Repositorio Institucional de la Universidad Autónoma de Madrid
https://repositorio.uam.es

Esta es la **versión de autor** del artículo publicado en:
This is an **author produced version** of a paper published in:

IEEE Transactions on Neural Networks and Learning Systems 31.8 (2020): 2752-2763

DOI: http://doi.org/10.1109/TNNLS.2019.2906302

Copyright: © 2020 Institute of Electrical and Electronics Engineers

© 2020 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Deep Least Squares Fisher Discriminant Analysis

David Díaz-Vico and José R. Dorronsoro

***Abstract*— While being one of the first and most elegant tools for dimensionality reduction, Fisher linear discriminant analysis (FLDA) is not currently considered among the top methods for feature extraction or classification. In this paper, we will review two recent approaches to FLDA, namely, least squares Fisher discriminant analysis (LSFDA) and regularized kernel FDA (RKFDA) and propose deep FDA (DFDA), a straightforward nonlinear extension of LSFDA that takes advantage of the recent advances on deep neural networks. We will compare the performance of RKFDA and DFDA on a large number of two-class and multiclass problems, many of them involving class-imbalanced data sets and some having quite large sample sizes; we will use, for this, the areas under the receiver operating characteristics (ROCs) curve of the classifiers considered. As we shall see, the classification performance of both methods is often very similar and particularly good on imbalanced problems, but building DFDA models is considerably much faster than doing so for RKFDA, particularly in problems with quite large sample sizes.**

***Index Terms*— Deep neural networks (DNNs), Fisher discriminant analysis (FDA), kernel discriminant analysis, nonlinear classifiers.**

NOMENCLATURE

c	Number of classes.
d	Number of pattern features.
$N, N_1, \ldots, N_c$	Total and class sample sizes.
I, I_d	$d \times d$ identity matrix.
$\mathbf{1}_N$	All ones N-dimensional vector.
X	$N \times d$ sample data matrix.
m	Sample mean.
E	$N \times c$ one hot encoding label matrix.
H	$N \times N$ centering matrix.
Π	$c \times c$ diagonal matrix with $\Pi_{ii} = N_i$.
S_B, S_T	Between-class and total sample covariance matrices.
s_B, s_T	Between-class and total projected covariance matrices.
λ	L_2 regularization parameter.
D_λ	Inverse of $S_T + \lambda I$.
$k(\cdot, \cdot)$	Kernel.
K	Kernel matrix.
C	HKH matrix.
$\mathcal{W}$	Generic weight set of a DNN.
w_0, W	Linear output components of $\mathcal{W}$.
$\widetilde{\mathcal{W}}$	Components of $\mathcal{W}$ excluding w_0 and W.
$s_k(x)$	Generic scoring for class k acting on x.
π_i	Prior probability of class i.
$P(0\|x), P(1\|x)$	Posterior probabilities of classes 0, 1 conditioned on x.
TPR(t), FPR(t)	True and false positive rates associated with a threshold t.

Manuscript received April 27, 2018; revised October 2, 2018, January 25, 2019, and February 21, 2019; accepted March 11, 2019. This work was supported in part by Mineco under Grant TIN2016-76406-P, in part by the Comunidad de Madrid under Grant S2013/ICE-2845 CASI-CAM-CM, in part by the Project FACIL—Ayudas Fundación BBVA a Equipos de Investigación Científica 2016, in part by the Universidad Autónoma de Madrid (UAM)–Asociación para el Desarrollo de la Ingeniería del Conocimiento (ADIC) Chair for Data Science and Machine Learning, and in part by the Instituto de Ingeniería del Conocimiento. *(Corresponding author: David Díaz-Vico.)*

The authors are with the Dpto. Ing. Informática, Instituto de Ingeniería del Conocimiento, Universidad Autónoma de Madrid, 28049 Madrid, Spain (e-mail: david.diaz.vico@outlook.com).

Digital Object Identifier 10.1109/TNNLS.2019.2906302

I. INTRODUCTION

IMBALANCED classification is certainly among the most important problems in machine learning, and as such, it has received a wide attention [1]. This has been particularly the case since the mid-2000s, where articles such as [2] and [3] drew the scientific community to work on a topic that, although recognized as relevant, up to that, the moment had received only scattered attention [4]–[7]. Over time, it has been possible to group [8] the many proposals for handling imbalanced problems into two general approaches, sample-based procedures or algorithm-based ones. Sample-based algorithms usually try to correct the imbalance by undersampling the largest class (see [9]), oversampling the smaller class (as in the celebrated Synthetic Minority Over-Sampling Technique approach [10]), or applying both in combination with an ensemble classifier [11]. Given the computational cost of rebalancing and of working with ensembles, simple classifiers such as decision trees are often used. On the other hand, algorithm-based methods involve stronger classifiers and seek to correct their natural bias toward the larger classes. A natural way to achieve this is to introduce imbalance correcting classification costs [7], or to modify a classifier's loss function so that the influence of the smaller classes is increased [12]. Neural networks for imbalanced classification were studied under this second approach. In fact, it was early recognized [13] that the usual one-hot target encoding often used in neural classification implicitly and strongly favored the larger class. To correct this, it was proposed in [13] that an alternate coding of class patterns which incorporated class size information into the network's training.

These observations suggest that classification methods that are implicitly aware of class information should be of interest when seeking efficient classifiers in imbalanced problems. One such method is Fisher linear discriminant analysis (FLDA [14], [15]). FLDA is based on having competing spreads for the within-class covariances (small) and the between-class covariance (large), and it is most often applied to obtain useful data representations. However, while it can

2162-237X © 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

also be used to build distance-based classifiers, it is not frequently applied as such. This is often the case of purely linear classifiers, but another reason is that its goal of keeping class means as far as possible while at the same time having small class variances, can be optimally achieved only when all classes have Gaussian-distributed features with a common covariance matrix.

Nevertheless, FLDA is at the core of several successful methods for important problems. For instance, in face recognition and person reidentification, the well known Fisher face method [16], which initially relied on models relatively close to Fisher's original proposal, has been progressively improved by adding a nonlinear processing of the images to be identified. Another well-known extension is kernel Fisher discriminant analysis (KFDA [17]), which essentially applies linear FDA over nonlinear extensions of the initial features and takes advantage of the kernel trick to avoid having to explicitly build them. KFDA initially addressed binary classification but has been extended to multiclass problems in [18]. Moreover, the approach in [18], regularized KFDA (RKFDA) puts on a clear footing on how to deal with regularization in KFDA. However, on the other hand, working with kernels requires $N \times N$ matrix computations, with N sample size, which may make it too expensive, if not unfeasible, on large problems. Other kernel extensions of FLDA have been proposed, such as [19] for face recognition, [20] for radial basis function-based classification, or [21] for general feature extraction and recognition.

After the huge success of deep neural networks (DNNs), a clear way for the nonlinear extension of linear methods is to place them at the last hidden layer of a possibly deep multilayer perceptron (MLP). In fact, the recent advances in DNNs have greatly simplified the training of networks with a very large number of layers and hundreds of thousands of weights. To this, we can add the widespread availability of DNN tools such as Theano [22], CNTK [23], Torch [24], MXNET [25], or TensorFlow [26], endowed with compilation procedures that automatically compute backpropagation gradients for cost functions much more general than the cross entropy or square errors that had traditionally been applied in classification or regression. In a slight abuse of language, here, we will use the term "deep" with precisely this meaning of networks defined, initialized, and optimized using these new advanced techniques, rather than implying a large number of hidden layers. In fact, a large number of layers and the correspondingly large number of weights are dependent on having very large training samples with sizes in the millions of patterns. Here, we will consider samples sizes up to the hundreds of thousands and networks with up to five layers; note that while their training is nowadays almost routine, it was simply not possible until a few years ago, as the modern initialization, activation, and optimization techniques were not yet available.

It is thus natural trying to take advantage of this for Fisher analysis, but it is not easy to blend eigencalculations on the covariance matrices of the last hidden layer with the usual backpropagation training of DNNs; see [27] for an early attempt in this direction, see [28] where FLDA is preceded by nonlinear transformations learned by DNNs in a semisupervised fashion, and see [29] where the direct optimization of FLDA's eigenvalue-based criterion is proposed. Other examples are [30] for person reidentification or [31] for gender detection. A much simpler way for such an extension is given by the initial results in [32] and [33], and particularly, the proposals in [34] and [18] show how to relate FLDA with a least squares regression (LSR) over properly defined targets (see also [35] for a different least squares approach). We shall make extensive use of this and also of the isometry that is shown in [18] and [34] to exist between the FLDA projections and those induced by the LSR solution. This isometry implies that equivalent FLDA and LSR distance-based classifiers can be defined in terms of k-nearest neighbors, minimum class-mean distances, or as done here, distance-based scores.

As a consequence, a simple way to nonlinearly extend the preceding is to replace linear LSR by a DNN counterpart, where the outputs of a DNN will now approximate appropriate class-based LSR targets. Of course, regularization is also mandatory for DNNs, which involves not only a careful choice of penalties but also an adequate criterion function to be minimized during hyperparametrization. This DNN approach to FDA is our main contribution here (see [36] for a preliminary version), to which we can add the following.

1) The proposal of suitable scoring functions for RKFDA and deep FDA (DFDA) based on the computation of receiver operating characteristic (ROC) curves and area under the curve (AUC) values.
2) An extensive comparison of RKFDA and DFDA over a large number of two-class and multiclass data sets, many of them involving imbalanced problems, which show that DFDA gives essentially the same classification results but with a much lower computational cost.

