To tune or not to tune: recommending when to adjust SVM hyper-parameters via Meta-learning

Rafael G. Mantovani *, André L. D. Rossi †, Joaquin Vanschoren ‡, Bernd Bischl § and André C. P. L. F. Carvalho *
* Universidade de São Paulo (USP), São Carlos - SP, Brazil
Email: {rgmantov, andre}@icmc.usp.br
† Universidade Estadual Paulista (UNESP), Itapeva - SP, Brazil
Email: alrossi@itapeva.unesp.br
‡ Eindhoven University of Technology (TU/e), Eindhoven, The Netherlands
Email: j.vanschoren@tue.nl
§ Ludwig-Maximilians-University Munich, Germany
Email: bernd.bischl@stat.uni-muenchen.de

***Abstract*—Many classification algorithms, such as Neural Networks and Support Vector Machines, have a range of hyper-parameters that may strongly affect the predictive performance of the models induced by them. Hence, it is recommended to define the values of these hyper-parameters using optimization techniques. While these techniques usually converge to a good set of values, they typically have a high computational cost, because many candidate sets of values are evaluated during the optimization process. It is often not clear whether this will result in parameter settings that are significantly better than the default settings. When training time is limited, it may help to know when these parameters should definitely be tuned. In this study, we use meta-learning to predict when optimization techniques are expected to lead to models whose predictive performance is better than those obtained by using default parameter settings. Hence, we can choose to employ optimization techniques only when they are expected to improve performance, thus reducing the overall computational cost. We evaluate these meta-learning techniques on more than one hundred data sets. The experimental results show that it is possible to accurately predict when optimization techniques should be used instead of default values suggested by some machine learning libraries.**

I. Introduction

The predictive performance of a Machine Learning (ML) algorithm is usually influenced by the choice of the values of its hyper-parameters. Several studies have been investigating optimization techniques to optimize the hyper-parameters of Support Vector Machines (SVMs) for data classification problems [1]–[3]. However, the optimization of SVMs' hyper-parameters usually has a high computational cost, since a large number of candidate solutions needs to be evaluated. Some tools implementing this ML algorithm suggest default values for the hyper-parameters. For some data sets, the use of these default values produce classification models with good predictive performance. Therefore, deciding if the optimization of the hyper-parameters will improve model accuracy compared to the default values is an important decision to reduce the computation cost.

This study investigates the development of a recommendation system able to predict whether a hyper-parameter tuning process is necessary when SVMs are applied to a new data set. This recommendation system is based on Meta-learning (MTL) [4] ideas to induce a classification model that, based on the characteristics of a data set, recommends the tuning of the hyper-parameters or the use of default values. MTL has frequently been employed to select [5], rank [6], or predict [7] the performance of ML algorithms on a new data set.

The recommendation system proposed here consists of three steps. First, we create a meta-data set where the predictive features consist of characteristics (meta-features) from many data sets, and the target feature indicates whether or not optimization techniques proved useful on that data set. Next, a meta-model is induced from this meta-data set. Then, this model can be employed to predict whether an optimization technique should be used to tune the hyper-parameter values for a new data set.

Three meta-heuristics for parameter optimization were investigated in the experiments: Genetic Algorithm (GA) [8], Particle Swarm Optimization (PSO) [9] and Estimation of Distribution Algorithms (EDA) [10]. GA and PSO are often explored in related work, while EDA has attracted the attention of the community in recent years. Of course, there exist many more techniques for optimization, even some that are more time-efficient [11], [12], but their optimization performance is typically similar to that of the meta-heuristics used here, given sufficient computation time. We compare the performance of the optimized models with the models generated using default hyper-parameter values provided by Weka [13] and LIBSVM [14].

This paper is structured as follows: section II contextualizes the hyper-parameter tuning problem and presents related work. Section III presents our experimental methodology and steps covered to obtain the results, which are discussed in section IV. The last section presents our conclusions and future work.

II. Related work

Finding a good configuration for the hyper-parameters of a ML algorithm requires specific knowledge, intuition and, often, trial and error. In [15], the tuning of hyper-parameters is seen as an optimization problem, aiming to optimize the predictive performance (e.g., accuracy) of the models induced by the algorithm.

978-1-4799-1959-8/15/$31.00 ©2015 IEEE

Many deterministic and probabilistic techniques have been proposed for the optimization of hyper-parameters of ML algorithms. Among the deterministic techniques, Grid Search (GS) is often used due to its simplicity and good results in previous studies [16]. GS is an exhaustive search method that requires the discretization of the hyper-parameters space. Hence, if execution time is important, it may not be a good choice. It may also yield suboptimal results for numeric hyper-parameters because not all values are considered. Although more robust deterministic techniques have been proposed, GS is still the most used [3].

For optimization problems of high dimensionality (with many hyper-parameters) and large data sets, GS becomes computationally infeasible. In these scenarios, probabilistic optimization methods, such as GAs, are generally preferred [17]. Other authors explored the use of Pattern Search (PS) [18], gradient descent [1], or simple but effective techniques such as Random Search (RS) [16]. Other approaches are model-based, and aim to model the effects of parameters based on the outcomes of previous evaluations. These include local search (ParamILS [19]), estimation of distributions (REVAC [20]) and Bayesian optimization [21].

MTL has also been used to adjust the hyper-parameters of ML algorithms. One approach is to regard different parameter settings as independent algorithms and predict the best setting based on characteristics of the data set in question [22], [23]. In these cases, the parameter settings are predicted without actually evaluating the model on the new data set. In [7], [24], MTL is used to estimate the training time of classification algorithms for different hyper-parameter configurations.

Other studies combined MTL with optimization techniques for the selection of hyper-parameter values [25]–[28]. In these studies, MTL recommends hyper-parameter values for the initial population of a search technique, leading optimization methods to a faster convergence.

Another MTL approach, named Active Testing [29], [30], selects the algorithm and its hyper-parameters simultaneously. It uses the evaluations of all hyper-parameter settings tested on the new data set, and all evaluations on prior data sets, to select the best new candidate algorithm-parameter combination in the next iteration. This procedure has also been employed for the hyper-parameter tuning of SVMs specifically [31].

