Optimized Nonlinear Discriminant Analysis (ONDA) for Supervised Pixel Classification

Jia Guo, Hu Huang, Cheng Chen, and Gustavo K. Rohde, *Member, IEEE*

***Abstract*—Filter bank-based methods for pixel classification are attractive due to the potential of fast implementation with convolution operations. The design of optimal filter sets, however, is a challenging task given the nonlinear aspects of the problem. This letter extends the well known linear discriminant analysis method into a novel framework for local texture feature discrimination tasks. It proposes a mixture of linear models as a nonlinear classifier, where a number of filters are optimized locally by minimizing the prediction error. Through these filters, the 'best separable' features are selected. Experiments performed on two standard texture databases show that our method produces results which are comparable to state-of-the-art techniques while at the same time maintaining low computational complexity.**

***Index Terms*—Feature selection, filter bank design, pixel classification.**

I. Introduction

CLASSIFICATION of images via linear shift invariant filtering operations has received considerable attention in the past decades. It has been considered as a common denominator for most signal processing approaches in texture classification problems, where a bank of filters is utilized to extract texture features. The filter responses are subsequently used to classify individual pixels (see [1], [2] for examples) or image patches (see [3] for an example). Over the years, several methods for texture segmentation using filter banks have been described. A key distinction that can be made between them is whether they utilize training data (supervised) or not (unsupervised). Generally speaking, supervised methods allow for optimization and hence at times can be especially adapted to a given problem or texture type. Consequently, in certain cases, they can allow for increased classification accuracies in comparison to unsupervised approaches (e.g. wavelets, Gabor-like filters, etc.) [4].

Manuscript received June 14, 2013; revised August 06, 2013; accepted August 07, 2013. Date of publication August 21, 2013; date of current version September 30, 2013. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Kjersti Engan.

J. Guo, H. Huang and C. Chen are with the Center for Bioimage Informatics, Department of Biomedical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: guojia514@gmail.com; hwangtiger@gmail.com; chengchen@cmu.edu).

G. K. Rohde is with the Center for Bioimage Informatics, Department of Biomedical Engineering, Department of Electrical and Computer Engineering, Lane Center for Computational Biology, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: gustavor@cmu.edu).

This letter contains supplemental Appendix material available online at http://ieeexplore.ieee.org.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LSP.2013.2278976

Over the past decades, several approaches have been developed for supervised filter design for texture classification. Works on filter design utilizing principles such as prediction error, energy separation, principal component analysis, are reviewed in [4]. More recent works have borrowed concepts from high dimensional statistics and machine learning, so as to treat the task of optimal filter design as a feature selection problem. Given a subspace based on which filters can be built using linear combination of basis vectors, the set of filters separating two classes can be obtained by finding the linear combinations of the basis vectors that 'best separate' two classes in feature (filter response) space. The algorithm described in [5], for example, utilizes the Mahalanobis distance between two or more classes as a measure of separation between them, while the method in [6] utilizes the F-test criterion. Once the set of filters is selected, a classifier (e.g. k-nearest neighbors (kNN) classifier, support vector machines (SVM), etc.) is then trained and any pixel in a test image can be classified by first convolving that image with the selected optimal filters. Once the 'features' have been extracted, each pixel is classified by applying the estimated classification function.

Here we expand on these works by describing a method that optimizes a set of filters in a given subspace by explicit minimization of an especially tailored risk function. The minimization is initialized using the well known linear discriminant analysis (LDA) method [7, p.186–189], and subsequently iterated utilizing the gradient descent minimization method. The minimization can be made efficient even for large images, given that both the evaluation of the risk function, as well as its gradient, can be performed using convolutions. In addition, the method we describe not only yields optimal filters, but also provides a computationally efficient classifier as a natural by-product. Below we show that the method can achieve similar accuracies such as the methods described in [5], [6], albeit at a much reduced computational cost.

