# Racing trees to query partial data

Vu-Linh Nguyen, Sébastien Destercke, Marie-Hélène Masson, Rashad Ghassani

**Noname manuscript No.**
(will be inserted by the editor)

# Racing trees to query partial data

**Vu-Linh Nguyen · Sébastien Destercke* · Marie-Hélène Masson · Rashad Ghassani**

Received: date / Accepted: date

**Abstract** Dealing with partially known or missing data is a common problem in machine learning. This work is interested in the problem of querying the true value of data to improve the quality of the learned model, when those data are only partially known. This study is in the line of active learning, since we consider that the precise value of some partial data can be queried to reduce the uncertainty in the learning process, yet can consider any kind of partial data (not only entirely missing one). We propose a querying strategy based on the concept of racing algorithms in which several models are competing. The idea is to identify the query that will help the most to quickly decide the winning model in the competition. After discussing and formalizing the general ideas of our approach, we study the particular case of decision trees in case of interval-valued features and set-valued labels. The experimental results indicate that, in comparison to other baselines, the proposed approach significantly outperforms simpler strategies in the case of partially specified features, while it achieves similar performances in the case of partially specified labels.

**Keywords** partial data · data querying · active learning · racing algorithms · decision trees

Vu-Linh Nguyen
Heinz Nixdorf Institute and Department of Computer Science
Paderborn University, Germany
E-mail: vu.linh.nguyen@uni-paderborn.de

Sébastien Destercke* (corr. author)
UMR CNRS 7253 Heudiasyc, Sorbonne Universités, Université de Technologie de Compiègne CS 60319 - 60203 Compiègne cedex, France
E-mail: sebastien.destercke@hds.utc.fr

Marie-Hélène Masson
UMR CNRS 7253 Heudiasyc, Sorbonne Universités, Université de Technologie de Compiègne CS 60319 - 60203 Compiègne cedex, France, Université de Picardie Jules Verne, France
E-mail: mmasson@hds.utc.fr

Rashad Ghassani
Lebanese University, Lebanon
E-mail: rashad.ghassani@gmail.com

## 1 Introduction

Classical learning schemes assume that every instance is fully specified, yet there are many cases where such an assumption is unlikely to hold, and where some features or the label (class) of an instance may be only partially known. Such situations are commonly encountered in various fields, such as biostatistics (Heitjan, 1993), agronomy (Lagacherie et al, 2000), or economy (McDonald et al, 2018). These data can be generated by coarse or censored measurements (see e.g., (Efron, 1981)), anonymization techniques Dobra and Fienberg (2000), or can be obtained from expert opinions.

Multiple solutions have been proposed to deal with such problems. A first solution is to try to automatically complete the data by imputing their values (Rubin, 1976; Farhangfar et al, 2008; Lobato et al, 2015). A second one is to adapt learning methods so that they can integrate directly imprecise data (Xia et al, 2017). This latter approach has recently gained an increasing attention, both from the theoretical perspective (Hüllermeier, 2014; Liu and Dietterich, 2014; Cabannnes et al, 2020) and the practical one (Zhang et al, 2016; Utkin, 2019), with applications in different fields like image or natural language processing (Cour et al, 2011, 2009). However, in both cases the imprecision in the data usually leads to less accurate models, which in turn can lead to wrong decisions. It is therefore desirable, if possible, to reduce this imprecision in meaningful and efficient ways, to obtain the best possible model.

This work explores this particular issue: if we have the possibility to gain more information on some of the partial instances, which instance and what feature of this instance should we query to increase the final quality of the learned model? In the case of a completely missing label (and to a lesser extent of missing features), this problem is known as active learning and has already been largely treated (Settles, 2009; Ma et al, 2016). However, except for some recent works focusing on k-nn models (Nguyen et al, 2017), we are not aware of such works for generic partial data. Moreover, we also deal with the problem of partially specified features (i.e., issued from an unreliable sensor, from expert opinions), a problem that to our knowledge has never been treated in an active learning setting (the problem of completely missing features itself has received much less attention (Settles, 2009, Sec. 5.2.)).

While it may not always be possible to perform such a querying of partial data, we can think of many situations where it would be possible. This will be typically the case when cheap rough measurements can be complemented with a limited number of precise yet costly ones. For instance, coarse satellite images (Prince, 1991) or expert opinions (e.g., of building features to detect problems (Feng et al, 2017)) can be enriched by selected field measurements. In medicine, data issued from general health checks can be made more precise by asking for complementary exams.

In this paper, we propose to apply an approach (Nguyen et al, 2018) inspired from the idea of racing algorithms (Maron and Moore, 1997), initially used to statistically select an optimal configuration of a given lazy learning model, and since then applied to other settings such as multi-armed bandits (Busa-Fekete et al, 2014). The idea of such racing algorithms is to oppose a (finite) set of alternatives in a race, and to progressively discard losing ones as the race goes along. In our case, the set of alternatives will be different possible models. As the data is partial, the performance of each model is uncertain and several candidate models can be optimal. The race will consist in iteratively making queries, i.e. in asking to an oracle the precise value of a partial data. The key question is then to identify those data that will help the most to reduce the set of possible winners in the race and to converge quickly to the optimal model, hence those data that help the most in differentiating bad from good models. Provided the set of alternatives is rich and diverse enough, this should help in improving the quality of the model finally learned from the updated data.

We investigate how this general approach can be applied to decision tree classifiers (Quinlan, 1986; Safavian and Landgrebe, 1991). Decision trees are well-known to be sensitive to changes in the data, hence the importance of querying meaningful data for such classifiers. We show that detecting the data to query in our method can be done efficiently for decision trees when either the labels or the features are imprecise.

As the paper provides many algorithmic as well as mathematical results, we now provide a detailed description of the article contents to help the reader to navigate through it:

- Section 2 provides the necessary notations and preliminaries to understand the paper content;
- Section 3 provides a generic description of the racing approach we intend to apply in this paper, that was previously described and applied to SVM in (Nguyen et al, 2018). In particular, Algorithm 1 provides a generic description of our partial data querying method, applicable to any learning method. Although its reading is not absolutely necessary to understand the peculiar application to decision trees, it is useful to get the general ideas of our approach;
- Section 4 investigates how our general approach can be applied to decision trees when output labels are partially known and features precisely known. The first part of the section, up to Section 4.4, describes the formal elements and properties used to implement the querying method. Section 4.4 then provide details about the algorithmic implementation, in which Algorithm 2 provides the details of a single query, and refer to specific algorithms used in each step of the process;
- Section 5 investigates how our general approach can be applied to decision trees when features are partially known, which turns out to be trickier to handle than partial labels. Again, the first part (up to Section 5.4) provides formal justification of the algorithmic procedures. Section 5.4 then provides details about the algorithmic implementation, with Algorithm 7 being central and describing a general query, with reference to sub-routines to be used within the process;
- Finally, some experiments are performed on different data sets in Section 6, showing that our method is particularly interesting in the case where features are imprecisely known.

As the paper can get heavy in terms of notations, Table 1 summarizes the main notations and indices used in the paper.

## 2 Preliminaries

In classical supervised setting, the goal of the learning approach is to find a model $m : \mathcal{X} \to \mathcal{Y}$ within a set $\mathcal{M}$

| Symbols | Description |
|---|---|
| | Data related notations |
| $\mathcal{X}$ | input space, dimension $P$ |
| $\mathcal{Y}$ | output space, dimension $G$ |
| $\mathbf{x}$, $y$ | precise input and output |
| $\mathbf{X}$, $Y$ | imprecise input and output |
| $x_n^p$ | $p$-th precise coordinate of instance $\mathbf{x_n}$ |
| $X_n^p$ | $p$-th imprecise coordinate of instance $\mathbf{X_n}$ |
| $\lambda_g$ | $g$-th class among the $G$ possible ones |
| $[N]$ | the set $\{1, 2, \ldots, N\}$ |
| | Hypothesis-space, model related notations |
| $m$ | a model |
| $\mathcal{M}$ | set of models, dimension $S$ |
| $m(\mathbf{x})$ | output of model $m$ for input $\mathbf{x}$ |
| $\ell(y, m(\mathbf{x}))$ | 0-1 loss |
| $\underline{\ell}(Y, m(\mathbf{X}))$,$\overline{\ell}(Y, m(\mathbf{X}))$ | lower and upper 0-1 losses |
| $R(m)$ | empirical risk of model $m$ |
| $\underline{R}(m)$,$\overline{R}(m)$ | lower and upper empirical risks of model $m$ |
| $m^*_{mm}$, $m^*_{mM}$ | minimin and minimax optimal models |
| $\underline{R}(m_{k-l})$ | lower difference of empirical risks between $m_k$ and $m_l$ |
| | Query related notations |
| $q_n^p$ | a query: asking for the precise value of the $p$-th coordinate of instance $n$ |
| $q_n$ | a query: asking for the precise label of instance $n$ |
| $E_{q_n^p}(m_k)$ | single effect of query $q_n^p$ |
| $J_{q_n^p}(m_k, m_l)$ | pairwise effect of query $q_n^p$ |
| | Decision tree related notations |
| $H$ | number of terminal nodes of a tree |
| $A_h$ | $h$-th terminal node of a tree |
| $A_h^p$ | projection of $A_h$ on the $p$-th axis |
| $\lambda_h$ | class associated to leaf $A_h$ |
| | Experiment related notations |
| $(\delta, \eta)$ | contamination parameters in the experiments |

**Table 1** Key notations used in this paper

of models from a data set $\mathbf{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N \subset \mathcal{X} \times \mathcal{Y}$ of $N$ input/output samples, where $\mathcal{X}$ and $\mathcal{Y}$ are respectively the input and the output spaces[1]. The empirical risk $R(m)$ associated to a model $m$ is evaluated as

$$R(m) = \sum_{n=1}^{N} \ell(y_n, m(\mathbf{x}_n)) \,, \tag{1}$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is the loss function, and $\ell(y, m(\mathbf{x}))$ is the loss of predicting $m(\mathbf{x})$ when observing $y$. The selected model is then the one that minimizes (1), that is

$$m^* = \arg\min_{m \in \mathcal{M}} R(m) \,. \tag{2}$$

Another way to see the model selection problem is to assume that a model $m_l$ is said to be better than $m_k$ (denoted $m_l \succ m_k$) if

$$R(m_k) - R(m_l) > 0 \,, \tag{3}$$

or, in other words, if the risk of $m_l$ is lower than the risk of $m_k$.

[1] As $\mathcal{X}$ is often multi-dimensional, we will denote its elements and subsets by bold letters.

In this work, we are interested in a more general case where data is potentially only partially known, that is where general samples are of the kind $(\mathbf{X}_n, Y_n) \subseteq \mathcal{X} \times \mathcal{Y}$. In such a case, Equations (1), (2) and (3) are no longer well-defined, and there are different ways to extend them. Two of the most common ways to extend them is either to use a minimin (optimistic) (Hüllermeier and Beringer, 2006) or a minimax (pessimistic) approach (Guillaume et al, 2017). That is, if we extend Equation (1) to a lower bound $\underline{R}(m)$ and an upper bound $\overline{R}(m)$ such that

$$\begin{aligned} \underline{R}(m) &= \inf_{(\mathbf{x}_n, y_n) \in (\mathbf{X}_n, Y_n)} \sum_{n=1}^{N} \ell(y_n, m(\mathbf{x}_n)) \\ &= \sum_{n=1}^{N} \inf_{(\mathbf{x}_n, y_n) \in (\mathbf{X}_n, Y_n)} \ell(y_n, m(\mathbf{x}_n)) \\ &:= \sum_{n=1}^{N} \underline{\ell}(Y_n, m(\mathbf{X}_n)) \,, \end{aligned} \tag{4}$$

$$\begin{aligned}\overline{R}(m) &= \sup_{(\mathbf{x}_n,y_n)\in(\mathbf{X}_n,Y_n)} \sum_{n=1}^{N} \ell(y_n, m(\mathbf{x}_n)) \\ &= \sum_{n=1}^{N} \sup_{(\mathbf{x}_n,y_n)\in(\mathbf{X}_n,Y_n)} \ell(y_n, m(\mathbf{x}_n)) \\ &:= \sum_{n=1}^{N} \overline{\ell}(Y_n, m(\mathbf{X}_n))\,, \end{aligned} \tag{5}$$

then the optimal minimin $m^*_{mm}$ and minimax $m^*_{mM}$ models are

$$m^*_{mm} = \arg\min_{m\in\mathcal{M}} \underline{R}(m)\,, \tag{6}$$

$$m^*_{mM} = \arg\min_{m\in\mathcal{M}} \overline{R}(m)\,. \tag{7}$$

The minimin approach usually assumes that data are distributed according to the model, and tries to find the best data replacement (or disambiguation) combined with the best possible model (Hüllermeier, 2014). Conversely, the minimax approach assumes that data are distributed in the worst possible way, and selects the model performing the best in the worst situation, thus guaranteeing a minimum performance of the model (Guillaume et al, 2017). However, such an approach, due to its conservative nature, may lead to sub-optimal models. When having to choose a preferred model in the race, we will follow the optimistic approach, that is also in line with the idea of racing algorithms. For a deeper discussion on the differences between the optimistic and pessimistic approaches, we refer to (Hüllermeier et al, 2019).

However, in this paper, we are not primarily interested into learning a single model from partial data, but want to determine which partial data makes the potentially best models incomparable, in order to complete such data through query. To define such a set of potentially optimal models,we will say that a model $m_l$ is better than $m_k$ (still denoted $m_l \succ m_k$) if

$$\underline{R}(m_{k-l}) = \inf_{(\mathbf{x}_n,y_n)\in(\mathbf{X}_n,Y_n)} \big[R(m_k) - R(m_l)\big] > 0\,, \tag{8}$$

which is a direct extension of Equation (3). That is, $m_l \succ m_k$ if and only if it is better under every possible precise instances $(\mathbf{x}_n, y_n)$ consistent with the partial instances $(\mathbf{X}_n, Y_n)$. Such an approach is similar to decision rules used, for instance, in imprecise probability (Troffaes, 2007). We can then denote by

$$\mathcal{M}^* = \{m \in \mathcal{M} : \nexists m' \in \mathcal{M}\, s.t.\, m' \succ m\} \tag{9}$$

the set of undominated models within $\mathcal{M}$, that is the set of models that are maximum with respect to the partial order $\succ$.

