Approximating a Distribution Using Weight Queries
ICML2021 Supplementary Material

A. Limitations of greedy algorithms

In this section we prove two lemmas which point to limitations of certain types of greedy algorithms for finding a pruning with a low discrepancy. The first lemma shows that without a restriction on the split quality of the input tree, the greedy algorithm which splits the node with the maximal discrepancy, as well as a general class of greedy algorithms, could obtain poor approximation factors.

Lemma A.1. *Consider a greedy algorithm which creates a pruning by starting with the singleton pruning that includes the root node, and iteratively splitting the node with the largest discrepancy in the current pruning. Then, for any even pruning size $K \geq 2$, there exists an input tree such that the approximation factor of the greedy algorithm is at least $K/4$.*

Moreover, the same holds for any greedy algorithm which selects the next node to split based only on the discrepancy of each node in the current pruning and breaks ties arbitrarily.

In both cases, the input tree that obtains this approximation factor does not have a split quality $q < 1$, but does satisfy the following property (equivalent to having a split quality of $q = 1$): For any two nodes v, u in T such that u is a child of v, $\mathbb{D}_u \leq \mathbb{D}_v$.

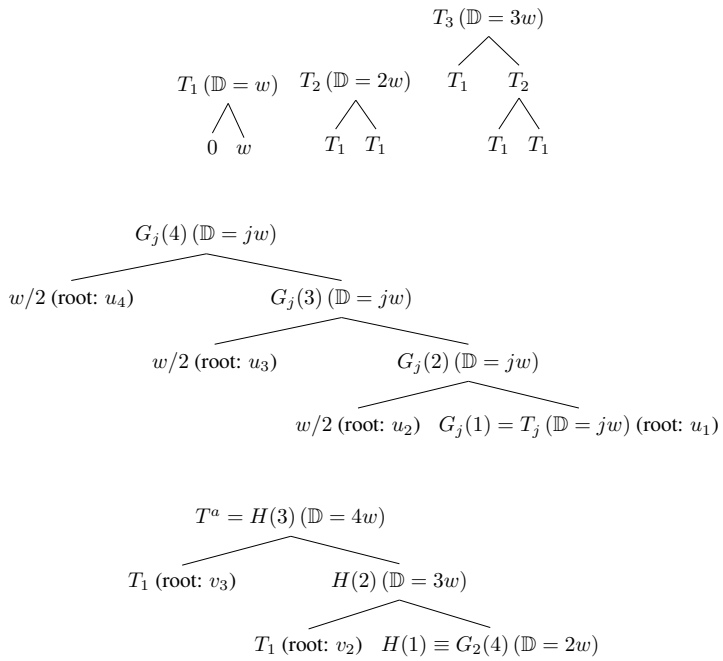

Figure 2. Illustrating the trees defined in the proof of Lemma A.1 for $k = 3$.

Proof. We define several trees; see illustrations in Figure 2. Let $w > 0$. Its value will be defined below. All the trees defined below have an average leaf weight of $w/2$. Therefore, when recursively combining them to a larger tree, the average weight remains the same, and so the discrepancy of any internal node (except for nodes with leaf children) is the total discrepancy of its two child nodes.

Let T_1 be a tree of depth 1, which has a root node with two child leaves with weights 0 and w. The root of T_1 has a discrepancy of w. For $i \geq 2$, let T_i be a tree of depth i such that one child node of the root is T_1 and the other is T_{i-1}. Note that T_i has a discrepancy of iw.

For positive integers i and j, define the tree $G_j(i)$ recursively, such that $G_j(1) := T_j$ (denote its root node u_1), and $G_j(i)$ has a root node with two children: a leaf with weight $w/2$ (denote it u_i) and $G_j(i-1)$. It is easy to verify that the discrepancy of $G_j(i)$ for all $i \geq 1$ is jw.

Let $k = K/2$. For the first part of the lemma, define $H(i)$ recursively. $H(1) = G_2(k+1)$, and $H(i)$ has a root with the children T_1 (denote its root v_i) and $H(i-1)$. The discrepancy of $H(i)$ is thus $(i+1)w$. Now, consider the greedy algorithm that iteratively splits the node with the largest discrepancy in the pruning, and suppose that it is run with the input tree $T^a := H(k)$ and a pruning size $K = 2k$. Set w so that the total weight of T^a is equal to 1. Due to the discrepancy values, the greedy algorithm splits the root nodes of $H(k), H(k-1), \ldots, H(1) = G_2(k+1)$ and then of $G_2(k+1), G_2(k), \ldots, G_2(2)$. The resulting pruning is $v_2, \ldots, v_k, u_1, \ldots, u_{k+1}$, with a total discrepancy of $(k-1)w + 2w = (k+1)w$. In contrast, consider the pruning of size K which includes the two leaf children of each of $v_2, \ldots, v_k$ and the children of the root of $G_2(k+1)$ (the sibling of v_2). This pruning has a discrepancy of $2w$. Thus, the approximation factor obtained by the greedy algorithm in this example is $(k+1)/2 \geq K/4$.

For the second part of the lemma, consider the tree T^b which has a root with the child nodes $G_1(2k)$ (which has a discrepancy of w) and T_{k-1} (which has a discrepancy of $(k-1)w$). Define w so that the total weight of T^b is equal to 1. Suppose that T^b and pruning size K are provided as input to some greedy algorithm that splits according to discrepancy values of nodes in the current pruning, and breaks ties arbitrarily. In the first round, the root node must be split. Thereafter, the current pruning always includes some pruning of $G_1(2k)$ (possibly the singlton pruning which includes just the root of this sub-tree). There is only one pruning of $G_1(2k)$ of size $i \leq 2k$, and it is composed of $i-1$ leaves of weight $w/2$ and the root of $G_1(2k-i+1)$. Therefore, at all times in the algorithm, the pruning includes some node with a discrepancy w which is the root of $G_1(i)$ for some $i \leq 2k$. It follows that the said greedy algorithm might never split any of the nodes which are the root of some T_1 under T_{k-1}, since these nodes also have a discrepancy of w. It also can never split $G_1(1) = T_1$, since this would require a pruning of size larger than $K = 2k$. As a result, such an algorithm might obtain a final pruning with a discrepancy of kw. In contrast, the pruning which includes all the child leaves of the sub-trees T_1 in T_{k-1} and two child nodes of $G_1(2k)$ has a discrepancy of w. This gives an approximation factor of $k = K/2 \geq K/4$. □

The next lemma shows that a different greedy approach, which selects the node to split by the maximal improvement in discrepancy, also fails. In fact, it obtains an unbounded approximation factor, even for a split quality as low as $1/2$.

Lemma A.2. *Consider a greedy algorithm that in each iteration splits the node v in the current pruning that maximizes $\mathbb{D}_v - (\mathbb{D}_{v_R} + \mathbb{D}_{v_L})$, where v_R and v_L are the child nodes of v. For any pruning size $K \geq 5$ and any value $N \geq 2$, there exists an input tree such that the approximation factor of this algorithm is larger than N. For any $\epsilon > 0$, there exists such a tree with a split quality $q \leq \frac{1}{2} + \epsilon$.*

Proof. We define a hierarchical tree; see illustration in Figure 3. Let $w > 0$. Its value will be defined below. Let $k \geq K - 2$. The input tree T has two child nodes. The left child node, denoted v_1, has two children, v_2 and v_3. Each of these child nodes has two leaf children, one with weight 0 and one with weight $Nw/2$. Thus, $\mathbb{D}_{v_2} = \mathbb{D}_{v_3} = Nw/2$, and $\mathbb{D}_{v_1} = Nw$.

