PSO-based Constrained Imbalanced Data Classification

Martin Hlosta, Rostislav Stríž, Jaroslav Zendulka, Tomáš Hruška
Department of information systems
Faculty of Information Technology, IT4Innovations Centre of Excellence
Brno University of Technology Technology
Božetěchova 2, Brno, Czech Republic
Email: {ihlosta,istriz,zendulka,hruska}@fit.vutbr.cz

***Abstract*—The paper deals with classification of highly imbalanced data with accuracy constraints for the minority class. We solve this problem by our proposed meta-learning method that uses cost-sensitive logistic regression to generate initial candidate models. These models can be used as an initial solutions for various optimization algorithms. This paper is aimed for using Particle Swarm Optimization (PSO) to handle the constrained imbalanced classification problem. Experiments, comparing with Genetic Algorithm (GA), show that the swarm intelligence approach is suitable for this problem and outperforms GA.**

***Index Terms*—Data mining, imbalanced classification, constraints, PSO, Genetic Algorithm.**

I. Introduction

Every day, a huge amount of data is generated from various sources. Usually, such amount of data contains some hidden and undiscovered knowledge that could prove useful. Knowledge discovery from databases, sometimes also referred to as data mining, is an approach that is trying to address this issue.

Classification belongs to one of the most used tasks of data mining. Given input data and their attributes, classification algorithm tries to learn and create a model that correctly classifies data into a number of different classes. There is an enormous number of different algorithms capable of learning various types of classification models. Most of these algorithms equally assume that the distribution of classes is balanced. However, in many real-world applications most of the data belong to some majority class and only a small fraction of them to the minority class. Medical datasets are the typical example of this phenomenon, e.g. , cancer tests contain very few records with positive test result compared to all performed tests. The issue with the standard learning algorithms is that they tend to classify all the data to the majority class. Thus, we are unable to reveal cases of the minority class, which is in most cases the more important one. The problem of learning from imbalanced data has been identified as a crucial problem of data mining and machine learning. Many methods suggesting how to deal with data imbalance has been presented, including[1].

Based on our experience, in some applications, accuracy constraints for the minority class can be defined. To give an example, it is desirable to reveal high percentage, e.g., 99 percent, of all rare examples. In this case, it is necessary to modify current algorithms for imbalanced data mining or to develop new ones. Recently, we presented in [2] a method that deals with this problem on a large dataset and results in a model capable of very fast prediction of the new data. This method utilizes both cost-sensitive learning using logistic regression and subsequent optimization using Genetic Algorithm (GA).

In this paper, we present a way of addressing this issue by using different optimization algorithm – Particle Swarm Optimization (PSO), which has been already used in various applications in the field of data mining. The contributions of this paper could be listed as follows:

1) Present Swarm Intelligence approach for solving constrained classification of imbalanced data and other strategies on how to deal with constrained learning.
2) Provide comparison of behavior of these strategies together with comparison of learning from initial weight models from Cost Sensitive Logistic Regression (CS-LR) or starting with random solutions.
3) Show comparison of this approach with the Genetic Algorithm optimization.

The paper is organized as follows. Section II describes the problem of imbalanced learning and evaluation metrics in greater detail. In Section III, we introduce related optimization algorithms. Section IV presents the strategies used for handling constraints in optimization algorithms and their usage for imbalanced classification. Finally, section V provides description of the experiments on the large dataset from our Internet research.

II. Learning from Imbalanced data

As already mentioned, imbalanced data are characterized by a large disproportion of data between the classes. Typically, in the binary classification problem, there is a huge amount of data belonging to majority class and only few instances from the minority class. The imbalance in the data can be either intrinsic or extrinsic, profoundly described in [3]. To deal with the imbalanced data problem, various methods have been published to work either at the *data level*, or at the *algorithmic level*. At the data level, sampling techniques are used; from the algorithmic perspective, cost-sensitive learning, active learning, ensemble methods are used. Other methods can be found in [3].

	Predicted minor	Predicted major
Actual minor	0	> 1
Actual major	1	0

Table I
COST MATRIX FOR MAJORITY AND MINORITY CLASS.

