The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

MERCS: Multi-Directional Ensembles of Regression and Classification Trees

Elia Van Wolputte, Evgeniya Korneva, Hendrik Blockeel
KU Leuven, Department of Computer Science, 3001 Leuven, Belgium
{elia.vanwolputte, evgeniya.korneva, hendrik.blockeel}@cs.kuleuven.be

Abstract

Learning a function $f_{\mathbf{X}\to\mathbf{Y}}$ that predicts $\mathbf{Y}$ from $\mathbf{X}$ is the archetypal Machine Learning (ML) problem. Typically, both sets of attributes $(\mathbf{X}, \mathbf{Y})$ have to be known before a model can be trained. When this is not the case, or when functions $f_{\mathbf{X}\to\mathbf{Y}}$ are needed for varying $\mathbf{X}$ and $\mathbf{Y}$, this may introduce significant overhead (separate learning runs for each function). In this paper, we explore the possibility of omitting the specification of $\mathbf{X}$ and $\mathbf{Y}$ at training time altogether, by learning a multi-directional, or *versatile* model, which will allow prediction of any $\mathbf{Y}$ from any $\mathbf{X}$. Specifically, we introduce a decision tree-based paradigm that generalizes the well-known Random Forests approach to allow for multi-directionality. The result of these efforts is a novel method called **MERCS**: **M**ulti-directional **E**nsembles of **R**egression and **C**lassification tree**S**. Experiments show the viability of the approach.

Introduction

Learning predictive functions from data is one of the most common tasks in machine learning or data mining. The task can be described as follows: given a dataset D with instances of the form $(\mathbf{x}, y)$ with $\mathbf{x} \in \mathcal{X}$ and $y \in \mathcal{Y}$, learn a function $f : \mathcal{X} \to \mathcal{Y}$ that can be used to predict the value of y, given an instance $\mathbf{x}$. Usually, these instances are described using a fixed set of variables: a set of variables $\mathbf{X} = \{X_1, \ldots, X_m\}$ called the input variables, and an output variable Y. If Dom maps a variable onto its domain, then $\mathcal{X} = Dom(\mathbf{X}) = \prod_{i=1}^{m} Dom(X_i)$ and $\mathcal{Y} = Dom(Y)$.

This predictive learning task is easily extended to the case where more than one target variable needs to be predicted; instead of a nominal or scalar value y, f then predicts a tuple $\mathbf{y} \in \mathcal{Y} = \prod_{i=1}^{k} Dom(Y_i)$ with k the dimensionality of the output space. This is sometimes called multi-target prediction or multivariate prediction; variants include multi-label prediction (Madjarov et al. 2012) and structured output prediction (Kocev et al. 2013).

In this paper, we consider a related but more general problem: given a dataset D with instances described by a set of variables $\mathbf{A} = \{A_1, \ldots, A_m\}$, learn a model M such that, for *any* subsets $\mathbf{X} \subseteq \mathbf{A}$ and $\mathbf{Y} \subseteq \mathbf{A}$, M can be used to predict $\mathbf{Y}$ from $\mathbf{X}$. We will call such a model a *versatile* model.

Definition 1 (Versatile model) *A* ***versatile model*** *is a model $M(\mathbf{A})$, learned from a dataset D, that can be used to predict any set of variables $\mathbf{Y} \subseteq \mathbf{A}$ from any other set $\mathbf{X} \subseteq \mathbf{A}$.*

The ability to omit this specification of $\mathbf{X}$ and $\mathbf{Y}$ at training time has a wide variety of potential applications, e.g.:

- **Imputation**: assume new instances are given with some attribute values missing, and the attributes missing vary from one instance to another. The missing values need to be filled in by the model M. This kind of missing value imputation is closely related to tasks such as pattern completion, e.g., image completion in vision (Drori, Cohen-Or, and Yeshurun 2003).
- **Specialization**: assume $\mathbf{X}$ and $\mathbf{Y}$ are constant, but only known at prediction time. One could learn the function $f_{X\to Y}$ from the data at that time, but this may be slow, and it requires that the training data are still available. If a versatile model M was learned in advance, this can be used to efficiently derive $f_{X\to Y}$ from it, without needing to re-access data.
- **Anomaly detection**: assume a new instance is given, with all attributes known, and the task is to determine if this instance is anomalous; that is, do some of the values in the instance differ from what one would expect after inspecting the earlier data D? (Aggarwal 2015) This can be tested by predicting each attribute value and comparing it to the observed value.

