Assessing variable importance in clustering: a new method based on unsupervised binary decision trees

Ghattas Badih[1] · **Michel Pierre**[1,2] · **Boyer Laurent**[2]

Abstract
We consider different approaches for assessing variable importance in clustering. We focus on clustering using binary decision trees (CUBT), which is a non-parametric top-down hierarchical clustering method designed for both continuous and nominal data. We suggest a measure of variable importance for this method similar to the one used in Breiman's classification and regression trees. This score is useful to rank the variables in a dataset, to determine which variables are the most important or to detect the irrelevant ones. We analyze both stability and efficiency of this score on different data simulation models in the presence of noise, and compare it to other classical variable importance measures. Our experiments show that variable importance based on CUBT is much more efficient than other approaches in a large variety of situations.

Keywords Unsupervised learning · CUBT · Deviance · Variable importance · Variables ranking

1 Introduction

In most statistical modeling and data analysis tasks, variables' scoring is essential. Its most frequent use is for dimension reduction or feature selection (Liu and Yu 2005), in order to reduce the complexity of the models, to reduce the noise in the data and hence to gain in model accuracy and interpretability. Variables are generally scored with respect to a model or a specific task.

✉ Ghattas Badih
badih.ghattas@univ-amu.fr

Michel Pierre
pierre.michel@univ-amu.fr

Boyer Laurent
laurent.boyer@ap-hm.fr

1 I2M UMR 7373, Aix Marseille Université, CNRS, Centrale Marseille, 13453 Marseille, France

2 SPMC EA3279, Aix Marseille Université, 13385 Marseille, France

In supervised learning, variable importance is often related to its correlation or dependence with a target variable Y. It may be assessed within the model learning process, or once the model is estimated. The most known approaches are classification and regression trees (CART, Breiman et al. 1984), random forests (RF, Breiman 2001) and support vector machines (SVM, Guyon et al. 2002; Rakotomamonjy 2003), where variable importance is strongly related to the model structure and accuracy.

Unsupervised learning concerns data clustering and density estimation. We consider here only clustering methods which aim to construct a partition of a set of n observations in k clusters, where k is specified a priori or determined by the method. Two methods, k-means and hierarchical clustering are among the very common approaches used in practice. In clustering, there is a need to determine which variables are the most important with respect to the obtained clusters.

CUBT (Fraiman et al. 2013; Ghattas et al. 2017) is a non-parametric top-down hierarchical method designed for both continuous and nominal data. A decision tree constructed with CUBT can be used to identify which variables of the dataset are active and take part directly in the growing stage of the tree. However, although some input variables may be irrelevant for the tree construction, they may be competitive with the active variables at different splits of the tree. Identifying these variables may be very useful for many applications.

The goal of this paper is to define variable importance based on CUBT. The role of important variables is analyzed through different data simulation models, comparing CUBT with other classical methods. Stability and efficiency of the score of variable importance are provided. The paper is structured as follows: Sect. 2 presents some classical methods to assess variable importance in clustering. Section 3 gives a brief description of CUBT. Section 4 presents the new methodology to assess variable importance based on CUBT. In Sect. 5, we give a real data example to illustrate our new method. Finally, Sect. 6 presents some experiments and results with different data simulation models and different clustering methods.

2 Variable importance in clustering

In this section we present some scoring approaches for variables in the context of clustering. Two of them are strongly related to the clustering approach, unsupervised random forests (URF) and two-level variable weighting clustering algorithm TWKM. A third scoring approach, the Laplacian score (LS), computes a variable importance score independently from any partitioning model or algorithm. Finally, we propose an intuitive scoring method, leave-one-variable out (LOVO) adapted for all clustering methods.

2.1 Unsupervised random forests

Random Forests (Breiman 2001) and their variable importance measure are one of the most known and used supervised methods both for regression and classification. Breiman (2001) suggested to use RF for clustering by resolving a binary classification

problem: the unlabeled data are assigned to the first class. The second class contains observations generated by independent sampling from the one-dimensional marginal distributions of the first class data. The clustering task is thus transformed into a binary classification problem and variable importance is computed accordingly. The importance of a variable corresponds to the mean decrease of the Gini criterion obtained if each split within the trees of the forest was replaced by the surrogate split over this variable (URF_{gini}). It can also be computed from permuting out-of-bag (OOB) data: for each tree, the prediction error over the OOB data is computed, the same error is computed after randomly permuting each predictor variable. The difference between the two errors are then averaged over all trees, and normalized by the standard deviation of the differences (URF_{perm}). URF may be used for both continuous and nominal data. The number of trees used in the forest was set to 500. The minimal size of a terminal node ($nodesize$) was set to 1 and the number of variables randomly sampled as candidates at each split ($mtry$) was set to $\lfloor\sqrt{p}\rfloor$, with p the number of variables in the data, and $\lfloor\ \rfloor$ stands for the integer part. To apply this method we have used the randomForest package (Liaw and Wiener 2002) from the R software (R Core Team 2013).

2.2 Weighting *k*-means clustering

The two-level variable weighting clustering algorithm TWKM (Chen et al. 2013) is a weighted subspace clustering algorithm for continuous data which can assign weights for each variable in the dataset or weights for groups of variables. This algorithm is an extension of the entropy weighted k-means clustering algorithm, which itself is an extension of the k-means algorithm (MacQueen 1967). First, a set of k initial centers are randomly selected. Observations are assigned to the nearest centers using a dissimilarity measure. Then, new centers are updated, based on these new clusters. Weights are computed for each variable within each cluster, based on the current clustering. The variable weight is inversely proportional to the sum of the within-cluster variances of the variable in the clusters. Weights are taken into account in the computation of the dissimilarity measure. The process continues in this way until convergence. The dissimilarity measure used is the same as the one used with k-means by default, i.e. it is based on the sum of squares between observations. To apply this method, we used the R package wskm (Williams et al. 2015). This approach has been implemented only for continuous data.

2.3 Laplacian score

The Laplacian score (LS, Belkin and Niyogi 2001) is a popular variable ranking index for unsupervised learning. LS assesses variable importance by evaluating the power of locality preserving of a variable. For each variable, this score is computed according to proximities between observations in a k-nearest neighbor graph, with the assumption that two observations are related if they are connected in the graph. A similarity matrix $\mathbb{S}$ between observations is first computed. For each pair of observations i and j,

$\mathbb{S}_{ij} = e^{-\frac{||x_i - x_j||^2}{t}}$, where t is a constant. In this study, we fix $t = 1$. For each observation not in the k-nearest neighborhood, similarities are set to zero. Let $D = diag(\mathbb{S}\mathbf{1})$ (where $\mathbf{1}$ is an identity vector) the diagonal matrix containing for each observation the sum of the similarities to its k-nearest neighbors, i.e. $D = \sum_{i=1}^{n} E_i \mathbb{S} \mathbf{1} e_i$, where E_i is a $n \times n$ matrix with 1 on the cell (i, i) and 0 elsewhere, and e_i is a $1 \times n$ row matrix with 1 on cell $(1, i)$ and 0 elsewhere. The Laplacian matrix is defined to be $L = D - \mathbb{S}$, each variable $X_{.j}$, $j \in \{1, \ldots, p\}$, is "centered" getting:

$$\tilde{X}_{.j} = X_{.j} - \frac{X'_{.j} D \mathbf{1}}{\mathbf{1}' D \mathbf{1}} \mathbf{1}$$

Finally, the LS of variable j is computed as follows:

$$Imp_{LS}(X_{.j}) = \frac{\tilde{X}'_{.j} L \tilde{X}_{.j}}{\tilde{X}'_{.j} D \tilde{X}_{.j}}$$

LS assigns highest scores to variables that best respect the nearest neighbor's graph structure. It is often used in filter methods for feature selection (Zhu et al. 2012). When compared to other classical scores such as data variance or Fisher score, LS showed better results in terms of efficiency (Zhu et al. 2012).

