Multi-Class Nonlinear Discriminant Feature Analysis

Tiene Andre Filisbino

Gilson Antonio Giraldi

tiene@lncc.br

gilson@lncc.br

Laboratório Nacional de Computação Científica

Getlio Vargas Av, 25651-075,Rio de Janeiro, Petrópolis, Brasil

Carlos Eduardo Thomaz

cet@fei.edu.br

Centro Universitário FEI

Av. Humberto de Alencar Castelo Branco, 09850-901, São Paulo, São Bernardo do Campo, Brasil

Abstract. *The problem of ranking features in N-class problems have been addressed by the multi-class discriminant principal component analysis (MDPCA) for texture and face image classification. In this paper we present a nonlinear version of the MDPCA, named multi-class nonlinear discriminant feature analysis (MNDFA), that is based on kernel support vector machines (KSVM) and AdaBoost techniques. Specifically, the problem of ranking features, computed from multi-class databases, is addressed by applying the AdaBoost procedure in a nested loop: each iteration of the inner loop boosts weak classifiers to a moderate one while the outer loop combines the moderate classifiers to build the global discriminant vector. The inner and outer loop procedures use AdaBoost techniques to combine learners. In the proposed MNDFA, each weak learner is a linear classifier computed through a separating hyperplane, defined by a KSVM decision boundary, in the feature space. In the computational experiments we analyse the obtained approach using a five-class granite image database. Our experimental results have shown that the features selected by the proposed technique allow competitive recognition rates when compared with related methods.*

Keywords: *Nonlinear,Multi-Class,KSVM, Discriminant Analysis, AdaBoost, Texture Analysis, Classification*

1 INTRODUCTION

Many areas such as computer vision, signal processing and medical image analysis, have as main goal to get enough information to distinguish sample groups in classification tasks Hastie *et al.* (2001). Hence, discriminant analysis should be performed for discarding redundancies and to select important features for pattern recognition Cunningham & Ghahramani (2015); Hastie *et al.* (2001).

In this avenue, we follow a statistical learning approach whose basic pipeline can be described as follows Giraldi *et al.* (2008): (a) The acquisition of information through feature extraction ; (b) Among the features obtained, apply machine learning to select the most discriminant ones for separating sample groups in classification problems Filisbino *et al.* (n.d.). The step (a) can be accomplished through classical works on features computation through histograms, object shape, image transforms, color patterns, besides the very known Haralick's descriptors for texture analysis Haralick *et al.* (1973); Zayed & Elnemr (2015), among others. The determination of discriminant features (step (b) above), in general, depends on the incorporation of prior information based on labeled data. The linear discriminant analysis (LDA) Hastie *et al.* (2001), Fisher criterion Filisbino *et al.* (2015a), discriminant principal components analysis (DPCA) Thomaz & Giraldi (2010) and its extension to multi-class problems, Multi-Class DPCA and Multi-Class.M2 DPCA Filisbino *et al.* (2015b, 2016), are methods reported in the literature for discriminant features selection.

In this work, we focus on N-class problems but we tackle discriminant analysis for nonlinear classification tasks. To perform this, we apply the following methodology: build an ensemble of moderate linear classifiers to get local discriminant weights that are combined through AdaBoost.M2 technique in order to determine the discriminant contribution of each feature Filisbino *et al.* (2015b, 2016). Ensemble methods, like AdaBoost.M2, find an accurate classifier by combining many moderate learners Zhou (2012). However, it is known that a strong learner does not work well as the base component for Adaboost Garcia & Lozano (2007). Therefore, we implement a strategy to compute the moderate learners through weak learners, useful as AdaBoost components, that are built through the kernel support vector machine (KSVM) hypersurface geometry. Each moderate linear classifier is constructed through a linear ensemble of weak classifiers that are computed through a sampling of the tangent bundle (set of tangent spaces) of the decision boundary. To implement this computation, each iteration of the novel algorithm calculates a separating hypersurface based on the "one-against-all" KSVM multi-class approach. In this way, we keep the idea of using a robust classifier to steer to process of discriminant analysis, started by the DPCA.

However, the decision boundary yielded by KSVM generates an infinite ensemble of weak linear classifiers given by tangent spaces, each one encapsulating a local discriminant vector. So, we construct a systematic method to get a finite ensemble of the tangent bundle, and import from Filisbino *et al.* (2015b) the idea of combining these weak components through AdaBoost approach to build a moderate classifier, at the end of each iteration of the internal loop. Finally, the global discriminant vector is constructed by executing the AdaBoost.M2 methodology, but now using the moderate classifiers as input. The proposed technique, called multi-class nonlinear discriminant feature analysis (MNDFA), is the main contribution of this paper. It extends the Multi-Class.M2 DPCA to nonlinear classification problems and more general feature spaces.

It is important to highlight that we do not deal with the problem of computing general

discriminant directions that are different from the original features, like LDA does. Rather, we apply the idea of using separating hypersurfaces computed by KSVM, linear classifiers (tangent hyperplanes) and ensemble methods (AdaBoost, in this case) to compute discriminant weights that are used to select, among the original features, the most discriminant ones. We have focused here on the KSVM Vapnik (1998) method but any other separating hypersurface could be used.

To evaluate the MNDFA algorithm, we perform group separation tasks using the same granite tiles data set applied in Bianconi *et al.* (2015). Firstly, we use the co-occurrence matrix and Haralick's descriptors to compute the feature vectors to represent the sample images Arvis *et al.* (2011); Haralick *et al.* (1973). A co-occurrence matrix for a given image is generated by computing the distribution of co-occurring pixel values (gray scale values, or colors) at a given offset. Hence, we can describe texture through a set of features considering multiple directions defined by a set of offsets. To summarize the information contained in the co-occurrence matrix we follow Zayed & Elnemr (2015) and compute the Haralick's texture features Haralick *et al.* (1973), in order to generate the feature space. These methodologies are classical in the pattern recognition and texture image analysis Bianconi *et al.* (2012). Our experimental results have shown that the features selected by the discriminant method MNFDA in nonlinear classification problems allow competitive recognition rates when compared with Fisher criterion Loog *et al.* (2001), and a straightforward variation of the Multi-Class LDA-DPCA and Multi-Class.M2 DPCA, both proposed in Filisbino *et al.* (2016).

The paper is organized as follows. In section 1.1 we survey related works for discriminant analysis. Next, section 1.2 presents the main stages of the proposed method. Then, in section 2 we review the theory behind MNDFA approach. Section 3 presents the MNDFA algorithm. The feature extraction technique applied in this paper is presented in section 4. The computational experiments are described in section 5. Finally, in section 6, we conclude the paper, summarizing its main contributions and describing further developments.

1.1 RELATED WORK

Given a feature space, the key question in this work is "how can we determine the most important discriminant features for a pattern recognition task, like classification?" Discriminant analysis techniques address this question, which is represented in Figure 1. Both Figures 1.(a) and 1.(b) pictures the same data set. Figure 1.(a) just shows the directions ($\widetilde{\mathbf{x}}$ and $\widetilde{\mathbf{y}}$) and the distribution of the samples over the space. However, in Figure 1.(b) we distinguish two patterns: plus $(+)$ and triangle $(\blacktriangledown)$. We observe that the direction $\widetilde{\mathbf{x}}$ can not discriminate samples of the considered groups because the projection of the data points over direction $\widetilde{\mathbf{x}}$ will mix the patterns in the corresponding one-dimensional subspace. Therefore, in terms of classification, the direction $\widetilde{\mathbf{y}}$, as well as the corresponding feature, is more discriminant than the other one.

In general, the LDA is used to identify the most important linear directions for separating sample groups Hastie *et al.* (2001); Zhu (2006). In the two-class case, its objective is to find a projection direction that maximizes the Fisher criterion.