The use of MTL to predict when hyper-parameters should be tuned was also studied by [32]. However, the investigation performed here contains some important differences:

- meta-labels were defined according to a conservative rule, instead of an empirical threshold [32];
- we analyzed a larger number of meta-heuristics (MTHs) in order to verify whether this SVM tuning problem is dependent on the technique;
- an additional set of meta-features, namely data complexity meta-features, was investigated in this paper;
- in evaluating our techniques, we used two alternative default hyper-parameter values as baselines
- although we used a smaller quantity of data sets (143 vs. 326), we considered both binary and multi-class problems instead of only binary problems.

TABLE I. SVM HYPER-PARAMETER RANGE VALUES ADOPTED [34].

Hyper-parameter	Minimum	Maximum
cost (C)	2^{-2}	2^{15}
gamma (γ)	2^{-15}	2^{3}

III. EXPERIMENTAL METHODOLOGY

To predict and understand in which cases optimizing SVM hyper-parameters is better than using the default values, we evaluate the predictive performance of models induced by SVMs on 143 public data sets using the meta-heuristics (MTHs) GA, PSO and EDA to tune SVM's hyper-parameters. The predictive performance is compared to the default values provided by the Weka [13] and LIBSVM library [14] (labelled DF-WEKA and DF-LIBSVM, respectively).

For the MTHs, each individual is a pair of real values representing the SVM hyper-parameter C (cost) and the width of the Gaussian kernel γ. Only the Gaussian kernel is considered here because it usually achieves good performance, can handle nonlinear decision boundaries, and has less numerical difficulties than other kernel functions (e.g. the value of the polynomial kernel may be infinite) [33]. The predictive accuracy obtained by the induced model was used as the fitness measure for all optimization techniques. Higher fitness values indicate better predictive performance, i.e., better hyper-parameter values. Table I shows the range of values for C and γ [34].

Figure 1 illustrates the MTL framework. In the hyper-parameter tuning process (Item 2), the performance of the base-level SVM models on the data sets (Item 1) is evaluated using a k fold cross-validation [27]. This k-CV methodology splits each data set into training, validation and test sets. Whenever a tuning technique is executed, the data set is divided into k stratified folds. For each candidate solution found by a MTH, the SVM technique is trained with $k-2$ folds (training folds). One of the remaining folds, referred to as validation fold, is used to select the optimal hyper-parameter setting, i.e., the setting that induced the model with the best predictive performance for this validation fold. Finally, the other remaining fold, referred as test fold, is used to evaluate the optimized model on new, previously unseen, data.

A. Data sets

For the experiments, 143 classification data sets with different characteristics were collected from the UCI repository (Figure 1 - Item 1). These data sets, listed in Table II, were characterized by extracting a set of meta-features (Figure 1 - Item 3), each describing an aspect of the data set (see Section III-C).

B. Hyper-parameter tuning process

In a hyper-parameter optimization task, time is an important factor to be considered in practical scenarios. Several authors have mentioned that the tuning process may take many hours to find good hyper-parameter values for a single data set [27], [32]. Thus, researchers and users of ML algorithms frequently do not tune their hyper-parameters.

We have performed MTH optimization using different budget sizes, from 50 to 10 000 evaluations of the fitness function. Results suggested that when we increase the number of

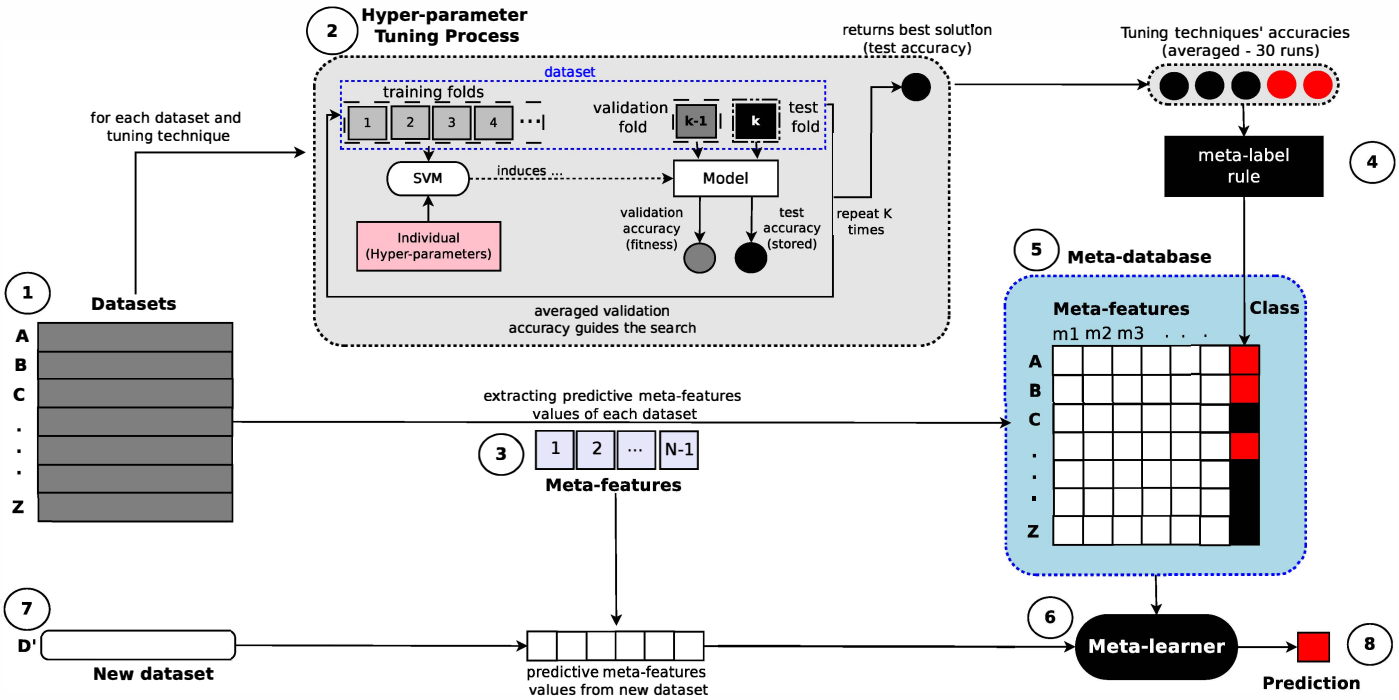

Fig. 1. Overview of the evaluation procedure Meta-learning for hyper-parameter prediction (when or not to tune).

