New features of the Statistical Pattern Recognition Toolbox [1)]

Vojtěch Franc and Václav Hlaváč
Czech Technical University, Faculty of Electrical Engineering
Department of Cybernetics, Center for Machine Perception
121 35 Praha 2, Karlovo náměstí 13, Czech Republic
{xfrancv,hlavac}@cmp.felk.cvut.cz, http://cmp.felk.cvut.cz

1 Introduction

In this contribution, our Statistical Pattern Recogntion Toolbox (abbreviated STPR toolbox, built on top of Matlab, freely downloadable and usable for academic or non-profit purposes) is briefly sketched and namely its newly developed features are introduced. The STPR toolbox was created within a diploma thesis project of the first author in early 2000 [FHS00] as a demonstration tool of a monograph M.I. Schlesinger, V. Hlaváč: Ten lectures on the statistical and pattern recognition [SH99]. The STPR toolbox has been developed since then and other methods that we found interesting were included. We focus primarily to these additions in this paper.

The aim of the toolbox is (i) to help the reader of the monograph to understand the pattern recognition algorithms, (ii) to provide to the user the tool for experimenting with various pattern recognition methods, and (iii) to provide the teachers of pattern recognition courses the tool they can use with their students in laboratory exercises.

2 The toolbox overview

We first briefly list methods which have been included in the STPR toolbox at the time it was written in February 2000. The newly added algorithms will be discussed in the coming sections.

The first part of the STPR toolbox consists of the methods for synthesis of the linear discriminant function. Both the algorithms for learning from finite point sets and infinite point sets

[1)] This work was supported by the Czech Ministry of Education, project Center for Applied Cybernetics, LN00B096.

(mixtures of Gaussians) are implemented. The main representatives of this part deal with finite point sets are Perceptron learning rule, Kozinec's algorithm, algorithms finding Fisher's classifiers and linear version of the Support Vector Machines. The second part deals with infinite point sets, e.g. mixtures of Gaussians, and its core is the Generalized Anderson's task proposed by M.I. Schlesinger. This task belongs to the class of non-Bayesian approaches.

The third part of the toolbox is devoted to synthesis of the quadratic discriminant function. It contains functions which map the original feature space into the new one with higher dimension. In the new feature space, linear algorithms can be used for effective recognition.

The fourth part of the STPR toolbox deals with learning algorithms, i.e. the algorithms which build a statistical model from given training data. The Minimax learning algorithms and several versions of the Expectation-Maximization (EM) algorithm are implemented. The Minimax learning algorithm estimates normally distributed statistical models for the case when training set is labeled and the data are not randomly selected. Implemented versions of the EM algorithm find statistical model as a weighted mixture of Gaussians from unlabeled data.

3 New features in the STPR toolbox

The toolbox has been extended by the EM algorithm finding conditionally independent statistical model, the Kernel Principal Component Analysis method and the implementation of the Support Vector Machines. We will give brief description of these methods in the following three subsections.

3.1 EM algorithm for conditionally independent statistical model

The EM algorithm searches for the statistical model m^* by solving the following optimization problem

$$m^* = \operatorname*{argmax}_m \sum_{i=1}^{n} \log \sum_{k \in K} p(k) p(x_i | k, a_k) \, .$$

The model $m = (p(k), a_k | k \in K)$ is described by a priori probabilities of classes $p(k)$, $k \in K$, and by parameters a_k, $k \in K$, which determine a conditional probabilities $p(x_i|k, a_k)$. Training data $x_i, i = 1, 2, \ldots, n$, are supposed to be independently and identically distributed feature vectors which represent the examined problem.

The most often used model is a mixture of Gaussians or, more generally, a model from the exponential family. The algorithm that finds weighted mixture of Gaussians for correlated and uncorrelated data is implemented in the toolbox as well. But there exist other less-known model for which the EM algorithm can be used. One of them is a model we call the *conditionally independent model*, which is defined as follows.

Suppose that the examined object is described by two observable features $x \in X$ and $y \in Y$ and by an unobservable state $k \in K$. The sets X, Y and K are supposed to be finite. The features x and y are conditionally independent, i.e. $p(x, y|k) = p(x|k) \cdot p(y|k)$, $\forall x \in X$, $\forall y \in Y$, $\forall k \in K$.

