An Experimental Study of the Combination of Meta-Learning with Particle Swarm Algorithms for SVM Parameter Selection

Péricles B. C. de Miranda[1], Ricardo B. C. Prudêncio[1], Andre Carlos P. L. F. de Carvalho[2] and Carlos Soares[3]

[1] Federal University of Pernambuco, CIn-UFPE
[2] University of São Paulo, USP
[3] University of Porto, FEP

Abstract. Support Vector Machines (SVMs) have become a well succeed algorithm due to the good performance it achieves on different learning problems. However, to perform well the SVM formulation requires adjustments on its model. Avoiding the trial and error procedure, the automatic SVM parameter selection is a way to deal with this. The automatic parameter selection is commonly considered an optimization problem whose goal is to find suitable configuration of parameters which attends some learning problem.
In the current work, we propose a study of the combination of Meta-learning (ML) with Particle Swarm Optimization (PSO) algorithms to optimize the SVM model, seeking for combinations of parameters which maximize the success rate of SVM. ML is used to recommend SVM parameters, to a given input problem, based on well-succeeded parameters adopted in previous similar problems. In this combination, initial solutions provided by ML are possibly located in good regions in the search space. Hence, using a reduced number of candidate search points, in the search process, to find an adequate solution, would be less expensive.
In our work, we implemented five benchmarks PSO approaches applied to select two SVM parameters for classification. The experiments consist in comparing the performance of the search algorithms using a traditional random initialization and using ML suggestions as initial population. This research analysed the influence of meta-learning on convergence of the optimization algorithms, verifying that the combination of PSO techniques with ML obtained solutions with higher quality on a set of 40 classification problems.

Keywords: particle swarm optimization; meta-learning; svm parameter selection.

1 Introduction

SVMs have achieved a considerable attention due to its theoretical foundations and 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 parameters 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 values of parameters is obviously not practical [3].

The process of selecting SVM parameters is commonly treated by different authors as an optimization problem in which a search algorithm is used to find the adequate configurations of parameters on the problem at hand [4]. Although it represents an automatic mode to select SVM parameters, this approach can still be very expensive, since a large number of candidate configurations of parameters is often evaluated during the search process [1].

An alternative approach to SVM parameter selection is the use of Meta-Learning (ML), which treats the SVM parameter selection as a supervised learning task [1] [5]. Each training example for ML (i.e. each meta-example) stores the characteristics of a past problem and 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 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 different candidate configurations (as performed using search techniques). However, ML is very 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 ML for SVM parameter selection may be not so good as the performance of search techniques [6].

Visualizing these drawbacks, a recent work, developed by [18], combined search techniques and ML to solve the SVM parameter selection problem for regression. In this proposal, configurations of parameters suggested by ML are adopted as initial solutions which will be later refined by the search technique. This work used as search technique the single objective version of PSO whose objective was to minimize the error rate. The results showed that the combination of ML with PSO generated better solutions in comparison to PSO with random initialization for SVM parameter selection in the cited problem. This work implemented a prototype of PSO, using the *inertia* weight and adopted *Star* as topology, and compared the PSO convergence with the convergence of PSO using ML, verifying that the suggestions of ML speed up and refine the algorithm's convergence.

Aiming to realize a more complete study of the influence of ML suggestions in the optimization process of search algorithms, we implemented five benchmarks PSO approaches for comparison: Basic PSO, Inertia Gbest (PSO using *inertia* weight and *Gbest* topology), Constricted Gbest (PSO using *constricted* factor and *Gbest* topology), Inertia Lbest (PSO using *inertia* weight and *Lbest* topology) and Constricted Lbest (PSO using *constricted* factor and *Lbest* topology).

In order to evaluate our study, we adapted the prototypes to select two SVM parameters: the parameter γ of the RBF kernel and the regularization constant C, which may have a strong influence in SVM performance [10].

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 classification problems. Each classification problem was described by a number of 8 meta-features proposed in [11] [1] [16]. All the implemented prototypes were used to optimize the parameters (γ, C) regarding the success rate on classification.

