Diagnosis determination: decision trees optimizing simultaneously worst and expected testing cost

Ferdinando Cicalese CICALESE@DIA.UNISA.IT
University of Salerno, Italy

Eduardo Laber LABER@INF.PUC-RIO.BR
PUC-Rio, Brazil

Aline Medeiros Saettler ASAETTLER@INF.PUC-RIO.BR
PUC-Rio, Brazil

Abstract

In several applications of automatic diagnosis and active learning a central problem is the evaluation of a discrete function by adaptively querying the values of its variables until the values read uniquely determine the value of the function. In general reading the value of a variable is done at the expense of some cost (computational or possibly a fee to pay the corresponding experiment). The goal is to design a strategy for evaluating the function incurring little cost (in the worst case or in expectation according to a prior distribution on the possible variables' assignments).

Our algorithm builds a strategy (decision tree) which attains a logarithmic approximation simultaneously for the expected and worst cost spent. This is best possible since, under standard complexity assumption, no algorithm can guarantee $o(\log n)$ approximation.

1. Introduction

An automatic agent that implements a high frequency trading strategy decides the next action to be performed as sending or canceling a buy/sell order, on the basis of some market variables as well as private variables (e.g., stock price, traded volume, volatility, order books distributions as well as complex relations among these variables). For instance in (Nevmyvaka et al., 2006) the trading strategy is learned in the form of a discrete function, described as a table, that has to be evaluated whenever a new scenario is faced and an action (sell/buy) has to be taken. The evaluation can be performed by computing every time the value of each single variable. However, this might be very expensive. By taking into account the structure of the function together with information on the probability distribution on the states of the market and also the fact that some variables are more expensive (or time consuming) to calculate than others, the algorithm can significantly speed up the evaluation of the function. Since market conditions change on a millisecond basis, being able to react very quickly to a new scenario is the key to a profitable strategy.

In a classical Bayesian active learning problem, the task is to select the right hypothesis from a possibly very large set $\mathcal{H} = \{h_1, \ldots, h_n\}$. Each $h \in \mathcal{H}$ is a mapping from a set $\mathcal{X}$ called the query/test space to the set (of labels) $\{1, \ldots, \ell\}$. It is assumed that the functions in $\mathcal{H}$ are unique, i.e., for each pair of them there is at least one point in $\mathcal{X}$ where they differ. There is one function $h^* \in \mathcal{H}$ which provides the correct labeling of the space $\mathcal{X}$ and the task is to identify it through queries/tests. A query/test coincides with an element $x \in \mathcal{X}$ and the result is the value $h^*(x)$. Each test x has an associated cost $c(x)$ that must be paid in order to acquire the response $h^*(x)$, since the process of labeling an example may be expensive either in terms of time or money (e.g. annotating a document). The goal is to identify the correct hypothesis spending as little as possible.

In an example of automatic diagnosis, $\mathcal{H}$ represents the set of possible hypotheses and $\mathcal{X}$ the set of symptoms or medical tests, with h^* being the exact diagnosis that has to be achieved by reducing the cost of the examinations. In (Bellala et al., 2012), a more general variant of the problem was considered where rather than the diagnosis it is important to identify the therapy (e.g., for cases of poisoning it is important to quickly understand which antidote to administer

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

rather than identifying the exact poisoning). This problem can be modeled by defining a partition $\mathcal{P}$ on $\mathcal{H}$ with each class of $\mathcal{P}$ representing the subset of diagnoses which requires the same therapy. The problem is then how to identify the class of the exact h^* rather than h^* itself. This model has also been studied by Golovin et al. (Golovin et al., 2010) to tackle the problem of erroneous tests' responses in Bayesian active learning.

The above examples can all be cast into the following general problem.

The Discrete Function Evaluation Problem (DFEP). An instance of the problem is defined by a quintuple $(S, C, T, \mathbf{p}, \mathbf{c})$, where $S = \{s_1, \ldots, s_n\}$ is a set of objects, $C = \{C_1, \ldots, C_m\}$ is a partition of S into m classes, T is a set of tests, $\mathbf{p}$ is a probability distribution on S, and $\mathbf{c}$ is a cost function assigning to each test t a cost $c(t) \in \mathbb{Q}^+$. A test $t \in T$, when applied to an object $s \in S$, incurs a cost $c(t)$ and outputs a number $t(s)$ in the set $\{1, \ldots, \ell\}$. It is assumed that the set of tests is complete, in the sense that for any distinct $s_1, s_2 \in S$ there exists a test t such that $t(s_1) \neq t(s_2)$. The goal is to define a testing procedure which uses tests from T and minimizes the testing cost (in expectation and/or in the worst case) for identifying the class of an unknown object s^* chosen according to the distribution $\mathbf{p}$.

The DFEP can be rephrased in terms of minimizing the cost of evaluating a discrete function that maps points (corresponding to objects) from some finite subset of $\{1, \ldots, \ell\}^{|T|}$ into values from $\{1, \ldots, m\}$ (corresponding to classes), where each object $s \in S$ corresponds to the point $(t_1(s), \ldots, t_{|T|}(s))$ which is obtained by applying each test of T to s. This perspective motivates the name we chose for the problem. However, for the sake of uniformity with more recent work (Golovin et al., 2010; Bellala et al., 2012) we employ the definition of the problem in terms of objects/tests/classes.

Decision Tree Optimization. Any testing procedure can be represented by a *decision tree*, which is a tree where every internal node is associated with a test and every leaf is associated with a set of objects that belong to the same class. More formally, a decision tree D for $(S, C, T, \mathbf{p}, \mathbf{c})$ is a leaf associated with class i if every object of S belongs to the same class i. Otherwise, the root r of D is associated with some test $t \in T$ and the children of r are decision trees for the sets $\{S_t^1, ..., S_t^\ell\}$, where S_t^i, for $i = 1, \ldots, \ell$, is the subset of S that outputs i for test t.

Given a decision tree D, rooted at r, we can identify the class of an unknown object s^* by following a path from r to a leaf as follows: first, we ask for the result of the test associated with r when performed on s^*; then, we follow the branch of r associated with the result of the test to reach a child r_i of r; next, we apply the same steps recursively for the decision tree rooted at r_i. The procedure ends when a leaf is reached, which determines the class of s^*.