This paper is organized as follows. In Section II, we shall briefly review classical FLDA, and in Section III, we give a streamlined expositions of the RKFDA approach in [18]. In Section IV, we will review the equivalence in [18] and [34] between a concrete LSR problem and FLDA. We will propose in Section V a distance-based scoring function for both DFDA and RKFDA that facilitates the computation of AUC values which we will use in order to optimize the regularization parameter of DFDA and RKFDA. In Section VI, we will extensively compare the performance of DFDA and that of RKFDA over a number of substantially imbalanced two-class problems, whereas, in Section VII, we will do so on several large scale problems that are either imbalanced or multiclass or both; we also include, here, results for the largest data sets among those considered in [17]. While, as we shall see, the classification performance of RKFDA and DFDA is essentially the same, the computational costs of DFDA are considerably smaller and make it possible to apply DFDA to large problems for which the large size of the kernel matrix makes the use of RKFDA too costly. Finally, a brief discussion will be given as well as pointers to further work. We also mention that Python code for building RKFDA and DFDA models and performing our experiments are available at a GitHub repository.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

II. Fisher's Linear Discriminant Analysis

A. Generalized Eigenproblem

Recall that FLDA seeks to concentrate its projections around their class means while, at the same time, keeping apart these class means. Several target functions have been proposed, see [14, Sec. 10.2], where many of them are shown to be equivalent. In this paper, we will maximize the trace criterion

$$g(A) = \text{trace}\left(s_T^{-1} s_B\right) = \text{trace}((A^t S_T A)^{-1}(A^t S_B A)) \quad (1)$$

where A is the $d \times q$ projection matrix, S_B and S_T are the sample between-class and total covariance matrices, respectively, and s_B and s_T denote the between-class and total covariances of the projections $z = A^t x$, see [18, Sec. 2.2], for more details. Assuming q to be such that s_B has rank q and s_T is invertible, solving $\nabla_A g = 0$ leads to $0 = -2\ S_T A s_T^{-1} s_B s_T^{-1} + 2\ S_B A s_T^{-1}$, i.e., to the problem of finding A such that $S_B A = S_T A s_T^{-1} s_B$.

Given that, for any M and an invertible Q, $\text{trace}(Q^{-1} M Q) = \text{trace}(Q Q^{-1} M) = \text{trace}(M)$, the solution of (1) is unique modulo any such a $q \times q$ transformation Q of the A projections, which will not change the cost function $g(A)$. In particular, if $s_T^{-1} s_B = U \Gamma U^t$ is the SVD of $s_T^{-1} s_B$, we can replace the previous problem with the following equivalent eigenproblem:

$$S_B A = S_T A \Gamma \quad (2)$$

with Γ the eigenvalues of $s_T^{-1} s_B$ and, hence, the nonzero eigenvalues of $S_T^{-1} S_B$. If S_T is invertible, solving (2) is equivalent to solving $S_T^{-1} S_B A = A\Gamma$, we then have

$$g(A) = \text{trace}\left(s_T^{-1} s_B\right) = \text{trace}\ \Gamma = \gamma_1 + \ldots + \gamma_q. \quad (3)$$

We can maximize this simply by selecting the q largest eigenvalues in Γ after sorting them in descending order, together with some convenient normalization of their associated eigenvectors. In fact, note that a normalization has to be introduced given that the maximizer of (1) is not uniquely defined; the usual choice is to have $A^t S_T A = I_q$ but, here, we will consider solutions B such that $B^t S_T B = \Gamma$. Observe that we can move from A to B simply by setting $B = A\Gamma^{1/2}$ and vice versa.

Finally, it may be the case that S_T does not have full rank. While often S_T^{-1} is then replaced with the Moore–Penrose inverse of S_T, here, we will consider regularized discriminant analysis [37], working with the positive definite matrix $S_T + \lambda I$ for some $\lambda > 0$ and solving then the eigenvalue problem

$$(S_T + \lambda I)^{-1} S_B A = A\Gamma. \quad (4)$$

B. Solving the Generalized Eigenproblem

At first sight, solving (4) would require the computation of:
1) S_B and $D_\lambda = (S_T + \lambda I)^{-1}$ at a cost $O(Nd^2) + O(d^3)$;
2) $D_\lambda S_B$ at a cost $O(d^3)$;
3) A and Γ from the SVD of $D_\lambda S_B$ at a cost $O(d^3)$.

Thus, besides computing S_T, three steps with a cost $O(d^3)$ are involved. To improve on this, first, observe that if X denotes the $N \times d$ data matrix, we have

$$S_T = X^t H X, \quad S_B = X^t H E \Pi^{-1} E^t H X \quad (5)$$

Algorithm 1 Solving Fisher Linear Discriminant Method

```
1 Read $X$, $\lambda$, $E$, $(N_1, \ldots, N_c)$
2 Compute $\Pi = \text{diag}(N_1, \ldots, N_c)$
3 Compute $S_T = X^T H X$
4 Compute $D_\lambda = (S_T + \lambda I_d)^{-1}$
5 Compute $Q = X^t H E \Pi^{-1/2}$ and $R = Q^t D_\lambda Q$
6 Compute the SVD $R = V \Gamma V^t$
7 Return $\Gamma$, $V$, $D_\lambda$
```

where the superscript t indicates the transpose, Π denotes the $c \times c$ diagonal matrix with $\Pi_{ii} = N_i$, the number of sample patterns in class i, E denotes the $N \times c$ one-hot encoding matrix, and H is the centering matrix

$$H = I_N - \frac{1}{N}\mathbf{1}_N \mathbf{1}_N^t \quad (6)$$

with $\mathbf{1}_N$ the all ones N-dimensional vector; in particular, the pth row of the $N \times d$ matrix HX equals $x_p^t - m^t$, with x_p the pth sample vector. Moreover

$$S_B = X^t H E \Pi^{-1/2}\ \Pi^{-1/2} E^t H X = Q Q^t \quad (7)$$

where we write $Q = X^t H E \Pi^{-1/2}$. Defining, now, the $c \times c$ matrix

$$R = Q^t D_\lambda Q \quad (8)$$

if $R = V \Gamma V^t$ is its reduced SVD decomposition (i.e., we only consider the $q \leq c$ nonzero eigenvalues in Γ), then it can be easily seen that the pair $(A = D_\lambda Q V,\ \Gamma)$ verifies $D_\lambda S_B A = A\Gamma$, i.e., it solves (4), for we have

$$\begin{aligned} D_\lambda S_B A &= D_\lambda Q Q^t D_\lambda Q V = D_\lambda Q R V \\ &= D_\lambda Q V \Gamma = A\Gamma. \end{aligned}$$

This suggests to solve (4) through the steps in Algorithm 1, whose computational cost is:
1) $O(Nd^2)$ to compute S_T at step 3 of the algorithm;
2) $O(d^3)$ to compute D_λ in step 4;
3) $O(dNc) + O(dc^2)$ to compute Q and $O(cd^2 + c^2\ d)$ to compute R in step 5;
4) $O(c^3)$ to compute V, Γ from the SVD of R in step 6;
5) $O(dq^2)$ to compute $A = D_\lambda Q V$ after the return in step 7.

There is thus only one step with a cost $O(d^3)$ and when $d > N > c$, this should improve on the first approach.

III. Regularized Kernel Fisher Discriminant Analysis

A. Plain Regularized KFDA Problem

In what follows we loosely follow the discussion in [18]. We also point out that, as mentioned in [18], the treatment below is slightly different from that in [17]. We assume starting patterns $x \in \mathbf{R}^d$ and their expansions $\widetilde{x} = \Phi(x) \in \mathbf{R}^D$ in some (quite) large dimensional space (i.e., $D \gg d$). We will denote the

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

expanded data matrix as $\widetilde{X}$ and we can define the between and total covariance matrices of the expansions as

$$\widetilde{S}_B = \widetilde{X}^t H E \Pi^{-1} E^t H \widetilde{X}, \quad \widetilde{S}_T = \widetilde{X}^t H \widetilde{X} \tag{9}$$

with H again the centering matrix $H = I_N - (1/N)\mathbf{1}_N \mathbf{1}_N^t$ and E the one-hot encoding matrix. The generalized regularized eigenvalue problem now to be solved is

$$\widetilde{S}_B \widetilde{A} = (\widetilde{S}_t + \lambda I_D)\widetilde{A}\,\widetilde{\Gamma} \tag{10}$$

with $\widetilde{A} \in \mathbf{R}^{D\times q}$. This is [18, eq. (14)], shown also in that paper to be equivalent to its kernel version problem in [18, eq. (16)].

Given that now $D \gg c$ is certain, the way to proceed according to the previous discussion for FLDA would be to set $\widetilde{Q} = \widetilde{X}^t H E \Pi^{-1/2}$ and to define again the $c \times c$ matrix R as

$$R = \widetilde{Q}^t \widetilde{D}_\lambda \widetilde{Q} = \Pi^{-1/2} E^t H \widetilde{X} \widetilde{D}_\lambda \widetilde{X}^t H E \Pi^{-1/2} \tag{11}$$

where $\widetilde{D}_\lambda = (\widetilde{S}_T + \lambda I)^{-1}$ and R has rank $q \le c-1$; we then get its reduced SVD decomposition $R = V\Gamma V^t$ and somehow compute the projection matrix $\widetilde{A} = \widetilde{D}_\lambda \widetilde{Q} V$. However, this is not feasible as all the preceding computations would have to be performed on the expanded $\widetilde{X}$, something we want to avoid.

To do so, we will rewrite $\widetilde{D}_\lambda \widetilde{X}^t H$ as follows. Set $C = H\widetilde{X}\widetilde{X}^t H = HKH$, where we will call $K = \widetilde{X}\widetilde{X}^t$ the *kernel* matrix as we assume its entries $\widetilde{x}_p \cdot \widetilde{x}_q$ to be computed on the initial features x through a suitable kernel $k(x_p, x_q)$, and set also $\Delta_\lambda = (C + \lambda I_N)^{-1}$. Then, we have

$$\begin{aligned}
\widetilde{X}^t H &= \widetilde{X}^t H\,(C + \lambda I_N)\,\Delta_\lambda \\
&= (\widetilde{X}^t H H \widetilde{X}\widetilde{X}^t H + \lambda \widetilde{X}^t H)\,\Delta_\lambda \\
&= (\widetilde{X}^t H H \widetilde{X} + \lambda I_D)\,\widetilde{X}^t H\,\Delta_\lambda \\
&= \widetilde{D}_\lambda^{-1}\,\widetilde{X}^t H\,\Delta_\lambda
\end{aligned}$$

therefore, it follows that

$$\widetilde{D}_\lambda \widetilde{X}^t H = \widetilde{X}^t H\,\Delta_\lambda \tag{12}$$

which, in turn, implies

$$\begin{aligned}
R &= \Pi^{-1/2} E^t H \widetilde{X}\,(\widetilde{D}_\lambda \widetilde{X}^t H E)\,\Pi^{-1/2} \\
&= \Pi^{-1/2} E^t H \widetilde{X}\,(\widetilde{X}^t H\,\Delta_\lambda)\,E \Pi^{-1/2} \\
&= \Pi^{-1/2} E^t\,(H \widetilde{X}\widetilde{X}^t H)\,\Delta_\lambda E \Pi^{-1/2} \\
&= \Pi^{-1/2} E^t\,C\,\Delta_\lambda\,E \Pi^{-1/2}
\end{aligned} \tag{13}$$

where we use parentheses for easier reading.

