weight queries, using a pruning of a hierarchical tree. We showed, both theoretically and experimentally, that such an approximation can be obtained using an efficient interactive algorithm which iteratively constructs a pruning. In future work, we plan to study the effectiveness of our algorithm under more relaxed assumptions, and to generalize the input structure beyond a hierarchical tree.

## References

Lorenzo Torresani Alessandro Bergamo. Exploiting weakly-labeled web images to improve object classification: a domain adaptation approach. In *Neural Information Processing Systems (NIPS)*, December 2010a.

Lorenzo Torresani Alessandro Bergamo. The Bing data set. `http://vlg.cs.dartmouth.edu/projects/domainadapt/data/Bing/`, 2010b.

Jean-Yves Audibert, Sébastien Bubeck, and Rémi Munos. Best arm identification in multi-armed bandits. *COLT 2010*, page 41, 2010.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In *Advances in neural information processing systems*, pages 137–144, 2007.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

Christopher Berlind and Ruth Urner. Active nearest neighbors in changing environments. In *International Conference on Machine Learning*, pages 1870–1879, 2015.

Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative learning for differing training and test distributions. In *Proceedings of the 24th international conference on Machine learning*, pages 81–88. ACM, 2007.

Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative learning under covariate shift. *Journal of Machine Learning Research*, 10(Sep):2137–2155, 2009.

John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman. Learning bounds for domain adaptation. In *Advances in neural information processing systems*, pages 129–136, 2008.

Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvári. X-armed bandits. *Journal of Machine Learning Research*, 12(May):1655–1695, 2011.

Adrian Corduneanu and Christopher M Bishop. Variational bayesian model selection for mixture distributions. In *Artificial intelligence and Statistics*, volume 2001, pages 27–34. Morgan Kaufmann Waltham, MA, 2001.

Corinna Cortes, Giulia Desalvo, Claudio Gentile, Mehryar Mohri, and Ningshan Zhang. Adaptive region-based active learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2144–2153. PMLR, 13–18 Jul 2020. URL `http://proceedings.mlr.press/v119/cortes20a.html`.

Sanjoy Dasgupta and Daniel Hsu. Hierarchical sampling for active learning. In *Proceedings of the 25th international conference on Machine learning*, pages 208–215. ACM, 2008.

Zhengming Ding, Ming Shao, and Yun Fu. Deep low-rank coding for transfer learning. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Mario A. T. Figueiredo and Anil K. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on pattern analysis and machine intelligence*, 24(3):381–396, 2002.

Alireza Ghasemi, Mohammad T Manzuri, Hamid R Rabiee, Mohammad H Rohban, and Siavash Haghiri. Active one-class learning by kernel density estimation. In *2011 IEEE International Workshop on Machine Learning for Signal Processing*, pages 1–6. IEEE, 2011.

Jacob Goldberger and Sam T Roweis. Hierarchical clustering of a mixture model. In *Advances in neural information processing systems*, pages 505–512, 2005.

Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2066–2073. IEEE, 2012.

Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. `http://www.vision.caltech.edu/Image_Datasets/Caltech256/`, 2007.

Judy Hoffman, Brian Kulis, Trevor Darrell, and Kate Saenko. Discovering latent domains for multisource domain adaptation. In *European Conference on Computer Vision*, pages 702–715. Springer, 2012.

Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *VLDB*, volume 4, pages 180–191. Toronto, Canada, 2004.

Samory Kpotufe and Guillaume Martinet. Marginal singularity, and the benefits of labels in covariate-shift. In Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet, editors, *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 1882–1886. PMLR, 06–09 Jul 2018.

Samory Kpotufe, Ruth Urner, and Shai Ben-David. Hierarchical label queries with data-dependent partitions. In *Conference on Learning Theory*, pages 1176–1189. PMLR, 2015.

Matej Kristan, Danijel Skočaj, and Ales Leonardis. Online kernel density estimation for interactive learning. *Image and Vision Computing*, 28(7):1106–1116, 2010.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 2010.

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. *arXiv preprint arXiv:0902.3430*, 2009.

Andreas Maurer and Massimiliano Pontil. Empirical Bernstein bounds and sample-variance penalization. In *Proceedings of the 22nd Annual Conference on Learning Theory, COLT 2009*, 2009.

Clare A McGrory and DM237087605559631 Titterington. Variational approximations in bayesian model selection for finite mixture distributions. *Computational Statistics & Data Analysis*, 51(11):5352–5367, 2007.

Geoffrey J McLachlan and Thriyambakam Krishnan. *The EM algorithm and extensions*, volume 382. John Wiley & Sons, 2007.

Daniel Müllner. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*, 2011.

Rémi Munos. Optimistic optimization of a deterministic function without the knowledge of its smoothness. In *Advances in neural information processing systems*, pages 783–791, 2011.

Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pages 213–226. Springer, 2010. URL https://drive.google.com/open?id=0B4IapRTv9pJ1WGZVd1VDMmhwdlE.

Bernard W Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.

Aleksandrs Slivkins. Multi-armed bandits on implicit metric spaces. In *Advances in Neural Information Processing Systems*, pages 1602–1610, 2011.

Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul V Buenau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in neural information processing systems*, pages 1433–1440, 2008.

Ruth Urner, Sharon Wulff, and Shai Ben-David. Plal: Cluster-based active learning. In *Conference on Learning Theory*, pages 376–397. PMLR, 2013.

Matt P Wand and M Chris Jones. *Kernel smoothing*. Crc Press, 1994.

EL Wilmer, David A Levin, and Yuval Peres. Markov chains and mixing times. *American Mathematical Soc., Providence*, 2009.