Combination of Single Feature Classifiers for Fast Feature Selection

Hassan Chouaib, Florence Cloppet, and Nicole Vincent

Abstract. Feature selection happens to be an important step in many classification tasks. Its aim is to reduce the number of features and at the same time to try to maintain or even improve the performance of the used classifier. The selection methods described in the literature present some limitations at different levels. For instance, some are too complex to be operated in reasonable time or too dependent on the classifier used for evaluation. Others overlook interactions between features. In this paper, in order to limit these drawbacks, we propose a fast feature selection method. Each feature is closely associated with a single feature classifier. The weak classifiers we considered have several degrees of freedom and are optimized on the training dataset. Within the genetic algorithm, the individuals who are classifier subsets are evaluated by a fitness function based on a combination of single feature classifiers. Several combination operators are compared. The whole method is implemented and extensive trials are performed on four databases built from the MNIST handwritten digits database using four different descriptors. Results show how robust is our approach and how efficient is the method. On average, the number of selected features is about 70% smaller than the initial set while keeping the level of recognition rate.

1 Introduction

In many domains such as computer vision or pattern recognition, solving a problem is based on processing data extracted from a set of real world data acquired by means of sensors or resulting from some data processing. Data are structured as vectors. The quality of processing systems highly depends on the choice of the

Hassan Chouaib · Florence Cloppet · Nicole Vincent
Laboratoire LIPADE, Université Paris Descartes, France
e-mail: {hassan.chouaib,florence.cloppet,
nicole.vincent}@mi.parisdescartes.fr

F. Guillet et al. (eds.), *Advances in Knowledge Discovery and Management,*
Studies in Computational Intelligence 527,
DOI: 10.1007/978-3-319-02999-3_7,

vector contents. However, in many cases the vectors' high dimensionality makes it almost impossible to use them to solve the problem because of the data nature and of the learning set size. The high dimension of the representation space makes any learning set too sparse for using the common methods [Bins and Draper, 2001]. Hence it is usually recommended, and sometimes required, for example in bioinformatic studies or text analysis, to reduce the vector size in order to make data more usable. There is benefit even if the reduction might lead to loss of information. Sometimes, solving complex problems with large descriptors can also be accomplished using a small set of features selected from the initial data set. This can be done if the selected features are relevant enough with respect to the problem being considered [Zhou and Dillion, 1991]. According to the feature nature, feature selection can either improve the quality of the system if the eliminated features are the too noisy ones, or improve computation time when redundant or irrelevant features are present in the feature set.

Reducing vector dimensionality is often considered as a pre-processing step dedicated to noise and redundant information elimination. Among dimensionality reduction methods, feature extraction (the most representative is Principal Component Analysis) and feature selection can be considered. Here, we focus on feature selection. It consists of selecting the most relevant features from an initial set. Among the applications needing feature selection methods we can distinguish between clustering [Bouguila and Ziou, 2012] and classification. In this paper, only the classification problem is considered.

Existing feature selection methods reveal limitations on many levels such as complexity, interaction between the features, dependency on the evaluation classifier, and so on. In order to overcome these limitations, we introduce a new method for feature selection. It is based on selecting the best classifier combination from a set of simple classifiers. Each of these classifiers is built using a single feature and the selection is accomplished using a genetic algorithm. Moreover, the intermediate use of classifiers enable to handle a set of mixed numerical and symbolical features.

The paper is organized as follows. In Section 2, we motivate our choice for feature selection method by showing the limitations of existing methods. In Section 3, we introduce our **F**ast **F**eature **S**election **M**ethod (FFSM). In Section 4, different elements of the method are discussed and in Section 5, an extensive experimental study is carried out. Finally, conclusions are drawn and perspectives are given in Section 6.

2 Feature Selection

Feature selection is generally defined as a search process that finds a *relevant* feature subset from an initial feature set. The relevance of a feature subset always depends on the objectives and criteria of the problem to be solved.

A selection method [Liu and Yu, 2005] generally incorporates several phases (see Figure 1).

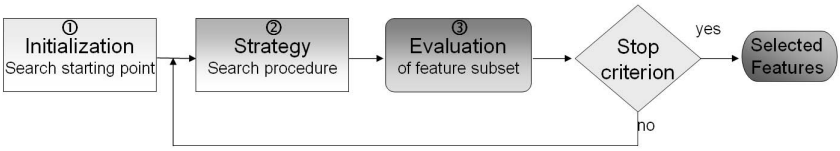

Fig. 1 Overview of a feature selection method

The two first steps initialize a search starting point and apply a search procedure. Once the subsets are generated, an evaluation is computed in the third step. Steps 2 and 3 are repeated until a stop criterion is satisfied. A search procedure consists of generating feature subsets which will be evaluated to select the best subset. In general, search strategies can be classified into three categories: exhaustive, heuristic and random. The evaluation function computes the suitability of the selected subset and compares it with the previous best candidate, replacing it if the current subset is estimated as being better.

Besides, feature selection algorithms may be classified into two categories depending on their evaluation procedure *filter or wrapper*. Feature statistical properties are taken into account in *filter* approach while *wrapper* methods are based on a classifier the efficiency of which is optimized by learning on a training data set while selecting the features. Pros and cons of both approaches are considered in the following sub-sections.