Therefore, the kernel matrix is just what we need to obtain the matrix R without having to handle the extended data matrix $\widetilde{X}$ and to compute Rs SVD to get V and Γ. Using (12) again, the resulting projecting matrix $\widetilde{A}$ would then be

$$\begin{aligned}
\widetilde{A} &= \widetilde{D}_\lambda \widetilde{Q} V = \widetilde{D}_\lambda \widetilde{X}^t H E \Pi^{-1/2} V \\
&= \widetilde{X} H \Delta_\lambda E \Pi^{-1/2} V.
\end{aligned} \tag{14}$$

At first sight, $\widetilde{A}$ seems to require $\widetilde{X}$ but, again, we can avoid this when computing the Fisher projections. In fact, the projection of the expansion $\widetilde{x}$ of a new x would be

$$\begin{aligned}
z &= \widetilde{A}^t(\widetilde{x} - \widetilde{m}) = V^t \Pi^{-1/2} E^t \Delta_\lambda H \widetilde{X}^t(\widetilde{x} - \widetilde{m}) \\
&= V^t \Pi^{-1/2} E^t \Delta_\lambda H \left(k_x - \frac{1}{N} K \mathbf{1}_N\right)
\end{aligned} \tag{15}$$

where k_x is the vector $(k(x, x_1), \dots, k(x, x_N))^t$ and which only involves kernel operations. Note that when $k(x, x') = x \cdot x'$ (i.e., $\widetilde{x} = x$), we simply recover the previous solution of FLDA.

We next make explicit the corresponding training and testing algorithms for this RKFDA procedure.

Algorithm 2 Training Phase of the Regularized Kernel Fisher Discriminant Method

```
1 Read $K$, $\lambda$, $E$, $(N_1, \dots, N_c)$
2 Compute $\Pi = \mathrm{diag}(N_1, \dots, N_c)$
3 Compute $C = HKH$
4 Compute $\Delta_\lambda = (C + \lambda I_N)^{-1}$
5 Compute $F = E^t C$ and $G = \Delta_\lambda E$
6 Compute $P = \Pi^{-1/2} F$, $Q = G \Pi^{-1/2}$ and $R = PQ$
7 Compute the condensed SVD $R = V\Gamma V^t$
8 Return $\Gamma$, $V$, $\Delta_\lambda$
```

B. Train and Test Algorithms for the RKFDA Problem

We describe, here, [18, Algorithm 5] in a more detailed form, handling separately what would be its training phase (Algorithm 2) and its testing phase (Algorithm 3) and with an eye to the detailed computational analysis we given in the following. In the training phase, we simply compute R in (13) and perform a SVD on it, getting V and Γ such that

$$R = V\Gamma V^t \tag{16}$$

where Γ contains the $q \le c-1$ nonzero eigenvalues of R and V is made of orthogonal eigenvectors.

These steps to obtain V and Γ are put together in Algorithm 2 for the training phase of RKFDA, whose computational costs run as follows.

1) Since we have $C = HKH = K - (1/N)(D^K + (D^K)^t) + (1/N)V^K$, with:
   a) D^K the $N \times N$ matrix $D^k = (d^K, \dots, d^K)$ with d^K the degree vector $d_K = (d_1^K, \dots, d_N^K)^t$ and $d_p^K = \sum_q K_{p,q}$
   b) $V_{pq}^K = v^K = \sum_{p,q} K_{p,q}$

   computing C in line 3 does not essentially involve float operations once the kernel matrix K is available; computing it has a cost of $O(N^2\kappa)$, where κ is the cost of a kernel computation $k(x, x')$.
2) Computing $\Delta_\lambda = (C + \lambda I_N)^{-1}$ in line 4 has a $O(N^3)$ cost.
3) Computing $F = E^t C$ and $G = \Delta_\lambda E$ in line 5 has a cost of $O(cN^2)$.
4) Computing $P = \Pi^{-1/2} F$, $Q = G\Pi^{-1/2}$ and $R = PQ$ in line 6 has a cost of $O(c^2\,N)$.
5) Computing the condensed SVD $R = V\Gamma V^t$ in line 7 has also a cost of $O(c^3)$.

Since we may expect $N \gg c$, the most expensive operation is computing $\Delta_\lambda = (C + \lambda I_N)^{-1}$.

With regards to the test phase, where we apply the projections $A^t \widetilde{x}$ to the expansions $\widetilde{x}$ of new, unseen x, the required steps are outlined in Algorithm 3 for getting these projections

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Algorithm 3 Test Phase of the Regularized Kernel Fisher Discriminant Method

```
1 Read $k(\cdot,\cdot)$, $S=\{x_1,\dots,x_n\}$, $\Delta_\lambda=(C+\lambda I_N)^{-1}$, $E$, $(N_1,\dots,N_c)$, $\Gamma_q$, $V_q$
2 Read test sample $S_{test}=\{x'_1,\dots,x'_M\}$
3 Compute the test kernel matrix $K_S=(k'_1,\dots,k'_M)$, with
      $$k'_p=\left(k\left(x'_p,x_1\right),\dots,k\left(x'_p,x_N\right)\right)^t-\overline{k'}_p\mathbf{1}_N \qquad (17)$$
  and $\overline{k'}_p=\frac{1}{N}\sum_q k(x'_p,x_q)$
4 Compute $C_S=HK_S$
5 Compute $\Pi=\mathrm{diag}(N_1,\dots,N_c)$
6 Compute $T=V^t\Pi^{-1/2}E^t\Delta_\lambda$
7 Return $TC_S$
```

on a test sample $S_{test}=\{x'_1,\dots,x'_M\}$ with M new patterns. Its main computational costs are those of the following.

1) Computing K_S in line 3 with a cost of $O(NM\kappa)$, where we recall that κ denotes the cost of a kernel computation $k(x,x')$.
2) Computing $C_S=HK_S=K_S-(1/N)\mathbf{1}_N\mathbf{1}_N^tK_S$ in line 4 only involves sums but no further kernel operations.
3) Computing $T=(V^t\Pi^{-1/2})(E^t\Delta_\lambda)$ in line 6 with a $O(cN^2)$ cost.
4) Computing TC_S in line 7 with a $O(cNM)$ cost.

Thus, assuming $N\ge M\ge c$, the overall test cost is dominated by the $O(cN^2)$ of line 6.

Finally, we observe that we have relied on the assumption of a finite dimension D for the $\widetilde{x}$ for the sake of motivating and deriving the above-mentioned algorithms. However, we ultimately only need a kernel $k(x,x')$ to compute the matrix R and the projections z in Algorithms 2 and 3. This makes possible to work, for instance, with projections $\Phi(x)$ in a countably infinitely dimensional Hilbert space when Gaussian kernels are used.

IV. Deep Fisher Discriminant Analysis

A. Least Squares Regression and FLDA

Let X be the $n\times d$ data matrix, $\mathbf{1}_n$ the all ones vector, and in a two-class problem, let y be the target vector defined by setting $y_p=n/n_1$ for patterns in class 1 and $y_p=-n/n_2$ for those in class 2. Then, it is well known [15] that solving

$$\min_{w_0,w}\frac{1}{2}\|y-\mathbf{1}_nw_0-Xw\|^2 \qquad (18)$$

gives a solution to FLDA. In fact, if m_1 and m_2 are the class means on the original features, then $S_B=(m_1-m_2)(m_1-m_2)^t$, and setting $w=S_T^{-1}(m_1-m_2)$, it is easy to check that $S_T^{-1}S_Bw=w\gamma$, with $\gamma=(m_1-m_2)^tS_T^{-1}(m_1-m_2)$. In other words, w solves the eigenproblem (4) and, therefore, coincides with a dilation of an FLDA's projection vector.

Many attempts have been made to extend this simple equivalence to multiclass problems. This has been achieved by the essentially equivalent proposals of Ye [34] and, particularly, Zhang *et al.* [18] (see also the proposals by Park and Park [33]). We describe it next.

With $\mathbf{1}_n$ and X as before, let now W be a $d\times q$ matrix, w_0 a $q\times 1$ vector and Y a target matrix to be chosen, and consider the LSR problem of minimizing

$$\min_{w_0,W}\frac{1}{2}\left\|Y-\mathbf{1}_n\,w_0^t-XW\right\|^2. \qquad (19)$$

Assuming for simplicity, a regular S_T (or working with $S_T+\lambda I$ for some $\lambda>0$ if not), the optimal W^* solving (19) is

$$W^*=S_T^{-1}X^tHY \qquad (20)$$

with H again the centering matrix. As in Section II, we have here $S_T=X^tHX$ and also

$$S_B=X^tHE\Pi^{-1/2}\ \Pi^{-1/2}E^tHX=QQ^t$$

with E the $N\times c$ one-hot encoding matrix. Choosing as targets in (19) the $N\times c$ matrix $Y=HE\Pi^{-1/2}$, the LSR matrix W^* solution is given by

$$W^*=S_T^{-1}X^tHY=S_T^{-1}X^tHE\Pi^{-1/2}=S_T^{-1}Q. \qquad (21)$$

We see now that we can recover from W^* a solution of (4). To do so, consider again the reduced SVD decomposition $R=V\,\Gamma V^t$ of the $c\times c$ matrix $R=Q^tS_T^{-1}Q$. Setting $B=W^*V$, we have

$$\begin{aligned}S_T^{-1}S_BB&=S_T^{-1}S_BW^*V\\&=S_T^{-1}QQ^tW^*V=S_T^{-1}QQ^tS_T^{-1}QV\\&=S_T^{-1}QRV=S_T^{-1}QV\Gamma\\&=W^*V\Gamma=B\Gamma.\end{aligned}$$

Moreover

$$\begin{aligned}B^tS_TB&=V^t(W^*)^tS_TW^*V=V^tQ^tS_T^{-1}S_TS_T^{-1}QV\\&=V^tQ^tS_T^{-1}QV=V^tRV\\&=\Gamma.\end{aligned}$$

In other words, (B,Γ) is a solution of (4) with the normalization $B^tS_TB=\Gamma$.

