A PSO-Based Framework for Dynamic SVM Model Selection

Marcelo N. Kapp
École de technologie supérieure
Université du Québec
1100 rue Notre-Dame ouest
Montreal, Canada
kapp@livia.etsmtl.ca

Robert Sabourin
École de technologie supérieure
Université du Québec
1100 rue Notre-Dame ouest
Montreal, Canada
robert.sabourin@etsmtl.ca

Patrick Maupin
Defence Research and Development Canada
DRDC Valcartier
2459 Pie XI Blvd North
Val-Bélair, Canada
patrick.maupin@drdc-rddc.gc.ca

ABSTRACT

Support Vector Machines (SVM) are very powerful classifiers in theory but their efficiency in practice rely on an optimal selection of hyper-parameters. A naïve or *ad hoc* choice of values for the latter can lead to poor performance in terms of generalization error and high complexity of parameterized models obtained in terms of the number of support vectors identified. This hyper-parameter estimation with respect to the aforementioned performance measures is often called the model selection problem in the SVM research community. In this paper we propose a strategy to select optimal SVM models in a dynamic fashion in order to attend that when knowledge about the environment is updated with new observations and previously parameterized models need to be re-evaluated, and in some cases discarded in favour of revised models. This strategy combines the power of the swarm intelligence theory with the conventional grid-search method in order to progressively identify and sort out potential solutions using dynamically updated training datasets. Experimental results demonstrate that the proposed method outperforms the traditional approaches tested against it while saving considerable computational time.

Categories and Subject Descriptors

I.5.2 [**Pattern Recognition**]: Design Methodology—*Classifier design and evaluation*

General Terms

Experimentation

Keywords

Particle Swarm Optimization, Dynamic Optimization, Support Vector Machines, Model Selection

GECCO'09, July 8–12, 2009, Montréal Québec, Canada.

1. INTRODUCTION

Support Vector Machines (SVM) [21] are very powerful classifiers in theory but their efficiency in practice rely on an optimal selection of hyper-parameters. A naïve or *ad hoc* choice of values for the latter can lead to poor performance in terms of generalization error and high complexity of parameterized models obtained in terms of the number of support vectors identified. This hyper-parameter estimation with respect to the aforementioned performance measures is the so-called model selection problem in the SVM research community.

Over the last years, many model selection approaches have been proposed in the literature. They differ basically in two aspects: in the selection criterion and in the searching methods used. Some selection criterion are specifically related to the SVM formulation, such as: radius margin bound [21], span bound [5], and support vector count [21]. Others are classical ones, such as the well-known cross-validation and hold-out estimations. On the other hand, the most common searching methods applied are: the gradient descent [1], grid-search [4], or evolutionary techniques, such as genetic algorithms (GA) [6], covariance matrix adaptation evolution strategy (CMA-ES) [11], and more recently Particle Swarm Optimization (PSO) [10, 14]. Although some of these methods have practical implementations, e.g. gradient descent, their application is usually limited by hurdles in the model selection process. For instance, gradient descent methods require a differentiable objective function w.r.t. the hyper-parameters and the kernel, which also needs to be differentiable. In this same vein, multiple local minima in objective functions also represents a nightmare for gradient descent based methods. To overcome this, the application of grid-search or evolutionary techniques is very attractive. Unfortunately, concerning the grid-search method, a good discretization of the search space in fixed values is crucial to reach high performances. Thus, the choice of the objective function, the presence of local minima in the search space, and the computational time required for the model selection task have been considered main challenges in the field.

In addition to the typical parameter estimation difficulties briefly discussed above, updates about the knowledge available on the pattern recognition problem to solve represents a challenge too. This takes typically to form of data arriving in batches and thus available for training. In fact, training data quality and dynamics can affect the general

model selection process in different ways. For example, if the knowledge on the problem is limited, the data is noisy or is arriving in batches overtime the model selection task and its performance can be degraded along time. In order to avoid the negative effects of these uncertainties and updates in available training data, we believe that an efficient option is to allow on-line re-estimation of the current model's fitness and if required to allow the production of new classification model more suitable to both historical and new data. Thus, if the goal is to obtain a performing single classifier, the model selection process must be able to select dynamically optimal hyper-parameters and train new models from new samples added to existing batches. In this work we propose to study the general SVM model selection task as a dynamic optimization problem in a gradual learning context, where solution revisions are required on-line to either (1) improve models existing or train new models or simply decide if (2) actual or past models already trained can be used to classify incoming data. These considerations are pertinent specially for applications for which the acquisition of labeled data is expensive to obtain, e.g. cancer diagnosis, signature verification, etc., in which the data available may initially not be available in sufficient quantity to perform an efficient model selection. However, more data can be available overtime and new models can be generated gradually for improving performance. On the other hand, as previously mentioned, apart from the optimality of models estimated, the computational time spent to search for their parameter values is also a relevant factor. Most of related works in the literature have considered cases involving only a fixed amount of data to systems aimed at producing a single best solution. In these approaches whenever the training set is updated with more samples, the entire search process must re-start from scratch.

In this paper we propose a Particle Swarm Optimization based framework to select optimal models in a dynamic fashion. The general concept behind it is to treat the SVM model selection process as being a dynamic optimization problem, which can have multiple solutions, since its optimal hyper-parameter values can shift or not over the search space depending on the data available about the classification problem at a given instant. Thus, the proposed method can be also useful for real-world applications that require the generation of new classifiers dynamically in a serial way, e.g. those involving streaming data. The key idea is to obtain solutions dynamically over training datasets via three levels: re-evaluations of previous solutions, dynamic optimization processes, or even by keeping the previous best solution found so far. In this way, by shifting among these three levels, the method is able to provide systematically adapted solutions. We implement the proposed method based on three main principles: change detection, adapted grid-search, and swarm intelligence theory (for self-organization capability), where the goal is to solve the model selection by overcoming the constraints of the methods described above.

