Combining Meta-Learning and Search Techniques to SVM Parameter Selection

Taciana A. F. Gomes, Ricardo B. C. Prudêncio
Centro de Informática, Universidade Federal de Pernambuco, Brazil
{tafg, rbcp}@cin.ufpe.br

Carlos Soares
LIAAD-INESC Porto LA, Faculdade de Economia
Universidade do Porto, Portugal
csoares@fep.up.pt

André L. D. Rossi, André Carvalho
Depto. Ciências da Computação
Universidade de São Paulo, Brazil
{alrossi, andre}@icmc.usp.br

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 a number of parameters, including for instance the kernel and the regularization parameters. In the current work, we propose the combination of Meta-Learning and search techniques to the problem of SVM parameter selection. Given an input problem, Meta-Learning is used to recommend SVM parameters based on well-succeeded parameters adopted in previous similar problems. The parameters returned by Meta-Learning are then used as initial search points to a search technique which will perform a further exploration of the parameter space. In this combination, we envisioned that the initial solutions provided by Meta-Learning are located in good regions in the search space (i.e. they are closer to the optimum solutions). Hence, the search technique would need to evaluate a lower number of candidate search points in order to find an adequate solution. In our work, we implemented a prototype in which Particle Swarm Optimization (PSO) was used to select the values of two SVM parameters for regression problems. In the performed experiments, the proposed solution was compared to a PSO with random initialization, obtaining better average results on a set of 40 regression problems.

1 Introduction

An increasing attention has been given to Support Vector Machines (SVMs) due to both the theoretical foundations of SVMs 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 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].

SVM parameter selection is commonly treated by different authors as an optimization problem in which a search technique is used 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 configurations of parameters is 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 later refined by the search technique. In previous work the search process starts evaluating solutions randomly sampled from the parameter space (e.g., [6, 7, 8]). In the proposed hybrid method in turn the

search process is started from solutions which were well-succeeded in 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 implemented a prototype 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 [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 20 meta-features proposed in [10, 1, 11]. In the implemented prototype, the Particle Swarm Optimization (PSO) algorithm [12] was used as the search technique to optimize the parameters (γ, C). In our experiments, we evaluated the PSO in two different versions: (1) PSO with initial population suggested by Meta-Learning and (2) PSO with random initial population. The experiments' results revealed that the hybrid method was able to converge more quickly to good solutions when compared to the randomly initialized PSO.

Section 2 brings a brief presentation on the SVM model selection. Section 3 presents details of the proposed work, as well as the case study. Section 4 describes the experiments and obtained results. Finally, Section 5 presents some conclusions and the future work.

2 SVM Parameter Selection

According to [13], 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 estimated using the problem's dataset. In order to automatize this process and to avoid an exhaustive or a random exploration of parameters, different authors have deployed search and optimization techniques [3, 4, 6, 7, 8, 13, 14, 15]. In this context, the search space consists on a set of possible configurations of parameters and the objective function corresponds to a performance measure (e.g., precision estimated by cross-validation) obtained by the SVM on the problem. Different search techniques were adopted in this approach, including gradient-based techniques [3], Evolutionary Algorithms [6, 7, 8], Tabu Search [13] and Particle Swarm Optimization [15].

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 being evaluated 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.

Alternatively, Meta-Learning has been proposed and investigated in recent years to SVM parameter selection [1, 5, 10, 11, 16, 17]. In this approach, the choice of parameters for a problem is based on well-succeeded parameters adopted to previous similar problems. In Meta-Learning, 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) the empirical evaluation 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 the problems' characteristics. In this sense, Meta-Learning is a more economic approach in terms of computational cost since SVM models can be suggested for new problems without executing the SVM on each candidate configuration of parameters.

As it will be seen, in the current work we propose to use Meta-Learning to recommend parameters which will be later refined by a search technique.

3 Proposed Solution

As discussed in the previous section, SVMs have a strong generalization power, however the performance of these algorithms depends on an adequate choice of their parameters. The work presented here proposes a new method to automate the design of SVMs based on the combination of Meta-Learning and search techniques. In the proposal, a meta-learner suggests the initial search points from well-succeeded parameters to problems which are similar to the current one. As discussed in [18], good solutions to a particular search problem can be used to indicate promising regions of the search space for similar problems. Hence, we expected that the initialization provided by Meta-Learning enables the search technique to speed up its convergence to good solutions.

Figure 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 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 a prototype 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 [9, 19]. It is known that the γ pa-

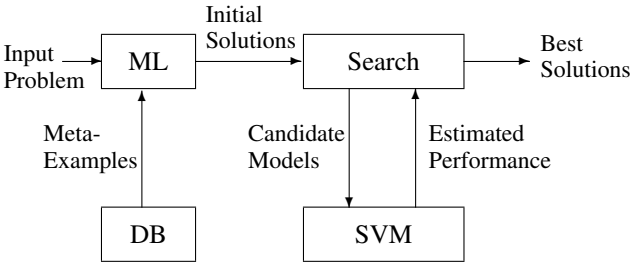

Figure 1. General Architecture

rameter 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 [19].

