A New Training Algorithm for Pattern Recognition Technique Based on Straight Line Segments

João Henrique Burckas Ribeiro and Ronaldo Fumio Hashimoto
Universidade de São Paulo
Instituto de Matemática e Estatística
Departamento de Ciência da Computação
Rua do Matão, 1010 - São Paulo -SP
burckas@ime.usp.br and ronaldo@ime.usp.br

Abstract

Recently, a new Pattern Recognition technique based on straight line segments (SLSs) was presented. The key issue in this new technique is to find a function based on distances between points and two sets of SLSs that minimizes a certain error or risk criterion. An algorithm for solving this optimization problem is called training algorithm. Although this technique seems to be very promising, the first presented training algorithm is based on a heuristic. In fact, the search for this best function is a hard nonlinear optimization problem. In this paper, we present a new and improved training algorithm for the SLS technique based on gradient descent optimization method. We have applied this new training algorithm to artificial and public data sets and their results confirm the improvement of this methodology.

1. Introduction

Recently, a new Pattern Recognition technique based on straight line segments (SLSs) was presented [16]. Basically, given a training data set, this technique estimates a function where the value of this function is calculated based on distances between points and two sets of SLSs. The key issue of this new technique is to find two sets of SLSs that minimizes a certain error or risk criterion. An algorithm for solving this optimization problem is called training algorithm. Although this technique seems to be very promising, the first training algorithm presented in [16] is based on a heuristic. In fact, the search for this best function is a hard nonlinear optimization problem. In this paper, we present a new and improved training algorithm for the SLSs technique based on gradient descent optimization method. We have applied this new training algorithm to artificial and public data sets and their results confirm the improvement of this methodology.

Following this brief introduction, Section 2 introduces the Pattern Recognition problem. Section 3 recalls some concepts of the SLS technique presented in [16]. Section 4 presents the new training algorithm. Section 5 shows the experimental results and, finally, Section 6 concludes this paper and points some directions for future research in this topic.

2. Pattern Recognition Problem Definition

In [17], Vapnik described a model for Pattern Recognition Theory for which the *supervised classification* problem can be briefly defined in the following way: (1) let $\mathcal{F}_\Lambda = \{f_\alpha \in \{0,1\}^{\mathbb{R}^d},\ \alpha \in \Lambda\}$ be a set of functions, where Λ is a set of parameter vectors; (2) let $S_n = \{(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \{0,1\}, i = 1, 2, \ldots, n\}$ be a sample set with n examples $(\boldsymbol{x}_i, y_i)$, such that $(2.a)$ each vector $\boldsymbol{x}_i \in \mathbb{R}^d$ is drawn independently from a fixed but unknown probability distribution $P(\boldsymbol{x})$; $(2.b)$ for each input vector $\boldsymbol{x}_i$ a supervisor returns an output value $y_i \in \{0,1\}$ according to a conditional probability distribution $P(y \mid \boldsymbol{x})$, also fixed but unknown; (3) it is desired to find the parameter $\alpha^* \in \Lambda$ such that the function $f_{\alpha^*} \in \mathcal{F}$ minimizes a certain error or risk function in order to classify (that is, to give labels $y \in \{0,1\}$ to) new query points $\boldsymbol{x} \in \mathbb{R}^d$.

3. Straight Line Segment Technique

It is important to observe that in the literature we can find some works that use straight lines for solving Pattern Recognition problems. The Nearest Feature Line Method (NFL) was proposed by Li and Lu in [13]. This method is similar to Nearest Neighbor (NN) proposed by Cover and Hart in 1967 [4]. While NN compares distance from the query point to training examples, NFL compares distance from the query point and straight lines defined by pairs of examples. This method has shown good performance [3, 9, 10, 12, 13]. However, it has some drawbacks: (1) high computational complexity ($O(N_C^2)$, where N_C is the number of examples); (2) extrapolation inaccuracy; and (3) interpolation inaccuracy. Many other authors have proposed several methods improving the original idea of NFL [7, 15, 19, 18]. Using a different approach from NFL method, Ribeiro and Hashimoto [16] proposed a new Pattern Recognition technique based on *straight line segments* (SLS). The advantages of this new technique over the NFL methods are that of it avoids extrapolation and interpolation inaccuracy and has low computational complexity. In this section, we recall some concepts of the SLS technique presented in [16].

Let $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{R}^{d+1}$. The straight line segment with extremities $\boldsymbol{p}$ and $\boldsymbol{q}$, denoted $\overline{\boldsymbol{pq}}$, is the closed segment of a line in $\mathbb{R}^{d+1}$ with endpoints $\boldsymbol{p}$ and $\boldsymbol{q}$. More formally,

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

Given a point $\boldsymbol{x} \in \mathbb{R}^d$, the extension of $\boldsymbol{x}$ to $\mathbb{R}^{d+1}$, denoted by $\boldsymbol{x}_e \in \mathbb{R}^{d+1}$, is the point $\boldsymbol{x}_e = (\boldsymbol{x}, 0)$, that is, the point $\boldsymbol{x}$ is extended to $\mathbb{R}^{d+1}$ by adding one more coordinate with zero value.

Given a point $\boldsymbol{x} \in \mathbb{R}^d$ and a SLS $\overline{\boldsymbol{pq}}$, with $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{R}^{d+1}$, we define the pseudo-distance between $\boldsymbol{x}$ and $\overline{\boldsymbol{pq}}$ as

$$pdist(\boldsymbol{x}, \overline{\boldsymbol{pq}}) = \frac{dist(\boldsymbol{x}_e, \boldsymbol{p}) + dist(\boldsymbol{x}_e, \boldsymbol{q}) - dist(\boldsymbol{p}, \boldsymbol{q})}{2}, \quad (2)$$

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

Note that if $\boldsymbol{x}_e \in \overline{\boldsymbol{pq}}$, then $pdist(\boldsymbol{x}, \overline{\boldsymbol{pq}}) = 0$. In addition, the larger is the distance between $\boldsymbol{x}_e$ and one extremity of $\overline{\boldsymbol{pq}}$, the greater is $pdist(\boldsymbol{x}, \overline{\boldsymbol{pq}})$. Besides, if $\boldsymbol{p} = \boldsymbol{q}$, then $pdist(\boldsymbol{x}, \overline{\boldsymbol{pq}}) = dist(\boldsymbol{x}_e, \boldsymbol{p}) = dist(\boldsymbol{x}_e, \boldsymbol{q})$.

