Evolutive Algorithms Applied To The Straight Line Segment Classifier

Rosario Medina Rodríguez
Institute of Mathematics and Statistics
University of São Paulo
São Paulo, Brazil
rmedinar@vision.ime.usp.br

Ronaldo Fumio Hashimoto
Institute of Mathematics and Statistics
University of São Paulo
São Paulo, Brazil
ronaldo@ime.usp.br

***Abstract*—During the past years, the use of machine learning techniques has become one of the most frequently performed tasks, due to the large amount of pattern recognition applications. Thus, a great number of techniques dealing with this kind of problem have been developed until now. In this work, we propose an alternative training algorithm to improve the accuracy of the SLS binary Classifier, which produces good results that can be compared to Support Vector Machines. That classifier uses the Gradient Descent method to optimize the final positions of two sets of straight line segments that represent each class. Although, this method quickly converges to a local optimum, it does not guarantee a global minimum. Given that problem, we combine evolutive optimization algorithms with the gradient descent method to improve the classifier accuracy. In addition, we also propose two experiments: (i) explore the use of different number of straight line segments to represent the data distribution and (ii) estimate the optimal number of segments to represent each class using the X-Means algorithm. The proposed methodology showed good results which can be used to solve some other real problems using the SLS classifier with the proposed hybrid training algorithm.**

***Keywords*-straight line segments; supervised machine learning; evolutive algorithms; optimization algorithms; pattern recognition**

I. Introduction

Supervised Classification is one of the most frequently performed and studied tasks due to its multiple applications such as: voice recognition, character recognition, data mining, among others [1], [2]. The most commonly known methods are Linear Classifiers, Neural Networks and Support Vector Machines [3]. The last one, represents the state of the art for machine learning methods due to they strong mathematical background and good performance on practical cases [4].

A new technique for Pattern Recognition (focused on supervised binary classification), called as Straight Line Segment (SLS) classifier [5], [6], [7] was published. The key issue in this technique is to find optimal positions of the straight line segments given a training data set. An algorithm for finding these optimal positions is called *training algorithm*.

(*) This manuscript is a summary of the Msc. dissertation "Algoritmos Evolutivos aplicados ao Classificador baseado em Segmentos de Reta" defended at July 03, 2012.

This classifier has not been explored yet, so we decided to study new forms of improving the classification accuracy to compared it results with the most commonly used classifiers.

Research objective and Contributions: The main research objective of our work was the development of a new training algorithm for the SLS classifier. The proposed method is based on a combination of evolutive algorithms with the gradient descent method in order to improve the accuracy of the mentioned classifier.

The contributions were the following:

- Study and implementation of different hybrid optimization algorithms to train the SLS classifier;
- Explore and evaluate the use different number of straight line segments to represent each class;
- Propose the use of a clustering algorithm (X-Means) to estimate the number of straight line segments;
- Verify the viability of our proposal by applying the classifier in custom and public datasets. In addition, a comparison with original SLS classifier and Support Vector Machines results were done, too.

It is worth noting that, this modified classifier present results that can be compared with SVM and also improved in most of the cases the accuracy of the original Straight Line Segment Classifier.

Following this introduction, Section II provides a briefly description of the original SLS Classifier. Section III, presents one of our proposals which describes new forms to estimate the number of straight line segments to be used in each class. A recall of evolutive optimization algorithms can be found in Section IV and our last proposal of combining these algorithms with gradient descent method in Section V. Section VI shows results from the different experiments. Finally, we give some conclusions and directions for future work in Section VII.

II. Straight Line Segment Classifier

A recent publication on Pattern Recognition presents a new technique based on straight line segments [5], [6], [7]. Its main contribution is to introduce a new type of classifier based on distances between a set of points and two sets of straight line segments. So far, this technique is focused on *binary* classification, so there is only two possible classes in the data

set. For the sake of completeness, we briefly describe in this section the SLS classifier.