2.1 Filter Approach

The *filter* model (see Figure 2) was the first one used in feature selection. The used criterion for feature relevance evaluation is based on measures that rely on training data properties. Different measures [Guyon and Elisseeff, 2003] may be used such as correlation criterion [Hall, 2000], Fischer criterion [Furey et al., 2000], mutual information [Ben-Bassat, 1983], consistency [Dash and Liu, 2003] and signal to noise ratio. This type of method is usually considered as a pre-processing step (filtering) done before the training phase. In other words, evaluation is generally done independently of any classifier [John et al., 1994]. Methods that are based on this feature evaluation model often use a heuristic approach as search strategy [Chapelle and Vapnik, 2000].

The main advantage of filtering methods is their computational efficiency and robustness against over-fitting. Unfortunately, these methods do not take into account interactions between features and tend to select features that are redundant rather than complementary. Furthermore, these methods do not absolutely take into account the performance of classification methods subsequent to selection [Kohavi and John, 1997].

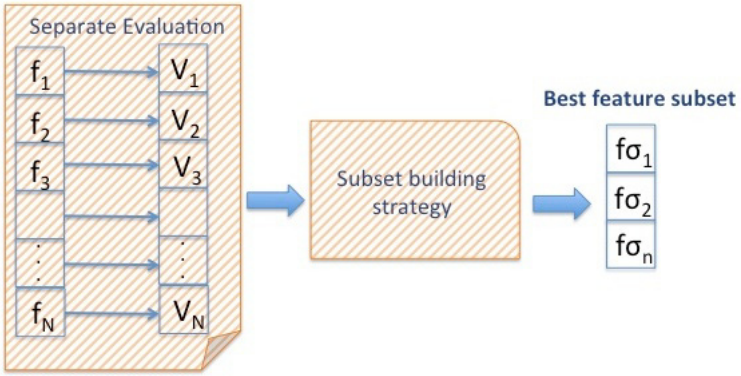

Fig. 2 General overview of a filter selection method

2.2 *Wrapper Approach*

As seen in the previous section, the main drawback of *filter* approaches is that they ignore the potential influence of the selected features on the performance of the classifiers to be used later. To solve this problem Kohavi and John introduced the concept of *wrapper* for feature selection [Kohavi and John, 1997]. The *wrapper* methods (see Figure 3) evaluate feature subsets on the basis of their classification performances using a learning algorithm.

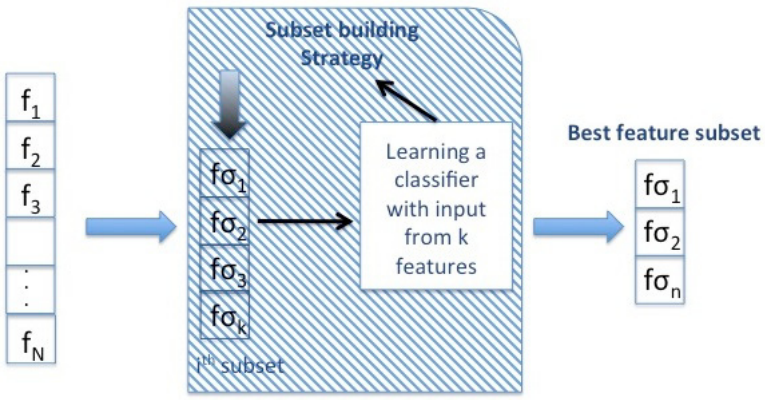

Fig. 3 General overview of a wrapper selection method

This evaluation is done using a classifier that enables to estimate the relevance of a given feature subset. The feature subset selected is always well adapted to the used classification algorithm but it is not necessarily valid if the classifier is changed. The

complexity of the learning algorithm makes the wrapper methods rather expensive regarding time complexity. It was shown that these methods generally give better results than filter methods, as they take into account both feature interactions and interactions between the classifier and the data set [Kohavi and John, 1997; Li and Guo, 2008; Huang et al., 2008].

However, they are time consuming because the learning step is performed with each feature subset. This main drawback makes it impossible to use an exhaustive search strategy (NP-complete problem), heuristics or random search strategies are then often preferred. Though, even in this case, the search becomes more and more unconceivable as the initial feature set size increases. An other drawback is the dependence of the relevant selected features on the used classifier. The feature evaluation is done using a chosen classifier during the selection phase. Each classifier has its own specificities and assumptions. But, if a change of classifier is required in order to better fit evolving data, the all selection process has to be restarted.

Genetic Algorithms seem to be a well suited search strategy used to take into account the dependency between features and to come as near as possible to the optimum. They are more global than forward or backward strategies.

2.3 Genetic Algorithms and Feature Selection

Genetic algorithms (GA) are one of the latest techniques in the field of feature selection [Kitoogo and Baryamureeba, 2007; Kim et al., 2000b; Oliveira et al., 2002; Yang and Honavar, 1998; Leardi, 1994]. Unlike classical feature selection strategies where one solution is optimized, a population of solutions can be modified at the same time. This can result in several optimal feature subsets as output. To apply a GA to solve a given problem, one should encode its potential solutions by finite strings of bits forming chromosomes. The main opened questions are the definition of an evaluation function, the *fitness* function, that allows good chromosome discrimination as well as the definition of genetic operators that will be used. The *fitness* function can be used either in *filter* or *wrapper* models.

The fitness evaluation of all chromosomes (coding each feature subset) in all generations can be very costly. This is particularly a problem for *wrapper* approaches where each chromosome is associated with a classifier that has to be trained and evaluated. To limit this problem we propose and describe in next section a new fast feature selection method (FFSM) that takes advantage of both *filter and wrapper* approaches

- As in *filter* methods, quality is associated with each feature.
- As in *wrapper* methods, the efficiency of a classifier is optimized. It is built on a feature subset, and do not need a new learning phase for each feature subset.