2.4 Leave-one-variable-out score

We propose an intuitive scoring method that may be used with any clustering approach. Once a partition $\mathscr{P}$ is obtained using all the variables we compute a within–cluster heterogeneity criterion denoted $R(\mathscr{P})$. This heterogeneity criterion is defined in Sect. 3.2. Denote $\mathscr{P}^{-j}$ the partition obtained using the same algorithm but omitting variable j from the data, and $R(\mathscr{P}^{-j})$ the within-cluster heterogeneity measure for $\mathscr{P}^{-j}$. The importance of variable j is defined as follows:

$$Imp_{LOVO}(X_{.j}) = R(\mathscr{P}^{-j})$$

We denote this method LOVO for leave one variable out and we use it later for CUBT, k-means and k-modes.

3 Clustering using unsupervised binary trees

CUBT (Fraiman et al. 2013) is a clustering method inspired by CART. It defines a set of clusters using binary decision rules over the original variables. The obtained partition is represented by a binary decision tree interpretable in terms of the original variables, and may be used to assign new observations to a cluster. The algorithm is parallelizable, thus usable for large data sets.

CUBT proceeds using three stages. In the growing stage the sample is split recursively into two subsamples reducing the heterogeneity of the data within the new

subsamples according to a heterogeneity measure. In the second and third stage the maximal tree obtained from the growing stage is pruned using two different criteria, a robust dissimilarity measure and a heterogeneity measure.

In recent works, an extension of CUBT was proposed for both ordinal and nominal data (Ghattas et al. 2017), using Shannon entropy and mutual information. Comparisons with other classical methods using several data simulation models and real data applications showed that CUBT outperforms most of the other methods, especially in terms of prediction, that is using the clustering result to assign a cluster to new observations not used in the clustering process. Some heuristics were proposed for a fine tuning of the parameters used in the method.

We give now a brief description of CUBT.

3.1 Notations

Let $X \in E = \prod_{j=1}^{p} supp(j)$, be a random p-dimensional vector with coordinates $X_{.j}$, $j \in \{1, \ldots, p\}$, and $supp(j)$ the support of variable j, i.e. the set of values it can take. We have a set S of n random observations of X, denoted X_i with $i \in \{1, \ldots, n\}$ and X_{ij} is the ith observation of component j of X. Similar notations are used with small letters to denote the realizations of these variables: x, x_i, $x_{.j}$ and x_{ij}.

3.2 Heterogeneity criteria

For any node t (a subset of S), let n_t be the number of observations in t, let $X^{(t)}$ be the restriction of X to node t, i.e. $X^{(t)} = \{X | X \in t\}$. We define $R(t)$, the heterogeneity measure of t for continuous data as follows:

$$\begin{aligned} R(t) &= \frac{n_t}{n} trace(cov(X^{(t)})) \\ &= \frac{\sum_{X_i \in t} || X_i - \bar{X}_t ||^2}{n} \end{aligned}$$

where $cov(X^{(t)})$ is the covariance matrix of $X^{(t)}$ and:

$$\bar{X}_t = \frac{\sum_{X_i \in t} X_i}{n_t}$$

For nominal data, the heterogeneity measure of t is defined as follows:

$$\begin{aligned} R(t) &= trace(\mathbf{MI}(X^{(t)})) \\ &= -\sum_{j=1}^{p} \sum_{l \in supp(j)} p_{lj}^{(t)} \log_2 p_{lj}^{(t)} \end{aligned}$$

where $\mathbf{MI}(X^{(t)})$ is the mutual information matrix of $X^{(t)}$ and $p_{lj}^{(t)}$ is the probability for the component j of X to take value l within node t. For continuous data, te

heterogeneity criterion $R(t)$ is the sum of the variables' variances while for nominal data, it is the sum of their entropies.

3.3 Growing stage

Initially, the root node of the tree contains all the observations of S. The sample is split recursively into two disjoint subsamples using binary splits of the form $x_{.j} \in \mathscr{A}_j$, where $j \in \{1, \ldots, p\}$ and $\mathscr{A}_j$ is a subset of $supp(j)$. For continuous variables, $\mathscr{A}_j$ will be a real interval of the form $\mathscr{A}_j =]\inf(supp(j)); a_j]$, with $a_j \in supp(j)$.

Thus, a split $s_j(t)$ of a node t into two subnodes t_l and t_r is defined by a pair $(j, \mathscr{A}_j)$ as follows:

$$t_l = \{x \in E : x_{.j} \in \mathscr{A}_j\} \text{ and } t_r = \{x \in E : x_{.j} \notin \mathscr{A}_j\}$$

The best split of t into two sibling nodes t_l and t_r is defined by:

$$argmax_{(j,\mathscr{A}_j)}\{\Delta(t, s_j(t))\}$$

where

$$\Delta(t, s_j(t)) = R(t) - R(t_l) - R(t_r)$$

The new subnodes are recursively split, and the growing process may stop when at least one of the two following criteria is satisfied:

$$n_t < minsize \text{ or } \Delta(t, s_j(t)) < mindev \times R(S)$$

where $minsize$ and $mindev$ are fixed thresholds, and $R(S)$ is the deviance of the entire sample. The $minsize$ parameter is the minimum number of observations within a terminal node while $mindev$ is the minimum proportion of deviance attained for a split to be considered. Once the algorithm stops, a class label is assigned to each leaf and the obtained tree is called the maximal tree. A partition of the initial dataset is obtained, and each leaf of the tree corresponds to a cluster.

Details of both pruning stages of CUBT are described in previous works (Fraiman et al. 2013; Ghattas et al. 2017), as they are not needed to assess variable importance.

The main difference between CUBT and CART are the follows:

- The splitting criterion is based on the output variable Y in CART whereas in CUBT it is based on all input variables, no output variable being available.
- Pruning in CUBT is done in two stages using a dissimilarity measure when CART uses a one-step cost complexity measure.

4 Assessing variable importance in CUBT

To define variable importance in CUBT we follow similar ideas used in CART. We first define the surrogate splits for each variable within each node of a tree. Then we use the surrogate splits to compute the variable importance score.

4.1 Surrogate splits

Let s be the best split of a node t in the tree T, based on variable j_0 which splits t into t_L and t_R. Let $s_j = s_j(t)$ be any split of t using any variable $j \neq j_0$ splitting t into t'_L and t'_R.

The probability for an observation to be sent to the left node for both splits s and s_j is defined as follows:

$$p(t_L \cap t'_L) = \frac{Card\{t_L \cap t'_L\}}{n_t}$$

where $Card$ is the cardinal of a set.

Then, the probability that both splits send an observation to the left node is:

$$p_{LL}(s, s_j) = \frac{p(t_L \cap t'_L)}{p(t)}$$

where $p(t) = \frac{n_t}{n}$. $p_{RR}(s, s_j)$ is defined equivalently.

The probability that s_j predicts s is:

$$p(s, s_j) = p_{LL}(s, s_j) + p_{RR}(s, s_j)$$

The *surrogate split* for s over variable j in node t, denoted $\tilde{s_j}(t)$ is defined by:

$$p(s, \tilde{s_j}(t)) = \max_{s_j \in \mathscr{S}_j} p(s, s_j)$$

where $\mathscr{S}_j$ is the set of all splits over variable j.

Surrogate splits are used to compute variable importance. They may also be used when predicting new observations with missing data.

4.2 Variable importance

We define the importance of variable j as follows:

$$Imp(X_{.j}) = \sum_{t \in T} \Delta(t, \tilde{s_j}(t))$$

The score of importance is the total loss of deviance induced if each split in the tree T is replaced by the surrogate split over $X_{.j}$.

Variable importance may be assessed using the dataset at hand, but using bootstrap may give more stable results for the scores (Ghattas 1999). It is defined exactly like in CART except that the criterion used in is not the same.

5 A real data example

As an example we use the *Iris* data set (Fisher 1936) from a supervised classification context. The output variable has three levels corresponding to different types of iris and the other are continuous variables corresponding to measurements of the plant (length and width of petals and sepals). The data set contains 150 observations, 50 for each iris type. The goal is to predict the type of the plant (the species) from the dimensions of the plant.

The output of CUBT applied to the *Iris* dataset is given in Fig. 1 (the class variable is not used). The leaves of the tree correspond to the discovered clusters labeled arbitrarily C1, C2 and C3.

