Using Metalearning to Predict When Parameter Optimization Is Likely to Improve Classification Accuracy

Parker Ridd[1] and **Christophe Giraud-Carrier**[2]

Abstract. Work on metalearning for algorithm selection has often been criticized because it mostly considers only the default parameter settings of the candidate base learning algorithms. Many have indeed argued that the choice of parameter values can have a significant impact on accuracy. Yet little empirical evidence exists to provide definitive support for that argument. Recent experiments do suggest that parameter optimization may indeed have an impact. However, the distribution of performance differences has a long tail, suggesting that in most cases parameter optimization has little effect on accuracy. In this paper, we revisit some of these results and use metalearning to characterize the situations when parameter optimization is likely to cause a significant increase in accuracy. In so doing, we show that 1) a relatively simple and efficient landmarker carries significant predictive power, and 2) metalearning for algorithm selection should be effected in two phases, the first in which one determines whether parameter optimization is likely to increase accuracy, and the second in which algorithm selection actually takes place.

1 Introduction

The availability of a large number of classification learning algorithms together with the No Free Lunch theorem for classification present business users with a significant challenge, namely that of deciding which algorithm is likely to induce the most accurate model for their particular classification task. This selection process is further compounded by the fact that many classification learning algorithms include parameters, and the various possible settings of these parameters may give rise to models whose accuracy on the target classification task varies significantly. As a result, the algorithm selection problem in machine learning consists not only in choosing an algorithm, but rather in choosing an algorithm *and* an associated parameter setting. Formally, let:

- $\mathcal{L} = \{L_1, L_2, \ldots, L_n\}$ be a finite set of n classification learning algorithms (e.g., C4.5, Naïve Bayes-NB, Backpropagation-BP, Support Vector Machine-SVM).
- $\mathcal{P}_{L_i} = P^1_{L_i} \times P^2_{L_i} \times \ldots P^{k_i}_{L_i}$ be the set of parameter settings associated with L_i, where each $P^k_{L_i}$ represents one of the parameters of L_i (e.g., $P^1_{C4.5}$ =pruning indicator, $P^2_{C4.5}$ =splitting criterion, ..., P^1_{BP} =number of hidden layers, P^2_{BP} =learning rate, P^3_{BP} =momentum term, ...).
- $T = \mathcal{X} \times \mathcal{Y}$ be a training set for a classification task where $\mathcal{X}$ is a set of features and $\mathcal{Y}$ is a finite set of labels.
- $\mathcal{I}(L, P, T)$ be the model induced by algorithm L with parameter setting P on some classification learning task T. Hence, $\mathcal{I}$ maps objects in $\mathcal{X}$ to labels in $\mathcal{Y}$.
- $\mathcal{A}(I)$ be the predictive accuracy of model I (typically measured by cross-validation).

The classification learning algorithm selection problem can be formulated as follows.

Classification Learning Algorithm Selection Given a training set T for some classification learning task, find the pair (L^*, P^*) where $L^* \in \mathcal{L}$ and $P^* \in \mathcal{P}_{L^*}$, such that

$$\forall (L, P) \in \mathcal{L} \times \mathcal{P}_L \;\; \mathcal{A}(\mathcal{I}(L, P, T)) \leq \mathcal{A}(\mathcal{I}(L^*, P^*, T)).$$

The above formulation is generic in that it says nothing about the process used to search the combined spaces of classification learning algorithms and parameter settings to find the optimal algorithm. Metalearning for classification learning algorithm selection is the specific instance of that general problem wherein the search is effected by a learning algorithm [6]. In other words, the past performance of classification learning algorithms on a variety of tasks is used to induce a predictive model that takes as input a classification learning task and produces as output a classification learning algorithm and its associated parameter setting.