It has been proven that for the case of two states only, i.e. $k = \{1, 2\}$, the EM algorithms finds a *global maximum* of the likelihood function [Sch97]. This conclusion does not hold for a model based on the mixture of Gaussians. The conditionally independent model has several further interesting properties which are analyzed in [SH99]. In addition to the mentioned EM algorithm, the STPR toolbox contains useful functions for visualization and for working with obtained results.

3.2 Kernel Principal Component Analysis

The method called Kernel Principal Component Analysis (Kernel PCA) [SSM98, RLdT01] implies two steps. The first one is a non-linear mapping of the feature space which generalizes linear algorithms (e.g. Perceptron learning rule) to non-linear algorithms. The second step is the standard Principal Component Analysis (PCA). In the non-linear mapping method, vectors from the original feature space $x \in X$ are mapped by the function $\phi \colon X \rightarrow Y$ onto the vectors $y \in Y$ from the new feature space of considerably higher dimension.

The PCA is a well-known statistical analysis technique which linearly projects data onto the principal subspace. The PCA is based on finding the eigenvalues and eigenvectors of the covariance matrix $C = \frac{1}{N} Y Y^T$, where $Y = \{y_1, y_2, ..., y_N\}$ are the training column vectors.

Performing these two steps separately can become computationally demanding because the dimension of the new feature space Y can be very high or even infinite. For instance, the dimension of the new feature space for a polynomial function $\phi(x)$ is equal to $\left(\begin{array}{c} d+p-1 \\ p \end{array} \right)$, where p is a degree of the polynomial and d is a dimension of the original feature space. Thus for the polynomial of degree $p = 4$ and data of dimension $d = 256$ is $dim(Y) = 183,181,376$.

A way how to avoid this problem is to use kernel functions. It is known that linear algorithms, which depend only on the data by dot products, denoted $\langle x_i, x_j \rangle$, can be made non-linear by replacing dot products by the *kernel function* $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$. Its computation does not require an explicit mapping of vectors x_i, x_j into a higher dimensional space. This relation is expressed by the function $k(x_i, x_j)$ in an easier manner. It has been proven in [SSM98] that there is one-to-one mapping between the non-zero eigenvectors of the matrix $C = \frac{1}{N} X X^T$ and the non-zero eigenvectors of matrix $K = \frac{1}{N} X^T X$. As mentioned above, the matrix K can be substituted by the kernel matrix $K_{i,j} = \langle \phi(x_i), \phi(x_j) \rangle = k(x_i, x_j)$ in the non-linear case.

The toolbox contains standard kernels (i.e. polynomial, quadratic, Gaussian Radial Basis functions, sigmoid kernels, splines and Fourier expansion) but other ones can be simply added. The Kernel PCA essentially extends capability of the linear algorithms implemented in the toolbox, i.e. Perceptron learning rule, Kozinec's algorithm, Fisher's classifiers or algorithms for Anderson's task.

3.3 Support Vector Machines for classification problems

We implemented Support Vector Machines (SVM) [Vap95], which is the training algorithm having two main features: (1) it automatically tunes the capacity of the classifier (trade-off between the ability to learn complicated discriminant functions and the ability of generalization) by maximizing the margin between the classes, (2) it allows to learn variety of non-linear discriminant functions by the use of non-linear kernels (cf. Section 3.2).

The SVM in its basic form learns a classifier from two training point sets X_1 and X_2. Considering the linear case, the SVM finds such decision hyperplane, which maximizes a margin between the training classes and minimizes an overlap of the training points. This learning task can be expressed as the quadratic optimization problem

$$(w, b) = \operatorname*{argmin}_{w,b} \frac{1}{2} ||w||^2 + C \sum_i \xi_i \tag{1}$$

with linear constrains

$$\begin{aligned}
\langle x_i, w \rangle + b &\geq +1 - \xi_i \,, \qquad \text{for} \quad x_i \in X_1 \,, && (2) \\
\langle x_i, w \rangle + b &\geq -1 + \xi_i \,, \qquad \text{for} \quad x_i \in X_2 \,, && (3) \\
\xi_i &\geq 0 \,, \quad \forall i \,, \\
C &> 0 \,,
\end{aligned}$$

where w is a normal vector of the decision hyperplane and b is its threshold. The number ξ_i is 0 if the point x_i does not overlap the hyperplane or is $\xi_i > 0$ if it does. The first addend $\frac{1}{2}||w||^2$ in the objective function (1) mirrors the requirement of the maximal margin between classes and the second member $C \sum_i \xi_i$ expresses that the overlap of the training points should be as small as possible. The constant C determines the trade-off between points overlap and maximal margin. This parameter must be prescribed by the user. The conditions (2) and (3) state that the training points from the first class X_1 and the second class X_2 must lie in different subspaces determined by the found hyperplane.

