FAST TRAINING AND EFFICIENT LINEAR LEARNING MACHINE

Abdenour BOUNSIAR, Pierre BEAUSEROY and Edith GRALL

Université de Technologie de Troyes
Institut des Sciences et Technologies de l'Information de Troyes (CNRS FRE 2732)
Équipe Modélisation et Sûreté des Systèmes
12 rue marie curie, 10010, Troyes Cedex, France
email: {abdenour.bounsiar, pierre.beauseroy, edith.grall}@utt.fr
phone: + (33) 325718450, fax: + (33) 325715699, web: www.utt.fr

ABSTRACT

Time complexity is a challenge for learning machines. In this paper, a fast training and efficient linear learning machine is presented. Starting from a simple linear classifier, a new one is proposed based on an improvement on the first one. The machine obtained is characterized by a weight vector that can be processed immediately without any complex calculus or optimization step, which allows for considerable training time savings. A geometric interpretation of the proposed method is given. Experiments show that this classifier is competitive to other state of the art linear learning methods such as Support Vector Machines and Kernel Fisher Discriminant.

1. INTRODUCTION

In statistics, the problem of guessing or estimating a decision function from a set of input-output pairs is called *supervised learning*. The function is determined from a given *training set* that contains n input-output pairs $\{(x_i, y_i)\}_{i=1}^{n}$, where x_i are real vectors and y_i are discrete scalars.

For supervised binary classification, the outputs or the labels y_i take generally the value -1 or $+1$. We then speak respectively about *positive* and *negative classes*. The classification is made by using a real-valued function f so that a pattern $x \in \mathbb{R}^d$ is assigned to the positive class , if $f(x) \geq 0$, otherwise it is assigned to the negative class.

Statisticians and neural network researchers have largely used this simple kind of classifier, calling them respectively *linear discriminants* and *perceptrons*. The theory of linear discriminants was developed by Fisher in 1936 [1] [2], while neural network researchers studied perceptrons in the early 1960s. Both consider the case where $f(x)$ is a linear function of x, so that it can be written as $f(x) = \langle w, x\rangle + b$, where $(w, b) \in \mathbb{R}^d \times \mathbb{R}$ are the parameters of the function and are referred to as *weight vector* and *bias*, respectively. The input space X is split into two parts by the hyperplane defined by the equation $f(x) = 0$, each subspace corresponds to the decision of a distinct class.

Research in linear classifiers has been recently revamped by the popularity of kernel methods [3][4], a set of mathematical tools used to efficiently represent complex nonlinear decision surfaces in terms of linear classifiers in a high-dimensional feature space defined by kernel functions. Using such methods, more efficient linear learning machines have been developed such as SVM (*Support Vector Machines*) [5] and KFD (*Kernel Fisher Discriminant*) [6].

Time complexity is one of the challenges of linear learning machines. Many suffer from a rapid growth of time complexity with the growth of the number of training patterns. In this paper, a fast training linear learning machine is proposed. It uses kernel feature spaces yielding an efficient and highly flexible classifier.

In the next section, a simple linear classifier is presented. Then in section 3, the proposed method is developed by further improvements and new contributions to the formulation of the earlier one. The validity of these contributions is then verified by geometric interpretations. In section 4, experiments are carried out in order to compare the proposed method with other state of the art classifiers on different standard data bases. Conclusions and perspectives are given in section 5.

2. A SIMPLE LINEAR CLASSIFIER

In [4], authors have presented a simple linear classifier. The basic idea is to assign a new pattern to the class with closer mean. The means of the two classes are estimated from training samples, they are denoted $c_+ = \sum_{\{y_i=+1\}} \frac{x_i}{m_+}$ and $c_- = \sum_{\{y_i=-1\}} \frac{x_i}{m_-}$ for classes with positive and negative labelled samples respectively, where m_+ and m_- are the number of positive and negative labelled training patterns. Half way between c_+ and c_- lies the point $c = (c_+ + c_-)/2$. The class of an input x is determined by comparing the absolute angle between the vector $x - c$ and the vector $w = c_+ - c_-$ to $\pi/2$. This leads to the output of the classifier :

$$f(x) \quad = \quad \text{sign}\left(\langle x - c, w\rangle\right)$$

$$= \text{sign}\Big(\sum_{\{i,y_i=+1\}} \frac{\langle x, x_i \rangle}{m_+} - \sum_{\{i,y_i=-1\}} \frac{\langle x, x_i \rangle}{m_-} + b \Big),$$

with $b = \frac{1}{2}(\|c_-\|^2 - \|c_+\|^2)$. Other values of the bias may lead to better performances.

