DEPARTMENT OF ENGINEERING MANAGEMENT

Active Learning-based Pedagogical Rule Extraction

Enric Junqué de Fortuny & David Martens

UNIVERSITY OF ANTWERP
Faculty of Applied Economics

City Campus
Prinsstraat 13, B.226
B-2000 Antwerp
Tel. +32 (0)3 265 40 32
Fax +32 (0)3 265 47 99
www.uantwerpen.be

FACULTY OF APPLIED ECONOMICS

DEPARTMENT OF ENGINEERING MANAGEMENT

Active Learning-based Pedagogical Rule Extraction

Enric Junqué de Fortuny & David Martens

RESEARCH PAPER 2014-016
AUGUST 2014

University of Antwerp, City Campus, Prinsstraat 13, B-2000 Antwerp, Belgium
Research Administration – room B.226
phone: (32) 3 265 40 32
fax: (32) 3 265 47 99
e-mail: joeri.nys@uantwerpen.be

The research papers from the Faculty of Applied Economics are also available at www.repec.org (Research Papers in Economics - RePEc)

D/2014/1169/016

Active Learning-based Pedagogical Rule Extraction

Enric Junqué de Fortuny David Martens

Both authors are with the department of Applied Data Mining, Faculty of Applied Economics, University of Antwerp, Belgium

Abstract

Many of the state-of-the-art data mining techniques introduce non-linearities in their models to cope with complex data-relationships effectively. Although such techniques are consistently included among the top classification techniques in terms of predictive power, their lack of transparency renders them useless in any domain where comprehensibility is of importance. Rule-extraction algorithms remedy this by distilling comprehensible rulesets from complex models that explain how the classifications are made. The present article considers a new rule extraction technique, based on active learning. The technique generates artificial data points around training data with low confidence in the output score, after which these are labelled by the black-box model. The main novelty of the proposed method is that it uses a pedagogical approach without making any architectural assumptions of the underlying model. It can therefore be applied to any black-box technique. Furthermore, it can generate any rule format, depending on the chosen underlying rule induction technique. In a large-scale empirical study, we demonstrate the validity of our technique to extract trees and rules from Artificial Neural Networks, Support Vector Machines and Random Forests, on 25 datasets of varying size and dimensionality. Our results show that not only do the generated rules explain the black-box models well (thereby facilitating the acceptance of such models), the proposed algorithm also performs significantly better than traditional rule induction techniques in terms of accuracy as well as fidelity.

Index Terms

Rule Extraction, Active Learning, Comprehensibility, Pedagogical

I. Introduction

Data mining is a relatively young and interdisciplinary field of computer science that concerns itself with discovering new patterns from large datasets. Benchmarking studies reveal that non-linear techniques, such as Artificial Neural Networks (ANN) [1], Support Vector Machines (SVM) [2] and ensemble methods [3] perform consistently well in terms of predictive accuracy [4], [5], [6]. Their ability to capture non-linearities is simultaneously their greatest weakness, as the generated predictive models are often

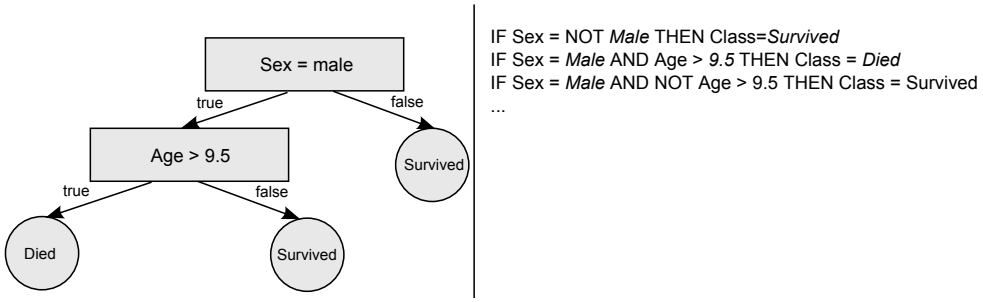


Fig. 1 : Both decision tree algorithms (left) as well as a ruleset-generators (right) generate comprehensible models that allow one to reason about the data. This example comes from the famous Titanic example and illustrates the “Women and children first” principle.

incomprehensible to a human interpreter. Comprehensibility is required in any domain where the model needs to be validated before it can be used in practice, such as medical diagnosis [7] or audit mining [8]. In credit scoring this requirement is even a legal one [9], where financial institutions need to be able to explain to any rejected loan applicant why credit has been denied. The Basel III capital accord has similar requirements with regards to the models for internal capital requirement calculations [10]. Furthermore, prior work suggests that when users do not understand the inner workings of a decision making system, they will be sceptical and reluctant to use the model, even if the model is known to improve the decision performance, see e.g., [11], [12], [13], [14], [15]. Although the importance of comprehensibility has been established decades ago [16], current data mining research seems focused on predictive accuracy only. Rule extraction techniques have been proposed as a way to generate predictive rules that mimic the classifications made by the black-box technique [17], [18], [19], and take an important role in data mining, which was originally defined as ”the non-trivial process of identifying valid, novel , potentially useful, and *ultimately understandable* patterns in data.” [20].

As shown in Figure 1, the generated rule sets from such a model provide insights

into the logics underlying the black-box model in a human-readable form. The extent to which the extracted rules explain the black-box model is measured in terms of the percentage of test data that are classified the same by both the black-box model and the rules. If this so called fidelity metric is high enough, one can decide that enough insight into the black-box model is obtained and it can be used in practice. The rationale behind this is that, as the fidelity increases (measured on a test set), the decision boundaries of both models resemble each other more and more. Interestingly, previous research has shown that performing rule extraction can even lead to an improved test accuracy and comprehensibility, when compared to traditional rule induction techniques [18]. A result that will be confirmed in our empirical section.

The general structure of this paper is as follows: first we elaborate on the rationale behind rule extraction and discuss the important design factors and evaluation metrics that have to be taken into account (Section II). In Section III we cover the general rule extraction methodology in more detail after which we explain the data mining techniques used in the empirical study in Section IV. In Section V and VI we perform large-scale experiments to study if our method performs significantly well in different contexts. The results are then summarized in the concluding section, together with prospects into future developments of the reported technique.

II. Rule Extraction: Overview

A. Rationale behind rule extraction

For many applications, it is important to build classification models that are both very accurate and easily understood. Using traditional techniques, these requirements often work in a contradictory manner and either one must be sacrificed for the other or as Breiman stated in 2011 [21]: *"Unfortunately, in prediction, accuracy and simplicity (interpretability) are in conflict"*. For instance, using a complex non-linear SVM might yield very good performance, but it is uninterpretable in most realistic settings. Rule induction techniques such as Quinlan's C4.5 [22] construct very comprehensible models

but often comes at the cost of losing considerably on accuracy. Rule extraction is a technique that attempts to find a compromise between both requirements by building a simple rule set that mimics how the well performing complex model (*black-box*) makes its decisions. In the presented approach, rule sets are generated by a rule induction method, hereafter called the *white-box* technique.

There are two sub-scenarios in which rule extraction techniques are commonly used [18]. First of all, one might be interested in the logics or inner workings of a black-box model with strong predictive power. That is, we want to know the rationale behind the decisions made, and verify and whether its results make sense in practice. This can be useful in many safety-critical applications ranging from the operation of power plants and air traffic control to decision support and medical diagnosis [23]. In this case, the aim is to extract rules that mimic the black-box well, measured by the *fidelity*.

Another way in which rule-extraction can be used is to attempt to improve the performance of a rule induction technique. In this second scenario we are only concerned with improving the rule set *accuracy*, while maintaining comprehensibility (for instance by removing noise from the data). We will come back to this point in Section V-A. Although this manuscript focusses on global explanations for black-box models, one should note that also instance based explanations can be generated to explain the classification for a single data instance [24], [25], [26].

B. Overview of existing rule extraction techniques

A chronological overview of some rule extraction techniques and their translucency/rule expressiveness is given in Table I. A myriad of techniques have been proposed to extract rules from complex methods, each having different characteristics and outputs. Andrews et al [23] propose a taxonomy for rule extraction techniques according to five dimensions. Although these were presented for the special case of ANNs, four of these dimensions are applicable on a more general class of rule extraction algorithms as well.

The expressive power of the extracted rules: Many types of rules have been suggested in the literature. Propositional rules take the form of **If ... Then ...** expressions, where the clauses are defined in propositional logic (e.g. $x \geq 3$ *and* $y \leq 5$). Another type of rules is the M-of-N rule and differs in that the clause can take the form of *at least/exactly/at most M of the N conditions*. Although the first form is easier to understand, the second form allows the generation of richer rule sets. Breaking away from traditional logic, fuzzy rules allow partial truths instead of Boolean true/false outcomes. These allow

Algorithm	Ref.	Transl.	BB	Rule Format	Comment
Trepan (1996)	[17]	P	ANN	M-of-N splits tree	decision tree induction
ANN-DT (1999)	[27]	D	ANN	decision tree	induction
DecText (2000)	[28]	D	ANN	decision tree	induction
SVM+Prototypes (2002)	[29]	D	SVM	propositional rules	clustering using RBF
REFNE (2003)	[30]	D	ANN	propositional rules	breadth-first search with sampling
REX (2003)	[31]	P	ANN	fuzzy rules	genetic algorithm
Rabuñal (2004)	[32]	P	ANN	propositional rules	genetic programming
BUR (2004)	[33]	P	SVM	propositional rules	based on gradient boosting machines
Barakat (2005)	[34]	D	SVM	decision tree	train decision tree on support
Fung (2005)	[35]	D	SVM	propositional rules	only for linear classifiers
Re-RX (2006)	[36]	P	ANN	hierarchical rule sets	first splits are based on discrete attributes
Iter (2006)	[37]	P	SVM	propositional rules	iterative growing of hypercubes, regression
SQRex-SVM (2007)	[38]	D	SVM	propositional rules	sequential covering
ALBA (2008)	[18]	D	SVM	open	active learning with support vectors
Farquad (2010)	[39]	D	SVM	propositional rules	regression
Su (2011)	[40]	D	SVM	propositional rules	genetic algorithm based on support vectors
LORE (2011)	[41]	D	ANN	decision diagrams	eclectic
ALPA (2013)		P	open	open	present paper

TABLE I : Chronological Overview of Rule Extraction Algorithms with an indication of the translucency (pedagogical 'P' or decompositional 'D'), the black-box techniques it can be applied to and the rule format.

for more comprehensive rules that are still clear to understand by human experts since many of them bare close resemblance to linguistic concepts.

Translucency: The translucency refers to the relation between the extracted rules and the internal architecture of the underlying complex model. In *decompositional* rule extraction methods, the algorithm is intertwined with the workings of the complex model. These algorithms are usually specifically built for the complex method and not portable to other methods. For example, the ALBA [18] and SVM+Prototypes [29] techniques make explicit use of support vectors and can therefore only be used for SVM rule extraction, while DecText [28] and ANN-DT [27] can only be used for ANN rule extraction.