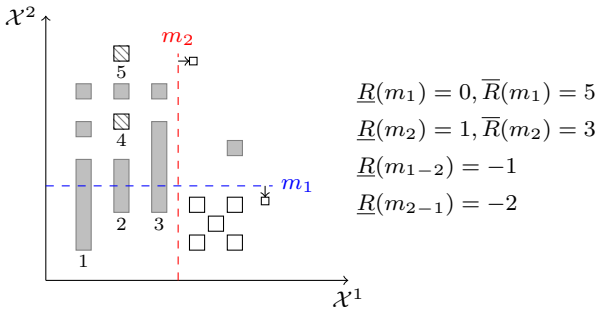


**Fig. 1** Illustration of partial data and competing models

*Example 1* Figure 1 illustrates a situation where $\mathcal{Y}$ consists of two different classes (gray and white), and $\mathcal{X}$ of two dimensions. Only imprecise data are numbered. Squares are assumed to have precise features. Points 1, 2 and 3 are imprecise with respect to their second feature. Shaded squares (points 4 and 5) have unknown labels. Assuming that $\mathcal{M} = \{m_1, m_2\}$ (the models could be decision stumps, i.e, one-level decision trees (Rodríguez and Maudes, 2008), we would have that $m_2 = m^*_{mM}$ is the minimax model and $m_1 = m^*_{mm}$ the minimin one. The two models would however be incomparable according to (8), hence $\mathcal{M}^* = \mathcal{M}$ in this case, and the minimax and minimin rules would have given us different answers.

## 3 Partial data querying: a racing approach

This section presents a general querying scheme based on racing idea and then investigates the computational issue of such a scheme for the specific setting of decision trees.

Both the minimin and minimax approaches (6)-(7) have the same goal: obtaining a unique model from partially specified data. Our objective in this paper is different: we want to query those data that will increase the most the accuracy of a learned model. To do so, we propose to start from a set $\mathcal{M}$ of potentially optimal models, and to identify in a racing scheme those data that will be the most efficient in reducing our uncertainty about which model is the optimal one within $\mathcal{M}$, hence are likely to be determinant in differentiating model quality in general. Much like querying-by-committee in classical active learning (Abe and Mamitsuka, 1998), the purpose of the race is to select the query to be made, as $\mathcal{M}$ is unlikely to contain the risk minimizing model. Once the queries have been made, a new model should be learned from the updated data set. How we quantify the usefulness of a query within the race is formalized in what follows.

Let us first assume that $\mathcal{X} = \mathcal{X}^1 \times \ldots \times \mathcal{X}^P$ is a Cartesian product of $P$ spaces, that a partial data $(\mathbf{X}_n, Y_n)$ can be expressed as $(\times_{p=1}^P X_n^p, Y_n)$, and furthermore that if $\mathcal{X}^p \subseteq \mathbb{R}$ is a subset of the real line, then $X_n^p$ is a closed interval.

A query on a partial data $(\times_{p=1}^P X_n^p, Y_n)$ consists in transforming one of its dimension $X_n^p$ or $Y_n$ into the true precise value $x_n^p$ or $y_n$, provided by an oracle (an expert, a precise measuring device). More precisely, $q_n^p$ denotes the query made on $X_n^p$ or $Y_n$, with $p = P + 1$ for $Y_n$. Given a model $m_k$ and a data $(\times_{p=1}^P X_n^p, Y_n)$, we are interested in knowing two things:

- whether the result of a query can have an effect on the the empirical risk bounds

  $$\mathbf{R}(m_k) = [\underline{R}(m_k), \overline{R}(m_k)]\,,$$

  which will be the case only if the query can have an effect on the interval

  $$\mathbf{L}(Y_n, m_k(\mathbf{X}_n) = [\underline{\ell}(Y_n, m_k(\mathbf{X}_n)), \overline{\ell}(Y_n, m_k(\mathbf{X}_n))]\,.$$

  We will then speak about the single effect of a query, as we will consider a single model;
- whether the model $m_l$ can be preferred to $m_k$ after performing a query, in which case we have to assess whether the query can have an influence on the lower bound $\underline{R}(m_{k-l})$ or not, since $m_l$ will be preferred to $m_k$ as soon as this bound becomes positive.

This can be formalized by two functions:

$$E_{q_n^p} : \mathcal{M} \to \{0, 1\}\,, \tag{10}$$

$$J_{q_n^p} : \mathcal{M} \times \mathcal{M} \to \{0, 1\}\,, \tag{11}$$

such that:

$$E_{q_n^p}(m_k) = \begin{cases} 1 \text{ if } \exists x_n^p \in X_n^p \text{ that reduces } \mathbf{R}(m_k) \\ 0 \text{ else}\,. \end{cases}$$

$$J_{q_n^p}(m_k, m_l) = \begin{cases} 1 \text{ if } \exists x_n^p \in X_n^p \text{ that increases } \underline{R}(m_{k-l}) \\ 0 \text{ else}\,. \end{cases}$$

When $p = P+1$, $X_n^p$ is to be replaced by $Y_n$. $E_{q_n^p}$ simply tells us whether or not the query can affect our evaluation of $m_k$ performances, while $J_{q_n^p}(m_k, m_l)$ informs us whether the query can help to differentiate $m_k$ and $m_l$. Let $[S] := \{1, 2, \ldots, S\}$. If we denote by

$$k^* = \arg\min_{k \in [S]} \underline{R}(m_k)$$

the currently winning model (racing algorithms do focus on this model, trying to determine if it is really the winner of the race), the total effect of a query $q_n^p$ is defined as

$$Value(q_n^p) = E_{q_n^p}(m_{k^*}) + \sum_{k \neq k^*} J_{q_n^p}(m_k, m_{k^*})\,. \tag{12}$$

This value or utility is then used to assess which data (label or feature) should be queried next. It should be noticed that scores (10) and (11) can be modified, for example to account for different loss functions. Unless there are other reasons to change it, our choice appears to be the simplest and most straightforward.

*Example 2* In Figure 1, questions related to partial classes (points 4 and 5) and to partial features (points 1, 2 and 3) have respectively the same potential effect, so we can restrict our attention to $q_4^3$ (the class of point 4) and to $q_3^2$ (the second feature of point 3). For these two questions, we have

- $E_{q_4^3}(m_1) = E_{q_4^3}(m_2) = 1$.
- $J_{q_4^3}(m_1, m_2) = J_{q_4^3}(m_2, m_1) = 0$.
- $E_{q_3^2}(m_1) = 1$, $E_{q_3^2}(m_2) = 0$.
- $J_{q_3^2}(m_1, m_2) = J_{q_3^2}(m_2, m_1) = 1$.

This example shows that while some questions may reduce our uncertainty about many model risks ($q_4^3$ reduce risk intervals for both models), they may be less useful than other questions to tell two models apart ($q_3^2$ can actually lead to declare $m_2$ better than $m_1$, while $q_4^3$ cannot).

The effect of a query being now formalized, we can propose a method inspired by racing algorithms. To create the initial set of racing models, a convenient method is to sample $S$ times a precise data set

$$\mathbf{D} = \{(\mathbf{x}_n, y_n) \in (\mathbf{X}_n, Y_n)\}_{n=1}^N$$

and then to learn an optimal model for each such selection. Algorithm 1 summarises the general procedure applied to find the best query and to update the race. This algorithm simply searches the query that will have the biggest impact on the minimin model and its competitors, adopting the optimistic attitude of racing algorithms. Once a query has been made, the data set as well as the set of competitors are updated, so that only potentially optimal models remain. Note that in practice, such a sampling is close to methods used in query-by-committee approaches (Abe and Mamitsuka, 1998; Nigam and McCallum, 1998), and makes no specific assumption about the process that has led to imprecision. Also, as in usual query-by-committee and racing approaches, we also assume that we work with models of the same nature and of comparable complexity.

The next Section presents the practical computations of racing algorithm (Algorithm 1) for the case of set-valued labels, when the ones for the case of interval-valued features will be given in the Section 5.

**Algorithm 1:** One iteration of the racing algorithm to query data.

**Input:** data $\{(X_n, Y_n)\}_{n=1}^N$, set of models $\mathcal{M} = \{m_1, \ldots, m_S\}$
**Output:** updated data and set of models

$k^* = \arg\min_{k\in[S]} \underline{R}(m_k)$;
**foreach** *query* $q_n^p$ **do**
  $Value(q_n^p) = E_{q_n^p}(m_{k^*}) + \sum_{k\neq k^*} J_{q_n^p}(m_k, m_{k^*})$;
$(n^*, p^*) = \arg\max_{(n,p)} Value(q_n^p)$;
Get value $x_{n^*}^{p^*}$ of $X_{n^*}^{p^*}$ ;
**foreach** $k, \ell \in [S] \times [S]$, $k \neq \ell$ **do**
  Compute $\underline{R}(m_{k-\ell})$ ;
  **if** $\underline{R}(m_{k-\ell}) > 0$ **then** remove $m_k$ from $\mathcal{M}$ ;

## 4 Application to decision trees: set-valued labels

In this section, we consider that the labels of some instances are partially given, but that all the features are precise. A query will simply be denoted by $q_n$ meaning that the label of instance $\mathbf{x}_n$ is queried. In the classical setting of decision trees, the input space is $\mathcal{X} = \mathcal{X}^1 \times \ldots \times \mathcal{X}^P \subseteq \mathbb{R}^p$ (where $\mathbb{R}$ is the real line) and the output space is $\mathcal{Y} = \{\lambda_1, \ldots, \lambda_G\}$, where $\lambda_g$, $g \in [G]$, encode all the possible classes. A decision tree $m_k$ is formally a rooted tree structure consisting of terminal nodes and non-terminal nodes (Quinlan, 1986; Safavian and Landgrebe, 1991):

- each non-terminal node of the tree is associated to an attribute $X^p$ ($p \in [P]$), and to each branch issued from this node is associated a condition on this attribute that determines which data of the sample $\mathbf{D}$ go into that branch.
- terminal nodes are called leaves. Each leaf is associated to a predicted class $\lambda_h \in \mathcal{Y}$ and a partition element $\mathbf{A}_h = A_h^1 \times \ldots \times A_h^P$ where $A_h^p \subseteq \mathcal{X}^p$. In the rest of this paper, we will adopt, for each leaf $t_h$, the following notation

  $t_h = (\mathbf{A}_h, \lambda_h)$

  as such information is enough for the purpose of making prediction for new instances: for any instance $\mathbf{x}_n \in \mathbf{A}_h$, we have $m(\mathbf{x}_n) = \lambda_h$.

  The next small example illustrates those notations.

*Example 3* We consider a given tree trained from data set $\mathbf{D} \in \mathcal{X}$ with $P = 2$ attributes, and $M = 3$ classes. Input and output spaces are described as follows:

$$\mathcal{X}^1 = [1, 10], \mathcal{X}^2 = [10, 20], \mathcal{Y} = \{a, b, c\}.$$

Figure 2 illustrates a possible decision tree $m_k$ for the above setting.

Assume we have new instances $\mathbf{x}_1 = (2, 17)$ and $\mathbf{x}_2 = (6, 11)$. Then $\mathbf{x}_1$ will reach leaf $t_4$ and be assigned to class $m_k(\mathbf{x}_1) = c$ while $\mathbf{x}_2$ will reach leaf $t_1$ with an assigned class $m_k(\mathbf{x}_2) = a$.

We will focus on the classical $0-1$ loss function defined as follows: for a given instance $(\mathbf{x}_n, y_n)$

$$\ell(y_n, m_k(\mathbf{x}_n)) = \begin{cases} 0 \text{ if } y_n = m_k(\mathbf{x}_n) \\ 1 \text{ otherwise}. \end{cases} \tag{13}$$

In case of partially labelled data, the label is a set $Y_n \subseteq \mathcal{Y}$ instead of a single label. The loss in (13) becomes an interval $\mathbf{L}(Y_n, m_k(\mathbf{X}_n)$ whose bounds are

$$\underline{\ell}(Y_n, m_k(\mathbf{x}_n)) = \min_{\lambda\in Y_n} \ell(\lambda, m_k(\mathbf{x}_n)), \tag{14}$$

$$\overline{\ell}(Y_n, m_k(\mathbf{x}_n)) = \max_{\lambda\in Y_n} \ell(\lambda, m_k(\mathbf{x}_n)). \tag{15}$$

*Example 4* Let us now continue with the data set and the decision tree from Example 3. Assume that instances $\mathbf{x}_1$ and $\mathbf{x}_2$ are partially labelled with $Y_1 = \{a, c\}$ and $Y_2 = \{b, c\}$, respectively. Then using (14) and (15), we can easily get

$$\mathbf{L}(Y_1, m_k(\mathbf{X}_1) = [0, 1], \mathbf{L}(Y_2, m_k(\mathbf{X}_2) = [1, 1].$$

The next sections investigate how we can apply the racing algorithm presented in Section 3 to the practical case of partial labels in decision trees. We first study under which conditions a given partial label introduces imprecision in the empirical risks, before detailing the computation of querying value scores.

### 4.1 Instances introducing imprecision in empirical risk

For a given instance $(\mathbf{x}_n, Y_n)$ and a decision tree $m_k$, the lower and upper losses in (14) and (15) can be determined as follows:

$$\underline{\ell}(Y_n, m_k(\mathbf{x}_n)) = \begin{cases} 0 \text{ if } m_k(\mathbf{x}_n) \in Y_n, \\ 1 \text{ otherwise,} \end{cases} \tag{16}$$

$$\overline{\ell}(Y_n, m_k(\mathbf{x}_n)) = \begin{cases} 0 \text{ if } \{m_k(\mathbf{x}_n)\} = Y_n, \\ 1 \text{ otherwise.} \end{cases} \tag{17}$$

Given a decision tree $m_k$, we will say that an instance is imprecise w.r.t. $m_k$ if

$$\underline{\ell}(Y_n, m_k(\mathbf{x}_n)) \neq \overline{\ell}(Y_n, m_k(\mathbf{x}_n)). \tag{18}$$

The next proposition characterizes simple conditions under which an instance is imprecise w.r.t. $m_k$.

