Int J Comput Vis (2009) 83: 164–177
DOI 10.1007/s11263-008-0180-2

A Regularized Framework for Feature Selection in Face Detection and Authentication

Augusto Destrero · Christine De Mol · Francesca Odone · Alessandro Verri

Received: 13 February 2008 / Accepted: 23 September 2008 / Published online: 15 November 2008

Abstract This paper proposes a general framework for selecting features in the computer vision domain—i.e., learning descriptions from data—where the prior knowledge related to the application is confined in the early stages. The main building block is a regularization algorithm based on a penalty term enforcing sparsity. The overall strategy we propose is also effective for training sets of limited size and reaches competitive performances with respect to the state-of-the-art. To show the versatility of the proposed strategy we apply it to both face detection and authentication, implementing two modules of a monitoring system working in real time in our lab. Aside from the choices of the feature dictionary and the training data, which require prior knowledge on the problem, the proposed method is fully automatic. The very good results obtained in different applications speak for the generality and the robustness of the framework.

Keywords Feature selection · Learning from examples · Regularized methods · Lasso regression · Thresholded Landweber · Face detection · Face authentication · Real-time system

A. Destrero · F. Odone (✉) · A. Verri
DISI, Università degli Studi di Genova, Via Dodecaneso 35, 16146 Genova, Italy
e-mail: odone@disi.unige.it

A. Verri
e-mail: verri@disi.unige.it

A. Destrero
Imavis s.r.l., Via Greto di Cornigliano 6R, 16152 Genova, Italy
e-mail: augusto.destrero@imavis.com

C. De Mol
Department of Mathematics and ECARES, Universite Libre de Bruxelles, Campus Plaine CP 217, Boulevard du Triomphe, 1050 Brussels, Belgium
e-mail: Christine.De.Mol@ulb.ac.be

1 Introduction

Many computer vision problems can be cast into a feature selection and classification pipeline for which learning from examples provides an ideal framework. Unfortunately, learning from data is not an automatic procedure and requires, not only the collection of a dataset, but also the design or the choice of data representation methods, dimensionality reduction techniques, appropriate similarity measures or kernel functions, and model selection protocols. Satisfactory learning-based solutions to specific problems have appeared (Viola and Jones 2004; Papageorgiou and Poggio 2000), but a major disadvantage is that they often sacrifice generality to the purpose of increasing the performance. Here we explore the possibility of developing a general framework while confining the domain-dependent knowledge in a well-defined early stage.

Aside from a number of specific contributions related to the proposed optimization algorithm, the most interesting trait of this work is the design, implementation, and assessment of a single data driven system for providing successful solutions to different computer vision problems. While the classification step is quite standard, the feature selection step is the core of the proposed framework. The prior knowledge is limited to the choice of a feature dictionary of large cardinality and the feature selection, fully automatic, is learned from examples. The face domain in computer vision is presumably an ideal testbed for this framework since a relatively large amount of data for a given problem is available and the state-of-the-art is well established. The fact that we obtain very good results for both face detection and authentication with different feature dictionaries speaks for the generality and the effectiveness of the proposed framework.

Before discussing the related state-of-the-art, we briefly describe the application motivating our work: a face authentication system installed in our department. All the data we use to train and validate the modules we developed have been gathered by the system through semi-automatic procedures. The first system requirement is a face detection module able to process the video in *real-time* and return a good localization of the frontal faces appearing in the video. The set-up assumes that the person walks roughly in the direction of the camera optical axis, towards the camera. We assume a face authentication setting, thus the individual is carrying information on his/her identity $\mathcal{I}$. Then, once the face has been localized, it is used as input of the authentication module $\mathcal{I}$. Face authentication is performed on a video portion starting from when the person is close enough to the camera (i.e., his/her face is at least 40×40 pixels). Both face detection, and all the face authentication modules make use of high dimensional dictionaries of local features in order to be tolerant to relative view-point changes and noise. Since our focus is not on the initial representation, we adopt sets of features well established in the literature—rectangle features (Viola and Jones 2004) for face detection and LBP features (Ahonen et al. 2006) for face authentication. Feature selection is a crucial step allowing us to select automatically the local features which are most descriptive for a given problem. All the problems we consider can be cast in a binary classification setting. This is easily explained in the case of face detection, where positives are examples of the class of interest while the negative examples may be images containing other common regions (areas of motions except faces, for instance). In the case of face authentication we train a module for each individual $\mathcal{I}$ to discriminate between $\mathcal{I}$ and others. This time, instead of computing features directly from positive and negative images, we adopt the approach based on modeling intra-personal vs extra-personal features (described, for instance, in Moghaddam and Pentland 1997).

We now address related work on feature selection in machine learning. Sparsity-enforcing algorithms have been studied in depth by the machine learning community as they raise interesting theoretical issues and carry useful properties. Sparse solutions with respect to the training set allow to identify the most representative (or most difficult) prototypes for a given problem. On this respect we mention the well known Support Vector Machines (SVM) (Vapnik 1998) from statistical learning theory and the Relevance Vector Machines (RVM) (Tipping 2001) in the Bayesian framework. Notice that, when adopting kernelized versions of such algorithms, sparsification is applied to a "dictionary" of basis functions induced by the kernel.

Alternative to this approach are methods which sparsify with respect to a dictionary of features describing input data—we refer in this case to *feature selection*, and observe that this sparsification produces a reduction on the data dimensionality. Such a sparsification is usually adopted in conjunction with over-complete or redundant data representations. We mention the Basis Pursuit approach (Chen et al. 1998) also referred to as the Lasso approach (Tibshirani 1996)—the latter derived applying the concept of sparsity for a linear regression model. A sparse code related to receptive field properties is proposed in Olshauser and Field (1997) and gives a biological motivation to feature selection from over-complete feature sets. On the feature selection topic see also the review paper (Guyon and Elisseeff 2003) and the more recent references (Weston et al. 2003; Viola and Jones 2004; Zhu et al. 2004).

More recently, kernelized versions of these sparsity-enforcing methods have been proposed. These approaches, though, do not keep the ability of selecting subsets of features from the original representation since they sparsify with respect to dictionaries of the form

$$\mathcal{D} = \{K(x_1, x), \ldots, K(x_n, x)\}.$$

We mention for instance the Kernelized Lasso (Roth 2004). A similar dictionary was adopted in Girosi (1998) to discuss the equivalence between the Basis Pursuit and SVM, showing that with a dictionary $\mathcal{D}$ as the one above the two approaches have very strong connections: the Basis Pursuit sparsity is enforced on the basis functions, therefore a sparsity on data is obtained.

In this paper we consider a purely linear setting since we are primarily interested in obtaining *automatically* a compact data description (i.e., a small feature subset) from an initially over-complete representation. To this purpose we explore the Lagrangian formulation of the so called Lasso scheme (Tibshirani 1996) for selecting features, and we find a solution to the Lasso by means of a simple iterative algorithm recently proposed (Daubechies et al. 2004). The algorithm was previously applied with success to computational biology (De Mol et al. 2007) and to the face detection problem (Destrero et al. 2007b). The main merits of such a regularized approach are its effectiveness even in the presence of a very small number of data (De Mol et al. 2007) coupled with the fact that it is supported by well-grounded theory. These are the two main reasons that make this approach a possible alternative to other feature selection mechanisms popular in the computer vision community (Freund and Schapire 1995; Friedman et al. 1998; Schapire and Singer 1999; Li and Zhang 2004). In this paper we propose a strategy that allows us to apply the above-mentioned method to computer vision problems, where large datasets are common and features are often strongly correlated because of the spatial correlation of images. Also, we provide a speeding up method leading to a substantial saving of computational time *on the training phase*. Our work is somewhat related to Brown et al. (2004). Similarly to our approach, the authors discuss the use of Lasso regression as a feature selection strategy in the context of computer vision applications. However, they are mostly interested in analysing the stability of the feature selection process and to this purpose they devise an iterative algorithm leading to the optimal parameter locus, i.e., a *complete* set of sparse classifiers depending on the regularization parameter. On this respect they report results showing the stability of the obtained solutions for synthetic data as well as a small set of face images.

