ARTICLE IN PRESS

Available at www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Neurocomputing ▮▮▮ (▮▮▮▮) ▮▮▮–▮▮▮

NEUROCOMPUTING

www.elsevier.com/locate/neucom

Feature vector selection and projection using kernels

G. Baudat*, F. Anouar

MEI, Mars Electronics International, 1301 Wilson Drive, West Chester, PA 19380, USA

Received 28 February 2002; accepted 6 March 2003

Abstract

This paper provides new insight into kernel methods by using data selection. The kernel trick is used to select from the data a relevant subset forming a basis in a feature space F. Thus the selected vectors define a subspace in F. Then, the data is projected onto this subspace where classical algorithms are applied. We show that kernel methods like generalized discriminant analysis (Neural Comput. 12 (2000) 2385) or kernel principal component analysis (Neural Comput. 10 (1998) 1299) can be expressed more easily. Moreover, it will turn out that the size of the basis is related to the complexity of the model. Therefore, the data selection leads to a complexity control and thus to a better generalization. The approach covers a wide range of algorithms. We investigate the function approximation on real classification problems and on a regression problem.

Keywords: Kernel methods; Feature space; Data selection; Principal component analysis; Discriminant analysis

1. Introduction

Although the kernel theory is old [12], it was first used in pattern recognition domain by Aizerman and recently in support vector machines (SVM) [7,15,23]. SVM have been applied to a wide class of problems such as classification and regression. The idea behind kernel methods is to map the data in a feature space F that usually has a huge dimension. The transformed data in F has often a shape that can be handled using simple methods such as linear algorithms. It has been shown [24] that the capacity of generalization depends on the geometrical characteristics of the training data and not

* Corresponding author. Tel.: +1-610-430-27-51; fax: +1-610-430-27-95.
E-mail addresses: gaston.baudat@eu.effem.com (G. Baudat), fatiha.anouar@effem.com (F. Anouar).

0925-2312/03/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S0925-2312(03)00429-6

on their dimensionality. Therefore, if these characteristics are well chosen, the expected generalization error could be small even if the feature space has a huge dimensionality.

SVM and kernel-based methods have inspired recent work on the extension of the classical linear algorithms [4,19] and on improving computational efficiency [8]. The extension of classical algorithms to kernel methods is not usually straightforward and leads to complex and memory-consuming solutions like generalized discriminant analysis (GDA) and kernel principal component analysis (KPCA). These algorithms need to use matrices, as large as the size of all the training samples even at the utilization phase. The idea of reducing the size of such matrices has been explored in [21] and is related to kernel feature analysis (KFA) [20]. KFA provides sparse feature extractors by using a modified setting of KPCA. Unlike KPCA which uses a l_2 norm, the KFA deals with a l_1 penalty on the expansion coefficients, leading to simple vertex solutions. But this simplification has a cost, inasmuch as the admissible vectors may have a high l_2 norm and generate a bias in the variance criterion. None of the existing feature extraction algorithms deal with the centering step. Actually, this step in KPCA is memory consuming because it uses all the data. Setting aside the centering can yield less relevant feature extractors.

Our approach initially aimed to answer the question: knowing that the transformed data in a huge dimensionality space F span on a subspace F_S, what is the useful dimension of this subspace? We defined a global criterion to characterize F_S. Once the relevant vectors that we call feature vectors (FVs) are selected from the training set, any data can be projected onto these FVs. Therefore, algorithms can be carried out with this projected data and no longer require the kernel function, except through the projection. The idea is to get a non-linear version of an algorithm while avoiding formulating it in terms of dot product. The feature vector selection (FVS) is the first step in this process; the second step is the projection onto the FVs and then application of the classical algorithms. In particular, a good approximation of KPCA with l_2 penalty and dealing with the centering can be obtained with few FVs.

This paper is motivated by the following objectives: an easy implementation of the kernel methods which reuse the classical algorithms taking advantage of the kernel trick; and computation speed up during the utilization phase by performing any algorithm into a subset of the feature space.

In the following we present the FVS. Then, we investigate the combination of the data selection with various algorithms. We present an extension of the linear regression to get a non-linear function approximation. The latter algorithm is applied to classification and regression problems. Many other algorithms can be extended easily. We also review the GDA and the KPCA algorithms.

2. Feature vector selection

Consider a mapping function ϕ operating from an input space X in a feature Hilbert space F:

$$\begin{aligned} \phi : X &\to F, \\ x &\to \phi(x). \end{aligned} \tag{1}$$

Suppose M is the number of samples; then, we define the kernel matrix K of dot products as

$$K = (k_{ij})_{1 \leqslant i \leqslant M, 1 \leqslant j \leqslant M} \quad \text{where } k_{ij} = \phi^t(x_i)\phi(x_j). \tag{2}$$

The transformed data lie in a subspace F_S of F with a dimension up to M. In practice, the dimension of this subspace is significantly lower than M and equal to the numerical rank of the kernel matrix K.

We developed a preprocessing method that selects relevant data (FVs), forming an approximation of a basis of F_S and capturing the geometrical structure of the whole data in F. It should be noted that this algorithm is independent of any a priori target. The data selection in F exploit the kernel trick that uses the dot products k_{ij} without knowing explicitly the mapping function ϕ [1,6]. We look for vectors that are sufficient to express all the remaining data as a linear combination of those selected vectors in F.

In the following development, the data are assumed centered using the estimation of the center as given in Appendix A. Let L be the number of FVs. For notation simplification: for each x_i the mapping is noted $\phi(x_i) = \phi_i$ for $1 \leqslant i \leqslant M$. We note the selected vectors by, x_{s_j} and $\phi(x_{s_j}) = \phi_{s_j}$ for $1 \leqslant j \leqslant L$, where $L \leqslant M$. For a given set of selected vectors $X_S = \{x_{s_1}, \ldots, x_{s_L}\}$ in F where $S = \{s_1, \ldots, s_L\}$, we can estimate the mapping of any vector x_i as a linear combination of X_S, which is formulated as dot products:

$$\hat{\phi}_i = \Phi_S.a_i, \tag{3}$$

where $\Phi_S = (\phi_{s_1}, \ldots, \phi_{s_L})$ is the matrix of the selected vectors in F and $a_i = (a_{i1}, \ldots, a_{iL})^t$ is the coefficient vector that weighted this matrix.