Object	t1	t2	t3	Class	Probability
1	1	1	2	A	0.1
2	1	2	1	A	0.2
3	2	2	1	B	0.4
4	1	2	2	C	0.25
5	2	2	2	C	0.05

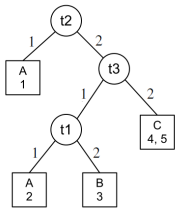

Figure 1. Decision tree for 5 objects presented in the table in the left, with $c(t1) = 2$, $c(t2) = 1$ and $c(t3) = 3$. Letters and numbers in the leaves indicate, respectively, classes and objects.

We define $cost(D, s)$ as the sum of the tests' cost on the root-to-leaf path from the root of D to the leaf associated with object s. Then, the *worst testing cost* and the *expected testing cost* of D are, respectively, defined as

$$cost_W(D) = \max_{s \in S}\{cost(D, s)\} \tag{1}$$

$$cost_E(D) = \sum_{s \in S} cost(D, s)p(s) \tag{2}$$

Figure 1 shows an instance of the DFEP and a decision tree for it. The tree has worst testing cost $1 + 3 + 2 = 6$ and expected testing cost $(1 \times 0.1) + (6 \times 0.2) + (6 \times 0.4) + (4 \times 0.3) = 4.9$.

Our Results. Our main result is an algorithm that builds a decision tree whose expected testing cost and worst testing cost are at most $O(\log n)$ times the minimum possible expected testing cost and the minimum possible worst testing cost, respectively. In other words, the decision tree built by our algorithm achieves simultaneously the best possible approximation achievable with respect to both the expected testing cost and the worst testing cost. In fact, for the special case where each object defines a distinct class—known as the *identification problem*— both the minimization of the expected testing cost and the minimization of the worst testing cost do not admit a sub-logarithmic approximation unless $\mathcal{P} = \mathcal{NP}$, as shown in (Chakaravarthy et al., 2007) and in (Laber & Nogueira, 2004), respectively. In addition, in Section 5, we show that the same inapproximability results holds in general for the case of m classes for any $m \geq 2$.

It shoud be noted that in general there are instances for which the decision tree that minimizes the expected testing cost has worst testing cost much larger than that achieved by the decision tree with minimum worst testing cost. Also there are instances where the converse happens. Therefore,

it is reasonable to ask whether it is possible to construct decision trees that are efficient with respect to both performance criteria. This might be important in practical applications where only an estimate of the probability distribution is available which is not very accurate. Also, in scenarios like the one depicted in (Bellala et al., 2012), very high cost (or time) might have disastrous consequences. In such a case, one could try to minimize the expected testing cost with the additional constraint that the worst testing cost shall not be much larger than the optimal one.

With respect to the minimization of the expected testing cost, our result improves upon the previous $O(\log 1/p_{\min})$ approximation shown in (Golovin et al., 2010) and (Bellala et al., 2012), where p_{min} is the minimum positive probability among the objects in S. From the result in these papers an $O(\log n)$ approximation could be attained only for the particular case of uniform costs via a technique used by Kosaraju in (Kosaraju et al., 1999).

From a high-level perspective, our method closely follows the one used by Gupta *et al.* (Gupta et al., 2010) for obtaining the $O(\log n)$ approximation for the expected testing cost in the identification problem. Both constructions of the decision tree consist of building a path (backbone) that splits the input instance into smaller ones, for which decision trees are recursively constructed and attached as children of the nodes in the path.

A closer look, however, reveals that our algorithm is much simpler than the one presented in (Gupta et al., 2010). First, it is more transparently linked to the structure of the problem, which remained somehow hidden in (Gupta et al., 2010) where the result was obtained via an involved mapping from adaptive TSP. Second, our algorithm avoids expensive computational steps as the Sviridenko procedure (Sviridenko, 2004) and some non-intuitive/redundant steps that are used to select the tests for the backbone of the tree. In fact, we believe that providing an algorithm that is much simpler to implement and an alternative proof of the result in (Gupta et al., 2010) is an additional contribution of this paper.

State of the art. The DFEP has been recently studied under the names of class equivalence problem (Golovin et al., 2010) and group identification problem (Bellala et al., 2012) and long before it had been described in the excellent survey by Moret (B.M.E. Moret, 1982). Both (Golovin et al., 2010) and (Bellala et al., 2012) give $O(\log(1/p_{min}))$ approximation algorithms for the version of the DFEP where the exptected testing cost has to be minimized and both the probabilities and the testing costs are non-uniform. In addition, when the testing costs are uniform both algorithms can be converted into a $O(\log n)$ approximation algorithm via the approach of Kosaraju (Kosaraju et al., 1999). The algorithm in (Golovin et al., 2010) is more general because it addresses multiway tests rather than binary ones. The minimization of the worst testing cost—for which our algorithm provides the best possible approximation—had been left as an open problem in (Bellala et al., 2012).

The particular case of the DFEP where each object belongs to a different class—known as the *identification problem*—has been more extensively investigated (Dasgupta, 2004; Adler & Heeringa, 2008; Chakaravarthy et al., 2007; 2009). Both the minimization of the worst and the expected testing cost do not admit a sublogarithmic approximation unless $\mathcal{P} = \mathcal{NP}$ as proved by (Laber & Nogueira, 2004) and (Chakaravarthy et al., 2007). For the expected testing cost, in the variant with multiway tests, non uniform probabilities and non uniform testing costs, an $O(\log(1/p_{min}))$ approximation is given by Guillory and Blimes in (Guillory & Bilmes, 2009). Gupta *et al.* (Gupta et al., 2010) improved this result to $O(\log n)$ employing new techniques not relying on the Generalized Binary Search (GBS)—the basis of all the previous strategies.

An $O(\log n)$ approximation algorithm for the minimization of the worst testing cost for the identification problem has been given by Arkin et. al. (Arkin et al., 1993) for binary tests and uniform cost and by Hanneke (Hanneke, 2006) for case with mutiway tests and non-uniform testing costs. The worst case cost is also investigated in (Guillory & Bilmes, 2011) under the framework of covering and learning.