The left child node is defined recursively as follows. For an integer m, let $F(m)$ be some complete binary tree with m leaves, each of weight $w' := w/3^k$. By definition, the discrepancy of the root of $F(m)$, for any integer m, is zero. We define $J(i)$ for $i \geq 0$ recursively, as follows. Let $J(0)$ be a tree such that its left child node is $F(1)$ and its right child node is the root of some complete binary tree with 3^k leaves of weight zero. Let $J(i)$ be a tree such that its left child is $F(2 \cdot 3^{i-1})$ and its right child is $J(i-1)$. Note that $J(0)$ has $1 = 3^0$ leaf of weight w', and by induction, $J(i)$ has $2 \cdot 3^{i-1} + 3^{i-1} = 3^i$ such leaves. In addition, $J(i)$ has 3^k leaves of weight 0. Thus, the root of $J(i)$ has an average weight of $\frac{3^i w'}{3^i + 3^k} = \frac{w}{3^k + 3^{2k-i}}$, and a discrepancy of

$$\alpha_i := 3^k \cdot \frac{w}{3^k + 3^{2k-i}} + 3^i(w' - \frac{w}{3^k + 3^{2k-i}}) = \frac{w}{1 + 3^{k-i}} + \frac{w}{3^{k-i}} - \frac{w}{3^{k-i} + 3^{2k-2i}} = \frac{2w}{1 + 3^{k-i}}.$$

The last equality follows by setting $a = 3^{k-i}$ and $b = 3^{2k-2i}$ so that $\frac{w}{3^{k-i}} - \frac{w}{3^{k-i} + 3^{2k-2i}} = w(\frac{1}{a} - \frac{1}{a+b})$, and noting

that

$$\frac{1}{a} - \frac{1}{a+b} = \frac{b}{a(a+b)} = \frac{1}{a^2/b + a} = \frac{1}{1+3^{k-i}}.$$

Now, consider running the given algorithm on the input tree T with pruning size K. Splitting v_1 into v_2 and v_3 does not reduce the total discrepancy. On the other hand, for any i, splitting the root of $J(i)$ reduces the total discrepancy, since it replaces a discrepancy of α_i with a discrepancy of zero (for $F(2 \cdot 3^{i-1})$) plus a discrepancy of $\alpha_{i-1} < \alpha_i$ (for $J(i-1)$). Therefore, the defined greedy algorithm will never split v_1, and will obtain a final pruning with a discrepancy of at least $\mathbb{D}_{v_1} = Nw$. On the other hand, any pruning which includes the leaves under v_2, v_3 will have a discrepancy of at most that of $J(k)$, which is equal to w. Therefore, the approximation factor of the greedy algorithm is at least N.

To complete the proof, we show that for a large k, the split quality of the tree is close to $1/2$. First, it is easy to see that the split quality of the tree rooted at v_1 is $1/2$. For the tree $J(k)$, observe that the discrepancy of $J(i)$ is α_i and the discrepancy of its child nodes is 0 and α_{i-1}. Since $\alpha_{i-1}/\alpha_i \leq 1/2$, $J(i)$ also has a split quality $\leq 1/2$, for all i. We have left to bound the ratio between the discrepancy of the root node and each of its child nodes. The left child node of the root has 4 leaves of total weight Nw, and the right child node has $2 \cdot 3^k$ leaves of total weight $3^k \cdot w' = w$. Therefore, the average weight of the root node is $\bar{w} := \dfrac{(N+1)w}{4 + 2 \cdot 3^k}$. The discrepancy of the root node is thus

$$2|Nw/2 - \bar{w}| + 2|\bar{w}| + 3^k|\bar{w} - w'| + 3^k|\bar{w}| = (N-1)w + 2 \cdot 3^k \bar{w} = (N-1)w + \frac{(N+1)w}{2 \cdot 3^{-k} + 1}.$$

For $k \to \infty$, this approaches $2Nw$ from below. Thus, for any ϵ, there is a sufficiently large k such that the discrepancy of the root is at least $Nw/(\frac{1}{2} + \epsilon)$. Since the discrepancy of each child node is $\leq Nw$, this gives a split quality of at most $\frac{1}{2} + \epsilon$. □

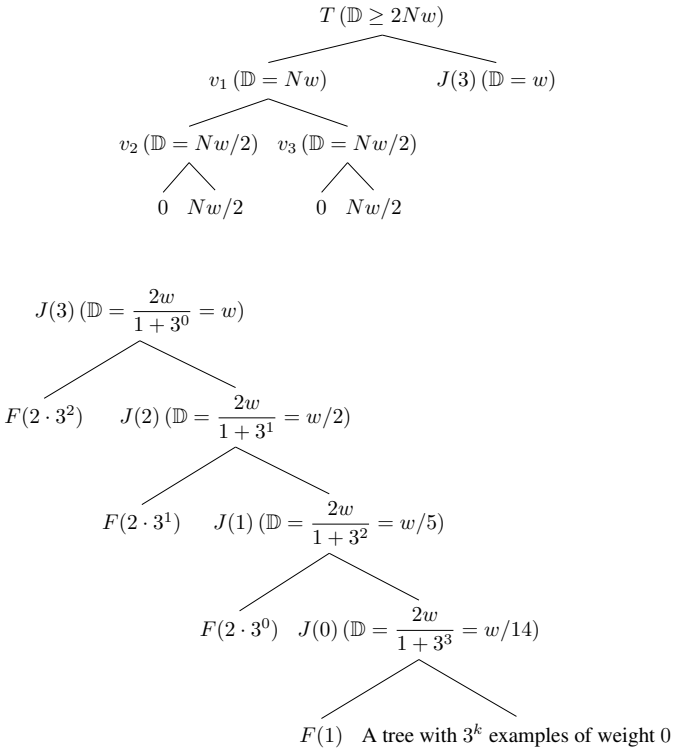

Figure 3. Illustrating the trees defined in the proof of Lemma A.2 for $k = 3$.

B. Limitations of other discrepancy estimators

First, we show that the discrepancy cannot be reliably estimated from weight queries of examples alone, unless almost all of the weights are sampled. To see this, consider a node v with $n+1$ descendant leaves (examples), all with the same weight $w = 1/(n+n^2)$, except for one special example with true weight either $n^2 w$ (first case) or w (second case). In the first case, the average weight of the examples is nw, and $\mathbb{D}_v = n \cdot |nw - w| + |nw - n^2 w| = 2(n^2 - n)w = 2 - 4/(n+1)$. In the second case, $\mathbb{D}_v = 0$. However, in a random sample of size $\leq n/2$, the probability that the special example is not observed is $(1 - 1/(n+1))^{n/2} \geq \frac{1}{2}$. When this example is not sampled, it is impossible to distinguish between the two cases unless additional information is available. This induces a large estimation error in this scenario.