However, we would like to avoid working with B as it would require the SVD of R and, instead, derive a suitable projection from the LSR solution W. To achieve this, let us denote by $z=(W^*)^tx$ and $\omega=B^tx$ the W^* and B projection, respectively, of a pattern x. We then have [18]

$$\begin{aligned}\|\omega-\omega'\|_2^2&=(x-x')^tBB^t(x-x')\\&=(x-x')^tW^*VV^t(W^*)^t(x-x')\\&=\|z-z'\|_2^2\end{aligned}$$

which implies that

$$\|z-\overline{z}_k\|_2=\|\omega-\overline{\omega_k}\|_2, \qquad (22)$$

where $\overline{z}_k,\overline{\omega}_k$ denote the kth class means for the z and ω projections.

Thus, any distance classifier or, more generally, any score based on distances to class means will give the same results when computed either over the least squares z projection or over the linear FDA ones ω. In other words, the LSR solution can be used to define distance-based scores equivalent to the ones which could be defined using Fisher's projection;

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

we will discuss them in Section V below. This LSR procedure opens the way to a nonlinear, DNN-based approach to Fisher's analysis which we describe in Section IV-B. Before doing so, we point out that it can be easily seen that the row in the target matrix $Y = HE\Pi^{-1/2}$ for a pattern x_p of class k is given by

$$Y_{pk} = \frac{n - n_k}{n\sqrt{n_k}} \tag{23}$$

and $Y_{pk'} = -(\sqrt{n_k}/n)$ for the other components $k' \neq k$. (In [34], these Y values are just multiplied by $\sqrt{n}$.)

B. Deep Neural Fisher Discriminant Networks

As just argued, a distance-based classifier for a c class problem equivalent to the one resulting from Fisher projections can be obtained through the following steps.

1) For a given training matrix X_{tr}, class indicator matrix E_{tr} and targets $Y_{tr} = HE_{tr}\Pi^{-1/2}$, obtain the c-dimensional vector w_0^* and $d \times c$ matrix W^* which solve the LSR problem

$$\min_{w_0, W} \frac{1}{2}\left\|Y_{tr} - \mathbf{1}_n\, w_0^t - X_{tr}W\right\|^2. \tag{24}$$

2) Use them to compute the projections $y = w_0^* + (W^*)^t x$ for $x \in D_{tr}$ and their class means $\overline{y}_k = w_0^* + (W^*)^t \overline{x}_k$.
3) Assign a test pattern x to a class according to scores defined in terms of the distances between the projection $y = w_0^* + (W^*)^t x$ and the class means $\overline{y}_k$.

Now, it is natural to define a nonlinear extension by applying the previous LSR steps to nonlinear extensions $z = \Phi(x)$ of the original features x. A simple way is to apply the previous linear steps to the z features on the last hidden layer of a DNN; more precisely, we

1) Solve the LSR problem

$$\min_{\mathcal{W}} \frac{1}{2}\|Y_{tr} - f(X_{tr}, \mathcal{W})\|^2 \tag{25}$$

to get an optimal DNN weight set $\mathcal{W}^*$; here, X_{tr} is the training matrix, Y_{tr} is the training target matrix defined previously and the matrix $f(X_{tr}, \mathcal{W})$ has rows of the form $f(X_{tr}, \mathcal{W})_p = f(x_p, \mathcal{W})$, with $f(x, \mathcal{W})$ the linear outputs of a deep network with weights $\mathcal{W}$.
2) Compute the DNN projections $y_p = f(x_p, \mathcal{W}^*)$ over X_{tr} and their class means $\overline{y}_c$.
3) Compute for the rows x in a test matrix X_{ts} their DNN projections $y = f(x, \mathcal{W}^*)$ and corresponding scores according to their distances to the $\overline{y}_c$ means and assign them to the class with the highest score.

Let us write the optimal weight as $\mathcal{W}^* = (w_0^*, W^*, \widetilde{\mathcal{W}}^*)$, where w_0^*, W^* are the linear weight vector and matrix that connect the last hidden layer with the network outputs; let us also denote as z the last hidden layer features $z = \Phi(x, \widetilde{\mathcal{W}}^*)$, with Φ the partial DNN transformation that computes them. Then, the previous w_0^*, W^* also solve the LSR problem (18) over the z features: if not and there were better choices, say $\widehat{w}_0$ and $\widehat{W}$ for w_0 and W with a smaller square error over the z features, the DNN weight set $(\widehat{w}_0, \widehat{W}, \widetilde{\mathcal{W}}^*)$ would yield a smaller error than the one defined by the previously optimal $\mathcal{W}^*$. As a consequence, any score-based classifier built on the full DNN projections is equivalent to the same score classifier acting over the FLDA projections of the last hidden layer patterns.

We will call this DFDA, as it effectively applies Fisher's standard linear discriminant analysis over the features z learned by training a deep neural model. As mentioned in Section I, we point out that our use of the term "deep network" has more to do with the underlying network architectures and initialization and training techniques that we will use, than with the networks having a large number of hidden layers (which will be at most five in our experiments).

To avoid singularities, we can simply add a regularization term. In its simplest form, we would solve

$$\min_{w_0, W, \widetilde{\mathcal{W}}} \frac{1}{2}\|Y - f(X, w_0, W, \widetilde{\mathcal{W}})\|^2 + \frac{\lambda}{2}\mathrm{trace}(W^t W + \widetilde{W}^t \widetilde{W}) \tag{26}$$

where $\widetilde{W}$ are the components of $\widetilde{\mathcal{W}}$ when layer biases are removed. We shall use this cost function in our experiments. Note that other regularization procedure, such as dropout, could be used for the $\widetilde{W}$ weights; on the other hand, $(\lambda/2)\mathrm{trace}(W^t W)$ should be the regularizer of the linear output weights W.

V. Scoring Functions and AUC Computation

A. Scoring Functions for RKFDA and DFDA

Since we intend to use the AUC as our merit function for model hyperparametrization and test set evaluation, we will transform the RKFDA and DFDA outputs into vector scores with components in a $[0, 1]$ range, where higher values of the score components should reflect outputs closer to a class centroid. The desired $[0, 1]$ score range is reminiscent of that of posterior Probabilities, and in principle, one way of obtaining such scores could be to try to exploit the fact that Fisher's LDA maximizes the posterior class probabilities assuming all sample class densities are given by Gaussians with different means but the same covariance. However, this is most likely not being true of the original features, cannot be checked on the implicit features of RKFDA, and is not guaranteed at all for the deep features at the last hidden layer of a DFDA network. Because of this, we prefer to follow the simple heuristic we describe next.

Note that, for the RKFDA z projections defined by (15), class centroids $\overline{z}_k$ are easily computed and so are the distances to them of new pattern projections. Moreover, for FDA, we have just observed in (22) that the least squares projections have the same centroid distances than those of a Fisher projection. This obviously extends to the outputs of a DFDA network, as they are the least squares projections of the last hidden layer representations.

Thus, for both RKFDA and DFDA, we can easily compute the class-means distances $\|z - \overline{z}_k\|$ of their projections z, distances that we will transform into $[0, 1]$ scores better suited to our subsequent work. To do so, given a validation or test Fisher projection z, we first compute its distances $d_k = d_k(z)$

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

to the centroids $\overline{z}_k$, $1 \leq k \leq c$, of the training class projections and then we define the kth score of z as the following normalization of the inverses $1/d_k$, namely,

$$s_k = s_k(z) = \frac{\frac{1}{d_k}}{\sum_j \frac{1}{d_j}} = \frac{1}{\sum_j \frac{d_k}{d_j}} = \frac{1}{1 + \sum_{j \neq k} \frac{d_k}{d_j}}.$$

Of course, we have to watch out for zero or near-zero values of d_k. Assuming different class centroids, $d_j = 0$ can only happen at one k; in such a case, the previous expression would clearly give $s_k = 1$ while we simply take $s_j = 0$ for other j values as they would involve a fraction $1/d_k = \infty$ in the denominator.

B. AUC for Two-Class and Multiclass Problems

Considering first two-class problems, let $s(x)$, $0 \leq s(x) \leq 1$ be a scoring function which gives higher values to positive patterns, i.e., those in class 1; an example could be any estimate $\hat{P}(1|x)$ of the conditional probability of a pattern being from class 1 given its features x. Let f_0, f_1 be the densities of the class conditioned random scores $S_0(x) = s(x|0)$, $S_1(x) = s(x|1)$ which we assume to be independent. Then, for any threshold t, let $c_t(x)$ be the classifier such that $c_t(x) = 1$ iff $s(x) > t$ and let true positive ratios (TPR(t)), false positive ratios (FPR(t)) be its true and false positive ratios, that is,

$$\text{TPR}(t) = P(\{s(x) > t | x \in C_1\}) = \int_t^1 f_1(s)ds$$

$$\text{FPR}(t) = P(\{s(x) > t | x \in C_0\}) = \int_t^1 f_0(s)ds.$$

Note that TPR is equivalent to recall or sensitivity and $1-\text{FPR}$ is equal to specificity. The ROC curve is defined by the points $(\text{FPR}(t), \text{TPR}(t))$ and the area below the ROC is called the AUC. We define AUC in terms of the 1 class but the same value is obtained if defined in terms of the 0 class.

