Available online at www.sciencedirect.com

SCIENCE DIRECT®

Pattern Recognition Letters 25 (2004) 1107–1113

Pattern Recognition Letters

www.elsevier.com/locate/patrec

Pattern recognition using higher-order local autocorrelation coefficients

Vlad Popovici [*], Jean-Philippe Thiran

Signal Processing Institute, Swiss Federal Institute of Technology (EPFL), CH-1015 Lausanne, Switzerland

Received 15 July 2003; received in revised form 9 February 2004
Available online 15 April 2004

Abstract

The autocorrelations have been previously used as features for 1D or 2D signal classification in a wide range of applications, like texture classification, face detection and recognition, EEG signal classification, and so on. However, in almost all the cases, the high computational costs have hampered the extension to higher orders (more than the second order). In this paper we present an effective method for using higher order autocorrelation functions for pattern recognition. We will show that while the autocorrelation feature vectors (described below) are elements of a high dimensional space, one may avoid their explicit computation when the method employed can be expressed in terms of inner products of input vectors. Different typical scenarios of using the autocorrelations will be presented and we will show that the order of autocorrelations is no longer an obstacle.
© 2004 Elsevier B.V. All rights reserved.

1. Introduction

Usually, in the framework of statistical pattern recognition, one pattern can be viewed as a function of time and/or spatial coordinates. Moreover, in most cases the class membership does not change as the pattern is translated or scaled. In such situations we would like to design a classifier that is invariant to a given class of affine transformations. One approach consists in transforming each pattern through a function which would induce this invariancy: if we consider each pattern as a point in a vector space, we wish to map all points corresponding to translated (and/or scaled) versions of one pattern in a single point. In addition, patterns which differ in other ways should map into distinct points, and in some sense, patterns which are similar should map into points that are close together.

As it will be shown, the autocorrelation functions possess the uniqueness property for even orders (McLaughlin and Raviv, 1968) and they are translation invariant. The autocorrelations have been used in a wide range of applications: character recognition (McLaughlin and Raviv, 1968), geo-spatial data mining (Chawla et al., 2000), affine-invariant texture classification (Chetverikov and Foldvari, 2000), time series classification (Keogh and Pazzani, 1998) and face detection and recognition (Popovici and Thiran, 2001; Keogh

[*] Corresponding author. Tel.: +41-21-6935646; fax: +41-21-6937600.
E-mail addresses: vlad.popovici@epfl.ch (V. Popovici), jp.thiran@epfl.ch (J.-P. Thiran).

0167-8655/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2004.03.007

and Pazzani, 1998; Goudail et al., 1996). However, in most cases, the applicability of the autocorrelations has been limited to first or second order, due to high computational costs. An interesting approach is presented in (Kreutz et al., 1996) where the authors use the autocorrelations up to the third order and obtain a scale invariant classification by integrating over different scales. They succeed to reduce the number of computations by using the shift-invariant property of the autocorrelations and by a priori determining the lags for which the autocorrelations are not equivalent. The same approach is taken in (Kurita et al., 1998; Hotta et al., 1998), but it has the disadvantage of not being easily generalizable for higher orders or larger local domains.

Our interest is in finding an approach which scales well with the increase of the autocorrelation order, allowing us to generalize the use of autocorrelations. We will present the properties of the autocorrelation functions and will explore different approaches for autocorrelation-based pattern recognition. The paper is organized as follows: Section 2 discusses the properties of the autocorrelation function and sets the background for the subsequent sections, where two possible scenarios of autocorrelation-based pattern recognition are presented. In Section 3 we present a PCA-based technique and we show how one can perform the PCA decomposition in the autocorrelation space. Next section describes the use of autocorrelation features in the general framework of kernel-based techniques for pattern recognition. These two techniques are exploited in Section 5 where we report some experiments performed using autocorrelation features. Finally, in Section 6 we draw some conclusions and indicate some possible directions for further investigation.

2. Generalized autocorrelation functions

2.1. Definition and properties

Definition 1 (*Higher-order autocorrelation*). Let $\psi : D \subseteq \mathbb{R}^m \to \mathbb{R}$ be a real-valued function. The n-order autocorrelation function associated with function ψ is defined as follows:

$$r_\psi^{(n)}(\tau_1,\ldots,\tau_n) \triangleq \int \psi(t) \prod_{k=1}^{n} \psi(t+\tau_k)\,\mathrm{d}t \tag{1}$$

It is easy to see that $r_\psi^{(n)}$ is shift-invariant, in the sense that $\psi(t)$ and $\psi(t+\tau)$ have the same nth order autocorrelation.