In practice, one has access to a set $\mathcal{T} = \{T_1, T_2, \ldots, T_m\}$ of m training sets (corresponding to m classification learning tasks). For each T_j, the learning algorithms in $\mathcal{L}$, together with their parameter settings, are used one at a time to induce a model on T_j. The pair of classification learning algorithm and parameter setting that maximizes $\mathcal{A}(\mathcal{I}(L, P, T_j))$ is recorded. Each T_j with its corresponding winning pair becomes a training example for a (meta)learning algorithm. Since storing complete learning tasks is unfeasible and likely undesirable, one uses instead some characterization of learning tasks by meta-features. Meta-features may be drawn from basic statistics and information-theoretic measures (e.g., ratio of nominal features, skewness, class entropy) [14, 7, 22], landmarking measures (i.e., performances of simple learners that serve as signpost for more complex ones) [2, 17, 8], and model-based measures (e.g., properties of induced decision trees) [1, 3, 16]. Given a training set T for some classification learning task, let $\mathcal{C}(T)$ be the characterization of T by some set of meta-features.

One can now take a number of classification tasks, characterize them via the function $\mathcal{C}$, and record the corresponding meta-features together with the accuracy of the best learning algorithm and associated parameter setting. The classification learning algorithm selec-

[1] Brigham Young University, USA, email: parker.ridd@byu.net
[2] Brigham Young University, USA, email: cgc@cs.byu.edu

tion problem, as solved by metalearning, can then be reformulated as follows.[3]

Metalearning for Classification Learning Algorithm Selection Given a set $\mathcal{T}^m = \{(\mathcal{C}(T), \text{argmax}_{L \in \mathcal{L}, P \in \mathcal{P}_L} \mathcal{A}(\mathcal{I}(L, P, T))\}$ of past classification learning episodes, together with a classification learning algorithm L^m, which may belong to $\mathcal{L}$, and an associated parameter setting P_{L^m}:

1. Construct $\mathcal{M} = \mathcal{I}(L^m, P_{L^m}, \mathcal{T}^m)$.
2. For any training set T', $(L^*, P^*) = \mathcal{M}(T')$.

By convention, let $P^0_{L_i}$ denote the default parameter setting of L_i, as determined by the implementation of L_i under consideration (e.g., Weka, IBM SPSS Modeler). Most of the work in metalearning so far has addressed the above problem with the further assumption that for all learning algorithms the parameter setting is fixed to its default. This, of course, creates a much restricted, yet also greatly simplified, version of the selection problem, since the large, possibly infinite, space of parameter settings need not be considered at all. However, that restriction has also been the source of much criticism, and sometimes dismissal, by a part of the machine learning community, who has maintained that:

Parameter Optimization Claim Parameter setting has a significant impact (for the better) on the predictive accuracy of classification learning algorithms.

It would seem that most metalearning researchers, and indeed most machine learning researchers, have taken this claim to be well founded, and considered ways to address it. There have been two main approaches.

- Two-stage Metalearning. Some metalearning researchers have adopted a two-stage approach to metalearning, wherein they continue to use the restricted form of the Metalearning for Classification Learning Algorithm Selection problem to choose a learning algorithm, but then follow up with an optimization phase to find the best set of parameter values for the selected algorithm.[4] The problem in this case, however, is that one may reach a suboptimal solution. Indeed, let L_1 and L_2 be two classification learning algorithms, such that, for some classification task T', $\mathcal{M}(T') = (L_1, P^0_{L_1})$. Then, L_1 would be selected and its parameter setting optimized to $P^*_{L_1}$. Yet, despite the fact that $\mathcal{A}(\mathcal{I}(L_1, P^0_{L_1}, T')) > \mathcal{A}(\mathcal{I}(L_2, P^0_{L_2}, T'))$ (assuming the metalearner is accurate), it is entirely possible that there exists some parameter setting $P^k_{L_2}$ of algorithm L_2 such that $\mathcal{A}(\mathcal{I}(L_1, P^*_{L_1}, T')) < \mathcal{A}(\mathcal{I}(L_2, P^k_{L_2}, T'))$. In other words, the early greedy commitment to L_1 makes it impossible to explore other parts of the combined spaces of learning algorithms and parameter settings.

- Unrestricted Metalearning. Other metalearning researchers have lifted the traditional restriction, and recently begun to design solutions from the unrestricted Metalearning for Classification Learning Algorithm Selection problem. Their systems seek to select both a learning algorithm an associated parameter setting (e.g., see [13, 24, 21]).