A. Basic Definitions for SLS Classifiers

Let p and $q \in \mathbb{R}^{d+1}$. The straight line segment (SLS) with extremities p and q is defined as:

$$L_{p,q} = \{x \in \mathbb{R}^{d+1} : x = p + \lambda \cdot (q - p), 0 \leq \lambda \leq 1\} \quad (1)$$

Given a point $x \in \mathbb{R}^d$, let $x_e = (x, 0)$ denote the extension of x to $\mathbb{R}^{d+1}$.

Given a point $x \in \mathbb{R}^d$ and $L_{p,q} \subseteq \mathbb{R}^{d+1}$. The *pseudo-distance* between x and L is given by:

$$distP(x, L) = \frac{dist(x_e, p) + dist(x_e, q) - dist(p, q)}{2} \quad (2)$$

where $dist(a, b)$ denotes the Euclidean distance between two points $a, b \in \mathbb{R}^{d+1}$.

Let $\mathcal{L}$ denote a set of SLSs, defined as:

$$\mathcal{L} = \{L_{p_i,q_i} : p_i, q_i \in \mathbb{R}^{d+1}, i = 1, ..., m\} \quad (3)$$

where m represents the number of SLSs for each class.

Given a point $x \in \mathbb{R}^d$, the discriminative function is defined as:

$$T_{\mathcal{L}_0,\mathcal{L}_1}(x) = \sum_{L \in \mathcal{L}_1} \frac{1}{distP(x, L) + \varepsilon} - \sum_{L \in \mathcal{L}_0} \frac{1}{distP(x, L) + \varepsilon} \quad (4)$$

where ε is a small positive constant to avoid division by zero.

Considering the discriminative function, the classification function is defined as:

$$F_{\mathcal{L}_0,\mathcal{L}_1}(x) \quad \begin{cases} 0, & if \ \mathcal{S}_{\mathcal{L}_0,\mathcal{L}_1}(x) < 0.5; \\ 1, & \text{otherwise} \end{cases} \quad (5)$$

where $\mathcal{S}_{\mathcal{L}_0,\mathcal{L}_1}(x)$ is a sigmoid function denoted by:

$$S_{\mathcal{L}_0,\mathcal{L}_1}(x) = \frac{1}{1 + e^{-g(T_{\mathcal{L}_0,\mathcal{L}_1}(x))}} \quad (6)$$

The following relationships can be found among the discriminative, sigmoid and classification functions:

- If x has approximately the same "distance" from both $\mathcal{L}_0$ and $\mathcal{L}_1$ (that is, $\forall L \in \mathcal{L}_0, \exists L' \in \mathcal{L}_1 : distP(x, L) \approx distP(x, L')$ and vice-versa) then, by Eq. 4, $T_{\mathcal{L}_0,\mathcal{L}_1}(x) \approx 0$ and consequently, by Eq. 6, $S_{\mathcal{L}_0,\mathcal{L}_1}(x) \approx 0.5$. So, the more x is equally separated from $\mathcal{L}_0$ and $\mathcal{L}_1$, the more it is difficult to discriminate whether x belongs to class 0 or 1.
- The closer x is to $\mathcal{L}_1$ (that is, $\exists L \in \mathcal{L}_1 : distP(x, L) \approx 0$) and, at the same time, x is farther from $\mathcal{L}_0$ (that is, $\forall L \in \mathcal{L}_0 : distP(x, L) \approx +\infty$) then, by Eq. 4, the bigger is $T_{\mathcal{L}_0,\mathcal{L}_1}(x)$ (that is, it tends to $+\infty$) and, consequently, by Eq. 6, the closer $S_{\mathcal{L}_0,\mathcal{L}_1}(x)$ is to 1, and, therefore, by Eq. 5, the closer is more x can be discriminated as belonging to class 1 (that is, the closer the classification function $F_{\mathcal{L}_0,\mathcal{L}_1}(x)$ is to 1).
- Analogously, the closer x is to $\mathcal{L}_0$ and (at the same time) farther from $\mathcal{L}_1$, the closer the classification function $F_{\mathcal{L}_0,\mathcal{L}_1}(x)$ is to 0.

