Hierarchical Meta-Rules for Scalable Meta-Learning

Quan Sun and Bernhard Pfahringer

Department of Computer Science
The University of Waikato
Hamilton, New Zealand
quan.sun.nz@gmail.com, bernhard@cs.waikato.ac.nz

Abstract. The Pairwise Meta-Rules (PMR) method proposed in [18] has been shown to improve the predictive performances of several meta-learning algorithms for the algorithm ranking problem. Given m target objects (e.g., algorithms), the training complexity of the PMR method with respect to m is quadratic: $\binom{m}{2} = m \times (m-1)/2$. This is usually not a problem when m is moderate, such as when ranking 20 different learning algorithms. However, for problems with a much larger m, such as the meta-learning-based parameter ranking problem, where m can be 100+, the PMR method is less efficient. In this paper, we propose a novel method named Hierarchical Meta-Rules (HMR), which is based on the theory of orthogonal contrasts. The proposed HMR method has a linear training complexity with respect to m, providing a way of dealing with a large number of objects that the PMR method cannot handle efficiently. Our experimental results demonstrate the benefit of the new method in the context of meta-learning.

1 Meta-Learning

It is commonly accepted in the machine learning community that each algorithm has its own specific strengths and weaknesses or a restricted hypothesis space bias. This is due to the assumptions any algorithm must make in order to learn a model for a given dataset. This phenomenon has also been confirmed by a series of empirical studies [20, 1, 5]. There is no single best algorithm to be used in all problems since most algorithms have an inductive bias, theoretically due to the No Free Lunch theorem [21].

The choice of an algorithm and its parameter settings for a given dataset is guided by the performance estimation methodology that an analyst uses. One common practice is to use the “trial and error” strategy. Although feasible, this strategy may still require a reasonable amount of computing time, especially when there are many algorithms available. Also, in a business or industrial environment, the end-users of machine learning techniques are not necessarily machine learning experts. Therefore, the choice of which algorithm(s) to use depends on the dataset at hand, and systems that can provide such recommendations would be very useful.

Meta-learning uses a general machine learning approach to generate meta-knowledge that maps the characteristics of a dataset, captured by meta-features, to the relative performances of the available algorithms. The advantage of meta-learning is that high-quality algorithm or parameter ranking can be done on the fly. This promise is particularly important for business domains that require rapid deployment of analytical techniques.

The next question is "Which type of recommendation should a meta-learning system provide to the end-user?" In this paper we follow the reasons and motivations described in [4, 3, 17]: "... when searching a topic on the Web, one may investigate several links. The same can also apply to a data analysis task if enough resources are available to try out more than one algorithm. Since we do not know how many algorithms the user might actually want to select, we provide a ranking of all the algorithms...".

We first introduce the mechanics of meta-learning. The basic steps of constructing a meta-dataset for a meta-learning task are as follows: (a) a set of datasets is collected; (b) define some meta-features as the characteristics of each dataset, e.g., the number of instances, the number of numeric or categorical features, and many more; (c) estimate the predictive performance of the available algorithms, e.g., using cross-validation, for every dataset in the dataset collection.

Thus, for each dataset we get a list of available algorithms with their performance estimates. Given the above information, we can construct a meta-dataset, which is a $n \times v$ matrix, where $v = u + m$. Here, v is the sum of the number of meta-features u and the number of algorithms m, and n is the number of datasets. Below is an example dataset, where $n = 3$, $u = 3$ and $m = 5$.

$$M = \begin{array}{c} \\ d_1 \\ d_2 \\ d_3 \end{array} \begin{array}{c} \begin{array}{cccccccc} f_1 & f_2 & f_3 & \text{C4.5} & \text{LG} & \text{k-NN} & \text{RF} & \text{GBT} \end{array} \\ \begin{pmatrix} 100 & 0.52 & -1.0 & 0.85 & 0.86 & 0.77 & 0.93 & 0.92 \\ 300 & 0.45 & 2.0 & 0.55 & 0.52 & 0.70 & 0.85 & 0.81 \\ 450 & 0.77 & 1.5 & 0.71 & 0.83 & 0.69 & 0.74 & 0.78 \end{pmatrix} \end{array}$$

For algorithm/parameter ranking, our goal is not to predict the absolute expected performance of any object (be it algorithms or parameter settings), but rather the relative performance between objects. Thus, the original meta-dataset can be transformed to represent the ranks:

$$\Gamma = transform(M) \equiv$$

$$\begin{array}{c} \\ d_1 \\ d_2 \\ d_3 \end{array} \begin{array}{c} \begin{array}{cccccccc} f_1 & f_2 & f_3 & \text{C4.5} & \text{LG} & \text{k-NN} & \text{RF} & \text{GBT} \end{array} \\ \begin{pmatrix} 100 & 0.52 & -1.0 & 4 & 3 & 5 & 1 & 2 \\ 300 & 0.45 & 2.0 & 4 & 5 & 3 & 1 & 2 \\ 450 & 0.77 & 1.5 & 4 & 1 & 5 & 3 & 2 \end{pmatrix} \end{array}$$

In this case, ranking is a special case of the general multi-target regression setting. Then, a meta-learner (ranking algorithm) will take the new meta-dataset, such as Γ, as its training input to learn a ranking model.