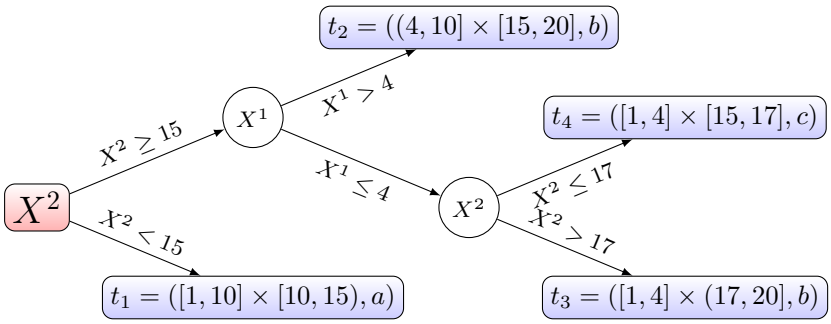


**Fig. 2** Decision tree illustration $m_k$

**Proposition 1** *Given a model $m_k$ and instance $(\mathbf{x}, Y)$, then $(\mathbf{x}, Y)$ is imprecise w.r.t. $m_k$ if and only if*

$$m_k(\mathbf{x}) \in Y \text{ and } |Y| > 1\,. \tag{19}$$

*Proof* Let us first note that by definitions we always have

$$\underline{\ell}(Y, m_k(\mathbf{x})) \leq \overline{\ell}(Y, m_k(\mathbf{x}))\,.$$

Then combining with condition (18) the lower and upper losses of an imprecise instance can be determined explicitly by

$$\underline{\ell}(Y, m_k(\mathbf{x})) = 0 \text{ and } \overline{\ell}(Y, m_k(\mathbf{x})) = 1\,. \tag{20}$$

Conditions in (13) guarantee that $\underline{\ell}(Y, m_k(\mathbf{x})) = 0$ is equivalent to condition that $m_k(\mathbf{x}) \in Y$. Furthermore, condition $|Y| > 1$ ensures that $\overline{\ell}(Y, m_k(\mathbf{x})) = 1$ (otherwise, $\{m_k(\mathbf{x})\} = Y$, and both lower and upper losses will be 0). □

Proposition 1 simply translates the fact that imprecision can happen only if a partial label could contain the prediction of $m_k$. Using Proposition 1, we can conclude that in Example 4, instance $\mathbf{x}_1$ is imprecise w.r.t. $m_k$ while $\mathbf{x}_2$ is precise, even if it has a partial label.

We are now going to investigate the practical computation of the empirical risk bounds of a single model, the pairwise risk bounds in a given set $\mathcal{M}$ of models and the effect of querying partial labels on those risks. It is easy to see that the empirical risk bound of a given model can be changed only by querying imprecise instances and the pairwise risk bounds can be changed if the chosen instance is imprecise w.r.t. at least one model. We will then focus on those cases in the next Sections.

### 4.2 Empirical risk bounds and single effect

Equation (4) (resp. (5)) implies that the computation of $\underline{R}(m_k)$ (resp. $\overline{R}(m_k)$) can be done by computing $\underline{\ell}(Y_n, m_k(\mathbf{x}_n))$ (resp. $\overline{\ell}(Y_n, m_k(\mathbf{x}_n))$), $\forall\ n \in [N]$, and then by summing the obtained values. Therefore, the computation of the lower and upper risks of a given model can be carried out easily after determining the lower and upper losses of each instance.

Before going to present conditions under which a query $q_n$ have an effect on modifying the interval $\mathbf{R}(m_k)$ (or in other words $E_{q_n}(m_k) = 1$), let us first note that a query $q_n$ is effective if and only if $\mathbf{L}(Y_n, m_k(\mathbf{X}_n)$ can be modified. Then, as pointed out in the next proposition, such effect (i.e $E_{q_n}(m_k) = 1$) will simply hold for all imprecise instances.

**Proposition 2** *Given an instance $(\mathbf{x}_n, Y_n)$ and a model $m_k$, then $E_{q_n}(m_k) = 1$ if and only if $(\mathbf{x}_n, Y_n)$ is imprecise w.r.t. $m_k$.*

*Proof* Firstly, it is easy to see that querying any instance that is precise w.r.t. $m_k$ will not help to modify $\mathbf{L}(Y_n, m_k(\mathbf{X}_n)$. Furthermore, (20) implies that $q_n$ have an effect by either increasing $\underline{\ell}(Y_n, m_k(\mathbf{x}_n))$ or decreasing $\overline{\ell}(Y_n, m_k(\mathbf{x}_n))$. We will now show that at least one of such losses can be changed after querying any imprecise instance $(\mathbf{x}_n, Y_n)$.

Assuming that $\lambda$ is the label we get after query $q_n$, then either $\lambda = m_k(\mathbf{x}_n)$ or $\lambda \neq m_k(\mathbf{x}_n)$. In the first case, both of lower and upper losses will be 0 after performing $q_n$ while both lower and upper losses will be 1 in the latter case. In other words, $E_{q_n}(m_k) = 1$ if $(\mathbf{x}_n, Y_n)$ is imprecise w.r.t. $m_k$. □

Computation of pairwise risk bounds and the effect $J_{q_n}(m_k, m_l)$ will be investigated in the next Section. Again, if an instance is precise w.r.t. both models, then querying it will not affect the pairwise risk bounds. Therefore, we will focus our interest on instances that are imprecise with respect to at least one model.

### 4.3 Pairwise risk bounds and effect

Let us now focus on how to compute, for a pair of models $m_k$ and $m_l$, the corresponding pairwise risk $\underline{R}(m_{k-l})$ and whether a query $q_n$ can increase this risk. The computation will be treated in two cases: when an instance

is imprecise w.r.t. only one model and when an instance is imprecise w.r.t. both.

First note that, similarly to the empirical risk bounds of a unique model, the computation of $\underline{R}(m_{k-l})$ can be carried out by simply summing up the values $\underline{\ell}_{k-l}(Y_n, \mathbf{x}_n)$ for all $(\mathbf{x}_n, Y_n)$ with

$$\underline{\ell}_{k-l}(Y_n, \mathbf{x}_n) = \inf_{y_n \in Y_n} \left[\ell(y_n, m_k(\mathbf{x}_n)) - \ell(y_n, m_l(\mathbf{x}_n))\right].$$

Furthermore, a query $q_n$ can increase $\underline{R}(m_{k-l})$ if and only if it can increase the value $\underline{\ell}_{k-l}(Y_n, \mathbf{x}_n)$. This is why, in this section, we focus on computing $\underline{\ell}_{k-l}(Y_n, \mathbf{x}_n)$ and its possible change after a query $q_n$.

#### *4.3.1 Imprecision with respect to one model*

We are going to present the computation of $\underline{\ell}_{k-l}(Y_n, \mathbf{x}_n)$ and the conditions under which $J_{q_n}(m_k, m_l) = 1$.

**Proposition 3** *Given $(\mathbf{x}_n, Y_n)$ and two models $m_k$ and $m_l$ s.t. $\mathbf{x}_n$ is imprecise w.r.t. one and only one model, we have the following relations:*

- *if $\mathbf{x}_n$ is imprecise w.r.t. $m_l$, then*

$$\underline{\ell}_{k-l}(Y_n, \mathbf{x}_n) = \ell(Y_n, m_k(\mathbf{x}_n)) - 1\,. \tag{21}$$

- *if $\mathbf{x}_n$ is imprecise w.r.t. $m_k$, then*

$$\underline{\ell}_{k-l}(Y_n, \mathbf{x}_n) = 0 - \ell(Y_n, m_l(\mathbf{x}_n))\,. \tag{22}$$

*Proof* Since $\mathbf{x}_n$ is imprecise w.r.t. only one model, the imprecision is only associated to such a model, and we can select the worst case: $\ell(y_n, m_\ell(\mathbf{x}_n)) = 1$ for Equation (21) and $\ell(y_n, m_k(\mathbf{x}_n)) = 0$ for Equation (22). □

Now we are going to study under which conditions a query $q_n$ can increase $\underline{\ell}_{k-l}$. As the given instance is imprecise w.r.t. only one model, it can only increase the pairwise risk by either increasing $\underline{\ell}(y_n, m_k(\mathbf{x}_n))$ or decreasing $\overline{\ell}(y_n, m_l(\mathbf{x}_n))$. As shown in the next Proposition, this can always happen, meaning that we systematically have $J_{q_n}(m_k, m_l) = 1$ in this case.

**Proposition 4** *Given $(\mathbf{x}_n, Y_n)$ and two models $m_k$ and $m_l$ s.t $\mathbf{x}_n$ is imprecise w.r.t. one and only one model, then query $q_n$ can always increase $\underline{\ell}_{k-l}$, or in other words $J_{q_n}(m_k, m_l) = 1$.*

*Proof* We investigate the case where $(\mathbf{x}_n, Y_n)$ is imprecise w.r.t. $m_k$, the case for $m_l$ can be treated similarly.

Assuming that $(\mathbf{x}_n, Y_n)$ is imprecise w.r.t. $m_k$, then Proposition 2 ensures that there always exists a label $\lambda \in Y_n$ such that the lower bound $\underline{\ell}(y_n, m_k(\mathbf{x}_n))$ will be increased to 1 after query $q_n$.

Similar claim about decreasing the upper bound $\overline{\ell}(y_n, m_l(\mathbf{x}_n))$ can be carried when $(\mathbf{x}_n, Y_n)$ is imprecise w.r.t. $m_l$. □

#### *4.3.2 Imprecision with respect to both models*

For the cases where $\mathbf{x}_n$ is imprecise w.r.t. both models $m_k$ and $m_l$, the computation of $\underline{\ell}_{k-l}(Y_n, \mathbf{x}_n)$ and the conditions under which $J_{q_n}(m_k, m_l) = 1$ will be investigated separately in two circumstances: when $m_k(\mathbf{x}_n) = m_l(\mathbf{x}_n)$ and when $m_k(\mathbf{x}_n) \neq m_l(\mathbf{x}_n)$.

**Proposition 5** *Given $(\mathbf{x}_n, Y_n)$ and two models $m_k$ and $m_l$ s.t $\mathbf{x}_n$ is imprecise w.r.t. both models, then the following results hold*

- *if $m_k(\mathbf{x}_n) = m_l(\mathbf{x}_n)$, then*

$$\underline{\ell}_{k-l}(Y_n, \mathbf{x}_n) = 0 \text{ and } J_{q_n}(m_k, m_l) = 0\,.$$

- *if $m_k(\mathbf{x}_n) \neq m_l(\mathbf{x}_n)$, then*

$$\underline{\ell}_{k-l}(Y_n, \mathbf{x}_n) = -1 \text{ and } J_{q_n}(m_k, m_l) = 1\,.$$

*Proof* - Let $m_k(\mathbf{x}_n) = m_l(\mathbf{x}_n)$, then

$$\ell(\lambda, m_k(\mathbf{x}_n)) = \ell(\lambda, m_l(\mathbf{x}_n))\,, \forall \lambda \in Y_n\,.$$

Therefore, we always have

$$\underline{\ell}_{k-l}(Y_n, \mathbf{x}_n) = \overline{\ell}_{k-l}(Y_n, \mathbf{x}_n) = \ell_{k-l}(Y_n, \mathbf{x}_n) = 0\,.$$

Furthermore, for any label $\lambda \in Y_n$ to be given after performing query $q_n$, the difference $\underline{\ell}_{k-l}(Y_n, \mathbf{x}_n)$ will be 0. Or in other words, if $m_k(\mathbf{x}_n) = m_l(\mathbf{x}_n)$, then we can conclude that $\underline{\ell}_{k-l}(Y_n, \mathbf{x}_n) = 0$ and $J_{q_n}(m_k, m_l) = 0$.

- In case $m_k(\mathbf{x}_n) \neq m_l(\mathbf{x}_n)$, as pointed out in Proposition 1, $\mathbf{x}_n$ being imprecise w.r.t. both models implies that $m_k(\mathbf{x}_n) \in Y_n$ and $m_l(\mathbf{x}_n) \in Y_n$. Then there always exists a label $\lambda$ in $Y_n$ (i.e $\lambda = m_k(\mathbf{x}_n)$) s.t. model $m_k$ returns a true prediction while $m_l$ returns a wrong one. In other words, we have $\underline{\ell}_{k-l}(Y_n, \mathbf{x}_n) = -1$. The effect $J_{q_n}(m_k, m_l) = 1$ follows simply by assuming that label $\lambda = m_l(\mathbf{x}_n)$ will be given after querying $\mathbf{x}_n$ which implies that $\underline{\ell}_{k-l}(Y_n, \mathbf{x}_n)$ will be increased into 1. □

The next section provides practical algorithms to perform a single querying step.

### 4.4 Algorithms

Algorithm 2 summarizes the complete procedure to perform an iteration of our querying strategy. Sub-routines are described in other algorithms. Algorithm 3 computes the individual risk bounds of every model, according to the corresponding values of $\underline{\ell}(Y_n, m_k(\mathbf{x}_n))$ and $\overline{\ell}(Y_n, m_k(\mathbf{x}_n))$. Algorithm 4 simply summarises the model selection procedure, that will also be used in the case of interval-valued features.

Finally, Algorithm 5 summarises the main procedure that determines the value of the different possible queries, allowing us to pick the best one among all the possible ones. Let us now analyse the complexity of this procedure. Lines 1-2 of Algorithm 2 are performed in $\mathcal{O}(S \times N)$, as Algorithm 3 is called $S$ times and is in $\mathcal{O}(N)$. Line 3 of Algorithm 2 is in $\mathcal{O}(S)$. Finally, since Algorithm 5 is in $\mathcal{O}(S)$, lines 4-5 of Algorithm 5 are in $\mathcal{O}(S \times N)$. So the overall complexity of Algorithm 5 is in $\mathcal{O}(S \times N)$, meaning that the approach is computationally affordable.

**Algorithm 2:** A single step to query set-valued data.

**Input:** Training data set $\{(\mathbf{x}_n, Y_n\}_{n=1}^N$, label set $\Omega = \{\lambda_1, \ldots, \lambda_G\}$, set of undominated model $\mathcal{M}$.
**Output:** The optimal query $q_{n^*}$

1 **foreach** $m_k \in \mathcal{M}$ **do**
2  Compute empirical risk $(\underline{R}(m_k), \overline{R}(m_k))$ bounds using Alg. 3;
3 Determine the best model $m_{k^*}$ and the undominated model set $\mathcal{M}$ using Alg. 4;
4 **foreach** $n \in [N]$ **do**
5  Determine the query effect value $Value(q_n)$ using Alg. 5;
6 Determine $q_{n^*} = \arg\max_n Value(q_n)$;

**Algorithm 3:** Compute the empirical risk bounds $(\underline{R}(m_k), \overline{R}(m_k))$.

**Input:** Training data set $\{(\mathbf{x}_n, Y_n\}_{n=1}^N$, label set $\Omega = \{\lambda_1, \ldots, \lambda_G\}$, model $m_k$.
**Output:** Empirical risk bounds $(\underline{R}(m_k), \overline{R}(m_k))$

1 $\underline{R}(m_k) = 0$, $\overline{R}(m_k) = 0$;
2 **foreach** $n \in [N]$ **do**
3  **if** $|Y_n| > 1$ **then**
4  **if** $m_k(\mathbf{x}_n) \notin Y_n$ **then** $\underline{R}(m_k) = \underline{R}(m_k) + 1$;
5  $\overline{R}(m_k) = \overline{R}(m_k) + 1$
6  **else if** $\{m_k(\mathbf{x}_n)\} \neq Y_n$ **then**
  $\underline{R}(m_k) = \underline{R}(m_k) + 1$, $\overline{R}(m_k) = \overline{R}(m_k) + 1$;

## 5 Application to decision trees: interval-valued features

We now deal with the case of interval-valued features, which is much more involved than the case of partial labels, yet still manageable from a computational point of view. Such additional difficulties may explain why there are very few active learning methods dealing with missing features, and none (to our knowledge) dealing with partially known features, at least to our knowledge.