B. Training Algorithm

Given a set of n examples $E_n = \{(x_i, y_i) \in \mathbb{R}^d \times \{0, 1\} : i = 1, 2, .., n\}$, the main purpose of the training algorithm in the context of SLS classifiers, is to find two sets of SLSs $\mathcal{L}_0$ and $\mathcal{L}_1$ in order to minimize the mean square error function defined as:

$$E(F_{\mathcal{L}_0,\mathcal{L}_1}) = \frac{1}{n} \sum_{i=1}^{n} [F_{\mathcal{L}_0,\mathcal{L}_1}(x_i) - y_i]^2 \quad (7)$$

This algorithm can be divided into two phases [6]:

1) *Placing:* This phase consists of pre-allocating (finding initial positions of) the SLSs (in $\mathcal{L}_0$ and $\mathcal{L}_1$) based on the fact that points x closer to $\mathcal{L}_0$ (or $\mathcal{L}_1$, respectively) and farther from $\mathcal{L}_1$ (or $\mathcal{L}_0$, respectively) lead the classification function $F_{\mathcal{L}_0,\mathcal{L}_1}(x)$ to 0 (or 1, respectively). To achieve this goal, the set of examples E_n is split into two groups, $X_i = \{x \in \mathbb{R}^d : (x, y) \in E_n \text{ and } y = i\}$ (for $i = 0, 1$), and then the clustering algorithm k-means is applied to each group. Later, with the objective to obtain the initial extremities of the SLSs for each cluster, the k-means algorithm (with $k = 2$) is applied again, but at this time to each cluster obtained from the previous k-means application.
2) *Tuning:* The purpose of this phase is to minimize the mean square error function. One possible way to accomplish this task is to use the gradient descent technique [8] to find the positions of the SLSs in $\mathcal{L}_0$ and $\mathcal{L}_1$ such that the mean square function derivate is equal to zero. Despite of the gradient descent method does not guarantee the global minimum and the final solution (positions of the SLSs) depends on the initial placing phase, this method was successfully applied in [6].

III. Selecting the number of Straight Line Segments

At SLS Classifier training phase (see II-B), the number of straight line segments (SLSs) that represents each class, besides being equal for $\mathcal{L}_0$ and $\mathcal{L}_1$, it is also an user predefined parameter ($nSLS$) in the algorithm. Despite of the good results obtained by the classifier in [7], where the number of SLSs is the same for both classes and considering that this number might have a great influence in the classification rate. We decided to explore the performance of the classifier when a different number of SLSs is used to represent each one of the classes.

The main problems we found when selecting the number of SLSs, were the following:

- We needed a graphical representation of the data distribution to determine $nSLS$ parameter;
- If the data distribution is unknown (most common case), it would be necessary to train the classifier multiple times, testing all the possible combination of SLSs until we find

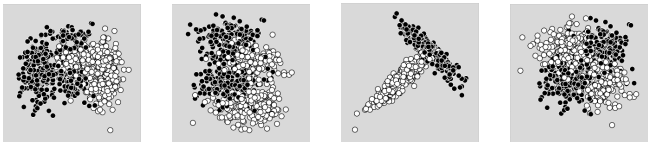

Fig. 1. The four data distributions used in our tests (proposed in [6]).

an optimal one. This lead us to another problem: finding the range of SLSs number to be tested;
- The original SLS classifier solution requires the $nSLS$ parameter to be provided by the user which may not lead to a minimum classification error.

Given that problems, we also propose the application of a clustering algorithm at Placing phase, in order to automatically determine $nSLS$ for each class. For this purpose, we used the *X-Means* algorithm [9], which finds clusters in a dataset by optimizing some information criteria like Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC) [10].

In the following subsections will be described our custom test dataset and the two approaches we choose to reach an optimal number of SLSs for each class.

A. Custom Test Data Sets

In these data sets, each region has a uniform probability density function. This property makes possible to apply the Bayes classifier [1] to obtain the ideal classification rate and compare it with our result as it was proposed in [6]. Furthermore, since the probability density function is known, it is possible to use numerical integration to calculate the classification rate of the SLS classifier.