Given x_i, the goal is to find the coefficients a_i such that the estimated mapping $\hat{\phi}_i$ is as close as possible to the real mapping ϕ_i for a given set S. For this purpose we minimize in F, the normalized Euclidean distance given by the following ratio:

$$\delta_i = \frac{\|\phi_i - \hat{\phi}_i\|^2}{\|\phi_i\|^2}. \tag{4}$$

Note that the numerator of δ_i is the residual error and the ratio δ_i is equal to $\sin^2(\theta_i)$ where θ_i is the angle between the original vector ϕ_i and the reconstructed vector $\hat{\phi}_i$. In other words δ_i is a measure of collinearity between the two vectors, regardless their lengths. This turns out to be dealing with angles instead of residual errors, looking for a maximum of collinearity reconstruction. We show in Appendix B that the minimum of (4) for a given S can be expressed with dot products only and leads to (5):

$$\min_{a_i} \delta_i = 1 - \frac{\vec{K}_{Si}^t K_{SS}^{-1} \vec{K}_{Si}}{k_{ii}}, \tag{5}$$

where $K_{SS} = (k_{s_p s_q})_{1 \leqslant p \leqslant L, 1 \leqslant q \leqslant L}$ is a square matrix of dot products of FVs and $\vec{K}_{Si} = (k_{s_p i})_{1 \leqslant p \leqslant L}$ is the vector of dot products between x_i and the FVs.

The goal is to find the set S that minimizes (5) in average over all samples x_i:

$$\min_S \left(\sum_{x_i \in X} \left(1 - \frac{\vec{K}^t_{Si} K^{-1}_{SS} \vec{K}_{Si}}{k_{ii}} \right) \right). \tag{6}$$

We define the global and the local fitness J_S and J_{Si} for a given set S by

$$J_S = \frac{1}{M} \sum_{x_i \in X} J_{Si}, \tag{7a}$$

where

$$J_{Si} = \frac{\vec{K}^t_{Si} K^{-1}_{SS} \vec{K}_{Si}}{k_{ii}} = \frac{\|\hat{\phi}_i\|^2}{\|\phi_i\|^2}. \tag{7b}$$

(6) is equivalent to:

$$\max_S (J_S). \tag{8}$$

Note that the maximum of (7a) is one and for $x_i \in S$, (5) is zero. Therefore when L increases, we only need to explore $(M - L)$ remaining vectors to evaluate (8).

The selection is an iterative process, which is a sequential forward selection: at the first iteration $(L = 1)$ we look for the sample that gives the maximum global fitness J_S (7a) and at the same time the local fitness (7b) is used to select the next best candidate, as the one having the lowest local fitness J_{Si} for the current basis (maximum angle θ_i). Except for iteration 1, the algorithm uses the lowest local fitness to elect the new FV while evaluating the global fitness. J_S is monotonic since the new basis will reconstruct all the samples at least as well as the previous basis did. On the other hand, the new FV is rebuilt without error leading to an even larger global fitness. This procedure has a tendency to provide FVs close to orthogonality, ending to well-conditioned numerical solution. And finally retrieving the next FV from the current iteration reduces the computing complexity as well. Next iterations compute the new global fitness and look for the next best candidate using the local fitness again. To avoid computing the inverse of the matrix at each iteration, we used the partition inversion method [16]. The procedure ends when $\mathbf{K}_{SS}$ is no longer invertible numerically, which means that X_S is a good approximation of a basis for the data in F. We can also stop as soon as the global fitness J_S reaches a given value bounded by one, or when a predefined number of FVs have been selected. In our experiments, the stopping parameters are chosen by cross validation. We note interestingly that these parameters allow us to control the sparsity and consequently to control the complexity of the model.

2.1. Principle of the FVS Algorithm

```
Arguments:
  Training sample $X = \{x_1, \ldots, x_M\}$
  Stopping criterion: number of feature vectors nbFV or maxFitness
  or basisFound
• Return: $S$ subset of selected feature vectors
• Initialization: fitness $= 0$, $L = 0$, basisFound $=$ False, $S = \{\}$
• Compute and center the kernel matrix $K = (k_{ij})_{1 \leqslant i \leqslant M, 1 \leqslant j \leqslant M}$
  (see Appendix A)
• First iteration:
$L = 1$;
for $i = 1$ to $M$ do
  $S = \{x_i\}$; $J_S = 0$;
  Compute all local fitness $J_{Sj}$ for $1 < j < M$;
  Store the index $\hat{j}$ such that $\hat{j} = \arg\min_j J_{Sj}$;
  Compute $J_S$;
  if ($J_S \geqslant$ fitness)
    fitness $= J_S$;
    firstFV $=$ i;   %keep the vector $x_i$ giving the maximum $J_S$
    nextFV $= \hat{J}$;   %keep the vector $x_{\hat{j}}$ giving the minimum $J_{Sj}$
  end if
end for
$s_1 =$ firstFV; $S = \{s_1\}$;
Catch the necessary data $\vec{H}_{S(L),i}$, $\psi_{S(L),i}$ for next steps (see Appendix C)
• Iteration $L > 1$
while ($L <$ nbFV) and (fitness $<$ maxFitness) and (not basisFound)
  $L = L + 1$; $s_L =$ nextFV; $S = S \cup \{s_L\}$;
  Compute $c_L = \psi_{S(L-1)s_L} - k_{s_L s_L}$;  %for recursive fitness (see Appendix C)
  if $c_L \neq 0$                    %$K_{SS}$ matrix still invertible
    compute and catch $\vec{H}_{S(L),i}$, $\psi_{S(L),i}$ for $1 < i < M$;
    Compute all local fitness $J_{Sj}$ for $1 < j < M$;
    Store the index $\hat{j}$ such that $\hat{j} = \arg\min_j J_{Sj}$;
    Compute $J_S$; fitness $= J_S$; nextFV $= \hat{j}$;
  else basisFound $=$ true;
  end if
end while
if not basisFound                  %can be added to the FV set
  $L = L + 1$; $s_L =$ nextFV; $S = S \cup \{s_L\}$;
end if
```

The computation of the first iteration requires $O(M^2)$ operations like the computation of the kernel matrix. The following iterations ($L > 1$) use an iterative computation based on the partition inversion method [16] (see Appendix C). The cost per iteration is O(ML). Therefore, the computational cost overall L iterations ($L > 1$) is O(ML^2). Since the center is approximated with one nearest sample, the complexity stays in O(ML^2). For the first iteration, one may use a probabilistic speed-up by performing the search only on a subset of a size N sampled among M vectors [18]. The computational cost in this case will be O(NM). Note that K terms can be computed when needed if we want to avoid storing it. The selected set S may not be unique. But our concern is to find one good set that preserves the intrinsic structure of the data in F. We show, in practice, that for some kernels like the polynomial, the optimal value of J_S is reached with only few selected vectors.