The problem reported in Figure 1 is very studied in the context of principal components analysis (PCA). In Zhu & Martinez (2006) it is proposed a specific method for selecting principal components that is based on the spectral decomposition of the LDA characteristic equation, selecting the features that most correlate to the between class scatter matrix. In Thomaz & Giraldi (2010) authors proposed the DPCA technique, based on the idea of ranking the principal

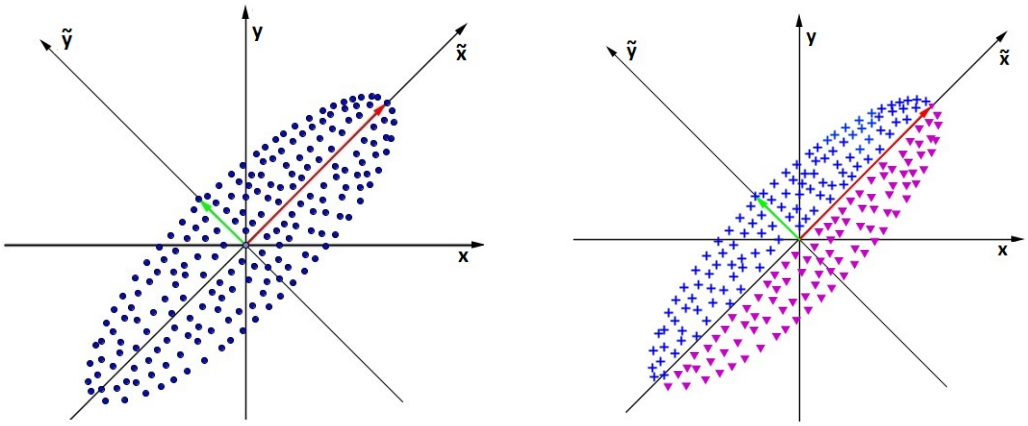


Figure 1: (a) Scatter plot of samples. (b) The same population but distinguishing patterns plus $(+)$ and triangle $(\blacktriangledown)$.

components by how well they align with separating hyperplane directions, determined by the corresponding discriminant weights. Such a set of principal components ranked in decreasing order of the discriminant weights is called in Thomaz & Giraldi (2010) the discriminant principal components. The Multi-Class DPCA, described in Filisbino *et al.* (2015b, 2016), consists of the following steps: (a) apply PCA technique for dimensionality reduction in order to eliminate redundancy. (b) Compute a linear ensemble, based on the one-against-all SVM multi-class approach. (c) Combine the discriminant weights computed through the separating SVM hyperplanes in order to determine the discriminant contribution of each feature. The DPCA and its multi-class version have been successfully applied to facial expression and texture classification experiments Filisbino *et al.* (2015b, 2016).

1.2 TECHNIQUE OVERVIEW

The whole MNDFA methodology is shown in Figure 2. Firstly, we perform feature extraction using a suitable technique. Each iteration of the main loop is composed by the following steps: (a) Compute a separating hypersurface using the KSVM approach and a weighted version of the original data set (KSVM surface in Figure 2); (b) Among the support vectors, select a subset, represented by $\mathbf{x}_1, \cdot\cdot, \mathbf{x}_4$ in Figure 2, useful to generate AdaBoost components and that preserve the geometry of KSVM decision boundary; (c) For each selected support vector, calculate the corresponding tangent hyperplane (weak classifiers C_i in the figure) and AdaBoost weight; (d) Perform linear combination of the weak classifiers to get a moderate accurate learner.

In the last step of the MNDFA procedure, the moderate classifiers are linearly combined through their AdaBoost.M2 weights to produce the global discriminant vector. The key idea of this step is based on the fact that AdaBoost.M2 linearly combines components classifiers to get the final hypothesis. So, it is straightforward to obtain the global discriminant weights from the expression that defines the strong classifier.

Finally, we follow the traditional DPCA proposal (see section 2) and sort texture features in the decreasing order of the global discriminant weights. The output of MNDFA algorithm is

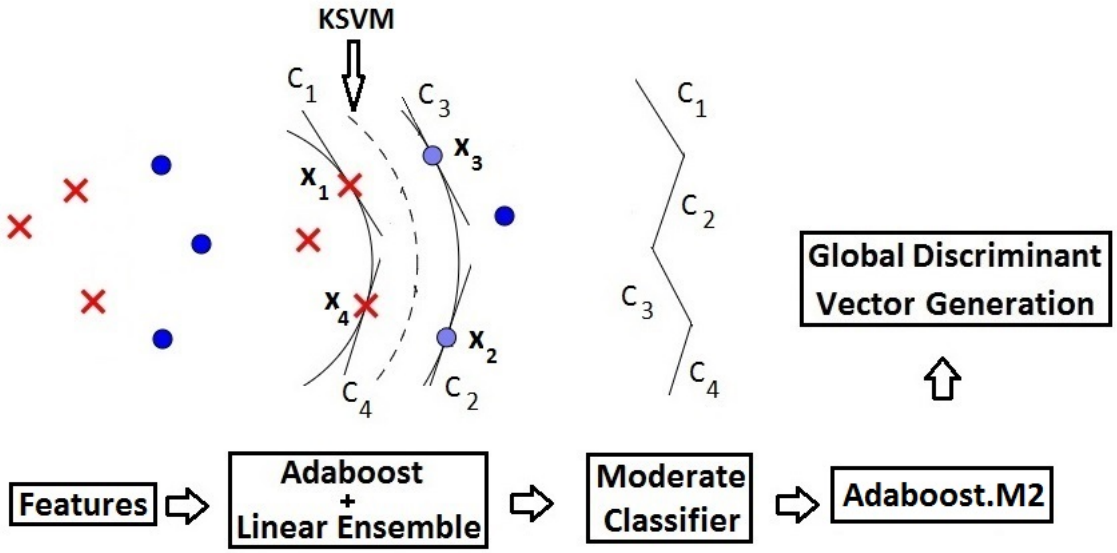


Figure 2: Main steps of the MNDFA technique.

the canonical texture directions arranged according to the discriminant weights. The method is not restricted to any application or particular probability density function of the sample groups and the number of meaningful discriminant directions is not limited to the number of groups, like LDA Hastie *et al.* (2001).

2 TECHNICAL BACKGROUND

The foundation of MNDFA technique includes the original DPCA methodology Thomaz & Giraldi (2010) and KSVM Vapnik (1998), which are described bellow. Let the training observations $\mathbf{x}_i \in \mathbb{R}^n$, $i = 1, \cdots, M$ that generate a $M \times n$ training set matrix $\widetilde{\Theta}$ centered respect to the global mean $\widehat{\mathbf{x}}$. The DPCA methodology works in the PCA space. The PCA algorithm computes a transformation matrix $P_{pca} = [\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_{m'}]$ whose columns $\mathbf{p}_i$, $i = 1, \ldots, m'$ minimize the mean square reconstruction error, being the $m' \leq n$ eigenvectors of the covariance matrix Ω of $\widetilde{\Theta}$ that correspond to the m' largest eigenvalues Turk & Pentland (1991).

If to each training sample $\mathbf{x}_i$ it is associated a label $y_i \in \{-1, 1\}$, then we have a projected labeled training set:

$$X = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \ldots (\mathbf{x}_M, y_M)\}, \tag{1}$$

and, we can apply the DPCA technique to select the most discriminant principal components to separate sample groups.

The original DPCA is implemented taking as input a training set X, like in expression (1). Firstly, for discarding redundancies, the PCA transformation matrix $P_{pca} = [\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_{m'}]$ is computed and each zero mean data vector $\widetilde{\mathbf{x}}_i$ is projected generating a vector $\overline{\mathbf{x}}_i = (P_{pca})^T \widetilde{\mathbf{x}}_i$. Afterwards, the obtained $M \times m'$ data matrix and their corresponding labels are used as input to calculate the separating hyperplane, generically represented by the discriminant vector $\phi = (w_1, w_2, \cdots, w_{m'})$ given by a linear classifier. If we multiply the $M \times m'$ most expressive features

matrix by the $m' \times 1$ discriminant vector:

$$\begin{aligned} c_1 &= \overline{\mathbf{x}}_{11} w_1 + \overline{\mathbf{x}}_{12} w_2 + ... + \overline{\mathbf{x}}_{1m'} w_{m'}, \\ c_2 &= \overline{\mathbf{x}}_{21} w_1 + \overline{\mathbf{x}}_{22} w_2 + ... + \overline{\mathbf{x}}_{2m'} w_{m'}, \\ &... \\ c_M &= \overline{\mathbf{x}}_{N1} w_1 + \overline{\mathbf{x}}_{N2} w_2 + ... + \overline{\mathbf{x}}_{Nm'} w_{m'}. \end{aligned} \tag{2}$$

we get the most discriminant feature $c_i \in \mathbb{R}$ of each one of the m'-dimensional vectors $\overline{\mathbf{x}}_i$. Therefore, we can determine the discriminant contribution of each feature by investigating the weights $[w_1, w_2, ..., w_{m'}]$. In fact, weights that are estimated to be 0 or approximately 0 have negligible contribution on the discriminant scores c_i described in equation (2), indicating that the corresponding features are not significant to separate the sample groups. In contrast, largest weights (in absolute values) indicate that the corresponding features contribute more to the discriminant score and consequently are important to characterize the differences between the groups.