3 Proposed FFSM Method

On one side, filtering methods for feature selection have limitations with regard to the consideration of potential interactions between features. On the other side, *wrapper* methods present a very high time complexity and dependence on the classifier evaluation. Filtering methods derive their rapidity from taking into account features in an individual manner. We retain this idea by building a set of classifiers, each associated with one feature. They will be defined in Section 3.2. The overall vision of the *wrapper* method is preserved while considering a selection criterion that takes into account all the used features. This is implemented in a GA whose *fitness* function will be detailed in Section 3.3. Thus, we consider interactions between features. Finally, the features associated with the subset of classifiers selected at the last iteration represent the final feature subset. But first, in Section 3.1, an overall vision of FFSM method citeChouaib12 is given.

3.1 Selection Process

Let set $F = \{f_1, f_2, \ldots, f_N\}$ be composed of N features and $B_{app} = \{X_1, X_2, \ldots, X_M\}$ be a training dataset consisting of M samples where each $X_i = (f_{i1}, f_{i2}, \ldots, f_{iN})$ represents the i^{th} sample. A sample of dimension N is represented by a vector whose components are the values of features (f_i), where N is the total number of features. Let $Y = \{y_1, y_2, \ldots, y_M\}$ be the sample labels. For a bi-class classification problem we have $y_i \in \{-1, 1\}$. In order to minimize overfitting possibilities, the training set was divided into two parts: a training dataset A which contains M_A samples used to build the classifier set, a validation dataset V which contains other M_V samples used by the GA algorithm. Figure 4 represents the two-step process of our feature selection method:

- The construction of N simple classifiers H_i through a learning based on the dataset A. For each H_i, only the i^{th} feature f_i is taken into account.
- A selection among classifiers (H_i) by mean of a GA using the V dataset.

The first selection step consists in building a set of classifiers that represent the initial features which will be given as inputs to the GA. Each classifier is a simple model trained on a single feature. Once this classifier set is built, in the second step, we apply a genetic algorithm to select, after several generations, a *good* subset of classifiers. The features associated with the final selected models represent the final feature subset.

3.2 Classifier Set

In this step, a set of classifiers is built so that each classifier input is based on only one feature that can be either ordinal, with one or several dimensions, or symbolic. A classifier learnt on a single feature is a simple model defined using a learning

algorithm based on the content of the A training dataset. Such a classifier must be simple and as efficient as possible on a single feature.

It is among these classifiers that a subset, optimizing the defined criterion, will be extracted. This optimization will be carried out by a GA. It is then necessary to encode the subsets. The most classical way consists in encoding each possible solution by a binary string of size equal to the total number of classifiers, N. A gene of index i has the value 1 if the initial set i^{th} classifier is present in the current subset and 0 otherwise. We denote by $C = (c_1, c_2, \ldots, c_N)$ a chromosome where each c_i belongs to $\{0, 1\}$ and S_c is the set $\{i/c_i = 1\}$.

This type of coding prevents selecting more than once the same classifier for an individual. The control of the selected classifier number is left to the GA, which can be a major drawback in some applications.

The selection criterion, expressed in the GA's fitness function, is specified in the following section.

3.3 Selection Criterion

The problem is to find a subset having a reduced number of highly efficient classifiers. In *wrapper* methods, the *fitness* function is related to the building of a new classifier based on features that are involved in the individual (feature subset). To overcome the heaviness of this approach, we made a compromise. We build a new classifier that does not need a training phase but involves all the features present in the individual. Thus each selected classifier participates in the decision making. Therefore, we introduce, without any new learning phase, a classifier built as a combination of classifiers:

$$H^c = \boldsymbol{Comb}_{i \in S_c}(H_i) \tag{1}$$

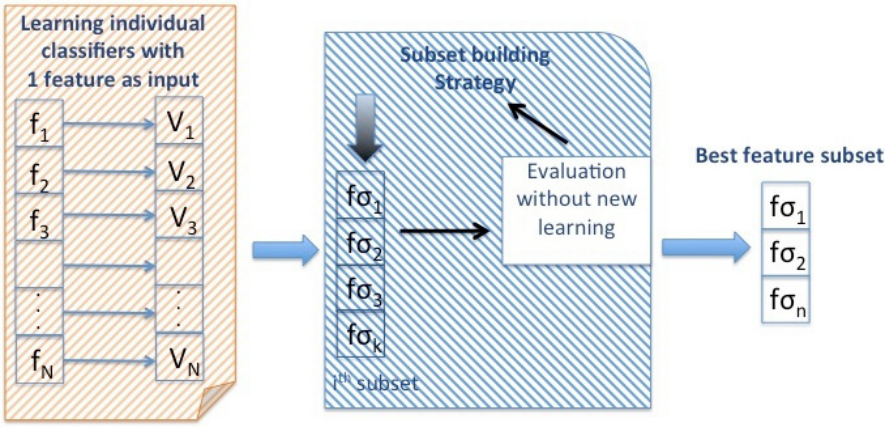

Fig. 4 General flow chart of the FFSM method

where $\boldsymbol{Comb}_{i \in S_c}(H_i)$ is the combination of the classifiers present in an individual and S_c is the set $\{i/c_i = 1\}$. Thus, for the GA fitness function, we compute the error given by this new classifier, related to the V sample set. It can be written as:

$$fitness = error(H^c) \tag{2}$$

Let see below an example showing how to compute the $fitness$ of an individual: Given a set of classifiers $H = \{H_1, H_2, \ldots, H_{12}\}$ which contain twelve classifiers. Suppose we want to combine classifiers whose answers are between -1 and +1 (-1 being associated with an element of the first class labeled -1 and +1 characterizing an element of the second class labeled +1). Let $I =$ "100110010110" be an individual for which six out of twelve classifiers are present. If we use the mean as a method for combining the classifiers then $\boldsymbol{Comb}_{i \in S_c}(H_i) = \frac{1}{6}(H_1 + H_4 + H_5 + H_8 + H_{10} + H_{11})$ the *fitness* function on I will be calculated as follows:

$$fitness(I) = error(sign(\boldsymbol{Comb}_{i \in S_c}(H_i)))$$

The ***Comb*** operator can take many forms, such as voting methods or mean approaches. Some of them will be discussed in Section 4.5.

