Budget learning based on equivalent trees and genetic algorithm : application to fall detection algorithm embedding

Mounir Atiq, Sergio Peignier, Mathilde Mougeot

Budget learning based on equivalent trees and genetic algorithm : application to fall detection algorithm embedding.

Mounir Atiq[1], Sergio Peignier[2], and Mathilde Mougeot[1]

[1]Centre Borelli, ENS Paris-Saclay, Gif, France
[2]Univ Lyon, INSA Lyon, INRAE, BF2I, UMR0203, F-69621 Villeurbanne, France

Abstract

Budget learning is a research field of growing interest that aims at including real world resource constraints into the design of machine learning models, mainly to reduce real environment prediction time. One common way of doing it is by modifying a pre-trained machine learning model, to fit the prediction time constraints while keeping as best as possible the model's prediction quality. However, in this case, the performance of these kinds of methods depends on the pre-trained model structure. To overcome this dependence, we propose to tackle the budgeted optimization problem, by using equivalent models with different structures and therefore different computation costs. The contribution of this work is to propose a genetic algorithm to decrease prediction time of random forest classifiers, by using equivalent decision trees. The first step of our method consists in building, from a pre-trained random forest, an initial population of random forests, that share the same decision function but have different structures. Then a genome reduction operation, is iteratively applied on the individuals via pruning based mutations.

Our experiments show an important impact of using equivalent decision trees on reachable random forest solutions with a budgeted prediction time. Results obtained on a synthetic data made of gaussian-shaped clusters and on a real industrial fall detection dataset, advocate for the use of equivalent random forest models in budget learning.

Keywords: Budget learning, equivalent decision trees, random forests, prediction time cost, genetic algorithm

1 Introduction

1.1 Budget learning

One major success of machine learning is its today applicability in a wide-range of industrial systems in numerous fields like IoT, security, healthcare or finance. In this context, the prediction model complexity is often adjusted through hyper-parameter tuning in the learning process, without taking deeply into account the capacity of

the device intended to run the model. Indeed several variables, involving real resource consumption necessary for building a model and/or making it work in a limited device, are not directly included in classical machine learning, whereas these real world applications need resources both to be trained and to stay functioning. These resources are often subject to practical constraints and taking them into account during the learning and/or prediction phase is the purpose of budget learning. Thus, budget learning is a vast and real concern for modern ground applications of machine learning.

Depending on the applications, resources can be critical either in the *training* or the *prediction* phase [1]. For instance, Active Learning methods focus on budgets defined on training instances, whereas other approaches consider budgeted computation time of prediction. Even both can be considered at the same time, for instance, this is the case for budgeted reinforcement learning [2]. In our case, we assume that the training is done under unrestricted conditions, with hypothetically unlimited resources. Conversely, as our model is intended to be embedded in a small electronic device, the cpu-time consumption is extremely constrained in the prediction phase occurring in real conditions.

Several recent works on budget learning on decision trees show the increasing interest in finding new ways to deal with real-world resource constraints in the context of models based on decision trees. Some methods incorporate directly budget constraints in a tree building algorithm to learn from scratch a budget sensitive model [3] [4]. Others are designed to alter a pre-trained decision trees based model to make it fit some budget constraints [5]. Among all known methods for updating a pre-trained tree-based model conditionally to a budget, one vast family is based on genetic programming methods [6].

In this work we present a new approach to modify a random forest, previously trained on an experimental setup with almost no constraints, to fit a prediction-time budget, defined by the capacities of an embedded system. It consists of a genetic algorithm that explores the space of pruned predictors from an initial population composed by random forests. This genetic exploration is based on a fitness function that is a trade-off between accuracy and the prediction time, and it is initialized by a randomized equivalent tree procedure.

1.2 Budget learning on decision trees

Budget learning aims at considering resource needs while applying a machine learning method. At the same time, the complexity of machine learning models is growing, as the scale of their application in real world, and the hard-wares they are implemented in are more and more portable and small [7, 8]. Thus, several recent works involving this topics on various kinds of models and considering different resource constraint frameworks. Some of them refer directly to industrial applications [9, 10], whereas others focus more on the theoretical framework of budgeted optimization [11,12]. One major categorisation among budget learning situations, as presented by the authors in [1,2], is whether the resource costs appear in the training or the prediction phase of a machine learning model. In this paper, we focus on the prediction time of random forests models.

One common approach while dealing with prediction time costs on decision tree based models is to include a penalisation to these costs directly in the greedy building operations of decision trees. This can be done by adapting boosting trees methods, as the Greedy-Miser algorithm [13], that adds sequentially a new decision tree in a step-wise regression way at each step of the algorithm. Each new decision tree is built using modified version of CART with a budget sensible purity function which takes into account both empirical error and feature usage with a regularization parameter λ. In [14], authors present several boosting algorithms on decision trees for budget learning. Another local approach is proposed in [3], for inducing budget sensitive

ensemble of decision trees. Authors define a family of budget sensitive purity functions (class of admissible impurity functions) that can be used to greedily construct decision trees.

In [5], the same authors propose an alternative algorithm with a more global optimization approach. The budget learning algorithm starts from random forest of pre-trained decision trees and then prunes them under budget constraints. They propose to solve a linear program corresponding to the best pruning combination on decision trees of a random forest, relatively to a dataset and feature acquisition costs, according to the trade-off between prediction accuracy and prediction computation time.