In our experiments, we, initially, performed an analysis comparing the performance of the benchmark algorithms using random initialization. After that, we realized a comparison between the benchmark algorithm which achieved the best performance with the same algorithm using ML (hybrid approach). The experiments' results revealed that the hybrid approach was able to generate better solutions along the generations when compared to the randomly initialized PSOs.

The paper is organized as follows: Section 2 brings a brief presentation on the SVM model selection. Section 3 presents details of the proposed work. Section 4 describes the experiments, obtained results and statistical analysis. Finally, Section 5 presents some conclusions and the future work.

2 SVM Parameter Selection

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 performance [12]. Different authors have deployed search and optimization techniques aiming to automatize this process and to avoid an exhaustive or a random exploration of parameters [14][4][7][8][9][12][13].

In this context, solutions are combinations of parameters and the objective function corresponds to SVM execution. Different techniques were proposed as based on gradients [14], evolutionary algorithms [7][8][9], Tabu Search [12], and PSO [13] [18].

Although the use of search techniques automatize the parameter selection process, this solution may still be very expensive since for each configuration being evaluated during the search it is necessary to train the SVM [1]. The impact of this limitation can be even more drastic depending on the problem at hand and the number of parameters to be optimized.

ML is an alternative that has been studied in recent years to SVM parameter selection [1][15][11][16][17][5]. In this approach, the choice of parameters for a problem is based on well-succeeded parameters adopted to previous similar problems. In ML, it is necessary to maintain a set of meta-examples where each meta-example stores: (1) a set of features (called meta-features) describing a learning problem; and (2) evaluations of a set of candidate parameters on the problem. A meta-learner is then used to acquire knowledge from a set of such meta-examples in order to recommend (or predict) the adequate configurations of parameters for new problems based on past problems' characteristics.

In this way, using ML, SVM parameters can be suggested for new problems without executing the SVM on each candidate configuration of parameters making this approach more economic in terms of computational cost. However, ML is very 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 ML for SVM parameter selection may be not so good as the performance of search techniques [6].

Thus, a recent work was performed combining ML with particle swarm optimization algorithms [18] in such a way that ML is used to recommend parameters which will be later refined by a search algorithm. This research handled the SVM parameter selection task as a single objective problem where the hybrid approach achieved good results for regression problems.

As it will be seen, in this study we propose a more complete analysis of the ML application to recommend parameters which will be later refined by search approaches applied for classification problems.

3 Developed Work

The work presented here proposes a comparison of benchmarks search techniques and hybrid search techniques (search techniques combined with ML) aiming to analyse the influence of suggestion of solutions in the search process. As context, we adopted the SVM model selection problem for classification.

Figure 1 depicts the general architecture [18] used to perform the combination of search techniques with ML. 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. 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 configuration of parameters will be the best one generated by the Search module up to its convergence or another stopping criteria.

In the current work, we implemented five benchmark particle swarm optimization algorithms to select two specific SVM parameters: 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 [19][20]. 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 of the induced SVMs [20]. As it will be seen, the prototypes were implemented to select the parameters (γ, C) for classification problems according one objective: success rate. Details of implementation will be presented in the next subsections.

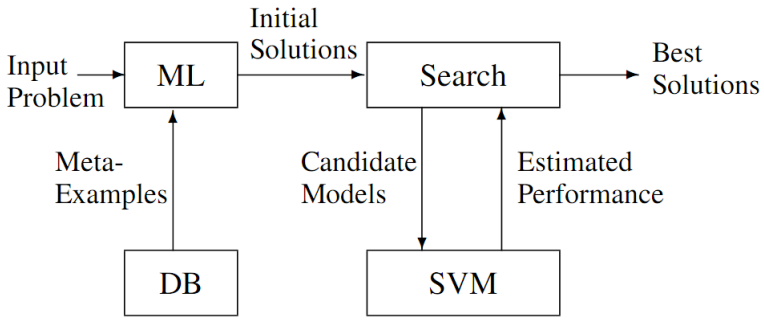

Fig. 1. Proposed Architecture.

