A Memetic Algorithm to Select Training Data for Support Vector Machines

Jakub Nalepa
Institute of Informatics
Silesian University of Technology
Akademicka 16
44-100 Gliwice, Poland
jakub.nalepa@polsl.pl

Michal Kawulok
Institute of Informatics
Silesian University of Technology
Akademicka 16
44-100 Gliwice, Poland
michal.kawulok@polsl.pl

ABSTRACT

In this paper we propose a new memetic algorithm (MASVM) for fast and efficient selection of a valuable training set for support vector machines (SVMs). This is a crucial step especially in case of large and noisy data sets, since the SVM training has high time and memory complexity. The majority of state-of-the-art methods exploit the data geometry analysis, both in the input and kernel space. Although evolutionary algorithms have been proven to be very efficient for this purpose, they have not been extensively studied so far. Here, we propose a new method employing an adaptive genetic algorithm enhanced by some refinement techniques. The refinements are based on utilizing a pool of the support vectors identified so far at various steps of the algorithm. Extensive experimental study performed on the well-known benchmark, real-world and artificial data sets clearly confirms the efficacy, robustness and convergence capabilities of the proposed approach, and shows that it is competitive compared with other state-of-the-art techniques.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*; I.2.0 [**Artificial Intelligence**]: General

General Terms

Algorithms

Keywords

Memetic algorithm; self-adaptation; support vector machines; training data selection

1. INTRODUCTION

Support vector machines (SVMs) [5] is a supervised classifier which has been successfully applied to a variety of pattern recognition problems. However, its main limitation lies in high training complexity, which makes it virtually inapplicable in case of huge training sets. SVM training is a constrained quadratic programming (QP) problem of $O(n^3)$ time and $O(n^2)$ memory complexity, where n is the cardinality of the training set. During training, SVMs determine a hyperplane that linearly separates two classes in higher-dimensional kernel spaces, and it is later used to classify the data. The hyperplane is defined by the *support vectors* (SVs), which are selected from the entire training set.

The number of SVs is usually small compared with the number of vectors in the training set. Taking that into account, some attempts have been made to reduce the training sets and use only those samples, from which the SVs are predicted to be selected. State-of-the-art techniques are focused either on random selection or analysis of the data geometry in the input or kernel spaces. An alternative approach is to use a genetic algorithm (GA) for selecting the training data [11]. It is an effective method, but requires that the size of the refined set is given prior to the GA optimization. In practice, it is difficult to determine the optimal size, and the GA should be run for various sizes of the reduced set.

Our contribution lies in introducing a new memetic algorithm (MASVM) to select a good training set. We introduce a concept of the support vectors pool, in which valuable samples identified so far are stored. They are used to enhance other individuals for guiding the search. We propose to generate *super individuals* representing training sets composed of SVs only. A new selection scheme is incorporated to balance the exploitation and exploration of the solution space. Since MASVM adapts its parameters, including the training set size, it is not necessary to determine it *a priori*. The experiments show that MASVM is very robust and competitive compared with other state-of-the-art techniques.

The paper is organized as follows. State of the art regarding training set reduction techniques is outlined in Section 2. The details of the proposed memetic algorithm are detailed in Section 3, and the results obtained in our experimental study are presented and discussed in Section 4. Conclusions and directions for our future work are given in Section 5.

2. RELATED LITERATURE

There were numerous approaches proposed over the years towards dealing with large data sets in SVM training. Their purpose was to decrease the training time or to make it feasible in some cases, but also to select valuable data from noisy or poorly labeled data sets. The initial efforts were aimed at reducing the time complexity of the QP optimization by

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.

http://dx.doi.org/10.1145/2576768.2598370.

splitting it into a number of sub-problems [10]. Although this substantially reduces the training time, the size of many real-world data sets is too large to make this method applicable. It concerns various domains, including bioinformatics, genomics, document categorization, and more [25]. In such cases, the solution is to select a subset of training samples from the entire set, and train SVM using the reduced set.

There exist approaches for dealing with large sets based on approximating the answer of a non-linear kernel machine [23, 24]. Here, the input data are mapped into a low-dimensional randomized feature space, and fast linear learning methods are applied to estimate the answer of the corresponding non-linear kernel machine. The approximation of the decision function by means of multiplying the input with a Gaussian random matrix and applying a non-linearity was the basis of the recently proposed *Fastfood* algorithm [13].

A simple, yet effective, approach to reduce a training set size is to sample a subset of a large set randomly [2]. This method was the basis for other sampling algorithms, including reduced support vector machines (RSVMs) [14], later applied and enhanced in many works [3, 7]. RSVMs select a small subset randomly to build a thin rectangular kernel matrix instead of the full kernel matrix, which significantly reduces the training computational complexity. It means that the SVs are chosen from a small random subset, but the optimization is performed using the entire training set.

Randomized methods proved to be quite effective, however they ignore the relations between the training set vectors. Active sampling procedures emerged to take advantage of the data geometry. Crisp clusters were applied to reject samples, which are positioned inside the clusters formed by the vectors with the same class label. The vectors inside the uniformly-labeled clusters are seldom determined as SVs [12]. The k-means clustering and an iterative process were used to reduce the number of misclassified samples [4]. Hierarchical clustering is applied as the initial step of sample reduction by data structure analysis (SR-DSA) [30], which was used for comparisons in the research reported here. It is executed for each class separately, and the interior samples from each cluster are removed based on the Mahalanobis distance in the kernel space. The exterior samples that are far from the opposite class clusters are discarded, since they should not influence the decision boundary [26].

Some methods use alternative clustering techniques to analyze the data geometry. In [31], convex hulls embedding the training data are determined, and the samples are selected using Hausdorff distance between the convex hulls of opposite classes. A method based on the convex-concave hull analysis was recently discussed [16]. In [33], the points from the training set are interpreted as a graph and subject to β-skeleton algorithm. It enables to reduce both training and testing time while being almost as effective as utilizing the entire set. The minimum enclosing and the smallest enclosing ball with a ring region analyses were applied in [29,32].

