Kernel KMeans clustering splits for end-to-end unsupervised decision trees

Louis Ohl [1,2] & Pierre-Alexandre Mattei [1] & Mickaël Leclercq [2] & Arnaud Droit [2] & Frédéric Precioso [1]

[1] *Université Côte d'Azur, Inria équipe-projet Maasai, I3S / LJAD, CNRS*
[2] *Université Laval, Centre de recherche du CHU de Québec*
louis.ohl@inria.fr, pierre-alexandre.mattei@inria.fr,
mickael.leclercq@crchudequebec.ulaval.ca,
arnaud.droit@crchudequebec.ulaval.ca, frederic.precioso@univ-cotedazur.fr

Résumé. Les arbres de décisions sont des modèles utiles pour obtenir des prédictions avec explications pour des jeux de données de tailles raisonnables. Alors qu'il existe de nombreuses propositions d'algorithmes pour construire de tels arbres en supervisé d'un seul coup, aucune proposition d'algorithme d'apprentissage en une étape n'existe pour le cas non-supervisé du clustering. Les travaux connexes se concentrent plutôt sur l'apprentissage supervisé via un arbre du résultat d'un premier algorithme de clustering. Nous présentons ici une première proposition de construction d'arbre de décisions pour clustering en une seule étape : Kauri. Kauri utilise une optimisation gloutonne du score K-Moyennes à noyau sans calculer ni définir des centroïdes. Nous comparons Kauri à la concaténation K-Moyennes + CART et montrons de meilleurs performances de clustering lorsque le noyau n'est pas linéaire.

Mot-clefs. Arbres de décisions, clustering, apprentissage non-supervisé, méthodes de noyaux, K-Moyennes

Abstract. Trees are convenient models for obtaining explainable predictions on relatively small datasets. Although there are many proposals for the end-to-end construction of such trees in supervised learning, learning a tree end-to-end for clustering without labels remains an open challenge. As most works focus on interpreting with trees the result of another clustering algorithm, we present here a novel end-to-end trained unsupervised binary tree for clustering: Kauri. This method performs a greedy maximisation of the kernel KMeans objective without requiring the definition of centroids. We compare this model with a KMeans + CART combination and show that Kauri displays better performances for other kernels than the linear kernel.

Keywords. Decision trees, clustering, unsupervised learning, kernel methods, KMeans

1 Introduction

Decision tree classifiers are one of the most intuitive models in machine learning owing to their intrinsic interpretability (Molnar, 2020, Section 3.2). Trees consist of a set of hierarchically sorted nodes that start from one single root node. Each node comprises two or more

conditions, called rules, each of which leads to a different child node. Once a node does not have any children, a decision is returned. A childless node is named a leaf.

While the end model is eventually interpretable, building it implies some questions to be addressed, notably regarding the number of nodes, the feature (or set of features) on which to apply a decision rule, the construction of a decision rule i.e. the number of thresholds and hence the number of children per node. Learning the structure is easier in the case of supervised learning, whereas the absence of labels makes the construction of unsupervised trees more challenging. In recent related works, the problem was oftentimes addressed with twofold methods (Tavallali et al., 2021; Laber et al., 2023): first learning clusters using another algorithm e.g. KMeans, then applying a supervised decision tree to uncover explanations of the clusters. However, such *unsupervised trees* are not fully unsupervised in fact, since their training still requires the presence of external labels for guidance which are provided by KMeans.

To alleviate this dependence on an exterior clustering algorithm:

- We show how the kernel KMeans objective can be rephrased to avoid the computations of centroids, leveraging simple gains to compute for split proposals in decision trees.
- We then introduce an end-to-end unsupervised tree for clustering: KMeans as unsupervised reward ideal (Kauri), which is not restricted to a fixed number of leaves. To the best of our knowledge, this is the first kernel-based end-to-end clustering tree.
- We show that Kauri often displays better performance in clustering to kernel KMeans + Tree using end-to-end training for kernels other than the linear kernel.
- We finally show that Kauri addresses some limitations of the kernel KMeans algorithm.