Table 1 gives the score of variable importance for each input variable, computed with CUBT, unsupervised random forests (URF_{gini} and (URF_{perm})), leave-one-variable-out using CUBT and k-means (LOVO-cubt and LOVO-km), two-level variable weighting clustering algorithm TW-k-means (TWKM) and Laplacian score (LS). Regarding CUBT, the only active variable ($Petal.Length$) has the highest score. This is also

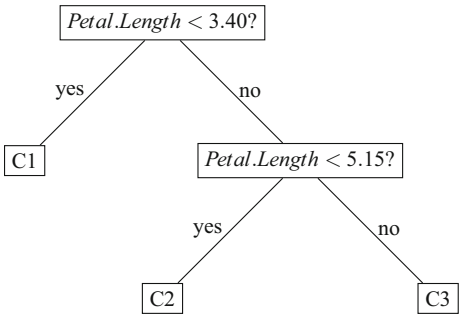

Fig. 1 Three clusters (C1, C2, C3) obtained by CUBT with the *Iris* dataset. The only active variable is $Petal.Length$

Table 1 The four input variables of the *Iris* dataset with their corresponding importance score using different methods

Method	$Sepal.Length$	$Sepal.Width$	$Petal.Length$	$Petal.Width$
CUBT	0.82	0.04	**1.00**	0.96
URF_{gini}	0.99	0.89	**1.00**	0.72
URF_{perm}	0.77	0.44	**1.00**	0.88
LOVO-cubt	0.83	0.98	0.71	**1.00**
LOVO-km	0.96	0.93	**1.00**	0.93
TWKM	2.53×10^{-6}	1.31×10^{-5}	2.53×10^{-6}	**1.00**
LS	0.45	**1.00**	0.08	0.20

Highest scores for each approach are indicated in bold

the case for URF_{gini}, URF_{perm} and LOVO-km. The other variables ($Sepal.Length$, $Sepal.Width$, $Petal.Width$) are not active in the tree, however two of them have high scores.

6 Experimental analysis

In this section, we analyze both the stability and the efficiency of variable importance scores based on CUBT. Decision trees are known to be unstable when the data is subject to small variations (Breiman 1996). Besides, CUBT like CART is very sensitive to the choice of the $minsize$ parameter. First, the stability of the score computed with CUBT is analyzed using two datasets where the clusters are known, *Toys* ($k = 2$) and *Iris* ($k = 3$). Then, the efficiency of important variables detection is analyzed using nine data simulation models designed for both continuous and nominal data, and CUBT is compared to other classical variable importance scoring methods (URF_{gini}, URF_{perm}, LOVO-cubt, LOVO-km, TWKM and LS).

6.1 Stability of variable importance in CUBT

We focus here on the stability of variable importance obtained with CUBT, with respect to data modifications and the choice of $minsize$.
For this analysis, we use two known datasets from supervised classification, the first is the *Toys* simulated dataset (Weston et al. 2003) and the second is the real *Iris* dataset.

The *Toys* dataset has two levels ($Y \in \{-1, 1\}$) and n observations may be drawn for $p \geq 6$ variables as follows:

- For $j \in \{1, 2, 3\}$, $P(X_{.j} \sim N(yj, 1)) = 0.7$ and $P(X_{.j} \sim N(0, 1)) = 0.3$,
- For $j \in \{4, 5, 6\}$, $P(X_{.j} \sim N(0, 1)) = 0.7$ and $P(X_{.j} \sim N(y(j-3), 1)) = 0.3$,
- For $j > 6$, $P(X_{.j} \sim N(0, 1)) = 1$.

The first six variables are the true important variables, the others are irrelevant for the model.
For each dataset, we draw 100 simulated datasets using either the simulation model (for *Toys*) or bootstrap samples (for *Iris*). For each sample a CUBT tree is constructed, and variable importance is computed. This is repeated for different integer values of $minsize$ ranging from 2 to $\lfloor \frac{n}{4} \rfloor$. Thus for each value of $minsize$ we get 100 estimates for the importance of each variable. An optimal value of $minsize$ may be obtained using the heuristic proposed in Ghattas et al. (2017) minimizing the within deviance score of the CUBT trees using ten fold cross validation. For the *Toys* dataset we fix $n = 100$ and $p = 12$, i.e. there are as many irrelevant variables as the true important ones.

Figures 2 and 3 show the boxplots of variables' importance for different values of $minsize$ for the *Toys* and the *Iris* datasets respectively. Figures 4 and 5 show these boxplots for the optimal values of $minsize$, 8 for *Toys* and 16 for *Iris* respectively.

The score of variable importance is sensitive to the value of the $minsize$ parameter. It decreases when increasing the $minsize$ value. As the score is additive with the

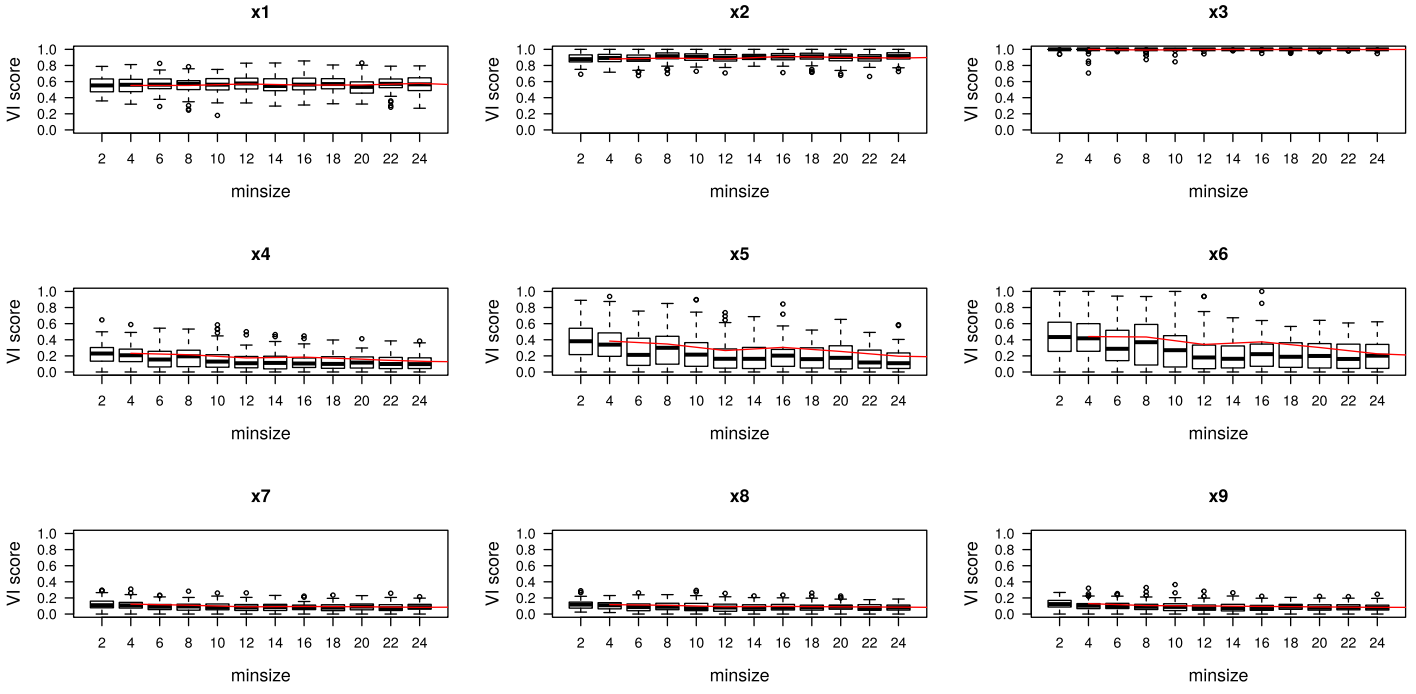

Fig. 2 Stability of CUBT Variables' importance for the *Toys* dataset, obtained using 100 simulated datasets, for different values of $minsize$. The six first variables are true important variables, the others are noise variables. VI scores were normalized. Red lines indicate the means of the variables. This figure shows variables' importance for the nine first variables, for different values of $minsize$