1.3 Genetic algorithms on decision trees

Among global optimization for budget learning in decision trees, genetic algorithms are widely used as non-greedy alternatives to deal with large combinatorial and multi-objective optimizations. They are already largely used in feature subset selection [15, 16], which is very linked to budget learning. For instance, genetic algorithms are introduced to explore the extremely high-dimensional discrete space of all possible feature subsets and retrain successively decision tree models over the selected subset until achieving a fitness or maximum iteration criterion in [17] and [18]. While exploring decision tree models space through genetic algorithms, some inherent drawbacks of local an greedy induction can be avoided [6] as well as keeping the ability of decision tree models to extract compact and relevant set of decision rules [19, 20]. Moreover, as shown in [21] tree structure is well-suited for defining intuitive cross-over and mutation operators and is compatible with string encoding which allows to use standard genetic programming methods. Most of existing genetic algorithms on decision trees use a standard cross-over consisting in the swapping sub-trees between two decision tree individuals [22]. In this work, as described in Section 3, individuals are random forests and the cross-over is based on swapping full decision trees between two random forests. Considering mutation, existing works on genetic algorithm for decision trees propose different mutation operations, depending on the optimization problem they consider, and the correspondent explored space of models. All of these budget learning methods on decision tree based models aim at minimizing a prediction error while keeping in a limited range of prediction time cost, but they mainly differ on the exact cost function they consider and the way this function is included during the optimization.

The work presented in this paper is organised as follows: Section 2 describes the usual framework for modelling prediction time on random forests and the constrained optimization we are dealing with. Section 3 explains the different genetic operations used in our algorithm and describes how they are assembled to give an iterative method for reduction of the prediction computation time. In particular Section 3.2 presents the notion of equivalent decision trees which is used to initialize the genetic algorithm. The experimental setup and results obtained on simulated and real industrial data are presented in Sections 4 and 5.

2 Budgeted prediction time using random forests

This Section defines prediction time costs for random forests and the associated constrained minimization problem considered in our algorithm.

2.1 Prediction time costs for decision tree based models

There are two kinds of computations that contribute to the prediction time of a decision tree or a random forest : the **evaluation cost** and the **feature acquisition cost**.

For a given sample x, the feature acquisition cost corresponds to the time spent to compute needed features for the prediction of x, and the evaluation cost corresponds to the time spent to compute internal nodes for the prediction of x (once the feature have been computed).

We deal here with a pre-trained random forest model called $\mathcal{M}$, a training set of n labelled data point couples $S = (X, Y) = \left((x^{(1)}, y^{(1)}), ...(x^{(n)}, y^{(n)})\right)$ with a feature space of dimension d. We assume that each feature f_j (with $1 \leq j \leq d$) has an individual *acquisition cost* c_j and a *evaluation cost* for each internal node k_{ev}, which is constant for all node splits.

For a given decision tree, any sample x is associated with a precise path leading to a leaf value which is the prediction of x by this tree. Then, we can define a *feature usage* vector $\Phi[x] \in \mathbb{N}^d$ corresponding to how many times each feature is used in this path. Then, the length of this path in terms of number of internal nodes is the sum of this vector's elements $\|\Phi[x]\|_1$. By taking the element-wise indicator function on this vector, we define the binary vector $\phi^{bin}[x]_j = 1_{]0,+\infty[}[\Phi[x]_j]$ that indicates for each feature whether or not it is needed in the path of x ($\phi^{bin}[x]_j = 1$ if the feature f_j is needed and 0 if not).

Feature acquisition cost: It corresponds to the computation time spent to compute features needed in the path of x indicated by the ones of the vector $\phi^{bin}[x]$. Then the feature acquisition cost is defined as $\sum_{j=1}^{d} c_j \phi^{bin}[x]_j$.

Evaluation cost: It corresponds to the computation time spent, having already computed the needed features, to assess all the node on the path of x. Then the evaluation cost is proportional to the length of this path and is modelled by : $k_{ev}\|\Phi[x]\|_1$.

For a random forest, if we consider the sum of all the different paths of x in the trees of the forest, we can define in the same way vectors Φ and ϕ^{bin} describing the feature usage for the random forest prediction of x. Then, for a given sample x, the total prediction time cost of a decision tree is defined as :

$$C(\mathcal{M}, x) = \sum_{j=1}^{d} c_j \phi^{bin}_{\mathcal{M}}[x]_j + k_{ev}\|\Phi_{\mathcal{M}}[x]\|_1$$

2.2 Budgeted optimization

In this work, the budget learning task is formalized as a constrained minimization problem. Indeed, we aim at minimizing a mean empirical loss relatively to a loss function (as in standard machine learning problems), but in a constrained space of models that have a prediction time bounded by a budget $B > 0$. More formally, let $\mathcal{H}$ be the space of models based on decision tree classifiers (e.g., random forests). To each model $\mathcal{M} \in \mathcal{H}$ corresponds a decision function $h_{\mathcal{M}}(x)$ and a prediction time cost $C(\mathcal{M}, x)$ for every sample x. Moreover, let $\ell(h_{\mathcal{M}}(x), y)$ be the loss function measuring the error between the prediction $h_{\mathcal{M}}(x)$ and the true label y.