In addition, we try to answer the following questions: Is PSO really efficient to select optimal SVM models? Can the proposed method be more efficient than the traditional grid-search or even a PSO based strategy? Is it possible to obtain satisfactory results by spending less computational time than the application of PSO for each set of data? What is the impact in terms of classification errors, model complexities, and computational time for the most promising strategies? Experimental results demonstrate that our method can outperform the model selection approaches tested. Moreover, we show that for datasets with a fair amount of samples, the gradual application of the proposed method over sets of data can reach similar results to those obtained by optimization processes with PSO over all data, but saving considerable computational time.

This paper is organized as follows. In Section 2 we briefly describe the SVM classifier method. In Section 3 we explain the SVM model selection problem. The proposed method is introduced in Section 4. Finally, experimental results and conclusions are outlined in Sections 5 and 6, respectively.

2. SUPPORT VECTOR MACHINES

The Support Vector Machine (SVM) classifier is a machine learning approach based on the structural risk theory introduced by Vapnik in [21]. We can summarize the construction of a SVM classifier as follows. Consider a set of training labeled samples represented by $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, where $\mathbf{x}_i \in \Re^d$ denotes a d-dimensional vector in a space and $y_i \in \{-1, +1\}$ is the label associated to it. The SVM training process produces a linear decision boundary (optimal hyperplane) which separates two classes (-1 and +1). It is formulated by minimizing the training error while maximizing the separating margin. This optimization problem is usually solved through the Lagrange dual, which can be reformulated as:

$$\begin{array}{l} \max \quad \frac{1}{2}\sum_i^n \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j, \\ \text{subject to} \\ 0 \leq \alpha_i \leq C, \sum_i^n \alpha_i y_i = 0 \end{array} \quad (1)$$

Where C is a tradeoff parameter between error and margin, $(\alpha_i)_{i \in n}$ are lagrangian multipliers computed during the optimization for each training sample. This process selects a fraction l of training samples $\mathbf{x}_i$ that have $\alpha_i > 0$. These samples are called support vectors and are used to define the decision boundary. This SVM formulation works only for linearly separable classes. However, since real-world classification problems are hardly solved by a linear classifier, an extension is needed to nonlinear decision surfaces. To solve this, the dot products $(\mathbf{x}_i.\mathbf{x}_j)$ in the linear algorithm are replaced for a non-linear kernel function $K(.)$, where $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i).\Phi(\mathbf{x}_j)$ and Φ is a mapping function $\Phi : \Re^d \mapsto H$. Such a replacement is the so-called kernel trick, which enables the linear algorithm to map the data from the original input space $\Re^d$ to some different space H called feature space. In the feature space, non-linear SVMs can be generated since linear operations in the feature space are equivalent to non-linear operation in input space. In this paper, we use the RBF kernel $K(\mathbf{x}_i, \mathbf{x}_j) = exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$, therefore with the hyperparameter $\gamma > 0$ to be selected. Finally, the decision function derived by the SVM classifier for a test sample $\mathbf{x}$, training samples $\mathbf{x}_i$, and a bias term b can be computed as follows for a two-class problem:

$$sign(f(\mathbf{x})) \quad \text{with} \quad f(\mathbf{x}) = \sum_i^l \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \quad (2)$$

SVM is a powerful classifier but it needs a fine tuning of its hyper-parameters. However, finding the best values for such hyper-parameters is a non trivial and usually time consuming task. It is the so-called model selection problem.

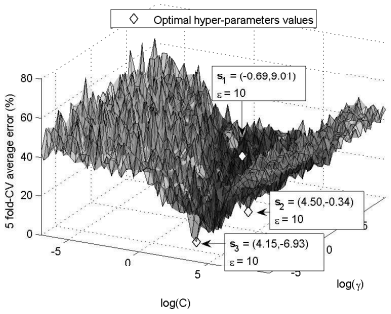
3. SVM MODEL SELECTION AND DYNAMIC OPTIMIZATION PROBLEMS

In order to generate highly performing SVM classifiers capable of dealing with continuous updates of training data an efficient model selection is required. The model selection task can be divided into two main phases: the searching phase and the final training/test phase. The searching phase involves solving an optimization problem whose goal is to find optimal values for the SVM hyper-parameters considered in this paper (C and γ) with respect to some preference, or selection criterion. In our case this criterion is expressed as an objective function $\mathcal{F}$ evaluated over a training dataset $\mathcal{D}$, in terms of the cross-validation error ϵ. Our model parameter selection problem takes thus the following form $min(\epsilon((C,\gamma),\mathcal{D}))$ or to simplify for the rest of the paper $min(\epsilon(\mathbf{s},\mathcal{D}))$. The final training/test phase is concerned with the production and evaluation on a test set of the final SVM model created based on the optimal hyper-parameters set found so far in the searching phase.