Given a new dataset, we first calculate its meta-features and use the meta-features (e.g., the f_1, f_2, f_3 values for the above example) as input to the ranker. The ranker finally returns the predicted ranks for each algorithm.

Existing meta-learning systems are mainly based on three types of meta-features: statistical, information-theoretic and landmarking-based meta-features, or **SIL** for short. Thorough reviews of these meta-features can be found in [2, 13, 10, 17, 15, 14].

1.1 Meta-Learning Approaches

We first briefly review several meta-learning approaches from an *algorithmic* perspective. A more detailed review of the following approaches can be found in [4, 3, 17].

***k*-Nearest Neighbors**—The k-NN ranking approach has two steps: the nearest neighbor search step and the ranking generation step. In the first step, given a new dataset, we first calculate its meta-features to construct an instance as a query (a u-value array). Then, we select a set of instances (nearest neighbors) in the training set (the $n \times v$ data matrix) that are similar to the query instance. In the second step, we combine the rankings of the nearest neighbors to generate an aggregated algorithm ranking for the new dataset.

Pairwise Classification—Given the $n \times m$ data matrix as the training data, multiple binary (pairwise) classification models can be used to construct a ranking model. Given a new dataset, we first calculate its meta-features as a query. Then, we use the binary classification models to classify the query. The final algorithm ranking list for the new dataset is computed based on how many times each algorithm has been predicted as "is better".

Label Ranking—Label ranking can be seen as an extension of the conventional setting of classification. The former can be obtained from the latter by replacing single class labels by complete label rankings. Meta-learning for algorithm ranking using the multi-target setting can also be transformed to a label ranking problem, so that label ranking algorithms can be used directly.

1.2 Pairwise Meta-Rules (PMR) for Meta-Learning

In this section, we briefly review the Pairwise Meta-Rule (PMR) method proposed in [18]. The main motivation of PMR is that existing meta-feature sets have ignored the logical pairwise relationships between each pair of the target algorithms to rank. Explicitly adding this information to the meta-feature space might improve a meta-learner's predictive accuracy. In [18], the authors proposed to use a rule learner to learn pairwise rules first, and then use these rules as new meta-features.

Two steps are involved: the first step is similar to the binary pairwise classification ranking approach, where $\binom{m}{2}$ binary classification training datasets are constructed from the original $n \times v$ data matrix. Denote the rank value of an algorithm with index z, $\Phi(z)$, binary classification training dataset $A^{(ij)}$ is constructed as:

$$A^{(ij)} = \begin{array}{c} \\ d_1 \\ d_2 \\ \vdots \\ d_n \end{array} \left(\begin{array}{ccccc} f_1 & f_2 & \cdots & f_u & \text{class label} \\ a_{1,1} & a_{1,2} & \cdots & a_{1,u} & l_1 = \begin{cases} 1 \text{ if } \Phi(i) > \Phi(j); \\ 0 \text{ otherwise.} \end{cases} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,u} & l_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,u} & l_n \end{array} \right)$$

Whether an algorithm is better than the other is determined by each algorithm's ranking position ($\Phi(z)$) in Γ.

Then, we build $\binom{m}{2}$ rule-based binary classification models based on the above $\binom{m}{2}$ binary classification meta-datasets. In [18], the authors have shown that the RIPPER algorithm [8] works well for this purpose. For each pair of algorithms $(i, j, i < j)$, a RIPPER ruleset is built for describing in which situation(s) an algorithm is to be preferred over another. These rules are called the Pairwise Meta-Rules (PMR). Following is an example rule model (set) for two algorithms `SVM` and `C4.5`:

```
If Num.Features > 100 AND Num.Instances < 3000 Then SVM is
better
If Num.NumericFeatures > 80% Then SVM is better
Otherwise C4.5 is better.
```

This RIPPER ruleset comprises three rules. The first two are *individual* rules, whereas the third one is a default catch-all rule. **PMR** turns each *individual* rule into one boolean meta-feature. For example, an *individual* rule may look like:

`If BaseMetaFeature-X` $\leq$ `0.5 AND BaseMetaFeature-Y` $\geq$ `0,`
`Then Algorithm A is better than Algorithm B.`

The value of the new meta-feature constructed from this pairwise meta-rule will be determined by looking at the (base-level) meta-feature values of a new dataset defined by the pairwise meta-rule. For a new dataset, the PMR-based meta-feature value is set to *true (1)* if the rule conditions: "`BaseMetaFeature-X` $\leq$ `0.5 AND BaseMetaFeature-Y` $\geq$ `0`" are met, or to *false (0)* otherwise.