B. Exhaustive Search of SLSs Number

With this method, we try to find the best SLSs number for each class in order to obtain the best classification rate. As its name explains itself, every possible combination of numbers in the range of 1 to 4, is tested. So, the classifier was trained $nSLS^2$ times, which means 16 for this case, to achieve our objective. Some results can be seen in Figure 2, using the "F-Distribution" with 1600 examples. For each train, we show the final positions of the SLSs and its classification rate. The highlighted figures into squares represent the best training percentages: two using the same number of SLSs per class and the other two using a different number.

C. Applying X-Means on Data Distributions

To avoid multiple trainings with *Exhaustive Search* or *guessing* the optimal pair of numbers for the SLSs, we propose using the X-Means clustering algorithm. The main advantage of this algorithm [10], [11] is that we do not need to provide the value of K as in K-Means algorithm. So, it can estimates the number of clusters within the data distribution which will represent the $nSLS$ parameter for each class.

The X-Means algorithm starts with a small number of clusters. Then by making locally decisions for every centroid

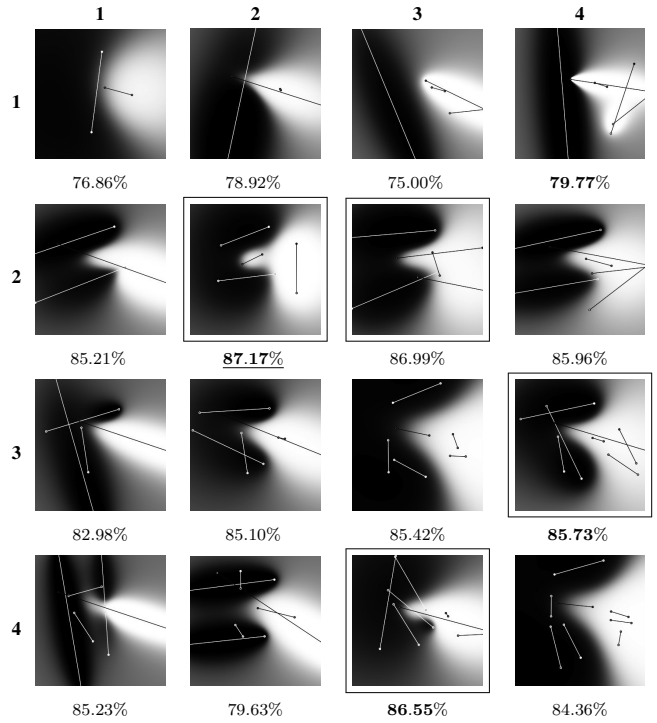

Fig. 2. Exhaustive Search: Final SLSs positions after training phase for "F-Distribution". Bayes classification rate: 87.66%

it decides to divide one centroid into two new centroids or not. The criterion to decide this division is the Bayesian Information Criterion (BIC) [12]. However, it can also be used the Anderson-Darling Criterion (AD) [13], because it presents a good performance with not spherical distributions. As we can not assume the data distributions we will deal with, would be spherical. We choose Anderson-Darling criterion to decide the centroids division.

So, we propose a new Placing algorithm for SLS classifier, by using the X-Means algorithm before the K-Means application to find the extremities of the SLSs. The approach we used to discover if it could find an optimal pair of SLSs number, is described as follows:

- Divide the training data set into two groups, $X_i = \{x \in \mathbb{R}^d : (x, y) \in E_n \text{ and } y = i\}$ (for $i = 0, 1$);
- Then X-Means is applied on each group to obtain $nSLS_0$ clusters for class 0 and $nSLS_1$ clusters for class 1;
- Finally, to get the extremities of the SLSs the K-Means algorithm with $K = 2$, is applied on each cluster obtained from the previous step, as it was proposed in the original placing algorithm.

An example of the results obtained is shown in Figure 3. On the right side, there is a points "F-Distribution" with 1600 examples. The X-Means algorithm found 4 clusters for class 0 and 2 for class 1. The SLSs final positions obtained a classification rate of 87.03% for 3200 examples and 83.22% for 1600 examples. It is worth noting that the positions obtained in Figure 2 differ from these ones, due to the centroid positions founded by the clustering algorithms. In addition,

the classification rate is also better than the obtained with Exhaustive Search (86.55% and 79.63%). As a conclusion of these experiments, we can say that the use of X-Means at Placing phase would help on the improvement of the accuracy of the SLS classifier.