The idea behind sampling methods is to modify the class distribution so that the proportion of instances of each class is balanced. We can distinguish between random and informed methods for over and under-sampling. *Random oversampling* randomly selects a group of minority class instances and duplicates them. In contrast, *random under-sampling* randomly selects a group of majority class instances and removes them from the training set. The over-sampling approach has an advantage in that no information is lost, on the other hand, under-sampling is more time and memory efficient. To improve the quality of sampling, more complex informed sampling methods were presented. The best known is the *Synthetic Minority Oversampling Technique* (*SMOTE*) [4], which takes advantage of both approaches. It creates new artificial instances using k-nearest neighbors algorithm and also uses under-sampling of majority class instances. The drawback of informed methods is that they are less time efficient.

Cost-sensitive learning methods do not modify input data but they penalize the misclassification of minority class instances more than than the instances from the majority class. To achieve this, they utilize the cost-matrix, which contain penalties/costs for all right and wrong classifications. Based on user demands, it is necessary to specify these costs as the input for an algorithm. The example of a cost-matrix, which can be used for binary imbalanced classification, is in Tab. I. It is clear that correct classifications are not penalized and that classifying a minor class instance as major is penalized more than a major class instance as minor. Notice, that in the case of binary classification, it is worth setting only one cost. To perform cost-sensitive learning using existing algorithms, *MetaCost* was presented in [5] as an effective cost-sensitive framework/wrapper. The other way is to incorporate this functionality directly into an algorithm. According to some studies such as [6], cost-sensitive learning beats using sampling methods. On the other hand, some researchers combine both approaches to achieve better results [7], [8].

As in other classification tasks, the approach of building ensemble of various base classifiers has been utilized to improve performance. For example, according to [9], using ensembles can improve performance dramatically.

A. Evaluation

Accuracy and error rate are considered as typical evaluation metrics for classifiers. These measures are irrelevant for an imbalanced data set because they are unable to prioritize the minority class. Most used metrics instead of accuracy are *Area under ROC curve* (*AUC*), and *Geometric mean* (*G-mean*), which is defined as:

$$G{-}mean = \sqrt{TPR \times TNR} = \sqrt{\frac{TP}{TP+FN} \times \frac{TN}{TN+FP}}, \quad (1)$$

where *TP, FP, TN* and *FN* comes from a confusion matrix; i.e., number of True/False Positives and True/False Negatives. TPR and TNR represent the ability of the classifier to recognize the positive and negative class respectively. In the medical application, TPR is often called as *sensitivity* and it refers to the more important class, i.e., in the context of imbalanced data usually the minority class. Similarly, *TNR* is denoted as *specificity.* Because of that and for easier readability we use *sensitivity* for *accuracy on the minority class* and *specificity* for *accuracy on the majority class*.

B. Constrained classification of imbalanced data

As already mentioned in the introduction, there are some applications where there are accuracy constraints imposed on the minority class. For example, given constraint is to have sensitivity at least *0.99,* the task is to find the solution with highest specificity. To deal with such problem, we recently presented a method in [2] based on Cost Sensitive Logistic Regression *(CS-LR)* and optimization using Genetic Algorithm (shortened as *LR/GA*). The method is depicted in Figure 1. First, meta-learning procedure using CS-LR is run to find initial models for the subsequent optimization. It uses various cost settings for CS-LR and threshold moving with heuristic optimizations to find these models faster. After initial models are found, they are given as the input solution for the Genetic Algorithm optimization, which is used to improve the overall quality of the proposed solution. Having the minority class accuracy constraint set to *0.99*, the GA always improved the candidate solution given by the CS-LR for a large input dataset from our Internet research with over 5mil. records. The experiments showed that the use of genetic algorithm improved the final model generated by CS-LR dramatically. Moreover, resulting model of weights enables very fast classification of unknown data in the production environment.

III. STOCHASTIC OPTIMIZATION ALGORITHMS

Many stochastic methods have been utilized for classification in the past. These methods incorporate randomness into the learning. Two stochastic algorithms that have been used for data mining are described in this section, i.e., Genetic Algorithm and Particle Swarm Optimization.