Second, we show that even when w^*_v is known, a naive empirical estimator of the discrepancy can have a large estimation error. Recall that the discrepancy of a node v is defined as $\mathbb{D}_v := \sum_{x \in \mathcal{L}_v} |w^*_v/N_v - w^*(x)|$. Denote $n := |\mathcal{L}_v|$. Given a sample S_v of randomly selected examples in $\mathcal{L}_v$ whose weight has been observed, the naive empirical estimator for the discrepancy is $\frac{n}{|S_v|} \sum_{x \in S_v} |w^*_v/N_v - w^*(x)|$. Now, consider a case where $n-1$ examples from $\mathcal{L}_v$ have weight 0, and a single example has weight 1. We have $\mathbb{D}_v = (n-1) \cdot |1/n - 0| + |1/n - 1| = 2 - 2/n$. However, if the heavy example is not sampled, the naive empirical estimate is equal to 1. Similarly to the example above, if the sample size is of size $\leq n/2$, there is a probability of more than half that the heavy example is not observed, leading to an estimation error which is close to 1.

C. Proof of Lemma 5.1

Proof of Lemma 5.1. Fix a node v in T, and consider the value of $\hat{\mathbb{D}}_v$ after drawing M_v random samples from $\mathcal{L}_v$. To apply Lemma 3.1, set $\mathbf{w}$ to be the sequence of weights of the examples in $\mathcal{L}_v$. Then $\mathbb{D}_v = \mathbb{D}(\mathbf{w})$ and $\hat{\mathbb{D}}(\mathbf{w}) = \hat{\mathbb{D}}_v$. For an integer M, let $\delta(M) := \frac{3\delta}{K\pi^2 M^2}$. By Lemma 3.1, with a probability at least $1 - \delta(M_v)$, $|\mathbb{D}_v - \hat{\mathbb{D}}_v| \leq w^*_v \sqrt{2\ln(2/\delta(M_v))/M_v} \equiv \Delta_v$. We have $\sum_{n=1}^{\infty} \delta(n) = \frac{3\delta}{K\pi^2} \sum_{n=1}^{\infty} \frac{1}{n^2} = \delta/(2K)$. Thus, for any fixed node v, with a probability at least $1 - \delta/(2K)$, after any number of samples M_v, $|\mathbb{D}_v - \hat{\mathbb{D}}_v| \leq \Delta_v$. Denote this event D_v.

Now, the pruning P starts as a singleton containing the root node. Subsequently, in each update of P, one node is removed and its two children are added. Thus, in total $2K-1$ nodes are ever added to P (including the root node). Let v_i be the i'th node added to P, and let $V = \{v_1, \ldots, v_{2K-1}\}$. We have,

$$\mathbb{P}[E_0] \geq \mathbb{P}[\forall v \in V, D_v] = 1 - \mathbb{P}[\exists v \in V, \neg D_v].$$

Now, letting $\mathcal{N}$ be the nodes in T,

$$\mathbb{P}[\exists v \in V, \neg D_v] \leq \sum_{i=1}^{2K-1} \sum_{v \in \mathcal{N}} \mathbb{P}[v_i = v]\mathbb{P}[\neg D_v \mid v_i = v].$$

Now, $\mathbb{P}[\neg D_v \mid v_i = v] = \mathbb{P}[\neg D_v]$, since the estimate $\hat{\mathbb{D}}_v$ uses samples that are drawn after setting $v_i = v$. Therefore, $\mathbb{P}[\neg D_v \mid v_i = v] \leq \delta/(2K)$. Since $\sum_{v \in \mathcal{N}} \mathbb{P}[v_i = v] = 1$, it follows that $\mathbb{P}[\exists v \in V, \neg D_v] \leq \delta$. Therefore, We have $\mathbb{P}[E_0] \geq 1 - \delta$, as claimed.

□

D. Proof of Lemma 5.2

Proof of Lemma 5.2. To prove the first part, denote the nodes in P by $v_1, \ldots, v_n$ and let $v_0 := r$ be the root node. For $i \in \{0, \ldots, n\}$, let $\mathbf{w}_i$ be a sequence of length N_{v_i} of the weights $w^*(x)$ of all the leaves $x \in \mathcal{L}_{v_i}$. Let $\bar{w}^*_i := \frac{w^*_{v_i}}{N_{v_i}}$. Then

$$\begin{aligned}\mathbb{D}_{v_i} &= \|\mathbf{w}_i - \bar{w}^*_i \cdot \mathbf{1}\|_1 = \|\mathbf{w}_i - \bar{w}^*_0 \cdot \mathbf{1} + \bar{w}^*_0 \cdot \mathbf{1} - \bar{w}^*_i \cdot \mathbf{1}\|_1 \\ &\leq \|\mathbf{w}_i - \bar{w}^*_0 \cdot \mathbf{1}\|_1 + \|\bar{w}^*_0 \cdot \mathbf{1} - \bar{w}^*_i \cdot \mathbf{1}\|_1 = \|\mathbf{w}_i - \bar{w}^*_0 \cdot \mathbf{1}\|_1 + N_{v_i}|\bar{w}^*_i - \bar{w}^*_0|.\end{aligned}$$

Now, observe that

$$N_{v_i}|\bar{w}^*_i - \bar{w}^*_0| = |w^*_i - N_{v_i}\bar{w}^*_0| = |\sum_{x \in \mathcal{L}_{v_i}} (w^*(x) - \bar{w}^*_0)| \leq \sum_{x \in \mathcal{L}_{v_i}} |w^*(x) - \bar{w}^*_0| = \|\mathbf{w}_i - \mathbf{1} \cdot \bar{w}^*_0\|_1.$$

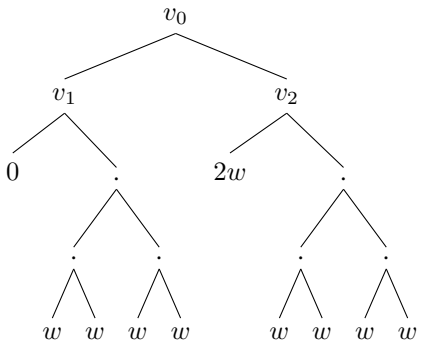

Figure 4. Illustrating the tree constructed in the proof of the second part of Lemma 5.2, for $n = 8$.

Therefore, $\mathbb{D}_{v_i} \leq 2\|\mathbf{w}_i - \mathbf{1} \cdot \bar{w}_0^*\|_1$. Summing over all the nodes, and noting that $\mathbf{w}_0 = \mathbf{w}_1 \circ \ldots \circ \mathbf{w}_n$, we get:

$$\sum_{i\in[n]} \mathbb{D}_{v_i} \leq \sum_{i\in[n]} 2\|\mathbf{w}_i - \bar{w}_0^* \cdot \mathbf{1}\|_1 = 2\|\mathbf{w}_0 - \bar{w}_0^* \cdot \mathbf{1}\|_1 = 2\mathbb{D}_{v_0},$$

which proves the first part of the lemma.

For the second part of the lemma, let n be an integer sufficiently large such that $\frac{2}{1+2/n} \geq 2 - \epsilon$. We consider a tree (see illustration in Figure 4) with $n + 2$ leaves. Denote the leaves by $x_1, \ldots, x_{n+2}$. Denote $w := 1/(n + 2)$, and define $w^*(x_1) = 0$, $w^*(x_2) = 2w$, and $w^*(x_i) = w$ for $i \geq 2$. Denote by v_0 the root node of the tree, and let its two child nodes be v_1 and v_2. The tree is organized so that x_1 and $n/2$ of the examples with weight w are descendants of v_1, and the other examples are descendants of v_2. v_1 has two child nodes, one is the leaf x_1 and the other is some binary tree whose leaves are all the other $n/2$ examples. Similarly, v_2 has a child node which is the leaf x_2, and the other examples are organized in some binary tree rooted at the other child node.