The AUC captures in a single number of the performance of the underlying classifier across all thresholds $0 \leq t \leq 1$. Moreover, since TPR(t) and FPR(t) are computed on the rows and columns on the confusion matrix, the AUC should be more robust on imbalanced problems, as it uses no prior probability information. Finally, it can be shown [38] that

$$\text{AUC} = P(\{s(x) > s(x') : x \in C_1, x' \in C_0\})$$

i.e., the AUC measures the probability that the score of a random positive pattern is larger than that of negative one. In particular, this supports the intuition of a given classifier being preferable to another with smaller AUC.

Contrary to the two-class situation, no clear cut extension of the AUC to a multiclass setting has been given. Conceptually, the volume under the surface (VUS) in [39] is possibly closest to the previous two-class formulation, but it may be quite difficult to compute, especially for more than three classes. Simpler approaches can be derived by computing and combining several two-class AUC values, such as the *total* AUC [40]

$$\text{AUC}_{\text{total}} = \sum_{i=1}^{c} \pi_i \ \text{AUC}_i \tag{27}$$

where, for each i, π_i is the class prior and AUC$_i$ is computed as in a two-class problem with C_i as the positive class; the *macroaverage* AUCs [41], either on its arithmetic

$$\text{AUC}_{\text{macro}} = \frac{1}{c} \sum_i \text{AUC}_i \tag{28}$$

or geometric

$$\text{AUC}_{\text{geom}} = \left(\prod_1^c \text{AUC}_i \right)^{\frac{1}{c}} \tag{29}$$

mean variants; the *microaverage* AUCs of [42, Ch. 13], or finally, the M-AUC [43]

$$M = \frac{2}{c(c-1)} \sum_{i<j} \text{AUC}(i, j) \tag{30}$$

which combines $c(c-1)$ two class AUC(i, j). See [44] for more details on the AUC and [38], [41], and [45] for examples of its use in multiclass problems. As just mentioned, there does not seem to be a general agreement on which AUC variant should be used in multiclass problems. Given its simplicity and relative robustness on imbalanced data sets, we shall use geometric macro-AUC in our multiclass experiments.

VI. DFDA Versus RKFDA on Imbalanced Two-Class Problems

RKFDA and DFDA over 30 data sets taken from the Keel repository [46]. They all derive from an original set of seven problems (some of them multiclass) whose samples are grouped in various ways to produce 30 different two-class problems with imbalance ratios that range from a minimum of 9.22 (when the classes 0 and 4 of the `glass` problem are to be classified against the class 5) to a maximum of 129.44 (when class 19 of the `abalone`problem is pitted against all others). Their sample sizes, dimensions, and imbalance ratios are given in Table I where the data set names to follow the Keel naming conventions. For all problems considered, we have used the five train-test folds provided in the Keel repository.

We will compare the performance of RKFDA models against DFDA models with a feedforward architecture. The quite popular ReLU function is our choice for the hidden layer activations; as usual in a regression setting, we have linear outputs, and we use Adam over minibatches as the backpropagation optimizer.

In principle, these choices could imply a substantial hyperparameterization cost but we will simplify this as follows. First, we will use Adam's default values for the initial learning rate (0.001) and its $beta_1$ (0.9) and $beta_2$ (0.999) parameters, as they are quite reliable and robust; similarly, we leave minibatch size at its `scikit-learn` default (200). The second source of hyperparameters could be the number of hidden layers and of units on each of them in the DFDA

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE I
DIMENSIONS, SAMPLE SIZES, AND IMBALANCE RATIOS FOR THE TWO-CLASS PROBLEMS

Dataset	dimension	n. patterns	imbalance ratio
abalone9-18	8	731	16.4
abalone19	8	4174	129.44
ecoli4	7	336	15.8
ecoli-0-1-3-7_vs_2-6	7	281	39.14
ecoli-0-1-4-6_vs_5	6	280	13
ecoli-0-1-4-7_vs_2-3-5-6	7	336	10.59
ecoli-0-1-4-7_vs_5-6	6	332	12.28
ecoli-0-1_vs_5	6	240	11
ecoli-0-3-4-6_vs_5	7	205	9.25
ecoli-0-3-4-7_vs_5-6	7	257	9.28
ecoli-0-6-7_vs_5	6	220	10
glass2	9	214	11.59
glass4	9	214	15.47
glass5	9	214	22.78
glass-0-1-4-6_vs_2	9	205	11.06
glass-0-1-6_vs_2	9	192	10.29
glass-0-1-6_vs_5	9	184	19.44
glass-0-4_vs_5	9	92	9.22
glass-0-6_vs_5	9	108	11
page-blocks-1-3_vs_4	10	472	15.86
shuttle-c0-vs-c4	9	1829	13.87
shuttle-c2-vs-c4	9	129	20.5
vowel0	13	988	9.98
yeast4	8	1484	28.1
yeast5	8	1484	32.73
yeast6	8	1484	41.4
yeast-0-5-6-7-9_vs_4	8	528	9.35
yeast-1-2-8-9_vs_7	8	947	30.57
yeast-1-4-5-8_vs_7	8	693	22.1
yeast-2_vs_8	8	482	23.1

TABLE II
DIMENSIONS, TRAIN SAMPLE SIZES, NUMBER OF CLASSES, AND IMBALANCE RATIOS FOR THE LARGE SIZE DATA SETS

Problem	n. patterns	dimension	n. classes	imbalance ratios
a4a	4,781	123	2	3.024
a8a	22,696	123	2	3.122
dna	2,000	180	3	2.265
ijcnn1	49,990	22	2	9.301
letter	15,000	16	26	1.128
pendigits	7,494	16	10	1.085
satimage	4,435	36	6	2.583
shuttle	43,500	9	7	5,684.667
usps	7,291	256	10	2.203
w7a	24,692	300	2	32.368
w8a	49,749	300	2	32.637
banana	5,300	2	2	1.231
german	1,000	20	2	2.333
image	2,086	18	2	1.323
ringnorm	7,400	20	2	1.020
splice	2,991	60	2	1.225
twonorm	7,400	20	2	1.002
waveform	5,000	21	2	2.036
combined	78,823	100	3	2.156
covtype.binary	581,012	54	2	1.051
skin_nonskin	245,057	3	2	3.818

models but here we will simply report our results for networks with a number of hidden layers ranging from 0 to 5, and 100 units each.

This leaves us with the L_2 (or Tikhonov) regularization penalty λ as our only DFDA hyperparameter, for which we explore 50 values evenly spaced on a log scale in the interval $[2^{-30}, 2^{10}]$ selecting the optimal one by k-fold cross validation as described in the following. Prior to this, the DFDA features have been normalized to 0 mean and 1 standard deviation.

The RKFDA models require two hyperparameters, the L_2 regularization penalty λ and the width γ of the Gaussian kernels $\exp(-\gamma \|x - x'\|^2)$; both will be also selected here by cross validation. As before, for λ, we will explore 50 values in the interval $[2^{-30}, 2^{10}]$. In order to select γ, we scale featurewise the RKFDA inputs to a [0, 1] range; note that after this we will have $\|x - x'\|^2 \leq d$, with d pattern dimension. Because of this, we will explore γ values of the form $(2^k/d)$, with k in the $[-10, 10]$ range; in other words, the considered kernels will essentially be powers of a basic exponential e^{-z^2}.

As just mentioned, for each of the five train partitions provided for each data set, optimal λ regularization parameters, and in the RKFDA case, γ values have been obtained by fivefold stratified cross validation using the AUC of the positive class as the scoring function (recall that, as mentioned, $\mathrm{AUC}_1 = \mathrm{AUC}_0$). Once the optimal λ and, for RKFDA, γ are chosen, we have used them to train individual DFDA and RKFDA models on the train partition and applied them on the test partition, computing afterward the test AUC scores. In Table III, we show, for each problem, the average AUC values over five test splits of the RKFDA and all the DFDA classifiers except the one with a single hidden layer that we omit for space and formatting reasons. For easier reading of these values, the table also gives at the bottom the average of the rankings of the six models considered for each problem.

As it can be seen, these average rankings are quite similar for all methods except, as it was to expected, the DFDA_0 model which, being linear, it is in fact equivalent to a standard linear Fisher model (note that, nevertheless, it gives the best test average AUC in some problems). RKFDA gives the smallest AUC in five problems, as also does the DFDA_5 model but, in general, all models except DFDA_0 appear to have a similar performance. We also observe that the best test AUC values in Table III are higher in almost all cases than the values computed for the same data sets using six state-of-the-art procedures in imbalanced classification and reported in [47]. In any case, note that the methods here are much stronger than the relatively weak decision tree classifiers used in [47].

The results in Table III are mostly descriptive. In order to achieve a more precise analysis, observe that considering all the 30 problems and their five train-test splits, we have a total of 150 AUC values for each one of the six models considered. This suggests that a more objective comparison can be achieved by applying a paired Wilcoxon signed-rank test for each model pair over these 150 AUC values, using Bonferroni corrections to compensate for multiple comparisons. The resulting p-values under the null hypothesis are shown in Table IV. As it could be expected from the previous discussion, the null hypothesis can only be rejected when the DFDA_0 model is compared against the others; this is not the case in all other comparisons and we can conclude that the performance of DFDA models with two or more hidden units is similar to that of RKFDA models.