Furthermore, versatile models are especially useful in the context of big or streaming data, where learning many different, specialized, predictive models can be slow. Here, it is particularly desirable that versatile model learners scale well in terms of model size, training time, and prediction time.

In this paper, we present a versatile model learner that is primarily based on decision tree learning. Decision trees, and ensembles of them (*forests*), are known to have excellent scalability both in terms of learning and prediction (Domingos and Hulten 2000). However, their learning algorithms assume a fixed set of input and output variables.

	ANN	NN	PGM	MERCS
Scalable learning	-	+	-	+
Scalable inference	+	-	-	+
Interpretability	-	+	+	+
Data summarization	+	-	+	+
Nominal/numerical	-	+	-	+

Table 1: MERCS combines desirable properties not encountered simultaneously in other approaches.

The main contribution of this work is that we propose a method, MERCS, which constructs an ensemble of trees of which the input and output variables vary, to the extent that the ensemble contains enough trees to allow for predictions of any $\mathbf{Y}$ from any $\mathbf{X}$. Additionally, the benefits of decision trees carry over to the MERCS model, ensuring efficient learning and fast predictions.

In the remainder of the paper, we first discuss possible alternative approaches to versatile model learning. We next describe the MERCS approach in detail, and present an experimental comparison with Bayesian networks, which are among the most closely related methods in terms of versatility. The results confirm the usefulness of MERCS for learning versatile models from big data.

Related Work

While the term *versatile models* is not canonical nomenclature, the concept itself is not entirely new. Nearest neighbor methods and neural networks can under certain circumstances be used as versatile models, and probabilistic graphical models (PGMs) constitute versatile models in the true sense of the word, by virtue of the joint probability distribution that they represent. Each of these approaches has its strengths and weaknesses, but the MERCS approach combines a set of desirable properties that are not encountered simultaneously in any of these competitors, as discussed below, and summarized in Table 1.

Probabilistic graphical models (PGMs) can be used to estimate the probability distribution $p(\mathbf{Y}|\mathbf{X})$ for any set of variables $\mathbf{Y}$ and $\mathbf{X}$. A PGM represents a joint distribution over all the variables. From this joint distribution, any conditional or marginal distribution can be derived, and hence any expected value, most probable value, etc. can be computed. While PGMs are arguably the theoretically best-founded approach, they are computationally expensive.

Learning a PGM is relatively easy if the structure of the PGM is given and only parameters need to be learned, but learning the structure itself is challenging. It often involves a search through possible structures, where for each candidate structure the parameters need to be learned and its fit to the data evaluated.

Additionally, inference with a PGM is NP-hard: its complexity is exponential in the tree-width of the graph. For this reason, approximate inference is often used; but even approximate inference is NP-hard, if guarantees about the accuracy of the inference are desired (Neapolitan 2004).

Most PGMs do not naturally handle continuous variables; such variables are usually discretized for that reason.

Among PGMs, probabilistic dependency networks (Heckerman et al. 2000) are closest to MERCS, in the sense that they also employ decision trees. However, inference with them is still probabilistic, and suffers from the computational complexity that this entails.

We should note that, while probabilistic inference is inherently hard, in many cases it is in some sense overkill: if the aim is simply to predict a variable with high accuracy, without requiring the exact probability that this prediction is correct, probabilistic inference is not needed. This motivates research into non-probabilistic versatile models: wherever PGMs are used, MERCS can offer a scalable alternative, minus probabilistic information.

Nearest neighbor methods (NN) can, in principle, be used to fill in any missing value in exactly the same way they are used to predict class labels. In this case the model M is the input dataset D itself. However, there are at least two issues in the multi-directional setting. First, NN methods typically require feature weighting to attain good performance in high-dimensional spaces. Unfortunately, it is not clear how a set of weights could be found that works well regardless of what variable is to be predicted. Second, prediction can be slow because it requires a search through the data D at prediction time. A straightforward solution to this is indexing, but it is again not obvious how this translates to a multi-directional setting, i.e.: indexing the data in such a way that the k nearest neighbors in *any* subspace $\mathbf{X}$ of the original space can be found efficiently.

In conclusion, the desired multi-directionality introduces highly non-trivial bottlenecks in the nearest neighbors paradigm, which are, to our knowledge, as yet unsolved.

Artificial neural networks (ANN) can be constructed that have as many outputs as there are inputs, and that try to learn the identify function. These are known as auto-encoders. Such networks are often used for pattern completion. The classical disadvantages of neural networks also apply here, i.e.: notoriously long training times, the need for enormous training sets, high demands on computing power and most importantly a lack of interpretability.