PMR-based meta-features are then added to the original feature space. Figure 1 shows the concept diagram of the meta-rule generation procedure.

The PMR method has been shown to improve the predictive performance of several meta-learning algorithms, including k-Nearest Neighbor (k-NN) for meta-learning, Pairwise Classification and Approximate Ranking Tree Forests [18].

Given n training instances (datasets), and m algorithms to rank, the training complexity of PMR is $\mathcal{O}(g(n)\binom{m}{2}) = \mathcal{O}(g(n)(m(m-1)/2))$, where $g(n)$ is the training complexity of the RIPPER algorithm. Since $g(n)$ is a constant when

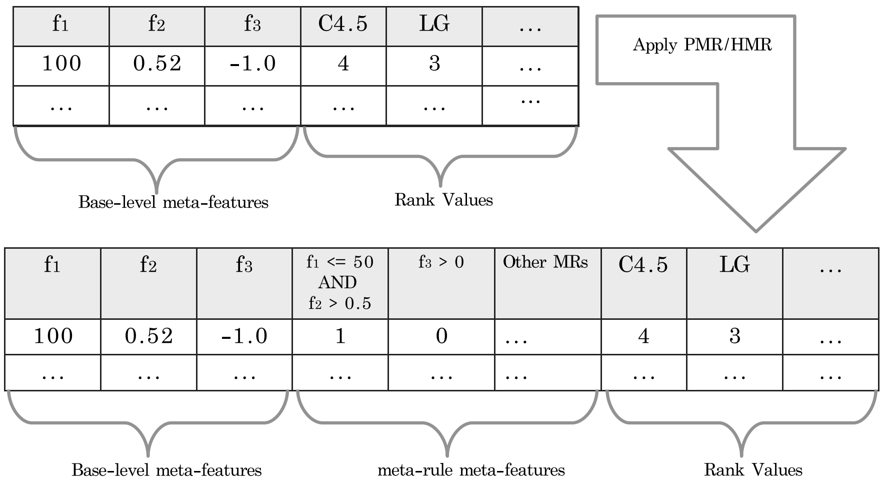


Fig. 1. Meta-rule generation conceptual graph

n is fixed, we can see that the training complexity of PRM is quadratic with respect to m.

In the next section, we propose an alternative meta-rule method named Hierarchical Meta-Rules (HMR), which is based on the theory of orthogonal contrasts. The proposed HMR method has a linear training complexity with respect to m, providing a way of dealing with a large number of objects that the PMR method cannot handle efficiently.

2 Hierarchical Meta-Rules

In this section, we propose a novel meta-rule method named Hierarchical Meta-Rules (HMR). The section is organised in two parts. The first part focuses on the steps of the proposed HMR method. The second part attempts to give a more formal explanation of the fundamental principle of the HMR method.

Suppose we have m objects to rank, e.g., algorithms or parameter settings. Instead of generating meta-rules from $\binom{m}{2}$ pairwise comparisons as in the PMR method, in HMR, we first find a set of $(m-1)$ group-wise comparisons (technically called orthogonal contrasts) by object-wise clustering. For ease of reading, we leave the theory of orthogonal contrasts and why we use object-wise clustering here to the second part of this section. Next, we generate meta-rules by training only $(m-1)$ binary classification rule models based on the $(m-1)$ group-wise comparisons found by the object-wise clustering procedure. Finally, as in the PMR method, we add these group-wise meta-rules as new meta-features to the original feature space. Algorithm 1 shows the steps of the HMR procedure. Figure 1 also shows the concept graph of the HMR procedure.