On the other hand, for two functions ψ_1 and ψ_2 it can be proven (McLaughlin and Raviv, 1968) that the second order (and higher even order) autocorrelation functions are equal only if $\psi_1(t) = \psi_2(t+\tau)$, meaning that the two patterns have the same representation in autocorrelation space if ψ_2 is a shifted version of ψ_1. This also means that ψ_1 can be recovered from $r_{\psi_1}^{(2k)}$ except for an unknown translation τ. Generally, in the case of pattern recognition, this is a valuable property.

Another important property (McLaughlin and Raviv, 1968) of the autocorrelation functions, which will be fully exploited later on, is given by the following.

Lemma 2. *Let ψ_1 and ψ_2 be two real-valued functions defined over the same domain. Then, the inner product of their corresponding autocorrelation functions $r_{\psi_1}^{(n)}$ and $r_{\psi_2}^{(n)}$ is given by*

$$\langle r_{\psi_1}^{(n)}, r_{\psi_2}^{(n)} \rangle = \int \left\{ \int \psi_1(v)\psi_2(v+s)\,\mathrm{d}v \right\}^{n+1} \mathrm{d}s \tag{2}$$

Proof. We successively have

$$\begin{aligned}
&\langle r_{\psi_1}^{(n)}, r_{\psi_2}^{(n)} \rangle \\
&= \int \cdots \int r_{\psi_1}^{(n)}(\tau_1,\ldots,\tau_n)\cdot r_{\psi_2}^{(n)}(\tau_1,\ldots,\tau_n)\,\mathrm{d}\tau_1\ldots\mathrm{d}\tau_n \\
&= \int \cdots \int \left\{ \int \psi_1(t)\psi_1(t+\tau_1)\ldots\psi_1(t+\tau_n)\,\mathrm{d}t \right\} \\
&\quad \cdot \left\{ \int \psi_2(u)\psi_2(u+\tau_1)\ldots\psi_2(u+\tau_n)\,\mathrm{d}u \right\} \mathrm{d}\tau_1\ldots\mathrm{d}\tau_n \\
&= \int\int \psi_1(t)\psi_2(u) \left\{ \int \psi_1(t+\tau)\,\psi_2(u+\tau)\,\mathrm{d}\tau \right\}^n \mathrm{d}u\,\mathrm{d}t \\
&= \int\int \psi_1(t)\psi_2(s+t) \left\{ \int \psi_1(v)\psi_2(v+s)\,\mathrm{d}v \right\}^n \mathrm{d}s\,\mathrm{d}t \\
&= \int \left\{ \int \psi_1(v)\psi_2(v+s)\,\mathrm{d}v \right\}^{n+1} \mathrm{d}s \qquad \square
\end{aligned}$$

Considering the set of admissible values for τ_k being discrete and having m_k distinct values, it follows that the space of nth order autocorrelations has $\prod_{k=1}^{n} m_k$ dimensions, making the explicit computation of autocorrelations prohibitively expensive.

For any ψ, we build the corresponding autocorrelation feature vector $\boldsymbol{r}_{\psi}^{(n)}$ (column vector), by considering the values of $r_{\psi}^{(n)}(\tau_1, \dots, \tau_n)$ ordered sequentially (for example, by letting the variables τ_i run faster than τ_j over the set of admissible values, for any $i > j$). To simplify the notation, in the following we will denote by $\boldsymbol{r}_k$ the nth order autocorrelation vector corresponding to the function ψ_k.