TABLE II. UCI DATA SETS USED IN EXPERIMENTS

Data sets				
1. abalone-3class	**2.** abalone-7class	**3.** abalone-11class	**4.** abalone-28class	**5.** accute-inflammations-nephritis
6. acute-inflammations-urinary	**7.** annealing	**8.** appendicitis	**9.** arrhythmia	**10.** audiology
11. autoUniv-au1-100	**12.** autoUniv-au4-2500	**13.** autoUniv-au6-1000	**14.** autoUniv-au6-drift-au6-cd1-500	**15.** autoUniv-au6-cd1-400
16. autoUniv-au7-300-drift-au7-cpd1-800	**17.** autoUniv-au7-700	**18.** autoUniv-au7-cpd1-500	**19.** balance-scale	**20.** balloons-adult+stretch
21. ballons-adult-stretch	**22.** ballons-yellow-small+adult-stretch	**23.** ballons-yellow-small	**24.** banana	**25.** banknote-authentication
26. blogger	**27.** blood-transfusion-service	**28.** breast-cancer-wisconsin	**29.** breast-tissue-4class	**30.** breast-tissue-6class
31. bupa	**32.** climate-simulation-craches	**33.** cmc	**34.** colon32	**35.** connectionist-mines-vs-rocks
36. connectionist-vowell-reduced	**37.** connectionist-vowell	**38.** credit-approval	**39.** dbworld-subjects-stemmed	**40.** dermatology
41. digits2	**42.** dresses-sales	**43.** ecoli	**44.** energy-efficienty-yl	**45.** energy-efficiency-y2
46. fertility-diagnosis	**47.** flags	**48.** flare	**49.** glass	**50.** glioma16
51. habermans-suervival	**52.** heart-disease-processed-cleveland	**53.** heart-disease-processed-hungarian	**54.** heart-disease-processed-switzerland	**55.** heart-disease-processed-va
56. heart-disease-reprocessed-hungarian	**57.** hepatitis	**58.** hill-valey-with-noise	**59.** hill-valley-without-noise	**60.** horsec-olic-surgical
61. indian-liver-patient	**62** ionosphere	**63.** iris	**64.** kr-vc-kp	**65.** leaf
66. led7digit	**67.** lenses	**68.** leukemia-haslinger	**69.** lymphography	**70.** molecular-promotor-gene
71. monks1	**72.** monks2	**73.** monks3	**74.** parkinsons	**75.** pima-indian-diabetes
76. planning-relax	**77.** qsar-biodegradation	**78.** qualitative-bankruptcy	**79.** robot-failure-lpl	**80.** robot-failure-lp2
81. robot-failure-lp3	**82.** robot-failure-lp4	**83.** robot-failure-lp5	**84.** robot-nav-sensor-readings-2	**85.** robot-nav-sensor-readings-24
86. robot-nav-sensor-readings-4	**87.** saheart	**88.** seeds	**89.** seismic-bumps	**90.** semeion
91. shuttle-landing-control	**92.** spambase	**93.** spect-heart	**94.** spectf-heart	**95.** spectometer
96. statlog-australian-credit	**97.** statlog-german-credit-numeric	**98.** statlog-german-credit	**99.** statlog-heart	**100.** statlog-image-segmentation
101. statlog-landsat-satellite	**102.** statlog-vehicle-silhouettes	**103.** steel-plates-faults	**104.** systhetic-control	**105.** teaching-assistant-evaluation
106. texture	**107.** thoracic-surgery	**108.** thyroid-alldp	**109.** thyroid-allhyper	**110.** thyroid-allhypo
111. thyroid-allrep	**112.** thyroid-ann	**113.** thyroid-dis	**114.** thyroid-hypothyroid	**115.** thyroid-newthyroid
116. thyroid-sick-euthyroid	**117.** thyroid-sick	**118.** tica-tac-toe	**119.** trains	**120.** turkiye-student
121. user-knowledge	**122.** vertebra-column-2c	**123.** vertebra-column-3c	**124.** volcanoes-a1	**125.** volcanoes-a2
126. volcanoes-a3	**127.** volcanoes-a4	**128.** volcanoes-e1	**129.** volcanoes-e2	**130.** volcanoes-e3
131. volcanos-e4	**132.** volcanos-e5	**133.** voting	**134.** waveform-vl	**135.** waveform-v2
136. wdbc	**137.** wholesale-channel	**138.** wholesale-region	**139.** wilt	**140.** wine-quality-red
	141. wine-quality-white-5class	**142.** wine	**143.** wpbc	

evaluations, the generalization power (test accuracy) decreases for imbalanced data sets, and improves for more balanced data sets. This behavior suggests that overfitting occurs for the imbalanced data sets while there were small gains for balanced data sets.

Furthermore, we noted that while we allowed a large number of optimization iterations, MTHs already converged to a common solution after few iterations. This result can be explained by the relatively small search space, and consequently, the low complexity of the optimization problem, since we are tuning only two hyper-parameters. Based on these results, we decided to use a restricted budget of 200 evaluations when performing experiments with SVMs in particular.

Returning to Figure 1, the hyper-parameter tuning process (Item 2) provides the information needed to define the target values (labels) on the meta-data set. The target values depend on the predictive performance of the induced models with and without hyper-parameter tuning by MTHs. In the experiments, all MTHs were run over the 143 data sets for SVM hyper-parameter tuning with the same initial configuration: 20 individuals in the initial population and at most 10 iterations. Thus, the MTHs can evaluate 200 different combinations of hyper-parameter values (individuals) per execution.