**Algorithm 4:** Determine the best model $m_{k^*}$ and the undominated model set $\mathcal{M}^*$.

**Input:** Model set $\mathcal{M}$, empirical risk bounds $\{(\underline{R}(m_k), \overline{R}(m_k))|\forall m_k \in \mathcal{M}\}$
**Output:** The best model $m_{k^*}$ and the undominated model set $\mathcal{M}$

1 $k^* = \arg\min_{m_k \in \mathcal{M}} \underline{R}(m_k)$;
2 $\overline{R}_{\min} = \min_{m_k \in \mathcal{M}} \overline{R}(m_k)$ ;
3 **foreach** $m_k \in \mathcal{M}$ **do**
4  **if** $\underline{R}(m_k) > \overline{R}_{\min}$ **then** Remove $m_k$ from $\mathcal{M}$;

**Algorithm 5:** Determine the effect value of a query $Value(q_n)$.

**Input:** Training instance $(\mathbf{x}_n, Y_n)$, undominated model set $\mathcal{M}^*$.
**Output:** The querying effect value $Value(q_n)$

1 Initialize $E_{q_n}(m_{k^*}) = 0$, $J_{q_n} = 0$;
2 **if** $|Y_n| > 1$ *and* $m_{k^*}(\mathbf{x}_n) \in Y_n$ **then**
3  $E_{q_n}(m_{k^*}) = 1$;
4  **foreach** $m_k \in \mathcal{M}$ *and* $k \neq k^*$ **do**
5  **if** $m_k(\mathbf{x}_n) \in Y_n$ *and* $m_k(\mathbf{x}_n) \neq m_{k^*}(\mathbf{x}_n)$ **then** $J_{q_n} = J_{q_n} + 1$;
6  **else if** $m_k(\mathbf{x}_n) \notin Y_n$ **then** $J_{q_n} = J_{q_n} + 1$;
7 **else if** $|Y_n| > 1$ **then**
8  **foreach** $m_k \in \mathcal{M}$ *and* $k \neq k^*$ **do**
9  **if** $m_k(\mathbf{x}_n) \in Y_n$ **then** $J_{q_n} = J_{q_n} + 1$;
10 $Value(q_n) = E_{q_n}(m_{m^*}) + J_{q_n}$;

### 5.1 Instances introducing imprecision in empirical risk

Before going further, let us remind that, for a given tree $m$, each terminal node (which is sufficient in later analysis) is associated with a partition element

$$\mathbf{A}_h = A_h^1 \times \ldots, \times A_h^P, \tag{23}$$

where $A_h^p$ can be a closed, open or semi-closed interval in our case. However, for the sake of practical implementation and exposure, we will from now on assume that $A_h^p$ is a closed interval.

Since we work with interval-valued feature data, for each instance $(\mathbf{X}_n, y_n)$, its feature $\mathbf{X}_n$ can be represented as a hyper-cube (similar to terminal node in (23)) denoted by

$$\mathbf{X}_n = X_n^1 \times \ldots \times X_n^P\,. \tag{24}$$

The intersection between partition elements and/or partial instances is nothing else but the one of two hyper-cubes. Given two such hyper-cubes $\mathbf{U} = U^1 \times \ldots \times U^P$

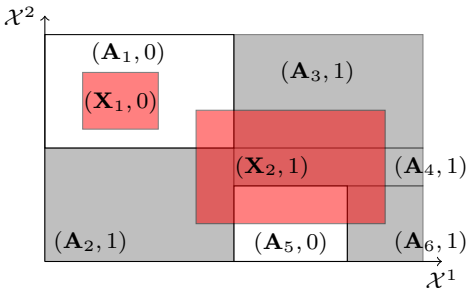


**Fig. 3** Example of imprecise instance

and $\mathbf{V} = V^1 \times \ldots \times V^P$, their corresponding intersection, denoted by $U \cap V$ is

$$U \cap V = \times_{p=1}^{P} U^p \cap V^p \,. \tag{25}$$

(25) provides a practical way to check whether the intersection of two cubic forms is non-empty. More precisely, we have that $\mathbf{U} \cap \mathbf{V} \neq \emptyset$ iff

$$U^p \cap V^p \neq \emptyset, \forall p \in [P] \,. \tag{26}$$

As for the case of partial labels, an instance $(\mathbf{X}, y)$ is said to be imprecise w.r.t. a decision tree $m_k$ if

$$\exists \mathbf{x}, \mathbf{x}' \in \mathbf{X} \text{ s.t } \ell(y, m_k(\mathbf{x})) \neq \ell(y, m_k(\mathbf{x}')) \,. \tag{27}$$

Furthermore, as an instance can intersect several partition elements which are possibly associated to different labels, then (27) is equivalent to the following relation

$$\begin{aligned} \exists \mathbf{A}_h, \mathbf{A}_{h'} \text{ s.t } & \mathbf{X} \cap \mathbf{A}_h \neq \emptyset, \mathbf{X} \cap \mathbf{A}_{h'} \neq \emptyset \\ & \text{and } \lambda_h = y \text{ and } \lambda_{h'} \neq y \,. \end{aligned} \tag{28}$$

Note that (28) can be easily determined using (26). The following Example gives illustrations of an imprecise instance w.r.t. a given decision tree.

*Example 5* Figure 3 gives an example of a tree $m_k$ and two instances, $(\mathbf{X}_1, 0)$, $(\mathbf{X}_2, 1)$.

It is easy to see that $(\mathbf{X}_1, 0)$ is a precise instance since it only intersects with a partition element associated to label 0. However $\mathbf{X}_2$ is imprecise since (28) holds. More precisely, $(\mathbf{X}_2, 1)$ intersects with $\mathbf{A}_5$ and $\mathbf{A}_6$ whose associated labels are different.

## 5.2 Empirical risk bounds and single effect

We are now going to investigate how risk bounds $\mathbf{R}(m_k)$ can be computed efficiently from data $\{(\mathbf{X}_n, y_n)\}_{n=1}^N$ by computing bounds $\mathbf{L}(y_n, m_k(\mathbf{X}_n))$, and how the potential effect of a query $q_n^p$ ($q_n^p$ corresponding to ask the true value within $X_n^p$) on those bounds can be estimated.

Let us first study how bounds on loss functions can be estimated. Similarly to the case of set-valued labels, an instance $\mathbf{X}_n$ will get the imprecise empirical risk bounds $\mathbf{L}(y_n, m_k(\mathbf{X}_n)) = [0, 1]$ iff it satisfies condition (28). Otherwise, the corresponding loss is precise and such an instance can be discarded from the querying process. For example, in Figure 3, we can see that $\mathbf{L}(y_1, m_k(\mathbf{X}_1)) = [0, 0]$ while $\mathbf{L}(y_2, m_k(\mathbf{X}_2)) = [0, 1]$. Note that a training instance $(\mathbf{X}_n, y_n)$ is precise if and only if partition elements that intersect with it either are all of label $y_n$ or all different from $y_n$. To determine whether such a condition holds, let us firstly introduce the following information vectors

$$\mathbf{K} = (k^1, \ldots, k^H), k^h = \begin{cases} 1 \text{ if } \mathbf{X}_n \cap \mathbf{A}_h \neq \emptyset \,, \\ 0 \text{ otherwise,} \end{cases} \tag{29}$$

$$\mathbf{B}_{y_n} = (b_{y_n}^1, \ldots, b_{y_n}^H), b_{y_n}^h = \begin{cases} 0 \text{ if } \lambda_h = y_n \,, \\ 1 \text{ otherwise} \,, \end{cases} \tag{30}$$

$$\mathbf{C}_{y_n} = (c_{y_n}^1, \ldots, c_{y_n}^H), c_{y_n}^h = \begin{cases} 1 \text{ if } \lambda_h = y_n \,, \\ 0 \text{ otherwise} \,, \end{cases} \tag{31}$$

with $H$ the number of terminal nodes of the decision tree $m_k$. Note that $\mathbf{K}$ can easily be built using (26), and that $\mathbf{B}, \mathbf{C}$ have to be built only once. A given training instance $\mathbf{X}_n$ is imprecise w.r.t. $m_k$ if and only if $(\mathbf{K}\mathbf{B}_{y_n}^\top)(\mathbf{K}\mathbf{C}_{y_n}^\top) \neq 0$, where $\mathbf{a}\mathbf{b}^\top$ is the dot product of two vectors $\mathbf{a}$ and $\mathbf{b}$. Before going further, let us note that we can use information vectors to deduce that $\ell(y_n, m_k(\mathbf{X}_n))$ has the precise value 0 and 1, as this happens when $\mathbf{K}\mathbf{B}_{y_n}^\top = 0$ and $\mathbf{K}\mathbf{C}_{y_n}^\top = 0$, respectively.

One can see that performing a query $q_n^p$ can only change $\mathbf{K}$. Denoting by $\mathbf{K}_{q_n^p}$ the vector resulting from $q_n^p$, the single effect $E_{q_n^p}(m_k) = 1$ if and only if

$$(\mathbf{K}\mathbf{B}_{y_n}^\top)(\mathbf{K}\mathbf{C}_{y_n}^\top) \neq 0$$

and

$$\exists x_n^p \in X_n^p \text{s.t.} (\mathbf{K}_{q_n^p}\mathbf{B}_{y_n}^\top)(\mathbf{K}_{q_n^p}\mathbf{C}_{y_n}^\top) = 0 \,.$$

Verifying whether such a situation happens can be done by checking the two following conditions

$$\begin{aligned} & \exists x_n^p \in X_n^p \text{ s.t } (\mathbf{K}_{q_n^p}\mathbf{B}_{y_n}^\top) = 0 \\ \text{or } & \exists x_n^p \in X_n^p \text{ s.t } (\mathbf{K}_{q_n^p}\mathbf{C}_{y_n}^\top) = 0 \end{aligned}$$

We will present detailed developments and computations for the first condition and then present the result for the second one (which can be developed in a similar manner). The definition of $\mathbf{K}$ ensures that only elements of value 1 can change to zero after a query,

since reducing $X_n^p$ can only lead to the fact that a non-empty intersection with $A_h$ becomes empty. Furthermore, (26) implies that if $k^h = 1$ then for all dimensions $p = 1, \ldots, P$, we have $X_n^p \cap A_h^p \neq \emptyset$. Such an observation ensures that the results after performing $q_n^p$, $k^h = 0$ if and only if $\exists x_n^p \in X_n^p$ s.t $x_n^p \cap A_h^p = \emptyset$, that is if the intersection with $A_h$ on dimension $p$ can become empty after querying $X_n^p$. It then implies that the condition $\exists x_n^p \in X_n^p$ s.t $(\mathbf{K}_{q_n^p} \mathbf{B}_{y_n}^\top) = 0$ is equivalent to the following condition

$$\exists x_n^p \in X_n^p \text{ s.t } x_n^p \cap A_h^p = \emptyset, \forall h \text{ where } k^h b_{y_n}^h = 1,$$

or, in other words, there is a value $x_n^p \in X_n^p$ s.t $x_n^p$ does not belong to any of $A_h^p$ for which the condition $k^h b_{y_n}^h = 1$ holds, that is for this value the resulting hyper-cube intersects with no leaves having $y_n$ as prediction. Such a condition comes down to check whether the following assertion is true:

$$X_n^p \setminus \left( \cup_{k^h b_{y_n}^h = 1} A_h^p \right) \neq \emptyset. \tag{32}$$

Similarly, to determine whether

$$\exists x_n^p \in X_n^p \text{ s.t } (\mathbf{K}_{q_n^p} \mathbf{C}_{y_n}^\top) = 0\,,$$

we can simply investigate whether

$$\exists x_n^p \in X_n^p \text{ s.t } x_n^p \cap A_h^p = \emptyset, \forall h \text{ when } k^h c_{y_n}^h = 1,$$

which can be done by checking the condition

$$X_n^p \setminus \left( \cup_{k^h c_{y_n}^h = 1} A_h^p \right) \neq \emptyset. \tag{33}$$

The general problem we have to solve is to check whether an interval $X_n^p = \left[a_n^p, b_n^p\right]$ contains a value that is outside the union of some collection of intervals $\left[\underline{d}^i, \overline{d}^i\right]$ (here, the intervals $A_h^p$ satisfying the conditions in (32) and (33)). Once we notice this, we can rewrite the computational problem in the following form

$$\begin{aligned} &\left[a_n^p, b_n^p\right] \setminus \cup_{i=1}^I \left[\underline{d}^i, \overline{d}^i\right] \neq \emptyset\,, \\ &\text{when } \forall i = 1, \ldots, I, \left[a_n^p, b_n^p\right] \cap \left[\underline{d}^i, \overline{d}^i\right] \neq \emptyset. \end{aligned} \tag{34}$$

The intuitive idea is that (34) is not satisfied if and only if $\cup_{i=1}^I \left[\underline{d}^i, \overline{d}^i\right]$ is a closed interval including $\left[a_n^p, b_n^p\right]$. Then to check whether (34) is satisfied, we just have to firstly check whether $\cup_{i=1}^I \left[\underline{d}^i, \overline{d}^i\right]$ is a closed interval, and if it is, whether it includes $\left[a_n^p, b_n^p\right]$. To check that $\cup_{i=1}^I \left[\underline{d}^i, \overline{d}^i\right]$ is a closed interval comes down to check whether there is a gap in the union of intervals. Let $\{\underline{d}^{(1)}, \ldots, \underline{d}^{(I)}\}$ be the ordered list of lower bounds, or starts of intervals. A gap happens if, when increasing values from $a_n^p$ to $b_n^p$, all intervals that have been opened

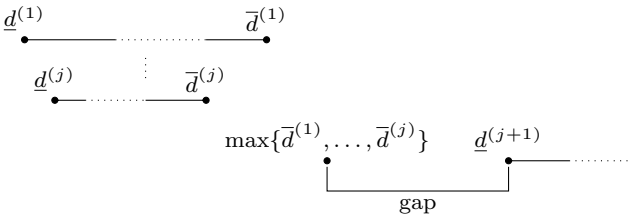


**Fig. 4** Case where $\cup_{i=1}^I \left[\underline{d}^i, \overline{d}^i\right]$ is not an interval

are closed before another one starts (as illustrated in Figure 4). In formal terms, there exists an index $j$ s.t.

$$\left|\{\overline{d}^i : \overline{d}^i < \underline{d}^{(j)}\}\right| = j - 1\,,$$

which expresses the fact that before the $j$th interval $[\underline{d}^{(j)}, \overline{d}^{(j)}]$ starts, the $j-1$ previous ones are closed, hence their union is not a closed interval. Provided $\cup_{i=1}^I \left[\underline{d}^i, \overline{d}^i\right]$ is a closed interval, then checking whether it includes $\left[a_n^p, b_n^p\right]$ can simply be done by checking that

$$\underline{d}^{(1)} \leq a_n^p \leq b_n^p \leq \overline{d}^{(I)}.$$

For a given interval $\left[a_n^p, b_n^p\right]$ and a set of interval $\left\{\left[\underline{d}^i, \overline{d}^i\right] | i \in [I]\right\}$, then whether there is a value within $\left[a_n^p, b_n^p\right]$ that is not included in $\cup_i \left[\underline{d}^i, \overline{d}^i\right]$ (i.e., whether condition (34) is satisfied) can be checked using Algorithm 6.

