Noname manuscript No.
(will be inserted by the editor)

Ultra fast classification and regression of high-dimensional problems projected on 2D

Heba Alateyat · Manuel Fernández-Delgado · Eva Cernadas · Senén Barro

Received: date / Accepted: date

Abstract We propose the two-dimensional visual map classifier and regressor, which project the high-dimensional patterns on a 2D map, for human visualization and understanding of the data, and afterwards define a classification or regression map that predicts, for each 2D pattern, the class label (in classification) or the output value (in regression). The 2D projection is performed using the linear discriminant analysis, due to its high performance, speed and ability to project unseen (out-of-sample) patterns. The map is defined in an efficient way by assigning the proper output value to each square (or pixel) in the 2D map. The experiments show that the maps defined by both methods: 1) allow to understand visually the data distribution of a classification or regression problem; 2) their performances are very near to the state-of-the-art support vector classification and regression, including wrappers; and 3) they are very fast, between 1 and 5 orders of magnitude faster than the other approaches, spending less than 1 minute to classify datasets with 5 million patterns. Matlab code is available.

Keywords Classification · regression · mapping · data visualization · dimensionality reduction · linear discriminant analysis

1 Introduction

Classification and regression are important fields in machine learning, while data visualization is another interesting part of statistics that is tightly related to dimensionality reduction (DR). Many DR techniques are unsupervised methods oriented to discover and visualize low-dimensional structures

M. Fernández-Delgado
Centro Singular de inverstigación en Tecnoloxías Intelixentes da USC (CiTIUS), University of Santiago de Compostela, 15782, Spain
Tel.: +34-8818-16458
Fax: +34-8818-16405
E-mail: manuel.fernandez.delgado@usc.es

in high-dimensional unlabeled data [16]. They include principal component analysis (PCA), Sammon mapping, kernel PCA, multi-dimensional data scaling (MDS), stochastic neighbor embedding (SNE), locally linear embedding (LLE) and Isomap, among others. There are also supervised DR methods which have been applied to classification, such as the classical linear discriminant analysis (LDA) and neighborhood component analysis (NCA) proposed by [18]. [20] formulated the large margin nearest neighbor (LMNN) and [33] proposed a supervised version of LLE. The supervised Isomap was lauched by [10], and [32] published the dubbed multi-manifold discriminant Isomap (MMD-Isomap). Recent approaches include manifold learning [26], canonical kernel DR [25], feature unionization [14], joint dictionary learning [30] and probabilistic structure learning DR [28].

Another application of DR is data visualization in 2D, often oriented to explain the classification problems in terms of class distributions, borders and overlap. When DR is combined to classification, the classical "filter" approach consists of training a classifier using the low-dimensional patterns. The alternative "wrapper" approach designs the DR method to optimize in the low-dimension space the performance of some classifier, such as support vector machines [29], neural networks [23], genetic algorithms [22] or nearest neighbor classifiers [27], among others. Visualization of classification problems using grayscale images has been performed for malware detection [19], although [31] performs visualization using entropy graphs and deep convolutional neural networks. Some visualization methods have also been applied to regression. , such as supervised PCA [3] and LMNN [2].

In a previous study [1] we developed a comparison of 34 DR methods, including many of the previous approaches, in the 2D visualization on a collection of 71 classification datasets. The visualization is performed over two dimensions in order to display the classification problem for human inspection and understanding, which is more difficult in one or three, and not possible with more, dimensions. This study used 2D classification maps defined by a Gaussian kernel support vector classifier (GSVC), trained on the 2D mapped patterns using the filter approach, in order to visualize the classification problem. The agreement between the original high-dimensional and the 2D classification problems was measured by the performance of a GSVC on both problems. Although in this study all the methods achieve lower performance than the original high-dimensional data, the LDA-based methods achieved the highest performances and lowest times. As well, the study is limited to classification, so in the current paper we: 1) extend the LDA mapping to regression problems and use the 2D mapped patterns to explain and visualize the landscape of the function, discrete or continuous for classification or regression, respectively; 2) propose a method to define a 2D classification or regression map that, opposed to [1], does not train any other machine learning classifier or regressor on the 2D patterns; 3) use exclusively the 2D patterns mapped by LDA to create a "photography" or image of the dataset, allowing to visualize and understand the data properties; and 4) design the proposed methods in an efficient way in order to visualize and predict almost instantaneously any