In the following, we will investigate the properties of those vectors, using the discrete version of (2):

$$\langle \boldsymbol{r}_1, \boldsymbol{r}_2 \rangle = \sum_{\tau} \left\{ \sum_{t} \psi_1(t)\psi_2(t+\tau) \right\}^{n+1} \tag{3}$$

Let now $\{\boldsymbol{r}_1, \dots, \boldsymbol{r}_m\}$ be a set of linearly independent autocorrelation vectors (not necessarily orthogonal), let $R = [\boldsymbol{r}_1 | \cdots | \boldsymbol{r}_m]$ be the transformation matrix having these vectors as its columns, and let $\boldsymbol{r}$ be a new vector to be projected on the space spanned by $\{\boldsymbol{r}_k\}_{k=1}^{m}$. The vector $\boldsymbol{r}$ can be decomposed into two components:

$$\boldsymbol{r} = \boldsymbol{r}_W + \boldsymbol{r}_{\perp W} \tag{4}$$

where $\boldsymbol{r}_W \in W = \mathrm{Span}(\{\boldsymbol{r}_k\})$ and $\boldsymbol{r}_{\perp W}$ is orthogonal to W. Then, the orthogonal projection $\boldsymbol{r}_W$ onto W is given by (see Appendix A for a derivation of this result):

$$\boldsymbol{r}_W = R(R'R)^{-1}R'\boldsymbol{r}. \tag{5}$$

Note that all products $R'\boldsymbol{r}$ and $R'R$ imply only computations of inner products between autocorrelation vectors which can be computed by means of (2) and (3), avoiding the explicit computation of autocorrelations. This method of avoiding the explicit computation of autocorrelation vectors is similar to the *kernel trick*, used, for example, in the context of support vector machines (Vapnik, 1995).

2.2. Extended feature vectors

One may wish to combine different autocorrelation orders in a single and, hopefully, more descriptive feature vector. For this, it is enough to define $I = \{i_1, \dots, i_m\}$ as a set of indices and $\boldsymbol{r}_{\psi}^{(I)}$ as the vector obtained by concatenating the autocorrelations $\boldsymbol{r}_{\psi}^{(k)}$, where $k = i_1, \dots, i_m$. Then it is easy to see that

$$\langle \boldsymbol{r}_{\psi_1}^{(I)}, \boldsymbol{r}_{\psi_2}^{(I)} \rangle = \sum_{k=i_1,\dots,i_m} \langle \boldsymbol{r}_{\psi_1}^{(k)}, \boldsymbol{r}_{\psi_2}^{(k)} \rangle \tag{6}$$

meaning that computing the inner product of two compound feature vectors can be done by simply summing the inner products of the components.

Another consequence of this observation is the fact that the autocorrelations may be computed over any topology of the local neighborhood: one may consider a partition of the domain $D = \cup_i D_i$ and use an extended feature vector composed of autocorrelations computed on D_i for discriminating the patterns, still one can use a similar formula to (6) to compute the inner products. In the following two sections we use simple autocorrelation vectors when deriving our results, but the same techniques remain valid for extended feature vectors as well. Moreover, in the experimental section we present an application of extended feature vectors for face class modeling.

3. Applying PCA to autocorrelation feature vectors

Principal component analysis (PCA) is a technique for extracting the structure from a high-dimensional data set. PCA can be viewed as an orthogonal transformation of the coordinate system in which the data is described. This transformation is performed in the hope that a small number of principal directions will suffice to well-approximate the data. There are different methods to perform PCA, the most common requiring the diagonalization of the covariance matrix, or, equivalently, to solve the eigenproblem

$$C\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i \tag{7}$$

where C is the covariance matrix and λ_i are the eigenvalues corresponding to the eigenvectors $\boldsymbol{v}_i$.

Naturally, we are interested only in the non-trivial solution of (7). Following a similar approach as in (McLaughlin and Raviv, 1968), we will derive a method for performing PCA in autocorrelation space that employs only dot products between feature vectors.

Let $\{\boldsymbol{r}_k\}$ be a set of mean-centered autocorrelation vectors (later we will remove this assumption). The equivalent problem of (7) is

$$RR'\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i \tag{8}$$

where the elements of the matrix RR' are sums of outer products $\boldsymbol{r}_i\boldsymbol{r}_i'$ ($RR' = \sum_i \boldsymbol{r}_i\boldsymbol{r}_i'$). The rank of RR' cannot exceed m, the number of data/autocorrelation vectors, even if its dimensionality is usually much bigger than $m \times m$. We can solve the problem (8) indirectly, starting from a more manageable eigenproblem:

$$R'R\boldsymbol{w}_i = \lambda_i \boldsymbol{w}_i. \tag{9}$$

Note that the size of $R'R$ is $m \times m$ and that its eigenvectors are $\boldsymbol{w}_i$. Now, by left-multiplying with R we obtain