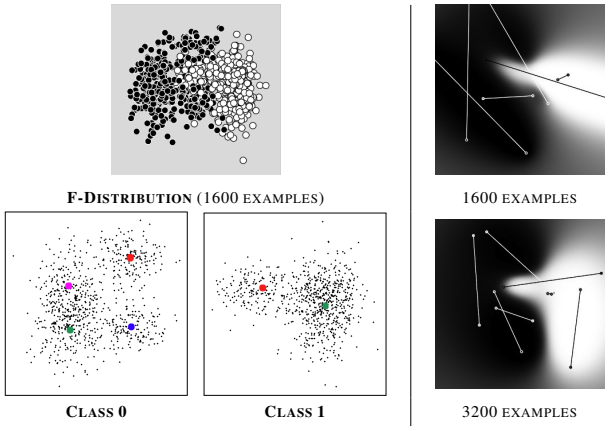

Fig. 3. Applying X-Means algorithm - On the right: "F-Distribution" and its respective clusters for each class. On the left: the final positions for the SLSs after training using 4 SLSs for class 0 and 2 or 3 for class 1, respectively.

IV. Optimization Algorithms

It is known that optimization is the process of making something better. The optimization problem could be divided in two classes: global and local. The first one, always find the best solution in a space of all the possible candidate solutions. Unlike global optimization, the local one, finds the best solution within a set of neighboring candidate solutions.

Optimization Algorithms are commonly used at training phases of supervised machine learning in order to minimize a specific cost function. In addition, considering the Theorem 1, proposed in [14], which explains that it is impossible to create a universal optimization strategy [15]. Several optimization algorithms inspired on different areas where developed.

Theorem 1. *("No Free Lunch" for Optimization Algorithms): Any two algorithms are equivalent when their performance is averaged across all possible problems [16].*

A briefly description of the algorithms used in this work at training phase of the SLS classifier, is presented in the following subsections.

A. Genetic Algorithms

The genetic algorithm (GA) is an optimization and search technique based on the principles of genetics and natural selection. A GA allows a population composed of many individuals to evolve under specified selection rules to a state that maximizes the "fitness" (minimizes the cost function) [17].

B. Dialectical Optimization Method

The Dialectical Optimization Method (DOM) it is an evolutionary method based on the materialist dialectics for solving search and optimization problems [18], [19]. In addition, it is based on the dynamics of contradictions between their integrating dialectical poles. Each pole is considered as a possible solution to the problem. These poles are affected by process that make the system tend to stability called as *pole struggle* and *revolutionary crises*, where some poles can be absorbed by others or new poles may came up. The cost function to be minimized is called as *social force*.

C. K-Beam Search

Sometimes, there are several local minimum points in the optimization problem, so the global optimization is not guaranteed. To avoid that problem, the optimization algorithm is executed with different start points, saving the best solution of them. The K-Beam Search, which is also a search and optimization algorithm, works in the same way. Instead of saving only the best solution, it saves a set of K solutions and iteratively creates a predefined number of successors for each of them, evaluating its neighboring solutions [20], to find a global solution.

V. Hybrid Optimization Methods

The main advantage of evolutive algorithms is their capability of escaping from local optimum by multi-point stochastic searching. On the other hand, the strength of gradient descent method [8] is the ability of finding local optimum by pointing the direction that maximizes the objective function.

Thus, based on these properties, the combination of gradient descent and evolutive optimization algorithms, described on the Algorithm 1, could improve the solution "quality" of optimization problem. Also, as it was proposed in [21], the evolutive algorithms (DOM, in that case) can assist the gradient descent method by providing a new set of initial positions for the two sets of SLSs $\mathcal{L}_0$ and $\mathcal{L}_1$, helping the gradient to escape from local minima.

