Neurocomputing 75 (2012) 3–13

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Combining meta-learning and search techniques to select parameters for support vector machines

Taciana A.F. Gomes [a], Ricardo B.C. Prudêncio [a,*], Carlos Soares [b], André L.D. Rossi [c], André Carvalho [c]

[a] Centro de Informática, Universidade Federal de Pernambuco, Recife, Brazil
[b] LIAAD-INESC Porto LA, Faculdade de Economia, Universidade do Porto, Portugal
[c] Depto. de Ciências da Computação, Universidade de São Paulo, São Carlos, Brazil

ARTICLE INFO

Available online 31 July 2011

Keywords:
Support vector machines
Meta-learning
Search

ABSTRACT

Support Vector Machines (SVMs) have achieved very good performance on different learning problems. However, the success of SVMs depends on the adequate choice of the values of a number of parameters (e.g., the kernel and regularization parameters). In the current work, we propose the combination of meta-learning and search algorithms to deal with the problem of SVM parameter selection. In this combination, given a new problem to be solved, meta-learning is employed to recommend SVM parameter values based on parameter configurations that have been successfully adopted in previous similar problems. The parameter values returned by meta-learning are then used as initial search points by a search technique, which will further explore the parameter space. In this proposal, we envisioned that the initial solutions provided by meta-learning are located in good regions of the search space (i.e. they are closer to optimum solutions). Hence, the search algorithm would need to evaluate a lower number of candidate solutions when looking for an adequate solution. In this work, we investigate the combination of meta-learning with two search algorithms: Particle Swarm Optimization and Tabu Search. The implemented hybrid algorithms were used to select the values of two SVM parameters in the regression domain. These combinations were compared with the use of the search algorithms without meta-learning. The experimental results on a set of 40 regression problems showed that, on average, the proposed hybrid methods obtained lower error rates when compared to their components applied in isolation.

1. Introduction

An increasing attention has been given to Support Vector Machines (SVMs) due to both their theoretical foundations and the good empirical performance when compared to other learning algorithms in different applications [1]. However, the SVM performance strongly depends on the adequate choice of its parameter values, including for instance, the kernel function, the values of kernel parameters, the regularization parameter, among others [2]. An exhaustive trial-and-error procedure for selecting good parameter values is obviously not practical [3].

SVM parameter selection is commonly treated by different authors as an optimization problem in which a search technique is employed to find the configuration of parameters which maximizes the SVM performance estimated on the problem at hand [4]. Although it represents a more systematic approach to parameter selection, this approach can still be very expensive, since a large number of candidate parameter configurations are often evaluated during the search process [1].

An alternative approach to SVM parameter selection is the use of meta-learning, which treats the SVM parameter selection as a supervised learning task [1,5]. Each training example for meta-learning (i.e. each *meta-example*) stores the characteristics of a past problem and the performance obtained by a set of candidate configurations of parameters on the problem. By receiving a set of such meta-examples as input, a *meta-learner* is able to predict the best configuration of parameters for a new problem based on its characteristics. meta-learning is a less expensive solution compared to the search approach. In fact, once the knowledge is acquired by the meta-learner, configurations of parameters can be suggested for new problems without the need of empirically evaluating different candidate configurations (as performed using search techniques).

In the current work, we propose the combination of search techniques and meta-learning to the problem of SVM parameter selection. In this proposal, configurations of parameters suggested by meta-learning are adopted as initial solutions which will be

* Corresponding author.
E-mail addresses: tafg@cin.ufpe.br (T.A. Gomes), rbcp@cin.ufpe.br (R.B. Prudêncio), csoares@fep.up.pt (C. Soares), alrossi@icmc.usp.br (A.L. Rossi), andre@icmc.usp.br (A. Carvalho).

0925-2312/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2011.07.005

later refined by the search technique. In previous work the search process starts evaluating solutions randomly sampled from the parameter space (e.g., [6–8]). In the proposed hybrid approach, the search process starts with successful solutions from previous similar problems. Hence, we expect that meta-learning guides the search directly to promising regions of the search space, thus speeding up the convergence to good solutions.

In order to evaluate our proposal, we investigated the selection of two parameters for SVMs on regression problems (i.e. Support Vector Regressors): the parameter γ of the RBF kernel and the regularization constant C, which may have a strong influence in SVM performance [9]. In our work, a database of 40 meta-examples was produced from the evaluation of a set of 399 configurations of (γ, C) on 40 different regression problems. Each regression problem was described by a number of 17 meta-features proposed in [10,1,11]. Two search algorithms were used to optimize the parameters (γ, C): Particle Swarm Optimization (PSO) [12] and Tabu Search (TS) [13] These algorithms were used in two different scenarios: with the initial population suggested by meta-learning, leading to hybrid methods, and with a random initial population. According to experimental results, the hybrid methods were able to converge faster to good solutions when compared to the randomly initialized PSO and TS.

This paper is organized as follows. Section 2 brings a brief presentation on the SVM parameter selection task. Section 3 presents the proposed work, followed by Section 4 which presents the implementation details. Section 5 describes the experiments performed and the obtained results. Finally, Section 6 discusses the main conclusions and possibilities for future work.

2. SVM parameter selection

According to [14], the SVM parameter selection task is often performed by evaluating a range of different combinations of parameters and retaining the best one in terms of an *objective function*. There are two important issues in this sense: (1) the objective function to be optimized, which is in general a functional estimating the SVM performance using the problem's dataset (e.g., leave-one-out and cross-validation estimates, error bounds, model complexity, among others); (2) the strategy adopted to explore the space of parameters. Regarding the second issue, an exhaustive procedure to explore the parameter space can lead to good results however it is a strategy that should be avoided due to practical reasons.

In order to improve the search process and to avoid an exhaustive or a random exploration of parameters, different authors have deployed search and optimization techniques [3,4,6–8,14–16]. In this context, the search space consists of a set of possible configurations of parameters. Each search technique deploys specific search operators and mechanisms to explore the search space, aiming to reach optimized parameters with good values for the chosen objective function. Among the search techniques adopted in the literature, we can mention gradient-based techniques [3], Evolutionary Algorithms [6–8], Tabu Search [14] and Particle Swarm Optimization [16].

Although the use of search techniques is more efficient when compared to an exhaustive process of parameter selection, this solution may still be very expensive since for each configuration generated during the search it is necessary to train the SVM [1]. This limitation can be even more drastic depending on the problem at hand and the number of parameters to be optimized. Another limitation of this solution is that the search process usually starts with random configurations uniformly sampled from the search space. This can result on slow convergence and sensibility to local minima depending on the adopted search technique.

Alternatively, *meta-learning* has been proposed and investigated in recent years to SVM parameter selection [1,5,10,11,17,18]. In this approach, the choice of parameters for a problem is based on well-succeeded parameters adopted to previous similar problems. For this, it is necessary to maintain a set of *meta-examples* where each meta-example stores: (1) a set of characteristics (called *meta-features*) describing a learning problem; and (2) the best configuration of parameters (among a set of candidates) empirically evaluated on the problem. A *meta-learner* is then used to acquire knowledge from a set of such meta-examples in order to recommend (predict) adequate configurations of parameters for new problems based on their meta-features. Meta-learning is able to predict not only one configuration of parameters but also to recommend *rankings* of configurations (as performed in [1]). This is interesting since the user has more alternatives if the first configuration recommended by meta-learning does not achieve adequate results.