2. Preliminaries

Given an instance $I = (S, C, T, \mathbf{p}, \mathbf{c})$ of the DFEP, we will denote by $OPT_E(I)$ ($OPT_W(I)$) the expected testing cost (worst testing cost) of a decision tree with minimum possible expected testing cost (worst testing cost) over the instance I. When the instance I is clear from the context, we will also use the notation $OPT_W(S)$ ($OPT_E(S)$) for the above quantity, referring only to the set of objects involved. We use p_{min} to denote the smallest non-zero probability among the objects in S.

Let $(S, T, C, \mathbf{p}, \mathbf{c})$ be an instance of DFEP and let S' be a subset of S. In addition, let C', $\mathbf{p}'$ and $\mathbf{c}'$ be, respectively, the restrictions of C, $\mathbf{p}$ and $\mathbf{c}$ to the set S'. Our first observation is that every decision tree D for $(S, C, T, \mathbf{p}, \mathbf{c})$ is also a decision tree for the instance $I' = (S', C', T, \mathbf{p}', \mathbf{c}')$. The following proposition immediately follows.

Proposition 1. *Let $I = (S, C, T, \mathbf{p}, \mathbf{c})$ be an instance of the DFEP and let S' be a subset of S. Then, $OPT_E(I') \leq OPT_E(I)$ and $OPT_W(I') \leq OPT_W(I)$, where $I' = (S', C', T, \mathbf{p}', \mathbf{c}')$ is the restriction of I to S'.*

In order to measure the progress of our strategy, we consider the set of objects which satisfy all the tests performed and count the number of pairs of such objects which be-

long to different classes. The following definition formalizes this concept of pairs for a given set of objects.

Definition 1 (Pairs)**.** *Let $I = (S, T, C, \mathbf{p}, \mathbf{c})$ be an instance of the DFEP and $G \subseteq S$. We say that two objects $x, y \in S$ constitute a pair of G if they both belong to G but come from different classes. We denote by $P(G)$ the number of pairs of G. In formulae, we have*

$$P(G) = \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} n_i(G) n_j(G)$$

where for $1 \leq i \leq m$ and $A \subseteq S$, $n_i(A)$ denotes the number of objects in A belonging to class C_i.

As an example, for the set of objects S in Figure 1 we have $P(S) = 8$ and the following set of pairs

$$\{(1,3), (1,4), (1,5), (2,3), (2,4), (2,5), (3,4), (3,5)\}.$$

Let $\mathbf{t}$ be a sequence of tests applied to identify the class of s^* (it corresponds to a path in the decision tree) and let G be the set of objects that agree with the outcomes of all tests in $\mathbf{t}$. If $P(G) = 0$, then all objects in G belong to the same class, which must coincide with the class of the selected object s^*. Hence, $P(G) = 0$ indicates the identification of the class of the chosen object s^*.

For each test $t \in T$ and for each $i = 1, \ldots, \ell$, let $S_t^i \subseteq S$ be the set of objects for which the outcome of test t is i. For a test t, the outcome resulting in the largest number of pairs is of special interest for our strategy. We denote with S_t^* the set among $S_t^1, \ldots, S_t^\ell$ such that $P(S_t^*) = \max\{P(S_t^1), \ldots, P(S_t^\ell)\}$ (ties are broken arbitrarily). We denote with $\sigma_S(t)$ the set of objects not included in S_t^*, i.e., we define $\sigma_S(t) = S \setminus S_t^*$. Whenever S is clear from the context we use $\sigma(t)$ instead of $\sigma_S(t)$.

Given a set of objects S, each test produces a tripartition of the pairs in S: the ones in $\sigma(t)$, those in $S \setminus \sigma(t)$ and those with one element in $\sigma(t)$ and one element in $S \setminus \sigma(t)$. We say that the pairs in $\sigma(t)$ are *kept* by t, the pairs in $S \setminus \sigma(t)$ are *avoided* by t and finally the pairs with one element from $\sigma(t)$ and one element in $S \setminus \sigma(t)$ are *separated* by t. We also say that a pair is *covered* by the test t if it is either kept or separated by t. Analogously, we say that a test t covers an object s if $s \in \sigma(t)$; otherwise, we say that t avoids s.

For any set $Q \subseteq S$ we define $p(Q) = \sum_{s \in Q} p(s)$.

3. Minimization of the Expected Testing Cost

In this section, we describe our algorithm `DecTree` and analyze its performance w.r.t. the expected testing cost. In this section, the term cost is generally intended to mean expected cost.

The concept of the separation cost of a sequence of tests will turn useful for defining and analyzing our algorithm.

The separation cost of a sequence of tests. Given an instance I of the DFEP, for a sequence of tests $\mathbf{t} = t_1, t_2, \ldots, t_q$, we define the separation cost of $\mathbf{t}$ in the instance I, denoted by $sepcost(I, \mathbf{t})$, as follows: Fix an object x. If there exists $j < q$ such that $x \in \sigma(t_j)$ then we set $i(x) = \min\{j \mid x \in \sigma(t_j)\}$. If $x \notin \sigma(t_j)$ for each $j = 1, \ldots, q-1$, then we set $i(x) = q$. Define the *cost of separating x in the instance I by means of the sequence* $\mathbf{t}$ as $sepcost(I, \mathbf{t}, x) = \sum_{j=1}^{i(x)} c(t_j)$.

The *separation cost of* $\mathbf{t}$ (in the instance I) is defined by

$$sepcost(I, \mathbf{t}) = \sum_{s \in S} p(s) sepcost(I, \mathbf{t}, s). \qquad (3)$$

Lower bounds on the cost of an optimal decision tree for the DFEP. We denote by $sepcost^*(I)$ the minimum separation cost in I attainable by a sequence of tests in T which covers all the pairs in S.

The following theorem shows that for any instance $I = (S, C, T, \mathbf{p}, \mathbf{c})$ of the DFEP the minimum separation cost of a sequence of tests covering all pairs in S is a lower bound on the cost of an optimal solution for the instance I.

Theorem 1. *For any instance $I = (S, C, T, \mathbf{p}, \mathbf{c})$ of the DFEP, it holds that $sepcost^*(I) \leq OPT(I)$.*

Proof. (Sketch)

Let D^* be a decision tree for the instance I with expected cost equal to $OPT(I)$.