3. Projection onto the feature vectors

Once the feature vectors are selected, they define a subspace F_S in F. We transform all of the samples in this subspace. For a given sample x_i, we apply the dot product projection:

$$z_i = \Phi_S^t \phi_i. \tag{9}$$

Actually, this transformation is an empirical kernel map [17], $\phi_{\text{emp1}} : x_i \to z_i = (k(x_i, x_{s_1}), \ldots, k(x_i, x_{s_L}))^t$. Note that z_i is obtained through a transformation which is not using the canonical dot product. Other transformations can be considered, in particular, an orthonormal projection (10), which requires more computation and is not needed for many algorithms applied to the transformed data.

$$z_i = (\Phi_S^t \Phi_S)^{-1/2} \Phi_S^t \phi_i. \tag{10}$$

That latter projection corresponds to an orthogonal empirical kernel map such as

$$\phi_{\text{emp2}} : x_i \to z_i = K_{SS}^{-1/2}(k(x_i, x_{s_1}), \ldots, k(x_i, x_{s_L}))^t$$

The FVS algorithm can be combined with many methods since the selection is a data preprocessing. In fact, the FVS combined with an algorithm that is invariant to transformation (9) leads to a good approximation of its kernalized version. Indeed, if we select all the samples as FVs we eventually converge to the kernalized version. The procedure needs an adaptation for algorithms like KPCA. In order to get the same results from either the KPCA or the FVS combined with the classical PCA (FVS-PCA), projection (10) that incorporates a rescaling operation preserving the dot product in F should be used [17]. A closer look to (10) shows that when $L \to M$, i.e. $K_{SS}^{-1/2} \to K^{-1/2}$ then FVS-PCA is equivalent to KPCA. In the section devoted to the applications, we will show how the KPCA is well approximated by FVS-PCA.

4. Feature vector selection and linear regression (FVS-LR)

The data selection can be viewed as the definition of the hidden layer of a multi-layer neural network [3]. The number of hidden neurons on which the data are projected

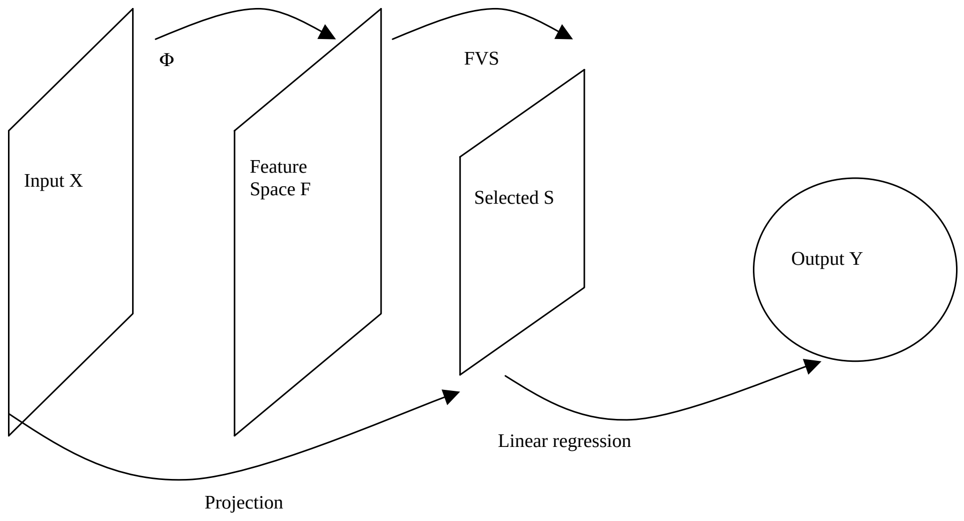

Fig. 1. Architecture of FVS-LR.

corresponds to the number of FVs selected in F. However, the FVS has the advantage of providing a number of hidden neurons for a given kernel. The kernel trick makes it possible to proceed and select data in the feature space F. For a neural network, the feature space remains hidden and is never explored. This main attribute gives new insight into neural networks.

The generalization of different kinds of neural networks is obtained by choosing the appropriate kernel function: Gaussian kernel gives an RBF neural network, while a sigmoid function, which fulfills the Mercer conditions [23], leads to a multi-layer perceptron. Fig. 1 shows the architecture of a neural network using the FVS and linear regression that we call FVS-LR.

Given a set of data (z_i, y_i) where z_i is projection (9) onto F of the sample x_i, the goal of the FVS-LR algorithm is to estimate the vector $\hat{y}_i$, output of the function. It minimizes the mean square error MSE of the learning set T and results in an optimal solution (11) using the Moore–Penrose pseudo-inverse [2]

$$\hat{y}_i^t = z_i^t A + \beta^t, \tag{11}$$

where $A = (Z_T^t Z_T)^{-1} Z_T^t Y_T$ and $Z_T = (z_j^t)_{j \in T}$, $Y_T = (y_i^t)_{j \in T}$.

The vector β can be included in the estimation of A by adding a constant component in each vector z_i.

It has been shown that (11) leads to the posterior probability estimation and gives the best approximation of the Bayes decision function [9].