We finally point out that in our experiments in this and Section VII we have used the `MLPRegressor` class in `scikit-learn` [48] for our implementation of DFDA networks, which makes its programming and execution very

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE III
TEST AUC FOR RKFDA AND THE 0- AND 2- TO 5-HIDDEN LAYER DFDA MODELS ON THE TWO-CLASS PROBLEMS

	RKFDA	DFDA_0	DFDA_2	DFDA_3	DFDA_4	DFDA_5
abalone9-18	88.67±7.13 (6)	95.44±2.13 (1)	91.68±6.21 (3)	90.67±4.34 (5)	90.95±4.77 (4)	91.72±5.46 (2)
abalone19	69.01±8.86 (6)	70.61±8.43 (5)	71.71±9.49 (4)	73.59±6.66 (2)	73.40±6.31 (3)	73.96±7.62 (1)
ec4	99.52±0.63 (4)	99.84±0.32 (1)	99.68±0.30 (2)	98.90±1.09 (5)	99.53±0.46 (3)	89.13±19.57 (6)
ec-0-1-3-7_vs_2-6	95.62±6.15 (1)	87.63±22.01 (3)	81.80±33.66 (5)	86.51±22.63 (4)	78.11±34.32 (6)	91.07±13.92 (2)
ec-0-1-4-6_vs_5	95.10±6.22 (5)	90.67±9.52 (6)	98.08±2.65 (2)	97.88±3.11 (3)	98.75±1.41 (1)	96.63±4.88 (4)
ec-0-1-4-7_vs_2-3-5-6	96.97±3.13 (1)	91.50±9.70 (6)	94.23±6.59 (3)	93.05±5.11 (5)	95.89±2.56 (2)	93.47±6.33 (4)
ec-0-1-4-7_vs_5-6	96.38±4.48 (3)	95.22±5.18 (4)	97.92±1.49 (2)	94.12±6.36 (6)	94.70±5.92 (5)	98.05±1.63 (1)
ec-0-1_vs_5	97.16±3.72 (3)	92.73±8.38 (6)	98.52±1.17 (1)	97.39±1.37 (2)	93.07±9.30 (5)	94.09±6.35 (4)
ec-0-3-4-6_vs_5	98.11±2.15 (4)	87.57±9.41 (6)	98.24±1.79 (3)	98.78±2.11 (2)	97.70±2.25 (5)	98.92±1.52 (1)
ec-0-3-4-7_vs_5-6	97.78±3.80 (1)	94.19±8.08 (6)	96.25±6.64 (3)	95.30±6.21 (5)	96.75±4.65 (2)	95.72±4.23 (4)
ec-0-6-7_vs_5	94.88±4.25 (1)	92.88±4.85 (5)	93±7.97 (4)	94.25±3.90 (2)	93.12±4.94 (3)	90.88±8.45 (6)
glass2	67.61±22.70 (6)	81.53±11.72 (4)	87±9 (3)	89.22±7.33 (1)	78.69±19.07 (5)	88.06±8.94 (2)
glass4	97.18±3.06 (4)	95.52±2.44 (6)	96.87±2.99 (5)	98.09±1.97 (1)	97.83±3.56 (2)	97.76±2.52 (3)
glass5	98.54±1.42 (3)	92.68±4.69 (6)	99.27±1.46 (1)	98.78±1.54 (2)	94.39±5.95 (5)	97.32±4.25 (4)
glass-0-1-4-6_vs_2	79.07±13.16 (5)	80.53±19.42 (3)	79.77±8.51 (4)	82.93±6.63 (1)	81.32±8.93 (2)	78.36±9.98 (6)
glass-0-1-6_vs_2	71.81±10.62 (5)	83.86±9.45 (1)	76.19±19.48 (3)	68.71±13.12 (6)	73.67±14.53 (4)	77.43±12.55 (2)
glass-0-1-6_vs_5	98.57±1.28 (4.5)	93.43±2.80 (6)	98.86±1.07 (2)	98.57±1.28 (4.5)	98.86±1.07 (2)	98.86±1.40 (2)
glass-0-4_vs_5	100±0 (3)	97.50±3.64 (6)	100±0 (3)	100±0 (3)	100±0 (3)	100±0 (3)
glass-0-6_vs_5	100±0 (3)	92.97±6.19 (6)	100±0 (3)	100±0 (3)	100±0 (3)	100±0 (3)
page-blocks-1-3_vs_4	100±0 (3)	96.38±0.95 (6)	100±0 (3)	100±0 (3)	100±0 (3)	100±0 (3)
shuttle-c0-vs-c4	100±0 (3)	100±0 (1.5)	89.98±19.99 (6)	100±0 (1.5)	99.98±0.03 (4)	99.92±0.15 (5)
shuttle-c2-vs-c4	100±0 (2)	100±0 (2)	99.20±1.60 (5)	99.20±1.60 (5)	100±0 (2)	99.20±1.60 (5)
vowel0	100±0 (3)	96.17±2.06 (6)	100±0 (3)	100±0 (3)	100±0 (3)	100±0 (3)
yeast4	85.46±3.24 (5)	86.55±2.21 (2)	85.68±10.21 (4)	86.69±7.26 (1)	86.49±5.44 (3)	84.96±12 (6)
yeast5	98.70±0.46 (4)	98.30±0.74 (6)	98.58±0.62 (5)	99.03±0.41 (3)	99.07±0.22 (2)	99.09±0.13 (1)
yeast6	91.88±8.17 (1)	91.28±8.29 (2)	89.85±8.56 (5)	89.65±8.99 (6)	91.03±6.12 (3)	90.61±8.92 (4)
yeast-0-5-6-7-9_vs_4	85.20±7.43 (2)	83.81±5.34 (6)	84.21±8.89 (4)	85.76±5.89 (1)	84.06±7.71 (5)	84.54±2.75 (3)
yeast-1-2-8-9_vs_7	73.68±5.15 (5)	73.93±7.32 (4)	77.17±12.61 (1)	72.10±13.30 (6)	76.57±11.92 (2)	76.48±13.03 (3)
yeast-1-4-5-8_vs_7	69.21±8.21 (3)	66.51±8.15 (6)	68.25±9.27 (5)	72.04±9.24 (2)	68.86±8.53 (4)	72.54±4.41 (1)
yeast-2_vs_8	83.72±4.47 (4)	76.80±5.13 (6)	80.11±11.32 (5)	84.21±12.10 (3)	87.59±8.27 (1)	86.92±13.25 (2)
rank mean	3.45	4.4833	3.4	3.2333	3.2333	3.2

TABLE IV
WILCOXON TESTS FOR TWO-CLASS PROBLEMS

	p-value	Hypothesis
RKFDA vs DFDA_0	0.00	Rejected
RKFDA vs DFDA_2	0.32	Not rejected
RKFDA vs DFDA_3	0.60	Not rejected
RKFDA vs DFDA_4	0.52	Not rejected
RKFDA vs DFDA_5	0.49	Not rejected
DFDA_0 vs DFDA_2	0.00	Rejected
DFDA_0 vs DFDA_3	0.00	Rejected
DFDA_0 vs DFDA_4	0.00	Rejected
DFDA_0 vs DFDA_5	0.00	Rejected
DFDA_2 vs DFDA_3	0.88	Not rejected
DFDA_2 vs DFDA_4	0.61	Not rejected
DFDA_2 vs DFDA_5	0.57	Not rejected
DFDA_3 vs DFDA_4	0.65	Not rejected
DFDA_3 vs DFDA_5	0.83	Not rejected
DFDA_4 vs DFDA_5	0.37	Not rejected

easy. We have also used `scikit-learn`'s routines and pipelines to implement the data scaling and cross-validation-based hyperparameterization. As for RKFDA, we have based our implementation on the `numpy` and `scipy` routines for eigenvalue computations, matrix inversion, and matrix multiplication. Thus, while the execution of general Python code may imply some computational overheads, the numerically heavier parts of our algorithms rely on a computationally efficient core. We have run our programs on a Fujitsu Primergy RX2540 server with 512 GB of RAM memory and Xeon E5-2640v4 processors at 2.4 GHz. Recall that the code used to implement RKFDA in our experiments is available at a https://github.com/daviddiazvico?tab=repositoriesGitHub repository.

VII. DFDA VERSUS RKFDA ON LARGE SCALE PROBLEMS

Large scale problems have an obvious importance in applications, particularly so when imbalanced and/or multiclass data sets are considered, but they seem not have been widely discussed in the literature [49]. We will compare here the performance of RKFDA and DFDA on two different data subsets with a number of moderate to large problems, some of them multiclass and/or imbalanced. The first set is taken from the https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/data sets section of the LIBSVM web site. To determine the optimal values of the hyperparameters λ and γ, we will use the predefined train, validation, and test splits available for the `dna`, `ijcnn1`, `letter`, `satimage`, and `shuttle` data sets. The remaining data sets have only train and test splits and for them, we will hyperparameterize λ and γ by tenfold stratified cross validation on the train subset. The second comparison is made of the seven data sets with at least 1000 patterns among the 13 considered in the original RKFDA paper by Mika *et al.* [17], namely, `banana`, `German`, `image`, `ringnorm`, `splice`, `twonorm`, and `waveform`. These data sets are available in several web sites; we have used those at http://theoval.cmp.uea.ac.uk/g̃cc/matlab/default.html #benchmarks.In [17], RKFDA is compared against other machine learning algorithms using as the performance measure the accuracy of a threshold-based Fisher classifier (all problems have only two classes) which is chosen applying a 1-D linear support vector machine (SVM) classifier on

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE V
TEST AUC FOR RKFDA AND THE 0- AND 2- TO 5-HIDDEN LAYER MLPs ON THE LARGE SIZE PROBLEMS