There are methods which predict the location of the separating hyperplane and select those vectors for RSVM training that are positioned close to it. This hyperplane may be estimated based on the samples heterogeneity measured by entropy [28] or using preliminary classification based on Mahalanobis distances [1]. Training sets can also be selected using active learning, which operates on a large unlabeled set, and dynamically determines labels for the samples [19]. It was shown that these approaches select the labels near the hyperplane [27]. Overall, the mentioned geometry-based methods are dependent on the training set size, which is a significant drawback in case of very large data sets.

Evolutionary algorithms (EAs) have not been extensively explored for selecting SVM training sets. In our earlier work we proposed a genetic algorithm (GA) [11], which consists in evolving a population of N solutions that define small training sets. Each solution, i.e., a *chromosome*, represents a subset of $2K$ samples from the entire training set. The population is successively improved in the biologically-inspired algorithm, in which chromosomes are selected, crossed-over and mutated. The *fitness* of each individual corresponds to the classification accuracy. The most important shortcoming of the GA is an unclear selection of the chromosome size ($2K$) to ensure the proper convergence speed, exploration and exploitation capabilities. We addressed this problem recently [21] using an adaptive approach. Here, the GA process is initiated with a set of individuals of a small chromosome size, which is subsequently increased.

Memetic algorithms (MAs), also referred to as hybrid genetic algorithms, are built upon the population-based approach. They combine EAs to explore the solution space with local refinement procedures applied for expoiting the solutions already found [18]. Due to their high convergence capabilities and wide practical applicability, a number of MAs have been proposed for solving various pattern recognition and optimization problems [8, 9, 15, 17, 20].

3. MEMETIC ALGORITHM

In this section we elaborate on the proposed MA to select training set for SVM (MASVM). We discuss the concept of the support vectors pool $\boldsymbol{P}$, containing valuable samples used in refining solutions and creating *super individuals* (SIs) that are composed of SVs only, and are likely to be well-fitted. A new adaptive selection scheme balancing the exploration and exploitation of the solution space is presented.

3.1 Algorithm outline

Let $\boldsymbol{T}$ denote the entire training set containing $t = t_A + t_B$ samples ($t_A \neq t_B$ e.g., for imbalanced sets), each belonging to one out of two classes, C_A and C_B, respectively. We aim at reducing the entire set $\boldsymbol{T}$ and finding a refined set $\boldsymbol{T}'$ with $t' = t'_A + t'_B$ valuable samples, where $t' \ll t$. Clearly, t' is not known *a priori* and should be determined carefully.

3.1.1 Chromosomes

In MASVM (Alg. 1), each individual (*chromosome*) p_i, $i \in \{1, 2, \ldots, N\}$, represents a refined training set $\boldsymbol{T}'$. The initial population of N individuals is drawn using random sampling (line 1). A new individual p_i contains $t^{(i)}$ (which is independent from t, and $t^{(i)} \ll t$) distinct samples taken from the entire training set $\boldsymbol{T}$ (see Fig. 1). Initially, the same number of samples from both classes are selected ($t_A^{(i)} = t_B^{(i)} = K$ thus $t^{(i)} = 2K$, where K is small at the beginning and increases during the MA execution) to avoid biasing $\boldsymbol{T}'$ with vectors from the more numerous class. Importantly, the MA is not dependent on the training set size t, which may be enormously large for real-life data, and making SVMs inapplicable for such sets in practice. Finding a fitness η of an individual p_i consists in training the SVM using the refined set $\boldsymbol{T}'$, and determining the area under receiver operating characteristic curve (AUC) for the entire set $\boldsymbol{T}$.

Algorithm 1 A memetic algorithm to select refined SVM training set (MASVM).

1: Initialize population of size N; ▷ Random sampling
2: $c_s \leftarrow 0$; $p_B \leftarrow$ **null**; $\boldsymbol{P} \leftarrow \emptyset$;
3: $\eta_I \leftarrow 0$; $\eta_P \leftarrow 0$; $\eta_B \leftarrow 0$;
4: *regenerate* $\leftarrow$ **false**; ▷ Re-generation condition
5: **repeat**
6: Determine N pairs (p_a, p_b); ▷ Pre-selection
7: **for all** (p_a, p_b) **do**
8: $p'_c \leftarrow$ Crossover(p_a, p_b, K);
9: $p''_c \leftarrow$ Compensate$(p'_c, \boldsymbol{T}, K)$;
10: $p'''_c \leftarrow$ Educate$(p''_c, \boldsymbol{P})$; ▷ With probability $\mathcal{P}_e$
11: $p_c \leftarrow$ Mutate(p'''_c); ▷ With probability $\mathcal{P}_m$
12: $\eta(p_c) \leftarrow$ FindFitness$(p_c, \boldsymbol{T})$;
13: **if** $(\eta(p_c) > \eta_B)$ **then**
14: $\eta_B \leftarrow \eta(p_c)$; $p_B \leftarrow p_c$; ▷ Update best fitness
15: **end if**
16: **end for**
17: Update support vectors pool $\boldsymbol{P}$;
18: Generate and assess $\alpha \cdot N$ *super individuals*;
19: Form the next generation G; ▷ Post-selection
20: $\Delta_T \leftarrow \eta_B - \eta_I$; ▷ Improvement after increase
21: $\Delta_R \leftarrow \eta_B - \eta_P$; ▷ Recent improvement
22: **if** $(c_s < \mathcal{T}_s)$ **or** $(\Delta_R \geq 0.5 \cdot \Delta_T)$
23: **or** $(\varrho_{SV}(p_B) < \mathcal{T}_{SV})$ **or** $(\Delta_T = 0)$ **then**
24: $c_s \leftarrow c_s + 1$; ▷ K is not increased
25: **else if** (LGA mode is local) **then**
26: Set LGA mode to global;
27: **else**
28: $\delta_K \leftarrow 1 + |\varrho_{SV}(p_B) - \mathcal{T}_{SV}| \,/\, (1 - \mathcal{T}_{SV})$;
29: $K \leftarrow \delta_K \cdot K$; $\eta_I \leftarrow \eta_B$; $c_s \leftarrow 0$;
30: Set LGA mode to local;
31: **end if**
32: $\eta_P \leftarrow \eta_B$;
33: *regenerate* $\leftarrow$ CheckRegenerationCondition();
34: **if** (*regenerate*) **then**
35: Re-generate population;
36: **end if**
37: **until** StopCondition();
38: **return** p_B with the highest η_B; ▷ $t' = t^{(B)}$