In general, real world applications require discriminant functions that are more complex than linear ones. Kernel representations offer a solution by projecting the data from X into a high dimensional *feature space* $F = \{\phi(x) | x \in X\}$. The mapping $\phi(.)$ is performed by a kernel function $K_\theta(.,.)$ depending on a set of parameters θ, such that $K_\theta(x,y) = \langle \phi_\theta(x), \phi_\theta(y) \rangle$ defines the dot product in that space. The kernels having these properties satisfy the Mercer's conditions [5]. Using such kernels, the decision rule of the previous classifier can be expressed as :

$$\sum_{\{i,y_i=+1\}} \frac{K_\theta(x,x_i)}{m_+} - \sum_{\{i,y_i=-1\}} \frac{K_\theta(x,x_i)}{m_-} + b \underset{D_-}{\overset{D_+}{\gtrless}} 0, \quad (1)$$

where D_+ and D_- are the decisions to affect a pattern to the positive and negative class, respectively.

3. THE PROPOSED METHOD

3.1. Description of the method

Assuming that :

- K_θ is a probability density i.e., it is positive and has unit integral: $\int_X K_\theta(x,y)dx = 1$ for all $y \in X$.
- the conditional probability density of each classes (+ or -) is estimated by the *Parzen windows* estimator :

$$\hat{p}_\pm(x) \sim \sum_{\{i,y_i=\pm 1\}} \frac{K_\theta(x,x_i)}{m_\pm}$$

- $b = 0$.

Thus equation (1) takes the form :

$$\frac{\hat{p}_-(x)}{\hat{p}_+(x)} \underset{D_+}{\overset{D_-}{\gtrless}} \lambda, \quad (2)$$

with $\lambda = 1$, which corresponds to a likelihood ratio based classifier. Varying the decision threshold λ in $[0, +\infty[$, (2) may embrace a large scope of likelihood ratio based decision rules (Bayes rule, Neyman-Pearson test, Mini-Max test).

Using a parameter $\rho \in [0, 1[$, decision rule (2) with $\lambda \in [0, +\infty[$ becomes :

$$\frac{\hat{p}_-(x)}{\hat{p}_+(x)} \underset{D_+}{\overset{D_-}{\gtrless}} \frac{\rho}{1-\rho}, \quad (3)$$

which gives the decision rule :

$$\rho\hat{p}_+(x) - (1-\rho)\hat{p}_-(x) \underset{D_-}{\overset{D_+}{\gtrless}} 0.$$

This corresponds to a linear classifier in the feature space without bias ($b = 0$) and the following weight vector :

$$\begin{aligned} w_\theta &= \rho \sum_{\{i,y_i=+1\}} \frac{\phi_\theta(x_i)}{m_+} - (1-\rho) \sum_{\{i,y_i=-1\}} \frac{\phi_\theta(x_i)}{m_-} \\ &= \rho C_+ - (1-\rho) C_-, \end{aligned} \quad (4)$$

where C_+ and C_- are the class means in the feature space.

3.2. The decision rule

The decision rule (4) corresponds to a linear classifier without bias. Generally, for linear classifiers, the bias is one of the classifier parameters that must be optimized in order to obtain high performances. Considering a bias in (4) leads to :

$$\begin{aligned} \rho\hat{p}_+(x) \quad - \quad & (1-\rho)\hat{p}_-(x) + b \underset{D_-}{\overset{D_+}{\gtrless}} 0 \Leftrightarrow \\ \frac{\hat{p}_-(x)}{\hat{p}_+(x)} \quad & \underset{D_+}{\overset{D_-}{\lessgtr}} \quad \frac{\rho}{1-\rho} + \frac{b}{(1-\rho)\hat{p}_+(x)} \Leftrightarrow \\ \frac{\hat{p}_-(x)}{\hat{p}_+(x)} \quad & \underset{D_+}{\overset{D_-}{\lessgtr}} \quad \frac{\rho}{1-\rho} + \delta(x). \end{aligned} \quad (5)$$

Equation (5) is a likelihood ratio based decision rule, where the probability densities are estimated by *Parzen windows* estimators. The decision threshold consists of a constant term that is defined through the parameter ρ, and a variable term $\delta(x)$ that depends on ρ, the bias b and the pattern considered.