**Algorithm 6:** Checking whether the condition (34) is satisfied.

**Input:** $\left[a_n^p, b_n^p\right]$, sets $\left\{\left[\underline{d}^i, \overline{d}^i\right] | i \in [I]\right\}$ s.t, for $\forall i$, $\left[a_n^p, b_n^p\right] \cap \left[\underline{d}^i, \overline{d}^i\right] \neq \emptyset$
**Output:** Return $In = 1$ if (34) is satisfied and 0 otherwise

```
Order {d^1, ..., d^I} into {d^(1), ..., d^(I)} ;
foreach i ∈ [I] do
    if |{d̄^k : d̄^k < d^(i)}| = i − 1, then
        Return In = 1 and Stop the Algorithm
if min_i d^i > a_n^p then
    Return In = 1 and Stop the Algorithm
else if b_n^p > max_i d̄^i then
    Return In = 1 and Stop the Algorithm
Return In = 0;
```

Let us now illustrate how to practically determine the single effect using a simple example.

*Example 6* Consider the tree $m_k$ and two instances $\mathbf{X}_1$, $\mathbf{X}_2$ illustrated in Figure 3. Instance $\mathbf{X}_1$ is precise w.r.t. the model $m_k$, hence querying its feature is useless for

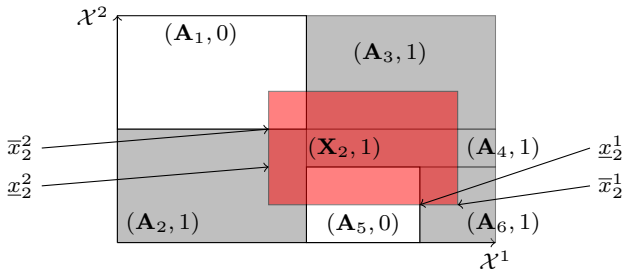


**Fig. 5** Example of determining the single effect

this model. We then focus on determining the effect of querying the features of $\mathbf{X}_2$.

Using (29)-(31), the information vectors associated to $\mathbf{X}_2$ are

$$\begin{aligned}\mathbf{K} &= (1,1,1,1,1,1)\,,\\ \mathbf{B}_{y_2} &= (1,0,0,0,1,0)\,,\\ \mathbf{C}_{y_2} &= (0,1,1,1,0,1)\,.\end{aligned}$$

Let us now investigate whether $\mathbf{X}_2$ can become precise (w.r.t. the model $m_k$) by querying its feature $X_2^1$. We have that

$$\cup_{k^h b_{y_n}^h = 1} A_h^1 = A_1^1 \cup A_5^1$$

as leaves $A_1$ and $A_5$ are overlapping with $\mathbf{X}_2$ and predict a different class from its true one. We can see on the picture that $A_1^1 \cup A_5^1$ is a closed interval that does not includes $X_2^1$. Then, for any value $x_2^1$ belonging to the interval $(\underline{x}_2^1, \overline{x}_2^1]$ as illustrated in the Figure 5, we have that $\mathbf{K}_{q_n^p}\mathbf{B}_{y_2}^\top = 0$. In other words, we have that instance $\mathbf{X}_2$ can become a precise instance after querying its feature $X_2^1$.

Similarly, for the case of querying $X_2^2$, we have that

$$\cup_{k^h b_{y_n}^h = 1} A_h^2 = A_1^2 \cup A_5^2.$$

Since $A_1^2 \cup A_5^2$ is not a closed interval, then, for any value $x_2^2$ belonging to the interval $(\underline{x}_2^2, \overline{x}_2^2)$ (illustrated in the Figure 5), we have that $\mathbf{K}_{q_n^p}\mathbf{B}_{y_2}^\top = 0$.

Finally, we conclude that instance $\mathbf{X}_2$ can become a precise instance after querying either $X_2^1$ or $X_2^2$.

## 5.3 Pairwise risk bounds and effect

This section focuses on how to compute, for a pair of models $m_k$ and $m_l$, the corresponding pairwise risk bounds $\underline{\ell}_{k-l}(y_n, \mathbf{X}_n)$ for all instance $\mathbf{X}_n$ and whether a query $q_n^p$ can increase this risk. In a way similar to the case of set-valued labels (Section 4.3.2), computations will be treated in two cases: when the instance is imprecise w.r.t. only one model; and when it is imprecise for both.

### *5.3.1 Imprecision with respect to one model*

In case an instance $\mathbf{X}_n$ is imprecise w.r.t. one model (either $m_k$ or $m_l$), the pairwise risk bound $\underline{\ell}_{k-l}(y_n, \mathbf{X}_n)$ can be determined in a way similar to the case of set-valued labels (Proposition 3). Note that this bound is, in the context of imprecise features, defined as:

$$\underline{\ell}_{k-l}(y_n, \mathbf{X}_n) = \inf_{\mathbf{x}_n \in \mathbf{X}_n} \big[\ell(y_n, m_k(\mathbf{x}_n)) - \ell(y_n, m_l(\mathbf{x}_n))\big].$$

**Proposition 6** *Given $(\mathbf{X}_n, y_n)$, and two models $m_k$ and $m_l$ s.t $\mathbf{X}_n$ is imprecise w.r.t. one and only one model, then the following results hold*

- *if $\mathbf{X}_n$ is imprecise w.r.t. $m_l$, then*

$$\underline{\ell}_{k-l}(y_n, \mathbf{X}_n) = \ell(y_n, m_k(\mathbf{X}_n)) - 1\,. \tag{35}$$

- *if $\mathbf{X}_n$ is imprecise w.r.t. $m_k$, then*

$$\underline{\ell}_{k-l}(y_n, \mathbf{X}_n) = -\ell(y_n, m_l(\mathbf{X}_n))\,. \tag{36}$$

*Proof* Similar to proof of Proposition 3. □

Then a query $q_n^p$ will have an effect $J_{q_n^p}(m_k, m_l) = 1$ if either $q_n^p$ increases $\underline{\ell}(y_n, m_l(\mathbf{X}_n))$ or $q_n^p$ decreases $\overline{\ell}(y_n, m_k(\mathbf{X}_n))$. The detailed arguments can be found in the next proposition.

**Proposition 7** *Given $(\mathbf{X}_n, y_n)$ and two models $m_k$ and $m_l$ s.t. $\mathbf{X}_n$ is imprecise w.r.t. one and only one model, then $J_{q_n^p}(m_k, m_l) = 1$ if and only if one of the following conditions holds*

- *if $\mathbf{X}_n$ is imprecise w.r.t. $m_k$, then $J_{q_n^p}(m_k, m_l) = 1$ if and only if Equation (33) holds for the model $m_k$.*
- *if $\mathbf{X}_n$ is imprecise w.r.t. $m_l$, then $J_{q_n^p}(m_k, m_l) = 1$ if and only if Equation (32) holds for the model $m_l$.*

*Proof* Let us start with the case when $\mathbf{X}_n$ is imprecise w.r.t. model $m_k$. The condition that Equation (33) holds for the model $m_k$ simply implies that after performing a query $q_n^p$, the loss $\ell(y_n, m_k(\mathbf{X}_n))$ becomes precisely 1. Hence it is clear that the pairwise risk bound is increased.

Similarly, when $\mathbf{X}_n$ is imprecise w.r.t. model $m_l$, that Equation (32) holds implies that after performing a query $q_n^p$, the loss $\ell(y_n, m_l(\mathbf{X}_n))$ is precisely 0 which results in increasing $\underline{\ell}_{k-l}(y_n, \mathbf{X}_n)$. □

### *5.3.2 Imprecision with respect to both models*

Note that when an instance $\mathbf{X}_n$ is imprecise with respect to both models $m_k$ and $m_l$, the pairwise risk bounds $\underline{\ell}_{k-l}(y_n, \mathbf{X}_n)$ can get values in $\{-1, 0, 1\}$. Let us denote by $\lambda_h^{m_l}$ the label associated to the partition

$\mathbf{A}_h$ of a tree $m_l$, then the relation between $\mathbf{X}_n$ and leaves of $m_k$ and $m_l$ can be encoded in matrix form as follows

$$\mathbf{W}^{k,l} = \left(w_{i,j}^{k,l}\right)_{i=1,\ldots,H_k, j=1,\ldots,H_l} \tag{37}$$

where the values of $w_{i,j}^{k,l}$ and the corresponding conditions are

| $w_{i,j}^{k,l}$ | Corresponding condition |
|---|---|
| 2 | $\mathbf{X}_n \cap \mathbf{A}_i^{m_k} \cap \mathbf{A}_j^{m_l} = \emptyset$ |
| 1 | $\mathbf{X}_n \cap \mathbf{A}_i^{m_k} \cap \mathbf{A}_j^{m_l} \neq \emptyset, \lambda_i^{m_k} \neq y_n, \lambda_j^{m_l} = y_n$ |
| 0 | $\mathbf{X}_n \cap \mathbf{A}_i^{m_k} \cap \mathbf{A}_j^{m_l} \neq \emptyset, \lambda_i^{m_k} = \lambda_j^{m_l}$ |
| $-1$ | $\mathbf{X}_n \cap \mathbf{A}_i^{m_k} \cap \mathbf{A}_j^{m_l} \neq \emptyset, \lambda_i^{m_k} = y_n, \lambda_j^{m_l} \neq y_n$ |

(38)

It is easy to see that the matrix $\mathbf{W}^{k,l}$ covers all possible values of $\ell_{k-l}(y_n, \mathbf{X}_n)$, with 2 being an arbitrary value to denote that $\mathbf{X}_n$ prediction does not depend on $\mathbf{A}_i^{m_k} \cap \mathbf{A}_j^{m_l}$. The pairwise lower risk bound is then simply the minimum value of elements in matrix $\mathbf{W}^{k,l}$ i.e.,

$$\underline{\ell}_{k-l}(y_n, \mathbf{X}_n) = \min_{i,j} w_{i,j}^{k,l}. \tag{39}$$

Before going to determine whether a query $q_n^p$ can increase the pairwise risk bound $\underline{\ell}_{k-l}(y_n, \mathbf{X}_n)$, note that whether $\mathbf{X}_n \cap \mathbf{A}_i^{m_k} \cap \mathbf{A}_j^{m_l} = \emptyset$ can be easily determined as a consequence of Equation (25), as we have

$$\mathbf{X}_n \cap \mathbf{A}_i^{m_k} \cap \mathbf{A}_j^{m_l} = \times_{p=1}^{P} X_n^p \cap A_{i,p}^{m_k} \cap A_{j,p}^{m_l}.$$

Then for an instance $\mathbf{X}_n$, its corresponding pairwise risk bound w.r.t. two models $m_k$ and $m_l$ can be determined explicitly using Equations (37) and (38). A query $q_n^p$ can increase the pairwise risk bound if and only if it can increase the value of all elements of value $\min_{i,j} w_{i,j}^{k,l}$. Let

$$\mathbf{S}_{min} = \left\{w_{i',j'}^{k,l} | w_{i',j'}^{k,l} = \min_{i,j} w_{i,j}^{k,l}\right\} \tag{40}$$

be the set of such elements, then $J_{q_n^p}(m_k, m_l) = 1$ if $\exists$ $x_n^p \in X_n^p$ s.t after querying $X_n^p$, all elements in the set $\mathbf{S}_{min}$ are increased.

Note that for a given pair $\left(\mathbf{A}_i^{m_k}, \mathbf{A}_j^{m_l}\right)$, using (26), we have that their intersection is

$$\begin{aligned}\mathbf{A}_{i,j} = \mathbf{A}_i^{m_k} \cap \mathbf{A}_j^{m_l} &= \times_{p=1}^{P} A_{i,p}^{m_k} \cap A_{j,p}^{m_l} \\ &:= \times_{p=1}^{P} A_{i,j}^{p},\end{aligned} \tag{41}$$

where $A_{i,j}^p$ is a closed interval, for $p = 1, \ldots, P$. Furthermore, a query $q_n^p$ can increase the pairwise risk bound if $\exists$ $x_n^p \in X_n^p$ s.t $x_n^p \notin A_{i,j}^p$, for all $w_{i,j}^{k,l} \in \mathbf{S}_{min}$. The next Proposition provides a practical procedure to check whether such a condition holds.