3.1.2 Selection

The population of solutions evolves in time. First, N pairs of individuals from the *i-th* generation G_i are selected (Alg. 1, line 6) to create the $(i+1)$-*th* generation G_{i+1} using a new *local-global adaptation* (LGA) scheme. In this scheme, the individuals are sorted according to their η values. Afterwards, the population is divided into two parts: one containing $\epsilon \cdot N$ well-fitted individuals, and the other $(N - \epsilon \cdot N)$ less-fitted ones. Finally, $\epsilon \cdot N$ pairs (p_a, p_b) are selected from the fittest individuals along with the $(N - \epsilon \cdot N)$ pairs drawn from the less-fitted part of the population (the local mode). Thus, both parts of the population are exploited independently. Finally, if the MA reaches the steady state (e.g., the improvement of the best individual is neglectable), the parent p_a is selected from the fittest part, whereas p_b from the other one to increase the exploration capabilities (the global mode). The mode can be changed back to exploit a promising part of the solution space. Here, we incorporated the LGA into the proposed MA and compared its performance with our multi-parent selection (AMPC) [21], which outperformed well-known high-low fit (HLF) pre-selection scheme [6]. The latter was proved to be asymptotically best for optimizing SVM training sets [11].

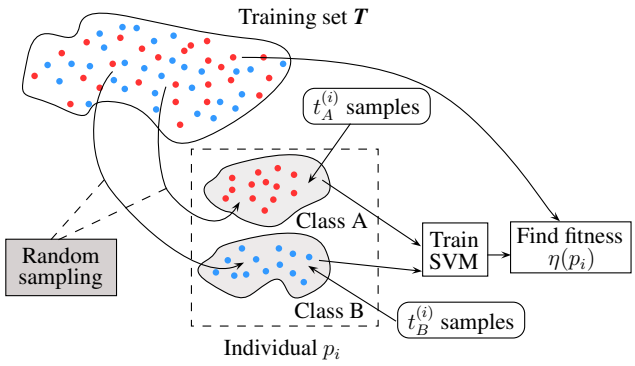

Figure 1: Creation and evaluation of an individual p_i in MASVM.

3.1.3 Crossover

For each pair (p_a, p_b), a new individual p_c is generated using crossover (line 8) with compensation (line 9) if necessary (see Fig. 2). The individuals p_a and p_b create an intermediate child p'_c with $(t^{(a)} + t^{(b)})$ inherited vectors. The size of p_c is determined randomly $(\max(t^{(a)}, t^{(b)}) \leq t^{(c)} \leq 2K)$, and the number of samples is equal for both classes. It is easy to see that the sum of distinct parent samples may be less than $t^{(c)}$. In this case, the compensation (Alg. 1, line 9) is performed to select remaining samples from $\boldsymbol{T}$ randomly.

3.1.4 Education and mutation

In the education procedure (line 10), each p''_c sample which was not selected as a SV during the last SVM training is treated as a "weak sample". Each weak sample is substituted by a SV drawn from the support vectors pool $\boldsymbol{P}$ with the probability $\mathcal{P}_e$. Clearly, only those vectors from $\boldsymbol{P}$ that are not included into p''_c already are considered. p'''_c is mutated with the probability $\mathcal{P}_m$. It consists in random selection of the p'''_c vectors from both classes and substituting them with new samples drawn from the entire set $\boldsymbol{T}$. The fitness of the final p_c is found as discussed earlier (line 12), and the best fitness in the population η_B is updated if necessary (line 14).

3.1.5 Super individuals

Once the children are generated, the newly determined SVs are added to the support vectors pool $\boldsymbol{P}$ (line 17). Then, we generate and assess $\alpha \cdot N$, $\alpha < 1$, SIs (line 18). They contain at most K samples from each class that were selected as SVs and reside in $\boldsymbol{P}$. The number of samples from each class is not necessarily the same in case of the SIs, since the number of class A and B vectors kept in $\boldsymbol{P}$ may be different (and less than K). The individuals from the set $\boldsymbol{P}$, those from the current generation G_i, and the children are used to form the next generation G_{i+1} (line 19). The best N solutions out of $(2N + \alpha \cdot N)$ ones are chosen.

3.1.6 Self-adaptation

After the post-selection, it is verified whether the number of samples K is to be increased (line 23). In order to bal-

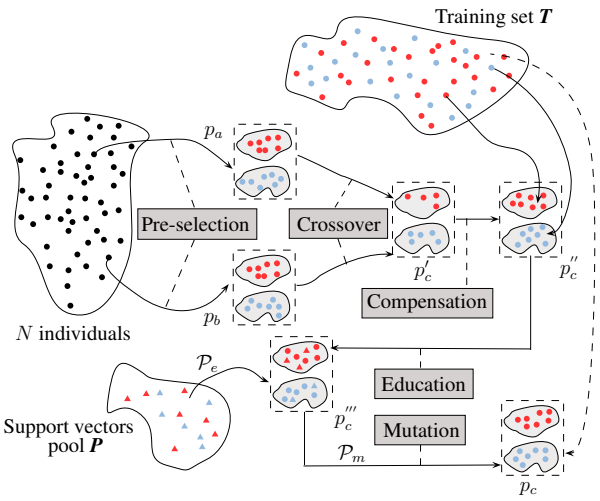

Figure 2: Creation of a child p_c in MASVM.

ance the exploitation and exploration of the solution space, we assume that it can be increased once in $\mathcal{T}_s$ consecutive generations. Also, K is not changed, if the fitness did not grow since the last increase ($\Delta_T = 0$), or the fitness improvement Δ_R was significant (i.e., greater than $0.5 \cdot \Delta_T$). If the LGA mode is local, then it is swapped to global (line 26) before K is updated to further explore the solution space.

The increase factor δ_K is determined based on the ratio $\varrho_{SV} = s/t'$, where s is the number of SVs in the best individual p_B, and $t' = t^{(B)}$ (line 28). Clearly, if this ratio is relatively small (below a threshold $\mathcal{T}_{SV}$), then the current refined set size has not been exploited. Once the value of K is updated (line 29), the LGA mode is set back to local for intensive exploitation of the population (line 30).