Pedagogical techniques "view rule extraction tasks as a learning task where the target concept is the function computed by the [complex model] and the input features are simply the [complex model] input features" [42].[1] An advantage of such techniques is the broad applicability to any black-box model.

The quality of the extracted rule: The quality of the extracted rules is usually measured in terms of accuracy, fidelity and comprehensibility. Accuracy is defined as the number of test data points correctly classified by the rules, divided by the total number of data points in the test set. Fidelity is a similar metric that uses the number of data points where the rule set and the complex model agree as the numerator, it is an indicator of

[1]Sometimes, techniques that use a combination of both approaches are said to follow the *eclectic* approach. We refer to these as being decompositional as well [23].

Algorithm	Approach	Explored Region	Rule format
TREPAN (1996)	P	Adaptive	M-of-N splits tree
ALBA (2008)	D	Fixed	open
ALPA (2012)	P	Adaptive	open

TABLE II: Comparison with TREPAN and ALBA.

how similar the extracted rule set and the complex model are. For more information on how to measure comprehensibility see Section II-C.

The algorithmic complexity: This dimensions refers to the universal requirement for the algorithm to be as efficient as possible. Some algorithms are only applicable to toy examples due to their complexity while others can perform well in terms of time resources on real datasets.

Two techniques deserve special attention because they contain very useful and crucial concepts for rule extraction that are also present in the proposed algorithm: Trepan and ALBA. Trepan [17] is a rule extraction technique initially designed by Craven and Shavlik for ANN that builds M-of-N decision trees. In order to handle the problem of the decreasing number of training observations as the depth in the tree increases, Trepan generates additional random data instances, which are labelled by the ANN model. Using this mechanism, a certain minimum number of observations is ensured on each level of the tree, leading to more fine-grained rule sets. Trepan is one of the first pedagogical approaches to rule extraction in the literature. Later this concept was further developed in ALBA [18] (Active Learning-Based Approach to SVM Rule Extraction) by Martens et al. Here, an active learning component generates additional artificial data points near the decision boundary, where most of the noise is present in the data. ALBA uses the support vectors as proxies for the decision boundaries, seen that these specific training points are typically located near the decision boundary. The main limitations of ALBA are that it is rather slow and the decompositional approach can only be applied to SVM models. We expand on the concept of generating extra data in specific regions of the input space, but confine these interesting regions even more. Furthermore, in order to avoid any generality problems, we propose a pedagogical algorithm that does not use specific architectural properties of the algorithm to define these regions. A comparison of the methodological similarities and differences between our method, Trepan and ALBA is displayed in Table II. As can be seen from Tables I and II-C, our proposed ALPA rule extraction technique is the first that is applicable to any black-box model[2] and has no limitations on the rule format.

[2]Beyond the approach of only changing the class labels of the training data to the black-box predicted labels [38].

C. Design and evaluation considerations

Besides the taxonomy provided by Andrews et al. [23] (discussed above), the rule extraction literature also provides us five important criteria for evaluating rule extraction methods [43].

a) Comprehensibility: is the extent to which extracted representations are humanly comprehensible. There are two design choices to consider when evaluating comprehensibility. A first aspect to be taken into account is the model complexity. A decision tree with more than one hundred branches is clearly not very comprehensible. Empirical research by Dejaeger et al [44] has shown that larger representations of rules result in a decrease in expert answer accuracy, an increase in answer time, and a decrease in confidence. A second aspect is the type of rule structure used. There are many rule types and structures and each of these has a different level of comprehensibility. Although cognitive science research is still ongoing, preliminary research [44] has shown that decision tables are the most comprehensible in general, but that the choice often depends on situational - and thus hard to control - aspects such as the expert's experience with a rule set structure.

b) Generality: The generality depends on the extent to which a method requires special training regimes or restrictions on the model architecture. The decompositional approach cannot be ported to other techniques without some prior modifications, making it very specific and limited in applicability. A pedagogical approach, in principle, require no internal knowledge or structure and can therefore be applied to other black-box methods as well. Even so, as can be seen from Table I (column BB) existing techniques in the literature are still custom tailored with a specific black-box technique in mind. One exception to this are metaheuristics-based approaches such as REX [31], GEX [45].

c) Scalability: Scalability is the ability of the method to scale to problem instances with large input spaces and large number of data both in comprehensibility and time.

d) Fidelity and accuracy: As discussed before, accuracy and fidelity are the two most important evaluation criteria for extracted rule sets.

e) Software availability: Software availability is the extent to which researchers make their models available to potential users.

III. Uncertainty based rule extraction

A. Proposed Rule Extraction Technique

The central idea on which ALPA relies is that in order to improve a rule set in terms of accuracy or fidelity, we should train a comprehensible model (the *white-box*) to mimic

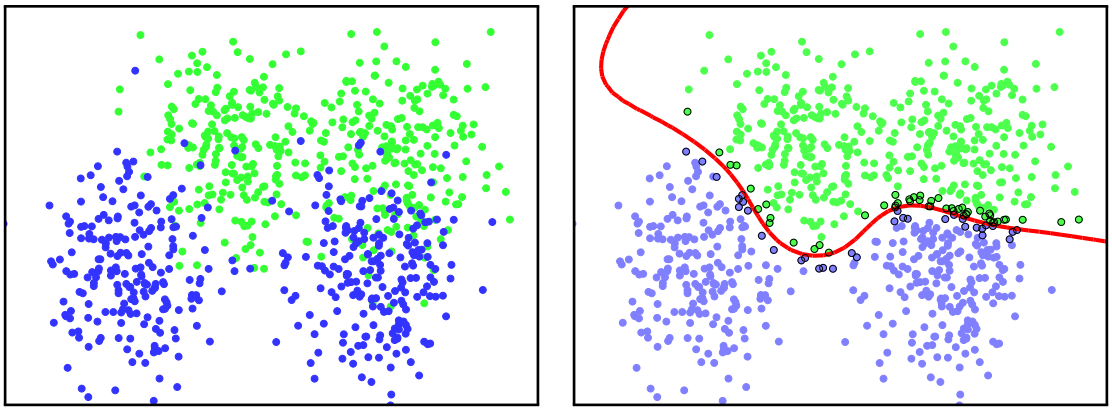
the output of a more complex[3] black-box model that performs better. Given a training set and a black-box technique, by presenting the *predicted* target values of the training set to the white-box algorithm instead of the *original* target values associated with the training set, we can improve the similarity between the black-box and the white-box substantially. As such, the black-box becomes an oracle for the white-box and we call the training data linked with the predictions of that black-box, the *oracle set* (as opposed to the *original set*). The similarity between the white-box and the black-box models is defined as the *fidelity* of the rule set. As the fidelity improves, the behaviour of the rule set given by the white-box models converges to that of the black-box. This relabling step can also be seen as a step to remove noise from the input, thus reducing ambiguity.

Active Learning As brought up in [17], we do not have to limit ourselves to the original data: since we are using an oracle, we can generate new artificial data points and their predictions (provided by the oracle) without restrictions. Further improvements of fidelity can be achieved by offering the white-box technique more data. This pedagogical approach is not a new idea and similar lines of reasoning have been used in other lines of research, including *optimal design of experiments* [46] and *active learning* [47]. The main concern that both of these fields deal with, is choosing which input vector adds the most information for further predictions. In our case this concern is translated into finding the region in which we should generate new training data vectors to achieve optimal fidelity under some constraints (e.g. limited number of rules).

Active learning recommends focussing on the problem areas and for rule extraction these are the areas where the noise is the highest [47]. The key observation used in our method, is that most of the dissimilarities between both models are found near the decision boundary, which marks the transition from one class to another [18]. Thus, one way to improve the model is to shift focus from the regions where the algorithm is very certain of its predictions and the black-box and white-box models concur largely in their predictions, to the boundary regions where there is more uncertainty. In order to do so, we must generate points in those (boundary) regions. There are a few practical problems that arise when trying to do so. First, not all boundary regions prove to be interesting (e.g. in parts of the input space where there are no nearby data points). Even worse, often no closed boundary is given by the black-box model or the formula is simply not known. In summary, we have to find some subregion of the boundary region where it is feasible and interesting to generate new data points.

Valley points A solution to both of these problems can be found in the observation that we know which data points of the original dataset lie near to the boundary: those with the lowest prediction confidence. Generating data points near these makes sense, since they are close to the decision boundary and are certain to lie in regions where some of the input data lies. In addition, dense regions near the decision boundary will

[3]Note that the complexity of a classification model is defined in terms of transparency to a human interpreter.



(a) Original Ripley dataset before processing.

(b) Oracle dataset: The SVM decision boundary is shown by a red curved line and all valley points are marked.

Fig. 2 : A toy example: the Ripley dataset was drawn from a mixture of overlapping Gaussian distributions.

have more data points with high uncertainty. As such, the distribution of the data is implicitly taken into account.

We first choose a subset of the training set, containing points that are near to the decision boundary. The points in this set are called *valley points*, because they lie near the decision boundary in the valley of the *confidence function* π of the black-box model[4]. That is, points nearby the decision boundary have low prediction confidence and are considered to be in a problem region (a point lying on the decision boundary has the lowest possible confidence since picking either class is equally likely). Consider the Ripley dataset displayed in Figure 2(a), which we will be using as a toy example to illustrate the algorithm [48]. This dataset stems from a binary classification problem, where input vectors were generated from a mixture of overlapping Gaussian distributions. As explained before, first a black-box model is trained (in this example, a Support Vector Machine (SVM), see Section IV-B) on the training set, followed by a relabeling of the training data so as to get the oracle set. Figure 2(b) displays the relabeled set along with the separating hyperplane given by the SVM and shows no more overlap is present in the data. We marked the top 15% most uncertain points with a black circle, these are the valley points for this particular dataset and black-box model. The number of valley points chosen depends on the black-box technique used, but our empirical analyses show that generally about 20 percent of the training set should be used.

Number of valley points: Two aspects are important when we generate extra data in the problem regions. Not only is the region in which we generate new points important, but also the number of extra data we generate. As mentioned before, new information

[4]These points serve a similar role as the support vectors in ALBA.