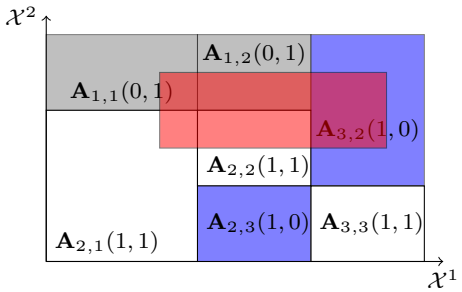


**Fig. 6** Example of determining the pairwise effect

**Proposition 8** *Given a training instance $\mathbf{X}_n$ which is imprecise w.r.t. both models $m_k$ and $m_l$, the corresponding $\mathbf{W}^{k,l}$ matrix, assuming that $\min_{i,j} w_{i,j}^{k,l} < 1$, then $J_{q_n^p}(m_k, m_l) = 1$ if and only if*

$$X_n^p \setminus \left(\cup_{i,j | w_{i,j}^{k,l} \in \mathbf{S}_{min}} A_{i,j}^p\right) \neq \emptyset. \tag{42}$$

*Proof* Let us first note that if (42) holds, then $\exists$ $x_n^p \in X_n^p$ s.t $x_n^p \notin A_{i,j}^p$, for all $w_{i,j}^{k,l} \in \mathbf{S}_{min}$. Then the corresponding elements $w_{i,j}^{k,l}$ are increased to be 2. It is then resulting in the increasing of $\min_{i,j} w_{i,j}^{k,l}$, or in other words, the pairwise risk bound $\underline{\ell}_{k-l}(y_n, \mathbf{X}_n)$. □

Checking whether Equation (42) is true can easily be reformulated in the form of Equation (34), Algorithm 6 can then be used to perform the check. In practice, such a check is quadratic in the number of leaves of the models $m_k$, $m_l$, which remains affordable from a computational standpoint. The next example illustrates how to practically determine the effect of queries on the pairwise risk bounds.

*Example 7* Assume that we have two models $m_1$ and $m_2$ with 3 leaves each, whose intersection of partition elements is illustrated in Figure 6.

Instance $\mathbf{X}$ covers the red region and has label $y = 1$. From Figure 6, we can see that $\mathbf{X}$ is imprecise w.r.t. both models $m_1$ and $m_2$, and its corresponding information matrix $\mathbf{W}^{1,2}$ can be determined as follows

$$\mathbf{W}^{1,2} = \begin{bmatrix} 1 & 1 & 2 \\ 0 & 0 & 2 \\ 2 & -1 & 2 \end{bmatrix}$$

Then it is clear that $\underline{\ell}_{1-2}(y, \mathbf{X}) = -1$ and

$$\mathbf{S}_{min} = \left\{w_{i',j'}^{k,l} | w_{i',j'}^{k,l} = \min_{i,j} w_{i,j}^{k,l}\right\} = w_{3,2}^{1,2}.$$

Let us now investigate whether the empirical risk bound $\underline{\ell}_{1-2}(y, \mathbf{X})$ can increase by querying the features of $\mathbf{X}$. It is easy to see that $A_{3,2}^1$ is a closed interval that does not include $X^1$. Then we always can find value

$x^1 \in X^1$ s.t $A^1_{3,2} \cap x^1 = \emptyset$. In other words, we can increase the bound $\underline{\ell}_{1-2}(y, \mathbf{X})$ by querying $X^1$.

However, as $A^2_{3,2}$ is a closed interval that includes $X^2$, there is no value $x^2 \in X^2$ s.t $A^2_{3,2} \cap x^2 = \emptyset$, or in other words, the bound $\underline{\ell}_{1-2}(y, \mathbf{X})$ can not be increased by querying $X^2$.

### 5.4 Algorithms

Algorithm 7 summarizes how to determine the optimal query for a single querying step in case of imprecise features. It is very similar to Algorithm 2, but takes different sub-routines specific to the case of partially known features.

**Algorithm 7:** A single step to query interval-valued data.

**Input:** Training data set $\{(\mathbf{X}_n, y_n)\}_{n=1}^N$, label set $\Omega = \{\lambda_1, \ldots, \lambda_G\}$, undominated model set $\mathcal{M}$.
**Output:** The optimal query $q_{n^*}$

1 **foreach** $m_k \in \mathcal{M}$ **do**
2  Compute empirical risk $[\underline{R}(m_k), \overline{R}(m_k)]$ bounds using Alg. 8;
3 Determine the best model $m_{k^*}$ and the undominated model set $\mathcal{M}^*$ using Alg. 4;
4 **foreach** $n \in [N]$ **do**
5  Determine $(E_{q_n^1}(m_{k^*}), \ldots, E_{q_n^P}(m_{k^*}))$ using Alg. 9 with model $m_{k^*}$;
6  Determine the cumulative pairwise effects $(J_{q_n^1}, \ldots, J_{q_n^P})$ using Alg. 10;
7  **foreach** $p \in [P]$ **do**
8  $Value(q_n^p) = E_{q_n^p}(m_{k^*}) + J_{q_n^p}$;
9 Determine $(n^*, p^*) = \arg\max_{(n,p)} Value(q_n^p)$;

**Algorithm 8:** Compute the empirical risk bounds $(\underline{R}(m_k), \overline{R}(m_k))$.

**Input:** Training data set $\{(\mathbf{X}_n, y_n)\}_{n=1}^N$, label set $\Omega = \{\lambda_1, \ldots, \lambda_G\}$, model $m_k$.
**Output:** Empirical risk bounds $(\underline{R}(m_k), \overline{R}(m_k))$

1 $\underline{R}(m_k) = 0$, $\overline{R}(m_k) = 0$;
2 **foreach** $n \in [N]$ **do**
3  Compute $\mathbf{K}$, $\mathbf{B}_{y_n}$ and $\mathbf{C}_{y_n}$ using (29)-(31);
4  **if** $\mathbf{K}\mathbf{C}^\top_{y_n} = 0$ **then** $\underline{R}(m_k) = \underline{R}(m_k) + 1$, $\overline{R}(m_k) = \overline{R}(m_k) + 1$;
5  **if** $(\mathbf{K}\mathbf{B}^\top_{y_n})(\mathbf{K}\mathbf{C}^\top_{y_n}) \neq 0$ **then** $\overline{R}(m_k) = \overline{R}(m_k) + 1$;

Algorithm 8 summarises how risk bounds, from which can be deduced the best potential model (through Algorithm 4, that remains unchanged), can be computed. Algorithms 9 and 10 describe how potential effects of querying an instance $(\mathbf{X}_n, y_n)$, respectively on empirical risk bounds and on pairwise risk bound, can be determined. Note that Algorithm 10 computes the sum of the pairwise effects between the best potential model $m_{k^*}$ and the other ones. Let us now look at the complexity of Algorithm 8, assuming that all decision trees have $H$ leaves. Before doing that, note that checking whether two hyper-cubes do intersect is in $\mathcal{O}(P)$, according to Equation (26). Lines 2-3 are in $\mathcal{O}(S \times N \times H \times P)$, since Algorithm 8 is in $\mathcal{O}(N \times H \times P)$, as computing vector $\mathbf{K}$ (line 3 of Algorithm 8) is in $\mathcal{O}(H \times P)$. Algorithm 4 remains in $\mathcal{O}(S)$. Lines 4-9 of Algorithm 8 is in $\mathcal{O}(N \times S \times P \times H^4)$: indeed, in Algorithm 10, lines 7-9 are in $\mathcal{O}(P \times H^4)$, as we must apply Algorithm 6 to at most $H^2$ intervals.

In particular, Algorithm 10 treats both the cases of an instance that is imprecise with respect to both models, as well as the other cases (other loops): Line 2 determines whether the instance is imprecise w.r.t $m_{k^*}$, Line 4 whether it is imprecise w.r.t $m_k$. So Lines 4-9 correpond to imprecision with respect to both models, lines 10-13 to imprecision w.r.t only $m_{k^*}$, and lines 15-19 w.r.t only $m_k$.

**Algorithm 9:** Determine the single effects $(E_{q_n^1}(m), \ldots, E_{q_n^P}(m))$.

**Input:** Training instance $(\mathbf{X}_n, y_n)$, a model $m$.
**Output:** The single effects $(E_{q_n^1}(m), \ldots, E_{q_n^P}(m))$

1 Initialize $(E_{q_n^1}, \ldots, E_{q_n^P}) = (0, \ldots, 0)$;
2 **if** $\underline{\ell}(m(\mathbf{X}_n), y_n) \neq \overline{\ell}(m(\mathbf{X}_n), y_n)$ **then**
3  **foreach** $p \in [P]$ *with* $\|X_n^p\| > 0$ **do**
4  $In \leftarrow$ Alg. 6 with inputs $X_n^p$, $\{A_h^p | k^h c^h_{y_n} = 1\}$;
5  **if** $In = 1$ **then** $E_{q_n^p}(m) = 1$;
6  $In \leftarrow$ Alg. 6 with inputs $X_n^p$, $\{A_h^p | k^h_{y_n} b^h_{y_n} = 1\}$;
7  **if** $In = 1$ **then** $E_{q_n^p}(m) = 1$;

The overall complexity is polynomial in all parameters, which may be considered as reasonable when the number of partial data, and the complexity of the trees both remain limited. Also, this is a worst-case complexity, assuming that every feature of every training data is imprecise, and that every resulting hyper-cube intersect all leaves of all the decision trees in $\mathcal{M}$. In practice, we may expect partial features to be quite less numerous, as well as their intersections with tree leaves.

It should also be noticed that since the models will not change during the race, and that data will only be queried iteratively, one can in principle compute all matrices at the start of the race, and then proceed to

**Algorithm 10:** Determine the cumulative pairwise effects $(J_{q_n^1}, \ldots, J_{q_n^P})$.

**Input:** Training instance $(\mathbf{X}_n, y_n)$, undominated model set $\mathcal{M}$, best model $m_{k^*}$.
**Output:** The cumulative pairwise effects $(J_{q_n^1}, \ldots, J_{q_n^P})$

Initialize $(J_{q_n^1}, \ldots, J_{q_n^P}) = (0, \ldots, 0)$;
**if** $\underline{\ell}(m_{k^*}(\mathbf{X}_n), y_n) \neq \overline{\ell}(m_{k^*}(\mathbf{X}_n), y_n)$ **then**
  **foreach** $m_k \in \mathcal{M}$ *and* $k \neq k^*$ **do**
    **if** $\underline{\ell}(m_k(\mathbf{X}_n), y_n) \neq \overline{\ell}(m_k(\mathbf{X}_n), y_n)$ **then**
      Compute matrix $\mathbf{W}^{k,k^*}$ defined in (37);
      **if** $\min \mathbf{W}^{k,k^*} < 1$ **then**
        **foreach** $p \in [P]$ *and* $\|X_n^p\| > 0$ **do**
          $In \leftarrow$ Alg. 6 with inputs $X_n^p$, $\{A_{i,j}^p : w_{i,j}^{k,k^*} = \min \mathbf{W}^{k,k^*}\}$;
          **if** $In = 1$ **then** $J_{q_n^p} = J_{q_n^p} + 1$;
    **else**
      **foreach** $p \in [P]$ *and* $\|X_n^p\| > 0$ **do**
        $In \leftarrow$ Alg. 6 with inputs $X_n^p$, $\{A_h^p$ of $m_{k^*} | k_{y_n}^h b_{y_n}^h = 1\}$;
        **if** $In = 1$ **then** $J_{q_n^p} = J_{q_n^p} + 1$;
**else**
  **foreach** $m_k \in \mathcal{M}$ *and* $k \neq k^*$ **do**
    **if** $\underline{\ell}(m_k(\mathbf{X}_n), y_n) \neq \overline{\ell}(m_k(\mathbf{X}_n), y_n)$ **then**
      **foreach** $p \in [P]$ *and* $\|X_n^p\| > 0$ **do**
        $In \leftarrow$ Alg. 6 with inputs $X_n^p$, $\{A_h^p$ of $m_k | k_{y_n}^h c_{y_n}^h = 1\}$;
        **if** $In = 1$ **then** $J_{q_n^p} = J_{q_n^p} + 1$;

a minimal update at each query, thus considerably reducing the time to determine optimal queries. Finally, it should be noticed that querying data mainly makes sense when data are scarce (as an increased quantity of data improves the model accuracy even in the presence of imperfections).

## 6 Experiments

In this section, we run experiments on a "contaminated" version of 4 standard benchmark data sets as described in Table 2. To evaluate the efficiency of our proposal, we compare our racing algorithm with baseline algorithms whose details will be described separately in each setting of partial data. Note that when data are partial and in contrast with classical active learning, it is usually difficult to divide the data between a set of training data and a set of data with missing values, especially if all data are partial. This is why we will do the queries on the same data we use to train the models. As the situation where both input and output are partially given rarely happens in practice, we only focus on two settings: partiality in inputs; and partiality in outputs. In both cases and for each selection of precise data, we use a standard CART decision tree learned in which splits are selected by Gini scores. It should be noted that changing the scores, e.g. for entropy, has usually little influence on the result (Dubois et al, 2007), as most of them are refinements of a unique ordinal notion. The next two subsections present details about the experimental settings and the results for interval-valued features and set-valued labels data, respectively. It should be noted here that our primary goal is to verify that our approach query useful data, in the sense that those data are those that make the notion of best model ambiguous. Once this is confirmed, and after data have been queried, one could use other learning or aggregation methods such as ensemble learning to optimize accuracy from queried data.

**Table 2** Data set used in the experiments

| Name | # instances | # features | # classes |
|---|---|---|---|
| wine | 178 | 13 | 3 |
| breast-cancer | 569 | 30 | 2 |
| vowel | 990 | 10 | 11 |
| segment | 2310 | 19 | 7 |

### 6.1 Interval-valued features

We follow a $2 \times 5$ fold cross-validation procedure: Each data set is randomly split into 5 folds. Each fold is in turn considered as the training set $\mathbf{D}$, while other folds are used for testing $\mathbf{T}$. For each feature $x_n^p$ in the training set, a biased coin is flipped in order to decide whether or not this example will be contaminated; the probability of contamination is $\delta$. The level of partiality $\delta$ is fixed to two values (0.3 and 0.6) which correspond to a low and a high level of imprecision. In case $x_n^p$ is contaminated, a width $\epsilon_n^p$ is generated from a uniform distribution on the unit interval. Then the generated interval valued data is $X_n^p = [x_n^p + \epsilon_n^p(\underline{D}^p - x_n^p), x_n^p + \epsilon_n^p(\overline{D}^p - x_n^p)]$ where $\underline{D}^p = \min_n(x_n^p)$ and $\overline{D}^p = \max_n(x_n^p)$.

*Example 8* Assume that the initial precise observed value is $x = 1$, that the domain is $[\underline{D}, \overline{D}] = [0, 10]$, and that we have randomly picked $\epsilon = 0.5$. In this case, the resulting interval-valued data is $X = [0.5, 5.5]$.

To get the initial set of models for the race, we randomly generate 100 completions of interval-valued data. From each completion, one tree model (with a minimal number of training observations in any terminal node fixed to 3 for first two small data sets and 5 for the two later ones) is trained and the set of such models is considered as the initial set of undominated models. The

budget will be fixed to be the total number of partially featured values. After each query, we discard the dominated models and determine the best potential model. In case of multiple minimum risk models, the one with a minimum value of $\overline{R}_m$ will be chosen as the best potential model.