3.1.7 *Re-generation*

To diversify the search, we re-generate the population if necessary (line 35), based on analyzing the change of the best individual's fitness η_B and the population diversity ϕ [21]. Here, $r \cdot N$ best individuals are copied, and $(N - r \cdot N)$ ones are drawn randomly from $\boldsymbol{T}$ to keep N constant. The MA may be stopped when the fitness does not grow in a given number of consecutive generations, K is not further increased or the desired fitness is reached.

4. EXPERIMENTAL VALIDATION

Extensive experiments were conducted to compare the performance of the proposed MA with the following state-of-the-art techniques: 1) random sampling [2], 2) data structure analysis method (SR-DSA) [30], 3) genetic algorithm (GASVM) [11], and 4) adaptive genetic algorithm (AGA) [21]. The considered data sets include: 1) *Adult* benchmark set from the UCI repository[1], 2) *Skin* – real-world set derived from the ECU skin image database [22], and 3) *2D* – an artificially generated set of 6371 points positioned on a 500×500 surface. This set of points was used for both training and validation. *Skin* set consists of the training set $\boldsymbol{T}$ with $4 \cdot 10^6$ pixels in the RGB color space and the validation set $\boldsymbol{V}$ with 97560 pixels. For *Adult*, we omitted the samples with missing data, and the set was split into training $\boldsymbol{T}$ (15082 vectors) and validation $\boldsymbol{V}$ (15080 vectors) sets. The presented scores were obtained for $\boldsymbol{V}$. The validation sets were not used during the MASVM set selection for *Adult* and *Skin*.

[1] `http://archive.ics.uci.edu/ml/datasets.html`

Table 1: Settings used for different data sets.

Data set ↓	γ	C	τ (seconds)
2D	100	10	30
Skin	1	10	300
Adult	0.1	0.1	300

4.1 Experimental settings

In this study, we used the LIBSVM for SVM implementation, while the remaining methods were implemented in C++. The experiments were run on an Intel Xeon 3.2 GHz computer with 16 GB RAM. We used SVMs with the RBF kernel: $K(\boldsymbol{u}, \boldsymbol{v}) = \exp\left(-\gamma \|\boldsymbol{u} - \boldsymbol{v}\|^2\right)$. The SVM parameters (i.e., C and γ) were set relying on a grid search with an exponential step [21]. They are summarized in Tab. 1, along with the maximum MASVM execution time τ. Its parameters were set as follows: $N = 10$ (population size), $K = 10$ (initial number of samples per class), $\mathcal{P}_m = 0.3$ (mutation probability), $\mathcal{P}_e = 0.3$ (education probability), $\epsilon = 0.5$ (LGA selection coefficient), $\mathcal{T}_s = 3$ (minimal number of generations without the K increase), $r = 0.1$ (re-generation coefficient), S–10 re-generation (see Section 4.3), $\alpha = 0.2$ (SIs generation coefficient). N and K are relatively small to decrease the MA execution time. We used $\mathcal{T}_{SV} = 0.25$ (SVs ratio) to balance the exploitation and exploration of the solution space (increasing $\mathcal{T}_{SV}$ causes converging to local minima).

In the hierarchical clustering (the first step in the SR-DSA), the distance between clusters to merge increases with the decrease of the number of clusters [30]. It is worth to note that the selection of the SR-DSA parameters is not trivial and strongly influences its performance, especially in case of imbalanced data sets. The following numbers of clusters were used: $c = 255$ for *2D*, $c = 505$ for *Adult*, and $c = 4048$ for *Skin*. Once the clusters are found, $\lambda|M_i|$ samples are removed from each cluster M_i (along with the samples that are distant from the other class vectors), where $|M_i|$ is the number of its samples. In this study, we used $\lambda = 0.8$ for *2D* set, $\lambda = 0.92$ for *Adult* set, and $\lambda = 0.9995$ for *Skin* set.

4.2 Analysis and discussion

In this section we present the results obtained using the proposed MASVM along with the state-of-the-art techniques. All tests for GASVM [11], AGA [21] and MASVM were run 30 times and terminated after τ seconds (see Tab. 1). Their execution time includes the SVM training time for each individual to find its fitness. Contrary to that, in case of the random sampling and SR-DSA, this training is omitted.

In Figs. 3–5, we compare AUC obtained for *Adult*, *Skin* and *2D* data sets for g_M generations ($g_M = 100$, $g_M = 40$ and $g_M = 60$, respectively). The plots present the AUC value of the best individual (averaged over 30 runs) along with its standard deviation. We investigated the performance of two variants of GASVM for each data set, termed GASVM–K, where K is the number of samples per class in the refined set $\boldsymbol{T}'$. Since this value is not modified during the GASVM execution, it should be carefully selected beforehand [21]. We compare the efficacy of the approach for

two values of K: 1) $K = 10$ (which is the initial K value in MASVM), and 2) $K = \bar{t}'/2$ ($\bar{t}'$ is the average size of the refined set of the best individual obtained using MASVM).

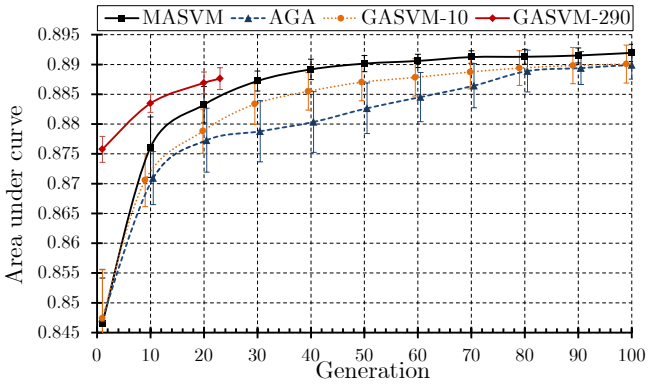

Figure 3: AUC obtained using different methods for *Adult* data set.