We now cover the state-of-the-art in face detection and authentication relevant to our approach. Face-related problems are a mature field in computer vision, and many solutions based on examples have been shown to be effective. Learning from examples dominated the face detection scene since the early 90s and contributed to obtaining promising solutions that deal with view-point changes, rotations, scale and illumination variations (Yang et al. 2002; Osuna et al. 1997; Schneiderman and Kanade 2000). Among such a vast literature, component-based approaches highlighted the fact that local areas are often more descriptive and more appropriate for dealing with occlusions and deformations (Mohan et al. 2001; Ullman et al. 2002). Very popular approaches are the ones derived from Adaboost (see, for instance, Verschae and Ruiz del Solar 2003; Viola and Jones 2004; Li and Zhang 2004; Zhang et al. 2004; Wang and Zhang 2008), that appear to be one of the few attempts of treating feature selection not as an application-oriented problem. As for face recognition, the survey by Zhao et al. (2003) provides a comprehensive reference to the state-of-the-art.

Eigenfaces (Turk and Pentland 1991) are with no doubt one of the most popular holistic approaches to face detection and recognition, and they have been adopted and extended by many authors (Etemad and Chellappa 1997; Belhumeur et al. 1997; Zhao et al. 2003). In Moghaddam and Pentland (1997) the eigenface method is extended to use a probabilistic similarity that models the *intra-personal* variations versus the *extra-personal* variations. The former are related to variations due to illumination, view-point, expression changes within the same individual; the latter refer to the difference between an individual and another. Feature-based approaches include methods for precise localization of specific points (Pentland et al. 1998; Wiskott et al. 1997; Lanzarotti et al. 2006) and methods based on templates, where each area of interest is described by approximately located patches. Possibly the first local approach to face recognition is again due to Pentland and his co-workers (Pentland et al. 1994). Among local approaches, Local Binary Patterns (LBP) (Ahonen et al. 2006) are becoming very popular for their ability to capture descriptive features of a given texture. They have been applied with success in face recognition by many authors (see, for instance, Ahonen et al. 2006; Hadid et al. 2007; Tan and Triggs 2007).

The structure of the paper is as follows. In Sect. 2 we review the iterative algorithm that we adopt and describe the strategies proposed to deal with large problems and to speed up training. Section 3 is devoted to face detection and describes the final 3-stages architecture allowing us to obtain automatically a very small set of features from a large dictionary. Section 4 deals with the face authentication problem. Section 5 is left to a final discussion and a brief account of future developments.

2 Feature Selection for Large Computer Vision Problems

In this section we present and discuss our approach to feature selection. First we describe the adopted optimization algorithm, then the sampled version we developed for dealing with large data sets, and finally we motivate the need of a further classification stage.

2.1 Thresholded Landweber

We start off describing the basic algorithm on which our feature selection method is built upon.[1] We consider the case of a *linear* dependence between input and output data, which means that the problem can be reformulated as the solution of the following linear system of equations:

$$g = Af \tag{1}$$

where $A = \{A_{ij}\}, i = 1, \dots, n; j = 1, \dots, p$ is the $n \times p$ feature matrix obtained representing the training set of n elements with a dictionary of p features. The element A_{ij} is thus the j-th feature of the i-th example. The $n \times 1$ vector $g = (g_1, \dots, g_n)^\top$, instead, contains the output labels. Since we focus on a binary classification setting, we associate to each row of A a label $g_i \in \{-1, 1\}$. $f = (f_1, \dots, f_p)^\top$ is the vector of the unknown weights, where each entry is associated to one feature and intuitively describes the importance of the feature in determining the membership of a given data vector to one of the two classes. Since we are looking for a compact representation of the image for the problem of interest, we perform feature selection looking for a sparse solution where features corresponding to non-zero weights f_i are relevant to model the diversity of the two classes.

In the problems that we consider, typically, the number of features p is much larger than the dimension n of the training set, so that the system is hugely under-determined. Also,

[1]The source code of the algorithm is available for download at http://slipguru.disi.unige.it/.

because of the redundancy of the feature set, we may have to deal with the collinearities between feature vectors that are responsible for severe ill-conditioning. Both difficulties call for some form of regularization and can be obviated by turning problem (1) into a penalized least-squares problem.

Since we are dealing with a feature selection problem we choose the L_1 penalty that automatically enforce the presence of (many) zero weights in the vector f. Thus we consider the following problem, usually referred to as *Lasso regression* (Tibshirani 1996):

$$f_L = \arg\min_f \{|g - Af|_2^2 + 2\tau |f|_1\} \tag{2}$$

where $|f|_1 = \sum_j |f_j|$ is the L_1-norm of f and τ is a regularization parameter regulating the balance between the data misfit and the penalty. In feature selection problems, this parameter also allows to vary the degree of sparsity (number of true zero weights) of the vector f. Notice that the L^1-norm penalty makes the dependence of lasso solutions on g nonlinear. Hence the computation of L^1-norm penalized solutions is more difficult than with L^2-norm penalties. To solve (2) in this paper we adopt a simple iterative strategy:

$$f_L^{(t+1)} = S_\tau[f_L^{(t)} + A^\top(g - Af_L^{(t)})] \quad t = 0, 1, \ldots \tag{3}$$

with arbitrary initial vector $f_L^{(0)}$, where S_τ is the following "soft-thresholder"

$$(S_\tau h)_j = \begin{cases} h_j - \tau \operatorname{sign}(h_j) & \text{if } |h_j| \geq \tau, \\ 0 & \text{otherwise.} \end{cases}$$

In the absence of soft-thresholding ($\tau = 0$) this scheme is known as the Landweber iteration, which converges to the generalized solution (minimum-norm least-squares solution) of (1). The soft-thresholded Landweber scheme (3) has been proved in Daubechies et al. (2004) to converge to a minimizer of (2), provided the norm of the matrix A is renormalized to a value strictly smaller than 1.

Experimental evidence showed that the choice of the initialization vector is not crucial, therefore we always initialize the weight vector f with zeros: $f^{(0)} = \mathbf{0}^\top$. The stopping rule of the iterative process is related to the stability of the solution reached and it is based on comparing the solution obtained at the tth iteration $f^{(t)}$ with the previous one.

2.2 Sampled Version of the Thresholded Landweber Algorithm

The linear problems that we are about to build will be rather large. Assuming that each row of A is associated to a training image and that our training set is made of 1000 positive and 1000 negative examples, then a matrix A of 1 Gb size will be easily obtained—for instance by storing in single precision the rectangle features of 40×40 pixels images. As we will see in Sects. 3 and 4 such size may be quite likely in real-world applications, considering that a 2000 elements training set is common in the computer vision domain.

For this reason applying the iterative algorithm described in (3) directly to the whole matrix may not be feasible on all PCs: the matrix multiplication needs to be implemented carefully so that we do not keep in primary memory the entire matrix A. One possibility is to compute intermediate solutions with multiple accesses to secondary memory.

We implemented a different approach, based on resampling the features set and obtaining many smaller problems, which can be described as follows: we build S feature subsets *each* time extracting with replacement m features from the *original* set of size p ($m \ll p$); we then obtain S smaller linear sub-problems of the type: $A_s f_s = g$ for $s = 1, \ldots, S$, where A_s is a sub-matrix of A containing the columns relative to the features in s; f_s is computed accordingly. As for the choice of the number S of sub-problems and their size we observe that the subset size should be big enough to be descriptive, small enough to handle the matrix easily; thus, we consider subsets with about 10% of the original feature set size. To choose the number of sub-problems S, we rely on the binomial distribution and estimate how many extractions are needed so that each feature is extracted at least 10 times with high probability.

After we build the S sub-problems we look for the S solutions running S iterative methods as in (3). At the end of the process we are left with S overlapping sets of features. The final set is obtained choosing the features that have been selected *each time they appear in the subsets*.

The computational complexity of the iterative algorithm (3) is caused by the two matrix-vector multiplications and thus is $O(np)$ for each iteration and a fixed τ. Since the number of iterations I is not negligible we consider $O(npI)$. In the model selection phase this should be repeated as many times as the number of τ that we evaluate.

It is worth mentioning the fact that the re-sampled version of the method is suitable for parallel computation and would allow for a saving of computation time in inverse relation to the number of processors used. In the case only a single processor is available one should design carefully the model selection phase in order to limit the computational cost of the training phase. In our experiments we consider two methods: (i) to choose τ that includes a fixed number of 0s in the solution (or, equivalently, that selects a given number features) in about I iterations. (ii) to choose τ on the basis of the generalization performance, using cross-validation: we select τ's leading to classification rates below a certain threshold and then choose among them the value providing the smallest number of features. Here, an example $x = (x_1, \ldots, x_p)$ is classified according to the sign of $\sum_{i=1}^p f_i x_i$ where $f = (f_1, \ldots, f_p)$ is the obtained Lasso solution.