It is easy to see that $\mathbb{D}_{v_0} = 2w$. To calculate $\mathbb{D}_{v_1}$, note that the average weight of node v_1 is $\frac{nw/2}{n/2+1} = nw^2$. Thus,

$$\mathbb{D}_{v_1} \equiv \sum_{x\in\mathcal{L}_{v_1}} |nw^2 - w^*(x)| = \frac{n}{2}(w - nw^2) + nw^2 = \frac{nw}{2}(1 - \frac{n}{n+2}) + nw^2 = 2nw^2.$$

A similar calculation shows that $\mathbb{D}_{v_2} = 2nw^2$. Define the pruning $P = \{v_1, v_2\}$ of the tree rooted at v_0. Then

$$\mathbb{D}_P = 4nw^2 = 2nw \cdot \mathbb{D}_{v_0} = \frac{2}{1 + 2/n}\mathbb{D}_{v_0} \geq (2 - \epsilon)\mathbb{D}_{v_0},$$

as required.

To show that this tree has a split quality of less than 1, note that $\mathbb{D}_{v_1} = \mathbb{D}_{v_2} = \frac{1}{1+2/n}\mathbb{D}_{v_0}$, and that the discrepancy of each of the child nodes of v_1 and v_2 is zero, since all their leaves have the same weight. Therefore, this tree has a split quality $\frac{1}{1+2/n} < 1$. □

E. Proof of Lemma 5.3

Proof of Lemma 5.3. 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.

Proof of Claim 1: Observe that if any of P_a, P_d, Q_a, Q_d is empty then all of these sets are empty: By definition, $P_a = \emptyset \Leftrightarrow Q_d = \emptyset$ and $P_d = \emptyset \Leftrightarrow Q_a = \emptyset$. Now, suppose that $P_a = Q_d = \emptyset$. Since $|R| + |P_d| + |P_a| = |P_o| = |Q| =$

$|R| + |Q_d| + |Q_a|$, we deduce that $|P_d| = |Q_a|$. But for each node in Q_a, there are at least two descendants in P_d, thus $|P_d| \geq 2|Q_a|$. Combined with the equality, it follows that $P_d = Q_a = \emptyset$. The other direction is proved in an analogous way. Now, if $P_a = P_d = Q_a = Q_d = \emptyset$ then $P_o = Q$, thus in this case $\mathbb{D}_P = \mathbb{D}_Q$, which means that the statement of the lemma holds. This concludes the proof of Claim 1.

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,

$$\mathbb{D}_{P_d} = \sum_{u \in Q_a} \mathbb{D}_{P(u)} \leq \sum_{u \in Q_a} 2\mathbb{D}_u = 2\mathbb{D}_{Q_a}. \tag{7}$$

Let P be the pruning when AWP decided to split node r. By the definition of the splitting criterion SC (Eq. (3)), for all $v \in P \setminus \{r\}$, 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 θ. 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, and then prove the required upper bound on θ in Claim 5.

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

Proof of Claim 4: Fix some $u \in Q_a$, and let $P_u(t)$ be the set of nodes in the pruning P in iteration t which have u as an ancestor. Let $P'_u(t)$ be the set of nodes both of whose child nodes are in $P_u(t)$, and denote $L_u(t) := \sum_{v \in P'_u(t)} l_v$. We prove that for all iterations t, $L_u(t) \geq |P_u(t)| - 2$. First, immediately after u is split, we have $P'_u(t) = \{u\}$, $|P_u(t)| = 2$, $L_u(t) = 0$. Hence, $L_u(t) \geq |P_u(t)| - 2$. Next, let t such that $P_u(t)$ grows by 1, that is some node u_t in $P_u(t)$ is split. If u_t is the child of a node $v_t \in P'_u(t)$, then $P'_u(t+1) = P'_u(t) \setminus \{v_t\} \cup \{u_t\}$. In this case, $L_u(t+1) = L_u(t) + 1$, since $l_{u_t} = l_{v_t} + 1$. Otherwise, u_t is not a child of a node in $P'_u(t)$, so $P'_u(t+1) = P'_u(t) \cup \{u_t\}$, and so $L_u(t+1) = L_u(t) + l_{u_t} \geq L_u(t) + 1$. Thus, $L_u(t)$ grows by at least 1 when the size of $P_u(t)$ grows by 1. It follows that in all iterations, $L_u(t) \geq |P_u(t)| - 2$. Summing over $u \in Q_a$ and considering the final pruning, we get $L \geq |P_d| - 2|Q_a|$. Now, since $|P| = |Q|$, we have

$|P_d| - |Q_a| = |Q_d| - |P_a|$. From the definition of Q_d, $|Q_d| \geq 2|P_a|$. Therefore, $|P_d| - |Q_a| \geq |P_a|$. It follows that $L \geq |P_a| - |Q_a|$. Lastly, every node $u \in Q_a$ was split by `AWP`, and has at least one descendant in P'_d. Therefore, $|Q_a| \leq |P'_d| \equiv n$. Hence, $L \geq |P_a| - n$, which concludes the proof of Claim 4.

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 α, θ 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 α, 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. □

F. An auxiliary lemma

Lemma F.1. *Let $\mu > 0, \phi \geq 0, p \geq 0$. If $p < \phi \ln(\mu p)$ then $p < e\phi \ln(e\mu\phi)$.*

Proof. We assume that $p \geq e\phi \ln(e\mu\phi)$ and prove that $p \geq \phi \ln(\mu p)$. First, consider the case $\mu\phi < 1$. In this case, $p/\phi > \mu p \geq \ln(\mu p)$. Therefore, $p \geq \phi \ln(\mu p)$. Next, suppose $\mu\phi \geq 1$. Define the function $f(x) := x/\log(\mu x)$, and note that it is monotone increasing for $x \geq e/\mu$. By the assumption, we have $p \geq e\phi \ln(e\mu\phi)$. In addition, $\phi \geq 1/\mu$, hence $e\phi \ln(e\mu\phi) \geq e\phi \geq e/\mu$. Therefore, $f(p) \geq f(e\phi \ln(e\mu\phi))$, and we can conclude that

$$\frac{p}{\ln(\mu p)} \equiv f(p) \geq f(e\phi \ln(e\mu\phi)) \equiv \frac{e\phi \ln(e\mu\phi)}{\ln(e\mu\phi \ln(e\mu\phi))} \geq \frac{e\phi \ln(e\mu\phi)}{2\ln(e\mu\phi)} \geq e\phi/2 \geq \phi.$$

Note that we used the fact $\ln(x \ln(x)) \leq 2\ln(x)$, which follows since for any x, $\ln(x) \leq x$. This proves the claim. □

G. Tightening Δ_v using empirical Bernstein bounds

We give a tighter definition of Δ_v, using the empirical Bernstein bound of Maurer and Pontil (2009). This tighter definition does not change the analysis, but can improve the empirical behavior of the algorithm, by allowing it to require weight queries of fewer examples in some cases. The empirical Bernstein bound states that for i.i.d. random variables $Z_1, \ldots, Z_m$ such that $\mathbb{P}[Z_i \in [0,1]] = 1$, with a probability $1 - \delta$,

$$|\mathbb{E}[Z_1] - \frac{1}{m}\sum_{i\in[m]} Z_i| \leq \sqrt{8V_m \ln(2/\delta)/m} + 14\ln(2/\delta)/(3(m-1)), \tag{8}$$

where $V_m := \frac{1}{m(m-1)} \sum_{1\leq i<j\leq m}(Z_i - Z_j)^2$. The following lemma derives the resulting bound. The proof is similar to the proof of Lemma 3.1, except that it uses the bound above instead of Hoeffding's inequality.

