Journal of Machine Learning Research (2008) Submitted 01/07; Published

# Particle Swarm Full Model Selection

**Hugo Jair Escalante** HUGOJAIR@CCC.INAOEP.MX
**Manuel Montes** MMONTESG@INAOEP.MX
**Luis Enrique Sucar** ESUCAR@INAOEP.MX
*Department of Computational Sciences*
*National Institute of Astrophysics, Optics and Electronics*
*Puebla, México, 72840*

**Editor:** Isabelle Guyon and Amir Saffari

## Abstract

In this paper we propose the application of particle swarm optimization (*PSO*) to the problem of *full model selection (FMS)* for classification tasks. We define *FMS* as follows: given a pool of preprocessing methods, feature selection and learning algorithms, select the combination of these that obtains the lowest classification error for a given data set; the task also includes the selection of hyperparameters for the considered methods. This problem generates a vast search space to be explored, well suited for stochastic optimization techniques. *FMS* can be applied to any classification domain as it does not require domain knowledge. Different model types and a variety of algorithms can be considered under this formulation. Furthermore, competitive yet simple models can be obtained with *FMS*. We adopted *PSO* for the search because of its proved performance in different problems and because of its simplicity, since neither expensive computations nor specialized methods are needed. We compared *PSO* against random search in *FMS* using cross validation for assessing models. Experimental results in benchmark data give evidence that *PSO* outperforms random search at the end and through the search process. Also, *PSO* exhibited better convergence behavior and it is less prone to overfitting than random search. Results obtained in a model selection competition show the competitiveness of models selected with *PSO* compared to models selected with more sophisticated techniques and with methods that use domain knowledge.

**Keywords:** Full model selection, particle swarm optimization, hyperparameter optimization, cross validation, machine learning challenges

## 1. Introduction

Model selection consists of the task of picking the model that best describes a data set (Hastie et al., 2001). Since the phrase *describing a data set* can be interpreted in several different ways, the model selection task can denote diverse related problems, including: variable and feature selection (Bengio and Chapados, 2003; Guyon et al., 2006a; Guyon and Elisseeff, 2003), system identification (Voss and Feng, 2002; Nelles, 2001), parameter-hyperparameter optimization (Guyon et al., 2006b; Kim et al., 2002; Hastie et al., 2001; Cawley and Talbot, 2007; Escalante et al., 2007), and discretization (Boullé, 2007; Hue and Boullé, 2007). In this paper we give a broader sense interpretation to this task that we have named *full model selection* (*FMS*). The *FMS* problem consists of the following:

given a pool of preprocessing methods, feature selection and learning algorithms, select the combination of these that obtains the lowest classification error for a given data set. This task also includes the selection of hyperparameters for the considered methods, resulting in a vast search space to be explored that is well suited for stochastic optimization techniques. Adopting a broader interpretation to the model selection problem allows us to consider different model types and a variety of methods, opposed to specialized techniques that consider a single model type (i. e. either learning algorithm or feature selection method, but not both) and a single method (e. g. neural networks). Also, since neither prior domain knowledge nor machine learning knowledge is required, *FMS* can be applied to any classification problem without modification. This is a clear advantage over ad-hoc model selection methods that perform well only for a single domain or that work for a fixed algorithm. This will help users with limited machine learning knowledge, since *FMS* can be seen as a black-box tool for model selection. Experts on machine learning can be also benefited as well. For example, several authors make use of search strategies for the selection of candidate models (Lutz, 2006; Boullé, 2007; Reunanen, 2007; Wichard, 2007), our approach can be used for obtaining such candidate models with no modification. It would be expected a considerable lost of accuracy by gaining generality. However, this is not the case of of our approach since it showed comparable performance to sophisticated methods designed for a single algorithm and to methods that take into account domain knowledge (Guyon et al., 2007b). The main drawback of this interpretation is the computational cost it involves exploring the vast search space. However, preliminary results show that the method can be efficient by adopting a simple subsampling strategy (see Section 4.2). The lack of interpretability of the selected models can be another limitation of our approach, as users may have no idea of why a given model was selected by *PSMS* for a given data set. This, however, is a minor limitation as this method is intended for users that may have limited knowledge on machine learning techniques.

We propose the application of particle swarm optimization (*PSO*) for exploring the full-models search space. *PSO* is a bio-inspired search technique that has shown comparable performance to that of evolutionary algorithms (Angeline, 1998; Reyes and Coello, 2006). Like evolutionary algorithms, *PSO* is useful when other techniques such as gradient descend or direct analytical discovery are not possible, combinatoric and real-valued function optimization in which the optimization surface possesses many locally optimal solutions are well suited for swarm optimization. In *FMS* we should find the best combination of methods (for preprocessing, feature selection and learning) and simultaneously optimizing real valued functions (finding pseudo-optimal parameters for the considered methods), in consequence the application of *PSO* is straightforward. The methodological differences between swarm optimization and evolutionary algorithms have been highlighted by several authors (Angeline, 1998; Kennedy and Eberhart, 1995, 2001). However a difference in performance has not been demonstrated in favor of either method. Such demonstration would be a difficult task because no black-box stochastic optimization algorithm can outperform another (not even to random search) overall optimization problems (Wolpert and Macready, 1997; van den Bergh, 2001). We selected *PSO* instead of evolutionary algorithms because of its simplicity and generality as no ad-hoc modification was made to *PSO* for applying it to *FMS*. *PSO* is easier to implement than evolutionary algorithms because it only involves a single operator for updating solutions. Opposed to evolutionary algorithms where methods for represen-

tation, cross-over, mutation, speciation and selection should be considered. Furthermore, *PSO* has been found to be very effective in a wide variety of applications, being able to produce good solutions at a very low computational cost (Gudise and Venayagamoorthy, 2003; Hernández et al., 2004; Hu et al., 2003; Yoshida et al., 2001; Robinson, 2004; Kennedy and Eberhart, 2001; Reyes and Coello, 2006).

We compare *PSO* to a random search implementation in order to evaluate the added value of using the swarm strategy. For both search methods we used cross validation ($CV$) for assessing the *goodness* of models, trying to avoid overfitting. Experimental results in benchmark data give evidence that both *PSO* and random search are effective strategies for *FMS*. However, we found that *PSO* outperforms random search through and at the end of the search process, showing better convergence behavior and overfitting avoidance. This result gives evidence that *PSO* can work as an any-time method for *FMS*, which means that the search algorithm can be interrupted at any time and yet return a competitive model. Hopefully, the more time we spend on searching the better models we obtain; note, however, that search methods are prone to overfitting. One should note that the random search implementation is not an easy baseline, as random-search solutions are created in the same way in which the *PSO* method is initialized (See section 4.1). Some models selected with *PSO* were evaluated in a model selection competition, where several specialized, sophisticated and prior-knowledge based methods for model selection were proposed. This in order to compare the performance of our *agnostic* method against other state of the art strategies in a real scenario. Models selected with *PSO* appeared at the top of the list of ranked models; in the $8^{th}$ position overall, in the $5^{th}$ position of methods that did not use domain knowledge and in the $2^{nd}$ position by using the software provided by the organizers (Guyon et al., 2006c, 2007a,b).

*PSO* has been widely used for parameter selection in supervised learning (Kennedy and Eberhart, 1995, 2001; Salerno, 1997; Gudise and Venayagamoorthy, 2003). However, parameter selection is related with the first level of inference in which, given a learning algorithm, we would like to find parameters for such algorithm in order to describe the data. For example, in neural networks the adjustment of weights between units according to some training data is a parameter selection problem. Hyperparameter optimization, on the other hand, is related with the second level of inference, that is finding parameters for the methods that in turn should find parameters for describing the data. In the neural network example, selecting the optimal number of input-output units, the learning rate, and the number of epochs for training the network is a hyperparameter optimization problem. The *FMS* problem is lying between levels 2 and 3 of inference, since we should select feature selection, preprocessing methods and classifier for a given data set, and performing simultaneously hyperparameter optimization for the selected methods. *PSO* has been used for hyperparameter optimization by Voss et al. (Voss and Feng, 2002), however they restricted the problem to linear systems in univariate data sets, considering one hundred data observations. In this paper we are going several steps further: we applied *PSO* for *FMS* (an even harder problem than that of hyperparameter optimization) considering non-linear models in multivariate data sets with a large number of observations.

The way we interpret the model selection problem (*FMS*) and the stochastic-search approach we propose are our main contributions. To the best of our knowledge there are not similar works that consider the *FMS* problem for classification tasks. Parallel to our

work, Gorissen et al. have proposed the use of genetic algorithms for meta-model selection in regression tasks (Gorissen, 2007; Gorissen et al., 2008), a similar approach that emerged totally independent to our proposal. One should note, however, that this method has been used for a different task in low dimensional data sets, most results are reported for 2D data, (Gorissen, 2007; Gorissen et al., 2008). Even with this dimensionality the method has been run for only a few iterations (10-15) with a population size of 10-15 individuals (Gorissen, 2007; Gorissen et al., 2008). Furthermore, the use of a genetic algorithm required the definition of specific operators *for each of the considered models*. Gorissen et al. considered seven different models (including neural networks and kernel methods) that required of 18 different genetic operators for creation, mutation and cross-over (Gorissen, 2007; Gorissen et al., 2008). Additionally, general operators for speciation and selection were also defined. In this work, however, we only used a single operator for updating solutions, regardless of the considered models. This clearly illustrates the main advantage of *PSO* over genetic algorithms, namely generality and simplicity.