Algorithm 1 ALPA for N_v valley points and N training points.

procedure EXTRACT($blackbox$, $whitebox$, $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, N_v)
  // Build the oracle set.
  for $i = 1 \cdots N$ **do**
    $o_i \leftarrow \text{predict}(blackbox, \mathbf{x}_i)$
  end for
  $\mathcal{O}_{train} = \{(\mathbf{x}_i, o_i)\}_{i=1}^{i=2/3 \cdot N}$ ▷ Oracle training data.
  $\mathcal{O}_{val} = \{(\mathbf{x}_i, o_i)\}_{i=2/3 \cdot N+1}^{N}$ ▷ Oracle validation data.
  for $\rho \in \{0\%, \cdots, 250\%\}$ **do** ▷ Reweighing factor.
    $N_r = 2/3 \cdot \rho \cdot N$ ▷ Generate N_r extra points.
    Generate a set $\mathcal{R} = \{\mathbf{a}_i\}_{i=0}^{N_r}$ of artificial points using Algorithm 2 or 3
    // Relabel the data
    for $i = 1 \cdots N_r$ **do**
      $u_i \leftarrow \text{predict}(oracle, \mathbf{a}_i)$
    end for
    // Add the points to the training set and generate a model
    $\mathcal{O}^{(\rho)} \leftarrow \mathcal{O}_{train} \cup \{(\mathbf{a}_i, u_i)\}_{i=1}^{N_r}$
    $rulemodel = \text{train}(whitebox, \mathcal{O}^{(\rho)})$
    $[fid, comp] = \text{evaluate}(rulemodel, \mathcal{O}_{val})$
  end for
  Return the rule model with the best fidelity.
end procedure

is made available to the white-box technique which allows it to make better predictions. An important side-effect is that the importance of the boundary region as opposed to the rest of the input space is increased. This could be interpreted as a reweighing process of the two regions of space: generating no points at all gives more focus to the space outside of the boundary regions, generating an infinite amount of extra information would remove all importance of the outside region. It is clear that we have to find some balance between both of these behaviours. This trade-off can be quantified in the fraction $\rho = \frac{\#\text{boundary region}}{\#\text{outside region}}$, where $\rho = 1$ gives approximately equal importance to both regions. Through empirical experimentation we found that the optimal boundary/outside region importance always lies within the interval $[0, 2.5]$ (where setting $\rho = 0$ is the same as using the original oracle set). The exact value is instance-dependent but it is not computationally difficult to perform a linear search for a set of feasible ρ values. Combining all of the previous ideas leads to Algorithm 1.

Data generation Given the valley points, we want to generate extra data in the neighbourhood thereof, on both sides of the decision boundary and not too far away. The only remaining question is how to define 'not too far away'. The relationship between the distance in output space (e.g. score) and the difference in input space is generally not known except in some specific cases. Depending on the type of black-box technique used, and more specifically whether a continuous output score is provided, we propose two viable ways of making this design choice.

Algorithm 2 Valley–Boundary Approximation

procedure GENERATEARTIFICIAL(oracle,N_v,N_r) ▷ N_v valley and boundary points, N_r random points
  $\mathcal{V} \leftarrow \{(\mathbf{x}_i, \pi_i) \,|\, (\mathbf{x}_j, \pi_j) \notin \mathcal{V} \quad \pi_i \leq \pi_j\}_{j=1}^{N_v}$
  $\mathcal{B} \leftarrow \{\mathbf{b}_i = \arg\min_{\mathbf{x}} (\text{confidence}(\mathbf{x}_i))\}_{i=1}^{N_v}$
  for $\mathbf{b}_i \in \mathcal{B}$ **do**
    for $\mathbf{v}_i \in \mathcal{V}$ **do**
      $\mathbf{d}_{\mathbf{b}_i,\mathbf{v}_i} = |\mathbf{b}_i - \mathbf{v}_i|$
    end for
  end for
  // Generate N_r extra data points in the problem region.
  $\mathcal{R} \leftarrow \{\mathbf{b}_r + \mathbf{c} \cdot \mathbf{d}_{\mathbf{b}_r,\mathbf{v}_r}\}^{N_r}$
  with $\mathbf{c} \in [-1, 1]^m$,
    $r \in \{0, 1, ..., N_v\}$
end procedure

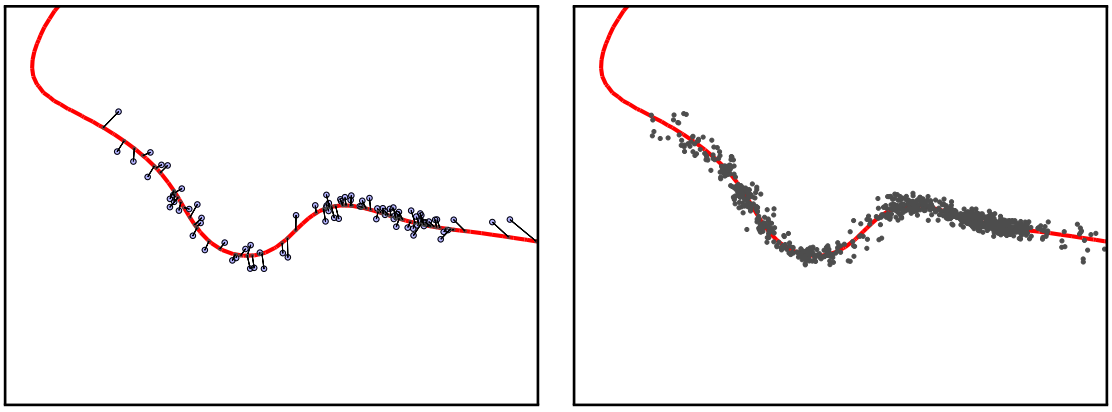
1) Valley–Boundary Approximation: The valley–boundary approximation requires that a (locally convex) continuous confidence measure of the black-box model's prediction be known. The confidence of a prediction can be formalized by introducing the confidence function $\pi(C_i|\mathbf{x}) : (C, \mathbf{x}) \rightarrow \mathbb{R}$. A high output value indicates high confidence of classifying input vector $\mathbf{x}$ as belonging to class C_i. Note that the exact return value of this function does not matter since we are only interested in a ranking of (un)certainty.

In a first step, for each valley point, we look for a point nearby on the decision boundary, we will call these points *boundary points* (see Figure 5(a)). Once we have found these points, we can define the interesting region as that which contains all of the points in the input space that lie closer to the decision boundary points than the original valley points. Finding these boundary points boils down to solving the optimization problem:

$$\arg\min_{\mathbf{x}} \quad \pi(C_i \,|\, \mathbf{x}), \tag{1}$$

when starting from the respective valley points. Function π is a measure for the confidence of the class prediction C_i made by the black-box model for input vector $\mathbf{x}$ and $\mathbf{v}$ is the starting valley point. As the confidence approaches 0%, the oracle gives very uncertain predictions for the input vector $\mathbf{x}$ and thus $\mathbf{x}$ must lie very close to the decision boundary. The minimization algorithm that we used for finding the boundary points is the Broyden-Fletcher-Goldfarb-Shanno Quasi-Newton method (with a cubic line search procedure). This Quasi-Newton method uses the BFGS formula for updating the approximation of the Hessian matrix of the confidence function [49].

If we give the optimization procedure the valley points as initial starting points and repeat the minimization process for each valley point, the output will be a list of points that lie close to the valley points, on the decision boundary. In some rare cases, an



(a) Finding the boundary points: the original valley points are connected to their nearest minima (the boundary points).

(b) Once the boundary region is found, random points are generated (illustrated for gradient descent).

Fig. 3 : Graphical representation of the algorithm with the valley-boundary approximation when applied to the Ripley dataset.

initial overshoot might cause a point to lie far away but this is easily remedied using Grubbs test for outlier detection [50], [51]. Following the Ripley example, the result of this procedure is displayed in Figure 3(a), where each valley point is connected to it is associated boundary point, found by the BFGS Quasi-Newton algorithm.

Once we have correctly identified the boundary points, we choose the generation region to be that which contains all of the points that lie as far from a boundary point as its corresponding valley point (in each feature dimension in the input space) and generate points uniformly in this region. The distance in each of the m dimensions is given by the component-wise distance vector $\mathbf{d}_{\mathbf{x},\mathbf{y}}$:

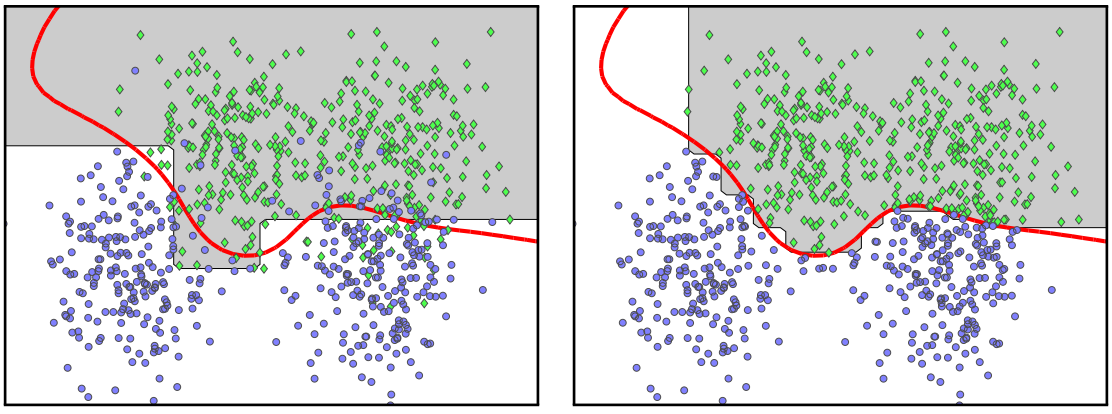
$$d^{(i)}_{\mathbf{x},\mathbf{y}} = |\mathbf{x}_i - \mathbf{y}_i| \tag{2}$$

$$\mathbf{d}_{\mathbf{x},\mathbf{y}} = \left[d^{(1)}_{\mathbf{x},\mathbf{y}}, d^{(2)}_{\mathbf{x},\mathbf{y}}, \cdots, d^{(m)}_{\mathbf{x},\mathbf{y}}\right]. \tag{3}$$

The set of points lying in the boundary region associated with a boundary point $\mathbf{b}$ and a valley point $\mathbf{v}$ is therefore defined as:

$$\{\mathbf{d}_{\mathbf{x},\mathbf{b}} \preceq \mathbf{d}_{\mathbf{v},\mathbf{b}} \mid \forall\, \mathbf{x} \in S\}, \tag{4}$$

where '$\preceq$' is the element-wise smaller than order-relation and S the input space. The resulting region is displayed in Figure 5(a). This choice allows us to generate random numbers computationally fast (as opposed to previous methods such as [18]). A more exact scheme would be to perform *rejection sampling* or *Monte Carlo sampling* to determine which points of the underlying distribution lie as near to the confidence value as the original confidence value. This turns out to be computationally infeasible to perform for all valley/boundary point pairs in very high dimensional data. The resulting procedure is summarized in Algorithm 2, where we introduced the π_i shorthand notation



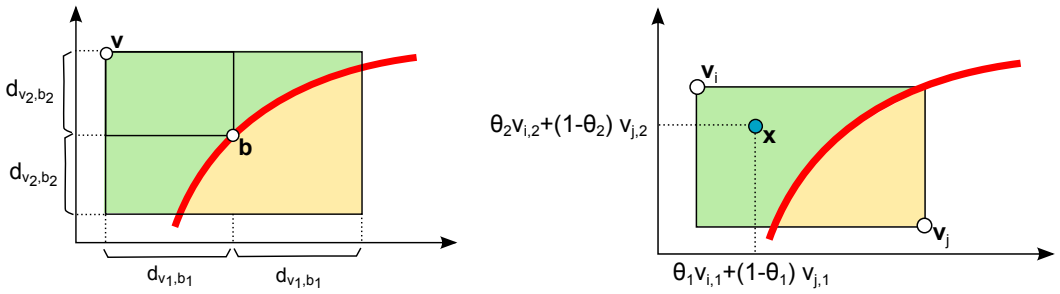
(a) C4.5 boundary. (b) ALPA boundary.