The optimization problem described above can be transformed into a dual optimization problem [Fle87] where training points appear only in dot products $\langle x_i, x_j \rangle$. For this formulation, we have to introduce class label $y_i, i = 1, 2, ..., m$, which is $y_i = +1$ for $x_i \in X_1$ and $y_i = -1$ for $x_i \in X_2$. Then the dual optimization problem can be written as

$$\begin{aligned}
&\lambda = \underset{\lambda}{\operatorname{argmax}} \sum_{i=1}^{m} \lambda_i - \tfrac{1}{2} \sum_{i,j=1}^{m} \lambda_i \lambda_j y_i y_j \langle x_i, x_j \rangle \,, \\
&0 \le \lambda_i \le C \,, \quad i = 1, 2, \ldots, m \,, \\
&\sum_{i=1}^{m} \lambda_i y_i = 0 \,.
\end{aligned}$$

The found values λ determine the vector w as

$$w = \sum_{i=1}^{m} \lambda_i y_i x_i.$$

Let us note, that we can sum up only over such vectors x_i those corresponding λ_i is non-zero. These vectors are called Support Vectors. The threshold b can be computed using Karush-Kuhn-Tucker conditions, playing a main role in analysis of this problem. Specifically, we find the subset $I = \{i = 1, 2, \ldots, m \colon 0 < \lambda_i < C\}$ and compute threshold as

$$b = \frac{1}{|I|} \sum_{i \in I} y_i \cdot (1 - \langle w, x_i \rangle) \,,$$

where the average value of the threshold is used since it is numerically more feasible. Finally, a given classified point x is assigned to the class according to the sign of function

$$f(x) = \sum_{i=1}^{m} \lambda_i \, y_i \, \langle x_i, x \rangle + b \,. \tag{4}$$

Again, the above summation can be done only over Support Vectors.

The dual expression makes the learning problem independent of the dimension. Moreover, when these dot products are replaced by the non-linear kernel function $k(x_i, x_j)$ then the SVM can be used to find non-linear classifiers (see Section 3.2).

SVM implemented in the toolbox uses the Matlab Optimization toolbox to numerical optimization. Let us note that implemented basic version of SVM is applicable for small to moderate values of input data size. For large problems (tens of thousands of patterns), standard algorithms for quadratic optimization (like those in Matlab) cannot be used directly. Effective methods based on decomposition of the initial problems to smaller computationally feasible ones were proposed in [Vap95], [Pla98]. We plan to include these methods into next version of the STPR toolbox.

The further restriction of the basic SVM comes from its formulation for two classes only. We implemented a simple extension which decomposes the initial c-class problem into c, $c > 2$, dichotomies solvable by the basic algorithm. Specifically, for the i-th class we learn classifier which assigns points $x \in X_i$ into one class and the other points $x \in \cup_{i \neq j, j=1,\ldots,c} X_j$ into the second class. We obtain one discriminant function $f_i(x)$ for each class, which is defined similarly as in Equation (4). Given points are assigned to the class c_x with the maximal value of discriminant function, i.e. $c_x = \underset{i=1,\ldots,c}{\operatorname{argmax}} f_i(x)$.

4 Experiments

We will demonstrate the use of the newly implemented algorithms on the Ripley's dataset [Rip94]. This dataset constitutes a well-known synthetic problem of small size and in two-dimensional space. The set consists of 250 training and 1000 testing patterns belonging into two classes. Both the testing set and the training set are heavily overlapped. In this experiment, the SVM are compared to the non-linear version of the simple linear algorithm which follows the similar recently published approaches [RLdT01, FCC98]

We learnt three types of discriminant functions - linear, quadratic and a discriminant corresponding to the Radial Basis Function kernel (denoted RBF). As a learning algorithm we always used SVM with corresponding kernel compared with the algorithm solving Anderson's task. The Anderson's task produces only linear decision boundary but it can be easily extended to non-linear boundaries using a non-linear mapping. We used both the explicit mapping for quadratic discriminant function and the Kernel PCA with RBF kernel.