Compared to the search approach, meta-learning tends to be more efficient in terms of computational cost. In fact, in the search approach it may be necessary to evaluate a large number of configurations until good objective values are achieved. In meta-learning in turn, the quality of a SVM configuration on a specific problem is *predicted* instead of directly estimated (as it occurs in the search approach). Meta-learning however is very much dependent on the quality of its meta-examples. In the literature, it is usually difficult obtaining good results since meta-features are in general very noisy and the number of problems available for meta-example generation is commonly limited. Hence, the performance of meta-learning for SVM parameter selection may be not so good as the performance of search techniques. The previous statement however has to be taken with caution since no previous authors have performed experiments comparing meta-learning to search techniques.

In this section we present the advantages and limitations of the search and the meta-learning approaches to SVM parameter selection. In our work, we combine the two approaches in such a way that meta-learning is used to recommend parameters which will be later refined by a search technique.

3. Proposed solution

As discussed in the previous section, although SVMs have a strong generalization capability, their performance depends on an adequate choice of their parameter values. This work proposes a new hybrid method to automate the design of SVMs based on the combination of meta-learning and search algorithms. In the proposal, a meta-learner suggests the initial search points as successful parameter values from previous similar problems.

As discussed in [19], good solutions to a particular search problem can be used to indicate promising regions of the search space for similar problems. We highlight that the target output of meta-learning is essentially the same objective function optimized by the search techniques (i.e. a given measure of learning performance). The difference is that in meta-learning the objective function is *predicted* using the meta-features instead of directly estimated. The value of meta-features to predict learning performance has already been demonstrated in different previous work on meta-learning (see [20]). Meta-learning has been also applied to improve optimization tasks but in very different contexts (e.g., job shop scheduling [19], quadratic programming problems [21], traveling salesman problems [22]). The positive results in these contexts motivated us to apply similar ideas for optimizing SVM parameters. We expect that the initialization

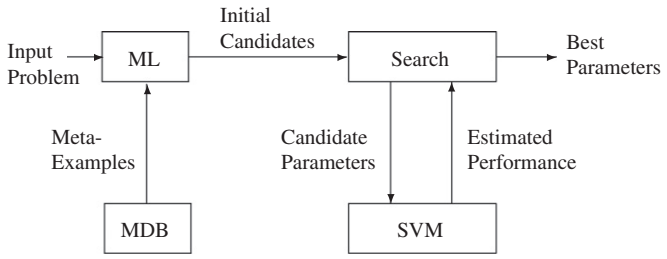

Fig. 1. General architecture.

provided by meta-learning enables the search algorithm to speed up its convergence to good solutions.

Fig. 1 depicts the general architecture of the proposed solution. Initially, the meta-learner (ML) module retrieves a predefined number of past meta-examples stored in a Meta-Database (MDB), selected on the basis of their similarity to the input problem. Following, the Search module adopts as initial search points the configurations of successful parameter values on the retrieved meta-examples. In the Search module, a search process iteratively generates new candidate parameters to be evaluated in the SVM. The output configuration of parameters will be the best one generated by the Search module up to its convergence or other stopping criteria.

In [23], we presented the preliminary experiments which evaluated the proposed hybrid method (using PSO in the Search module). In the current work, we provide a more detailed description of the proposed method, by formally presenting each component of the proposed architecture (see next sub-sections). We also implemented a new technique in the Search module: the Tabu Search algorithm which presents different characteristics compared to the PSO. As it will be seen, different conclusions can be yielded by considering different search techniques in our solution. Finally, new experiments were performed in the current paper. In [23], the hybrid solution was only compared to the search technique in isolation. In the current work, we compared the hybrid method to both the meta-learning and the search technique. For this, we adapted the TopN procedure originally adopted in [1] for evaluating the SVM parameters recommended by meta-learning. We also performed statistical tests for performance evaluation and provided more detailed reports of our results that helped us to assess the advantages and limitations of each evaluated method.

3.1. Search module

The SVM parameter selection problem can be defined as an optimization problem considering a *search space* of parameters $\mathcal{S}$ and an objective function $\mathcal{O}: \mathcal{S} \rightarrow \Re$, which evaluates the configurations in $\mathcal{S}$. The optimization task is to find a set of good configurations $S^{opt} \in \mathcal{S}$ in terms of the objective function $\mathcal{O}$.

In an iterative optimization technique, the search process starts with an initial set $S^{ini} \subset \mathcal{S}$ of search points and progressively *moves* toward betters solutions. The set S^{ini} can be just a single configuration or a population of search points (in the case of population-based techniques). Given a learning problem d, the optimized configurations of parameters S^{opt} found during the search can be seen as a function of the initial search points:

$$S^{opt} \leftarrow Search(d, S^{ini}) \tag{1}$$

In a conventional search approach, S^{ini} is randomly and uniformly sampled from $\mathcal{S}$. Ideally, the search technique would be less dependent as possible from the initial search points. In our work, in order to minimize this dependency, the set S^{ini} is recommended by a meta-learning technique.

An important aspect in the Search module is the technique adopted to perform the exploration of the search space. Global search techniques, Genetic Algorithms (GA) and PSO, are more complex and would be potentially better to perform the parameter optimization. However, as the search in our hybrid method starts in a promising region of the search space, simple local techniques could also be adopted. In fact, the viability of using a local technique in our hybrid search method was indicated in our experiments (see Section 5).

3.2. Meta-database

The meta-database is derived from a set of learning problems $\mathcal{D}$, which can be collected from dataset repositories or artificially generated datasets. Formally, let $\mathcal{M}$ be the set of meta-examples derived from $\mathcal{D}$. Each meta-example $e_i \in \mathcal{M}$ is related to a learning problem $d_i \in \mathcal{D}$ and stores: (1) a vector of p meta-features $\mathbf{x}_i = (x_i^1, \ldots, x_i^p)$ describing the problem d_i; (2) the best configuration of parameters $s_i^* \in \mathcal{S}$ evaluated on the problem. Two important issues are considered for generating the meta-examples: (1) the meta-features adopted to describe the learning problems; (2) the definition of the best configuration for a problem.

Considering the first issue, a large amount of work has been developed to define suitable meta-features, including general, statistical and information-theoretic measures, landmarkers and model-based measures (see [24] for a review). According to [25], one can follow some advices to choose the meta-features. First, meta-features have to be feasible for the class of problem being solved (for instance, classification or regression). Second, meta-features should be reliably identified, avoiding subjective analysis, such as visual inspection of plots. In fact, subjective feature extraction is expensive and requires a lot of expertize. Finally, one should use a manageable number of simple meta-features in order to avoid a time consuming feature extraction process.

Finally, the configuration s_i^* stored in a meta-example is defined from an empirical evaluation of a set of candidates $S^C \subset \mathcal{S}$ on the associated problem (i.e. the best configuration in S^C considering the objective function $\mathcal{O}$). The quality of the stored configurations closely depends on the number of candidates available in S^C. By considering a very small set of candidates, the quality of meta-learning itself may be harmed. In turn, by considering a very large set of candidates, the cost of system development increases.

3.3. Meta-learner

Given a meta-database, a meta-learner $\mathcal{L}$ is used to predict good configurations of parameters based on the description of the problem at hand. Formally, let $\mathbf{x}$ be the description of a new problem d and let $\mathcal{M}$ be the set of meta-examples. The meta-learner returns as output a set S^{rec} of recommended configurations:

$$S^{rec} \leftarrow \mathcal{L}(\mathbf{x}, \mathcal{M}) \tag{2}$$

In our work, a simple meta-learner was adopted. Initially, given the description $\mathbf{x}$, the meta-learner retrieves the most similar meta-examples from $\mathcal{M}$. Similarity of meta-examples is defined in terms of the values of meta-features. Afterwards, the best configurations stored in the retrieved meta-examples are inserted in the recommended set S^{ini}.