2 Training trees

In clustering, we do not have access to labels making all notions of gains such as the Gini criterion (Gini, 1912; Breiman, 1984) or the information gain (Quinlan, 1986, 2014) unusable, so we need other tools to guide the decision tree splitting procedure. A common approach is then to keep the algorithm supervised as described in the previous section, yet providing labels that were derived from a clustering algorithm e.g. KMeans (Laber et al., 2023; Held and Buhmann, 1997). In this sense, centroids derived from KMeans can also be involved in split procedures (Tavallali et al., 2021), even to the point that the data from which the centroids are derived do not need to be collected (Gamlath et al., 2021). However, such methods do not properly construct the tree *from scratch* in an unsupervised way despite potential changes in the gain formulations. We are interested in a method that can provide a directly integrated objective to optimise tree training. Other gains derived from entropy formulations have also been proposed (Bock, 1994; Basak and Krishnapuram, 2005). We even note the use of mutual information to achieve deeper and deeper refinements of binary clusters (Karakos et al., 2005).

Oftentimes, these approaches assume that a leaf describes fully a cluster, e.g. Blockeel et al. (1998). Combining leaves into a single cluster requires then post hoc methods (Fraiman

et al., 2013). In such a case, an elegant approach for constructing an unsupervised tree was proposed by Liu et al. (2000) by adding uniform noise to the data and assigning a decision tree to separate the noise from the true data. Such trees put in different leaves dense areas of the data, which can then be labelled manually.

To ensure that several leaves can be assigned to a single cluster, related work also focused on the complete initialisation of a tree and refinement according to a global objective function. For example, Bertsimas et al. (2021) directly maximise the silhouette score or the Dunn index, which are internal clustering metrics and require the initialisation of the tree through greedy construction or KMeans labels. The objective is optimised using a mixed integer optimisation formulation of the tree structure. Lately, Gabidolla and Carreira-Perpinan (2022) proposed to optimise an oblique tree, a structure with logistic regressions at each node, through the alternative optimisation of a distance-based objective, e.g. KMeans, providing pseudo-labels to a tree alternating optimisation problem (Carreira-Perpinan and Tavallali, 2018).

3 Kauri: KMeans as unsupervised reward ideal

The Kauri tree is a non-differentiable binary decision tree that looks in many ways alike the CART algorithm. It constructs from scratch a binary tree giving hard clustering assignments to the data by using an objective equivalent to the optimisation of a kernel KMeans. In the Kauri structure, a cluster can be described by several leaves.

3.1 Objective function

We consider that we have a dataset of n samples: $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^n$. We write the partition into $K_{\max}$ clusters as:

$$\mathcal{C}_k = \{\boldsymbol{x}_i \in \mathcal{X}_k\}, \forall k \leq K_{\max}, \tag{1}$$

with $\{\mathcal{X}_k\}_{k=1}^{K_{\max}}$ a partition of the data space $\mathcal{X} \subseteq \mathbb{R}^d$.

The kernel KMeans algorithm minimises the cluster sum of squares in a Hilbert space $\mathcal{H}$ with projection φ and kernel κ with respect to $K_{\max}$ centroids $\boldsymbol{\mu}_k \dots \boldsymbol{\mu}_{K_{\max}}$:

$$\mathcal{L}_{\text{KMeans}} = \sum_{k=1}^{K_{\max}} \sum_{\boldsymbol{x} \in \mathcal{C}_k} \|\varphi(\boldsymbol{x}) - \boldsymbol{\mu}_k\|_{\mathcal{H}}^2. \tag{2}$$

Instead of computing the sample-wise distance to the centroid using the kernel trick (Dhillon et al., 2004), Kauri optimises this objective without explicitly computing centroids per cluster. To that end, we use the following simple equality:

$$\sum_{\boldsymbol{x} \in \mathcal{C}_k} \|\varphi(\boldsymbol{x}) - \boldsymbol{\mu}_k\|_{\mathcal{H}}^2 = \frac{1}{2\,|\mathcal{C}_k|} \sum_{\boldsymbol{x},\boldsymbol{y} \in \mathcal{C}_k} \|\varphi(\boldsymbol{x}) - \varphi(\boldsymbol{y})\|_{\mathcal{H}}^2. \tag{3}$$

Once inserted into the kernel KMeans objective, we get an alternative formulation without centroids:

$$\mathcal{L}_{\text{KMeans}} = \sum_{k=1}^{K_{\max}} \frac{1}{2\,|\mathcal{C}_k|} \sum_{\boldsymbol{x},\boldsymbol{y} \in \mathcal{C}_k} \|\varphi(\boldsymbol{x}) - \varphi(\boldsymbol{y})\|_{\mathcal{H}}^2. \tag{4}$$