A. Genetic algorithm

Genetic algorithm (*GA*) can be described as a stochastic optimization method based on principle of natural selection and biological evolution. The input problem is encoded into a set of chromosome-like structures, so called *population*, which is then randomly modified by the algorithm using the typical GA operations: 1) *selection* – algorithm simply copies selected chromosomes into the new population, 2) *mutation* – new chromosome is produced by randomly altering parts of the original chromosome within defined bounds (attribute domain,

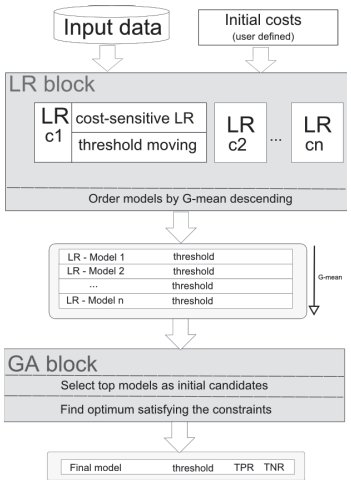

Figure 1. Cost Sensitive Logistic Regression and Genetic Algorithm.

etc.), 3) *crossover* - two chromosomes split into (usually) 2 parts are used to produce 2 new chromosomes by joining the parts together respectively. GA is an iterative algorithm that is trying to produce *improved* population as a result of each iteration. The quality of a population is based on the quality of each chromosome from that population. The quality of the individual chromosome is determined by its *fitness value* that is computed by the algorithm's *fitness function*. The definition of a suitable fitness function is the crucial part of the whole process; the function should be able to evaluate each chromosome with regards to the desired outcome as well as given restrictions.

The use of genetic algorithm for data mining has been proposed in [10] . Its main potential lies in the search performance since the algorithm can be very easily parallelized. Nowadays, genetic algorithm is being used as an optimization technique for other data mining methods, mostly for feature subset selection [11], in hybrid decision structures [12], or for rule induction [13].

B. Particle Swarm Optimization

Opposed to GA, PSO is a typical candidate from the swarm intelligence algorithm family. Swarm intelligence studies the collective behavior of unsophisticated agents that interact locally through their environment [14]. The research takes inspiration from a social behavior of insects such as ants or bees, or flock of birds. PSO was firstly used to simulate birds searching for food. Every bird in a population is denoted as *particle*, and the whole population of particles as *swarm.* The basic idea of PSO is that every particle has its velocity, which is continuously updated based on two factors – 1) towards own personal best solution and 2) towards the best solution of particles in its neighborhood (the neighborhood can be defined in various ways). In the global variant the particle moves toward the best solution of all the particles in the swarm. The initial particles are initialized by random and then, continuous update of their velocities causes the swarm to move towards the desired optimum.

More formally, a new updated position x_i of the particle i is computed as follows:

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{2}$$

where $i = 1, 2, \ldots N$ and N is the size of the population; x_i^t is the old position and $v_i^{t+1}$is the current velocity, which is calculated as:

$$v_i^{t+1} = \chi(\omega v_i^t + c_1 r_{i1}^t (pbest_i^t - x_i^t) + c_2 r_{i2}^t (gbest^t - x_i^t)) \tag{3}$$

where $pbest$ and $gbest$ are personal and global best solution; χ is a *constriction factor* used to control velocities; ω is the *inertia factor*; $c_1$and $c_2$are two learning factors, called *cognition* and *social* learning rate respectively; r_{i1}^t and r_{i2}^t are uniformly distributed random numbers between 0 and 1.

The PSO has been used in data mining for various purposes. For classification, in [15] an algorithm for mining classification rules based on PSO was proposed. Recently, it has been used in [8] for solving imbalanced data problem, where PSO optimized the input parameters for Cost Sensitive SVM. More usages of PSO and other swarm intelligence methods can be found in [16].

Besides PSO, there are also two known algorithms from the swarm intelligence family that are used for data mining – Ant Colony Optimization (ACO) algorithm and Artificial Bee Colony.

IV. Modifications for constrained imbalanced learning

Optimization methods described in the section III are intended for searching unconstrained optimum. This section describes the modifications for the constrained classification imbalanced data.

A. Genetic algorithm

As already mentioned, it is the fitness function what determines the course of genetic algorithm. The most straightforward way of employing class accuracy constraints is , naturally, to modify the fitness function accordingly. The general pattern we use in our application is fitness value fv defined as follows:

$$\begin{aligned} fv &= (Acc_{minor} * C_1 + Acc_{major} * C_2) * Constr \\ &\quad + (Acc_{minor} * C_3 + Acc_{major} * C_4), \end{aligned} \tag{4}$$

where Acc_{minor}, Acc_{major}denotes minority and majority class accuracy respectively; $Constr \in \{0, 1\}$; $Constr = 1$ when the given accuracy constraint is met, otherwise $Constr = 0$;C_1, C_2, C_3, C_4 are user-defined constants that determine the importance ratio between class accuracies, in some cases some of them can be 0, which depreciates the related class accuracy. This fitness pattern proved functional during performed experiments while still being flexible enough to leave some space for user-tuning and customization so the results are as desired.