The *budgeted prediction time* problem is to find a model $\hat{\mathcal{M}}$ such that:

$$\hat{\mathcal{M}} = \underset{\mathcal{M} \in \mathcal{H}}{\operatorname{argmin}} \ \frac{1}{n} \sum_{i=1}^{n} \ell(h_{\mathcal{M}}(x^{(i)}), y^{(i)}) \tag{1}$$

$$\text{subject to } \ \frac{1}{n} \sum_{i=1}^{n} C(\mathcal{M}, x^{(i)}) \leq B \tag{2}$$

3 Genetic pruning algorithm

This Section presents the genetic representation of random forests and the genetic operations used in our algorithm and defines the fitness function on random forest individuals, which is the objective function used to perform the budgeted minimization presented in 2.2 through individual selection. The notion of equivalence between decision trees is developed as well as how and why it is used to initialize the population of our genetic algorithm. As possible splits on which are built random forest individuals is a finite set, the space of random forest explored by our algorithm is also finite, but drastically large. Moreover, optimal equivalent decision trees considering model's size are proved to be NP-hard to find [21, 23] and genetic algorithms are a common way of dealing with such optimization problem [24].

3.1 Genetic representation of random forests

We propose a genetic representation of random forests where each decision tree is treated as a separate gene with two parts which correspond to the ordered sequences of internal nodes and leaves values. Internal nodes are split sequences (feature/threshold couples) and are the part of the genome subject to mutations. These splits are fixed by the initial pre-trained random forest and the genetic algorithm re-orders them in different decision tree structures with randomized equivalent tree procedure. On the other hand, leaf values are label sequences and are updated following the training data of the budget algorithm, their values being regulated by the environment.

After the population initialization, each new generation is obtained using the following successive genetic operations : the cross-over reproduction, to increase population diversity and fasten convergence towards a best individual; the pruning based mutation, to explore random forests with lower prediction time costs; and population reduction, to keep a reasonable population size and to increase the mean overall fitness of the whole population. As explained in 3.5 a selection and guided mutations are employed to optimize a *fitness function* representing the quality of random forests both in terms of accuracy and prediction time cost.

3.2 Initialization with randomized equivalent trees

Equivalent decision trees are classifiers that differ in their structure although they share the same decision function, Figure 3.2 shows an example of two equivalent decision trees. In 2, we explained that the prediction time cost of a random forest is importantly linked to the structure of the trees, as for a given sample x, the cost depends on its paths on the random forest trees. This means that two equivalent decision trees can have different prediction time costs.

Definition 1: Two decision trees T_1 and T_2 are *equivalent* if their decision functions are equal on all the feature space $\mathcal{X}$. $\forall x \in \mathcal{X}, h_{T_1}(x) = h_{T_2}(x)$.

Consequently, any budgeted prediction time algorithm applied on a random forest might lead to other solutions while considering an equivalent random forest composed by equivalent decision trees. Our motivation is that exploring the space of equivalent trees from an initial random forest can allow to find other solutions to the budgeted prediction time problem defined in 2.2. To illustrate it, we initialize our genetic algorithm with a randomized equivalent trees procedure to give structural variety to the initial population of random forest individuals. This procedure builds, for each tree of the pre-trained random forest, a randomized equivalent tree using the same splits but drawing them randomly in a top-down manner until getting one-class leaves according to the initial tree. A pseudo-code of this procedure is given in Algorithm 1.

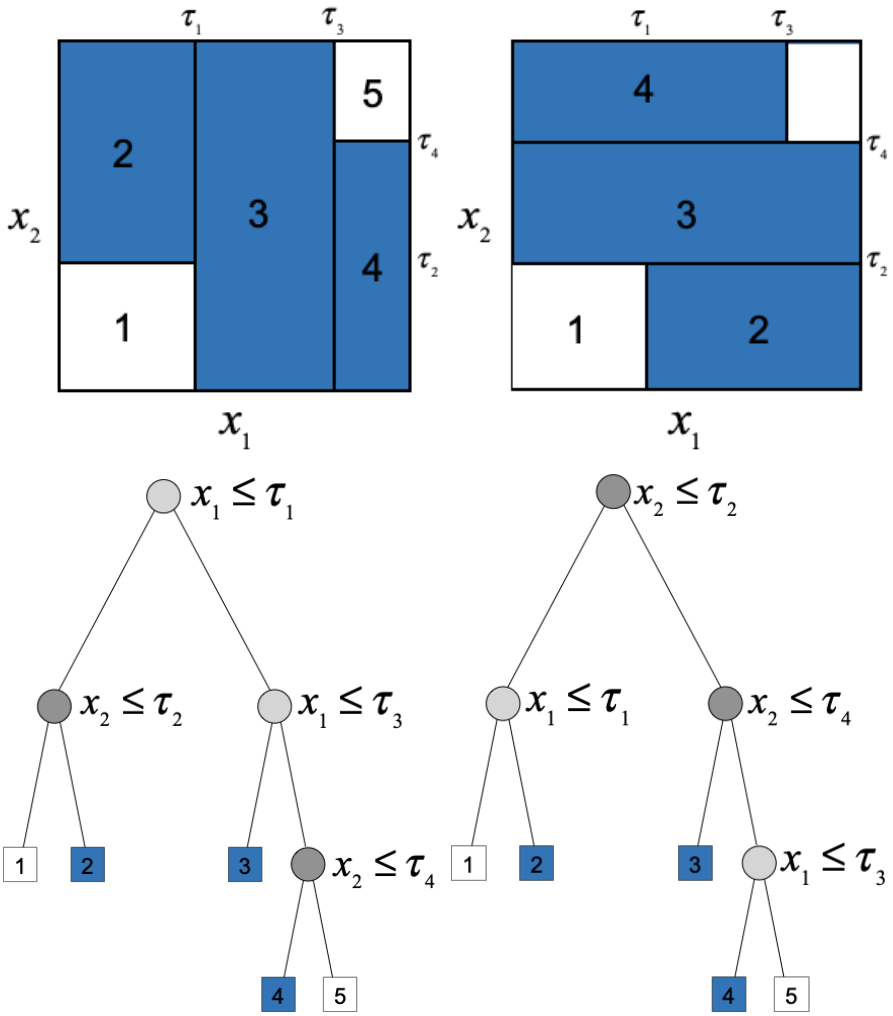


Fig. 1: Different partitioning for the same classification (a) and corresponding equivalent trees (b).