II. Methods

The LDA technique is one of the most popular methods for classifying linearly separable multidimensional data. For images the linear operation associated with the LDA technique can be written as a convolution between an input image g (a discrete array of size $M_x \times M_y$), and the optimal filter f (the linear direction computed by the LDA procedure). Mathematically, the classification operation applied to each pixel in an input image g can be written as

$$\tilde{C}_{\mathrm{LDA}}(g;\mathbf{k}) = (f * g)[\mathbf{k}] + b \gtrless 0 \tag{1}$$

where $\mathbf{k}$ denotes a 2D image coordinate, f and b are given by the LDA procedure, and $*$ denotes the 2D convolution operation. Note that for classifying color images (as we do below), the

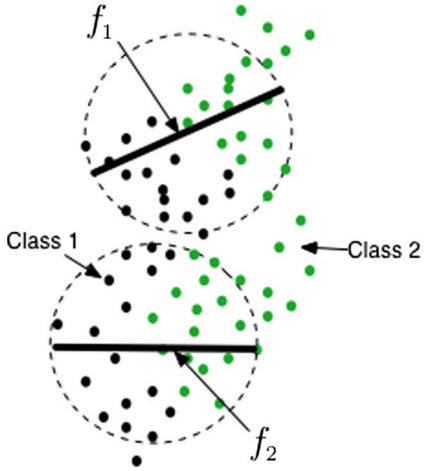

Fig. 1. Our method attempts to classify pixel intensities through a set of filters $(f_1, \cdots, f_{N_f})$ that act as local linear classifiers. Each point in this diagram corresponds to a pixel's feature vector, comprised of a local sub-window.

operation above is repeated channel (color) by channel, with filter g being a multichannel filter. Testing whether $\tilde{C}_{\mathrm{LDA}}(g;\mathbf{k})$, the classification output, is positive or negative at each pixel tells one whether the particular pixel $\mathbf{k}$ belongs to class 1 or 2.

While simple to use and extremely effective at times, the framework above often fails to separate textures of common interest in image processing and computer vision (i.e. Fig. 2). Therefore, our goal here is to extend the framework above so that it can be used to estimate nonlinear separation boundaries between two or more classes. The idea is to build on the LDA framework (i.e. utilize it as a starting point) by computing several digital filters $f_1, \cdots, f_{N_f}$ that act as locally linear separators in 'feature space.' By feature space of a given pixel $\mathbf{k}$, we mean the set of pixel values that form the local neighborhood (an array of the same size of filters $f_1, \cdots, f_{N_f}$). As a result, our method is able to deal with the nonlinear classification problem by utilizing a local LDA method, which we denote as the optimized nonlinear discriminant analysis (ONDA). Fig. 1 shows a illustration of this idea.

In our approach, each filter in $f_1, \cdots, f_{N_f}$ is modeled as a linear combination of known 'basis' vectors (filters) $\phi_1, \cdots, \phi_{N_a}$ (described below) via

$$f_i = \sum_{j=1}^{N_a} a_{i,j}\phi_j, \tag{2}$$

where coefficients $a_{i,j}$, $i = 1, \cdots, N_f$, $j = 1, \cdots, N_a$. Our classifier model is then written as

$$\tilde{C}_{\mathbf{c},\mathbf{r},\mathbf{a},\mathbf{b}}(g, v; \mathbf{k}) = \sum_{i=1}^{N_f} \Psi^i_{c_i,r_i}(g, v_i; \mathbf{k}) \left\{(f_i * g)[\mathbf{k}] + b_i\right\} \tag{3}$$

where $c_i, r_i, a_{i,j}, b_i$ are parameters to be optimized. We note that $\Psi^i_{c_i,r_i}(g, v_i; \mathbf{k})$ is a function meant to play the role of an 'influence field' for linear filter f_i, as depicted by the large circles in Fig. 1. In our implementation we define it as

$$\Psi^i_{c_i,r_i}(g, v_i; \mathbf{k}) = c_i e^{-r_i\left((g^2 * \chi)[\mathbf{k}] - 2(g * v_i)[\mathbf{k}] + \|v_i\|^2\right)} \tag{4}$$

where v_i is a sub-window of pixel intensities (patch) corresponding to the center location (in feature space) over which filter f_i acts, χ corresponds to a filter with each entry set to 1, and $\|v_i\|^2$ corresponds to the sum of squared values of patch v_i. Here f_i, v_i, ϕ_j, and χ are arrays whose sizes are equal. In this model, scalar r_i corresponds to the radius of influence of filter f_i, scalar c_i acts as the mixture weight which satisfies $0 \leq c_i \leq 1, \sum_{i=1}^{N_f} c_i = 1$, and v_i is computed as described below in the *Training procedure*.