Interestingly, and somewhat surprisingly, very few researchers have bothered to check the validity of the Parameter Optimization Claim. Yet, to the best of our knowledge —and as presented by most, it is just that: a claim. We have been hard-pressed to find any systematic study in the literature that addresses the impact of parameter settings over a wide variety of classification learning algorithms. What if the Parameter Optimization Claim does not hold in general? What if it only holds in some specific cases? Would it not be useful to know what these cases are? And what about using metalearning to characterize when the claim holds? We address these questions here.

2 Impact of Parameter Optimization on Classification Learning Algorithm Performance

There is one exception to our statement that the literature contains no systematic study of the impact of parameter optimization on the performance of classification learning algorithms, found in [23]. In that paper, the authors considered 466 datasets and for each, computed the difference in accuracy between their default parameter setting and the best possible parameter setting after optimization. The optimization procedure is based on particle swarm optimization (PSO), wherein the authors specify which parameters should be optimized (e.g., kernel function and complexity constant for SVM) and the range of values that PSO should consider. We obtained the data from the authors and reproduced their results, with two small exceptions: 1) we found that one of the datasets in the list was redundant, so we removed it; and 2) our dataset characterization tool (see below) did not terminate on two of the datasets after several days so we stopped it, and omitted the corresponding datasets from our study. Hence, the results here are for 463 datasets. For each dataset, Figure 1 shows the percentage of improvement of the best AUC score among 20 classification learning algorithms after parameter optimization over the best AUC score among the same classification learning algorithms with their default parameter setting. The data is ordered by increasing value of improvement.

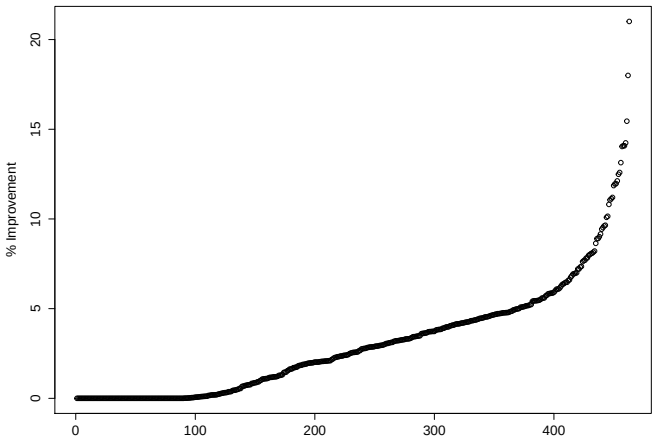

Figure 1. Impact of Parameter Optimization on 463 Datasets

Figure 1 makes it clear that the impact of parameter optimization

[3] We recognize that it is possible to use metalearning to predict rankings of learning algorithms (e.g., see [5, 23]), or even the actual performance of learning algorithms via regression (e.g., see [4, 10, 20]). We restrict our attention here to the prediction of a single best algorithm although much of the discussion extends naturally to these settings.

[4] Some have also simply picked a classification learning algorithm manually, and used metalearning to choose the best parameter setting (e.g., see [9, 19]).

is highly variable across datasets, and seems to be rather small for a large number of them. We are a little surprised that with such a skewed distribution, the authors of [23] concluded that "the result demonstrates the benefit of using the performances of optimised algorithms for generating algorithm rankings", and thus carried on with a blanket application of their combined classification learning algorithm / parameter setting selection approach.

To make the relationship even clearer, consider Figure 2 that shows the cumulative distribution of the same datasets, where each successive bar represents the proportion of datasets for which the improvement in AUC due to parameter optimization is less than or equal to the value indicated on the x-axis in increments of 1%.

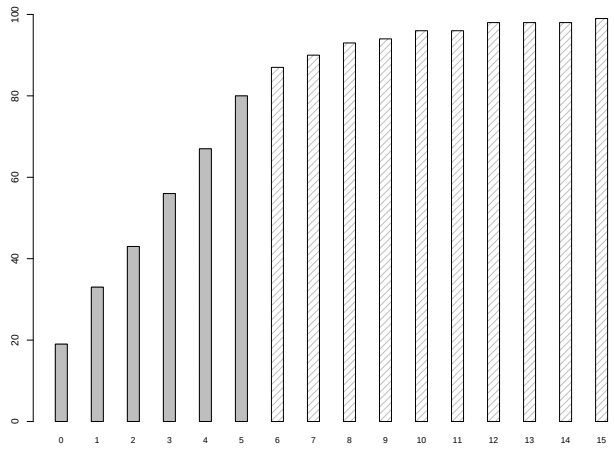

Figure 2. Cumulative Distribution of the Impact of Parameter Optimization on 463 Datasets

