

Parameterized Complexity of Small Decision Tree Learning

Code: github.com/sueszli/optimal-tree-solver

Yahya Jabary, TU Wien

Source: A Ordyniak, S., & Szeider, S. (2021). Parameterized Complexity of Small Decision Tree Learning. Proceedings of the AAAI Conference on Artificial Intelligence, 35 (7), 6454-6462. <https://www.ac.tuwien.ac.at/files/tr/ac-tr-21-002.pdf>.

motivation

- small decision trees = fewer tests, easier to interpret, generalize better - but expensive to train (np-hard)
- parametrized complexity = input as problem parameters that provide runtime guarantees

problem

- DTS/DTD minimum decision tree size/depth
- decision problem, but our algorithms also return the solution tree

preliminaries

- sol - solution size = can be s, d
- s - size = count of non-leaf nodes
- d - depth = longest root-to-leaf path
- F , $feat(E)$ - features = can have a domain value, finite
- D - domain values = some integers, possibly infinite
- $D_E(f)$ - feature domain values = all mappings to that feature from all examples
 - $D_E(f) = \{e(f) \mid e \in E\}$
- D_{\max} - maximum domain size = maximum size of $D_E(f)$ for any feature
 - $D_{\max} = \max_{f \in feat(E)} |D_E(f)|$
- E - examples, classification instance = mapping of domain values D to features f
 - E^+, E^- = positive or negative examples
 - all examples have the same features $\forall e_i, e_j \in E : feat(e_i) = feat(e_j)$
 - all examples have a unique label (disjoint/non-overlapping union of labels)
 - uniform examples = all have the same labels
- $E[\alpha]$ - agreeing examples = query of examples with the same values for features as in the mapping α
 - $E[\alpha] = \{e \mid e(f) = \alpha(f) \wedge f \in F'\}$ where $\alpha : F' \mapsto D$
- $\delta(e, e')$ - difference = set of features that two examples have different feature values for / disagree on
 - $\delta(e_1, e_2) = \{f : e_1(f) \neq e_2(f)\}$ = hamming distance of two examples
- δ_{\max} - maximum difference = maximum difference between any two examples with different labels
 - $\delta_{\max}(E) = \max_{e^+ \in E^+ \wedge e^- \in E^-} |\delta(e^+, e^-)|$
- S - support set = set of features that can distinguish between (some but not all) positive and negative examples
 - any two examples with different labels must disagree in at least one feature of S
 - $\forall e_1 \in E^+, e_2 \in E^- : \exists f \in S : \delta(e_1, e_2) \neq \emptyset$
- R - additional set = helps support set to distinguish between all positive and negative examples
 - $R = feat(T) \setminus S$ of an optimal tree T using support set S
- T - decision tree = unbalanced binary tree $T = (V, A)$ that partitions decision space
 - has vertices/tests, arcs/edges
 - each inner node v has a feature $feat(v)$ and threshold value $\lambda(v)$ assigned to it
 - correctly classifies every example
- pseudo tree = balanced binary tree, no features and thresholds, can't correctly classify examples
 - if a valid threshold γ can be found for a pseudo tree with assigned features (T, α) , then it can be extended to a valid decision tree
- $\alpha : v(T) \mapsto feat(E)$ - feature assignment function = defines test features $feat(T)$ in tree
 - this overloads alpha which previously mapped values to features, not features to decision nodes
- $\gamma : v(T) \mapsto D_E(\alpha(v))$ - threshold assignment function = defines test thresholds λ in tree

observation 1

- intuition: trees use some support set S for their test nodes $feat(T)$
- proof: if examples with different labels wouldn't disagree on a value, they would end in the same leaf and the tree would be invalid

hardness results

theorem 2 reduces the hitting set problem HS to DTS/DTD and shows that $\{sol, D_{\max}\}$ alone do not yield FTP tractability, even if all the features are booleans.

however, when the hitting set has a bounded size, the problem becomes FTP tractable. this is equivalent to δ_{\max} in our original problem and is naturally small for most datasets (see table 1). this approach to finding parameters is called “deconstruction of intractability”.

theorem 8 confirms this finding by showing the final algorithm.

for the sake of simplicity most proofs only mention the DTS problem.

complexity landscape

- we can assume the problem parameters to be small
- we assume $|E| = |E| \cdot (|feat(E)| + 1) \cdot \log D_{\max}$
- FTP tractable = $\{sol, \delta_{\max}, D_{\max}\}$
 - runtime is exponential in a function of the problem params, polynomial in the input size
 - runtime is bounded by $f(k) \cdot n^{O(1)}$, where f is any computable function of parameter k and n is the input size
 - author’s assume D_{\max} might be redundant
 - sol - solution size, can be s or d
- XP tractable = $\{sol\}, \{sol, \delta_{\max}\}, \{sol, D_{\max}\}$
 - the degree of the polynomial can depend on the parameter
 - runtime is bounded by $n^{f(k)}$, where f is any computable function of parameter k and n is the input size
- w[2] tractable = $\{sol\}, \{sol, D_{\max}\}$
 - problem can be solved by a circuit with t layers of gates with many inputs
 - strong evidence that a problem is not FTP
- paraNP hard = $\emptyset, \{\delta_{\max}\}, \{D_{\max}\}, \{\delta_{\max}, D_{\max}\}$
 - problem remains NP-hard even when the parameters are fixed to a constant

theorem 2

- intuition: $\{sol, D_{\max}\}$ does not yield FTP tractability, even if all the features are booleans – because it’s W[2] tractable
- proof: you can reduce the hitting set problem HS to the optimal decision tree problem DTS/DTD
- any valid decision tree must distinguish the single positive example from all negative examples with a single split. this forces it to identify all features that “hit” all the negative examples

hitting set problem:

- in some universe U of elements, given a collection of sets \mathcal{F} – find a subset H containing at most k elements that intersect with every single set in the collection $\forall F \in \mathcal{F} : F \cap H \neq \emptyset$
 - U = universe of all possible elements
 - $\mathcal{F} \subseteq U$ = collection of sets
 - $H \subseteq U$ = hitting set of size of max size k
 - Δ - maximum arity = size of the largest set $F \in \mathcal{F}$
- reduction: hitting set \rightarrow decision tree:
 - reduction in polynomial time, preserves parameter k (hitting set size becomes solution size)
 - elements in the universe - are represented as features
 - collection of sets - are represented as examples, using boolean flags to encode whether an element from the universe belongs to that set
 - steps to generate $E(\mathcal{I})$ from hitting set instance \mathcal{I} :
 - * i. create the empty set \emptyset – set all feature flags to **false**, set label to **positive**
 - * ii. create the collection of sets – set feature flag based on belonging, set label to **negative**

theorem 3

- intuition: $\{\delta_{\max}(E)\}$ does not yield FTP tractability, even if all the features are booleans – because it’s paraNP-hard tractable
 - proof: $\delta_{\max}(E(\mathcal{I})) = \Delta_{\max}(\mathcal{I})$
 - the highest number of features two examples with a different classification can disagree on, is equivalent to the size of the largest set in the collection
 - the only positive example has all zeros, each negative example has ones exactly in positions corresponding to elements of its set F . therefore, the number of disagreements between the only positive and any negative examples equals the size of set F
- intuition: $\{\min_{\#}(E)\}$ does not yield FTP tractability – because it’s paraNP-hard tractable
 - where: $\min_{\#}(E) = \min\{|E^+|, |E^-|\}$

- proof: in our reduction $\min_{\#}(E(\mathcal{I})) = 1$ because we have a single positive example
- intuition: num of inner nodes does not yield FTP tractability – because it's paraNP-hard tractable
 - proof: due to $\min_{\#}(E(\mathcal{I})) = 1$ we have 0 branching nodes, just two leafs

algorithm

the hardness of decision tree learning comes primarily from feature selection, rather than training

stage 2: training

theorem 4

- intuition: we can compute an optimal decision tree that only uses the given support set S features in $O(2^{\mathcal{O}(s^2)} \|E\|^{1+o(1)} \log \|E\|)$
- runtime
 - the runtime of lemma 6 dominates that of lemma 5 when they're multiplied
 - we enumerate all (T, α) pairs, then search for a valid γ recursively, starting from the root node
 - monotonicity property of thresholds = it suffices to find the maximum threshold $t \in D_E(f)$ for each node, which can be done with binary search
- proof: **findTH** algorithm

lemma 5

- intuition: we can enumerate all (pseudo tree T , feature assignment α) pairs where the assigned features are from the support set S in $O(s^s)$
- the number of trees we have to consider is defined by k which is bounded by the solution size $k = |S| \leq s$
- this means we can just enumerate/brute force all tree structures and feature assignments to nodes, but not all possible thresholds - that will need a heuristic

lemma 6

- intuition: we can find valid threshold assignments γ for (pseudo tree T , feature assignment α) pairs in $O(2^{\mathcal{O}(s^2)} \|E\|^{1+o(1)} \log \|E\|)$ where $d \leq s$
- = number of recursive findTH calls · runtime of each call
 - number of recursive calls = $O(\log \|E\|^d)$ because it calls it self at most $\log \|E\| + 2$ times due to binary search
 - runtime of each call = $O(\|E\| \log \|E\|)$

theorem 7

- intuition: without any feature selection, just by enumerating the $O(|\text{feat}(E)|^s)$ possible support sets, we can solve the entire problem in $O(|\text{feat}(E)|^s \cdot 2^{\mathcal{O}(s^2)} \|E\|^{1+o(1)} \log \|E\|)$
- this runtime is XP-tractable parametrized by s , but not FPT tractable because the degree of the polynomial depends on parameter s

stage 1: feature selection

theorem 8

- intuition: $\{sol, \delta_{\max}, D_{\max}\}$ yields FTP tractability
- proof: **minDT** algorithm

corollary 9

- intuition: we can enumerate all minimal support sets S that are smaller than k in $O(\delta_{\max}(E)^k \cdot |E|)$
- proof: this is the runtime of the hitting set $O(\Delta^k \cdot |F|)$
- however, finding minimal support sets isn't sufficient as shown in lemma 10

lemma 10

- intuition: all test features $\text{feat}(T)$ are a support set S , but not all support sets are valid test features for optimal trees
- optimal decision trees need additional features beyond minimal support sets
- proof: counter example in figure 3

lemma 11

- intuition: additional features $R = \text{feat}(T) \setminus S$ of an optimal tree T using support set S are useful
- additional features separate examples that look identical when only looking at the support features (= equivalence class)

definition of “usefulness”:

- $\forall \beta : R \rightarrow D, \exists \alpha : S \rightarrow D$ such that:
 - $E[\alpha] \neq \emptyset$ (examples exist that match α)
 - $E[\alpha \cup \beta] = \emptyset$ (no examples match both α and β)
- where:
 - $E[\alpha] = \{e \in E \mid \forall f \in S : e(f) = \alpha(f)\}$
 - $E[\alpha \cup \beta] = \{e \in E \mid \forall f \in S : e(f) = \alpha(f) \wedge \forall f \in R : e(f) = \beta(f)\}$
- for every possible “query” of examples using with R , one can exclude all examples for at least one “query” with S , allowing those to be partitioned

proof by contradiction:

- assume: R is not empty, otherwise proof is trivial
- If the lemma were false it would imply:
 - $\exists \beta : R \rightarrow D, \forall \alpha : S \rightarrow D$ such that $E[\alpha \cup \beta] \neq \emptyset$ (some examples match both α and β).
- construction of T' :
 - i.) for nodes in T with features in R , recursively prune one child (left if $\beta(f) > \lambda$, else right). after removing those subtrees, T'' is formed where each node with R features has only one child.
 - ii.) contract maximal paths of R nodes, making those paths into single edges.
 - T' ends up with non-leaf nodes only from S , each having two children.
 - if T' is a valid decision tree, it contradicts our assumption that T had minimum size.
- correctness of T' :
 - assuming T' were invalid, we'd find a leaf $l' \in T'$ containing both positive (e^+) and negative (e^-) examples. these examples would come from different assignments α^+ and α^- over the feature set S .
 - now, in the original tree T , the path to a leaf is determined by both S and R features. but because we're assuming the lemma is false, this means there's an assignment β for R that doesn't split any equivalence classes of S .
 - because of this β , the examples that reached the mixed leaf l' in T' would also end up in the same leaf l in T . why? because β doesn't cause any additional splits.
 - since assuming T' is invalid leads to a contradiction (it would make T invalid too), we must conclude that T' is actually valid.
- assuming R is not useful leads to a valid decision tree T' that is smaller $|T'| < |T|$, which contradicts the minimality of T . this forces the conclusion that R must be useful, ensuring that no smaller valid decision tree exists.

lemma 12

- intuition: for any way you assign values to R , there's always one feature in R that breaks some equivalence class
- $\forall \beta : R \mapsto D : R \cap \{\delta(e, \beta) \mid e \in E(S)\} \neq \emptyset$
- $E(S)$ = exactly one example from each possible query α that has a non empty result $E[\alpha] \rightarrow$ because the same values assigned to a support set feature $f \in S$ might match multiple examples
- $\delta(e, \beta)$ = set of features where the example and the value assignment to R have different values

lemma 13

- intuition: copying an arbitrary example $e \in E : \gamma(f) := e(f)$ to construct an assignment function $\gamma : \text{feat}(E) \mapsto D$ limits the number of features any example can disagree with $\forall e' \in E : \delta(e, \gamma) \leq 2\delta_{\max}(E)$
- proof: for any example $e' \in E$, there exists an example with a different label e'' such that $\delta(e', e'') \leq \delta_{\max}(E)$ (by definition of δ_{\max}). since $\delta(\gamma, e'') \leq \delta_{\max}(E)$, it follows that $\delta(\gamma, e') \leq 2\delta_{\max}(E)$

lemma 14

- intuition: we can compute the branching set R_0 for S that is at most $D_{\max}^{|S|} \cdot 2\delta_{\max}(E)$ large
- R_0 - branching set = contains at least one feature from every possible R there is
 - $R_0 = \{\delta(e, \gamma) \mid e \in E(S)\}$ where γ is from lemma 13
 - by fixing γ as an arbitrary example, disagreements between γ and any $e \in E(S)$ are bounded by $2\delta_{\max}$, and collecting ALL such disagreements across $E(S)$ ensures R_0 contains every feature needed to “break” non-uniform equivalence classes.
- proof:
 - first factor: $|E(S)| \leq D_{\max}^{|S|}$
 - * $D_{\max}^{|S|}$ is an upper bound for the total number of examples you can construct with features from S .
 - * each example in $E(S)$ corresponds to a unique combination of feature values from S .
 - second factor: $2\delta_{\max}(E)$
 - * by copying arbitrary values from examples to $\beta : R \mapsto D$ we can be sure that $\{\delta(e, \beta) \mid e \in E(S)\}$ has at most $2\delta_{\max}(E)$ features