



Repositorio Institucional de la Universidad Autónoma de Madrid

<https://repositorio.uam.es>

Esta es la **versión de autor** de la comunicación de congreso publicada en:
This is an **author produced version** of a paper published in:

16th International Conference on Artificial Intelligence and Soft Computing,
ICAISC 2017, Zakopane; Poland, June 11-15, 2017. Lecture Notes in Computer
Science, Volumen 10306. Springer, 2017. 501-5012

DOI: http://dx.doi.org/10.1007/978-3-319-59147-6_43

Copyright: © Springer International Publishing AG 2017

El acceso a la versión del editor puede requerir la suscripción del recurso
Access to the published version may require subscription

Deep Fisher Discriminant Analysis

David Díaz-Vico, Adil Omari, Alberto Torres-Barrán, José R. Dorronsoro

Dpto. Ing. Informática and Instituto de Ingeniería del Conocimiento
Universidad Autónoma de Madrid

Abstract. Fisher Discriminant Analysis' linear nature and the usual eigen-analysis approach to its solution have limited the application of its underlying elegant idea. In this work we will take advantage of some recent partially equivalent formulations based on standard least squares regression to develop a simple Deep Neural Network (DNN) extension of Fisher's analysis that greatly improves on its ability to cluster sample projections around their class means while keeping these apart. This is shown by the much better accuracies and g scores of class mean classifiers when applied to the features provided by simple DNN architectures than what can be achieved using Fisher's linear ones.

Keywords: Linear Discriminant Analysis, Deep Neural Networks, Non-linear Classifiers.

1 Introduction

Fisher' Linear Discriminant Analysis (FLDA from now on) is a very well known linear dimensionality reduction/feature extraction technique that, while able to provide useful data representations, does not intend, in principle, to solve a given classification problem and, thus, it has known only a limited use as a tool to build classifiers. There may be two main reasons for this. The first one is its linear nature. In fact, while quite attractive, its main goal of concentrating the projected features around their class means while keeping those means apart can, for most problems, only be partially achieved by FLDA's linear projections. Moreover, in order to build a powerful classifier, we would most likely need to apply a non-linear classifier to the FLDA features, but this combination of a linear projection followed by a non-linear classifier may at best be only competitive with the direct application of the non-linear classifier over the initial features.

In any case, FLDA has been successfully applied in a number of problems, most notably on face recognition where the original Fisher Face method [1] has been progressively improved to become the state of the art in this area. A natural idea is thus to somehow extend FLDA to a nonlinear procedure by applying it after some non-linear pre-processing of the original features. This is the goal of Kernel Discriminant Analysis (KDA, [12]), which addresses binary problems and where the well known reduction for such problems of FLDA to a linear regression problem [4], Chap. 5, is extended into a kernel setting.

As just said, KDA only is available in principle for binary problems. A different non-linear extension that works for multiclass problems was proposed in [14,8], where FLDA is applied on the nonlinear features obtained after processing the original features by the hidden layers of a standard feed-forward multilayer perceptron (MLP). For this, the non-linear z features are first obtained on the MLP's last hidden layer as $z = f(x, W)$, where W denotes the MLP weights and biases up to the last hidden layer and $f(x, W)$ the effect of the MLP forward pass on the original features x . Then, FLDA's standard criterion function is used on these z to get FLDA's projecting matrix A by minimizing one of the several criterion functions J proposed in FLDA. Thus, we can view the overall cost function $J(A, W)$ as depending separately on the FLDA's projection matrix A and on the MLP's weight and bias set W , which suggests to optimize $J(A, W)$ alternating the minimization on W and A . More precisely, for a given W_k and $z^k = f(x, W_k)$, we first derive the A_k matrix by minimizing $J(A, W_k)$ by FLDA's standard eigen-procedure. Then, the new W_{k+1} are derived minimizing $\mathcal{J}(W) = J(A_k, W)$; as shown in [8], the gradient $\nabla_W \mathcal{J}$ can be explicitly computed by backpropagation. Notice that this ensures $J(A_k, W_{k+1}) < J(A_k, W_k) < J(A_{k-1}, W_k)$ and this alternating two-step process can be iterated towards a minimum (A^*, W^*) of J . As in [14,8], we shall refer to this procedure as Non-linear Discriminant Analysis (NLDA).