2.3 Speeding up Feature Selection

In this section we describe a technique allowing us to obtain an approximate solution of the feature selection process with a considerable saving of computational time.

We start by observing that the computational cost of the iterative algorithm is related to the size of the feature matrix A, and that when dealing with small matrices the computational cost of the method is limited. The speeding up heuristics that we discuss here is suggested by two empirical considerations: (a) for a given τ, a high percentage of the features is discarded in the first iterations; (b) when a feature weight goes to zero it will not change in the next iterations.

At a given step t, after we have computed $f_L^{(t)}$, we may rewrite a smaller linear system to be used in the next iteration. We consider a smaller feature vector f' where we discard the zero entries and, consequently, obtain a smaller data matrix A'. We apply this "shrinking" procedure only when there is a computational advantage in manipulating a smaller matrix, i.e., when the number of zero entries in f is more than 10%. The stopping rule of the iterative procedure is the same as for the original system.

We experimentally observed that, as the matrix become significantly smaller, the solution to the problem is reached from 2 to 6 times faster, depending on the original size of the matrix. To date the use of this procedure is justified by convincing experimental results but no proof of equivalence of the solutions is yet available.

2.4 The Choice of a Classifier

So far we have discussed issues related to feature selection allowing us to obtain an appropriate description from the available data. As for the choice of the most appropriate classification algorithm applied to the selected features, it is well known that the Lasso solution may be used directly as a classifier. In the previous section, for example, we used it for finding the optimal value of the regularization parameter. However, it has been experimentally observed that the Lasso leads to solutions which select effectively the meaningful features but underestimate the corresponding weights due to the soft thresholding step. In the literature a typical approach to correct this bias (and to increase classification performance) is to adopt an alternative classification or regression algorithm on the restricted features set. In Candes and Tao (2007) the bias is corrected by recomputing the weights f of the non-zero features by means of least squares. Here we adopt an SVM classifier, well known for its good generalization ability, on the selected features. In this way the final solution is sparse with respect to both features and data. Since the obtained data representation is not redundant (the features we obtain at the end of the selection process are at most weakly correlated) the choice of a *linear kernel* is appropriate (for more details on this respect, and a comparison with other kernels, see Destrero et al. 2007a) and may be computationally advantageous (for instance if applying Sequential Minimal Optimization (SMO) to solve the quadratic optimization problem, as in SVMLight, Joachims 1999). Apart from algorithmic considerations this method allows us to achieve considerable space-time efficiency at run time, and this suits our requirement of real-time processing.

As pointed out in Guyon and Elisseeff (2003) when an appropriate feature selection strategy is applied and model selection is performed, the quality of the obtained subset of features should be evaluated on an independent test set. If the purpose of feature selection is to obtain a description of the data for a subsequent classification step, it may be the case to evaluate the features quality by estimating the performance of a classifier trained and tuned on data represented with the selected features. This is the strategy we will adopt in Sects. 3 and 4, using the same classifier adopted in the final system, an SVM classifier, starting from which we compute Receiver Operating Characteristic (ROC) curves varying the SVM offset value b.

3 Face Detection

In this section we describe how we specialize the feature selection of Sect. 2 to the case of face detection. We start from an image representation based on the rectangle features (Viola and Jones 2004) as they are widely retained a good starting point for many object detection problems. Rectangle features are meant to be computed over different locations, sizes, and aspect ratios for each image or image patch under consideration. The reference image size that we consider corresponds to the size of our training images: 19×19 pixels. We will therefore compute about 64 000 features per image/patch.

Such a redundant description calls for some feature selection. Notice that, in principle, the redundancy of image features does not compromise generalization performance, but certainly affects the computational efficiency of the classifier. Dealing with redundancy of information means selecting one or few delegates for each group of correlated features to represent the other elements of the group. As for the choice of the delegate, unlike in other application domains (such as micro-array data analysis) in most computer vision problems one could choose a random delegate for the correlated feature sets. Image features are not important *per se* but for the appearance information that they carry, which is resemblant to all other members of their group.

On this respect a remark is in order. The overcomplete features we compute are correlated, not only because of the intrinsic spatial short range correlation of all natural images, but also because of the dependencies related to the class of face images (which contain multiple occurrences of similar patterns at different locations—like eyes and mouth, for

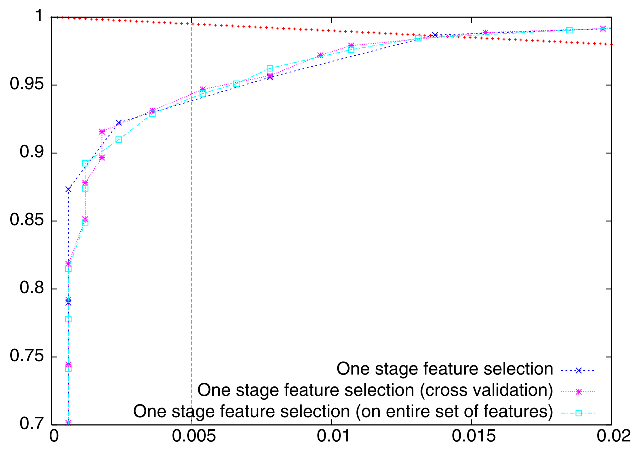


Fig. 1 Generalization performance of a linear SVM with different feature sets: a direct solution of the linear problem and the re-sampling strategy (with the two different parameter choices). The intersection of the ROC with the vertical line gives the hit rate for 0.5% false positives, the intersection with $f(x) = 1 - x$ gives the e.e.r.

example). Correlation due to the representation chosen produces redundant descriptions, while correlation due to the class of interest may carry important information on its peculiarities.

Let us now describe how we apply the iterative algorithm described in Sect. 2 to the selection of face features.

3.1 Feature Selection for Faces

We consider a dataset of 4000 training data, evenly distributed between positive and negative, 2000 validation and 3400 test data. The dataset has been gathered by means of our monitoring system, and manually labeled in positive and negative examples. The linear system that we build for feature selection is rather big, the data matrix A is $4000 \times 64\,000$. As a first experiment, we apply the thresholded Landweber to select a set of meaningful features and test the effectiveness of the obtained solution on the test set. The results obtained are encouraging (equal error rate (e.e.r.) $< 2\%$, see Fig. 1) but the problem (1) has to be solved on a PC equipped with at least 1 Gb RAM, otherwise we run out of memory. This is an appropriate setting to test the re-sampling strategy discussed previously. We consider $S = 200$ sub-problems built each time extracting 10% of the original set of features—with this choice of the number and the size of subproblems, in practice, only 256 features over 64 000 are extracted less than 10 times.

Figure 1 compares the results obtained with and without the re-sampling strategy, and it shows that, not only there is no loss when applying the re-sampling strategy, but there is actually a small gain. As for the choice of parameter τ, considering the number of problems we need to solve, a simple model selection is advisable, as far as it leads to satisfactory results (for more details, the interested reader is referred to Destrero et al. 2007a). We thus tune the parameter τ by setting the number of 0s to be reached within I iterations. The final set of selected features, S_1, contains 4636 elements—less than 10% of the original set size. The selected features are a good synthesis of the original description as confirmed by the classification results on our test set (see Fig. 1 that also shows the results obtained tuning the parameter τ with cross validation) and they maintain all the descriptiveness, representing all meaningful areas of a face. Nonetheless, the number of selected features is still high, higher than the size of the feature set obtained with Adaboost on our same dataset (about 70 features), than the number of principal components (about 400), and also than the intrinsic size of original input data (19×19 pixels). To reduce the number of features we first chose a different τ, so to force a higher number of zeros. Doing so, we noticed that the features descriptiveness decreases. We tested our feature selection setting 99% of zeros to be obtained in about 1000 iterations: we got 345 features that showed to a visual inspection a higher degree of short range correlation than the previous output sets, while many interesting patterns were missing. This loss of information corresponds, as expected, to a drop in performance of about 3%. The solutions to this problem of redundancy that we investigate later in this section consider a possible subsequent stage, aiming at decreasing the size of S_1.