MERCS

MERCS stands for **M**ulti-directional **E**nsembles of **R**egression and **C**lassification tree**S**. Compared to traditional ensembles of decision trees, the key difference is in the term "multi-directional".

Let us denote a single decision tree with input attributes $\mathbf{X}$ and output attributes $\mathbf{Y}$ as $T(\mathbf{X}, \mathbf{Y})$. Though standard decision trees are often assumed to have a single target attribute, variants with multiple targets have also been developed (Blockeel, De Raedt, and Ramon 2000). The term "decision tree" here refers to trees that can have multiple targets.

Learning decision trees is typically almost linear in dataset size[1]. Furthermore, numerical and nominal attributes are naturally dealt with, and decision trees are readily interpretable.

Aggregating multiple trees in ensembles is known to yield significant improvements in predictive performance (Breiman 2001). Moreover, the attractive properties of single trees are preserved (Criminisi and Shotton 2013), though interpretability is somewhat lessened. A traditional ensemble can be written as,

$$E(\mathbf{X},\mathbf{Y}) = \{\mathbf{T_i}(\mathbf{X},\mathbf{Y}) | \mathbf{X},\mathbf{Y} \in \mathcal{P}(\mathbf{A}) \setminus \emptyset \wedge \mathbf{X} \cap \mathbf{Y} = \emptyset\} \tag{1}$$

To obtain a versatile model (def. 1), one more generalization step is required. The sets of attributes, $\mathbf{X}, \mathbf{Y}$ in Eq. 1 are still fixed, all trees use the same $\mathbf{X}$ and $\mathbf{Y}$. It is a rather straightforward affair to postulate a more general ensemble that allows for multi-directionality. Consider the ensemble $M(\mathbf{A})$ in which attribute sets $\mathbf{X^i}$ and $\mathbf{Y^i}$ are allowed to differ from tree to tree,

$$M(\mathbf{A}) = \{\mathbf{T_i}(\mathbf{X^i},\mathbf{Y^i}) | \mathbf{X^i},\mathbf{Y^i} \in \mathcal{P}(\mathbf{A}) \setminus \emptyset \wedge \mathbf{X^i} \cap \mathbf{Y^i} = \emptyset\} \tag{2}$$

By no longer limiting the component trees of an ensemble to one specific prediction task, we get a multi-directional model, one that allows predictions in multiple directions. The elephant in the room here is the number of possible prediction tasks,

$$N_{Preds} = 3^m - 2^{m+1} + 1 \tag{3}$$

which increases exponentially with the total number of attributes m in the attribute set $\mathbf{A}$. Constructing an ensemble that includes for *each* possible prediction task $\mathbf{X} \rightarrow \mathbf{Y}$ one or more trees $T_i(\mathbf{X},\mathbf{Y})$ is infeasible for all but the most trivial cases. We need to construct ensembles that will *not* contain a tree for each possible task. This leads to the following two overarching research themes:

T1: How do we select which trees $T_i(\mathbf{X^i},\mathbf{Y^i})$ should constitute the ensemble? This mostly boils down to the question: how do we select $\mathbf{X^i}$ and $\mathbf{Y^i}$ (the construction of the tree itself uses a standard algorithm).

Several research questions arise: do we predict every attribute individually, or do we predict sets of attributes together? The latter could be more efficient, but what is the trade-off between efficiency and performance? Is there a way to decide which attributes to be predicted jointly, or does a random selection work as well? These questions will be tackled by the **selection algorithms**.

T2: How do we predict with such an ensemble? When the ensemble does not contain any trees that can be used directly for predicting $\mathbf{Y}$ from $\mathbf{X}$ (i.e., there are no trees $T_i(\mathbf{X}^i,\mathbf{Y}^i)$ such that $\mathbf{X}^i \subseteq \mathbf{X}$ and $\mathbf{Y} \subseteq \mathbf{Y}^i$), how can we combine the trees in the ensemble to obtain a prediction? This issue will be tackled by the **prediction algorithms**. Their purpose is to ensure the proper multi-directionality at inference time, generalizing aggregating techniques of traditional unidirectional ensembles.

The following two sections discuss the selection and prediction algorithms proposed in this paper.

Selection Algorithms

Given a dataset D with attributes $\mathbf{A} = \{A_1, A_2, \ldots, A_m\}$, the most naive way to build a versatile tree-based model is to include a separate decision tree for every possible combination of the input and target variables in the ensemble. As argued before, this does not scale (Eq. 3).