The two following baseline algorithms are employed to query interval-valued data and make comparison about the evolution of the size of the sets of undominated models and the performance of the best potential model:

- **a random querying strategy** where, at each iteration, the queried example and feature will be chosen randomly,
- and the **most partial querying strategy** designed such that, at each iteration, examples with the largest imprecision will be queried.

Those baselines are quite straightforward, and we expect our method to perform better than those. As we mentioned in the introduction, we are not aware of other methods that intends to query the most influential data when features are partial, hence can only compare ourselves to those standard baselines in the case of partial features.

In practice, it may be the case that not all features appear in the set of racing trees. In those cases, keeping all the features in the instances would disadvantage both random and most partial querying in the race, since in this latter only the features present in the trees are relevant (i.e., will play a role to discard racing models). To make a comparable setting and to not give an unfair advantage to our method, we thus eliminate the features that do not appear in the trained trees.

In order to evaluate the performances of those different strategies, we will use three measures:

- the similarity of the best potential model $m_{k^*}$ with a reference model $m_{ref}$ is computed on the precise test set $\mathbf{T} = \{(\mathbf{x}_t, y_t)\}_{t=1}^{T}$. This similarity is computed as

$$\frac{|\{t \in [T] | m_{ref}(\mathbf{x}_t) = m_{k^*}(\mathbf{x}_t)\}|}{T}.$$

  This similarity is 1 if the two models make identical predictions on the test set (hence have the same performances), and 0 if they systematically disagree. The reference model is chosen to be the one in the initial undominated set that has the best accuracy on the fully precise training set. It is thus the model towards which any querying strategy, and the race in particular, should converge;
- the size of the undominated set $\mathcal{M}^*$, that should decrease as fast as possible, both to ensure computational efficiency and model performances.

| Data set | $\delta = 0.3$ | | | $\delta = 0.6$ | | |
|---|---|---|---|---|---|---|
| # undominated models | | | | | | |
| | Rac. | Rand. | Most | Rac. | Rand. | Most |
| Wine | **11.6** | 96.1 | 82.8 | **8.8** | 100 | 100 |
| Breast | **5.8** | 98.1 | 100 | **44.2** | 100 | 100 |
| Vowel | **25.9** | 100 | 100 | 100 | 100 | 100 |
| Segment | **1.5** | 100 | 100 | **48.8** | 50.6 | 50.6 |
| Similarity to best model | | | | | | |
| | Rac. | Rand. | Most | Rac. | Rand. | Most |
| Wine | **1** | 0.96 | 0.94 | **0.95** | 0.84 | 0.83 |
| Breast | **0.97** | 0.92 | 0.92 | **1** | 0.92 | 0.94 |
| Vowel | **0.68** | 0.57 | 0.55 | 0.35 | 0.35 | **0.40** |
| Segment | **0.98** | 0.89 | 0.86 | 0.74 | 0.75 | **0.8** |

**Table 3** Partial features: results after querying 10% of the partial data

The 5-folds process is repeated 2 times. The average size of the sets of models and the average similarity of the best potential model are reported.

The experimental results are presented in Figure 7 and 8. They show that, using the racing approach, the size of the undominated set can be quickly reduced and that the best potential model converges very fast to the desired model when knowing a small number of the precise data. The reduction of the size of the set is much slower for other querying strategies. This is true for the four tested data sets, and the advantage of using the racing approach is obvious whether we have little ($\delta = 0.3$) or a lot ($\delta = 0.6$) of imprecision. The exception observed for high imprecision ($\delta = 0.6$) in the case of the segment data set is due to the fact that few features are used in the different trees, hence all models are quite similar, and all querying strategies focus on those features, converging at comparable speeds. For convenience, Table 3 also summarizes the obtained results after having queried 10% of the partial data. From the table, we can see that our approach is usually the best one, both at discarding sub-optimal models and at getting closer to the optimal one, except for the Vowel and Segment data sets in case of high imprecision. For the Segment data set, Figure 8.h confirms that all methods are comparable, while for the Vowel data set Figure 8.f shows that our approach is outperforming the others, yet only at the end of the querying process.

### 6.2 Set-valued labels

We perform on the same data sets as before (cf. Table 2) and the $2 \times 5$ cross-validation procedure as described for partially featured data (without the feature filtering step as we only consider the partial labels here).

In order to contaminate a given data set, we used the following strategy: for each example in the training

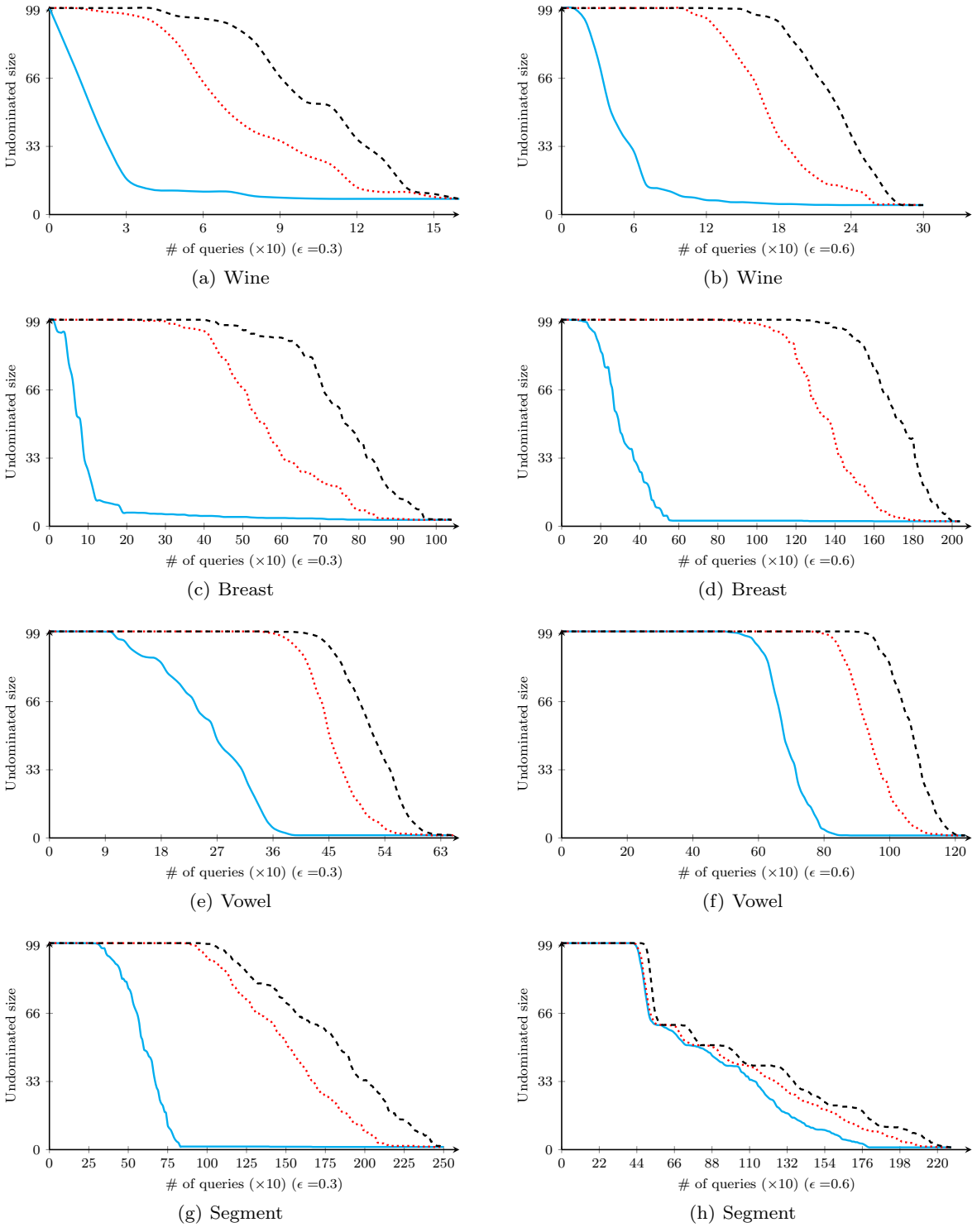


**Fig. 7** Interval-valued features: Size of undominated model sets

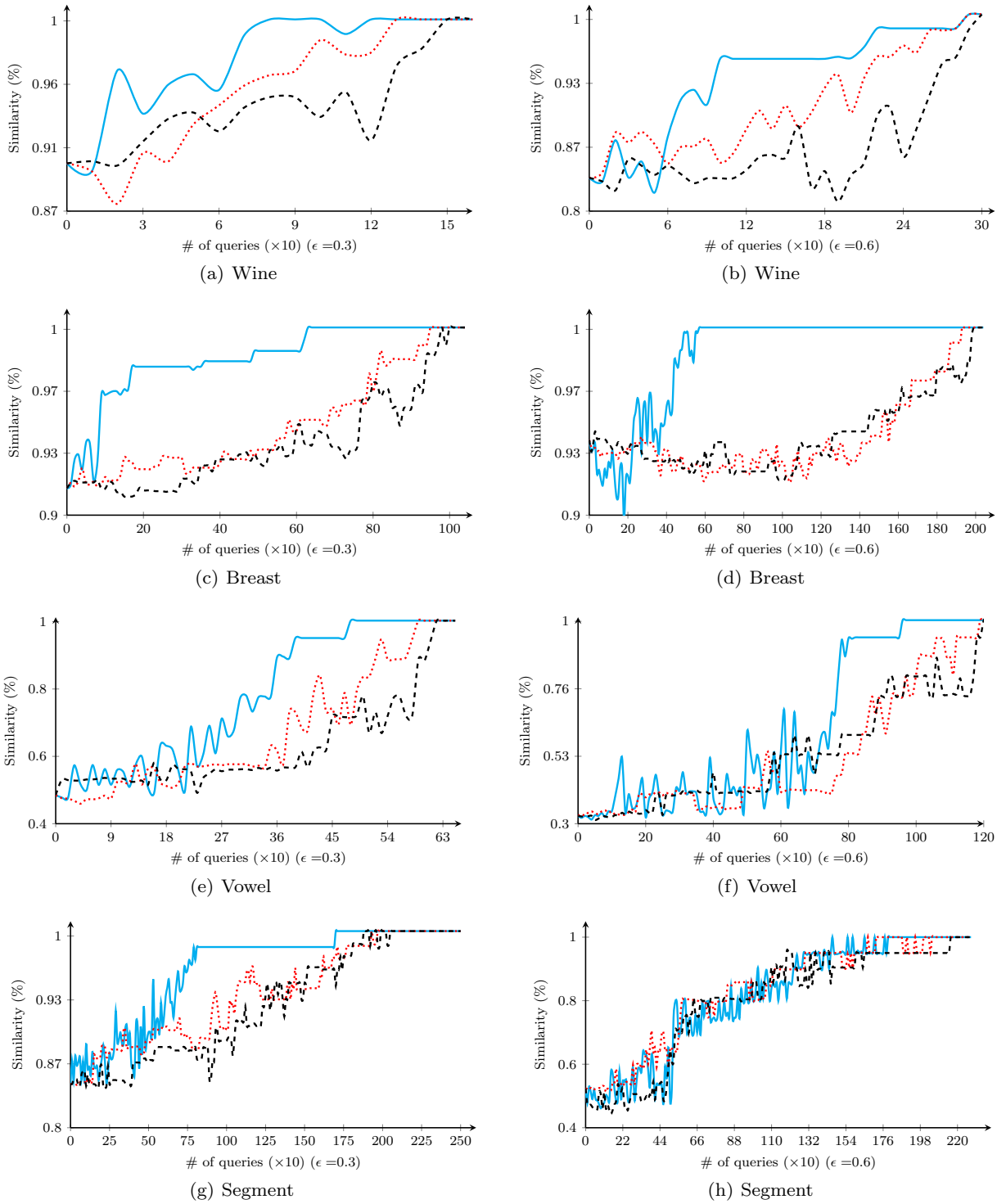


**Fig. 8** Interval-valued features: Similarity between the current best and reference models

set, a biased coin is flipped in order to decide whether or not this example will be contaminated; the probability of contamination is $\delta$. When an example is contaminated, the class candidates are added with probability $\eta$, independently of each other. Thus, the contamination procedure is parametrized by the probabilities $\delta$ and $\eta$, where $\delta$ corresponds to the expected fraction of imprecise examples in a data set, and $\eta$ reflects the average number of classes added to contaminated examples. The expected cardinality of a label set, in case of contamination, is given by $1+(G-1)\eta$. In all experiments, $\delta$ and $\eta$ are fixed respectively to 0.3 and 0.8. To start the race, 100 precise replacements for each imprecise labels are randomly chosen. From each selection, one classification tree is trained. Similarly to the case of partial features, the minimal number of observations in any terminal node is fixed to 3 and 5 for the first two data sets and the later ones, respectively.

To make comparisons, the two baseline querying schemes are also employed to query set valued-labels:

- **a random query** strategy where, each time, imprecise examples will be chosen randomly to be queried.
- **a query by committee** approach (QBC) in which each model is allowed to vote on the labellings of query candidates. The most informative query is considered to be the instance for which they most disagree. The disagreement measure used is the **vote entropy**:

$$\mathbf{x}^*_{\text{VE}} = \underset{\mathbf{x}}{\text{argmax}} - \sum_{g=1}^{G} \frac{V_{\mathbf{x}}(\lambda_g)}{S} \log \frac{V_{\mathbf{x}}(\lambda_g)}{S}$$

  where $V_{\mathbf{x}}(\lambda_g)$ denotes the number of models predicting class $\lambda_g$ for a given instance $\mathbf{x}$, and $S = |\mathcal{M}^*|$ denotes the number of models in the committee. QBC approaches usually give good performance in active learning settings with missing labels, and give state-of-art results (Vandoni et al, 2019).

In a way similar to the interval-valued setting, the size of the sets of models and the similarity of the best potential model $m_{k^*}$ w.r.t. to the reference model are reported and used to make comparison.