Lemma G.1. *Consider the same definitions and notations as in Lemma 3.1. Let*

$$V := \frac{1}{m(m-1)} \sum_{1\leq i<j\leq m} (|Z_i - W| - Z_i - |Z_j - W| + Z_j)^2.$$

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

$$|\mathbb{D}(\mathbf{w}) - \hat{\mathbb{D}}(\mathbf{w})| \leq n\sqrt{8V \ln(2/\delta)/m} + 28\|\mathbf{w}\|_1 \ln(2/\delta)/(3(m-1)).$$

Proof. Let $Z'_i = |Z_i - W| - Z_i$. If $Z_i \geq W$, then $Z'_i = W$. Otherwise, we have $Z_i \leq W$, in which case $Z'_i = W - 2Z_i \geq -W$. Therefore, $\mathbb{P}[Z'_i \in [-W, W]] = 1$. Thus, applying Eq. (8) and normalizing by $2W$, we get that with a probability $1 - \delta$,

$$|\mathbb{E}[Z'_1] - \frac{1}{m}\sum_{i\in[m]} Z'_i| \leq \sqrt{8V\ln(2/\delta)/m} + 28W\ln(2/\delta)/(3(m-1)).$$

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

$$\left|\mathbb{D}(\mathbf{w}) - \|\mathbf{w}\|_1 - (\hat{\mathbb{D}}(\mathbf{w}) - \|\mathbf{w}\|_1)\right| \leq n\sqrt{8V\ln(2/\delta)/m} + 28nW\ln(2/\delta)/(3(m-1)).$$

By noting that $nW = \|\mathbf{w}\|_1$, this completes the proof. □

The tighter definition of Δ_v is obtained by taking the minimum between this bound and the one in Lemma 3.1. Thus, we set

$$\Delta_v := \min(w^*_v \cdot \sqrt{2\ln(2K\pi^2 M_v^2/(3\delta))/M_v}, N_v\sqrt{8V\ln(2/\delta)/M_v} + 28w^*_v\ln(2/\delta)/(3(M_v-1))).$$

The entire analysis is satisfied also by this new definition of Δ_v. Its main advantage is obtaining a smaller value when V is small. This may reduce the number of weight queries required by the algorithm in some cases.

H. Full experiment results

In this section, we provide the full results and details of all the experiments described in Section 6. Implementation in python of AWP and of all the experiments, as well as the data files containing the raw results, are provided in the supplementary material. For each experiment, we report the average normalized output distance over 10 runs, as a function of the pruning size. Error bars, represented by shaded regions, represent the maximal and minimal normalized disrepancies obtained in these runs. Note that the error bars are sometimes too small to observe, in cases where the algorithms behave deterministically or very similarly in different runs of the same experiment.

Figure 5 provides the full results for the experiments on the Adult data set. We give here more details on the procedure which we used to create the hierarchical tree: we started with a tree that includes only the root node, and then iteratively selected a random node to split and a random attribute to use for the split. For numerical attributes, the split was based on a threshold corresponding to the median value of the attribute. For discrete attributes, the attribute values were divided so that the split is fairly balanced. We generated several target distributions by partitioning the data set into ordered bins, where in each experiment the partition was based on the value of a different attribute. The tested attributes were all discrete attributes with a small number of possible values: "occupation", "relationship", "marital status" and "education-num". For the last attribute, all values up to 8 were mapped to a single bin and similarly for all values from 14 and above, to avoid very small bins. We then allocated the target weight to each bin so that each example in a given bin is N times more heavier than each example in the next bin. We tested $N = 2, 4$, which appear in the left and right columns of Figure 5, respectively. It can be seen in Figure 5, that except for a single configuration ($N = 2$ and the "relationship" attribute), AWP always performs better than the baselines.

We now turn to the visual data sets. In all these data sets except for MNIST, images were resized to a standard 224×224 size and transformed to grayscale. Figure 6 provides the full results for first experiment on the MNIST and Caltech256 data sets. In this experiment, the examples were divided into 10 bins by image brightness, and weights were allocated such that the weight of an example is N times heavier than an example in the next bin. The plots show results for $N = 2$ (left) and for $N = 4$ (right). The top row gives the results for MNIST and the bottom row gives the results for Caltech256. Here too, it can be seen that AWP obtains significantly better approximations of the target.

Figure 7 provides the results for the MNIST data set for bins allocated by class, using the same scheme of weight allocation for each bin as in the previous experiment. Results for $N = 2$ (left column) and $N = 4$ (right column) are reported for three random bin orders. Figure 8 provides the results of an analogous this experiment for the Caltech256 data set. For this data set, the 10 bins were generated by randomly partitioning the 256 classes into 10 bins with (almost) the same number of classes in each. The allocation of classes to bins and their ordering, for both data sets, are provided as part of the submitted code. It can be seen that AWP obtains an improvement over the baselines in the MNIST experiments, while the Caltech256 experiment obtains about the same results for all algorithms, with a slight advantage for AWP.

Figure 9 and Figure 10 provides the results of the experiments with the data set pairs. In all experiments, the input data set was Caltech256. In each experiment, a different target data set was fixed. The weight of each Caltech256 example was set to the fraction of images from the target data set which have this image as their nearest neighbor. The target data sets were the Office dataset (Saenko et al., 2010), out of which the 10 classes that also exist in Caltech256 (1410 images) were used, and The Bing dataset (Alessandro Bergamo, 2010a;b), which includes 300 images in each Caltech256 class. For Bing, we also ran three experiments where images from a single super-class from the taxonomy in Griffin et al. (2007) were used as the target data set. The super-classes that were tested were "plants" "insects" and "animals". The classes in each such super-class, as well as those in the Office data set, are given in Table 1. In these experiments as well, the advantage of AWP is easily observed.