Therefore, instead of sorting these features by selecting the corresponding principal components in decreasing order of eigenvalues, as PCA does, DPCA selects as the most important features for classification the ones with the highest discriminant weights, that is, $|w_1| \geq |w_2| \geq ... \geq |w_{m'}|$.

The background of KSVM belongs to the reproducing kernel Hilbert spaces and Mercer theory. A remarkable results in this scenario is the Mercer theorem, which we summarized bellow in order to set the mathematical machinery that we are going to use in what follows Vapnik (1998). So, let the space $\mathbb{R}^n$ and μ a finite measure in $\mathbb{R}^n$. We define also the function spaces $L_2(\mathbb{R}^n) = \{f : \mathbb{R}^n \longrightarrow \mathbb{R};\ |f|^2\ is\ \mu - integrable\}$ and $L_\infty(\mathbb{R}^n) = \{f : \mathbb{R}^n \longrightarrow \mathbb{R};\ \exists K > 0,\ |f(\mathbf{x})| \leq K\}$.

Theorem 1 (Mercer): Suppose $k : \mathbb{R}^n \times \mathbb{R}^n \longrightarrow \mathbb{R}$ is a continuous symmetric positive definite function (kernel) such that $k \in L_\infty(\mathbb{R}^n \times \mathbb{R}^n)$. Under certain conditions, the integral operator $T_k : L_2(\mathbb{R}^n) \longrightarrow L_2(\mathbb{R}^n)$:

$$(T_k f)(\mathbf{x}) = \int_{\mathbb{R}} k(\mathbf{x}, \mathbf{y}) f(\mathbf{y}) d\mu(\mathbf{y}), \tag{3}$$

has a set of normalized eigenfunctions $\psi_j : \mathbb{R}^n \longrightarrow \mathbb{R}$, with associated eigenvalues $\lambda_j > 0$, sorted in nonincreasing order, such that: $k(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{n_F} \lambda_j \psi_j(\mathbf{x}) \psi_j(\mathbf{y})$. Either $n_F \in \mathbb{N}$ or $n_F = \infty$.

With this result, the KSVM generalizes the linear support vector machines through the kernel function k which allows to write the hypersurface that separates positive from negative samples in the input space as Vapnik (1998):

$$F(\mathbf{x}) \equiv \sum_{i=1}^{M} y_i \alpha_i k(\mathbf{x}_i, \mathbf{x}) + \widetilde{b} = 0, \tag{4}$$

where $\alpha_i \geq 0, i = 1, 2, \cdot, \cdot, \cdot, M$, are Lagrange multipliers in the quadratic optimization problem behind KSVM technique Vapnik (1998). The samples $\mathbf{x}_i$ with $\alpha_i \neq 0$ are named support vectors. If $n_F \in \mathbb{N}$ then, there exists a map $\Phi(\mathbf{x}) \equiv (z_1(\mathbf{x}), z_2(\mathbf{x}), z_3(\mathbf{x}), \ldots, z_{n_F}(\mathbf{x}))$ such that the separating surface in the feature space $\Re^{n_F}$ is given by:

$$\sum_{r=1}^{n_F} \omega_r z_r(\mathbf{x}) + \widetilde{b} = 0 \tag{5}$$

where $\omega_r = \sum_{i=1}^{M} y_i \alpha_i z_r(\mathbf{x}_i)$. In this work, we consider radial basis function (RBF) kernel, given by:

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(\frac{||\mathbf{x} - \mathbf{y}||^2}{2\sigma^2}\right), \tag{6}$$

where $\sigma \in \mathbb{R}$.

3 MNDFA DISCRIMINANT METHODOLOGY

The MNDFA procedure is described by the Algorithm 1 that has as input: (i) The training instances in the image labeled database $X \subset \mathbb{R}^n \times Y$, where $Y = \{1, 2, 3, .., N\}$, that are supposed independently and identically distributed from an uniform distribution D; (ii) The maximum number of iterations T; (iii) Kernel type k; (iv) Accumulation parameter τ, percentage μ; (v) Thresholds T_{Rmin} and T_{Rmax}.

Following the pipeline in Figure 2, the first stage of MNDFA builds a labeled feature space $(\overline{\mathbf{x}}_i, y_i) \in \mathbb{R}^m \times Y$, where $\overline{\mathbf{x}}_i \in \mathbb{R}^m$ is the feature vector of the image $\mathbf{x}_i$ that belongs to the class y_i. The labeled feature vectors are built in line 2 of Algorithm 1 and compose the input to the whole methodology. The key idea behind MNDFA is to apply the AdaBoost procedure in a nested loop: given a class label y, each iteration of the inner loop (lines 6-16) uses an one-against-all technique to boost weak classifiers (tangent hyperplanes to KSVM decision boundary), denoted by $h^t_{j,y}$ in line 9, to a moderate one ($\overline{h}^t_y$, in line 16) while the outer loop computes AdaBoost.M2 weights (α^t, line 21) that allow to combine the moderate classifiers to build the global discriminant vector. So, as we have N classes, the internal loop in the Algorithm 1 (line 6 to 16) constructs N weakened learners using the Algorithm 2 and an AdaBoost procedure. To do this, in line 7 of Algorithm 1 we build the set $\overline{\Theta}^y$, generated by taking all k_y feature vectors from class y and label them as 1. Then, using random sampling we choose $(2k_y)/(N-1)$ samples from classes other than y and label them as -1. The obtained set of feature vectors $\overline{\mathbf{x}}^y_m \in \mathbb{R}^m$ and corresponding labels $y_m \in \{-1, 1\}$:

$$\overline{\Theta}^y = \left\{ (\overline{\mathbf{x}}^y_1, l_1), (\overline{\mathbf{x}}^y_2, l_2), ... \left(\overline{\mathbf{x}}^y_{3k_y}, l_{3k_y}\right) \right\}, \tag{7}$$

are the input to call the Algorithm 2, in line 8 of Algorithm 1, which constructs a set of weak classifiers $h^t_{j,y}$, each one represented by a tangent hyperplane to the KSVM decision boundary that separates sample groups in the set $\overline{\Theta}^y$. Basically, the procedure HL yields the KSVM hypersurface using the weighted data X^* as training set, selects a list L_{sv} of support vectors (see line 7 from Algorithm 2) and computes an hyperplane for each element in L_{sv}. In general, the obtained linear learners $h^t_{1,y}, h^t_{2,y}, \cdots, h^t_{N^t_{sv},y}$, are weak classifiers. In order to apply KSVM with AdaBoost we weaken the KSVM methodology (following the linear approach proposed by Garcia & Lozano (2007)) using a parameter $0 < \mu < 1$ to discard a percentage μ of the samples to generate the training set (line 2 of Algorithm 2). Besides, in order to eliminate unrepresentative support vectors we consider the accumulation parameter τ in line 7.