A better strategy is to learn for each separate target A_i a single-target tree $T(\mathbf{A} \setminus \{A_i\}, \{A_i\})$. This yields m trees. Given a target set $\mathbf{Y}$, each $A_i \in \mathbf{Y}$ is then predicted using a separate tree in the ensemble. When a tree requires attribute values that are not available (i.e., one of the tests in T is not in $\mathbf{X}$), it is treated as a missing value; this can be done using standard techniques for handling missing values. In this way, each variable is predicted using a single tree, which may or may not rely on many missing values. By learning an ensemble of k trees, rather than a single tree, for each target, a multi-directional ensemble of km trees is obtained.

For large m and typical k (e.g., $k = 30$), the approach above still yields a large model. The use of multi-target trees can reduce its size. If the m attributes are partitioned into m/p disjoint subsets of p targets, and an ensemble of k multi-target trees is learned for each subset, the resulting model contains $\frac{km}{p}$ trees, rather than km.

Additionally, such a partitioning could be made q times. For each individual target, the number of trees predicting it is then kq. Using different partitions introduces additional variety among the learned trees: when A_i is predicted by multiple trees, the trees differ not only due to data resampling (as in bagging) and attribute subsampling (as in random forests), but also due to variation in A_i's co-targets. Thus, increasing q introduces more variety than increasing k.

The selection strategy used in this paper is the following: construct q random partitions of $\mathbf{A}$; for each partition $P_i = \{\mathbf{Y}_{i,1}, \mathbf{Y}_{i,2}, \ldots, \mathbf{Y}_{i,\frac{m}{p}}\}$, learn for each $\mathbf{Y}_{i,j}$ k trees $T(\mathbf{A} \setminus \mathbf{Y}_{i,j}, \mathbf{Y}_{i,j})$. Thus, the total number of trees included in the model is $\frac{m}{p}kq$. While m is given, the quantity $\frac{kq}{p}$ can be influenced and affects scalability. Given some value for $\frac{kq}{p}$, the question is what values for k, p and q are optimal.

Prediction Algorithms

Multi-directionality implies sacrificing our ability to tailor the ensemble towards specific prediction tasks. Considering the sheer amount of those (Eq. 3), maybe not even a single component tree is a perfect match with the

[1]Typically $\mathcal{O}(mn \log n)$ for a dataset with n tuples and m attributes.

required task, $f_{\mathbf{X}\rightarrow\mathbf{Y}}$. As a result, performing inference with multi-directional ensembles becomes more intricate than its unidirectional counterpart, mainly because the straightforward aggregation procedures of the latter[2] no longer apply.

Difficulties arise when $f_{\mathbf{X}\rightarrow\mathbf{Y}}$ "matches poorly" with the predictive functions $f_{\mathbf{X^i}\rightarrow\mathbf{Y^i}}$ explicitly expressed by the component trees, $T(\mathbf{X^i}, \mathbf{Y^i})$. This discrepancy can occur in both target ($\mathbf{Y}$) and descriptive ($\mathbf{X}$) attributes, which we will discuss separately.[3]

First, assume that all the matching issues stem exclusively from the target attributes. Here, if suffices to aggregate the predictions of all *relevant* models (i.e. $\{T(\mathbf{X^i}, \mathbf{Y^i}) | \mathbf{Y^i} \cap \mathbf{Y} \neq \emptyset\}$). Second, assume that not a single model in the ensemble has all its descriptive attributes $\mathbf{X^i}$ available. The prediction algorithm revolves mainly around handling these missing descriptive attributes $\mathbf{X^i}$, and does so by means of two strategies.

The first strategy, *missing value imputation*, directly addresses this issue in the component models. The second, *model activation*, determines the most appropriate models. Both approaches can -up until a certain point- be used simultaneously, but as one shifts the emphasis from one method to another, the behavior of the resulting algorithm changes qualitatively. It is this change of emphasis that underlies the two prediction strategies implemented in the MERCS system.

S1: Missing value Imputation (MI). This straightforward prediction strategy first imputes all missing descriptive attributes of all relevant component models (i.e. $\{T(\mathbf{X^i}, \mathbf{Y^i}) | \mathbf{Y^i} \cap \mathbf{Y} \neq \emptyset\}$). Afterwards, the algorithm continues by aggregating all their predictions, treating all models equally.