3.1 Search Module

In our study, we implemented five benchmark search algorithms based on PSO: basic PSO, Inertia *Gbest*, Inertia *Lbest*, Constricted *Gbest* and Constricted *Lbest*. This subsection presents, initially, how the basic PSO works and after that we present its variations.

The main forms of PSO In our basic PSO implementation, each particle i represents a solution for a given problem, 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:

$$\begin{aligned} \boldsymbol{v}_i(t+1) = \boldsymbol{v_i}(t) + c_1 r_1(\boldsymbol{p}_i(t) - \boldsymbol{x_i}(t)) + \\ c_2 r_2(\boldsymbol{n}_i(t) - \boldsymbol{x_i}(t)), \end{aligned} \tag{1}$$

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i + \boldsymbol{v}_i(t+1). \tag{2}$$

In equation 1, $\boldsymbol{p}_i(t)$ is the best position achieved by the particle so far, and $\boldsymbol{n}_i(t)$ 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 c_1 and c_2 are positive accelerators which control the trade-off between exploring good global regions in the search space and refining the search in local regions around the particle. In equation 1, r_1 and r_2 are random numbers used to enhance the diversity of particle positions.

As it was mentioned, two parameters (c_1 and c_2) accelerate the convergence process, however, this acceleration can lead the swarm to an explosion state,

where the particles achieve high velocity values [21]. Thus, a few years after the initial PSO publications, a new parameter ω, called *inertia* weight, was introduced in an effort to strike a better balance. The velocity update equation was altered to the form:

$$\begin{aligned} \boldsymbol{v}_i(t+1) \;=\; \omega \boldsymbol{v_i}(t) \;+\; c_1 r_1(\boldsymbol{p}_i(t) \;-\; \boldsymbol{x_i}(t)) + \\ c_2 r_2(\boldsymbol{n}_i(t) \;-\; \boldsymbol{x_i}(t)). \end{aligned} \tag{3}$$

By adjusting the value of ω, the swarm has a greater tendency to eventually constrict itself down to the area containing the best fitness and explore that area in detail.

Another method of balancing global and local searches known as *constriction* was being explored simultaneously with the *inertia* weight method. Similar to the *inertia* weight method, this method introduced a new parameter χ, known as the *constriction* factor. χ is derived from the existing constants in the velocity update equation:

$$\chi = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}, \varphi = c_1 + c_2. \tag{4}$$

It was found that when $\varphi < 4$, the swarm would slowly "spiral" toward and around the best found solution in the search space with no guarantee of convergence, while for $\varphi > 4$ convergence would be quick and guaranteed.

This *constriction* factor is applied to the entire velocity update equation:

$$\begin{aligned} \boldsymbol{v}_i(t+1) \;= \chi(\boldsymbol{v_i}(t) \;+\; c_1 r_1(\boldsymbol{p}_i(t) \;-\; \boldsymbol{x_i}(t)) + \\ c_2 r_2(\boldsymbol{n}_i(t) \;-\; \boldsymbol{x_i}(t))). \end{aligned} \tag{5}$$

The effects are similar to those of *inertia* weight, resulting in swarm behaviour that is eventually limited to a small area of the feasible search space containing the best known solution. In our prototypes, we fixed PSO parameters using c_1 and $c_2 = 2.05$ and the $\omega = 0.8$.

Besides changes in the velocity structure the way the information is exchanged among the particles is crucial for the convergence. To disseminate information within a swarm is the key of any swarm intelligence based algorithm. PSO, like others swarm algorithms, make use of its own information exchange methods to distribute the best positions found during the algorithm execution [21] [22]. The way used by the swarm to distribute this information is the social structure formed by the particles. Variations in this structure can improve the algorithm performance.

Even when using different types of velocity update equations, the algorithm can work better by exploring the information exchange mechanism inside the swarm. This information exchange influences the particles in the velocity evaluation. The most common communication mechanisms between particles are *Star* and *Ring* topologies, shown in Figure 2.