The rest of this paper is organized as follows. In the next section we describe the general *PSO* algorithm, analyzing its parameters and convergence properties. Next, in Section 3, we describe how we applied *PSO* to the *FMS* problem, describing the representation and fitness function we adopted and discussing complexity issues. Then, in Section 4, we present experimental results in benchmark data, comparing the performance of *PSO* and random search on the *FMS* task, we also describe the application of *PSO* for hyperparameter optimization of a neural network and the results we obtained in a model selection competition. Finally, in Section 5, we outline the conclusions and discuss future work directions.

## 2. Particle swarm optimization (*PSO*)

*PSO* is a population-based search algorithm inspired on the behavior of biological communities that exhibit both individual and social behavior, examples of these communities are flocks of birds, schools of fishes and swarms of bees. Members of such societies share common goals (finding food, for example) that are realized by exploring its environment while interacting among them. Proposed by Kennedy et al. (Kennedy and Eberhart, 1995), *PSO* has become an established optimization algorithm with application in neural network training (Kennedy and Eberhart, 1995; Salerno, 1997; Kennedy and Eberhart, 2001; Gudise and Venayagamoorthy, 2003); though other applications include control and engineering design (Hernández et al., 2004; Hu et al., 2003; Yoshida et al., 2001; Robinson, 2004). The popularity of *PSO* is due in part to the simplicity of the algorithm (Kennedy and Eberhart, 1995; Reyes and Coello, 2006), but mainly because it has shown to be very effective for producing good results at a very low computational cost (Gudise and Venayagamoorthy, 2003; Kennedy and Eberhart, 2001; Reyes and Coello, 2006). Like evolutionary algorithms, *PSO* is appropriate for problems with immense search spaces that present many local minima, from this it follows that we use it for *FMS*.

In *PSO* each solution to the problem at hand is called a particle. Each particle, $i$, has a position in the search space as described in Equation (1)

$$\mathbf{x}_i =< x_{i,1}, x_{i,2}, \ldots, x_{i,d} > \tag{1}$$

where $d$ is the dimensionality of the solutions. A set of particles $\mathbf{S} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$ is called a swarm. Particles have an associated velocity value that they use for *flying* (exploring) through the search space. The velocity of particle $i$ is given by

$$\mathbf{v}_i =< v_{i,1}, v_{i,2}, \ldots, v_{i,d} > \tag{2}$$

where $v_{i,k}$ is the velocity for dimension $k$ of particle $i$. Particles adjust their flight trajectories by using the following updating equations

$$v_{i,j} = W \times v_{i,j} + c_1 \times r_1 \times (p_{i,j} - x_{i,j}) + c_2 \times r_2 \times (p_{g,j} - x_{i,j}) \tag{3}$$

$$x_{i,j} = x_{i,j} + v_{i,j} \tag{4}$$

where $v_{i,j}$ is the velocity of the particle $i$ in the $j^{th}$ dimension; $p_{i,j}$ is the value in dimension $j$ of the best solution found so far by particle $i$; $\mathbf{p}_i =< p_{i,1}, \ldots, p_{i,d} >$ is often called personal best. $p_{g,j}$ is the value in dimension $j$ of the best particle found so far within the entire swarm ($\mathbf{S}$); $\mathbf{p}_g =< p_{g,1}, \ldots, p_{g,d} >$ is considered the leader particle. Note that through $\mathbf{p}_i$ and $\mathbf{p}_g$ each particle $i$ takes into account individual and social information for updating its velocity and position. In that respect, $c_1, c_2 \in \mathbb{R}$ are values that weight the contribution of the individual and social information respectively. $r_1, r_2 \in [0, 1]$ are uniformly distributed random numbers that introduce randomness into the search process. $W$ is the so called inertia weight, whose goal is to control the impact of the history of the velocities of a particle over the current velocity, influencing the local and global exploration abilities of the algorithm. Once that velocity is updated, the new position of the particle $i$ in its $j^{th}$ dimension is recomputed, Equation (4). This process is repeated for each dimension of each particle $i$ and for each of the $m$ particles in the swarm ($\mathbf{S}$). The pseudo code of the general *PSO* algorithm is shown in Algorithm 1. The swarm is initialized randomly, taking into account restrictions on the values that each dimension can take. Then, by using Equations (3) and (4), particles in the swarm fly through the search space until a stop criteria is met. Usually, the process stops when either a maximum number of iterations ($I$) is reached or a minimum error value is obtained by a particle in the swarm (we used the first criteria for *FMS*); eventually, an (locally) optimal solution is found.

### 2.1 Fitness function

We have already talked about solutions that are better than others, however we have not introduced yet the way we evaluate the *goodness* of solutions. As in most search algorithms, in *PSO* a fitness function is needed to evaluate the aptitude of candidate solutions. The definition of a fitness function depends on the problem at hand, but in general should reflect the proximity of the solutions to the optima. A fitness function $F : \Psi \to \mathbb{R}$, where $\Psi$ is the space of particles positions $\mathbf{x}_{1,\ldots,Z}$, should return a value $f_{\mathbf{x}i}$ for each particle position $\mathbf{x}_i$, indicating how far particle $i$ is from the (optimal) solution of the problem at hand. In our case, the goal is to improve prediction accuracy of learning algorithms. Therefore, we can use a prediction accuracy measure as our $F$, see Section 3.3.

### 2.2 Analysis of *PSO* parameters

Selecting the best parameters $(W, c_1, c_2, m, I)$ for *PSO* is another model selection problem. In the application of *PSO* for *FMS* we would be dealing with a very complex problem

**Algorithm 1** *PSO.*

**Require:** $c_1, c_2$ individual/social behavior weights; $m$: swarm size; $I$: number of iterations; $F(\Psi \rightarrow \mathbb{R})$: fitness function; $W$: Inertia weight.
  Initialize swarm ($\mathbf{S} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$)
  Locate leader ($\mathbf{p}_g$)
  $i = 0$
  **while** $i < I$ **do**
    **for all** $\mathbf{x}_{1,\ldots,m} \in \mathbf{S}$ **do**
      Calculate velocity $\mathbf{v}_i$ for $\mathbf{x}_i$(Equation (3))
      Update position of $\mathbf{x}_i$ (Equation (4))
      Evaluate the *goodness* of $\mathbf{x}_i$ ($F(\mathbf{x}_i) = f_{\mathbf{x}i} \in \mathbb{R}$, see Section 2.1)
      Update $\mathbf{p}_i$ (*if necessary*)
    **end for**
    Update $\mathbf{p}_g$ (*if necessary*)
    $i^{++}$
  **end while**
  **return** $\mathbf{p}_g$

lying in the third level of inference. Fortunately, several empirical and theoretical studies have been performed about the parameters of *PSO* from which useful information can be obtained (Shi and Eberhart, 1998, 1999; Kennedy and Mendes, 2002; Reyes and Coello, 2006; Ozcan and Mohan, 1998; Clerc and Kennedy, 2002; van den Bergh, 2001).

We will start analyzing $c_1, c_2 \in \mathbb{R}$, the weighting factors for individual and social behavior. It have been shown that under certain values for $c_1, c_2$, convergence is guaranteed[1] for *PSO* (van den Bergh, 2001; Reyes and Coello, 2006; Ozcan and Mohan, 1998; Clerc and Kennedy, 2002). Note that most convergence studies simplify the problem to a single one-dimensional particle, setting $\phi = c_1 \times r_1 + c_2 \times r_2$, $\mathbf{p}_g$ and $\mathbf{p}_i$ constant (van den Bergh, 2001; Reyes and Coello, 2006; Ozcan and Mohan, 1998; Clerc and Kennedy, 2002). A complete study including the inertia weight is carried out by Van Den Bergh (van den Bergh, 2001). In agreement with this study the value $\phi < 3.8$ guarantees eventual convergence of the algorithm when the inertia weight $W$ is close to 1.0. For most experiments in this work we fixed $c_1 = c_2 = 2$, with these values we have $\phi < 3.8$ with high probability $P(\phi < 3.8) \approx 0.9$, given the uniformly distributed numbers $r_1, r_2$ (van den Bergh, 2001). The configuration $c_1 = c_2 = 2$ also has been proved, empirically, to be an effective choice for these parameters (Kennedy and Eberhart, 1995; Shi and Eberhart, 1998, 1999; Kennedy and Mendes, 2002). Note, however, the restriction that $W$ remains close to 1.0. We considered a value of $W = 0.95$ at the beginning of the search, decreasing it to $W = 0.2$ through the *PSO* iterations. In consequence it is possible that at the end of the search process the swarm may show divergent behavior. The choice of $W \in [0.95, 0.2]$ is justified by the fact that at the very beginning we would like that particles velocities are mostly influenced by the

1. Note that in *PSO* we say that the swarm converges iff $\lim_{t \rightarrow \inf} \mathbf{p}_{gt} = \mathbf{p}$, where $\mathbf{p}$ is an arbitrary position in the search space and $t$ indexes iterations of *PSO*. Since $\mathbf{p}$ refers to an arbitrary position, this definition do not mean neither convergence to local or global optimum, but convergence to the global best position in the swarm (van den Bergh, 2001; Reyes and Coello, 2006).

personal best solution ($\mathbf{p}_i$), allowing global search. While at the end of the search process most of the influence will be given by the leader particle ($\mathbf{p}_g$), allowing local search. Our selection of $W$ should not be suspiring since even the value $W \in [1.0, 0]$ has been proven to be better than using a fixed $W$ (Shi and Eberhart, 1998). Furthermore, empirical studies suggest that $W \in [1.1, 0.4]$ results in better performance for *PSO*, even with $c_1 = c_2 = 2$ (Shi and Eberhart, 1999, 1998). The inertia weight is a way to compensate the lack of a mutation operator like in evolutionary algorithms (Shi and Eberhart, 1999, 1998; Reyes and Coello, 2006). The decreasing of $W$ through the search process, called *adaptive inertia weight* (Shi and Eberhart, 1999, 1998), is a process similar to that of simulated annealing in which temperature is decreased exponentially, allowing global and local search (Kirkpatrick et al., 1983).

