Combining Meta-Learning with Multi-Objective Particle Swarm Algorithms for SVM Parameter Selection: An Experimental Analysis

Péricles B. C. Miranda, Ricardo B. C. Prudêncio
Universidade Federal de Pernambuco
Recife, Brazil
{pbc,mrbc}@cin.ufpe.br

Andre Carlos P. L. F. de Carvalho
Universidade de São Paulo
São Carlos, Brazil
andre@icmc.usp.br

Carlos Soares
Universidade do Porto
Porto, Portugal
csoares@fep.up.pt

Abstract—Support Vector Machines (SVMs) have become a well succeeded technique due to the good performance it achieves on different learning problems. However, the SVM performance depends on adjustments of its parameters’ values. The automatic SVM parameter selection is treated by many authors as an optimization problem whose goal is to find a suitable configuration of parameters for a given learning problem. This work performs a comparative study of combining Meta-Learning (ML) and Multi-Objective Particle Swarm Optimization (MOPSO) techniques for the SVM parameter selection problem. In this combination, configurations of parameters provided by ML are adopted as initial search points of the MOPSO techniques. Our hypothesis is that, starting the search with reasonable solutions will speed up the process performed by the MOPSO techniques. In our work, we implemented three MOPSO techniques applied to select two SVM parameters for classification. Our work’s aim is to optimize the SVMs by seeking for configurations of parameters which maximize the success rate and minimize the number of support vectors (i.e., two objective functions). In the experiments, the performance of the search algorithms using a traditional random initialization was compared to the performance achieved by initializing the search process using the ML suggestions. We verified that the combination of the techniques with ML obtained solutions with higher quality on a set of 40 classification problems.

Keywords—Multi-Objective optimization; Meta-learning; Parameter Selection Problem; SVM;

I. INTRODUCTION

The SVM performance strongly depends on the adequate choice of its parameters, and an exhaustive trial-and-error procedure for selecting good values of parameters is not practical for computational reasons[6]. Hence, the selection of SVM parameters is commonly treated by different authors as an optimization problem in which a search technique is used to find adequate configurations of parameters for the problem at hand. In literature, different techniques were applied to this problem including Evolutionary Algorithms (EA) [2], Particle Swarm Optimization (PSO) [2] and Tabu Search [3]. Previous work commonly used single objective techniques for SVM parameter selection, however this is not totally adequate since this task is inherently a Multi-Objective Optimization (MOO) problem [4]. In this context, we can mention the use of Multi-Objective EA (MOEA) [4], Multi-Objective PSO (MOPSO) [5] and the use of Gradient-Based techniques [6], which considered multiple objectives. Although the application of search techniques represents an automatic solution to select SVM parameters, this approach can be very expensive, since a large number of candidate configurations of parameters is often evaluated during the search [1].

An alternative approach to SVM parameter selection is the use of Meta-Learning (ML), which treats parameter selection as a supervised learning task [1][10]. Each training example for ML (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, a meta-learner predicts the most suitable configuration of parameters for a new problem based on its characteristics. ML 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 several candidate configurations. However, ML is very dependent on the quality of its meta-examples. In general the number of problems available for meta-example generation is limited and the data is noisy, requiring a careful treatment of the data.

In a recent work [11], ML and search techniques were combined for SVM parameter selection. In this work, ML was adopted to suggest a number of solutions (configurations of parameters) which are adopted as the initial population of the search technique. The search technique just refined a promising solution returned by ML, speeding up the optimization process. The authors adopted as search technique the single objective PSO, whose objective was to minimize the error rate obtained by the SVM. Despite the good results obtained by this combination, as said the use of multiple objectives would be more adequate. In order to overcome this limitation, in [12], we proposed the use of ML initialization of MOO search techniques. More specifically, ML suggested configurations of parameters which were optimized by the MOPSO algorithm. The choice of swarm optimization in our research is due to its good performance.
in difficult optimization tasks and its simplicity. In fact, PSO became a popular, easy-to-use and extendable technique. The results presented in [12] showed that ML can be used as an interesting alternative to improve the MOPSO algorithm’s convergence.