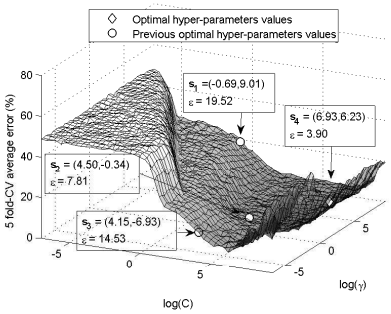
It is important to note that the model selection process depends on the data available at time k. In the type of situations considered in this paper, hyper-parameters must be re-estimated in order to retrain a SVM classifier from a dataset $\mathcal{D}$ updated at different instants k while the data is stored in a cumulative fashion, i.e. $\mathcal{D}(k)$ denotes the state of the dataset after merging previous collected data with new incomming data up to time k, as it occurs for applications where data collecting is expensive, such as cancer diagnosis, signature verification, etc. From this point of view, the optimal hyper-parameters for SVM model selection process can change dynamically with respect to the data available and used at time k.

Such dynamic optimization problems are complex problems whose optimal solution can change overtime in different ways [15], and require that the optimization algorithm be able of re-adapting past solutions to new environments. In particular, a dynamic environment concerning the optimization field can be categorized into three types, being: the location of the optimum changes over time and the amount of shift is quantified by a severity parameter (type I), the location remains fixed, but the value of the objective function changes (type II), and eventually both, the location and the value change (type III) [18]. Regarding the SVM model selection problem, we are dealing with a type III, since optimal hyper-parameters and respective objective function values can change depending on the data in both cases.

To demonstrate this fact, we depict a case study in Figure 1 regarding the second situation explained previously, i.e $min(\epsilon(\mathbf{s},\mathcal{D}(k)))$. Firstly, in Figure 1(a) one can see a SVM hyper-parameter search space and optimal solutions obtained with a certain number of data samples from a classification problem. Then, all the search space was recomputed with the same objective function (five-fold cross-validation average error), but now from more data. The resulting search space is shown in Figure 1(b). It can be seen that the search space and the optimal points actually may change depending on the amount of knowledge about the problem. It is truth concerning both objective function values, since the new objective values of previous optimal solutions $\mathbf{s}^*$ have changed from $\epsilon = 10\%$ to worst (e.g. $\mathbf{s}_1$ and $\mathbf{s}_3$) or to better (i.e. $\mathbf{s}_2$), and positions (hyper-parameters) since a new optimal solution emerged, $\mathbf{s}_4$=(6.93,6.23). The classification problem used for this case study is called P2 and it is introduced with other datasets in section 5.1. Therefore, the SVM model selection problem can be seen as a dynamic optimization problem when its goal is to produce solutions overtime. Because of this, it claims for sophisticated methods in which such dynamism be considered in order to adapt new solutions and save computational time, rather than for example, to start searching processes from scratch everytime.



(a) 40 samples



(b) 922 samples

Figure 1: Hyper-parameters search space for P2 problem with different number of samples.

4. THE PROPOSED DYNAMIC SVM MODEL SELECTION STRATEGY - DMS

In this work, the goal of the proposed method is to point out dynamically optimum solutions $\mathbf{s}^*(k)$ for dataset $\mathcal{D}(k)$ by switching among three levels: 1) use best solution $\mathbf{s}^*(k-1)$ found so far, 2) search for a new solution over an adapted grid composed of a set of solutions $\mathcal{S}(k-1)$, or 3) start a dynamic optimization process. In this paper, each solution $\mathbf{s}$ denotes the set of SVM hyper-parameters (C,γ). The final SVM model trained from a solution $\mathbf{s}^*(k)$ and dataset $\mathcal{D}(k)$ is denoted by $\mathcal{M}$. The switching among the levels is governed by change detection mechanisms that monitors novelties in the objective function $\mathcal{F}$. Such changes correspond to degradation of performance or no improvement at all (stability) with respect to new data, what will indicate that the system must act or not. We depict a general framework of the proposed method which is composed of three main modules in Figure 2: change detection, adapted grid-search, and dynamic particle swarm optimization (DPSO). We detail them below.

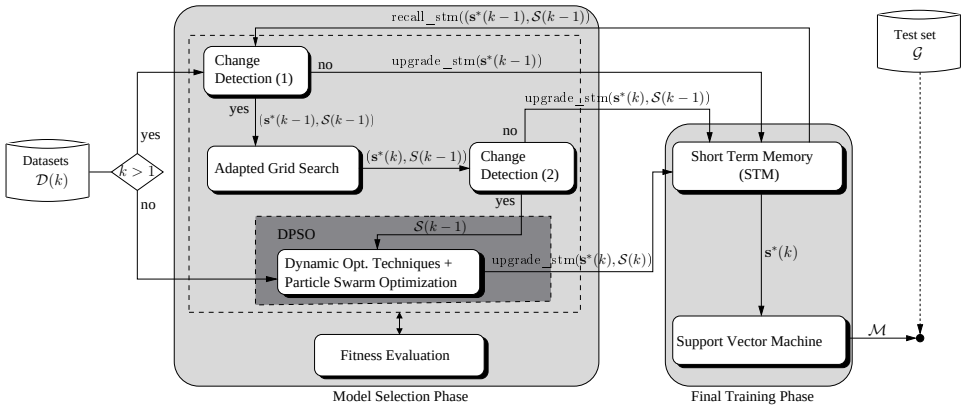


Figure 2: General framework of the proposed method for the dynamic SVM model selection.