Let $\mathcal{L}$ denote a collection of SLSs, that is, $\mathcal{L} = \{\overline{\boldsymbol{p}_i\boldsymbol{q}_i} : \boldsymbol{p}_i, \boldsymbol{q}_i \in \mathbb{R}^{d+1}, i = 1, \ldots, n\}$. Given $\mathcal{L}_0$ and $\mathcal{L}_1$, two collections of SLSs in $\mathbb{R}^{d+1}$, let us define two functions. The first is called *balance function* and the second one *classification function*.

The *balance function* $T_{\mathcal{L}_0,\mathcal{L}_1} : \mathbb{R}^d \to \mathbb{R}$ is defined as

$$T_{\mathcal{L}_0,\mathcal{L}_1}(\boldsymbol{x}) = \left(\sum_{\overline{\boldsymbol{pq}} \in \mathcal{L}_1} \frac{1}{pdist(\boldsymbol{x}, \overline{\boldsymbol{pq}}) + \varepsilon} - \sum_{\overline{\boldsymbol{pq}} \in \mathcal{L}_0} \frac{1}{pdist(\boldsymbol{x}, \overline{\boldsymbol{pq}}) + \varepsilon} \right), \quad (3)$$

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

The *classification function* $y_{\mathcal{L}_0,\mathcal{L}_1} : \mathbb{R}^d \to [0, 1]$ is defined as

$$y_{\mathcal{L}_0,\mathcal{L}_1}(\boldsymbol{x}) = \frac{1}{1 + e^{-g \cdot T_{\mathcal{L}_0,\mathcal{L}_1}(\boldsymbol{x})}}, \quad (4)$$

where g is a positive real number for scale purposes (see Section 4.2). The classification of a point $\boldsymbol{x} \in \mathbb{R}^d$ by the classification function is obtained by thresholding its output in the following way: label $\boldsymbol{x}$ as the class 0 if and only if $y_{\mathcal{L}_0,\mathcal{L}_1}(\boldsymbol{x}) \leq 0.5$.

In order to give an intuition of what these functions do, let us analyze their behavior in a particular situation. Given a point $\boldsymbol{x} \in \mathbb{R}^d$, let $d_{\mathcal{L}_0}(\boldsymbol{x}) = \min\{pdist(\boldsymbol{x}, \overline{\boldsymbol{pq}}) : \overline{\boldsymbol{pq}} \in \mathcal{L}_0\}$ and $d_{\mathcal{L}_1}(\boldsymbol{x}) = \min\{pdist(\boldsymbol{x}, \overline{\boldsymbol{pq}}) : \overline{\boldsymbol{pq}} \in \mathcal{L}_1\}$. For all $\overline{\boldsymbol{p}_i\boldsymbol{q}_i} \in \mathcal{L}_0$ and for all $\overline{\boldsymbol{p}_j\boldsymbol{q}_j} \in \mathcal{L}_1$, let $d(\mathcal{L}_0, \mathcal{L}_1) = \min\{dist(\boldsymbol{p}_i, \boldsymbol{p}_j), dist(\boldsymbol{p}_i, \boldsymbol{q}_j), dist(\boldsymbol{q}_i, \boldsymbol{p}_j), dist(\boldsymbol{q}_i, \boldsymbol{q}_j)\}$. Consider the case where $d(\mathcal{L}_0, \mathcal{L}_1) \ggg 0$ is large sufficient in order to have the following situation: as $d_{\mathcal{L}_0}(\boldsymbol{x})$ tends to 0 (denoted by $d_{\mathcal{L}_0}(\boldsymbol{x}) \to 0$), $d_{\mathcal{L}_1}(\boldsymbol{x})$ tends to a very big positive real number $d(\mathcal{L}_0, \mathcal{L}_1)$. In this case, as $\varepsilon \to 0$ and $d_{\mathcal{L}_0}(\boldsymbol{x}) \to 0$, the function $T_{\mathcal{L}_0,\mathcal{L}_1}(\boldsymbol{x}) \to +\infty$ and thereby $y_{\mathcal{L}_0,\mathcal{L}_1}(\boldsymbol{x}) \to 0$. This means that query points that are "near" to at least one SLS in $\mathcal{L}_0$ are classified as class 0. Similarly, as $\varepsilon \to 0$ and $d_{\mathcal{L}_1}(\boldsymbol{x}) \to 0$, the function $T_{\mathcal{L}_0,\mathcal{L}_1}(\boldsymbol{x}) \to -\infty$ and thereby $y_{\mathcal{L}_0,\mathcal{L}_1}(\boldsymbol{x}) \to 1$.

Now, before ending this section, let us give some intuition why the extremities of SLSs are placed in $\mathbb{R}^{d+1}$ while query points are in $\mathbb{R}^d$. Fig. 1 illustrates a situation where query points are in $\mathbb{R}^2$ and SLSs are in $\mathbb{R}^3$. If the SLSs are placed in a space with one more dimension than the feature space, we increase the possibilities for placing the extremities of the SLSs, and therefore, obtaining a richer set for balance functions. Therefore, we have more flexibility for the classification function in the sense that this approach provides more complex classification boundaries.