The set of configurations recommended by meta-learning is adopted as initial search points for the Search module. Hence, the hybrid method can be formally defined as:

$$Hybrid(d, \mathbf{x}, \mathcal{M}) = Search(d, \mathcal{L}(\mathbf{x}, \mathcal{M})) \tag{3}$$

In the next section, we present an implemented prototype which followed the hybrid solution described here. Experiments

evaluating the implemented prototype will be presented in Section 5.

4. Implementation

In the current work, we implemented a prototype to select SVM parameters for regression problems. In this case, we actually deployed Support Vector Regressors (SVRs). Although our implementation and case study has been focused on regression, we highlight that the proposed solution can be also adopted for classification problems. Case studies evaluating the proposed solution for SVMs in classification problems will be developed in future work.

In our work, we adopted the LibSVM library to implement the SVMs for regression problems [26].[1] Two specific parameters were considered to select: the γ parameter of RBF kernel and the regularization parameter C. The choice of RBF kernel is due to its flexibility in different problems compared to other kernels [9,28]. It is known that the γ parameter has an important influence in learning performance since it controls the linearity of the induced SVM. The parameter C is also important for learning performance since it controls the complexity (flatness) of the regression function derived by the SVMs [28]. The other SVR parameters which were not the focus of the current work were defined as the default values suggested by the LibSVM tool.

In the implemented prototype, a meta-database was generated from the application of LibSVM to 40 different regression problems collected from data repositories. An instance-based learning method was employed in our work to provide the initial parameter configurations. In the Search module, we evaluated two search algorithms with different characteristics: Particle Swarm Optimization (PSO) [12] and Tabu Search (TS) [13]. The former algorithm, based on population-based search, deploys adaptive mechanisms to combine both global and local exploration of the search space. The second, TS, is a simpler search technique that locally explores the search space. Information regarding their implementation will be presented in the next sub-sections.

4.1. Search module—PSO

In our prototype, we implemented the version of PSO originally proposed in [29] and adapted here to perform the search for configurations (γ, C). The *objective* function evaluated the quality of each configuration of parameters on a given regression problem. In our work, given a SVM configuration, we defined the objective function as the *Normalized Mean Squared Error* (NMSE) obtained by the SVM in a 10-fold cross - validation experiment. So, the objective of PSO was to find the configuration (γ, C) with lowest NMSE value for a given regression problem.

In our PSO implementation, each particle j represents a configuration $s_j = (\gamma, C)$, indicating the *position* of the particle in the search space. Each particle also has a *velocity* which indicates the current search direction performed by the particle. PSO basically works by updating the position and velocity of each particle in order to progressively explore the best regions in the search space. The update of position and velocity in the basic PSO is given by the following equations:

$$v_j \leftarrow \omega v_j + c_1 r_1(\hat{s}_j - s_j) + c_2 r_2(\hat{g} - s_j) \quad (4)$$

$$s_j \leftarrow s_j + v_j \quad (5)$$

In Eq. (4), $\hat{s}_j$ is the best position achieved by the particle so far, and $\hat{g}_j$ is the best position achieved by any particle in the population so far. Hence, each particle is progressively moved in direction of the best *global* positions achieved by the population (the *social* component of the search) and the best *local* positions obtained by the particle (the *cognitive* component of the search).

The parameters ω, c_1 and c_2 control the trade-off between exploring good global regions in the search space and refining the search in local regions around the particle. In Eq. (4), r_1 and r_2 are random numbers (uniformly sampled from the interval [0,1]) used to enhance the diversity of particle positions. In our prototype, we adopted decreasing values for the inertia parameter ω (decreasing from 0.9 to 0.4 during the PSO generations). Hence, the PSO performs a more global exploration in its initial generations and a fine-tuned local exploration in the last generations. Regarding the parameters c_1 and c_2, we adopted fixed values: $c_1 = c_2 = 2$, commonly adopted in the literature.

In our work, the PSO was implemented to perform a search in a space represented by a discrete *grid* of SVM configurations, consisting of 399 different settings of parameters γ and C. By following the guidelines provided in [28], we considered the following exponentially growing sequences of γ and C as potentially good configurations: the parameter γ assumed 19 different values (from 2^{-15} to 2^{3}) and the parameter C assumed 21 different values (from 2^{-5} to 2^{15}), thus yielding $19 \times 21 = 399$ different combinations of parameter values in the search space. We highlight that in order to facilitate the search process the particles were defined in the log space of these configurations.

4.2. Search module—TS

In the previous section, we described our implementation of PSO, which represents as a powerful population-based search algorithm. PSO is able to balance global and local exploration during the search, thus leading to faster and better results in different applications. In our work, we also evaluated the TS algorithm, which deploys a simpler search mechanism performing a local exploration of the search space. Our motivation here is to evaluate to which extent the hybrid solution proposed in our work is dependent on a potentially powerful search technique. We envisioned that a local search mechanism would be good enough to refine the solutions provided by meta-learning.

As previously stated, each position j in the search space represents a configuration $s_j = (\gamma, C)$ of SVM parameters. At each iteration, the TS generates a set of *neighbors* from the current position of the search. Following, the current position is updated to the best solution observed in the generated neighborhood. In order to prevent repeating movements and cycles in the search space, each previously visited position is recorded on a *tabu list*.

An important aspect of the TS algorithm is how to generate neighbors from a position. In our work, we added and subtract a unity on the parameter values stored in the current position. Hence, for each position a number of four neighbors can be generated. This operator is adopted just to produce small variations on the current solution of TS.

4.3. Meta-database

In order to generate meta-examples, we collected 40 datasets corresponding to 40 different regression problems, available for downloading in the WEKA project website.[2] The list of regression problems adopted in our work for meta-example generation is

[1] In our work, the ε-Support Vector Regression [27] formulation was adopted. More details on this implementation can be found in http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[2] These datasets are specifically the sets provided in the files *numeric* and *regression* available in http://www.cs.waikato.ac.nz/ml/weka/

Table 1
Regression problems adopted for meta-example generation.

Auto-price	Cloud	Hungarian	Pyrim
Auto93	Cpu	Lowbwt	Quake
AutoHorse	Detroid	Lowglew	Schlvote
AutoMpg	Diabetes-numeric	Machinecpu	Sensory
Baskball	EchoMonths	Mbagrade	Sleep
Bodyfat	Elusage	Meta	Stock
Bolts	Fishcatch	Pbc	Triazines
BreastTumor	Fruitfly	Pharynx	Veteran
Cholesterol	Gascons	Pollution	Vineyard
Cleveland	Housing	PwLinear	Wisconsin

presented in Table 1. These problems correspond to benchmarking datasets associated to different domains of application. This diversity of domains is positive in our context since the characteristics of the datasets tend to vary a lot as well. For instance, the adopted datasets have 253.7 examples on average with a high standard deviation (363.8). The minimum and maximum values of number of examples are 13 and 2178 respectively. Concerning the number of attributes, the datasets have 9.5 attributes on average with 6.87 of standard deviation. The minimum and maximum values in this case are 2 and 38 attributes. This variation is convenient to meta-learning, since it is expected that a learning algorithm present significantly different patterns of performance, depending on the problem being solved.