In the current work, we extend our previous research by performing a more complete study of ML influence on the optimization process of search algorithms. We implemented three well-known MOPSO techniques for comparison: MOPSO, MOPSO-CDR and CSS-MOPSO. These techniques were applied in our work to select two SVM parameters: the parameter $\gamma$ of the RBF kernel and the regularization constant $C$, which may have a strong influence in SVM performance [6]. In our work, a database of 40 meta-examples was produced from the evaluation of a set of 399 configurations of ($\gamma$, $C$) on 40 different classification problems. Each classification problem was described by a number of 8 meta-features proposed in [1]. All the implemented MOO algorithms were used to optimize the parameters ($\gamma$, $C$) regarding the success rate and number of support vectors observed in the SVMs.

In our experiments, we performed three analysis comparing the performance of the implemented algorithms. The first one compares the techniques which use random initialization of the swarm (traditional approach), the second analysis compares the techniques which use ML suggestions as initial solutions (hybrid approach) and the third analysis compares all involved techniques evaluating the influence of ML in the search process. The results reveal that the hybrid approach was able to generate better solutions along the generations when compared to the traditional approach.

Section II presents the basic concepts of MOO. Section III presents details of the proposed work. Section IV describes the experiments and obtained results. Finally, Section V presents some conclusions and the future work.

II. MULTI-OBJECTIVE OPTIMIZATION

In contrast to single objective techniques, MOO aims to optimize more than one objective at the same time. MOO can be defined as the problem of finding a decision variable vector which satisfies constraints and optimizes a vector of functions whose elements represent objective functions. A general multi-objective minimization problem can be defined as [7]:

$$\text{minimize } \vec{f}(\vec{z}) := [f_1(\vec{z}), f_2(\vec{z}), ..., f_n(\vec{z})],$$  \hspace{1cm} (1)

subject to:

$$g_i(\vec{z}) \leq 0 \quad i = 1, 2, ..., p,$$  \hspace{1cm} (2)

$$h_j(\vec{z}) = 0 \quad j = 1, 2, ..., q,$$  \hspace{1cm} (3)

where $\vec{z} = (z_1, z_2, ..., z_m) \in \mathbb{R}^n$ is the vector on the decision search space; $n$ is the number of objectives and $g_i(\vec{z})$ and $h_j(\vec{z})$ are the constraint functions and $p+q$ is the number of constraints of the problem. Given two vectors $\vec{z}, \vec{u} \in \mathbb{R}^m$, $\vec{z}$ dominates $\vec{u}$ (denoted by $\vec{z} \preceq \vec{u}$) if $\vec{z}$ is better than $\vec{u}$ in at least one objective and $\vec{z}$ is not worse than $\vec{u}$ in any objective. $\vec{z}$ is not dominated if does not exist another current solution $\vec{z}_i$ in the current population, such that $\vec{z}_i \prec \vec{z}$. The set of non-dominated solutions in the objective space is known as pareto front.

III. DEVELOPED WORK

The work presented here performs a comparison between the traditional and hybrid approach aiming to analyze the influence of ML in the search process. As context, we adopted the SVM parameter selection problem for classification. Figure 1 presents the general architecture of the proposed solution. Initially, the Meta-Learner module retrieves a predefined number of past meta-examples stored in a Database (DB), selected on the basis of their similarity to the input problem. The process of suggesting meta-examples is not trivial as in a single objective approach. As we are dealing with a multi-objective problem, the dominance evaluation, showed in the previous section, defines the set of non-dominated solutions among all configurations. After that, the Meta-learner is able to perform the suggestion of non-dominated meta-examples. Following, the Search module adopts as initial search points the configurations of parameters which were well-succeeded on the retrieved meta-examples. The search module iterates its search process by generating new candidate configurations to be evaluated in the SVM. The output configurations of parameters will be the non-dominated solutions generated by the Search module up to its convergence or another stopping criterion.