Using the kernel trick, we can rephrase this objective as:

$$\mathcal{L}_{\text{KMeans}} = \sum_{k=1}^{K_{\max}} \frac{1}{2\,|\mathcal{C}_k|} \sum_{\boldsymbol{x},\boldsymbol{y}\in\mathcal{C}_k} (\kappa(\boldsymbol{x},\boldsymbol{x}) + \kappa(\boldsymbol{y},\boldsymbol{y}) - 2\kappa(\boldsymbol{x},\boldsymbol{y})), \tag{5}$$

where the two first kernel terms can be summarised as the size of clusters weighting the diagonal elements of the kernel. Finally, the third term is the grand sum of the kernel of the cluster, and thus:

$$\mathcal{L}_{\text{KMeans}} = \sum_{\boldsymbol{x}\in\mathcal{D}} \kappa(\boldsymbol{x},\boldsymbol{x}) - \sum_{k=1}^{K_{\max}} \frac{1}{|\mathcal{C}_k|} \sum_{\boldsymbol{x},\boldsymbol{y}\in\mathcal{C}_k} \kappa(\boldsymbol{x},\boldsymbol{y}). \tag{6}$$

For the sake of simplicity, we introduce the function σ that sums the kernel values $\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j) = \langle\varphi(\boldsymbol{x}_i),\varphi(\boldsymbol{x}_j)\rangle$ of samples indexed by two sets:

$$\sigma(E\times F) = \sum_{\substack{\boldsymbol{x}_i\in E\\ \boldsymbol{x}_j\in F}} \kappa(\boldsymbol{x}_i,\boldsymbol{x}_j). \tag{7}$$

We will refer to the σ function as the *kernel stock*. This function is bilinear with respect to the input spaces. We provide in Figure 1 a visual explanation of its different usages.

We finally note that the first term of Eq. (6) is a constant because it does not depend on the clustering. With the only the second term remaining, we can remove the minus sign and hence maximise the objective:

$$\mathcal{L}_{\text{Kauri}} = \sum_{k=1}^{K_{\max}} \frac{\sigma(\mathcal{C}_k^2)}{|\mathcal{C}_k|}, \tag{8}$$

Therefore, maximising our objective function $\mathcal{L}$ is equivalent up to a constant to minimising a KMeans objective for any kernel. However, in contrast to Eq. (2), it is not a function of centroids, but a function of a partition.

3.2 Tree branching

For supervised trees like CART or ID3, the types of splits are binary and guided by the labels which tell us to which class each child node should go. For unsupervised trees, we must consider all possibilities: to which cluster goes the left child, to which cluster goes the right child on what set of features to do the split, on what threshold in this feature to split and on which nodes. We note $\mathcal{T}_p$ the set of samples reaching the p-th node. For a split, let $\mathcal{S}_L$ be the subset of samples from the node samples $\mathcal{T}_p$ that will go to the left child node and $\mathcal{S}_R$ the complementary subset of samples that will go to the right child node. Each child node will be assigned to a different cluster, whether new, already existing, or equal to the parent node's cluster assignment. Let k_p be the current cluster membership of the parent node p, k_L the future cluster membership for the left child node and k_R the future cluster membership of the right child node, then $\mathcal{S}_L \cup \mathcal{S}_R = \mathcal{T}_p \subseteq \mathcal{C}_{k_p}$ and after splitting: $\mathcal{S}_L \subseteq \mathcal{C}_{k_L}$ and $\mathcal{S}_R \subseteq \mathcal{C}_{k_R}$.

We enforce the following constraints: a child node must stay in the parent node's cluster if both children leaving would empty the parent's cluster; the creation of a new cluster can only be done under the condition that the number of clusters does not exceed a specified limit $K_{\max}$. We also impose a maximum number of leaves $T_{\max}$ which can be equal to at most the number of samples n. It is nonetheless possible that the algorithm stops the splitting procedure if all gains become negative before reaching the maximum number of leaves allowed. Unlike Gabidolla and Carreira-Perpinan (2022) who recently proposed unsupervised oblique trees, we choose to keep splits on a single feature because this lowers the complexity of greedy exploration.