According to Figure 2, for 19% of the datasets considered parameter optimization offers no gain in performance. The ascent is actually very steep, as shown by the shaded portion of the distribution, reaching 80% of the datasets for an improvement in performance of no more than 5%. From 0% to 5%, the relationship is virtually linear (r=0.999) with a slope of 12. These results seem robust as an independent analysis of 129 datasets and 9 classification learning algorithms reveals that there is no gain in performance with optimization for about 15% of the datasets and 96% of the datasets exhibit less than 5% improvement.[5]

We note that the above analysis was not performed on a per-algorithm basis. As stated, the differences in performance are computed from among the best in 20 algorithms, which means that the best optimized version could be obtained with algorithm A, while the best default version for the same dataset would be obtained with algorithm B. It is possible, however, that some classification learning algorithms are more sensitive to parameter settings and may thus be more likely to exhibit significant differences with parameter optimization. It may be worthwhile in a future study to consider such algorithm-level variation. In spite of this limitation, several conclusions seem inescapable from the foregoing analysis:

1. Parameter optimization does not improve performance uniformly across datasets.
2. For many datasets, parameter optimization yields very little improvement and may be viewed as computational overkill.
3. For a few datasets, parameter optimization makes a significant difference and should thus be performed.

From a practitioner's standpoint, the last two conclusions are particularly relevant. If no improvement is to be expected from parameter optimization, then one would gladly save the extra computational time required to effect it. Conversely, if a significant improvement can be expected and one is focused on maximizing predictive accuracy, then one would be willing to bear the extra cost. The question then, for such practitioners, is: Will the predictive accuracy on the classification learning task I am considering be improved by parameter optimization?

It should be obvious to machine learning researchers that if one labels the datasets falling under conclusion 2 above as *no advantage* and the datasets falling under conclusion 3 as *advantage*, one obtains a training dataset for a classification learning task. We propose to do exactly that and essentially take our own medicine, by applying machine learning to the problem of distinguishing between datasets that may benefit from parameter optimization and datasets that would not. Because our data consists of information about the performance of learning algorithms, this is an instance of metalearning.

3 Metalearning the Impact of Parameter Optimization

As per standard metalearning practice, we build our training metadata by characterizing each of our 463 datasets by a set of meta-features. However, because smaller datasets may produce unreliable meta-features, we remove from the analysis all of the datasets containing less than 100 instances. This leaves us with 326 datasets.

We use an existing R script that, for each dataset, creates a meta-feature vector by extracting over 68 meta-features, including statistical and information-theoretic meta-features, landmarkers, model-based meta-features, and timing information [18]. Prior to our experiments, we manually remove a number of meta-features that carry little information in this context (e.g., timing data, model-based meta-features, redundant meta-features). In each experiment, we also use correlation-based feature subset selection (CFS) [11], as implemented by the function `CfsSubsetEval` in Weka [12], to further reduce the number of meta-features.

The metalearning task consists in distinguishing datasets where parameter optimization is deemed to offer no advantage (class 0) from datasets where parameter optimization is likely to yield a performance advantage (class 1). Each meta-feature vector is labeled appropriately based on the observed difference in performance over its corresponding dataset. We use various threshold values to separate class 1 from class 0, as shown below.

There are two types of error our (meta)models can make. A false positive (FP) error is made when the model considers a class 0 dataset but labels it as class 1. When this happens, there is wasted effort in performing parameter optimization when no significant gain will be achieved by such optimization. Conversely, a false negative (FN) error is made when the model considers a class 1 dataset but labels it as class 0. When this happens, valuable parameter optimization is omitted and there is a resulting loss in predictive accuracy in the classification task under consideration. Assuming that a practitioner's ultimate goal is to get the best accuracy possible on their specific classification task, one can argue that FN errors are more costly than FP errors, and thus should be minimized. This is, of course, is equivalent to maximizing recall, $R = \frac{TP}{TP+FN}$ (where TP is the number of

[5] There may be some overlap in the datasets used in this study and those used in [23], although the datasets were not transformed into binary classification tasks in the former as they were in the latter.

true positive classifications). Yet, one must be careful as it is trivial to maximize R by simply assigning all classification tasks to class 1. Clearly, this would defeat the purpose of the model and is unacceptable due to the unnecessary computational burden it places on the system. Hence, we focus on obtaining models with high recall, but also good precision, $P = \frac{TP}{TP+FP}$.

