PSMS for Neural Networks on the *IJCNN 2007* Agnostic vs Prior Knowledge Challenge

H. Jair Escalante, Manuel Montes y Gómez, and Luis Enrique Sucar

***Abstract*— Artificial neural networks have been proven to be effective learning algorithms since their introduction. These methods have been widely used in many domains, including scientific, medical, and commercial applications with great success. However, selecting the optimal combination of preprocessing methods and hyperparameters for a given data set is still a challenge. Recently a method for supervised learning model selection has been proposed: *Particle Swarm Model Selection* (*PSMS*). *PSMS* is a reliable method for the selection of optimal learning algorithms together with preprocessing methods, as well as for hyperparameter optimization. In this paper we applied *PSMS* for the selection of the (*pseudo*) optimal combination of preprocessing methods and hyperparameters for a fixed neural network on benchmark data sets from a challenging competition: the *(IJCNN 2007) agnostic vs prior knowledge challenge*. A forum for the evaluation of methods for model selection and data representation discovery. In this paper we further show that the use of *PSMS* is useful for model selection when we have no knowledge about the domain we are dealing with. With *PSMS* we obtained competitive models that are ranked high in the official results of the challenge.**

I. Introduction

Many supervised learning algorithms have been proposed so far, such as, neural networks and kernel methods [10], [17]. These algorithms, together with preprocessing methods, have been widely used in many domains, including scientific, medical, and commercial applications with great success. However, selecting the best combination of learning algorithm with preprocessing methods, for analyzing a given data set, is still a challenge. Even a harder problem is the estimation of the parameters for the selected model, often referred to as *hyperparameter* optimization. Both difficult problems are known as model selection. Traditionally, application's developers using statistical and learning methods choose algorithms and tune their parameters empirically, commonly by trial and error; or in the best case, by using prior knowledge of experts on the domain. However these methodologies are both impractical and inaccurate.

Model selection is concerned with the automated selection of the (pseudo) optimal model for describing a dataset. We can use prior knowledge of the task at hand for improving the model selection process, or, instead we can develop general purpose model selection algorithms, called *"agnostic"* methods. The phrase *"describing a dataset"* in the context of supervised learning consist of selecting a learning algorithm together with their optimal hyperparameters for a supervised learning task, as well as on the selection of the preprocessing methods that could improve the algorithm's accuracy. In this paper the supervised learning task approached is classification, with the objective of selecting the model that minimizes the misclassification rate.

In this paper the results our participation on the *agnostic versus prior knowledge challenge (IJCNN 2007)* [5], [8] are reported. The method used in this contest is *particle swarm model selection* (*PSMS*) [4]. It is an implementation of a particle swarm optimization (*PSO* [12], [11]) algorithm for dealing with the model selection problem. In *PSMS* models are represented as particles in the search space. Particles fly through this search space using knowledge acquired from previous iterations in order to find the particle (model) that optimizes a given fitness function. *PSO* has been already used for training a neural network, that is adjusting the weights of the learning algorithm [11]. It has also been applied with other algorithms for hyperparameter optimization [19]. However *PSO* has not been applied for selection of preprocessing methods, learning algorithm and hyperparameter optimization of multiple models at the same time. *PSMS* was recently proposed [4], and it has been partially evaluated on the model selection game [6], [9]. In this paper we applied *PSMS* for the selection of preprocessing methods and hyperparameter optimization for a neural network learning algorithm. Experimental results show that *PSMS* is a reliable method for supervised learning model selection when we have no knowledge about the domain we are dealing with.