$$(RR')(R\boldsymbol{w}_i) = \lambda_i(R\boldsymbol{w}_i). \tag{10}$$

If we denote by $\boldsymbol{u}_i = R\boldsymbol{w}_i/\sqrt{\lambda_i}$, we see that $\boldsymbol{u}_i$ are the eigenvectors of $(RR')\boldsymbol{u}_i = \lambda_i\boldsymbol{u}_i$, so they are actually the eigenvectors $\boldsymbol{v}_i$ of problem (8). It follows that if $\lambda_i \neq 0$ (the only case we are interested in), then

$$\boldsymbol{v}_i = \frac{R\boldsymbol{w}_i}{\sqrt{\lambda_i}} \tag{11}$$

is the solution of (8) when $\boldsymbol{w}_i$ is the solution of (9). Since the ranks of RR' and $R'R$ are equal, there are no eigenvectors missed or added by this indirect method.

Then, the projections of the vectors $\{\boldsymbol{r}_k\}$ on the principal directions will be given by

$$\boldsymbol{a}_i = R'\boldsymbol{v}_i = \frac{R'R\boldsymbol{w}_i}{\sqrt{\lambda_i}}. \tag{12}$$

Generally, $\boldsymbol{v}_i$ are not valid nth order autocorrelations so the projection of a vector $\boldsymbol{r}$ on $\boldsymbol{v}_i$ cannot be computed directly as a simple inner product. Instead we have to use

$$\boldsymbol{r}'\boldsymbol{v}_i = \frac{\boldsymbol{r}'R\boldsymbol{w}_i}{\sqrt{\lambda_i}}. \tag{13}$$

All of the above development has been done supposing that the vectors $\boldsymbol{r}_k$ are centered around their mean. We will remove now this restriction and we will prove that the centering in the autocorrelation space can be carried out indirectly, without computing the autocorrelations. In Eqs. (8)–(13) we have to replace the matrix R with R_* where $R_* = [\boldsymbol{r}_1 - \bar{\boldsymbol{r}}|\cdots|\boldsymbol{r}_m - \bar{\boldsymbol{r}}]$, with $\bar{\boldsymbol{r}}$ being the mean autocorrelation vector. Computing the product $R_*'R_*$ reduces to computing the inner products $\langle \boldsymbol{r}_i - \bar{\boldsymbol{r}}, \boldsymbol{r}_j - \bar{\boldsymbol{r}}\rangle$ for all $i,j = 1,\ldots,m$:

$$\begin{aligned}\langle \boldsymbol{r}_i - \bar{\boldsymbol{r}}, \boldsymbol{r}_j - \bar{\boldsymbol{r}}\rangle = {} & \langle \boldsymbol{r}_i, \boldsymbol{r}_j\rangle - \frac{1}{m}\sum_{k=1}^{m}\langle \boldsymbol{r}_i, \boldsymbol{r}_k\rangle \\ & - \frac{1}{m}\sum_{k=1}^{m}\langle \boldsymbol{r}_j, \boldsymbol{r}_k\rangle + \frac{1}{m^2}\sum_{k,l=1}^{m}\langle \boldsymbol{r}_k, \boldsymbol{r}_l\rangle\end{aligned} \tag{14}$$

which translates into

$$\begin{aligned}R_*'R_* = {} & R'R - \frac{1}{m}1_{mm}(R'R) - \frac{1}{m}(R'R)1_{mm} \\ & + \frac{1}{m^2}1_{mm}(R'R)1_{mm}\end{aligned} \tag{15}$$

where 1_{mm} is an $m \times m$ matrix of ones.

Finally, we have to compute the projection of $\boldsymbol{r}_* = \boldsymbol{r}_0 - \bar{\boldsymbol{r}}$ on the principal axis, where $\boldsymbol{r}_0$ is a new autocorrelation vector which has to be projected on the principal directions. Similar to (15), we have:

$$\begin{aligned}\boldsymbol{r}_*'R_* = {} & \boldsymbol{r}_0'R - \frac{1}{m}1_{1m}(R'R) - \frac{1}{m}(\boldsymbol{r}_0'R)1_{mm} \\ & + \frac{1}{m^2}1_{1m}(R'R)1_{mm}\end{aligned} \tag{16}$$

and from (13) we have the projection on the ith principal direction:

$$\boldsymbol{r}_*'\boldsymbol{v}_{*i} = \frac{\boldsymbol{r}_*'R_*\boldsymbol{w}_{*i}}{\sqrt{\lambda_{*i}}} \tag{17}$$

where $\boldsymbol{v}_{*i}$, $\boldsymbol{w}_{*i}$ and λ_{*i} are obtained by considering Eqs. (9) and (11) with R_* replaced for R.

