IDIAP RESEARCH REPORT

Face Verification using MLP and SVM

Fabien Cardinaux Sébastien Marcel

IDIAP–RR 02-21

May 2002

SUBMITTED FOR PUBLICATION

Dalle Molle Institute
for Perceptual Artificial
Intelligence • P.O.Box 592 •
Martigny • Valais • Switzerland

phone +41 – 27 – 721 77 11
fax +41 – 27 – 721 77 12
e-mail secretariat@idiap.ch
internet http://www.idiap.ch

IDIAP Research Report 02-21

Face Verification using MLP and SVM

Fabien Cardinaux Sébastien Marcel

May 2002

submitted for publication

Abstract. The performance of machine learning algorithms has steadily improved over the past few years, such as MLP or more recently SVM. In this paper, we compare two successful discriminant machine learning algorithms apply to the problem of face verification: MLP and SVM. These two algorithms are tested on a benchmark database, namely XM2VTS. Results show that a MLP is better than a SVM on this particular task.

1 Introduction

Identity verification is a general task that has many real-life applications such as access control, transaction authentication (in telephone banking or remote credit card purchases for instance), voice mail, or secure teleworking.

The goal of an *automatic identity verification system* is to either accept or reject the identity claim made by a given person. Biometric identity verification systems are based on the characteristics of a person, such as its face, fingerprint or signature. A good introduction to identity verification can be found in [1]. Identity verification using face information is a challenging research area that was very active recently, mainly because of its natural and non-intrusive interaction with the authentication system.

In this paper, we investigate the use of Multi-Layer Perceptrons and Support Vector Machines to train face verification systems. The paper is structured as follows. In Section 2 we introduce the reader to the problem of identity verification, based on face image (face verification). Then, in section 3 we present the models used. In section 4 we compare these face verification systems on the well-known benchmark database XM2VTS using its associated Lausanne protocol. Finally, in Section 5, we analyze the results and conclude.

2 Face Verification

An identity verification system has to deal with two kinds of events: either the person claiming a given identity is the one who he claims to be (in which case, he is called a *client*), or he is not (in which case, he is called an *impostor*). Moreover, the system may generally take two decisions: either *accept* the *client* or *reject* him and decide he is an *impostor*.

The classical face verification process can be decomposed into several steps, namely *image acquisition* (grab the images, from a camera or a VCR, in color or gray levels), *image processing* (apply filtering algorithms in order to enhance important features and to reduce the noise), *face detection* (detect and localize an eventual face in a given image) and finally *face verification* itself, which consists in verifying if the given face corresponds to the claimed identity of the client.

In this paper, we assume (as it is often done in comparable studies, but nonetheless incorrectly) that the detection step has been performed perfectly and we thus concentrate on the last step, namely the face verification step. A good survey on the different methods used in face verification can be found in [2].

3 The Proposed Approach

3.1 Face features

In face verification, we are interested in particular objects, namely faces. The representation used to code input images in most state-of-the-art methods are often based on gray-scale face image. In our approach we use an additional feature to the face image: the skin color. This additional feature improve face verification results [3].

The face is cropped and the extracted sub-image is downsized to a 30x40 image. After enhancement and smoothing, the face image becomes a feature vector of dimension 1200. The objective of image enhancement is to modify the contrast of the image in order to enhance important features. On the other hand, smoothing is a simple algorithm which reduces the noise in the image (after image enhancement for example) by applying a Gaussian to the whole image. In addition to the gray scale feature, we use a feature vector of dimension 96 representing the skin color [3]. Then, the final input of the face verification system (Fig. 1) is a vector of dimension 1296.