Algorithm 1 : Hybrid Optimization Methods Algorithm

Require: $iterations$, $initialSolutionSize$, $minimumError$: set of initial parameters
Ensure: $minValue$: optimized value.
1: $w[0 : initialSolutionSize] \leftarrow generateInitialSolutions()$
2: **while** $(it < iterations)$ **do**
3: **if** $(minValue > minimumError)$ **then**
4: $w \leftarrow applyGradientDescent(w)$
5: $minValue \leftarrow evolutiveAlgorithmFunctions(w)$
6: **end if**
7: **end while**
8: **return** $minValue$

It is important to emphasize that the initial set of solutions includes the solution obtained from one single application of gradient descent so that our hybrid approach can generate a solution that is equal to or better than the one obtained by just using the gradient descent method. In this sense, our approach is conservative.

In order to apply the evolutive algorithms to the straight line segments problem, adaptations were made to some concepts

of Genetic Algorithms / Dialectical Optimization Method / K-Beam Search to resolve the find the best positions of the SLSs in $\mathcal{L}_0$ and $\mathcal{L}_1$:

1) Chromosome/ Poles/ States: Representing each initial solution, is a vector consisting of the concatenated extremities of the SLSs belonging to $\mathcal{L}_0$ and $\mathcal{L}_1$ (i.e. $[\mathcal{L}_0|\mathcal{L}_1]$).
2) Fitness Function/ Social Force/ Cost Function: Represented by the MSE function, defined in Equation 5, which is the classification function from the SLS classifier.

As result, we present 5 choices for training the SLS classifier: (1)Gradient Descent (GD), (2)Genetic Algorithms (GA); and the hybrid methods: (3)Genetic Algorithms with Gradient Descent (GA&GD, (4) Dialectical Optimization Method with Gradient Descent (DOM&GD) and (5)K-Beam Search with Gradient Descent (KBM&GD).

VI. Obtained Results

In order to evaluate our proposal we performed three experiments on the 4 custom test distributions described on Subsection III-A, with 100, 200, 400, 800, 1600 and 3200 examples; 3 samples for each one, totalizing 18 samples for each distribution. The experiments conducted were: (i) To find the best hybrid optimization algorithm to train the SLS classifier (GD, GA, GA&GD, MOD&GD, KBM&GD); (ii) To find the best set of parameters, by conducting a sensitivity analysis and (iii) To find the best way of estimating the SLSs number for each class. Finally, we apply the best training option on a set of public data sets [1].

All the experiments were performed in computers with 16 processors Intel 2.4 GHz and 5.8 GB of RAM under Ubuntu/Linux operating system. In this work we only show 1 experiment for custom distributions and the results obtained for public data sets.

A. Sensitivity Analysis

Here, we described the second experiment. The SLS classifier was trained with each of the 5 hybrid training method with the best SLSs number obtained from the exhaustive search for "F-Distribution": $nSLS = 2$ for each class, using 3 variations of parameters described on Table I. One result from this analysis is presented on Figure 4, we can say that the best classifications rates were using Param 2 and 3, but for most of the other distributions were not significant difference.

B. Public Data Set

The proposed hybrid methods were also applied to eight public data sets extracted from the UCI Machine Learning Repository [22]. The experiments were conducted by separating each sample randomly in two parts: $2/3$ of the examples for training and $1/3$ for testing, using 10-Fold-Validation with the 5 hybrid algorithms and the third set of parameters. To estimate the number of SLS we use X-Means Algorithm.

[1] All the results and experiments analysis are available on: http://vision.ime.usp.br/~rmedinar/Sibgrapi_Experiment_Results.

TABLE I
Parameters Sets for each Hybrid Optimization Method

	Parameters		
GA	Param. 1	Param. 2	Param. 3
Population	200	500	1000
Generations	100	1000	500
GA&GD	Param. 1	Param. 2	Param. 3
Population	2	10	50
Generations	2	5	3
DOM&GD	Param. 1	Param. 2	Param. 3
Poles	30	15	50
Phases	20	10	15
Iterations	15	15	15
KBM&GD	Param. 1	Param. 2	Param. 3
States	2	5	10
Successors	10	20	15
Iterations	3	10	5

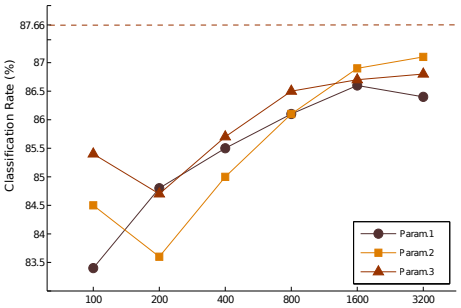

Fig. 4. Sensitivity Analysis for "F-Distribution" using Genetic Algorithm training method and different sizes of data sets. The pointed line represent the Bayes classification rate (87.6%).