Let $t_1, t_2, \ldots, t_q, l$ be the root-to-leaf path in D^* such that for each $i = 2, \ldots, q$, the node associated to t_i is on the branch stemming from t_{i-1} which is associated with $S \setminus \sigma(t_{i-1})$, and l is the child of t_q associated wih the objects in $S \setminus \sigma(t_q)$. This is the path followed when the object chosen is some x such that for each test t performed following the strategy D^* we have that $x \notin \sigma(t)$.

It can be shown that $\mathbf{t} = t_1, t_2, \ldots, t_q$ is a sequence of tests covering all pairs of S, hence $sepcost(\mathbf{t}) \geq sepcost^*(I)$ and also $sepcost(I, \mathbf{t}, s) \leq cost(D^*, s)$, for any object s.

These two inequalities together with (3) imply

$$\begin{aligned} sepcost^*(I) &\leq sepcost(I, \mathbf{t}) = \sum_{s \in S} p(s) sepcost(I, \mathbf{t}, s) \\ &\leq \sum_{s \in S} p(s) cost(D^*, s) = OPT(I). \quad \square \end{aligned}$$

The following subadditivity property will be useful.

Proposition 2 (Subadditivity)**.** *Let $S_1, S_2, \ldots, S_q$ be a partition of the object set S. We have $OPT_E(S) \geq \sum_{j=1}^{q} OPT_E(S_j)$, where $OPT_E(S_j)$ is the minimum expected testing cost when the set of objects is S_j.*

The optimization of submodular functions of sets of tests. Let $I = (S, T, C, \mathbf{p}, \mathbf{c})$ be an instance of the DFEP. A set function $f : 2^T \mapsto \mathbb{R}_+$ is submodular non-decreasing if for every $R \subseteq R' \subseteq T$ and every $t \in T \setminus R'$, it holds that $f(R \cup \{t\}) - f(R) \geq f(R' \cup \{t\}) - f(R')$ (submodularity) and $f(R) \leq f(R')$ (non-decreasing).

It is not hard to verify that the functions

$$f_1 : R \subseteq T \mapsto P(S) - P(S \setminus \bigcup_{t \in R} \sigma_S(t))$$

$$f_2 : R \subseteq T \mapsto p(S) - p(S \setminus \bigcup_{t \in R} \sigma_S(t))$$

are non-negative non-decreasing submodular set functions. In words $f_1(R)$ (resp. $f_2(R)$) is the function mapping a set of tests R into the number of pairs (resp. probability) of the set of objects covered by the tests in R

Consider the following optimization problem defined over a non-negative, non-decreasing, sub modular function f:

$$\mathcal{P} : \max_{R \subseteq T} \left\{ f(R) : \sum_{t \in R} c(t) \leq B \right\}. \tag{4}$$

In (Wolsey, 1982), Wolsey studied the solution to the problem $\mathcal{P}$ provided by Algorithm 1 below, called the adapted greedy heuristic.

Algorithm 1 Wolsey greedy algorithm

Procedure `Adapted-Greedy`$(S, T, f, \mathbf{c}, B)$

1: $spent \leftarrow 0,\ A \leftarrow \emptyset,\ k \leftarrow 0$
2: **repeat**
3: $k \leftarrow k + 1$
4: let t_k be a test t maximizing $\frac{f(A \cup \{t\}) - f(A)}{c(t)}$ among all $t \in T$ s.t. $c(t) \leq B$
5: $T \leftarrow T \setminus \{t_k\},\ spent \leftarrow spent + c(t_k),\ A \leftarrow A \cup \{t_k\}$
6: **until** $spent > B$ or $T = \emptyset$
7: **if** $f(\{t_k\}) \geq f(A \setminus \{t_k\})$ **then** Return $\{t_k\}$ **else** Return $\{t_1\, t_2\, \ldots\, t_{k-1}\}$

The following theorem summarizes results from (Wolsey, 1982)[Theorems 2 and 3].

Theorem 2. *(Wolsey, 1982) Let R^* be the solution of the problem $\mathcal{P}$ and $\overline{R}$ be the set returned by Algorithm 1. Let e be the base of the natural logarithm and χ be the solution of $e^\chi = 2 - \chi$. Then we have that $f(\overline{R}) \geq (1 - e^{-\chi}) f(R^*) \approx 0.35 f(R^*)$. Moreover, if there exists c such $c(t) = c$ for each $t \in T$ and c divides B, then we have $f(\overline{R}) \geq (1 - 1/e) f(R^*) \approx 0.63 f(R^*)$.*

Corollary 1. *Let $\mathbf{t} = t_1 \ldots t_{k-1} t_k$ be the sequence of all the tests selected by* `Adapted-Greedy`*, i.e., the concatenation of the two possible outputs in line 7. Then, we have that the total cost of the tests in $\mathbf{t}$ is at most $2B$ and $f(\{t_1, \ldots, t_{k-1}, t_k\}) \geq (1 - e^{-\chi}) f(R^*) \approx 0.35 f(R^*)$.*

We will employ this greedy heuristic for finding approximate solutions to the optimization problem $\mathcal{P}$ over the submodular set functions f_1 and f_2 defined above.

3.1. Achieving logarithmic approximation

We will show that Algorithm 2 attains a logarithmic approximation for DFEP. The algorithm consists of 4 blocks. The first block (lines 1-2) is the basis of the recursion, which returns a leaf if all objects belong to the same class. The second block (line 3) calls procedure `FindBudget` to define the budget B allowed for the tests selected in the third and fourth blocks.