A classification task into N classes is a particular regression problem for which the output $y_i = (y_{ij})_{1 \leqslant j \leqslant N}$ is a binary vector such that

$$y_{ij} = \begin{cases} 1 & \text{if } z_i \in j\text{th cluster}, \\ 0 & \text{otherwise.} \end{cases}$$

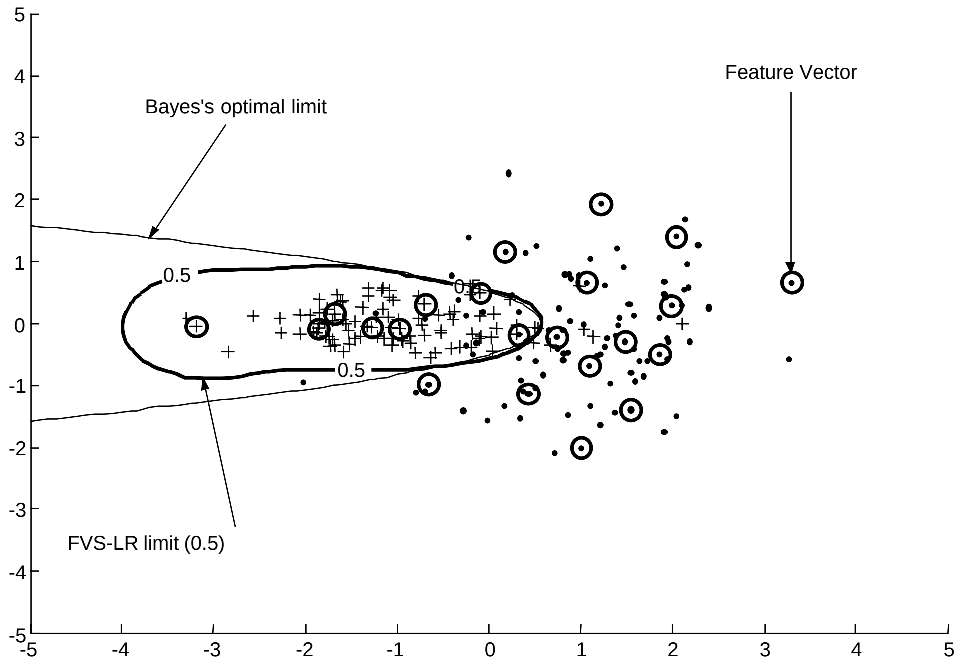

Fig. 2. Optimal boundary and FVS-LR boundary; the feature vectors are represented by circles.

The classification optimization performed in F is simple and leads to the same formula as presented above for regression. A sample is assigned to the class c such that $c = \arg\max_j \hat{y}_{ij}$.

5. Experiments

5.1. Comparison to optimal classification

We simulate two clusters with normal distributions

$$N((-1,0),I) \quad \text{and} \quad N\left((1,0),\begin{pmatrix}1 & 0\\ 0 & 0.1\end{pmatrix}\right)$$

displayed in Fig. 2. I is the identity matrix. The learning set contains 100 examples from each class. Since we know the theoretical distribution, we computed the optimal separation given by the Bayesian decision function [10]. Therefore, we can compare the theoretical results and the results obtained with the FVS-LR in F. Fig. 2 shows the results of the Bayesian classification and kernel function approximation using a Gaussian kernel function $k(x_i,x_j) = \exp(-\|x_i - x_j\|^2/\sigma^2)$ with $\sigma = 1$.

Table 1
Test performance comparison

	Bayes	SVM	FVS-LR
Testing performance (%)	88.8	88.2	88.6
Number of vectors	—	54 SV	23 FV

We evaluated the classification performances shown in Table 1 using a testing set of 100,000 samples. FVS-LR classification gives almost the same results as the SVM ($\sigma = 1$, OSU SVM Matlab toolbox). Nevertheless, our algorithm reaches the best performance with only 23 selected vectors, while the SVM needs 54 support vectors.

5.2. Function regression

We consider the approximation of the sinc function $f(x) = (\sin \pi x)/\pi x$ corrupted by a normal noise. Results on this function using SVM are given in [25]. We used the same kernel function, which is linear spline:

$$K(x_i, x_j) = 1 + x_i, x_j + x_i x_j \min(x_i, x_j) - \frac{(x_i + x_j)}{2}(\min(x_i, x_j))^2 + \frac{(\min(x_i, x_j))^3}{3}.$$

The added normal noise has a standard deviation of 0.5. Fig. 3 shows the approximated function and the real function.

The approximation given in [25] leads to 27 support vectors while FVS-LR requires only six feature vectors. Another interesting result related to the approximation of 'sinc' is obtained by Relevance Vector Machine [22]. For a standard deviation of the noise equal to 0.2, the approximation also uses six relevance vectors. Although the data are not exactly the same, the number of feature vectors is close to the number of relevance vectors and significantly smaller than the number of support vectors.

5.3. Feature vector selection and GDA

This section shows that GDA [5] can be obtained more easily by applying linear discriminant analysis (LDA) [10] after running the FVS and data projection onto the FV subspace. For writing simplification, we denote the whole procedure as FVS-LDA.

Example 1: *feature space versus input space.* This simple example illustrates visually the relation between the input space X and the feature space F when using a polynomial kernel of the second order. If $X = R^2$, The space F for this kernel has a dimension of three and can be displayed. We can express the mapping function ϕ in this particular case. The homogenous polynomial kernel in R^2 is defined by $k_{ij} = (x_i^t x_j)^2$, and can be rewritten as:

$$(x_i^t, x_j)^2 = \phi^t(x_i)\phi(x_j) \quad \text{where } \phi^t(x_i) = \left(x_{i1}^2, \sqrt{2}x_{i1}x_{i2}, x_{i2}^2\right).$$