	RKFDA	DFDA_0	DFDA_2	DFDA_3	DFDA_4	DFDA_5
a4a	84.33 (6)	84.66 (1)	84.56 (5)	84.63 (3)	84.63 (3)	84.63 (3)
a8a	85.16 (4)	84.80 (6)	85.08 (5)	85.48 (1)	85.43 (2)	85.41 (3)
dna	99.35 (1)	98.79 (6)	99.19 (3)	99.18 (4.5)	99.18 (4.5)	99.22 (2)
ijcnn1	99.07 (5)	87.74 (6)	99.42 (1)	99.37 (2)	99.20 (4)	99.36 (3)
letter	99.98 (1)	95.63 (6)	99.86 (5)	99.94 (4)	99.96 (2)	99.96 (3)
pendigits	99.97 (1)	97.54 (6)	99.97 (2)	99.95 (4)	99.94 (5)	99.97 (3)
satimage	78.28 (6)	93.26 (5)	97.66 (1)	94.43 (3)	94.38 (4)	97.64 (2)
shuttle	99.92 (4)	86.22 (6)	99.83 (5)	99.95 (3)	99.98 (1)	99.96 (2)
usps	99.77 (1)	98.30 (6)	99.74 (2)	99.66 (3.5)	99.66 (3.5)	99.66 (5)
w7a	92.76 (4)	90.23 (6)	92.55 (5)	93.73 (3)	95.22 (1.5)	95.22 (1.5)
w8a	94.68 (5)	89.79 (6)	96.30 (4)	99.07 (1)	98.82 (2)	98.73 (3)
banana	86.52±1.58 (5)	55.44±2.67 (6)	90.48±1.19 (3)	90.59±1.28 (1)	90.58±1.25 (2)	90.39±1.25 (4)
german	71.78±1.58 (1)	71.59±1.48 (3)	71.66±1.55 (2)	71.30±2.23 (4)	70.15±2.31 (6)	70.90±1.57 (5)
image	96.14±0.61 (2)	82.47±1.02 (6)	95.20±1.05 (5)	96.38±0.62 (1)	95.64±0.80 (4)	95.68±0.83 (3)
ringnorm	96.24±1.13 (5)	76.78±1.73 (6)	98.01±0.70 (3)	98.07±0.62 (1)	98.03±0.67 (2)	97.91±0.72 (4)
splice	90.69±1.25 (5)	84.43±1.09 (6)	93.65±0.96 (1)	93.53±0.92 (2)	93.29±0.80 (4)	93.33±0.99 (3)
twonorm	97.78±0.65 (6)	97.92±0.60 (2)	97.92±0.59 (3.5)	97.92±0.60 (5)	97.92±0.61 (3.5)	97.93±0.58 (1)
waveform	91.08±1.54 (4)	82.96±1.85 (6)	91.71±1.46 (1)	91.23±1.51 (2)	91.14±1.56 (3)	89.99±1.55 (5)
rank mean	3.6667	5.2778	3.1389	2.6667	3.1667	3.0833
combined	–	89.37 (5)	92.96 (1)	92.28 (3)	92.48 (2)	92.25 (4)
covtype.binary	–	55.74 (5)	59.32 (4)	61.12 (2)	61.91 (1)	59.83 (3)
skin_nonskin	–	97.80 (5)	99.26 (3)	99.87 (1)	99.81 (2)	99.16 (4)
rank mean	–	5	2.6667	2	1.6667	3.6667

the RKFDA outputs. In order to have a simpler comparison with our other results, we have compared the performance of RKFDA and DFDA using the test AUC values of our score-based classifiers.

Their dimensions, train sample sizes, number of classes, and imbalance ratios (defined now as the ratio between the maximum class size and the minimum one) are given in Table II. While some data sets are rather small (1000 patterns in `german`), others are quite large, particularly `cvotype.binary`, with 581 012 patterns. The maximum number of classes is 26 in `letter` and imbalance ratios go from near 1 in several data sets to the very large one for `shuttle`.

We will again compare here the performance of RKFDA models against DFDA models comprising 0–5 hidden layers of width 100, following the procedure of Section VI; recall that we will scale DFDA inputs to 0 mean and 1 standard deviation featurewise, while, for RKFDA models, the inputs will be scaled also featurewise to a [0, 1] range.

Optimal λ and γ values for the LIBSVM data sets are obtained as those giving a higher geometric macro-AUC in the validation subsets and these values are used to build the final models over the train or train plus validation subsets when the latter exist, and to compute the multiclass geometric macro-AUC over the test subsets. The data sets used by Mika *et al.* have 100 predefined train-test splits; as in [17], we use the first five splits for hyperparameter tuning and then report the average and the standard deviation of the test AUCs over the 100 splits (recall that the LIBSVM data sets only have a single test set and no standard deviation can be computed).

These final AUC values are given in Table V; we also omit again the results for the DFDA_1 classifier. For convenience, it is divided into three parts for the medium size LIBSVM data sets, the Mika data sets and, finally, the large LIBSVM data sets. While in several problems, DFDA and RKFDA models give similar geometric macro-AUC values, in `satimage`, `w7a`, `w8a`, `banana`, `ringnorm`, and `splice`, the AUC values of RKFDA are lower, while still in other problems, namely, `combined`, `covtype.binary`, and `skinnonskin`, we have not been able to properly hyperparameterize RKFDA on them. In fact, RKFDA and DFDA training required about the same times on the rather small two-class problems but RKFDA training took much longer times as the data set sizes increased. In particular, given that we were not able to obtain RKFDA hyperparameters for the three largest data sets, we do not give RKFDA's AUC values for them at the table's bottom. As before, each method's ranking is shown for each problem in the table; note the rankings of the first two groups run from 1 to 6 but those at its bottom going from 1 to 5. Here, also, the DFDA_0 model seems to give worse results and the DFDA_2 also appears to have a worse performance than the models with three or more hidden layers; note also that the DFDA_3 seems to have a slight edge over the others.

As in the two-class case, the results in Table V and, particularly, its rankings, also have here a descriptive nature. The number of observations of the models for which proper hyperparameters are obtained for RKFDA is now 18, a relatively low number at the edge of what is usually taken as to justify a more precise Wilcoxon-based comparison between the DFDA and RKFDA models. Nevertheless, even with these caveats in mind, we have also applied here a paired Wilcoxon signed-rank tests for the first two problem sets in Table V. Its results appear in Table VI and, as it can be seen, if the Wilcoxon approximation hypotheses hold, the null hypothesis could be rejected when comparing the DFDA_0 model against the others. On the other hand, when RKFDA is compared with the DFDA_3 and DFDA_4, we can reject the null hypothesis

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE VI
WILCOXON TESTS FOR THE LARGE SIZE PROBLEMS

	p-value	Hypothesis
RKFDA vs DFDA_0	0.00	Rejected
RKFDA vs DFDA_2	0.13	Not rejected
RKFDA vs DFDA_3	0.01	Rejected
RKFDA vs DFDA_4	0.04	Rejected
RKFDA vs DFDA_5	0.08	Not rejected
DFDA_0 vs DFDA_2	0.00	Rejected
DFDA_0 vs DFDA_3	0.00	Rejected
DFDA_0 vs DFDA_4	0.00	Rejected
DFDA_0 vs DFDA_5	0.00	Rejected
DFDA_2 vs DFDA_3	0.59	Not rejected
DFDA_2 vs DFDA_4	0.87	Not rejected
DFDA_2 vs DFDA_5	0.74	Not rejected
DFDA_3 vs DFDA_4	0.05	Rejected
DFDA_3 vs DFDA_5	0.38	Not rejected
DFDA_4 vs DFDA_5	0.72	Not rejected

at the 0.05 level and could also do so at the 0.1 level when it is compared with DFDA_5 (we could also reject it when comparing the DFDA_3 and DFDA_4 models). Therefore, while the application of a Wilcoxon test may not be rigorously justified, the results in Tables V and VI point to a slightly better performance of some of the DFDA models over the RKFDA ones on larger sample size problems.

VIII. CONCLUSION

In this paper, we have reviewed the classical (FDA) and kernel (RKFDA) approaches to Fisher's discriminant analysis following the analysis of Zhang *et al.* [18]. We have emphasized the computational complexity of RKFDA, which is cubic on the sample's size N and, hence, could become prohibitive for large data sets. Aiming to overcome this, we have proposed DFDA networks, a simple yet very powerful DNN alternative to achieve a nonlinear form of FDA adapting the least squares formulation of FDA proposed in [34] and [18].

We have compared DFDA and RKFDA on a large number of highly imbalanced two-class problems of relatively small sizes as well as in a number of two-class and multiclass problems with substantially larger sizes and some of them with large class imbalances. Our experimental results show that while RKFDA and the deep DFDA models give similar results on the smaller data sets (excluding, of course, the linear model DFDA_0), the deeper DFDA models seem to improve the purely classification performance of RKFDA on the large data sets: although the application of a Wilcoxon test is not fully justified, if done, we could reject the null hypothesis at the 0.05 level in two cases and could do so at the 0.1 level in another. Moreover, when computational complexity considerations are taken into account, DFDA models seem to clearly beat the RKFDA ones, whose training times on the larger data sets are much higher (and even failed to finish in some cases).

We can thus conclude that the new deep DFDA networks we propose here are a simple yet powerful alternative to the more established regularized kernel-based RKFDA models in general classification problems and particularly so in imbalanced ones. We have seen both approaches to have a similar (and quite good) performance on highly imbalanced, relatively small two-class problems; on larger problems, DFDA networks appear to perform better than RKFDA models from a pure classification point of view, while being clearly superior from a computational perspective. In fact, the need of handling kernel matrices makes the direct application of RKFDA models quite costly, putting large but clearly not big data problems out of their reach (a situation not dissimilar to what other kernel-based methods such as SVMs face over large data sets).

In any case, there are clear ways to improve the performance of DFDA models that seem precluded to RKFDA ones, of which we mention three. First, better results are to be expected if problem-tailored numbers of hidden layers and units are chosen instead of the fixed architectures used here. Second, the fully connected networks we have considered can be combined or substituted with any other of the many processing layer proposals that have been made for deep networks. A clear example is convolutional layers: they directly fit in the proposed deep Fisher approach and could certainly lead to an improved classification performance on problems such as image classification whose inputs have a spatial structure upon which convolutional filters act naturally.

Finally, we point out that, even after a nonlinear preprocessing, the Fisher criterion may not lead to the strongest classifiers. In fact, in the linear case, the cross-entropy loss used in logistic regression often produces better models. On the other hand, the structure induced by the Fisher criterion on the last hidden layer of a DFDA network yields new features that a more powerful classifier could take advantage of. This naturally suggests that one could add the least squares DFDA loss as a companion to another, problem-specific loss that then takes advantage of the within-class concentration and between class separation structure of the Fisher-like representation on the last hidden layer.

While that might have been rather difficult a few years ago, the most widely used deep net frameworks (Torch, TensorFlow, Theano, MXNET, or CNTK) provide backpropagation gradients automatically through their network "compilation" procedures. This means that more general loss functions than square error or cross entropy can be considered without having to program their gradients "by hand," as it was needed in the early 2000s. We can thus add the DFDA loss function into any strong deep model (such as AlexNet or VGG-16 for image processing) in a way that can enhance its performance. We are currently pursuing these and other related research goals.

ACKNOWLEDGMENT

The authors would like to thank the Centro de Computación Científica (CCC) at Universidad Autónoma de Madrid (UAM) for the use of their facilities.