4. Training Algorithm

The key issue in the technique presented in [16] is to find two collections of SLSs $\mathcal{L}_0$ and $\mathcal{L}_1$ such that the classification function minimizes a certain error or risk criterion. An algorithm for solving this optimization problem is called *training algorithm*. Although the technique given in [16] seems to be very promising, its training algorithm is based on a heuristic. In fact, the search for this best function is a hard nonlinear optimization problem. The main contribution of this work is to present a new and improved training algorithm for the SLS technique based on gradient descent optimization method.

According to Vapnik [17], the risk criterion to be minimized can be defined as

$$R(\alpha) = \int Loss(y, f_\alpha(\boldsymbol{x})) \cdot dP(\boldsymbol{x}, y), \quad (5)$$

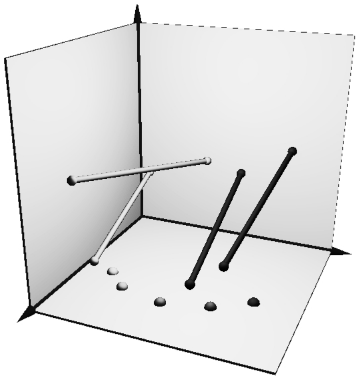

Figure 1. A Representation of SLS Technique. The isolated points represent the query points. While query points are in $\mathbb{R}^2$, SLSs are in $\mathbb{R}^3$.

where $Loss(y, f(\boldsymbol{x}, \alpha))$ is a function that represents the loss in case of misclassification. Although it is possible to use many types of loss function, we choose the classical least square function. In this case, the loss function is known as mean square error (MSE) and it is defined as

$$Loss(y, f_\alpha(\boldsymbol{x})) = (y - f_\alpha(\boldsymbol{x}))^2. \tag{6}$$

In our case, $f_\alpha(\boldsymbol{x}) = y_{\mathcal{L}_0,\mathcal{L}_1}(\boldsymbol{x})$ and the parameter vector α is defined by the SLSs in the collections $\mathcal{L}_0$ and $\mathcal{L}_1$. Since the probability distribution $P(\boldsymbol{x}, y)$ is not known, it is not possible to calculate the value of the risk function $R(\alpha)$ analytically. So we have to estimate it from the sample set $S_n = \{(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \{0,1\}, i = 1, 2, \ldots, n\}$ with n examples. The estimated value for the risk function is called *empirical risk* and it is calculated as

$$\hat{R}(\mathcal{L}_0, \mathcal{L}_1) = \frac{1}{n} \sum_{i=1}^{n} [err_i(y_{\mathcal{L}_0,\mathcal{L}_1})]^2 \tag{7}$$

where

$$err_i(y_{\mathcal{L}_0,\mathcal{L}_1}) = y_{\mathcal{L}_0,\mathcal{L}_1}(\boldsymbol{x}_i) - y_i \tag{8}$$

In [17], Vapnik studied the necessary and sufficient conditions for which the empirical risk minimization leads a good generalization for the classification function. In fact, this is a key problem in Supervised Pattern Classification field.

4.1. Placing the First Straight Line Segments

The first step for the training algorithm consists of placing some SLSs in $\mathcal{L}_0$ and $\mathcal{L}_1$. These SLSs will be adjusted later by the second step of the training algorithm. This first step is called *placing algorithm.*

The placing algorithm is based on the fact that regions in the feature space near to the SLSs in $\mathcal{L}_0$ lead the classification function $y_{\mathcal{L}_0,\mathcal{L}_1}(\boldsymbol{x})$ to 0 and the regions near to the SLSs in $\mathcal{L}_1$ lead the classification function $y_{\mathcal{L}_0,\mathcal{L}_1}(\boldsymbol{x})$ to 1.

The first step of the placing algorithm splits the points $\boldsymbol{x}_i \in \mathbb{R}^d$ of sample set $S_n = \{(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \{0,1\}, i = 1, 2, \ldots, n\}$ into two groups, X_0 and X_1 (see Lines 4 and 5 of the placing algorithm). The points in X_0 are used to form the SLSs that will be placed into the set $\mathcal{L}_0$. Similarly, the points in X_1 are used to form the SLSs that will be placed into the set $\mathcal{L}_1$.

The second step of the placing algorithm is to find the extremities of the SLSs. For this, each group X_j $(j = 0, 1)$ will be split into $k = 2 \cdot MaxSLS$ clusters (subsets), where $MaxSLS$ denotes the number of SLSs in each class (that is, $|\mathcal{L}_0| = |\mathcal{L}_1| = MaxSLS$). The centroids of these clusters (say, $\boldsymbol{c}_{j,i} \in \mathbb{R}^d$, $i = 1, \ldots, k$) will be used to compute the extremities of the SLSs in $\mathcal{L}_j$. In this way, each set $\mathcal{L}_j$ will have $MaxSLS$ SLSs. These centroids $\boldsymbol{c}_{j,i} \in \mathbb{R}^d$ can be computed by using clustering techniques [5, 11]. Although any clustering algorithm could be used in this step, in our case, we choose the k-means clustering algorithm (Line 12 of the PLACING procedure), since it is simple to implement and usually converges quickly [5, 11].

For a fixed group X_j, let C_j the set of $k = 2 \cdot MaxSLS$ centroids of X_j, that is, $C_j = \{\boldsymbol{c}_{j,i} \in \mathbb{R}^d : i = 1, \ldots, k\}$, repeat the following process until there are no more centroids in C_j: (1) select the two nearest centroids $\boldsymbol{c}_0$ and $\boldsymbol{c}_1$ of C_j to be used to form a SLS in order to put it into the set $\mathcal{L}_j$; (2) take the centroids $\boldsymbol{c}_0$ and $\boldsymbol{c}_1$ out from C_j.