We are faced at the metalevel with the same challenge of selecting a (meta)learning algorithm adequate for the task at hand. We are guided here by a desire for comprehensibility of the induced model and the results of preliminary experiments. Hence, we use for our metalearner, a decision tree learning algorithm, specifically Weka's J48, and implement the training and testing processes in RapidMiner [15]. Interestingly, J48 also resulted in higher accuracy than other algorithms such as SVM and Random Forest.

3.1 Threshold = 1.5

We begin by setting the threshold relatively low, assuming that there is an advantage to parameter optimization if the performance improvement exceeds 1.5%.

The default accuracy in this case is 61.96%, obtained by predicting class 1 uniformly. This, of course, produces maximum recall, R=100%, but very poor precision, P=61.96%.

Applying CFS selects the following mixed set of meta-features: `attributes`, `kurtosis`, `joint entropy`, `NB`, `LDA` and `NN1`. Further experimentation shows that performance can be improved still by using only `joint entropy`, `NB` and `NN1`. The confusion matrix is as follows.

		Predicted	
		0	**1**
Actual	**0**	84	40
	1	28	174

This yields an accuracy of 79.17%, significantly above default, with both high recall R=86.14% and high precision P=81.31%. The decision tree is shown below.

```
nn_1 <= 0.902439
|   joint_entropy <= 0.606044: 0 (14.0/2.0)
|   joint_entropy > 0.606044
|   |   naive_bayes <= 0.952: 1 (208.0/30.0)
|   |   naive_bayes > 0.952
|   |   |   nn_1 <= 0.71134
|   |   |   |   joint_entropy <= 2.08461: 0 (2.0)
|   |   |   |   joint_entropy > 2.08461: 1 (5.0)
|   |   |   nn_1 > 0.71134: 0 (12.0/2.0)
nn_1 > 0.902439: 0 (85.0/15.0)
```

To further test the metamodel, we collected 42 independent datasets with more than 100 instances each. It is possible that some of these correspond to some of the tasks used in the training data. However, all 463 training tasks have been binarized [23], while these were left untouched. Hence, we are training on 2-class datasets and testing on n-class datasets, which may be somewhat unfair, but still interesting.

We extracted the meta-features of the 42 datasets, and ran the corresponding meta-feature vectors against the metamodel.[6] The accuracy on the test datasets is 54.76%, which is rather poor given a default accuracy of 73.81%. However, the default is obtained by predicting class 0 uniformly. While the test accuracy is not very good, recall is very high at R=90.91% (compared to R=0 for the default), with 10 of the 11 datasets of class 1 being predicted in class 1. This suggests that the model has picked up useful information to identify learning tasks where parameter optimization would be advantageous (i.e., expected improvement greater than 1.5%).

In addition to its own performance at the metalevel, we consider how the metamodel affects performance at the base level. To do so, we compare $maxImp$, the total amount of possible improvement due to parameter optimization to $predImp$, the amount of improvement one would obtain by using the metamodel. For each dataset d, let l_d be d's label, p_d be d's predicted class, and I_d be the improvement one would experience if parameter optimization were used with d. Then,

$$maxImp = \sum_d I_d$$

i.e., $maxImp$ is the sum of the individual improvement values across all of our 326 datasets. Here, $maxImp = 949.26$. Similarly,

$$predImp = \sum_d \begin{cases} I_d & \text{if } (p_d = l_d) \wedge (l_d = 1) \\ 0 & \text{otherwise} \end{cases}$$

i.e., $predImp$ is the sum of the improvement values for all datasets where the metamodel makes the correct class 1 prediction. We do not include correct class 0 predictions as these would artificially inflate the results. Here, $predImp = 789.92$. Hence, the metamodel would allow us to claim 83.21% of the total performance improvement available at the base level.

3.2 Threshold=2.5

We next raise the threshold, assuming that there is an advantage to parameter optimization if the performance improvement exceeds 2.5%.