S2: Model Activation (MA). Whereas the MI-prediction strategy uses every relevant model, MA-prediction is more restrictive. First, it determines the *most appropriate* models for the task at hand, and only when necessary resorts to imputation of missing descriptive attributes. The appropriateness criterion is defined as the ratio of available vs. missing descriptive attributes of the component models:

$$C_{app} = \frac{|\mathbf{X^i} \cap \mathbf{X}|}{|\mathbf{X^i}|} \tag{4}$$

This algorithm thus favors knowledge present in the ensemble over ad hoc fixes (i.e. imputation) in the component models directly. Additionally, the threshold T (Alg. 1) that C_{app} has to exceed to determine appropriateness can vary, as opposed to the MI-prediction algorithm which always uses every relevant model.

Both prediction strategies can be captured in algorithm 1. The switch between both strategies is controlled by the `canModelAct` method: herein lies the judgment of the inclusion of a certain component model. The more restrictive this method is, the more we prioritize the selection of appropriate models to rectifying deficiencies in the component models (i.e. MA-prediction). Conversely, if this method selects every model that is relevant (i.e. $\{T(\mathbf{X^i}, \mathbf{Y^i}) | \mathbf{Y^i} \cap \mathbf{Y} \neq \emptyset\}$), the algorithm implements the MI-prediction strategy.

Algorithm 1 Prediction algorithm

```
Input: $descAtts, targAtts = \mathbf{X}, \mathbf{Y}$
    $allModels = M$
    $input = \mathbf{x}$
    $T, \alpha =$ Threshold, Stepsize
Output: $result = \mathbf{y}$
 1: $activeModels \leftarrow \emptyset$
 2: $result \leftarrow \emptyset$
 3: while $activeModels = \emptyset$ do
 4:     for all $m \in allModels$ do
 5:         if canModelAct(m, descAtts, targAtts, T) then
 6:             $activeModels \leftarrow m$
 7:         end if
 8:     end for
 9:     $T \leftarrow T - \alpha$
10: end while
11: for all $m_a \in activeModels$ do
12:     if hasMissingDesc($m_a$, descAtts) then
13:         imputeMissingDesc($m_a$, $\mathbf{x}$, descAtts)
14:     end if
15: end for
16: for all $y \in targAtts$ do
17:     $p$=predict(y,$\mathbf{x}$, activeModels)
18:     $result \leftarrow p$
19: end for
```

Experiments

This section is divided in three parts. First, we study the selection strategies. Second, we investigate the effects of the different prediction strategies and third, we compare MERCS to an external baseline, in this case PGMs.

As argued in the related work section, we consider PGMs as the most natural competitor to the MERCS system. We rely on the `SMILE` library[4] to learn the structure and parameters of these networks, as well as for performing inference afterwards. This `SMILE` library underlies the commercial `BayesFusion` system, and is considered a popular[5] and powerful tool for Bayesian networks, motivating its role as a meaningful external baseline.

[2]E.g. a weighted majority vote in classification or an averaging procedure in regression.

[3]This is merely an assumption for the sake of simplicity. Since these are disjoint sets, both cases can be discussed separately without loss of generality.

[4]Cf. download.bayesfusion.com.

[5]For an extensive list of other publications where this system is used, cf. bayesfusion.com/publications.

Dataset	# instances n	# attributes m
msnbc	291326	17
adult	30718	9
plants	17412	68
nltcs	16181	16
netflix	15000	100
audio	15000	100
nursery	12960	9
accidents	12758	110
pumsb_star	12262	163
jester	9000	100
dna	1600	180
voting	1214	1358

Table 2: Summary of datasets used in this study.

Next to typical evaluation criteria such as speed and predictive performance, we will also gauge MERCS' versatility. To do so, we evaluate MERCS on a variety of prediction tasks ($f_{\mathbf{X^j} \rightarrow \mathbf{Y^j}}$), to which from this point on, we will refer to as *queries*.

Evidently, this makes the query-generating mechanism an important part of our experiments. We generate queries with a varying number of known attributes, as follows. Initially, $\mathbf{X^j}$ contains all attributes except $\mathbf{Y^j}$. We then consecutively drop 9% of the attributes at random, leading to an $\mathbf{X^j}$ that contains $100\%, 91\%, \dots, 9\%$ of the attributes (i.e., 11 different versions). Note that these queries correspond to prediction tasks with increasing amounts of missing values. We only consider single-target queries ($|\mathbf{Y^j}| = \mathbf{1}$), and repeat this procedure on every dataset for 50 starting queries. Eventually, this yields 550 queries per dataset.[6] With these queries we assess both speed and macro F1-score (averaged over all queries), as well as gain some insight regarding the actual versatility of our model.

The datasets used in the following experiments are summarized in Table 2. All datasets previously occurred in studies on density estimation[7] (also inherently a multi-directional setting), except for nursery and adult, which were lifted from the UCI[8] repository.