In the following, we present the PLACING procedure. The function k-means(X_j) returns the set C_j with $k = 2 \cdot MaxSLS$ centroids. The function Nearest(C_j) returns the two nearest centroids in the set C_j.

```
1:  PLACING ($S_n, MaxSLS$)
2:  **Input**: A sample $S_n = \{(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \{0,1\}, i = 1, 2, \ldots, n\}$ and $MaxSLS$ (number of SLSs
           in $\mathcal{L}_0$ and $\mathcal{L}_1$).
3:  **Output**: Two sets of SLSs $\mathcal{L}_0$ e $\mathcal{L}_1$ (with $MaxSLS$ in
           each one).
4:  $X_0 \leftarrow \{\boldsymbol{x}_i \in \mathbb{R}^d : (\boldsymbol{x}_i, y_i) \in S_n$ and $y_i \leq 0.5\}$;
5:  $X_1 \leftarrow \{\boldsymbol{x}_i \in \mathbb{R}^d : (\boldsymbol{x}_i, y_i) \in S_n$ and $y_i > 0.5\}$;
6:  $\sigma_0 \leftarrow$ standard deviation of $X_0$;
7:  $\sigma_1 \leftarrow$ standard deviation of $X_1$;
8:  $desl \leftarrow |\sigma_0| + |\sigma_1|$;
9:  $k \leftarrow 2 \cdot MaxSLS$;
10: **for** $j \leftarrow 0$ to $1$ **do**
11:     $\mathcal{L}_j \leftarrow \emptyset$;
12:     $C_j \leftarrow k$-means($X_j$);
13:     **repeat**
14:         $[\boldsymbol{c}_0, \boldsymbol{c}_1] \leftarrow$ Nearest($C_j$);
```

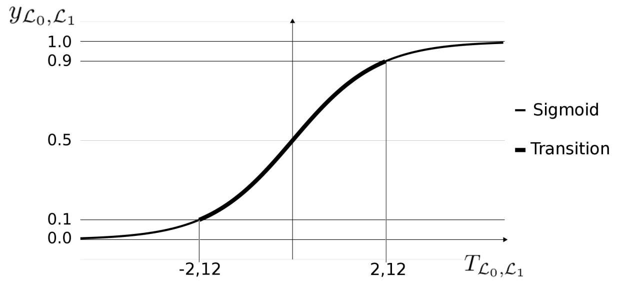

Figure 2. Sigmoid Curve. The proposed method selects *g* in order to have the most part of the training examples in sigmoid transition curve. In this way, this method avoids very large or small values for *g*.

```
15:         $C_j \leftarrow C_j \setminus \{\boldsymbol{c}_0, \boldsymbol{c}_1\}$;
16:         $\boldsymbol{d}_0 \leftarrow (\boldsymbol{c}_0, desl)$; /* Extending $\boldsymbol{c}_0$ to $\mathbb{R}^{d+1}$ */
17:         $\boldsymbol{d}_1 \leftarrow (\boldsymbol{c}_1, desl)$; /* Extending $\boldsymbol{c}_1$ to $\mathbb{R}^{d+1}$ */
18:         $\mathcal{L}_j \leftarrow \mathcal{L}_j \cup \overline{\boldsymbol{d}_0\boldsymbol{d}_1}$;
19:     until $(C_j = \emptyset)$;
20: end for
21: return $\mathcal{L}_0, \mathcal{L}_1$;
```

The computational complexity of k-means when applied to a set with n elements is $O(n \cdot d \cdot k \cdot t)$, where t is the number of iterations, typically lower than n. The computation complexity of Nearest(C_j) is $O(k^2 \cdot d)$, since $|C_j| = k$. So, the computational complexity of the PLACING procedure is $O(n^2 \cdot d \cdot k + k^3 \cdot d)$. Since topically $n^2 \cdot k \gg k^3$ (assuming $n \gg k$), we can consider its complexity as $O(n^2 \cdot d \cdot k)$.

4.2. The Value of Constant *g*

The constant g fits the scale factor of the sigmoid $y_{\mathcal{L}_0,\mathcal{L}_1}(\boldsymbol{x})$. Fig. 2 shows a sigmoid curve (in this example, $y_{\mathcal{L}_0,\mathcal{L}_1}(\boldsymbol{x}) \times g \cdot T_{\mathcal{L}_0,\mathcal{L}_1}(\boldsymbol{x})$). In the sigmoid curve, we call the *sigmoid transition curve* the domain points $g \cdot T_{\mathcal{L}_0,\mathcal{L}_1}(\boldsymbol{x})$ for which $y_{\mathcal{L}_0,\mathcal{L}_1}(\boldsymbol{x}) \in [0.1; 0.9]$. In Fig. 2, the sigmoid transition curve is marked in bold.

The aim of this part of the training algorithm is to select the value for g such that the most part values of $g \cdot T_{\mathcal{L}_0,\mathcal{L}_1}(\boldsymbol{x}_i)$, $i = 1, 2, \ldots, n$, are in the sigmoid transition curve. For this, consider the function $\Theta : \mathbb{R}^d \to \{0, 1\}$ defined as:

$$\Theta(\boldsymbol{x}) = \begin{cases} 1 & g < \ln((2-\eta)/\eta)/|T_{\mathcal{L}_0,\mathcal{L}_1}(\boldsymbol{x})| \\ 0 & g \geq \ln((2-\eta)/\eta)/|T_{\mathcal{L}_0,\mathcal{L}_1}(\boldsymbol{x})|. \end{cases} \tag{9}$$

In order to have approximately $1 - \eta$ values of $g \cdot T_{\mathcal{L}_0,\mathcal{L}_1}(\boldsymbol{x}_i)$, $i = 1, 2, \ldots, n$, in the sigmoid transition curve, the objective is to select g such that

$$\frac{1}{n} \sum_{i=1}^{n} \Theta(\boldsymbol{x}_i) \cong 1 - \eta. \tag{10}$$

Taking a careful look at Eqs. 9 and 10, the parameter η $(0 < \eta < 0.5)$ defines the percentage (that is, $1 - \eta$) of the training examples that will be placed into the sigmoid transition curve.

