Periodic split method: learning more readable decision trees for human activities

Matthieu Boussard [1] Clodéric Mars[1] Rémi Dès[1] Caroline Chopinaud[1]

[1] craft ai

matthieu@craft.ai cloderic@craft.ai remi@craft.ai caroline@craft.ai

Abstract

Placing your trust in algorithms is a major issue in society today. This article introduces a novel split method for decision tree generation algorithms aimed at improving the quality/readability ratio of generated decision trees. We focus on human activities learning that allow the definition of new temporal features. By virtue of these features, we present here the periodic split method*, which produces similar or superior quality trees with reduced tree depth.*

Keywords

Machine Learning, Decision trees, Split, Temporal data, C4.5

Résumé

La confiance dans les algorithmes est un problème de société majeur. Cet article introduit une nouvelle méthode de calcul de split pour la génération d'arbres de décisions permettant l'amelioration du rapport qualité/lisibilité des arbres générés. En nous focalisant sur l'apprentissage d'activité humaines, nous définissons des nouveaux attributs temporels. Grâce à ces nouveaux attributs, nous introduisons la méthode de split périodique *qui permet de réduire la profondeur des arbres pour des performances similaires.*

Mots Clef

Apprentissage automatique, Arbres de décisions, Split, Données temporelles, C4.5

1 Introduction

Trust in algorithms is a major question for societies today. Crucially, the justification of the decision made by an AI-based system regarding humans will become an essential requirement. When a system interacts with humans, it should be transparent enough to check decision results and detect illicit discrimination [3]. This topic has taken on such importance, that a law[1] was passed on March 14th in France; it states that *"rules making decision concerning individuals based on algorithms can be subject to explanation on user request"*

[1]LOI n° 2016-1321 du 7 octobre 2016 pour une République numérique

User-centric applications such as home automation, coaching or personal assistant services can take advantage of user data to provide personalized and adaptive interactions. To this end, applications must be able to continuously learn from a user's habits and preferences in order to predict their needs, in particular when bearing in mind the time factor.

Machine learning algorithms are powerful tools that autonomously find hidden rules or relations in data, and can be used to classify, predict or decide based on what they have learned. In this context, one of the difficulties is to balance predicting power and explainability. For instance, with sufficient data, deep-neural networks (DNNs) now have a impressive power to predict. However, the neural network on which decisions are being make can be considered as a *"black box"* where, even though it is possible to read every weight of the DNN, is is almost impossible to explain specific decisions.

Decision trees are a way to describe predictive models, and up to a certain size they can be understood and explained, and even considered *"white boxes"*. Algorithms like CART [1] or C4.5 [8] generate predictive decision trees from training data. Compared to *"black box"* learning algorithms, such as neural networks, they can learn quickly and perform well even when provided with a very limited amount of data. However, decision trees tend to be less accurate than the aforementioned

Decision tree learning algorithms are therefore very interesting in the context of learning from human behavior to enable automation, in the main because of their explainability. Furthermore, one area of particular importance lies in the repetitions and seasonality in the data. For instance, if we want to learn the sleeping habits of a person, we may want to learn when he/she falls asleep, when he/she wakes up in the morning, and whether he/she changes his/her sleeping habits during the weekend, etc. It is already possible to model time data such as the day of the week or the time of day as continuous values to generate decision trees with standard methods, tough this can lead to decision trees that are unnecessarily complex and difficult to read, and does not accurately represent the periodic nature of repetitive time data.

In this paper we introduce a new representation and associated decision tree splitting method for periodic time properties such as the time of day or day of the week. This new

method aims to improve the quality of predictive decision trees in terms of explainability and simplicity, especially when applied to data describing human activity. The paper is organized as follows : section 2 presents the related works, section 3 the formal definition of a split, and finally we present the main contribution of the paper, the periodic split, in section 4, experiments are shown in section 5.

2 Related works

Much research has been carried out in the field of decision trees, in an effort to improve their quality/readability. We would like to present here related works, but also to distinguish this work from previous approaches using similar topics.