large-scale classification or regression problem. The proposed methods, called **ultra-fast 2D classifier** (UF2DC) and **ultra-fast 2D regressor** (UF2DR), are described in section 2, being evaluated experimentally and discussed in section 3. The conclusions of this study are compiled in section 4.

2 Materials and methods

The proposed methodology first describes the LDA projection on 2D (subsection 2.1), introduces the 2D map (subsection 2.2) and describes how the map is defined in UF2DC and UF2DR for classification and regression (subsections 2.3 and 2.4), respectively.

2.1 Projection of the high-dimensional patterns in 2D

Let $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})$, with $i = 1, \ldots, N$, be the original high-dimensional training patterns (row vectors), p the number of inputs and N the number of training patterns. In classification problems, let C be the number of classes, $c_i \in \{1, \ldots, C\}$ the true class label of $\mathbf{x}_i$ and N_j, with $j = 1, \ldots, C$, the number of training patterns of class j. In regression problems, let o_i be the true output for $\mathbf{x}_i$. Following the LDA method, the total covariance matrix $\mathbf{T}$ (squared of size p) is defined as:

$$\mathbf{T} = \frac{(\mathbf{X} - \overline{\mathbf{X}})^T (\mathbf{X} - \overline{\mathbf{X}})}{N - 1} \tag{1}$$

where the matrices $\mathbf{X}$ and $\overline{\mathbf{X}}$, of size $N \times p$, contain, respectively, the training patterns $\{\mathbf{x}_i\}_{i=1}^N$ by rows, and the vector $\overline{\mathbf{x}} = (1/N) \sum_{i=1}^N \mathbf{x}_i$ (mean over all the patterns) repeated in each row. On the other hand, the matrix $\mathbf{W}$, also squared of size p, is calculated by averaging the within-class covariance matrices of the C classes:

$$\mathbf{W} = \frac{1}{C} \sum_{j=1}^{C} \frac{\left(\mathbf{X}_j - \overline{\mathbf{X}}_j\right)^T \left(\mathbf{X}_j - \overline{\mathbf{X}}_j\right)}{N_j - 1} \tag{2}$$

Both $\mathbf{X}_j$ and $\overline{\mathbf{X}}_j$ are matrices of size $N_j \times p$. The matrix $\mathbf{X}_j$ contains, by rows, the N_j training patterns $\mathbf{x}_i$ of class j (i.e., that verify $c_i = j$). Each row of matrix $\overline{\mathbf{X}}_j$ is equal to $\overline{\mathbf{x}}_j = (1/N_j) \sum_{c_i = j} \mathbf{x}_i$ (mean over patterns of class j). Additionally, the between-class covariance matrix $\mathbf{B}$, of size p, is calculated as $\mathbf{B} = \mathbf{T} - \mathbf{W}$. Once the matrices $\mathbf{W}$ and $\mathbf{B}$ are calculated, the matrix $\mathbf{A}$ of size $2 \times p$, which defines the mapping from $\mathbb{R}^p$ to $\mathbb{R}^2$, is composed by the two (row) eigenvectors associated to the largest eigenvalues (henceforth named "two main eigenvectors") of the matrix product $(\mathbf{W} + \alpha \mathbf{I}_p)^{-1} \mathbf{B}$, where $\mathbf{I}_p$ is the identity matrix of size p and $\alpha = 0.001$ regularizes the matrix $\mathbf{W}$, that is often

singular. Finally, the LDA method maps linearly a original test pattern $\mathbf{x} \in \mathbb{R}^p$ into a 2D pattern $\mathbf{y} = \mathbf{A}(\mathbf{x} - \overline{\mathbf{x}}) \in \mathbb{R}^2$. The literature includes many uses of LDA for classification and dimensionality reduction [17], as the incremental version proposed by [11], the robust LDA [15], with low sensitivity to noise and outliers, that achieves sparse discriminant directions, and the relevance weighting LDA [24], that weights the inter-class relations in order to estimate the within-class scatter matrix for multi-class problems.