To find a initial value for g, we need first to compute $\ln((2-\eta)/\eta)/|T_{\mathcal{L}_0,\mathcal{L}_1}(\boldsymbol{x}_i)|)$ for each example $\boldsymbol{x}_i$; second, select the $\lfloor \eta \cdot n \rfloor$-th value of these numbers; third, set up g as this value. The following algorithm describes this procedure.

```
1: COMPUTEG $(\mathcal{L}_0, \mathcal{L}_1, S_n, \eta)$
2: Input: two set of SLSs $\mathcal{L}_0$ and $\mathcal{L}_1$, a sample set
       $S_n$ and $\eta$ $(0 < \eta < 0.5)$.
3: Output: the constant $g$.
4: for each $\boldsymbol{x}_i \in S_n$ do
5:     $vg[i] \leftarrow \ln((2-\eta)/\eta)/|T_{\mathcal{L}_0,\mathcal{L}_1}(\boldsymbol{x}_i)|$;
6: end for
7: return the $\lfloor \eta \cdot n \rfloor$-th element of $vg$.
```

The computational complexity of the algorithm COMPUTEG($\mathcal{L}_0, \mathcal{L}_1, S_n$) is $O(n \ln(\lfloor \eta \cdot n \rfloor))$. Depending on the implementation of the algorithm to find the i-th element, the computational complexity of COMPUTEG($\mathcal{L}_0, \mathcal{L}_1, S_n$, η) can be $O(n)$.

For our experiments we choose $\eta = 0.2$, therefore, we consider approximately 80% of samples will be in the training in the transition curve.

4.3. Gradient Descent Method

The criterion to find the best positions for the extremities of the SLSs needed for the classification function is to minimize the empirical risk (Eq. 7) with respect to the SLSs in $\mathcal{L}_0$ and $\mathcal{L}_1$. In [16], Ribeiro and Hashimoto presented a training algorithm for finding these positions. However, their training algorithm is based on a heuristic. In fact, the search for this best function is a hard nonlinear optimization problem. In this section, we present a new and improved training algorithm for the SLSs technique based on gradient descent optimization method [6].

In order to find the best classification function, we need to minimize the empirical risk (Eq. 7) with respect to the SLSs in $\mathcal{L}_0$ and $\mathcal{L}_1$. In our case, the parameter vector α of Eq. 7 is defined by:

$$\alpha = \begin{bmatrix} p_h^{j,k} \\ \vdots \\ q_h^{j,k} \\ \vdots \end{bmatrix}, \begin{array}{l} j = \{0,1\}; \\ k = \{1,\ldots,|\mathcal{L}_j|\}; \\ h = \{1,\ldots,d+1\}; \end{array} \qquad (11)$$

where $p_h^{j,k}$ and $q_h^{j,k}$ are the values of the h-th coordinate of the k-th SLS $(\overline{\boldsymbol{pq}})_k \in \mathcal{L}_j$. The function VETORIZE$(\mathcal{L}_0, \mathcal{L}_1)$ in the TRAINING procedure returns a vector α as described by Eq. 11.

The gradient of the empirical risk with respect to the SLSs extremities, $\bigtriangledown_\alpha \hat{R}(\alpha)$, gives the direction in which the parameter vector α (that is, the extremities of the SLS in $\mathcal{L}_0$ and $\mathcal{L}_1$) must be displaced in order to get the most risk variation.

If $|\bigtriangledown_\alpha \hat{R}(\alpha)| = 0$, the empirical risk will not change for any infinitesimal extremities displacement in any direction. Therefore, this point may be a maximal point, minimal point or saddle point.

The adjustment of the extremities of the SLSs $\mathcal{L}_0$ and $\mathcal{L}_1$ is done by using the gradient descent method. In this methodology, the parameter vector is displaced gradually in the gradient direction for minimizing the empirical risk.

Each displacement is proportionally to γ, where $\gamma \in \mathbb{R}$ and $\gamma > 0$. If the γ is very large, then the displacement may be exaggerated making the empirical risk increasing (instead of decreasing). On the order hand, if the value of γ is very small the rate of the minimization of empirical risk may be very slow.