2.1 Spatial decision trees

In the initial C4.5 decision tree algorithms three types of data are taken into account : continuous, discrete and categorical. Each type leads to a different split method. To improve the quality of the split, it is possible to add new types. For instance, many researchers have worked on learning based on spatial data. In these cases, the classical splits perform poorly, and the typing of certain features as *"spatial types"* can improve results. For instance, in [5], the authors use oblique splits for spatial data because they are better suited to this type of data. In [7] the authors generate predicate functions to create new features and then learn the tree using both spatial and non spatial features.

2.2 Handling time in decision trees

This paper presents a new way of handling time in decision trees and differs from previous work by not applying first a transformation on the temporal information and then applying a classical decision tree algorithm, but instead, by modifying the way the splits are computed on a temporal feature

There are also various ways of representing temporal information. For instance, decision tree algorithms have been adapted to split on time series by introducing a similarity measure between time series [10]. However this work, as well as others interested in time series, does not focus on human activities, which is where our prime interest lies.

3 Splits and multisplits algorithms

Split algorithms are a crucial part of decision tree generation. Multisplit is an extension of the classical split aimed at improving the quality and readability of the resulting decision tree. In this section we formally define both.

Let S be a training set. A sample $s \in S$ is a vector $\vec{s} = \langle s_0, \dots s_n, c \rangle$, composed of i features and a class c. The value of the feature i for sample s is noted as $val_i(s)$. Classical kinds of feature value types are numerical (discrete or continuous) and categorical.

A split function over S takes an feature i and splits it into k subsets $S = \bigcup_{j=0}^{k} S_j$ according to the values of $val_i(s)$. For instance, equation 1 represents a split on a continuous feature according to a threshold value T.

$$\begin{cases} S_{inf}\{s \in S | val_i(s) \leq T\} \\ S_{sup}\{s \in S | val_i(s) > T\} \end{cases} \tag{1}$$

To build decision trees, classical algorithms, like C4.5, compute for every feature the maximum (possible) gain if S is split into subsets according to every possible split value. This gain is based on various measures, depending on the algorithm : for instance C4.5 uses information gain measures. Once every maximum gain has been computed, the split feature i is selected with its best split value T by selecting the couple with the best gain. Following this, two subsets are computed using Eq.1 and the algorithm proceeds recursively on the two created subsets.

However, as stated in [2], non-binary splits [4] on continuous features make the trees easier to understand and also seem to lead to more accurate trees in some domains. A n-split on a continuous value is described by Eq.2.

$$\begin{cases} S_0\{s \in S | val_i(s) \leq T_0\} \\ S_1\{s \in S | val_i(s) \in]T_0, T_1]\} \\ \vdots \\ S_{n-1}\{s \in S | val_i(s) > T_{n-1}\} \end{cases} \tag{2}$$

Multisplit algorithms belong to two families depending on how their arity is defined. The number of splits, or cut points, can either be fixed in advance [4], or discovered automatically during the process [9]. We introduce a method that belongs to the former to create a ternary split on periodic data.

4 Split on periodic data

In this paper, we introduce a new split mechanism, namely the *periodic split operator*. This operator, given two bounds, will split data into two complementary subsets. The decision tree algorithm remains unchanged, but the possible splits are computed using Eq.3. The motivation behind introducing this new operator is that many common data types do not fit into classical, or even non-binary, splits. We will further discuss these types in section 4.1.

$$\begin{cases} S_{in}\{s \in S | val_i(s) \in [T_{min}, T_{max}]\} \\ S_{out}\{s \in S | val_i(s)) \notin [T_{min}, T_{max}]\} \end{cases} \tag{3}$$