3.2 A Refinement of the Solution

In order to obtain a smaller set of meaningful features we further apply the feature selection procedure to set S_1 looking for a new, sparser, solution. The new data matrix is obtained selecting from A the columns corresponding to S_1, and $f_{S_1}^{(0)}$ is initialized again to a vector of zeros. As for the parameter tuning strategy we choose cross validation to be used in this second stage, since it takes explicitly into account generalization and therefore is more appropriate whenever it is computationally feasible. The selection procedure leaves us with a set S_2 of 247 features. The performance reached is shown in Fig. 2.

To compare our approach with other dimensionality reduction methods we first consider PCA. Figure 3 (left) compares the classification results obtained with PCA on the set S_1 and the ones obtained with a second step of regularized feature selection. Our set of features allows us to obtain higher classification rates and, also, only 247 rectangle features per each test need to be computed, instead than combinations of about 2400 features. Then we consider the Adaboost feature selection proposed in Viola and Jones (2004).[2] Our comparison focuses on the quality of the selected features more than on the classifier: again, we eval-

[2]We used the Intel OpenCV implementation http://www.intel.com/.

uate the goodness of features with respect to their generalization ability, so in both cases we check how good they are on a classification task, for a fixed classifier (SVM). Figure 3 (right) shows a comparison between classification performances obtained with our features and Adaboost features selected on our same set of data. Since in Viola and Jones (2004) feature selection and training are performed on a unique Adaboost loop we verified the fairness of our comparison by including in our plot the *false positive / hit* ratio obtained with a cascade of Adaboost classifiers: they are marked as a black dot. Notice that these results are in line with the curve obtained with Adaboost features and the SVM classifier. The impressive performance obtained by the face detection in Viola and Jones (2004) seems to rely on the use of a very big set of negative examples. At each stage of the cascade the current classifier is trained with new negative examples, while the ones that were correctly classified in the previous stages are discarded.

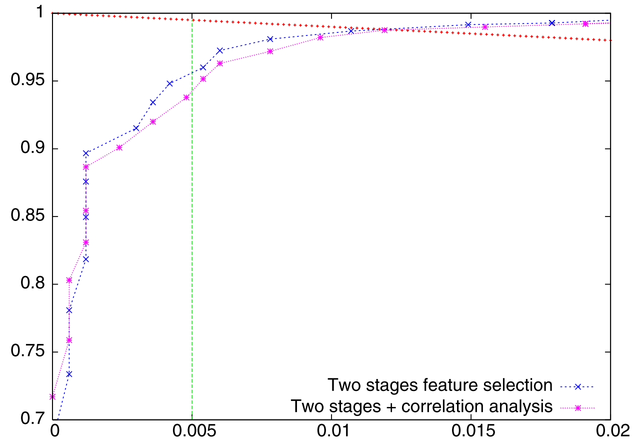


Fig. 2 Evaluation of the features obtained from the second selection stage, with and without a 3rd stage of correlation analysis

In order to obtain a minimal set of features carrying all the information without redundancy, we apply a further selection on the features that survived the previous two stages, based on choosing only one delegate for groups of short range correlated features. Our evaluation is based on the principle of discarding features of the same type, correlated, and spatially close to a feature already included in the final set. We evaluate the correlation between two features using the well known Spearman's correlation test. Using this further analysis as a third stage of our selection procedure we obtain very compact descriptions with little degradation of the results. Figure 4 shows the final set S_3 of 42 features. The classification results on the test set are reported in Fig. 2. Summarizing, Fig. 5 shows the structure of the final 3 stages feature selection protocol. This very same protocol will be applied to the various face authentication modules.

3.3 The Final System

We conclude with an account on the final face detector and the performance obtained.

The features in S_3 are used to build a cascade of small SVMs that is able to process video frames in real time. The cascade of classifiers analyzes an image according to the usual coarse-to-fine approach. Each image patch at a certain location and scale is the input of the classifiers cascade: if the patch fails one test it is immediately discarded; if it passes all tests a face is detected. Each classifier of the cascade is built starting by 3 mutually distant features, training a linear SVM on such features, and adding further features until a target performance is obtained on a validation set. Target performance is chosen so that each classifier will not be likely to miss faces: we set the minimum hit rate to 99.5% and the maximum false positive rate to 50%. Assuming a

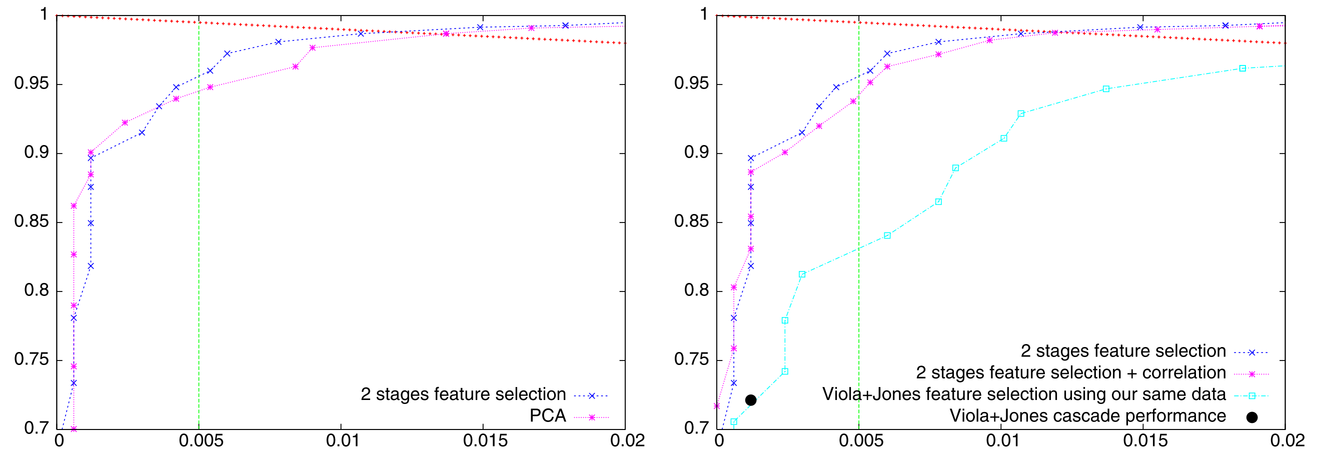


Fig. 3 Comparison of our second selection stage against other dimensionality reduction methods applied to set S_1. *Left*: PCA; *Right*: Adaboost features obtained from our same pool of data (see text)

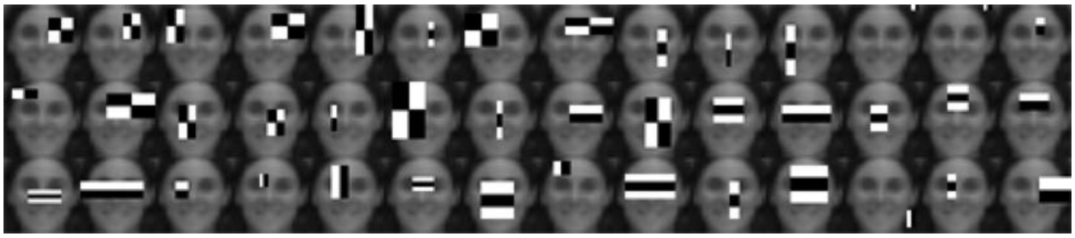
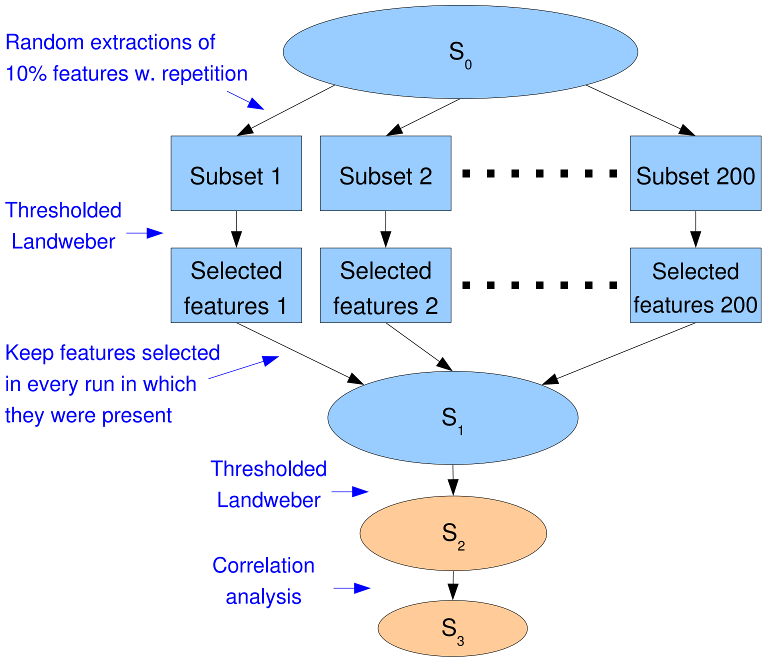


Fig. 4 The 42 features that are left after a third stage of correlation analysis (S_3)

Fig. 5 The structure of the 3 stages feature selection that we adopt



cascade of 10 classifiers, we would get as overall performance: $HR = 0.995^{10} \sim 0.9$ and $FPR = 0.5^{10} \sim 3 \times 10^{-5}$ (Viola and Jones 2004).