Fig. 4 : Example of C4.5 and ALPA boundaries on the Ripley dataset using Ripper as a rule extraction algorithm. The SVM decision boundary is shown by a curved red line. The rule set decision boundaries are drawn by straight black lines.

to denote the highest confidence prediction for an input vector $\mathbf{x_i}$. Continuing the Ripley example, after finding the valley/boundary pairs we can generate points uniformly distributed in the region defined using Equation 4. The result of applying the procedure to each valley point and generating random points is displayed in Figure 3(b).

The resulting model of the thus far described method using uncertainty descent is displayed in Figure 4. Here, a rule extraction algorithm was trained on the original dataset as a baseline (Figure 4(a)) and a second, baseline rule extraction algorithm was trained using the method explained in this section (Figure 4(b)). Although the white-box technique (C4.5) achieved a fairly good explanation of the black-box model, using our proposed method allowed the rule extraction method to fine-tune on the important regions and the fidelity increased. Both the C4.5 and the ALPA boundary regions were generated using the same pruning factor. As one can tell by looking at the figure choosing the right pruning factor for the rule extraction algorithm remains very important to not produce overcomplicated models. The reason why ALPA is able to arrive at more complicated models is due to the fact that the relabling process removes noise and reduces ambiguity. This in turn allows for more rules to be generated since the effect of pruning is less noticeable.

Despite the fact that we have constructed the algorithm with computational complexity in mind, we should note that the algorithm does not scale well for datasets with large input dimension m. The culprit of this is the Quasi-Newton search procedure which must be repeated for each valley point. When using secant updating methods, the complexity of each Quasi-Newton iteration takes $\mathcal{O}(m^3)$ operations. Although the number of iterations is linear for quadratic objective functions, it behaves superlinear (but not quadratic) in terms of convergence for more general problems. Since the process



(a) Valley–Boundary Approximation. (b) Valley–Valley Approximation.

Fig. 5 : Problem regions as defined by (a) the valley–boundary approximation and (b) the valley–valley approximation.

has to be repeated for each of the $\mathcal{O}(N)$ chosen valley points, the total amount work of work is bounded by $\mathcal{O}(C \cdot N \cdot m^5)$, where C is the time required to evaluate a prediction by the black-box model.

2) Valley–Valley Approximation: Defining a good confidence function can be hard or the dimensionality of the dataset might be very high. In these cases, a rougher estimate of the neighbourhood can be defined by considering the region between two points that lie in different sides of the decision space. This eliminates the need for the gradient descent step and thus the need for an explicit confidence function. Given two valley points at opposite half-spaces of a decision boundary (e.g. a hyperplane), the line segment connecting both valley points will certainly cross the boundary and lie at least partly in a region of high uncertainty (and due to continuity contain more points of low confidence)[5]. This line segment is defined by the convex combination of two nearby valley points:

$$\mathbf{r} \quad = \quad \theta \cdot \mathbf{v}_i + (1-\theta) \cdot \mathbf{v}_j, \quad \theta \in [0,1], \tag{5}$$

where $\mathbf{v}_i$ and $\mathbf{v}_j$ are nearby valley points and all the $\mathbf{r}$ are points on the line segment. By extension we can define the region in a similar way as before as those points that lie in the cuboid region in input space, defined by the convex combination in each coordinate dimension:

$$R \quad = \quad \{\mathbf{x} \in S \,|\, \exists\, \boldsymbol{\theta} \in [0,1]^m : \mathbf{x} = \boldsymbol{\theta} \cdot \mathbf{v}_i + (\mathbf{1}-\boldsymbol{\theta}) \cdot \mathbf{v}_j\}, \tag{6}$$

where S is the original input space, m the number of dimensions, $\boldsymbol{\theta}$ a vector of values, each of which lies in $[0,1]$ and $\mathbf{1}$ a vector of ones. This leads to Algorithm 3, where

[5]When the black-box technique maps the input data in some feature space, the above statement is only valid for continuous mappings (e.g. for SVMs and ANNs). A full proof for all black-boxes considered in this manuscript is provided in the Appendix.

Algorithm 3 Valley–Valley Approximation

1: //N_v valley points, N_r random points.
2: **procedure** GENERATEARTIFICIAL(oracle,N_v,N_r)
3: // Determine the N_v valley points.
$\mathcal{V} \leftarrow \{(\mathbf{x}_i, \pi_i) \,|\, (\mathbf{x}_j, \pi_j) \notin \mathcal{V} \quad \pi_i \leq \pi_j\}_{i=1}^{N_v}$
4: // Calculate the pairwise distance matrix.
5: **for** $(\mathbf{v}_i, \mathbf{v}_j) \in \mathcal{V} \times \mathcal{V}$ **do**
6: **if** $\text{predict}(oracle, \mathbf{v}_i) \neq \text{predict}(oracle, \mathbf{v}_j)$ **then**
7: $D_{i,j} = ||\mathbf{v}_i - \mathbf{v}_j||_2$
8: **else**
9: $D_{i,j} = \infty$
10: **end if**
11: **end for**
12: // Determine the set of neighbours and generate
// N_r extra data points in the problem region.
13: $\mathcal{N} = \{(\mathbf{v}_i, \mathbf{v}_j) \in \mathcal{V} \times \mathcal{V} \,|\, \arg\min_{i,j} \min_{j} D_{i,j}\}$
14: $\mathcal{R} \leftarrow \{\boldsymbol{\theta} \cdot \mathbf{v}_i + (1 - \boldsymbol{\theta}) \cdot \mathbf{v}_j\}^{N_r} \quad \boldsymbol{\theta} \in [0,1]^m, (\mathbf{v}_i, \mathbf{v}_j) \in \mathcal{N}$
15: **end procedure**

we introduced the π_i shorthand notation to denote the confidence of a prediction for an input vector $\mathbf{x}_i$.

This method does not use any information of the black-box model or its transformed space beyond the output score or a certainty ranking of the original data. The drawback of the line segmentation method is that although we are ensured that we will cover the most uncertain points, we could be covering quite a lot of points that are not so uncertain at all thereby including not so informational ones as well, thus slowing down the progress of the algorithm. Consider the extreme case in which only one valleypoint lies in half-space S_1 and all the other valleypoints lie in half-space S_2. We would then cover a very large part of both half-spaces by using this method.

The advantage, however, is that it is usually much faster in the generating phase and that it does not require an uncertainty function. The valley–valley approximation method does not suffer from dimensionality scaling issues, but is more sensitive to the number of data presented. This is due to the fact that calculating the distance matrix requires $\frac{N(N-1)}{2}$ operations, each of which considers the dimension m only once, resulting in a complexity bound of $\mathcal{O}(N^2 \cdot m + N \cdot C)$. The second term comes from the valley point prediction phase (step 2) in which in the order of N points have to be predicted in order to determine their confidence, with C again the blackbox prediction time. This operation is usually very fast (i.e. the time required to evaluate a non-linear function), but it depends on the black-box technique used.

Note that the computationally most expensive steps in both Algorithm 2 (step 4 to 8) and Algorithm 3 (step 3 to 8) have to be performed only once since their results can

Black-box	$\pi(C_i\|\mathbf{x})$
SVM	$h_i(\mathbf{x}) - \max\limits_{j \neq i} h_j(\mathbf{x})$
ANN	$(e^{y_i(\mathbf{x},\mathbf{w})}) / \left(\sum\limits_{k=1}^{n} e^{y_k(\mathbf{x},\mathbf{w})} \right)$
RF	$\underset{k}{\text{avg}}\, \mathrm{I}(\mathrm{h}_k(\mathbf{x}) = y) - \max\limits_{j \neq y} \underset{k}{\text{avg}}\, \mathrm{I}(\mathrm{h}_k(\mathbf{x} = j)$

TABLE III: Confidence functions

be stored outside of the loop at line 5 in Algorithm 1.

IV. Covered Techniques

In order to test the validity of the proposed rule extraction technique, we will consider several problem instances with varying characteristics. We consider SVMs, ANNs and Random Forests (RF) as black-box techniques in this study. For each of these techniques, we deduce how we come to a mathematically and semantically sound formula for its uncertainty functions. In order to explain these complex models, we use C4.5 [22] and Ripper [52] as white-box rule inducers. This choice is motivated by the fact that these produce distinctly different rule sets, respectively a tree structure and a list of many-term rules. An overview of the confidence function for all three black-box techniques is given in Table III. In order to understand the semantics of these functions, we must take a look into the inner workings of the black-box techniques first.

A. Artificial Neural Network

A feed-forward neural network [1] can be described as a series of functional transformations, applied in sequence. This sequence is described in layers, where each layer defines a process step. Given a dataset of N data points $\{\mathbf{x}_i, y_i\}_{i=1}^{N}$, with input data $\mathbf{x}_i \in \mathbb{R}^m$, each neuron of the input layer calculates the linear combination of the m input feature values $x_1, \ldots, x_m$. This is then transformed using a non-linear activation function. Repeating this process for each layer in the network (two in our case), we end up with the final output value:

$$y_k(\mathbf{x}, \mathbf{w}) \;=\; \sigma \left(\sum_{j=0}^{J} w_{k,j}^{(2)} h \left(\sum_{i=0}^{m} w_{j,i}^{(1)} \right) \right), \tag{7}$$

where h is the activation function for the hidden layer and σ the sigmoid function.

As Equation (7) shows, this two stage process results in a complex non-linear function, fully parametrized by the weight vector $\mathbf{w}$. The trained model is very difficult to

understand without some help from rule extraction techniques. By applying the softmax function to the output of the neural network we get an estimation of the posterior class probabilities (cfr. [53]), which we can use as a surrogate for the prediction certainty:

$$\pi\left(C_i \,|\, \mathbf{x}\right) \;=\; \Pr\left(C_i \,|\, \mathbf{x}\right) = \frac{e^{y_i(\mathbf{x},\mathbf{w})}}{\sum_{k=1}^{n} e^{y_k(\mathbf{x},\mathbf{w})}} \tag{8}$$

B. The Support Vector Machine

The Support Vector Machine is a learning procedure based on the statistical learning theory [2]. Given a training set and corresponding binary class labels $y_i \in \{-1, +1\}$, the SVM classifier constructs a hyperplane in a feature space, induced by the non-linear function φ.

This hyperplane, $\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}) + b = 0$, discriminates between the two classes. By minimizing $\mathbf{w}^T\mathbf{w}$, the margin between both classes is maximized. In primal weight space the classifier then takes the form

$$y(\mathbf{x}) = \operatorname{sign}\left(\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}) + b\right), \tag{9}$$

but, on the other hand, is never evaluated in this form.