Let us give a brief description of the Anderson's task. The aim of the original Anderson's task is to separate two classes, each is described by one Gaussian[1)] and to minimize the probability of classification error. No weights of these distributions are required, which is especially useful when a priori probabilities of classes are not available. Moreover, it has been proven [SH99] that use of this classifier is admissible, even if distributions are not Gaussian and there is no additional information about their shape. Implementation of the algorithm solving the Anderson's task is considerably simpler compared with the SVM. Roughly speaking, the algorithm tries to find such a hyperplane that maximizes Mahalanobis distance from mean value of both class to the hyperplane. For detailed description we refer to [SH99].

To use this classifier, we have to know parameters of Gaussians, i.e. mean value μ and covariance matrix σ, of each class. We estimated these parameters by sample mean and sample covariance matrix. For linear decision boundary, we estimated μ and σ directly in the input space. In the case, the quadratic boundary and the RBF boundary were learned, we first non-linearly mapped the input space to a new space with higher dimension. In the case of quadratic boundary, the function for explicit mapping was used. In the case of RBF boundary, its corresponding feature space has infinite dimension, the Kernel PCA with the RBF kernel was used. The dimension of the space was reduced to $d = 10$.

The obtained results are summarized in Table 1. Discriminant boundaries found classifiers are shown in Figure 1. Notice that boundaries found by SVM and non-linear algorithm for Anderson's task are similar even though both approaches are very different.

[1)] Let us note that the toolbox contains not only algorithm for this task but also others which solve so called Generalized Anderson's task where the classes are supposed to be described by a finite mixture of Gaussians.

Table 1: SVM compared to non-linearized Anderson's task.

	Class. error (testing data)	Class. error (training data)	Floating point operations
SVM (linear, C=100)	10 %	15 %	7309.8×10^6
Anderson (linear)	12 %	16 %	0.0056×10^6
SVM (quadratic, C=100)	10 %	16 %	7556.9×10^6
Anderson + quadratic mapping	9 %	15 %	0.0256×10^6
SVM (RBF, C=100)	10 %	12 %	8746.2×10^6
Anderson + Kernel PCA (RBF, dim = 10)	10 %	12 %	275.7691×10^6