The result obtained is interesting because $\delta(x)$ can be interpreted as a correction term to the decision rule (2). Once the probability density functions are estimated by choosing the convenient kernel function K_θ, the decision thre- shold giving the best performance is defined through the parameter ρ, then the bias b is chosen to give the best average correction to the decision threshold. Note that because the estimation error of the likelihood ratio $\hat{p}_-(x)/\hat{p}_+(x)$ is a function of the pattern x, the correction of the decision threshold must also be a function of pattern x.

Decision rule (5) can be reformulated as :

$$\begin{aligned} \rho\Big[\hat{p}_+(x) + b\Big] - (1-\rho)\Big[\hat{p}_-(x) - b\Big] \quad & \underset{D_-}{\overset{D_+}{\gtrless}} \quad 0 \Leftrightarrow \\ \frac{\hat{p}_-(x) - b}{\hat{p}_+(x) + b} \quad & \underset{D_+}{\overset{D_-}{\lessgtr}} \quad \frac{\rho}{1-\rho}. \end{aligned}$$

In this case, the bias modifies the discriminant function rather than the decision threshold. It can be considered as an offset for the two estimated probability density functions, which gives a correction to the estimated likelihood ratio. For different patterns x, this correction is not the same. Using this formulation or that of (5), the bias b appears to provide decision rule (3) with a correction of the estimation errors.

Figure 1 shows the separation boundaries (dash-dot line and solid line) obtained on a toy data, by using (3) and (5),

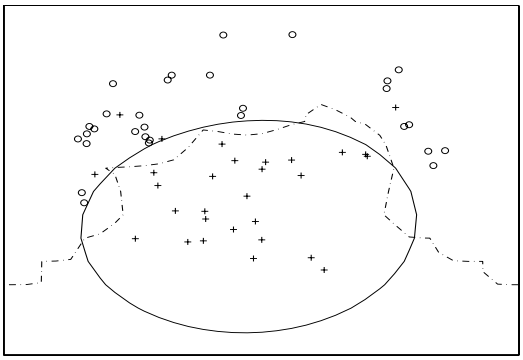

Fig. 1. Separation boundaries obtained by (3)(dash-dot) and (5)(solid) for a small training set (30 samples per class).

respectively. The data consists of two classes. One has a uniform disc probability density function, and the other one has a noisy bow probability density function, situated on the top edge of the first one. The RBF kernel is used :

$$K_\sigma(x, y) = \exp\left(-\sigma\|x - y\|^2\right). \quad (6)$$

The boundary obtained by (5) is smoother and discriminates the two classes in an almost optimal way. This example shows the beneficial effect of $\delta(x)$ in (5) on the correction of the decision rule (3) to get a better classifier. Note that the training set is small (30 samples per class). The optimal parameters of the two classifiers are $\sigma = 5$ (kernel function width (6)) for (3), $\sigma = 0.1$ and $\rho = 0.7$ for (5). These parameters were optimized on a separate validation set to give minimum error using a grid of parameters. It is to be noticed that these two classifiers do not have the same value σ, which means that the classifier (5) is not only the extension of (2) to the case with bias, but also defines a different classifier.

3.3. Geometric interpretation

The family of classifiers defined by (1) is included in the proposed more general family of classifiers defined by (5). Not taking into consideration parameter estimation problem, this category of classifiers achieves better performances. The estimated hyperplane separating the two classes is perpendicular to the weight vector w_θ whose direction varies from $-C_-$ to C_+ when varying the value of the parameter ρ from 0 to 1, as illustrated in figure 2. The position of this hyperplane in w_θ's direction is set by the value of the bias b.