The MTHs GA, PSO and EDA were implemented in R using packages ”GA”, ”pso”, and ”copulaedas”, respectively.

TABLE III. WIN-TIE-LOSS OF THE OPTIMIZATION TECHNIQUES FOR 143 DATA SETS.

Technique	Win	Tie	Loss
GA	2	121	20
PSO	1	125	17
EDA	2	117	24
DF-LIBSVM	0	10	123
DF-WEKA	2	20	121

They available on CRAN[1]. The parameter values for GA and PSO were chosen base on [34]. We used an EDA with a Gaussian copula function (GCEDA) [10]. Its parameter values are the default values provided by the package.

Each technique was run 30 times for each data set, returning the mean and standard deviation of the accuracies on both the validation and test partitions. In each of the executions, the initial population of MTHs was started randomly. The predictive performance of the models induced by the tuned SVMs was compared with models induced by SVMs using the default values provided by DF-WEKA and DF-LIBSVM.

To statistically analyse the optimization effect over all data sets, the Friedman statistical test with the Nemenyi post-hoc test and a confidence level of 95% was applied. According to the test, all the MTHs found overall significantly better hyper-parameter values than the default values (DF-WEKA, DF-LIBSVM). Moreover, the test showed that there is no significant difference in the predictive performance among the MTHs. All tuning techniques presented very similar accuracies, with a small standard deviation (about 0.01 to 0.03) over the executions. Table III shows win-tie-loss occurrences in base-level learning. A technique 'wins' if its Friedman rank on a dataset is one *critical difference* better than the next technique. The critical difference is defined by the Nemenyi test over all datasets.

The use of MTHs to tune the hyper-parameters increased the computational cost by around 600 times compared to the use of the default values. Therefore, for some data sets, the default parameter values can be used to significantly lower the computational cost, with relatively little accuracy loss.

C. Meta-features

The meta-data set (Figure 1 - Item 5) is constructed out of the meta-features (Figure 1 - Item 3) of the 143 data sets, and the results of the SVM hyper-parameter tuning process described earlier. Since each data set results in one meta-example, the meta-data set is composed out of 143 unlabeled meta-examples, each one representing one of the original data sets. Two meta-data sets were created according to the meta-features used to describe the data sets:

- one with 17 STATLOG meta-features, originally used in the STATLOG project [35], and later in many similar studies [22], [25], [26]. These meta-features are simple measures based on statistics, such as the number of classes, number of attributes, class distribution and so on;
- one with 13 data complexity meta-features: based on data complexity measures [36], [37] that have been explored in some recent work [38]–[40]. These measures analyze the complexity of a classification problem considering the overlap in the feature values, the separability of the classes, and geometry/topological properties.

[1] http://cran.r-project.org/

TABLE IV. META-FEATURES USED IN EXPERIMENTS.

Type	Name	Meta-feature
STAT	#cls	Number of classes
STAT	#attr	Number of attributes
STAT	#NumAttr	Number of numeric attributes
STAT	#NomAttr	Number of nominal (symbolic) attributes
STAT	#smpl	Number of samples
STAT	#dim	Dimensionality (attributes / samples)
STAT	#NumRate	Numerical attributes rate
STAT	#NomRate	Nominal attributes rate
STAT	#SymMin	Minimum number of levels of the symbolic attributes
STAT	#SymMax	Maximum number of levels of the symbolic attributes
STAT	#SymAvg	Average number of levels of the symbolic attributes
STAT	#SymSd	Standard deviation of levels of the symbolic attributes
STAT	#SymSum	Total number of levels of the symbolic attributes
STAT	%Cmin	Percentage of elements of the minority class
STAT	%Cmax	Percentage of elements of the majority class
STAT	%CMean	Average number of elements by class
STAT	%CSd	Standard deviation of elements by class
COMP	F1	Fisher's discriminant ratio
COMP	F2	Overlapping the per-class bounding boxes
COMP	F3	Maximum individual feature efficiency
COMP	L1	Distance of erroneous instances to a linear classifiers
COMP	L2	Training error of a linear classifier
COMP	L3	Non-linearity of a linear classifier
COMP	N1	Fraction of points lying on the class boundary
COMP	N2	Average intra/inter class nearest neighbor distances
COMP	N3	LOO error rate of the 1-NN
COMP	N4	Non-linearity of the 1-NN
COMP	T1	Fraction of maximum covering spheres on data
COMP	F1v	Fisher's discriminant ratio
COMP	F4	Collective Feature Efficiency

All the meta-features are listed in Table IV. The last meta-feature is the target, whose values indicate if the hyper-parameter tuning improves the predictive performance of the model or not. The target values are equal for both meta-databases. These values were defined according to the rule described in the next subsection.

D. Meta-data set

In order to label the meta-examples of the meta-data set, we followed a confidence interval rule (Figure 1 - Item 4). Given a data set x, the best MTH M and the best DF D obtained for x, the deviation of M and D techniques are defined by:

$$\begin{array}{rcl} M_{dev}(x) & = & \mu(M(x)) - \sigma(M(x)) \\ D_{dev}(x) & = & \mu(D(x)) + \sigma(D(x)) \end{array} \quad (1)$$

where μ is the averaged accuracy and σ is the standard deviation over the 30 executions of the technique. A meta-example x' created from x receives a label according to the rule:

$$L(x') = \begin{cases} \text{DF}, & \text{if } (M_{dev}(x') - D_{dev}(x')) \leq 0, \\ \text{TUN}, & \text{otherwise.} \end{cases} \quad (2)$$

Thus, if the difference between the ranges of the best MTH (black ball on Figure 1 - Item 4) and best DF (red ones) for a meta-example is smaller than or equal to zero, the meta-example receives the label *'DF'*. Otherwise, it receives the label *'TUN'*, meaning that a tuning step is recommended. With these labels and the meta-features previously computed, we obtain the meta-data set.

E. Meta-learner

Six ML classification algorithms were used as meta-learners (Figure 1 - Item 6): J48 Decision Tree (J48), Naïve Bayes (NB), k-Nearest Neighbors (k-NN) with $k = 3$, Multilayer Perceptron (MLP), Random Forest (RF) and Support Vector Machines (SVM). These techniques follow different learning paradigms, each representing a distinct bias, and may result in different predictions.