4 Experiments

In this section we present the experiments carried out to illustrate the FFSM method. We first describe the used databases. In different studies the experiments are performed on different databases that have different properties. Then the comparisons are very difficult to draw. So we have decided to use a single application and several kinds of descriptors are applied to this single problem. Thus, this approach enables to define several databases with different dimensions as input of the system. The difficulty of the problem is exactly the same in the different cases but the descriptors have different properties (projection on base with loss of information, or raw data, ...). Before presenting the results and comparisons with other selection methods, we describe the problem and the experimental protocol. The different descriptors that are used are presented, they enable to build four different databases on which our trials are based. Then, we present the choice of implementation of our method at different levels, the classifiers, the combination method and the GA.

4.1 Problem and Material

For our experiments we used the *MNIST* database. It is a database of isolated handwritten digits (from 0 to 9) built in 1998 [Lecun et al., 1998]. Each digit is associated with an image of size 28×28 in 256 grey levels (example in Figure 5). The *MNIST* database is divided into two subsets, a training set of 60 000 examples and a test set of 10 000 examples.

We process *a priori* two-class problems, in the more general case of n classes it is necessary to build subsets within the labelled sample set to allow the use of a *one*

Fig. 5 Samples of images extracted from the *MNIST* database

versus all approach. Each subset is associated with a class. Let $A = \{A_i\}$, $V = \{V_i\}$ and $T = \{T_i\}$, which represents respectively training, validation and test datasets. A_i, V_i and T_i are constructed for the use of a *one versus all* method. On the one hand, each of the A_i datasets and T_i contain $2 * N_p$ samples: N_p samples of class i and N_p samples of all the other classes. On the other hand V_i only contains N_p elements: $\frac{N_p}{2}$ samples of class i and $\frac{N_p}{2}$ samples of all the other classes. For the *MNIST* dataset we have $i \in \{0,1,2,..,9\}$ and $N = 1000$.

4.2 Descriptors – Databases

We have used four descriptors for the representation of these data:

- Generic Fourier descriptor (GFD)[Zhang and Lu, 2002] is a descriptor based on the Fourier transform. The radial (***R***) and angular resolutions (***T***) represent two of its parameters.
- *R*-signature [Tabbone and Wendling, 2003] uses a Radon transform to represent an image.
- Zernike descriptor [Kim et al., 2000a] is a descriptor based on Zernike moments.
- Luminance of the 28×28 pixels.

Table 1 resumes the size of the feature vectors of each of these four data representations, constituting four experimental databases.

Table 1 Vector dimension (number of features) for each database

Name of database		***Dimension***
GFD	R=8,T=12	96
	R=10,T=15	150
R-signature		180
Zernike		66
Pixels		784

4.3 Classifier Sets

As we deal with numerical features, we propose to take advantage of the two following approaches to build a classifier set. One is to compute a single classification

threshold, the other, in order to improve the classifier efficiency, is to combine different weak classifiers in a strong classifier in order to obtain classifiers with multiple classification thresholds. This makes our approach original.

Case of Single Classification Threshold

In [Alamdari, 2006], a simple binary classifier is proposed in order to evaluate features individually. It was used as a selection criterion in a feature filtering method. The threshold that was used is the midpoint of a segment whose endpoints are the barycentre of data feature values in each class. In the following, we use the name *Classif_Alamdari* for this classifier. This classifier, for the i^{th} feature is defined by: Let $X^i = \{f_{1i}, f_{2i}, .., f_{Mi}\}$ be a feature value set where each element is the value of the i^{th} feature of one of the training samples. Let $X^{i,1} = \{f_{ki} | y_k = 1\}$ and $X^{i,-1} = \{f_{ki} | y_k = -1\}$.

$$y = sign((f_i - \frac{\mu_i^1 + \mu_i^{-1}}{2})(\mu_i^1 - \mu_i^{-1})) \tag{3}$$

Where μ_i^1 and μ_i^{-1} represent means of data for the i^{th} feature of class "1" and class "-1" respectively.

Another classifier of this type is the *decision stump*. It is a decision tree with only one internal node (the root) which is immediately connected to the terminal nodes [Iba and Langley, 1992]. It defines the best threshold that minimizes the classification error on a single feature.

Case of Multiple Classification Thresholds

To improve the efficiency of so simple classifiers presented in the previous paragraph, we propose to introduce multiple classification thresholds in the classifier building. To do that, we associate a threshold with the nodes of a decision tree [Breiman et al., 1984], or we use an AdaBoost [Freund and Schapire, 1995] algorithm from weak classifiers of type *decision stump* computed on different sample sets. Then, an H classifier is associated with feature f. This is illustrated in figure 6.

4.4 Genetic Algorithm

A genetic algorithm is composed of several parts. First a population has to be initialised, then the population evolves along iterations through genetic operators. The individuals are evaluated by the fitness function value and are introduced in the next generation by means of an elitist process. The evolution is stopped when some criterion is reached. The fitness function has been described in Section 3.3. As the Genetic Algorithm is not the purpose of the paper we just present here the choices that have been made and the parameter values that have been experimentaly fixed.