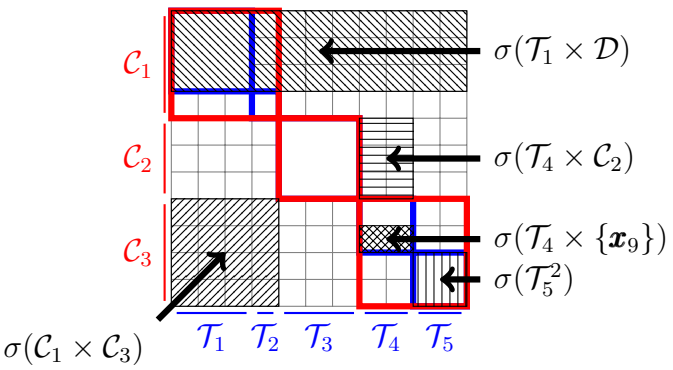


Figure 1: A toy example with a dataset consisting of 11 samples partitioned in 3 clusters using 5 leaves in a tree. The matrix represents the kernel between all pairs of samples and dashed areas correspond to the sum of kernel elements according to the *kernel stock* function σ.

Thus, learning consists in greedily exploring from all nodes the best split and either taking this split to build a new cluster or merging with another cluster. We now present the objective function and related gains depending on the children's cluster memberships.

3.3 Gain metrics

We can derive from the objective of Eq. (8) four gains that evaluate how much score we get by assigning one child node to a new cluster, assigning both child nodes to two new clusters, merging one child node to another cluster, or merging both child nodes to different clusters. We denote by $\mathcal{C}'_\bullet$ the clusters after the split operation and by $\mathcal{C}_\bullet$ the clusters before the split. Hence, the global gain metric is:

$$\Delta\mathcal{L}(\mathcal{S}_L : k_p \to k_L, \mathcal{S}_R : k_p \to k_R) = \frac{\sigma(\mathcal{C}'^{\,2}_{k_L})}{|\mathcal{C}'_{k_L}|} + \frac{\sigma(\mathcal{C}'^{\,2}_{k_R})}{|\mathcal{C}'_{k_R}|} + \frac{\sigma(\mathcal{C}'^{\,2}_{k_p})}{|\mathcal{C}'_{k_p}|} - \frac{\sigma(\mathcal{C}^2_{k_L})}{|\mathcal{C}_{k_L}|} - \frac{\sigma(\mathcal{C}^2_{k_R})}{|\mathcal{C}_{k_R}|} - \frac{\sigma(\mathcal{C}^2_{k_p})}{|\mathcal{C}_{k_p}|}, \quad (9)$$

which corresponds to subtracting the contribution of the kernel stocks of the former clusters and adding the kernel stocks of the new clusters after splitting.

From this global gain metric, we derive four different gains: the *star gain* $\Delta\mathcal{L}^\star$ for assigning either the left or right child of a leaf to a new cluster, the *double star gain* $\Delta\mathcal{L}^{\star\star}$ for assigning the left and right children of a leaf to two new clusters, the *switch gain* $\Delta\mathcal{L}^{\rightleftarrows}$ for assigning either the left or right child of a leaf to another existing cluster and the *reallocation gain* $\Delta\mathcal{L}^{\hookrightarrow}$ for assigning respectively the left and right children to different existing clusters. In

Table 1: Summary of the datasets used in the experiments.

Name	Breast Cancer	Iris	Lsun	Mice protein	Target	Twodiamonds
Samples	683	150	404	552	770	800
Features	9	4	3	77	2	2
Classes	2	3	3	8	6	2

Table 2: ARI scores $_{\text{std}}$ (greater is better) after 30 runs on random subsamples of 80% of the input datasets for varying kernels. All models are limited to finding 4 times more leaves than clusters.

Kernel	Additive χ^2		χ^2		Laplacian		RBF	
Dataset	Kauri	KMeans+DT	Kauri	KMeans+DT	Kauri	KMeans+DT	Kauri	KMeans+DT
Cancer	$0.86_{0.01}$	$0.84_{0.02}$	$0.82_{0.02}$	$0.86_{0.02}$	$\mathbf{0.87_{0.01}}$	$0.79_{0.03}$	$\mathbf{0.87_{0.02}}$	$0.78_{0.03}$
Iris	$0.67_{0.04}$	$0.62_{0.09}$	$0.67_{0.04}$	$0.61_{0.11}$	$\mathbf{0.78_{0.05}}$	$0.71_{0.14}$	$0.72_{0.03}$	$0.71_{0.05}$
Lsun	$\mathbf{0.98_{0.00}}$	$0.89_{0.21}$	$\mathbf{0.98_{0.01}}$	$0.81_{0.26}$	$\mathbf{0.98_{0.01}}$	$0.96_{0.01}$	$0.88_{0.02}$	$0.93_{0.02}$
Twodiamonds	$0.98_{0.00}$	$0.95_{0.02}$	$0.98_{0.00}$	$0.97_{0.02}$	$\mathbf{1.00_{0.00}}$	$0.77_{0.43}$	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$
Wine	$0.87_{0.03}$	$0.74_{0.03}$	$\mathbf{0.90_{0.03}}$	$0.73_{0.10}$	$0.89_{0.03}$	$0.83_{0.03}$	$0.85_{0.03}$	$0.80_{0.04}$