Figure 3 shows on a two dimensional case, the influence of parameter ρ on the determination of the separating hyperplane. The hyperplane $\langle w_\theta, \phi_\theta(x)\rangle + b = 0$ given by (1) clearly gives poor separation. The best separating hyperplane in the sense of minimum error $\langle w'_\theta, \phi_\theta(x)\rangle + b' = 0$ is a member of the family of hyperplanes given by (5), it can be obtained by rotation of the first hyperplane (rotation angle ω in figure 3) and a suitable value of b. Note that for

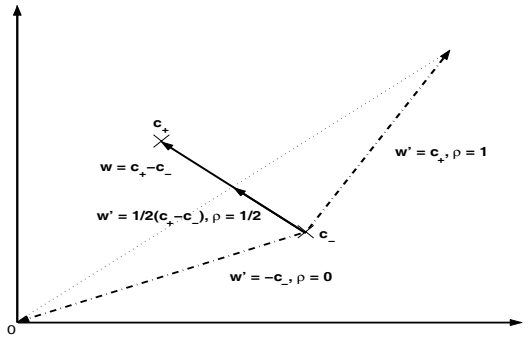

Fig. 2. Varying the variable ρ from 0 to 1, makes the weight vector w' varying from $-c_-$ to c_+.

bi-dimensional classification problems such as the one of figure 3, the weight vector of the best separating hyperplane is obviously in the plane defined by (C_+, C_-) except if these two vectors are collinear. This is not necessarily true for higher dimensional problems, in such cases the best separating hyperplane may not be member of the family of hyperplanes defined by (5).

Special attention as to the choice of the kernel function K_θ is necessary to verify that the two centers C_+ and C_- are not collinear. If they are, the rotation of the weight vector w_θ in (4) will not be possible.

3.4. The case of a RBF kernel

In the case of a RBF kernel (6), since $K_\sigma(x, x) = 1$ for all patterns x, all the vectors $\phi_\sigma(x_i)$ in the feature space are located on a unit radius hyper-sphere: $\|\phi_\sigma(x_i)\| = 1, \forall i$. Furthermore, for all patterns x and y: $0 \leq \widehat{\phi_\sigma(x)\phi_\sigma(y)} < +\pi/2$ because $0 < K_\sigma(x, y) \leq 1$. This means that all data in feature space are located on a surface delimited by a solid angle of $\pi/2$. So, the two centers can be collinear only in the case where $C_+ = C_-$, a situation that is extremely improbable.

4. EXPERIMENTS

In order to evaluate the performance of the proposed method, we compared it to other state of the art classifiers : Kernel Fisher Discriminant, regularized AdaBoost and Support Vector Machines [6][7]. We used 13 artificial and real world data sets from the **UCI**, **DELVE** and **STATLOG** benchmark repositories (except for **banana**) [7]: **banana, breast cancer, diabetis, german, heart, image, ringnorm, flare solar, splice, thyroid, titanic, twonorm** and **waveform**. Some of these problems are originally not binary classification ones, hence a random partition into two classes is used. For each data sets 100 pairs of test and training sets were created. The proposed method was trained and tested on each of these set pairs.

For each data base, the parameters (σ, ρ, b) of the proposed classifier were optimized on the first five pairs of train-

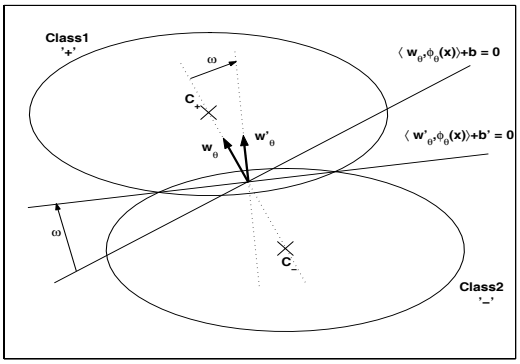

Fig. 3. The best separating hyperplane may be obtained by rotation of the one of (1).

ing and test sets using a grid of values for σ and ρ. For each pair of sets the classifier was trained with each couple of values (σ, ρ) and then we searched the value of b by minimizing the average of the five validation errors obtained with the first five test sets (each one with its corresponding training set). Finally, the model parameters σ, ρ and b are chosen to minimize the average error rate with the first five pairs of test sets. The results obtained are summarized on table 1. The values on the table represent mean and standard-deviation of the test errors for each data using our method (METH.). Results for other methods are found in [6].

The results show that the proposed method is competitive and in some cases even better than the other methods on almost all data sets except for **diabetis** and **splice**. Considering these results and despite the fact that the best solution for high dimensional classification problems may not have been obtained using (5), for such cases by choosing adequate mapping ϕ_θ, we suggest that the best possible solution can be approximated.