Selection: Internal Evaluation

In this part of the experiments, we varied the parameter p of the selection strategy in order to investigate how the usage of multi-target trees affects model performance, as well as its impact upon induction and inference times. Specifically, for each dataset in Table 2, a MERCS model composed of single target trees ($p = 1$) is learned. These are then compared to their counterparts composed of multi-target decision trees ($p \in \{2, 4, 6, 8, 10\}$).

[6]In adult & nursery, only 400 queries are generated. This is due to the limited number of attributes in those datasets.

[7]github.com/UCLA-StarAI/Density-Estimation-Datasets

[8]archive.ics.uci.edu/ml

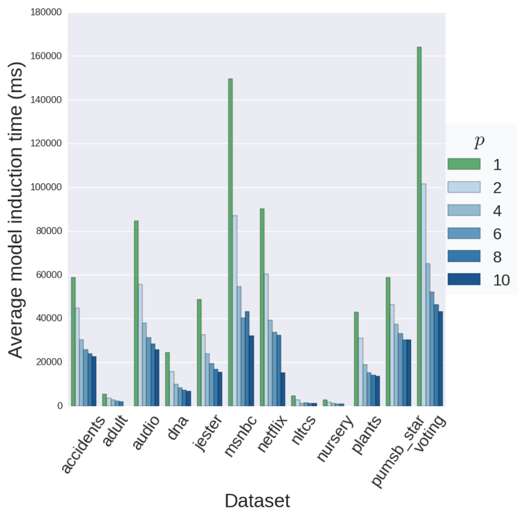

Figure 1: Average model induction time. By relying on multi-target trees ($p > 1$), one can significantly reduce the induction time.

As can be seen from Fig. 1, the multi-target setting makes it possible to significantly reduce model learning time. The gain is especially high for the datasets with a large number of attributes and/or instances (e.g., voting and msnbc). At the same time, predictive performance of an ensemble of multi-target trees model often stays approximately at the same level as in the single-target setting (Fig. 2).

This observation offers an interesting direction for future work. It has been shown (Kocev et al. 2013) that a multi-target tree can be more accurate than a set of single-target trees, since it exploits the information related attributes carry about each other. In the current version of MERCS, target attributes are still grouped randomly. However, if attributes interdependencies are taken into account while constructing multi-target trees, there may be a chance to increase the predictive performance of the resulting ensemble while reducing the learning time. To that end, a more sophisticated selection strategy is needed.

Prediction: Internal Evaluation

The MA-prediction strategy assumes that using only the most appropriate trees in the ensemble might be preferable to relying exclusively on imputation. But our default selection strategy is such that each attribute appears in every component tree, either as target or as descriptive attribute. Therefore we do not expect significant advantages from using the MA-prediction strategy; all models will be more or less equally appropriate anyway.

To experimentally discern the differences between our two approaches to prediction, we introduce an extra variation of the selection strategy, i.e. *random selection*. Its behavior is exactly what the name suggests: it builds random

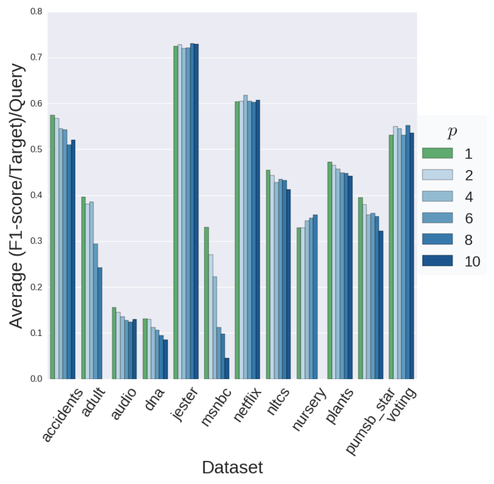

Figure 2: Average macro F1-score per query. Increasing the number of targets typically has a limited impact on the predictive performance.