As previously mentioned, each meta-example is related to a single regression problem and stores: (1) the values of its meta-features; and (2) the best configuration of parameters evaluated for the problem.

4.3.1. Meta-features.

In this work, a total number of 17 meta-features was used to describe the datasets of regression problems. From these 17 meta-features, 14 were based on the set of features defined in [10], corresponding to descriptive measures of the regression datasets (see Table 2). Here we provide explanations for some meta-features, which are not straightforward to understand. Concerning for instance the meta-feature *Proportion of continuous attributes with outliers*, in a regression problem with a attributes this meta-feature is computed as:

$$\frac{\sum_{i=1}^{a} I(H_{outl}(a_i) < 0.7)}{a}$$

where I is the indicator function,[3] a_i is the i-th (continuous) independent attribute and $H_{outl}(a_i)$ is the ratio between the standard deviation of the mean and the standard deviation of the α-trimmed mean of a_i (with α set to 0.05).

The meta-feature *Sparsity of the target* brings a categorical value derived from the coefficient of variation ($\sigma_T/\overline{T}$) of the target attribute T of a regression problem:

$$\begin{cases} 0 & (\text{“not sparse”}) \quad \sigma_T/\overline{T} < 0.2 \\ 1 & (\text{“sparse”}) \quad 0.2 \le \sigma_T/\overline{T} \le 0.5 \\ 2 & (\text{“extremely sparse”}) \quad \sigma_T/\overline{T} > 0.5 \end{cases}$$

The meta-feature *Presence of outliers in the target* is computed following the rule:

$$\begin{cases} 1 & (\text{“yes”}) \quad H_{outl}(T) < 0.07 \\ 0 & (\text{“no”}) \quad H_{outl}(T) \ge 0.07 \end{cases}$$

[3] The indicator function $I(C)$ returns 1 if the condition C is true and 0, otherwise.

Table 2
Meta-features proposed in [10] for regression problems.

Number of examples
Number of attributes
Proportion of symbolic attributes
Ratio of the number of examples to the number of attributes
Proportion of the attributes with outliers
Coefficient of variation of the target (ratio of the standard deviation to the mean)
Sparsity of the target (coefficient of variation discretized into three values)
Presence of outliers in the target
Stationarity of the target (the standard deviation is larger than the mean)
R2 coefficient of linear regression (without symbolic attributes)
R2 coefficient of linear regression (with binarized symbolic attributes)
Average absolute correlation between numeric predictor attributes
Average absolute correlation between numeric predictor attributes and the target attribute
Average dispersion gain

Table 3
Meta-features proposed in [1] for regression problems.

Mean of off-diagonal values
Variance of the off-diagonal values
Kernel-target alignment

The meta-feature *Stationarity of the target* is defined as:

$$\begin{cases} 1 & (\text{“yes”}) \quad \sigma_T > \overline{T} \\ 0 & (\text{“no”}) \quad \sigma_T \le \overline{T} \end{cases}$$

Finally, *Average dispersion gain* is defined as the mean error of the best decision stump for each predictor attribute. The other meta-features are computed based on well-known statistics and are not explained in detail here.

The remaining three meta-features were defined in [1], corresponding to features computed from the kernel matrix (see Table 3). These meta-features evaluate the quality of the kernel matrix derived from the examples of a regression problem. Both sets of meta-features have shown to be useful for model selection purposes (see [10,1]). In our work, we combined these sets in order to provide a more complete description of the regression problems.

4.3.2. Best configuration

The best configuration stored on a meta-example is defined from the empirical evaluation of the same 399 configuration settings defined in the Search module (see Section 4.1). For each of the 399 configurations, a 10-fold cross-validation experiment was performed to evaluate the SVM performance. The 399 NMSE values obtained were compared and the lowest value was retained. The configuration with lowest NMSE was then stored in the meta-example. As previously observed, in these experiments, we deployed the LibSVM library to implement the SVMs and to perform the cross-validation procedure.

We highlight here that the candidate set of parameters adopted for meta-example generation is equivalent to the parameter space explored by the Search module. Hence, we could evaluate which configurations of parameters were the best ones in the regression problems (i.e. the best points in a search space) and use this information to guide the search process for new similar problems.

4.4. Meta-learner

Given a new input problem described by the vector $\mathbf{x} = (x^1, \ldots, x^p)$, the meta-learner retrieves the k most similar meta-examples from the database, according to the lowest

distance between the meta-features. The distance function (*dist*) implemented in the prototype was the unweighted L_1-Norm, defined as:

$$dist(\mathbf{x},\mathbf{x}_i) = \sum_{j=1}^{p} \frac{|x^j - x_i^j|}{\max_i(x_i^j) - \min_i(x_i^j)} \quad (6)$$

We adopted the L_1-Norm following the original proposal of meta-learning for SVM parameter selection (see [1]). For each retrieved meta-example, the meta-learner collects the best configuration stored in the meta-example. Then, the meta-learner suggests as initial configurations for searching the set of k best configurations observed on the retrieved meta-examples.

Considering the PSO method, the k recommended configurations are directly adopted as the initial population of particles. Considering the TS, in turn, the k configurations are initially evaluated and the best one is adopted as the first position of the TS. In both cases, the derived hybrid methods (named here as *Hybrid-PSO* and *Hybrid-TS*) starts with a number of k search points (i.e. those ones suggested by meta-learning). From the $(k+1)$-th point, the space is explored only using a search method (either PSO or TS).

There are other alternatives that can be adopted to recommend the SVM configurations once the most similar meta-examples are retrieved. For instance, the meta-learner could return the configurations that obtained the highest average or alternatively the highest median performance on the retrieved meta-examples. Also, the configurations could be ranked for each retrieved meta-example and the meta-learner could return the configurations with best average rank. In our implementation, we choose to return the best configuration of each retrieved meta-example since this approach is simple and fast however other alternatives can be evaluated in future work.

5. Experiments

In this section, we present the experiments performed to evaluate the proposed hybrid solution for SVM parameter selection. Initially, we present the experiments which evaluated the Hybrid-PSO method (Section 5.1). Following, we present the experiments performed to evaluate the Hybrid-TS method, and also discuss the relative performance of this method compared to the Hybrid-PSO (Section 5.2).

Besides the empirical evaluation of the hybrid solution, we also performed an empirical evaluation comparing individually the meta-learning and the search approaches to SVM parameter selection. To the best of our knowledge, no previous authors have compared these two approaches in order to have a better assessment of their relative costs and benefits. Hence, this comparison can be pointed out as a specific contribution of our work.

5.1. Hybrid-PSO method

The proposed hybrid solution was evaluated on the set of 40 regression problems by following a leave-one-out methodology. At each step of leave-one-out, one meta-example was left out to evaluate the implemented prototype and the remaining 39 meta-examples were considered in the MDB to be selected by the ML module. Initially, a number of k configurations were suggested by the ML module as the initial PSO population (see Section 4.4) (in our experiments, we adopted $k=5$). The PSO then optimized the SVM configurations for the problem left out up to the number of 10 generations. In each generation, we recorded the lowest NMSE value obtained so far (i.e. the best fitness). Hence, for each problem left out a curve of N values of NMSE was generated aiming to analyze the search progress on the problem. Finally, the curves of NMSE values were averaged over the 40 steps of the leave-one-out experiment in order to evaluate the quality of the PSO search on optimizing SVM parameters for the 40 regression problems considered.