Above we have described the HMR method. Next, we introduce the theory of orthogonal contrasts (OC) proposed and extended by Marden and Chung in

Algorithm 1 The Hierarchical Meta-Rules (HMR) Procedure

```
Input:
$\mathcal{T}$—training data, similar to matrix $\Gamma$ in Section 1.2;
$\mathcal{C}$—a hierarchical clustering algorithm
$\Omega$—a rule learner
$m$—number of objects

$A \leftarrow$ getRankMatrix($\mathcal{T}$); // get the rank matrix
$c \leftarrow \mathcal{C}(A^T)$; // build a clustering model on the transpose of $A$, so each row of $A^T$ is
a vector of rank values of an object; the number of rows (datasets) is therefore $m$.
transform $c$ to a set of $m-1$ orthogonal contrasts $\mathcal{Q}$
for each (two-group) contrast $q \in \mathcal{Q}$ do
    build rule models with rule learner $\Omega$ between the best algorithm in each of the
    two groups in a contrast $q$.
end for
for each meta-rule $r \in$ all $m-1$ rule models do
    add $r$ as new features to the original feature space
end for
```

[6, 7, 12]. The OC theory provides a theoretical foundation for the design of the HMR method.

Traditionally, the "target" part of the matrix Γ in Section 1.2 is called the rank data. Marden [12] provides an excellent review of models and methods that can be used for rank data analysis. The basic unit of analysis consists of n experiments ranking a set of m objects. Here, the experiments can be seen as "judges" to rank objects (i.e. algorithms in a meta-learning experiment). Denote the set of objects by $\mathcal{O} = \{O_1, O_2, ..., O_m\}$. A full ranking of the objects represents a complete ordering to the objects. There are two (mathematically equivalent) representations of a ranking, namely, the rank vector and the order vector. In a rank vector, the objects are listed in a prespecified order, where "1" denotes best and m denotes worst. In contrast, an object vector lists the objects in order from best to worst. The objects themselves can be identified with integers, therefore both rankings and orderings are permutations of the first m integers [12]. To alleviate confusion, we used the rank vector representation in this paper. A ranking y resides in the space:

$$S_m \equiv \{\text{Permutations of the ranks } \{1, 2, ..., m\}\}. \tag{1}$$

The rank part of Γ can be seen as a sample of n rank vectors:

$$y^{(1)}, y^{(2)}, ..., y^{(n)} \in S_m. \tag{2}$$

Consider five ($m = 5$) objects $\equiv$ machine learning algorithms: C4.5 decision trees, Logistic Regression (LG), k-Nearest Neighbors algorithm (k-NN), Random Forests (RF) and Gradient Boosted Trees (GBT).

$$\mathcal{O}_{m=5} = \{\text{C4.5}, \text{LG}, \text{k-NN}, \text{RF}, \text{GBT}\}. \tag{3}$$

If we ask a data miner to rank the above five algorithms, she might wish to compare $\underline{\text{C4.5, LG and k-NN}}$ to $\underline{\text{RF and GBT}}$, that is, to compare single-model algorithms to ensemble algorithms; or compare $\underline{\text{k-NN}}$ to $\underline{\text{C4.5 and LG}}$ if she cares more about the trade-off between the runtime of training and prediction accuracy; or compare $\underline{\text{LG}}$ to $\underline{\text{C4.5}}$ for better probability outputs; or compare $\underline{\text{RF}}$ to $\underline{\text{GBT}}$ for better predictive performance.

Imagine there are a large number of objects to be ranked. For an experienced data miner the above comparison strategy could be more efficient than pairwise comparisons (as in the PMR method). Technically, a comparison of groups of objects is called a *contrast*. For the above example, we could define the following (two-group) contrasts:

C_1 = ({C4.5, LG, k-NN}, {RF, GBT});
C_2 = ({k-NN}, {C4.5, LG});
C_3 = ({LG}, {C4.5});
C_4 = ({RF}, {GBT});

In order to reveal the properties of contrasts, we next introduce some formal definitions.

Definition 1. (Marden (1995) [12]) *a) A partition of the integer λ_+ is a vector $\lambda = (\lambda_1, ..., \lambda_K)$ of positive integers that sum to λ_+ and are in nonincreasing order. b) An ordered partition lifts the ordering restriction.*

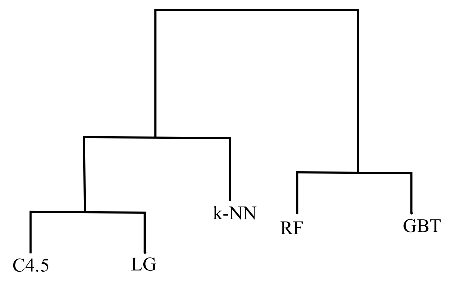
*For any set Ω and an ordered partition λ with $\lambda_+ \leq \#\Omega$, let $\lambda(\Omega)$ denote the set of all ordered sets of disjoint subsets Ω with sizes given by λ. That is, $\lambda(\Omega) = \{(\mathcal{O}_1, ..., \mathcal{O}_K) | \mathcal{O}_K \subset \Omega, \mathcal{O}_k \cap \mathcal{O}_l = \emptyset \text{ if } k \neq l,$
$\text{and } \#\mathcal{O}_k = \lambda_k, k = 1, ..., K\}$.*