![General Architecture](image.png)

**Figure 1.** General Architecture, where (1) is the success rate and (2) the number of support vectors.

A. Search Module

In this work, we implemented three MOO techniques, shown in Section III-A1, and adapted to perform the search for configurations ($\gamma$, $C$). The choice of RBF kernel is due to its flexibility in different problems compared to other kernels [10]. It is known that the $\gamma$ and $C$ parameters have an important influence in learning performance since they control the linearity and complexity of the SVM [13].

The objective functions evaluate the quality and complexity of each configuration of parameters on a given
classification problem. In our work, given a SVM configuration, we define the objective functions as the success rate (SR), the most direct way to evaluate the quality of a SVM model; and the number of support vectors (NSV), which influences the space and time complexity of the SVM [4]. Both are obtained by the SVM in a 10-fold cross validation experiment. So, the objectives were to find the non-dominated configurations of \((\gamma, C)\) trying to maximize the SR and minimize the NSV for a given classification problem.

In our implementation, each particle \(i\) represents a configuration \(x_i = (\gamma, C)\), 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. The basic MOPSO algorithm 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 MOPSO is given by the following equations:

\[
\vec{v}_i(t + 1) = w\vec{v}_i(t) + c_1r_1(\vec{p}_i(t) - \vec{x}_i(t)) + c_2r_2(\vec{n}_i(t) - \vec{x}_i(t)), \quad (4)
\]

\[
\vec{x}_i(t + 1) = \vec{x}_i(t) + \vec{v}_i(t + 1), \quad (5)
\]

In Equation 4, \(\vec{p}_i(t)\) is the best position achieved by the particle so far, and \(\vec{n}_i(t)\) is the best position achieved by any particle in the population so far. This equation seems to be similar to the single PSO equation, however, the process of updating the \(\vec{n}_i(t)\) makes each particle moving in direction of the best global positions achieved from the repository of non-dominated solutions (pareto front) through roulette wheel. The parameters \(w\), \(c_1\) 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. The parameters \(r_1\) and \(r_2\) are random numbers used to enhance the diversity of particle positions.

In our work, the algorithms were implemented to perform a search in a space represented by a discrete grid of SVM configurations, consisting of 399 different settings of parameters \(\gamma\) and \(C\). By following the guidelines provided in [13], we considered the following exponentially growing sequences of \(\gamma\) and \(C\) as potentially good configurations: the parameter \(\gamma\) assumed 19 different values (from \(2^{-15}\) to \(2^{5}\)) and the parameter \(C\) assumed 21 different values (from \(2^{-5}\) to \(2^{15}\)), thus yielding \(19 \times 21 = 399\) different combinations of parameters values in the search space.

1) MOO Techniques: Here, we briefly review three well-known MOPSO techniques, previously reported in the literature, which were used in this work for comparison.

MOPSO was proposed by Coello Coello et. al [7]. The MOPSO algorithm uses an External Archive \(EA\) or repository of non-dominated solutions and the space of objectives is divided in hypercubes. The fitness of each hypercube depends on the current number of solutions of the \(EA\) inside the hypercube. A roulette-wheel is used to select a hypercube based on its fitness. Once a hypercube is selected, one of the particles inside the hypercube is randomly chosen to be the social leader. MOPSO also uses a greedy local search operator.

CSS-MOPSO was proposed by Chiu et. al [9]. The CSS-MOPSO does not use cognitive leaders. On the other hand, two social leaders are selected simultaneously \(\vec{n}_i^1(t)\) and \(\vec{n}_i^2(t)\). \(\vec{n}_i^1(t)\) is selected based on the angle between the datum line, which connects the analyzed solution of the \(EA\) and the datum point, and the line that connects the the analyzed solution of the \(EA\) and the particle. \(\vec{n}_i^2(t)\) is selected according to the fitness value \(f_i\) of a randomly selected objective at each iteration. For a particle with even index, CSS-MOPSO chooses the solution in the \(EA\) whose \(f_i\) is higher and nearest. For a particle with odd index, CSS-MOPSO chooses the solution in the \(EA\) whose \(f_i\) is lower and nearest.