With respect to $m$, the size of the swarm, experimental results suggest that the size of the population does not damage the performance of *PSO* (Shi and Eberhart, 1999), although slightly better results are obtained with a large value of $m$. We would like to use a large value of $m$ for *FMS* as well, however, since the evaluation of solutions for *FMS* is expensive (see Section 3.4), we fixed $m = 10$ for all of our experiments. Regarding the number of iterations, to the best of our knowledge, there is not work on the subject. This is mainly due to the fact that this issue depends mostly on the hardness of the problem at hand and therefore a general rule can not be derived. For *FMS* the number of iterations should not be large because we may overfit the data, for most experiments in this paper we fixed $I = 100$.

Note that in Equation (3) every particle in the swarm knows the best position found so far by any other particle within the swarm, that is $\mathbf{p}_g$. With this formulation we are considering a fully-connected swarm topology in which every member knows the leader particle. This topology has shown to converge faster than any other topology (Kennedy and Mendes, 2002; Reyes and Coello, 2006; Kennedy and Eberhart, 2001). With this topology, however, the swarm is prone to converge to local minima. We tried to overcome this limitation by introducing an adaptive inertia weight ($W$).

## 3. Particle swarm full model selection

Since one of the strong advantages of *PSO* is its simplicity, its application to *FMS* is almost direct. Actually, particle swarm full model selection (*PSMS*, that is the application of *PSO* to *FMS*) is described by the pseudocode in Algorithm 1, with the only modification that $W$ is decreased through the iterations. In this section we describe additional details about *PSMS*. We start with a description of the pool of methods to choose from that we have considered. Then we describe the representation of particles and the fitness function we used and next we briefly discuss some complexity issues.

### 3.1 The challenge learning object package (*CLOP*)

In order to implement *PSMS* we would need to define the models search space. For doing this, we consider the set of methods in a machine learning toolbox from which full models can be generated. Currently, there exist several machine learning toolboxes, some of them publicly available (Smola and Schölkopf, 2006; Franc and Hlavac, 2004; van der Heijden et al., 2004; Wichard and Merkwirth, 2007; Witten and Frank, 2005; Saffari and Guyon,

| Object name | F | Hyperparameters | Description |
|---|---|---|---|
| *s2n* | *FS* | $f_{max}$, $w_{min}$ | Signal-to-noise ratio for feature ranking |
| *relief* | *FS* | $f_{max}$, $w_{min}$, $k_{num}$ | Relief ranking criterion |
| *gs* | *FS* | $f_{max}$ | Forward feature selection with Gram-Schmidt orth. |
| *rffs* | *FS* | $f_{max}$, $w_{min}$, *child* | Random forest used as feature selection filter |
| *svcrfe* | *FS* | $f_{max}$, *child* | Recursive feature elimination filter using svc |
| *standardize* | *Pre* | center | Standardization of the features |
| *normalize* | *Pre* | center | Normalization of the lines of the data matrix |
| $shift - scale$ | *Pre* | offset, factor, $take_{log}$ | Shifts and scale data |
| $pc - extract$ | *Pre* | $f_{max}$ | Extraction of features with *PCA* |
| *subsample* | *Pre* | $p_{max}$, balance | Subsample of the training patterns |
| *bias* | *Pos* | none | Finds the best threshold for the output of the classifiers |

Table 1: Feature selection (*FS*), preprocessing (*Pre*) and postprocessing (*Pos*) objects available in *CLOP*. A brief description of the methods and their hyperparameters is presented.

2006; Weston et al., 2005); even there is a track of this journal (*JMLR*) dedicated to machine learning software (JMLR, 2007). This is due to the increasing interest from the machine learning community in the dissemination and popularization of this research field (Sonnenburg, 2006; JMLR, 2007). The *Challenge Learning Object Package*[2] (*CLOP*) is one of such development kits distributed under the GNU license (Saffari and Guyon, 2006; Guyon et al., 2006c, 2007a,b) . *CLOP* is a $Matlab^{TM}$ toolbox with implementations of feature-variable selection methods and machine learning algorithms. The list of available preprocessing, feature selection and postprocessing methods in the *CLOP* toolbox is shown in Table 1. A description of the learning algorithms available in *CLOP* is presented on Table 2.

In consequence, our pool[3] of methods to select from are those methods described on Tables 1 and 2. A typical model consists of the *chain* (a grouping object that allows serial concatenation of different methods) combination of one (none) feature selection algorithm followed by a (several/none) preprocessing method, in turn followed by a learning algorithm and finally a (none) postprocessing algorithm. For example, the model given by:
***chain***$\{gs(fmax = 8)$*,standardize(center=1),neural(units=10,s=0.5,balance=1,iter=10)*$\}$
uses *gs* for feature selection, *standardization* of data and a balanced *neural network* classifier with 10 units, learning rate of 0.5, and trained for 10 iterations. The search space in *FMS* is given by all the possible combinations of methods and hyperparameters, an infinite search space due to the real valued parameters.

### 3.2 Representation

In *PSO* each potential solution to the problem at hand is considered particle, each particle has a position in the $d-$dimensional search space. Usually, the $d-$dimensions of particles positions are real valued, though also exist discrete representations (van den Bergh, 2001; Reyes and Coello, 2006). In *FMS* potential solutions are models, in consequence for *PSMS*

2. http://clopinet.com/CLOP/
3. Notice that the *CLOP* package includes also the spider package (Weston et al., 2005) which in turn includes other implementations of learning algorithms and preprocessing methods, however, in this work we only used *CLOP* objects.

| Object name | Hyperparameters | Description |
|---|---|---|
| *zarbi* | none | Linear classifier |
| *kridge* | shrinkage, kernel parameters (coef0, degree, gamma), balance | Kernel ridge regression |
| *naive* | none | Naive Bayes |
| *neural* | units number, shrinkage, maxiter, balance | Neural network (Netlab) |
| *rf* | units number, mtry | Random forest |
| *svc* | shrinkage, kernel parameters (coef0, degree, gamma) | SVM classifier |
| *gentleboost* | child, units number, balance, subratio | Gentle Boost algorithm |

Table 2: Available learning objects with their respective hyperparameters in the CLOP package.

we represented the position of each particle $i$ with a $d-$dimensional vector $\mathbf{x}_i$ as described in Equation (5).

$$\mathbf{x}_i =< x_{i,1}, y^1_{i,1\ldots a}, x_{i,2}, y^2_{i,2,\ldots b}, \ldots\ldots x_{i,n}, y^n_{i,1,\ldots c} > \tag{5}$$

Where each $x_{i,j}$ is a binary element whose value (0 or 1) indicates the absence or presence of method $j$ in particle (full-model) $i$; each entry $y^w_{i,1\ldots h}$ represents the $h-$parameters for method $w$ in particle $i$, this elements can be binary or real valued, depending on the parameters of method $w$. In this representation we have $n-$methods, each with their respective parameters. Note that in case $n = 1$, the problem will be reduced to single hyperparameter optimization for method $x_{i,1}$.

### 3.3 Fitness function for *PSMS*

As introduced in Section 2.1, a fitness function is a correspondence rule between solutions (particles' positions) and real numbers. These numbers should indicate how far a particle is from the optimal solution to the problem at hand. In *FMS* our goal is minimizing classification error (two class problems) in unseen (test) data. Therefore, our fitness function should relate models with an estimate of classification error in these unseen data. There are several options for the selection of a fitness function for *FMS*, including mean absolute error, balanced error rate, squared root error, recall, precision, area under the *ROC* curve, etcetera. All of these measures can give an estimate of models accuracy in unseen data. For this work we used the balanced error rate (*BER*) as our fitness function, because this measure takes into account misclassification rates in both classes. Furthermore, *BER* has been used in machine learning challenges as leading error measure for ranking participants (Guyon et al., 2007a,b). The *BER* of model $\psi$ is the average of the misclassifications obtained by $\psi$ over the classes in a data set, as described in Equation (6)

$$BER(\psi) = \frac{E_+ + E_-}{2} \tag{6}$$

where $E_+$ and $E_-$ are the misclassifications rates for the positive and negative classes, respectively. Note that the selection of the fitness function is a crucial aspect in *PSO*; the quality of the best solution found depends on it. However, the selection of a particular classification error measure is not crucial for the performance of *PSMS*. What it is actually critical is the way we obtain an estimate of classification error in unseen data, given a classification error measure and training data. This is one of the main challenges in model

selection, since error estimates using training data are very optimistic about the behavior of models on unseen data, this phenomenon is known as overfitting (Mitchell, 1997; Hastie et al., 2001; Nelles, 2001).