As it will be seen, the current prototype was implemented to select the parameters (γ, C) for regression problems. Details of implementation will be presented in the next subsections.

3.1 Search Module

In our prototype, we implemented the version of the Particle Swarm Optimization (PSO) algorithm originally proposed in [20] 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 i represents a configuration $s_i = (\gamma, C)$, 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. 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:

$$v_i \leftarrow \omega v_i + c_1 r_1(\hat{s}_i - s_i) + c_2 r_2(\hat{g} - s_i) \quad (1)$$

$$s_i \leftarrow s_i + v_i \quad (2)$$

In Equation 1, $\hat{s}_i$ is the best position achieved by the particle so far, and $\hat{g}_i$ 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 equation 1, r_1 and r_2 are random numbers used to enhance the diversity of particle positions. In our prototype, we fixed PSO parameters using ω=0.8, c_1=2 and c_2=2.

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 [19], 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 Database

In order to generate meta-examples, we collected 40 datasets corresponding to 40 different regression problems, available in the WEKA project[1]. Each meta-example is related to a single regression problem and stores: (1) a vector of *meta-features* describing the problem; and (2) the *performance grid* which stores the performance obtained by the SVM in the search space of configurations (γ, C).

3.2.1 Meta-Features

In the developed work, a total number of 20 meta-features was used to describe the datasets of regression problems. A number of 17 meta-features were based on the set of features defined in [10], corresponding to descritive measures of the regression datasets (see Table 1). The remaining 3 meta-features were defined by [1], corresponding to features computed from the kernel matrix (see Table 2). Both sets of meta-features have shown to be useful for model selection purposes (see [10] and [1]). In our work, we combined these sets in order to provide a more complete description of the regression problems.

3.2.2 Performance Grid

The performance grid stores the empirical performance obtained by the SVM on a problem considering different SVM configurations. The performance grid was built considering the same 399 settings of configurations defined in PSO

[1]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. Meta-features proposed by [10] for regression problems

Number of examples
Number of attributes
Ratio of the number of examples to the number of attributes
Correlation matrix between attributes and target
Correlation between continuous attributes
Ratio of the standard-deviation to the standard-deviation of alpha trimmed mean.
Number of continuous attributes with outliers
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 3 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 attributes
Average absolute correlation of numeric attributes to the target
Average dispersion gain

Table 2. Meta-features proposed by [1] for regression problems

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

search space in Section 3.1. For each of the 399 configurations, a 10-fold cross validation experiment was performed to evaluate SVM performance. The obtained 399 NMSE values were stored in the performance grid. In these experiments, we deployed the LIBSVM library [21] 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 $\mathbf{x} = (x^1, \ldots, x^p)$, the Meta-Learner retrieves the k most similar meta-examples from the database, according to the distance between meta-attributes. The distance function (*dist*) implemented in the prototype was the unweighted L_1-Norm, defined as:

$$dist(\mathbf{x}, \mathbf{x}_i) = \sum_{j=1}^{p} \frac{|x^j - x_i^j|}{max_i(x_i^j) - min_i(x_i^j)} \qquad (3)$$

For each retrieved meta-example, the meta-learner selects in the performance grid the configuration of parameters (among the 399 candidates) that obtained the lowest NMSE 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 section, we present the experiments which evaluated the proposed solution on the set of 40 regression problems considered in our work. The proposed solution was evaluated by following a leave-one-out methodology described bellow.

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 (see Section 3.3) (in our experiments, we adopted k = 3, 5 and 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 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 for each value of k but using a randomly initialized population for PSO. Despite its simplicity, the random initialization has the advantage of performing a uniform initial exploration of the search space. Finally, we highlight that each evaluated version of PSO (randomly initialized PSO and the hybrid solution) was executed 100 times and the average results were recorded.

Figures 2, 3 and 4 show the average NMSE curves (over 100 runs) obtained by the two evaluated methods for different sizes of the PSO population (k= 3, 5 and 7 respectively). As it can be seen, in all experiments the Meta-Learner was able to suggest an initial population of models with lower NMSE values compared to the values obtained by a random initial population. For k=3 and k=5 (see figures 2 and 3), the NMSE values obtained by the randomly initialized PSO