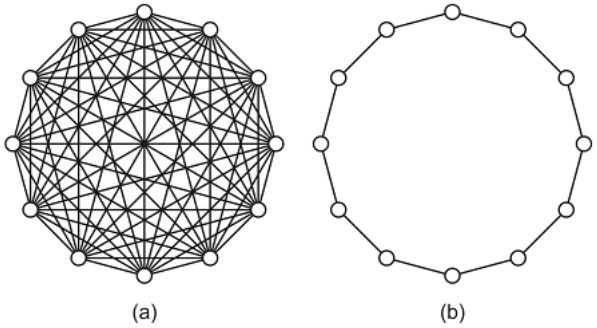

Fig. 2. Star and Ring topology.

Particles can share information globally through a fully-connected structure called star topology represented by Figure 2 a. This topology uses a global neighbourhood mechanism known as *Gbest* to share information. By using *Gbest*, particles can spread information quickly through the swarm. It is analogous to a large community where all decisions taken are instantaneously known by everyone. Each particle in this topology is attracted towards the best solution found by the entire swarm.

An information exchange mechanism based on local neighbourhood is known as *Lbest*. The particles only share information with their own neighbours. The structure used by this mechanism is ring topology and is illustrated in Figure 2 b. Different regions of the search space can be explored at the same time; however, the successful information from these regions takes a longer time to be sent to all other particles.

PSO to SVM Model Selection In the basic PSO and its variations, we adapted here to perform the search for configurations (γ, C). The objective function evaluates the success rate (SR) of each configuration of parameters, trying to maximize it, on a given classification problem.

In our work, the implemented search techniques performed 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 [20], 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 x 21 = 399 different combinations of parameters in the search space.

3.2 Meta-Database

In order to generate meta-examples, we collected 40 datasets corresponding to 40 different classification problems, available in the WEKA project [23] and UCI

Repository [24]. Each meta-example is related to a single classification problem and stores: (1) a vector of meta-features describing the problem; and (2) the performance grid which stores the success rate obtained by the SVM in the search space of configurations (γ, C). The performance grid consists of 399 different settings of parameters γ and C.

Meta-Features In the developed work, a total number of 8 meta-features was used to describe the datasets of classification problems. These meta-features were based on the set of features defined in [25], which proposed 16 meta-features for classification problems. However, according the restrictions (numerical, no missing values, no binary attributes) of the selected databases, we adopted the meta-features listed in Table 1, which is divided in 3 parts: Simple, Statistical and Information Theory meta-features.

Table 1. Meta-Features for Classification Problems.

Simple
Number of examples
Number of attributes
Number of classes
Statistical
Mean correlation of attributes
Skewness
Kurtosis
Geometric mean of attributes
Information Theory
Entropy of class

Values as *number of examples*, *attributes* and *classes* are data already discriminated in databases, being considered simple data. The group of statistical values is composed by the *mean correlation of attributes*; *Skewness*, which measure the asymmetry of the distribution regarding the central axis [25]; *Kurtosis*, which measure the dispersion (characterized by the flatness of the distribution curve) [25] and the *geometric mean of the attributes* which evaluates the mean of the data standard deviation. Ultimately, the information theory group measure the randomness of the instances; being composed by the *Entropy* which defines the degree of uncertainty of classification [25][26].

Performance Grid The performance grid stores the success rate obtained by the SVM on a problem considering different SVM configurations. For each of the 399 configurations, a 10-fold cross validation experiment was performed to collect SVM performance. The obtained 399 objective values were stored in the

performance grid. In these experiments, we deployed the Scikits Learn library [20] to implement the SVMs and to perform the cross-validation experiments.

We highlight here that the performance grid is equivalent to the search space explored by PSO. By generating a performance 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.

3.3 Meta-Learner

Given a new input problem described by the vector $\boldsymbol{x} = (x_1, ..., x_p)$, the Meta-Learner selects the k most similar problems according to the distance between the meta-attributes. We applied this method to make the initial population more diverse (retrieving good solutions from different similar problems) to avoid suggesting local maximum regions. The distance function (dist) implemented was the Euclidean Distance, defined as:

$$dist(vec_i, vec_l) = \sum_{j=1}^{p} \sqrt{(i_j - l_j)^2}. \quad (6)$$

For each retrieved meta-example, the meta-learner selects in the performance grid the configuration of parameters (among the 399 candidates) that obtained the highest SR value, i.e. the best SVM configuration. Hence, the meta-learner will suggest as initial PSO population the set of k best configurations selected in the performance grids of the retrieved meta-examples.