In the literature, LDA is usually applied for classification by taking the $C-1$ main eigenvectors and mapping the high-dimensional patterns into the low-dimensional space generated by them. Using LDA for visualization, only the two main eigenvectors are used. In binary classification problems (with $C = 2$ classes), the usual approach in the literature is to use $C - 1 = 1$ eigenvectors and to project to 1D. Since we are interested in a two-dimensional representation of the data, in these cases we will also use the two main eigenvectors, although the first one will provide the largest power of discrimination between both classes. Obviously, when information is transformed from $p > 2$ dimensions into a 2D space some information might be lost, so the performance is expected to be lower on the 2D mapped dataset than on the original high-dimensional dataset. To evaluate the impact of this loss of information in the classification and regression performance is one of the objectives of the current work. Note that the proposed methods are designed for 2D data visualization, classification and regression, and they are not specifically oriented to dimensionality reduction.

In regression problems, where the continuous output o_i replaces the class label c_i for a training pattern $\mathbf{x}_i$, the previous methodology can also be used by replacing the classes by levels of output values, i.e., by quantifying the output values in a set of levels. The number $\mathcal{L}$ of output levels should not be very low, because a certain number of levels are required to quantify ouput values, nor very high, because many within-level covariance matrices (eq. 2) should be calculated. A reasonable value might be $\mathcal{L} = 10$, which provides a trade-off between ability to quantify the output values and excessive number of levels. Note that $\mathcal{L}$ is not the number of possible output values given by UF2DR (see subsection 2.4), but the number C of classes (or output levels) used to define the LDA mapping in regression problems. The number l of patterns per level is calculated as $l = \lfloor N/\mathcal{L} \rfloor$, where $\lfloor \cdot \rfloor$ is the integer `floor` function. Sorting the true output values $o_1, \ldots, o_N$ and denoting by σ_i the position of pattern $\mathbf{x}_i$ according to this sorting, a class label $c_i = 1 + \lfloor \sigma_i / l \rfloor \in \{1, \ldots, C\}$ is assigned to pattern $\mathbf{x}_i$, with $i = 1, \ldots, N$, and LDA can be applied analogously to the classification case. The use of LDA for regression problems through categorization of the value to predict may seem an artificial approach, given that a plethora of methods do exist specifically for regression. However, this categorization allows us to extend the LDA mapping, and the methods proposed to develop a 2D map, for regression problems in an efficient way. The experimental work (section 3.2) will develop an empirical evaluation of this efficiency and the performance loss compared to existing regression methods working on the original p-dimensional data.

2.2 Definition of the 2D map

Once the 2D training patterns are projected from $\mathbb{R}^p$ and standarized (i.e., scaled and translated to have zero mean and standard deviation one in each dimension), the proposed methods UF2DC or UF2DR create a map, defined by the square matrix $\mathbf{M}$, of the squared region $\mathcal{R}$ of $\mathbb{R}^2$ where the mapped patterns are located. Let us consider $\mathcal{R}$ divided in s^2 squares, so that s is the size of the map $\mathbf{M}$ of $\mathcal{R}$. The element M_{nm}, with $n, m = 1, \ldots, s$, is the output value associated to the n, m-th square in $\mathcal{R}$. Thus, a new p-dimensional test pattern $\mathbf{x}_t$ can be projected to a 2D pattern $\mathbf{y}_t = \mathbf{A}(\mathbf{x}_t - \overline{\mathbf{x}})$, the indices n, m of the square where $\mathbf{y}_t$ is located are calculated, and the output predicted by UF2DC or UF2DR for $\mathbf{y}_t$ will be $z_t = M_{nm}$.