4.1 Change Detection Module

The change detection module controls the intensity of the search process by pointing out how the solutions are found thereby the levels of the framework. In particular, it is responsible for at the same time monitoring the quality of the model selection process and avoiding "unnecessary" searching processes. We implement it by monitoring differences in the objective function values, in this case error estimations obtained for a best solution $\mathbf{s}^*(\cdot)$ on datasets $\mathcal{D}(\cdot)$, i.e. $\epsilon(\mathbf{s}^*(\cdot), \mathcal{D}(\cdot))$. In this work, as we are interested in finding performing solutions, we consider that if the objective function value computed gets worst or does not lie in a "stable" region, further searches are needed. The stable region is computed through the maximum expected difference δ_{max} between the objective functions values at the 90% confidence level using a normal approximation to the binomial distribution (see Equations 3 and 4) [9]. In this setting, if there is a degradation of performance, i.e. $(\epsilon(\mathbf{s}^*(k-1), \mathcal{D}(k-1)) < \epsilon(\mathbf{s}^*(k-1), \mathcal{D}(k)))$ or significant variation in the objective function (i.e. $|\epsilon(\mathbf{s}^*(k-1), \mathcal{D}(k-1)) - \epsilon(\mathbf{s}^*(k-1), \mathcal{D}(k))| \geq \delta max$), then other levels are activated for additional searches. In order to make this criterion more robust when small datasets are used, we combine it with a rule related to the compression capability of the classifier. The compression capability is calculated as the proportion of support vectors over the number of training samples. If the δ_{max} rule and a minimal compression required are attained, the situation is characterized as stable and no further searches are computed. Otherwise, the model selection process continues by activating the other modules.

$$\delta_{max} = z_{0.9} \times \sqrt{\sigma} = 1.282 \times \sqrt{\sigma} \tag{3}$$

Where σ is computed by, where $W(\cdot)$ is the dataset size:

$$\sigma = \frac{\epsilon(\mathbf{s}^*, \mathcal{D}(k-1)) \times (1 - \epsilon(\mathbf{s}^*, \mathcal{D}(k-1)))}{W(\mathcal{D}(k-1))} + \frac{\epsilon(\mathbf{s}^*, \mathcal{D}(k)) \times (1 - \epsilon(\mathbf{s}^*, \mathcal{D}(k)))}{W(\mathcal{D}(k))} \tag{4}$$

Thus, the change detection module may sometimes denote a trade-off controller between computational time spent and quality of solutions. For instance, if we ignore this module, then dynamic re-optimization processes will be always conducted, what can produce either satisfactory results but being unnecessarily time consuming for stable cases.

4.2 Adapted Grid-Search

The adapted grid search module provides optimum solutions by re-evaluating the knowledge acquired from previous optimizations performed by the DPSO module. Basically, this module uses the best positions of preceding optimized solutions as a grid of new possible candidate solutions to be evaluated over the current data $\mathcal{D}(k)$. At the end of the process, the best candidate is selected. Usually, this method finds better solutions than the traditional grid-search method, since unlike this latter, the adapted grid-search module reduces the number of trials by focusing the search in an optimal region. As a result, this module can save a considerable computational time. If the solution pointed is not satisfactory for final learning purposes, then the dynamic optimization module is activated as indicated in the framework in Figure 2.

4.3 Dynamic Particle Swarm Optimization - DPSO

The DPSO module is responsible for finding new solutions by re-optimization processes. We implement it based on the Particle Swarm Optimization (PSO) algorithm combined with dynamic optimization techniques.

The PSO method was firstly introduced by Kennedy and Eberhart in 1995 [17]. In short, it is a population-based optimization technique inspired by the social behavior of bird flocking or fish schooling. We use PSO because it has a continuous codification, what is ideal for search of optimal SVM hyper-parameters. Moreover, the potential for adaptive control and flexibility that the swarm intelligence provides (e.g. self-organization and division of labor), is very interesting to be explored for solving dynamic optimization problems. In this section, we simplify the index notation (e.g. time or datasets) and use only those needed to the PSO technique.

PSO involves a set $\mathcal{S} = \{\mathbf{s}_i, \dot{s}_i, \mathbf{s}'_i\}_{i=1}^{P}$[1] of particles that fly in the search space looking for an optimal point in a given d-dimensional solution space. The $\mathbf{s}_i = (s_i^1, s_i^2, \ldots, s_i^d)$ which is a vector that contains the set of values of the current hypothesis. It represents the current location of the particle

[1]We use this functional notation for sake of generality, the equivalent to traditional PSO would be: $\mathcal{S} = \{\mathbf{x}_i, \mathbf{v}_i, \mathbf{p}_i\}_{i=1}^{P}$

in the solution space, where the number of dimensions is problem dependent. The vector $\dot{s}_i = (\dot{s}_i^1, \dot{s}_i^2, \ldots, \dot{s}_i^d)$ which stores the velocities for each dimension of the vector $\mathbf{s}_i$. The velocities are responsible for changing the direction of the particle. The vector $\mathbf{s}'_i = (s_i'^1, s_i'^2, \ldots, s_i'^d)$ is a copy of the vector $\mathbf{s}_i$ which produced the particle's individual best fitness. Together, $\mathbf{s}'_i$ and $\mathbf{s}_i$ represent the particles' memories. Regarding the model selection problem, the vector positions $\mathbf{s}_i$ encodes the SVM hyper-parameter set to be optimized and $\mathbf{s}^*$ denotes the best solution found.