The inner loop of the Algorithm 1 computes in lines $10-15$ an AdaBoost weight $\overline{\alpha}_{k,y}^{t}$ for each weak classifiers $h_{k,y}^{t}$. Then, in line 16, the moderate classifier $\overline{h}_{y}^{t}$ is produced following the manner that AdaBoost combines the component learners. The computation of the AdaBoost weights $\overline{\alpha}_{k,y}^{t}$ is based on the idea of deriving AdaBoost method by using the linear combination of learners, generically denoted in Zhou (2012) as $h^1, h^2, \cdots, h^T$. In Zhou (2012) the stronger classifier is given by:

$$H\left(\mathbf{x}\right) = \sum_{t=1}^{T} \alpha^t h^t\left(\mathbf{x}\right), \tag{8}$$

where α^t does not depend on $\mathbf{x}$, Zhou (2012). This formulation seeks a strong learner H that minimizes the exponential loss function:

$$l_{exp}\left(H|D^t\right) = E_{\mathbf{x}\sim D^t}\left[e^{-\chi(\mathbf{x})H(\mathbf{x})}\right] = \sum_{i} e^{-\chi(\mathbf{x}_i)H(\mathbf{x}_i)} D^t\left(\mathbf{x}_i\right),$$

where χ is the ground-truth label function (replaced by g in line 11), with $\chi\left(\mathbf{x}\right) \in \{-1, +1\}$, and D^t is a probability distribution Zhou (2012). The strong learner H is produced by iteratively generating h^t and α^t. When a classifier h^t is generated under the distribution D^t, its coefficient α^t in expression (8) is to be determined in order to minimizes the exponential loss Zhou (2012):

$$l_{exp}\left(\alpha^t h^t|D^t\right) = e^{-\alpha^t}\left(1-\epsilon^t\right) + e^{\alpha^t}\epsilon^t,$$

where:

$$\epsilon^t = \left[\sum_{i;\chi(\mathbf{x}_i)\neq h_t(\mathbf{x}_i)} D^t\left(\chi\left(\mathbf{x}_i\right) \neq h_t\left(\mathbf{x}_i\right)\right)\right].$$

To obtain the optimal α_t, we must solve the equation:

$$\frac{\partial l_{exp}\left(\alpha_t h^t|D^t\right)}{\partial \alpha^t} = -e^{-\alpha_t}\left(1-\epsilon^t\right) + e^{\alpha_t}\epsilon^t = 0, \tag{9}$$

whose solution, after a simple algebra, is given by:

$$\alpha^t = \frac{1}{2} ln\left(\frac{1-\epsilon^t}{\epsilon^t}\right), \tag{10}$$

which justifies lines $10-16$ of MNDFA procedure.

Next, in the outer loop, each moderate classifier $\overline{h}_{y}^{t}$ receives a normalized AdaBoost weight $\widetilde{\alpha}^t$, calculated in line 24. Each hypothesis $\overline{h}^t$, in line 17 of Algorithm 1, has the form $\overline{h}^t : \overline{\Theta} \rightarrow [0, 1]$, and can be interpreted as the probability that y is the correct label associated with instance $\mathbf{x}$. It is generated through a moderate classifier $\overline{h}_{y}^{t}$ and the following normalization function:

$$f\left(z\right) = \frac{z - z_{min,y}^{t}}{z_{max,y}^{t} - z_{min,y}^{t}}, \tag{11}$$

where $f : [z^t_{min,y}, z^t_{max,y}] \rightarrow [0, 1]$, with $z^t_{min,y}$ and $z^t_{max,y}$ being the minimum and maximum values, respectively, of the set $\left\{ < \overline{\mathbf{x}}_i, \mathbf{\Phi}^t_y > +\Psi^t_y, \quad i = 1, 2, \ldots, M \right\}$. So, given a sample $\overline{\mathbf{x}}_i$, the probability of choosing an incorrect label y is: $Pr = \frac{1}{2}\left(1 - \overline{h}^t(\overline{\mathbf{x}}_i, y_i) + \overline{h}^t(\overline{\mathbf{x}}_i, y)\right)$ Freund & Schapire (1997). However, we have $|Y| - 1$ possibilities to obtain the incorrect answer. So, we can define the loss of the hypothesis through a weighted average according to some $q^t_{i,y}$, called the label weighting function, that assigns to each example i in the training set a load, with $\sum_{y \neq y_i} q^t_{i,y} = 1$. The resulting formula is called the pseudo-loss of $\overline{h}^t$ on training instance i with respect to q^t Freund & Schapire (1997):

$$ploss_q\left(\overline{h}^t, i\right) = \frac{1}{2}\left(1 - \overline{h}^t(\overline{\mathbf{x}}_i, y_i) + \sum_{y \neq y_i} q_{i,y}\overline{h}^t(\overline{\mathbf{x}}_i, y)\right). \tag{12}$$

So, following the AdaBoost.M2 strategy Freund & Schapire (1997), in each iteration t of the Algorithm 1, the weak learner's goal is to minimize the expected pseudo-loss, computed in line 18 of the Algorithm 1, for a distribution D^t and weighting function q^t. The algorithm uses a second weight vector whose values at time t are denoted by $w^t_{i,y}$, $i = 1, \cdots, M$, $y \in Y - \{y_i\}$, which is initialized in line 1, based on the initial distribution D. The main loop of the algorithm aims to update these weights in order to minimize the expected pseudo-loss. So, the weighting function q^t and the distribution D^t are computed using the $w^t_{i,y}$ (line 5 of Algorithm 1). The lines 16-18 of the Algorithm 1 are based on the AdaBoost.M2 idea of deriving a strong learner $\overline{h}_f$ by using the linear combination of weak learners $\overline{h}^1, \overline{h}^2, \cdots, \overline{h}^T$:

$$\overline{h}_f(\overline{\mathbf{x}}) = \arg\max_{y \in Y} \sum_{t=1}^{T} \widetilde{\alpha}^t \overline{h}^t(\overline{\mathbf{x}}, y), \tag{13}$$

where $\widetilde{\alpha}^t$ is computed in line 24. To see this, we shall remember that $h^t(\overline{\mathbf{x}}, y)$ in line 17 is computed through the function f, in expression (11), and rewrite expression (13) as:

$$\overline{h}_f(\overline{\mathbf{x}}) = \arg\max_{y \in Y} \left[\sum_{t=1}^{T} \widetilde{\alpha}^t f\left(< \overline{\mathbf{x}}, \mathbf{\Phi}^t_y > +\Psi^t_y\right)\right]. \tag{14}$$

But, from equation (11), we get:

$$f\left(< \overline{\mathbf{x}}, \mathbf{\Phi}^t_y > +\Psi^t_y\right) = \frac{< \overline{\mathbf{x}}, \mathbf{\Phi}^t_y > +\Psi^t_y - z^t_{min,y}}{z^t_{max,y} - z^t_{min,y}}. \tag{15}$$

Therefore, by substituting this expression into equation (14), and using the linearity of the inner product, we can show that:

$$\overline{h}_f(\overline{\mathbf{x}}) = \arg\max_{y \in Y} \left[< \overline{\mathbf{x}}, \boldsymbol{\Upsilon}_y > +\xi_y\right], \tag{16}$$

where:

$$\boldsymbol{\Upsilon}_y = \sum_{t=1}^{T} \widetilde{\alpha}^t \frac{\Phi^t_y}{z^t_{max,y} - z^t_{min,y}}, \quad \xi_y = \sum_{t=1}^{T} \widetilde{\alpha}^t \frac{\left(\Psi^t_y - z^t_{min,y}\right)}{z^t_{max,y} - z^t_{min,y}},$$

with $\boldsymbol{\Upsilon}_y \in \mathbb{R}^m$ and $\xi_y \in \mathbb{R}$. The bias ξ_y can be incorporated in the inner product through a translation $\overline{\boldsymbol{T}}_y$ such that $< \overline{\boldsymbol{T}}_y, \boldsymbol{\Upsilon}_y >= \xi_y$, which renders:

$$\overline{h}_f\left(\overline{\mathbf{x}}\right) = \arg\max_{y \in Y} \left[\sum_{i=1}^{n} \left(\overline{x}_i + \overline{T}_{i,y} \right) \Upsilon_{i,y} \right]. \tag{17}$$

This expression is the key to apply the DPCA methodology, described in section 2. Specifically, each feature i has a vector of weights $\Upsilon_{i,y}$ with size $|Y|$. So, for each feature i we need to seek for the most important weight, in absolute value $|\Upsilon_{i,y}|$, which can be interpreted as a measure of the discriminant contribution of the corresponding feature. These values are used to generate the vector $\mathbf{v}$ in line 27 of Algorithm 1. Next, we shall sort the obtained array in decreasing order, as performed in line 28 of the Algorithm 1, to get the global discriminant weights. The output of the MNDFA procedure is the discriminant directions $\overline{\mathbf{q}}_1, \overline{\mathbf{q}}_2, \cdots, \overline{\mathbf{q}}_m$ where $\overline{\mathbf{q}}_i$ is a feature direction selected according to its discriminant weight $v\left(i\right)$.