In our previous work [1], the 2D patterns mapped by LDA are used to train a GSVC which afterwards classifies a 2D pattern for each square in $\mathcal{R}$. This "filter" approach is very intuitive and easy to understand, but it has several drawbacks. 1) The classification map is conditioned by the classifier, so different classifiers will define different maps, while a 2D map of a classification problem should be intrinsic to the dataset, i.e., unique. 2) Some classifiers might be limited defining complex 2D maps. An example is random forest: despite being a state-of-the-art classifier, it splits binarily each input dimension, so in 2D it only would learn class borders parallel to both axes. Other classifiers might exhibit other restrictions. 3) Since the classifier must be tested on all the squares of the map in order to achieve their output values, its training and testing may be slow with large, and even with small, datasets. Specifically, to train a GSVC in 2D may be slow with high class overlap or when the high-dimensional classification problem is already difficult. With large datasets, to train the classifier (or regressor) on the 2D mapped dataset might be slow despite the speed of the LDA mapping, so fast methods to define 2D maps using only the mapped training patterns without any training would be useful. Such methods are proposed in subsections 2.3 and 2.4 for classification and regression, respectively.

2.3 Definition of the 2D classification map in UF2DC

In order to create the classification map used by the ultra-fast 2D classifier (UF2DC), let $\mathbf{y}_i = (y_{i1}, y_{i2})$, with $i = 1, \ldots, N$, be the standarized 2D mapped training patterns, and let L_q, U_q be the lower and upper bounds in the dimension $q = 1, 2$, of region $\mathcal{R}$. Since the patterns are standarized, it is expectable that $-2 \leq y_{i1}, y_{i2} \leq 2$ for most patterns, so $L_q = -2$ and $U_q = 2$ for $q = 1, 2$, should be a good choice. In those datasets with many 2D patterns outside the square $[-2, 2] \times [-2, 2]$, the values L_q and U_q should be set adequately and they even might depend on q, so the squares might be replaced by rectangles. The 2D classification map will cover the square $\mathcal{R} = \{(y_1, y_2) : L_q \leq y_q \leq U_q, q = 1, 2\}$, which is divided in a grid of s^2 squares. The map size s is selected as:

$$s = \max(S_L, \min(\lfloor\sqrt{\eta N}\rfloor, S_U)) \tag{3}$$

which is increasing with the number N of training patterns and with the parameter η, a pre-specified ratio between the number of squares and training patterns, so that $\eta \approx s^2/N$. For classification, a value $\eta = 1$ is a good choice in order to have roughly so many squares as training patterns. In order to avoid excesively large or small s when N is too high or too low, a lower and an upper bound ($S_L = 10$ and $S_U = 300$, respectively) are set. The upper bound S_U limits the size of matrix $\mathbf{M}$, and the computational cost of the map definition, to $S_U^2 = 90,000$ squares in large classification datasets where $N > S_U^2/\eta = 90,000$ patterns.

The element M_{nm} of $\mathbf{M}$ is the class prediction for the test patterns that are mapped by LDA in the square with indices n and m, with $n, m = 1, \ldots, s$, square that henceforth will be denoted as $\langle n, m\rangle$. This prediction M_{nm} is selected as the class with the highest relative population in $\langle n, m\rangle$, i.e., the class with the highest proportion of training patterns located in that square. In order to create $\mathbf{M}$, a matrix $\mathbf{P}^j$, named "population matrix", of size s is defined for each class $j = 1, \ldots, C$, in such a way that P^j_{nm}, with $n, m = 1, \ldots, s$, is the relative population of class j in $\langle n, m\rangle$. Given $\mathbf{y}_i = (y_{i1}, y_{i2})$, in order to calculate P^j_{nm} we define:

$$u_{iq} = \max\left[1, \min\left(\frac{s(y_{iq} - L_q)}{U_q - L_q}, s\right)\right], \quad q = 1, 2 \tag{4}$$