While in principle any number of hidden layers could be considered, only a single hidden layer was used in [14,8], as was customarily done before the advent of deep neural networks (DNN). These have provided two main insights. The first one is a better understanding of network initialization plus efficient minimization and regularization procedures, which have made largely routine the previously near impossible training of many layered networks. The second one is the availability of symbolic gradient computation in platforms such as Theano [2] or TensorFlow [9] that make it possible the consideration of cost functions much more general than the square error or cross entropy usually applied in neural network-based regression or classification.

Both could be applied to improve on NLDA, either by keeping the alternating minimization of $J(A, W)$ but working with deeper networks or, simply by applying symbolic differentiation on A, W directly to the joint $J(A, W)$ cost function. Here, however, we will follow a much more direct approach by taking advantage of the results in [13], [15] and [16], where a link is established between a concrete formulation of FLDA and a related Least Squares Regression (LSR) problem with a particular, class-based target choice. We shall make extensive use of this approach which we will call Least Squares Discriminant Analysis (LSDA). More precisely, it is shown in [16,15] that there is an isometry between projections derived from a specific FLDA formulation and those derived from the solution of the LSR problem. In turn, this implies that if distance based classifiers such as k -Nearest Neighbors or (as done here) minimum class-mean distances are used, either a renormalized FLDA or LSR approaches result in equivalent classifiers.

Once the previous set up is available, it is straightforward to carry the LSR problem into a DNN setting, working with the same targets as in the linear

case but which now are to be approximated by the outputs of a suitable DNN. This is the approach we shall follow here and, besides a short, self-contained presentation of the LSR and FLDA equivalence in [16,15] our contributions are the following:

- The proposal of Deep Fisher Discriminant Analysis, DFDA, along the lines just summarized.
- A comparison of DFDA with classical FLDA over several, large size, binary and multiclass datasets, that shows a much better performance of DFDA.

The paper is organized as follows. In Section 2 we shall review classical FLDA as well as the distance-based classifier equivalence established in [16,15] between classical FLDA and a concrete LSR problem. Deep Fisher Discriminant Analysis is introduced in Section 3 and in Section 4 we will compare its performance with that of classical LFDA over several relatively large multiclass and, in some cases, imbalanced problems. As we shall see, the accuracies and g scores of the DFDA classifiers are substantially better. Finally the paper closes with a brief discussion and pointers to further work.

2 Fisher's Linear Discriminant Analysis and Least Squares Counterparts

2.1 Fisher's Linear Discriminant Analysis

We first briefly review classical FLDA. As mentioned, its goal is to linearly project the original patterns in such a way that these projections are close to their class means while these class means are kept apart. Several criterion functions can be used for this goal and many of them are in fact equivalent; see [6], Section 10.2. Here we will seek to maximize the trace criterion

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

where A is the projection matrix, S_B and S_T denote the between-class and total covariance matrices respectively of the sample patterns and s_B and s_T are their counterparts for the projections $z = Ax$; see [16], Subsection 2.2 for more details. Solving $\nabla_A g = 0$ leads to

$$0 = -2S_T A s_T^{-1} s_B s_T^{-1} + 2S_B A s_T^{-1}$$

i.e., $S_T^{-1} S_B A = A s_T^{-1} s_B$ or, up to an invertible transformation of A (which won't change the cost function $g(W)$), to

$$S_T^{-1} S_B A = A \Lambda, \quad (2)$$

with Λ the non-zero eigenvalues of $S_T^{-1} S_B$ (and of $s_T^{-1} s_B$). Thus, for such an A we have

$$g(A) = \text{trace}(s_T^{-1} s_W) = \text{trace } \Lambda = \lambda_1 + \dots + \lambda_q,$$

which we maximize by sorting the eigenvalues in A in descending order and selecting the $q = \min\{d, C - 1\}$ largest ones and some conveniently normalized associated eigenvectors; here d is pattern dimension and C the number of classes; q is then the rank of S_B . Notice that the minimizer of (1) is not uniquely defined, and some normalization has to be introduced; an usual choice is to impose $A^t S_T A = I_q$.