4 Experiments

In this work, we realized two experiments: 1) comparison of the benchmark algorithms' performance and 2) comparison between the benchmark algorithm which achieved the best performance with the hybrid algorithm.

In the first experiment, we executed each one of the five benchmark approaches using random initialization, computing the mean of the success rate values for each generation of all problems. The output of this experiment is the mean curve, among all problems, representing the performance along the generations.

In the second experiment, to realize the comparison between the best benchmark algorithm with the hybrid algorithm, it was necessary to execute the hybrid technique 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. Initially, a number of k configurations were suggested by the ML module as the initial PSO population (in our experiments, we adopted $k = 7$). 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 highest SR value obtained so far (i.e. the best fitness). Hence, for each problem left out a curve of N values of success rate was generated aiming to analyse the search progress on the problem. Finally, the curves of SR 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 classification problems considered.

After executing both, best benchmark algorithm and hybrid algorithm, we analysed their performance along the generations and, moreover, evaluated the number of wins of the algorithms per generation regarding all problems, where the algorithm which achieved better performance, according an specific problem, is the winner.

Finally, we highlight that each evaluated version of PSO was executed 30 times and the average results were recorded.

4.1 Results

Initially, we analysed the success rate curve of all PSO approaches using random initialization, and selected the technique which achieved the best performance to be used as basis of comparison with its own version, but, using ML suggestions as initial population.

Figure 3 shows the mean performance curve of all benchmark search techniques along the generations. As we can see, the basic PSO performance, after the fourth generation, outperformed the other techniques. As we adopt the policy of selecting the k most similar problems to augment the diversity, the basic PSO exploration converged faster with 10 generations. For the same reason, the other techniques which used the *Gbest* mechanism achieved better results than the approaches that used *Lbest* mechanism. The approaches using *Ring* topology were overcome by the other techniques since its policy of information exchange is based on exploitation, reducing the velocity of convergence.

As it was discussed above, the approach which obtained the best results was the basic PSO. So, in order to analyse the influence of ML on search approaches performance, we compared the performance of the basic PSO with basic PSO using ML (hybrid PSO). Figure 4 makes a comparison between the basic PSO and the basic PSO using ML performance. As it can be seen, the combination of the basic PSO with ML makes the search process start in well succeed regions, for this reason in the first iteration the Hybrid PSO presented a higher performance than basic PSO. Along the generations, while the basic PSO sought by good regions, the hybrid PSO refined the found solutions augmenting the convergence.

The analysis realized above was about the mean of the success rate values for each generation of all problems. The next analysis intend to count the number of classification problems that each algorithm won per iteration, shown in Figure 5.

As it can be seen, the hybrid PSO won the basic PSO in most of the classification problems in all generations. It achieved a mean of victories of 78.5% considering all generations. Possibly, the problems the hybrid approach lost for the basic PSO are classification problems with few or none similar problems.

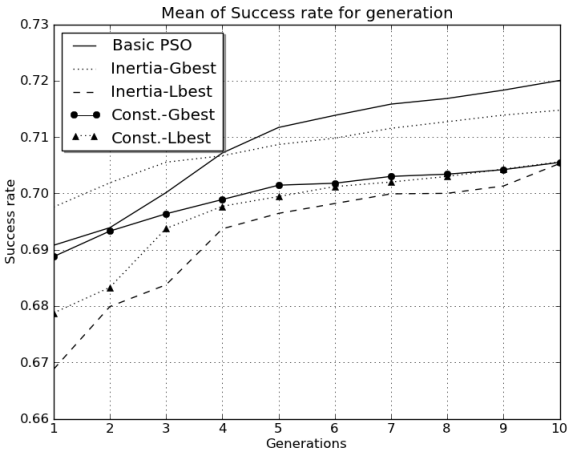

Fig. 3. Success rate curve of all PSO approaches using random initialization with $k = 7$.

