Pairwise Meta-Rules for Better Metalearning-Based Algorithm Ranking

Quan Sun and Bernhard Pfahringer

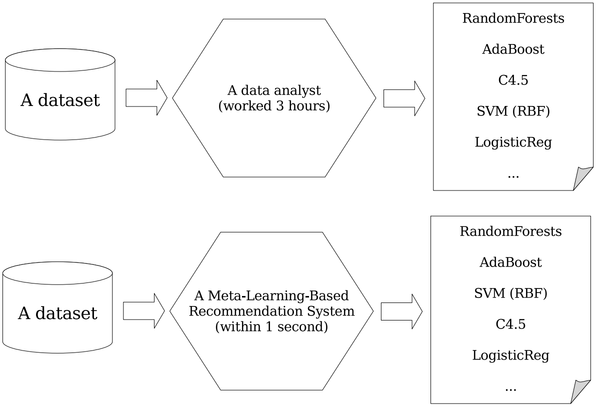
Machine Learning Group
Department of Computer Science
The University of Waikato, New Zealand

September 2013

Metalearning

- Metalearning is usually explained as "learning to learn".
- In this paper, the term is used in the sense of "**metalearning for algorithm ranking or recommendation**".

A Successful Metalearning System



- Metalearning tries to support and automate algorithm selection, by generating meta-knowledge mapping the properties of a dataset to the relative performances of algorithms.

The Metalearning Task

The basic steps of building a metalearning system:

1. collect a set of datasets
2. define some meta-features of each dataset, e.g., the #. of instances, the #. of numeric or categorical features...
   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.
3. estimate the predictive performance of the available algorithms (eg, CV), for every dataset in the dataset collection

Meta-Dataset

Given the above information, we can construct a meta-dataset:

$$M = \begin{array}{c} \\ d_1 \\ d_2 \\ d_3 \end{array} \begin{array}{cccccccc} f_1 & f_2 & f_3 & \text{C4.5} & \text{LG} & \text{k-NN} & \text{RF} & \text{SVM} \\ \end{array}$$

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

- For algorithm ranking, our goal is to predict the relative performance between algorithms. Thus, the (raw) meta-dataset can be transformed to represent the rankings of the algorithms.

Meta-Dataset cont'd

$$M = \begin{array}{c} \\ d_1 \\ d_2 \\ d_3 \end{array} \begin{array}{cccccccc} f_1 & f_2 & f_3 & \text{C4.5} & \text{LG} & \text{k-NN} & \text{RF} & \text{SVM} \\ \end{array}$$

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

$$M^* = transform(M) \Longrightarrow$$

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

Metalearning Approaches

- The k-Nearest Neighbors approach
- The pairwise classification approach
- The learning to rank approach
- The label ranking approach
- The single/multi-target regression approach

Three Contributions

- Pairwise Meta-Rules (a new meta-feature generator)
- Approximate Ranking Tree Forests (a new meta-learner)
- Parameter-Optimisation-Based Ranking Generation (a new experimental configuration)

Pairwise Meta-Rules (PMR)

- Explicitly adding the logical pairwise information between each pair of the target algorithms to the meta-feature space might improve a meta-learner's predictive accuracy.
- We propose to use a rule learner to learn pairwise rules first, and then use these rules as new meta-features.

Pairwise Meta-Rules: Step 1

Construct a binary classification dataset for each algorithm pair. Each binary dataset (i, j pair, $i < j$) has two class labels:

$$A^{(ij)} = \begin{array}{c} \\ d_1 \\ d_2 \\ \vdots \\ d_n \end{array} \begin{array}{ccccc} f_1 & f_2 & \cdots & f_u & \text{class label} \end{array} \left(\begin{array}{ccccc} a_{1,1} & a_{1,2} & \cdots & a_{1,u} & l_1 = \begin{cases} \texttt{Yes} & \text{if Algorithm } i \text{ is better;} \\ \texttt{No} & \text{otherwise.} \end{cases} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,u} & l_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,u} & l_n \end{array} \right)$$

In total, there are $\frac{m \times (m-1)}{2}$ (m is the #. of target algorithms) binary classification datasets.