In summary, for a fixed set of input parameters $\mathbf{c}, \mathbf{r}, \mathbf{a}, \mathbf{b}$, as well as v_i, the model defined in (3) can be evaluated at each pixel $\mathbf{k}$ in an input image g. Testing whether the model is greater than or less than zero indicates whether the pixel is likely to correspond to class 1 or 2, for example.

We also note that the modeling procedure described above also depends on the subspace formed by the span of $\phi_1, \cdots, \phi_{N_a}$. Rather than choosing an *a priori* set of filters, we utilize training data to choose N_a linearly independent filters. In the computations shown below, for a given dataset, we utilized the principal component analysis (PCA) technique to choose the N_a filters associated with the largest N_a eigenvalues of the covariance matrix of a randomly chosen pixel windows (neighborhoods) from the dataset to be analyzed, although we note that alternative methods for choosing N_a linearly independent filters as described in [6] and [8].

A. Training Procedure

In a supervised learning problem, a certain amount of labeled data is used to estimate the parameters for the classification function that is to be used in the testing procedure. In our case, taking as an example a binary class problem, labeled training data consists of image pixels that have class indexes ($+1$ or -1) assigned to them. Mathematically, we make use of a function $C(g;\mathbf{k})$ that expresses this. Denote our error (risk) function as $E(\alpha)$, where $\alpha = \{\mathbf{c}, \mathbf{r}, \mathbf{a}, \mathbf{b}\}$ represents the set of unknown parameters. The goal of the training procedure is then to find the unknown parameters such that the following error functional is minimized

$$E(\alpha) = \sum_{\mathbf{k}} \left(I_\epsilon\left(\tilde{C}_{\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{r}}(g, v; \mathbf{k})\right) - C(g;\mathbf{k})\right)^2 \tag{5}$$

where $I_\epsilon(x) = \tanh(x/\epsilon)$, and the sum is taken over all pixels in input labeled image g. If more than one image is available for training, the sum above is repeated to include each training image.

The filter design procedure described below takes as input a labeled image g (an array of size $M_x \times M_y$, which can also contain color information), where the class of each pixel is given. Let $C(g)$ represent the set of labels ($+1$ or -1 in the binary classification case) for the input image g. That is, $C(g)$ is also an array of size $M_x \times M_y$, where each entry is populated by a $+1$ or a -1. The output of our procedure is a set of linear shift invariant digital filters $f_1, \cdots, f_{N_f}$ that are optimal in the sense of allowing for feature space separation between the multiple classes present, together with their corresponding influence fields $\Psi^1, \cdots, \Psi^{N_f}$. In addition, the procedure also outputs a nonlinear classification function (classifier), denoted as $\tilde{C}_\alpha(g, v; \mathbf{k})$, where α are optimally selected parameters, that can accurately predict the class of each pixel in any image g. Our ONDA framework in binary classification case is described as Algorithm 1. We later expand the framework to multiple classes.

The training algorithm is composed of the following steps: obtaining model vectors, computing centers of influence regions, and model optimization.

Obtaining Model Vectors ϕ_j, $j = 1, \cdots, N_a$: Given input labeled images $g_1, \cdots, g_{N_{train}}$ for training, we apply the PCA procedure to the set of local window neighborhoods obtained from the training data. The eigenvectors associated with the N_a largest eigenvalues are chosen as the model vectors ϕ_j, $j = 1, \cdots, N_a$.