PSO starts the search process by initializing the particles' positions randomly over the search space. Then, it searches for optimal solutions iteratively by updating them to fly through a multidimensional search space by following the current optimum particles. The direction of the particle's movement is governed by the velocity vector $\dot{s}_i$, which is denoted by the sum between the information from the best particle's informant found in its neighborhood (e.g. $s'_{i,net(i,\lambda)}(q)$, regarding a *lbest* topology of size λ [16]) with the particle's own experience $\mathbf{s}'_i$. For a new iteration $q+1$ and dimension d, the update is computed as follows:

$$\dot{s}_i^d(q+1) = \chi(\dot{s}_i^d(q) + \phi r_1(s_i'^d(q) - s_i^d(q)) + \phi r_2(s'^d_{net(i,\lambda)}(q) - s_i^d(q))) \quad (5)$$

where χ is the constriction coefficient introduced by Clerc [8], and r_1 and r_2 are random values. A constriction coefficient value of $\chi = 0.7298$ and $\phi = 2.05$ are recommended [16]. Eventually the trajectory of a particle is updated by the sum of its updated velocity vector $\dot{s}_i(q+1)$ to its current position vector $\mathbf{s}_i(q)$ to obtain a new location, as depicted in Equation 6. Therefore, each velocity dimension $\dot{s}_{id}$ is updated in order to guide the particles' positions s_{id} to search across the most promising areas of the search space.

$$s_i^d(q+1) = s_i^d(q) + \dot{s}_i^d(q+1) \quad (6)$$

However, even though PSO is a powerful optimization method, if the optimization problem suffers changes in the objective function, for example between blocks of data, the particles can get stuck in local minima. To avoid this, an alternative should be to start a full PSO optimization process from scratch for each time that the module is activated. But it should be very time consuming and sometimes even unnecessary if the changes occur around the preceding optimum region.

Taking this into account, we implement the DPSO module to re-start optimization processes from preceding results in order to save computational time. In order to conceive it, we combine two dynamic optimization techniques: re-randomization and re-evaluation of solutions and apply them into this module. The main steps are listed in Algorithm 1, which are explained as follows. First of all, once the DPSO module is activated, every fitness is updated from the re-evaluation of the current position $\mathbf{s}_i$ and best position $\mathbf{s}'_i$ of each particle $\mathbf{s}_i$ in the swarm $\mathcal{S}(k)$ (steps: 3 to 6). It is performed to avoid that the particle's memory become obsolete [3]. In fact, the fitness of the best positions $\mathbf{s}'_i$ can be profited from the preceding level (adapted grid-search), what dispenses a second evaluation. Thereafter, a re-optimization process is launched by keeping ρ% of the best particles positions from the swarm $\mathcal{S}(k-1)$, which was computed in the previous optimization, and by creating randomly new particles over the search space [13]. Some of them located nearly to the previous optimum region. In this manner, we guarantee that fine searches are realized based on previous information, what can have faster adaptation to new data than full optimization processes (steps 7 and 8). At the same time, we also add more diversity in the algorithm for searching new solutions, what avoids situations in which the whole swarm has already converged to a specific area. Finally, the steps 9 to 23 correspond to main steps of the PSO implementation, but slightly modified by adding a mechanism that updates the connections among the particles, if no improvement is observed between iterations (steps 19 to 21). These latter are suggested by Clerc [7] as an alternative to improve the adaptability and hence the performance of the swarm.

In fact, the re-randomization and re-evaluation of solutions are both techniques already applied in the PSO literature [3, 13] to solve dynamic optimization problems. In particular, these PSO variants are commonly called DPSO (Dynamic PSO), so for the sake of simplicity, we name this module as DPSO to refer such a combination of approaches. Nevertheless, it is important to distinguish that existing dynamic PSO algorithms apply such techniques and change detection mechanisms in each iteration, since they suppose that objective function changes can happen during the optimization. In here, as the optimization over a dataset $\mathcal{D}(k)$ at a given instant t is indeed static, we apply these dynamic techniques to prepare the optimization module for transitions from preceding optimizations knowledge to launch new ones. As a result, we take advantage of these techniques to provide diversity in the solutions and clues on optimal starting points before the optimization. Thus, unlike actual dynamic PSO versions, no extra computational effort is added at each iteration. In light of this, in Figure 2 and in the rest of this paper, our DPSO module represents the application of these dynamic techniques to cooperate with the optimization algorithm, but not in its interior in each iteration. Thus, through the use of these modules, the proposed method allows the searching process to evolve and adapt itself dynamically. In the next section, we present some experiments carried out to validate the proposed method.

5. EXPERIMENTAL PROTOCOL

In order to test the effectiveness of the proposed method, a series of experiments were carried out. In particular, we compare our method with other model selection strategies under a gradual learning scenario. In a gradual learning scenario, a SVM classifier must be built gradually from scratch whenever more data is available. We have used datasets generated from synthetic and real-world problems. For each dataset, the following experimental setup was conceived. First of all, the original training sets were divided into sets of data. The total number of samples for each dataset was progressively increased according to a logarithmic rule [12], from about 16 examples per class to the total number of samples available in the dataset. For datasets in which the original distribution of samples was unbalanced among the classes, we have maintained the original class-priors for each dataset. Then we have applied each SVM model selection strategy over the datasets. Once the model selection was finished for each dataset, the performance of the classifiers were assessed in terms of its generalization error on the test set after each simulation. The generalization error was es-