It is clear that the best initial AUC is strongly dependent on the training set size (see GASVM–290, GASVM–24 and GASVM–253 in Figs. 3–5). However, it can be noted that GASVM has been terminated after a relatively small number of generations, which indicates a high computational cost in case of large K's (the execution time exceeded τ). Although MASVM, which utilizes the adaptive approach for K tuning, starts from a relatively poor AUC, it is much faster and converges to the initial GASVM (with large K) results rapidly. The desired K is unknown *a priori*, and selecting an improper, i.e., too large, K will drastically increase the execution time of GASVM. Thus, it would have to be run multiple times to determine a reasonable size of $\boldsymbol{T}'$ [11]. This issue is resolved in the AGA and MASVM by the adaptive setting of K according to the current search state. It is easy to see that GASVM with small K has large exploitation capabilities, but lacks the exploration abilities (it converges fast to the steady state). Since MASVM balances the exploitation and exploration, it delivers the best asymptotic results among all investigated techniques.

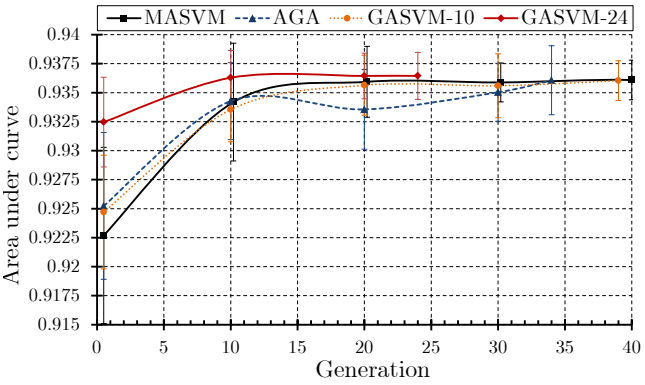

Figure 4: AUC obtained using different methods for *Skin* data set.

Selecting valuable training vectors is crucial for noisy data sets. The results obtained for *Skin* set using the AGA

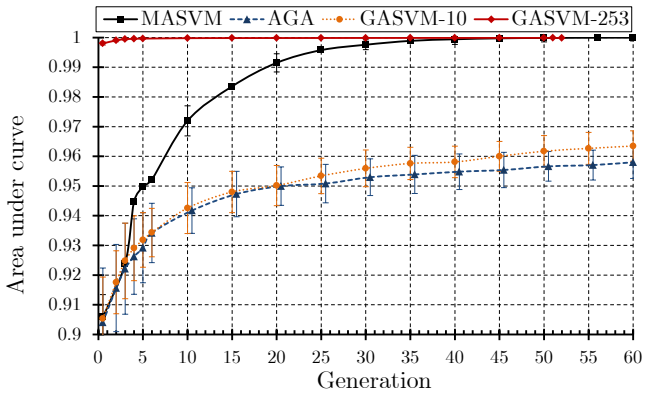

Figure 5: AUC obtained using different methods for *2D* data set.

(Fig. 4) show that the performance may be affected by misleading vectors (see generations 10–30), but the adaptive approach copes well with this problem and converges to the high-quality results. K is increased by a larger factor in AGA than in MASVM, and getting rid of "bad" samples takes more time. The same issue is illustrated in Fig. 6, showing the AUC obtained by random sampling. The bars indicate the minimal and maximal AUC values for various sizes of $\boldsymbol{T}'$. Its performance substantially drops for *Skin* set when t' is large. It confirms the noisiness of the data.

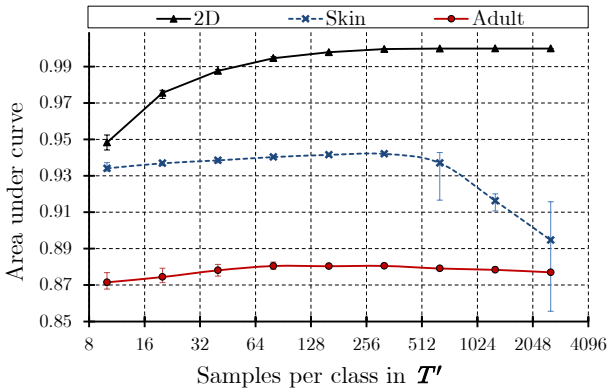

Figure 6: AUC obtained using random sampling for various numbers of samples per class in $\boldsymbol{T}'$.

In Fig. 7, we visualized the exemplary refined training sets obtained by SR-DSA, GASVM, AGA and MASVM. Black and white points indicate the $\boldsymbol{T}$ vectors, and those marked with white and black crosses show the data selected to the set $\boldsymbol{T}'$ (the colors are swapped for clarity). MASVM gives the best AUC and maximizes the ratio ϱ_{SV} in $\boldsymbol{T}'$.

In Tab. 2, we compare the detailed results obtained using the evaluated algorithms. Also, we trained the SVM using $\boldsymbol{T}$. It was not possible in the case of *Skin* set because of its size. MASVM outperformed other state-of-the-art algorithms and delivered the largest AUC scores. It is easy to see that the number of support vectors s is kept small without lowering the score. Thus, it allows for faster SVM classification. The ϱ_{SV} values are slightly smaller in case of MASVM compared with the AGA. The higher AUC for

Table 2: Comparison of AUC (in %) obtained using various methods.

Set		SVM	SR-DSA [30]	GASVM (t_1/t_2) [11]	AGA [21]	**MASVM**
Adult	AUC	88.7607	82.7171	$88.7532 \pm 57.24 \cdot 10^{-4}$/ $88.7508 \pm 18.25 \cdot 10^{-4}$	$88.7580 \pm 56.05 \cdot 10^{-4}$	**89.0854** $\pm 19.63 \cdot 10^{-4}$
	t'	15082	751	20/290	36	290 ± 101
	s	7536	369	20/290	36	281 ± 95
	ϱ_{SV}(%)	49.97	49.13	100/100	100	96.90
	g	–	–	195/22	94	50
	τ (s)	71	16.2	300/300	300	300
Skin	AUC	–	92.4454	$93.5859 \pm 22.14 \cdot 10^{-4}$/ $93.6365 \pm 16.91 \cdot 10^{-4}$	$93.6285 \pm 33.11 \cdot 10^{-4}$	**93.6471** $\pm 23.47 \cdot 10^{-4}$
	t'	$4 \cdot 10^6$	58	20/48	20	47 ± 14
	s	–	4	$17 \pm 2/38 \pm 5$	17 ± 3	37 ± 12
	ϱ_{SV}(%)	–	6.70	85/79.17	85	78.72
	g	–	–	38/23	34	25
	τ (s)	–	162.99	300/300	300	300
2D	AUC	99.9980	99.5733	$96.8038 \pm 46.04 \cdot 10^{-4}$/ $99.9906 \pm 26.18 \cdot 10^{-6}$	$97.8285 \pm 51.71 \cdot 10^{-4}$	**99.9999** $\pm 24.22 \cdot 10^{-7}$
	t'	6371	573	20/506	38 ± 6	506 ± 72
	s	279	142	$20/184 \pm 6$	36 ± 6	239 ± 11
	ϱ_{SV}(%)	46.50	24.78	100/36.36	94.74	47.23
	g	–	–	302/26	50	59
	τ (s)	0.6	1.07	30/30	30	30

