Fast Feature Selection for Handwritten Digit Recognition

Hassan Chouaib, Florence Cloppet and Nicole Vincent
Laboratoire LIPADE
Université Paris Descartes
Paris, France
Email: firstname.lastname@mi.parisdescartes.fr

***Abstract*—Feature selection happens to be an important step in any classification process. 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. Variability of handwriting makes features more or less efficient and gives a good support for evaluation of selection method. The selection methods described in the literature present some limitations at different levels. Some are too complex or too dependent on the classifier used for evaluation. Others overlook interactions between features. In this paper, we propose a fast selection method based on a genetic algorithm. Each feature is closely associated with a single feature classifier. The weak classifiers we consider have several degrees of freedom and are optimized on the training dataset. The classifier subsets are evaluated by a fitness function based on a combination of single feature classifiers. Results on the MNIST handwritten digits database show how robust our approach is and how efficient the method is.**

***Keywords*-Feature selection; Classifier combination; Handwritten Digit Recognition; Genetic algorithm; AWFO**

I. Introduction

In the field of handwritten digits recognition, variability of handwriting represents a challenge. Then many recognition system have been proposed and many works have been devoted to feature definition. High dimension spaces are often used as representation spaces. Many handwritten recognition system have been proposed and tested on the *MNIST* database [1] [2]. In our case the aim is not to propose a handwritten digit recognition system, but to show the efficiency of our fast feature selection method for such systems. Reducing vector dimensionality is often considered as a preprocessing step dedicated to noise and redundant information elimination. One type of dimensionality reduction methods is feature selection.

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.

The paper is organized as follows. In section II, we motivate our choices, in section III, we introduce our **F**ast **F**eature **S**election **M**ethod (FFSM). In section IV an extensive experimental study on the *MNIST* database is carried out. Finally, conclusions are drawn and perspectives are given in section V.

II. Feature selection

Feature selection is generally defined as a search process to find a *"relevant"* feature subset from an initial feature set. The relevance of a feature set always depends on objectives and criteria. A selection method [3] generally incorporates several steps: First a search starting point is set then subsets are evaluated and a search strategy is applied to make the subset evolve.

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*.

A. Filter approach

The *"filter"* model 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. This type of method is considered more as a pre-processing step (filtering) done before the training one. In other words, evaluation is generally done independently of any classifier [4]. Methods that are based on this feature evaluation model were developed in [5][6][7][8]. Most of them use an heuristic or random approach as search strategy.

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 [9]. Furthermore, these methods do not absolutely take into account the performance of classification methods subsequent to selection [10].

B. Wrapper approach

As seen in the previous section, the main drawback of *"filter"* approaches is that they ignore the potential influence

978-0-7695-4774-9/12 $26.00 © 2012 IEEE
DOI 10.1109/ICFHR.2012.203

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 [10]. The *"wrapper"* methods evaluate feature subsets on the basis of their classification performances using a learning algorithm.
This evaluation is done using a classifier that estimates 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. As the complexity drawback of this technique makes it impossible to use exhaustive search strategy (NP-complete problem), heuristics or random search strategies are often preferred. However, even in this case, the search becomes more and more inconceivable as the initial feature set size increases.

The wrapper methods are generally considered to be better than the filtering methods according to [11], [10]. They are able to select small feature subsets which are efficient for the used classifier, nevertheless the wrapper methods have two main drawbacks :

- Complexity and computation time
- Dependency of selected features on the classifier

C. Genetic algorithms and feature selection

Genetic algorithms (GA) are one of the latest techniques in the field of feature selection [12],[13]. Unlike classical feature selection strategies where one solution is optimized, a population of solutions can be modified at the same time. one should encode its potential solutions by finite strings of bits forming chromosomes of the candidate points population. The *fitness* function can be defined using filter or wrapper models.
The fitness evaluation of a genetic algorithm for feature selection, can be very costly as there are many generations of many feature subsets that must be evaluated. This is particularly a problem for wrapper approaches where classifiers are induced and evaluated for each chromosome. To limit this problem we propose to find a classifier that takes advantage of both *filter and wrapper* approaches.
- *filter* methods: quality is associated with each feature.
- *wrapper* methods: a classifier efficiency is optimized.
These objectives represent the main idea of our new fast feature selection method described in the next section.