3.3 Genome cross-over between random forests

The cross-over operation is a classic method of reproduction with variation. This operation corresponds to the exchange of sub-parts of the parental genomes after at a random location known as the cross-over point. For instance, one common way of doing a cross-over between decision trees is by exchanging sub-trees after having selected cross-over nodes in the two parents. In this work, the cross-over operation aims to exchange decision trees between two random forest parents. Accordingly to the previously presented random forests genetic representation, it is in other words a restricted locations cross-over that can happen only at genes start/end points. Then, over the genetic algorithm iterations, each gene of the random forest individuals has a corresponding ancestor gene corresponding to an initial decision tree or one of its equivalents (See 3.2). Indeed, when dealing with numerical an not quantitative feature like our case, exchanging sub-trees that correspond to non-overlapping regions of the

Algorithm 1 Randomized equivalent tree

procedure $RandEqRec(T, T_{eq}, path)$
  if $path = null$ **then**
    $T_{eq} \leftarrow NewTree()$
  $\phi, \tau \leftarrow CohSplits(T, path)$
  $C = h_T(path)$
  if $|C| > 1$ **then**
    // π : splits random drawing policy
    $\phi_k, \tau_k \leftarrow ChooseNewSplit(\phi, \tau, \pi)$
    $T_{eq} \leftarrow NewNode(\phi_k, \tau_k, \pi)$
    $path_l, path_r \leftarrow Childs(T_{eq}, path)$
    $T_{eq} \leftarrow RandEqRec(T, T_{eq}, path_l)$
    $T_{eq} \leftarrow RandEqRec(T, T_{eq}, path_r)$
  else
    $C = \{c_i\}$
    $T_{eq} \leftarrow NewLeaf(c_i, T_{eq}, path)$
return T_{eq}

feature space can end up creating unreachable leaves, if it is made completely randomly. Secondly, we want to keep track of the initial random forest model through genetic operations in order to be able, for data analysis purposes, to read the final model as a simplified version of the initial one. Concerning genetic algorithm using decision trees, encoding trees into strings is possible before applying genetic operations but it is not necessarily the case as decision trees already have suitable structure for that [6]. For practical implementations, we do not use this string encoding in our work but it is feasible, as long as cross-over is restricted to certain locations and genome reduction mutation still keeps strings structure that can be reversely decoded into decision trees.

3.4 Mutations using random pruning

The second main genetic operation is the mutation and corresponds to the random pruning of random forests individuals. Regarding the prediction time of a random forest, pruning operation ensures a reduction of this cost. According to cost definition in Section 2, the prediction time reduction obtained by pruning a decision tree at a given node depends on how much feature usage this pruning avoids and is weighted by the proportion of training samples reaching this node. So each time a random forest individual is selected for mutation a pruning occurs randomly in each of its decision tree, creating a new leaf of the dominant label among training data. To ensure the mutation process to be progressive over iterations, pruning nodes are drawn using an exponential policy relatively to their depth in order to encourage small pruning in terms of deleted sub-trees. As we consider a decision tree as a gene in our representation, this random pruning mutation is a genome reduction mechanism occurring on each gene. Before mutation, concerned individuals are replicated in order to avoid loosing them in the situation where mutation would degrade their fitness value.

3.5 Selection, guided mutation and fitness function

To deal with our budgeted prediction time cost optimization we choose, as the ranking value of random forest individuals, the trade-off between *mean prediction error* and

Algorithm 2 Genetic pruning algorithm

procedure $GenBudPr(\mathcal{R}_f{}^{(0)}, X, Y, N_0, i_M)$
  $pop_0 = RandEqTree(\mathcal{R}_f{}^{(0)}, N_0)$
  for $i = 0 :: i_M$ **do**
    $rep \leftarrow Select(pop_i, \beta_r, X, Y)$
    $pop_i \leftarrow Rep(pop_i, rep, \tau_r)$
    $mut \leftarrow Select(pop_i, \beta_m, X, Y)$
    $pop_i \leftarrow Mut(pop_i, mut)$
    $del \leftarrow Select(pop_i, \beta_d, X, Y)$
    $pop_{i+1} \leftarrow Eliminate(pop_i, del)$
    $\hat{\mathcal{R}}_f \leftarrow$ Min. $fitness(\mathcal{R}_f, X, Y, \lambda)$
    on pop_{i+1}
  return $\hat{\mathcal{R}}_f$
  // Best individual of pop_{i_M} according
  to the fitness function.

the *mean prediction time cost* of a random forest with parameter λ :

$$V_{fit}(\mathcal{M}) = \frac{1}{n}\sum_{i=1}^{n}\left(\ell(h_{\mathcal{M}}(x^{(i)}), y^{(i)}) + \lambda C(\mathcal{M}, x^{(i)})\right)$$

In this work, the trade-off function defines the *fitness* of the individuals. Depending on their fitness, organisms undergo a selection step, inspired on the natural selection phenomenon. This operation chooses the individuals with the best fitness (here, lowest $V_{fit}(\mathcal{M})$) to reproduce, and it eliminates the worst individuals (highest $V_{fit}(\mathcal{M})$).