We focused ourselves on the agnostic track of the challenge, since we believe that agnostic methods are more useful than those using prior knowledge. Mainly in one important respect: agnostic methods can be applied to many data sets and domains with little (if any) changes. Providing general purpose methods that can be used by any person, even when she/he has no knowledge on the domain nor on machine learning. On the other hand, prior knowledge based methods offer, in general, better models for specific tasks. However, this sort of methods require of having both: prior knowledge of the task at hand (an expert on the domain) and knowledge on machine learning methods (an expert on machine learning), resulting on a *long-run* and expensive development process. Nevertheless, accuracy of models obtained by using previous knowledge should improve models selected with agnostic methods. We do not sustain that better methods can be obtained with agnostic methods than with their *gnostic* counterparts. Both approaches have their own advantages and limitations. While agnostic methods offer a wide range of applicability, generality of prior knowledge methods is

H. Jair Escalante, Manuel Montes y Gómez and Luis Enrique Sucar are with the Department of Computer Science, at the National Institute of Astrophysics, Optics and Electronics (INAOE), Tonantzintla, Pue. 72840, Mexico (email: hugojair@ccc.inaoep.mx, {mmontesg,esucar}@inaoep.mx).

limited to a specific domain.

The rest of this document is organized as follows. In the next section, we briefly describe the agnostic vs prior knowledge challenge. In Section III, the *PSMS* algorithm is introduced. Next, in Section IV results of *PSMS* on the competition are presented; we ranked 3^{rd} on the agnostic track of the challenge. Finally, in Section V conclusions and future work directions are discussed.

II. The agnostic vs prior knowledge challenge

The agnostic versus prior knowledge challenge [5] is a competition with the aim of assessing the real added value of using prior knowledge into classification problems. There are two tracks on the challenge: *1) the agnostic track*, which objective is to evaluate models selected by using no knowledge of the task at hand and preprocessed data; and *2) the prior knowledge track*, in which participants are provided with the original (raw) data and information about the origin of data; they are required to obtain the best representation for the data as well as the model that best performs on the selected data representation. This sort of competitions are very useful for comparing methods on both model selection and data representation discovery, by providing models to choose from, data and a fair evaluation.

Details about this contest are further described by Guyon et al [8], in the rest of this section we briefly introduce some aspects of the challenge in order to make this document self contained. The rules of the challenge are quite simple: the organizers provide five datasets for classification, together with a $Matlab^R$ toolbox[1]. Then, the task is to select the model (and data representation in the prior knowledge track) that achieves the lowest *balanced error rate* (Equation (4)) over the five data sets on unseen test data.

The data sets provided for the agnostic track of the challenge [5], are summarized in Table I. The data come from real domains, and were split into separate training, validation and test sets. Given the distribution of instances for the different sets performance on the test set provides objective comparisons among models; note that we can not trust on the performance results obtained on the validation data, since this data set may be not representative of the test set. Labels for the training data were made available to participants but labels of the validation and test sets were not provided. Though competitors could obtain immediate feedback by submitting results on the validation set to the challenge website; performance on the test set is not revealed to competitors until the end of the challenge.

For our participation on the challenge we used the *Challenge Learning Object Package* (*CLOP*), provided by the organizers and publicly available for academic purposes[2]. This $Matlab^R$ toolbox contains feature and attribute selection (among other preprocessing) methods, as well as several machine learning algorithms. This software is object oriented facilitating the implementation of our methods. The available methods in the *CLOP* package are briefly described in Table II. This toolbox provides two grouping methods: chain and ensemble [7], [8]. *Chain* returns a model constructed from an array of learning sub-models (preprocessing, feature selection and learning algorithms). It is a very important object for *PSMS*, since *PSMS* uses this constructor for *translating* particles into *CLOP* models. Another available method is *ensemble* that allows the construction of ensembles by combining and weighting sub-models.