For example, if $\Omega = \{\text{C4.5}, \text{LG}, \text{k-NN}, \text{RF}\}$ and $\lambda = (2, 1, 1)$, then $\lambda(\Omega)$ contains $\binom{4}{2,1,1} = \dfrac{4!}{2!1!1!} = 12$ sets of subsets, such as ({C4.5,k-NN},{LG},{RF}), ({C4.5,LG},{RF},{k-NN}) and so on. If $\Omega = \{\text{C4.5}, \text{LG}, \text{k-NN}, \text{RF}, \text{GBT}\}$ and $\lambda = (2, 2, 1)$, than $\lambda(\Omega)$ contains $\binom{5}{2,2,1} = \dfrac{5!}{2!2!1!} = 30$ sets of subsets.

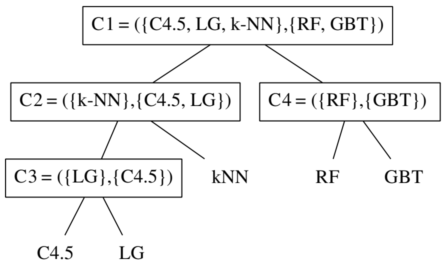
Definition 2. (Marden (1995) [12]) *A contrast is an element $C \in \lambda(\mathcal{O})$ of Definition 1 for some ordered partition λ with $\lambda_+ \leq m$.*

Given a contrast $C = (\mathcal{O}_1, \mathcal{O}_2, ..., \mathcal{O}_K)$, we have the full set of objects $\mathcal{O}^C \equiv \cup_{k=1}^K \mathcal{O}_k$. The vector λ corresponding to contrast C is: $\lambda^C = (\#\mathcal{O}_1, ..., \#\mathcal{O}_K)$. For example, $\lambda^{C_1} = (3, 2)$, $\lambda^{C_2} = (1, 2)$, $\lambda^{C_3} = (1, 1)$and $\lambda^{C_4} = (1, 1)$. So we can see that contrast C represents a comparison between the sets of objects $\mathcal{O}_k$.

Definition 3. *Two contrasts, $C = (\mathcal{O}_1, \mathcal{O}_2, ..., \mathcal{O}_K)$ and $D = (\mathcal{O}_1^*, \mathcal{O}_2, ..., \mathcal{O}_L^*) \in \lambda^D(\mathcal{O})$, are orthogonal if either $\mathcal{O}^C \cap \mathcal{O}^D = \emptyset$; or by virtue of nesting if $\mathcal{O}^C \subseteq \mathcal{O}_l^*$ for some l; or if $\mathcal{O}^D \subseteq \mathcal{O}_k$ for some k;*



(a) Hierarchical clustering tree (dendrogram)



(b) Full binary tree

Fig. 2. Tree diagrams for orthogonal contrasts

In this paper, we are particularly interested in the two-group orthogonal contrasts, which is the set of orthogonal contrasts (Definition 3) with the restriction of $K = L = 2$. The reason is that a set of two-group orthogonal contrasts can be represented by a hierarchical clustering tree model $\equiv$ a full binary tree.

Definition 4. (Full Binary Trees) *A binary tree T is full if each node is either a leaf or possesses exactly two child nodes.*

Proposition 1. *With m objects, a set of two-group orthogonal contrasts can have at most $m - 1$ contrasts.*

Outline of proof: The crux is to relate a hierarchical clustering model to a full binary tree. For example, Figure 2 (a) shows a hierarchical clustering model for $\mathcal{O}_m = \{\text{C4.5}, \text{LG}, \text{k-NN}, \text{RF}, \text{GBT}\}$. By labelling the internal nodes with a contrast between child nodes, we could obtain the full binary tree presentation (see Figure 2 (b)). Orthogonal contrasts C_1, C_2, C_3 and C_4 are labelled as internal nodes. Then, the proof follows immediately from the well-known Full Binary Tree Theorem: If T has m leaves, the number of internal nodes is $m - 1$.

The main theoretical reason for using orthogonal contrasts as the core component of the HMR method is: given enough orthogonal contrasts, it is possible to reconstruct the original ranking [12]. An analogy to this is the pairwise comparison model. Given enough pairwise comparisons, it is possible to reconstruct the original ranking.