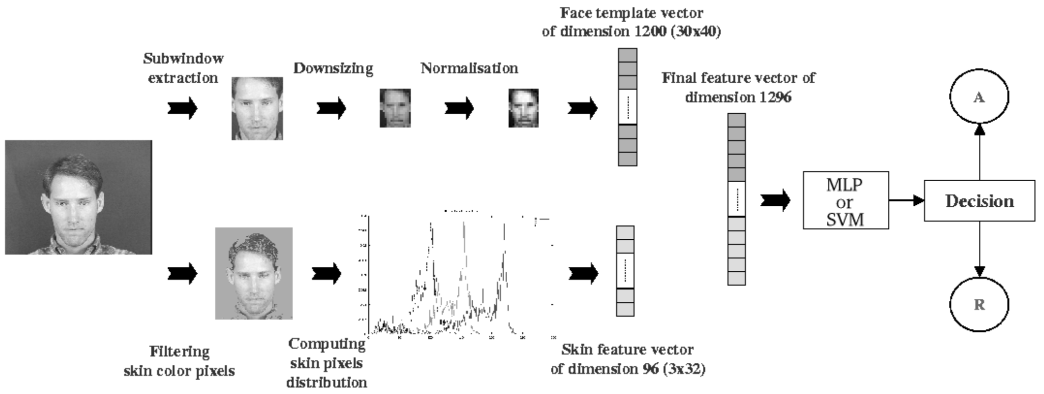
Face template vector
of dimension 1200 (30x40)
Subwindow extraction
Downsizing
Normalisation
Final feature vector of dimension 1296
A
MLP or SVM
Decision
R
Filtering skin color pixels
Computing skin pixels distribution
Skin feature vector of dimension 96 (3x32)

FIG. 1 – *A Face verification system using the image of the face and its skin color*

3.2 Face models

We propose to use and to compare two machine learning algorithms for face verification: Multi-Layer Perceptrons (MLPs) and Support Vector Machines (SVMs). For each client, a model (MLP or SVM) is trained to classify an input to be either the given client or not. The input of the model is a feature vector corresponding to the face image. The output of the model is either 1 (if the input corresponds to the client) or -1 (if the input corresponds to an impostor). The model is trained using both client images and impostor images, often taken to be the images corresponding to other available clients. In the present study, we used the other 199 clients of the XM2VTS database (see next section).

Finally, the decision to accept or reject a client access depends on the score obtained by the corresponding model which could be either above (accept) or under (reject) a given threshold, chosen on a separate validation set to optimize a given criterion.

3.3 Introduction to MLPs and SVMs

MLPs and SVMs are learning machines used in many classification problems. A good introduction to machine learning algorithms can be found in [4, 5]. We will assume that we have access to a training dataset of l pairs $(\mathbf{x}_i, y_i)$ where $\mathbf{x}_i$ is a vector containing the pattern, while y_i is the class of the corresponding pattern often coded respectively as 1 and -1.

3.3.1 Multi-Layer Perceptron

A MLP is a particular architecture of artificial neural networks [4, 5], composed of layers of non-linear but differentiable parametric functions. For instance, the output $\hat{y}$ of a 1-hidden-layer MLP can be written mathematically as follows

$$\hat{y} = b + \mathbf{w} \cdot \tanh\left(\mathbf{a} + \mathbf{x} \cdot \mathbf{V}\right) \tag{1}$$

where the estimated output $\hat{y}$ is a function of the input vector $\mathbf{x}$, and the parameters $\{b, \mathbf{w}, \mathbf{a}, \mathbf{V}\}$. In this notation, the non-linear function tanh() returns a vector which size is equal to the number of hidden units of the MLP, which controls its capacity and should thus be chosen carefully, by cross-validation for instance.

An MLP can be trained by gradient descent using the backpropagation algorithm [10] to optimize any derivable criterion, such as the *mean squared error* (MSE):

$$\text{MSE} = \frac{1}{l} \sum_{i=1}^{l} (y_i - \hat{y}_i)^2 \ . \tag{2}$$

3.3.2 Support Vector Machine

SVMs has been introduced by V. Vapnik [9].The SVM algorithm constructs a separating hypersurface in the input space. It acts as follows:

- maps the input space into a higher dimensional feature space through some nonlinear mapping chosen *a priori* (kernel);
- constructs the maximal margin hyperplane in this feature space (MMH); the MMH maximizes the distance of the closest vectors belonging to the different classes to the hyperplane.

