Recommending Learning Algorithms and Their Associated Hyperparameters

Michael R. Smith[1] and **Logan Mitchell**[2] and **Christophe Giraud-Carrier**[3] and **Tony Martinez**[4]

Abstract. The success of machine learning on a given task depends on, among other things, which learning algorithm is selected and its associated hyperparameters. Selecting an appropriate learning algorithm and setting its hyperparameters for a given data set can be a challenging task, especially for users who are not experts in machine learning. Previous work has examined using meta-features to predict which learning algorithm and hyperparameters should be used. However, choosing a set of meta-features that are predictive of algorithm performance is difficult. Here, we propose to apply collaborative filtering techniques to learning algorithm and hyperparameter selection, and find that doing so avoids determining which meta-features to use and outperforms traditional meta-learning approaches in many cases.

1 Introduction

Most previous meta-learning work has focused on selecting a learning algorithm or a set of hyperparameters based on meta-features used to characterize datasets [5]. As such, it can be viewed as a form of content-based filtering, a technique commonly-used in recommender systems that captures a set of measured characteristics of an item and/or user to recommend items with similar characteristics. On the other hand, collaborative filtering (CF), also used by some recommender systems, predicts the rating or preference that a user would give to an item, based on the past behavior of a set of users, characterized by ratings assigned by users to a set of items [9]. The underlying assumption of CF is that if users A and B agree on some issues, then user A is more likely to have the same opinion on a new issue X as user B than another randomly chosen user. A key advantage of CF is that it does not rely on directly measurable characteristics of the items. Thus, it is capable of modeling complex items without actually understanding the items themselves.

Here, we propose *meta-CF* (MCF) a novel approach to meta-learning that applies CF in the context of algorithm and/or hyperparameter selection. MCF differs from most previous meta-learning techniques in that it does not rely on meta-features. Instead, MCF considers the similarity of the performance of the learning algorithms with their associated hyperparameter settings from previous experiments. In this sense, the approach is more similar to landmarking [12] and active testing [10] since both also use the performance results from previous experiments to determine similarity among data sets.

While algorithm selection and hyperparameter optimization have been mostly studied in isolation (e.g., see [12, 4, 1, 2, 3, 15]), recent work has begun to consider them in tandem. For example, Auto-WEKA simultaneously chooses a learning algorithm and sets its hyperparameters using Bayesian optimization over a tree-structured representation of the combined space of learning algorithms and their hyperparameters [16]. All of these approaches face the difficult challenge of determining a set of meta-features that capture relevant and predictive characteristics of datasets. By contrast, MCF does consider both algorithm selection and hyperparameter setting at once, but alleviates the problem of meta-feature selection by leveraging information from previous experiments through collaborative filtering.

Our results suggest that using MCF for learning algorithm/hyperparameter setting recommendation is a promising direction. Using MCF for algorithm recommendation has some differences from the traditional CF used for human ratings. For example, CF for humans may have to deal with concept drift, where a user's taste may change over time; working with learning algorithms and hyperparameter settings is deterministic.

2 Empirical Evaluation

For MCF, we examine several CF techniques implemented in the Waffles toolkit [6]: baseline (predict the mean of the previously seen results), Fuzzy K-Means (FKM) [11], Matrix Factorization (MF) [9], Nonlinear PCA (NLPCA) [13], and Unsupervised Backpropagation (UBP) [7].