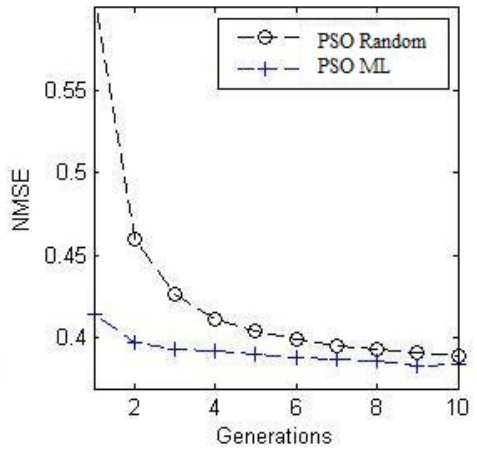

Figure 2. Average NMSE results obtained by PSO using k=3 particles

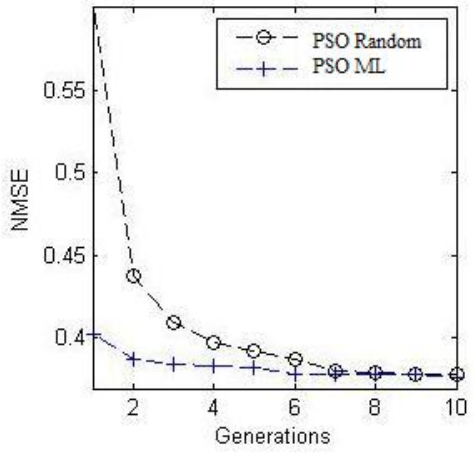

Figure 3. Average NMSE results obtained by PSO using k=5 particles

were equivalent to the initial values obtained by the hybrid solution only after four generations. For k=7, the results of the random PSO was equivalent to the results obtained by the hybrid solution on its first population only after two generations (see figure 4). The hybrid proposed solution started its search process in better regions 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 found solutions.

5 Conclusion

In the current work, we combined Meta-Learning and search techniques to the problem of SVM parameter selection. A prototype was implemented in which PSO was adopted as the search technique to select the parameter γ of the RBF kernel and the regularization parameter C. In our implementation, a number of 40 regression problems was used to generate meta-examples. In the performed experiments, we observed that the proposed approach was able to find adequate parameters in a lower number of iterations compared to a randomly initialized PSO.

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. Also, 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 can suppose 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 in the 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, and P. Kuba, "A meta-learning approach to select the kernel width in support vector regression," *Machine Learning*, vol. 54, no. 3, pp. 195–209, 2004.

[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*, vol. 46, no. 1, pp. 131–159, 2002.

[4] N. Cristianini, C. Campbell, and J. Shawe-Taylor, "Dynamically adapting kernels in support vector machines," in *NIPS*, 1998, pp. 204–210.

[5] S. Ali and K. A. Smith-Miles, "A meta-learning approach to automatic kernel selection for support vector machines," *Neurocomputing*, vol. 70, no. 1-3, pp. 173–186, 2006.

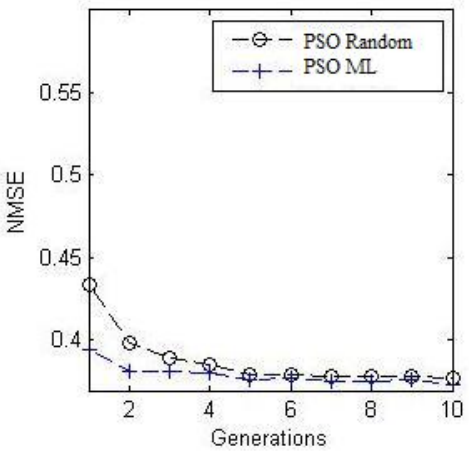

Figure 4. Average NMSE results obtained by PSO using k=7 particles

[6] S. Lessmann, R. Stahlbock, and S. Crone, "Genetic algorithms for support vector machine model selection," in *International Joint Conference on Neural Networks*, 2006, pp. 3063–3069.

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

[8] A. Lorena and A. de Carvalho, "Evolutionary tuning of svm parameter values in multiclass problems," *Neurocomputing*, vol. 71, pp. 16–18, 2008.

[9] S. S. Keerthi and C.-J. Lin, "Asymptotic behaviors of support vector machines with gaussian kernel," *Neural Computation*, vol. 15, no. 7, pp. 1667–1689, 2003.

[10] P. Kuba, P. Brazdil, C. Soares, and 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 and P. Brazdil, "Selecting parameters of svm using meta-learning and kernel matrix-based meta-features," in *SAC*, 2006, pp. 564–568.

[12] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, 1995, pp. 1942–1948.

[13] G. Cawley, "Model selection for support vector machines via adaptive step-size tabu search," in *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms*, 2001, pp. 434–437.

[14] X. Guo, J. Yang, C. Wu, C. Wang, and Y. Liang, "A novel ls-svms hyper-parameter selection based on particle swarm optimization," *Neurocomputing*, vol. 71, pp. 3211–3215, 2008.

[15] B. de Souza, A. de Carvalho, and R. Ishii, "Multiclass svm model selection using particle swarm optimization," in *Sixth International Conference on Hybrid Intelligent Systems*, 2006, pp. 441–446.

[16] S. Ali and 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.

[17] S. Ali and K. Smith-Miles, "On optimal degree selection for polynomial kernel with support vector machines: Theoretical and empirical investigations," *KES Journal*, vol. 11, no. 1, pp. 1–18, 2007.

[18] S. Louis and J. McDonnell, "Learning with case-injected genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 4, pp. 316–328, 2004.

[19] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," Department of Computer Science and Information Engineering, National Taiwan University, Taipei," Technical report, 2003. [Online]. Available: http://www.csie.ntu.edu.tw/ cjlin/libsvm/

[20] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *IEEE World Congress on Computational Intelligence*, 1998, pp. 69–73.

[21] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.