4.1 Relevant human periodic time types

The motivation behind introducing the periodic split is the need for a proper way of handling temporal data that occurs during day-to-day human activities. For instance, if we need to learn the sleeping habits of a person, as shown in Fig.2, he/she may fall asleep at 10pm, and wake up at 6am. In this use case the usual $<$ and $\geq$ operators don't make sense in terms of readability : 1am is after 10pm but 1 is not more than 22. The mathematical operators we use on

continuous values do not match how humans think about time.
We propose 4 temporal types, namely *"Time of day", "Day of week", "Day of month" and "Month of year"* for use in the model definition. These types define a period after which the feature value returns to its initial value. Periods for each type are represented in Fig.1.
In order to clarify and unify operators in the tree we propose the following notation for periodic intervals, with a and b between 0 and a period P , $[a, b[=$

$$\begin{cases} \{nP + r : n \in \mathbb{Z}, r \in \mathbb{R} | r \geq a \wedge r < b\} & \text{if a < b} \\ \varnothing & \text{if a=b} \\ (\{nP + r : n \in \mathbb{Z}, r \in \mathbb{R} | r \geq a \wedge r < P\} \cup \\ \{nP + r : n \in \mathbb{Z}, r \in \mathbb{R} | r < b \wedge r \geq 0\}) & \text{if a>b} \end{cases} \tag{4}$$

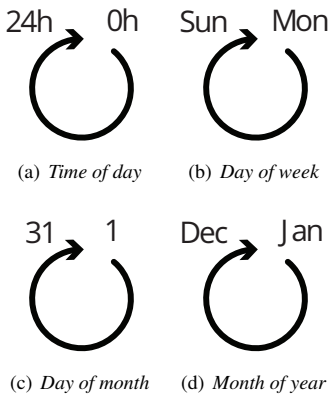
A feature of type *"Time of day"* can be used to focus on activities occurring on a daily basis, permitting the detection of any partitioning S_{in}, S_{out} of the period $]0, 24]$, as follows :

$$\begin{cases} S_{in} =]10pm, 6am] \\ S_{out} =]6am, 10pm] \end{cases} \tag{5}$$

This notation is a short for :

$$\begin{cases} S_{in} =]10pm, 12am] \cup]12am, 6am] \\ S_{out} =]6am, 10pm] \end{cases} \tag{6}$$

The other types are handled in the same way. Additionally, we always consider that a *"Day of month"* is in $[1; 31]$ even if some months are shorter. This assumption has no impact on the split quality.



(a) *Time of day* (b) *Day of week*

(c) *Day of month* (d) *Month of year*

Figure 1: Usual time periods

Once a decision tree has been learned, we simply use the in operator to make a decision,

4.2 Algorithm

We provide a naive algorithm to compute splits on periodic types. While it is straightforward to use non-binary split