III. 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 III-B. 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 III-C. 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. First, in section III-A, an overall vision of FFSM method is given.

A. Selection process

Let set $F = \{f_1, f_2, .., f_N\}$ be composed of N features and $B_{app} = \{X_1, X_2, .., X_M\}$ be a training dataset consisting of M samples where each $X_i = (f_{i1}, f_{i2}, .., f_{iN})$ 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 $Y = \{y_1, y_2, .., y_M\}$ be the sample labels. For a bi-class classification problem we have $y_i \in \{-1, 1\}$. The training set B_{app} is divided into two parts: a training dataset A and a validation dataset V. Our feature selection method is in two steps:

- Construction of N simple classifiers H_i. For each H_i, only the i^{th}feature f_i is taken into account.
- Selection of classifiers (H_i) by using a GA.

B. Classifier set

In this step, a set of classifiers is built so that each element is trained on only one feature. A classifier learnt on a single feature is a simple model defined using a learning algorithm based on the training dataset A and which only takes into account one feature at a time. Such classifier must be simple and as efficient as possible on a single feature.
To build such classifiers, we propose to take advantage of two approaches. One is to compute a single classification threshold, the other is to combine different weak classifiers in a strong classifier in order to obtain classifiers with multiple classification thresholds. This makes our approach original.
[14] use single classification threshold hereafter named *"Classif_Alamdari"*. The threshold that was used is the midpoint of a segment whose endpoints are the barycentre of data feature values in each class. Another classifier of this type is the *"decision stump"* [15]. It defines the best threshold that minimizes the classification error on a single feature.

As only one feature is considered at a time, 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 [16], or use AdaBoost algorithm [17] from weak classifiers of type *"decision stump"* computed

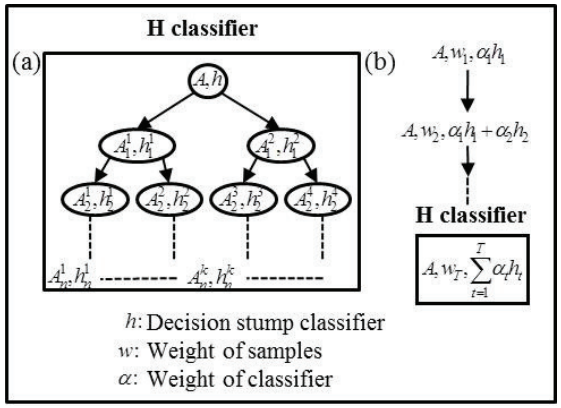

Figure 1. Multiple threshold classifier: (a)***Decision stump*** case (b)***Adaboost*** case

on different sample sets. Then, an H classifier is associated with feature f. This is illustrated in figure 1.
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. 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 $C = (c_1, c_2, .., c_N)$ a chromosome where each $c_i \in \{0, 1\}$ and S_c is the set: $\{i/c_i = 1\}$.

C. Selection criterion

The matter 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. 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. We can notice that the already built classifiers provide more adapted information to the problem than features themselves. Thus each selected classifier participates in the decision making. Therefore, we introduce a classifier built as a combination of classifiers.