Proposition 2. *If $(C_1, C_2, ..., C_T)$ is a set of two-group orthogonal contrasts and $Y \sim Uniform(S_m)$ (for any finite set Ω, $W \sim Uniform(\Omega) \Rightarrow P[W = w] = \dfrac{1}{\#\Omega}, w \in \Omega$.), then $Y_{C_t}, t = 1, 2, ..., T$ are independent.*

The proof follows from the Lemma 5.1, 7.5 in [12] for the more general case of multi-group orthogonal contrasts. The details are omitted due to space constraints. We here sketch the basic principle: the idea is to relate contrasts to rankings, so that the relative rankings within a group are independent of the rankings between groups.

The next question is how to find a proper set of orthogonal contrasts. In this paper, as we have showed in the first half of this section, we employ object-wise hierarchical clustering in our HMR method for finding a set of two-group orthogonal contrasts.

Readers may note that the pairwise comparison method (e.g., in PMR) actually constructs $\binom{m}{2}$ two-group contrasts (not all of which are orthogonal). So we can see that pairwise comparisons in PMR might be redundant and less efficient but indeed contain much richer information than group-wise comparisons in HMR. In future research, we also plan to investigate how to quantify the information contained in PMR and HMR from a probabilistic perspective.

3 Experiments

We have two experimental goals. The first is to compare meta-learners that use the HMR method to those that do not use any of the meta-rule methods. The second is to compare the performance of the HMR method to PMR. Especially, the runtime comparison between PMR and HMR, since in theory the proposed HMR method has a better scalability than PMR – being linear instead of quadratic.

3.1 Datasets

Four meta-datasets[1], namely **algo20**, **rf70**, **lg100** and **smo110** are used in our experiment. The feature values of the four datasets are identical, and they all have 466 instances and 80 features. However, the target parts of the datasets are different. Dataset **algo20** has 20 target ranks, which was generated based on the AUC scores of 20 WEKA [9] machine learning algorithms. Datasets **rf70**, **lg100** and **smo110** have 70, 100, and 110 target ranks respectively. Their target rank values were generated based on the AUC scores of 70, 100, and 110 parameter settings of the three algorithms (logistic regression, random forests and support vector machines) respectively.

3.2 Clustering Algorithms and Rule Learner

The clustering algorithm to be used for the object-wise clustering part has to satisfy one requirement—the final clustering model needs be consistent with a full binary tree structure. Otherwise we will not be able to construct orthogonal contrasts from the clustering model. In the following experiments, we employ two well-known hierarchical clustering algorithms: Hierarchical Clustering with complete linkage (HC) [19] and Bisecting k-means clustering (BKM) [16], because both of these models can be represented as full binary trees, similar to

[1] Supplementary materials can be downloaded from http://quansun.com/pricai2014/

the one shown in Figure 2 (a). We also use the RIPPER rule learner for training rule models, as the RIPPER algorithm has been shown to work well as a meta-rule learner in [18]. The reason for testing two clustering algorithms is: HC provides a bottom-up construction of contrasts; whereas in contrast BKM provides a top-down construction procedure.

3.3 Experimental Setup

We use two meta-learners in the following experiment. Namely, the k-NN for meta-learning algorithm and the Approximate Ranking Tree Forests (ARTF) algorithm. For details of the two algorithms, we refer the reader to [18]. For k-NN, we set $k = 15$ as suggested in [18]. We use 100 ART trees for ARTF. Parameter setting should not significantly affect our general conclusion since we are comparing the same parameter setting of a meta-learner with two different input features. We did not use a pairwise classification based meta-learner because it is not feasible for the parameter ranking datasets used in this paper (due to too many targets).

We assess ranking accuracy by comparing the rankings predicted by a meta-learner (ranker) for a given dataset with the corresponding target rankings. Given two sets of m-value rankings:

$T = [T_1, T_2, ..., T_{m-1}, T_m]$ and $P = [P_1, P_2, ..., P_{m-1}, P_m]$,

which are targets and predictions, respectively, and letting $d_i^2 = (T_i - P_i)^2$, the Spearman's Rank Correlation Coefficient (SRCC) is used in our experiments:

$$\rho_{SRCC} = 1 - \frac{6 \sum_{i=1}^{m} d_i^2}{m(m^2 - 1)}. \tag{4}$$

SRCC assesses how well the relationship between the true and predicted rankings can be described using a monotonic function [11].

The actual evaluation metric scores of each meta-learner were estimated based on multiple runs of train/test split evaluations. We use the average scores obtained from 30 runs of 66% vs. 34% train/test evaluation for result visualization.