Table 1 shows its detection performance as a face detector of our real-time system in two different situations. The first row of the table refers to the results obtained on images acquired in controlled situations (people were asked to walk towards the camera one at a time), while the second row refers to uncontrolled detection: we manually labeled the events occurring in a 5 hours recording of a busy week day; the recording was entirely out of our control and it included changes of the scene, people stopping for unpredictable time, lateral faces. Notice that at run time, the classifiers have been tuned so to minimize the number of false positives. We observe that the amount of data analyzed is huge since, on average, the detector analyzes 20 000 patches per image or frame. Our system running in various environmental conditions can be appreciated on the video available on our website.[3]

[3]The video is available at the url http://slipguru.disi.unige.it/Downloads/multimedia/faces_eyes.mpg.

Table 1 The performance of the face detection system on a controlled set of images (top row) and on a 5 hours unconstrained live video (bottom row)

Test data	False pos rate	Hit rate
test images	0.1%	94%
5 hours live	4×10^{-7}%	76%

The face detector module is associated to an *eye detector* obtained from the same protocol and trained with eye-pairs from the FERET dataset, and again based on the choice of the most representative set of features from the pool of rectangle features. At run time the eye detector is applied to the face detector hits to the purpose of discarding false positives and non frontal faces. A second important use of the eye detector is to automatically register faces to the purpose of authentication (see Fig. 6). As we mentioned previously, the face authentication module will process faces that are "big enough" for recognition, thus we keep only face images whose eye size is 20×40 pixels and automatically discard small faces. The final set of faces is resized to 40×40 pixels.

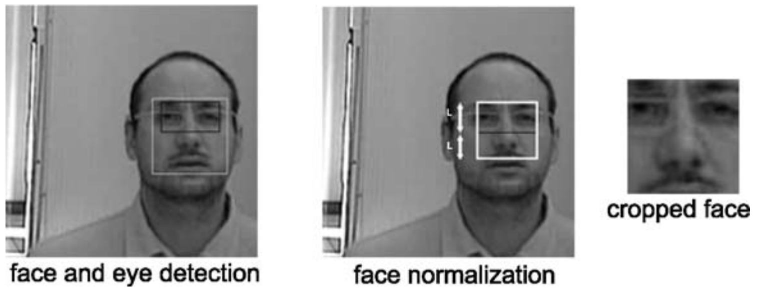


Fig. 6 Preparing the face region, output of face detection, for face authentication: eyes are detected and a smaller face region, whose upper half is the eye region, is cropped

We also considered benchmark datasets. We tested our face detection method on set A and C of the MIT-CMU dataset and on the BioID dataset obtaining consistent results to the ones obtained with our dataset: our method performance is consistently above an Adaboost detector trained with our same dataset. Keeping the false positive rate fixed, the hit rate gain with our approach is on average 8% on the various test sets (Destrero et al. 2007a). If the number of negative examples increases by three order of magnitudes (as in the standard Adaboost setting), the overall performance of Adaboost is superior. This result is consistent with the reported ability of Adaboost of taking advantage of large data sets.

4 Face Authentication

In this section we focus on the *face authentication* (or *validation* or *verification*) problem, closely related to face recognition. The main difference between the two problems is that, in the case of face authentication, the *probe* (test) is composed by a test image (or test sequence) and an associated identity. Therefore the individual to be recognized declares to the system his/her identity and the system has to verify whether he/she is a genuine or an impostor. Similarly to Moghaddam and Pentland (1997), Hadid et al. (2007) we analyze intra-personal and extra-personal variations. We adopt a procedure based on modeling variations of the individual, as opposed to variations of the people "universe". This approach, adopted for instance in Liu et al. (2003), is appropriate for face authentication, and has the advantage of simplifying the *enrollment phase* (i.e., the procedure where a new individual is added to the system database) of a new individual. Again we start off with a high dimensional description, exploiting LBPs in the original formulation described in Ahonen et al. (2006), and then apply feature selection to reduce the number of features.

4.1 Feature Selection for Authentication

The dataset we consider for face authentication is gathered and registered automatically by the face detection module (Sect. 3). In particular, the experiments reported in this paper are obtained from a dataset built from several weeks of acquisition: at the end of two weeks we manually labeled the stored videos (a video is stored if faces are detected in it) and built models for all the individuals that had a rich dataset. From the third week on data were gathered and labeled for testing: individuals that did not previously appear were stored to be used as negative examples (impostors) for all models. In total, 15 individuals were included in the training phase, while a total of 64 individuals were gathered for testing.

Let us briefly describe how we represent each individual $\mathcal{I}$:

- We start off from a set of 40×40 positive images I_p, $p = 1, \dots, P$ of individual $\mathcal{I}$ and a set of negative images I_n $n = 1, \dots, N$ randomly sampled from other individuals.
- For each image I_i, each neighborhood is represented as a LBP. The neighborhood size we consider is obtained choosing 8 sampling points on a circle of radius 2 (Ahonen et al. 2006). Then, for each image, we compute L LBP histograms on rectangular regions of at least 3×3 pixels, on all locations and aspect ratios; thus the description of image I_i is a list of histograms $H_i^1, \dots, H_i^L$. Notice that a feature selection procedure is important, since we obtained $L = 23\,409$ LBP histograms.
- We resort once again to the regularized feature selection of Sect. 2. The linear system is built so to express the intra-personal and extra-personal variation of each histogram. With a similar approach to Hadid et al. (2007) we compute each row of the matrix A as follows: for each pair of images I_A and I_B we compare corresponding LBP histograms (after normalization) using the χ^2 distance. That is, for each pair of input images we obtain a feature vector x_i whose elements are χ^2 distances: $x_i = (\chi^2(I_A^1, I_B^1), \dots, \chi^2(I_A^L, I_B^L))$. We associate a label $g_i \in \{+1, -1\}$, where $g_i = 1$ if both I_A and I_B are positive images, $g_i = -1$ if either I_A or I_B is negative.

For a given set of examples, the number of feature vectors we compute is very high—for a set of positive images of size P we have $\binom{P}{2}$ positive feature vectors and many more negative feature vectors. In order to obtain balanced systems of reasonable size we randomly sample at most 2000 positive and 2000 negative feature vectors and build matrix A. The vector g is composed of the corresponding labels g_i and the vector f of unknowns will weight the importance of each LBP histogram for intra-person and extra-person comparison. Once we have built a system for a given individual, according to the procedure previously described, we select features following the same 3-stages protocol illustrated in Sect. 3. Notice that for each individual we obtain a different set of distinctive features, the ones that best represent his/her peculiarity.

Again, we evaluate the goodness of a feature set in terms of its generalization ability. First, we briefly comment on the appropriateness of the 3-stages strategy for the problem of face authentication. We report similar experiments to the ones shown in Sect. 3 for a randomly chosen individual $\mathcal{I}_A$. After feature selection we train and tune a classifier that should discriminate between images of $\mathcal{I}_A$ and images of impostors. Figure 7 shows a comparison of the results obtained for the 3 stages. Similarly to face detection we notice that the three results are comparable, with a slight performance loss when adding the third stage, which is although responsible of a consistent decrease in the number of features and therefore allows for a very compact representation of each person (see Table 2).