An ensemble was also built with the predictions from these classifiers (ENS). The ensemble prediction is defined by majority voting. Since there is an even number of classifiers, ties are broken by choosing the majority class (*'TUN'*). The meta-learners were evaluated based on the following measures: classification error rate, precision, recall and F-Score.

F. MTL experiments

Based on our confidence interval rule (2), 46 of the 143 data sets were labeled with the *DF* class: the induced models presented a predictive performance similar to those induced when the hyper-parameters were tuned by the optimization techniques. The other 97 meta-examples received the label of the tuning (*TUN*) class. Due to the small number of meta-examples, the Leave-One-Out Cross-Validation (LOO-CV) methodology was adopted to evaluate the predictive performance of the meta-learners.

At each LOO-CV iteration a single meta-example (testing partition) represents the new unknown data set (Figure 1 - Item 7) for which the meta-features are known but not the actual target label (to tune or not to tune). The training partition of the meta-learner uses all 142 remaining meta-examples, already labeled according to the meta-label rule. Thus, the meta-learner will predict whether the use of *default values* for SVM hyper-parameters on the new test data set is the best alternative (Figure 1 - Item 8). The accuracy measures of the meta-learner are averaged over all LOO-CV iterations/executions.

IV. Experimental results

According to some studies, the hyper-parameters of SVMs should always be tuned when looking for the best predictive performance [14], [25], [27]. Conversely, other studies reported experimental results where the default values provided predictive performances similar to those obtained by optimized hyper-parameters [3]. In this section, the main empirical results are presented and discussed. We run meta-learners twice considering the two meta-data sets: one with simple STATLOG measures, and another one with data complexity measures.

A. MTL predictive performance

Table V summarizes the predictive performance obtained by the meta-learners. The first column contains the ML algorithms used as meta-learner. The second one emphasizes the meta-feature set used to describe the meta-examples. Subsequent columns present the results for different performance measures: mean classification error rate, precision, recall, and F-Score. Since the meta-database is imbalanced, with 32% of the examples in the minority class, a trivial classifier would have a mean classification error rate equal to 0.322.

TABLE V. Meta-learning results with STATLOG and Complexity meta-features and LOO-CV.

Classifier	Meta-feature Set	Error	Precision	Recall	F-Score
J48	STATLOG	0.252	0.667	0.435	0.526
MLP	STATLOG	0.266	0.643	0.391	0.486
NB	STATLOG	0.455	0.408	0.913	0.564
3-NN	STATLOG	0.238	0.625	0.652	0.638
RF	STATLOG	0.196	0.765	0.565	0.650
SVM	STATLOG	0.245	0.867	0.283	0.426
ENS	STATLOG	0.182	0.750	0.652	0.698
J48	COMPLEXITY	0.308	0.538	0.304	0.389
MLP	COMPLEXITY	0.266	0.611	0.478	0.537
NB	COMPLEXITY	0.476	0.369	0.674	0.477
3-NN	COMPLEXITY	0.266	0.571	0.696	0.627
RF	COMPLEXITY	0.259	0.645	0.435	0.519
SVM	COMPLEXITY	0.203	0.947	0.391	0.554
ENS	COMPLEXITY	0.252	0.639	0.500	0.561

In this recommendation problem, a *false positive* (FP) is a wrong recommendation to use default hyper-parameter values and a *false negative* (FN) is a wrong recommendation to use tuned hyper-parameter values. If the predictive performance is more important that runtime, a *'false positive - FP'* is considered worse than a FN, because it would result in SVMs with lower predictive performance. In this case, we should look at precision values in Table V. The best precision value was obtained by the SVM algorithm performed on meta-data generated by data complexity meta-features.

On the other hand, if processing time is more important, the recall values should be used. Although the induced SVMs will probably be less accurate, the time required for their induction will be smaller. The NB algorithm trained on STATLOG measures obtained the highest recall value in our experiments, but presented a low precision value, due to the high number of FPs in predictions.

In table V, we can also observe some low recall values. This might be due the class imbalance, making it difficult to predict the minority class. Correcting the class balancing may be not enough to solve the problem. An alternative approach would be to define more meta-features to extract more information and help in the prediction.

A more general picture of the meta-learner's predictive performance is provided by the F-Score measure, which is a balance between precision and recall measures. According to these values, ENS using STATLOG meta-features was the best overall. The 3-NN with data complexity meta-features also obtained a high F-Score value.

B. Hits and Misses

Figure 2 depicts the hits and misses of all meta-models over all meta-examples. The y-axis represents the meta-models: the algorithm and the set of meta-features used in executions. STAT denotes a meta-model built with STATLOG meta-features, and COMP with data complexity ones. The x-axis represents all the 143 meta-examples of the meta-database. In the figure, a hit is represented by a light gray square, and a miss by a black one.

Here, we can observe that only one dataset is misclassified for all meta-models: *'robot-failure-lp4'* (it has 116 examples, 90 attributes and 61% of elements in the majority class). Few datasets are classified correctly by all meta-models. The RF

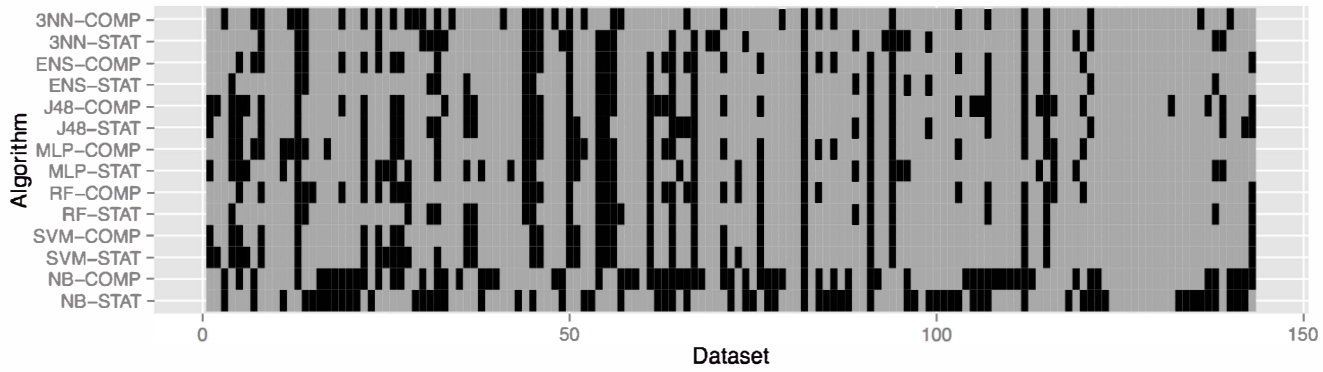

Fig. 2. Hits and erros of all meta-learners in 143 meta-examples.