Algorithm 2 Decision tree with cost $O(\log n) OPT_E(I)$

Procedure `DecTree`$(S, T, C, \mathbf{p}, \mathbf{c})$

1: **if** all objects in S belong to the same class **then**
2: Return a single leaf l asssociated with S
3: B=`FindBudget`$(S, T, C, \mathbf{c})$, $spent \leftarrow 0$, $spent_2 \leftarrow 0$, $U \leftarrow S$, $k \leftarrow 1$
4: **while** there is a test in T of cost $\leq B - spent$ **do**
5: let t_k be a test which maximizes $\frac{p(U) - p(U \setminus \sigma_S(t))}{c(t)}$ among all tests t s.t. $t \in T$ and $c(t) \leq B - spent$
6: **If** $k = 1$ **then** make t_1 root of D **else** t_k child of t_{k-1}
7: **for** every $i \in \{1, \ldots, \ell\}$ such that $(S^i_{t_k} \cap U) \neq \emptyset$ **and** $S^i_{t_k} \neq S^*_{t_k}$ **do**
8: Make $D^i \leftarrow$ `DecTree`$(S^i_{t_k} \cap U, T, C, \mathbf{p}, \mathbf{c})$ child of t_k
9: $U \leftarrow U \setminus \sigma_S(t_k),\ spent \leftarrow spent + c(t_k)\ , T \leftarrow T \setminus \{t_k\},\ k \leftarrow k + 1$
10: **end while**
11: **repeat**
12: let t_k be a test which maximizes $\frac{P(U) - P(U \setminus \sigma_S(t))}{c(t)}$ among all tests $t \in T$ s.t. $c(t) \leq B$
13: Set t_k as a child of t_{k-1}
14: **for** every $i \in \{1, \ldots, \ell\}$ such that $(S^i_{t_k} \cap U) \neq \emptyset$ **and** $S^i_{t_k} \neq S^*_{t_k}$ **do**
15: Make $D^i \leftarrow$`DecTree`$(U \cap S^i_{t_k}, T, C, \mathbf{p}, \mathbf{c})$ child of t_k
16: $U \leftarrow U \setminus \sigma_S(t_k),\ spent_2 \leftarrow spent_2 + c(t_k)\ , T \leftarrow T \setminus \{t_k\},\ k \leftarrow k + 1$
17: **until** $B - spent_2 < 0$ **or** $T = \emptyset$
18: $D' \leftarrow$ `DecTree`$(U, T, C, \mathbf{p}, \mathbf{c})$; Make D' a child of t_{k-1}
19: Return the decision tree D constructed by the algorithm

Procedure `FindBudget`$(S, T, C, \mathbf{c})$

1: Let $f : R \in 2^T \mapsto P(S) - P(S \setminus \cup_{t \in R} \sigma(t))$ and let $\alpha = 1 - e^\chi \approx 0.35$
2: Do a binary search in the interval $[1, \sum_{t \in T} c(t)]$ to find the smallest B such that `Adapted-Greedy`$(S, T, f, \mathbf{c}, B)$ returns a set of tests R covering at least $\alpha P(S)$ pairs
3: Return B

The third (lines 4-10) and the fourth (lines 11-17) blocks are responsible for the construction of the backbone of the decision tree (see Fig. 2) as well as to call `DecTree` recursively to construct the decision trees that are children of the nodes in the backbone. The third block (the **while** loop) constructs the first part of the backbone (sequence $\mathbf{t}^A$ in Fig. 2) by iteratively selecting the test that covers

the maximum uncovered mass probability per unit of testing cost (line 5). The selected test t_k induces a partition $(U^1_{t_k}, \ldots, U^\ell_{t_k})$ on the current set of objects U. In lines 7 and 8, the procedure is recursiverly called for each set of this partition but for the one that is contained in the subset $S^*_{t_k}$. With reference to Figure 2, these calls will build the subtrees rooted at the children of the nodes in $\mathbf{t}_A$

Similarly, the fourth block (the **repeat-until** loop) constructs the second part of the backbone (sequence $\mathbf{t}^B$ in Fig. 2) by iteratively selecting the test that covers the maximum number of uncovered pairs per unit of testing cost (line 12). It easy to see that also this procedure is based on the adapted greedy heuristic of Algorithm 1. In fact, here the sequence constructed is the concatenation of the two possible outputs of Algorithm 1. As a consequence, we can apply Corollary 1 to analyze the cost and the coverage of this sequence.

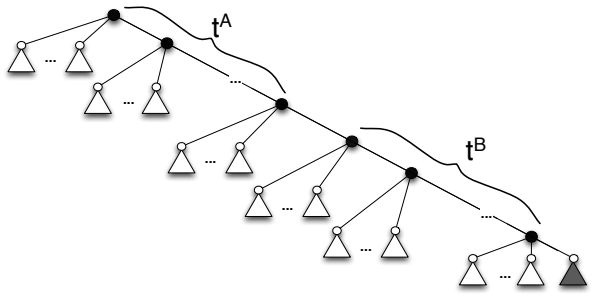

Figure 2. The structure of the decision tree built by `DecTree`: white nodes correspond to recursive calls. In each white subtree, the number of pairs is at most $P(S)/2$, while in the lowest-right gray subtree it is at most $8/9P(S)$ (see the proof of Theorem 4).

Let $\mathbf{t}_I$ denote the sequence of tests obtained by concatenating the tests selected in the **while** loop and in the **repeat-until** loop of the execution of `DecTree` over instance I. We delay to the next section the proof of the following key result.

Theorem 3. *Let χ be the solution of $e^\chi = 2 - \chi$, and $\alpha = 1 - e^\chi \approx 0.35$. There exists a constant $\delta \geq 1$, such that for any instance $I = (S, C, T, \mathbf{p}, \mathbf{c})$ of the DFEP, the sequence $\mathbf{t}_I$ covers at least $\alpha^2 P(S) > \frac{1}{9}P(S)$ pairs and $sepcost(I, \mathbf{t}_I) \leq \delta \cdot sepcost^*(I)$.*

Theorem 4. *For any instance $I = (S, C, T, \mathbf{p}, \mathbf{c})$ of the DFEP, the algorithm `DecTree` outputs a decision tree whose expected testing cost is at most $O(log(n)) \cdot OPT_E(I)$*

Proof. For any instance I, let $D^{\mathbb{A}}(I)$ be the decision tree produced by the algorithm `DecTree`. Let $\mathcal{I}$ be the set of instances on which the algorithm `DecTree` is recusively called in lines 8,15 and 18.