In the case of face authentication, having to deal with a high number of models, that will grow at each new individual inserted in the training phase, we also adopted the speeding up strategy of the feature selection process. In order to assess the reliability of the approach we compared the results obtained with and without speedup for all the 201 (200 for the first stage, 1 for the second) feature selection phases of the model that we used so far. We observed that all the features selected in the various stages except 1 were the same with the two methods. We attribute to numerical approximations the reason for this small difference.

Figure 8 shows the top 5 features extracted for some individuals of our training set. Although the detected features

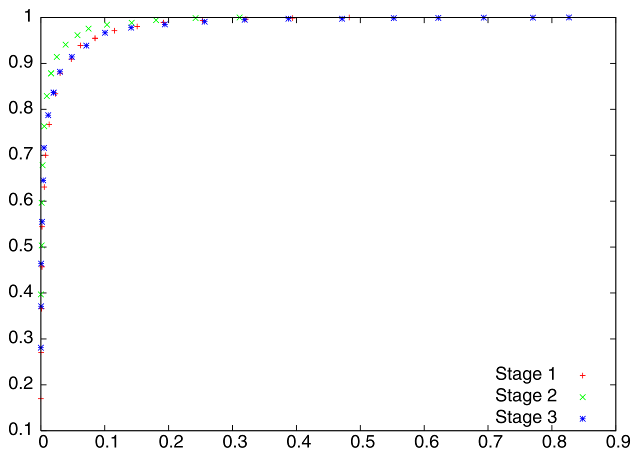


Fig. 7 Evaluation of the features obtained from the three stages, for an individual arbitrarily chosen

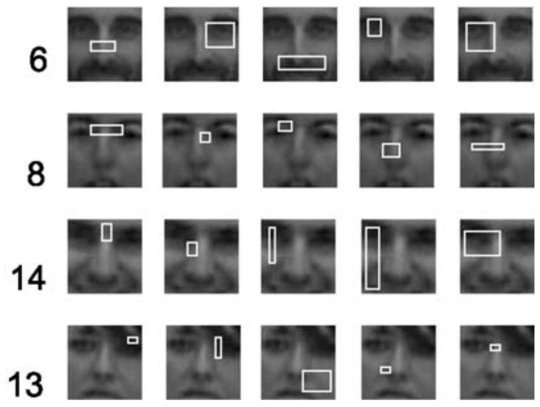


Fig. 8 Top 5 features for some individuals: they often capture distinctive face regions

Table 2 Results by individual: we compare the performance of our method with PCA to the purpose of showing that with the proposed framework we can successfully deal with a complex real world scenario

		LBP		PCA	
Individual	# of features	Precision	Recall	Precision	Recall
1	17	0.92	1.00	0.91	0.98
2	12	1.00	0.86	0.65	0.82
3	19	1.00	0.96	0.78	0.95
4	19	0.99	0.86	0.54	0.88
5	16	0.96	0.90	0.81	0.64
6	13	1.00	1.00	0.78	1.00
7	14	0.95	0.94	0.84	0.92
8	13	0.98	1.00	0.65	0.91
9	25	0.94	0.87	0.67	0.91
10	14	0.97	0.78	0.65	0.61
11	19	0.99	0.98	0.65	0.98
12	20	0.96	0.96	0.52	0.79
13	7	0.91	0.88	0.50	0.28
14	16	1.00	0.99	0.63	0.99
15	23	0.98	0.99	0.73	0.99

do not necessarily reflect elements that a human observer would find meaningful, they are localized in the most distinctive areas of the face for each individual. For person 6, for instance, the beard is spotted. The most distinctive area of person 8 seems to be the top part, the eyes region. Person 14 is mainly characterized by vertical features. The top 2 features of person 13 are localized where she wears a fringe.

4.2 The Final System

Let us now describe briefly the actual testing procedure. After we have selected the appropriate set of features for each individual, we are ready to train and tune a classifier to discriminate between pairs of images of the individual $\mathcal{I}$ under consideration, and pairs of $\mathcal{I}$ with impostors. The rationale of this classifier is that when a probe claims to be $\mathcal{I}$ it will be coupled with all the elements of gallery $\mathcal{I}$, and as many *test pairs* as the size of the gallery will be built. These test pairs will be classified to see whether they represent the same person, $\mathcal{I}$, or not. Thanks to feature selection the classification will be entirely based on the features that are important to $\mathcal{I}$. As a classifier, we consider again linear SVMs, this time without the need for a coarse-to-fine strategy, as we perform only one evaluation per frame.

To test the effectiveness of our authentication we run each classifier on the test set that we gathered, containing about 1700 probes, assuming an equal *a priori* probability of genuine people and impostors (similarly to the evaluation procedure proposed in Messer et al. 2004): we use all positive data available for each classifier, and an equal number of negatives randomly selected. Each test image T fed on a given classifier $\mathcal{I}_i$ produces G_i test pairs. In order to associate a unique output to each probe we compute the percentage of the test pairs generated by it that were classified as positives.

Figure 9 shows the trend of average precision-recall by varying the percentage of positive scores q in the range 10–90%. Table 2 shows the number of selected features per each individual, at the end of the 3-stages procedure, and the authentication results in terms of precision-recall obtained for each individual. We associate to the image a label “genuine” if at least $q = 50\%$ of the intermediate outputs were positives. We compare the results obtained with our local approach based on automatic selection of LBP features, and the one obtained using PCA as a reference method. Here again we compute the “individual” eigenspace. PCA suffers from the registration quality, which being automatic, occasionally fails. Also, as a holistic approach, it fails in case of occlusions or view-point changes, quite common in real environments. Finally, the individual eigenspace may not be enough descriptive if the number of training images is small (as it happens for individual 14).

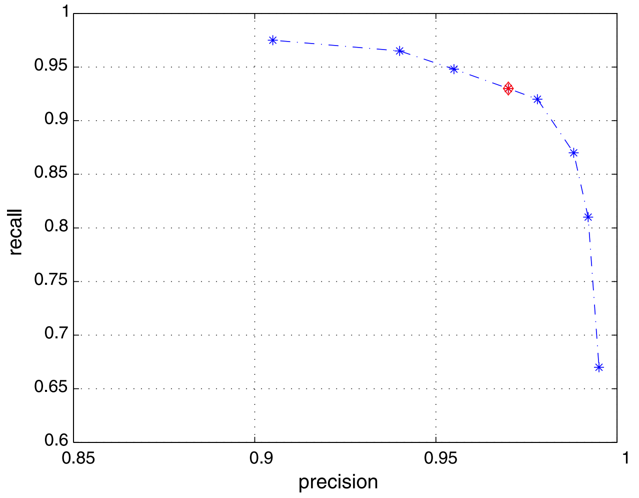


Fig. 9 The trend of precision-recall obtained by varying from 20 to 90 the percentage of positive test data obtained from a single probe image. The *darker spot* indicates the precision-recall at 50% currently used by our system

Table 3 Comparison between average authentication results obtained with automatic feature selection of the LBP features most appropriate for each individual (averages of the results reported in Table 2) and manually set LBP features according to Ahonen et al. (2006)

Automatic LBP selection		Manual selection	
Precision	Recall	Precision	Recall
0.97	0.93	0.93	0.86

A more challenging comparison can be obtained by replacing in the presented pipeline our automatic feature selection with the manual selection of LBP features proposed in Ahonen et al. (2006). The average results on all the classes of Table 2 are reported in Table 3. The considerable gain obtained by adopting our automatic procedure is possibly due to the fact that our approach selects the most meaningful features per each individual, while the manual selection is based on extracting meaningful areas for an average face (eyes, nose, and mouth regions).

We conclude this section by observing that the obtained results are quite promising, especially considering that the entire procedure is fully automatic. When gathering data for face authentication, the labeling was done on a sample frame for each video shot. As a consequence of this, our system actually found “real impostors” (people that were wrongly labeled) in our dataset: they were spotted because they produced unusually bad results compared to other data temporally close to them—Fig. 10 left shows an example frame where the person leaving the scene (in the foreground) enters by mistake in the test set of the person on the back. Figure 10 (right) shows that not all the views extracted by the face detector are ideal for recognition.