B. Particle Swarm Optimization

In a general optimization problem, there are usually mechanisms for dealing with constraints. In [17], feasible initial solutions for PSO were generated by random. Unfortunately, in some cases this approach has a high computation cost. This method is slightly similar to our solution in a way that we also feed the initial swarm population with some feasible or almost feasible solutions. In contrast to their method, we do not generate it randomly but with the use of Cost Sensitive Logistic Regression. Other researchers in [18] tried to handle the problem with small population size. We dealt with the constraints using two strategies – 1) the penalty function approach, which is based on the work of [19] and 2) using strategy with modified updating.

1) Penalty function: The idea of this approach is penalizing the solutions violating given constraints. In a maximization task, the penalty value is subtracted from the original objective function and the resulting *penalty function* becomes a new objective. According to [19], the penalty function is defined as:

$$F(x) = f(x) - h(k)H(x), \qquad x \in S \subset \mathbb{R}^n, \tag{5}$$

where $f(x)$ is the original objective function, $h(k)$ is dynamically modified with the current algorithm iteration k and $H(x)$ is the penalty factor, defined as:

$$H(x) = \sum_{i=0}^{m} \theta(q_i(x)) q_i(x)^{\gamma(q_i(x))}, \tag{6}$$

where $q_i(x) = \max\{0, g_i(x)\}$, $i = 1, \ldots, m$ and is denoted as *relative violated function*; $\theta(q_i(x))$ is *multi-stage assignment function*; $\gamma(q_i(x))$ the power of the penalty function and $g_i(x)$ are the constraints in the form:

$$g_i(x) \leq 0, \quad x \in S \subset \mathbb{R}^n. \tag{7}$$

For our purpose, we have only one constraint: $sensitivity > minSensitivity$ and following the given form $minSensitivity - sensitivity \leq 0$. The functions $h(.)$,$\theta(.)$ and $\gamma(.)$ are problem dependent. In experiments we used $h(k) = 1$, causing that the penalty factor is still the same. For the two other functions we adapted and modified the strategy that was recommended in [20] and [19]. The $\gamma(q(x))$ is computed as:

$$\gamma((q(x)) = \begin{cases} 1 & \text{if} \quad 0 < q(x) < 1 \\ 2 & \text{if} \quad q(x) >= 1 \end{cases} \tag{8}$$

The $\theta((q(x))$ function is defined as follows:

$$\theta((q(x)) = \begin{cases} 5 & \text{if} \quad 0 < q(x) < 0.001 \\ 10 & \text{if} \quad 0.001 \leq q(x) \leq 0.1 \\ 50 & \text{if} \quad 0.1 < q(x) \leq 1 \\ 100 & \text{if} \quad q(x) >= 1 \end{cases} \tag{9}$$

Recall that $q(x) \geq 0$ and, as expected, for $q(x) = 0$ the penalty $H(x)$ is always 0. To sum up, the optimization process is the same as for PSO described in the previous section, the only difference is the modified objective function with a penalty.

2) Strategy with modified updating: As opposed to the previous unchanged optimization process, this approach slightly modifies updating strategy of the personal best solution of a particle. Typically, the personal best solution is updated, if the new particle has better value of an objective function. In our case, this means greater specificity. However, the update does not account for constraints so the personal best is replaced by the new one if at least one of the following condition is true:

1) $gbest.Sensitivity < minSensitivity \wedge x_{i+1}.Sensitivity > pbest.Sensitivity$ – the global best doesn't satisfy the sensitivity constraint and the new particle has greater sensitivity than its personal best.
2) $x_{i+1}.Specificity > pbest.Specificity \wedge (x_{i+1}.Sensitivity \geq minSensitivity \vee gbest.Sensitivity \geq minSensitivity)$ – the particle has greater specificity than its best, and either the particle or the best neighbor satisfies the constraint.

Here, the dot notation is used for the measure of a particle, thus $gbest.Sensitivity$ is a sensitivity of the global best particle. At first, the algorithm tries to reach a solution that satisfies the conditions. After such solution is found, particles follow other particles with greater specificity.