REFERENCES

[1] B. Krawczyk, "Learning from imbalanced data: Open challenges and future directions," *Prog. Artif. Intell.*, vol. 5, no. 4, pp. 221–232, 2016.
[2] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intell. Data Anal.*, vol. 6, no. 5, pp. 429–449, 2002.
[3] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.
[4] N. Japkowicz, C. Myers, and M. A. Gluck, "A novelty detection approach to classification," in *Proc. 14th Int. Joint conf. Artif. Intell.*, vol. 1, Aug. 1995, pp. 518–523.
[5] M. Kubat and S. Matwin, "Addressing the curse of imbalanced training sets: One-sided selection," in *Proc. 14th Int. Conf. Mach. Learn.*, vol. 97, 1997, pp. 179–186.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

[6] T. Fawcett and F. J. Provost, "Adaptive fraud detection," *Data Mining Knowl. Discovery*, vol. 1, no. 3, pp. 291–316, 1997.

[7] P. M. Domingos, "MetaCost: A general method for making classifiers cost-sensitive," in *Proc. 5th ACM SIGKDD int. conf. Knowl. Discovery Data Mining*, vol. 99, Aug. 1999, pp. 155–164.

[8] B. Tang and H. He, "GIR-based ensemble sampling approaches for imbalanced learning," *Pattern Recognit.*, vol. 71, pp. 306–319, Nov. 2017.

[9] X. Liu, J. Wu, and Z. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Trans. Syst., Man, Cybern. B* , vol. 39, no. 2, pp. 539–550, Apr. 2009.

[10] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002.

[11] M. Galar, A. Fernandez. E. Barrenechea. H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches," *IEEE Trans. Sys. Man, Cybern. C, Appl. Reviews*, vol. 42, no. 4, pp. 463–484, Jul. 2012.

[12] Z. Zhou and X. Liu, "Training Cost-Sensitive neural networks with methods addressing the class imbalance problem," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 1, pp. 63–77, Jan. 2006.

[13] A. R. Webb and D. Lowe, "The optimised internal representation of multilayer classifier networks performs nonlinear discriminant analysis," *Neural Netw.*, vol. 3, no. 4, pp. 367–375, 1990.

[14] K. Fukunaga, *Introduction to statistical pattern recognition*, Boston, MA, USA: Academic, 1990.

[15] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. NewYork, NY, USA: Wiley, 2000.

[16] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 711–720, Jul. 1997.

[17] S. Mika, G. Rätsch, J. Weston, B.Schölkopf, A. J. Smola, and K. R. Müller, "Invariant feature extraction and classification in kernel spaces," in *Proc. Adv. Neural Inf. Process. syst.*, 1999, pp. 526–532.

[18] Z. Zhang, G. Dai, C. Xu, and M. I. Jordan, "Regularized discriminant analysis, ridge regression and beyond," *J. Mach. Learning Res.*, vol. 11, pp. 2199–2228, Aug. 2010.

[19] J. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos, "Face recognition using kernel direct discriminant analysis algorithms," *Trans. Neural Netw.*, vol. 14, no. 1, pp. 117–126, Jan. 2003.

[20] V. Sydorov, M. Sakurada, and C. H. Lampert, "Deep fisher kernels–end to end learning of the fisher kernel GMM parameters," in *Proc. Conf. Comput. Vision Pattern Recognit.*, Washington, DC, USA, Jun. 2014, pp. 1402–1409.

[21] J. Yang, A. F. Frangi, J.-Y. Yang, D. Zhang, and Z. Jin, "KPCA plus LDA: A complete kernel fisher discriminant framework for feature extraction and recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 2, pp. 230–244, Feb. 2005.

[22] J. Bergstra *et al.*, "Theano: A CPU and GPU math expression compiler," in *Proc. Python sci. Comput. conf. (SciPy)*, Jun. 2010, vol. 4, no. 3, pp. 1–25.

[23] F. Seide and A. Agarwal, "CNTK: microsoft's open-source deep-learning toolkit," in *Proc. 22nd ACM SIGKDD Inter. Conf. Knowledge Discovery Data Mining*, 2016, p. 2135.

[24] R. Collobert and K. Kavukcuoglu, "Torch7: A matlab-like environment for machine learning," in *Proc. BigLearn, NIPS Workshop*, vol. 5, no. 10, Dec. 2011, pp. 1–6.

[25] T. Chen *et al.* (2015). "MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems." [Online]. Available: https://arxiv.org/abs/1512.01274

[26] Google. *Tensorflow, An Open Source Software Library for Machine Intelligence*. Accessed: Mar. 2019. [Online]. Available: https://www.tensorflow.org/

[27] C. S. Cruz and J. R. Dorronsoro, "A nonlinear discriminant algorithm for feature extraction and data classification," *IEEE Trans. Neural Netw.*, vol. 9, no. 6, pp. 1370–1376, Nov. 1998.

[28] A. Stuhlsatz, J. Lippel, and T. Zielke, "Feature extraction with deep neural networks by a generalized discriminant analysis," *IEEE Trans. Neural Netw. Learn. Sys.*, vol. 23, no. 4, pp. 596–608, Apr. 2012.

[29] M. Dorfer, R. Kelz, and G. Widmer. (Nov. 2015). "Deep linear discriminant analysis." [Online]. Available: https://arxiv.org/abs/1511.04707

[30] L. Wu, C. Shen, and A. van den Hengel, "Deep linear discriminant analysis on fisher networks: A hybrid architecture for person re-identification," *Pattern Recognit.*, vol. 65, pp. 238–250, May 2017.

[31] Q. Tian, T. Arbel, and J. J. Clark, "Deep LDA-pruned nets for efficient facial gender classification," in *Proc.IEEE Conf. Comput. Vision Pattern Recognit. (CVPR) Workshops*, Honolulu, HI, USA, Jul. 2017, pp. 512–521.

[32] K. Lee and J. Kim, "Font Size: On the equivalence of linear discriminant analysis and least squares," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 2736–2742.

[33] C. H. Park and H. Park, "A relationship between linear discriminant analysis and the generalized minimum squared error solution," *SIAM J. Matrix Anal. Applications*, vol. 27, no. 2, pp. 474–492, Jun. 2005.

[34] J. Ye, "Least squares linear discriminant analysis," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 1087–1093.

[35] D. Cai, X. He, and J. Han, "SRDA: An efficient algorithm for large-scale discriminant analysis," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 1, pp. 1–12, Jan. 2008.

[36] D. Díaz-Vico A. Omari, A. Torres-Barrán, and J. R. Dorronsoro, "Deep fisher discriminant analysis," in *Proce. 14th Int. Work-conf. Artif. Neural Netw.*, 2017, pp. 501–512.

[37] J. H. Friedman, "Regularized discriminant analysis," *J. Amer. Statist. Assoc.*, vol. 84, no. 405, pp. 165–175, 1989.

[38] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, 2006.

[39] C. Ferri, J. Hernandez-Orallo, and M. Salido, "Volume under the ROC surface for multi-class problems," in *Proc. 14th Eur. Conf. Machine Learn.*, 2003, pp. 108–120.

[40] F. Provost and P. Domingos, "Tree induction for probability-based ranking," *Machine Learn.*, vol. 52, no. 3, pp. 199–215, Sep. 2003.

[41] G. Tsoumakas, I. Katakis, and I. P. Vlahavas, "Mining multi-label data," in *Data Mining Knowl. Discovery Handbook*, 2nd ed. New York, NY, USA: Springer, 2010, pp. 667–685.

[42] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[43] D. J. Hand and R. J. Till, "A simple generalisation of the area under the ROC curve for multiple class classification problems," *Machine Learn.*, vol. 45, no. 2, pp. 171–186, Nov. 2001.

[44] D. Díaz-Vico, A. R. Figueiras-Vidal, and J. R. Dorronsoro, "Deep MLPS for imbalanced classification," in *Proc. Int. Joint Conf. Neural Netw.*, 2018, pp. 1–7.

[45] Y. Yang, "An evaluation of statistical approaches to text categorization," *Inf. Retrieval*, vol. 1, nos. 1–2, pp. 69–90, 1999.

[46] J. Alcalá-Fdez *et al.* "Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework," *J. Multiple-Valued Logic Soft Comput.*, vol. 17, nos. 2–3, pp. 255–287, 2011.

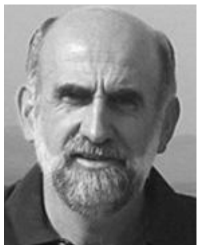
[47] S. Gónzalez S. García, M. LázaroA. R. Figueiras-Vidal, and F. Herrera, "Class switching according to nearest enemy distance for learning from highly imbalanced data-sets," *Pattern Recognit.*, vol. 70 pp. 12–24, 2017.

[48] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in python," *J. achine Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[49] S. Wang and X. Yao, "Multiclass imbalance problems: Analysis and potential solutions," *IEEE Trans. Sys. Man Cybern., B (Cybern.)*, vol. 42, no. 4, pp. 1119–1130, 2012.

David Díaz-Vico received the M.Sc. degree in mathematics and the M.S.E. degree in computer science from the Universidad Autónoma de Madrid, Madrid, Spain, in 2012, where he is currently pursuing the Ph.D. degree in computer science.

He was a Data Scientist with Accenture Analytics, Madrid, Telefónica Research and Development, Madrid, and Instituto de Ingeniería del Conocimiento (IIC), Madrid, for more than 8 years. He has authored several papers. He holds patents in machine learning and applications.



José R. Dorronsoro received the Ph.D. degree with Washington University in St. Louis, St. Louis, MO, USA, in 1981.

He is currently a Professor of computer science with the Universidad Autónoma de Madrid, Madrid, Spain. He has directed eight Ph.D. theses and has been a Leader of a large number of research and innovation projects. He is also a Senior Scientist with the Instituto de Ingeniería del Conocimiento (IIC), Madrid, where he leads IIC's research and innovation on the application of machine learning to areas such as renewable energy. He has authored more than 100 scientific papers in mathematical analysis, machine learning, and applications.