The resulting function is of the form:

$$\hat{y} = sign\left(\sum_{i=1}^{l} y_i \alpha_i K(\mathbf{x},\mathbf{x}_i) + b\right) \tag{3}$$

where $\mathbf{x}$ is the input vector of a example to test, $\hat{y}$ is the decision of the model (accept if $\hat{y}$ is positive, reject otherwise), $\mathbf{x}_i$ is the input vector of the i^{th} training example, l is the number of training examples, the α_i and b are the parameters of the model, and $K(\mathbf{x},\mathbf{x}_i)$ is a kernel function that can have different forms, such as:

$$K(\mathbf{x},\mathbf{x}_i) = (\mathbf{x}^t \ \mathbf{x}_i + 1)^d \tag{4}$$

which leads to a Polynomial SVM with parameter d, or

$$K(\mathbf{x},\mathbf{x}_i) = \exp\left(\frac{-\|\mathbf{x} - \mathbf{x}_i\|^2}{\sigma^2}\right) \tag{5}$$

which leads to a Radial Basis Function (RBF) SVM with parameter σ. Either d or σ must be selected using methods such as cross-validation.

In order to train such SVMs, one needs to solve the following quadratic optimization problem: find the parameter vector $\alpha = \{\alpha_1,\alpha_2,\ldots,\alpha_l\}$ that maximize the objective function

$$Q(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j) \tag{6}$$

subject to the constraints

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \tag{7}$$

and

$$0 \le \alpha_i \le C \ \forall i. \tag{8}$$

It is important to note that the training complexity of SVMs is quadratic on the number l of examples, which makes the use of SVMs for large datasets difficult. Note however that in the resulting solution (3), most α_i are equal to 0, and the examples with non-zero α_i are called *support vectors*.

4 Experimental Results

In this section, we present an experimental[1] comparison between MLPs and SVMs. This comparison has been done using the multi-modal XM2VTS database, using its associated experimental protocol, the *Lausanne Protocol* [6].

[1] The machine learning library used for all experiments is Torch http://www.torch.ch.

4.1 The Database and the Protocol

The XM2VTS database contains synchronized image and speech data recorded on 295 subjects during four sessions taken at one month intervals. On each session, two recordings were made, each consisting of a speech shot and a head rotation shot.

The database was divided into three sets: a training set, an evaluation set, and a test set. The training set was used to build client models, while the evaluation set was used to compute the decision (by estimating thresholds for instance, or parameters of a fusion algorithm). Finally, the test set was used only to estimate the performance of the two different features.

The 295 subjects were divided into a set of 200 clients, 25 evaluation impostors, and 70 test impostors. Two different evaluation configurations were defined. They differ in the distribution of client training and client evaluation data. Both the training client and evaluation client data were drawn from the same recording sessions for Configuration I which might lead to biased estimation on the evaluation set and hence poor performance on the test set. For Configuration II on the other hand, the evaluation client and test client sets are drawn from different recording sessions which might lead to more realistic results. For each client, there is 4 training shot.

The system may make two types of errors: *false acceptances* (FA), when the system accepts an *impostor*, and *false rejections* (FR), when the system rejects a *client*. In order to be independent on the specific dataset distribution, the performance of the system is often measured in terms of these two different errors, as follows:

$$\text{FAR} = \frac{\text{number of FAs}}{\text{number of impostor accesses}} , \tag{9}$$

$$\text{FRR} = \frac{\text{number of FRs}}{\text{number of client accesses}} . \tag{10}$$

A unique measure often used combines these two ratios into the so-called *Half Total Error Rate* (HTER) as follows:

$$\text{HTER} = \frac{\text{FAR} + \text{FRR}}{2} . \tag{11}$$

Most verification systems output a score for each access. Selecting a threshold over which scores are considered genuine clients instead of impostors can greatly modify the relative performance of FAR and FRR. A typical threshold chosen is the one that reaches the *Equal Error Rate* (EER) where FAR=FRR on a separate validation set.

4.2 Comparative Results

We have compared a face verification system using MLPs and SVMs. The Configuration I of the **Lausanne Protocol** is chosen for all experiments

4.2.1 Training set

For each client model, the training database is composed of a client training set (3 images for each client) and an impostor training set. Again, the client training set is enlarged by shifting (8 directions and 4 pixel shifts), scaling (2 scales) and mirroring the original face bounding box.