3.4 Experimental Results

Table 1 shows the predictive performances of two meta-learners using a meta-rule method against using only the **b**ase-level meta-**f**eatures (BF). For both the ARTF and k-NN meta-learners, we can see that the PMR-based approach always significantly outperforms the respective meta-learner without using meta-rules. This is consistent with the result shown in [18].

For the ARTF meta-learner, the HMR-based meta-rule approaches significantly outperformed BF on 2 out of 4 (50%) datasets. For the k-NN meta-learner, the HMR-based meta-rule approaches significantly outperformed BF on 3 out of 4 (75%) datasets.

Table 1. Ranking performances of two meta-learners that use or without using meta-rule methods. • or ◦ means the predictive performance of a meta-learner using the respective meta-rule method is significantly better or worse than that without using the predictive meta-rule method

Dataset	Base Features (BF)	BF plus HMR-HC	BF plus HMR-BKM	BF plus PMR
algo20	0.601±0.018	0.599±0.018	0.599±0.018	0.608±0.016 •
rf70	0.558±0.030	0.569±0.032 •	0.571±0.030 •	0.584±0.033 •
lg100	0.666±0.016	0.674±0.017 •	0.673±0.017 •	0.675±0.017 •
smo110	0.218±0.017	0.217±0.017	0.218±0.020	0.219±0.016 •

(a) Meta-learner: ARTF

Dataset	Base Features (BF)	BF plus HMR-HC	BF plus HMR-BKM	BF plus PMR
algo20	0.552±0.019	0.559±0.018 •	0.559±0.019 •	0.592±0.017 •
rf70	0.500±0.033	0.532±0.036 •	0.539±0.033 •	0.557±0.035 •
lg100	0.653±0.016	0.665±0.017 •	0.664±0.016 •	0.667±0.018 •
smo110	0.174±0.015	0.171±0.014 ◦	0.174±0.015	0.189±0.015 •

(b) Meta-learner: k-NN

Table 2. Runtime (in seconds) of different meta-rule construction methods; Values are the average of 30 runs

Dataset	HMR-HC	HMR-BKM	PMR
algo20	2	2	24
rf70	13	13	854
lg100	18	17	1343
smo110	22	22	2070

The performances of the two HMR-based methods are very close and have no significant difference on 7 out of 8 (87.5%) comparisons.

Table 2 shows the runtime of the three meta-rule construction methods. We can see that the HMR-based methods are much faster than the PMR-based method, which is consistent with their theoretical runtime complexities.

To see if the HMR-based meta-rule methods have a linear training complexity when the number of targets increases. We run an experiment to show the relationship between the training time and the number of targets. Figure 3 shows the training time of the HMR-HC method on the four datasets. We can see that the figures show clear linear relationship. Runtime figures for HMR-BKM are not included since the linear patterns are the same.

For simplicity, in our previous analysis, we ignored the training runtime of the two clustering algorithms (HC and BKM) because in our context m is relatively small to them (both algorithms could build a clustering model within a second). However, when m is a relatively large number, the faster clustering algorithm may be preferred.

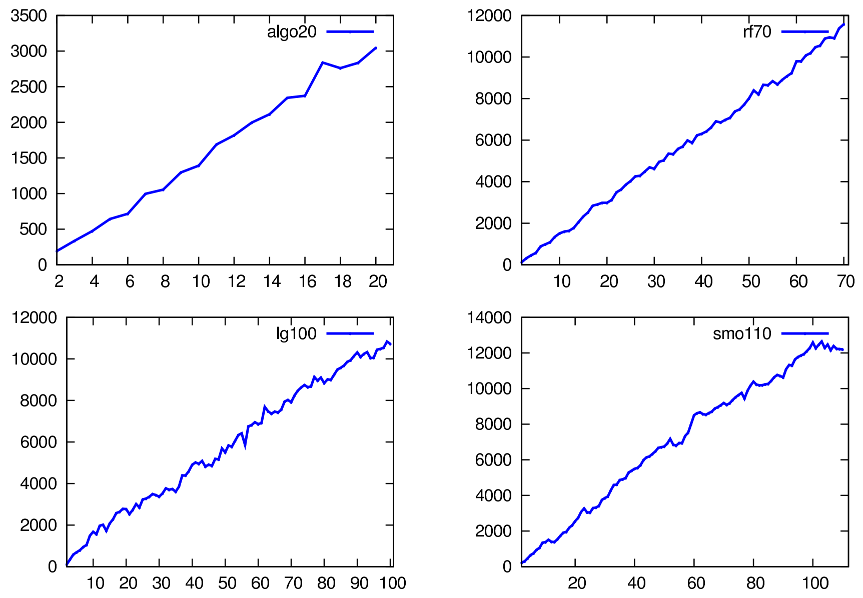


Fig. 3. Meta-rule construction runtime of the HMR-HC method on four datasets. Values in the figures represent the mean of 10 runs. X-axis represents the number of targets; Y-axis represents the meta-rule construction time in milliseconds.