The results can be observed on Table II. The higher percentages for our proposal, are highlighted in bold and the best results from the original SLS classifier and SVM are underlined. Finally, the highest classification rate from them is highlighted into a square for each data set. These results showed that for most of the cases our proposal is better than the original SLS classifier and even sometimes better than the Support Vector Machines. About the number of SLSs, in most of the cases, it was increased improving the accuracy of the classifier.

TABLE III
SLSs number after applying X-Means and K-Means

Data Sets	SLSs - Number	
	X-Means	K-Means
AUSTRALIAN	9-1	8-8
BREAST-CANCER	3-1	4-4
DIABETES	**3-5**	1-1
GERMAN	**5-3**	2-2
HEART	3-2	10-10
IONOSPHERE	2-8	10-10
LIVER-DISORDERS	**4-2**	3-3
SONAR	**2-2**	4-4

VII. Discussion and Conclusions

In this paper we presented two approaches to estimate the required number of straight line segments to be used

TABLE II
RESULTS OBTAINED USING THE 5 HYBRID TRAINING ALGORITHMS WITH THE BEST SET OF PARAMETERS.

	HYBRID TRAINING ALGORITHMS - PROPOSALS					BEST % - ORIGINAL	
PUBLIC DATA SETS	GD	AG+GD	AG	MDO+GD	KBM+GD	CLASS. SLS	SVM
AUSTRALIAN	**76.9** (0.6)	75,0(2,7e-04)	75,0 (2,7e-04)	75,0 (1,9e-04)	75,4 (0,6)	87.0 (1.8)	87.4 (1.6)
BREAST-CANCER	**76.9** (2.4)	75.2 (0.4)	75.0 (0.0)	75.0 (0.0)	75.0 (0.0)	98.1 (0.7)	97.9 (0.9)
DIABETES	75.9 (0.2)	81.2 (0.9)	75.1 (0.03)	**81.5** (1.1)	81.4 (0.7)	76.4 (1.8)	77.8 (1.8)
GERMAN	76.5 (0.5)	80.5 (0.8)	75.2 (0.03)	80.7 (1.1)	**80.9** (0.7)	76.7 (2.2)	77.3 (0.5)
HEART	75.5 (0.5)	77.0 (1.1)	75.0 (0.01)	**77.9** (1.7)	77.1 (0.9)	82.2 (3.3)	85.1 (3.3)
IONOSPHERE	81.0 (3.5)	94.7 (1.1)	74.5 (3.7)	**95.1** (1.1)	93.1 (1.8)	95.2 (2.6)	96.0 (2.1)
LIVER-DISORDERS	75.8 (0.5)	**78.9** (1.9)	75.5 (0.3)	77.3 (1.8)	76.3 (1.2)	70.1 (2.8)	72.7 (2.7)
SONAR	80.9 (1.8)	**88.9** (0.8)	75.3 (0.09)	87.6 (2.8)	87.8 (1.4)	86.3 (4.1)	88.4 (4.2)

for representing each class. An exhaustive search which as expected take a lot of time and resources and the application of a clustering algorithm X-Means. From the results, we suggest the alternative of using the X-Means algorithm in future, works due to the improvement of the accuracy and better representation of the data distribution. Although, it has been showed that X-Means can also fail when finding the clusters.

The application of different evolutive optimization algorithms combined with the gradient descent method allows the accuracy improvement of the SLS classifier. This can be confirmed with the results obtained with the public data set. While these hybrid algorithms improves the classification rate, the computation time for the training algorithm increases because of the multiple iterations of the evolutive algorithms and gradient descent method. In addition it has been studied the use of threads on the implementation to reduce the training time. From the results, a good alternative for training the SLS classifier could be the combination of Dialectical Optimization Method and Gradient Descent, because it present good classification rates. Also, in contrast with the other optimization methods, it reduces the space of solutions in each iteration.