Algorithm 1 Our implementation of Dynamic PSO

Input: PSO parameters and previous swarm $\mathcal{S}(k-1)$.
Output: Optimized solutions.
for all particles i from $\mathcal{S}(k-1)$ such that $1 \leq i \leq P$ **do**
  Compute fitness values for $\mathbf{s}_i$ using $\mathcal{D}(k)$
  Update $\mathbf{s}'_i$ if $\mathbf{s}_i$ is better ($\mathbf{s}'_i \leftarrow \mathbf{s}_i$)
end for
Initialize dynamically the new swarm $\mathcal{S}(k)$ by keeping ρ% of the best information (positions $\mathbf{s}'_i$) from the preceding swarm $\mathcal{S}(k-1)$ and by creating new particles.
Initialize the links among the particles based on a nearest neighborhood rule according to the topology chosen.
$q \leftarrow 0$;
repeat
  for all particles i such that $1 \leq i \leq P$ **do**
    Compute fitness value for the current position $\mathbf{s}_i(q)$
    Update $\mathbf{s}'_i(q)$ if position $\mathbf{s}_i(q)$ is better ($\mathbf{s}'_i(q) \leftarrow \mathbf{s}_i(q)$)
  end for
  Select the best fitness of this iteration q
  for all particles i such that $1 \leq i \leq P$ **do**
    Update $\dot{s}_i(q)$ (Equation 5) and $\mathbf{s}_i(q)$ (Equation 6)
  end for
  if $\mathcal{F}(\mathbf{s}^*(q)) = \mathcal{F}(\mathbf{s}^*(q-1))$ {No improvement. Change particle communication structure} **then**
    Randomly change the particles' links based on the topology chosen.
  end if
  $q = q + 1$
until maximum iterations or other stop criteria be attained

timated as the ratio of misclassified test set samples over the total number of test samples. As some strategies tested were stochastic algorithms based, the results represent averages drawn over 10 replications. The kernel chosen for the SVM classifier was the RBF (Radial Basis Functions). Hence, the model selection methods were carried out to find optimal values for the hyper-parameters set (C, γ). More specifications on the approaches tested and information on the datasets are outlined in next section.

5.1 Datasets

We have used six synthetic and real-world datasets in the experiments. They are listed in Table 1 with more details. Adult, Dna, and Satimage are databases from the UCI Repository [2][2]. NistDigit is a dataset composed of samples from the NIST digits Special database 19 (NIST SD19). Two distinct test sets were used NistDigit 1 (60,089 samples) and NistDigit 2 (58,646 samples), which are partitions hsf-4 and hsf-7, respectively. The features set is the same employed in [19]. IR-Ship database is a military database that consists of images of eight different classes of ships [20]. P2 problem is a two-class problem, where each class is defined in multiple decision regions [12].

In order to speed up the execution of our experiments, we have conducted on a Beowulf cluster with 20 nodes using Athlon XP 2500+ processors with 1GB of PC-2700 DDR RAM (333MHz FSB). The optimization algorithms were implemented using LAM MPI v6.5 in master-slave mode with a simple load balance. It means that while one master node executes the main operation related to the control of the processes, the evaluations of fitness are performed by several slave processors.

[2]More information on these databases can be found in http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/

Table 1: Specifications on the databases used.

Database	# of Class.	# of Feat.	# of Train. Samples	# of Test Samples
Adult	2	123	3,185	29,376
Dna	3	180	1,400	1,186
IR-Ship	8	11	1,785	760
NistDigit	10	132	5,860	60,089 /58,646
P2	2	2	3,856	10,000
Satimage	6	36	3,104	2,000

5.2 SVM Model Selection Strategies Tested

The following model selection strategies were compared:

- *Traditional Grid-Search (GS)*: This method selects the best solution by evaluating several combinations of possible values. The best combination is kept to train the final SVM classifier. In this study, we consider a grid of 70 (7x10) positions, where the possible combinations lie in these values: $C = \{0.01, 0.1, 100, 150, 170, 250, 600\}$, and $\gamma = \{0.08, 0.15, 15, 20, 50, 100, 300, 500, 1000, 1500\}$.
- *Grid-Search on 1-st (1st-GS)*: This strategy applies a traditional grid-search only over the first dataset and retain the same solution found for the subsequent subsets.
- *Full Particle Swarm Optimization (FPSO)*: The optimal hyper-parameters values are selected by the standard PSO algorithm for each new set of data.
- *Chained PSO (CPSO)*: It also applies PSO to search for optimal solutions. However, the solutions in here are optimized among sequence of datasets in a chained way, like a serial process. It means that the optimization process is performed continuously over the datasets, and not by fully re-initializing the swarm between sets.
- *Dynamic Model Selection (DMS)*: This strategy represents the proposed method introduced in Section 4.

5.3 Experiments Parameters Setting

The following parameters setting was employed:

- *Optimization Algorithms Parameters*: The swarm size was set to 20. The dimensions of each particle is denoted by the set of SVM hyper-parameters (C, γ), where the maximum and minimum values of such dimensions were set to $[2^{-6}, 2^{14}]$, $[2^{-15}, 2^{10}]$, respectively. The topology used in PSO and DPSO was the *lbest* with $\lambda = 3$. This topology was selected because unlike the *gbest* topology, which has a tendency toward premature convergence because all particles are influenced by the same global source, the *lbest* topology is more sophisticated for exploring multiple regions in parallel [16]. Two stop criteria were implemented for the optimization processes. The first one was implemented based on the maximum iteration permitted (100). The second one was related to improvement, i.e. if the best value of fitness did not improve over 10 consecutive iterations, then the optimization process was stopped.

- *Objective Function*: we use the minimization of the generalization error from five-fold cross-validation (CV) procedures over a training set as suggested in [4].