In consequence, comparing models accuracy using the same data for training and assessing them is not straightforward (Hastie et al., 2001; Nelles, 2001). For this reason we calculated the *BER* using a $k-$fold cross validation *(k-fold CV)* approach[4]. *k-fold CV* is the most used hold-out approach for model selection (Nelles, 2001; Hastie et al., 2001; Cawley and Talbot, 2007). Using a high value for $k$, say $k = N - 1$ where $N$ is the training set size (*leave one out - CV*), the error estimate is almost unbiased, but can have high variance because the $N$ training sets are so similar to each other (Nelles, 2001; Cawley and Talbot, 2007); furthermore, computation time could be a serious problem (Hastie et al., 2001; Nelles, 2001). With a low value for $k$, the estimate can have low variance but the bias could be a problem. The selection of an adequate $k$ for *k-fold CV* could improve the performance of our *PSMS* implementation. However, the selection of an optimal $k$ is still an open problem in the model selection field, it will be interesting to use *PSO* for the selection of $k$ as future work. For most experiments in this work we used $k = 5$ because it offers a good tradeoff between processing time and variance-bias.

### 3.4 Complexity issues

As we have seen the search space in the *FMS* problem is composed by all the possible models that can be built given the considered methods and their hyperparameters. This is an infinite search space even with the restriction we imposed into the values that models can take (see Section 3.2). This is the main drawback of the model selection interpretation that we have adopted. However, we should remark that this is a complexity drawback of the *FMS* interpretation and not of *PSMS*. Actually, any search algorithm used for *FMS* would be limited because of the vast search space to be explored. Properties of *PSO* are fast convergence rate, its simplicity and the fact that this method is well suited for complex search spaces (Kennedy and Eberhart, 2001; Reyes and Coello, 2006). The last property of *PSO* is the main reason for using fit for *FMS*. As we can see from Algorithm 1, *PSO* is very simple and does not involve expensive operations attributed to itself.

The computational expensiveness of *PSMS* is due to the fitness function we used. For each selected model the fitness function should train and evaluate such model $k-$times. Depending on the model complexity this process can be performed on linear, quadratic or higher order times. Clearly, computing the fitness function using the entire training set, opposed to *k-fold CV*, could reduce *PSMS* complexity, although we could easily overfit the data (see Section 3.3). For a single run of *PSMS* the fitness function should be evaluated $\rho = m \times (I + 1)$ times, with $m$ the swarm size and $I$ the number of iterations. Suppose the complexity of model $\lambda$ is bounded by $\lambda_O$ then the complexity of *PSMS* will be bounded by $\rho \times (k \times \lambda_O)$. Since $\lambda_O$ is related to the complexity of model $\lambda$, which depends on the size and dimensionality of the data set, its value may vary dramatically. For instance, computing the fitness function for a naive Bayes model in a high dimensional data set takes around

4. Note that the *BER* is still obtained from training data.

| ID | Data set | Training Patterns | Testing Patterns | Number of Replications | Input Features |
|---|---|---|---|---|---|
| 1 | Banana | 400 | 4900 | 10 | 2 |
| 2 | Breast cancer | 200 | 77 | 10 | 9 |
| 3 | Diabetis | 468 | 300 | 10 | 8 |
| 4 | Flare solar | 666 | 400 | 10 | 9 |
| 5 | German | 700 | 300 | 10 | 20 |
| 6 | Heart | 170 | 100 | 10 | 13 |
| 7 | Image | 1300 | 1010 | 10 | 20 |
| 8 | Splice | 1000 | 2175 | 10 | 60 |
| 9 | Thyroid | 140 | 75 | 10 | 5 |
| 10 | Titanic | 150 | 2051 | 10 | 3 |

Table 3: Data sets used in the comparison of *PSMS* and random search.

two seconds[5], however computing the same fitness function for the same data set could take several minutes if we use a support vector classifier. In the future we could introduce a complexity penalty term into the fitness function, in order to alleviate the complexity problem for *PSMS*. A simpler alternative solution maybe computing the fitness function for each particle by using only a subset of the available data, different each time. Preliminary results with the later solution show a significant processing time reduction, without loss of accuracy, see Section 4.2. We emphasize once again that high complexity is due to the nature of the *FMS* problem. With this interpretation, however, users will be able to obtain models for their data without requiring knowledge on the data or on machine learning techniques. Furthermore, as we will see in Section 4, models selected with *PSMS* have proven to be competitive among others selected with more sophisticated strategies (Guyon et al., 2006c, 2007a,b).

## 4. Experimental results

In this section we present results of experiments carried out using benchmark data from two different sources. Firstly, we present results on a suite of benchmark machine learning data sets[6], used by several authors (Mika et al., 2000; Rätsch et al., 2001; Cawley and Talbot, 2007). These data are described in Table 4, 10 replications of each data set were considered. We used these data sets to compare *PSMS* against random search in the *FMS* problem. Secondly, we applied *PSMS* to the data used in a model selection competition. The goal of these experiments was to compare the performance of *PSMS* against other sophisticated model selection approaches, a complete analysis of the results in the competition is carried out by Guyon et al. (Guyon et al., 2006c,b, 2007a,b).

5. Most of the experiments were carried out on a workstation with $Pentium^{TM}$ 4 processor at 2.5 GHz, and 1 gigabyte in RAM.
6. The original data are available at *http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm*, the same data in the *CLOP* format is available at *http://ccc.inaoep.mx/∼hugojair/*. The code of *PSMS* is also available in the same link.

### 4.1 *PSO* against random search

In order to evaluate the gain we can have by using *PSMS* instead of randomly picking models we implemented a random-search strategy and compare it to *PSMS*. This strategy generates solutions randomly and it evaluates the *goodness* of solutions by using *5-fold CV*, the solution with lowest error is kept as the best solution ($\mathbf{rs}_g$). The pseudocode of the random search strategy is shown in Algorithm 2. Solutions are created by considering uniformly-distributed random numbers[7], using the same vectorial representation for models that we adopted for *PSMS*. The way that solutions are created in random search is the same way that the swarm is initialized for *PSMS*. Therefore, random-search solutions are generated using randomness but also taking into account the restriction on the different parameters for the methods. Because of this last statement and of the nature of the *FMS* problem, the random search implementation is not, as one may think, an easy baseline.

**Algorithm 2** Random search.

**Require:** $I_r$: number of iterations; $F(\Psi \rightarrow \mathbb{R})$: fitness function;
  **while** $i < I_r$ **do**
    Create a solution ($\mathbf{c}_i$)
    **if** $F(\mathbf{c}_g) < F(\mathbf{rs}_g)$ **then**
      $\mathbf{rs}_g = \mathbf{c}_i$
    **end if**
    $i^{++}$
  **end while**
  **return** $\mathbf{rs}_g$

In each experiment described below we let *PSMS* and random search performing the same number of fitness function evaluations (that is, $I_r = m \times (I+1)$, where $m$ is the size of the swarm and $I$ the number of *PSMS* iterations), using exactly the same settings and data sets, which guarantees a fair comparison. Complexity is the same for both methods and processing time was approximately equal. Since both methods use the same fitness function and perform the same number of fitness function evaluations, the difference in performance will indicate the gain we have by guiding the search according to Equations (3) and (4).

We compare the *FMS* ability of *PSMS* and random search by evaluating accuracy of the selected models at the end of the search process; that is, we compare the performance of models $\mathbf{p}_g$ and $\mathbf{rs}_g$, selected with *PSMS* and random search respectively (see Algorithms 1 and 2). As recommended by Demsar (Demsar, 2006), we used the Wilcoxon signed-ranks test for the comparison of the resulting models. In this test the difference in accuracy of two models through $N-$data sets is obtained, the absolute values of the differences are ranked and sums of the ranks for each model are calculated. The null-hypothesis that both models perform equally well is rejected with 95% of confidence if[8] the smallest of the sums is lower or equal to $N_\kappa$, the critical value for $N$ and $\alpha = 0.05$. In the following we will refer to this statistical test with 95% of confidence when mentioning statistical significance.