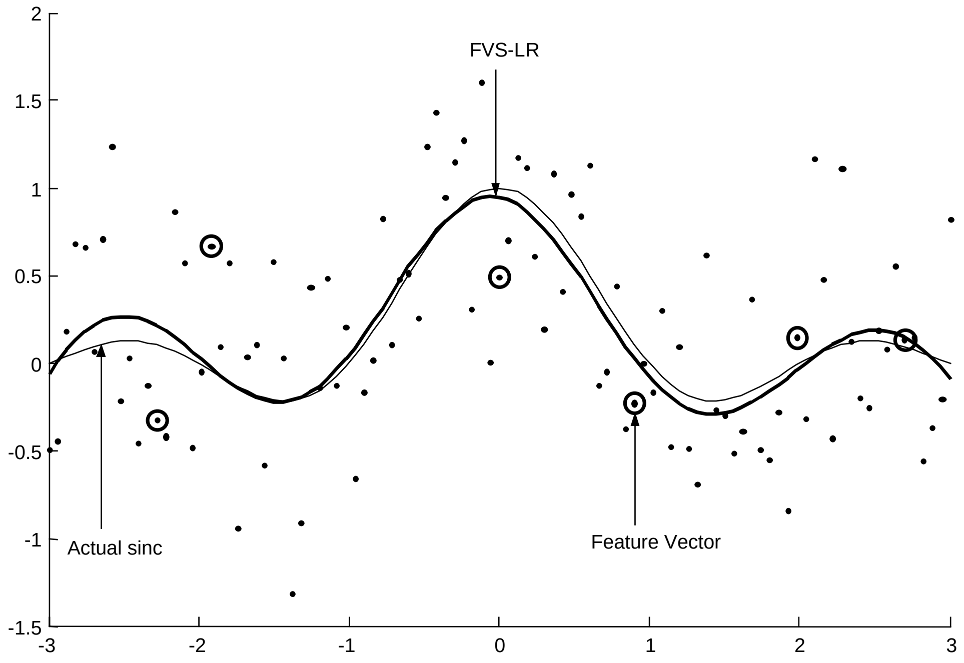

Fig. 3. Sinc function corrupted by a normal noise and the approximated function obtained by FVS-LR with the feature vectors shown circled.

Consider two clusters along two circles (2D) of radius 1 and 0.5 plotted with crosses and small circles in Fig. 4a. The images of these two circles into F are displayed in Fig. 4b. The line in F is the discriminant axis obtained with FVS-LDA. The continuous circle in the input space (Fig. 4a) is the separation boundary assuming a threshold on the discriminant axis half way between the two circles into F of Fig. 4b.

This example shows that the discriminant axis requires only three feature vectors to be expressed in F. We obtained the same results using the GDA which requires more computation, because it expresses the axis with a dense expansion using all the samples.

Example 2: *non-separable data.* We simulate two clusters along two overlapping circles with added noise. Each cluster contains 200 samples: 100 samples for learning and 100 samples for testing. Note that the two clusters cannot be separated into the input space using the LDA. We applied the FVS-LDA using Gaussian kernel with $\sigma = 0.5$. Fig. 5a shows the clusters and the boundary function corresponding to FVS-LDA algorithm. We use 100 samples for each cluster for the learning step, but only 35 samples have been selected for the projection. When running the GDA with all data, the optimal inertia is 0.91. On the other hand, with only 35 selected feature vectors the eigenvalue of the FVS-LDA is 0.91 too. Fig. 5b presents the evolution of algorithm performance expressed by the correct classification rate on the testing set versus the number of selected feature vectors. The classification has been performed using the

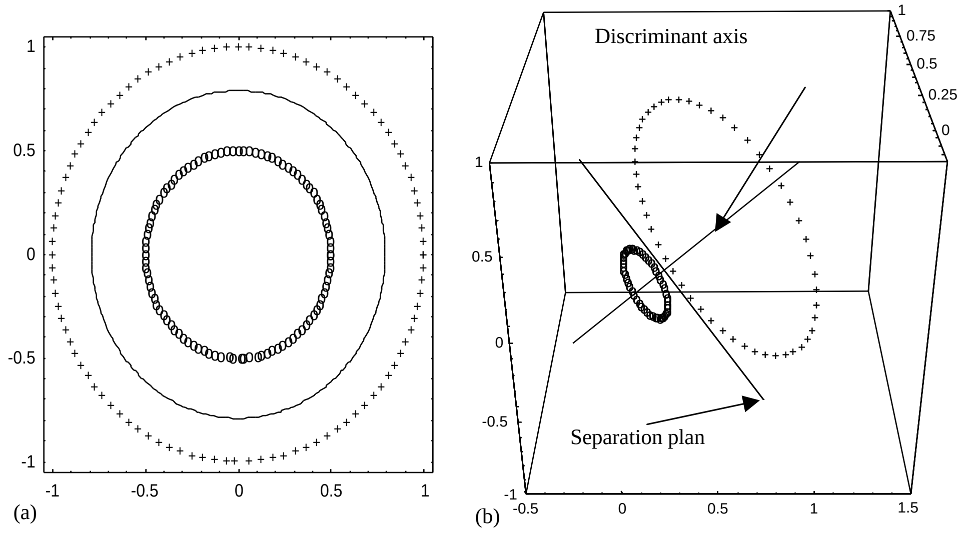

Fig. 4. (a) Data along two circles, the continuous circle is the separation boundary corresponding to the separation plan into F; (b) the image of the clusters into F and the FVS-LDA discriminant axis.

projection of the data on the discriminant axis and the Bayes's rule scheme. We note that the rate increases until it reaches its maximum. Therefore, the FVS algorithm can find sufficient samples that give maximum performance.

Note that FVS cannot go further when the maximum of the selected vectors is reached because the matrix K_{SS} is no longer invertible. The GDA classification rate is the same as the FVS-LDA with 35 FV. It should be noted that, the GDA, which is formulated in terms of dot products, solves an eigenvalue system that faces the problem of singular matrix. In a certain point, the FVS algorithm deals with these singularities and makes the solution easier.

5.4. Benchmarks

To validate FVS approach, we applied the FVS-LR to three of a standardized benchmark datasets and compared with known best results (IDA Benchmark repository). The data are given in 100 predefined splits into training and testing samples. On each partition the FVS is applied and the number of FVs, as well as the kernel parameter, are selected by cross validation much in spirit of [13]. Then a linear regression is trained after projection (9) onto the FVs. The results of FVS-LR as well as those of other methods are given in Table 2. The performances for SVM, kernel Fisher analysis (KFD) and radial basis function (RBF) are reported from [11,14]. Using a Gaussian kernel, we applied the FVS-LR and the KPCA-LR (linear regression on the principal components of KPCA). For the sake of comparison, the number of components used for KPCA-LR is the same than the number of FVs.