In some problems S_T may be ill conditioned and not have full rank. One possibility in this case is to use the Moore–Penrose inverse of S_T ; another, and the one we follow here, is Regularized Discriminant Analysis [5], where we work with $S_t + \lambda I$ for an appropriate $\lambda > 0$.

2.2 Least Squares Regression and Fisher’s Linear Discriminant Analysis

It is a very well known result [4] that for 2-class problems, a solution to FLDA can be obtained solving a Least Squares Regression (LSR) problem

$$\min \frac{1}{2} \|Y - \mathbf{1}_n w_0 - Xw\|^2 \quad (3)$$

where $\mathbf{1}_n$ is the all ones vector, X is the $n \times d$ data matrix and Y is an appropriate target matrix defined by setting $Y_p = n/n_1$ if x_p belongs to class 1 and $Y_p = -n/n_2$ if x_p belongs to class 2. If m_1 and m_2 denote the class means on the original features, it can be then checked that $w = S_T^{-1}(m_1 - m_2)$. Since now $S_B = (m_1 - m_2)(m_1 - m_2)^t$, it follows that $S_T^{-1} S_B w = w\gamma$, with $\gamma = (m_1 - m_2)^t S_T^{-1} (m_1 - m_2)$, i.e., w is an eigen-solution of (2) and, hence, a dilation of a FLDA’s projection vector computed as in the previous subsection.

There have been several attempts to carry this result to a multiclass setting. Among the most successful ones are the proposals by Park and Park [13] and the somewhat simpler one in Zhang *et al.* [16], which we follow here and briefly explain next in a much more concise way.

Consider again the LSR problem (3), where the target matrix Y to be adequately chosen. For simplicity we assume that S_T is regular; if not, we can simply replace it with $S_T + \lambda I$ for some $\lambda > 0$. The optimal LSR solution is then

$$w = S_T^{-1} X^t H Y,$$

where H is the **centering matrix**

$$H = I_n - \frac{1}{n} \mathbf{1}_n \mathbf{1}_n^t;$$

in particular, $X^t H$ is the $n \times d$ matrix whose p -th row equals $x_p^t - m^t$. The key assumption is now that we can choose a target matrix Y such that we can write S_B as

$$S_B = X^t H Y Y^t H X.$$

Assuming this, let us write $Q = X^t H Y$ (and, thus, $S_B = Q Q^t$); we show next how can we transform the solution $w = S_T^{-1} X^t H Y = S_T^{-1} Q$ of (3) into an eigen-solution A of (2). Consider the semidefinite positive matrix $R = Q^t S_T^{-1} Q$ whose

SVD decomposition is $R = \tilde{V}\tilde{\Gamma}\tilde{V}^t$. Assuming for simplicity that $\text{rank}(Q) = \text{rank}(S_B) = C - 1$, at least one of the diagonal elements in $\tilde{\Gamma}$ will be zero; reordering $\tilde{\Gamma}$ if necessary, we assume it to be the element in the (C, C) matrix entry. We can thus drop the last row and column of $\tilde{\Gamma}$ (that are 0) to get a $(C - 1) \times (C - 1)$ diagonal matrix Γ , and the last column of \tilde{V} to get a $C \times (C - 1)$ matrix V that verifies $V^t V = I_{C-1}$ and for which we can also write R as $R = V\Gamma V^t$.

Now it is easy to check that the pair $(A = wV\Gamma^{-1/2}, \Gamma)$ is an eigensolution of (2) with normalization $A^t S_T A = I_q$, for we have

$$\begin{aligned} S_T^{-1} S_B A &= S_T^{-1} Q Q^t w V \Gamma^{-1/2} = w Q^t S_T^{-1} Q V \Gamma^{-1/2} = w R V \Gamma^{-1/2} \\ &= w V \Gamma V^t V \Gamma^{-1/2} = A \Gamma. \end{aligned}$$

In other words, if we choose Y adequately, from $Q = X^t H Y$ we can derive the LSR solution w , the SVD decomposition (V, Γ) of $Q^t S_T^{-1} Q$ and the FLDA eigensolution $A = w V \Gamma^{-1/2}$. This combines Algorithm 4 in [16] and the discussion in its section 6 to derive a FLDA solution A from the LSR solution w .