Computing Centers of Influence Regions v_i, $i = 1, \cdots, N_f$: The goal here is to compute the center 'location' of v_i, $i = 1, \cdots, N_f$, in feature space over which a given filter f_i will act. In this work we select each v_i (a patch of the same size as f_i and ϕ_j) to be associated with regions near the feature space boundary between classes of the feature space. To that end we use a preliminary feature selection criterion based on the 'Neighborhood Entropy' method described in [9]. The idea in this method is to select points near the decision boundaries between two classes as regions of interest. It does so by making use of a threshold applied to a neighborhood (in feature space) entropy measure. In our case we further modify the technique by selecting only pixels which were misclassified with the LDA procedure as potential data points for influence regions. Briefly, for each feature vector in the training set its Q nearest neighbors (in this work we set $Q = 11$ empirically), and compute the entropy of the probability of its neighbors belonging to each class. Pixels for which this entropy is greater than 1.75 (this value chosen empirically in our work) are selected. The K-means algorithm (with $k = N_f$) is then used to select the feature vectors to be set as v_i, $i = 1, \cdots, N_f$.

Algorithm 1 ONDA for Binary Classification

Input

A. Training images $\{g_1, \cdots, g_{N_{train}}\}$
B. Image labels $C(g_i) \in \{+1, -1\}, i = 1, \cdots, N_{train}$

Training steps

1) Generate PCA 'basis' filters $\{\phi_1, \cdots, \phi_{N_a}\}$
2) Compute influence regions
3) Initialize parameters for optimization
4) Optimization by gradient descent $\alpha^{(\tau+1)} = \alpha^{(\tau)} - \eta \bigtriangledown E(\alpha^{(\tau)}); \alpha = \{c, r, a, b\}$

Outputs

A. Optimal digital filters $\{f_1, \cdots, f_{N_f}\}$
B. Corresponding influence fields $\{\Psi^1, \cdots, \Psi^{N_f}\}$
C. Nonlinear classifier $\tilde{C}_\alpha(g, v; \mathbf{k})$

Model Optimization: Once the filter basis vectors ϕ_j and the centers of each influence regions v_i are known, we optimize the cost function (5) via a standard gradient descent procedure. That is, given an initial guess for parameter vector $\alpha^{(0)}$, its subsequent iterations are computed via $\alpha^{(\tau+1)} = \alpha^{(\tau)} - \eta \bigtriangledown E(\alpha^{(\tau)})$ with an approximate line minimization for computing η at each iteration. As for the initial parameters, we set $c_i = 1/N_f$, and $r_i = r_0$, where r_0 is the optimal radius selected by the least squares cross-validation of the supervised kernel density estimation in training feature space. As for the coefficients $a_{i,j}$ and b_i, we initially set these to correspond to the coefficients given by the linear discriminant analysis technique. Finally, we mention that the algorithm above is for binary classification. Extension to problems with more than 2 classes is possible using one of several popular procedures, including one versus one, as well as one versus all classifiers with voting [7, p.182–183].

B. Pixel Classification Procedure

Once the training procedure outline has been completed, one may use discriminating filters $f_i, i = 1, \cdots, N_f$ as feature extraction operations in conjunction with other nonlinear classifiers (e.g. kNN, SVM, etc), as done in [5], [6], for example. In these case the 'features' are the outputs of the filtering operation applied to each sub window. An alternative is to utilize the region of interest framework described above to compute and array $\mathcal{F}_i, \cdots, \mathcal{F}_{N_f}$ of 'weighted' features instead: $\mathcal{F}_i(g, v_i; \mathbf{k}) = \Psi^i_{c_i, r_i}(g, v_i; \mathbf{k})\{(f_i * g)[\mathbf{k}] + b_i\}$. Finally, one may also utilize our nonlinear classifier $\tilde{C}_\alpha(g, v; \mathbf{k})$ to classify the pixels in an arbitrary input image g, with optimal parameter vector α computed as above. As shown below this classifier does not seem to produce significantly lower classification results than several other methods, while its computational complexity is on the order of N_f convolutions with the input image.