4 Conclusions

In this paper, we proposed a novel meta-rule method—Hierarchical Meta-Rules (HMR), which has a linear training complexity with respect to the number of target objects. Compared to the ARTF meta-learner, the k-NN meta-learner is more likely to achieve a better ranking performance when it is used together with HMR-based methods. In terms of the runtime for meta-rule construction, HMR-based methods are much faster than PMR. This is consistent with our theoretical judgement, where PMR's training complexity is quadratic and HMR is linear with respect to the number of target objects.

We also introduced the orthogonal contrasts (OC) theory. To the best of our knowledge, the HMR-based meta-rule method is the first application of the OC theory to machine learning. Also, the HMR-based ranking approach for meta-learning is the only meta-rule method available that is designed for a relatively large number of objects (be it algorithms or parameter settings). This contribution makes the meta-rule method feasible for large scale meta-learning based recommendation problems.

References

1. Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: bagging, boosting, and variants. Machine learning 1(38) (1998)
2. Brazdil, P., Gama, J., Henery, B.: Characterizing the applicability of classification algorithms using meta-level learning. In: Proceedings of the European Conference on Machine Learning (1994)
3. Brazdil, P., Giraud-Carrier, C., Soares, C., Vilalta, R.: Metalearning: Applications to Data Mining. Springer (2009)
4. Brazdil, P., Soares, C., Da Costa, J.P.: Ranking learning algorithms: Using ibl and meta-learning on accuracy and time results. Mach. Learn. 50(3), 251–277 (Mar 2003)
5. Caruana, R., Niculescu-mizil, A.: An empirical comparison of supervised learning algorithms. In: In Proc. 23 rd Intl. Conf. Machine learning (ICML06. pp. 161–168 (2006)
6. Chung, L., Marden, J.I.: Use of nonnull models for rank statistics in bivariate, two-sample, and analysis of variance problems. Journal of the American Statistical Association 86(413), 188–200 (1991)
7. Chung, L., Marden, J.I.: Extensions of mallows ϕ model. In: Probability Models and Statistical Analyses for Ranking Data, pp. 108–139. Springer (1993)
8. Cohen, W.W.: Fast effective rule induction. In: Proceedings of the 12th International Conference on Machine Learning. Morgan Kaufmann (1995)
9. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.: The weka data mining software: An update. SIGKDD Explorations 11(1) (2009)
10. Kalousis, A.: Algorithm Selection via Meta-Learning. Ph.D. thesis, Department of Computer Science, University of Geneva (2002)
11. Kendall, M.G.: Rank correlation methods. Griffin (1970)
12. Marden, J.: Analyzing and Modeling Rank Data. Monographs on Statistics and Applied Probability, Chapman and Hall (1995)
13. Pfahringer, B., Bensusan, H., Giraud-Carrier, C.: Meta-learning by landmarking various learning algorithms. In: Proceedings of the 17th International Conference on Machine Learning (2000)
14. van Rijn, J.N., Holmes, G., Pfahringer, B., Vanschoren, J.: Algorithm selection on data streams. In: Discovery Science. Springer (2014)
15. Rossi, A.L.D., De Carvalho, A.C.P.D.L.F., Soares, C., De Souza, B.F.: Metastream: A meta-learning based method for periodic algorithm selection in time-changing data. Neurocomputing 127, 52–64 (2014)
16. Steinbach, M., Karypis, G., Kumar, V.: A comparison of document clustering techniques. In: In KDD Workshop on Text Mining (2000)
17. Sun, Q.: Meta-Learning and the Full Model Selection Problem. Ph.D. thesis, The University of Waikato (2014)
18. Sun, Q., Pfahringer, B.: Pairwise meta-rules for better meta-learning-based algorithm ranking. Machine Learning 93(1), 141–161 (2013)
19. Ward Jr, J.H.: Hierarchical grouping to optimize an objective function. Journal of the American statistical association 58(301), 236–244 (1963)
20. Weiss, S.M., Kapouleas, I.: An empirical comparison of pattern recognition, neural nets, and machine learning classification methods. In: In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence. pp. 781–787. Morgan Kaufmann (1989)
21. Wolpert, D., Macready, W.: No free lunch theorems for optimization. Evolutionary Computation, IEEE Transactions on 1(1), 67–82 (apr 1997)