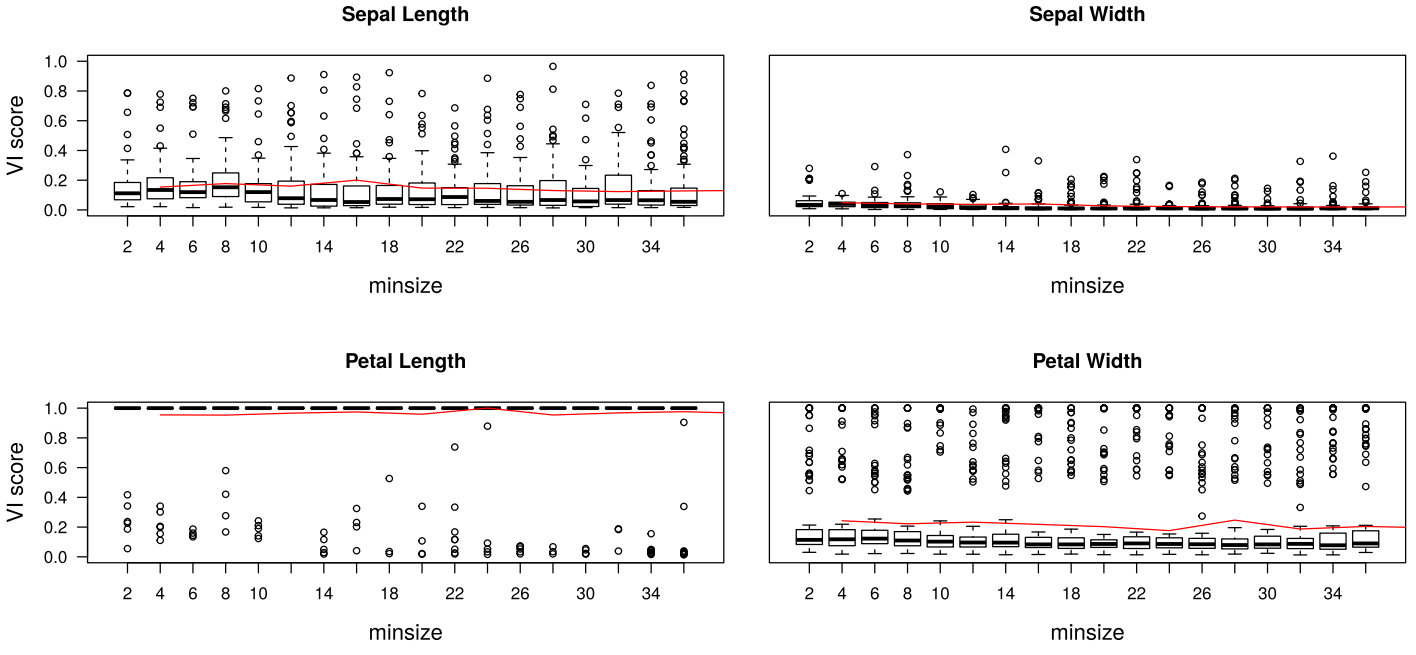

Fig. 3 Stability of CUBT Variables' importance for the *Iris* dataset, obtained using 100 bootstrap samples, for different values of $minsize$. VI scores were normalized. Red lines indicate the means of the variables. This figure shows variables'importance for different values of $minsize$

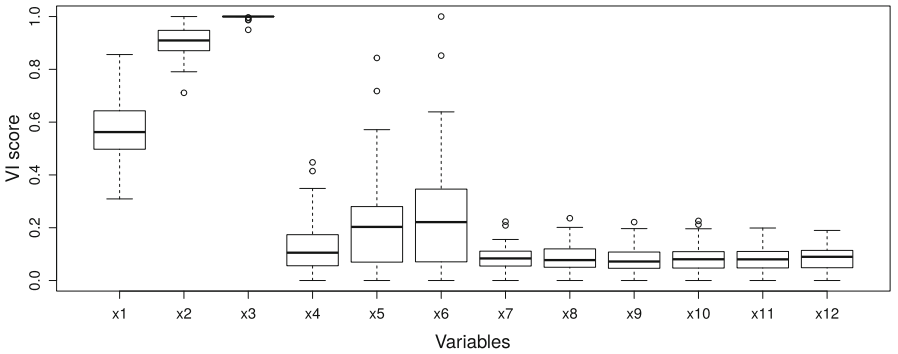

Fig. 4 Stability of CUBT Variables' importance for the *Toys* dataset, obtained using 100 simulated datasets, for different values of $minsize$. The six first variables are true important variables, the others are noise variables. VI scores were normalized. This figure shows variables' importance for $minsize = 8$, which is the optimal value obtained by cross validation

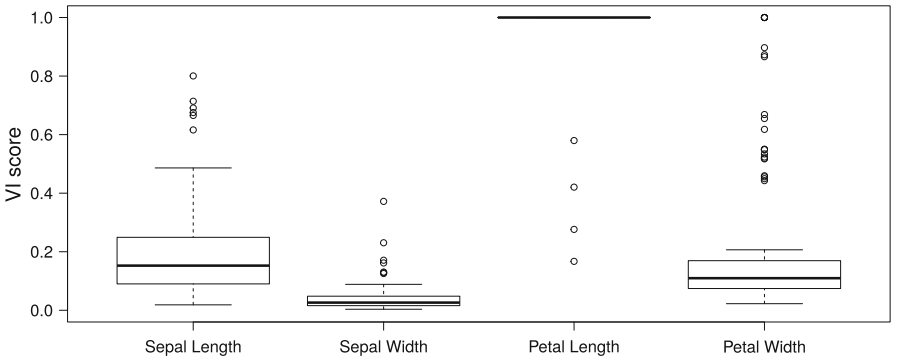

Fig. 5 Stability of CUBT Variables' importance for the *Iris* dataset, obtained using 100 bootstrap samples, for different values of $minsize$. VI scores were normalized. This figure shows variables' importance for $minsize = 16$, which is the optimal value obtained by cross validation

number of nodes in the tree, high values of $minsize$ give trees with less nodes thus lower variable importance scores.

For the *Toys* dataset, the boxplots show that the three first variables are always detected as the most important variables, the next three variables are less important, as expected due to generating the data model. The six remaining variables (the noisy variables) show a very stable variable importance score, lower than all others. Variables having lower scores also have a lower dispersion.

For the *Iris* dataset, the score of variable importance is more stable for the less important variable ($Sepal.Width$), while the boxplots for "intermediate" variables ($Sepal.Length$ and $Petal.Width$) are very dispersed. The variable $Petal.Length$ is appearing as the most important in this dataset, confirming the results given in Table 1.

6.2 Efficiency of variable importance

The efficiency of variable importance scoring corresponds to its ability to detect the most important variables in a dataset, providing the highest scores for the most important variables and the lowest scores to irrelevant or redundant ones. To test the efficiency of variable importance scoring methods we use nine data simulation models where the true important variables defining the clusters are known. We choose the same simulation models previously defined in Fraiman et al. (2013) and Ghattas et al. (2017) for continuous and nominal data, considering different numbers of variables and clusters. We also test the efficiency of this method using again the *Toys* dataset for $p = 18$ and in high dimension for $p = 1000$.

6.2.1 Simulation models

We consider nine data simulation models. The first four models are designed for continuous data while the five remaining are designed for nominal data.

M1: 2D-model In this model, we fix $k = 4$ and $X \in \mathbb{R}^2$ following a multivariate normal distribution $N(\mu_l, \Sigma)$, $l \in \{1, \ldots, k\}$, where $\mu_1 = (-1, 0)$, $\mu_2 = (1, 0)$, $\mu_3 = (0, -1)$ and $\mu_4 = (0, 1)$, and the covariance matrix $\Sigma = diag(\sigma^2 \mathbf{1})$, with $\sigma = 0.1$.