[1]Although it is not mandatory to use the *CLOP* toolbox, the organizers encourage its use in order to provide objective comparisons

[2]http://clopinet.com/ isabelle/Projects/modelselect/Clop.zip

Name	Features	Train	Validation	Test
Ada	48	4174	415	41471
Gina	970	3153	315	31532
Hiva	1617	3845	384	38449
Nova	16969	1754	175	17537
Sylva	216	13086	1309	130857

TABLE I

DATA SETS FOR THE AGNOSTIC TRACK OF THE COMPETITION [8], [5], COLUMNS $3-5$ SHOW THE NUMBER OF OBSERVATIONS FOR TRAINING, VALIDATION AND TESTING

Acronym	Name	# Pars.
	Feature selection	
GS	Gram-Schmidt	1
s2n	S2N, corr. coef.	2
Rlf	Relief	3
Rffs	Random Forest	2
SVREC	SVM Rec. elimination	1
	Preprocessing	
std	Standardize	1
Nmlz	Normalize	1
slng	Scaling	3
PCA	PCA	1
subs	Observations subsample	2
	Learning Algorithms	
Zarbi	Linear classifier	0
kridge	Kridge regression	4
Bayes	Naive Bayes	0
NNs	Neural Network	4
Rf	Random forest	3
SVC	SVM classifier	4
Boost	Boosting	4

TABLE II

AVAILABLE METHODS IN THE *CLOP* PACKAGE

III. Particle swarm model selection

The particle swarm optimization algorithm (*PSO*), was proposed by Kennedy and Eberhart more than a decade ago [12]. *PSO* is a population-based search algorithm that aims to simulate the social behavior of birds within a flock. It was originally proposed for training neural networks [12], [11]. Although it has been also applied to many other optimization problems too. Mainly in problems in which the features are real valued [16].

We decided to use *PSO* instead of other search strategies mainly because *PSO* has outperformed other optimization strategies such as hill climbing methods, simulated annealing and evolutionary algorithms in several domains. Furthermore,

as originally proposed, *PSO* has been used for hyperparameter selection and for training neural networks [19], [12], [11]. In this paper we are going one step further in this direction by using *PSO* for the (*quasi-*)full[3] model selection problem. This task consist of the selection of the (*pseudo-*)optimal combination of preprocessing methods and hyperparameters for a fixed learning algorithm. Note that the search space of this problem is much more larger than the one we have when we want to train a neural network or optimize its parameters only. In consequence we must apply more efficient techniques and be willing to spent more time to converge. Unfortunately we can not present an analysis of complexity and convergence of the algorithm. Though we are currently performing a complexity and convergence analysis of the algorithm for the full model selection problem; as well as a systematic comparison of *PSO* with other standard search strategies for the model selection problem [4].

PSO is inspired on the social behavior of biological societies, in which each individual (*particle*) of the community shares a common goal (*getting food, for example*) with the other members of the population (*swarm*). It is assumed that individuals know how far the goal is, though they do not know the exact position of the goal.

In *PSO* each particle represents a candidate solution to the optimization problem at hand and it is treated as a point in the $d-$dimensional search space. Each particle adjusts its flight (*search direction*) over the search space based on its own previous flight experience and that of its neighbors. As in most heuristic search algorithms, an aptitude measure should defined. The aptitude measure should evaluate the proximity of the candidate solution to the optimum. Particles have memory in the sense that each particle is able to know the best position it has achieved so far. Social behavior in particles allows them to follow leader particles, that is, the global best solutions according the aptitude measure. Particles adjust their flight trajectories using the following equations:

$$v_{i,j}^t = \rho * v_{i,j}^{t-1} + c_1 * r_1 * (p_{i,j} - x_{i,j}) + c_2 * r_2 * (p_{g,j} - x_{i,j}) \quad (1)$$

$$x_{i,j} = x_{i,j} + v_{i,j}^t \quad (2)$$

where ρ is the inertia weight, whose goal is to control the impact of the previous velocities over the current one. $v_{i,j}^t$ is the velocity of the particle i in the j^{th} dimension, at time t. c_1 and c_2 are weights applied to the influence of the best position found so far ($p_{i,j}$) by particle i and by the best particle in the swarm $p_{g,j}$. $r_1, r_2 \in [0, 1]$ are random values with a uniform distribution. After the velocity is updated, the new position of the particle i in its j^{th} dimension is recomputed, see Equation (2). This process is repeated for each dimension of the particle i and for all the particles in the swarm.