As mentioned, one thing to consider is the possibility of S_T being singular. This can be easily handled now by working with a Ridge Regression problem, i.e., solving for an appropriate $\lambda > 0$

$$\min \frac{1}{2} \|Y - \mathbf{1}_{nC} w_0^t - X w\|^2 + \frac{\lambda}{2} \text{trace}(w^t w),$$

where $\mathbf{1}_{nC}$ denotes the $n \times C$ all ones matrix and w_0 is a C -dimensional vector.

2.3 Equivalence of Distance Classifiers

As in [16], let's consider the projection matrix $B = A\Gamma^{1/2} = wV$ instead of FLDA's standard A . It is easy to see that B is also an eigen-solution of (2) associated to the normalization $B^t S_T B = \Gamma$. Let $y = w_0 + w^t x$ and $\omega = B^t x$ be the LSR and B -eigen-projections of a pattern x , respectively. We then have

$$\|\omega - \omega'\|^2 = (x - x')^t B B^t (x - x') = (x - x')^t w V V^t w^t (x - x') = \|y - y'\|^2.$$

Thus, any Euclidean distance-based classifier will give the same results when applied to the B -eigen-projections ω than when applied to the LSR ones y . This will be the case for a k -Nearest Neighbor classifier and also for the nearest class mean classifier

$$\delta_{NCM}(x) = \arg \min_c \|w^t x - w^t \bar{x}_c\| = \arg \min_c \|y - \bar{y}_c\|,$$

which we will use here. In other words, we can obtain a FLDA-like nearest class mean classifier directly from the LSR solution, without having to perform the eigen-analysis that FLDA requires. We will call this procedure, proposed in [16,15], Least Squares Discriminant Analysis, LSDA, and take advantage of this in Section 3 to define our deep Fisher classifiers but, before that, we close this section with two examples of suitable target matrices Y .

2.4 Two Examples

The well known relationship between the LSR and FLDA solutions for 2 class problems mentioned above also follows easily from the previous discussion. In fact, using the target vector $Y_p = n/n_1$ if x_p is in class 1 and $Y_p = -n/n_2$ if x_p is in class 2, then we have $Q = X^t H Y = m_1 - m_2$ and, hence, $S_B = (m_1 - m_2)(m_1 - m_2)^t = Q Q^t$. Besides, $R = Q S_T^{-1} Q = (m_1 - m_2)^t S_T^{-1} (m_1 - m_2) = \gamma$, with a trivial SVD decomposition $R = 1\gamma 1$ and, thus, we have here $A = w \cdot 1 \cdot \gamma^{-1/2} = S_T^{-1} (m_1 - m_2) \gamma^{-1/2}$ and $B = A \gamma^{1/2} = S_T^{-1} (m_1 - m_2) = w$, i.e, the ω and y projections now coincide.

For the general multi-class case, it is shown in [16], equation (4), that we can write

$$S_B = X^t H E \Pi^{-1} E^t H X = X^t H E \Pi^{-1/2} \Pi^{-1/2} E^t H X,$$

where Π is the diagonal matrix with $\Pi_{cc} = n_c$ and E is the $n \times C$ indicator matrix with rows e_p such that if x_p is in class c , $e_{pc} = 1$ and $e_{pc'} = 0$ for $c' \neq c$. Thus here we can take $Q = X^t H E \Pi^{-1/2}$ and it is also shown in [16], equation (25) that we can write the LSR solution $w = S_T^{-1} X^t H Y$ as

$$w = S_T^{-1} X^t H E \Pi^{-1/2} = S_T^{-1} Q$$

if we use $Y = H E \Pi^{-1/2}$ as the target matrix. It is now easy to see that for such Y and x_p in class c , we have $Y_{pc} = \frac{n - n_c}{n \sqrt{n_c}}$, and $Y_{pc'} = -\frac{\sqrt{n_c}}{n}$ otherwise. These are the targets we shall use in the next section.

3 Deep Fisher Discriminant Analysis

We have just argued how we can obtain for a general C class problem a nearest class-mean classifier equivalent to one acting on the B -based Fisher projections by performing the following steps:

1. For a given training set D_{tr} , solve for a data matrix X_{tr} , class indicator matrix E_{tr} and targets $Y_{tr} = H E_{tr} \Pi^{-1/2}$ the LSR problem

$$\min \frac{1}{2} \|Y_{tr} - \mathbf{1}_{nC} w_0^t - X_{tr} w\|^2,$$

obtaining the optimal $d \times C$ matrix w^* and C -dimensional vector w_0^* .