M2: 5D-model This model generates $k = 10$ clusters of observations in $\mathbb{R}^5$, having different multivariate normal distributions $N(\mu_l, \Sigma)$, $l \in [1, k]$, with $\mu_1 = (1, 0, 0, 0, 0)$, $\mu_2 = (0, 1, 0, 0, 0)$, $\mu_3 = (0, 0, 1, 0, 0)$, $\mu_4 = (0, 0, 0, 1, 0)$, $\mu_5 = (0, 0, 0, 0, 1)$ and $\mu_i = -\mu_{i-5}$ for $i \in \{6, \ldots, 10\}$. The covariance matrix is $\Sigma = diag(\sigma^2 \mathbf{1})$, with $\sigma = 0.1$.

M3: Concentric cluster Model In this model, we consider two concentric clusters in $\mathbb{R}^2$; each cluster has observations distributed uniformly between two concentric circles centered in $(0, 0)$. The first cluster is delimited by circles having radius between 50 and 80, the second by circles with radius from 200 to 230.

M4: High-dimensional Model In this model, three clusters are normally distributed in $\mathbb{R}^{50}$, with $\mu_1 = (-1, \ldots, -1)$, $\mu_2 = (0, \ldots, 0)$ and $\mu_3 = (1, \ldots, 1)$, and the covariance matrix is $\Sigma = diag(\sigma^2 \mathbf{1})$, with $\sigma = 0.01$.

M5: Labels distribution-based Model In this model, each variable $X_{.j}$, $j \in \{1, \ldots, p = 9\}$ has $m = 5$ levels. We define $k = 3$ clusters, each characterized by a high frequency of one level. For observations from cluster 1, $P(X_{.j} = 1) = q$, and a uniform probability is used for the other levels i.e. $P(X_{.j} = l) = \frac{1-q}{m-1}$ for $l \neq 1$. For clusters 2 and 3, the frequent levels are 3 and 5, respectively, using the same probabilities. We fix $q = 0.8$.

M6: Tree Model 1 We use here a tree structure model. We fix the dimension $p = 3$ and the number of groups $k = 4$. Each variable $X_{.j}$, $j \in \{1, \ldots, p\}$, has $m = 4$ levels, $X_{.j} \in \{1, 2, 3, 4\}$. Clusters are defined as follows:

- $C1$: x_1 and x_2 have odd levels, and x_3 is arbitrary
- $C2$: x_1 has odd levels, x_2 has even levels, and x_3 is arbitrary
- $C3$: x_1 has even levels, x_3 has odd levels, and x_2 is arbitrary
- $C4$: x_1 and x_3 have even levels, and x_2 is arbitrary

M7: Tree Model 2 We use the same tree structure model. As in the previous case, we fix $p = 3$ and the number of groups $k = 4$. Here, each variable $X_{.j}$, $j \in \{1, \ldots, p\}$, has $m = 4$ levels. The only difference is that variable levels are not uniformly distributed in each cluster. Here, we consider a parameter p_0 that controls the non-uniformity of the distribution of levels. In our experiments, we fix $p_0 = 0.8$. Clusters are defined as follows:

- $C1$: x_1 and x_2 have odd levels with $P(x_1 = 1) = P(x_2 = 1) = p_0$, and x_3 is arbitrary
- $C2$: x_1 has odd levels, x_2 has even levels with $P(x_1 = 1) = P(x_2 = 2) = p_0$, and x_3 is arbitrary
- $C3$: x_1 has even levels, x_3 has odd levels with $P(x_1 = 2) = P(x_3 = 1) = p_0$, and x_2 is arbitrary
- $C4$: x_1 and x_3 have even levels with $P(x_1 = 2) = P(x_3 = 2) = p_0$, and x_2 is arbitrary

M8: Nominal IRT-based model We use here an item response theory (IRT) model designed for nominal data. We fix the dimension $p = 9$ and the number of groups $k = 3$. Each variable has $m_j = 5$ levels. We now suppose that variables are representing multiple-choice items. The nominal response model (NRM, Bock 1972) can address nominal data. It is a specialization of the general model for multinomial response relations and is defined as follows:

Let θ be a level of latent ability underlying the response to the items. The probability that a subject of ability θ responds category l for item j is given by

$$\Psi_{jl_j}(\theta) = \frac{\exp[z_{jl_j}(\theta)]}{\sum_{h=1}^{m_j} \exp(z_{jh}(\theta))}$$

where $z_{jh}(\theta) = c_{jh} + a_{jh}\theta$ with $h = 1, 2, \ldots, l_j, \ldots, m_j$, θ is a latent trait, and c_{jh} and a_{jh} are item parameters associated with the h-th category of item j. We generate random datasets using the NRM by simulating latent trait values for the four groups. For $c \in \{1, 2, 3\}$, we simulate a vector of latent trait values for each group c using $N(\mu_c, \sigma^2)$, $\mu = (-3, -1, 1, 3)$ and $\sigma^2 = 0.2$. For $j \in \{1, \ldots, p\}$, the values of c_{jh} range uniformly between -2 and 2 while c_{jh} are distributed as $N(1, 0.1)$. Simulations are performed using the *NRM.sim* function of the *mcIRT* package (Reif 2014) with **R**.

M9: IRT-based Model We use IRT models again. These models allow us to assess the probability of observing a level for each variable given a latent trait level. The latent trait is an unobservable continuous variable that defines the individual's ability, measured by the observed variables. In the IRT framework, the variables called items are ordinal. The observations can be either binary or polytomous. Here, we introduce a polytomous IRT model to generate data in a probabilistic way. The generalized partial credit model (GPCM, Muraki 1992) is an IRT model that can address ordinal data. It is an extension of the 2-parameter logistic model for dichotomous data. The model is defined as follows:

$$p_{jx}(\theta) = P(X_{ij} = x|\theta) = \frac{\exp \sum_{l=0}^{x} \alpha_j(\theta_i - \beta_{jl})}{\sum_{r=0}^{m_j} \exp \sum_{l=0}^{r} \alpha_j(\theta_i - \beta_{jl})}$$

where θ is the latent trait and θ_i represents the latent trait level of individual i. β_{jl} is a difficulty threshold parameter for the level l of the item j. For $j \in \{1, \ldots, p\}$, β_j is a vector of dimension $m-1$. α_j is a discrimination parameter represented by a scalar. We generate random datasets using the GPCM by simulating latent trait values for the three groups. For $c \in \{1, 2, 3\}$, we simulate a vector of latent trait values for each class c using $N(\mu_c, \sigma^2)$, $\mu = (-3, 0, 3)$ and $\sigma^2 = 0.2$. For $j \in \{1, \ldots, p\}$, α_j is distributed as $N(1, 0.1)$, and β_j is a vector of ordered values that range uniformly between -2 and 2. Simulations are performed using the *rmvordlogis* function of the *ltm* package (Rizopoulos 2006) with **R**.

6.2.2 Adding noise to the simulated datasets

Each of the simulation models suggested in the previous subsection is based on p variables. These variables may be considered as the true important variables with respect to the clusters identification. For each case, we add p' irrelevant variables to the model, also called noise variables. We test two configurations: $p' = p$ and $p' = 2p$.

For continuous datasets (models M1 to M4), we simulate $\left\lfloor \frac{p'}{2} \right\rfloor$ variables following the normal distribution $N(0, \frac{\sigma_0}{2})$, where $\sigma_0 = \min_{j \in \{1,\ldots,p\}} \sigma_j$ is the minimum standard deviation observed in the initial set of variables. The other $p' - \left\lfloor \frac{p'}{2} \right\rfloor$ additional variables follow the uniform distribution $U\left(-\sigma_0\sqrt{\frac{3}{4}}, \sigma_0\sqrt{\frac{3}{4}}\right)$. This method allows us to consider two types of distributions for the noise variables, that have the same variance.

For nominal simulation models M5 to M9, for each of the p relevant variables in the model, we generate p' corresponding irrelevant variables as follows: if the original variable has m levels, its corresponding noise variable has the distribution $(p_1, \ldots, p_m)$ over the same support where $p_l = 0.8$ and $p_j = \frac{0.2}{m-1}$ for $j \neq l$ and level l is chosen arbitrarily. This approach ensures that the noise variables generated will have a low entropy.