7. Random numbers are scaled or discretized so that they represent the selection of certain method or hyperparameter value in the vectorial representation, see Section 3.2.
8. Note that a table of critical values is used for tests with $N \leq 25$, for a large number of data sets a $z$ statistic is calculated. A comprehensive description of the test is described by Demsar (Demsar, 2006).

| ID | Data set | RS-test-BER | PSMS-test-BER | RS-CV-BER | PSMS-CV-BER |
|---|---|---|---|---|---|
| 1 | Banana | **11.37 ± 0.46** | 12.72 ± 0.46 | 35.70 ± 1.18 | **16.64 ± 1.22** |
| 2 | Breast-cancer | 40.60 ± 5.85 | **37.54 ± 5.73** | 32.95 ± 0.16 | **29.84 ± 0.05** |
| 3 | Diabetis | 27.14 ± 2.68 | **26.50 ± 2.71** | 29.97 ± 0.43 | **24.11 ± 0.08** |
| 4 | Flare-solar | 32.95 ± 2.06 | **32.87 ± 2.10** | 32.84 ± 0.01 | **32.01 ± 0.02** |
| 5 | German | 27.26 ± 2.81 | **27.22 ± 2.80** | 31.04 ± 0.30 | **27.65 ± 0.06** |
| 6 | Heart | **20.26 ± 3.97** | 22.10 ± 4.11 | 19.73 ± 1.84 | **12.97 ± 0.12** |
| 7 | Image | **3.12 ± 0.55** | 3.23 ± 0.55 | 22.94 ± 1.06 | **6.78 ± 0.83** |
| 8 | Splice | 8.21 ± 0.57 | **7.20 ± 0.54** | 21.47 ± 0.95 | **10.53 ± 0.34** |
| 9 | Thyroid | **5.85 ± 2.7** | 6.94 ± 2.77 | 18.88 ± 0.94 | **5.81 ± 0.41** |
| 10 | Titanic | **30.26 ± 1.07** | 30.65 ± 1.08 | 27.01 ± 0.02 | **26.31 ± 0.02** |

Table 4: Average *test-BER* with error bars obtained by models selected with random search (**RS**) and *PSMS*; the best results are shown in **bold**. Also, we show *CV-BER*, the averaged fitness *BER* obtained by each of the candidate solutions through the search process.

In the first experiment we compared the models selected by *PSMS* and random search at the end of the search process. For each trial we fixed the number of iterations for *PSMS* to $I = 100$ with a swarm size of $m = 10$. In consequence both search strategies performed $m \times (I + 1) = 1010$ evaluations of the fitness function in each run. In each trial the training set is used for the search process, and the resulting model is evaluated in the test set. We called the error obtained through the search process *CV-BER* and the one obtained in the test set *test-BER*. We repeated this process for each replica of each data set described in Table 4. Averaged results of this experiment are showed in Table 4.

As we can see, both search strategies tied the number of data sets in which they outperformed the other in *test-BER*. Within the data sets, only in the *Heart* data there is a statistical significant difference favoring random search. Globally, however, we noticed that from the 100 runs ($10-$data sets,$\times$ $10-$replicas for each, see Table 4), 53.1% of the models selected with *PSMS* outperformed those obtained with random search. While only in 36.7% of the runs random search outperformed *PSMS*, in the rest (10.2%) both methods tied in performance. We performed a statistical test over these 100 results and we found that there is a statistical-significant difference favoring *PSMS*. These results are due to the fact that for each data set the models selected with random search obtained low *test-BER* in a small portion of the replications. On the other hand, models selected with *PSMS* outperformed those selected with random search in most of the replications, however, the error (*test-BER*) in these replications was not too low. This result gives evidence that *PSMS* can outperform random search in the *FMS* problem.

Columns five and six in Table 4 show the *BER* obtained with each strategy through the 1010 evaluations for each data set (averaged over replications); these results are also showed in Figure 1, left. *CV-BER* value reflects the behavior of the search strategies through the search process. A low *CV-BER* value indicates that solutions were close to the optimal in most of the times (convergence); while a high value indicates divergent behavior of the search method. Clearly, *PSMS* outperformed random search in this aspect, the difference is statistical significant. In Figure 1 we can graphically appreciate this fact, also it can be seen that variance of *PSMS* is lower than that of random search, which illustrates

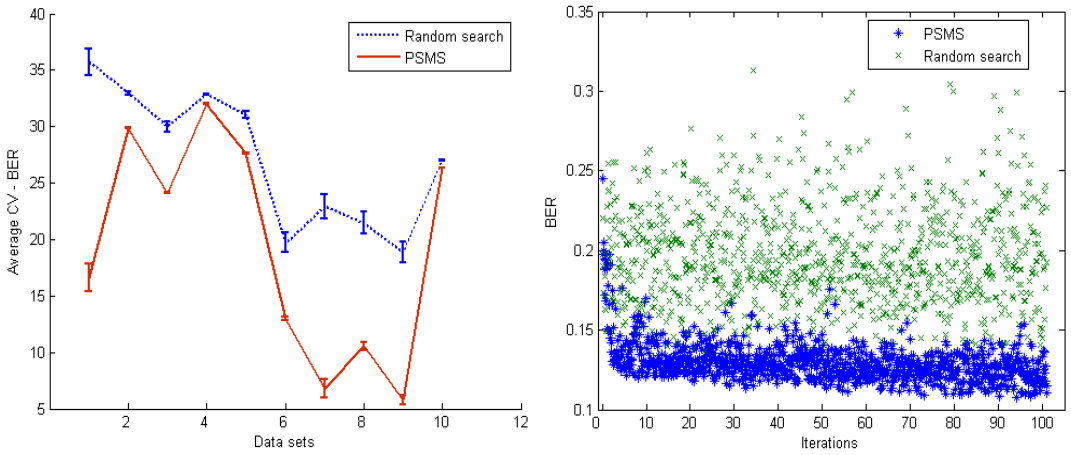


Figure 1: Left: mean and variance of *CV-BER* obtained for each data set in Table 4, the x-axis shows the ID of the data sets in Table 4. Right: mean *CV-BER* obtained through the search process by *PSMS* and random search for the *Heart* data set.

the better convergence behavior of *PSMS*. This result indicates that *PSMS* obtain better models through the search process than random search (according to the *CV-BER*), making *PSMS* a good candidate for the implementation of an any-time algorithm. This result also indicates that *PSO* is doing fine its job, since its objective is the optimization of the fitness function. The larger difference in *CV-BER* is obtained in the *Image* and *Thyroid* data sets, although random search obtained lower average *test-BER*. This last result implies that *PSMS* overfitted the data, at least in the last model and that the other method was benefitted of its random property. However, as we will see below, models selected with *PSMS* do not overfit the data in average. For illustration purposes we show in Figure 1 (right) the *CV-BER* obtained by *PSMS* through the search process for the *Heart* data set, the one in which random search outperformed *PSMS*. It is clear that with *PSMS* particles follow best-evaluated solutions eventually converging to a minima of the *CV-BER*. On the other hand, as expected the random search technique shows totally divergent behavior. This pattern was observed in all of the runs we performed.

In Figure 2 we show the average frequency, over all runs, of the methods preferred by random search and *PSMS*. Since randomly distributed numbers were used for generating solutions in random search, this method equally selected among the available classifiers (Figure 2, left). On the other hand, we can see that *PSMS* preferred support vector classifiers, which gave better results than the other models in most of the times. It is interesting the fact that in second term *PSMS* preferred *Zarbi* over other sophisticated classifiers like ridge regression and neural networks. For feature selection we restricted the search strategies by considering a subset of the available methods to choose from, see Table 1. This because of the data characteristics and for saving processing time However, as stated above, the same configuration of parameter settings and restrictions was considered for both search strategies in each data set. From Figure 2 we can see that *s2n* and *gs* have high selection-frequency for both methods, this is is due to the fact that these techniques were allowed for

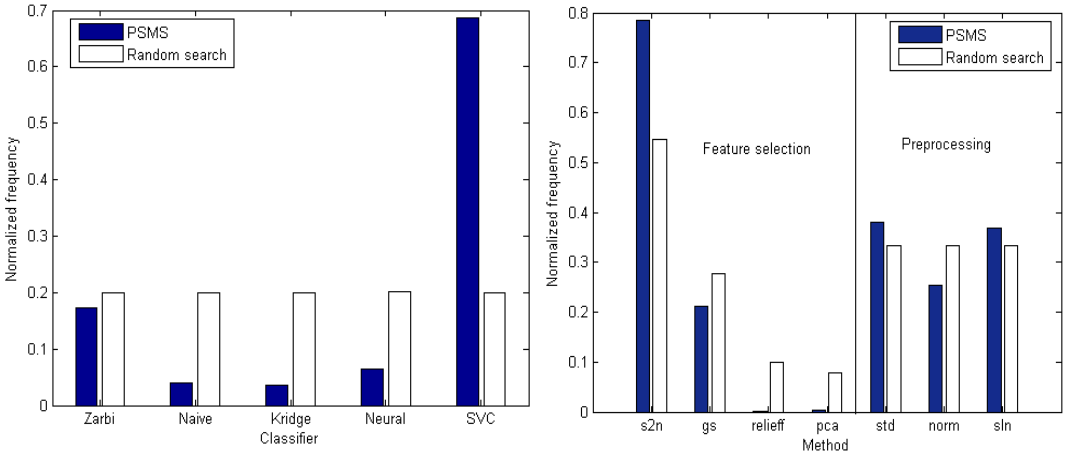


Figure 2: Normalized frequency of the selection of classifiers (left) and preprocessing-feature selection methods (right) by *PSMS* (shaded) and random search (unshaded).

all of the data sets. However, it is interesting to see that *PSMS* preferred *s2n* and *gs* instead of relief and *pca*, opposed to random search that selected uniformly. In the preprocessing methods it seems that *PSMS* found that *normalization* damaged the performance of its solutions, while random search again chose uniformly. As we can see, *PSMS* could also be used for analyzing the adequacy of certain models for a given data set and for comparison of learning methods.