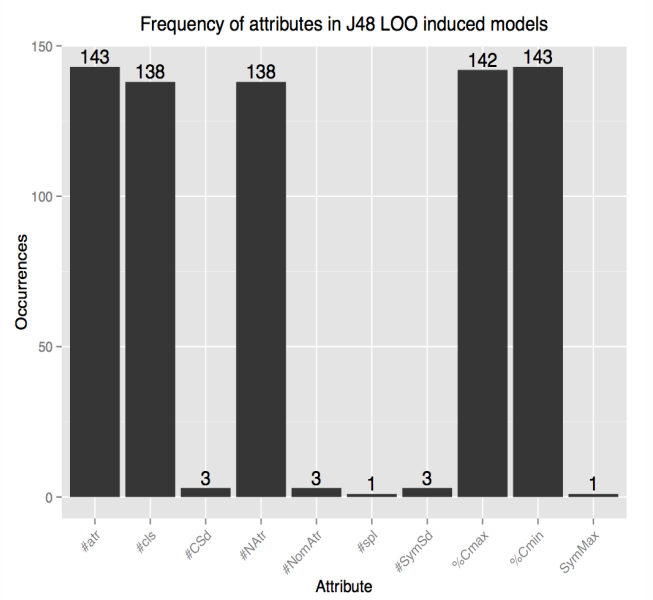

Fig. 3. The frequency of each attribute were selected by J48 models.

and SVM algorithms are correct on a lot of datasets, but classify several cases differently.

There are a few gains made by the ensemble (ENS) and its predictions present a pattern similar to the RF. The exception is the NB algorithm, which is a good complement of the other techniques. It hits almost all of DF meta-examples, but misses a lot of TUN examples. In general, there is no improvement obtained by using only data complexity meta-features.

C. Looking for patterns

The meta-database can also be used to analyze which dataset characteristics can help us decide whether the default hyper-parameter values are a good choice for the given dataset, or not. In Figure 4, we show a meta-decision tree, trained on the meta-data set, predicting when tuning is definitely recommended.

Figure 3 shows the frequency with which each attribute was selected by the induced decision trees. Only the attributes that were selected at least once are shown. Since the meta-data set has 143 meta-examples, each attribute can be selected up to 143 times using LOO-CV.

The meta-attributes selected with the highest frequency in the models induced with STATLOG meta-features were: *%CMax* (percentage of examples in the majority class), *#NumAttr* (number of numerical predictive attributes), *#cls* (number of classes), *#attr* (number of predictive attributes), and *%CMin* (percentage of examples in the minority class).

The decision tree in Figure 4 was the most frequently induced during the meta-level learning. This decision tree was generated 139 times and has 15 nodes, 8 of them leaf nodes. The predictive attribute most frequently selected as root node was *%CMax*, and obtained an accuracy value of 0.881.

The tree shows that if a data set is highly imbalanced, with more than 79% of the examples in the majority class, the meta-learner usually recommends the use of default hyper-parameter values. Perhaps, the tuning does not work well for imbalanced datasets, so performing the tuning process without a balancing method will not adequately improve the SVM accuracy.

The second-most important meta-feature is the number of predictive attributes. When there is a large number of predictive attributes, tuning becomes important. For large numbers of predictive attributes, SVMs tend to overfit. In order to deal with this problem, the gamma parameter needs to be wider (set to a lower value). This issue was previously reported and explained in [41], [42]. Indeed, recommending the tuning when a dataset has many attributes (#attr $>$ 13) might make sense if we consider the SVM data normalization process (carried out internally by LIBSVM). Although LIBSVM takes it into account by using $\gamma = 1/p$ as the default value, tuning is still recommended. Even so, this rule should not be taken as a golden rule, since this may be due to learning artifacts from the involved datasets.

The tree also recommends tuning when the proportion of numerical attributes is low (#NumAttr $\leq$ 11). In general, all categorical/factor attributes are converted into a binary encoding. In such a mixed/discrete space, 'non default' kernel hyper-parameter values might make more sense here then in the purely numerical case, as distances and geometrical properties will work differently.

We also looked at the importance of the attributes in the

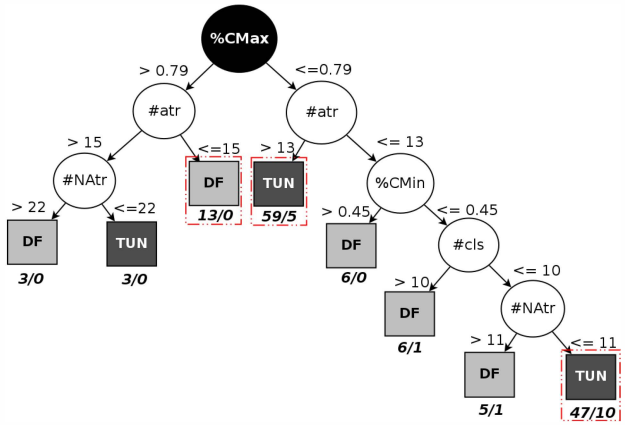

Fig. 4. The most frequent decision tree obtained by the J48 algorithm.