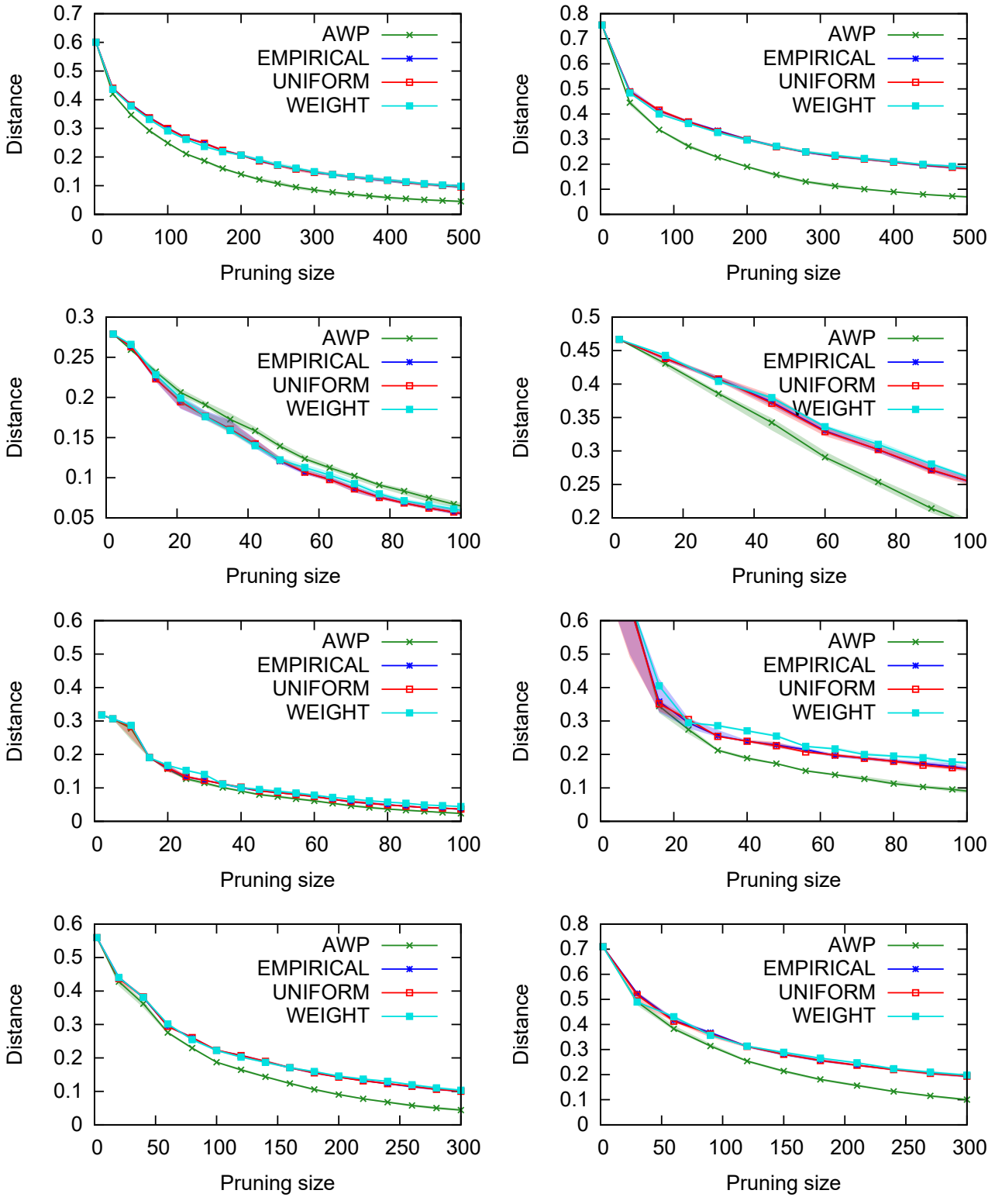

Figure 5. The Adult dataset experiments. Each row report the results for experiments on a different parameter in the following order: "occupation", "relationship", "marital-status", "education-num". Left: $N = 2$. Right: $N = 4$.

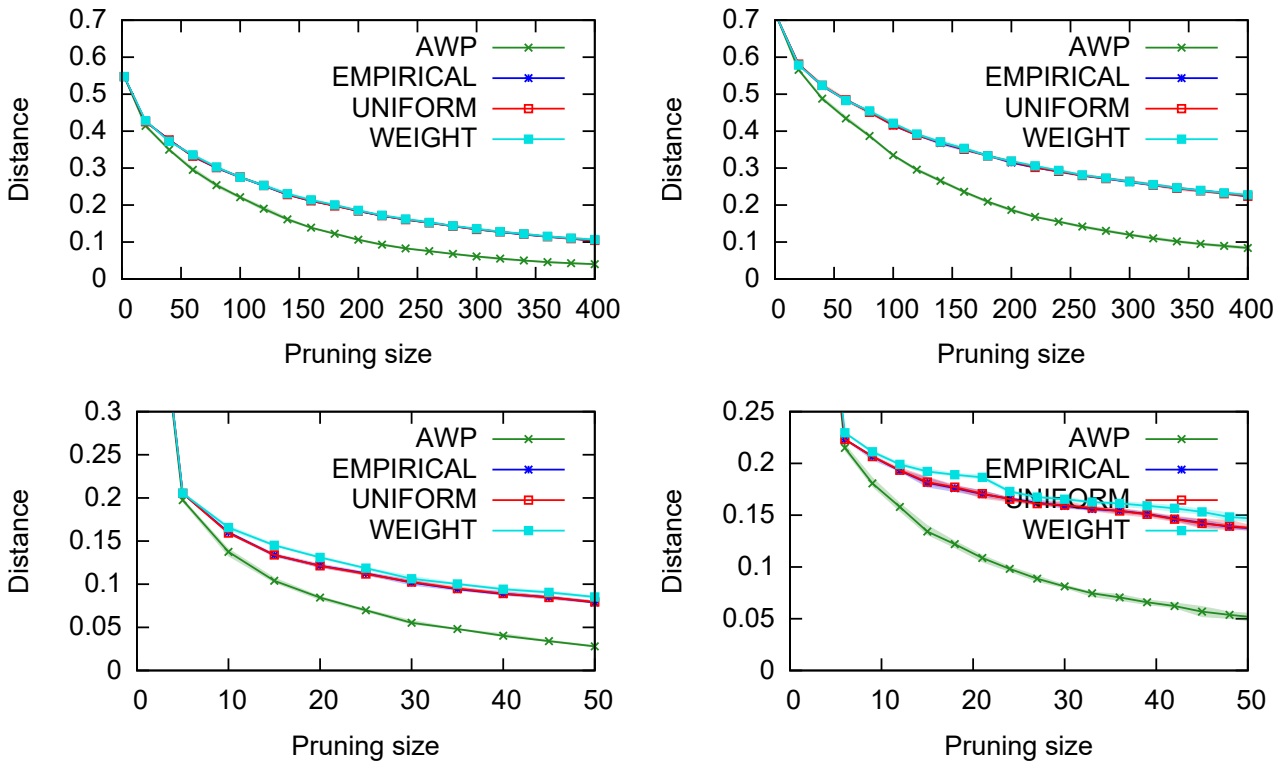

Figure 6. Experiments with bins allocated by brightness. The weight of an example is N times heavier than an example in the next bin. Left: $N = 2$, right: $N = 4$. Top: MNIST, Bottom: Caltech.

Table 1. The classes in each of the super-classes used in the reported experiments. The numbers in parentheses refer to the number of classes in the super-class.