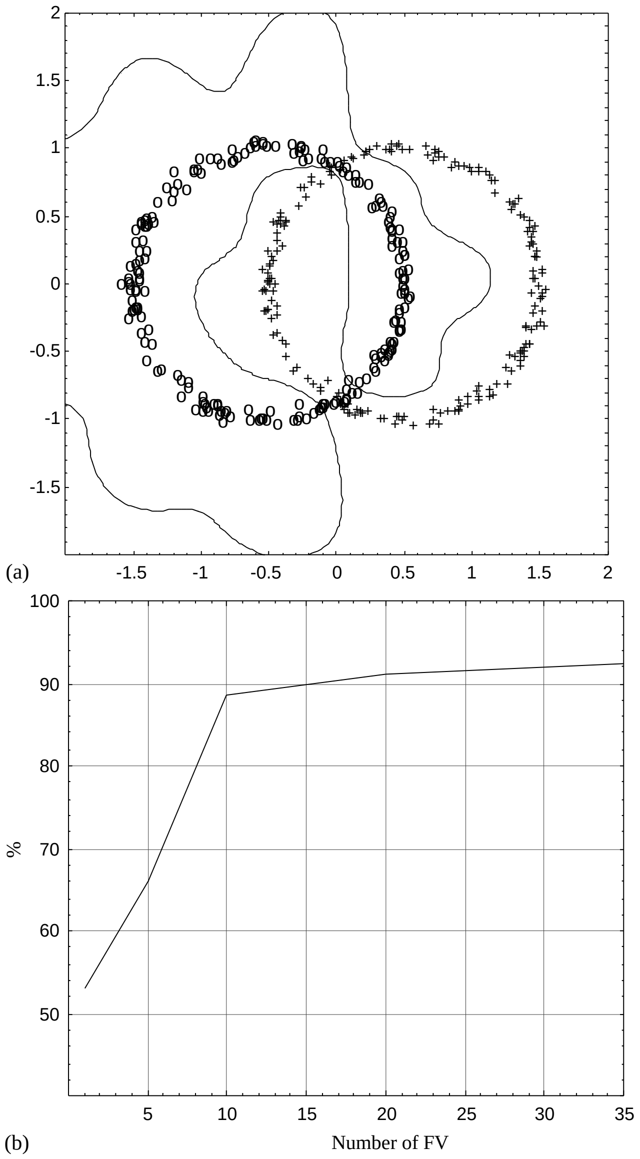

Fig. 5. (a) Boundary function using FVS-LDA with a Gaussian kernel; (b) classification performance for test set versus the number of selected vectors.

We note that FVS-LR gives better results than SVM and other methods for the three datasets. But applying a linear regression to the projected data on the principal components of KPCA gives the same results than FVS-LR. However, recall that unlike the KPCA, which needs all samples, FVS-LR reduces dramatically the number of

Table 2
Performances of: SVM, kernel Fisher discriminant (KFD), RBF, linear regression, KPCA-LR and FVS-LR on three benchmark datasets

Databases	Banana	Heart	Thyroid
SVM test error (%)	11.5±0.7	*16.0±3.3*	4.8±2.2
KFD test error (%)	*10.8±0.5*	16.1±3.4	*4.2±2.1*
RBF test error (%)	10.8±0.6	17.6±3.3	4.5±2.1
Linear regression LR test error (%)	47.0±4.5	**15.9±3.1**	14.8±3.3
KPCA-LR test error (%)	**10.6±0.5**	**15.9±3.1**	**4.1±2.0**
FVS-LR, test error (%)	**10.6±0.5**	**15.9±3.1**	**4.1±2.0**
Number of FVs or KPCA axis	35	13	40
Number of SVs in average	90	81	22
FVS σ	1	120	$7^{1/2}$

Best results in bold face, second best in italics.

needed samples. This number is even smaller than the number of support vectors. In the case of the thyroid dataset, the first 22 FVs give the same test error than SVM but 40 FVs give better results. Surprisingly, the heart disease data set is using very few FVs (13 FVs). It turns out that 13 is the dimension of the input space. Actually, for this dataset a linear regression (LR) in the input space is doing as well as FVS-LR with a Gaussian kernel with the same sigma as given in [14]. The analysis of the fitness *Js* provides important information on the dimensionality of the data and on the meaningful samples. Fig. 6 shows the average fitness *Js* versus the number of FVs for the three datasets on 100 partitions. It shows also the average of the classification error on the testing sets for FVS-LR and KPCA-LR. We note that the selected number of FVs corresponds to the minimum of testing error. For the heart disease data we can see a clear break point around 13 FVs.

6. Discussion

We have presented a selection algorithm in F using the kernel trick. The FVS algorithm captures the structure of the data by approximating a basis of the subspace of the data. All the samples are projected onto the selected subspace of the FVs; therefore, the statistical information is preserved. After the data selection and projection steps, many algorithms can be easily applied. We investigated linear PCA, linear regression and LDA. The selected FVs were sufficient to obtain high performances and reduce significantly the computation time at the utilization phase. We showed on simulated and real data that the results of the FVS-LR compete with the SVM. However, the number of FVs is usually smaller than the number of support vectors required for SVM. We noted the regularization effect of the FVS. Actually, the adjustment of the FVS parameters like the number of FVs or the fitness is a way to control the complexity of the model. We also presented results of the LDA and PCA combined with the FVS, which provides a new method for getting GDA and KPCA. The memory requirement