2. Compute the projections $y = w_0^* + (w^*)^t x$ for $x \in D_{tr}$ and their class means $\bar{y}_c = w_0^* + (w^*)^t \bar{x}_c$.
3. Assign an x in a test sample D_{ts} to the class whose mean the projection $y = w_0^* + (w^*)^t x$ it is closest to; that is, to the class c^* for which

$$c^* = \arg \min_c \{\|y - \bar{y}_c\|\}.$$

Now, a natural idea to extend this to a non linear setting is to perform the LSR computations on features z obtained by a non-linear processing $z = \Phi(x)$ of the original features x . An example of this is Kernel Discriminant Analysis, KDA [12,16], where a certain generalized eigenvalue problem involving the kernel matrix $K = ZZ^t$ is solved, with the matrix Z being $Z = \Phi(X)$; in particular the projections z are not needed explicitly as they enter the computations through a kernel k such that $z \cdot z' = \Phi(x) \cdot \Phi(x') = k(x, x')$. As it is often the case in kernel methods, handling the $n \times n$ matrix K can be too costly in large sample problems and some suitable low rank approximation would have to be used.

A simpler alternative, better suited in principle for large sample problems, is to derive the z features using a straightforward DNN extension of the previous linear setup; more precisely, in a DNN setting we would

1. Solve over a training set D_{tr} the LSR problem

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

with Y_{tr} the previous training target matrix and the p -th row of the matrix $f(X_{tr}, \mathcal{W})_p$ is given by $f(X_{tr}, \mathcal{W})_p = f(x_p, \mathcal{W})$, where $f(x, \mathcal{W})$ is the transfer function of a deep network with linear outputs and overall weight set \mathcal{W} ; we thus obtain an optimal DNN weight set \mathcal{W}^* .

2. Compute the projections $y_p = f(x_p, \mathcal{W}^*)$ over D_{tr} .
3. Assign a new $x \in D_{ts}$ to the class whose mean the projection $y = f(x, \mathcal{W}^*)$ is closest to.

Writing $\mathcal{W}^* = (w_0^*, w^*, W^*)$ with w_0^*, w^* the linear output weights, these optimal w_0^*, w^* solve the LSR problem (3) over the last hidden layer features $z = \Phi(x, W^*)$, with Φ the DNN transfer function up to the last hidden layer. Thus, the class mean classifier of the full DNN is equivalent to a class mean classifier over some FLDA projections of the z patterns in the last hidden layer which, in turn, are also learned by tuning the W component of the overall weight \mathcal{W} . In other words, the DNN also performs a particular kind of representation learning, as in this case it learns in its last hidden layer new features that have been optimized to perform FLDA on them. We will call this Deep Fisher Discriminant Analysis, or DFDA.

As in the linear case, we may avoid singularity issues here by adding a regularization term, i.e., solving for instance

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

with \widetilde{W} the components of W excluding the biases at each hidden layer. This is the cost function we will use. Of course, in the deep case one may use other regularization terms for the W components of the overall weight structure \mathcal{W} (such as, for instance, dropout), but the term $\frac{\lambda}{2} \text{trace}(w^t w)$ should be kept in any case for the linear weights w .

Table 1. Train sample size, dimension, number of classes and ratio between the maximum and minimum class sizes for the considered datasets.

Problem	n. patterns	dimension	n. classes	class ratios
combined	78,823	100	3	2.156
dna	2,000	180	3	2.265
ijcnn1	49,990	22	2	9.301
letter	15,000	16	26	1.128
mnist	60,000	784	10	1.244
pendigits	7,494	16	10	1.085
satimage	4,435	36	6	2.583
shuttle	43,500	9	7	5,684.667
w7a	24,692	300	2	32.368
w8a	49,749	300	2	32.637
usps	7,291	256	10	2.203