As a basis of comparison, the same above experiment was adopted to evaluate a randomly initialized population for PSO (also using $k=5$ as the population size). As said, the randomly initialization corresponds to the conventional strategy adopted in the literature. We also evaluated a purely random procedure for parameter selection. Despite its simplicity, the random search has the advantage of performing a uniform initial exploration of the search space and has been commonly used to verify the utility of more complex search techniques. Finally, we highlight that each search procedure (the Hybrid-PSO search, the randomly initialized PSO and the purely random search) was executed 1000 times and the average results were recorded.

Fig. 2 shows the NMSE curves (over 1000 runs) obtained by the three evaluated search techniques. As expected, the random initialized PSO had a similar performance compared to random search in the initial iterations (i.e. generations), and later it progressively increased its relative performance during the search. By performing a statistical test (the Wilcoxon signed-rank test [31] at a 95% level of confidence), we verified that the PSO was equivalent to the random search in the first and second generations and statistically better after the third generation. This result was also indicated by observing Fig. 3(a) which presents the performance gain of PSO over the random search in percentage terms as well as the 95% confidence intervals (as error bars) of the differences. A slight advantage (different from zero) is observed using PSO starting from the third generation. These results indicate the viability of the PSO search technique. However, the convergence of the PSO to good solutions is slower than the convergence observed for the Hybrid-PSO. In fact, after the third generation the Hybrid-PSO yields better results than PSO after ten generations (see Fig. 2). The Hybrid-PSO converges faster to good solutions since it starts its search on more promising solutions in the search space. The Search module in our hybrid solution just refined the initial solutions provided by meta-learning, which were in fact closer to the best solutions found. The Wilcoxon test revealed that the Hybrid-PSO is statistically better than PSO and the random search in all search generations. This result was also indicated by considering the confidence intervals for the performance differences presented in Fig. 3(b) and (c). The lower bounds

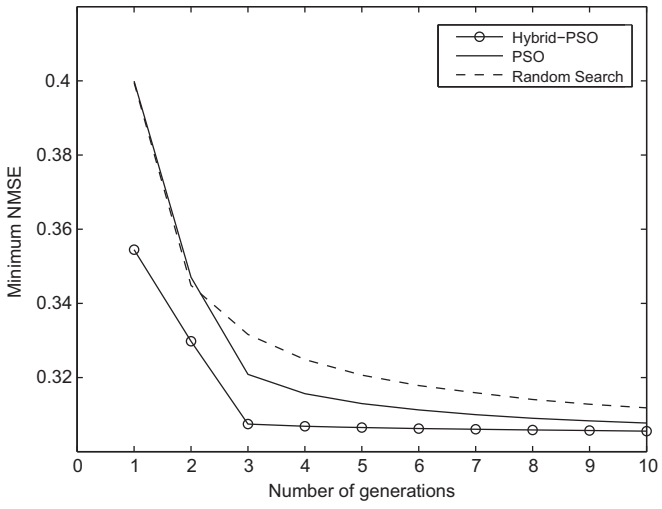

Fig. 2. Lowest NMSE results obtained at each generation.

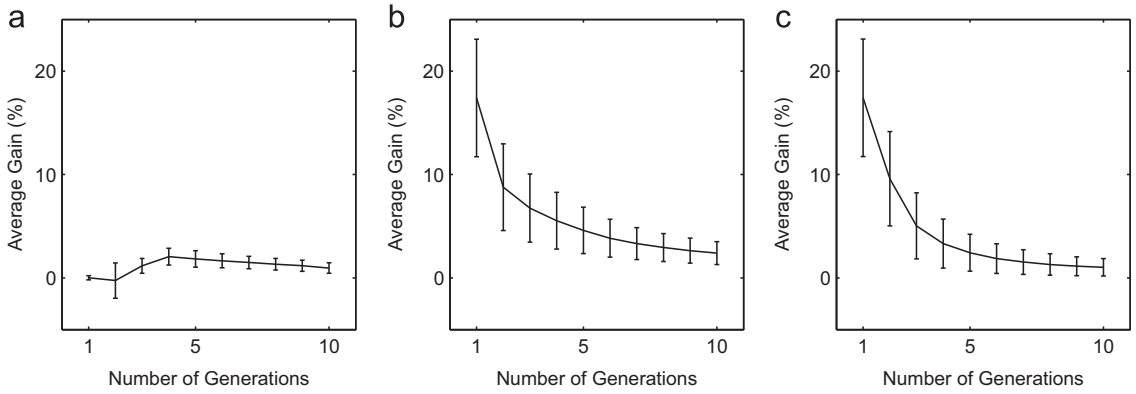

Fig. 3. Average gain in percentage terms at each generation for each pair of evaluated search methods: Random search, PSO and Hybrid-PSO.

of the intervals are greater than zero specially in the initial generations of the Hybrid-PSO.

In the previous experiment, we evaluated the relevance of using meta-learning to speed up the convergence of PSO. In our work, we also evaluated the performance of the Hybrid-PSO compared to its components in isolation: the meta-learning and the search procedure. For this, we adopted a TopN evaluation procedure as described in [1]. The TopN curve is just a plot of the minimum NMSE achieved vs. the number of different SVM configurations recommended by a method. By analyzing a TopN curve, we can balance the trade-off between benefits and costs of a method for parameter selection. In a TopN curve, we can evaluate how many alternatives configurations can be tested on the SVM in order to achieve a given level of performance. In our experiments, a TopN curve is generated for each method on each regression problem (up to 20 different recommended configurations). Then, the average TopN curve across the 40 problems is presented.

Fig. 4 presents the average TopN curves for the three approaches: (1) PSO (with random initialization); (2) meta-learning; and (3) the Hybrid-PSO. We also present in Fig. 4 the average NMSE achieved by the default heuristic adopted by the LibSVM tool (γ = inverse of the number of attributes and C=1). Although simple, this heuristic may be adequate for naive users and it is used here as baseline for benchmarking comparison. Finally, Fig. 4 shows the average NMSE that would be achieved if the best parameter configuration had been chosen on all problems.

By comparing PSO and meta-learning in isolation, we can identify a trade-off in their relative performances. Meta-learning is better than PSO when one considers a small number of recommended parameter configurations. It is also better than the default LibSVM parameters. Hence, meta-learning alone would be indicated in situations in which the SVM user had strong resources constraints. In these situations, the meta-learning could recommend a lower number of configurations with intermediate performance levels. We highlight that there is an additional cost in recommending the configurations by the hybrid approach which is the cost of the meta-learning process (specially the cost of computing the meta-features). However, by following the guidelines of previous work in meta-learning, we deployed non-expensive meta-features, in such a way that the total cost of computing them is in general lower than the cost of directly training a single SVM. The gain in performance using the hybrid approach in this case compensates this additional cost. The PSO in turn is able to find better configurations along its search and then it is more adequate considering a higher number of recommended configurations. Hence, PSO search would be more indicated in situations in which the user could afford the costs of evaluating a higher number of parameter configurations. The preference

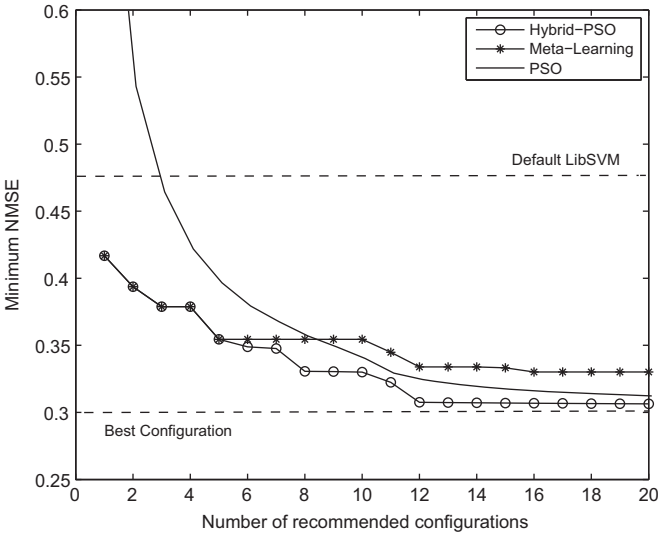

Fig. 4. TopN—NMSE result obtained at each recommended configuration.

among quality and cost of the recommendation process is clearly dependent on the user's domain. Multi-criteria metrics could be adopted to support the user in this decision (e.g., [30]).