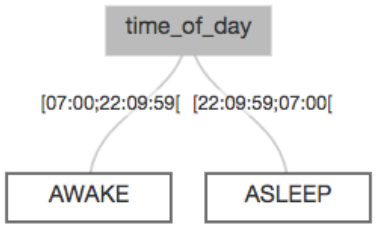


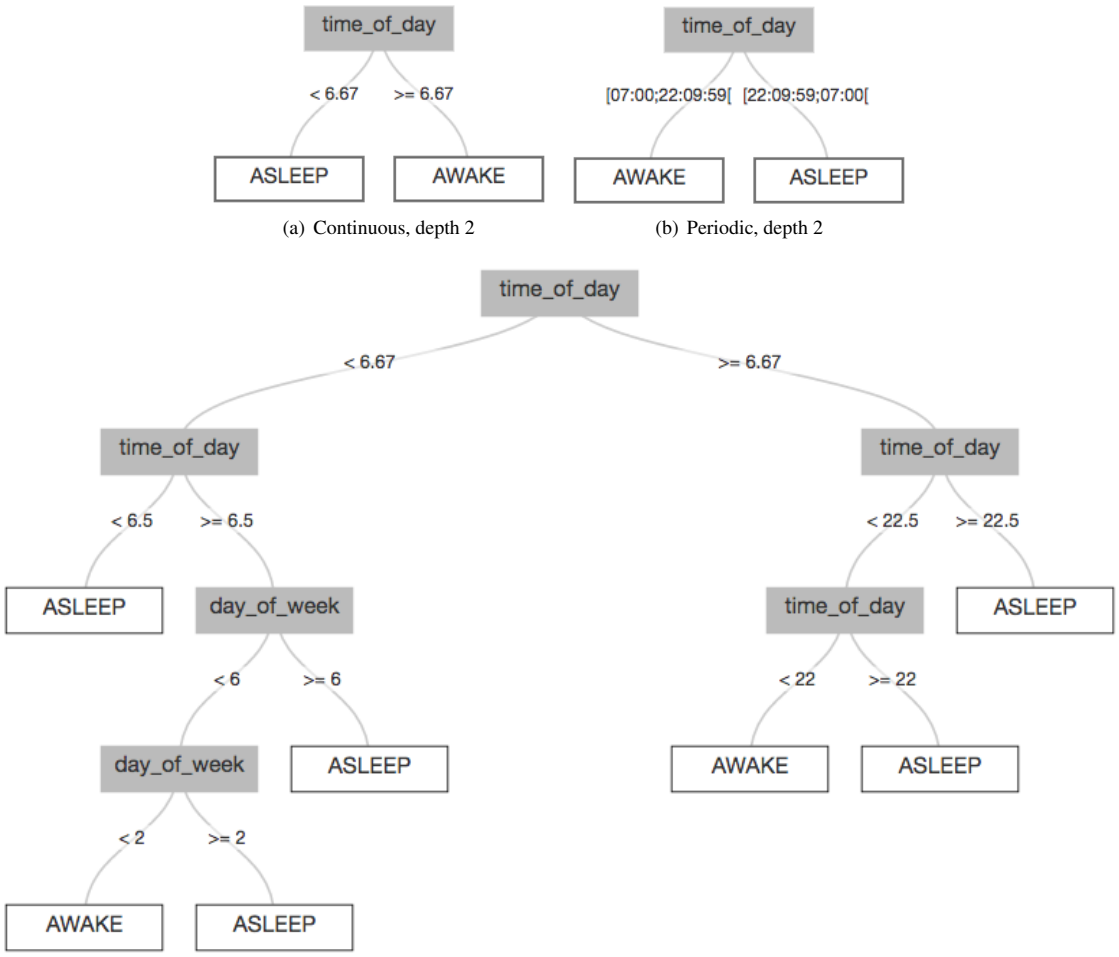
Figure 2: Example of periodic split

methods, for the sake of clarity, we stay focused on a simpler binary split algorithm. The interested reader can refer to [9] for further details on non-binary splits.
Since splits are a partition of the input dataset, one and only one partition will contain the start/end point of the period (an interval $]a, b]$ with $a > b$). We are only considering binary splits, hence we first find the non-looping split, while the other is the complement of this interval over the period. The algorithm to find the best gain for a periodic feature i is :

1. Sort S in ascending order according to $val_i(s)$
2. For each $a \in S'$
   (a) For each $b \in S' > a$
   (b) Compute the information gain when splitting in $]a, b]$ and $]b, a]$
   (c) Save the gain and the splits if better that current maximum
3. return best gain and intervals

5 Results

We use data from a connected watch (a Withings Activity) to predict whether a person is sleeping or not at a given moment in time. For clarity, we built a very simple model where each observation is defined by $s = \langle time\_of\_day, day\_of\_week, state? \rangle$, where state is either 'AWAKE' or 'ASLEEP' depending on the sleeping status of the person. The structure of the dataset is presented Table.1 : it is made of the status of a person, measured every 10 minutes. The overall activity for our two users is presented 4. We then compare the results obtained using classical splits against the periodic splits introduced in this paper. Data for two different and independent persons, c and s, are collected over a three month period and used for classification, while a fourth month is used for validation. We use the $F1$ score (computed by $2 * \frac{precision * recall}{precision + recall}$) and show how it evolves with the maximum depth of the tree. The results are presented in Fig.5. For a depth of 2, the two models using periodic intervals perform far better than those using classical continuous splits. This is because our method performs 2 successive binary splits in one single step. We have to increase the



(a) Continuous, depth 2 (b) Periodic, depth 2

(c) Continuous, depth 5

Figure 3: Impact of periodic types in generated decision trees, dataset c

depth to 5 before the continuous splits attain the same quality as the periodic methods. Further increasing the depth does not impact performance, both algorithms achieving the same quality.