MOPSO-CDR was proposed by Santana et. al [8] in 2009. It was inspired by MOPSO-CDLS algorithm and it incorporates a roulette wheel selection based on the crowding distance to select the \(\vec{n}_i(t)\) and to prevent an excessive number of non-dominated solutions in the \(EA\). Solutions with lower Crowding Distance have more chance to be selected as a social leader. Furthermore, MOPSO-CDR presents a novel procedure to update the \(\vec{p}_i(t)\). The \(\vec{p}_i(t)\) of a particle is updated if the new position of the particle dominates the current \(\vec{p}_i(t)\). If the new position and the \(\vec{p}_i(t)\) are incomparable, the \(EA\) is used. The algorithm searches in the \(EA\) for the nearest solution to the \(\vec{p}_i(t)\) and for the nearest solution to the new position. If the closer solution in the \(EA\) to the new position is in a less crowded region than the closer solution in the \(EA\) to the \(\vec{p}_i(t)\), the new position will be chosen as the new \(\vec{p}_i(t)\). Otherwise, the old \(\vec{p}_i(t)\) remains. The mutation operator is the same used in the MOPSO [7] and it is applied at each iteration with a bounded influence.

B. Meta-Database

To create meta-examples, we collected 40 datasets corresponding to 40 different classification problems, available in the UCI Repository [14]. Each meta-example is related to a single classification problem and stores: (1) a vector of meta-features describing the problem; and (2) the objective grid that stores the success rate and number of support vectors obtained by the SVM for each configuration \((\gamma, C)\). The objective grid consists of 399 different settings of parameters \(\gamma\) and \(C\).

1) Meta-Features: In this work, we used 8 meta-features to describe the datasets of classification problems. These meta-features were selected from the set of features defined in [15]. We adopted meta-features divided in three categories: 1) Simple, composed of Number of examples, attributes and classes, 2) Statistical, composed by Mean
correlation of attributes, Skewness and Kurtosis, and 3) Information Theory, composed by Entropy of class.

The group of statistical values is composed by the mean correlation of attributes; Skewness, which measures the asymmetry of the distribution regarding the central axis [15]; Kurtosis, which measure the dispersion (characterized by the flatness of the distribution curve) [15] and the geometric mean of the attributes which evaluates the mean of the data standard deviation. Finally, the information theory group measure the randomness of the instances; being composed by the Entropy which defines the degree of uncertainty of classification [15].

2) Objective Grid: The objective grid stores the SR and NSV obtained by the SVM on a problem considering different SVM configurations. For each one of the 399 configurations, a 10-fold cross validation experiment was performed to collect SVM SR and NSV. The obtained 399 objective values were stored in the objective grid. We highlight here that the objective grid is equivalent to the search space explored by the search technique. By generating an objective grid for a problem, we can evaluate which configurations of parameters were the best ones in the problem (i.e., the best points in a search space) and we can use this information to guide the search process for new similar problems.

C. Meta-Learner

Given a new input problem described by the vector $\vec{i} = (i_1, ..., i_p)$, the Meta-Learner selects the $k$ most similar problems according to the distance between meta-attributes. The distance function implemented was the Euclidean Distance. After that, we apply the dominance evaluation in the 399 configurations and generate a pareto front for each one of the $k$ most similar problems. In order to suggest an initial population, we select one random solution of each produced pareto front. Random non-dominated solutions of different problems were sampled in order to enhance diversity of the initial population.

IV. EXPERIMENTS

All traditional and hybrid techniques were evaluated by following a leave-one-out methodology described as follows: 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 DB to be selected by the ML module. A number of $k$ meta-examples were suggested by the ML module as the initial population of a technique (in our experiments, we adopted $k = 5$). The technique then optimized the SVM configurations for the problem left out up to the number of 10 generations. In each generation, a pareto front is formed and evaluated. This procedure was repeated 30 times and the average of the results was used to guarantee reliability. This experiment performed the mean of the metrics values of all problems for each generation.

A. Metrics

In our experiments, we evaluated the results (i.e., the pareto fronts) obtained by all hybrid and traditional algorithms for each problem according to different quality metrics usually adopted in the literature of MOO. The adopted metrics were: Hypervolume, Maximum Spread and Coverage. Each metric considers a different aspect of the pareto front.