Hence, the client training set contains 990 patterns ($3 * P$). On the other hand, the impostor training set contains 1194 patterns. (the mirrored 3 original patterns of each of the 199 other clients).

4.2.2 Models

All training sets are used to train the 200 models (one MLP or SVM for each client). Note than the chosen architecture for MLPs has 1 hidden layer with 90 hidden units. The chosen kernel for SVMs is a gaussian kernel ($\sigma = 11$).

4.2.3 Results

The trained model is used on the evaluation set to evaluate the global threshold that optimized the EER. Results on evaluation set with the optimized threshold are shown in Table 1. This threshold is then used with the same trained model on the test set to compute the HTER. Results are shown in Table 2. This table provides the FAR, FRR and HTER on the test set, both for MLPs and for SVMs. These results show that face verification using MLPs gives better results than using SVMs.

Features	FAR	FRR	HTER
MLP	1.67	1.67	**1.67**
SVM	1.70	1.67	**1.69**

Tab. 1 – *Comparative results of MLPs and SVMs on the Evaluation set*

Features	FAR	FRR	HTER
MLP	1.75	2.00	**1.87**
SVM	1.58	1.84	2.54

Tab. 2 – *Comparative results of MLPs and SVMs on the Test set*

These results are competitive when compared to recent results published on the same database. In [7] for instance, the best face HTER (with global thresholds) was 1.5 on the same data using LDA [8] and 61x57 face images from all the XM2VTS training set, i.e more training data and images 3 times bigger than proposed in this paper.

5 Conclusion

MLPs and SVMs have been applied to many classification problems, generally yielding good perfomance compared to other algorithms. In this paper, we have compared these two machine learning algorithms on face verification. This experimental comparison have been carried out using the XM2VTS benchmark database. Results have shown that the two algorithms are competitive when compared to recent results published on the same database. They have shown also that a MLP outperform a SVM on this particular task. Indeed, HTER obtained using MLPs and SVMs for the evaluation set are equivalent. However on the test set, the HTER obtained by MLPs is quite better than HTER obtained by SVMs. This experimental comparison show that MLPs have a better generalisation performance than SVMs in face verification tasks.

Acknowledgments The authors wish thank S. Bengio for fruitful discussions and collaboration.

Références

[1] P. Verlinde, G. Chollet and M. Acheroy, *Multi-modal identity verification using expert fusion,* **Information Fusion**, vol. 1, pp. 17–33, 2000.

[2] J. Zhang, Y. Yan and M. Lades, *Face recognition: Eigenfaces, elastic matching, and neural nets,* in **Proceedings of IEEE**, 1997, vol. 85.

[3] S. Marcel and S. Bengio, *Improving Face Verification using Skin Color Information,* in **Proceedings of the 16th ICPR (to appear)**, IEEE Computer Society Press, 2002.

[4] C. Bishop, **Neural Networks for Pattern Recognition**, Clarendon Press, Oxford, 1995.

[5] S. Haykin, **Neural Networks, a Comprehensive Foundation, second edition**, Prentice Hall, 1999.

[6] J. Lüttin and G. Maître, *Evaluation protocol for the extended M2VTS database (XM2VTSDB),* Techn. Report RR-21, **IDIAP**, 1998.

[7] J. Matas *et. al,Comparison of face verification results on the XM2VTS database,* in A. Sanfeliu, J. J. Villanueva, M. Vanrell, R. Alqueraz, J. Crowley and Y. Shirai (eds.), **Proceedings of the 15th ICPR**, IEEE Computer Society Press, 2000, vol. 4, pp. 858–863.

[8] Y. Li, J. Kittler and J. Matas, *On Matching Scores of LDA-based Face Verification,* in Pridmore and Elliman (eds.), **Proceedings of the British Machine Vision Conference BMVC2000**, British Machine Vision Association, 2000.

[9] V. Vapnik, *The Nature of Statistical Learning Theory,* Springer-Verlag, New York, 1995.

[10] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. *Learning internal representations by error propagation.* In Rumelhart and McClelland, editors, **Parallel Distributed Processing, volume 1.** MIT Press, Cambridge, MA., 1986.