III. Experimental Setup

A. Dataset and Experimental Setup

We evaluate the efficacy of the method described above on two well-known texture datasets including Brodatz Textures [10] and VisTex [11]. In the binary-class problems, for each dataset, we randomly selected 10 different textures for the experiments, taking half of the data for training and the other half for testing. The experiments have been processed in MATLAB(R2012a) on a Intel(R) Core(TM) i5-3320M CPU at 2.60 GHz with 8 GB of RAM. We compared the method to other filter bank optimization algorithms: the standard Fisher LDA approach [12], Mahanalobis(MAH) separability-based filter optimization[5], and the F-test score-based filter optimization method of [6]. In these comparisons, training and testing data never overlapped, while the same training and test images were used to compare all methods. To make the comparison uniform, the filter optimization approach of [5] and [6] utilized the same underlying PCA basis filters as used in our method. The kNN classifier was utilized to compare the classification accuracy of the features derived from our method, and the ones derived from [5] and [6]. In the multi-class problems, classification was performed by a "Max-Wins" voting strategy with one-vs.-one method [7, p.182]. In addition, we also compared our method with the following other state-of-the-art texture classification techniques [4], [13]–[16] on the same test suite (a 5-class Brodatz texture image P1 generated by Randen and Husøy [4]) utilized in these works.

B. Results

Fig. 2 shows the ONDA method applied to an image from the VisTex database. Here we see that the linear classification method is not sufficient to discriminate the classes and we are

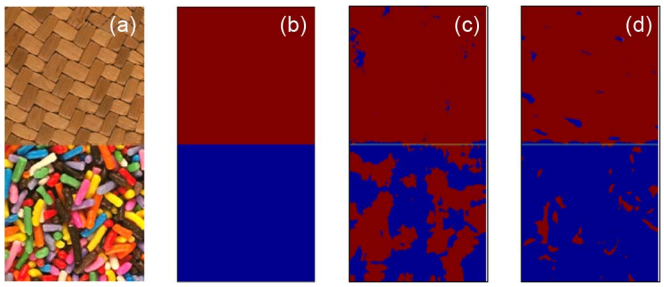

Fig. 2. Example of linearly nonseparable textures classification results by LDA and ONDA. (a) Binary VisTex textures with size 512 × 256. (b) Ground truth. (c) LDA classification results with accuracy 72.58%. (d) ONDA classification results with accuracy 96.84%.

TABLE I
COMPARISON OF CLASSIFICATION RESULTS ON BRODATZ AND VISTEX TEXTURES (EVALUATED ON $2 \times \binom{10}{2} = 90$ PAIRWISE BINARY TEXTURES)

Method	Accuracy(%)	CPU time(s)
Proposed	$\mathbf{97.21 \pm 4.57}$	$\mathbf{1.768}$
ONDA+kNN	96.95 ± 3.32	1957.084
Fisher+kNN [5]	95.29 ± 4.90	2011.023
MAH+kNN [5]	93.31 ± 8.06	1995.020
F-test+kNN [6]	92.56 ± 8.31	1950.447

TABLE II
COMPARISON BETWEEN THE CLASSIFICATION ACCURACIES(%)

Test Image	[4]	[13]	[14]	[15]	ONDA	D2 [16]
P1	92.8	93.3	94.5	96.63	96.72	98.39

able to appreciate the improvements provided by ONDA. Quantitative comparisons with other methods are provided in Table I for the Brodatz database (where we compare ONDA to other filter bank optimization methods), and Table II for the problem P1 (where we compare ONDA to other state-of-the-art pixel classification methods). Supplement Figs. S1, S2, and S3 allow one to visualize ONDA classification results on sample images, as well as to compare ONDA to other methods. We note that in Table II we show the results of ONDA after using the same voting procedure described in [16]), for example.

IV. SUMMARY AND DISCUSSION

We described a nonlinear pixel classification method that, in the binary case, can be applied with $O(kn_q \log n_q)$ operations, with n_q being the number of pixels in the test image and k the number of local centers. This represents significantly fewer operations than all other comparison methods in this letter (including the sparse optimization approach described in [16]). The method works by utilizing the standard LDA approach as a starting point, and then improves upon this initial guess by optimizing a carefully designed cost (risk) function using the standard gradient descent method. Given that the cost function is non convex, the method is prone to local optima. While the method is fast to compute, experiments demonstrated that it does not come at a cost of lower classification accuracies in comparison to several other nonlinear classification methods.