Using the above described optimization framework we can map the model selection problem into a particle's environment. In this work we wanted to evaluate the applicability of *PSO* for model selection, that is *PSMS*. As starting point we used the base *PSO* algorithm with standard parameters though different parameters, updating schemas and operators can be used in future experimentation as well [18], [16]. In the following we describe core components of the *PSMS* method, these are: the representation of models as particles and the aptitude measure that we will use to evaluate candidate solutions.

A. Representation

The problem we approached in this paper is that of selecting the best model for minimizing misclassifications on a test set. Therefore for applying *PSMS* in this problem we should represent each candidate model as a particle. In *PSO* particles are represented as $d-$dimensional numerical vectors. In consequence we represented each model λ as a $d-$dimensional vector P_i, as shown on Equation (3).

$$P_i = [M_1, p_{1,1}, \ldots, p_{1,k}, \ldots\ldots M_n, p_{n,1}, \ldots, p_{n,l}] \quad (3)$$

Where each M_j is a binary-valued element whose value (0 or 1) indicates the absence of presence of method j; each entry $p_{m_j,1\ldots t}$ represents the $t-$parameters for method j, these sort of elements can have binary or real values, depending on the methods' parameters. Therefore, in such a 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 M_1. For the full model selection problem with fixed neural network as learning algorithm the elements of the particle representation (using the *CLOP* package) have the following codification:

R_i = [**s2n**, f_{max}, w_{min}, **gs** , f_{max} , **relief**, f_{max} , w_{min} , k_{num} , **svcrfe** , f_{max} , **standarize**, *center*, **normalize** , *center* ,$shift_{n-scale}$, $take_{log}$, **subsample** , p_{num} , *balance* , **rffs** , f_{max} , w_{min} , **pc extract** , f_{max} , **NN** , $units$, $shrinkage$, $balance$, $epochs$]

Where the elements in **bold** are binary valued, representing absence or presence on the method indicated with the label. While the *italic* elements can be both real or binary valued, representing parameters for the method preceding them. **NN** stands for neural network; elements following **NN** are the neural network hyperparameters that we want to optimize. The methods we considered in this work were already presented on Section II, see Table II. For practical reasons we omitted some methods[4]: *PCA, subsample, Rffs* and *Rf*.

Each vector in the above codification is used with the *chain* grouping object of the *CLOP* package to create the model represented by each particle. A typical particle is shown below.

Particle = [1, 4, 0.2461, 0.31, 0.4, 0.786, 6, 0,23, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0.693, 0.577, 0.844, 2, 0, 0, 0, 0]

[3]The full model selection problem would consist of selecting learning algorithm together with its hyperparameters as well

[4]PCA was not considered for reducing computational cost, subsample was omitted because in some experiments the minority class was highly reduced to achieve a zero training error, the rf and rffs methods were not considered because we do not used the R software, though we will use this methods in the future

The CLOP model that the above particle represents is shown below
CLOP-Model:
chain
{
1: s2n (fmax = 4, wmin =0.2461),
2:standardize (center=1),
3:naïve,
4:bias (op=1)
}

B. Aptitude function

An aptitude measure (or function) ($\Psi(P_i) \rightarrow$ **lR**) should return a real value ψ_i for each particle P_i, indicating how far particle P_i is from the optimal solution of the problem at hand. In our case, the goal is to improve classification accuracy of learning algorithms. Therefore, we can use a classification accuracy measure for Ψ. There are several possible options, including mean absolute error, squared root error, recall, precision and area under the ROC curve. However, given that in the challenge [7], [8] a particular measure is used, we adopted it here; although, it would be interesting to test another prediction accuracy measures. The measure used in the challenge is the *balanced error rate* (*BER*), which is the average of the errors on each class for a data set as defined in Equation (4):

$$BER = \frac{E_+ + E_-}{2} \tag{4}$$

Where E_+ and E_- are the misclassifications rates for the positive and negative classes, respectively. Therefore, the aptitude measure could be defined as $\Psi(P_i) = BER_{\lambda_i}$, that is the *BER* obtained by model λ_i with particle representation P_i.