We use a dynamic value for γ. At each iteration, if the displacement of the SLSs decreases the empirical risk, the value of γ is increased by the factor γ_{inc} (Line 14 of the TRAINING procedure); and, conversely, if the empirical increases, then the value of γ is decreased by the factor γ_{dec} (Line 18 of the TRAINING procedure). So, in order to improve the rate of convergence, we can increase the value of γ when the displacement of the SLSs decreases the empirical risk and, on the other hand, we can decrease γ by half when the displacement of the SLSs increases the empirical risk avoiding exaggerated displacements.

```
1: TRAINING ($S_n$, $nSLS$, $i_{max}$, $\gamma_{min}$, $\gamma_{init}$, $\gamma_{inc}$, $\gamma_{dec}$,
   $\eta$)
2: Input: A sample set $S_n = \{(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \{0,1\}, i =$
          $1,\ldots,n\}$; the number $nSLS \geq 1$ of SLSs in $\mathcal{L}_0$
          and $\mathcal{L}_1$; the maximum number of iterations $i_{max}$;
          the minimum value $\gamma_{min}$ for $\gamma$ (with $0 < \gamma_{init} <$
          $\gamma_{inc}$); the initial value $\gamma_{init}$ (with $\gamma_{init} > 0$) for
          $\gamma$; constant $\gamma_{inc}$ (with $\gamma_{inc} > 0$) to increase $\gamma$;
          the constant $\gamma_{dec}$ (with $0 < \gamma_{dec} < 1$) to decrease $\gamma$ and the parameter for the sigmoid transition curve.
3: Output: two sets of SLSs $\mathcal{L}_0$ and $\mathcal{L}_1$.
4: $[\mathcal{L}_0, \mathcal{L}_1] \leftarrow$ PLACING$(S_n, nSLS)$;
5: $g \leftarrow$ COMPUTEG$(\mathcal{L}_0, \mathcal{L}_1, S_n, \eta)$;
6: $\alpha \leftarrow$ VETORIZE$(\mathcal{L}_0, \mathcal{L}_1)$;
7: $R0 \leftarrow$ compute $\hat{R}(\alpha)$;
8: $i \leftarrow 0$; $\alpha_{best} \leftarrow \alpha$; $\gamma \leftarrow 0.1$;
9: repeat
10:   $dR \leftarrow$ compute $\bigtriangledown_\alpha \hat{R}(\alpha)$;
11:   $\alpha \leftarrow \alpha - \gamma \cdot dR$;
12:   $R1 \leftarrow$ compute $\hat{R}(\alpha)$;
13:   if $R0 \geq R1$ then
14:     $\gamma \leftarrow \gamma \cdot (1 + \gamma_{inc})$ ;
15:     $\alpha_{best} \leftarrow \alpha$;
16:     $R0 \leftarrow R1$;
17:   else
18:     $\gamma \leftarrow \gamma \cdot \gamma_{dec}$ ;
19:     $\alpha \leftarrow \alpha_{best}$;
20:   end if
21:   $i \leftarrow i + 1$;
22: until $(i > i_{max})$ or $(\gamma < \gamma_{min})$;
23: return $\mathcal{L}_0$ e $\mathcal{L}_1$ extracted from $\alpha_{best}$;
```

The computational complexity of TRAINING procedure is the sum of the computational complexity of PLACING$(S_n, nSLS)$, COMPUTEG$(\mathcal{L}_0, \mathcal{L}_1, S_n, \eta)$ and the loop **repeat**...**until** (Lines 10 through 23) for the gradient descent method. Thus, the time complexity is $O(n^2 \cdot d \cdot nSLS) + O(n \ln(\lfloor \eta \cdot n \rfloor)) + O(n \cdot d \cdot nSLS \cdot i_{max})$.

5. Experimental Results

We conducted three experiments. In the first one, we used artificial datasets. There are two main purposes for this experiment: 1) to assess the behavior of SLS technique using this new training algorithm and 2) to compare this new training algorithm against the previous training algorithm described in [16]. To improve this comparison, we conducted a second experiment using the same public datasets used in [16]. For the third experiment, we used the public datasets used by Hsu and Lin used in [8] in order to evaluate the SLS method using real data and compare it to SVM method.

5.1. Artificial Datasets

The artificial datasets were created using the space $\mathbb{R}^2$ as the feature space to make possible a better analysis and visualization of the results. Considering the purpose of this experiment, we applied the SLS technique to two-class classification problem.

The probability distributions used in this experiment are defined by rectangular regions. Each region is associated to one class. Inside of each region, the probability density function is uniform. Since we known the probability distribution, we can use the Bayes classifier [5] to compute the best rate of correct classification for each artificial dataset.

Fig. 3.a represents the probability distribution called *Simple*. The intersection region has 50% probability for each class, therefore, this region has 50% probability of misclassification error. Thus, since 11.11% of examples are in the intersection region, the ideal classification function (or Bayes classifier [5]) will reach 94.44% of correct classification rate.

Using the same approach, we built three more probability distributions named *DistX*, *DistS* and *DistF* (Fig. 3.b, 3.c and 3.d, respectively). These probability distributions have ideal classification functions that reach at most 72.22% of correct classification rate for *distX*, 84.85% for *distS* and 80.00% for *distF*.

From each probability distribution, we have generated five samples with the following numbers of examples: 25, 50, 100, 200, 400, 800, 1600, 3200, making 40 samples in total. Then, for each sample, we applied the SLS technique. Since the probability distribution is known, we can use numerical integration to compute the classification rate of the classifiers obtained by the SLS technique.

In this experiment, we used as parameters for the training algorithm the following values: $i_{max} = 1000$, $\gamma_{min} = 10^{-5}$, $\gamma_{init} = 0.1$, $\gamma_{inc} = 0.1$, $\gamma_{dec} = 0.5$ and $\eta = 0.2$. By empirical observations, we noted that the training algorithm achieves a minimal local solution with these parameters values.

In order to show the classification regions obtained by the SLS technique, we use a graphic representation that exhibits the value of function $y_{\mathcal{L}_0,\mathcal{L}_1}(\boldsymbol{x})$ for all values of $\boldsymbol{x} \in \mathbb{R}^2$ in the square domain $[(0,0)..(1,1)]$. The values of function $y_{\mathcal{L}_0,\mathcal{L}_1}(\boldsymbol{x})$ are represented by a grayscale image. Since the SLSs are in $\mathbb{R}^3$, only the projection of the SLSs is shown on the graphic representation. Figs. from 3.e to 3.h are examples of this graphic representation.

The experiments were conducted as follow: (a) one using probability distribution called *Simple* with 1 SLS; (b) another one using probability distribution *DistX* with 2 SLSs for each class; (c) another one using distribution *DistS* with 1 SLS for each class and finally (d) three experiments using probability distribution *DistF* with 2, 3, and SLSs for each class.

In order to evaluate our training algorithm, we also applied the previous algorithm presented in [16], but, instead of using their placing algorithm [16], we used ours with a fixed number of SLSs. So, the comparisons were done using always the same output of the placing algorithm for both algorithms.

Table 1 shows all results with artificial datasets. The columns labeled “Heur.” (“Grad.”, respectively) indicate the results obtained by the previous (our, respectively) training algorithm. All the results on this table are average of five run tests. The boldface numbers indicate the best classification rates. Note that all results are very close to the ideal classification rate and the training using gradient descent usually obtains the best performance.