5.4 Obtained Results

The results are reported in Tables 2 and 3, in terms of generalization error rates, number of support vectors stored, and computational time spent, respectively. It is important to mention that these results were tested on multiple comparisons using the Kruskal-Wallis nonparametric statistical test by testing the equality between means values. The confidence level was set to 95% and Dunn-Sidak correction was applied to the critical values. The best results for each classification problem are shown in bold.

From the results, it was demonstrated how important a careful selection of hyper-parameters is to generate high performing classifiers. For instance, the results for the GS and 1st-Grid approaches in Table 2 show us that searching for optimal hyper-parameters given a new dataset can reach better results, in both classification accuracy and model complexity, than those that apply a searching process just once. Additionally, we have observed that PSO based approaches are very promising, since they have overcome the two grid-search methods through the experimental results in Table 2. Furthermore, the most important fact is that the proposed method (DMS) was able to attain similar results, but being less time consuming, than the full PSO (FPSO) strategy.

All these results, mainly comparing GS *vs* 1st-GS and CPSO *vs* DMS, are particularly interesting because they confirm the importance of tracking optimal solutions when new data is available and show the relevance of the proposed method. By analyzing the results, we can say that by shifting between re-evaluations and re-optimizations of previous swarms can be quite effective for building new solutions. The adapted grid module is less time consuming and more performing than evaluating, for instance a grid randomly composed of 70 different combinations or starting a whole new optimization process (FPSO). For a deeper analysis of the proposed method, we have depicted in Table 4 the frequencies of how many times a module was responsible for the selection of the final solution. From these results, it is possible to guess even the different degree of difficulty among the databases. For example, databases whose the final solutions were pointed out more often by the DPSO module, e.g. Dna seems to have a major degree of uncertainty, due perhaps a greater overlapping between classes, than other databases, such as NistDigit for example. In addition, we have also seen that DPSO module is advantageous, mainly in terms of processing time demanded to search for solutions, since unlike FPSO, which requires to perform several iterations, because it starts a new search every time randomly, DPSO saves time by applying dynamic optimization techniques, such as: the use of previous knowledge, increasing of diversity, etc. As a result, when the DPSO module is activated, it has converged faster and with similar results to that obtained with FPSO and better than CPSO.

The results have also shown an important advantage of our dynamic model selection strategy (DMS) in relation to the common FPSO strategy. While the FPSO optimization approach took a huge computation time to perform the model selection processes, mainly considering it for each set of data, our proposed method was capable of finding satisfactory solutions by consuming less computational time. It is because the FPSO strategy requires more evaluations than the proposed method, especially over each dataset, or still because when applied gradually over the datasets, the proposed method usually accelerates the searching process by approximating solutions before to reach the total size of training sets. Based on these results, we could see that the proposed method DMS has spent less computational time than the other strategies. Besides, it could be also noted that sometimes instead of spending much time by applying an optimization process over a large amount of data, the use of DMS gradually over subsets of data can be faster than a full optimization process over the entire original training set.

6. CONCLUSION

In this work we presented the SVM model selection problem as a dynamic optimization problem which depends on available data. In particular, it was shown that if one intends to build efficient SVM classifiers from different, gradual, or serial source of data, the best way is to consider the model selection process as a dynamic process which can evolve, change, and hence require different solutions overtime depending on knowledge available about the problem and uncertainties in the data. This kind of reasoning is particularly useful for real-world applications that require the generation of new classifiers dynamically in a serial way (e.g. those involving streaming data).

In order to solve this problem and concern also such dynamism, we proposed a PSO-based framework (DMS) which is based on the ideas of self-organization, change detection, and dynamic optimization techniques to track the optimal solutions and save computational time. The relevance of the proposed method was confirmed through experiments conducted on six databases. In short, the results have shown that: (1) if PSO is applied sequentially over datasets as a whole optimization process (Chained PSO) with the purpose of saving computational time, the resulting optimized solutions may stay trapped in local minima after successive model hyper-parameters selection processes. On the other hand, (2) full optimization processes with PSO (Full PSO strategy) is indeed efficient to reach good results, but very time consuming mainly when applied to each new dataset. (3) DMS was significantly similar to full optimization processes, but being less computationally expensive, mainly to the use of the dynamic optimization techniques. Eventually, by taking less iterations and with capabilities of adaptation, the proposed method is very promising to be used in a fully dynamic environment, mainly for those applications where the system must adapt itself to new data.

7. ACKNOWLEDGMENTS

This research was supported by Defence Research and Development Canada, DRDC-Valcartier under the contract W7701-2-4425 and in part by grant OGP0106456 to Robert Sabourin from the NSERC of Canada.

8. REFERENCES

[1] N. Ayat, M. Cheriet, and C. Suen. Automatic model selection for the optimization of SVM kernels. *Pattern Recognition*, 38(10):1733–1745, October 2005.

[2] C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998.

Table 2: Compilation of mean error rates and support vectors over the replications when the size of the dataset attained the size of the original training set.