The initial population is composed of 200 chromosomes, a higher number of individual does not improve the results while increasing processing time. The

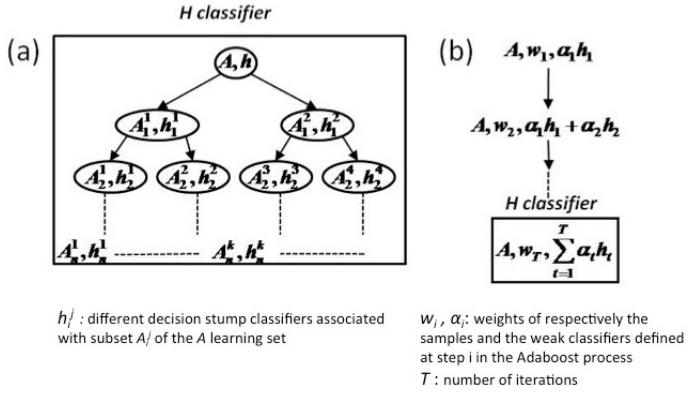

Fig. 6 Modeling of a Classifier H using multiple thresholds (a)Decision Tree (b)Adaboost

individuals have been randomly initialised with genes value being 0 or 1 according to a Bernouilli probability law with parameter p. This p parameter enables to handle the decrease of the number of selected features. The p value is here experimentaly fixed to 0.5. The genetic operators used are quite common. The crossover operator is a one-point crossover. And the mutation operator concernes the switching of one gene form 1 to 0 or from 0 to 1 according to the initial value with a probability fixed to 0.005. The members of a generation are becoming parents of next generation individuals using a tornament process where the best individual among 3 randomized individuals is selected. The stopping criterion is the maximal number of generations, it is set to 50 as we have experimented the evolution of the best individual quality is not more significant when this value is increased.

4.5 Classifier Combination

For combining classifiers, we used several conventional combining methods such as majority voting, weighted majority voting, mean, weighted mean and median. Another method that we used, is called *AWFO* (**A**ggregation **W**eight-**F**unctional **O**perators) [Dujet and Vincent, 1998]. In *AWFO* method, the assigned weights to each of the values given by the classifiers are adaptive. They do not only depend on each value but also on the general distribution of data. For the AWFO method we propose a modification to make it better adapted to our case. In the original version, it is assumed that the set of values to aggregate belongs to an interval on which the quality of values with respect to a goal is monotone (see Figure 7a). In our case of a two-class classification, the two classes both have an equivalent status, making it impossible to define a *distinguished value* with a significant value with respect to the problem (see Figure 7b).

In our case we want to combine classifiers whose answers are between -1 and +1. We can say the more a positive value is near +1, the more the element has a chance to belong to class labeled +1, and the more a negative value is near -1, the more the

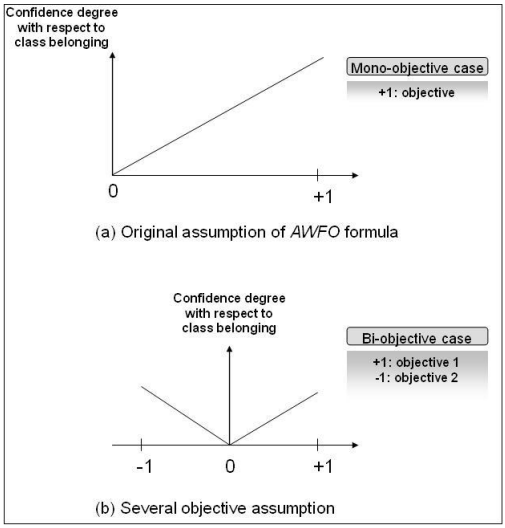

Fig. 7 Mono(a) and bi-objective (b) contexts of *AWFO* aggregation method

element has a chance to belong to class labeled -1. Therefore, we have chosen to aggregate separately the positive and the negative values. Refering to the original method, we have two *optimal* values. Thus, we have two *distinguished values*, -1 for negative values and +1 for positive values. The *AWFO* method does not only consider the classifier's answer but also the distribution of all answers to achieve aggregation.

Let us give an example to better understand the principle of the method. If we have the answers of ten classifiers (x_i for $i \in \{1,2,..,10\}$) with five positive answers and five negative answers, we propose to compute a weight for each positive answer while taking into account the other positive answers (respectively a weight for each negative answer while taking into account only the other negative answers). The weight of each answer ($W(x_i)$) is calculated using equation 4:

$$W(x_i) = \frac{d_{cum}(x_i)}{\sum\limits_{sign(x_j)=sign(x_i)} d(x_j)} \tag{4}$$

where

$$\begin{cases} d_{cum}(x_i) = \sum\limits_{j \in E_i} d(x_j) \text{ with } E_i = \{j/(d(x_j) \geq d(x_i)) \ \& \ (sign(x_j) = sign(x_i))\} \\ \text{and} \\ d(x) = 1 - |x| \end{cases}$$

In this formula, $d(x)$ is the distance between x and the associated distinguished value +1 or -1 according to the sign of x. Table 2 shows the details of this example.