4 FEATURE EXTRACTION

In this paper, the process of image feature extraction consists in the computation of Haralick's texture features through the gray level co-occurrence matrix (GLCM), computed considering multiple offsets, to represent the visual information Zayed & Elnemr (2015). So, given an $M \times N$ image I coded using L grey levels in the set $\{0, 1, 2, \ldots, L-1\}$ and an offset $(\Delta x, \Delta y)$, the GLCM matrix is a $L \times L$ two-dimensional array define by:

$$GLCM\left(i, j, \Delta x, \Delta y, I\right) = \sum_{x=1}^{M} \sum_{y=1}^{N} \delta\left(I\left(x, y\right) - i, I\left(x + \Delta x, y + \Delta y\right) - j\right), 0 \leq i, j \leq (L-1) \tag{19}$$

where the function δ is defined as: $\delta\left(m, n\right) = 1$, if $m = n = 0$, and, $\delta\left(m, n\right) = 0$, otherwise.

Therefore, the GLCM characterizes the texture of an image by calculating how often pairs of pixels $\left(\left(x, y\right), \left(x + \Delta x, y + \Delta y\right)\right)$ with the gray level intensities (i, j) occur in a specific offset $(\Delta x, \Delta y)$ Zayed & Elnemr (2015). We shall emphasize that we are working in the digital context and, consequently, expression (19) only makes sense if $\Delta x, \Delta y \in \mathbb{N}$.

Figure 3 illustrates the process to build the GLCM by only walking in horizontal direction, $(\Delta x, \Delta y) = (1, 0)$. In the Figure 3, $GLCM(1, 1, 1, 0, I) = 1$ because there is only one instance in the input image I where two horizontally adjacent pixels $I\left(x, y\right)$ and $I\left(x + 1, y\right)$ have values $I\left(x, y\right) = 1$ and $I\left(x + 1, y\right) = 1$. Analogously, the $GLCM(1, 2, 1, 0, I) = 2$ because there are two pairs of pixels such that $I\left(x, y\right) = 1$ and $I\left(x + 1, y\right) = 2$, and so on.

In this paper, the process of image feature extraction consists in calculating the Haralick's texture features through the GLCM matrix computed considering multiple directions to represent the visual information Felix *et al.* (2016); Zayed & Elnemr (2015).

Haralick extracted 14 descriptors from the co-occurrence matrix Haralick *et al.* (1973), but only five texture features are frequently used due to correlations between the descriptors Arvis *et al.* (2011). So, in this work, we use only the five texture features presented in Table

Algorithm 1: MNDFA Procedure

Input: Samples: $X = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \dots (\mathbf{x}_M, y_M)\}$; where $y_i \in Y$ and $Y = \{1, 2, 3, .., N\}$; Distribution D over the M examples; Parameters τ, T, and μ; (v) Thresholds T_{Rmin} and T_{Rmax}; Kernel type k;

Initialize the weight vector: $w_{i,y}^1 = \frac{D(i)}{|Y|-1}$, $for \quad i = 1, \cdots, M$; $y \in Y - \{y_i\}$;

Build the labeled feature vectors $\overline{\Theta} = \{(\overline{\mathbf{x}}_1, y_1), (\overline{\mathbf{x}}_2, y_2) \dots (\overline{\mathbf{x}}_M, y_M)\} \subset \mathbb{R}^m \times Y$;

for $t = 1, ...$ *to* T **do**

  Set $W_i^t = \sum_{y \neq y_i} w_{i,y}^t$ and $\mathcal{Y} = \{-1, 1\}$;

  for $y \neq y_i$: $q_{i,y}^t = \frac{w_{i,y}^t}{W_i^t}$; and set $D^t(i) = \frac{W_i^t}{\sum_{i=1}^N W_i^t}$;

  for $y = 1, ...$ *to* N **do**

    Build the subset $\overline{\Theta}^y$, given by expression (7);

    $\left\{(\boldsymbol{\phi}_{j,y}^t, b_{j,y}^t);\ 1 \leq j \leq N_{sv}^t\right\} = HL(\overline{\Theta}^y, D^t, \mathcal{Y}, \tau, \mu)$;

    Define $h_{j,y}^t(\overline{\mathbf{x}}) =< \overline{\mathbf{x}}, \boldsymbol{\phi}_{j,y}^t > + b_{j,y}^t$;

    for $k = 1, ...$ *to* N_{sv}^t **do**

      $\overline{\epsilon}_k^t = P_{x \sim D^t}(g\left(h_{k,y}^t(\overline{\mathbf{x}}_i)\right) \neq l_i)$;

      if $\overline{\epsilon}_{k,y}^t > T_{Rmax}$ *or* $\overline{\epsilon}_{k,y}^t < T_{Rmin}$ **then**

        $\overline{\alpha}_{k,y}^t = 0$;

      else

        $\overline{\alpha}_{k,y}^t = \frac{1}{2} ln(\frac{1-\overline{\epsilon}_{k,y}^t}{\overline{\epsilon}_{k,y}^t})$;

    $\overline{h}_y^t = \sum_{k=1}^{N_{sv}^t} \overline{\alpha}_{k,y}^t h_{k,y}^t =< \overline{\mathbf{x}}, \sum_{k=1}^{N_{sv}^t} \overline{\alpha}_{k,y}^t \phi_{k,y}^t > + \sum_{k=1}^{N_{sv}^t} \overline{\alpha}_{k,y}^t b_{k,y}^t \equiv < \overline{\mathbf{x}}, \boldsymbol{\Phi}_y^t > + \Psi_y^t$;

  Set $\overline{h}^t : \overline{\Theta} \rightarrow [0, 1]$, given by $\overline{h}^t(\overline{\mathbf{x}}, y) = f\left(\overline{h}_y^t(\overline{\mathbf{x}})\right) = f\left(< \overline{\mathbf{x}}, \boldsymbol{\Phi}_y^t > + \Psi_y^t\right)$;

  Compute:

$$\epsilon^t = \frac{1}{2} \sum_{i=1}^{N} D^t(i) \left(1 - \overline{h}^t(\overline{\mathbf{x}}_i, y_i) + \sum_{y \neq y_i} q_{i,y}^t \overline{h}^t(\overline{\mathbf{x}}_i, y)\right);$$

  if $\epsilon_t > 0.5$ **then**

    break;

  Calculate AdaBoost.M2 weights: $\alpha^t = \frac{1}{2} \ln\left(\frac{1-\epsilon^t}{\epsilon^t}\right)$;

  for $i = 1, ...N$ *and* $y \in Y - \{y_i\}$ **do**

    Update: $w_{i,y}^{t+1} = w_{i,y}^t exp(-\alpha^t(1 - \overline{h}^t(\overline{x}_i, y_i) + \overline{h}^t(\overline{x}_i, y)))$;

Normalize $\widetilde{\alpha}^t = \alpha^t / \sum_{j=1}^T \alpha^j$, $t = 1, 2, \dots, T$;

for $i = 1, ...$ *to* m **do**

$$|\Upsilon_{i,y}| = \left| \sum_{t=1}^{T} \widetilde{\alpha}^t \frac{\Phi_{i,y}^t}{z_{max,y}^t - z_{min,y}^t} \right|, y \in Y; \tag{18}$$

Compute $v(i) = \max_{y \in Y} \{|\Upsilon_{i,y}|\}$, $i = 1, 2, \dots, m$;

Sort discriminant weights: $v(1) \geq v(2) \geq \dots v(m)$;

Select texture feature following $v(i)$;

Output: Discriminant texture directions: $\overline{\mathbf{q}}_1, \overline{\mathbf{q}}_2, \cdots, \overline{\mathbf{q}}_m$.