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

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

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

We present below an example that shows how to compute the $fitness$ of an individual:

Given a set of classifiers (in this example, Adaboost classifiers) $H = \{H_1, H_2, .., H_{12}\}$ which contain twelve classifiers, where $H_i = \sum_{t=1}^{T} \alpha_{it} h_{it}$, $i \in \{1, 2, 3, .., 12\}$ and T is the number of AdaBoost iterations. Let $I =$ "100110010110" an individual for which six out of twelve classifiers are present. If we use the mean as a method for combining the classifiers then $\boldsymbol{Comb}_{i \in S_c}(H_i) = \frac{1}{6} \sum_k (\sum_{t=1}^{T} \alpha_{kt} h_{kt})$, the *fitness* function of I will be calculated as follows:

$$fitness(I) = error(sign(\tfrac{1}{6} \textstyle\sum_k (\sum_{t=1}^{T} \alpha_{kt} h_{kt}))) \text{ where } k \in \{1, 4, 5, 8, 10, 11\}$$

The ***Comb*** operator can take many forms that will be studied in the following section.

For combining classifiers, we used several conventional combining methods such as majority voting, weighted majority voting, mean, weighted mean, median. Another method was used called *AWFO* (**A**ggregation **W**eight-**F**unctional **O**perators) [18]. For the AWFO method we propose a modification to make it better adapted to our case. In our case of a two-class classification, the two classes have an equivalent status, making it impossible to define a *distinguished value*with a significant value with respect to the problem. We want to combine classifiers whose answers are between -1 and +1 (-1 being associated with an element of the first class and +1 characterizing an element of the second class). 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*. The *AWFO* method does not only consider the classifier's answer but also the distribution of all answers to achieve aggregation. Then weight formula becomes:

$$W(x_i) = \frac{d_{cum}(x_i)}{\sum\limits_{sign(x_j)=sign(x_i)} d_j} \tag{3}$$

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

Finally, in 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.

IV. Experiments and validation

In this section we present the experiments carried. We first describe several representation spaces and we compare the results with those obtained by other feature selection methods.

A. Databases and experimental protocol

For our experiments we used five feature databases. They have been built using different descriptors:

- Generic Fourier descriptor (GFD)[19]: is a descriptor based on Fourier transform. The radial (***R***) and angular resolutions (***T***) represent two of its parameters. We have used two different values of R and T. For (R=8,T=12), we obtain 96 features and for (R=10,T=15) we obtain 150 features. In the following, we use names *"GFD1"* and *"GFD2"* for these representation spaces.
- *R*-signature [20]: uses Radon transform to represent an image. There are 180 features in this descriptor hereafter named *"R_sig"*.
- Zernike descriptor [21]: is a descriptor based on Zernike moments. In our case, 66 moments are used.
- Luminance of the pixels. This descriptors is composed of 784 features hereafter named *"Pixel"*.

Features of each descriptor are calculated on the *MNIST* database. It is a database of isolated handwritten digits (from 0 to 9) built in 1998 [22]. Each digit is associated with an image of size 28 x 28 in 256 grey levels (example in Figure 2). The *MNIST* database is divided into two subsets, a training set of 60 000 examples and a test set of 10 000 examples.

Figure 2. Samples of images extracted from the *MNIST* database

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 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 samples: N samples of class i and N samples of all the other classes. On the other hand V_i only contains N elements: $\frac{N}{2}$ samples of class i and $\frac{N}{2}$ samples of all the other classes. For the *MNIST* dataset we have $i \in \{0, 1, 2, .., 9\}$ and $N = 1000$.

B. Results

In this section we made the different elements of our method vary: the nature of the classifiers, the nature of combination classifiers involved in the fitness function.

Table I shows the average number of selected features for each descriptor on the ten classes of *MNIST* database using *Adaboost* classifiers and *AWFO* combination method. We note that the final subsets are on average 69.9% smaller than the initial set. To evaluate the quality of the subsets found by the FFSM method, we used an SVM classifier learnt on training datasets A_i and tested on datasets T_i. The first and

Table I
NUMBER OF SELECTED FEATURES FOR EACH DESCRIPTOR USED FOR DIGIT RECOGNITION

	Zer	***GFD1***	***GFD2***	***R-sig***	***Pix***
Initial	66	96	150	180	784
Mean	25	30	46	42	245
% reduction	66.12	68.75	69.33	76.66	68.75

the last line of table II shows the classification average rate for each descriptor before and after selection. we can notice that we managed to select feature subsets 69.9% smaller than the originals sets but with roughly the same quality. Table

Table II
COMPARISON OF RESULTS DRAWN FROM SEVERAL CLASSIFIERS

	Zer	***GFD1***	***GFD2***	***R-sig***	***Pix***
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
No selection	92.47	92.38	91.97	75.95	97.73

II shows also the best results obtained for each classifier set and for each descriptor. From such results we can conclude that AdaBoost classifiers give the best selection result and the single threshold classifier are less efficient.