We have seen that *PSMS* outperformed random search in *CV-BER*, however we also saw that *PSMS* overfitted the data for the *Image* and *Thyroid* data sets. In order to evaluate how much *PSMS* overfitted the data through the search process we ran *PSMS* and random search again, but this time measuring testing error for each candidate solution through the 1010 iterations. That is, for each candidate solution during the search process we obtained both the *CV-BER* and the *test-BER*$^*$[9]. For this experiment we used a single replica for all of the data sets in Table 4, results of this experiment are showed in Table 5. As we can see *PSMS* obtained lowest *CV-BER* for all data sets, agreeing with results in Table 4. Also, *PSMS* obtained lower *test-BER*$^*$ than random search in 8 out of 10 data sets, note that for the *Flare solar* and *Titanic* data sets the difference can be neglected. The difference in *test-BER*$^*$ is statistical significant favoring *PSMS*.

In Figure 3 we plot the average *CV-BER* and the *test-BER*$^*$ for all of the data sets. As we can see *PSMS* is superior in both *CV* and testing error. This result illustrates that *PSMS* is less prone to overfitting than random search, it also shows the adequacy of *PSMS* for working in an any-time schema. Furthermore, the variance showed by *PSMS* is lower than that of random search, which shows the better convergence behavior of *PSMS*. Results in Table 5 and Figures 1 and 3 give evidence that *PSMS* is less prone to overfitting than random search, although it is not free of it at all; furthermore, we can appreciate that *PSMS* shows better convergence behavior.

9. Note that for this experiment *test-BER*$^*$ is obtained for each solution tried during the search process, instead of obtaining *test-BER* using the final model only.

| Data set | RS-CV-BER | PSMS-CV-BER | RS-test*-BER | PSMS-test*-BER |
|---|---|---|---|---|
| Banana | $32.97^{+}_{-}0.972$ | $\mathbf{26.57^{+}_{-}0.151}$ | $36.72^{+}_{-}0.969$ | $\mathbf{30.34^{+}_{-}0.207}$ |
| Breast-cancer | $32.74^{+}_{-}0.133$ | $\mathbf{29.31^{+}_{-}0.038}$ | $37.85^{+}_{-}0.175$ | $\mathbf{34.14^{+}_{-}0.038}$ |
| Diabetis | $29.57^{+}_{-}0.412$ | $\mathbf{25.33^{+}_{-}0.118}$ | $31.31^{+}_{-}0.558$ | $\mathbf{27.39^{+}_{-}0.179}$ |
| Flare-solar | $32.19^{+}_{-}0.007$ | $\mathbf{31.21^{+}_{-}0.003}$ | $\mathbf{32.98^{+}_{-}0.014}$ | $33.06^{+}_{-}0.009$ |
| German | $31.36^{+}_{-}0.301$ | $\mathbf{28.05^{+}_{-}0.033}$ | $30.16^{+}_{-}0.442$ | $\mathbf{28.08^{+}_{-}0.061}$ |
| Heart | $19.21^{+}_{-}0.925$ | $\mathbf{12.31^{+}_{-}0.031}$ | $26.58^{+}_{-}0.655$ | $\mathbf{20.74^{+}_{-}}0.068$ |
| Image | $22.39^{+}_{-}0.986$ | $\mathbf{6.23^{+}_{-}0.681}$ | $22.41^{+}_{-}1.094$ | $\mathbf{7.04^{+}_{-}0.692}$ |
| Splice | $23.10^{+}_{-}0.819$ | $\mathbf{14.24^{+}_{-}0.389}$ | $20.39^{+}_{-}1.111$ | $\mathbf{11.61^{+}_{-}}0.386$ |
| Thyroid | $21.38^{+}_{-}1.140$ | $\mathbf{10.41^{+}_{-}0.109}$ | $20.65^{+}_{-}1.371$ | $\mathbf{9.89^{+}_{-}}0.152$ |
| Titanic | $26.14^{+}_{-}0.043$ | $\mathbf{24.81^{+}_{-}0.027}$ | $\mathbf{29.62^{+}_{-}0.017}$ | $29.71^{+}_{-}0.043$ |
| Average | $27.105^{+}_{-}0.573$ | $\mathbf{20.847^{+}_{-}0.158}$ | $28.867^{+}_{-}0.641$ | $\mathbf{23.2^{+}_{-}}0.186$ |

Table 5: Average *CV-BER* and *test-BER* obtained by candidate models tried during the search process by random search (**RS**) and *PSMS*, we also show the variance.

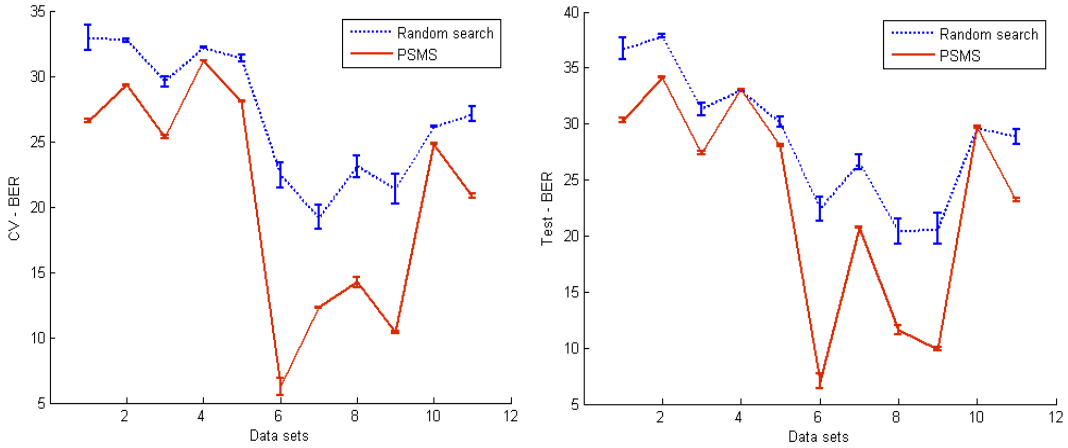


Figure 3: Average *CV-BER* (left) and *test-BER*$^*$ (right) obtained by candidate solutions tried during the search process by random search and *PSMS*, variance is also plotted. The x-axis shows the ID of the data sets in Table 4; $x_{11}$ shows the average.

### 4.2 Model selection competition

Recently a model selection competition called *agnostic learning vs. prior knowledge challenge* (*ALPK*) has concluded (Guyon et al., 2007a,b). Through its different stages the competition evaluated novel strategies for model selection as well as the added value of using prior knowledge for improving classification accuracy (Guyon et al., 2007b). This sort of competitions are very useful because, through them, the real effectiveness of methods can be evaluated; motivating further research in the field and collaborations among participants. The rules of the challenge are quite simple: organizers provide five data sets for classification (two class problems $[-1, 1]$) together with the *CLOP* toolbox (Saffari and Guyon, 2006). The task consists of obtaining the model that achieves the lowest *BER* over the five data sets on unseen data. Participants may elect using *CLOP* or their own machine learning implementations. A complete description of the challenge and a comprehensive analysis of the results are described by Guyon et al. (Guyon et al., 2006c, 2007a,b).

The competition was divided into two stages. The first stage, called *the model selection game* (Guyon et al., 2006c) was focused on the evaluation of pure model selection strategies. In the second, the goal was to evaluate the gain we can have by introducing prior knowledge into the model selection process (Guyon et al., 2007a,b). In this last stage, participants could introduce knowledge of the data domain into the model selection process (prior knowledge track). Also, participants could use *agnostic* methods in which no domain knowledge is considered in the selection process (agnostic track). In both stages of the competition we evaluated models obtained with *PSMS* under different settings. As we will see below, and as we may see in overview papers (Guyon et al., 2006c, 2007a,b), models obtained by *PSMS* were always ranked high in the participants list, showing the competitiveness of *PSMS* for model selection. Our models were ranked $2^{nd}$ in the agnostic track by using *CLOP* objects only. Furthermore, the difference with methods that used prior knowledge was relatively small, showing that *FMS* can be a viable solution for the model selection problem without the need of investing time in introducing domain knowledge, and by considering a wider variety of methods.