4. Higher-order autocorrelations in the context of kernel-based methods

A standard technique of transforming a linear classifier into a non-linear one, consists in projecting the initial space into a feature space, through a non-linear mapping $\Phi(\cdot)$. Now, being given a fixed mapping $\Phi : X \to K$, we define the *kernel* function as the inner product function $k : X \times X \to \mathbb{R}$, i.e., for all $\boldsymbol{x}_1, \boldsymbol{x}_2 \in X$:

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) \overset{\Delta}{=} \langle \Phi(\boldsymbol{x}_1), \Phi(\boldsymbol{x}_2) \rangle. \tag{18}$$

For a set of m vectors $\boldsymbol{x}_i \subset X^m$, the Gram matrix

$$G_{ij} \overset{\Delta}{=} \langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j) \rangle = k(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{19}$$

is called *kernel matrix*.

Usually we are not given the function $\Phi(\cdot)$, but the kernel function $k(\cdot,\cdot)$. Some of the kernels used in practice are

$$\begin{aligned}
&\text{polynomial} \quad k_{\mathrm{P}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + 1)^p \\
&\text{sigmoidal} \quad k_{\mathrm{S}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(\kappa \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + \delta) \\
&\text{radial basis function} \quad k_{\mathrm{RBF}}(\boldsymbol{x}_i, \boldsymbol{x}_j) \\
&\quad = \exp(-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2) \\
&\quad = \exp(-\gamma(\langle \boldsymbol{x}_i, \boldsymbol{x}_i \rangle + \langle \boldsymbol{x}_j, \boldsymbol{x}_j \rangle - 2\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle))
\end{aligned}$$

It follows that for all kernel functions that can be expressed in terms of inner products of data, we can use the technique developed above to carry out the computations of the kernel matrix. Then, performing a kernel PCA (Schölkopf et al., 1998) or training a Support Vector Machine (Vapnik, 1995; Cristianini and Shawe-Taylor, 2000) is an immediate task (Popovici and Thiran, 2001). We use this approach for computing the kernel matrix corresponding to a SVM trained in autocorrelation space and we present an application in Section 5.2.

5. Applications

In this section we will present different scenarios of using the autocorrelation feature vectors for pattern classification. The point here is to exemplify the use of autocorrelation features and not necessarily to build a state-of-the-art classifier for either of them. The first example uses the technique developed in Section 3, while the second shows how a kernel-based classifier can be trained in the autocorrelation space and is based on the considerations from Section 4.

5.1. 1D Functions

In order to assess the validity of the method presented here, we carried out a number of tests on the *Waveform* dataset from the UCI database (Blake and Merz, 1998). The set consists of 5000 samples of 1D signals, distributed equally in three classes. The goal of the experiment was to study the influence of different parameters in a binary classification task: discriminate between the first class (called A) and the other two (B and C).

The discrimination function was based on the *distance from feature space* (DFFS): a sample is classified as belonging to class A if its DFFS to the feature space of class A is less than a threshold (the DFFS is defined as the length of the orthogonal component of a vector that is to be projected on the space spanned by the eigenvectors of the covariance matrix—see for details Turk and Pentland (1991).

In all experiments, 500 vectors from class A (randomly chosen) have been used to perform PCA in autocorrelation space, and to determine the threshold. Another 500 vectors from class A and 500 from classes B and C have been used to test the classification. Fig. 1 presents some results obtained by autocorrelations features in comparison with some other methods (notation ACorr(n,d) is used to designate an autocorrelation function of order n having d distinct values for each of the variables τ_i (lags)).