Office (10)	Plants (10)	Insects (8)	Animals (44)			
backpack	palm tree	butterfly	bat	hummingbird	horse	horseshoe-crab
touring-bike	bonsai	centipede	bear	owl	iguana	crab
calculator	cactus	cockroach	camel	hawksbill	kangaroo	conch
headphones	fern	grasshopper	chimp	ibis	llama	dolphin
computer keyboard	hibiscus	house fly	dog	cormorant	leopards	goldfish
laptop	sun flower	praying-mantis	elephant	duck	porcupine	killer-whale
computer monitor	grapes	scorpion	elk	goose	raccoon	mussels
computer mouse	mushroom	spider	frog	iris	skunk	octopus
coffee mug	tomato		giraffe	ostrich	snail	starfish
video projector	water melon		gorilla	penguin	toad	snake
			greyhound	swan	zebra	goat

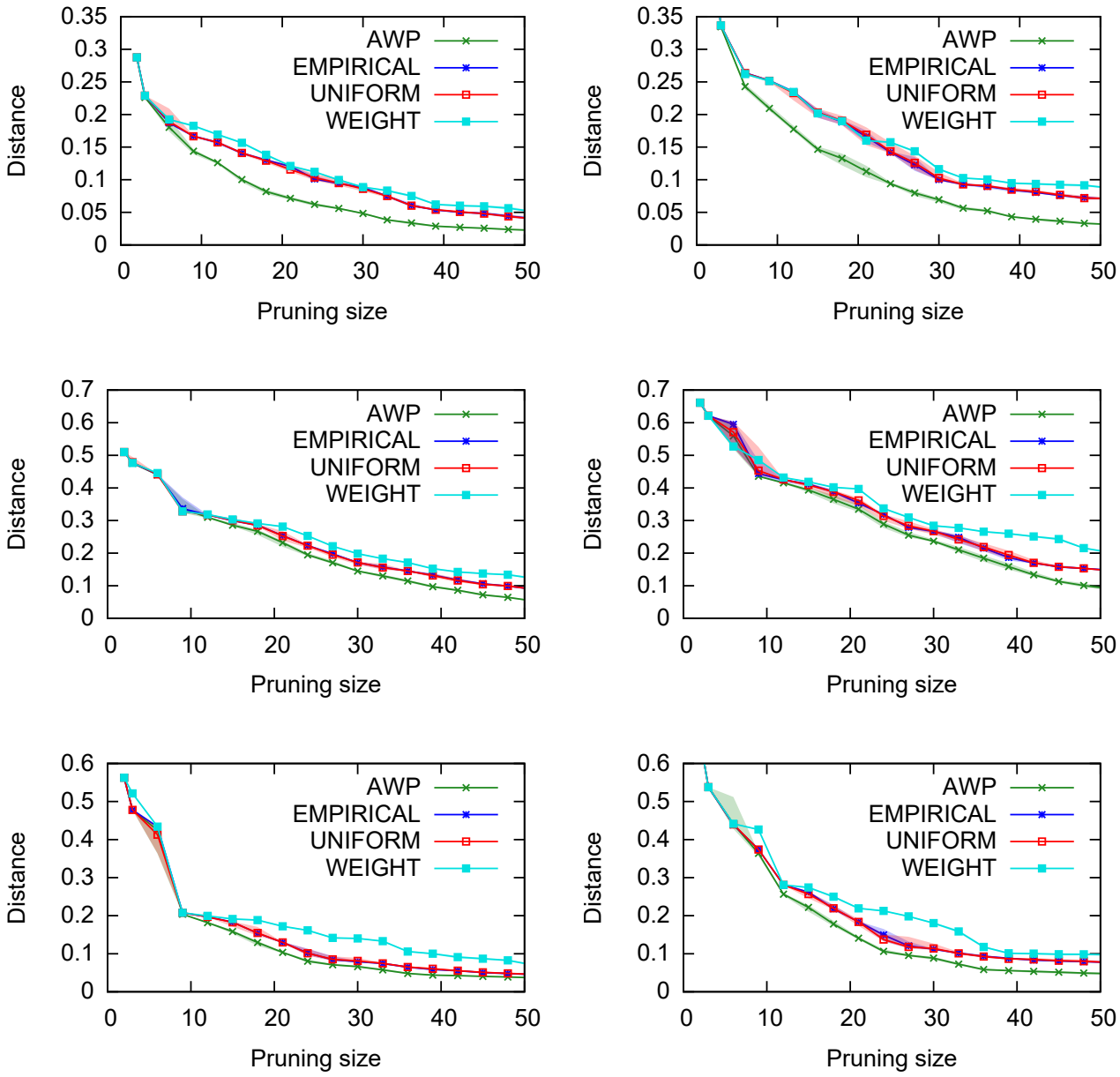

Figure 7. Experiments on the MNIST data set, with bins allocated by class. The weight of an example is N times heavier than an example in the next bin. The three plots in each column show results for three random orders of classes. The left column shows $N = 2$, and the right column shows $N = 4$, for the same class orders.

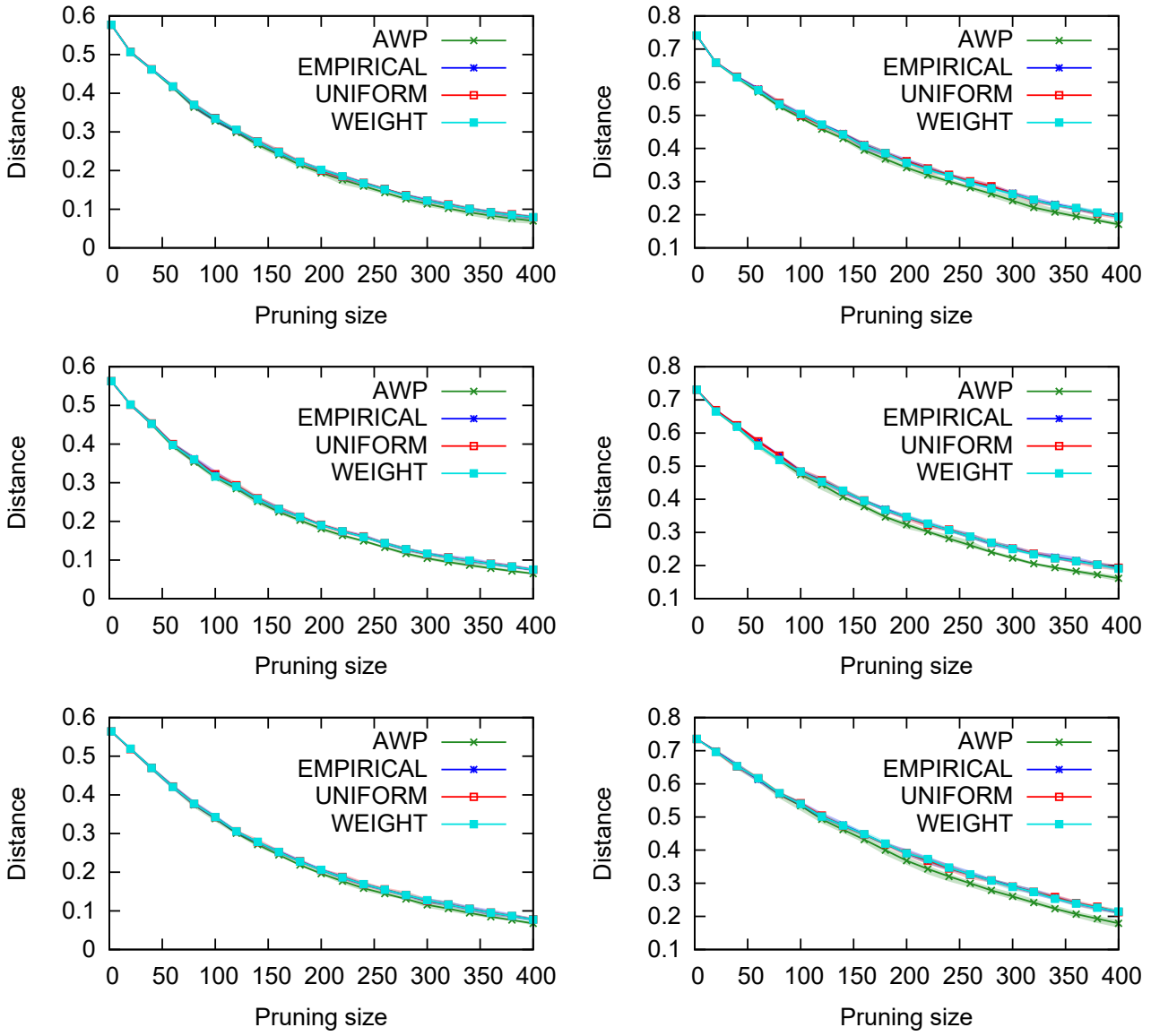

Figure 8. Experiments on the Caltech256 data set, with bins allocated by class. The weight of an example is N times heavier than an example in the next bin. The three plots in each column show results for three random orders of classes. The left column shows $N = 2$, and the right column shows $N = 4$.