5.2. Public Datasets

In order to have a systematic comparison to the previous training algorithm, we did experiments with the same public datasets used in [16]. Thus, we applied the SLS technique using leave-one-out estimation error to wine and breast-cancer datasets obtained from [1]. For this experiment, as we did with the artificial datasets, we used the same output of placing algorithm for both training methods (the previous one and the gradient descent). In this experiment, we used from 1 to 4 SLSs for each class. The obtained results are shown in Table 2. Note that, for both training algorithms, we have comparable performances, but, again, the training algorithm using gradient descent usually provides the best performance.

To compare the SLS classification technique using this new training algorithm to another one (Support Vector Machine -SVM- method), we used the same datasets that Hsu and Lin used in [8]. The datasets are from *Statlog* [14] and from *UCI Repository* [1]. All these datasets are accessible at *LIBSVM* website [2]. Table 3 shows the characteristics of these datasets. Since all these datasets have more than two class, we have used the method *One-Against-All* [8] for classification. In this method, we solve one two-class classification problem for each class: one class against the other ones. Some of these datasets have separated training and test data; in this case, we used cross-validation in order to estimate the classification rate. For the others, which have just training data, we used *10-fold-cross-validation*, as Hsu and Lin used in their work [8]. We always select a value to i_{max} large enough to guarantee convergence of the empirical risk to a local minimum. All other input parameters for the training algorithm have the same values that we used in the artificial data experiments. The comparison of our results to the ones obtained by Hsu and Lin [8] for the one-against-all classification are summarized in Table 3.

The datasets *vehicle*, *segment*, *satimage* and *letter* are applications from Image Processing and Computer Vision areas. The experiment using *vehicle* dataset is an application to identify the model of vehicles by inspection 2D images; the first aim of this experiment is to identify 3D objects from 2D images. The objective of the experiment using *segment* database is to classify seven different types of textures. The purpose of the experiment using *satimage* dataset is to process multispectral satellite images. Finally, the experiment using *letter* database is the classical problem of handwriting recognition.

The experiments with the public datasets show that the

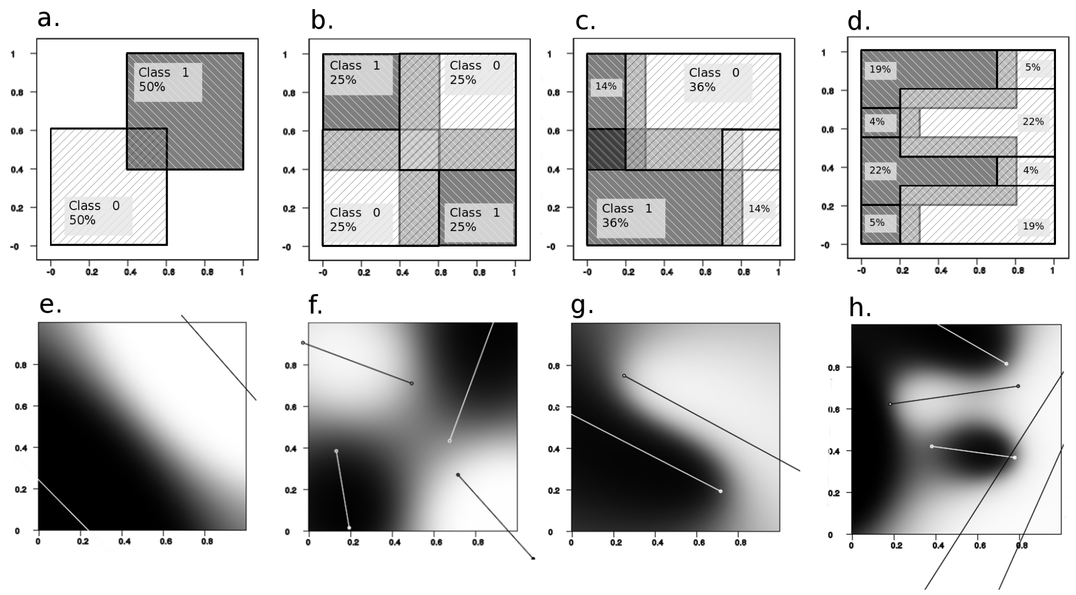

Figure 3. On top, probability distribution called: a) *Simple*, b) *DistX*, c) *DistS* and d) *DistF*.
On bottom, representation of classification regions obtained by the SLS technique using training sets drawn from probability distribution called: e) *Simple*, using 1 SLS for each class and a sample with 3200 examples; f) *DistX*, using 2 SLSs for each class and a sample with 1600 examples; g) *DistS*, using 1 SLS for each class and a sample with 3200 examples and h) *DistF*, using 3 SLSs for each class and a sample with 1600 examples.

	Simple (1)		DistX (2)		DistS (1)		DistF (2)		DistF (3)		DistF (4)	
# of examples	Heur.	Grad.	Heur.	Grad.	Heur.	Grad.	Heur.	Grad.	Heur.	Grad.	Heur.	Grad.
25	**93.99**	93.27	65.09	**68.82**	79.42	**79.71**	**69.53**	69.49	**72.50**	69.21	70.51	**70.69**
50	**94.09**	92.85	64.95	**69.29**	80.70	**83.10**	67.95	**71.47**	69.01	**73.25**	70.30	**71.96**
100	**94.21**	92.95	64.43	**71.87**	80.83	**83.57**	71.46	**76.76**	73.80	**76.70**	75.24	**75.83**
200	93.97	**94.24**	71.82	**72.11**	82.34	**83.15**	70.27	**73.98**	74.09	**77.41**	75.77	**78.19**
400	**94.28**	94.23	71.35	**72.22**	82.14	**83.67**	72.06	**77.69**	73.36	**76.55**	77.26	**78.03**
800	**94.37**	94.36	71.76	**72.22**	82.13	**84.09**	73.91	**76.83**	74.48	**78.80**	73.81	**79.43**
1600	94.41	**94.43**	72.21	**72.22**	82.96	**84.20**	74.88	**77.32**	72.63	**78.27**	76.17	**79.23**
3200	94.36	**94.43**	71.99	**72.22**	82.96	**84.55**	**75.85**	75.64	72.40	**78.09**	75.24	**79.33**
Ideal Classification Rate (%)	94.44		72.22		84.85		80.00					

Table 1. Experimental results using artificial datasets.