Algorithm 2: HL Procedure: Build Hyperplane List

Input: Labeled samples $X = \{(\mathbf{x}_i, y_i), i = 1, 2, \cdots, M\}$; Samples probability distribution $D(\mathbf{x}_i)$; Parameter $0 < \tau, \mu \leq 1$;
Select $\mathcal{J}$ so that, $\sum_{j \in \mathcal{J}} D(x_j) \leq (1 - \mu)$;
Select $(\mathbf{x}_i, y_i); i \in \mathcal{J}$, and define $D^* = D_{\mathcal{J}}$;
Compute the weighted data $X^* = \{(D_i^* \cdot \mathbf{x}_i, y_i), i = 1, 2, \cdots, M\}$
Compute KSVM hypersurface using X^* in equation (4);
Normalize and sort non null Lagrange multipliers β_i such that $\beta_1 > \beta_2 > \ldots > 0$
Let $sv < n$ and the list $L_{sv} = \{(\mathbf{x}_i, \beta_i)\,;\ \sum_{i=1}^{sv} \beta_i \leq \tau\}$,
Calculate $\boldsymbol{\phi}^j = \nabla_{\mathbf{x}} F(\mathbf{x}_j)$, $b^j = -\langle \nabla_{\mathbf{x}} F(\mathbf{x}_j), \mathbf{x}_j \rangle$, for $(\mathbf{x}_j, \beta_j) \in L_{sv}$ and F defined in expression (4),

Output: Tangent hyperplane list $(\boldsymbol{\phi}^j, b^j),\ j = 1, 2, \ldots |L_{sv}|$

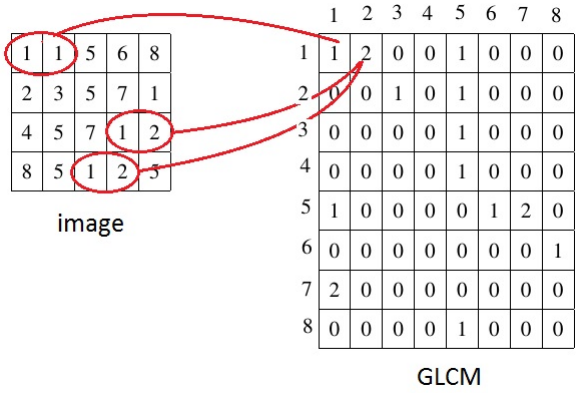


Figure 3: Process to create GLCM for image I, with $L = 8$ in expression (19).

Haralick's Descriptors	
Energy	$\sum_i \sum_j (GLCM(i, j, \Delta x, \Delta y, I))^2$
Contrast	$\sum_i \sum_j \lvert i - j \rvert^2 GLCM(i, j, \Delta x, \Delta y, I)$
Correlation	$\frac{\sum_i \sum_j (i - \mu_x)(j - \mu_y) GLCM(i,j,\Delta x,\Delta y,I)}{\sigma_x \sigma_y}$
Homogeneity	$\frac{\sum_i \sum_j GLCM(i,j,\Delta x,\Delta y,I)}{1 + \lvert i-j \rvert}$
Entropy	$\sum_i \sum_j GLCM(i, j, \Delta x, \Delta y, I) log(GLCM(i, j, \Delta x, \Delta y, I))$

Table 1: Five texture features

1. According to Arvis *et al.* (2011), these descriptors are adequate to give good results in classification task.

In this work, for each Haralick's Descriptors presented in Table (1), we consider the offsets $(\Delta x, \Delta y) \in \{(1, 0), (1, 1), (0, 1), (-1, -1)\}$, composing a feature space with 20 descriptors. Besides, $GLCM\,(i, j, \Delta x, \Delta y, I)$ that is defined by expression (19), is normalized such that the sum of its elements is equal to one Haralick *et al.* (1973). In the above table, μ_x and μ_y are the horizontal and vertical mean while σ_x and σ_y denote the horizontal and vertical standard

deviations, defined by expressions (20)-(21):

$$\mu_x = \sum_i \sum_j iGLCM\,(i,j,\Delta x,\Delta y,I) \quad and \quad \mu_y = \sum_j \sum_i jGLCM\,(i,j,\Delta x,\Delta y,I) \tag{20}$$

$$\sigma_x = \sqrt{\sum_{i,j}(i-\mu_x)^2 GLCM\,(i,j,\Delta x,\Delta y,I)} \quad and \quad \sigma_y = \sqrt{\sum_{i,j}(j-\mu_y)^2 GLCM\,(i,j,\Delta x,\Delta y,I)} \tag{21}$$

5 COMPUTATIONAL EXPERIMENTS

In this section we perform experiments using the granite image database described in Bianconi *et al.* (2015). It contains 25 classes with 40 images per class, composing a total of 1000 full color images with spatial resolution 1500×1500. There are 100 tiles in standard orientation and other 900 created by rotations of 10°, 20°, $\cdots$, 80° and 90° angles. We take 5 classes of the standard set to perform the experiments (see Figure 4). The high spatial resolution of each image allows to subdivide it into smaller blocks that keep the texture patterns. So, we convert each image to gray scale and subdivide it into blocks using crop windows with size 100×100 and 50×50, generating new databases that we call $DB100$ and $DB50$, with 1125 and 4500 images, respectively. Next, in order to save memory allocation along the algorithms execution, we take 500 block images (100 for each class) for training and 500 for test (100 for each class) from both $DB100$ and $DB50$ databases. Besides, Haralick's descriptors are obtained considering 32 gray levels (see section 4).

Figure 4: Samples from database representing the 5 classes of granite images used in the experiments.

In the following, we compare our approach with the Fisher criterion discriminant technique Zhu (2006), and a variation of the Multi-Class LDA-DPCA Filisbino *et al.* (2016) computed as follows: if we compute the LDA in the feature space, we get $N-1=4$ hyperplane directions $\phi^i_{lda} \in \mathbb{R}^m$, $i = 1,2,3,4$. Consequently, we obtain in this case a LDA weight matrix $\phi^{i,y}_{lda}$, which can be processed according to lines 27-28 of Algorithm 1, by just replacing $\Upsilon_{i,y}$ by $\phi^{i,y}_{lda}$. The obtained global discriminant weights are named Multi-Class LDA-DFA in the remaining of this paper. Also, we compare the MNDFA with the Multi-Class.M2 DPCA, proposed in Filisbino *et al.* (2016), where we replace the PCA features by the Haralick's descriptors. So, we call the obtained variation as MDFA in what follows. The MNDFA is a nonlinear version of MDFA. That is why we must compare these methods.

Let N be the number of classes, N_y the number of elements of class y, $\widehat{x}_y$ the average of the samples belonging to class y , $\widehat{x}$ the average of all the samples and $\widehat{x}_{y_i}$ the average of the class corresponding to the ith sample. With these elements, the Fisher criterion postulates that the larger is the value of W_j^{Fisher} computed by:

$$W_j^{Fisher} = \frac{\sum_{y=1}^{N} N_y \cdot \left(\widehat{\overline{x}}_{y;j} - \widehat{\overline{x}}_j\right)^2}{\sum_{i=1}^{N} \left(\overline{x}_{i;j} - \widehat{\overline{x}}_{y_i;j}\right)^2}, \tag{22}$$

then more discriminant is the $\overline{\mathbf{q}}_j$ feature direction for samples classification.

The considered multi-class discriminant analysis techniques are fed with the Haralick features that are ranked according to their discriminant weights. In this stage, we use the KNN classifier, with $K = 5$ and Euclidian distance, to analyze the performance of the discriminant feature spaces obtained.

Figure 5 presents a comparison between MNDFA and MDFA methodologies with images of the $DB100$ database for the five-class classification problem showed in Figure 4. In order to confront both

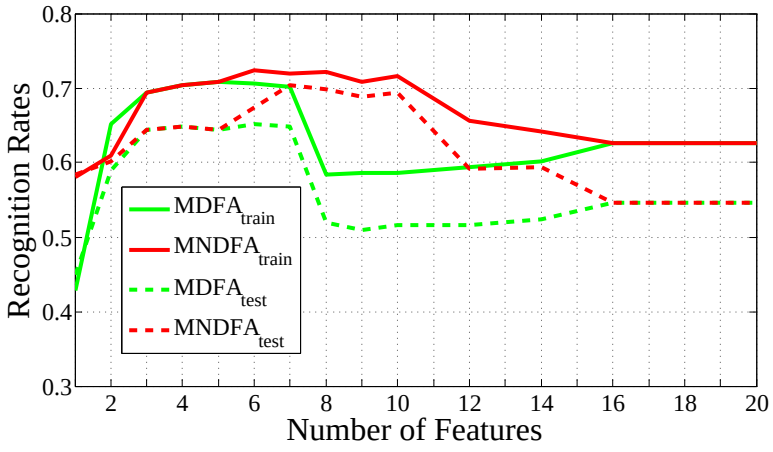
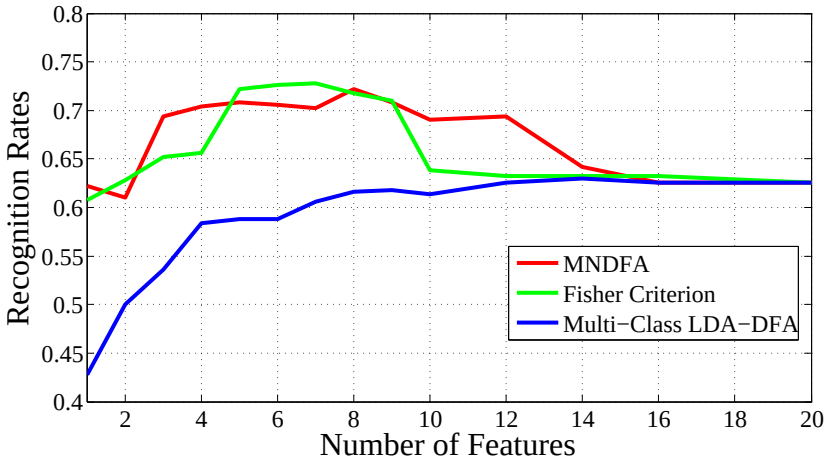


Figure 5: Comparing MNDFA and MDFA.