4 Numerical Experiments

4.1 Datasets and Quality Measures

We will consider the datasets `SensIT Vehicle (combined)`, `dna`, `ijcnn1`, `letter`, `mnist`, `pendigits`, `satimage`, `shuttle`, `w7a`, `w8a` and `usps`. All of them also have separate, well defined train-test splits and, except `mnist`, are available on the [Datasets section](#) of the LIBSVM web site; for `mnist` we have used Scikit-Learn to fetch it from [mldata](#). We have put an emphasis in relatively large, multiclass datasets; in Table 1 we give their dimension, total number of train patterns, number of classes and their maximum class size ratios, i.e., the ratio of the maximum class size to the minimum one. Data sizes go from 2,000 (`dna`) to 78,823 (`combined`) and the number of classes ranges from 2 to 26 (`letter`); while some of them are quite balanced, others (`w7a`, `w8a` and particularly `shuttle`) present large class imbalances, having class size ratios $\gg 1$. Because of this, the main quality measure we will use for model evaluation will be the g -score, i.e., the geometric mean of the different class sensitivities:

$$g = \left(\prod_{c=1}^C S_c \right)^{1/C} = \left(\prod_{c=1}^C \frac{m_{cc}}{\sum_j m_{cj}} \right)^{1/C},$$

where m_{cj} is the (c, j) entry of the confusion matrix, that is, the number of class c patterns that are assigned to class j . The g -score measure is often used in imbalanced classification as it is more robust to markedly different class sizes than accuracy, easily achieved by assigning small class patterns to the largest class. Because of this we will also use g as the merit function for hyper-parameter selection of both linear and deep models. Nevertheless, we shall also report the accuracies $a = \frac{\sum_c m_{cc}}{\sum_{c,j} m_{cj}}$.

4.2 Deep Model Universal Approximation Capabilities

The goal when using deep versions of FLDA is to obtain better representations in the last hidden layer of which the final, Fisher-like, linear transform can take advantage. In an extreme perspective, overfitting should concentrate each class around its mean while keeping these classes far apart. Once vanishing gradients are avoided and proper training is possible, the simplest way to overfit a dataset is to work with a deep enough network with rather large hidden layers, which ensures a large number of weights. We will do so here, building for each dataset four DFDA models having between 2 and 5 fully connected hidden layers with 100 units each but controlling overfit with a proper penalty. For most problems the number of weights approximately varies thus between 20,000 and 50,000; on the other hand, given its pattern dimension, for `mnist` the first hidden layer already has 78,400 weights. We stress that our main goal here is not to obtain top quality models; for instance, for `mnist`, convolutional networks would be needed for this and the fully connected layers be much larger than the ones considered here. Instead, our main goal here is to measure DFDA’s performance and compare it against that of FLDA.

As a benchmark reference we will build an LSDA model for each dataset using the class `Ridge` in `scikit-learn`; for DFDA models we will use the `MLPRegressor` class also in `scikit-learn`. `MLPRegressor` only allows for deep MLPs with fully connected layers and L_2 penalization; we will use `relu` activations and the `adam` solver. This solver is a faster, more stable version of gradient descent but convergence may still be slow. Moreover, the targets Y_{tr} are rather small to begin with, so the convergence tolerance should also be small. Because of this we will work with a maximum number of 20,000 iterations and use a tolerance of 10^{-9} ; other solver parameters are left at their default values. A more powerful alternative could have been to work with a general DNN framework such as Keras [3] that has Theano or TensorFlow as backends and offers a much wider range of network architectures (including for instance convolutional layers) or penalties (L_1 or dropout). However, most of the problems considered do not lend themselves to, say, using convolutional layers and, on the other hand, the structural simplicity of `MLPRegressor` models results in a much faster training.

We will work with mini-batch sizes `min(200, num_patterns)`, i.e., the default for the `MLPRegressor` class. Therefore, the only hyper-parameter we have to set is the L_2 penalty `alpha` of the LSDA and DFDA models. For both cases we will select it using the `RandomizedSearchCV` model selection framework in `scikit-learn`. To select the optimal `alpha` values, we will perform 100 uniform random searches of `alpha` values in a range $(0, \alpha_{\max})$, averaging for each one its g scores over 10 cross validation folds built on the training set and retaining the value giving the largest validation score. The test g -scores are reported in Table 2 and the accuracies in Table 3. In each case the reported test g and accuracy values are obtained training 10 DLDA models with different random initializations, averaging their outputs as well as those of the corresponding test patterns and computing the class predictions and the test confusion matrix over these averages. The tables also give the rankings of each model over the different problems.