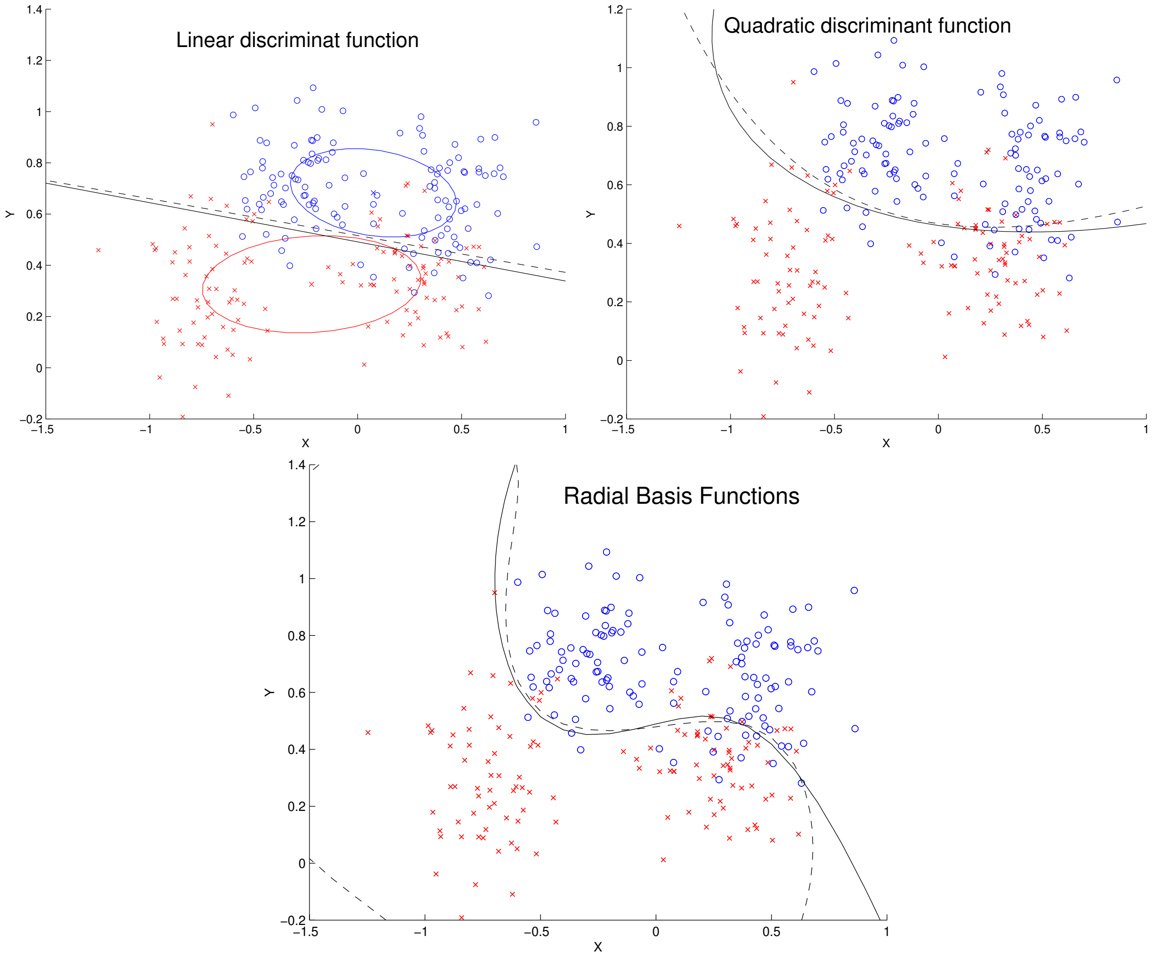

Figure 1: Decision boundaries of classifiers learnt on Riplay's data set by SVM (solid line) and algorithm for Anderson's task (dashed line). In the picture with linear discrimiant function, shapes of estimated covariance matrices are also displayed.

Results given by the non-linear algorithm for Anderson's task are well competitive with SVM in achieved errors rates. Moreover, they are considerably faster if judged by number of used floating operations. It shows that the simpler algorithm can compete with SVM very well.

5 Conclusions

The Statistical Pattern Recognition Toolbox built on top of Matlab has been briefly introduced. This paper concentrated to extensions of our toolbox unpublished so far, namely: (a) EM algorithm for conditionally independent statistical model, (b) Kerner Principal Component Analysis, (c) Support Vector Machines. Classifier learnt by (c) and linear algorithm solving Anderson's task non-linearized by (b) were experimentally tested. What might be surprising to the reader is that the originally linear algorithm for Anderson's task that is transformed to a nonlinear algorithm by the Kernel PCA yields comparable results to currently fashionable SVM even it is simpler.

We intend to keep developing the STPR toolbox in our future work.

We acknowledge the help of Prof. M.I. Schlesinger, who brought our attention to problems reported in this contribution and keeps showing us how to approach them.

References

[FCC98] T.T. Friess, N. Cristianini, and C. Campbell. The kernel adatron algorithm: A fast and simple learning procedure for support vector machines. In *Proc. 15th International Conference on Machine Learning*. Morgan Kaufman Publishers, 1998.

[FHS00] V. Franc, V. Hlaváč, and M. I. Schlesinger. Linear and quadratic classification toolbox for Matlab. In Tomaš Svoboda, editor, *Proceedings of the Czech Pattern Recognition Workshop*, pages 89–99, Peršlák, Czech Republic, February 2000. Czech Society for Pattern Recognition, Prague.

[Fle87] R. Fletcher. *Practical Methods of Optimization*. John Wiley and Sons, Inc., 2nd edition edition, 1987.

[Pla98] J.C. Platt. Fast training of support vectors machines using sequential minimal optimization, 1998.

[Rip94] B.D. Riplay. Neural networks and related methods for classification (withdiscusion). *J. Royal Statistical Soc. Series B*, 56:409–456, 1994.

[RLdT01] A. Ruiz and P.E. López-de Teruel. Nonlinear kernel-based statistical pattern analysis. *IEEE Trans. on Neural Networks*, 12(1):16–33, January 2001.

[Sch97] M. I. Schlesinger. Identifikacia statisticeskich parametrov v odnoj modeli uslovnoj nezavisimosti, in Russian (Statistical parameters identification of one conditional independence model). Technical Report 1895, UTIA AV CR, Prague, Czech Republic, January 1997.

[SH99] M. I. Schlesinger and V. Hlaváč. *Deset přednášek z teorie statistického a strukturního rozpoznávání, in Czech (Ten lectures on statistical and structural pattern recognition).* Czech Technical University Publishing House, Praha, Czech Republic, 1999. English version is supposed to be published by Kluwer Academic Publishers in 2001.

[SSM98] B. Schölkopf, A. Smola, and K.R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5),1299-1319, 1998.

[Vap95] V. N. Vapnik. *The nature of statistical learning theory.* Springer-Verlag, New York, 1995.