Table 2 Combination example with the AWFO method

Initial answers	-0.2	0.4	-0.5	-0.7	0.8	0.1	-0.9	0.6	0.5	-0.3
Sorted answers (x_i)	-0.9	-0.7	-0.5	-0.3	-0.2	0.1	0.4	0.5	0.6	0.8
Distance (d)	0.1	0.3	0.5	0.7	0.8	0.9	0.6	0.5	0.4	0.2
Cumulative distance (d_{cum})	2.4	2.3	2	1.5	0.8	0.9	1.5	2	2.4	2.6
Final weight ($W(x_i)$)	1	0.95	0.83	0.62	0.33	0.34	0.57	0.77	0.92	1

Finally, in the case of negative answer very close to -1 and if we have a lot of negative answers, the cumulative sum of distances increases and its weight will be high. Similarly for an answer very close to +1.

5 Results

In this section, we show the contribution of combining methods used in our approach and we compare the results with those obtained by other feature selection methods. Indeed, the aim of the method is to select the lowest number of features, while keeping the efficiency of the recognizer system or even improving it. The quality of a recognition method is linked to the quality of the features, our purpose is not to solve the problem of figure recognition but to prove the efficiency of the FFSM method according to the feature nature.

5.1 Evaluation

In this section, we show the results obtained on the databases defined in Section 4.2 using different types of classifiers. The construction of the initial set of simple classifiers is made using one of the classifiers described in Section 4.3. The first considered classifier is an AdaBoost classifier. On the one hand, it finds several thresholds adapted to the learning examples, this is an advantage compared to classifiers based on a single threshold. On the other hand, the answer of this classifier is numeric: the sign indicates the class and the module gives a kind of confidence degree between 0 and 1. This output format allows the implementation of different combining methods.

After simple classifiers set building using the AdaBoost algorithm, and after best classifier subset selection for the different databases, an experimental study was carried out to evaluate the combining method's influence on the selected subset quality. Table 3 shows the average number of selected features for each descriptor on the ten classes of our experimental databases. We notice that the final subsets are on average 69.9% smaller than the initial set. We can also notice the regular aspect of the dimension reduction ratio as the normalized standard deviation values are low and similar.

To evaluate the quality of the subsets found by the FFSM method, we did not used the classifier involved in the GA selection process but we chose a classifier the

Table 3 Number of selected features in each experimental databse used for digit recognition

	Zernike	*GFD_8×12*	*GFD_10×15*	*R-signature*	*Pixels*	*Mean*
Initial	66	96	150	180	784	-
Mean	25	30	46	42	245	-
Standard Deviation (SD)	3.72	4.57	5.36	10.91	17.43	-
Normalized SD	0.15	0.15	0.12	0.26	0.07	0.15
% reduction	66.12	68.75	69.33	76.66	68.75	69.92

efficiency of which is generaly admitted, a SVM classifier learnt on training datasets A_i and tested on datasets T_i. Table 4 shows the classification average rate for each experimental database before and after selection.

Table 4 Results of a SVM classifier with and without selection for each experimental database used for digit recognition

	Without selection	*With selection*	*% variation*
Zernike	92.47 ± 3.99	92.42 ± 4.19	-0.04
GFD_8×12	92.38 ± 3.48	92.55 ± 3.35	+0.17
GFD_10×15	91.97 ± 4.10	92.10 ± 3.66	+0.13
R-signature	75.95 ± 7.67	79.55 ± 6.97	+3.60
Pixels	97.73 ± 1.03	97.60 ± 1.05	-0.13

We can notice that the rates before and after selection are relatively close regardless of the descriptor. In all the cases we managed to select feature subsets 69.9% smaller than the originals sets but with similar recognition rate. We notice the improvment of recognition rate occurs when the initial recognition rate is the lowest. The nature of the features can explain this fact. Besides, the rates depend on the digit recognized.

Finally we compared the selection results using different combining methods. Table 5 shows the comparison results averaged on the ten classes. In this table, we note that the results may be gathered in two groups. Within the two groups, the results are not significantly different. The three combining methods *AWFO, mean* and *weighted mean* are close for different databases and are more efficient than the *majority voting*, *weighted majority voting* and *median combining methods*.

We have also tested different types of classifiers using the previous combining methods. As the results are not significantly different we present in table 6 the best ones using the three different classifiers *Decision stump*, *Classif_Alamdari* and *Decision trees*.

We can notice from such results that AdaBoost classifiers give the best selection result. The obtained results from decision tree classifiers are close to those of AdaBoost.

Then the FFMS method as a whole, combining multiple views of the database, is not too sensible to the different choices that may be made in the various steps. We

Table 5 Comparison of the different combining methods

	Zernike	*GFD_8×12*	*GFD_10×15*	*R-signature*	*Pixels*	*Mean*
AWFO	92.25	92.55	92.08	79.19	97.60	90.74
Mean	92.42	91.90	91.95	79.04	97.40	90.54
Weighted mean	92.28	92.34	92.10	79.55	97.55	90.76
Majority voting	91.71	90.55	91.65	77.38	96.80	89.61
Weighted voting	91.95	91.95	90.98	79.05	97.20	90.22
Median	92.14	91.06	92.05	78.25	97.5	90.20
Without selection	92.47	92.38	91.97	75.95	97.73	90.10

Table 6 Comparison of results drawn from several classifiers

	Zernike	*GFD_8×12*	*GFD_10×15*	*R-signature*	*Pixels*
AdaBoost	92.42	92.55	92.1	79.55	97.6
Classif_Alamdari	91.50	90.15	90.85	74.15	93.85
Decison_stump	92.05	92.05	91.95	79.25	97.45
Decision trees	91.95	92.17	91.85	79.35	97.50

have here made the classifiers and the combining vary. In any application the user of the FFMS method may incorporate the elements he is the most familiar with or adapt some to its specific problem.