The experimental results, presented in Figures 9 and 10, show that, among the three approaches, random queries usually converge more slowly towards the reference model (except for the vowel data set), while the set of undominated models decreases similarly for all data sets and all strategies (with a slight advantage for the QBC strategy, and a poorly performing random queries for the segment data set). This contrasts with the partial feature case, where our approach significantly outperforms the others. A reason for that maybe that the case of partial labels offers much less degrees of freedom, hence the impact of the querying strategy may be quite less important than for the feature case. Table 4 summarizes the results for partial labels in the same way as Table 3 did for the features. We can see that all results are quite close in this table, as they are in the figures, and that there is essentially no big differences between the approaches.

| Data set | $\delta = 0.3, \nu = 0.8$ | | |
|---|---|---|---|
| # undominated models | | | |
| | Rac. | Rand. | QBC |
| Wine | 93.8 | 91.7 | **94.6** |
| Breast | **95.5** | 87.8 | 90.4 |
| Vowel | 100 | 100 | 100 |
| Segment | 97.4 | **94.4** | 95.6 |
| Similarity to best model | | | |
| | Rac. | Rand. | QBC |
| Wine | 0.78 | 0.9 | **0.94** |
| Breast | 0.82 | 0.88 | **0.9** |
| Vowel | 0.54 | 0.59 | **0.74** |
| Segment | 0.82 | **0.97** | **0.97** |

**Table 4** Partial labels: results after querying 10% of the partial data

Note that it would have been possible to add many other more complex baselines, yet since both a random strategy and a more complex strategy such as QBC are on par with our proposed approach, we think this is sufficient to draw some conclusions.

## 7 Conclusion

The problem of actively learning with partial data has been little explored in the literature, in particular the case of partially known features. Indeed, active learning techniques usually focus on the case where a part of the labels are completely missing, while a few are known. To solve the problem, we apply in this paper an approach based on the idea of racing algorithms to the specific case of decision trees. To do so, we have developed a number of efficient algorithms to detect which data should be queried, in order to identify as soon as possible the best model among a set of racing ones.

We have then made some experiments to study the behaviour of our approach, compared to other querying strategies, starting from the same set of initial models. Our conclusion is that our approach significantly outperforms simpler strategies in the case of partially specified features, while it achieves similar performances in the case of partially specified labels. We think that this is due to the fact that partial labels offer much less degrees of freedom to the learning algorithms, meaning that most smart strategies, or even random ones will

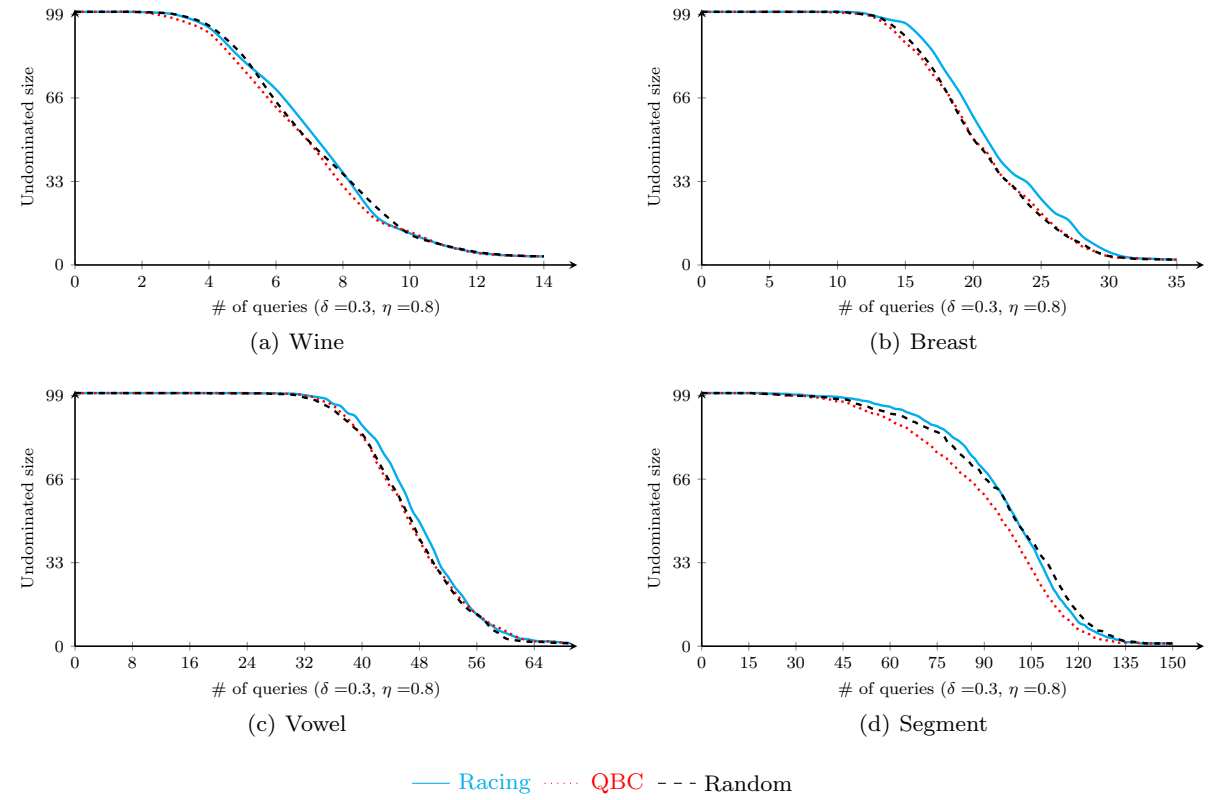


**Fig. 9** Set-valued labels: Size of undominated sets

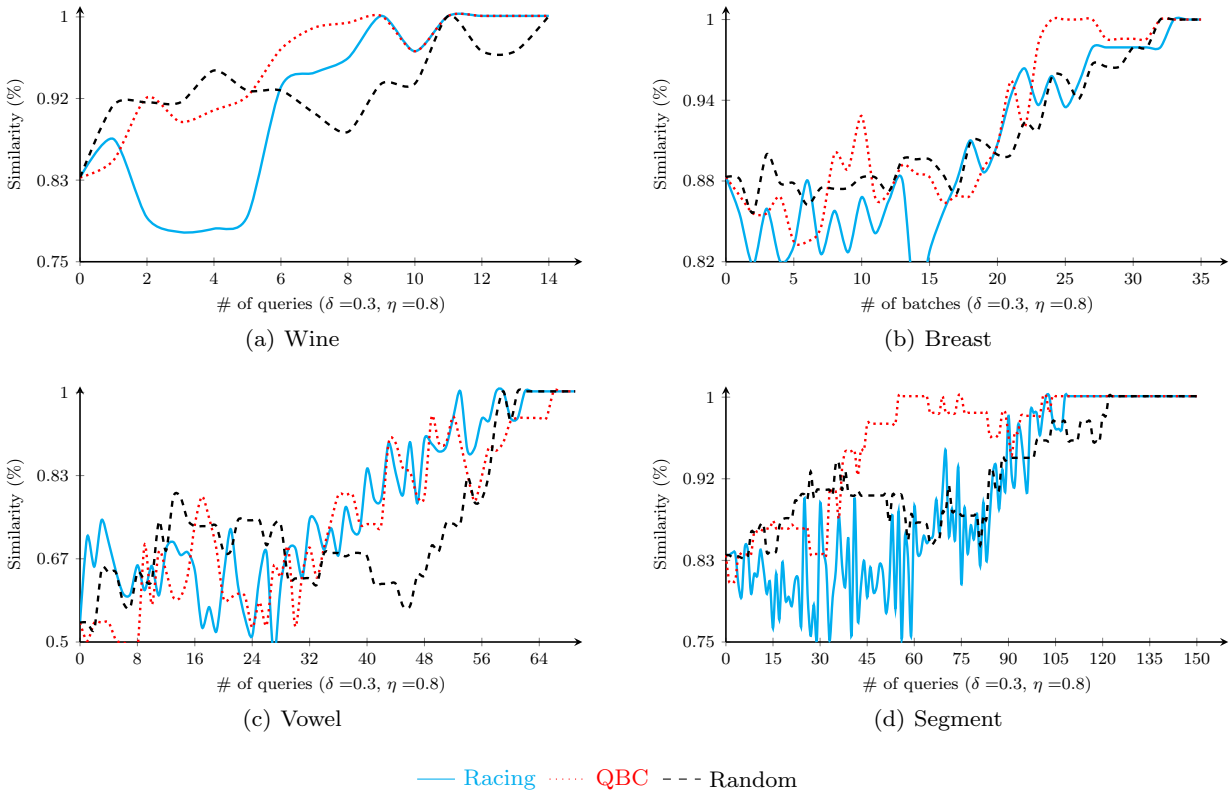


**Fig. 10** Set-valued labels: Similarity between the current best and reference models

perform similarly. This is not the case for partial features, where purely random strategies as well as querying the most partial features perform poorly.

This study considers a non-parametric classifier that can also be considered as a peculiar rule-based system, i.e., decision trees. As such, we can expect some of the conclusions and algorithmic procedures developed in this paper to also be applicable to generic rule-based systems (Hühn and Hüllermeier, 2009). Our paper therefore ideally complements our previous study (Nguyen et al, 2018), that focused on a parametric linear classifier, i.e., SVMs.

## Compliance with Ethical Standards

*Funding:* This work was carried out in the framework of Labex MS2T and UML-NET projects, which were funded by the French National Agency for Research (Reference ANR-11-IDEX-0004-02, ANR-14-CE24-0026).

*Conflict of Interest:* all authors declare that they have no conflict of interest.

*Ethical approval:* This article does not contain any studies with human participants or animals performed by any of the authors.

## References

Abe N, Mamitsuka H (1998) Query learning strategies using boosting and bagging. In: Proceedings of the fifteenth International Conference on Machine Learning (ICML), Morgan Kaufmann Pub, vol 1

Busa-Fekete R, Szörényi B, Weng P, Cheng W, Hüllermeier E (2014) Preference-based reinforcement learning: evolutionary direct policy search using a preference-based racing algorithm. Machine learning 97(3):327–351

Cabannnes V, Rudi A, Bach F (2020) Structured prediction with partial labelling through the infimum loss. In: International Conference on Machine Learning, PMLR, pp 1230–1239

Cour T, Sapp B, Jordan C, Taskar B (2009) Learning from ambiguously labeled images. In: Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, pp 919–926

Cour T, Sapp B, Taskar B (2011) Learning from partial labels. The Journal of Machine Learning Research 12:1501–1536

Dobra A, Fienberg SE (2000) Bounds for cell entries in contingency tables given marginal totals and decomposable graphs. Proceedings of the National Academy of Sciences 97(22):11,885–11,892

Dubois D, , Hullermeier E (2007) Comparing probability measures using possibility theory: A notion of relative peakedness. International Journal of Approximate Reasoning 45:364–385

Efron B (1981) Censored data and the bootstrap. Journal of the American Statistical Association 76(374):312–319

Farhangfar A, Kurgan L, Dy J (2008) Impact of imputation of missing values on classification error for discrete data. Pattern Recognition 41(12):3692–3705

Feng C, Liu MY, Kao CC, Lee TY (2017) Deep active learning for civil infrastructure defect detection and classification. In: Computing in civil engineering 2017, pp 298–306

Guillaume R, Couso I, Dubois D (2017) Maximum likelihood with coarse data based on robust optimisation. In: Proceedings of the Tenth International Symposium on Imprecise Probability: Theories and Applications (ISIPTA), pp 169–180

Heitjan DF (1993) Ignorability and coarse data: Some biomedical examples. Biometrics pp 1099–1109

Hühn J, Hüllermeier E (2009) Furia: an algorithm for unordered fuzzy rule induction. Data Mining and Knowledge Discovery 19(3):293–319

Hüllermeier E (2014) Learning from imprecise and fuzzy observations: Data disambiguation through generalized loss minimization. International Journal of Approximate Reasoning 55(7):1519–1534

Hüllermeier E, Beringer J (2006) Learning from ambiguously labeled examples. Intelligent Data Analysis 10(5):419–439

Hüllermeier E, Destercke S, Couso I (2019) Learning from imprecise data: adjustments of optimistic and pessimistic variants. In: International Conference on Scalable Uncertainty Management, Springer, pp 266–279

Lagacherie P, Cazemier DR, Martin-Clouaire R, Wassenaar T (2000) A spatial approach using imprecise soil data for modelling crop yields over vast areas. Agriculture, ecosystems & environment 81(1):5–16

Liu L, Dietterich T (2014) Learnability of the superset label learning problem. In: Proceedings of the 31st International Conference on Machine Learning (ICML), pp 1629–1637

Lobato F, Sales C, Araujo I, Tadaiesky V, Dias L, Ramos L, Santana A (2015) Multi-objective genetic algorithm for missing data imputation. Pattern Recognition Letters 68:126–131

Ma L, Destercke S, Wang Y (2016) Online active learning of decision trees with evidential data. Pattern Recognition 52:33–45

Maron O, Moore AW (1997) The racing algorithm: Model selection for lazy learners. In: Lazy learning,

Springer, pp 193–225
McDonald J, Stoddard O, Walton D (2018) On using interval response data in experimental economics. Journal of Behavioral and Experimental Economics 72:9 – 16
Nguyen VL, Destercke S, Masson MH (2017) Querying partially labelled data to improve a k-nn classifier. In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI), February 4-9, 2017, San Francisco, California, USA., pp 2401–2407
Nguyen VL, Destercke S, Masson MH (2018) Partial data querying through racing algorithms. International Journal of Approximate Reasoning 96:36 – 55
Nigam K, McCallum A (1998) Pool-based active learning for text classification. In: Proceedings of Conference on Automated Learning and Discovery (CONALD)
Prince S (1991) A model of regional primary production for use with coarse resolution satellite data. International Journal of Remote Sensing 12(6):1313–1330
Quinlan JR (1986) Induction of decision trees. Machine learning 1(1):81–106
Rodríguez JJ, Maudes J (2008) Boosting recombined weak classifiers. Pattern Recognition Letters 29(8):1049–1059
Rubin DB (1976) Inference and missing data. Biometrika 63(3):581–592
Safavian SR, Landgrebe D (1991) A survey of decision tree classifier methodology. IEEE transactions on systems, man, and cybernetics 21(3):660–674
Settles B (2009) Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison
Troffaes MC (2007) Decision making under uncertainty using imprecise probabilities. International Journal of Approximate Reasoning 45(1):17–29
Utkin LV (2019) An imprecise extension of svm-based machine learning models. Neurocomputing 331:18–32
Vandoni J, Aldea E, Le Hégarat-Mascle S (2019) Evidential query-by-committee active learning for pedestrian detection in high-density crowds. International Journal of Approximate Reasoning 104:166–184
Xia J, Zhang S, Cai G, Li L, Pan Q, Yan J, Ning G (2017) Adjusted weight voting algorithm for random forests in handling missing values. Pattern Recognition 69:52–60
Zhang ML, Zhou BB, Liu XY (2016) Partial label learning via feature-aware disambiguation. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 1335–1344