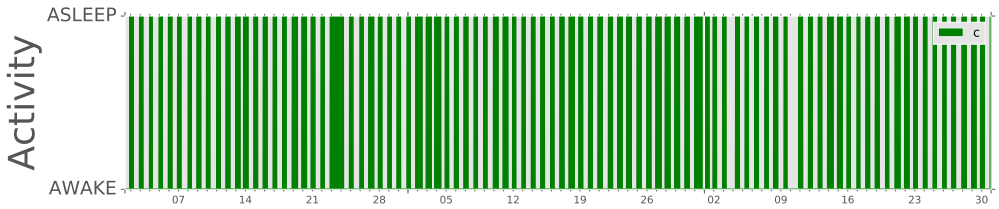
6 Conclusions

In this article, we present a new split method for decision tree generation algorithms. We demonstrated it could be applied to C4.5 to achieve a better performance/depth ratio compared to the traditional continuous split method in a classification use case. The same periodic split method can be used in other decision tree generation algorithms both for classifications, using other scoring method such as the gini coefficient, or for regressions with the same advantages.

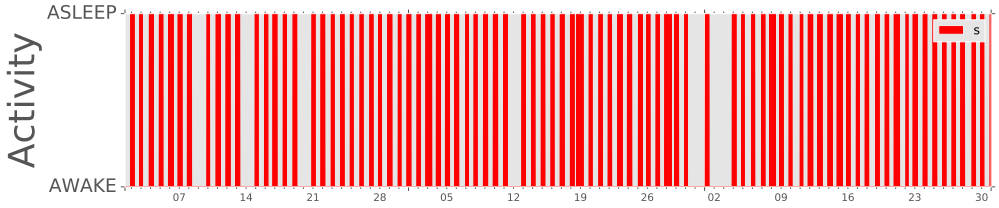
The naive implementation we describe in section 4.2 has a complexity of $\mathcal{O}(n^2)$ which is fine for simple use cases but leads to bad performances on large data sets. Now that we have validated the gain of our approach, we are investigating different strategies to achieve a linear complexity by reducing the search space of the two split points thanks to simple precalculations.

We focused on the application of the periodic split method on time, using basic time units as periods : the day, the week, and the month. These periods have a high relevance when it comes to modeling human activities. However, further to this, other time periods can be significant, such as fortnights, lunar cycles, quarters. Beyond human activities other phenomena follow different time periods. We are investigating the usage of feature engineering techniques detecting periods in data sets such as autocorrelation [6] in conjunction with our algorithm. Other kinds of features could also benefit from the same split method; for instance, geodetic coordinates periodic splits could lead to a better understanding of geographic features.

Decision trees have a very interesting property when it comes to creating models that will interact with humans : when kept under a certain size, they are explainable, re-



(a) Data from c



(b) Data from s

Figure 4: 3 month of sleep data for users c and s

Datetime	State	Day of week	Time of day
2017-02-01 00:00:00	ASLEEP	2.0	0.000000
2017-02-01 00:10:00	ASLEEP	2.0	0.166667
2017-02-01 00:20:00	ASLEEP	2.0	0.333333
⋮			
2017-03-01 22:10:00	AWAKE	2.0	22.166667

Table 1: Sample of the sleep dataset

maining *"white boxes"*. We have proven that the periodic split method we presented exhibits better results than the traditional methods for a constrained tree size, hence a net gain in explain ability. Furthermore, this new algorithm results in decision tree reasoning on time which is different from time boundaries and ranges. This representation is not only more compact but also more natural for humans to read, thus fostering the collaboration between humans and AI.