The sensitivity analysis of the evolutive algorithms on the custom data sets did not showed significative differences in the classification rates. It is our intention as future work, to rebuild this experiment with another distribution in order to confirm our results. We also plan to extend this work by finding the best type of data distributions or problems where the SLS classifier beat the Support Vector Machines with a bigger margin of classification rate and to develop an extended version of the classifier to be used on multi-class data distributions.

ACKNOWLEDGMENT

Financial support for this research has been provided by FAPESP and CNPq.

REFERENCES

[1] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. John Wiley and Sons, 2001.

[2] C. Bishop, *Pattern Recognition and Machine Learning*. Springer New York, 2006, vol. 4.

[3] S. Kotsiantis, "Supervised machine learning: A review of classification techniques," in *Proceeding of the 2007 conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*. IOS Press, 2007, pp. 3–24.

[4] S. Abe, *Support Vector Machines for Pattern Classification (Advances in Pattern Recognition)*, 2nd ed. Springer-Verlag New York, Inc., 2010.

[5] J. Ribeiro and R. Hashimoto, "A new machine learning technique based on straight line segments," in *ICMLA 2006: 5th International Conference on Machine Learning and Applications, Proceedings*. IEEE Computer Society, 2006, pp. 10–16.

[6] ——, "A new training algorithm for pattern recognition technique based on straight line segments," in *Proceedings of the 2008 XXI Brazilian Symposium on Computer Graphics and Image Processing*. IEEE Computer Society, 2008, pp. 19–26.

[7] ——, *Pattern Recognition, Recent Advances*. I-Tech, 2010, ch. Pattern Recognition Based on Straight Line Segments, book Chapter.

[8] D. Michie, D. Spiegelhater, and C. Taylor, "Introduction to optimization theory," 1973.

[9] D. Pelleg and A. Moore, "X-means: Extending k-means with efficient estimation of the number of clusters," in *Proceeding of the 17th International Conference on Machine Learning*. Morgan Kaufmann, 2000, pp. 727–734.

[10] A. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[11] R. Vatsavaia, C. Chandolaa, V. Junb, and G. Junb, "Gx-means: A model-based divide and merge algorithm for geospatial image clustering," *Procedia Computer Science*, vol. 4, pp. 186–195, 2011.

[12] G. Schwarz, "Estimating the dimension of a model," *The annals of statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[13] G. Hamerly and C. Elkan, "Learning the K in K-Means," in *Neural Information Processing Systems*. MIT Press, 2003.

[14] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[15] Y. Ho and D. Peyne, "Simple explanation of the no free lunch theorem of optimization," in *Proceedings of the 40th IEEE Conference on Decision and Control*, vol. 5. IEEE, 2001, pp. 4409–4414.

[16] D. Wolpert and W. Macready, "Coevolutionary free lunches," *Evolutionary Computation, IEEE Transactions on*, vol. 9, no. 6, pp. 721–735, 2005.

[17] R. Haupt and S. Haupt, *Practical Genetic Algorithms*. J. Wiley & Sons, 2004.

[18] W. D. Santos and F. D. Assis, "Optimization based on dialectics," in *Proceedings of the 2009 International Joint Conference on Neural Networks*, ser. IJCNN'09. IEEE Press, 2009, pp. 1095–1102.

[19] W. D. Santos, F. D. Assis, R. D. Souza, P. Mendes, H. Monteiro, and H. Alves, "Dialectical non-supervised image classification," in *Proceedings of the Eleventh conference on Congress on Evolutionary Computation*, ser. CEC'09. IEEE Press, 2009, pp. 2480–2487.

[20] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Prentice Hall, 2002.

[21] R. Medina and R. Hashimoto, "Combining dialectical optimization and gradient descent methods for improving the accuracy of straight line segment classifiers," in *Sibgrapi 2011. 24 Conference on Graphics, Patterns and Images.* IEEE, august 2011, pp. 321–328.

[22] A. Asuncion and D. Newman, "UCI machine learning repository," http://archive.ics.uci.edu/ml/, 2007.