Database	GS		1st-GS		FPSO		CPSO		DMS	
	Error	# SV	Error	# SV	Error	# SV	Error	# SV	Error	# SV
Adult	17.54	1508	24.06	1572	**15.55 (0.06)**	1176.50 (12.53)	23.85 (0.01)	3075.00 (10.00)	**15.56 (0.05)**	1174.80 (12.66)
Dna	12.82	1906	42.24	1914	**5.13 (0.18)**	628.40 (32.50)	6.37 (0.44)	436.10 (42.83)	**5.16 (0.56)**	810.60 (31.69)
IR-Ship	6.05	443	7.50	661	**4.86 (0.35)**	320.70 (13.34)	5.66 (0.45)	671.40 (21.74)	**4.72 (0.29)**	318.70 (9.53)
NistDigit 1	2.82	880	6.84	2912	**2.75 (0.04)**	898.40 (30.45)	3.02 (0.23)	1556.30 (62.56)	**2.74 (0.14)**	947.40 (55.09)
NistDigit 2	7.38	880	14.30	2912	**6.68 (0.15)**	898.40 (30.45)	7.33 (0.59)	1556.30 (62.56)	**6.72 (0.39)**	947.40 (55.09)
P2	1.79	226	3.71	430	**1.64 (0.10)**	161.40 (26.12)	2.03 (0.29)	383.50 (77.37)	**1.69 (0.14)**	152.80 (8.47)
Satimage	10.20	1117	10.50	1073	**8.06 (0.13)**	1888.00 (93.51)	14.32 (0.30)	1384.10 (60.64)	**8.26 (0.22)**	1849.00 (99.64)

Table 3: Mean computional time spent for model selection processes for the entire sequences of datasets with the most promising strategies. Results over the entire databases (FPSO-all data) are also reported.

Database	FPSO-all data	FPSO	CPSO	DMS
Adult	1:28:07 (0:38:13)	2:41:36 (0:02:53)	1:37:21 (0:01:07)	**0:32:31 (0:02:50)**
Dna	0:34:59 (0:15:59)	01:07:58 (0:01:34)	0:42:27 (0:00:39)	**0:14:21 (0:01:01)**
IR-Ship	0:19:08 (0:07:41)	0:30:42 (0:01:01)	0:15:17 (0:04:09)	**0:11:26 (0:05:00)**
NistDigit	6:47:51 (2:22:15)	13:46:00 (0:16:04)	3:46:24 (0:08:33)	**0:56:38 (0:05:34)**
P2	6:02:28 (0:48:29)	16:04:54 (0:17:44)	10:21:50 (0:13:47)	**5:35:55 (0:33:24)**
Satimage	1:45:55 (0:38:40)	2:46:18 (0:03:41)	1:41:29 (0:02:22)	**1:31:03 (0:05:04)**

Table 4: Modules' frequencies (%) to point final solution.

Database	Best Kept	Adapted Grid	DPSO
Adult	17.00	18.50	64.50
Dna	4.37	11.87	83.76
IR-Ship	21.82	12.73	65.45
NistDigit	5.62	61.25	33.13
P2	18.09	37.14	44.77
Satimage	33.12	6.25	60.63

[3] A. Carlisle and G. Dozier. Tracking changing extrema with adaptive particle swarm optimizer. In *Procs of the 5th BWAC*, pages 265–270, Orlando, USA, 2002.

[4] C. C. Chang and C. J. Lin. Libsvm: a library for support vector machines, 2005.

[5] O. Chapelle and V. Vapnik. Model selection for support vector machines. *In Advances in Neural Information Processing Systems*, pages 230–236, 1999.

[6] C. Chatelain, S. Adam, Y. Lecourtier, L. Heutte, and T. Paquet. Multi-objective optimization for SVM model selection. In *Procs of ICDAR*, pages 427–431, 2007.

[7] M. Clerc. *Particle Swarm Optimization*. ISTE Publishing Company, London, 2006.

[8] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, 2002.

[9] L. Cohen, G. Avrahami-Bakish, M. Last, A. Kandel, and O. Kipersztok. Real-time data mining of non-stationary data streams from sensor networks. *Information Fusion*, 9(3):344–353, 2008.

[10] B. de Souza, A. de Carvalho, R. Calvo, and R. Ishii. Multiclass SVM model selection using particle swarm optimization. In *Procs of ICHIS*, pages 31–34, 2006.

[11] F. Friedrichs and C. Igel. Evolutionary tuning of multiple SVM parameters. In *Proceedings of the 12th ESANN*, pages 519–524, 2004.

[12] P. Henniges, E. Granger, and R. Sabourin. Factors of overtraining with fuzzy ARTMAP neural networks. In *Procs of the IJCNN*, pages 1–4, 2005.

[13] X. Hu and R. C. Eberhart. Adaptive particle swarm optimization: Detection and response to dynamic systems. In *Procs of the CEC*, pages 1666–1670, 2002.

[14] M. Jiang and X. Yuan. Construction and application of PSO-SVM model for personal credit scoring. In *Procs of the ICCS*, pages 158–161, 2007.

[15] Y. Jin and J. Branke. Evolutionary optimization in uncertain environments - A survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317, 2005.

[16] J. Kennedy. Some issues and practices for particle swarms. In *Procs of IEEE SIS*, pages 801–808, 2007.

[17] J. Kennedy and R. C. Eberhart. Particle swarm intelligence. In *Procs of ICNN*, pages 1942–1948, 1995.

[18] A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh. DNPSO: A dynamic niching particle swarm optimizer for multi-modal optimization. In *Procs of the IEEE CEC*, pages 26–32, 2008.

[19] L. Oliveira, R. Sabourin, F. Bortolozzi, and C. Suen. Automatic recognition of handwritten numerical strings: a recognition and verification strategy. *IEEE Transactions on PAMI*, 24(11):1438–1454, 2002.

[20] Y. Park and J. Sklansky. Automated design of linear tree classifiers. *Pattern Recognition*, 23:1393–1412, 1990.

[21] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, NY, 1995.