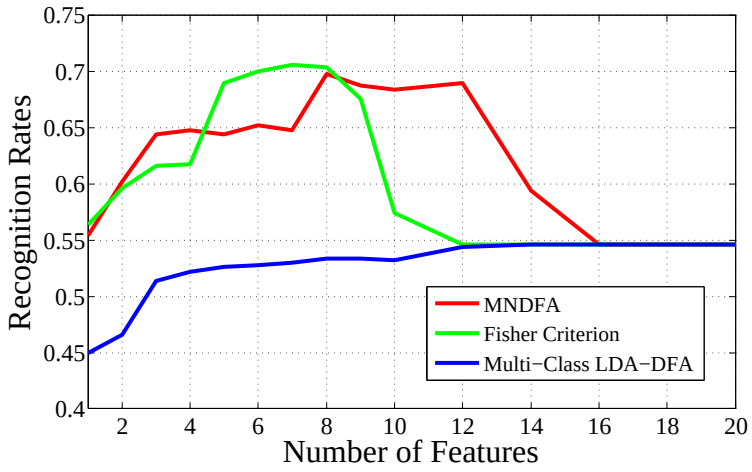
techniques we decided to instantiate them with the same parameters. So, to perform the training of MNDFA and MDFA, the KSVM and SVM relaxation parameter is set to $C = 0.001$. In function of this parameter, to weaken the SVM (see Algorithm 1 Filisbino *et al.* (2016)) without numerical problems, we experimentally observe that we should set $\mu \geq 0.5$ to compute the separating hyperplanes by MDFA. Besides, the pseudo-loss from Algorithm Filisbino *et al.* (2016) (line 10), reached the threshold for $T = 15$. Therefore, we take $T = 15$, $\mu = 0.5$ for both MNDFA and MDFA. The solid lines present recognition rates when taking the same samples to perform discriminant analysis and classification using 10-fold cross validation and KNN. On the other hand, the dashed line present results when using different sample sets for discriminant analysis and recognition rates computation. The results show that MNDFA outperforms or is equal the MDFA methodology in both tests, except for range $2 \leq k \leq 3$. Therefore, MNDFA is more competitive than MDFA for comparison with other discriminant techniques.

In order to observe the behavior of MNDFA as well as improve the recognition rates, we performed the remaining experiments using the Algorithm 1 with: $T = 30$, $\mu = 0.3$ to perform the weaken process of KSVM, and $\tau = 0.99$ in order to take representative support vectors in the Algorithm 2. Also, due to AdaBoost requirements, we consider the classification of the corresponding tangent hyperplanes by setting $T_{Rmin} = 0.5$ and $T_{Rmax} = 0.8$ which discards another subset of support vectors (line 12 of Algorithm 1). In average, we take 142 support vectors for the experiments performed in this section.

The Figure 6.(a)-(b) shows the average recognition rates where we also used the 10-fold cross validation experiments with KNN, using the Haralick feature space oriented through the discriminant techniques, for the five-class classification problem pictured in Figure 4, using images of the $DB100$ database. We must notice that the features selected by the MNDFA and Fisher criterion allow higher recognition rates using few texture features. The Figure 6.(a) represent the accuracy results when taking the same set for discriminant analysis and for classification using KNN. The result shows that our methodology is more efficient than the other ones in the interval $3 \leq k \leq 5$ and in the interval $8 \leq k \leq 15$. For $5 < k < 8$ the recognition rates of MNDFA are below the Fisher criterion but above the



(a)



(b)

Figure 6: Recognition rates for texture features ranked by MNDFA, Fisher criterion and a Multi-Class LDA-DFA when varying the number k of features ($DB100$ database).

Multi-Class LDA-DFA ones. In Figure 6.(b) the recognition rates were computed with different sets for discriminant analysis and classification using the database $DB100$. The MNDFA is better or equal than Multi-Class LDA-DFA for all ranges and when compared with Fisher criterion, MNDFA outperforms it for ranges $2 < k < 5$ and $8 < k < 16$.

The Figure 7 shows the average recognition rates for image database $DB50$. We can notice a decrease in the recognition rates for all methodologies, achieving no more than 60% of accuracy. From Figure 7, we observe that the MNDFA is better than the other methodologies in the range $1 < k < 5$ and $10 < k < 16$. However for $5 \leq k \leq 10$ Fisher criterion outperforms the MNDFA and Multi-Class LDA-DFA.

Regarding to the parameter σ in the RBF kernel (expression (6)), we follow the literature Bishop (1997) and estimate its value using the class means $\widehat{\mathbf{x}}_1$ and $\widehat{\mathbf{x}}_2$, according to:

$$\sigma = \frac{\|\widehat{\mathbf{x}}_1 - \widehat{\mathbf{x}}_2\|}{\sqrt{2}}, \tag{23}$$

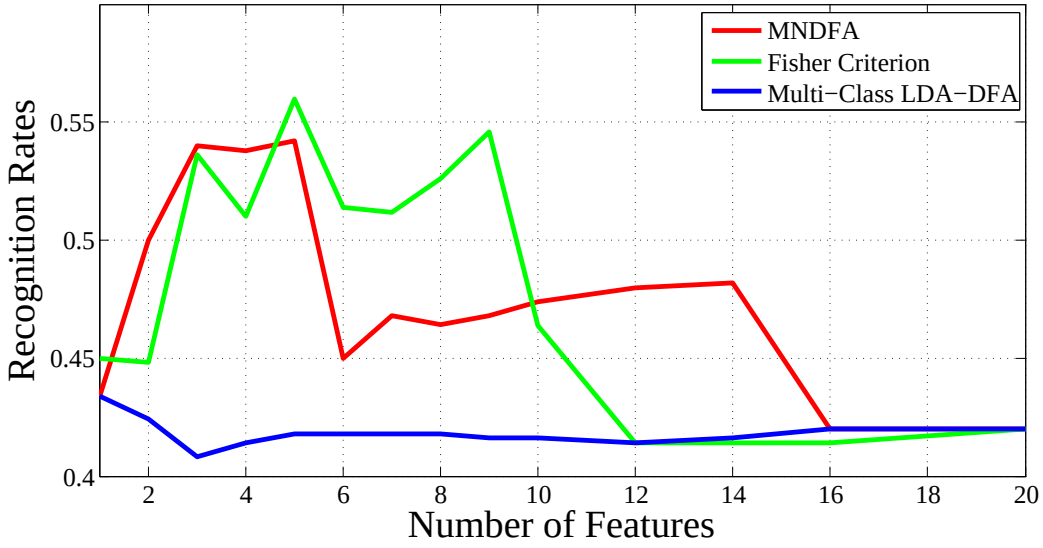


Figure 7: Recognition rates of the considered techniques for database $DB50$ when varying the number k of features.

where $\widehat{\mathbf{x}}_1$ is the mean of -1 samples in the set defined in expression (7) while $\widehat{\mathbf{x}}_2$ is the mean of the $+1$ samples in the same set. In order to experimentally analyse the influence of this parameter in the recognition rates obtained by MNDFA discriminant components, we set $\sigma_1 = \sigma/3$, $\sigma_2 = \sigma$ and $\sigma_3 = 3\sigma$, with σ set as above. The Figure 8 allows to check the effect of σ in MNDFA performance.