The default accuracy in this case is 50.31%, obtained by predicting class 1 uniformly. This, again, produces maximum recall, R=100%, but very poor precision, P=50.31%.

Applying CFS selects the following mixed set of meta-features: `kurtosis prep`, `normalized attribute entropy`, `joint entropy`, `NB`, `LDA`, `stump min gain` and `NN1`. Further experimentation shows that performance can be improved still by using only `kurtosis prep`, `normalized attribute entropy`, `joint entropy`, `NB` and `NN1`. The confusion matrix is as follows.

		Predicted	
		0	**1**
Actual	**0**	105	57
	1	26	138

This yields an accuracy of 74.52%, significantly above default, with both high recall R=84.15% and high precision P=70.77%. The decision tree is shown below.

```
nn_1 <= 0.857143
|   joint_entropy <= 0.606044
|   |   joint_entropy <= 0.508598
|   |   |   kurtosis_prep <= 26.479217: 1 (2.0)
|   |   |   kurtosis_prep > 26.479217: 0 (2.0)
|   |   joint_entropy > 0.508598: 0 (8.0)
|   joint_entropy > 0.606044
|   |   naive_bayes <= 0.969466: 1 (194.0/49.0)
|   |   naive_bayes > 0.969466
|   |   |   normalized_attribute_entropy <= 0.84991
|   |   |   |   kurtosis_prep <= 45.518635: 0 (2.0)
|   |   |   |   kurtosis_prep > 45.518635: 1 (2.0)
|   |   |   normalized_attribute_entropy > 0.84991: 0 (7.0)
nn_1 > 0.857143: 0 (109.0/15.0)
```

[6] As the results for the test datasets were obtained with a different set of classification learning algorithms and a different parameter optimization procedure, we are not entirely sure the comparison is fair. Yet, we feel there is value in including these results.

As with the threshold value of 1.5, using the metamodel against the test datasets produces poor accuracy (35.71% for a default of 88.10%), but recall is significantly better with R=60% (3 of the 5 datasets in class 1 are predicted correctly) against a default of R=0.

Considering performance at the base level, we have here $predImp = 698.61$, so that the metamodel would allow us to still claim 73.60% of the total performance improvement available.

3.3 Larger Threshold Values

Based on the distribution of performance improvements, raising the threshold will cause the majority class to shift to 0 and result in far fewer class 1 datasets. On the other hand, while fewer, correctly identifying these datasets offers the highest benefit to the practitioner as the expected improvement in accuracy with parameter optimization is rather significant. However, the metalearning task is also more difficult as the class distribution of the training data is rather skewed.

Setting the threshold to 5.0% results in a default accuracy of 83.44% with a model that predicts class 0 uniformly. While it was not possible in this case to improve on the default accuracy with the available set of meta-features, recall rose to R=25.94% (10 of the 54 datasets in class 1 are predicted correctly). Interestingly, CFS selected `LDA` and `NN1`, and while the tree induced without feature selection is a little larger than the previous ones, it also splits on `NN1` at the root node. Using the metamodel against the test datasets produces reasonable accuracy (76.19%) but only because the test data is skewed in favor of class 0. Only two datasets belong to class 1 and neither is predicted correctly.

Setting the threshold to 10.0% results in a default accuracy of 96.93% again with a model that predicts class 0 uniformly. Only 10 datasets belong to class 1, making for a very imbalanced class distribution. Using all the meta-features, the induced decision tree has a slightly improved accuracy (97.85%), with recall R=30%. The root of the tree is `mutual information`. There are no datasets in our test set for which the improvement with parameter optimization exceeds 10%.

4 Discussion

As mentioned previously, one of the biggest issues with parameter optimization is the amount of time it takes to optimize the parameters. In [24], it took 6,000 single core CPU hours to optimize the parameters of all their datasets, which means that it took on average about 13 hours per dataset. For obvious reasons, a practitioner would probably not wish to wait for that amount of time when it may result in little or no improvement from the models baseline performance.

What this short study demonstrates is that it is possible to predict, with good recall value, whether parameter optimization will significantly increase a classification learning models accuracy by using the metafeatures of any particular dataset, which are less computationally intensive than the parameter optimization procedure. Of course, as the threshold increases, or as the expected performance improvement gets larger, the prediction becomes less accurate. Nevertheless, it is better than default.