practice, for the p-th leaf with split proposals $\mathcal{S}_L$ and $\mathcal{S}_R$, we do not need to compute the future cluster kernel stocks $\sigma(\mathcal{C}'_\bullet)$ and instead use the stocks between the splits and the current clusters: $\sigma(\mathcal{S}_{L/R} \times \mathcal{C}_\bullet)$. The code of Kauri is available in the GemClus package: `https://gemini-clustering.github.io`.

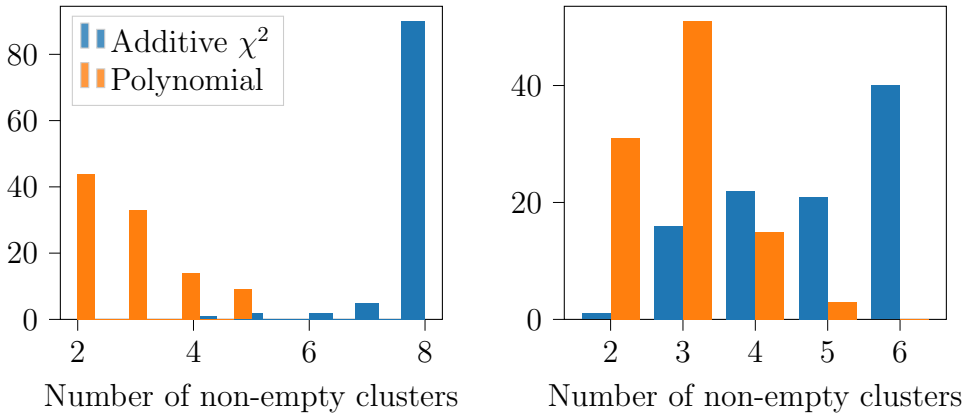
4 Experiments

The datasets used in the experiments are described in Table 1. We will assess the general clustering performances using the adjusted rand index (ARI, Hubert and Arabie, 1985) and the KMeans score normalised by the reference score of the sole kernel KMeans algorithm for cluster quality. We used minmax scaling for all datasets to ensure the computation of χ^2 kernels. We tossed away all samples that presented missing values for the sake of simplicity. Finally, we introduced stochasticity by measuring the performances on random samples of 80% of the dataset to be fair between the stochastic nature of kernel KMeans and the deterministic nature of Kauri due to its greedy construction.

We compare the performances of Kauri against the concatenation of a kernel KMeans algorithm and a supervised decision tree for different kernel (KMeans+DT). However, using a different kernel leads to the absence of a definition of centroids in the Euclidean space where the data lie. Consequently, the available implementations of the related work (Frost et al., 2020; Makarychev and Shan, 2022; Laber et al., 2023) are not compatible with this set-up because they generally require a centroid. We explore 4 different kernels with default parameters from `scikit-learn` (Pedregosa et al., 2011): χ^2, additive χ^2, Laplacian and RBF kernels with Table 2 for the ARI and Table 3 for the normalised KMeans score. We also allow the algorithms to find 4 times more leaves than clusters, as was done by Frost et al. (2020).

Table 3: Relative Kernel KMeans scores $_{\text{std}}$ (lower is better) of Kauri and Kernel-KMeans + Decision Tree after 30 runs on random subsamples of 80% of the input datasets for varying kernels. All models are limited to finding 4 times more leaves than clusters.