It should be noticed that almost all the data bases for which the proposed method is better, KFD comes second except for **titanic**. It may be due to the fact that KFD like the proposed method has no support vectors. Both use all training patterns in the formulation of decision function. In addition, the training of the coefficients of the weight vector of the proposed classifier is immediate whatever the size of the training set; a feature giving an advantage missing in other methods.

5. CONCLUSION

In this paper, a new linear learning machine has been proposed. This classifier uses kernel feature spaces which yield a highly flexible algorithm : this is competitive with other kernel based algorithms. The expression of the weight vector is a weighted combination of the means of the two classes. Consequently its training is immediate whatever is the size of the training set, contrary to other algorithms such as SVM for which the training time complexity is $\mathcal{O}(n^2)$, where n is the number of the training patterns.

	AB_R	SVM	KFD	METH.
Banana	10.9 ± 0.4	11.5 ± 0.7	10.8 ± 0.5	$\mathbf{10.8 \pm 0.4}$
B.Cancer	26.5 ± 2.3	26.0 ± 4.7	25.8 ± 4.6	$\mathbf{24.8 \pm 4.2}$
Diabetes	23.8 ± 1.8	23.5 ± 1.7	$\mathbf{23.2 \pm 1.6}$	26.2 ± 2.4
German	24.3 ± 2.1	$\mathbf{23.6 \pm 2.1}$	23.7 ± 2.2	23.9 ± 2.4
Heart	16.5 ± 3.5	$\mathbf{16.0 \pm 3.3}$	16.1 ± 3.4	16.9 ± 3.8
Image	$\mathbf{2.7 \pm 0.6}$	3.0 ± 0.6	4.8 ± 0.6	4.6 ± 0.7
Ringnorm	1.6 ± 0.1	1.7 ± 0.1	1.5 ± 0.1	$\mathbf{1.4 \pm 0.0}$
F.Solar	34.2 ± 2.2	$\mathbf{32.4 \pm 1.8}$	33.2 ± 1.7	33.9 ± 1.8
Splice	$\mathbf{9.5 \pm 0.7}$	10.9 ± 0.7	10.5 ± 0.6	12.8 ± 1.0
Thyroid	4.6 ± 2.2	4.8 ± 2.2	4.2 ± 2.1	$\mathbf{3.9 \pm 2.0}$
Titanic	22.6 ± 1.2	22.4 ± 1.0	23.2 ± 2.0	$\mathbf{21.8 \pm 1.1}$
Twonorm	2.7 ± 0.2	3.0 ± 0.2	2.6 ± 0.2	$\mathbf{2.4 \pm 0.1}$
Waveform	$\mathbf{9.8 \pm 0.8}$	9.9 ± 0.4	9.9 ± 0.4	10.9 ± 0.8

Table 1. Comparison between the proposed method, KFD, SVM and AB_R (see text). Best results are in bold face and underlined.

However while the space complexity of SVM scales with the number of support vectors, the complexity of the proposed method is $\mathcal{O}(n)$ depending on the number of training patterns because all of them are used in the formulation of the decision function. Future work will focus on the space complexity reduction of the proposed method.

6. REFERENCES

[1] Ronald A. Fisher, "The use of multiple measurments in taxonomic problems," *Annals of Eugenics*, vol. 7, pp. 179–188, 1936.

[2] Ronald A. Fisher, "The statistical utilization of multiple measurments," *Annals of Eugenics*, vol. 8, pp. 376–385, 1937.

[3] J. Shawe-Taylor and N. Cristianini, *Kernel methods for pattern analysis*, Cambridge University Press, Cambridge, UK, 2004.

[4] B. Schölkopf and A. J. Smola, *Learning with kernels*, MIT Press, MA, 2002.

[5] V. Vapnik, *The Nature of Statistical Learning Theory*, Spring Verlag, 1995.

[6] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller, "Fisher discriminant analysis with kernels," *Neural Networks for Signal Processing*, vol. IX, pp. 41–48, 1999.

[7] G. Rätsch, T. Onoda, and K.-R. Müller, "Soft margins for adaboost," *Machine Learning*, pp. 1–35, 2000.