One of the unexpected findings of our study, valid across almost all metamodels, is that the root node of the decision tree is `NN1`. In other words, a dataset's performance on `NN1` is indicative of whether parameter optimization will significantly improve the performance of learning algorithms trained against it. There are at least two interesting things about this finding. First, it confirms prior work on metalearning that suggest that landmarking meta-features are generally better than other meta-features, especially `NN1` and `NB` [20]. Second, and significantly more valuable, is that `NN1` is a very efficient algorithm, suggesting that it would be very fast to determine whether parameter optimization is likely to improve accuracy for any dataset.

To further test the predictive power of `NN1`, we ran Weka's linear regression on our dataset, without CFS, and with the target set to the actual percentage of accuracy improvement (`Imp`) observed for each dataset. The resulting model is shown below.

```
Imp =
      11.4405 * class_prob_min +
       0.1626 * skewness_prep +
      -0.0051 * kurtosis_prep +
      -2.003  * cancor_1 +
      -6.3391 * class_entropy +
      -8.2398 * mutual_information +
      -0.0008 * noise_signal_ratio +
      -0.5321 * attribute_mean +
      -5.7852 * stump_min +
      -4.8382 * stump_max +
      13.3081 * stump_mean +
       8.4397 * stump_sd +
       3.3769 * stump_min_gain +
      -7.5918 * nn_1 +
       6.4233
```

The correlation coefficient of the model is 0.40, and the relatively large multiplicative factor associated with `NN1` in the model suggests its influence on the value of `Imp`. Furthermore, if we regress `Imp` on `NN1` alone, we obtain the following very simple model.

```
Imp =
      -8.3476 * nn_1 +
       9.3173
```

The correlation coefficient of this simpler model is 0.44. There is still error in this model, and it will be interesting to see whether using only the `NN1` metafeature could in general reasonably predict the datasets improvement if classification learning algorithm parameters are optimized.

5 Conclusion

We have showed that the observed improvement in performance due to parameter optimization in classification learning has a rather skewed distribution, with most datasets seeing none or very little improvement. However, there remains a portion of datasets for which parameter optimization has a significant impact. We have used metalearning to build a model capable of predicting whether parameter optimization can be expected to improve classification accuracy. As a result, it would seem that both current approaches to metalearning, the one that ignores parameter optimization and considers only default settings, and the other that applies parameter optimization in all cases, are misinformed. Instead, a two-phase approach where one first determines whether parameter optimization is likely to produce an improvement, and then if so applies it, is more appropriate.

Furthermore, one unexpected and very interesting side effect of our metalearning study is that our results show that good recall can be achieved at the metalevel, and that, with a very small set of efficient meta-features. In particular, it would appear that `NN1` may serve as a great landmarker in this context.

For future work, it may be valuable to revisit error costs. In particular, we have argued here that FN errors were more costly than FP errors on the basis that practitioners are likely to wish to obtain the highest possible accuracy on their classification tasks. This is certainly also true when the computational cost of parameter optimization is prohibitive. Were computational cost not to matter, the best decision would be to always perform parameter optimization, and

there would consequently be no need for a metamodel. In practice, however, it is well-known that parameter optimization is computationally intensive. As a result, FP errors may actually be rather costly, and possibly even more so than FN errors. In any case, there would seem to exist a range of operating conditions between these extremes, or different cost-benefits between FN and FP associated with parameter optimization. It may be interesting to perform a cost-sensitive analysis at the metalevel, for example, relating expected gain in performance (or threshold) and incurred cost. We cannot do this with the data available since we only have final performance improvement (after some fixed search time). We would need to re-run experiments to gather information about improvement over time allowed for optimization.

REFERENCES

[1] Bensusan, H. (1998). Odd Bites into Bananas Don't Make You Blind: Learning about Simplicity and Attribute Addition. In *Proceedings of the ECML'98 Workshop on Upgrading Learning to the Meta-level: Model Selection and Data Transformation*, 30-42.

[2] Bensusan, H. and Giraud-Carrier, C. (2000). Discovering Task Neighbourhoods through Landmark Learning Performances. In *Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases, LNAI 1910*, 325-330.