MASVM shows that the training set is increased too slowly in case of the AGA. It is confirmed by the average number of generations g, which indicates that the small sized individuals were exploited too long by the AGA. It should be noted that SR-DSA executes quite fast and delivers reasonable scores for each considered data set. However, if its parameters are not selected very carefully (in a time-consuming process) then the performance drastically decays.

4.3 Sensitivity analysis on method components

Here, we analyze the role of several MASVM components and assess their impact on the average AUC and number of SVs found in the refined training sets obtained using the proposed MA. The investigated MASVM variants are summarized in Tab. 3. The population diversity ϕ is measured by the metric proposed in [21]. The s_ϕ value indicates the number of consecutive generations with the same ϕ.

Table 3: Abbreviations of the MASVM variants.

No–R	MASVM without the re-generation
CCR	Re-generation applied after each increase of K
DR–0.03	Re-generation applied if $\phi < 0.03$
DR–0.05	Re-generation applied if $\phi < 0.05$
S–5	Re-generation applied if $s_\phi = 5$
S–10	Re-generation applied if $s_\phi = 10$
AMPC	MASVM with the AMPC selection
No–SI	MASVM without the super individuals
No–E	MASVM without the education procedure
E–30%	MASVM with $\mathcal{P}_e = 0.3$
E–50%	MASVM with $\mathcal{P}_e = 0.5$

In Tab. 4, we present the results obtained for *Adult* data set sampled in various time moments τ. As the baseline for this investigation we assume $\mathcal{P}_e = 0.8$, DR–0.05 re-generation and the LGA selection scheme (the other settings as described in Section 4.1). The results confirm that too frequent re-generation of the population affects the MASVM exploitation capabilities (see CCR, DR–0.03 and DR–0.05). Also, in case of the AMPC variant the solutions are not intensively optimized. On the other hand, too intensive exploitation of a small subset of training samples may cause getting stuck in the local minima (see E–50% compared with No–E). Clearly, balancing both the exploitation and exploration of the solution space, and utilizing the available knowledge determined so far, result in the best AUC scores (E–30%). Fig. 8 shows the average number of support vectors visualized in various time intervals. It is easy to note that the number of training samples increased very fast in case of the AMPC variant. This highlights its exploration tendency, since the small sets were not analyzed properly and were quickly increased. It is also worth to mention that tuning the MASVM parameters may slightly change its efficacy, and setting them incorrectly will not degrade its performance significantly. Thus, MASVM is robust for various (not necessarily optimal) values of its parameters.

5. CONCLUSIONS AND FUTURE WORK

In this paper we proposed a new memetic algorithm for SVM training set selection (MASVM), which adapts itself according to the current search state, and exploits the available knowledge. It is capable of determining the size of a valuable training set in short time, and balances the exploitation and exploration of the solution space. We introduced a concept of the *super individuals* composed of the SVs only, and the support vectors pool, in which the valuable training samples are stored. They are used for enhancing other individuals to guide the search. Also, MASVM copes well with various (not necessarily optimal) parameters settings. An extensive experimental study clearly confirms the competitiveness of the proposed approach.

Our ongoing works are focused on combining MASVM with the data structure analysis for local search and the

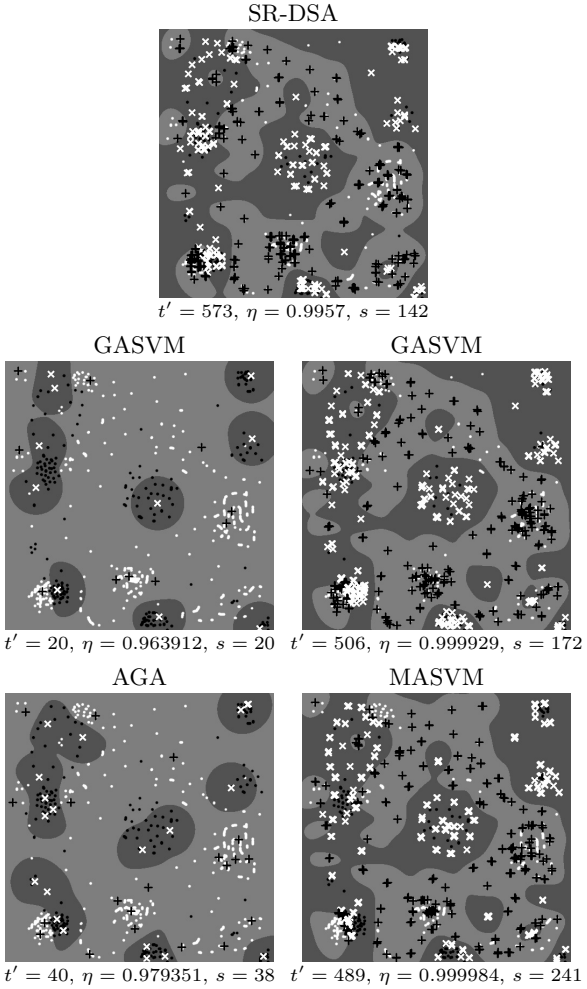

Figure 7: Training set selected using SR-DSA [30], GASVM [11], AGA [21] and MASVM.

initial assessment of training samples. Also, we aim at implementing the parallel MASVM for further decreasing the execution time, and evaluating MASVM on more data sets.