Note that in order to obtain *BER*, or any other evaluation measure, for a given model it should be trained first. Depending on the model complexity and dimensionality of data this process can be very expensive in terms of processing time. In the first experiments performed the *BER* value was obtained from the entire training set. However, comparing models' accuracy using the same data for fitting and assessing the model is not straightforward [10], [14]. Instead, on posterior experimentation we calculated the *BER* using a $k-$cross validation (*CV*) approach . As expected, with this approach the performance of the models is improved when tested on unseen data [4]. Although the computational cost increased with the value of k. This is the main concern with *PSMS*, as with any other search heuristic. Since when the *BER* is obtained from the entire training set and we have $\sigma_{size}-$particles in the swarm and β-iterations of *PSMS* are performed the model is trained and evaluated $(\sigma_{size} * \beta) + \sigma_{size}$ times. While by using *k-CV* the model must be trained and evaluated $(\sigma_{size} * \beta * k) + \sigma_{size}$ times. Even when the evaluation of each of the $k-$folds on *CV* is performed on sets of smaller size ($\frac{N}{k} - 1$, with N the size of the data set) the cost of *PSMS* is increased, though this is a common problem of any other population-based search algorithm. Furthermore, an adequate selection of k can result in speeding up *PSMS*: depending on the complexity of the learning machine and for small values of k, using a $k-CV$ approach can be faster than training the entire data set[5]. In consequence *PSMS* can be less computationally expensive by selecting an appropriate value of k. As we will see in Section IV a value of $k = 2$ represents a trade-off between model complexity, though for experiments using the challenge data sets we used a value of $k = 10$.

Model	NCV	$5-CV$	$10-CV$
Baseline	0	0	0
$PSMS_{2CV}$	61.04	64.47	64.44
$PSMS_{5CV}$	55.94	52.55	59.41
$PSMS_{10CV}$	**71.20**	**71.20**	**71.23**

TABLE III
PERCENTAGE OF *BER* REDUCTION OBTAINED BY THE MODELS SELECTED WITH *PSMS*, FOR DIFFERENT VALUES OF k, ON THE IONOSPHERE DATA SET.

IV. EXPERIMENTAL RESULTS

In order to evaluate the performance of *PSMS* in the model selection task several experiments were performed. We used the framework of the model selection game [6], [9] and the agnostic vs prior knowledge challenge [8]. Note that we are using the base *PSO* in the experiments reported in this paper.

Before showing results of *PSMS* on the challenge data sets, results of experiments on a small benchmark data set are reported. We used the ionosphere data set from the *ML-UCI* repository [1]. This data set contains 351 instances and a dimensionality of 34. We split the data into training (200 examples) and testing (151 examples) sets. In the first experiment we evaluated the performance of *PSMS* for different values of k. *PSMS* was ran for 1000 iterations using the training set, and then we evaluated performance on the test set. The simpler classifier available in the CLOP package was used, that is *zarbi*. The task of *PSMS* is then to select preprocessing methods and hyperparameter optimization for such methods. Results of these experiment for different values of k are shown in Table III.

From Table III we can appreciate the improvement over a *zarbi* classifier without any preprocessing. As we can see improvements on these set are very large. This reduction is more evident when $k = 10$, though $k = 2$ can be used as a trade off between processing time an accuracy. We would expect that $k = 5$ resulted in better models though $k = 2$ performed better. However we must emphasize that this results are illustrative only, and we can not generalize due to the size of the data set.