In practice, we used a deterministic selection scheme that picks the β_r top ratio of individuals for reproduction, and the β_d bottom ratio for elimination. Hence, parameters β_r and β_d, correspond to reproduction and death rates respectively. Moreover, we decided to guide the application of pruning mutation operations, using an exponential policy which is also a function of the fitness value. Then, worst random forest individuals are more likely to be pruned by mutation but creating mutants from best ones is still possible although less likely. This choice is based on the intuition that best individuals should be modified less often in order to preserve them for exploitation, while worst individuals should be modified more often, for exploration purposes.

4 Experimental setup

This Section describes the two data sets used to experiment our genetic budget learning algorithm, its parameter tuning and the methodology applied to compare and assess the results.

4.1 Synthetic data

Synthetic data used in experiments are generated from binary labeled gaussian clusters. The gaussian distributions are composed by 10 clusters of each label with mean μ and variance σ^2 parameters drawn randomly between bounded values (respectively $[-1, 1]$ and $[0.5, 1.5]$). Samples are generated with $d = 20$ features with 5 to 10 of them that are randomly chosen to be informative (the rest being white noise only).

One advantage of these synthetic gaussian controlled scenarios is that it is possible to generate as much data as needed for the evaluation process, which ensures performance measures as precise as desired. A Python implementation of the generator is available online [25].

4.2 Fall detection data

Our real ground application is a fall detection model that has to be embedded in a small device with limited cpu resources of 256kB ROM, 16kB RAM and 40 MIPS. We dispose of a dataset composed by 742 signals of falls and non-fall events, sampled at $f_s = 100Hz$, and recorded with 28 volunteers aged from 25 to 45 years old. A wide range of 87 features have been previously computed on these data from parts of the signal, its derivative and Fourier transform.

Ideally, we want the embedded system to give a prediction on the current signal every 10 ms, but all these features can not be computed with the device resources in this short time budget. Features costs c_i have been provided by the experts, as well as the evaluation cost k_{ev} of a unique decision tree node.

4.3 Experiments

To observe whether using equivalent decision trees can be relevant or not for budget learning purpose, each experiment is done on two separate population starting from the same initial random forest. The first population is obtained with equivalent trees, as described in Section 3.2, whereas the second one is obtained by exact duplication of the initial random forest. For each experiment, datasets are split into three subsets : the initial random forest training set, the training set used by genetic algorithm and a test set to assess final best random forest individual after all iterations. The main performance measure considered is the trade-off between mean accuracy and mean prediction time cost represented by the fitness function. Other interesting measures like mean error, depth, model size (in terms of number of nodes) or proportion of total feature acquisition cost used in the random forest are also presented in the next Section.

Population size at the start of the genetic algorithm is 40 random forests obtained from the initially pre-trained model (either by randomized equivalent trees procedure or duplication). At each iteration, new individuals are added to the population through mutation and reproduction whereas the less valuable ones according to the fitness function are removed. Mutation rate and reproduction rate are respectively set to $\beta_m = 0.6$ and $\beta_r = 0.25$. Birth rate and mortality rate are set to $\tau_r = 3$ and $\beta_d = 0.6$. Thus, the population size variation from a generation to the next one is given by the factor : $(1 + \beta_m + \beta_r * \tau_r)(1 - \beta_d)$ which is 1.02 in our experiments, meaning the population is slowly growing over iterations. As keeping the model's accuracy as best as possible is a crucial aspect of the considered industrial fall detection application, the trade-off parameter is fixed relatively low to $\lambda = 0.3$ for fall detection data and $\lambda = 0.6$ for synthetic data. Python implementation of algorithms 1 and 2 are available online [26, 27].

5 Results

This Section presents the results obtained with the aforementioned experiments, and our interpretations. Figures 2, 3, 4 and 5 illustrate the algorithm's behavior on the best random forest individual during iterations on datasets used for the budgeted training whereas Table 1 shows final best individual results assessed on the test datasets (to simplify the display of these results, we refer to the prediction time cost as "Budget" in Figure 3 and Table 1).

Figure 2 shows that, using parameter values given in previous Section, the genetic algorithm converges on all our experiments after a few iterations towards a best random forest individual regarding the fitness function value. This illustrates that using pruning mutation, in combination with tree exchanges based cross-over, is a valid method to optimize the trade-off between accuracy and prediction time cost of a

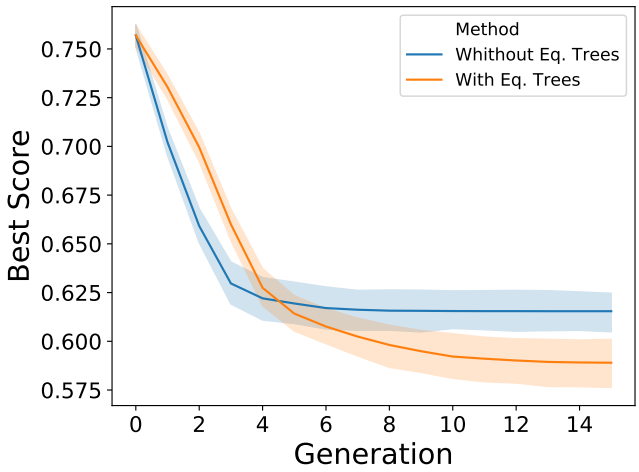
random forest. As we can observe both on synthetic data and fall detection data, the fitness value decrease in Figure 2 is very linked to the prediction time cost reduction (between 35% and 45% for the synthetic data and between 50% and 80% for the fall detection data) represented in Figure 3, which is the main purpose of the algorithm. Depending on the inherent complexity of data distribution regarding decision trees partitioning and the value of λ, this prediction time reduction can lead to a variable loss in accuracy, as represented in Figure 4.