6.2.3 Controlling the separability of the clusters

For each simulation model, we test two configurations of high and low cluster separability. For the continuous datasets (models M1, M2 and M4, all based on multidimensional gaussian distributions), the separability is reduced by fixing higher values for the standard deviations of the distributions within the clusters. The standard deviations are fixed to 0.8 for models M1 and M2, and to 0.1 for model M4. For the concentric model M3, the radius of the circles is changed so that the circles are closer. The first cluster is unchanged while the second is delimited by circles having radius between 100 and 130. For the nominal datasets (models M5 to M9), the separability is controlled in the same way as described in recent work (Ghattas et al. 2017).

Table 2 True positive rate (TPR) for each simulation model and each method, with *high separated clusters*

Model	n	CUBT	URF_{gini}	URF_{perm}	LOVO-cubt	LOVO-km	LS	TWKM
M1	100	**100(2)**	30.5(4)	65(2)	**100(2)**	**100(2)**	0(4)	23(4)
$p = 2$	300	**100(2)**	18.5(4)	65(2)	**100(2)**	99.5(2)	0(4)	56.5(4)
$p' = 2$	500	99(2)	19(4)	70(2)	**100(2)**	**100(2)**	0(4)	57(4)
M1	100	**100(2)**	23(6)	50(5)	**100(2)**	**100(2)**	0(4)	17.5(6)
$p = 2$	300	**100(2)**	16.5(6)	30(5)	**100(2)**	**100(2)**	0(4)	54.5(6)
$p' = 4$	500	99.5(2)	13.5(6)	45(2)	95(2)	**100(2)**	0(4)	58.5(4)
M2	100	**100(5)**	56.6(9)	46(9)	93(5)	91.4(5)	20(9)	29.8(9)
$p = 5$	300	96(5)	58.4(8)	54(8)	86.4(5)	**96.2(5)**	20(9)	75(9)
$p' = 5$	500	**100(5)**	54(9)	56(9)	85.2(5)	96.8(5)	20(9)	90.6(9)
M2	100	**100(5)**	36.8(10)	28(15)	86.4(5)	93.6(14)	0(14)	21.8(5)
$p = 5$	300	**96.2(5)**	39.2(6)	40(10)	82.6(5)	94.2(5)	0(13)	24.2(6)
$p' = 10$	500	**100(5)**	34.4(15)	38(13)	79.4(5)	96.2(5)	0(10)	88.8(14)
M3	100	**97.5(2)**	30(4)	55(4)	42(4)	80.5(2)	50(4)	61(4)
$p = 2$	300	**99.5(2)**	8(4)	55(2)	53(4)	78.5(2)	50(4)	55.5(4)
$p' = 2$	500	**100(2)**	5.5(4)	45(4)	81(2)	77.5(2)	50(4)	49(4)
M3	100	**97.5(2)**	19(4)	15(5)	40.5(3)	77(2)	50(6)	92(2)
$p = 2$	300	**100(2)**	18(5)	35(4)	52.5(3)	71(2)	50(4)	59(4)
$p' = 4$	500	**100(2)**	8(6)	50(4)	89(2)	71.5(2)	50(4)	41(4)
M4	100	**100(50)**	2.3(100)	62.4(88)	99.9(97)	73.4(100)	0(100)	78.5(50)
$p = 50$	300	**100(50)**	0.3(100)	68.8(92)	**100(50)**	72.4(100)	0(100)	79.3(50)
$p' = 50$	500	**100(50)**	0(100)	68.6(95)	**100(50)**	71.7(97)	0(100)	75.7(50)
M4	100	**100(50)**	0(150)	46.2(138)	99.8(136)	46(150)	0(147)	74.1(50)
$p = 50$	300	**100(50)**	0(146)	52.6(144)	**100(50)**	43.8(149)	0(150)	76(50)
$p' = 100$	500	**100(50)**	0(150)	56.6(145)	**100(50)**	45.3(147)	0(150)	80(50)
M5	100	**100(9)**	64.4(9)	33.3(18)	57.8(14)	86.7(18)	37.8(18)	–
$p = 9$	300	**100(9)**	78.9(9)	44.4(18)	55.6(16)	86.7(18)	28(18)	–
$p' = 9$	500	**100(9)**	84.4(9)	57.8(18)	57.8(18)	81.7(15)	18.1(18)	–
M5	100	**100(9)**	58.9(9)	15.6(27)	43.3(25)	51.1(23)	27.6(26)	–
$p = 9$	300	**100(9)**	71.7(9)	15.6(23)	41.7(22)	63.9(25)	22.4(27)	–
$p' = 18$	500	**100(9)**	80(9)	24.4(27)	38.9(25)	65(27)	19.1(26)	–
M6	100	96.7(3)	**100(3)**	26.7(6)	56.7(6)	53.3(6)	78(13)	–
$p = 3$	300	**100(3)**	**100(3)**	40(6)	28.3(6)	31.7(6)	12(6)	–
$p' = 3$	500	**100(3)**	**100(3)**	26.7(6)	21.7(6)	35(6)	0(6)	–
M6	100	95(3)	**100(3)**	10(8)	28.3(7)	38.3(7)	70.3(3)	–
$p = 3$	300	**100(3)**	**100(3)**	6.7(9)	21.7(7)	26.7(9)	68(3)	–
$p' = 6$	500	**100(3)**	**100(3)**	16.7(9)	25(9)	20(8)	26(9)	–
M7	100	96.7(3)	**98.3(3)**	33.3(6)	43.3(6)	58.3(4)	45(6)	–
$p = 3$	300	**100(3)**	**100(3)**	33.3(6)	55(6)	53.3(5)	1(6)	–
$p' = 3$	500	**100(3)**	**100(3)**	50(5)	41.7(6)	50(6)	0(6)	–
M7	100	98.3(3)	**100(3)**	3.3(8)	33.3(9)	45(3)	49(3)	–

Table 2 continued

Model	n	CUBT	URF_{gini}	URF_{perm}	LOVO-cubt	LOVO-km	LS	TWKM
$p = 3$	300	**100(3)**	**100(3)**	6.7(9)	45(4)	46.7(9)	20(9)	–
$p' = 6$	500	**100(3)**	**100(3)**	10(9)	35(6)	45(5)	1(9)	–
M8	100	73.9(9)	**83.3(9)**	45.6(16)	51.7(18)	55(17)	62.7(15)	–
$p = 9$	300	76.1(9)	**84.4(9)**	43.3(18)	53.3(16)	53.3(17)	60(14)	–
$p' = 9$	500	76.1(9)	**86.7(9)**	44.4(18)	50.6(15)	54.4(16)	66.9(9)	–
M8	100	68.3(9)	**77.2(9)**	25.6(23)	40(27)	36.7(27)	45.9(26)	–
$p = 9$	300	71.1(9)	**85.6(9)**	30(27)	37.8(26)	31.7(23)	46.2(17)	–
$p' = 18$	500	73.3(9)	**85(9)**	30(21)	31.7(25)	34.4(27)	48.4(22)	–
M9	100	**100(9)**	55.6(11)	64.4(16)	66.7(14)	61.1(15)	21.6(18)	–
$p = 9$	300	**100(9)**	59.4(16)	87.8(9)	52.2(18)	62.8(10)	3.8(18)	–
$p' = 9$	500	**100(9)**	60(16)	94.4(9)	55(16)	58.3(14)	0.3(18)	–
M9	100	**100(9)**	43.9(13)	33.3(27)	56.7(23)	40.6(23)	16(27)	–
$p = 9$	300	**100(9)**	47.8(26)	48.9(9)	50.6(23)	47.2(24)	9.2(25)	–
$p' = 18$	500	**100(9)**	56.1(22)	62.2(9)	35(11)	53.3(27)	2.6(27)	–

Values in parentheses represent the highest rank (HR) among the ranks of the p true important variables. Bold values correspond to the best performances. p is the number of true important variables and p' the number of irrelevant variables added to the data. TWKM cannot be computed on nominal data

6.2.4 Results

For each simulation model, we vary the sample size using $n = 100$, 300 and 500, and we run 100 simulated datasets computing importance scores from the following methods: CUBT, URF_{gini}, URF_{perm}, LOVO-cubt, LOVO-km, TWKM and LS. For both continuous and nominal datasets, clusters are equally sized. For CUBT, the score of variable importance is computed from the optimal clustering tree obtained after both the pruning stages are applied. To assess the efficiency of each scoring method, we compute the proportion of true important variables appearing among the top p highest score variables. This index is similar to a true positive rate (TPR). Highest values correspond to a highest ability to detect important variables. When it is far from 100% it may be interesting to see how are the "undetected" true important variables scored. We also report the highest rank (HR) among the ranks of the p true important variables. The first index (TPR) is averaged over the 100 simulated datasets while the HR is the rank computed from the mean variable importance score of each variable.