1) Hypervolume $HV$: this metric was proposed by Zitzler and Thiele [16] and is defined by the hypervolume in the space of objectives covered by the obtained pareto front ($P^*$). For MOP with $w$-objectives, $HV$ is defined by:

$$HV = \left\{ \bigcup_i a_i \mid s_i \in P^* \right\},$$

where $s_i$ ($i = 1, 2, ..., n$) is a non-dominated solution of the pareto front ($P^*$), $n$ is the number of solutions in the Pareto Front and $a_i$ is the hypervolume of the hypercube delimited by the position of solution $s_i$ in the space of objectives and the origin. In practice, this metric gives the size of the dominated space, which is also called the area under the curve. A large value of $HV$ is desired.

2) Maximum Spread $MS$: it was proposed by Zitzler et al [16] and evaluates the maximum extension covered by the non-dominated solutions in the pareto front. $MS$ is computed by using Eq. (7).

$$MS = \sqrt{\sum_{m=1}^{P} (\max_{i=1}^{n} f_{m}^i - \min_{i=1}^{n} f_{m}^i)^2},$$

where $n$ is number of solutions in the pareto front and $k$ is the number of objectives. This measure can be used to compare the techniques and thus define which of them covers a bigger extension of the search space. Hence, large values of this metric are preferred.

3) Coverage $CV$: was proposed by Zitzler et al [16]. $CV$ is evaluated by using (8):

$$CV(A, B) = \frac{|\{b \in B; \exists a \in A : a \succeq b\}|}{|B|},$$

where $A$ and $B$ are two sets of non-dominated solutions. The value $CV(A, B) = 1$ means that all solutions in $B$ are weakly dominated by $A$. On the other hand, $CV(A, B) = 0$ means that none of the solutions in $B$ are weakly dominated by $A$. Note that both $CV(A, B)$ and $CV(B, A)$ have to be evaluated, since $CV(A, B)$ is not necessarily equal to $1 - CV(B, A)$. If $0 < CV(A, B) < 1$ and $0 < CV(B, A) < 1$, then neither $A$ totally dominates $B$ nor $B$ totally dominates $A$.

B. Algorithm Settings

We present the values of the parameters adopted for the search algorithms. The same values suggested by Coello et al [7], Bastos-Filho[8] and Chiu[9] were used:
- Number of particles: 5
- Mutation rate: 0.5 for MOPSO and MOPSO-CDR [8], 0.01 for CSS-MOPSO [9]
- Inertia factor $\omega$: linearly decreases from 0.9 to 0.4
- Constants $c_1$ and $c_2$: 1.49
- Pareto front limit (solutions): 10 solutions
- Number of generations: 10

C. Results

In order to analyze our results adequately we performed statistical analysis. As the data does not follow a normal distribution, we applied the Wilcoxon test to verify hypothesis. All the following analysis used this methodology.

Figures 2 and 3 show the average metric values per generation including all problems for HV and MS respectively. In our analysis, initially we compare all traditional algorithms. After that, we compare all hybrid algorithms. Finally, we compare the hybrid with the traditional algorithms aiming to verify whether ML really improves the optimization process.

In Figure 2, considering only the traditional approach, the MOPSO-CDR and CSS-MOPSO achieved the best results for HV. Both were considered statistically equal in performance, out-performing the MOPSO technique with 95% of confidence since the second generation. Similarly, regarding MS, as we can see in Figure 3, the MOPSO-CDR and CSS-MOPSO achieved better results than MOPSO with 95% of confidence since the second generation. However, the CSS-MOPSO did not achieve as good results as MOPSO-CDR for MS, being overcome since the second generation with 95% of confidence.