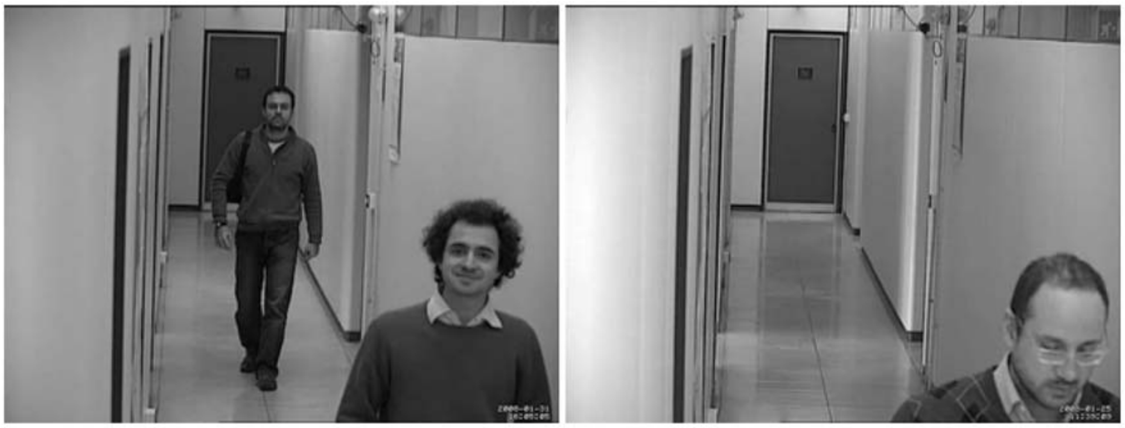


Fig. 10 The test data obtained automatically contains errors (i.e., "real" impostors) and difficult data (see text)

5 Discussion

This paper presented a framework for feature selection based on examples, formulated on the idea of obtaining a general procedure to be used for many applications. The core of the framework is to move all the domain-dependent information in a very early stage, during the preparation of an appropriate under-constrained linear system.

Our work is built around an iterative algorithm, the thresholded Landweber algorithm, implementing the so-called Lasso scheme that produces a sparse solution to a linear problem and can be used as a feature selection procedure. We explored the effectiveness of thresholded Landweber on the computer vision domain taking into account the peculiarity of image features and the availability of possibly large training sets. The final method that we devised includes two main stages: in the *first* stage, for computational reasons, we proposed to find a solution of a big problem by solving a number of smaller optimization problems and returns a smaller but still highly redundant set of features. The *second* stage is based on applying again the selection algorithm to the features resulting from the first stage; the obtained set is, on average, 0.5% the size of the initial representation. A further *third* stage, added for optimizing the classification step coming next, is based on finding a very small set of uncorrelated features that is suitable for real-time processing to the price of a very limited decrease in performance. From the algorithmic view-point we propose two effective solutions to two crucial problems: the need for solving large systems (thanks to the re-sampling strategy we deal with large matrices through intermediate solutions of smaller problems), the request for limiting the computational cost of the training phase coming from real world applications (we propose a speeding up heuristics that allows us perform training faster).

The computational time at training of our approach without speeding up is comparable to other general-purpose feature selection methods (e.g., the Adaboost, Viola and Jones 2004).[4] On top of this, thanks to the speed up heuristics, training may be up to 6 times faster.[5] For what concerns memory requirements, the biggest matrix we consider in the face detection module is in the first stage: a 4000×6400 matrix of real numbers. In all the different variants proposed the memory requirements are of about 100 Mb.

We applied our approach to various stages of a face authentication system under development. We fully exploited the versatility of our feature selection adopting it for two different detection modules (faces and eyes) and for a set of authentication modules (one per each individual enrolled in the system). A full version of the system has been running since the beginning of 2008 and the number of people enrolled is growing as we write.

The processing speed at run time is of about 8 frames per second with a PAL frame format. Notice that this is guaranteed thanks to (i) the cascade procedure (ii) a feature selection approach that allows us to compute only few features (as opposed to other dimensionality reduction methods). The obtained representations are very compact (42 real numbers, i.e. 168 bytes, for face detection and even less for face authentication). This is due to the choice of adopting a purely linear feature selection.

[4]For the case of face detection, with a dataset of 4000 positive and negative examples, on a PC Intel(R) Pentium(R) 4 CPU 3.00 GHz, training our method requires about 14 h, the time is approximately reduced to 5 h if the computation of the first stage is split on 4 similarly performing machines. Feature selection via Adaboost (OpenCV implementation) requires about 12 h.

[5]face detection training in about 3 h on a PC Intel(R) Pentium(R) 4 CPU 3.00 GHz.

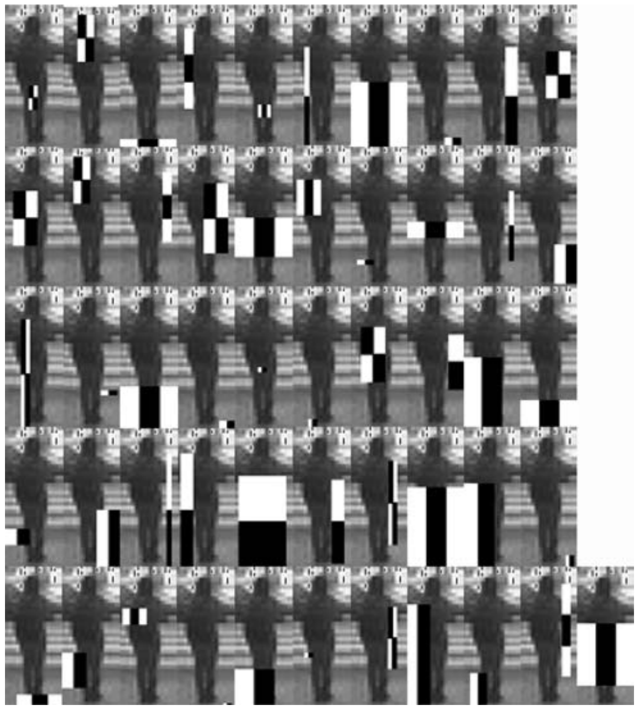


Fig. 11 The 51 features most representative of the pedestrian class automatically extracted by our 3-stages feature selection method

It should be noted that since the face authentication is a module of a more complex video-surveillance system the quality of the signal is low and it also suffers from the presence of artifacts due to video compression. We are working on saving face data at a higher resolution, and this should reflect positively on the obtained results. The current version of our approach does not exploit the fact that our data have a temporal continuity. A future work will be an extension of the face authentication that uses EVLBP features (Hadid et al. 2007). This modification will just reflect in the construction a new feature matrix, while the rest of the process will be unchanged.

As a final remark to this work, we would like to stress the versatility and effectiveness of the proposed framework. We first remark that face authentication relies on the outputs provided by the face and eye detector modules (the eyes detector is essential for automatic face registration). The very good authentication rates speak in favor of the overall accuracy of the various detection stages.

Secondly, we are applying the very same data-driven system to other image classification tasks. Figure 12, for example, shows the results obtained on a pedestrian detection problem. The used dataset combines three different datasets: MIT, USC, and (front and back views from the) Daimler Chrysler dataset.[6] The dataset size is of 1404 images which

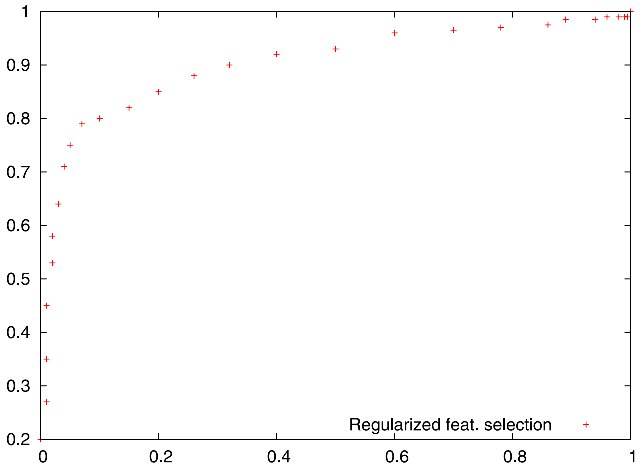


Fig. 12 Performance achieved with a pedestrian detection system automatically built with our data-driven procedure

have been resized to 15×37 pixels. The dataset is sampled to build a training set of 804 positive and 804 negative examples, a validation set of $300 + 300$ examples, and a test set of $300 + 300$ images. We are currently representing images by means of rectangle features, building a dictionary of size 125,605. The final classifier implements a component-based approach to pedestrian detection, vaguely reminiscent of the one proposed in Mohan et al. (2001). Indeed, the results obtained by our system are in line with the ones reported in that paper and approximately equal to the ones obtained using Adaboost. Figure 11 shows the 51 features automatically selected by our system as the best representation for the class "pedestrians".