Another aspect of these results is the impact of using equivalent decision trees. Indeed Figures 2 and 3 show that the genetic algorithm seems to reach better values of fitness function and prediction time with the population evolved from randomized equivalent trees initialization. This phenomenon is very pronounced on fall detection data and also noticeable in synthetic data regarding confidence intervals. Moreover this difference in prediction time reduction using equivalent trees is especially interesting insofar as it does not necessarily reflect an accuracy difference as shown in Figure 4. Overall, all these figures reveal that using equivalent trees with this genetic algorithm improves the budget learning task on our data but it takes more iterations to converge. This behavior is confirmed in Figure 5 with the best individual mean depth over iterations, which suggests in early iterations with Figure 3 that lower prediction time individuals are obtained with equivalent trees while random forests are deeper compared to the population initialized by duplication. These observations are consistent with the intuition that space of potential accuracy/prediction time couples of reachable solutions is highly wider by using equivalent trees because of the structural variety it induces.

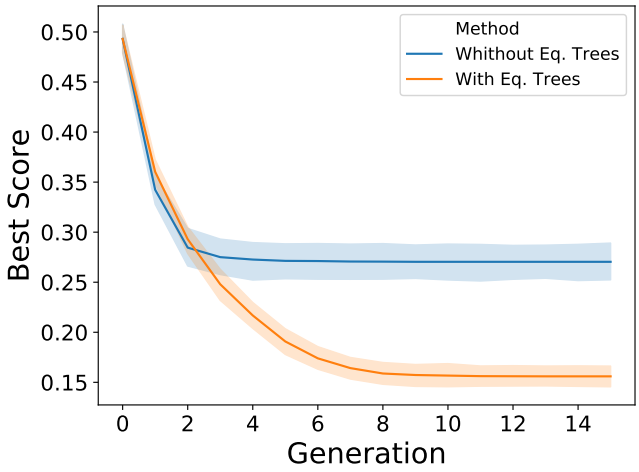
Concerning our embedded fall detection application, this genetic pruning algorithm allows broadly to divide by 2 the prediction time without randomized equivalent trees and to divide it by 4 while using them. From the 87 initial features, final best random forest individuals keep in mean around between 10 and 20 features representing in most cases less than 15 % of all features costs, and are also a lot lighter in terms of depth and number of nodes. To compare Table 1 presents the results obtained in these experiments in terms of accuracy and several prediction time aspects in front of the first version of random forest implemented in 2017 in our embedded systems for fall detection [7]. This model has been obtained by a first selection of a subset of features by embedded system experts based on features acquisition costs and features importance estimated on previous unconstrained model. Then a new random forest has been trained using this features subset. The pruning approach with equivalent trees presented in this work directly considers the mean prediction time budgeted by a given device to simultaneously reduce the feature acquisition cost (through features reordering and feature usage lowering) and the evaluation cost (through depth decrease).

Tab. 1: Budget-wise comparison of different random forest models for fall detection embedding.

	2017 version [7]	With Eq.	Without Eq.
Fitness value	0.31 ± 0.03	$\mathbf{0.20 \pm 0.04}$	0.29 ± 0.05
Budget	0.72 ± 0.08	$\mathbf{0.34 \pm 0.11}$	0.68 ± 0.16
Error	$\mathbf{0.09 \pm 0.01}$	$\mathbf{0.10 \pm 0.03}$	$\mathbf{0.09 \pm 0.02}$
Depth	5	$\mathbf{2.5 \pm 0.5}$	$\mathbf{2.5 \pm 0.4}$
N° nodes	93 ± 5	22 ± 7	21 ± 6
N° features	34	$\mathbf{16 \pm 3}$	$\mathbf{16 \pm 3}$
(%) total feature cost	25	$\mathbf{13 \pm 5}$	22 ± 6

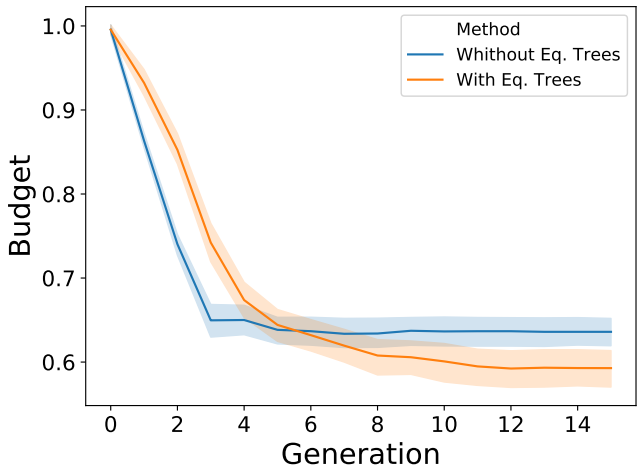


(a)

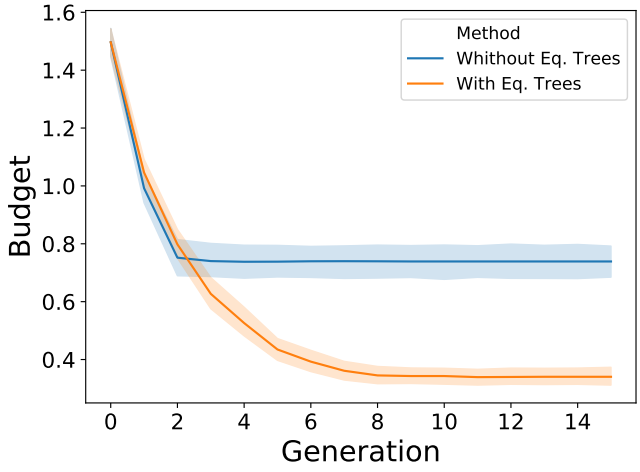


(b)

Fig. 2: Fitness value of the best random forest individual over iterations on synthetic (a) and fall detection datasets (b).