By considering the Hybrid-PSO solution, we can see that it was able to combine the advantages of its individual components, as it can be seen in Fig. 4. The performance of the Hybrid-PSO in the initial five recommended configurations is of course the same as the performance of meta-learning (since the initial configurations are recommended by meta-learning). From that point of the TopN curve, the Hybrid-PSO consistently achieves better results compared to both the PSO and the meta-learning. It converges earlier to solutions with similar NMSE values compared to the best configurations observed in the 40 problems.

The good performance of the hybrid approach can also be confirmed by considering the number of problems that it achieved lower NMSE values compared to the values yielded by its components (see Fig. 5). As it can be seen, the hybrid approach obtained lower NMSE values in more than a half of the problems (i.e. a higher number of successes on the 40 problems) compared to both PSO and meta-learning. It was also observed a positive relative gain of Hybrid-PSO in percentage terms when compared to its components, specially considering the comparison with the pure PSO (see Fig. 6). In order to take into account the variability of the results, we also applied in this experiment the Wilcoxon test at a 95% level of confidence for each number of recommended configurations. In this experiment, the Hybrid-PSO was statistically better than the randomly initialized PSO for all numbers of

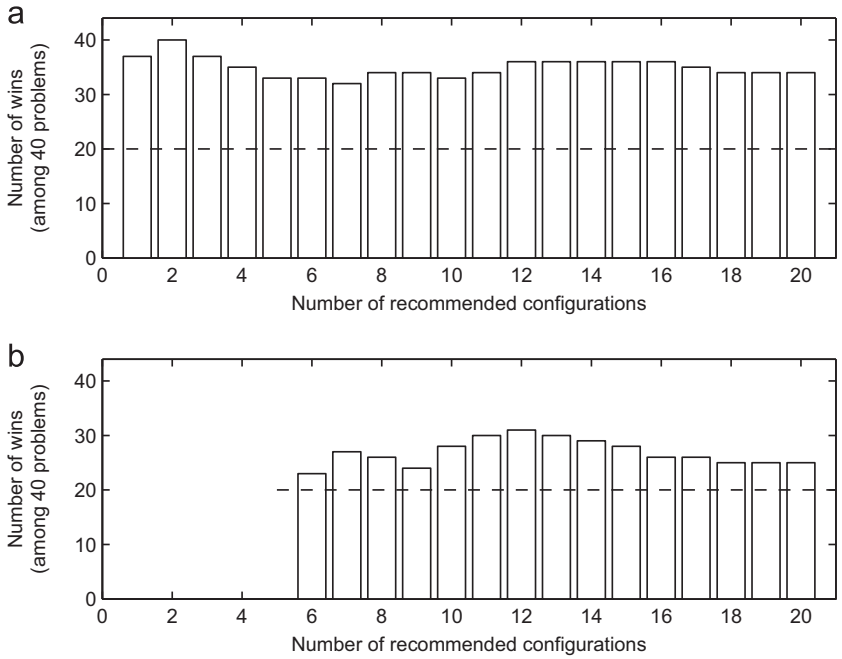

Fig. 5. Number of wins (among 40 problems) obtained by the hybrid method compared to its components.

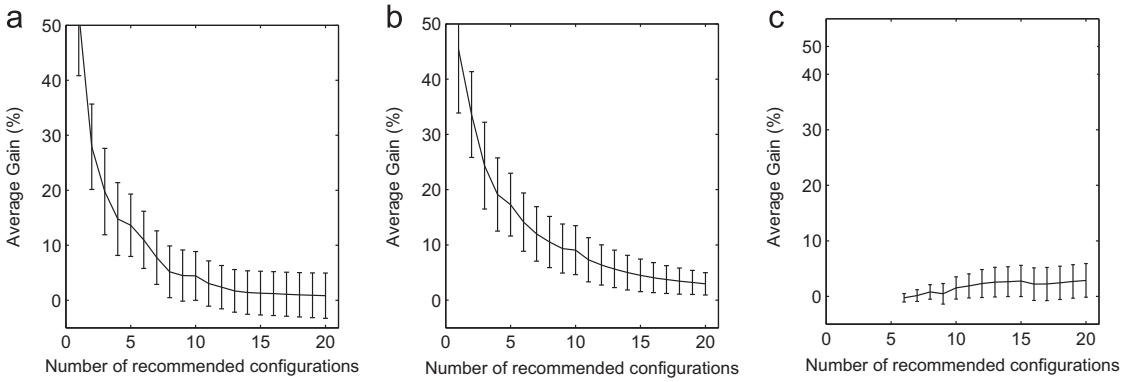

Fig. 6. Average gain in percentage terms at each number of recommended configurations for each pair of evaluated methods: PSO, Meta-learning and Hybrid-PSO.

recommendations. By comparing the Hybrid-PSO and the meta-learning in isolation, we verified a statistical gain from 11 to 14 recommendations and, in the other points, the methods were equivalent. This result closely reflects the number of wins which can be observed in Fig. 5.

In the above experiments, we assume that the meta-features are relevant to identify similar problems and hence to support the suggestion of good solutions for new problems. However, one could argue that good results in the hybrid solution could be achieved by initializing the PSO population with *any* parameter configuration well-succeeded in the past, independently on the meta-features' values. In this case, the meta-features would be just noise. In order to consider this issue, we performed an additional experiment evaluating the PSO initialized with the best configurations of parameters on $k=5$ meta-examples chosen at random in the MDB. This is essentially a new hybrid PSO in which the meta-features are not taken into account and the meta-examples are chosen randomly. Fig. 7 presents the TopN curves observed for the modified version of PSO initialized with the best configurations of random meta-examples and for the Hybrid-PSO. As it can be seen, the Hybrid-PSO which considers

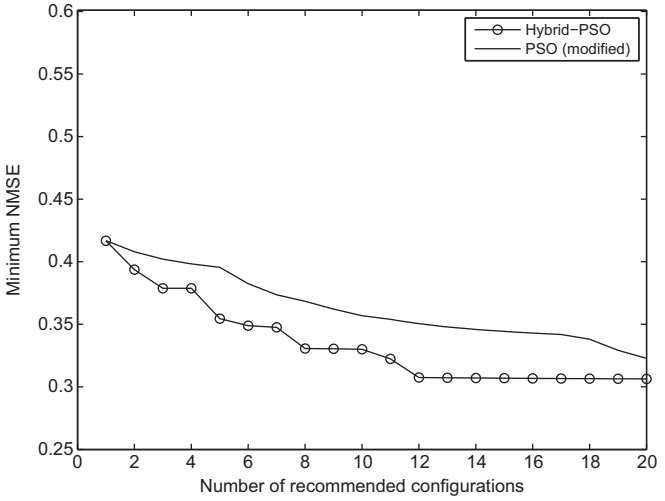

Fig. 7. TopN curves evaluating the Hybrid-PSO against PSO initialized with good solutions observed in randomly chosen meta-examples.

the meta-features to retrieve the most similar problems is better than the hybrid PSO using random meta-examples. This result provides an additional evidence of the usefulness of meta-features to recommend SVM parameters.