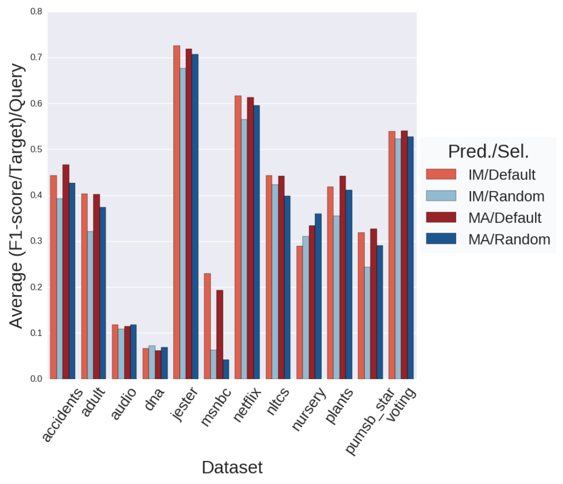

Figure 3: Average macro F1-score per query compared between prediction strategies. If the model was built using the random selection strategy (indicated in blue), MA-prediction is typically the better choice.

decision trees, while ensuring that all attributes do appear as targets in the process. The critical point is that now, component trees do not have all the non-target attributes as descriptive attributes. This means more variation in C_{app}, which should lead to MI-prediction and MA-prediction behaving differently.

Figure 3 confirms this hypothesis; the MA-prediction strategy is preferable when the MERCS ensemble consists of trees generated by the random selection strategy. MA-prediction is thus beneficial for those cases when e.g. time or memory constraints preclude a MERCS model composed of decision trees containing all attributes, otherwise MI-prediction appears equally good.

External Comparison

In the end, what matters most is how MERCS' performance compares to other approaches.[9] In this study, we opted for PGMs to serve as an external baseline. Just as in the internal evaluations, we will focus on the induction and inference times on the one hand, and predictive performance, measured in macro F1-score on the other hand. The results are summarized in figures 4, 5 and 6.

In terms of speed, the most striking result is undoubtedly to be found in the inference times (Fig. 4). Here, MERCS outperforms the PGMs in the `BayesFusion` implementation. This is to some extent unsurprising: inference with PGMs is well-known to be hard. Nevertheless, the sheer scale of the improvement, often approaching 3 orders of magnitude, is notable. De facto this means that in cases where using PGMs is impossible due to time constraints, MERCS may offer a viable alternative.

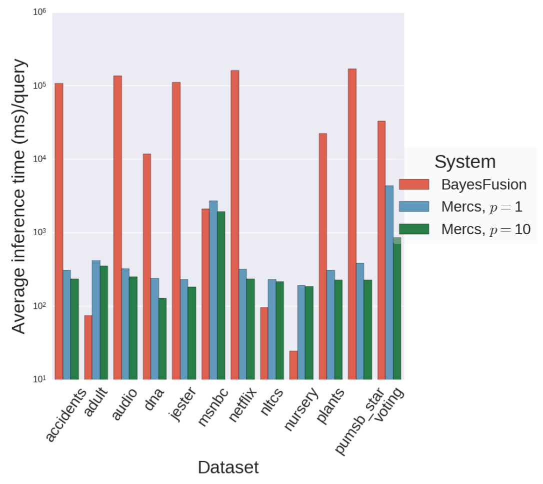

Figure 4: Average inference time per query for PGMs and MERCS. MERCS outperforms PGMs by several orders of magnitude in 8 cases.

In induction times, both systems are much more at par (Fig. 5), except when many attributes (e.g. the `voting` dataset) come into play: there, MERCS is clearly faster. Moreover, the induction times of the MERCS system can further be reduced by using multi-target trees.

Finally, when it comes to predictive performance (measured in macro F1-score), results are ambiguous (Fig. 6). MERCS loses from the `BayesFusion` system in 7 cases and outperforms its competitor in 5. We note that PGMs are

[9] In doing so, both for `BayesFusion` and MERCS we rely on default settings as much as possible. Concretely, for `BayesFusion` this means greedy thick thinning for structure learning and the clustering algorithm for inference. For MERCS, we rely on the default SciKit-Learn implementation of decision trees.

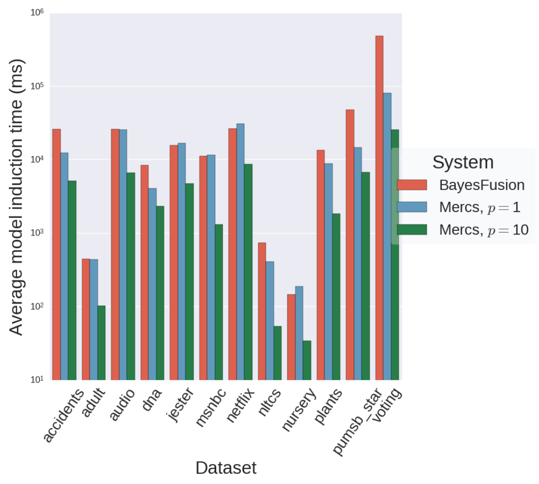

Figure 5: Average induction times for PGMs and MERCS. Both approaches appear equivalent, however there is an indication that MERCS scales better when many attributes are involved, e.g. the `voting` dataset.