Using genetic algorithm the results may depend on the various runs of the process. Then we have run the process several times and we present in table 7 the best, the average and the standard deviation while applying 5 times the process using the same environment and choices for the different elements. The classifiers are Adaboost classifiers, the combination is an AWFO operator. We can notice the results are stable as the standard deviation is low.

Table 7 Stability of selection on the recognition rates

	Zernike	*GFD_8×12*	*GFD_10×15*	*R-signature*	*Pixels*
best	92.25	92.34	92.08	79.19	97.6
mean	92.08	92.21	91.98	78.89	97.45
standard-deviation	0.14	0.11	0.11	0.19	0.09
without selection	92.47	92.38	91.97	75.95	97.73

5.2 Comparison with Other Selection Methods

We compared our selection method with three other existing methods. These methods are based on *filter* and *wrapper* evaluation approaches. We considered three methods: *Relief* [Kira and Rendell, 1992], *SAC* [Kachouri et al., 2010] and the third one is a classic *wrapper* method based on random search and using the same GA as our method, with the same parameters but with a different *fitness* function

defined by the classification error of a *SVM* classifier. Table 8 shows the comparison of results between our method and other selection methods. Results are computed on the mean of ten digit classes using the same databases as in Section 4.2 taken from the *MNIST* dataset.

Table 8 Comparison with other methods for each experimental database

	Relief	***Wrapper_SVM***	***SAC***	***FFSM method***
Zernike	89.85	92.61	91.11	92.42
GFD_8×12	90.05	92.55	91.15	92.55
GFD_10×15	90.15	92.01	91.45	92.10
R-signature	73.55	80.05	75.88	79.55
Pixels	95.85	97.68	96.35	97.60

We can notice that our method is significantly better than *Relief* and *SAC* methods. These results are very close to the *Wrapper_SVM* method for all experimental databases (Table 8), but the computation time of our method is significantly lower than the one of the *Wrapper_SVM* method. Table 9 shows on the one hand, that our method, in worst case is 125 times faster and 250 times faster in the best case (for the database *Pixels*) and on the other hand, that the size of feature subsets selected by our method is 6% smaller in worst case and 15% smaller in the best case. The

Table 9 Comparison of computation relative time for feature selection and number of selected features with FFSM method and the *Wrapper_SVM* method

		Zernike	***GFD_8×12***	***GFD_10×15***	***R-signature***	***Pixels***
FFSM method	Nb of features	25	30	46	42	245
	Time	0.001	0.0015	0.0022	0.0026	0.004
Wrapper_SVM	Time	0.13	0.22	0.28	0.36	1
	Nb of features	36	52	65	79	299

reference time concerns the case of 784 features on a bi-class problem, processed by a matlab (c) software on a 2GHz processor computer. With the classic wrapper method, the duration is equal to 489 minutes, where as with our FFSM method duration is less than 2 minutes.

6 Conclusion

In this paper, a combination of single feature classifiers and a genetic algorithm are used to define a new fast feature selection method. The used fitness function is based on a combination of single feature classifier. Many classifiers and combining methods are possible and we have illustrated some of them, showing their efficiency. It

is obvious the user of the FFMS process may introduce its own choices either for the classifiers or the combination process. Our experiments on the four databases issued from the digit recognition problem using the MNIST dataset show that similar results can be obtained using about 69.9% less features for different descriptors whatever their properties are. Moreover, the proposed method is faster, in the worst case, 125 times than a classical wrapper method. The method can be adapted in any context as the simple classifier construction is free, the only constrain is to have a numerical output comprised between -1 and +1. The features may mix numerical and symbolical data. The selection is here presented as a selection method but may be applied at another level for descriptor selection. The method can be applied several times and then enables to define some hierarchical subset among the features. The experiment we have conducted on these four databases as well as on other databases, more precisely on databases studied in bioinformatics showed the stochastic aspect of our method was not leading to results with a too heterogeneous quality.

In most applications, the real problem is to find the best representation space, in which the problem is solved in the easiest way, that is to say where the error rate or evaluation indexes are optimum. To do so, a feature selection process enables to consider only relevant and robust features. These common features are mathematical functions with generic properties. The nature of the data is not taken into account in the learning phase of a classifier for example.

In our work, we have changed the features' definitions to replace them by new features that are given by the classifier functions. When the feature values are very intricate, the classifier function is a modification of the feature according to the data. This makes our method robust and not too much dependant on the general classifier used in the chosen representation space.

Indeed some improvements can be added. We here indicate some hints. Whereas only one criterion is used in the optimization phase, an error rate, some other properties of the features might be considered such as the classifier's diversity that could minimize redundancy between the selected features. Thus, a multi-objective approach can be used to integrate this new objective.

References

[Alamdari, 2006] Alamdari, A.: Variable selection using correlation and single variable classifier methods: Applications. In: Guyon, I., Nikravesh, M., Gunn, S., Zadeh, L. (eds.) Feature Extraction. STUDFUZZ, vol. 207, pp. 343–358. Springer, Heidelberg (2006)

[Ben-Bassat, 1983] Ben-Bassat, M.: Use of distance measures, information measures and error bounds in feature evaluation. In: Krishnaiah, P., Kanal, L. (eds.) Classification, Pattern Recognition and Reduction of Dimensionality. HandBook of Statistics II, vol. 2, pp. 773–791. North Holland (1983)

[Bins and Draper, 2001] Bins, J., Draper, B.: Feature selection from huge feature sets. In: Proceedings of the International Conference on Computer Vision (ICCV), pp. 159–165. IEEE (2001)