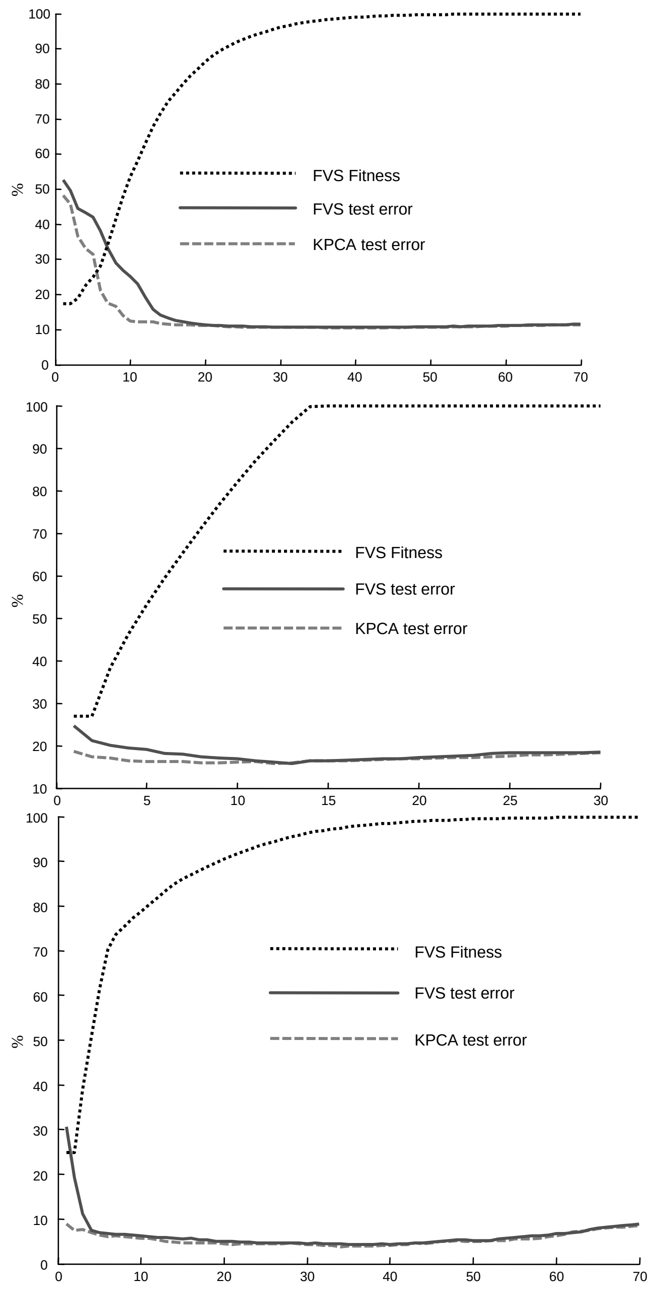

Fig. 6. Test errors and FVS fitness versus the number of FVs or principal axis (KPCA). From top to bottom: banana, heart disease and thyroid datasets.

for matrix storage for the utilization phase in the GDA or KPCA drops from the size of the dataset to the size of the selected dataset. Therefore, the computation time at the utilization phase is reduced and it is now more feasible to implement such algorithms for embedded applications.

Acknowledgements

We would like to thank Scott Barnes, Tim Harris (engineers at MEI, USA) for their comments on the manuscript and Philippe Jard (applied research manager at MEI, USA) for the fruitful discussions and his support of this work. Many thanks also to David Brown from the University of Cape Town (South Africa) for testing and improving the Matlab code. The Matlab code for FVS algorithm and FVS-LR is available for everyone who wants to test the method. Just contact us by e-mail at fatiha.anouar@effem.com or gaston.baudat@eu.effem.com.

Appendix A

In order to keep the advantage of the data selection of reducing the number of kernels at the utilization phase; the centering step is achieved by approximating the center with few or even one sample. Let us define the approximated center $\hat{\phi}_0$ as a combination of the samples by $\hat{\phi}_0 = \sum_{j=1}^{M} b_j \phi_j$, the real center ϕ_0 is obtained when $b_j = 1/M$ for all j. Among other methods, the distance $\|\phi_0 - \hat{\phi}_0\|$ can be minimized using the k-nearest neighbors. The number of neighbors is a trade-off between the computation time and the desired accuracy in approximating the center. In our test we used 1-nearest neighbor and even though we found that it improves the results versus not centering the data at all.

The centering itself is given by using the new kernel matrix:

$$\tilde{K} = (I_M - B_M)^t K (I_M - B_M),$$

where B_M is a matrix with each column is the vector $b = (b_1, \ldots, b_M)^t$ and I_M is the identity matrix. For the 1-nearest neighbor all b_j are zero except the index k of the nearest for which $b_k = 1$.

Appendix B

Substituting (3) in (4):

$$\delta_i = \frac{(\phi_i - \Phi_S.a_i)^t(\phi_i - \Phi_S.a_i)}{\|\phi_i\|^2}. \tag{12}$$

Putting the derivative of δ_i to zero gives the coefficient a_i:

$$\frac{\partial \delta_i}{\partial a_i} = \frac{2(\Phi_S^t \Phi_S) a_i - 2\Phi_S^t \phi_i}{\phi_i^t \phi_i},$$

$$\frac{\partial \delta_i}{\partial a_i} = 0 \;\Rightarrow\; a_i = (\Phi_S^t \Phi_S)^{-1} \Phi_S^t \phi_i, \tag{13}$$

$(\Phi_S^t \Phi_S)^{-1}$ exists if the FVs are linearly independent in F.

Replacing the coefficient values (13) in (12) we get

$$\min_{a_i} \delta_i = 1 - \frac{1}{\|\phi_i\|^2} (\phi_i^t \Phi_S (\Phi_S^t \Phi_S)^{-1} \Phi_S^t \phi_i), \tag{14}$$

(14) is equivalent to (5) given the definition of the kernel matrices.

Appendix C

The numerator of the ith local fitness (7b) at the step L needs the evaluation of the following quadratic form:

$$\psi_{S(L)i} = \vec{K}^t_{S(L)i} K^{-1}_{S(L)S(L)} \vec{K}_{S(L)i}, \tag{15}$$

where $S(L) = \{s_1, s_2, \ldots, s_L\}$ is the basis at the iteration L. Therefore, it includes the L already selected FVs. Without special care, and assuming $K^{-1}_{S(L)S(L)}$ given, (15) costs in order of L^2 operations per local fitness. Therefore, the overall cost per iteration will be at least $\mathrm{O}(ML)^2$. This is often more than we would like to accept. This problem can be overcome using a sequential scheme assuming some data is caught from the previous step $L-1$. Let us define:

$$\vec{H}_{S(L)i} = K^{-1}_{S(L)S(L)} \vec{K}_{S(L)i}. \tag{16}$$

Then:

$$\psi_{S(L)i} = \vec{K}^t_{S(L)i} \vec{H}_{S(L)i}, \tag{17}$$

$\vec{K}_{S(L)i} = (\vec{K}^t_{S(L-1)i}\, k_{s_L i})^t$ is retrieved from the iteration $L-1$, where s_L is the latest selected FV, added at the current iteration L, this means that $S(L) = S(L-1) \cup \{s_L\}$. When using a partitioned matrix inversion of $K_{S(L)S(L)}$ [16] (16) becomes

$$\vec{H}_{S(L)i} = \begin{pmatrix} \vec{H}_{S(L-1)i} + \vec{H}_{S(L-1)s_L}(k_{s_L i} - \vec{K}^t_{S(L-1)s_L} \vec{H}_{S(L-1)i}) c_L^{-1} \\ \vec{K}^t_{S(L-1)s_L} \vec{H}_{S(L-1)i} c_L^{-1} + k_{s_L i} k^{-1}_{s_L s_L} (1 - \psi_{S(L-1)s_L} c_L^{-1}) \end{pmatrix}, \tag{18}$$

where $c_L = \psi_{S(L-1)s_L} - k_{s_L s_L}$.