V. Experiments

In our experiments, we focused on comparing the behavior of the two strategies used for constrained optimization and with Genetic Algorithm. Also, we provide the comparison of learning from random initial weights to initial models from Cost Sensitive Logistic Regression (CS-LR) used in [2].

Experiments were performed on a large data set from our Internet security research. The data has two highly imbalanced classes with the ratio 1:34 between a minority and a majority class. The dataset contains more than $5\,000\,000$ data cases described by 120 binary attributes. The goal was to reach a model with the highest possible specificity (accuracy on the majority class) and satisfying the sensitivity constraint $minSensitivity = 0.99$.

We provide two types of evaluations. The first was one performed with widely used cross-validation and the second one shows, where the algorithms converge after reasonable number of iterations. In both types of evaluation, three different types of initial candidates were given as the input of the optimization algorithms:

1) Solutions with highest G-mean from CS-LR, denoted as *Uncons.* (unconstrained) in tables,
2) solutions with highest G-mean satisfying the $minSensitivity$ constraint, denoted as *Cons.*, and
3) uniformly distributed random weights, denoted as *Rand.*

In all experiments, we set the size of the swarm in PSO to $n = 70$ and population $n = 5000$ for GA. Moreover, we present a behavior of the strategies in a graphic form to demonstrate the growth of objective functions in the first 300 iterations. In all the following tables, *GA* stands for Genetic Algorithm, *PSO 1* is PSO with the penalty strategy and *PSO 2* the strategy with modified updating.

A. Cross-validation

The performance of a classifier is typically evaluated with the cross-validation (CV). Here we performed 5-fold stratified CV, where each fold has the same class ratio, which is necessary for imbalanced data. First, 1000 iterations of optimization algorithm were performed and then the resulting model was evaluated using testing data. Because of the stochastic character of the algorithms, we also ran the whole CV procedure 5 times with the same folds. We took the the mean and the standard deviation of all the objective function results and also mean and standard deviation of the maximum solutions for each fold. The results are in Tab. II.

We can see that best results were achieved by penalty strategy for PSO given initial models satisfying the *minSensitivity* constraint. This stands for both mean and the maximum. Genetic algorithm performed slightly worse. Updating penalty had worst results, especially when given models satisfying the *minSensitivity* constraint. Only initial models with the greatest G-mean achieved mean value above 0.55.

Comparing strategies, we can conclude that PSO 1 had better results with Cons. initial models, PSO 2 with the Cons. GA has very similar results for both initial models. Starting from randomly generated initial solutions, all the approaches achieved lower objective function values than with initial models from CS-LR. The greater standard deviation value for starting from random solution shows that the solutions are much less stable.

It is worth mentioning that the sensitivity constraint was narrowly violated in some measurements but did not drop under 0.9899.

B. Training and testing on the same dataset

This part shows, how good solutions the algorithms were able to produce during the optimization. In the terms of classification evaluation, this means evaluating on the training data set. Similarly as for cross validation, for each method, the mean, the best and also the worst solution achieved in 10 runs were obtained and reported in Tab. III.

The results show that the values are higher than in cross-validation, which was expected. For almost all cases the values slightly correlate with the previous evaluation. The difference is PSO 2 strategy, which performed significantly better than in cross validation. It is also clear that in all cases starting from initial weights from CS-LR leads to better results than from random ones.

C. Behavior of algorithms

To demonstrate different behavior of the approaches, we provide the following graphs with objective function value in the first 300 iterations. For each approach, it. GA, PSO1, PSO, there is separate figure each containing the three types of initial solutions. The first graph on the Fig. 2 shows the GA, the Fig. 3 the PSO 1 and the Fig. 4 the PSO 2.

The GA and PSO 1 has similar behavior for all the initial solutions. For Cons., the algorithm utilizes that the solution already satisfies the constraint and only grows, with steeper

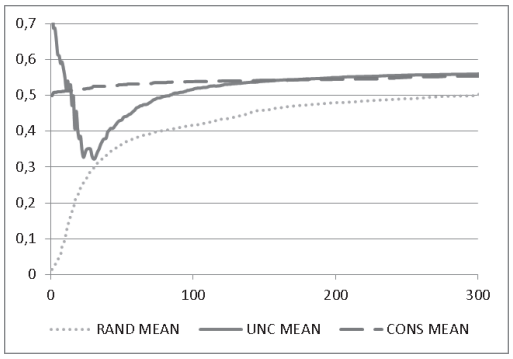

Figure 2. First 300 iterations of GA.