The value u_{iq} is the index of the interval where y_{iq} is located considering a set of s intervals between L_q and U_q, so that $\mathbf{y}_i$ is located in $\langle u_{i1}, u_{i2}\rangle$ and P^j_{nm} is calculated as:

$$P^j_{nm} = \frac{1}{N_j}\sum_{i=1}^{N} \delta(n, u_{i1})\delta(m, u_{i2})\delta(j, c_i) \tag{5}$$

where $n, m = 1, \ldots, s$ and $j = 1, \ldots, C$, while $\delta(a, b) = 1$ when $a = b$ and zero otherwise. Since each element of the class population matrices $\mathbf{P}^j$ is calculated separately, they contain noise that can be reduced by applying a smoothing process. This smooting can be developed using a 2D median filter [12]. Specifically, we used the Matlab comand $\mathbf{P}^j$=`medfilt2`($\mathbf{P}^j$,[K,K]), which applies this filter with a square mask of size K, being K=10 a reasonable value, padding the matrix with zeros on the edges. When $\langle n, m\rangle$ is not empty (i.e., exists $j \in \{1, \ldots, C\}$ such that $P^j_{nm} > 0$), the element M_{nm} is:

$$M_{nm} = \arg\max_{j=1,\ldots,C}\{P^j_{nm}\} \text{ when } \exists j \in \{1, \ldots, C\} : P^j_{nm} > 0 \tag{6}$$

Otherwise, when $\langle n, m\rangle$ is empty (i.e., it has ho training patterns, so that $P^j_{nm} = 0$ for $j = 1, \ldots, C$), the calculation of M_{nm} must consider a set of

squares of squared shape (also denoted as a "mask" in the literature), centered in $\langle n, m \rangle$, of the smallest size required to be non-empty. Let $2k+1$ the size of this mask, so that k must be the least value in the set $\{1, \ldots, s\}$ such that the mask includes some square $\langle r, v \rangle$ with $P_{rv}^j > 0$ for some class $j \in \{1, \ldots, C\}$.

Since the mask is centered on $\langle n, m \rangle$ with size $2k+1$, in order to be inside the map (of size s) it must be $r \in \{r_{nk}, \ldots, R_{nk}\}$ and $v \in \{v_{mk}, \ldots, V_{mk}\}$, where the bounds for r are $r_{nk} = \phi(n,k), R_{nk} = \Phi(n,k)$, the bounds for v are $v_{mk} = \phi(m,k), V_{mk} = \Phi(m,k)$, while $\phi(a,b) = \max(1, a-b)$ and $\Phi(a,b) = \min(a+b, s)$. The value of k is defined as:

$$k = \underset{l=1,\ldots,s}{\arg\min} \Big\{ \exists j \in \{1, \ldots, C\}, r \in \{r_{nl}, \ldots, R_{nl}\}, \\ , v \in \{v_{ml}, \ldots, V_{ml}\} : P_{rv}^j > 0 \Big\} \tag{7}$$

Once k is known, the element M_{nm} is the class j with the largest sum of P_{rv}^j over the squares of the mask with size $2k+1$:

$$M_{nm} = \underset{j=1,\ldots,C}{\arg\max} \{\mathcal{P}_{nm}^j\}, \quad \mathcal{P}_{nm}^j = \sum_{r=r_{nk}}^{R_{nk}} \sum_{v=v_{mk}}^{V_{mk}} P_{rv}^j \\ \text{when } P_{nm}^j = 0, \quad \forall j \in \{1, \ldots, C\} \tag{8}$$