Finally we compared the selection results using different combining methods. Table III shows the comparison results averaged on ten classes. We can notice that the results for the three combining methods *AWFO, mean* and *weighted mean* are close for various used descriptors and are more efficient than the *majority voting*, *weighted majority voting* and *median combining methods*.

Table III
COMPARISON OF THE DIFFERENT COMBINING METHODS

	Zer	***GFD1***	***GFD2***	***R-sig***	***Pix***
AWFO	92.25	92.55	92.08	79.19	97.60
Mean	92.42	91.90	91.95	79.04	97.40
Weighted mean	92.28	92.34	92.10	79.55	97.55
Majority voting	91.71	90.55	91.65	77.38	96.80
Weighted voting	91.95	91.95	90.98	79.05	97.20
Median	92.14	91.06	92.05	78.25	97.5
No selection	92.47	92.38	91.97	75.95	97.73

C. Comparison with other methods

We compared our selection method with three other existing methods. We considered three methods: *Relief* [5], *SAC* [6] and the third one is a classic *wrapper* method based on random search and using the same GA as our method, but with a different *fitness* function defined by the classification error of a *SVM* classifier. These methods are based on different evaluation approaches which are *filter*(*Relief* and *SAC*) and *wrapper* (*Wrapper_SVM*). Table IV shows the comparison of results between our method and other selection methods. Results are computed on the mean

Table IV
COMPARISON WITH OTHER METHODS FOR EACH DESCRIPTOR

	Relief	*Wrapper_SVM*	*SAC*	*FFSM*
Zernike	89.85	92.61	91.11	92.42
GFD1	90.05	92.55	91.15	92.55
GFD2	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

of ten classes taken from the *MNIST* dataset.

Table V
COMPARISON OF COMPUTATION RELATIVE TIME FOR FEATURE SELECTION AND NUMBER OF SELECTED FEATURES WITH *FFSM* METHOD AND THE *Wrapper_SVM* METHOD

	FFSM		*Wrapper_svm*	
	# features	Time	Time	# features
Zer	22	0.001	0.13	36
GFD1	30	0.0015	0.22	52
GFD2	46	0.0022	0.28	65
R-sig	42	0.0026	0.36	79
Pix	245	0.004	1	299

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 descriptors (Table IV), but the computation time of our method is significantly lower than the one of the *Wrapper_SVM* method . Table V 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 descriptor *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 reference time concerns the case of 784 features on a bi-class problem, processed by a matlab (c) software on a 2GHz processor computer. The duration is equal to 489 minutes.

V. 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 fitness function used is based on a combination of single feature classifiers. Many classifiers and combining methods are evaluated. Our experiments on the MNIST dataset show that similar classification rate can be obtained using about 69.9% less features for different descriptors. Moreover, the proposed method is faster, in the worst case, 125 times than a classical wrapper method. Therefore, the robustness of the proposed approach is confirmed.

Future works will be devoted to study the classifier's diversity to minimize redundancy. Thus, a multi-objective approach can be used to integrate this new objective. Another perspective is to select features using hierarchical approach applied at several levels (feature and descriptor).

REFERENCES

[1] J.-X. Dong, A. Krzyzak, and C. Y. Suen, "Fast svm training algorithm with decomposition on very large data sets," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, vol. 27, no. 4, pp. 603–618, 2005.

[2] C.-L. Liu, K. Nakashima, H. Sako, and H. Fujisawa, "Handwritten digit recognition using state-of-the-art techniques," in *Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition*, Washington, DC, USA, 2002, p. 320.