To establish a baseline, we first calculate the accuracy on a set of 125 data sets and 9 diverse learning algorithms (see [14] for a discussion on diversity) with default parameters as set in Weka [8]. The set of learning algorithms is composed of backpropagation (BP), C4.5, kNN, locally weight learning (LWL), naïve Bayes (NB), nearest neighbor with generalization (NNge), random forest (RF), ridor (Rid), and RIPPER (RIP). We select the accuracy from the learning algorithm that produces the highest classification accuracy. This represents algorithm selection with perfect recall. We also estimate the hyperparameter optimized accuracies for each learning algorithm using random hyperparameter optimization [3]. The results are shown in Table 1, where the accuracy from each learning algorithm is the average hyperparameter optimized accuracy for each data set, "Default" refers to the best accuracy from the learning algorithm with its default parameters, "ALL" refers to the accuracy from the best learning algorithm and hyperparameter setting, and "AW" refers to the results from running Auto-WEKA. For Auto-WEKA, each dataset was allowed to run as long as the longest algorithm took to run on the dataset when doing the random hyperparameter optimization. As Auto-WEKA is a random algorithm, we ran 4 runs each time with a different seed and chose the seed with highest accuracy. This can be seen as equivalent to allowing a user to run on average 16 learning

[1] Brigham Young University, USA, email: msmith@axon.cs.byu.edu
[2] Brigham Young University, USA, email: mitchlam711@gmail.com
[3] Brigham Young University, USA, email: cgc@cs.byu.edu
[4] Brigham Young University, USA, email: martinez@cs.byu.edu

algorithm and hyperparameter combinations on a data set.

Table 1. Average accuracy for the best hyperparameter setting for each learning algorithm, algorithm selection (Default), both algorithm selection and hyperparameter optimization (ALL), and Auto-WEKA (AW).

BP	C4.5	kNN	LWL	NB	NNge	RF	Rid	RIP	Default	ALL	AW
79.89	79.22	78.05	77.48	76.04	76.80	79.58	71.48	77.31	81.93	83.00	82.00

For MCF, we compiled the results from hyperparameter optimization. We randomly removed 10% to 90% of the results by increments of 10% and then used MCF to fill in the missing values. The top 4 learning algorithm/hyperparameter configurations are returned by the CF technique and the accuracy from the configuration that returns the highest classification accuracy is used. This process was repeated 10 times. A summary of the average results for MCF are provided in Table 2. The columns “Best”, “Median”, and “Average” refer to the accuracies averaged across all of the sparsity levels for the hyperparameter setting for the CF technique that provided the results. The columns 0.1 to 0.9 refer to the percentage of the results used for CF averaged over the hyperparameter settings. The row “Content” refers to meta-learning where a learning algorithm recommends a learning algorithm based on a set of meta-features.

Table 2. Average accuracy from the best of the top 4 recommended learning algorithm and hyperparameter settings from MCF.

	Best	Med	Ave	0.1	0.3	0.5	0.7	0.9
Baseline	81.11	81.11	81.11	**80.49**	80.91	81.12	81.33	81.54
FKM	81.52	81.04	81.29	80.13	80.65	81.07	81.45	81.88
MF	**82.12**	**82.06**	**81.95**	**80.49**	**81.63**	**82.12**	**82.44**	**82.65**
NLPCA	81.73	81.33	81.33	79.98	80.58	81.43	82.08	82.61
UBP	81.73	81.27	81.31	80.05	80.51	81.34	82.05	82.61
Content	81.35	80.47	78.91	-	-	-	-	-

Overall, MF achieves the highest accuracy values. The effectiveness of MCF increases as the percentage of the results increases. MCF significantly increases the classification accuracy compared with both hyperparameter optimization for a given learning algorithm and model selection with their default parameters as well as using the meta-features to predict which learning algorithm and hyperparameters to use. On average, MCF and Auto-WEKA achieve similar accuracy, which highlights the importance of considering *both* algorithm selection and hyperparameter optimization. However, provided one has access to a database of experiments, such as the ExperimentDB [17], MCF only requires the time to run a number of algorithms (often ran in parallel), and retraining the collaborative filter. In the current implementation, retraining takes less than 10 seconds. Thus, MCF presents an efficient method for recommending a learning algorithm and its associated hyperparameters.