Assuming $c_L \neq 0$, otherwise $K_{S(L)S(L)}$ is no longer invertible, (18) costs per local fitness $\mathrm{O}(L)$ as well as (17). Overall per iteration ($L > 1$) the complexity becomes $\mathrm{O}(ML)$. Globally, the algorithm costs $\mathrm{O}(ML^2)$ for L iterations. The first iteration costs $\mathrm{O}(M)^2$. Using a probabilistic speed up for the first iteration ($L = 1$) searching among only a subset of $N < M$ samples as proposed in [18], the complexity can be as low as O(MN). Often L is significantly lower than M.

References

[1] M.A. Aizerman, E.M. Braverman, L.I. Rozonoér, Theoretical foundations of the potential function method in pattern recognition learning, Automat. Remote Control 25 (1964) 821–837.

[2] A. Albert, Regression and the Moore–Penrose Pseudo-Inverse, Academic Press, New York, 1972.

[3] F. Anouar, F. Badran, S. Thiria, Probabilistic self organizing map and radial basis function, J. Neurocomput. 20 (1998) 83–96.

[4] G. Baudat, F. Anouar, Generalized discriminant analysis using a kernel approach, Neural Comput. 12 (2000) 2385–2404.

[5] G. Baudat, F. Anouar, Kernel based methods and function approximation, International Joint Conference on Neural Network, IJCNN 2001, Washington, DC, pp. 1244–1249.

[6] B.E. Boser, I.M. Guyon, V.N. Vapnik, A training algorithm for optimal margin classifiers, in: D. Haussler (Ed.), Fifth Annual ACM Workshop on COLT, Pittsburgh, PA, ACM Press, New York, 1992, pp. 144–152.

[7] C.J.C. Burges, A tutorial on support vector machine for pattern recognition, support vector web page, http://svm.first.gmd.de, 1998.

[8] C.J.C. Burges, B. Schölkopf, Improving the accuracy and speed of support vector machines, in: Neural Information Processing Systems, Vol. 9, MIT Press, Cambridge, MA, 1997.

[9] R.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.

[10] K. Fukunaga, Introduction to Statistical Pattern Recognition, 2nd Edition, Academic Press, New York, 1990.

[11] IDA Benchmark repository, http://ida.first.gmd.de/∼raetsch/data/benchmarks.htm.

[12] J. Mercer, Functions of positive and negative type and their connection with the theory of integral equations, Philos. Trans. Roy. Soc. London A 209 (1909) 415–446.

[13] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, K.R. Muller, Fisher discriminant analysis with kernels, in: Y.-H. Hu, J. Larsen, E. Wilson, S. Douglas (Eds.), Neural Networks for Signal Processing IX, IEEE, New York, 1999, pp. 41–48.

[14] K.R. Müller, S. Mika, G. Rätsch, K. Tsuda, B. Schölkopf, An introduction to kernel-based learning algorithms, IEEE Trans. Neural Networks 12 (2) (2001) 181–201.

[15] T. Poggio, On optimal nonlinear associative recall, Biol. Cybernet. 19 (1975) 201–209.

[16] L.L. Scharf, Statistical Signal Processing: Detection, Estimation and Times Series Analysis, Addison-Wesley, Reading, MA, 1991.

[17] B. Schölkopf, S. Mika, C.J.C. Burges, P. Knirsch, K.R. Müller, A.J. Smola, Input space vs. feature space in kernel-based methods, IEEE Trans. Neural Networks 10 (5) (1999) 1000–1017.

[18] A.J. Smola, B. Schölkopf, Sparse greedy matrix approximation for machine learning, International Conference on Machine Learning, ICML 2000, Stanford, CA, 2000.

[19] B. Schölkopf, A.J. Smola, K.R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (1998) 1299–1319.

[20] B. Schölkopf, A.J. Smola, Learning with Kernels, MIT Press, Cambridge, MA, 2002, pp. 443–445.

[21] A.J. Smola, B. Schölkopf, Sparse greedy matrix approximation for machine learning, in: International Conference on Machine Learning, 2000.

[22] M.E. Tipping, The relevance vector machine, in: S.A. Solla, T.K. Leen, K.-R. Müller (Eds.), Advances in Neural Information Processing System, Vol. 12, MIT Press, Cambridge, MA, 2000.

[23] V. Vapnik, The Nature of Statistical Learning Theory, Springer, Berlin, 1995.

[24] V. Vapnik, Three remarks on support vector method of function estimation, in: B. Schölkopf, C.J.C. Burges, A.J. Smola (Eds.), Advances in Kernel Methods, MIT Press, Cambridge, MA, 2000.

[25] V. Vapnik, S.E. Golowich, A. Smola, Support vector method for function approximation, regression estimation, and signal processing, in: Neural Information Processing Systems, Vol. 9, MIT Press, Cambridge, MA, 1997.

Gaston Baudat He received the diploma degree in computer science from the Swiss University of applied sciences (Yverdon-Les-Bains) in 1982 and a Master in 1999. He worked first at Landis & Gyr Geneva for document recognition designing new sensors and algorithms. In 1992, he moved to Mars Electronics International (MEI) as a technology manager where he developed and patented several algorithms for signal processing and pattern recognition. Since 1999, he is working at MEI headquarter in USA on advanced data processing. His fields of interest are classification, data analysis, signal processing, and kernel methods for practical implementations in embedded applications.

Fatiha Anouar She received the Diploma degree in mathematics in 1992 from the university of Paris XI and then a Ph.D. in the field of statistics and neural network from the "Consevatoire National des Arts et Metiers" in Paris 1996. In 1997, she was a Postdoc fellow developing a vision system for seed recognition at the French National Institute in Agronomy. Since 1999, she is working at Mars Electronics International headquarter in USA. Her interests are in the fields of statistics, machine learning, pattern recognition and modeling.