Let β be such that $\beta \log \frac{9}{8} = \delta$, where δ is the constant given in the statement of Theorem 3. Let us assume by induction on the number of pairs that the algorithm guarantees approximation $\beta \log P(G) + 1$ for every instance I' on a set of objects G with $2 \leq P(G) < P(S)$. We have that

$$\frac{cost(D^{\mathbb{A}}(I))}{OPT_E(I)} = \frac{sepcost(I, \mathbf{t}_I) + \sum_{I' \in \mathcal{I}} cost(D^{\mathbb{A}}(I'))}{OPT_E(I)} \quad (5)$$

$$\leq \frac{sepcost(I, \mathbf{t}_I)}{OPT_E(I)} + \max_{I' \in \mathcal{I}} \frac{cost(D^{\mathbb{A}}(I'))}{OPT_E(I')} \quad (6)$$

$$\leq \delta + \max_{I' \in \mathcal{I}} \frac{cost(D^{\mathbb{A}}(I'))}{OPT_E(I')} \quad (7)$$

$$\leq \delta + \max_{I' \in \mathcal{I}} \beta \log P(I') + 1 \quad (8)$$

$$\leq \delta + \beta \log 8P(S)/9 + 1 \quad (9)$$

$$= \beta log(P(S)) + 1. \quad (10)$$

The first equality follows by the recursive way the algorithm `DecTree` builds the decision tree. Inequality (6) follows from (5) by the subadditivity property (Proposition 2) and simple algebraic manipulations. The inequality in (7) follows by Theorem 3 together with Theorem 1 yielding $sepcost(I, \mathbf{t}_I) \leq \delta\, OPT_E(I)$. The inequality in (8) follows by induction (we are using $P(I')$ to denote the number of pairs of instance I').

To prove that the inequality in (9) holds we have to argue that every instance $I' \in \mathcal{I}$ has at most $8/9P(S)$ pairs. Let $U^i_{t_k} = S^i_{t_k} \cap U$ as in the lines 8 and 15. First we show that the number of pairs of $U^i_{t_k}$ is at most $P(S)/2$. In fact, because $S^i_{t_k} \neq S^*_{t_k}$ and $S^*_{t_k}$ is the set in the partition $\{S^1_{t_k}, \ldots, S^\ell_{t_k}\}$, induced by t_k on the set S, with the maximum number of pairs, it follows that $P(U^i_{t_k}) \leq P(S^i_{t_k}) \leq P(S)/2$. Now it remains to show that the instance I', recursively called, in line 18 has at most $8/9P(S)$ pairs. This is true because the number of pairs of I' is equal to the number of pairs not covered by $\mathbf{t}_I$ which is bounded by $(1 - \alpha^2)P(S) \leq 8P(S)/9$ by Theorem 3. □

3.2. The proof of Theorem 3

Given an instance I, for a sequence of tests $\mathbf{t} = t_1, \ldots, t_q$, we will denote by $totcost(I, \mathbf{t})$ the sum of the costs of the tests, i.e., $totcost(I, \mathbf{t}) = \sum_{j=1}^q c(t_j)$.

Due to the space limitations, the proof of the following lemmas is deferred to the full version of the paper.

Lemma 1. *For any instance $I = (S, C, T, \mathbf{p}, \mathbf{c})$, the value B returned by `FindBudget`$(S, T, C, \mathbf{c})$ satisfies $B \leq totcost(I, \tilde{\mathbf{t}})$, for every sequence $\tilde{\mathbf{t}}$ that covers all pairs of I. In addition, the execution of procedure `Adapted-Greedy` over the instance I, with budget B, and $f : R \subseteq S \mapsto P(S) - P(S \setminus \cup_{t \in R} \sigma(t))$ selects a sequence of tests that covers at least $\alpha P(S)$ pairs, where $\alpha = 1 - e^\chi \approx 0.35$ and χ is the solution of $e^\chi = 2 - \chi$.*

Given an instance I, for a sequence of tests $\mathbf{t} = t_1, \ldots, t_k$ and a real $K > 0$, let $sepcost_K(I, \mathbf{t})$ be the separation cost of $\mathbf{t}$ when every non-covered object is charged K, that is,

$$sepcost_K(I, \mathbf{t}) = \sum_{\text{x is covered by } \mathbf{t}} p(x) sepcost(I, \mathbf{t}, x) + \sum_{\text{x is not covered by } \mathbf{t}} p(x) \cdot K$$

Lemma 2. *Let $\mathbf{t}^A$ be the sequence obtained by concatenating the tests selected in the* **while** *loop of Algorithm 2. Then, $totcost(I, \mathbf{t}^A) \leq B$ and $sepcost_B(I, \mathbf{t}^A) \leq \gamma \cdot sepcost^*(I)$, where γ is a positive constant and B is the budget calculated at line 3.*

Lemma 3. *The sequence $\mathbf{t}_I$ covers at least $\alpha^2 P(S)$ pairs and it holds that $totcost(I, \mathbf{t}_I) \leq 3B$.*

The proof of Theorem 3 will now follow by combining the sequences provided by the previous three lemmas.

Proof of Theorem 3. First, it follows from Lemma 3 that $\mathbf{t}_I$ covers at least $\alpha^2 P(S)$ pairs.

To prove that $sepcost(I, \mathbf{t}_I) \leq sepcost^*(I)$, we decompose $\mathbf{t}_I$ into $\mathbf{t}^A = t_1^A, \ldots, t_q^A$ and $\mathbf{t}^B = t_1^B, \ldots, t_r^B$, the sequences of tests selected in the **while** and in the **repeat-until** loop of Algorithm 2, respectively.

For $i = 1, \ldots, q$, let $\pi_i = \sigma(t_i^A) \setminus (\bigcup_{j=1}^{i-1} \sigma(t_j^A))$. In addition, let π_A be the set of objects which are avoided by all the tests in $\mathbf{t}^A$. Thus,

$$\begin{aligned} sepcost(I, \mathbf{t}_I) &\leq \sum_{i=1}^{q} p(\pi_i) \left(\sum_{j=1}^{i} c(t_j^A) \right) + 3B \cdot p(\pi_A) \\ &\leq 3 sepcost_B(I, \mathbf{t}^{\mathbf{A}}) \leq 3\gamma sepcost^*(I), \end{aligned}$$

where the last inequality follows from Lemma 2. The proof is complete.

4. `DecTree` provides a logarithmic approximation for the worst testing cost

In this section, we prove that the `DecTree` algorithm provides $O(\log n)$-approximation (in fact best possible due to Theorem 6) also for the minimization of the worst testing cost.