To solve the system of inequalities, it is reformulated as a convex optimization problem and then solved using Lagrange multipliers. The exact details of this procedure are beyond the scope of this report but this leads to the following classifier:

$$y(\mathbf{x}) = \operatorname{sign}\left(\textstyle\sum_{i=1}^{N} \alpha_i\, y_i\, K(\mathbf{x}_i, \mathbf{x}) + b\right), \tag{10}$$

where $K(\mathbf{x}_i, \mathbf{x}) = \boldsymbol{\varphi}(\mathbf{x}_i)^T\boldsymbol{\varphi}(\mathbf{x})$ is a positive definite kernel satisfying the Mercer theorem.

As equation (10) shows, the SVM classifier can be a very complex function if a non-linear kernel is chosen. Trying to comprehend the logics of the classifications made is quite difficult, if not impossible.

In order to apply our method, we need to define an uncertainty measure. There exist measures to define the a posteriori probability of a prediction for the binary classification case [54], [55]. Unfortunately these methods are computationally resource intensive and not trivially generalizable to a one-vs-all multi-class setting. We therefore resort to an approximate measure that uses the decision value information, already given by SVM (the decision value is the argument given to the signum function of Equation (10)). These decision values should not be interpreted as probability measures, but they are a measure of confidence nonetheless since they represent the distance from the ”optimal” decision hyperplane. A prediction with high confidence is far away from the decision plane and will have a large value. If the confidence for more than one class is high, the

resulting output confidence should be lowered, this induces a ranking on the confidence (which is enough for our algorithm to work). To take these effects into account we define the confidence of a prediction as:

$$\pi\left(C \mid \mathbf{x}\right) = h_i(\mathbf{x}) - \max_{j \neq i} h_j(\mathbf{x}), \tag{11}$$

where C is the predicted class for input vector $\mathbf{x}$ and h_i is the decision value of classifier i for input vector $\mathbf{x}$.

C. Random Forest

A Random Forest is a classifier consisting of a collection of n tree-structured classifiers $\{h(\mathbf{x}, \mathbf{r}_k), k = 1, ..., n\}$ where the $\{\mathbf{r}_k\}$ are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input $\mathbf{x}$. This is also known as a bagging approach. The uncertainty of a random forest is defined by the margin function [56]:

$$\text{margin}(\mathbf{x}, y) = \underset{k}{\text{avg}}\, \text{I}(\text{h}_k(\mathbf{x}) = y) - \max_{j \neq y} \underset{k}{\text{avg}}\, \text{I}(\text{h}_k(\mathbf{x} = j), \tag{12}$$

where $I(\cdot)$ is the indicator function. The margin function is very similar to the previously defined uncertainty function for SVM (Equation 11). It could be used directly as a proxy for the certainty function in the algorithm. Unfortunately its discreteness makes it ill-suited for the Gradient Descent Algorithm. Given N trees in the collection of trees, the uncertainty function can only reach one of the $N + 1$ values of $\{\frac{0}{N}, \frac{1}{N}, ..., \frac{N}{N}\}$. The Gradient Descent approximation using Quasi-Newton needs small continuous changes in target function value to work properly since it updates the Hessian matrix based on these changes. In cases like these, it is therefore advisable to work with the valley-valley approximation instead (see Section III-A.2).

V. Experimental Setup

In this section we demonstrate the empirical usefulness of ALPA in two experimental benchmarks. First of all we confirm whether the rule extraction mechanism works well by evaluating the fidelity and accuracy of ALPA and comparing it to the original rule induction technique. Next, we determine the added value of the presented method vis-a-vis other rule extraction mechanisms.

A. Datasets

As mentioned in Section II-A, there are two scenarios in which rule extraction is useful and we selected datasets that comply with either one of these two use-cases. The main application is to obtain insight in the performance of the black-box model.

Dataset	Instances	Variables	Classes
arrhythmia	452	279	2
balance	576	4	3
bene	3122	24	2
breastcancer	568	31	2
cmc	1473	9	3
credit-a	690	16	2
diabetes	766	8	2
ecoli	336	8	8
ger	999	20	2
glass	214	10	7
haberman	306	3	2
heart	270	14	2
hepathitis	155	20	2
ionosphere	349	34	2
iris	150	4	3
lymph	148	19	4
mammography	829	6	2
mushroom	8124	22	2
ripley	1249	2	2
wine	129	13	3
krvskp	3196	36	2
ringnorm	7401	21	2
sick	3772	30	2
vowel	990	14	11
wisconsin	198	34	2

TABLE IV: Dataset Properties

Additionally, we could use our rule-extraction approach to improve the accuracy of the white-box technique's predictions as well. Both scenarios require the black-box model to outperform the white-box model with respect to the dataset. That is, we would probably not be interested in the behaviour of the black-box model if its performance were worse than that of the white-box model. Furthermore, there is little hope of improving the white-box model by using information from a black-box model that performs worse. As was mentioned by [18] this requirement is often overlooked in the literature, leading to overoptimistic results.

All of the datasets used in this study are listed in Table IV and were collected from the UCI machine learning repository [57]. We focus on datasets that have been used in various other rule extraction studies (e.g. [58], [59], [18]) and consequently allow us to compare the results to previous (and future) research. The accuracy requirement mentioned in the previous paragraph is not always met due to varying performances of the black-box and white-box techniques, but we explicitly filter out these infeasible situations ad hoc (cfr. Section VI). This leads to a heterogeneous selection of 25 datasets of varying size, number of variables and number of classes (see Table IV).

B. Rule extraction performance

To test whether the algorithm works well with a wide variety of black-box and white-box techniques, we carry out tests on a grid of black-box technique (SVM, ANN, RF) and white-box technique (Ripper, C4.5) combinations. Additionally, we test each white-box technique with default and extensive pruning settings (based on the pruning factor c[6]). This leads to a total of 12 experimental settings per dataset. We mentioned before (Section III) that the best number of valley points depends on the black-box and supplementary tests are performed to determine this parameter for each black-box technique as well.

For every dataset we evaluate the fidelity and the accuracy of our algorithm ten times (using a ten fold cross-validation scheme) for every ρ value (boundary/non-boundary trade-off parameter) in the linear search. In each fold iteration, we split the data in $\frac{9}{10}$ learning data and hold out $\frac{1}{10}$ of the data for evaluation purposes (test data). Subsequently in the learning procedure $\frac{2}{3}$ of the learning data is used as training data and the remaining $\frac{1}{3}$ is used as validation data for tuning ρ. After performing this linear search, the best trade-off parameter ρ^* can then be used to extract a candidate rule set. This rule set is then used and tested for similarity with the black-box model on the testing data. These test results are then compared to the fidelity and accuracy of the white-box technique when used as a rule induction method on the original dataset.[7]

C. Active learning component and the pedagogical approach

In these experiments we are interested in the performance of ALPA versus other techniques. The main issue that we encountered was that no open implementations were available for most of the techniques in the literature (even though software availability is reported to be one of the key criteria for rule extraction techniques by Andrews [23]) so we had to resort to new implementations.

A first comparison of interest is the difference of performance of ALPA with respect to randomly generating data (without a smart choice of boundary). If the main *active-learning* premises of ALPA is true (i.e. boundary regions are more important) the outcome should be that ALPA performs significantly better on the bench of dataset.

As a test of validity, we also compare ALPA to the technique on the ALBA technique on which it was inspired. This comparison is somewhat more limited because ALBA only works on SVMs, but it can provide us interesting insights nonetheless. A good outcome for this test would show that both methods perform more or less similar, because

[6]The pruning factor controls the number of rules in the rule set by limiting the width of the prediction confidence interval or the minimum total weight of the instances covered by a rule for C4.5 and Ripper respectively.

[7]This test set-up led to the generation of a total of 150.000 white-box models.

TABLE V : Comparison of the performance over the grid of experiments of ALPA versus the original rule induction techniques (C4.5, Ripper). The rule complexities were explicitly kept in the same order using differing pruning factors. $\mathrm{R}+$ indicates the number of wins, $\mathrm{R}=$ the number of draws and $\mathrm{R}-$ the number of losses of ALPA versus the original white-box technique.

			Fidelity				Accuracy				Median Rules	
	BB	ρ	R+	R=	R-	p-value	R+	R=	R-	p-value	WB	ALPA
C4.5 (0.25 vs 0.001)	RF	25	94	31	20	$\mathbf{5.72 \cdot 10^{-8}}$	59	34	52	0.3087	30	25
		20	89	32	24	$\mathbf{7.77 \cdot 10^{-10}}$	64	38	43	0.1629	30	27
		15	99	28	18	$\mathbf{9.00 \cdot 10^{-11}}$	67	32	46	0.0632	30	25
	SVM	25	105	23	23	$\mathbf{3.08 \cdot 10^{-12}}$	85	29	37	$\mathbf{2.30 \cdot 10^{-3}}$	23	29
		20	105	20	26	$\mathbf{8.37 \cdot 10^{-12}}$	87	29	35	$\mathbf{1.80 \cdot 10^{-4}}$	23	29
		15	95	28	28	$\mathbf{5.24 \cdot 10^{-9}}$	81	28	42	$\mathbf{8.41 \cdot 10^{-3}}$	23	29
	ANN	25	105	18	19	$\mathbf{1.01 \cdot 10^{-15}}$	78	22	42	$\mathbf{8.29 \cdot 10^{-3}}$	23	33
		20	106	23	13	$\mathbf{2.85 \cdot 10^{-16}}$	74	25	43	$\mathbf{1.80 \cdot 10^{-3}}$	23	34
		15	107	15	20	$\mathbf{1.16 \cdot 10^{-14}}$	73	20	49	$\mathbf{6.38 \cdot 10^{-3}}$	23	29
Ripper (2 vs 12)	RF	25	104	31	22	$\mathbf{8.01 \cdot 10^{-13}}$	78	32	47	**0.0110**	4	5
		20	100	31	26	$\mathbf{3.55 \cdot 10^{-12}}$	74	33	50	**0.0158**	4	6
		15	100	29	28	$\mathbf{2.73 \cdot 10^{-11}}$	72	30	55	**0.0408**	4	6
	SVM	25	88	34	22	$\mathbf{7.44 \cdot 10^{-12}}$	68	40	36	$\mathbf{3.72 \cdot 10^{-4}}$	4	6
		20	92	25	27	$\mathbf{2.18 \cdot 10^{-12}}$	69	30	45	$\mathbf{6.90 \cdot 10^{-3}}$	4	5
		15	94	26	24	$\mathbf{1.94 \cdot 10^{-12}}$	71	36	37	$\mathbf{9.06 \cdot 10^{-4}}$	4	5
	ANN	25	101	20	20	$\mathbf{4.46 \cdot 10^{-12}}$	77	25	39	$\mathbf{2.31 \cdot 10^{-4}}$	4	7
		20	100	21	20	$\mathbf{5.82 \cdot 10^{-12}}$	78	25	38	$\mathbf{2.30 \cdot 10^{-4}}$	4	7
		15	95	24	22	$\mathbf{7.89 \cdot 10^{-10}}$	68	29	44	$\mathbf{5.71 \cdot 10^{-3}}$	4	6

that would mean that we can achieve similar behaviour without making architectural assumptions: the main goal of our *pedagogical* technique.