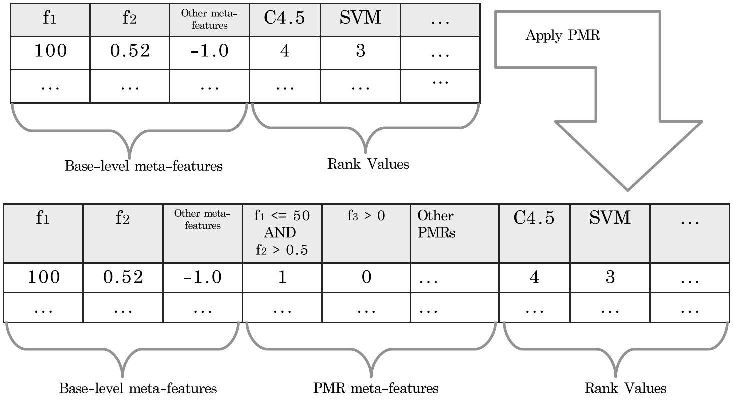
Pairwise Meta-Rules: Step 2

- Build a RIPPER rule model for each of the $\frac{m \times (m-1)}{2}$ binary datasets.
- Add meta-rules in each RIPPER model as new meta-features to the original feature space

A RIPPER rule model for `SGD` vs. `Naive Bayes` may look like:

```
If ObliviouTree.depth2.AUC  ≤ 0.55 AND MaxNominalFeatureDistinctValues  ≤ 7
  Then SGD is better;
If REPTree.depth2.AUC  ≤ 0.53 AND RandomTree.depth2.AUC  ≤ 0.51
  Then SGD is better;
Otherwise Naive Bayes is better.
```

Pairwise Meta-Rules (PMR)



f1	f2	Other meta-features	C4.5	SVM	…
100	0.52	-1.0	4	3	…
…	…	…	…	…	…

f1	f2	Other meta-features	f1 <= 50 AND f2 > 0.5	f3 > 0	Other PMRs	C4.5	SVM	…
100	0.52	-1.0	1	0	…	4	3	…
…	…	…	…	…	…	…	…	…

Approximate Ranking Tree Forests (ART Forests)

- Base-level $+$ PMR meta-features $\equiv$ a high-dimensional feature space
- We need a meta-learner that can handle the feature space efficiently
- ART Forests: an ensemble of random Approximate Ranking Trees using the random forests framework

The Approximate Ranking Trees (ART) Algorithm

Input:
D (training data);
u (number of features to test when splitting, default $log_2 M+1$, M is the $\#$. of features)
C (splitting and stopping criterions, details are given in the paper)

```
$bestSplit \leftarrow$ Randomly choose $u$ features and test them based on the splitting criterion $C$. Use the best feature among the $u$ features.

if $stopping\ criterion$ is met
    return a leaf node with the corresponding leaf ranking.
else
    $leftSubtree \leftarrow \mathbf{ART}(D^{+}_{bestSplit}, u, C)$
    $rightSubtree \leftarrow \mathbf{ART}(D^{-}_{bestSplit}, u, C)$
    return ($bestSplit$, $leftSubtree$, $rightSubtree$)
end if
```

ART's Splitting Criterion

In the ART algorithm, we use the median value of a meta-feature's range as the binary split point to split the data D, the current partition, into two sub-partitions D^+ and D^-. The best split point is determined to be the one that maximises the R^2 statistic:

$$R^2 = 1 - \frac{\sum_{l=1}^{L} \sum_{i=1}^{n^{(l)}} d_{Spearman}(y^{(li)}, \hat{z}^{(l)})}{\sum_{l=1}^{L} \sum_{i=1}^{n^{(l)}} d_{Spearman}(y^{(li)}, \hat{z}^{(D)})}, \quad (1)$$

where L is the number of partitions, and $n^{(l)}$ is the number of examples in partition l. R^2 is originally designed to measure the proportion of the spread explained by the differences between the two partitions.

In the paper, we showed that R^2 can be computed efficiently:

$$R^2 = 1 - \frac{n^{(D^+)}(h - 2||\bar{y}^{(D^+)}||^2) + n^{(D^-)}(h - 2||\bar{y}^{(D^-)}||^2)}{n^{(D)}(h - 2||\bar{y}^{(D)}||^2)}, \quad (2)$$

where $h = \frac{m(m+1)(2m+1)}{3}$.

Grow an ART forest using the Random Forests Framework

Input:
T (number of ART to use)
D (training data);
u (number of features to test when splitting, default $log_2 M+1$)
C (splitting and stopping criterions, details are given in the paper)

```
$ART_{ensemble} \leftarrow \emptyset$
**for** $i = 1$ **to** $T$
    $D_i \leftarrow getBootstrapSample(D)$
    $ART_i \leftarrow \mathbf{ART}(D_i, u, C)$
    $ART_{ensemble} \leftarrow ART_{ensemble} \cup ART_i$
**end for**
**return** $ART_{ensemble}$
```

Parameter-Optimisation-Based Ranking Generation

- Many previous meta-learning experiments have estimated algorithm performance using default parameter settings

  This approach is bound to be suboptimal. In practice, most algorithms need to be optimised separately for each specific dataset.