While our results show that MCF is a viable technique for recommending learning algorithms *and* hyperparameters, some work remains to be done. Future work for MCF includes addressing the cold-start problem which occurs when a data set is presented and no learning algorithm has been ran on it. MCF is adept at exploiting the space that has already been explored, but (like active testing) it does not explore unknown spaces at all. One way to overcome this limitation would be to use a hybrid recommendation system that combines content-based filtering and MCF.

REFERENCES

[1] S. Ali and K.A. Smith, ‘On Learning Algorithm Selection for Classification’, *Applied Soft Computing*, **6**2, 119–138, (2006).

[2] S. Ali and K.A. Smith-Miles, ‘A Meta-learning Approach to Automatic Kernel Selection for Support Vector Machines’, *Neurocomputing*, **70**, 173–186, (2006).

[3] J. Bergstra and Y. Bengio, ‘Random search for hyper-parameter optimization’, *Journal of Machine Learning Research*, **13**, 281–305, (2012).

[4] P. B. Brazdil, C. Soares, and J. Pinto Da Costa, ‘Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results’, *Machine Learning*, **50**(3), 251–277, (2003).

[5] P. Brazdil and C. Giraud-Carrier and C. Soares and R. Vilalta, ‘Metalearning: Applications to Data Mining’, Springer, (2009).

[6] M. S. Gashler, ‘Waffles: A machine learning toolkit’, *Journal of Machine Learning Research*, **MLOSS 12**, 2383–2387, (July 2011).

[7] M. S. Gashler, M. R. Smith, R. Morris, and T. Martinez, ‘Missing value imputation with unsupervised backpropagation’, *Computational Intelligence*, Accepted, (2014).

[8] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, ‘The weka data mining software: an update’, *SIGKDD Explorations Newsletter*, **11**(1), 10–18, (2009).

[9] Y. Koren, R. Bell, and C. Volinsky, ‘Matrix factorization techniques for recommender systems’, *Computer*, **42**(8), 30–37, (2009).

[10] R. Leite, P. Brazdil, and J. Vanschoren, ‘Selecting classification algorithms with active testing’, in *Machine Learning and Data Mining in Pattern Recognition*, ed., Petra Perner, volume 7376 of *Lecture Notes in Computer Science*, 117–131, Springer Berlin / Heidelberg, (2012).

[11] D. Li, J. Deogun, W. Spaulding, and B. Shuart, ‘Towards missing data imputation: A study of fuzzy k-means clustering method’, in *Rough Sets and Current Trends in Computing*, volume 3066 of *Lecture Notes in Computer Science*, 573–579, Springer Berlin / Heidelberg, (2004).

[12] B. Pfahringer, H. Bensusan, and C. G. Giraud-Carrier, ‘Meta-learning by landmarking various learning algorithms’, in *Proceedings of the 17th International Conference on Machine Learning*, pp. 743–750, San Francisco, CA, USA, (2000). Morgan Kaufmann Publishers Inc.

[13] M. Scholz, F. Kaplan, C. L. Guy, J. Kopka, and J. Selbig, ‘Non-linear pca: a missing data approach’, *Bioinformatics*, **21**(20), 3887–3895, (2005).

[14] M. R. Smith, T. Martinez, and C. Giraud-Carrier, ‘An instance level analysis of data complexity’, *Machine Learning*, **95**(2), 225–256, (2014).

[15] J. Snoek, H. Larochelle, and R. Adams, ‘Practical bayesian optimization of machine learning algorithms’, in *Advances in Neural Information Processing Systems 25*, eds., F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, 2951–2959, (2012).

[16] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, ‘Auto-weka: combined selection and hyperparameter optimization of classification algorithms’, in *proceedings of the 19th International Conference on Knowledge Discovery and Data Mining*, pp. 847–855, (2013).

[17] J. Vanschoren, H. Blockeel, B. Pfahringer, and G. Holmes, ‘Experiment databases - a new way to share, organize and learn from experiments’, *Machine Learning*, **87**(2), 127–158, (2012).