5.2. 2D Functions

In the case of pattern recognition for computer vision, a large number of classical algorithms have been adapted to perform the classification of image patterns. Usually an image pattern of size $w \times h$ is considered as a vector in a (wh)-dimensional space. A disadvantage of this perspective is the loss of 2D neighborhood information. On the other hand, the autocorrelation features are easily

Method	Rate
Optimal Bayes classifier (Blake and Merz (1998))	**86.0%**
Nearest Neighbor (Blake and Merz (1998))	78.0%
CART decision tree (Blake and Merz (1998))	72.0%
DFFS using ACorr(2,5)	**80.5%**
DFFS using ACorr(2,7)	79.0%
DFFS using ACorr(3,5)	**81.6%**
DFFS using ACorr(3,7)	78.0%
DFFS using ACorr(4,5)	**81.5%**
DFFS using ACorr(4,7)	**80.0%**

Fig. 1. Classifications rates for Waveform dataset (%). The "Optimal Bayes" rate is the maximum theoretical rate that can be achieved if the distribution of the classes is known.

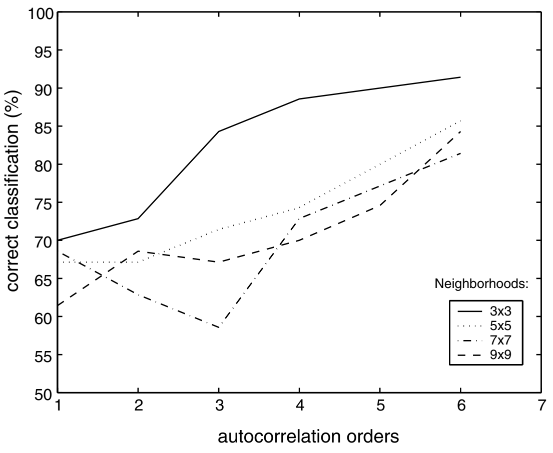

Fig. 2. The correct classification rates (%) on testing set using extended feature vectors.

adapted to the dimensionality of the problem, by choosing the correct shape of the neighborhood.

Here, we will briefly present a method for face class modelling that makes use of 2D autocorrelations for describing the face patterns. For a more detailed description the interested reader is referred to Popovici and Thiran (2001). We consider examples of human faces and non-faces in the form of intensity matrices of size 20×20 and we describe them in terms of their autocorrelation coefficients for orders from 1 to 6. More precisely, each image pattern (faces and non-faces alike) is considered to be described by an extended feature vector (see Section 2.2) of orders $1, \dots, k$ with $k = 1, \dots, 6$, and we test different neighborhood sizes. The positive examples (faces) have been cropped from the XM2VTS database (Messer et al., 1999). The negative examples (non-faces) have been collected in a bootstrap manner: an initial set was generated by randomly cropping 20×20 patterns from images without any human face. Then, the classifier was trained and tested on some other images that did not contain faces. The patterns that were classified as faces (so, actually false positives) have been collected and added to the negative set. The procedure was repeated several times, incrementally building a set of representative negative examples.

The classifier employed was a second-degree polynomial SVM and we applied the technique from Section 4 for computing the kernel matrix. No preprocessing has been used, so the result could be further improved by using some specific algorithms for image processing. Fig. 2 summarizes the results obtained.

It is interesting to note that generally (except for one case), an increase in number of autocorrelation orders led to an improvement in performances. However, it is still not clear how one should choose the neighborhood to be considered.

6. Conclusions

We have presented an effective method for using higher order autocorrelation coefficients for pattern recognition. While the autocorrelation feature vectors reside in a high dimensional space, making their direct use prohibitively expensive, we have shown that one may avoid their computation when the pattern recognition method can be expressed in terms of inner product of input data. Luckily, a large number of algorithms falls in this category, allowing the use of autocorrelation functions of any order.

Also, we have presented two possible scenarios of using these features for describing time series or image patterns. The generalization to higher dimensions of input data is immediate. For example, one may imagine the use of autocorrelation features for characterizing satellite images acquired in different spectra (so input patterns are of shape $w \times h \times b$ where b is the number of bands). Then one has just to define the topology over which the autocorrelation coefficients are to be considered, while keeping the same methodology of training the classifier as we have presented.

However, there are still two open questions: how can one choose the right order(s) for autocorrelation functions, without exploring a large range of them? and, how to choose the right topology of the neighborhood for a given task?

Acknowledgements

This work has been performed with the financial support of the Swiss OFES grant no. 99.0563.2 and of IM2-NCCR of the Swiss NFS.