In Figure 2, considering only the hybrid approach, it is difficult to define, visually, which technique achieved the best performance for HV. Applying the Wilcoxon Test we conclude that all hybrid techniques are statistically equal considering this metric. Regarding MS, in Figure 3, the HMOPSO-CDR out-performed the other techniques, in the three initial generations with 95% of confidence. This behaviour can be explained due to HMOPSO-CDR’s mechanism of selecting social leader from the EA using crowding distance, focusing on MS. However, the HMOPSO-CDR stabilizes its quality and from the third generation onwards, the HMOPSO-CDR and HCSS-MOPSO are statistically equal. In contrast, the HMOPSO did not achieve good performance, being out-performed by HMOPSO-CDR and HCSS-MOPSO with 95% in each generation.

After an individual analysis of traditional and hybrid approaches, we compared the performance of both algorithms. Considering HV, as shown in Figure 2, in the six initial generations, all hybrid techniques out-performed all traditional ones with at least 95% of confidence. In the following generations to the end, all techniques, hybrid and traditional, achieved a similar performance being considered equal statistically. Although there is equality in last generations, the techniques which achieved better results since the beginning are considered more suitable for costly problems.

The positive influence of ML-based suggestions along the optimization also can be seen in Figure 3, where two hybrid techniques, HMOPSO-CDR and HCSS-MOPSO, out-performed all traditional techniques from the first to the seventh generation with at least 95% of confidence. In the following generations, two traditional techniques, MOPSO-CDR and CSS-MOPSO, equalized the performance of their respective hybrid versions. In contrast, the MOPSO and HMOPSO did not achieve a good performance, being out-performed by all techniques from the second generation to the end with 95% of confidence.

Besides the previous metrics analysis, we studied the coverage of all generated paretos per generation. Table I
shows the average of the CV values in all generations for each combination of approaches. We used the following legends to represent the non-hybrid algorithms: CSS-MOPSO (NH1), MOPSO (NH2) and MOPSO-CDR (NH3), and the hybrid algorithms: HCSS-MOPSO (H1), HMOPSO (H2) and HMOPSO-CDR (H3). As it can be seen in Table I, the best approximated pareto fronts were generated by hybrid techniques, achieving at least 81% of dominance over the traditional techniques. According the CV, the HMOPSO-CDR (H3) presented the best pareto among all remnant approaches, being dominated in 2% by HMOPSO (H2) and 30% by HCSS-MOPSO (H1).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>NH1</th>
<th>NH2</th>
<th>NH3</th>
<th>H1</th>
<th>H2</th>
<th>H3</th>
</tr>
</thead>
<tbody>
<tr>
<td>NH1</td>
<td>0.88</td>
<td>0.20</td>
<td>0.00</td>
<td>0.13</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>NH2</td>
<td>0.12</td>
<td>0.08</td>
<td>0.00</td>
<td>0.06</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>NH3</td>
<td>0.77</td>
<td>0.85</td>
<td>-</td>
<td>0.31</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>H1</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>0.93</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>H2</td>
<td>0.85</td>
<td>0.89</td>
<td>0.81</td>
<td>0.03</td>
<td>-</td>
<td>0.02</td>
</tr>
<tr>
<td>H3</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>0.68</td>
<td>0.96</td>
<td>-</td>
</tr>
</tbody>
</table>

Table I
AVERAGE OF CV METRIC VALUES (%) PER GENERATION APPLIED FOR EACH ALGORITHM.

V. CONCLUSION
This work combines ML with MOPSO techniques to the problem of SVM parameter selection performing a more complete study of ML influence in the optimization process of search algorithms. Three well-known techniques were used to select the parameter \(\gamma\) of the RBF kernel and the regularization parameter \(C\). In the performed experiments, we observed that the hybrid techniques were able to generate better paretos than the traditional techniques. Besides generating paretos with high coverage, the hybrid approaches achieved good quality in metrics as \(HV\) and MS. In future work, we intend to augment the number of meta-examples since reducing the noisy data we believe that the performance of the proposed approach can be improved. Also, we intend to perform experiments using other MOO algorithms combined with meta-learning. Moreover, other quality measures could be studied to evaluate better the pareto front.

ACKNOWLEDGMENT
The authors would like to thank CAPES, FACEPE and CNPq (Brazilian Agencies) for their financial support.

REFERENCES