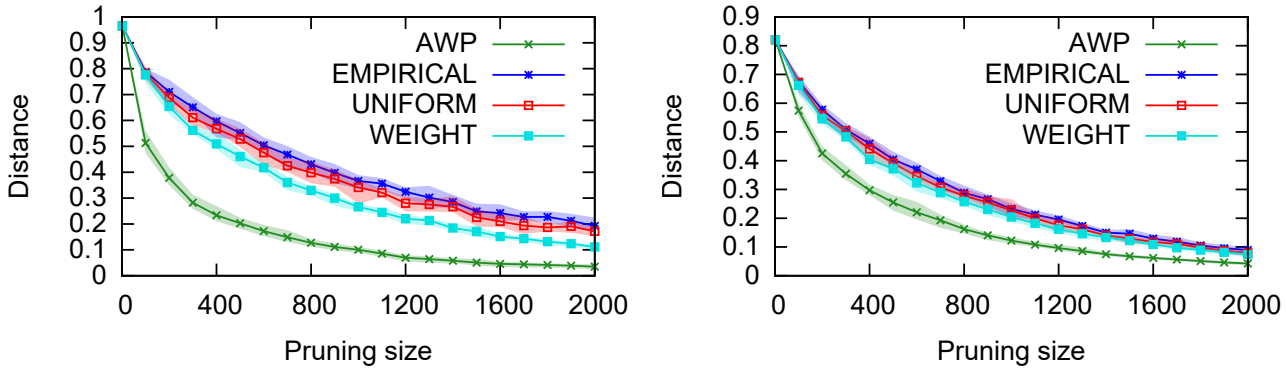

Figure 9. Experiments in which the input data set was Caltech256 and the target weights were calculated using other data sets. Left: the Office data set. Right: the Bing dataset.

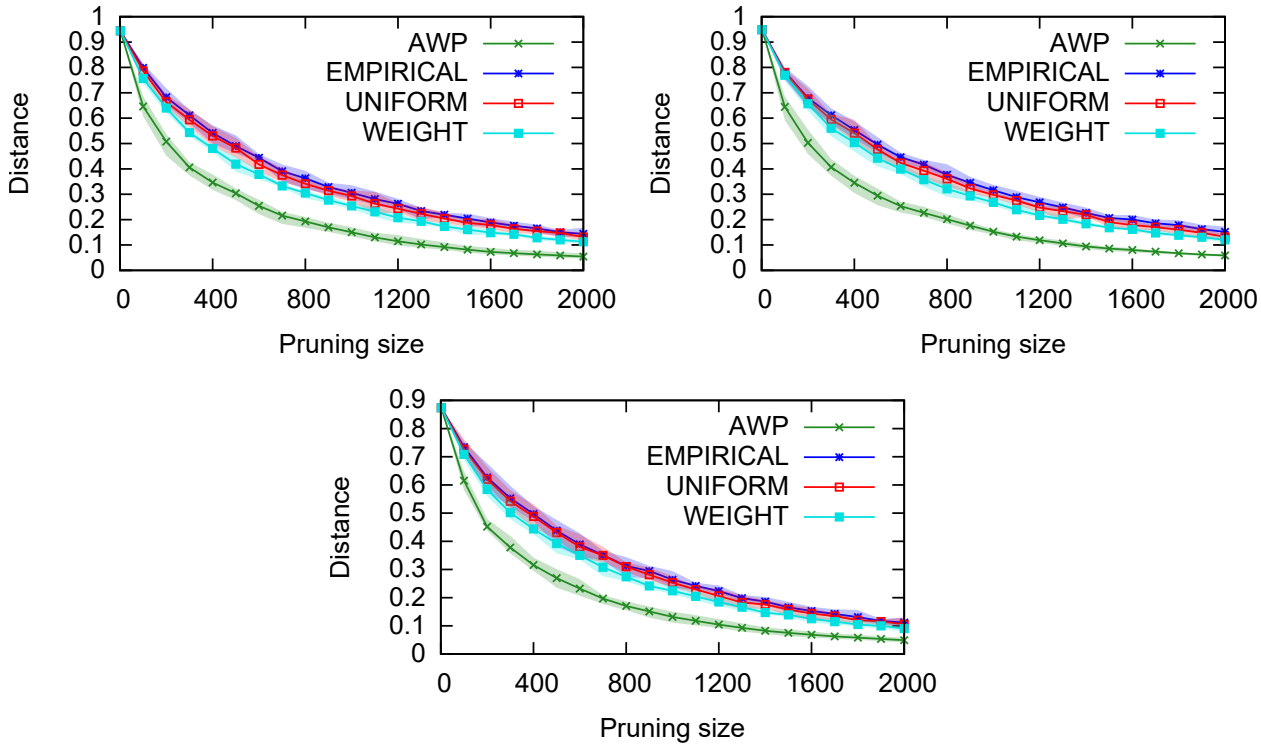

Figure 10. Experiments in which the input data set was Caltech256 and the target weights were calculated using super classes from the Bing dataset. Top Left: The "plants" super class. Top Right: The "insects" super class. Bottom: The "animals" super classes. See Table 1 for details on each super class.

I. Average split quality in experiments

As mentioned in Section 6, in our experiments, the split quality q was usually 1 or very close to one. This shows that AWP can be successful even in cases not strictly covered by Theorem 4.2. To gain additional insight on the empirical properties of the trees used in our experiments, we calculated also the average split quality of each tree, defined as the average over the set of values $\{\max(\mathbb{D}_{v_R}, \mathbb{D}_{v_L})/\mathbb{D}_v \mid v \in T,\ \mathbb{D}_v > 0\}$. The resulting values for all our experiments are reported in Table 2.

Table 2. The average split quality in each of the experiments. Left: experiments with target weight set by bins. In the "classes" experiments, the reported value is an average of the three tested random configurations. Right: domain adaptation experiments with Caltech as the input data set and another data set as the target distribution.

Data set	Binning criterion	Average split quality	
		$N = 2$	$N = 4$
Adult	"occupation"	0.574	0.544
Adult	"relationship"	0.618	0.541
Adult	"marital-status"	0.608	0.569
Adult	"education-num"	0.587	0.547
MNIST	classes	0.699	0.589
MNIST	brightness	0.677	0.592
Caltech	classes	0.645	0.652
Caltech	brightness	0.677	0.641

Target Data set	Average split quality
Office	0.763
Bing	0.756
Bing (plants)	0.751
Bing (insects)	0.755
Bing (animals)	0.762