Though the general framework has the advantages outlined above, here we highlight a few limitations. Firstly, as commonly the case in several other works, the filters derived in our method are not rotationally invariant. Secondly, the training procedure can be expensive to compute when numerous training images are available. The run time of the training procedure is dominated by Step 4, whose total cost is in the range of $O(tkn_p \log n_p)$ operations, with n_p the number of pixels in the training set and t the number of iterations of the gradient descent. In our experiments, the iteration number t is in the range between 500 and 1000. In addition, our model is implicitly regularized by the number of influence regions, but does not contain any explicit regularization. Thus over fitting may be possible when numerous influence regions are chosen. Finally, we note that the parameters c_i, $a_{i,j}$, and b_i in (3) are not uniquely specified. Several modifications rendering the model uniquely specified (e.g. regularization) can be devised. In this work we have opted for the model presented due to computational efficiency considerations.

We also mention that it is possible to improve the method in several aspects. Firstly, methods other than the standard PCA could be used to compute the initial 'basis' vectors. In addition, 'influence regions' for each filter were computed similarly to the method described in [9] and then fixed throughout the remaining of the procedure. Once optimization of the risk function achieves a stable point, the regions could be recomputed to better assess boundary values, and the whole procedure could be computed iteratively. Future work will include these and other improvements.

REFERENCES

[1] T. Randen and J. Husøy, "Texture segmentation using filters with optimized energy separation," *IEEE Trans. Image Process.*, vol. 8, no. 4, pp. 571–582, 1999.
[2] M. Unser, "Texture classification and segmentation using wavelet frames," *IEEE Trans. Image Process.*, vol. 4, no. 11, pp. 1549–1560, 1995.
[3] M. Varma and A. Zisserman, "A statistical approach to texture classification from single images," *Int. J. Comput. Vis.*, no. 1–2, pp. 61–81, Apr. .
[4] T. Randen and J. Husøy, "Filtering for texture classification: A comparative study," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 4, pp. 291–310, 1999.
[5] W. Li, K. Mao, H. Zhang, and T. Chai, "Designing compact gabor filter banks for efficient texture feature extraction," in *11th Int. Conf. Control Automation Robotics Vision (ICARCV)*, Dec. 2010, pp. 1193–1197.
[6] R. Jenssen and T. Eltoft, "Eltoft: Ica filter bank for segmentation of textured images," in *ICA 2003 4th Int. Symp. Independent Component Analysis and Blind Signal Separation*, Apr. 1–4, 2003, pp. 827–832.
[7] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag, 2006.
[8] W. Wang, Y. Mo, J. A. Ozolek, and G. K. Rohde, "Penalized fisher discriminant analysis and its application to image-based morphometry," *Pattern Recognit. Lett.*, vol. 32, no. 15, pp. 2128–2135, 2011.
[9] H. Shin and S. Cho, "Pattern selection for support vector classifiers," in *Intelligent Data Engineering and Automated LearningIDEAL 2002*. : Springer, 2002, pp. 469–474.
[10] P. Brodatz, *Textures—A Photographic Album for Artists and Designers*. New York, NY, USA: Dover, 1966.
[11] Color Image Database, VisTex, Media Lab., Mass. Inst. Technol.. Cambridge, MA, USA, 1995.
[12] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugenics*, no. 7, pp. 179–188, 1936.
[13] M. Topi, P. Matti, and O. Timo, "Texture classification by multi-predicate local binary pattern operators," in *Proc. 15th Int. IEEE Conf. Pattern Recognition*, 2000, vol. 3, pp. 939–942.
[14] K. Skretting and J. H. Husy, "Texture classification using sparse frame based representation," *EURASIP J. Appl. Signal Process.*, p. 2006, 2006.
[15] A. Di Lillo, G. Motta, and J. Storer, "Texture classification based on discriminative features extracted in the frequency domain," in *IEEE Int. Conf. Image Processing*, 2007, vol. 2, pp. II-53–II-56.
[16] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Discriminative learned dictionaries for local image analysis," in . *IEEE Conf. Computer Vision and Pattern Recognition, CVPR*, 2008, pp. 1–8, IEEE.