In the rest of this section we report results obtained by models selected with *PSMS* in the *ALPK* competition (Guyon et al., 2006c). The data sets used in this contest are described in Table 6, these data sets come from real domains. Data sets used for the agnostic and prior knowledge tracks were different. For the agnostic track the data were preprocessed and dummy features were introduced, while for the prior knowledge track raw data were used, together with a description of the domain. We did not applied *PSMS* to the *NOVA* data set due to its high dimensionality. We should emphasize that, although all of the approaches evaluated in the *ALPK* competition faced the same problem (that of choosing a model that obtain the lowest classification error for the data), such methods do not adopted the *FMS* interpretation. Most of the proposed approaches focused on a fixed machine learning technique like ensembles of trees (Lutz, 2006), or support-vector based methods (Cawley, 2006; Pranckeviciene et al., 2007; Guyon et al., 2007b), and do not took into account feature selection methods. Participants in the prior knowledge track could introduce domain knowledge. Further, most of participants used their own implementations, instead of the *CLOP sandbox*.

| Data set | Domain | Type | Features | Training | Validation | Testing |
|---|---|---|---|---|---|---|
| *Ada* | Marketing | Dense | 48 | 4174 | 415 | 41471 |
| *Gina* | Digits | Dense | 970 | 3153 | 315 | 31532 |
| *Hiva* | Drug discovery | Dense | 1617 | 3845 | 384 | 38449 |
| *Nova* | Text classification | Sparse binary | 16969 | 1754 | 175 | 17537 |
| *Sylva* | Ecology | Dense | 216 | 13086 | 1309 | 130857 |

Table 6: Benchmark data sets used for the model selection challenges (Guyon et al., 2006c, 2007a,b).

| Data | Prep. | Classifier | Parameters | Rank |
|---|---|---|---|---|
| *Ada*$^*$ | slng(1), std(1), norm(1) | NN | $u = 5$, $s = 0.008$, $e = 373$ | $7^{th}$ |
| *Gina* | norm(1) | SVC | $coef = 0.1$, $Kernel = poly$, $d = 5$,$s = 0.01$ | $3^{rd}$ |
| *Hiva* | std(1),norm(1) | **B-***kridge* | $coef = 1$, $Kernel = poly$, $d = 1$, $s = 1$ | $2^{nd}$ |
| *Nova* | norm(1) | **B-***NN* | $u = 1$, $s = 0.2$,$e = 50$ | $3^{rd}$ |
| *Sylva*$^*$ | std(1), norm(1) | NN | $u = 6$,$s = 0.028$, $e = 359$ | $7^{th}$ |

Table 7: Models selected by trial and error and *PSMS* for the model selection game. *slng* is for *shit-scale*, *std* is for *standardize* and *norm* is for *normalize*. Hyperparameters for the neural network classifier (*NN*) are: $u$ = *units*, $s$ = *shrinkage* and $e$ = *epochs* or training iterations. **B** indicates the use of a boosting ensemble. See Tables 1 and 2 for a description of methods and their hyperparameters. Models selected with *PSMS* are marked with $(*)$.

We tried two different *PSMS* configurations for selecting models through the two stages. For the first stage of the competition (Guyon et al., 2006c), we applied *PSMS* in the *ADA* and *SYLVA* data sets for the selection of preprocessing method and hyperparameter optimization of a neural network. Models for *GINA*, *HIVA* and *NOVA* were selected empirically by trial and error. In Table 7 we show the models selected for the first stage of the competition, together with the individual ranking of the models. This configuration of models was ranked $2^{nd}$ at this stage of the challenge, compared to methods that used only the *CLOP* toolbox and $3^{rd}$ overall participants (Guyon et al., 2006c). Note that *PSMS* was run 100 iterations for *ADA* and only 5 iterations for *SYLVA*.

For the second stage of the challenge we ran *PSMS* 100 iterations for *GINA*, *HIVA* and *SYLVA* and 500 iterations for *ADA*; for *NOVA* we used the model showed in Table 7. For these experiments we allowed *PSMS* to select preprocessing and feature selection methods as well as learning algorithm and to perform hyperparameter optimization[10]. Models selected in these settings are shown in Table 8. This was our best ranked entry in the competition. By the March's milestone (an intermediate stage) our run was ranked $3^{rd}$ in the agnostic track of the competition, and $2^{nd}$ by using *CLOP* objects only. For the final ranking this same entry was ranked $5^{th}$ overall agnostic entries and, again, $2^{nd}$ by using *CLOP* objects only. Considering the score of the top 21 entries, including agnostic and prior-knowledge based methods (those entries considered for the final ranking) our run was ranked $8^{th}$ overall.

10. Given the dimensionality of data we restricted the set of methods that *PSMS* could choose from. For preprocessing we dropped *subsample* and *pc-extract*; for feature selection we do not considered *relief* and *svcrfe*; as learning algorithms we just considered *zarbi, naive* and *neural*.

| Data set | Prep. | Models parameters | Train BER | Test BER | Rank |
|---|---|---|---|---|---|
| Ada | std(0),norm(1),slng(0) | $u = 5$, $s = 1.4323$, $b = 0$,$e = 257$ | 17.81 | 18.04 | $6^{th}$ |
| Gina | gs(48),slng(1) | $u = 16$, $s = 0.29$,$b = 1$, $e = 456$ | 2.66 | 6.14 | $9^{th}$ |
| Hiva | std(1),norm(0) | $u = 5$,$s = 3.028$, $b = 0$, $e = 448$ | 6.41 | 28.54 | $2^{th}$ |
| Sylva | std(0),norm(0),slng(1) | $u = 8$, $s = 1.2853$, $b = 0$, $e = 362$ | 0.56 | 0.84 | $2^{th}$ |

Table 8: Models selected with *PSMS* for the second stage of the challenge, this entry was ranked $5^{th}$ in the agnostic track and $8^{th}$ overall. We show training and testing *BER* obtained by each model, as well as the individual rank position for each data set in the agnostic track. See caption of Table 7 for a description of the abbreviations.

| Data set | Prep. | Models parameters | Train BER | Test BER |
|---|---|---|---|---|
| Ada | s2n(47),std(1),norm(0),slng(1) | $k = G$, $s = 1.312$, $G = 4.612$ | 19.30 | 18.01 |
| Gina | s2n(308),std(1),norm(1),slng(1) | $k = G$, $s = 0.23214$, $G = 0.86505$ | 0 | 4.14 |
| Hiva | relief(724),std(1),norm(1) | $k = G$, $s == 0.075666$, $G = 0.99429$ | 0.81 | 28.92 |
| Sylva | s2n(157),norm(0) | $u = 25$, $s = 0.0001$, $b = 1$, $e = 50$ | 3.07 | 2.92 |

Table 9: Models selected with *PSMS* by using subsamples of the available training data (see text). A support vector classifier was obtained for *ADA, GINA* and *HIVA*, and a neural network for *SYLVA*.

Furthermore, one should note that models selected with *PSMS* are very simple, compared with the models selected by other participants that used the *CLOP* package (Reunanen, 2007; Guyon et al., 2007b). This is, also, a desirable property in model selection.

We have stated that the main drawback of our model selection interpretation is the complexity of the problem, in particular, we are concerned with the processing time spend in the search. A simple solution to this problem is using only a subset of the available training data (chosen randomly) for evaluating each particle (model). With this approach processing time is significantly reduced, as the complexity most of the considered methods depends on the size of the data. Furthermore, overfitting can be avoided to some extend. We applied *PSMS* to the *ADA, GINA, HIVA* and *SYLVA* data sets. The number of iterations was fixed to $I = 50$, with a swarm size of $m = 10$ particles. Results obtained with the selected models are shown in Table 9. This entry is ranked immediately below our best entry of the challenge, that from Table 8. The difference in accuracy can be neglected, but we have reduced the processing time dramatically. For example, running *PSMS* for the *ADA* data set for 500 iterations took around 52 hours, with the modification it took around 25 minutes to get a better model, in the same computer. One should note that with the modified method we could consider most of the available models, this would not be possible to do if we were considered the complete data. We should emphasize that this are preliminary results.

The above results show the efficacy of *PSMS* for model selection. Besides its simplicity it has shown comparable performance to those obtained by more sophisticated model selection strategies and by participants that focused on robust machine learning algorithms, such as ensembles, or kernel methods (Lutz, 2006; Reunanen, 2007; Boullé, 2007; Pranckeviciene et al., 2007; Wichard, 2007). Models selected with *PSMS* are simple and yet very competitive, furthermore, with *PSMS* we do not need to have any knowledge on the methods we

can choose from, neither on the domain. Its application to any data set is direct; since we just need to load the data, press the *run* button and wait some time for obtaining a model.

## 5. Conclusions

In this paper we proposed *PSMS*, that is, the application of *PSO* for the *FMS* problem. Given a data set, the *FMS* problem consists of selecting the combination of preprocessing, feature selection and learning algorithm that obtains the lowest classification error. *FMS* also includes hyperparameter optimization for the selected methods. This broader sense interpretation of the model selection problem has the following advantages. First, the generality of the approach allows us to consider different model types (for preprocessing, feature selection and learning) and a variety of methods. Second, *PSMS* can be applied to any data set, since neither domain knowledge nor machine learning knowledge is required, therefore it can be considered a black-box model selection method. Third, and most important, competitive and yet very simple models can be obtained with *PSMS*. The main drawback of this interpretation is the computational cost it involves exploring the vast search space. However, we showed preliminary results with a simple subsampling technique that reduced significantly the processing time.

The simplicity of *PSO* and its proved performance, comparable to that of evolutionary algorithms, make this search algorithm well suited for *FMS*. However, the application of any other stochastic optimization strategy is straightforward. The main advantage of *PSO* is that a single equation for updating solutions is needed, opposed to evolutionary algorithms where methods for representation, mutation, cross-over, speciation and selection should be considered. Experimental results in benchmark data showed superior performance of *PSMS* when compared to random search with *CV*. Also, empirical evidence showed that *PSMS* is less prone to overfitting than random search and has better convergence behavior. These results make *PSMS* a good candidate for the implementation of an any-time *FMS* method. Results obtained by *PSMS* in a model selection challenge showed that this can be a very competitive method, besides its simplicity and generality. In such competition, models selected with *PSMS* were always ranked at the first positions, showing comparable performance to those selected with more sophisticated techniques, and with methods that used domain knowledge.