[Bouguila and Ziou, 2012] Bouguila, N., Ziou, D.: A countably infinite mixture model for clustering and feature selection. Knowledge and Information Systems 33, 351–370 (2012)
[Breiman et al., 1984] Breiman, L., et al.: Classification and Regression Trees. Chapman and Hall, New York (1984)
[Chapelle and Vapnik, 2000] Chapelle, O., Vapnik, V.: Model selection for support vector machines. In: Proceedings of the Neural Information Processing Systems, ANIPS 2000, Denver, Colorado, USA, pp. 230–236. MIT Press (2000)
[Dash and Liu, 2003] Dash, M., Liu, H.: Consistency-based search in feature selection. Artif. Intell. 151(1-2), 155–176 (2003)
[Dujet and Vincent, 1998] Dujet, C., Vincent, N.: Data fusion modeling human behavior. International Journal of Intelligent System 13, 27–39 (1998)
[Freund and Schapire, 1995] Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. In: Vitányi, P.M.B. (ed.) EuroCOLT 1995. LNCS, vol. 904, pp. 23–37. Springer, Heidelberg (1995)
[Furey et al., 2000] Furey, T.S., Cristianini, N., Duffy, N., Bednarski, D.W., Schummer, M., Haussler, D.: Support vector machine classification and validation of cancer tissue samples using microarray expression data. Bioinformatics 16(10), 906–914 (2000)
[Guyon and Elisseeff, 2003] Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. J. Mach. Learn. Res. 3, 1157–1182 (2003)
[Hall, 2000] Hall, M.: Correlation-based Feature Selection for Discrete and Numeric Class Machine Learning. In: 17th International Conference on Machine Learning, ICML 2000. LNCS, pp. 359–366. Morgan Kaufmann Publishers, San Fransico (2000)
[Huang et al., 2008] Huang, C.-J., Yang, D.-X., Chuang, Y.-T.: Application of wrapper approach and composite classifier to the stock trend prediction. Expert Syst. Appl. 34, 2870–2878 (2008)
[Iba and Langley, 1992] Iba, W., Langley, P.: Induction of one-level decision trees. In: Proceedings of the ninth International Workshop on Machine Learning, ML 1992, pp. 233–240. Morgan Kaufmann Publishers Inc., San Francisco (1992)
[John et al., 1994] John, G.H., Kohavi, R., Pfleger, K.: Irrelevant features and the subset selection problem. In: Machine Learning: Proceedings of the Eleventh International Conference, pp. 121–129. Morgan Kaufmann (1994)
[Kachouri et al., 2010] Kachouri, R., Djemal, K., Maaref, H.: Adaptive feature selection for heterogeneous image databases. In: Djemal, K., Deriche, M. (eds.) Second IEEE International Conference on Image Processing Theory, Tools 38; Applications, 10, Paris, France (2010)
[Kim et al., 2000a] Kim, H., Kim, J., Sim, D., Oh, D.: A modified zernike moment shape descriptor invariant to translation rotation and scale for similarity-based image retrieval. In: ICME 2000, p. MP5 (2000a)
[Kim et al., 2000b] Kim, Y., Street, W., Menczer, F.: Feature selection in unsupervised learning via evolutionary search. In: 6th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), pp. 365–369 (2000b)
[Kira and Rendell, 1992] Kira, K., Rendell, L.A.: The feature selection problem: Traditional methods and a new algorithm. In: AAAI, pp. 129–134. AAAI Press and MIT Press, Cambridge, MA, USA (1992)
[Kitoogo and Baryamureeba, 2007] Kitoogo, F.E., Baryamureeba, V.: A methodology for feature selection in named entity recognition. International Journal of Computing and ICT, 18–26 (2007)
[Kohavi and John, 1997] Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artif. Intell. 97, 273–324 (1997)

[Leardi, 1994] Leardi, R.: Application of a genetic algorithm to feature selection under full validation conditions and to outlier detection. Journal of Chemometrics 8(1), 65–79 (1994)

[Lecun et al., 1998] Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE, 2278–2324 (1998)

[Li and Guo, 2008] Li, Y., Guo, L.: Tcm-knn scheme for network anomaly detection using feature-based optimizations. In: Proceedings of the 2008 ACM Symposium on Applied Computing, SAC 2008, pp. 2103–2109. ACM, New York (2008)

[Liu and Yu, 2005] Liu, H., Yu, L.: Toward integrating feature selection algorithms for classification and clustering. IEEE Transations on Knowledge and Data Engineering 17, 491–502 (2005)

[Oliveira et al., 2002] Oliveira, L.S., Sabourin, R., Bortolozzi, F., Suen, C.Y.: Feature selection using multi-objective genetic algorithms for handwritten digit recognition. In: Proceedings of the 16th International Conference on Pattern Recognition, ICPR 2002, vol. 1. IEEE Computer Society, Washington, DC (2002)

[Tabbone and Wendling, 2003] Tabbone, S., Wendling, L.: Binary shape normalization using the Radon transform. In: Nyström, I., Sanniti di Baja, G., Svensson, S. (eds.) DGCI 2003. LNCS, vol. 2886, pp. 184–193. Springer, Heidelberg (2003)

[Yang and Honavar, 1998] Yang, J., Honavar, V.: Feature subset selection using a genetic algorithm. IEEE Intelligent Systems and their Applications 13(2), 44–49 (1998)

[Zhang and Lu, 2002] Zhang, D., Lu, G.: Shape based image retrieval using generic fourier descriptors. Signal Processing: Image Communication 17, 825–848 (2002)

[Zhou and Dillion, 1991] Zhou, X., Dillion, T.: A statistical-heuristic feature selection criterion for decision tree induction. IEEE Trans. Pattern Anal. Mach. Intell. 13, 834–841 (1991)