Kernel	Additive χ^2		χ^2		Laplacian		RBF	
Dataset	Kauri	KMeans+DT	Kauri	KMeans+DT	Kauri	KMeans+DT	Kauri	KMeans+DT
Cancer	$1.04_{0.02}$	$1.05_{0.02}$	$1.02_{0.01}$	$1.03_{0.01}$	$\mathbf{1.01_{0.02}}$	$1.04_{0.02}$	$1.04_{0.02}$	$1.07_{0.03}$
Iris	$1.11_{0.05}$	$1.18_{0.16}$	$1.10_{0.04}$	$1.20_{0.16}$	$\mathbf{1.05_{0.02}}$	$1.09_{0.08}$	$1.06_{0.05}$	$1.06_{0.09}$
Lsun	$1.08_{0.03}$	$1.13_{0.11}$	$\mathbf{1.04_{0.02}}$	$1.11_{0.12}$	$\mathbf{1.04_{0.01}}$	$\mathbf{1.04_{0.01}}$	$1.05_{0.03}$	$1.07_{0.04}$
Twodiamonds	$1.03_{0.02}$	$1.03_{0.02}$	$1.05_{0.02}$	$1.05_{0.02}$	$\mathbf{1.01_{0.01}}$	$1.06_{0.09}$	$1.04_{0.01}$	$1.03_{0.02}$
Wine	$1.03_{0.03}$	$1.05_{0.04}$	$1.05_{0.02}$	$1.08_{0.05}$	$\mathbf{1.02_{0.02}}$	$\mathbf{1.02_{0.02}}$	$1.06_{0.03}$	$1.08_{0.03}$



(a) Mice protein (8 classes)

(b) Target (6 classes)

Figure 2: Number of non-empty clusters for 100 runs of kernel KMeans with an additive χ^2 or polynomial kernel. The algorithm had to find the same number of clusters as classes per dataset.

We generally observe equal or stronger ARI scores for the KAURI algorithm in Table 2. The same observation goes for the KMeans score in Table 3, especially for the Laplacian kernel in both tables. Note that we could not run this experiment on the Mice protein and Target datasets. This is due to the phenomenon of empty clusters that arises in the kernel KMeans algorithm. Consequently, the basis on which the decision tree is learnt does not provide enough clusters, thus lowering the ARI and increasing the kernel KMeans score for these two excluded datasets. Similarly, the reference kernel KMeans score used for normalisation suffered from the same problem. Empirically, we observed this behaviour for 2 implementations of the kernel KMeans algorithm.[1]

As a simple example, we ran 100 kernel KMeans models with an additive χ^2 kernel or a polynomial kernel for the Mice protein dataset and the Target dataset. From Figure 2, we observe that for a relatively small number of clusters, the kernel KMeans may converge to several empty clusters, with the worst effect from the polynomial kernel in this example. To complete Figure 2, the number of clusters found with the best kernel KMeans score was 8

[1] `https://gist.github.com/mblondel/6230787`, and the `tslearn` v0.6.3 implementation (Tavenard et al., 2020).

for the Mice protein dataset and 6 for the Target dataset. In contrast, datasets with only 2 clusters to find often converged to the good number of clusters, making scores comparable.

This shows that with the end-to-end construction of the Kauri tree, we do not suffer from dependence to the basis KMeans algorithm, and manage to get the correct user-desired number of clusters.

5 Final words

We introduced a novel model optimising a kernel KMeans objective: Kauri. By rephrasing the kernel KMeans objective to drop the requirement for centroids, we leveraged an easy criterion to maximise and derived gains for an iterative splitting procedure in a CART-like binary tree. In contrast to related work on unsupervised binary decision trees, Kauri is compatible with kernels other than the linear kernel. Moreover, we showed that the algorithm does not suffer from an empty cluster phenomenon that arises in the kernel KMeans algorithm and will fill all clusters. When the kernel KMeans algorithm converges to the correct number of clusters, the performances of Kauri remain greater than the KMeans+DT baseline. Hence, the strong advantage of this method is building an interpretable by-nature clustering instead of seeking to explain another clustering output from a different algorithm. Future work will focus on the integration of linear splits using several features, such as in oblique trees (Gabidolla and Carreira-Perpinan, 2022).

Acknowledgements

This work has been supported by the French government, through the 3IA Côte d'Azur, Investment in the Future, project managed by the National Research Agency (ANR) with the reference number ANR-19-P3IA-0002. We would also like to thank the France Canada Research Fund (FFCR) for their contribution to the project. This work was partly supported by the Fonds de recherche du Québec – Santé (FRQS) and the Health-Data Hub, through the joint project AORTIC STENOSIS.