RF models. We analyzed only the RF-STAT meta-models since their predictions were better than those obtained by RF-COMP ones. We got a sorted vector from models with values indicating the mean decrease in accuracy when removing a specific attribute. The most significant attribute was %CMax, followed by %CSd and #attr. This corroborates what we observed in the decision tree models. Even while being a simpler and less accurate, decision trees allow a insight of the behavior of the learning process on the meta-level.

D. Overall considerations

The SVM hyper-parameter (C) and the width of the Gaussian kernel (γ) have been shown to be interdependent. There is no truly global optimum for this optimization, but instead there is a *ridge* of optimal solutions: if C goes up, γ can be adjusted to reach the same performance again. So, we defined a conservative confidence interval rule when choosing the class for meta-examples. It followed some conservative and empirical steps. Decision trees were induced to explain which characteristics from a data set could justify when one hyper-parameter setting approach should be favoured.

In [32] authors have found that the NN1 meta-feature (performance of a 1-NN algorithms on that data set) was the root node of the tree in meta-level. In our study we found a different meta-feature in the root node: %CMax, the percentage of examples in the majority class. Our tree shows that if data is highly imbalanced, the meta-learner tends to recommend the use of default hyper-parameter values. When analyzing RF meta-models, we observed that %Cmax and #attr were of great importance in the accuracy of these meta-models. Other experiments should be conducted to verify if the addition of the meta-feature NN1 on the set of meta-features evaluated in this paper will lead to a higher predictive performance of the meta-models.

V. Conclusions

This paper investigated the use of MTL to induce a classification model able to predict with a high predictive accuracy when optimization techniques should be used for tuning the hyper-parameters of SVMs instead of using default values suggested by both a well-known SVM library and a ML tool.

Experiments with MTHs using 143 data sets from the UCI repository were carried out to evaluate the effect of hyper-parameter tuning. From these experiments, a meta-data set was created for the induction of meta-models able to predict with high predictive performance when hyper-parameter tuning should be used instead of default values. Different ML algorithms, mainly RF and ENS, presented good prediction rates, better than the use of a single classifier.

Observing the most frequent decision tree, we claim that a small number of simple meta-features was sufficient to characterize the data sets. According to this decision tree, for highly imbalanced data sets, tuning does not obtain much higher performance. Probably, the tuning process might not work well for imbalanced datasets, so performing the tuning without a balancing method can not improve SVM accuracy.

As future work, we intend to investigate different approaches to extract meta-features and expand the number of data sets. We also plan to explore other methodologies on the meta level, including data balancing and meta-feature selection process to find the most relevant meta-features. Moreover, we should include a significance test to define the meta-target, and include experiments with other classification algorithms instead of SVMs, such as decision trees and *Deep Learning* algorithms, which have a larger number of sensitive hyper-parameters. Finally, we aim to make all our experiments available on OpenML [43], [44] for reproducibility and reusability.

Acknowledgment

The authors would like to thank CAPES, CNPq and FAPESP (Brazilian Agencies) for the financial support.

References

[1] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," *Machine Learning*, vol. 46, no. 1-3, pp. 131–159, Mar. 2002.

[2] S.-W. Lin, K.-C. Ying, S.-C. Chen, and Z.-J. Lee, "Particle swarm optimization for parameter determination and feature selection of support vector machines," *Expert Systems with Applications*, vol. 35, no. 4, pp. 1817 – 1824, 2008.

[3] I. Braga, L. P. do Carmo, C. C. Benatti, and M. C. Monard, "A note on parameter selection for support vector machines," in *Advances in Soft Computing and Its Applications*, ser. Lecture Notes in Computer Science, F. Castro, A. Gelbukh, and M. González, Eds. Springer Berlin Heidelberg, 2013, vol. 8266, pp. 233–244.

[4] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta, *Metalearning: Applications to Data Mining*, 2nd ed. Springer Verlag, 2009.

[5] S. Ali and K. A. Smith-Miles, "A meta-learning approach to automatic kernel selection for support vector machines," *Neurocomputing*, vol. 70, no. 13, pp. 173–186, 2006.

[6] J. Kanda, A. C. P. L. F. Carvalho, E. Hruschka, and C. Soares, "Selection of algorithms to solve traveling salesman problems using meta-learning," *Int. J. Hybrid Intell. Syst.*, vol. 8, no. 3, Aug. 2011.

[7] M. Reif, F. Shafait, and A. Dengel, "Prediction of classifier training time including parameter optimization," in *Proceedings of the 34th Annual German conference on Advances in artificial intelligence*, ser. KI'11. Springer-Verlag, 2011, pp. 260–271.

[8] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, 1989.

[9] J. Kennedy, "Particle swarms: optimization based on sociocognition," in *Recent Development in Biologically Inspired Computing*, L. Castro and F. V. Zuben, Eds. Idea Group, 2005, pp. 235–269.

[10] M. Hauschild and M. Pelikan, "An introduction and survey of estimation of distribution algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 3, pp. 111 – 128, 2011.

[11] H. H. Hoos, "Automated algorithm configuration and parameter tuning," in *Autonomous Search*, Y. Hamadi, E. Monfroy, and F. Saubion, Eds. Springer Berlin Heidelberg, 2012, pp. 37–71.

[12] J. Styles, H. H. Hoos, and M. Müller, "Automatically configuring algorithms for scalling performance," in *Proceedings of the 6th International Conference on Learning and Intelligent Optimization*, ser. LION'06, 2012, pp. 1–12.

[13] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009.

[14] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011.

[15] F. Hutter, H. H. Hoos, and T. Stützle, "Automatic algorithm configuration based on local search," in *Proceedings of the 22nd national conference on Artificial intelligence - Volume 2*, ser. AAAI'07. AAAI Press, 2007, pp. 1152–1157.

[16] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Mar. 2012.

[17] F. Friedrichs and C. Igel, "Evolutionary tuning of multiple svm parameters," *Neurocomput.*, vol. 64, pp. 107–117, 2005.

[18] T. Eitrich and B. Lang, "Efficient optimization of support vector machine learning parameters for unbalanced datasets," *Journal of Computational and Applied Mathematics*, vol. 196, no. 2, pp. 425–436, 2006.