6. ACKNOWLEDGMENTS

This work has been supported by the Polish Ministry of Science and Higher Education under research grant no. IP2012 026372 from the Science Budget 2013–2015. The work was performed using the infrastructure supported by POIG.02.03.01-24-099/13 grant: "GeCONiI–Upper Silesian Center for Computational Science and Engineering".

7. REFERENCES

[1] S. Abe and T. Inoue. Fast training of support vector machines by extracting boundary data. In *Proc. Int. Conf. on ANNs*, pages 308–313. Springer, 2001.

[2] J. L. Balcázar, Y. Dai, and O. Watanabe. A random sampling technique for training support vector machines. In *Proc. ALT*, pages 119–134. Springer, 2001.

[3] C.-C. Chang, H.-K. Pao, and Y.-J. Lee. An RSVM based two-teachers-one-student semi-supervised learning algorithm. *Neural Networks*, 25:57–69, 2012.

Table 4: AUC (in %) and s in time points τ (in seconds) for *Adult* data set.

	τ	AUC	s
No–R	0	$84.9053 \pm 76.52 \cdot 10^{-4}$	20
	120	$88.7224 \pm 27.16 \cdot 10^{-4}$	122 ± 37
	240	$88.8838 \pm 22.43 \cdot 10^{-4}$	192 ± 73
	360	$88.9680 \pm 19.90 \cdot 10^{-4}$	248 ± 90
	600	$89.0527 \pm 19.06 \cdot 10^{-4}$	362 ± 126
CCR	0	$84.8515 \pm 88.16 \cdot 10^{-4}$	20
	120	$88.3508 \pm 20.43 \cdot 10^{-4}$	98 ± 24
	240	$88.5961 \pm 18.21 \cdot 10^{-4}$	171 ± 42
	360	$88.6822 \pm 19.96 \cdot 10^{-4}$	248 ± 66
	600	$88.8230 \pm 14.45 \cdot 10^{-4}$	367 ± 78
DR–0.03	0	$84.6004 \pm 81.18 \cdot 10^{-4}$	20
	120	$88.6934 \pm 25.21 \cdot 10^{-4}$	107 ± 38
	240	$88.8770 \pm 22.08 \cdot 10^{-4}$	181 ± 79
	360	$88.9368 \pm 25.00 \cdot 10^{-4}$	225 ± 109
	600	$88.9914 \pm 25.48 \cdot 10^{-4}$	305 ± 146
DR–0.05	0	$85.0717 \pm 10.10 \cdot 10^{-3}$	20
	120	$88.5571 \pm 25.06 \cdot 10^{-4}$	111 ± 36
	240	$88.7672 \pm 19.96 \cdot 10^{-4}$	192 ± 78
	360	$88.8533 \pm 18.62 \cdot 10^{-4}$	270 ± 99
	600	$88.9116 \pm 17.35 \cdot 10^{-4}$	381 ± 139
S–5	0	$84.4017 \pm 10.87 \cdot 10^{-3}$	20
	120	$88.6967 \pm 24.14 \cdot 10^{-4}$	119 ± 37
	240	$88.8635 \pm 19.55 \cdot 10^{-4}$	200 ± 74
	360	$88.9419 \pm 19.39 \cdot 10^{-4}$	270 ± 106
	600	$89.0218 \pm 18.33 \cdot 10^{-4}$	393 ± 144
S–10	0	$84.8044 \pm 80.72 \cdot 10^{-4}$	20
	120	$88.6699 \pm 33.03 \cdot 10^{-4}$	129 ± 42
	240	$88.8490 \pm 28.43 \cdot 10^{-4}$	213 ± 86
	360	$88.9642 \pm 26.61 \cdot 10^{-4}$	286 ± 122
	600	$89.0450 \pm 22.75 \cdot 10^{-4}$	352 ± 156
AMPC	0	$84.7725 \pm 91.62 \cdot 10^{-4}$	20
	120	$88.5639 \pm 22.07 \cdot 10^{-4}$	162 ± 55
	240	$88.7418 \pm 19.96 \cdot 10^{-4}$	277 ± 95
	360	$88.8085 \pm 20.53 \cdot 10^{-4}$	361 ± 124
	600	$88.8793 \pm 18.62 \cdot 10^{-4}$	452 ± 147
No–SI	0	$84.8012 \pm 99.60 \cdot 10^{-4}$	20
	120	$\mathbf{88.7303} \pm 25.94 \cdot 10^{-4}$	154 ± 39
	240	$88.8755 \pm 19.77 \cdot 10^{-4}$	228 ± 60
	360	$88.9460 \pm 19.20 \cdot 10^{-4}$	297 ± 98
	600	$89.0499 \pm 22.73 \cdot 10^{-4}$	420 ± 147
No–E	0	$84.8580 \pm 95.82 \cdot 10^{-4}$	20
	120	$88.7162 \pm 24.33 \cdot 10^{-4}$	134 ± 27
	240	$\mathbf{88.9087} \pm 20.98 \cdot 10^{-4}$	234 ± 59
	360	$88.9739 \pm 20.71 \cdot 10^{-4}$	305 ± 93
	600	$89.0561 \pm 18.25 \cdot 10^{-4}$	394 ± 123
E–30%	0	$84.6679 \pm 97.00 \cdot 10^{-4}$	20
	120	$88.6598 \pm 26.44 \cdot 10^{-4}$	120 ± 35
	240	$88.8911 \pm 20.71 \cdot 10^{-4}$	231 ± 75
	360	$\mathbf{89.0041} \pm 18.53 \cdot 10^{-4}$	284 ± 84
	600	$\mathbf{89.0789} \pm 19.95 \cdot 10^{-4}$	362 ± 124
E–50%	0	$84.8592 \pm 11.17 \cdot 10^{-3}$	20
	120	$88.6674 \pm 22.35 \cdot 10^{-4}$	140 ± 34
	240	$88.8710 \pm 22.34 \cdot 10^{-4}$	245 ± 76
	360	$88.9452 \pm 20.40 \cdot 10^{-4}$	321 ± 124
	600	$89.0575 \pm 20.85 \cdot 10^{-4}$	410 ± 170