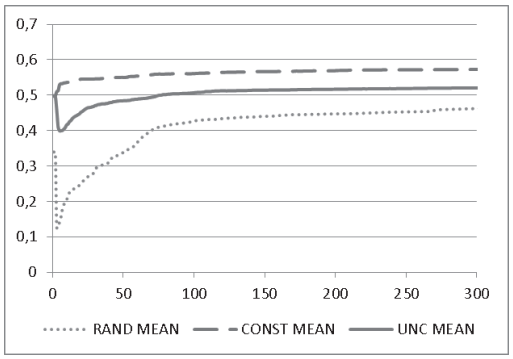

Figure 3. First 300 iterations of PSO with modified update strategy – PSO1.

growth in PSO 1. For Uncons. the objective function falls rapidly in the beginning because the initial solutions do not satisfy the constraint. PSO 1 begin to rise earlier than GA. For Rand. we can observe faster growth in the objective function for GA.

Starting from random solution, the modified version of updating strategy (PSO 2) has similar behavior as the other two, only the growth is much slower. For Cons. it very quickly gain local maximum but it is unable to find other solution. On the other hand, for Uncons. it grows slower but outperforms the Cons. initial models.

We also experimented with the neighborhood of particles. The local setting with 10 closest neighbors worked better compared to the taking the whole population.

VI. Conclusions

In this paper we presented a way how PSO stochastic optimization algorithm can be used for constrained classification of highly imbalanced data. Two strategies for constrained optimization together with Genetic Algorithm were compared in experiments performed on the large dataset. Experiments showed that the penalty function approach outperformed the GA and together with the initial weights provided by CS-LR, it can converge fast to a very good solution. Cross-validation also showed that the constraint is violated only narrowly while provided unknown data.

	Mean			Maximum		
	GA	PSO1	PSO2	GA	PSO1	PSO2
Uncons.	0.5661 ± 0.0049	0.5641 ± 0.0081	0.5596 ± 0.0029	0.5682 ± 0.0039	0.5679 ± 0.0058	0.5618 ± 0.0024
Cons.	0.5536 ± 0.0055	$\mathbf{0.5834 \pm 0.0037}$	0.4988 ± 0.0001	0.5595 ± 0.0042	$\mathbf{0.5855 \pm 0.0018}$	0.4989 ± 0.0000
Rand.	0.5452 ± 0.0127	0.4849 ± 0.0388	0.4970 ± 0.0276	0.5618 ± 0.0091	0.5353 ± 0.0165	0.5271 ± 0.0137

Table II
THE RESULTS FROM THE CROSS-VALIDATION.

	Mean			Maximum			Minimum		
	GA	PSO 1	PSO 2	GA	PSO 1	PSO 2	GA	PSO 1	PSO 2
Uncons.	0.5672 ± 0.0029	0.5469 ± 0.0130	0.5804 ± 0.0091	0.5700	0.5603	**0.5930**	0.5620	0.5286	0.5679
Cons.	0.5548 ± 0.0003	$\mathbf{0.5817 \pm 0.0074}$	0.5373 ± 0.0046	0.5601	0.5871	0.5426	0.5412	0.5666	0.4916
Rand.	0.5469 ± 0.0179	0.5506 ± 0.0190	0.5290 ± 0.0235	0.5680	0.5748	0.5566	0.5200	0.5221	0.4923

Table III
THE RESULTS FROM THE EVALUATION WITH TRAINING AND TESTING ON THE SAME DATASET.

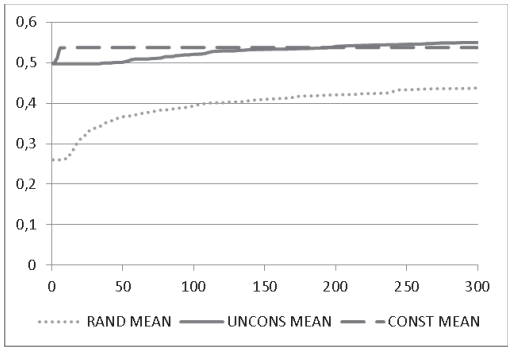

Figure 4. First 300 iterations of PSO with modified update strategy – PSO2.

In the future work, we would like to focus on utilization of unlabeled data with methods of semi-supervised learning. It would be also useful to obtain more data sets and explore how these optimization algorithms and strategies work with such data.