[3] Bensusan, H., Giraud-Carrier, C. and Kennedy, C. (2000). A Higher-order Approach to Meta-learning. In *Proceedings of the ECML-2000 Workshop on Meta-learning: Building Automatic Advice Strategies for Model Selection and Method Combination*, 109-118.

[4] Bensusan, H. and Kalousis, A. (2001). Estimating the Predictive Accuracy of a Classifier. In *Proceedings of the Twelfth European Conference on Machine Learning (LNCS 2167)*, 25-36.

[5] Brazdil, P., Soares, C. and Pinto da Costa, J. (2003). Ranking Learning Algorithms: Using IBL and Meta-Learning on Accuracy and Time Results. *Machine Learning*, **50**(3):251-277.

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

[7] Engels, R. and Theusinger, C. (1998). Using a Data Metric for Offering Preprocessing Advice in Data-mining Applications. In *Proceedings of the Thirteenth European Conference on Artificial Intelligence*, 430-434.

[8] Fuernkranz, J. and Petrak, J. (2001). An Evaluation of Landmarking Variants. In *Proceedings of the ECML/PKDD-01 Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-learning*, 57-68.

[9] Gomes, T.A.F., Prudêncio, R.B.C., Soares, C., Rossi, A.L.D. and Carvalho, A. (2012). Combining Meta-learning and Search Techniques to Select Parameters for Support Vector Machines. *Neurocomputing*, **75**:3-13.

[10] Guerra, S.B., Prudêncio, R.B.C. and Ludermir, T.B. (2008). Predicting the Performance of Learning Algorithms Using Support Vector Machines as Meta-regressors. In *Proceedings of the Eighteenth International Conference on Artificial Neural Networks (LNCS 5163)*, 523-532.

[11] Hall, M.A. (1999). Correlation-based Feature Subset Selection for Machine Learning. PhD Thesis, Department of Computer Science, The University of Waikato, Hamilton, New Zealand.

[12] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I.H. (2009). The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, **11**(1):10-18.

[13] Leite, R., Brazdil, P. and Vanschoren, J. (2012). Selecting Classification Algorithms with Active Testing. In *Proceedings of the Eighth International Conference on Machine Learning and Data Mining in Pattern Recognition (LNCS 7376)*, 117-131.

[14] Michie, D., Spiegelhalter, D.J. and Taylor, C.C. (1994). *Machine Learning, Neural and Statistical Classification*, Ellis Horwood.

[15] North, M. (2012). *Data Mining for the Masses*, Global Textbook Project.

[16] Peng, Y., Flach, P.A., Brazdil, P. and Soares, C. (2002). Improved Data Set Characterisation for Meta-learning. In *Proceedings of the Fifth International Conference on Discovery Science*, 141-152.

[17] Pfahringer, B., Bensusan, H. and Giraud-Carrier, C. (2000). Meta-learning by Landmarking Various Learning Algorithms. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 743-750.

[18] Reif, M. (2012). A Comprehensive Dataset for Evaluating Approaches of various Meta-Learning Tasks. In *Proceedings of the First International Conference on Pattern Recognition Applications and Methods*, 273-276.

[19] Reif, M., Shafait, F. and Dengel, A. (2012). Meta-learning for Evolutionary Parameter Optimization of Classifiers. *Machine Learning*, **87**(3):357-380.

[20] Reif, M., Shafait, F., Goldstein, M., Breuel, T. and Dengel, A. (2014). Automatic Classifier Selection for Non-experts. *Pattern Analysis & Applications*, **17**(1):83-96.

[21] Smith, M.R., Mitchell, L., Giraud-Carrier, C. and Martinez, T. (2014). Recommending Learning Algorithms and Their Associated Hyperparameters. Submitted.

[22] Sohn, S.Y. (1999). Meta Analysis of Classification Algorithms for Pattern Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **21**(11):1137-1144.

[23] Sun, Q. and Pfahringer, B. (2013). Pairwise Meta-rules for Better Meta-learning-based Algorithm Ranking. *Machine Learning*, **93**(1):141-161.

[24] Thornton, C., Hutter, F., Hoos, H.H. and Leyton-Brown, K. (2013). Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms. In *Proceedings of the Nineteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 847-855.