VI. Benchmark Results

A. Rule extraction performance

In our first experiments we kept all pruning factors of the white-boxes similar. As can be seen from Table **??** in the Appendix, the results of these experiments show superior performance of ALPA on all of the datasets in terms of both accuracy as well as fidelity. In all of the comparison tables, $\mathrm{R}+$ indicates the number of wins, $\mathrm{R}=$ the number of draws and $\mathrm{R}-$ the number of losses of ALPA in terms of fidelity and accuracy for that particular setting. Note that these numbers do not necessarily sum to the same amount, as for some datasets the non-linear black-box performed equally or worse, compared

to the white-box. An example of this are datasets that are particularly easy to learn or datasets where linear frontier-based models are superior to non-linear ones (an overview of the predictive power of all of the methods is given in Table **??** in the Appendix). These experiment instances do not fit the premises of the study and are therefore not included (as motivated by Martens et al. [18]).

A first concern with these results is that improving fidelity often means adding more rules. As mentioned in Section III this is not problematic since we are only working inside the constraints of the pruning factor of the rule algorithm (and this can be varied according to the user's preferences). Nevertheless, it could be argued that in order to perform a truly fair comparison of the performance, one should also compare ALPA with the more stringent pruning coefficient with the more relaxed white-box models, effectively creating the same rule-complexity for both type of models[8]. The results for these experiments are displayed in Table V. An overview of the actual fidelity values for all of the datasets for the setting of an SVM is shown in Table **??** in the Appendix.

As suggested by Demsar et al [60], we applied the Wilcoxon signed rank test to determine whether these results are significant or not. For each of the combinations, we are able to reject the null hypothesis that both algorithms perform equally well ($p \ll 0.01$) in terms of fidelity (significant results are marked in bold). The high fidelities show that the primary aim of explaining the black-box model is achieved, as the generated rules make the same predictions as the black-box model for most of the test data. Similar results can be observed for accuracy, though less articulate. This suggests that on many experiment instances, the original white-box technique performed suboptimal for these problem instances, when given only the training data. The surprising consequence is that the proposed methodology could therefore be used as well in contexts where the focus lies with accuracy improvements.

The fact that the accuracy results do not hold as strongly as the fidelity results is a general tendency and stems from two reasons. First of all, the ALPA method as presented and applied in this study is tuned to improve fidelity first and foremost (hence, to explain the black-box model). Secondly, improving accuracy is a more difficult problem to solve in general. Similar results have been reported in a previous SVM rule extraction study [18].

Although significant results are achieved across the board for ANNs and SVMs, the RF results are weaker. This can be explained by the fact that the first use the fine valley-boundary approximation, whereas the latter uses the coarser valley-valley approximation. Thus, ALPA works better for black-box models that output continuous uncertainty scores and we would recommend using the valley-boundary approximation over the valley-valley should the choice be available to the end user.

[8]Note that it is very hard to arrive at exactly the same number of rules due to the nature of the techniques, but a comparison of the median number of rules (last column) shows that the models have approximately similar rule complexity.

B. Active learning component and the pedagogical approach

Table VI shows the comparison of ALPA with the original white-box method, random data generation and ALBA on the same split-up over the bench of datasets using the same pruning factor for C4.5 and Ripper respectively. The results show that ALPA does significantly better than the original rule induction technique as well as random data generation across the board. One exception seems to be the case of Neural Networks with low number of valley points. One possible explanation of this aberrant behaviour is that the ANN has more concentrated regions of low confidence and we might miss out on some interesting regions when focussing on only a small part thereof. Based on these results, we advise to keep the boundary trade-off parameter high enough ($\rho > 0.15$) when working with ANNs in particular. Comparing the ALPA results to those form ALBA reveals that they perform very similar when applied to SVM models. In most cases ALPA slightly outperforms ALBA, but this result is not significant over all datasets so no conclusions can be made as to the superiority of one algorithm versus the other with respect to the fidelity and accuracy of the extracted rules. Simultaneously, the decompositional nature of ALBA makes it useless for rule extraction from ANNs or RFs, so on a broad applicability level ALPA has an unmistakable advantage.

VII. Discussion of the main results

The results from the previous section show that ALPA is very often able to formulate comprehensible models that are both a good representation of the black-boxes' inner workings (*fidelity*) as well as good stand-alone predictive models (*accuracy*). In Section II-C we mentioned three evaluation dimensions next to fidelity and accuracy. Let us briefly discuss how the proposed algorithm tackles each of the remaining aspects. In terms of *comprehensibility*, we leave the choice of rule structure to the end user: the rule structure can be changed by using a different white-box technique. Furthermore, most rule induction techniques have some complexity or pruning factor available, which the end user can use to limit the resulting rule set complexity. The proposed method is a pedagogical approach that does not depend on any specific architecture and is thus *general* in nature as well. It is worth mentioning that we believe that many of the decompositional approaches from Table I (e.g. those based on support vectors such as [18], [39], [40]) can be adapted to pedagogical variants using the uncertainty methodology provided in Section III.

As discussed in Section III, we tried to limit the computational complexity wherever possible to increase *scalability*. As a consequence, our algorithm is able to formulate good rule sets on realistic datasets in a matter of seconds for most of the problem instances we tested for. Although the algorithm we propose performs efficiently on the datasets included in this study, we realize that much larger datasets could be used in practice and the curse of dimensionality does apply to our algorithm to some extent. Our algorithm can however be applied in an online fashion so, again, the performance/time complexity trade-off can be chosen by the user in terms of execution time. Furthermore,

ALPA can rely on a flexible choice of rule induction technique for both the readability as well as the scalability.

Finally, we have made our implementation easy to use and easily *available* to data mining practitioners by providing an open WEKA package implementation of the proposed method available on our website[9].

VIII. Conclusion and Future Work

Recent advances in data mining focus mainly on improving the generalization behaviour of predictive models and have arguably led us further away from the data mining goal of creating "*ultimately understandable* patterns in data." [20]. Our approach attempts to leverage the very good predictive performance of the black-box models, while still resulting in a comprehensible set of rules. If we want to answer the call by Schmueli and Koppius [21] for the use of predictive analytics in social sciences and humanities, we need to have an eye on this comprehensibility issue. Our ALPA technique is based on an already rich rule extraction literature. However, previous approaches were either quite basic in their algorithmic working and had only limited performance (e.g. changing the class label to the black-box prediction [38]), or performed very well by using advanced concepts of the black-box (e.g. using the support vectors [18]) but were limited to one class of techniques only. ALPA is the first to be applicable to any black-box model while using advanced algorithmic concepts. The experiments have shown the suitability for rule extraction from SVMs, ANNs and Random Forests, which holds considerable promise for the broad applicability of ALPA to other black-box techniques and domains. We hope that the publicly available ALPA software, compatible with Weka, can spur a further investigation in the development and application of rule extraction.

References

[1] C. M. Bishop, "Neural networks: a pattern recognition perspective," *Neural Networks*, no. 1973, pp. 1–23, 1996.

[2] V. N. Vapnik, *The Nature of Statistical Learning Theory*, ser. Statistics for Engineering and Information Science, M. Jordan and S. L. Lauritzen, Eds. Springer, 1995, vol. 8, no. 6.

[3] T. G. Dietterich, "Ensemble Methods in Machine Learning," 1990.

[4] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings," *IEEE Transactions on Software Engineering*, vol. 34, no. 4, pp. 485–496, 2008.

[5] R. Caruana, N. Karampatziakis, and A. Yessenalina, "An empirical evaluation of supervised learning in high dimensions," *Proceedings of the 25th International Conference on Machine Learning (2008)*, pp. 96–103, 2008.

[6] A. Niculescu-mizil, C. Perlich, G. Swirszcz, and V. Sind, "Winning the KDD Cup Orange Challenge with Ensemble Selection," *JMLR Workshop and Conference Proceedings*, vol. 7, pp. 23–34, 2009.

[7] M. J. Pazzani, S. Mani, and W. R. Shankle, "Acceptance of rules generated by machine learning among medical experts." *Methods of Information in Medicine*, vol. 40, no. 5, pp. 380–385, 2001.

[9] http://www.applieddatamining.com, available, not publicly visible until publication

[8] D. Martens, L. Bruynseels, B. Baesens, M. Willekens, and J. Vanthienen, "Predicting going concern opinion with data mining," *Decision Support Systems*, vol. 45, no. 4, pp. 765–777, 2008.

[9] D. Martens, B. Baesens, T. Van Gestel, and J. Vanthienen, "Comprehensible credit scoring models using rule extraction from support vector machines," *European Journal of Operational Research*, vol. 183, no. 3, pp. 1466–1476, 2007.

[10] T. Van Gestel, B. Baesens, and L. Thomas, *Introduction to Modern Credit Scoring*. Oxford University Press.

[11] N. S. Umanath and I. Vessey, "Multiattribute Data Presentation and Human Judgment: A Cognitive Fit Perspective," *Decision Sciences*, vol. 25, no. 5-6, pp. 795–824, Sep. 1994.

[12] M. Limayem and G. DeSanctis, "Providing Decisional Guidance for Multicriteria Decision Making in Groups," *Information Systems Research*, vol. 11, no. 4, pp. 386–401, Dec. 2000.

[13] G. L. Lilien, A. Rangaswamy, G. H. Van Bruggen, and K. Starke, "DSS Effectiveness in Marketing Resource Allocation Decisions: Reality vs. Perception," *Information Systems Research*, vol. 15, no. 3, pp. 216–235, Sep. 2004.

[14] V. Arnold, N. Clark, P. A. Collier, S. A. Leech, and S. G. Sutton, "The differential use and effect of knowledgebased system explanations in novice and expert judgment decisions," *MIS Quarterly*, vol. 30, no. 1, pp. 79–97, Mar. 2006.

[15] U. Kayande, A. De Bruyn, G. L. Lilien, A. Rangaswamy, and G. H. van Bruggen, "How Incorporating Feedback Mechanisms in a DSS Affects DSS Evaluations," *Information Systems Research*, vol. 20, no. 4, pp. 527–546, Dec. 2008.

[16] Y. Kodratoff, "The Comprehensibility Manifesto," 1994.

[17] M. W. Craven and J. W. Shavlik, "Extracting tree-structured representations of trained neural networks," in *Advances in Neural Information Processing Systems*, D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, Eds., vol. 8. MIT Press, 1996, pp. 24–30.

[18] D. Martens, T. Van Gestel, and B. Baesens, "Decompositional rule extraction from support vector machines by active learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, pp. 178–191, 2009.

[19] B. Baesens, D. Martens, R. Setiono, and J. M. Zurada, "Guest editorial: special section on white box nonlinear prediction models." *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 2406–8, Dec. 2011.