Table 2. Test g scores of the LSDA and of the DFDA models with 2, 3, 4 and 5 100 unit, hidden layers. We write in parenthesis the g score ranking of each model over the different problems and the corresponding ranking means in the last line.

Problem	LSDA	DFDA			
		2 HL	3 HL	4 HL	5 HL
combined	0.780 (5)	0.800 (3)	0.791 (4)	0.817 (2)	0.831 (1)
dna	0.940 (5)	0.952 (3)	0.952 (3)	0.955 (1)	0.953 (2)
ijcnn1	0.772 (5)	0.826 (3)	0.809 (4)	0.946 (2)	0.955 (1)
letter	0.683 (5)	0.924 (4)	0.936 (3)	0.952 (2)	0.957 (1)
mnist	0.870 (5)	0.929 (4)	0.947 (3)	0.952 (2)	0.962 (1)
pendigits	0.805 (5)	0.975 (4)	0.976 (3)	0.978 (2)	0.981 (1)
satimage	0.808 (5)	0.883 (2)	0.887 (1)	0.876 (3)	0.873 (4)
shuttle	0.565 (3)	0.000 (5)	0.414 (4)	0.838 (2)	0.979 (1)
w7a	0.794 (3)	0.694 (5)	0.776 (4)	0.852 (1)	0.849 (2)
w8a	0.781 (5)	0.837 (1)	0.827 (3)	0.827 (3)	0.831 (2)
usps	0.870 (4)	0.943 (2)	0.949 (1)	0.939 (3)	0.939 (3)
rank mean	4.55	3.27	3.00	2.09	1.73

As it can be seen, the deep DFDA models clearly improve on the LSDA ones in terms of g scores and accuracies, with the best results usually obtained with the largest 5-hidden layer network. This is particularly remarkable on the `shuttle` problem. Notice in Table 2 the g score of the 2-hidden layer network is 0, due to no pattern in the smallest class being correctly classified (this class has about 50,000 times less patterns than the biggest one).

5 Discussion and Further Work

While elegant and enticing, classical Fisher Linear Discriminant Analysis (FLDA) has fallen into some disuse, partly because of its linear nature but also because of the eigenanalysis it requires, which doesn’t lend itself to be considered over very large datasets or to be learned in an iterative basis. Most of these difficulties are greatly alleviated when instead of a “pure” FLDA approach, one follows the equivalent LSR set-up proposed in [16,15] and discussed above. Moreover, this lends itself into a natural extension to a Deep Neural Network setting, pairing the LSR target matrix with a highly complex deep pattern processing.

This is our approach here, where we have shown how simple 2-to-5 layer networks can noticeably improve the performance of FLDA. We have applied some of the latest tools in deep networks, such as Glorot initialization [7], RELU activations [11] or ADAM optimizers [10] but, in any case, the networks considered are relatively small and rather simple. There are thus several venues we can follow to improve on the results reported here. For instance, we can use other, more specialized, DNN architectures, easily available through the `keras` wrapper for Theano or TensorFlow, which may include convolutional layers for highly structured inputs such as `mnist`. These layers could also be helpful in problems such

Table 3. Test accuracies of the LSDA and of the DFDA models with 2, 3, 4 and 5 100 unit, hidden layers. Again, we write in parenthesis the accuracy ranking of each model over the different problems and the corresponding ranking means in the last line.

Problem	LSDA	DFDA			
		2 HL	3 HL	4 HL	5 HL
combined	0.770 (5)	0.791 (3)	0.782 (4)	0.808 (2)	0.819 (1)
dna	0.927 (5)	0.957 (3)	0.954 (4)	0.960 (1)	0.958 (2)
ijcnn1	0.855 (5)	0.891 (3)	0.883 (4)	0.977 (2)	0.984 (1)
letter	0.694 (5)	0.925 (4)	0.937 (3)	0.952 (2)	0.957 (1)
mnist	0.873 (5)	0.931 (4)	0.947 (3)	0.952 (2)	0.962 (1)
pendigits	0.825 (5)	0.975 (4)	0.976 (3)	0.978 (2)	0.981 (1)
satimage	0.835 (5)	0.891 (2)	0.897 (1)	0.885 (3)	0.882 (4)
shuttle	0.913 (5)	0.943 (3)	0.934 (4)	0.989 (1)	0.986 (2)
w7a	0.984 (3)	0.843 (5)	0.978 (4)	0.988 (2)	0.989 (1)
w8a	0.984 (5)	0.988 (2)	0.988 (2)	0.987 (4)	0.989 (1)
usps	0.883 (5)	0.949 (2)	0.954 (1)	0.946 (3)	0.944 (4)
rank mean	4.82	3.18	3.00	2.18	1.73