References

Jayanta Basak and Raghu Krishnapuram. Interpretable Hierarchical Clustering by Constructing an Unsupervised Decision Tree. *IEEE transactions on knowledge and data engineering*, 17(1):121–132, 2005. Publisher: IEEE.

Dimitris Bertsimas, Agni Orfanoudaki, and Holly Wiberg. Interpretable Clustering: an Optimization Approach. *Machine Learning*, 110(1):89–138, January 2021. ISSN 1573-0565. doi: 10.1007/s10994-020-05896-2.

Hendrik Blockeel, Luc De Raedt, and Jan Ramon. Top-Down Induction of Clustering Trees. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 55–63, 1998.

HH Bock. Information and Entropy in Cluster Analysis. In *Proceedings of the First US/Japan Conference on the Frontiers of Statistical Modeling: An Informational Approach: Volume 2 Multivariate Statistical Modeling*, pages 115–147. Springer, 1994.

L Breiman. Classification and Regression Trees. *The Wadsworth & Brooks/Cole*, 1984. Publisher: Advanced Books & Software.

Miguel A Carreira-Perpinan and Pooya Tavallali. Alternating optimization of decision trees, with application to learning sparse oblique trees. *Advances in neural information processing systems*, 31, 2018.

Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Kernel K-Means: Spectral Clustering and Normalized Cuts. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 551–556, 2004.

Ricardo Fraiman, Badih Ghattas, and Marcela Svarc. Interpretable Clustering using Unsupervised Binary Trees. *Advances in Data Analysis and Classification*, 7:125–145, 2013. Publisher: Springer.

Nave Frost, Michal Moshkovitz, and Cyrus Rashtchian. ExKMC: Expanding Explainable k-Means Clustering. *arXiv preprint arXiv:2006.02399*, 2020.

Magzhan Gabidolla and Miguel A Carreira-Perpinan. Optimal Interpretable Clustering using Oblique Decision Trees. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 400–410, 2022.

Buddhima Gamlath, Xinrui Jia, Adam Polak, and Ola Svensson. Nearly-Tight and Oblivious Algorithms for Explainable Clustering. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 28929–28939. Curran Associates, Inc., 2021.

Corrado W Gini. Variability and Mutability, Contribution to the Study of Statistical Distributions and Relations. *Studi Economico-Giuridici della R. Universita de Cagliari*, 1912.

Marcus Held and Joachim Buhmann. Unsupervised on-line Learning of Decision Trees for Hierarchical Data Analysis. *Advances in neural information processing systems*, 10, 1997.

Lawrence Hubert and Phipps Arabie. Comparing Partitions. *Journal of classification*, 2(1): 193–218, 1985. Publisher: Springer.

Damianos Karakos, Sanjeev Khudanpur, Jason Eisner, and Carey E Priebe. Unsupervised Classification via Decision Trees: An Information-Theoretic Perspective. In *Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, volume 5, pages v–1081. IEEE, 2005.

Eduardo Laber, Lucas Murtinho, and Felipe Oliveira. Shallow Decision Trees for Explainable K-means Clustering. *Pattern Recognition*, 137:109239, 2023. Publisher: Elsevier.

Bing Liu, Yiyuan Xia, and Philip S Yu. Clustering Through Decision Tree Construction. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 20–29, 2000.

Konstantin Makarychev and Liren Shan. Explainable K-Means: Don't be Greedy, Plant Bigger Trees! In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pages 1629–1642, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 978-1-4503-9264-8. doi: 10.1145/3519935.3520056. event-place: Rome, Italy.

Christoph Molnar. *Interpretable Machine Learning*. Lulu. com, 2020.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

J. Ross Quinlan. Induction of Decision Trees. *Machine learning*, 1:81–106, 1986. Publisher: Springer.

J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

Pooya Tavallali, Peyman Tavallali, and Mukesh Singhal. K-means Tree: An Optimal Clustering Tree for Unsupervised Learning. *The Journal of Supercomputing*, 77:5239–5266, 2021. Publisher: Springer.

Romain Tavenard, Johann Faouzi, Gilles Vandewiele, Felix Divo, Guillaume Androz, Chester Holtz, Marie Payne, Roman Yurchak, Marc Rußwurm, Kushal Kolar, and Eli Woods. Tslearn, A Machine Learning Toolkit for Time Series Data. *Journal of Machine Learning Research*, 21(118):1–6, 2020.