[3] H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Transations on Knowledge and Data Engineering*, vol. 17, pp. 491–502, 2005.

[4] G. H. John, R. Kohavi, and K. Pfleger, "Irrelevant features and the subset selection problem," in *Machine Learning: Proceedings of the Eleventh International Conference*. Morgan Kaufmann, 1994, pp. 121–129.

[5] K. Kira and L. A. Rendell, "The feature selection problem: Traditional methods and a new algorithm." in *AAAI*. Cambridge, MA, USA: AAAI Press and MIT Press, 1992, pp. 129–134.

[6] R. Kachouri, K. Djemal, and H. Maaref, "Adaptive feature selection for heterogeneous image databases," in *Second IEEE International Conference on Image Processing Theory, Tools 38; Applications, 10*, K. Djemal and M. Deriche, Eds., Paris, France, 2010.

[7] H. Almuallim and T. G. Dietterich, "Learning with many irrelevant features," in *In Proceedings of the Ninth National Conference on Artificial Intelligence*. AAAI Press, 1991, pp. 547–552.

[8] L. P. Cordella, C. D. Stefano, F. Fontanella, and C. Marrocco, "A feature selection algorithm for handwritten character recognition." in *ICPR'08*, 2008, pp. 1–4.

[9] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, March 2003. [Online]. Available: http://portal.acm.org/citation.cfm?id=944919.944968

[10] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artif. Intell.*, vol. 97, pp. 273–324, December 1997.

[11] Y. Li and L. Guo, "Tcm-knn scheme for network anomaly detection using feature-based optimizations," in *Proceedings of the 2008 ACM symposium on Applied computing*, ser. SAC '08. New York, NY, USA: ACM, 2008, pp. 2103–2109. [Online]. Available: http://doi.acm.org/10.1145/1363686.1364194

[12] F. E. Kitoogo and V. Baryamureeba, "A methodology for feature selection in named entity recognition," *International Journal of Computing and ICT*, pp. 18–26, 2007.

[13] L. S. Oliveira, R. Sabourin, F. Bortolozzi, and C. Y. Suen, "Feature selection using multi-objective genetic algorithms for handwritten digit recognition," in *Proceedings of the 16 th International Conference on Pattern Recognition Volume 1*, ser. ICPR '02. Washington, DC, USA: IEEE Computer Society, 2002.

[14] A. Alamdari, "Variable selection using correlation and single variable classifier methods: Applications," in *Feature Extraction*, ser. Studies in Fuzziness and Soft Computing, 2006, vol. 207, pp. 343–358.

[15] W. Iba and P. Langley, "Induction of one-level decision trees," in *Proceedings of the ninth international workshop on Machine learning*, ser. ML92, San Francisco, CA, USA, 1992, pp. 233–240.

[16] L. Breiman *et al.*, *Classification and Regression Trees*. New York: Chapman and Hall, 1984.

[17] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Proceedings of the Second European Conference on Computational Learning Theory*, London, UK, 1995, pp. 23–37.

[18] C. Dujet and N. Vincent, "Feature selection for classification," *International journal of intelligent system*, vol. 13, pp. 131–156, 1998.

[19] D. Zhang and G. Lu, "Shape based image retrieval using generic fourier descriptors," in *Signal Processing: Image Communication 17*, 2002, pp. 825–848.

[20] S. Tabbone and L. Wendling, "Binary shape normalization using the Radon transform," in *11th International Conference on Discrete Geometry for Computer Imagery - DGCI'2003*, ser. Lecture Notes in Computer Science, vol. 2886. Naples, Italy: Springer, 2003, Colloque avec actes et comité de lecture. Internationale.

[21] H. Kim, J. Kim, D. Sim, and D. Oh, "A modified zernike moment shape descriptor invariant to translation rotation and scale for similarity-based image retrieval," in *ICME00*, 2000, p. MP5.

[22] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, 1998, pp. 2278–2324.