As an efficient procedure to calculate M_{nm} (algorithm 2 in Appendix A of the suplementary material), in the beginning of each row n of the map matrix $\mathbf{M}$ (when $m = 1$), the value of k is set to 1. This value is increased by 1 until $\mathcal{P}_{nm}^j > 0$ for some class j. Since the value of k selected for M_{nm} provides at least one $P_{rv}^j > 0$ (non-empty square) inside the mask, for $\langle n, m+1 \rangle$, i.e., for the square on the next column, the value $k-1$ also gives some $P_{rv}^j > 0$, so the starting value of k for $\langle n, m+1 \rangle$ is selected as $k = \max(1, k-1)$, where the max function avoids $k = 0$. Finally, when $P_{nm}^j > 0$ for some $j = 1, \ldots, C$ (non-empty square, eq. 6), the value of k is reset to 1, because $\langle n, m+1 \rangle$ will only need at most a mask of size 3 ($= 2k+1$) to include non-zero P_{rv}^j values.

Once the matrix $\mathbf{M}$ has been calculated, the classification of a new test pattern $\mathbf{x}_t \in \mathbb{R}^p$ requires: a) to project $\mathbf{x}_t$ to $\mathbf{y}_t = (y_{t1}, y_{t2}) = \mathbf{A}(\mathbf{x}_t - \overline{\mathbf{x}})$, with $\mathbf{A}$ defined in subsection 2.1; b) to calculate the indices $n = u_{t1}$ and $m = u_{t2}$ of the square where $\mathbf{y}_t$ is located using eq. 4; and c) to assign the test pattern $\mathbf{x}_t$ to the class $z_t = M_{nm}$. Note that u_{tq} in eq. 4, for $q = 1, 2$, is bounded between 1 and s when the mapped point $\mathbf{y}_t$ is outside $\mathcal{R}$, i.e., $y_{tq} < 1$ or $y_{tq} > s$.

The matrix $\mathbf{M}$ is a low-resolution map of size $s \leq S_U = 300$, which is enough to predict the class for a 2D pattern. However, an adequate data visualization requires a larger image matrix $\mathbf{I}$ of size $s' > s$. This matrix is defined in two steps. First, the resampled map matrix $\mathbf{M}'$ is defined by augmenting $\mathbf{M}$ to a size s' and processing it with a box-shaped kernel using e.g. with the Matlab command $\mathbf{M}'$=`imresize`($\mathbf{M}$,γ,`'box'`). The scale $\gamma > 1$ is defined as $\gamma = \min(\Gamma, S/s)$, where $\Gamma = 10$ is the maximum scale and

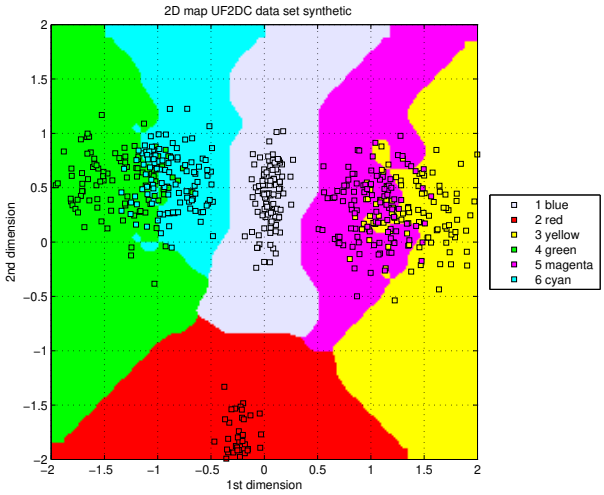

Fig. 1 Classification map defined by UF2DC and 2D training patterns for the UCI dataset `synthetic` (6 classes).