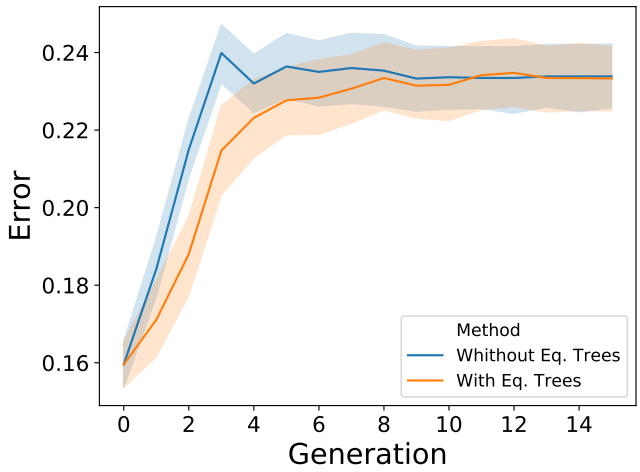


(a)

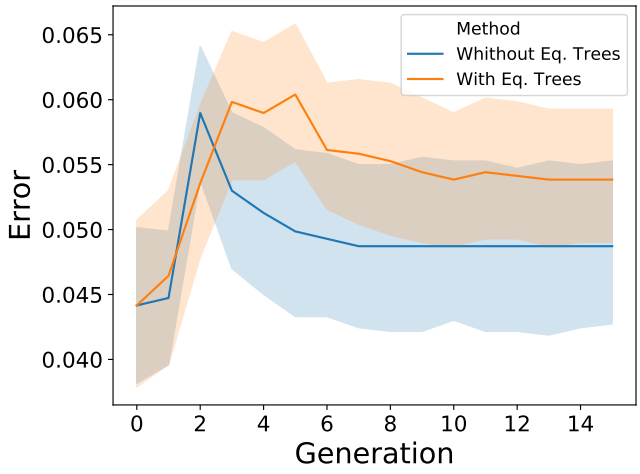


(b)

Fig. 3: Mean prediction time cost of the best random forest individual over iterations on synthetic (a) and fall detection datasets (b).

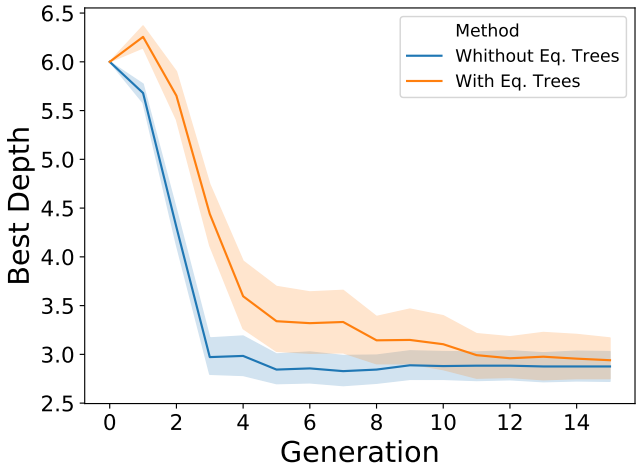


(a)

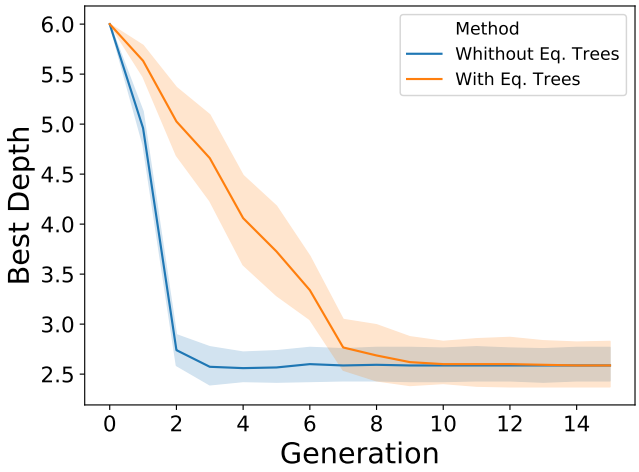


(b)

Fig. 4: Mean prediction error of the best random forest individual over iterations on synthetic (a) and fall detection datasets (b).



(a)



(b)

Fig. 5: Mean depth of the best random forest individual over iterations on synthetic (a) and fall detection datasets (b).

6 Conclusion and future work

This work tackles the problem of budgeted prediction time for random forest embedding in a small cpu capacity device. We manage to get, with our genetic pruning algorithm, low prediction time random forests in exchange of a very small accuracy decrease with shallow trees and using a restricted subset of features. Also, the final obtained classifier is still a pruned version of an equivalent of the initially pre-trained random forest model. This means that, for our fall detection application, the initial unconstrained model can be kept, if needed, offline for detailed interpretation of predictions, while the simplified version of this model is embedded for efficient real-time online predictions.