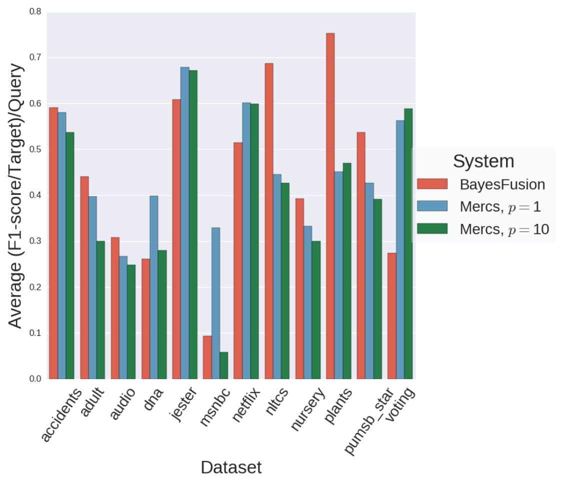

Figure 6: Average macro F1-score per query for PGMs and MERCS. Result are ambiguous, showing neither of both approaches to be clearly superior.

(arguably) theoretically the best-founded approach and consistently outperforming such a theoretically optimal competitor is not a realistic expectation.

Conclusions

In this paper we introduced MERCS, a novel class of tree-based models that allow for multi-directional reasoning. The main strength of MERCS approach is its ability to unify a set of attractive characteristics (i.e. scalability, interpretability, data summarization and easy handling of both nominal and numerical data) that are not encountered simultaneously in its competitors.

Experimental findings show MERCS' high potential. Combining unidirectional ensembles of decision trees by means of the selection strategy, one can learn a multi-directional model of data at least as fast as a PGM. Employing multi-target trees, especially in large problems, can significantly speed up the learning stage, while having a limited impact on the quality of the obtained predictions. In terms of predictive accuracy, neither MERCS nor PGMs managed to manifest itself as clearly superior. However, both methods differ strongly with respect to the speed with which predictions are made: MERCS is often several orders of magnitude faster at inference time (Fig. 4). This entails that MERCS is still applicable when PGM-based approaches are impossible to use in practice.

Our experimental results are particularly promising in light of the fact that, for practical reasons, we made our comparison in the application domain where PGMs are expected to excel (i.e.: moderately-sized datasets with only nominal attributes), and we compared a relatively basic implementation of MERCS to a state-of-the-art implementation of PGMs.

There are many ways in which MERCS can be improved or extended. The current implementation is based on SciKit-Learn. A special-purpose implementation of MERCS might be faster, and would make the efficiency comparison with BayesFusion (which is already in MERCS' favor) more fair. Other possible improvements include better selection or prediction algorithms, and an incremental version of MERCS (allowing for any-time prediction).

Acknowledgments

Research supported by ERC (ERC-ADG-2015, project 694980, SYNTH), Research Foundation - Flanders (project G079416N, MERCS), and KU Leuven (GOA/13/010).

References

Aggarwal, C. C. 2015. Outlier analysis. In *Data mining*, 237–263. Springer.

Blockeel, H.; De Raedt, L.; and Ramon, J. 2000. Top-down induction of clustering trees. *arXiv preprint cs/0011032*.

Breiman, L. 2001. Random forests. *Machine learning* 45(1):5–32.

Criminisi, A., and Shotton, J., eds. 2013. *Decision Forests for Computer Vision and Medical Image Analysis*. London: Springer London. DOI: 10.1007/978-1-4471-4929-3.

Domingos, P. M., and Hulten, G. 2000. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, Boston, MA, USA, August 20-23, 2000*, 71–80.

Drori, I.; Cohen-Or, D.; and Yeshurun, H. 2003. Fragment-based image completion. In *ACM Transactions on graphics (TOG)*, volume 22, 303–312. ACM.

Heckerman, D.; Chickering, D. M.; Meek, C.; Rounthwaite, R.; and Kadie, C. 2000. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research* 1(Oct):49–75.

Kocev, D.; Vens, C.; Struyf, J.; and Deroski, S. 2013. Tree ensembles for predicting structured outputs. *Pattern Recognition* 46(3):817–833.

Madjarov, G.; Kocev, D.; Gjorgjevikj, D.; and Džeroski, S. 2012. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition* 45(9):3084 – 3104.

Neapolitan, R. E. 2004. *Learning bayesian networks*, volume 38. Pearson Prentice Hall Upper Saddle River, NJ.