Acknowledgements The authors are grateful to Ernesto De Vito for many useful discussions and hints, and to the anonymous reviewers for their constructive comments and insights. The work has been partially supported by the FIRB Project LEAP (RBIN04PARL). C. De Mol acknowledges the hospitality of DISI during a sabbatical semester in Genova and the support of the grants ARC 02/07-281 and GOA62(VUB).

References

Ahonen, T., Hadid, A., & Pietikainen, M. (2006). Face description with local binary patterns: application to face recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, *28*(12), 2037–2041.

Belhumeur, P. N., Hespanha, J. P., & Kriegman, D. J. (1997). Eigenfaces versus fisherfaces. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, *19*, 711–720.

Brown, M., Costen, N. P., & Akamatsu, S. (2004). Efficient calculation of the complete optimal classification set. In *Proc. of the 17th ICPR*.

[6]Available for download respectively at http://cbcl.mit.edu/software-datasets/PedestrianData.html, http://iris.usc.edu/~bowu/DatasetWebpage/dataset.html, http://www2.science.uva.nl/sites/PedestrianClass/.

Candes, E., & Tao, T. (2007). The Dantzig selector: statistical estimation when p is much larger than n. *The Annals of Statistics*, *35*(6), 2313–2351.

Chen, S. S., Donoho, D., & Saunders, M. A. (1998). Atomic decomposition by basis pursuit. *SIAM Journal of Scientific Computing*, *20*(1), 33–61.

Daubechies, I., Defrise, M., & De Mol, C. (2004). An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, *57*, 1413–1457.

De Mol, C., Mosci, S., Traskine, M. S., & Verri, A. (2007). *A regularized method for selecting nested groups of relevant genes from microarray data* (Technical Report DISI-TR-07-04). Dipartimento di informatica e scienze dell'informazione, Universita' di Genova. Available at http://arxiv.org/abs/0809.1777.

Destrero, A., De Mol, C., Odone, F., & Verri, A. (2007a, to appear). A sparsity-enforcing method for learning face features. In *IEEE Transactions on Image Processing*.

Destrero, A., De Mol, C., Odone, F., & Verri, A. (2007b). A regularized approach to feature selection for face detection. In Yagi, Y. et al. (Eds.), *LNCS: Vol. 4844. Proc. of the Asian conference on computer vision, ACCV* (pp. 881–890). Berlin: Springer.

Etemad, K., & Chellappa, R. (1997). Discriminant analysis for recognition of human face images. *Journal of the Optical Society of America A*, *14*, 1724–1733.

Freund, Y., & Schapire, R. E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory* (pp. 23–37).

Friedman, J., Hastie, T., & Tibshirani, R. (1998). Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, *28*(2), 337–407

Girosi, F. (1998). An equivalence between sparse approximation and support vector machines. *Neural Computation*, *10*(6).

Guyon, I., & Elisseeff, E. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, *3*, 1157–1182.

Hadid, A., Pietikäinen, M., & Li, S. Z. (2007). Learning personal specific facial dynamics for face recognition from videos. In *AMFG07* (pp. 1–15).

Joachims, T. (1999). Making large-scale SVM learning practical. In Schölkopf, B., Burges, C., & Smola, A. (Eds.), *Advances in kernel methods—support vector learning*. Cambridge: MIT Press.

Lanzarotti, R., Arca, S., & Campadelli, P. (2006). A face recognition system based on automatically determined facial fiducial points. *Pattern recognition*, *39*(3), 432–443.

Li, S. Z., & Zhang, Z. Q. (2004). FloatBoost learning and statistical face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(9), 1112–1123.

Liu, X., Chen, T., & Vijaya Kumar, B. V. K. (2003). On modeling variations for face authentication. *Pattern Recognition*, *36*(2), 313–328.

Messer, K., Kittler, J., Sadeghi, M., Hamouz, M., Kostyn, A., Marcel, S., Bengio, S., Cardinaux, F., Sanderson, C., Poh, N., Rodriguez, Y., Kryszczuk, K., Czyz, J., Vandendorpe, L., Ng, J., Cheung, H., & Tang, B. (2004). Face authentication competition on the banca database. In *LNCS: Vol. 3072. Biometric authentication*. Springer: Berlin.

Moghaddam, B., & Pentland, A. (1997). Probabilistic visual learning for object representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*, 696–710.

Mohan, A., Papageorgiou, C., & Poggio, T. (2001). Example-based object detection in images by components. *IEEE Transactions on PAMI*, *23*(4), 349–361.

Olshauser, B. A., & Field, D. J. (1997). Sparse coding with an overcomplete basis set: a strategy employed by V1? *Vision Research*, *37*(23), 3311–3325.

Osuna, E., Freund, R., & Girosi, F. (1997). Training support vector machines: an application to face detection. In *Proceedings IEEE int. conference on computer vision and pattern recognition (CVPR)*.

Papageorgiou, C., & Poggio, T. (2000). A trainable system for object detection. *International Journal of Computer Vision*, *38*(1), 15–33.

Pentland, A., Moghaddam, B., & Starner, T. (1994). View-based and modular eigenspaces for face recognition. In *IEEE int. conf. on computer vision and pattern recognition (CVPR)* (pp. 84–91).

Pentland, A., Moghaddam, B., & Starner, T. (1998). Estimation of eye, eyebrow and nose features in videophone sequences. In *International workshop on very low bitrate video coding (VLBV 98)* (pp. 101–104).

Roth, V. (2004). The generalized lasso. *IEEE Transactions on Neural Networks*, *15*(1), 16–28.

Schapire, R. E., & Singer, Y. (1999). Improved boosting using confidence-rated predictions. *Machine Learning*, *37*(3), 297–336.

Schneiderman, H., & Kanade, T. (2000). A statistical method for 3D object detection applied to faces and cars. In *International conference on computer vision*.

Tan, X., & Triggs, B. (2007). Enhanced local texture feature sets for face recognition under difficult lighting conditions. In *AMFG* (pp. 168–182).

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of Royal Statistical Society B*, *58*(1), 267–288.

Tipping, M. (2001). Sparse Bayesian learning and the relevant vector machine. *Journal of Machine Learning Research*, *1*, 211–244

Turk, M. A., & Pentland, A. P. (1991). Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, *3*(1), 71–86.

Ullman, S., Vidal-Naquet, M., & Sali, E. (2002). Visual features of intermediate complexity and their use in classification. *Nature Neuroscience*, *5*(7), 1–6.

Vapnik, V. N. (1998). *Statistical learning theory*. New York: Wiley.

Verschae, R., & Ruiz del Solar, J. (2003). A hybrid face detector based on an asymmetrical Adaboost cascade detector and a wavelet-Bayesian-detector. In *Lecture notes in computer science: Vol. 2686. Computational methods in neural modeling*. Berlin: Springer.

Viola, P., & Jones, M. J. (2004). Robust real-time face detection. *International Journal on Computer Vision*, *57*(2), 137–154.

Wang, H., Li, P., & Zhang, T. (2008). Histogram feature-based fisher linear discriminant for face detection. *Neural Computing and Applications*, *17*(1), 49–58.

Weston, J., Elisseeff, A., Scholkopf, B., & Tipping, M. (2003). The use of zero-norm with linear models and kernel methods. *Journal of Machine Learning Research*, *3*, 1439–1461

Wiskott, L., Fellous, J., Kuiger, N., & von der Malsburg, C. (1997). Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*, 775–779.

Yang, M.-H., Kriegman, D. J., & Ahuja, N. (2002). Detecting faces in images: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(1), 34–58.

Zhang, D., Li, S. Z., & Gatica-Perez, D. (2004). Real-time face detection using boosting in hierarchical feature spaces. In *Proc. of ICPR*.

Zhao, W., Chellappa, R., Rosenfeld, A., & Phillips, P. J. (2003). Face recognition: A literature survey. *ACM computing surveys* (pp. 399–458).

Zhu, J., Rosset, S., Hastie, T., & Tibshirani, R. (2004). 1-norm support vector machines. In *Advances in neural information processing systems 16*. MIT Press: Cambridge.