On a second experiment we allowed *PSMS* to select preprocessing methods, learning algorithm as well as hyperparameter optimization for the full model, that is full model selection. The resulting model is shown in Table IV. With this model a *BER* on the test set of 0.089 was obtained,

[5]As pointed out by reviewers of this paper

Type	Model
Prep.	rel$\{f_{max=10}$,$w_{min=0.45}$,$k_{num=2}\}$,norm$\{c=0\}$
Classifier	svc$\{coef_{0=0}$,$degree=4$,$gamma=0$,$shr.=0.37$

TABLE IV

MODEL SELECTED WITH *PSMS* ON THE FULL MODEL SELECTION SETTING FOR THE IONOSPHERE DATA SET.

Data	Prep.	Classifier	Parameters
Ada^*	slng, std, nmlz	NN	$u=5$,$s=0.008$,$its=373$
$Gina^{**}$	nmlz	SVC	$c=0.1$,$K=p$,$d=5$,$s=0.01$
$Hiva^{**}$	std, nmlz	$kridge_{bost}$	$c=1$,$s=1$,$K=1$
$Nova^{**}$	nmlz	NN_{bost}	$u=1$,$s=0.2$,$its=50$
$Sylva^*$	std, nmlz	NN	$u=6$,$s=0.028$,$its=359$

TABLE V

MODELS SELECTED BY TRIAL AND ERROR ($*$) AND WITH $PSMS(**)$.

Data	Prep.	Parameters
Ada	slng(0), std(1), nmlz(1)	$u=5$,$s=1.43$,$b(0)$,$its=257$
$Gina$	gs(48), slng(1)	$u=16$,$s=0.29$,$b=1$,$its=456$
$Hiva$	std(1),nmlz(0)	$u=5$,$s=3.02$,$b=0$,$its=448$
$Nova$	nmlz	$u=1$,$s=0.2$,$its=50$
$Sylva$	std(0),nmlz(0) slng(1)	$u=8$,$s=1.285$,$b(0)$,$its=362$

TABLE VI

MODELS AND PARAMETERS SELECTED WITH $PSMS$ IN THE AGNOSTIC VS PRIOR KNOWLEDGE COMPETITION. A NEURAL NETWORK ALGORITHM IS FIXED.

Dataset	CV-BER	Ranking
Ada	$18.53+-0.93\%$	4^{th}
Gina	$6.99+-0.38\%$	4^{th}
Hiva	$24.76+-2.03\%$	3^{th}
Nova	$4.44+-0.70\%$	5^{th}
Sylva	$0.70+-0.10\%$	4^{th}

TABLE VII

RANK POSITIONS OF THE MODELS SELECTED WITH *PSMS*, *CV* ACCURACY IS ALSO SHOWN. THE MODELS FOR EACH DATA SET ARE SHOWN IN TABLE VI

which slightly outperformed the best model selected with *PSMS* in the last row of Table III, which obtained a test *BER* of 0.085. Note that the classifier for such a model was the simpler available. This result can be due to the fact that in the full model selection task we have a much more large search space than that we have when a learning algorithm is fixed. It is possible that running *PSMS* for a large number of iterations can result in larger improvements.

A. Results on the challenge data sets

In the model selection game our best entry was ranked 2^{nd} by using *CLOP* objects only, according the game results [8], [6], [9]. The selected models are shown in Table V, we obtained this result by combining models selected by both: *PSMS* for the *ADA* and *SYLVA* data sets and manual trial and error for *GINA*, *HIVA* and *NOVA*. For this stage of the competition we fixed preprocessing methods and we just performed hyperparameter optimization for a neural network. We used $k=5$ when calculating *CV* in the aptitude function and ran the *PSMS* algorithm 100 iterations for *ADA* and only 40 iterations for the *SYLVA* data set. These models were ranked high on that stage of the competition, even when we only performed hyperparameter optimization. Furthermore, this was our first participation on the challenge, while most of the other top-ranked participants already participated the previous year [9], [8], [2], [15], [13], [3]. Models selected with *PSMS* were much less complex than those selected by the cross-indexing method of the game winner [15], [9].