Current work includes the use of *PSMS* for the selection of members of ensembles. The idea is being tested in another machine learning challenge. We are performing experiments with the subsampling technique and studying its overfitting-avoidance properties. Also, *PSMS* is currently being applied to different tasks, including automatic image annotation and object recognition. Future work comprises a comparison of different stochastic optimization algorithms for the *FMS* problem, in order to bring some light in what type of search methods are specially well suited for this problem.

## Acknowledgments

We would like to thank the organizers and participants of the NIPS multi-level inference workshop and model selection game, and of the *IJCNN ALPK* challenge. The first author thanks the UNIPEN foundation and the NSF for travel support. We also thank editors and anonymous reviewers for their

useful comments that have help us to improve this paper. Finally, we thank Dr. Aurelio López and INAOE for the provided support.

## References

P. J. Angeline. Evolutionary optimization vs particle swarm optimization: Philosophy and performance differences. In *Proceedings of the 7th Conference on Evolutionary Programming*, volume 1447 of *LNCS*, pages 601–610, San Diego, CA, March 1998. Springer.

Y. Bengio and N. Chapados. Extensions to metric-based model selection. *Journal of Machine Learning Research*, 3:1209–1227, 2003.

M. Boullé. Report on preliminary experiments with data grid models in the agnostic learning vs prior knowledge challgenge. In *Proceedings of the 20th International Joint Conference on Neural Networks*, 2007.

G. Cawley. Leave-one-out cross-validation based model selection criteria for weighted ls-svms. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2006)*, pages 2970–2977, Vancouver, Canada, July 2006.

G. C. Cawley and N. L. C. Talbot. Preventing over-fitting during model selection via bayesian regularisation of the hyper-parameters. *Journal of Machine Learning Research*, 8:841–861, April 2007.

M. Clerc and J. Kennedy. The particle swarm: Explosion, stability and convergenge in a multidimensional complex space. *IEEE Transactions on on Evolutionary Computation*, 6(1):58–73, February 2002.

J. Demsar. Statistical comparisons of classfifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, January 2006.

H. J. Escalante, M. Montes, and E. Sucar. Psms for neural networks on the ijcnn 2007 agnostic vs prior knowledge challenge. In *Proceedings of the 20th International Joint Conference on Neural Networks*, Orlando, FL, USA., 2007.

V. Franc and V. Hlavac. The statistical pattern recognition toolbox. http://cmp.felk.cvut.cz/cmp/software/stprtool/index.html, 2004.

Dirk Gorissen. Heterogeneous evolution of surrogate models. Master's thesis, Katholieke Universiteit Leuven, Belgium, June 2007.

Dirk Gorissen, Luciano De Tommasi, Jeroen Croon, and Tom Dhaene. Automatic model type selection with heterogeneous evolution: An application to rf circuit block modeling. In *IEEE Proceedings of WCCI 2008, forthcoming*, 2008.

V.G. Gudise and G.K. Venayagamoorthy. Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium, 2003. (SIS03)*, pages 110–117, 2003.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3(Mar):1157–1182, 2003.

I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, editors. *Feature Extraction, Foundations and Applications*. Series Studies in Fuzziness and Soft Computing. Springer, 2006a.

I. Guyon, A. Saffari, G. Dror, and J. M. Buhmann. Performance prediction challenge. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2006)*, pages 2958–2965, Vancouver, Canada, July 2006b.

I. Guyon, A. Saffari, G. Dror, G. Cawley, and O. Guyon. Benchmark datasets and game result summary. In *NIPS Workshop on Multi-level Inference and the Model Selection Game*, Whistler, Canada, December 2006c.

I. Guyon, A. Saffari, G. Dror, and G. Cawley. Agnostic learning vs prior knowledge challenge. In *Proceedings of the 20th International Joint Conference on Neural Networks*, Orlando, Florida, 2007a.

I. Guyon, A. Saffari, G. Dror, and Gavin Cawley. Analysis of the ijcnn 2007 competition agnostic learning vs. prior knowledge. *Neural Network special anniversary issue*, (forthcoming), 2007b.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Verlag, New York, 2001.

E. Hernández, C. Coello, and A. Hernández. On the use of a population-based particle swarm optimizer to design combinational logic circuits. In *Evolvable Hardware*, pages 183–190, 2004.

Xiaohui Hu, R.C. Eberhart, and Y. Shi. Engineering optimization with particle swarm. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium, 2003. (SIS03)*, pages 53–57, 2003.

C. Hue and M. Boullé. A new probabilistic approach in rank regression with optimal bayesian partitioning. *Journal of Machine Learning Research*, (submitted), 2007.

JMLR. Jmlr track for machine learning open source software. http://jmlr.csail.mit.edu/mloss/, 2007.

J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of the International Conference on Neural Networks*, volume IV, pages 1942–1948. IEEE, 1995.

J. Kennedy and R. Eberhart. *Swarm Intelligence*. Morgan Kaufmann, 2001.

J. Kennedy and R. Mendes. Population structure and particle swarm performance. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002)*, volume 2, pages 1671–1676, 2002.

Y. Kim, N. Street, and F. Menczer. Evolutionary model selection in unsupervised learning. *Intelligent Data Analysis*, 6:531–556, 2002.

S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220(4598): 671–680, 1983.

R. Lutz. Logitboost with trees applied to the wcci 2006 performance prediction challenge datasets. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2006)*, pages 1657– 1660, Vancouver, Canada, July 2006.

S. Mika, G. Rätsch, J. Weston, B. Schölkopf, A. J. Smola, and K.-R. Müller. Invariant feature extraction and classification in kernel spaces. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 526–532, Cambridge, MA, 2000. MIT Press.

T. Mitchell. *Machine Learning*. McGraw-Hill, October 1997.

O. Nelles. *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models*. Springer, 2001.

E. Ozcan and C. K. Mohan. Analysis of a simple particle swarm optimization system. In *Intelligent Engineering Systems Through Artificial Neural Networks*, pages 253–258, 1998.

E. Pranckeviciene, R. Somorjai, and M. N. Tran. Feature/model selection by the linear programming svm combined with state-of-art classifiers: What can we learn about the data. In *Proceedings of the 20th International Joint Conference on Neural Networks*, 2007.

G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for adaboost. *Mach. Learn.*, 42(3):287–320, 2001. ISSN 0885-6125. doi: http://dx.doi.org/10.1023/A:1007618119488.

J. Reunanen. Model selection and assessment using cross-indexing. In *Proceedings of the 20th International Joint Conference on Neural Networks*, 2007.

M. Reyes and C. Coello. Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International Journal of Computational Intelligence Research*, 3(2):287308, 2006.

Y. Robinson, J. Rahmat-Samii. Particle swarm optimization in electromagnetics. *IEEE Transactions on Antennas and Propagation*, 52(2):397– 407, February 2004.

A. Saffari and I. Guyon. Quickstart guide for clop. Technical report, Graz University of Technology and Clopinet, May 2006. http://www.ymer.org/research/files/clop/QuickStartV1.0.pdf.

J. Salerno. Using the particle swarm optimization technique to train a recurrent neural model. In *Proceedings of the Ninth International Conference on Tools with Artificial Intelligence*, pages 45–49, 1997.

Y. Shi and R. C. Eberhart. Parameter selection in particle swarm optimization. In *Evolutionary Programming VII*, pages 591–600, New York, 1998. Springer-Verlag.

Y. Shi and R. C. Eberhart. Emprirical study of particle swarm optimization. In *Proceedings of the Congress on Evolutionary Computation*, pages 1945–1949, Piscataway, NJ, USA, 1999. IEEE.

A. Smola and B. Schölkopf. Software on kernel machines. http://www.kernel-machines.org/software.html, 2006.

S. Sonnenburg. Nips workshop on machine learning open source software. http://www2.fml.tuebingen.mpg.de/raetsch/workshops/MLOSS06/, December 2006.

F. van den Bergh. *An Analysis of Particle Swarm Optimizers*. PhD thesis, University of Pretoria, Sudafrica, November 2001.

F. van der Heijden, R. P.W. Duin, D. de Ridder, and D. M.J. Tax. Prtools: a matlab based toolbox for pattern recognition. http://www.prtools.org/, 2004.

M. Voss and X. Feng. Arma model selection using particle swarm optimization and aic criteria. In *Proceedings of the 15th IFAC World Congress on Automatic Control*, 2002.

J. Weston, A. Elisseeff, G. BakIr, and F. Sinz. The spider machine learning toolbox. http://www.kyb.tuebingen.mpg.de/bs/people/spider/, 2005.

J. Wichard. Agnostic learning with ensembles of classifiers. In *Proceedings of the 20th International Joint Conference on Neural Networks*, 2007.

J. Wichard and C. Merkwirth. Entool - a matlab toolbox for ensemble modeling. http://www.j-wichard.de/entool/, 2007.

I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.

D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 4:67–82, 1997.

H. Yoshida, K. Kawata, S. Takayama Y. Fukuyama, and Y. Nakanishi. A particle swarm optimization for reactive power and voltage control considering voltage security assessment. *IEEE Transactions on Power Systems*, 15(4):1232–1239, Jan 2001.