ACKNOWLEDGMENT

This work has been supported by the research funding TAČR TA01010858, BUT FIT grant FIT-S-11-2, the Research Plan No. MSM 0021630528 and the IT4Innovations Center of Excellence CZ.1.05/1.1.00/02.0070.

REFERENCES

[1] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Editorial: special issue on learning from imbalanced data sets," *SIGKDD Explor. Newsl.*, vol. 6, no. 1, pp. 1–6, Jun. 2004.

[2] M. Hlosta, R. Stríž, J. Kupčík, J. Zendulka, and T. Hruška, "Constrained classification of large imbalanced data by logistic regression and genetic algorithm," *Int'l Journal of Machine Learning and Computing*, vol. 2013, no. 3, pp. 214–218, 2013.

[3] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. on Knowl. and Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *J. Artif. Int. Res.*, vol. 16, no. 1, pp. 321–357, Jun. 2002.

[5] P. Domingos, "Metacost: a general method for making classifiers cost-sensitive," in *Proc. of the 5th ACM SIGKDD*, ser. KDD '99. New York, NY, USA: ACM, 1999, pp. 155–164.

[6] X.-Y. Liu and Z.-H. Zhou, "The influence of class imbalance on cost-sensitive learning: An empirical study," in *Proc. of the Sixth Int'l Conf. on Data Mining*, ser. ICDM '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 970–974.

[7] N. Thai-Nghe, Z. Gantner, and L. Schmidt-Thieme, "Cost-sensitive learning methods for imbalanced data," in *IJCNN*. IEEE, 2010, pp. 1–8.

[8] P. Cao, D. Zhao, and O. R. Zaïane, "An optimized cost-sensitive svm for imbalanced data learning," in *Proc. of Advances in Knowledge Discovery and Data Mining, PAKDD 2013*. Springer, 2013, pp. 280–292.

[9] Y. Sun, M. S. Kamel, A. K. C. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Pattern Recogn.*, vol. 40, no. 12, pp. 3358–3378, Dec. 2007.

[10] T. Jun-shan, H. Wei, and Q. Yan, "Application of genetic algorithm in data mining," in *ETCS'09*, vol. 2. IEEE, 2009, pp. 353–356.

[11] J. Yang and V. Honavar, "Feature subset selection using a genetic algorithm," in *Feature extraction, construction and selection*. Springer, 1998, pp. 117–136.

[12] D. R. Carvalho and A. A. Freitas, "A hybrid decision tree/genetic algorithm method for data mining," *Information Sciences*, vol. 163, no. 1, pp. 13–35, 2004.

[13] E. Noda, A. A. Freitas, and H. S. Lopes, "Discovering interesting prediction rules with a genetic algorithm," in *Evolutionary Computation - CEC 99*, vol. 2. IEEE, 1999.

[14] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm intelligence: from natural to artificial systems*. New York, NY, USA: Oxford University Press, Inc., 1999.

[15] Z. Wang, X. Sun, and D. Zhang, "A pso-based classification rule mining algorithm," in *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence, Third Int'l Conf. on Intelligent Computing, ICIC 2007.*, ser. Lecture Notes in Computer Science, vol. 4682. Springer, 2007, pp. 377–384.

[16] A. Abraham, C. Grosan, and V. Ramos, *Swarm intelligence in data mining*. Berlin; New York: Springer, 2006.

[17] X. Hu and R. Eberhart, "Solving constrained nonlinear optimization problems with particle swarm optimization," in *World Multiconference on Systemics, Cybernetics and Informatics SCI'02*, 2002, pp. 203–206.

[18] J. C. F. Cabrera and C. A. C. Coello, "Handling constraints in particle swarm optimization using a small population size," in *Proc. Mexican Int'l. Conf. on Advances in artificial intelligence*, ser. MICAI'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 41–51.

[19] K. E. Parsopoulos and M. N. Vrahatis, "Particle swarm optimization method for constrained optimization problems," in *In Proc. of the Euro-Int'l. Symposium on Computational Intelligence*. IOS Press, 2002, pp. 214–220.

[20] J.-M. Yang, Y.-P. Chen, J.-T. Horng, and C.-Y. Kao, "Applying family competition to evolution strategies for constrained optimization," in *Proc. of the 6th Int'l. Conf. on Evolutionary Programming VI*, ser. EP '97. London, UK, UK: Springer-Verlag, 1997, pp. 201–211.