For the agnostic vs prior knowledge competition we allowed *PSMS* to select preprocessing methods and to perform hyperparameter optimization for the full model with a fixed neural network learning algorithm (that is, *quasi*-full model selection). For this experiments $k=10$ in the *CV*. We ran *PSMS* 500 iterations for the *ADA* data set and 100 iterations for the *HIVA*, *GINA* and *SYLVA* data sets. The selected models are shown in Table VI. Note that the model for *NOVA* is the same as in Table V, this due to the fact that applying *PSMS* to such a dummy data representation (bag of words of documents) is impractical. Instead we are currently working on obtaining a smaller representation for this data set and then we will apply *PSMS* on it.

The best entry obtained at this stage of the challenge is *Corrida-final*, with overall score of 0.2857. The rank of each model as well as the cross validation error for each data set is shown is Table VII. The rank of *Corrida-final* is 3^{rd} in the agnostic track and 6^{th} on the general list. Models selected with *PSMS* even outperformed models from participants of last year competition. Furthermore, note that we only considered the methods available in the *CLOP* package. This results on *quasi*-full model selection show evidence that the use of *PSMS* in this task can result in robust models. Moreover, as in Table V, models selected by *PSMS* keep very simple. That is, low complexity models, even when we are not including a complexity *penalizer* for models in *PSMS*. This is a very important result since less complex models are more efficient than their complex counterparts.

In Figure 1 the *BER* value for each iteration of *PSMS* is shown. From this Figure we can appreciate that for all of the data sets a small number of iterations is needed to reach a local optimal solution. For SYLVA only 20 iterations were need for obtaining a good solution, for *HIVA* and *GINA* data sets the best solution was found at iteration 80. For the *ADA* data set the minimum *BER* value was obtained at iteration *250*, though it is very likely that at this point we are overfitting the data set.

Results from this Section show evidence that *PSMS* can be a very useful tool for full model selection. Its performance is competitive with other interesting methods [8], [2], [15], [13], [3]. Moreover, *PSMS* as an agnostic model selection strategy can be used by any user and any domain, even when we have no knowledge of the task at hand.

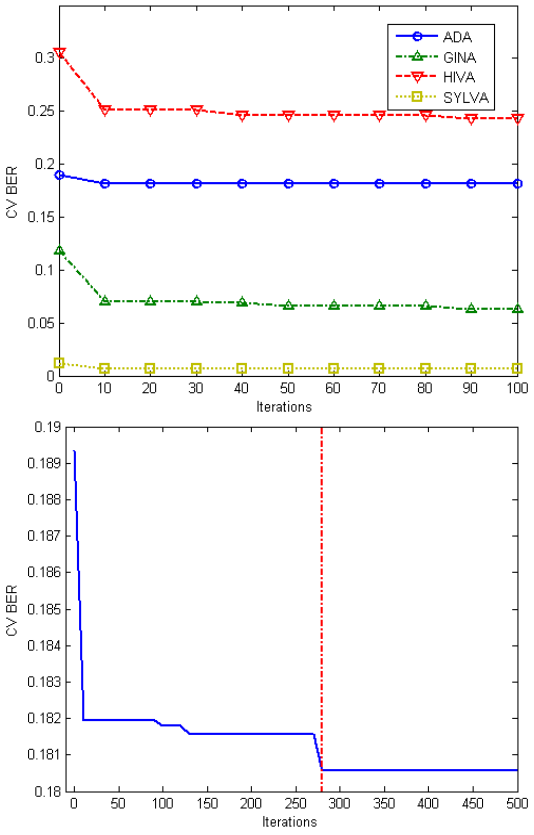

Fig. 1. Top: *BER* vs iterations for *ADA*, *HIVA*, *GINA* and *SYLVA*; bottom: BER vs iterations for the *ADA* data set