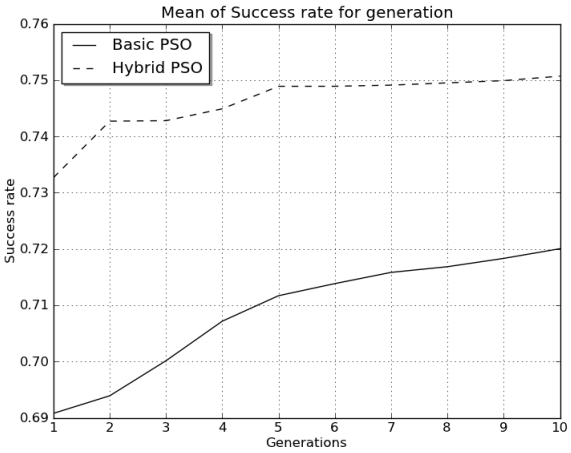

Fig. 4. Success rate curve of the basic PSO and the hybrid PSO with $k = 7$.

So, the suggestion of solutions performed by the ML can be impaired impacting negatively in the search process.

Although, the hybrid approach has achieved clearly better results than the basic PSO, a visual representation is not enough to conclude which approach is considered better. In this way, we realized statistical tests to guarantee our experiments.

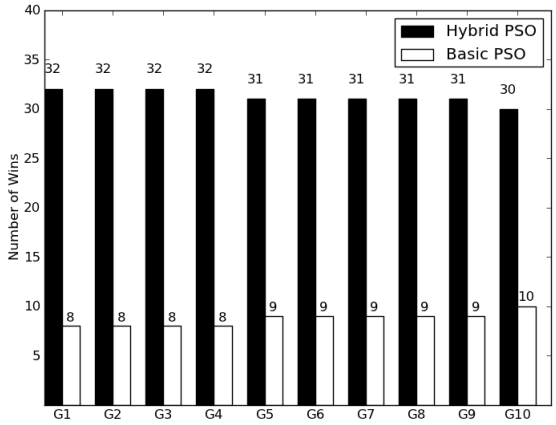

Fig. 5. Number of wins regarding success rate using $k = 7$ for 10 generations.

4.2 Statistical Analysis

Besides the analysis presented in last section, we realized two statistical tests: normality and hypothesis test; in order to evaluate which distribution, presented by the basic and hybrid PSO, is considered better. The Anderson-Darling method was used to test the data's normality using 5% of significance. After verifying and assuring the non-normality of the data, we applied the Wilcoxon test, a non-parametric test, to verify the hypothesis.

To perform this test, the hybrid PSO sample is presented as μ_1 and basic PSO sample as μ_2, using the following hypothesis:

$$H_0 : \mu_1 = \mu_2$$

$$H_a : \mu 1 > \mu_2.$$

The null hypothesis is that both distributions are equal, represented by H_0; and the alternative one is that the results of μ_1 are better than μ_2, represented by H_a; using 5% of significance. If the $p - value$ assumes a value less than 5%, the null hypothesis is rejected and the alternative one accepted.

After applying the Wilcoxon test, it returned $p - value = 0.00578$. As the $p - value$ is less than 5%, the null hypothesis is rejected. Hence, we conclude μ_1 (hybrid PSO results) achieved better solutions, per iteration, in comparison to μ_2 (basic PSO results), with 95% of reliability.

5 Conclusions

In the current work, we analysed the influence of meta-learning in the search process of particle swarm algorithms for the problem of SVM parameter selection. To perform the experimental study, we implemented the main particle swarm

approaches to select the parameter γ of the RBF kernel and the regularization parameter C. In our implementation, a number of 40 classification problems was used to generate meta-examples. In the performed experiments, we verified that the combination of meta-learning with particle swarm algorithms overcame all the benchmark results with 95% of reliability.

In future work, we intend to augment the number of meta-examples as we believe that the performance of the hybrid approaches can be improved as more meta-examples are considered. Also, other variations of search techniques can be considered in the future implementations.