Table 2 provides the results for all the models and all the methods, for two levels of noise, with high separated clusters. It gives the TPR together with the HR (between parentheses). For continuous data (models M1 to M4), LOVO-km uses k-means, and for the other models it uses k-modes. Table 3 provides the same results as Table 2, for low separated clusters. Results obtained using the *Toys* dataset are given in Table 4.

In the case of high separated clusters (Table 2), in terms of TPR, CUBT outperforms all the methods for all sample sizes except for model M8, retrieving correct important variables 90% of the times. For continuous datasets (models M1 to M4), URF_{gini} shows poor results, which are worse when increasing the sample size. URF_{perm} has

Table 3 True positive rate (TPR) for each simulation model and each method, with low separated clusters

Model	n	CUBT	URF_{gini}	URF_{perm}	LOVO-cubt	LOVO-km	LS	TWKM
M1	100	**100(2)**	45(4)	55(4)	90(2)	**100(2)**	0(4)	50(4)
$p = 2$	300	**100(2)**	40(3)	65(2)	70(2)	**100(2)**	0(4)	50(4)
$p' = 2$	500	**100(2)**	40(3)	65(4)	85(2)	**100(2)**	0(4)	50(4)
M1	100	**100(2)**	45(2)	35(5)	70(2)	**100(2)**	0(4)	50(6)
$p = 2$	300	**100(2)**	30(4)	35(5)	65(2)	**100(2)**	0(4)	50(4)
$p' = 4$	500	**100(2)**	15(6)	40(3)	80(2)	**100(2)**	0(4)	50(5)
M2	100	98(5)	58(9)	62(9)	80(5)	**100(5)**	20(9)	66(9)
$p = 5$	300	**100(5)**	58(8)	50(9)	66(6)	**100(5)**	20(9)	80(9)
$p' = 5$	500	**100(5)**	56(8)	58(8)	72(5)	**100(5)**	20(9)	80(9)
M2	100	96(5)	50(11)	34(15)	74(5)	**100(5)**	0(14)	28(14)
$p = 5$	300	**100(5)**	40(12)	32(13)	60(5)	**100(5)**	0(15)	80(13)
$p' = 10$	500	**100(5)**	38(14)	36(15)	40(12)	**100(5)**	0(15)	80(15)
M3	100	**100(2)**	20(4)	50(3)	55(4)	70(2)	45(4)	50(4)
$p = 2$	300	**100(2)**	10(4)	40(4)	55(3)	80(2)	45(4)	50(4)
$p' = 2$	500	**100(2)**	0(4)	45(3)	95(2)	85(2)	40(4)	50(4)
M3	100	**90(2)**	30(5)	40(6)	35(6)	55(2)	**90(6)**	50(6)
$p = 2$	300	**100(2)**	15(6)	20(6)	70(5)	65(2)	50(5)	50(4)
$p' = 4$	500	**100(2)**	5(5)	35(6)	80(2)	80(2)	50(3)	50(6)
M4	100	**100(50)**	34.6(100)	54(98)	75.2(100)	91.8(99)	0(100)	0(100)
$p = 50$	300	**100(50)**	27.8(100)	54.6 (100)	58.2(100)	98.8(87)	0(100)	0(100)
$p' = 50$	500	**100(50)**	23.8(100)	56(95)	68.6(99)	99.4(97)	0(100)	0(100)
M4	100	**99.8(50)**	19(150)	34.4 (146)	71.4(145)	81.6(149)	0(150)	0(150)
$p = 50$	300	**100(50)**	11.8(149)	36.8(150)	51.6(144)	95.6(136)	0(150)	0(145)
$p' = 100$	500	**100(50)**	5.8(149)	40(147)	53(150)	98(146)	0(149)	0(150)
M5	100	98.9(9)	**100(9)**	26.7(18)	50(18)	60(17)	80(15)	–
$p = 9$	300	98.9(9)	**100(9)**	26.7(18)	53.3(18)	60(17)	76.7(14)	–
$p' = 9$	500	100(9)	**100(9)**	15.6(18)	53.3(16)	71.1(18)	72.2(17)	–
M5	100	96.7(9)	**100(9)**	12.2(27)	37.8(22)	37.8(25)	50(24)	–
$p = 9$	300	**100(9)**	**100(9)**	5.6(27)	35.6(27)	43.3(21)	50(27)	–
$p' = 18$	500	**100(9)**	**100(9)**	3.3(26)	37.8(26)	48.9(26)	57.8(26)	–
M6	100	30(8)	**36.7(6)**	0(18)	20(16)	33.3(8)	23.3(14)	–
$p = 3$	300	23.3(7)	**40(7)**	0(15)	23.3(15)	26.7(16)	13.3(9)	–
$p' = 3$	500	**40(7)**	**40(5)**	3.3(18)	26.7(18)	33.3(9)	30(14)	–
M6	100	**46.7(9)**	43.3(5)	3.3(26)	13.3(15)	16.7(7)	13.3(8)	–
$p = 3$	300	**46.7(9)**	16.7(7)	0(20)	6.7(23)	26.7(22)	6.7(27)	–
$p' = 6$	500	**46.7(8)**	26.7(6)	0(25)	6.7(26)	23.3(16)	10(8)	–
M7	100	**23.3(9)**	10(9)	6.7(17)	16.7(13)	**23.3(8)**	16.7(13)	–
$p = 3$	300	**43.3(9)**	10(8)	6.7(17)	23.3(16)	26.7(18)	20(17)	–
$p' = 3$	500	23.3(7)	0(8)	0(18)	16.7(15)	**43.3(3)**	20(5)	–
M7	100	**26.7(8)**	10(7)	0(24)	20(27)	23.3(14)	10(26)	–

Table 3 continued

Model	n	CUBT	URF_{gini}	URF_{perm}	LOVO-cubt	LOVO-km	LS	TWKM
$p = 3$	300	**43.3(7)**	6.7(9)	3.3(27)	16.7(27)	16.7(25)	13.3(9)	–
$p' = 6$	500	**30(8)**	6.7(9)	3.3(19)	16.7(22)	10(22)	13.3(11)	–
M8	100	76.7(9)	**85.6(9)**	45.6(18)	52.2(18)	53.3(15)	58.9(15)	–
$p = 9$	300	71.1(17)	**88.9(9)**	46.7(17)	46.7(18)	45.6(18)	60(18)	–
$p' = 9$	500	73.3(10)	**87.8(9)**	47.8(18)	45.6(17)	47.8(16)	67.8(14)	–
M8	100	74.4(19)	**80(18)**	26.7(26)	36.7(21)	42.2(25)	48.9(26)	–
$p = 9$	300	64.4(9)	**83.3(9)**	34.4(26)	35.6(26)	27.8(26)	48.9(25)	–
$p' = 18$	500	73.3(9)	**90(9)**	22.2(27)	31.1(24)	26.7(26)	48.9(27)	–
M9	100	97.8(9)	**100(9)**	41.1(17)	56.7(18)	56.7(17)	68.9(18)	–
$p = 9$	300	**100(9)**	98.9(9)	40(17)	45.6(17)	54.4(18)	64.4(17)	–
$p' = 9$	500	**100(9)**	**100(9)**	53.3(18)	45.6(16)	56.7(18)	60(16)	–
M9	100	**97.8(9)**	95.6(9)	24.4(27)	32.2(27)	47.8(22)	48.9(19)	–
$p = 9$	300	**100(9)**	**100(9)**	28.9(27)	32.2(25)	38.9(23)	41.1(24)	–
$p' = 18$	500	**100(9)**	98.9(9)	33.3(27)	27.8(27)	45.6(25)	50(16)	–