V. Conclusions

In this paper we report results of our participation on the *IJCNN2007* challenge. We used a search strategy based on *PSO* for performing *quasi*-full model selection. The aptitude function of the search algorithm is based on the *CV* error. Results in the challenge show that *PSMS* can obtain competitive, yet simple, models for the data sets in which we applied it. Simple models are more useful than their complex counterparts because complexity is directly related to computational cost. Although models are simples, the models found by *PSMS* are competitive. Furthermore, the main advantage of *PSMS* is that it can be used by any user, even when she/he have no knowledge on the task at hand nor in machine learning at all. A concern with the current implementation of *PSMS*, (that is also a common concern for all population-based search algorithms), is that it can be expensive to compute. Since it depends on the models complexity, which in turn it depends on the size and dimensionality of the data set. At the moment we have just performed *quasi*-full model selection, though we believe that the advantages of *PSMS* will be further highlighted when experiments on full model selection being performed.

Currently a systematic analysis of complexity and convergence for *PSMS* is being carried out. A comparison of *PSMS* with evolutionary algorithms, simulated annealing, hill climbing and other widely used search strategies is an immediate step towards evaluating the performance of *PSMS*. Further work directions are the implementation of a multi-objective particle swarm optimization algorithm for model selection [16] and experiments with different parameters for *PSO*.

Acknowledgments: We would like to thank the organizers of the model selection game and the agnostic vs prior knowledge challenge for their motivation and support. Also, we would like to thank professor Aurelio Lopez and the Computer Science Department at INAOE, Mexico for their support.

References

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

[2] Marc Boullé. Report on preliminary experiments with data grid models in agnostic learning vs pk challgenge. In *Proc. of the IJCNN*, 2007.

[3] Gavin Cawley and N. L. C. Talbot. Agnostic learning vs prior knowledge in the design of kernel machines. In *Proc. of the IJCNN*, Orlando, Florida, 2007.

[4] H. Jair Escalante, Manuel Montes, and L. Enrique Sucar. Particle swarm model selection. *Journal of Machine Learning Research*, submitted, Jan 2007.

[5] Isabelle Guyon. Agnostic learning vs. prior knowledge challenge and workshop. http://agnostic.inf.ethz.ch/, 2006.

[6] Isabelle Guyon. Multi-level inference and model selection game, nips workshop. http://clopinet.com/isabelle/Projects/NIPS2006/, 2006.

[7] Isabelle Guyon, Amir Reza Saffari Azar Alamdari, Gideon Dror, and Joachim M. Buhmann. Performance prediction challenge. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2006)*, pages 2958–2965, Vancouver, Canada, July 2006.

[8] Isabelle Guyon, Amir Saffari, Gideon Dror, and Gavin Cawley. Agnostic learning vs prior knowledge challenge. In *Proc. of the IJCNN*, Orlando, Florida, 2007.

[9] Isabelle Guyon, Amir Saffari, Gideon Dror, Gavin Cawley, and Olivier Guyon. Benchmark datasets and game result summary. In *NIPS Workshop on Multi-level Inference and the Model Selection Game*, Whistler, Canada, December 2006.

[10] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Verlag, New York, 2001.

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

[12] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of the International Conference on Neural Networks*, volume IV, pages 1942–1948. IEEE, 1995.

[13] Roman Lutz. Logitboost with trees applied to the wcci 2006 performance prediction challenge datasets. In *Proc. of the IJCNN*, Vancouver, Canada, 2006.

[14] Oliver Nelles. *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models*. Springer, 2001.

[15] Juha Reunanen. Model selection and assessment using cross-indexing. In *Proc. of the IJCNN*, 2007.

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

[17] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

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

[19] 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.