This work shows through equivalent decision trees how much the model's structure is impacting on the computation time and that same decision function does not mean same computation time. Using a simple randomized equivalent tree procedure, our results on both synthetic data and fall detection data illustrate the potential of using equivalent trees for this kind of structurally dependent machine learning problems. This procedure builds equivalent decision trees with a homogeneous drawing policy over given splits and does not take into account the knowledge of features costs. Thus, an interesting continuation of this work would be to cost sensitive splits drawing policies while generating equivalent decision trees.

Acknowledgment

Part of this work has been founded by the Industrial Data Analytics And Machine Learning chairs of ENS Paris-Saclay. Mounir Atiq is partially supported by Tarkett.

References

[1] N. Cesa-Bianchi, S. Shalev-shwartz, and O. Shamir, "Some Impossibility Results for Budgeted Learning," *ICML Workshop on Budgeted Learning*, 2010.

[2] G. Contardo, L. Denoyer, and T. Artières, "Sequential cost-sensitive feature acquisition," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9897 LNCS, pp. 284–294, 2016.

[3] F. Nan, J. Wang, and V. Saligrama, "Feature-budgeted random forest," *32nd International Conference on Machine Learning, ICML 2015*, vol. 3, pp. 1983–1991, 2015.

[4] M. Chen, Z. Xu, K. Q. Weinberger, O. Chapelle, and D. Kedem, "Classifier Cascade for Minimizing Feature Evaluation Cost," *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. XX, pp. 218–226, 2012.

[5] F. Nan, J. Wang, and V. Saligrama, "Pruning random forests for prediction on a budget," *Advances in Neural Information Processing Systems*, pp. 2342–2350, 2016. [Online]. Available: http://arxiv.org/abs/1606.05060

[6] R. C. Barros, M. P. Basgalupp, A. C. De Carvalho, and A. A. Freitas, "A survey of evolutionary algorithms for decision-tree induction," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 42, no. 3, pp. 291–312, 2012.

[7] L. Minvielle, M. Atiq, R. Serra, M. Mougeot, and N. Vayatis, "Fall Detection Using Smart Floor Sensor and Supervised Learning," *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS (2017) 3445-3448*, pp. 3–6.

[8] Z. Hussain, Q. Z. Sheng, and W. E. Zhang, "A review and categorization of techniques on device-free human activity recognition," *Journal of Network and Computer Applications*, vol. 167, no. May, p. 102738, 2020. [Online]. Available: https://doi.org/10.1016/j.jnca.2020.102738

[9] T. Veniat and L. Denoyer, "Learning Time/Memory-Efficient Deep Architectures with Budgeted Super Networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. [Online]. Available: http://arxiv.org/abs/1706.00046

[10] W. Luo, C. Nam, and K. Sycara, "Online decision making for stream-based robotic sampling via submodular optimization," *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, vol. 2017-November, pp. 118–123, 2017.

[11] T. Gao and D. Koller, "Active Classification based on Value of Classifier." *Nips*, pp. 1–9, 2011.

[12] Q. Gu, T. Zhang, C. Ding, and J. Han, "Selective labeling via error bound minimization," *Advances in Neural Information Processing Systems*, vol. 1, pp. 323–331, 2012.

[13] Z. E. Xu, K. Q. Weinberger, O. Chapelle, S. Louis, and O. C. Cc, "The Greedy Miser: Learning under Test-time Budgets," *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pp. 1175–1182, 2012.

[14] Z. E. Xu, M. J. Kusner, K. Q. Weinberger, and A. X. Zheng, "Gradient Regularized Budgeted Boosting," 2019. [Online]. Available: http://arxiv.org/abs/1901.04065

[15] S. B. Kotsiantis, "Feature selection for machine learning classification problems: A recent overview," *Artificial Intelligence Review*, vol. 42, no. 1, p. 157, 2014.

[16] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A Survey on Evolutionary Computation Approaches to Feature Selection," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 4, pp. 606–626, 2016.

[17] J. Bala, J. Huang, H. Vafaie, K. DeJong, and H. Wechsler, "Hybrid learning using genetic algorithms and decision trees for pattern classification," *International Joint Conference on Artificial Intelligence*, vol. 14, pp. 719–724, 1995.

[18] Yang, "Feature Subset Selection Using a Genetic Algorithm," 2013.

[19] "A Genetic Algorithm for Constructing Compact Binary Decision Trees," 2013.

[20] Z. Fu, "An Innovative GA-Based Decision Tree Classifier in Large Scale Data Mining," vol. 5, pp. 348–353, 1999.

[21] Q. Zhao and M. Shirasaka, "A study on evolutionary design of binary decision trees," *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999*, vol. 3, no. 2, pp. 1988–1993, 1999.

[22] D. Jankowski, K. Jackowski, D. Jankowski, K. Jackowski, E. Algorithm, T. Induction, D. Jankowski, and K. Jackowski, "Evolutionary Algorithm for Decision Tree Induction To cite this version : HAL Id : hal-01405549," 2016.

[23] L. Hyafil and R. L. Rivest, "Constructing optimal binary decision trees is NP-Complete," *Information processing letters*, vol. 5, 1976.

[24] "Using Genetic Algorithms to Solve NP-Complete Problems," *In:ICGA*, no. November, pp. 124–132, 1989.

[25] "Synthetic Data Generator," https://github.com/SergioPeignier/TLSyntheticDataGenerator, [Online].

[26] "Randomized equivalent decision trees," https://github.com/atiqm/equivalent_decisiion_trees, [Online].

[27] "Genetic pruning algorithm ," https://github.com/atiqm/budget_learning/Gen/, [Online].