as person identification, where the current state of the art are the Fisher Face procedures, that apply a suitable version of Fisher analysis over face images. We could also consider DFDA networks as representation learners, using the network outputs of the last hidden layer as features upon which classifiers stronger than nearest neighbors or class mean distances could be applied. This may be particularly suitable for imbalanced problems. In this line it is also interesting to compare the performance of Deep Fisher networks with that of other non linear Fisher extensions, such as KDA. We are currently studying these issues.

Acknowledgments. With partial support from Spain’s grants TIN2013-42351-P, TIN2016-76406-P, TIN2015-70308-REDT and S2013/ICE-2845 CASI-CAM-CM. Work supported also by project FACIL–Ayudas Fundación BBVA a Equipos de Investigación Científica 2016, the UAM–ADIC Chair for Data Science and Machine Learning and Instituto de Ingeniería del Conocimiento. The third author is also supported by the FPU–MEC grant AP-2012-5163. We gratefully acknowledge the use of the facilities of Centro de Computación Científica (CCC) at UAM.

References

1. Belhumeur, P.N., Hespanha, J.P., Kriegman, D.J.: Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Trans. Pattern Anal. Mach. Intell.* 19(7), 711–720 (1997), <http://dx.doi.org/10.1109/34.598228>
2. Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., Bengio, Y.: Theano: a CPU and GPU math expression compiler. In: *Proceedings of the Python for Scientific Computing Conference (SciPy)* (Jun 2010), oral Presentation

3. Chollet, F.: Keras: Deep learning library for theano and tensorflow (2015), <https://github.com/fchollet/keras>
4. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification (2nd Edition). Wiley-Interscience (2000)
5. Friedman, J.H.: Regularized discriminant analysis. *Journal of the American Statistical Association* 84(405), 165–175 (1989)
6. Fukunaga, K.: Introduction to statistical pattern recognition. Computer science and scientific computing, Academic Press, Boston (1990)
7. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: *JMLR W&CP: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*. vol. 9, pp. 249–256 (May 2010)
8. González, A.M., Dorronsoro, J.R.: Natural learning in NLDA networks. *Neural Networks* 20(5), 610–620 (2007), <http://dx.doi.org/10.1016/j.neunet.2006.09.014>
9. Google: Tensorflow, an open source software library for machine intelligence, <https://www.tensorflow.org/>
10. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. *CoRR* abs/1412.6980 (2014), <http://arxiv.org/abs/1412.6980>
11. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C., Bottou, L., Weinberger, K. (eds.) *Advances in Neural Information Processing Systems* 25, pp. 1097–1105. Curran Associates, Inc. (2012), <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>
12. Mika, S., Rätsch, G., Weston, J., Schölkopf, B., Smola, A.J., Müller, K.: Invariant feature extraction and classification in kernel spaces. In: *Advances in Neural Information Processing Systems* 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]. pp. 526–532 (1999)
13. Park, C.H., Park, H.: A relationship between linear discriminant analysis and the generalized minimum squared error solution. *SIAM J. Matrix Analysis Applications* 27(2), 474–492 (2005), <http://dx.doi.org/10.1137/040607599>
14. Santa Cruz, C., Dorronsoro, J.R.: A nonlinear discriminant algorithm for feature extraction and data classification. *IEEE Trans. Neural Networks* 9(6), 1370–1376 (1998), <http://dx.doi.org/10.1109/72.728388>
15. Ye, J.: Least squares linear discriminant analysis. In: *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007)*, Corvallis, Oregon, USA, June 20–24, 2007. pp. 1087–1093 (2007)
16. Zhang, Z., Dai, G., Xu, C., Jordan, M.I.: Regularized discriminant analysis, ridge regression and beyond. *Journal of Machine Learning Research* 11, 2199–2228 (2010)