5.2. Hybrid-TS method

In our proposal, we emphasized that the Search module is used just to refine the initial configurations recommended by the meta-learner. Hence, we envisioned whether good results could also be achieved by deploying a local search technique in our hybrid solution. In this sub-section, we present the experiments performed using the Tabu Search algorithm combined to meta-learning.

As in the previous experiments with PSO, we adopted here a leave-one-out procedure to evaluate the Hybrid-TS method on the 40 available problems. At each step of leave-one-out, one meta-example was left out to evaluate the hybrid method. For each problem left out, a number of $k=5$ configurations was initially retrieved by the meta-learner. The best retrieved configuration is then defined as the initial search point for the Tabu Search (as described in Section 4.4). As a basis of comparison, this same experiment was performed with a randomly initialized Tabu search. Both evaluated methods were executed 1000 times and the average NMSE values were recorded.

Fig. 8 presents the average topN curves across the 40 problems considering both the PSO and the TS algorithms (with random initialization). As it can be seen, the PSO was better than Tabu search when the algorithms started their search with random initial points. The PSO starts its search by performing a global exploration of the search space and progressively performs a fine search (as a result of the time decreasing inertia parameter). This strategy was in fact adequate in our context. The Tabu search, in turn, only performs local exploration. Therefore, it was not as successful as PSO in quickly finding good regions in the search space.

On other hand, Fig. 9 shows that the relative performance of PSO and TS did not present the same behavior when they are part of the hybrid solution. As it can be seen, the Hybrid-TS method can be even better than Hybrid-PSO, as shown by the 8–12 recommended configurations. The good performance of the Hybrid-TS method can also be inferred by considering the number of wins obtained by this method compared to Hybrid-PSO (see Fig. 10). The Hybrid-TS consistently obtained a higher number of wins (among the 40 problems), specially if one considers its initial

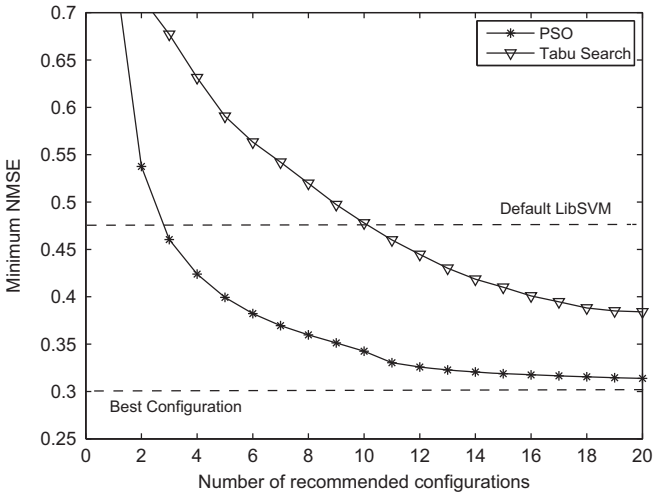

Fig. 8. TopN curves comparing PSO vs TS.

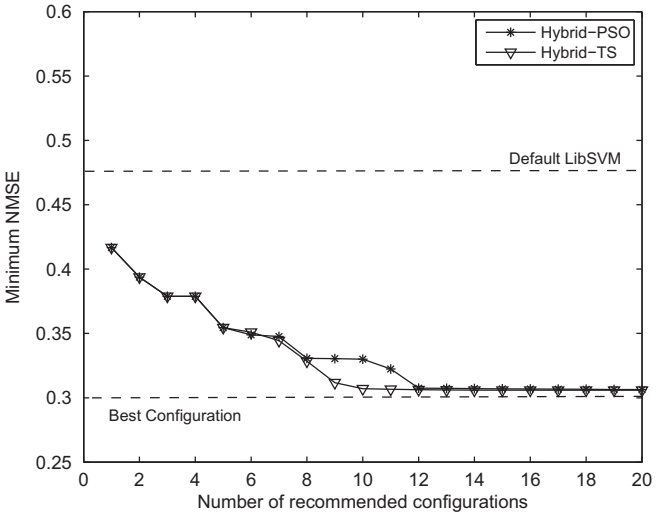

Fig. 9. TopN curves comparing Hybrid-PSO vs Hybrid-TS.

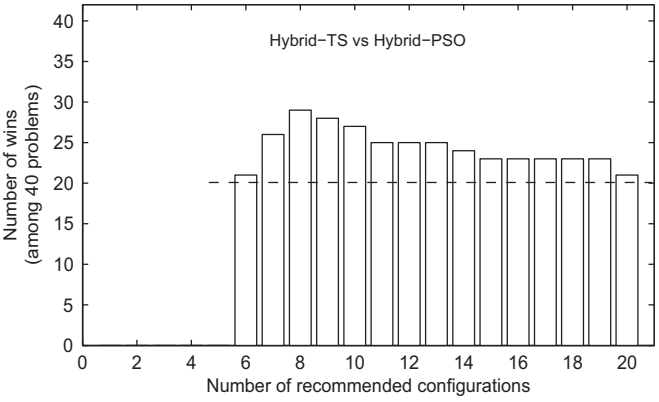

Fig. 10. Number of wins (among 40 problems) obtained by the Hybrid-TS vs Hybrid-PSO.

recommendations (a peak was observed at eight recommended configurations). The results suggest that TS was more successful to rapidly explore the neighborhood of the solutions recommended by the meta-learning module. In this experiment, we also applied the Wilcoxon test (as applied in the previous section). We verified that the Hybrid-TS was statistically better than TS for all number of recommendations. Compared to meta-learning in isolation, the Hybrid-TS was statistically better from 8 to 12 recommendations and equivalent in the other points. Finally, the Hybrid-TS was statistically better than Hybrid-PSO from 7 to 11 recommendations (which reflects the results shown in Fig. 10) and equivalent in the other points.

Although the PSO strategy of balancing between global and local exploration was more successful when starting from random points, it was not the best choice in the hybrid solution. In fact, by starting the search at promising solutions provided by meta-learning, a global exploration is no more necessary. Contrarily, a global exploration can move away from the regions that would have to be actually explored. In the final part of the curves, the hybrid methods become more similar, since PSO works more as a local mechanism (since the inertia value is smaller).

6. Conclusion

In this work, we combined meta-learning and search techniques to the problem of SVM parameter selection. Two search

algorithms, PSO and TS, were used to optimize two parameters of SMVs (γ, C). These algorithms were used in two different ways: as hybrid methods, using the initial population suggested by meta-learning, and with a random initial population. In the experiments performed, 40 regression problems were used to generate meta-examples. According to the experimental results, the proposed approach was able to find adequate parameters in a lower number of iterations compared to the randomly initialized algorithms. The results also showed, for the hybrid methods that the use of TS can lead to better parameter values than the use of PSO. The improvement in results achieved by our hybrid solution have shown to be statistically significant.

In future work, we intend to augment the number of meta-examples as we believe that the performance of the proposed approach can be improved as more meta-examples are considered. Besides, different search techniques can be considered in the future implementations. Our experiments suggested that the search process just refines the initial solutions provided by meta-learning. Hence, we believe that simpler techniques (e.g., hill climbing) once adopted in the Search Module can achieve good relative results compared to more complex technique (e.g., PSO). Finally, we intend to evaluate the proposed solution in other case studies, such as SVM parameter selection for classification problems.