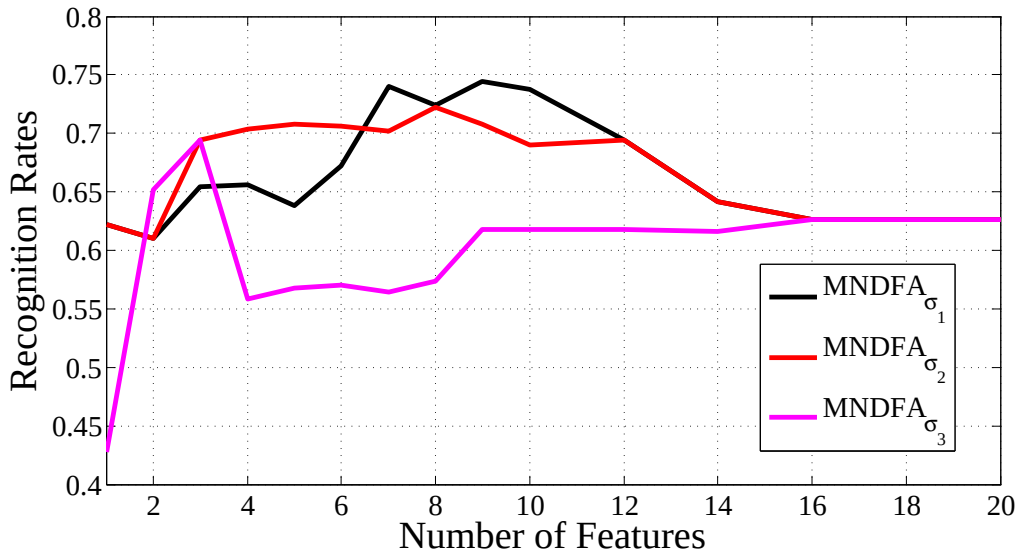


Figure 8: Influence of σ parameter in the $DB100$ database.

We notice from Figure 8 that, for $k < 2$ the recognition rates of MNDFA$_{\sigma_2}$ is equal MNDFA$_{\sigma_3}$ or higher than MNDFA$_{\sigma_1}$. For $2 \leq k \leq 3$ we notice that MNDFA$_{\sigma_3}$ > MNDFA$_{\sigma_2}$ > MNDFA$_{\sigma_1}$. On the other hand, in the range $3 < k < 7$ we have MNDFA$_{\sigma_2}$ > MNDFA$_{\sigma_1}$ > MNDFA$_{\sigma_3}$. Finally, for

$k > 7$ we see that $\text{MNDFA}_{\sigma_1} \geq \text{MNDFA}_{\sigma_2} \geq \text{MNDFA}_{\sigma_3}$. Therefore, the behavior of MNDFA respect to σ variation depends on the number of features considered. Specifically, from the performed tests we can not say that the MNDFA performance is a monotonically increasing (or decreasing) function of σ. However, we can improve the performance in a specific range through the variation the σ.

All experiments were executed in an Intel Core i7 computer, CPU 3.30GHz, with 12GB of RAM, video card NVidia GeForce 9800 GTX, and operating system Windows 7 64bits, using Matlab (Release 12) software. In such computational resource, the CPU time for the MNDFA computation for the discriminant analysis in the $DB100$ database was nearby 25 minutes.

6 CONCLUSION AND FUTURE WORKS

This paper presents the MNDFA algorithm, a generalization of the original MDPCA for ranking texture features in nonlinear classification problems. The basic loop of methodology is composed by the computation of KSVM in the feature space, followed by a nested AdaBoost procedure. The texture experiments show that, in general, for $DB100$ and $DB50$, the features selected by MNDFA allow higher or equal recognition rates than counterpart ones when considering number of discriminant features $k \leq 4$ and $k \geq 10$. The reduction of crop window size causes a decrease in recognition rates in all tested methodologies. The sigma parameter influences the recognition rates but without a defined pattern. Further work is being undertaken to improve MNDFA performance and to reduce its computational cost.

REFERENCES

Arvis, Vincent, Debain, Christophe, Berducat, Michel, & Benassi, Albert. 2011. Generalization of the cooccurrence matrix for colour images: application to colour texture classification. *Image Analysis & Stereology*, **23**(1), 63–72.

Bianconi, F., Bello, R., Fernndez, A., & Gonzlez, E. 2015. On comparing colour spaces from a performance perspective: Application to automated classification of Polished natural stones. *Lecture Notes in Computer Science*, 71–78.

Bianconi, Francesco, González, Elena, Fernández, Antonio, & Saetta, Stefano A. 2012. Automatic Classification of Granite Tiles Through Colour and Texture Features. *Expert Syst. Appl.*, **39**(12), 11212–11218.

Bishop, Christopher M. 1997. *Neural Networks for Pattern Recognition*. New York, NY, USA: Oxford University Press, Inc.

Cunningham, John P., & Ghahramani, Zoubin. 2015. Linear Dimensionality Reduction: Survey, Insights, and Generalizations. *J. of Mach. Learn. Research*, **16**, 2859–2900.

Felix, A., Oliveira, M., Machado, A., & Raniery, J. 2016 (Octuber). Using 3D Texture and Margin Sharpness Features on Classification of Small Pulmonary Nodules. *In: Graphics, Patterns and Images (SIBGRAPI), 2016 26th SIBGRAPI - Conference on*.

Filisbino, T.A., Giraldi, G.A., & Thomaz, C.E. 2015a. Comparing Ranking Methods for Tensor Components in Multilinear and Concurrent Subspace Analysis with Applications in Face Images. *IJIG-International Journal of Image and Graphics*, **15**.

Filisbino, T.A., Leite, D., Giraldi, G.A., & Thomaz, C.E. 2015b (Nov). Multi-Class Discriminant Analysis Based on SVM Ensembles for Ranking Principal Components. *In: 36th Ibero-Latin Am. Cong. on Comp. Meth. in Eng. (CILAMCE)*.

Filisbino, T.A., Giraldi, G.A., & Thomaz, C.E. 2016 (Octuber). Ranking Principal Components in Face Spaces Through AdaBoost.M2 Linear Ensemble. *In: Graphics, Patterns and Images (SIBGRAPI), 2016 26th SIBGRAPI - Conference on*.

Filisbino, Tiene A., Giraldi, Gilson A., Thomaz, Carlos Eduardo, Barros, Bruno M. N., & da Silva, Millena B. Ranking Texture Features Through AdaBoost.M2 Linear Ensembles for Granite Tiles Classification. *In: Xth EAMC, Petropolis, Brazil, Feb. 1-3, 2017*.

Freund, Yoav, & Schapire, Robert E. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, **55**(1), 119–139.

Garcia, Elkin, & Lozano, Fernando. 2007. Boosting Support Vector Machines. *Pages 153–167 of: Proceedings of International Conference of Machine Learning and Data Mining (MLDM'2007)*. Leipzig, Germany: IBal publishing.

Giraldi, G. A., Rodrigues, P. S., Kitani, E. C., & Thomaz, C. E. 2008. Dimensionality Reduction, Classification and Reconstruction Problems in Statistical Learning Approaches. *Revista de Informatica Teorica e Aplicada (RITA)*, **15**(1), 141–173.

Haralick, R. M., Shanmugam, K., & Dinstein, I. 1973. Texture features for image classification. *IEEE Transaction System Man Cybernet SMC-3*, **6**, 610–621.

Hastie, T., Tibshirani, R., & Friedman, J.H. 2001. The Elements of Statistical Learning. *Springer*.

Loog, Marco, Duin, R. P. W., & Haeb-Umbach, R. 2001. Multiclass Linear Dimension Reduction by Weighted Pairwise Fisher Criteria. *IEEE Trans. Patterns Anal. Mach Intell.*, **23(7)**, 762–766.

Thomaz, C. E., & Giraldi, G. A. 2010. A new ranking method for principal components analysis and its application to face image analysis. *Image Vision Comput.*, **28**(6), 902–913.

Turk, M., & Pentland, A. 1991. Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, **3**, 71–86.

Vapnik, Vladimir N. 1998. *Statistical Learning Theory*. John Wiley & Sons, INC.

Zayed, Nourhan, & Elnemr, Heba A. 2015. Statistical Analysis of Haralick Texture Features to Discriminate Lung Abnormalities. *Journal of Biomedical Imaging*, **2015**(Jan.), 12:12–12:12.

Zhou, Zhi-Hua. 2012. *Ensemble Methods: Foundations and Algorithms*. 1st edn. Chapman & Hall/CRC.

Zhu, M. 2006. Discriminant Analysis with common principal components. *Biometrika*, **93(4)**, 1018–1024.

Zhu, M., & Martinez, Aleix M. 2006 (June). Selecting Principal Components in a Two-Stage LDA Algorithm. *Pages 132–137 of: CVPR'06*.