[20] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *AI magazine*, vol. 17, no. 3, pp. 37–54, 1996.

[21] G. Shmueli and O. Koppius, "Predictive analytics in information systems research," *MIS Quarterly*, vol. 35, no. 3, pp. 553–572, 2010.

[22] J. R. Quinlan, *C4.5: Programs for Machine Learning*, M. Kaufmann, Ed. Morgan Kaufmann, 1993.

[23] R. Andrews, J. Diederich, and A. Tickle, "Survey and critique of techniques for extracting rules from trained artificial neural networks," *Knowledge-Based Systems*, vol. 8, no. 6, pp. 373–389, 1995.

[24] E. Strumbelj and I. Kononenko, "An efficient explanation of individual classifications using game theory." *Journal of Machine Learning*, vol. 11, pp. 1–18, 2010.

[25] M. Robnik-Sikonja and I. Kononenko, "Explaining classifications for individual instances." *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, pp. 589–600, 2008.

[26] D. Martens and F. Provost, "Explaining Data-Driven Document Classifications," *MIS Quarterly*, vol. In press, 2013.

[27] G. J. Schmitz, C. Aldrich, and F. S. Gouws, "ANN-DT: an algorithm for extraction of decision trees from artificial neural networks." *IEEE Transactions on Neural Networks*, vol. 10, no. 6, pp. 1392–1401, 1999.

[28] O. Boz, "Converting a trained neural network to a decision tree." Ph.D. dissertation, Lehigh University, 2000.

[29] H. Nuñez, C. Angulo, and A. Catala, "Rule-based learning systems for support vector machines," *Neural Processing Letters*, vol. 24, no. 1, pp. 1–18, 2002.

[30] Z.-H. Zhou, Y. Jiang, and S.-F. Chen, "Extracting symbolic rules from trained neural network ensembles," *AI Communications*, vol. 16, no. 1, pp. 3–15, 2003.

[31] U. Markowska-kaczmar and W. Trelak, "Extraction of Fuzzy Rules from Trained Neural Network Using Evolutionary Algorithm * The main idea of the fuzzy Rule Extraction Method (REX)," no. April, pp. 149–154, 2003.

[32] J. R. Rabuñal, J. Dorado, A. Pazos, J. Pereira, and D. Rivero, "A new approach to the extraction of ANN rules and to their generalization capacity through GP." *Neural Computation*, vol. 16, no. 7, pp. 1483–1523, 2004.

[33] F. Chen, "Learning accurate and understandable rules from SVM classifiers," Ph.D. dissertation, Simon Fraser University, 2004.

[34] N. Barakat and J. Diederich, "Eclectic rule-extraction from support vector machines," *International Journal of Computational Intelligence*, vol. 2, no. 1, pp. 59–62, 2005.

[35] G. Fung, S. Sandilya, and R. B. Rao, "Rule extraction from linear support vector machines," in *Neurocomputing*, ser. Lecture Notes in Computer Science, T. Ho, D. Cheung, and H. Liu, Eds., vol. 74, no. 1-3. ACM Press, 2005, pp. 107–112.
[36] R. Setiono, B. Baesens, and C. Mues, "Risk Management and Regulatory Compliance: a Data Mining Framework Based on Neural Network Rule Extraction," *Proceedings of the International Conference on Information Systems ICIS 2006*, 2006.
[37] J. Huysmans, B. Baesens, and J. Vanthienen, "ITER: an algorithm for Predictive Regression Rule extraction," in *8th International Conference on Data Warehousing and Knowledge Discovery*. Springer, 2006, vol. 4081, pp. 270–279.
[38] N. Barakat and A. Bradley, "Rule Extraction from Support Vector Machines: A Sequential Covering Approach," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 6, pp. 729–741, 2007.
[39] M. Farquad, V. Ravi, and S. B. Raju, "Support vector regression based hybrid rule extraction methods for forecasting," *Expert Systems with Applications*, vol. 37, no. 8, pp. 5577–5589, Aug. 2010.
[40] C.-T. Su and Y.-C. Chen, "Rule extraction algorithm from support vector machines and its application to credit screening," *Soft Computing*, vol. 16, no. 4, pp. 645–658, Sep. 2011.
[41] J. Chorowski and J. M. Zurada, "Extracting rules from neural networks as decision diagrams." *IEEE transactions on Neural Networks*, vol. 22, no. 12, pp. 2435–46, Dec. 2011.
[42] M. Craven and J. Shavlik, "Using sampling and queries to extract rules from trained neural networks," *Proceedings of the Eleventh International Conference on Machine Learning*, 1994.
[43] M. Craven, "Rule extraction: Where do we go from here," *Machine Learning Research Group Working Paper*, 1999.
[44] K. Dejaeger, W. Verbeke, J. Huysmans, C. Mues, J. Vanthienen, and B. Baesens, "An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models," *Decision Support Systems*, vol. 51, no. 1, pp. 141–154, 2010.
[45] U. Markowska-Kaczmar and M. Chumieja, "Discovering the Mysteries of Neural Networks," *International Journal of Hybrid Intelligent Systems*, vol. 1, no. 3-4, pp. 153–163, 2004.
[46] V. V. Fedorov, *Theory of optimal experiments*, V. V. Fedorov, Ed. Academic Press, 1972, vol. 59, no. 3.
[47] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Machine Learning*, vol. 15, no. 2, pp. 201–221, 1994.
[48] B. D. Ripley, "Neural Networks and Related Methods for Classification," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 56, no. 3, pp. 409–456, 1994.
[49] D. F. Shanno, "Conditioning of Quasi-Newton Methods for Function Minimization," *Mathematics of Computation*, vol. 24, no. 111, pp. 647–656, 1970.
[50] H. Czichos, T. Saito, and L. E. Smith, *Springer Handbook of Metrology and Testing*. Springer, 2011.
[51] F. Grubbs, "Procedures for detecting outlying observations in samples," *Technometrics*, vol. 11, no. 1, pp. 1–21, 1969.
[52] W. W. Cohen, "Fast Effective Rule Induction," in *Proceedings of the Twelfth International Conference on Machine Learning*, A. Prieditis and S. Russell, Eds., vol. 3. Morgan Kaufmann, 1995, pp. 115–123.
[53] K. Duan, S. Keerthi, W. Chu, S. Shevade, and A. Poo, "Multi-category classification by soft-max combination of binary classifiers," *Multiple Classifier Systems*, pp. 160–160, 2003.
[54] J. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999.
[55] S. Rüping, "A simple method for estimating conditional probabilities for SVMs," 2004.
[56] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
[57] A. Asuncion and D. J. Newman, "UCI Machine Learning Repository," 2007.
[58] D. Martens, M. De Backer, R. Haesen, J. Vanthienen, M. Snoeck, and B. Baesens, "Classification with Ant Colony Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 5, pp. 651–665, 2007.
[59] B. Baesens, R. Setiono, C. Mues, and J. Vanthienen, "Using neural network rule extraction and decision tables for credit-risk evaluation," *Management Science*, vol. 49, no. 3, pp. 312–329, 2003.
[60] J. Demšar, "Statistical Comparisons of Classifiers over Multiple Data Sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

TABLE VI : Comparison of the fidelity of ALPA versus alternative strategies when using C4.5 and Ripper, while keeping the same pruning factor.

			Original				Random				ALBA			
	BB	ρ	R+	R=	R-	p-value	R+	R=	R-	p-value	R+	R=	R-	p-v
C45 (0.25)	RF	25	92	49	1	$\mathbf{2.48 \cdot 10^{-16}}$	85	28	29	$\mathbf{9.30 \cdot 10^{-7}}$				
		20	92	48	2	$\mathbf{1.40 \cdot 10^{-16}}$	81	28	33	$\mathbf{6.27 \cdot 10^{-5}}$				
		15	93	47	2	$\mathbf{7.72 \cdot 10^{-17}}$	80	32	30	$\mathbf{1.56 \cdot 10^{-5}}$				
	SVM	25	108	22	11	$\mathbf{2.24 \cdot 10^{-17}}$	75	38	28	$\mathbf{9.56 \cdot 10^{-6}}$	57	40	44	0.1
		20	103	26	12	$\mathbf{4.16 \cdot 10^{-16}}$	67	41	33	$\mathbf{2.31 \cdot 10^{-3}}$	61	31	49	0.4
		15	105	26	10	$\mathbf{7.23 \cdot 10^{-17}}$	69	38	34	$\mathbf{8.20 \cdot 10^{-4}}$	62	32	47	0.3
	ANN	25	113	16	11	$\mathbf{2.36 \cdot 10^{-19}}$	64	35	41	$\mathbf{0.0326}$				
		20	114	12	14	$\mathbf{1.43 \cdot 10^{-18}}$	71	28	41	$\mathbf{0.0288}$				
		15	106	16	18	$\mathbf{5.93 \cdot 10^{-17}}$	64	27	49	0.2731				
C45 (0.001)	RF	25	98	46	1	$\mathbf{6.73 \cdot 10^{-18}}$	91	27	27	$\mathbf{4.47 \cdot 10^{-6}}$				
		20	100	44	1	$\mathbf{3.18 \cdot 10^{-18}}$	90	23	32	$\mathbf{3.06 \cdot 10^{-6}}$				
		15	102	42	1	$\mathbf{1.40 \cdot 10^{-18}}$	91	29	25	$\mathbf{5.00 \cdot 10^{-7}}$				
	SVM	25	107	34	10	$\mathbf{1.44 \cdot 10^{-17}}$	92	33	26	$\mathbf{1.99 \cdot 10^{-7}}$	66	36	49	0.6
		20	111	33	7	$\mathbf{2.85 \cdot 10^{-18}}$	98	33	20	$\mathbf{3.27 \cdot 10^{-9}}$	72	41	38	0.2
		15	102	37	12	$\mathbf{3.60 \cdot 10^{-14}}$	85	37	29	$\mathbf{1.84 \cdot 10^{-5}}$	65	32	54	0.5
	ANN	25	118	15	9	$\mathbf{1.32 \cdot 10^{-18}}$	88	27	27	$\mathbf{7.82 \cdot 10^{-7}}$				
		20	116	20	6	$\mathbf{7.47 \cdot 10^{-21}}$	86	28	28	$\mathbf{1.28 \cdot 10^{-6}}$				
		15	111	25	6	$\mathbf{4.84 \cdot 10^{-19}}$	85	28	29	$\mathbf{7.46 \cdot 10^{-6}}$				
Ripper (2)	RF	25	108	36	2	$\mathbf{2.24 \cdot 10^{-19}}$	75	31	40	$\mathbf{2.79 \cdot 10^{-3}}$				
		20	101	40	5	$\mathbf{2.74 \cdot 10^{-16}}$	75	26	45	$\mathbf{0.0348}$				
		15	103	38	5	$\mathbf{4.45 \cdot 10^{-17}}$	77	29	40	$\mathbf{5.84 \cdot 10^{-3}}$				
	SVM	25	106	25	15	$\mathbf{3.58 \cdot 10^{-16}}$	79	33	34	$\mathbf{1.66 \cdot 10^{-3}}$	64	34	48	0.1
		20	111	21	14	$\mathbf{1.35 \cdot 10^{-17}}$	87	24	35	$\mathbf{2.47 \cdot 10^{-5}}$	67	41	38	**2.04**
		15	103	30	13	$\mathbf{1.57 \cdot 10^{-16}}$	83	30	33	$\mathbf{3.40 \cdot 10^{-4}}$	67	30	49	0.1
	ANN	25	100	17	10	$\mathbf{1.74 \cdot 10^{-17}}$	65	29	33	$\mathbf{2.70 \cdot 10^{-3}}$				
		20	101	15	11	$\mathbf{6.52 \cdot 10^{-17}}$	71	26	30	$\mathbf{1.12 \cdot 10^{-3}}$				
		15	103	14	10	$\mathbf{3.49 \cdot 10^{-16}}$	65	29	33	$\mathbf{0.0213}$				
Ripper (12)	RF	25	114	39	4	$\mathbf{1.92 \cdot 10^{-19}}$	87	25	45	$\mathbf{5.10 \cdot 10^{-4}}$				
		20	105	50	2	$\mathbf{1.36 \cdot 10^{-18}}$	79	30	48	$\mathbf{9.42 \cdot 10^{-3}}$				
		15	102	50	5	$\mathbf{2.30 \cdot 10^{-18}}$	85	22	50	$\mathbf{3.08 \cdot 10^{-3}}$				
	SVM	25	107	25	12	$\mathbf{5.74 \cdot 10^{-16}}$	82	30	32	$\mathbf{1.29 \cdot 10^{-3}}$	54	33	57	0.5
		20	108	25	11	$\mathbf{5.97 \cdot 10^{-18}}$	83	30	31	$\mathbf{2.43 \cdot 10^{-5}}$	50	35	59	0.6
		15	106	25	13	$\mathbf{1.15 \cdot 10^{-15}}$	88	27	29	$\mathbf{8.34 \cdot 10^{-5}}$	52	31	61	0.5
	ANN	25	113	17	11	$\mathbf{2.49 \cdot 10^{-16}}$	71	28	42	$\mathbf{0.0390}$				
		20	106	24	11	$\mathbf{2.42 \cdot 10^{-15}}$	72	24	45	$\mathbf{0.0190}$				
		15	109	18	14	$\mathbf{2.56 \cdot 10^{-14}}$	65	31	45	0.4190				