SLS technique has a good classification rate in comparison to the SVM method. Many classification rates obtained by SLS technique were close to the SVM ones obtained by Hsu and Lin [8].

6. Conclusion

This paper presents a new training algorithm for the SLS technique presented in [16]. In fact, the key issue in this technique is to find two collections of SLSs such that the classification function minimizes a certain risk function. The search for this best classification function is a hard non-linear optimization problem. The main contribution of this work is to present a new and improved training algorithm for the SLSs technique based on gradient descent optimization method. Besides, this training algorithm is easier to implement compared to the one presented in [16]. Both algorithms have good results for classification, but usually, the new algorithm uses less SLSs, and thereby, leading to lower computational complexity for evaluating new query points. The experiments using public datasets show the viability of the SLS technique. For the most datasets, we achieved very good classification rates similar to SVM method, even when the training algorithm does not guarantee the best classification function. For future work, we will research a training algorithm that guarantees the optimum position for SLSs

Database	training	test	class	features	SVM [8]	1-SLS	2-SLS	3-SLS	k-SLS
iris	150	–	3	4	96.66	**96.25**	95.00	93.75	
wine	178	–	3	13	98.87	97.78	**98.33**	97.78	
glass	214	–	6	13	71.93	69.55	**73.18**	70.00	
vowel	528	–	11	10	98.48	82.08	85.47	**86.79**	
vehicle	846	–	4	18	87.47	**74.82**	73.53	73.88	
segment	2310	–	7	19	97.53	**93.45**	93.15	93.23	
satimage	4435	2000	6	36	91.70	89.33	90.21	**91.27**	
letter	15000	5000	26	16	97.88	73.73	82.33	82.33	**82.92** (k=8)
shuttle	43500	14500	7	9	99.91	99.60	99.56	**99.71**	99.68 (k=6)

Table 3. Datasets features and obtained results.

	Wine		**Breast-Cancer**	
# of examples	178		682	
# of classes	3		2	
# of SLSs	Heur.	Grad.	Heur.	Grad.
1	96.07	**98.32**	96.34	**96.78**
2	97.75	**97.76**	96.34	**96.92**
3	97.75	**98.32**	**96.63**	96.34
4	**99.44**	98.32	96.49	**96.78**

Table 2. Experimental results for Wine and Breast-Cancer datasets for the training algorithms using leave-one-out estimation error.

and modify the SLS machine learning to a more general method.

7. Acknowledgements

This work was partially funded by UOL ("Universo Online"), CNPq (Brazilian Council for Research and Development), CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) and FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo).

References

[1] A. Asuncion and D. J. Newman. UCI Machine Learning Repository, [http://archive.ics.uci.edu/ml/], 2007. Irvine, CA: University of California, School of Information and Computer Science.

[2] C.-C. Chang and C.-J. Lin. *LIBSVM: A Library for Support Vector Machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[3] K. Chen, T. Y. Wu, and H. J. Zhang. On the Use of Nearest Feature Line for Speaker Identification. *Pattern Recognition Letters*, 23:1735–1746, 2002.

[4] T. M. Cover and P. E. Hart. Nearest Neighbour Pattern Classification. *IEEE Transactions on Information Theory*, (13):21–27, January 1967.

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

[6] B. S. Gottfried and J. Weisman. *Introduction to Optimization Theory*. Prentice-Hall, 1973.

[7] H. Du and Y. Q. Chen. Rectified Nearest Feature Line Segment for Pattern Classification. *Patter Recognition*, 40:1486–1497, 2007.

[8] C.-W. Hsu and C.-J. Lin. A Comparison of Methods for Multiclass Support Vector Machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

[9] J. H. Chen and C. S. Chen. Object Recognition Based on Image Sequences by Using Inter-Feature-Line Consistencies. *Pattern Recognition*, 37:1913–1923, 2004.

[10] J. T. Chien and C. C. Wu. Discriminant Waveletfaces and Nearest Feature Classifiers for Face Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:1644–1649, 2002.

[11] A. K. Jain, M. N. Murty, and P. J. Flynn. Data Clustering: a Review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[12] S. Li, K. Chan, and C. L. Wang. Performance Evaluation of the Nearest Feature Line Method in Image Classification and Retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1335–1339, 2000.

[13] S. Li and J. Lu. Face Recognition Using the Nearest Feature Line Method. *IEEE Transactions on Neural Networks*, 10(2):439–443, Mar, 1999.

[14] D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.

[15] Qing-bin Gao and Zheng-Zhi Wang. Center-Based Nearest Neighbor Classifier. *Pattern Recognition*, 40:346–349, 2007.

[16] J. H. B. Ribeiro and R. F. Hashimoto. A New Machine Learning Technique Based on Straight Line Segments. In *ICMLA '06: Proceedings of the 5th International Conference on Machine Learning and Applications*, pages 10–16, Washington, DC, USA, 2006. IEEE Computer Society.

[17] V. N. Vapnik. An Overview of Statistical Learning Theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999.

[18] J. W. Y. Zhou, C. Zhang. *Extended Nearest Feature Line Classifier*, volume 3157. Springer Berlin / Heidelberg, 2004.

[19] W. Zheng, L. Zhao, and C. Zou. Locally Nearest Neighbor Classifiers for Pattern Classification. *Pattern Recognition*, 37:1307–1309, 2004.