Theorem 5. *For an instance $I = (S, C, T, \mathbf{p}, \mathbf{c})$ of the DFEP, the algorithm* `DecTree` *outputs a decision tree whose worst testing cost is at most $O(\log n) \cdot OPT_W(I)$.*

Proof. Let D be the decision tree produced by the algorithm `DecTree`. Let D^* be a decision tree such that $cost_W(D^*) = OPT_W(I)$ and $\mathbf{t}^*$ be the sequence of tests on a root-to-leaf path in D^* such that the sequence $\mathbf{t}^*$ covers all the pairs in the instance I (see the proof of Theorem 1 for how to choose such a sequence). We have $totcost(I, \mathbf{t}^*) \leq OPT_W(I)$.

Let $\mathbf{t}$ be a sequence of tests on a root to leaf path in D. The sequence $\mathbf{t}$ is constructed during several recursive calls to the algorithm `DecTree`. Let $\mathbf{t}^{(1)}, \ldots, \mathbf{t}^{(r)}$ be the disjoint subsequences of $\mathbf{t}$ where $\mathbf{t}^{(i)}$ is built in the ith recursive call in which some part of $\mathbf{t}$ is constructed. Let $I^{(i)}$ be the instance on which `DecTree` is called when the sequence $\mathbf{t}^{(i)}$ is constructed. Moreover, let $\mathbf{t}(i)_I$ be the complete sequence built during the recursive call of `DecTree` on $I^{(i)}$ (in particular $\mathbf{t}^{(i)}$ is some prefix of $\mathbf{t}(i)_I$). Let $B^{(i)}$ be the value returned by `FindBudget` on the instance $I^{(i)}$. We have

$$\begin{aligned} totcost(I, \mathbf{t}^{(i)}) &= totcost(I^{(i)}, \mathbf{t}^{(i)}) \leq totcost(I^{(i)}, \mathbf{t}_I^{(i)}) \\ &\leq 3B^{(i)} \leq 3totcost(I^{(i)}, \mathbf{t}^*) \\ &= 3totcost(I, \mathbf{t}^*), \end{aligned}$$

where the first and the last equality follow from the fact that the tests' costs are the same in I and $I^{(i)}$; the first inequality follows from $\mathbf{t}^{(i)}$ being a subsequence of $\mathbf{t}_I^{(i)}$; the second inequality follows from Lemma 3 and the last inequality from Lemma 1, because the sequence $\mathbf{t}^*$ which covers all the pairs in I, also covers, *a fortiori*, all the pairs in the sub-instance $I^{(i)}$. It follows that

$$\begin{aligned} \sum_{t \in \mathbf{t}} c(t) &= \sum_{i=1}^{r} \sum_{t \in \mathbf{t}^{(i)}} c(t) \leq 3r \cdot totcost \leq 3r OPT_W(I) \\ &= O(\log n) OPT_W(I), \end{aligned}$$

where the last inequality follows by noticing that for each $i = 2, \ldots, r$, it holds that the number of pairs in the instance $I^{(i)}$ are not more than $8/9$ of the number of pairs in the instance $I^{(i)}$, as shown in the proof of Theorem 5. Hence we have that $r = O(\log P(S)) = O(\log n)$.

Since $\sum_{t \in \mathbf{t}} c(t) \leq O(\log n) OPT_W(I)$ holds for any $\mathbf{t}$ which is on a root-to-leaf path in D it follows that $cost_W(D) = O(\log n) OPT_W(I)$. □

On the bi-criteria optimization of `DecTree`. By Theorems 4 and 5 we have that algorithm `DecTree` provides simultaneously the best possible approximation for the minimization of expected testing cost and worst testing cost. We would like to remark that this is a very strong feature of our algorithm. In this respect, let us consider the following instance of the DFEP[1]: Let $S = \{s_1, \ldots, s_n\}$; $p_i = 2^{-i}$, for $i = 1, \ldots, n-1$ and $p_n = 2^{-(n-1)}$; the set of tests

[1] This is also an instance of the identification problem

is in one to one correspondence with the set of all binary strings of length n so that the test corresponding to a binary string $\mathbf{b}$ outputs $0(1)$ for object s_i if and only if the ith bit of $\mathbf{b}$ is 0(1). Moreover, all tests have unitary costs. This instance is also an instance of the problem of constructing an optimal prefix coding binary tree, which can be solved by the Huffman's algorithm. Let D_E^* and D_W^* be, respectively, the decision trees with minimum expected cost and minimum worst testing cost for this example. Using Huffman's algorithm, it is not difficult to verify that $Cost_E(D_E^*) \leq 3$ and $Cost_W(D_E^*) = n - 1$. In addition, we have that $Cost_E(D_W^*) = Cost_W(D_W^*) = \log n$. This example shows that the minimization of the expected testing cost may result in high worst testing cost and vice versa the minimization of the worst testing cost may result in high expected testing cost. In cases like this it might provide a significant gain the ability of the algorithm to optimize with respect to both measures of cost.

5. $O(\log n)$ is the best possible approximation.

Let $U = \{u_1, \ldots, u_n\}$ be a set of n elements and $\mathcal{F} = \{F_1, \ldots, F_m\}$ be a family of subsets of U. The minimum set cover problem asks for a family $\mathcal{F}' \subseteq \mathcal{F}$ of minimum cardinality such that $\bigcup_{i \in \mathcal{F}'} F_i = U$. It is well known that this problem does not admit an $o(\log n)$ approximation unless $P = NP$.

First we show that an $o(\log n)$ approximation for the worst case version of the DFEP with exactly b classes for $b \geq 2$, implies the same approximation for the Minimum Set Cover problem so that one cannot expect to obtain a sublogarithmic approximation for the DFEP unless $P = NP$.

We construct an instance for the DFEP as follows. The set of objects is $S = U \cup \{o_1, \ldots, o_{b-1}\}$. All the objects of U belong to class 0 while the object o_i, for $i = 1, \ldots, b-1$, belongs to class i. For each F_i in $\mathcal{F}$ we create a test t_i such that t_i has value 0 for the objects in F_i and value 1 for the remaining objects. In addition, we create a test t_{m+1} which has value 0 for objects in U and value $i-1$ for object o_i. Each test has cost 1.