Acknowledgments

The authors would like to thank CNPq, CAPES, FAPESP and FACEPE (Brazilian Agencies) and FCT project Rank (PTDC/EIA/81178 /2006) for their financial support.

References

[1] C. Soares, P. Brazdil, P. Kuba, A meta-learning approach to select the kernel width in support vector regression, Machine Learning 54 (3) (2004) 195–209.
[2] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods, Cambridge University Press, 2000.
[3] O. Chapelle, V. Vapnik, O. Bousquet, S. Mukherjee, Choosing multiple parameters for support vector machines, Machine Learning 46 (1) (2002) 131–159.
[4] N. Cristianini, C. Campbell, J. Shawe-Taylor, Dynamically adapting kernels in support vector machines, in: Proceedings of Neural Information Processing Workshop, 1998, pp. 204–210.
[5] S. Ali, K.A. Smith-Miles, A meta-learning approach to automatic kernel selection for support vector machines, Neurocomputing 70 (1–3) (2006) 173–186.
[6] S. Lessmann, R. Stahlbock, S. Crone, Genetic algorithms for support vector machine model selection, in: International Joint Conference on Neural Networks, 2006, pp. 3063–3069.
[7] F. Friedrichs, C. Igel, Evolutionary tuning of multiple SVM parameters, Neurocomputing 64 (2005) 107–117.
[8] A. Lorena, A. de Carvalho, Evolutionary tuning of SVM parameter values in multiclass problems, Neurocomputing 71 (2008) 16–18.
[9] S.S. Keerthi, C.-J. Lin, Asymptotic behaviors of support vector machines with Gaussian kernel, Neural Computation 15 (7) (2003) 1667–1689.
[10] P. Kuba, P. Brazdil, C. Soares, A. Woznica, Exploiting sampling and meta-learning for parameter setting support vector machines, in: Proceedings of the IBERAMIA 2002, 2002, pp. 217–225.
[11] C. Soares, P. Brazdil, Selecting parameters of SVM using meta-learning and kernel matrix-based meta-features, in: Proceedings of SAC, 2006, pp. 564–568.
[12] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Joint Conference on Neural Networks, 1995, pp. 1942–1948.
[13] F. Glover, M. Laguna, Tabu Search, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.
[14] G. Cawley, Model selection for support vector machines via adaptive step-size tabu search, in: International Conference on Artificial Neural Networks and Genetic Algorithms, 2001, pp. 434–437.
[15] X. Guo, J. Yang, C. Wu, C. Wang, Y. Liang, A novel LS-SVMS hyper-parameter selection based on particle swarm optimization, Neurocomputing 71 (2008) 3211–3215.
[16] B. de Souza, A. de Carvalho, R. Ishii, Multiclass SVM model selection using particle swarm optimization, in: Sixth International Conference on Hybrid Intelligent Systems, 2006, pp. 441–446.
[17] S. Ali, K.A. Smith, Matching SVM kernel's suitability to data characteristics using tree by fuzzy c-means clustering, in: Third International Conference on Hybrid Intelligent Systems, 2003, pp. 553–562.
[18] S. Ali, K. Smith-Miles, On optimal degree selection for polynomial kernel with support vector machines: Theoretical and empirical investigations, KES Journal 11 (1) (2007) 1–18.
[19] S. Louis, J. McDonnell, Learning with case-injected genetic algorithms, IEEE Transactions on Evolutionary Computation 8 (4) (2004) 316–328.
[20] K. Smith-Miles, Cross-disciplinary perspectives on meta-learning for algorithm selection, ACM Computing Surveys 41 (1) (2008) 1–25.
[21] K. Smith-Miles, Towards insightful algorithm selection for optimisation using meta-learning concepts, in: Proceedings of the IEEE International Joint Conference on Neural Networks, 2008, pp. 4118–4124.
[22] J. Kanda, A. Carvalho, E. Hruschka, C. Soares, Using meta-learning to classify traveling salesman problems, in: Brazilian Symposium on Neural Networks, 2010, pp. 73–78.
[23] T. Gomes, R.B.C. Prudêncio, C. Soares, A. Rossi, A. Carvalho, Combining meta-learning and search techniques to SVM parameter selection, in: Brazilian Symposium on Neural Networks, 2010, pp. 79–84.
[24] P. Brazdil, C. Giraud-Carrier, C. Soares, R. Vilalta, Metalearning: Applications to Data Mining, Cognitive Technologies, Springer, 2009.
[25] R.B.C. Prudêncio, T.B. Ludermir, Meta-learning approaches to selecting time series models, Neurocomputing 61 (2004) 121–137.
[26] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines (2001).
[27] V. Vapnik, Statistical Learning Theory, Wiley, New York, NY, 1998.
[28] C.-W. Hsu, C.-C. Chang, C.-J. Lin, A practical guide to support vector classification, Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, 2003.
[29] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: IEEE World Congress on Computational Intelligence, 1998, pp. 69–73.
[30] P. Brazdil, C. Soares, J.P. Costa, S. Louis, J. McDonnell, Ranking learning algorithms—using IBL and meta-learning on accuracy and time results, Machine Learning 50 (3) (2003) 251–277.
[31] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics 1 (1945) 80–83.

Taciana A.F. Gomes received her B.Sc. degree in Computer Science from the Universidade Catolica de Pernambuco and her M.Sc. degree in Computer Science from the Universidade Federal de Pernambuco, Brazil. Her main interests are Machine Learning, Hybrid Intelligent Systems, Meta-learning, Optimization and Support Vector Machines.

Ricardo B.C. Prudêncio received his B.Sc. degree in Computer Science from the Universidade Federal do Ceara and his M.Sc. and Ph.D. degrees in Computer Science from the Universidade Federal de Pernambuco, Brazil. He is a lecturer at the Center of Informatics, Universidade Federal de Pernambuco, Brazil. His main interests are Machine Learning, Meta-learning, Hybrid Intelligent Systems, Time Series Forecasting and Text Mining.

Carlos Soares received his B.Sc. degree in Systems Engineering and Informatics from Universidade do Minho, Portugal. He received his M.Sc. degree in Artificial Intelligence and his Ph.D. in Computer Science from Universidade do Porto, Portugal. He is a lecturer at Faculdade de Economia da Universidade do Porto. His main interests are Machine Learning, Data Mining, Meta-learning and Data Streams.

Andre L.D. Rossi received his B.Sc. degree in Computer Science from the Universidade Estadual de Londrina and his M.Sc. degree in Computer Science from Universidade de São Paulo, Brazil. He is a Ph.D. student at Universidade de São Paulo, Brazil. His main interests are Machine Learning, Data Mining, Bioinformatics, Evolutionary Computation, Bioinspired Computing, Optimization and Data Streams.

André Carvalho received his B.Sc. and M.Sc. degrees in Computer Science from the Universidade Federal de Pernambuco, Brazil. He received his Ph.D. degree in Electronic Engineering from the University of Kent, UK. Prof. André de Carvalho is Full Professor at the Department of Computer Science, Universidade de São Paulo, Brazil. He has published around 80 Journal and 200 Conference refereed papers. He has been involved in the organization of several conferences and journal special issues. His main interests are Machine Learning, Data Mining, Bioinformatics, Evolutionary Computation, Bioinspired Computing and Hybrid Intelligent Systems.