Values in parentheses represent the highest rank (HR) among the ranks of the p true important variables. Bold values correspond to the best performances. p is the number of true important variables and p' the number of irrelevant variables added to the data. TWKM cannot be computed on nominale data

Table 4 True positive rate (TPR) for the *Toys* dataset

Data	n	CUBT	URF_{gini}	URF_{perm}	LOVO-cubt	LOVO-km	LS	TWKM
Toys	100	**85.7(6)**	34.3(12)	55.8(7)	75.7(12)	73(9)	16.7(12)	83.7(12)
$p = 6$	300	**95.5(6)**	25.7(12)	57.5(7)	65(11)	72.7(7)	6.7(12)	83.3(12)
$p' = 6$	500	**95.7(6)**	23.5(12)	60(12)	58.7(9)	75.2(10)	2.5(12)	83.8(12)
Toys	100	80.5(6)	18.2(18)	37.2(16)	68.8(17)	61.2(18)	11.2(18)	**83.5(16)**
$p = 6$	300	**91.7(6)**	11.2(18)	42.7(15)	50.8(17)	62.3(11)	4.2(18)	83.5(17)
$p' = 12$	500	**95.2(6)**	8.2(17)	40.8(8)	51.2(17)	62(17)	4.3(18)	83.3(18)
Toys								
$p = 6$	100	36.6(469)	0.3(589)	0.0(889)	25(869)	8.3(983)	0(936)	**83.3(895)**
$p' = 994$								

Values in parentheses represent the highest rank (HR) among the ranks of the p true important variables. Bold values correspond to the best performing method(s). p is the number of true important variables and p' the number of irrelevant variables added to the data

a better performance than URF_{gini}. This is not the case for TWKM and LS, whose performances remain stable and increase with sample size (except for model M3), but are still unsatisfactory. CUBT is often placed equal with both LOVO approaches in model M1 (and model M4 for LOVO-cubt).

For nominal datasets CUBT is largely better than the other methods. For models M6 and M7, CUBT and URF_{gini} performances are very close, with a TPR greater than 90%, and outperform the LOVO approaches. The results according to the HR are complementary with those already exposed (higher proportions induce lower ranks).

However, when CUBT is not the best performing method regarding the TPR (it happens 13 times out of the 54 cases), it is performing in the same way as the other best-performing methods (LOVO or URF_{gini}) regarding the HR. Even if the gap with the TPR of other methods is often negligible, the HR ensures that CUBT shows a good ability to detect the important variables in our simulated datasets.

In the case of low separated clusters, CUBT has the best performances except for models M5, M6 and M8, where it is at the second position just after or close to URF_{gini}. The gap is more significant for model M8 that corresponds to the nominal IRT model. For continuous datasets, the TPR retrieved by CUBT is similar to the one in the case of high separated clusters (i.e. $\geq 90\%$). The other methods (except LOVO-km) show poor results and the same behavior is observed for URF_{gini} and URF_{perm} (i.e. worse results when increasing the sample size). CUBT outperforms the other methods except for model M2 with $N = 100$. In this case, CUBT and LOVO-km have the same HR, see Table 3.

For nominal datasets (M5–M9), CUBT and URF_{gini} outperform the other methods, except for model M7 with a low level of noise (in which LOVO-km is better). For models M5 and M9, CUBT and URF_{gini} performances are very close (TPR is $\geq 90\%$). When CUBT is not the best performing approach, its results are similar to the ones obtained with the best performing method (URF_{gini} or LOVO-km), except for model M8 for which URF_{gini} is better than CUBT. In some situations (in models M6, M7 and M8), CUBT is placed equal with the best performing method in terms of HR, even it does not have the highest TPR.

For the *Toys* dataset (see Table 4), CUBT shows the highest values of TPR (except when $p' = 2p$ and $n = 100$ where TWKM is best performing regarding this index) and the best values of HR, detecting perfectly the p true important variables. With 1000 variables, CUBT and TWKM have a better performance than the other methods. While TWKM has the best TPR, CUBT is better than TWKM in terms of HR, meaning that all the true important variables are detected earlier with CUBT.

7 Conclusion

We have presented a new method to assess variable importance based on CUBT. We have compared this approach to other classical scoring methods over several data simulation models, in the presence of noise. CUBT Variable Importance score was the best performing in most experiments. More experiments may be undertaken to analyze the efficiency of the presented scores with respect to correlation or redundancy between the variables and in higher dimensions. We have also analyzed the stability of this score with respect to the data and to the tuning parameters of CUBT (especially the $minsize$ parameter), using heuristics defined in previous works.

Variable importance based on CUBT may be used for feature selection as for supervised learning. The surrogate splits used to compute the score may also be used to handle missing data within CUBT both for the learning process and for predictions. These ideas are now under study.

Acknowledgements We thank Claude Deniau and Pascal Auquier for their valuable comments. This work was partially supported by the Project ECOS SUD U14E02.

References

Belkin M, Niyogi P (2001) Laplacian eigenmaps and spectral techniques for embedding and clustering. Adv Neural Inf Process Syst 14:585–591

Bock RD (1972) Estimating item parameters and latent ability when responses are scored in two or more nominal categories. Psychometrika 37:29–51

Breiman L (1996) Heuristics of instability and stabilization in model selection. Ann Stat 24:6

Breiman L (2001) Random forests. Mach Learn 45(1):5–32

Breiman L, Friedman J, Olshen R, Stone C (1984) Classification and regression trees. Wadsworth and Brooks, London

Chen X, Xu X, Huang JZ, Ye Y (2013) Tw-*k*-means: automated two-level variable weighting clustering algorithm for multiview data. IEEE Trans Knowl Data Eng 25(4):932–944

Fisher R (1936) The use of multiple measurements in taxonomic problems. Ann Eugen 7:179–188

Fraiman R, Ghattas B, Svarc M (2013) Interpretable clustering using unsupervised binary trees. Adv Data Anal Classif 7:125–145

Ghattas B (1999) Importance des variables dans les méthodes cart. Modulad 24:29–39

Ghattas B, Michel P, Boyer L (2017) Clustering nominal data using unsupervised binary decision trees: comparisons with the state of the art methods. Pattern Recognit 67:177–185

Guyon I, Weston J, Barnhill S, Vapnik VN (2002) Gene selection for cancer classification using support vector machines. Mach Learn 46(1–3):389–422

Liaw A, Wiener M (2002) Classification and regression by randomforest. R News 2(3):12–22

Liu H, Yu L (2005) Toward integrating feature selection algorithms for classifcation and clustering. IEEE TKDE 17:491–502

MacQueen J (1967) Some methods for classification and analysis of multivariate observations. In: Neyman J, Le Cam LM (eds) Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, pp 281–297

Muraki E (1992) A generalized partial credit model: application of an em algorithm. Appl Psychol Measur 16:159–176

R Core Team (2013) R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna

Rakotomamonjy A (2003) Variable selection using SVM-based criteria. J Mach Learn Res 3:1357–1370

Reif M (2014) mcIRT: IRT models for multiple choice items. Technical report, R package version 0.41

Rizopoulos D (2006) ltm: an R package for latent variable modelling and item response theory analyses. J Stat Softw 17(5):1–25

Weston J, Elisseff A, Schoelkopf B, Tipping M (2003) Use of the zero norm with linear models and kernel methods. J Mach Learn Res 3:1439–1461

Williams G, Huang JZ, Chen X, Wang Q, Xiao L (2015) wskm: weighted k-means clustering. Technical report, R package version 1.4.28

Zhu L, Miao L, Zhang D (2012) Iterative Laplacian score for feature selection. Pattern Recognit 321:80–87