Let D^* be the decision tree with minimum worst case cost and let $\mathcal{F}^* = \{F_{i_1}, \ldots, F_{i_h}\}$ be a minimum set cover for instance $(U, \mathcal{F})$, where $h = |\mathcal{F}^*|$. Now, we argue that $h \leq OPT_W(D^*) \leq h + 1$.

We can construct a decision tree D of worst testing cost $h+1$ by putting the test t_1^* associated with F_{i_1} in the root of the tree, then the test associated with F_{i_2} as the child of t_1^* and so on. At the bottom of the tree we add the test t_{m+1}. Thus, the worst case cost of D^* is at most $h+1$. On the other hand, let P be the path from the root of the decision tree D^* to the leaf where object o_1 lies. It is easy to realize that the subsets associated with the tests in this path covers all elements in U. This establishes the hardness for the worst case version of the DFEP.

The same reduction works for the expected cost version. In this case, we set the probability of o_1 equals to $1 - (n + b - 2)\epsilon$ and the probability of the other objects equal to ϵ. Thus, we have the following theorem

Theorem 6. *Both the minimization of the worst case and the expected case of the DFEP don't admit an $o(\log n)$ approximation unless $\mathcal{P} = \mathcal{NP}$*

6. Final remarks and Future Directions

We presented a new algorithm for the discrete evaluation problem, a generalization of the classical Bayesian active learning also studied under the names of Equivalence Class Determination Problem (Golovin et al., 2010) and Group Based Active Query Selection problem of (Bellala et al., 2012). Our algorithm builds a decision tree which achieves the best possible approximation simultaneously for the expected and the worst testing cost unles $\mathcal{P} = \mathcal{NP}$. This closes the gap left open by the previous $O(\log 1/p_{\min})$ approximation shown in(Golovin et al., 2010) and (Bellala et al., 2012), where p_{min} is the minimum positive probability among the objects in S. In addition we close the open problem raised by (Bellala et al., 2012) about efficiently approximating the worst testing cost.

In the broader context of machine learning, given a set of samples labeled according to an unknown function, a standard task is to find a good approximation of the labeling function (hypothesis). In order to guarantee that the hypothesis chosen has some generalization power w.r.t. to the set of samples, we should avoid overfitting. When learning is performed via decision tree induction this implies that we shall not have leaves associated with a small number of samples so that we end up with a decision tree that have leaves associated with more than one label. There are many strategies available in the literature to induce such a tree.

In the problem considered in this paper our aim is to overfit the data because the function is known a priori and we are interested in obtaining a decision tree that allows us to identify the label of a new sample with the minimum possible cost (time/money). The theoretical results we obtain for the "fitting" problem should be generalizable to the problem of approximating the function. To this aim we could employ the framework of covering and learning (Guillory & Bilmes, 2011) along the following lines: we would interrupt the recursive process in Algorithm 2 through which we construct the tree as soon as we reach a certain level of learning (fitting) w.r.t. the set of labeled samples. Then, it remains to show that our decision tree is at logarithmic factor of the optimal one for that level of learning. This is an interesting direction for future research.

References

Adler, M. and Heeringa, B. Approximating optimal binary decision trees. APPROX/RANDOM '08, pp. 1–9, 2008.

Arkin, E. M., Meijer, H., Mitchell, J.S.B., Rappaport, D. and Skiena, S.S. Decision trees for geometric models. In *Proc. of SCG '93*, pp. 369–378, 1993.

Bellala, G., Bhavnani, S. K., and Scott, C. Group-based active query selection for rapid diagnosis in time-critical situations. *IEEE Trs. Inf. Theor.*, 58(1):459–478, 2012.

B.M.E. Moret. Decision Trees and Diagrams. *ACM Computing Surveys*, pp. 593–623, 1982.

Chakaravarthy, V. T., Pandit, V., Roy, S., Awasthi, P., and Mohania, M. Decision trees for entity identification: approximation algorithms and hardness results. In *Proc. PODS '07*, pp. 53–62, 2007.

Chakaravarthy, V. T., Pandit, V., Roy, S., and Sabharwal, Y. Approximating decision trees with multiway branches. In *Proc. ICALP '09*, pp. 210–221, 2009.

Dasgupta, S. Analysis of a greedy active learning strategy. In *NIPS'04*, 2004.

Golovin, D., Krause, A., and Ray, D. Near-optimal bayesian active learning with noisy observations. In *Proc. of NIPS'10*, pp. 766–774, 2010.

Guillory, A. and Bilmes, J. Average-case active learning with costs. In *Proc. of ALT'09*, pp. 141–155, 2009.

Guillory, A. and Bilmes, J. Interactive submodular set cover. In *Proc. ICML'10*, pp. 415–422. 2010.

Guillory, A. and Bilmes, J. Simultaneous learning and covering with adversarial noise. *ICML'11*, pp. 369–376. 2011.

Gupta, A., Nagarajan, V., and Ravi, R. Approximation algorithms for optimal decision trees and adaptive tsp problems. In *Proc. ICALP'10*, pp. 690–701, 2010.

Hanneke, S. The cost complexity of interactive learning. *unpublished*, 2006.

Kaplan, H., Kushilevitz, E., and Mansour, Y.. Learning with attribute costs. In *STOC*, pp. 356–365, 2005.

Kosaraju, S. Rao, Przytycka, Teresa M., and Borgstrom, Ryan S. On an optimal split tree problem. In *Proc. of WADS '99*, pp. 157–168, 1999.

Laber, Eduardo S. and Nogueira, Loana Tito. On the hardness of the minimum height decision tree problem. *Discrete Appl. Math.*, 144:209–212, 2004.

Nemhauser, G., Wolsey, L., and Fisher, M. L. An analysis of approximations for maximizing submodular set functions-i. *Math. Programming*, 14:265–294, 1978.

Nevmyvaka, Y., Feng, Y., and Kearns, M.. Reinforcement learning for optimized trade execution. In *Proc. of ICML'06*, pp. 673–680, 2006.

Sviridenko, M. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41 – 43, 2004.

Wolsey, L. Maximising real-valued submodular functions. *Math. of Operation Research*, 7(3):410–425, 1982.