Parameter-Optimisation-Based Ranking Generation

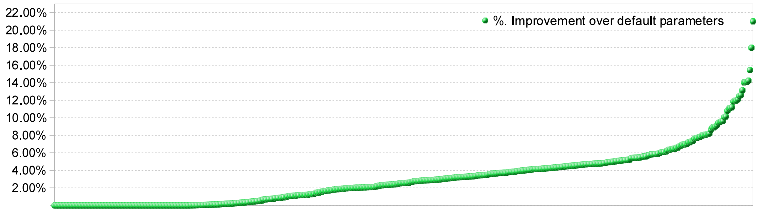


Figure: Percentage of improvement of the best AUC performance among 20 parameter-optimised algorithms for 466 datasets over the same 20 algorithm using their default parameters.

Parameter-Optimisation-Based Ranking Generation

- At the meta-dataset generation stage. We assume that a procedure is available for optimising each algorithm for each dataset, and then predict the ranking of the optimised algorithms.

Experimental Setup: Meta-dataset Construction

- rank 20 supervised machine learning algorithms.
- 466 binary classification datasets.
- we manually specify parameters and their respective value ranges for PSO to optimise. AUC is used as the target metric.

  We run the 20 algorithms, with PSO-based parameter optimisation, on the 466 binary classification datasets and use 10-fold cross-validation based AUC scores for ranking generation.

Experiments: Evaluation

8 ranking evaluation metrics and functions:

- Spearman's Rank Correlation Coefficient (SRCC)
- Weighted Rank Correlation (WRC)
- Loose Accuracy (LA@1, LA@3 and LA@5)
- Normalized Discounted Cumulative Gain (NDCG@1, NDCG@3 and NDCG@5)

Experiments: Meta-learners

7 rankers (meta-learners) are used in experiments:

- **DefRanker**: uses the average rank of each algorithm over all the training data; returns a fixed ranking
- ***k*-NN**: an instance-based algorithm
- **LRT**: a label ranking algorithm
- **RPC**: a ranking by pairwise comparison algorithm
- **PCTR**: the predictive clustering trees for ranking algorithm
- **AdaRank**: a learning to rank algorithm based on boosting
- **ARTForests**: the ART Forests algorithm

Experiments

1. compare meta-feature sets based on k-NN performance curves
2. compare ranking performances of multiple rankers

Experiment 1: compare meta-feature sets based on k-NN performance curves

Three meta-feature sets in comparison, including two PMR-based variants:

- **SIL-only**: 80 SIL meta-features
- **SIL+Meta-Rule-1**: 80 SIL meta-features plus PMR variant 1
- **SIL+Meta-Rule-2**: 80 SIL meta-features plus PMR variant 2

Experiment 1: compare meta-feature sets based on k-NN performance curves

- Overall, *k* values between 10 and 20 usually produce relatively good performance across all eight ranking metrics.
- Regarding the choice of meta-feature sets, the SIL+Meta-rules-1 set outperforms the SIL-only and the SIL+Meta-rules-2 meta-feature sets.

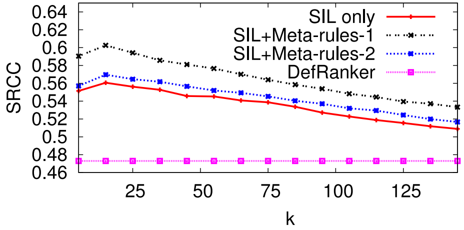


Figure: An example result for the Spearman's Rank Correlation Coefficient metric

Experiment 2: compare ranking performances of multiple rankers

- Overall, all the best ranbkers used the **SIL+PMR** set
- the **SIL+PMR** set significantly outperformed the **SIL** set in 79.1% comparision tests across 7 rankers
- ART Forests with the **SIL+PMR** set consistently produces positive performance gains for all 8 metrics
- ART Forests is placed as the best ranker for 7 out of 8 metrics

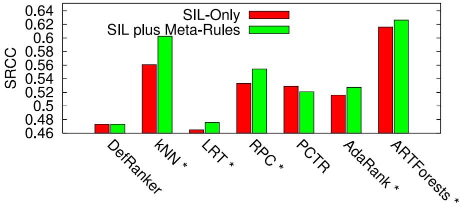


Figure: An example result for the Spearman's Rank Correlation Coefficient metric

Conclusions

- Pairwise Meta-Rules for meta-learning
- ART Forests for modelling and predicting rankings (can also be used for label ranking problems)
- Parameter-Optimisation-based meta-dataset generation
- The Art Forests software, source code and dataset can be downloaded from: http://www.cs.waikato.ac.nz/~qs12/ml/meta/

(poster stand #76)

Thank you :-)