References

1. C. Soares, P. Brazdil, and P. Kuba. A meta-learning approach to select the kernel width in support vector regression. (4):195–209, 2000.
2. N. Cristianini and 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, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 131–159, 2002.
4. N. Cristianini, C. Campbell, and J. Shawe-Taylor. Dynamically adapting kernels in support vector machines. *NIPS*, 204–210, 1998.
5. S. Ali and K. Smith-Miles. On optimal degree selection for polynomial kernel with support vector machines: Theoretical and empirical investigations. *KES Journal*, 1(1):1–18, 2007.
6. G. Narzisi. An Experimental Multi-Objective Study of the SVM Model Selection problem.
7. S. Lessmann, R. Stahlbock, and S. Crone. Genetic algorithms for support vector machine model selection. *International Joint Conference on Neural Networks*, 3063–3069, 2006.
8. F. Friedrichs and C. Igel. Evolutionary tuning of multiple svm parameters. *Neurocomputing*, 2005.
9. A. Lorena and A. de Carvalho. Evolutionary tuning of svm parameter values in multiclass problems. *Neurocomputing*, pages 16–18.
10. S.S. Keerthi. Efficient tuning of svm hyperparameters using radius/margin bound and iterative algorithms. IEEE Transactions on Neural Networks, 2002.
11. P. Kuba, P. Brazdil, C. Soares, and A. Woznica. Exploiting sampling and meta-learning for parameter setting support vector machines. *Proceedings of the IBERAMIA*, pages 217–225, 2001.
12. G. Cawley. Model selection for support vector machines via adaptive step-size tabu search. *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms*, 28(8):434–437, 2001.
13. B. de Souza, A. de Carvalho, and R. Ishii. Multiclass svm model selection using particle swarm optimization. *Sixth International Conference on Hybrid Intelligent Systems*, 2006.
14. T. Glasmachers and C. Igel. Gradient-based adaptation of general gaussian kernels. Neural Comput., 2005.
15. S. Ali and K. A. Smith-Miles. A meta-learning approach to automatic kernel selection for support vector machines. *Neurocomputing*, 173–186, 2006.
16. C. Soares and P. Brazdil. Selecting parameters of svm using meta-learning and kernel matrix-based meta-features. *SAC*, 2006.

17. S. Ali and K. A. Smith. Matching svm kernel's suitability to data characteristics using tree by fuzzy c-means clustering. *Third International Conference on Hybrid Intelligent Systems*, 2010.
18. T. Gomes, R. B. C. Prudencio, C. Soares, A. Rossi, A. Carvalho. Combining meta-learning and search techniques to svm parameter selection, in: Brazilian Symposium on Neural Networks, 2010, pp. 7984.
19. S. S. Keerthi and C.-J. Lin. Asymptotic behaviors of support vector machines with gaussian kernel. In Tomasz Imielinski and Hank Korth, editors, *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.
20. C.-W. Hsu, C.-C. Chang, and C.-J. Lin. Scikit-Learn: A Toolkit of Machine Learning in Python, Available: http://scikit-learn.org.
21. A. P. Engelbrecht. Fundamentals of Computational Swarm Intelligence. In John Wiley Sons, 2005.
22. A. M. Sutton, D. Whitley, M. Lunacek, and A. Howe. PSO and multi-funnel landscapes: how cooperation might limit exploration. in GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation, vol. 1. Seattle, Washington, USA: ACM Press, 8-12 Jul. 2006, pp. 7582.
23. WEKA, The University of Waikato, Available: http://www.cs.waikato.ac.nz/ml/weka/.
24. UCI Machine Learning Repository, Available: http://archive.ics.uci.edu/ml/.
25. Kate A. Smith-Miles. Cross-Disciplinary Perspectives on Meta-Learning for Algorithm Selection, 2005.
26. Ricardo B.C. Prudêncio and Teresa B. Ludermir. Selective generation of training examples in active meta-learning. *International Journal of Hybrid Intelligent Systems*, 2008.