[19] F. Hutter, H. Hoos, K. Leyton-Brown, and T. Stützle, "Paramils: an automatic algorithm configuration framework," *Journal of Artificial Intelligence Research*, no. 36, pp. 267–306, 2009.

[20] V. Nannen and A. E. Eiben, "Relevance estimation and value calibration of evolutionary algorithm parameters," in *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, ser. IJCAI'07. Morgan Kaufmann Publishers Inc., 2007, pp. 975–980.

[21] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms," in *Proc. of KDD-2013*, 2013, pp. 847–855.

[22] C. Soares, P. B. Brazdil, and P. Kuba, "A meta-learning method to select the kernel width in support vector regression," *Machine Learning*, vol. 54, no. 3, pp. 195–209, 2004.

[23] C. Soares and P. B. Brazdil, "Selecting parameters of svm using meta-learning and kernel matrix-based meta-features," in *Proceedings of the 2006 ACM symposium on Applied computing*, ser. SAC'06. ACM Press, 2006, pp. 564–568.

[24] R. Priya, B. F. De Souza, A. L. D. Rossi, and A. C. P. L. F. Carvalho, "Using genetic algorithms to improve prediction of execution times of ML tasks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7208 LNAI, no. PART 1, 2012, pp. 196–207.

[25] T. A. F. Gomes, R. B. C. Prudêncio, C. Soares, A. L. D. Rossi, and nd André C. P. L. F. Carvalho, "Combining meta-learning and search techniques to select parameters for support vector machines," *Neurocomput.*, vol. 75, no. 1, pp. 3–13, Jan. 2012.

[26] P. B. C. Miranda, R. B. C. Prudêncio, A. C. P. L. F. Carvalho, and C. Soares, "An experimental study of the combination of meta-learning with particle swarm algorithms for SVM parameter selection," *Lecture Notes in Computer Science*, vol. 7335 LNCS, no. PART 3, pp. 562–575, 2012.

[27] M. Reif, F. Shafait, and A. Dengel, "Meta-learning for evolutionary parameter optimization of classifiers," *Machine Learning*, vol. 87, pp. 357–380, 2012.

[28] M. Feurer, T. Springenberg, and F. Hutter, "Initializing bayesian hyperparameter optimization via meta-learning," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, Jan. 2015.

[29] R. Leite and P. Brazdil, "Active testing strategy to predict the best classification algorithm via sampling and metalearning," in *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2010, pp. 309–314.

[30] R. Leite, P. Brazdil, and J. Vanschoren, "Selecting classification algorithms with active testing," in *Proceedings of the 2012 Conference on Machine Learning and Data Mining (MLDM 2012)*, 2012, pp. 117–131.

[31] P. Miranda and R. Prudêncio, "Active testing for SVM parameter selection," in *Neural Networks (IJCNN), The 2013 International Joint Conference on*, Aug 2013, pp. 1–8.

[32] P. Ridd and C. Giraud-Carrier, "Using metalearning to predict when parameter optimization is likely to improve classification accuracy," in *Meta-learning and Algorithm Selection Workshop at ECAI 2014*, J. Vanschoren, P. Brazdil, C. Soares, and L. Kotthoff, Eds., August 2014, pp. 18–23.

[33] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, *A Practical Guide to Support Vector Classification*, Department of Computer Science - National Taiwan University, Taipei, Taiwan, 2007.

[34] A. L. D. Rossi and A. C. P. L. F. Carvalho, "Bio-inspired optimization techniques for svm parameter tuning," in *Proceedings of 10th Brazilian Symposium on Neural Networks*. IEEE Computer Society, 2008, pp. 435–440.

[35] P. B. Brazdil and R. J. Henery, "Analysis of results," in *Machine learning, neural and statistical classification*, D. Michie, D. J. Spiegelhalter, C. C. Taylor, and J. Campbell, Eds. Ellis Horwood, 1994, ch. 10, pp. 175–212.

[36] T. K. Ho and M. Basu, "Complexity measures of supervised classification problems," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 3, pp. 289–300, 2002.

[37] A. Orriols-Puig, N. Macia, and T. K. Ho, "Documentation for the data complexity library in c++," Barcelona, Spain, Tech. Rep., 2010.

[38] Y. Nojima, S. Nishikawa, and H. Ishibuchi, "A meta-fuzzy classifier for specifying appropriate fuzzy partitions by genetic fuzzy rule selection with data complexity measures," in *2011 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011)*. IEEE, Jun. 2011, pp. 264–271.

[39] L. P. F. Garcia, A. C. Lorena, and A. C. P. L. F. de Carvalho, "A study on class noise detection and elimination," in *SBRN*, 2012, pp. 13–18.

[40] L. P. F. Garcia, A. C. de Carvalho, and A. C. Lorena, "Noisy data set identification," in *Hybrid Artificial Intelligent Systems*, ser. Lecture Notes in Computer Science, J.-S. Pan, M. M. Polycarpou, M. Wo?niak, A. C. de Carvalho, H. Quintin, and E. Corchado, Eds. Springer Berlin Heidelberg, 2013, vol. 8073, pp. 629–638.

[41] J. Vanschoren, B. Pfahringer, and G. Holmes, "Learning from the past with experiment databases," in *Proceedings of PRICAI 2008*, ser. Lecture Notes in Computer Science, 2008, vol. 5351, pp. 485–496.

[42] J. Vanschoren, H. Blockeel, B. Pfahringer, and G. Holmes, "Experiment databases," *Machine Learning*, vol. 87, no. 2, pp. 127–158, 2012.

[43] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "OpenML: Networked science in machine learning," *SIGKDD Explorations*, vol. 15, no. 2, pp. 49–60, 2013.

[44] J. van Rijn, B. Bischl, L. Torgo, B. Gao, V. Umaashankar, S. Fischer, P. Winter, B. Wiswedel, M. Berthold, and J. Vanschoren, "OpenML: A collaborative science platform," in *Proceedings of ECMLPKDD 2013*, ser. Lecture Notes in Computer Science, 2013, vol. 8190, pp. 645–649.