APPENDIX I
PROOF OF LEAST CONFIDENCE

A. A proof for continuous functions

In these first theorems, we prove that the connecting line approximation as defined in the publication indeed contains a point of least confidence under the condition that the function mapping over which we approximate is continuous. This is the case for any SVM with a Mercer kernel (due to positive definiteness) as well as the before defined ANN (since every differentiable function is continuous).

Definition I.1 Let $T \subset S$. T is *simply connected* if for every $x_1, x_2 \in T$ there exists a continuous function $\varphi : [0,1] \to S$ such that $\varphi(0) = x_1$, $\varphi(1) = x_2$ and $\forall_{x \in (0,1)} \varphi(x) \in T$.

Definition I.2 Let S be an n-dimensional Euclidean vector space. A subset $E \subset S$ is called a *boundary* if it is a simply connected $n-1$-dimensional smooth manifold.

Definition I.3 Let $x \in S$ and $V \subset S$. The distance $d(x, V)$ from x to V is defined as

$$d(x, V) := \inf_{y \in V} \|x - y\|.$$

Theorem I.4 *Let S be a Euclidean vector space (in particular $\mathbb{R}^n$) and $E \subset S$ a boundary. Then there exist $S_1 \subset S$ and $S_2 \subset S$ such that*

(i) *S_1 and S_2 are simply connected open sets,*
(ii) *$S_1 \cup E \cup S_2 = S$,*
(iii) *$S_1 \cap S_2 = \emptyset$.*

Moreover, there exists a continuous function $\pi : S \to \mathbb{R}$ such that for $x \in S$

$$\begin{cases} \pi(x) > 0 & \text{if } x \in S_1, \\ \pi(x) = 0 & \text{if } x \in E, \\ \pi(x) < 0 & \text{if } x \in S_2. \end{cases}$$

Proof: Let $x_1 \in S$ such that $x_1 \notin E$. Then, we note first that since the distance of x_1 to E is larger than 0, there exists an open environment U_{x_1} of x_1 such that $\forall_{y \in U_{x_1}} y \notin E$. Define the set $S_1 \subset S$ as the set of all $y \in S$ such that there exists a function $\varphi : [0,1] \to S$ with $\varphi(0) = x_1$, $\varphi(1) = y$, and for all $\lambda \in [0,1]$ $\varphi(\lambda) \notin E$. Note that S_1 is simply connected and open. Now we define $S_2 := S \setminus (E \cup S_1)$. Define the function π by

$$\pi : S \to \mathbb{R}, x \mapsto \begin{cases} d(x, E) & \text{if } x \in S_1, \\ 0, & \text{if } x \in E, \\ -d(x, E) & \text{if } x \in S_2. \end{cases}$$

If S_2 is simply connected (as it is trivially so for $\mathbb{R}^n$), it follows that π is a continuous function. ■

We want to assert whether any curve connecting a point from S_1 with a point of S_2 contains a point of least confidence.

Theorem I.5 *Let $x_1 \in S_1$ and $x_2 \in S_2$. Consider any continuous function $\varphi : [0,1] \to \Phi$ that satisfies $\varphi(0) = x_1$ and $\varphi(1) = x_2$. Then there exists a $\lambda \in (0,1)$ such that $x_0 := \varphi(\lambda) \in S_0$.*

Proof: Let $x_1 \in S_1$ and $x_2 \in S_2$. Consider a continuous function $\varphi : [0,1] \to \Phi$ that satisfies $\varphi(0) = x_1$ and $\varphi(1) = x_2$. Define the function $\nu : [0,1] \to \mathbb{R}$ as $\nu := \pi \circ \phi \circ \varphi$. Since φ, ϕ and π are continuous functions, ν is also continuous. Moreover, since $x_1 \in S_1$ we have that $\nu(0) > 0$ and similarly $\nu(1) < 0$. By the intermediate value theorem we know that there exists a $\lambda \in (0,1)$ such that $\nu(\lambda) = 0$ and hence $x_0 := \varphi(\lambda) \in S_0$. ■

B. A proof for Random Forests and discrete classifiers in general

Definition I.6 Let S be an n-dimensional Euclidian vector space. A *binary decision tree*, consists of a decision function f, operating on S by splitting S into k distinct regions $R^{(j)}$ such that:

(i) $R^{(j)} \subset S$
(ii) $\bigcup_j R^{(j)} = S$
(iii) $f : R^{(j)} \to \{0,1\}$

Definition I.7 A *Binary Random Forest* is an ensemble of K binary decision trees f_k with associated aggregate classification function $g : S \to \{0,1\}$, defined as:[10]

$$g : \mathbf{x} \mapsto \begin{cases} 1 & \text{if} \quad \#\{j | f_j(\mathbf{x}) = 1\} > \#\{j | f_j(\mathbf{x}) = 0\} \\ 0 & \text{otherwise .} \end{cases}$$

Definition I.8 We say that the prediction of an input vector $\mathbf{x}$ has *maximal agreement* if more than $\lfloor K/2 \rfloor + 1$ of the K trees support that prediction (i.e. for more than $\lfloor K/2 \rfloor + 1$ trees the prediction $f_k(\mathbf{x})$ is equal to that of the target prediction). The intersection A of all of the cuboid regions that contain the input vector and agree with the prediction is called the (cuboid) *subspace of maximal agreement*.

[10]We can assume k to be an even number without loss of generality since, we can always enforce this by adding a placebo tree that agrees with the majority vote and picks a class in case of equal evidence for either class.

By definition, each point $\mathbf{x}_i$ has an associated subspace of maximal agreement A_i which is formed by the union of at least $\lfloor K/2 \rfloor + 1$ regions for which f_k is the maximal agreement vote. Since the subspace of maximal agreement is the intersection of cuboid regions, it must be a cuboid region itself (possibly of lower dimension).

Lemma I.9 *Let g be a Binary Random Forest g and $\mathbf{x}_i$, $\mathbf{x}_j$ two points for which $g(\mathbf{x}_i) \neq g(\mathbf{x}_j)$. The intersection of the subspaces of maximal agreement associated with these points is empty for a pair of input vectors with different prediction outcome: $A_i \cap A_j = \emptyset$.*

Proof: This follows immediately from Definition I.8. If this were not the case for a certain choice of i and j, this would imply that the agreement for both is greater than $\lfloor K/2 \rfloor + 1$ which is impossible since there are only K tree decision values. ■

Lemma I.10 *Given a boundary E that divides a vector space into two opposite half-spaces S_1 and S_2. The curve connecting any two points lying in opposite half-spaces will intersect the boundary E.*

Proof: This follows immediately from the (Hyper)Plane Separation Postulate in Euclidian spaces. ■

Theorem I.11 *Let g be a Binary Random Forest g and $\mathbf{x}_i$,$\mathbf{x}_j$ two points for which $g(\mathbf{x}_i) \neq g(\mathbf{x}_j)$. Any connected subspace that contains both $\mathbf{x}_i$ and $\mathbf{x}_j$ must go through a point of no agreement.*

Proof: Without loss of generality, we can choose one point $\mathbf{x}_i$ of class 0 and one point $\mathbf{x}_j$ of class 1. Due to Lemma I.9 these must lie in disjunct cuboid regions A_i and A_j. Since A_i and A_j are two convex, non-empty sets, the connected subspace defined by the convex combination of $\mathbf{x}_i$ and $\mathbf{x}_j$ (i.e. the line segment) must contain at least one point $\mathbf{x}_0$ that is contained within the space outside of A_i and A_j : $\mathbf{x}_0 \in \overline{A_i \cup A_j}$.

Let us consider the aggregation of all cuboid regions for which the predicted value is 0 and 1 respectively called A_0 and A_1. We know due to Lemma I.9 that these must be disjunct. Due to the connectedness of the subspace, we know that it must contain at least one point $\mathbf{x}_0$ that is contained within a region of the space outside of A_0 and A_1 : $\mathbf{x}_0 \in \overline{A_0 \cup A_1}$. Since for this point:

(i) $\mathbf{x}_0 \notin A_0 \implies \#\{j | f_j(\mathbf{x}_0) = 0\} < \lfloor K/2 \rfloor + 1$
(ii) $\mathbf{x}_0 \notin A_1 \implies \#\{j | f_j(\mathbf{x}_0) = 1\} < \lfloor K/2 \rfloor + 1$

Given that there are exactly K trees, it must follow that support for both is equal to $\lfloor K/2 \rfloor$, i.e. $\mathbf{x}_0$ lies within a boundary region for which no agreement can be reached. ■