Appendix A

A.1. Orthogonal projection of autocorrelation vectors

Using the notations from Section 2.1, any vector $\boldsymbol{r}$ can be written as a sum of two components: $\boldsymbol{r} = \boldsymbol{r}_W + \boldsymbol{r}_{\perp W}$. Since $\boldsymbol{r}_W \in W$, it is a linear combination of vectors $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_m$, so there exists a vector $\boldsymbol{c} \in \mathbb{R}^m$ such that

$$\boldsymbol{r}_W = R\boldsymbol{c}.$$

From the fact that $\boldsymbol{r}_{\perp W}$ is orthogonal on W (so $R'\boldsymbol{r}_{\perp W} = 0$) we have

$$R'\boldsymbol{r} = R'(\boldsymbol{r}_W + \boldsymbol{r}_{\perp W}) = R'\boldsymbol{r}_W = R'R\boldsymbol{c}.$$

It follows that

$$\boldsymbol{c} = (R'R)^{-1}R'\boldsymbol{r}$$
$$\boldsymbol{r}_W = R(R'R)^{-1}R'\boldsymbol{r}.$$

Thus we have the orthogonal projection of $\boldsymbol{r}$ on W. Further, we can obtain the modulus of the projection and the distance from the space W by

$$\|\boldsymbol{r}_W\|^2 = \langle \boldsymbol{r}_W, \boldsymbol{r}_W \rangle = (R'\boldsymbol{r})'(R'R)^{-1}(R'\boldsymbol{r})$$

$$\|\boldsymbol{r}_{\perp W}\|^2 = \|\boldsymbol{r}\|^2 - \|\boldsymbol{r}_W\|^2 = \langle \boldsymbol{r}, \boldsymbol{r} \rangle - (R'\boldsymbol{r})'(R'R)^{-1}(R'\boldsymbol{r}).$$

References

Blake, C., Merz, C., 1998. UCI repository of machine learning databases. URL http://www.ics.uci.edu/~mlearn/MLRepository.html.

Chawla, S., Shekhar, S., Wu, W., Ozesmi, U., 2000. Modeling spatial dependencies for mining geospatial data: An introduction.

Chetverikov, D., Foldvari, Z., 2000. Affine-invariant texture classification using regularity features. In: Pietikainen, M. (Ed.), Texture Analysis in Machine Vision. Series in Machine Perception and Artificial Intelligence. World Scientific.

Cristianini, N., Shawe-Taylor, J., 2000. An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods. Cambridge University Press.

Goudail, R., Lange, E., Iwamoto, T., Kyuma, K., Otsu, N., 1996. Face recognition system using local autocorrelations and multiscale integration. IEEE Trans. Pattern Anal. Machine Intell. 18 (10), 1024–1028.

Hotta, K., Kurita, T., Mishima, T., 1998. Scale invariant face detection method using higher-order local autocorrelation features extracted from log-polar image. In: Automatic Face and Gesture Recognition. Proc. 3rd IEEE Internat. Conf. on IEEE, pp. 70–75.

Keogh, E.J., Pazzani, M.J., 1998. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. Knowledge Discovery Data Mining, 239–243.

Kreutz, M., Volpel, B., Janssen, H., 1996. Scale-invariant image recognition based on higher order autocorrelation features. Pattern Recognition 29 (1) 19–26.

Kurita, T., Hotta, K., Mishima, T., 1998. Scale and rotation invariant recognition method using higher-order local autocorrelation features of log-polar image. In: Third Asian Conf. on Comput. Vision, pp. 89–96.

McLaughlin, J.A., Raviv, J., 1968. *N*th-order autocorrelations in pattern recognition. Inform. Control 12, 121–142.

Messer, K., Matas, J., Kittler, J., Luettin, J., Maitre, G., 1999. Xm2vtsdb: The extended m2vts database. In: Akunuri, S., Kullman, C. (Eds.), AVBPA'99, pp. 72–77.

Popovici, V., Thiran, J.P., 2001. Higher order autocorrelations for pattern classification. In: Proc. Internat. Conf. on Image Processing (ICIP). IEEE, pp. 724–727.

Schölkopf, B., Mika, S., Smola, A., Rätsch, G., Müller, K.-R., 1998. Kernel PCA pattern reconstruction via approximate pre-images. In: Niklasson, L., Boden, M., Ziemke, T. (Eds.), Proc. 8th Internat. Conf. on Artificial Neural Networks. Springer Verlag, pp. 147–152.

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

Vapnik, V., 1995. The Nature of Statistical Learning Theory. Springer Verlag.