References

[1] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[2] Quinlan Quinlan Cs and J. R. Quinlan. Improved use of continuous attributes in c4.5. *Journal of Artificial Intelligence Research*, 4:77–90, 1996.

[3] FRANCE. Conseil d'Etat. *Le numérique et les droits fondamentaux*. Etudes et documents, Conseil d'Etat. La Documentation française, 2014.

[4] Usama M. Fayyad and Keki B. Irani. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, 8(1):87–102, 1992.

[5] Jean Gaudart, Nathalie Graffeo, Guillaume Barbet, Stanilas Rebaudet, Nadine Dessay, Ogobara K Doumbo, and Roch Giorgi. SPODT: An R Package to Perform Spatial Partitioning. *Journal of Statistical Software*, 63(16), January 2015.

[6] J. Kmenta. *Elements of Econometrics*, pages 298––334. Maxwell Macmillan international editions. Macmillan Publ, 1986.

[7] Krzysztof Koperski, Jiawei Hah, and Nebojsa Stefanovic. An efficient two-step method for classification of spatial data. In *Symposium on Spatial Data Handling (SDH '98)*, pages 45–54, Vancouver, Canada, 1998.

[8] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[9] Juho Rousu and Tapio Elomaa. Efficient multisplitting revisited: Optima-preserving elimination of partition candidates. *Data Mining and Knowledge Discovery*, 8(2):97–126, 2004.

[10] Suzuki Einoshin Yokoi Hideto Takabayashi Katsuhiko Yamada, Yuu. Decision-tree induction from

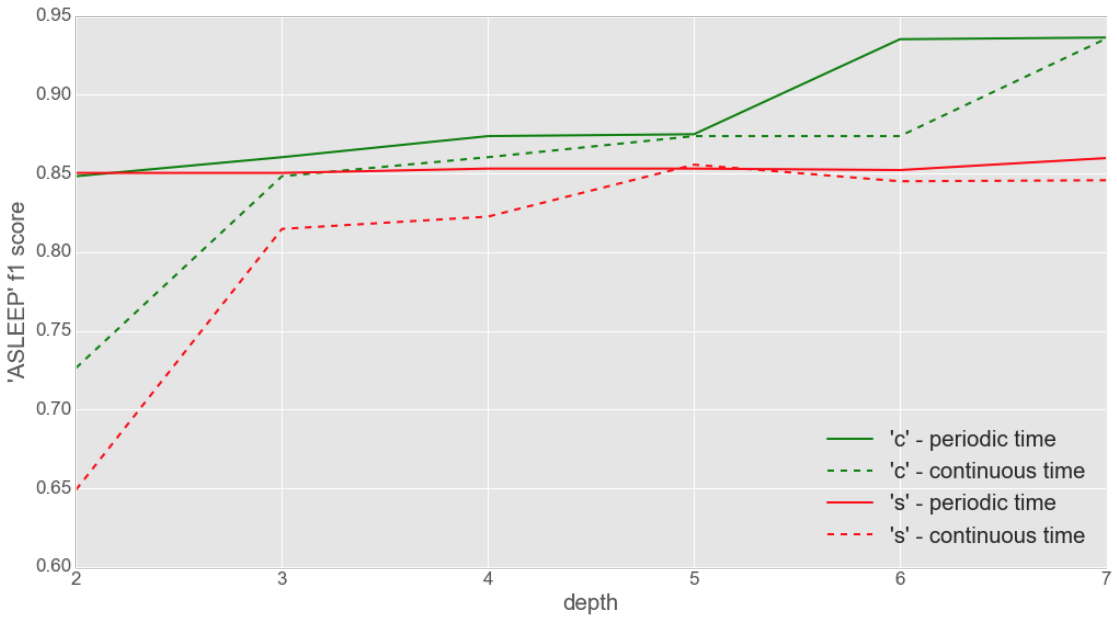


Figure 5: Comparison between periodic and classical splits

time-series data based on a standard-example split test. In *Twentieth International Conference on Machine Learning (ICML)*, page 840–847, 2003.