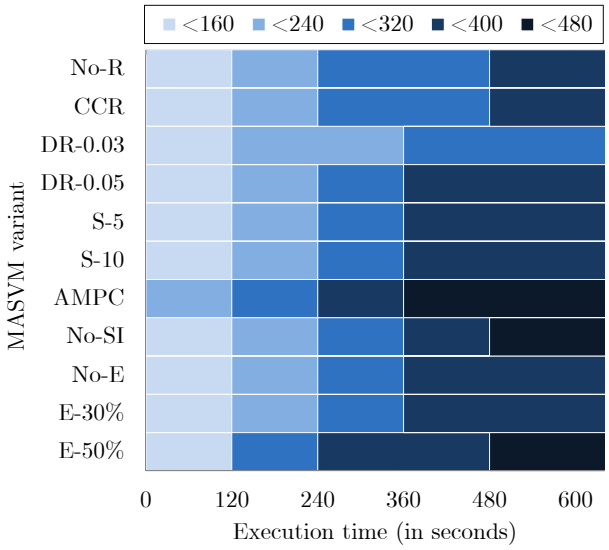

Figure 8: Average number of SVs obtained by various MASVM variants for *Adult* data set.

[4] L.-J. Chien, C.-C. Chang, and Y.-J. Lee. Variant methods of reduced set selection for reduced support vector machines. *JISE*, 26(1):183–196, 2010.

[5] C. Cortes and V. Vapnik. Support-Vector Networks. *Mach. Learn.*, 20(3):273–297, 1995.

[6] E. E. A. Elamin. A proposed genetic algorithm selection method. In *Proc. 1st NITS*, pages 1–8, 2006.

[7] E. Ferragut and J. Laska. Randomized sampling for large data applications of SVM. In *Proc. IEEE ICMLA*, volume 1, pages 350–355, 2012.

[8] X. Guan, X. Zhang, D. Han, Y. Zhu, J. Lv, and J. Su. A strategic flight conflict avoidance approach based on a memetic algorithm. *Chinese J. of Aeronautics*, 27(1):93 – 101, 2014.

[9] Y. Jin, J.-K. Hao, and J.-P. Hamiez. A memetic algorithm for the minimum sum coloring problem. *Comput. Oper. Res.*, 43:318 – 327, 2014.

[10] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Adv. in kernel methods*, pages 169–184. MIT Press, 1999.

[11] M. Kawulok and J. Nalepa. Support vector machines training data selection using a genetic algorithm. In *Proc. S+SSPR 2012*, volume 7626 of *LNCS*, pages 557–565. Springer, 2012.

[12] R. Koggalage and S. Halgamuge. Reducing the number of training samples for fast support vector machine classification. *Neural Information Process. – Lett. and Reviews*, 2(3):57–65, 2004.

[13] Q. Le, T. Sarlos, and A. Smola. Fastfood – approximating kernel expansions in loglinear time. In *Proc. ICML*, 2013.

[14] Y.-J. Lee and S.-Y. Huang. Reduced support vector machines: A statistical theory. *IEEE T. Neural Networ.*, 18(1):1–13, 2007.

[15] Y. Li, P. Li, B. Wu, L. Jiao, and R. Shang. Kernel clustering using a hybrid memetic algorithm. *Natural Comput.*, 12(4):605–615, 2013.

[16] A. Lopez-Chau, X. Li, and W. Yu. Convex-concave hull for classification with SVM. In *Proc. IEEE ICDMW*, pages 431–438, 2012.

[17] M. Marinaki and Y. Marinakis. An island memetic differential evolution algorithm for the feature selection problem. In *Proc. NICSO*, volume 512 of *SCI*, pages 29–42. Springer, 2014.

[18] P. Moscato. On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts – Towards Memetic Algorithms. Technical Report 158-79, California Institute of Technology, 1989.

[19] D. R. Musicant and A. Feinberg. Active set support vector regression. *IEEE T. on Neural Networ.*, 15(2):268–275, 2004.

[20] J. Nalepa and Z. J. Czech. New selection schemes in a memetic algorithm for the vehicle routing problem with time windows. In *Proc. ICANNGA*, volume 7824 of *LNCS*, pages 396–405. Springer, 2013.

[21] J. Nalepa and M. Kawulok. Adaptive genetic algorithm to select training data for support vector machines. In *EvoApp.*, LNCS. Springer, 2014. in press.

[22] S. L. Phung, D. Chai, and A. Bouzerdoum. Adaptive skin segmentation in color images. In *Proc. IEEE ICASSP*, pages 353–356, 2003.

[23] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Proc. NIPS*, pages 1177–1184. 2008.

[24] A. Rahimi and B. Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Proc. NIPS*, pages 1313–1320. 2009.

[25] I. Rodriguez-Lujan, C. S. Cruz, and R. Huerta. Hierarchical linear support vector machine. *Patt. Recogn.*, 45(12):4414 – 4427, 2012.

[26] S. Salvador and P. Chan. Determining the number of clusters / segments in hierarchical clustering / segmentation algorithms. In *Proc. IEEE ICTAI*, pages 576–584, 2004.

[27] G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *Proc. ICML*, pages 839–846, 2000.

[28] H. Shin and S. Cho. Neighborhood property–based pattern selection for SVMs. *Neural Comput.*, 19(3):816–855, 2007.

[29] I. W. Tsang, J. T. Kwok, and P.-M. Cheung. Core vector machines: Fast SVM training on very large data sets. *J. of Mach. Learn. Res.*, 6:363–392, 2005.

[30] D. Wang and L. Shi. Selecting valuable training samples for SVMs via data structure analysis. *Neurocomputing*, 71:2772–2781, 2008.

[31] J. Wang, P. Neskovic, and L. N. Cooper. Training data selection for SVMs. In *Adv. in Natural Comp.*, pages 554–564. Springer, 2005.

[32] Z.-Q. Zeng, H.-R. Xu, Y.-Q. Xie, and J. Gao. A geometric approach to train SVM on very large data sets. In *Proc. ISKE*, volume 1, pages 991–996, 2008.

[33] W. Zhang and I. King. Locating support vectors via β-skeleton technique. In *Proc. ICONIP*, pages 1423–1427, 2002.