$S = 1,000$ is the maximum allowed size for $\mathbf{M}'$ in each dimension, so that $s' = \min(\Gamma s, S)$. Second, the image matrix $\mathbf{I}$ is a smoothed version of $\mathbf{M}'$ defined by applying the **mode** function over sliding squares (columnwise) of size $\xi = \lfloor 3\gamma/2 \rceil$ in each dimension ($\lfloor \cdot \rceil$ is the nearest integer function), using the Matlab command $\mathbf{I}$=`colfilt`($\mathbf{M}'$,$[\xi, \xi]$,`'sliding'`,`@mode`). Note that the calculation of the larger $\mathbf{M}'$ and $\mathbf{I}$ matrices (both of size s') is only necessary for visualization, but not for classification, that only requires the matrix $\mathbf{M}$, so the time required to the calculation of $\mathbf{M}'$ and $\mathbf{I}$ can be saved. Figure 1 shows the image (matrix $\mathbf{I}$) of the classification map defined by UF2DC and the 2D mapped training patterns (small squares) for the UCI classification dataset *Synthetic control chart time series* (abreviated as `synthetic`), with 6 classes (see subsection 3.1). The map provides a coherent 2D representation that shows which classes are adjacent or nearby. The color of the training patterns is given by their class labels, so the figure also shows their distribution over the regions occupied by each class, and the level of class overlap, given by the number of patterns of one class located in regions of other classes. In this case, most patterns are located in the region of their own class, so the the class overlap is low and UF2DC discriminates fairly well among classes.

The proposed method UF2DC is able to transform, using LDA, the original high-dimensional classification or regression dataset into a two-dimensional space that allows, in classification problems: 1) To see how the classes are located relatively and how are the borders among them. This is already interesting for binary classification, but even more for multi-class problems where the relative class positions and the borders among them may be complex. 2) To know what pairs of classes are located near, or share a common border, and what pairs of classes are far or non-adjacent, even whether some class is

inside, or completely surrounded by, other class. 3) To know what is the level of overlap among each pair of classes; 4) To see how far the training patterns, either of the same or different classes, are in this space. And to answer other questions that a data engineer may formulate about the data in classification problems. Additionally, UF2DC performs classification in this 2D space, because the class label is predicted for a new test pattern using exclusively its location on this map, reflecting the class prediction if the original high-dimensional classification problem were as it is seen in two-dimensions. Note also that LDA is used only to map the high-dimensional patterns on 2D, but the creation of the classifier from the 2D-mapped patterns does not depend on LDA at all and is very fast even for large-sized datasets. In fact, the few approaches in the literature that perform classification on 2D do it by train a standard classifier (e.g., a SVM) using the 2D patterns, while UF2DC designs the classifier as an image created directly from the mapped patterns, thus being coherent with the 2D map created by the LDA projection.

2.4 Definition of the 2D regression map in UF2DR

In regresion problems the output is continuous, so the ultra-fast 2D regressor (UF2DR) uses a larger map matrix $\mathbf{M}$ compared to UF2DC in order to codify a high number of output values. Therefore, a value η higher than in classification is recommendable to achieve a higher resolution map. We used $\eta = 5$ so the number s^2 of squares in $\mathcal{R}$ is five times the dataset population N. The map size s is calculated as a function of η and N as in UF2DC (eq. 3). To calculate the elements M_{nm}, with $n, m = 1, \ldots, s$, of the map matrix $\mathbf{M}$, the selection of the most populated class in classification is replaced by an output averaging in regression, similarly to many other machine learning approaches. Two matrices, $\mathbf{P}$ (population) and $\mathbf{O}$ (output), both of size s, are defined. The element P_{nm} of matrix $\mathbf{P}$ is the number of training patterns inside the square $\langle n, m \rangle$:

$$P_{nm} = \sum_{i=1}^{N} \delta(n, u_{i1}) \delta(m, u_{i2}), \quad n, m = 1, \ldots, s \tag{9}$$

where u_{i1}, u_{i2} and $\delta(a, b)$ are defined in subsection 2.3. The element O_{nm} of matrix $\mathbf{O}$ is the sum of the true outputs o_i over the training patterns $\mathbf{y}_i$ located inside $\langle n, m \rangle$:

$$O_{nm} = \sum_{i=1}^{N} \delta(n, u_{i1}) \delta(m, u_{i2}) o_i \tag{10}$$

When $P_{nm} > 0$, that happens when $\langle n, m \rangle$ is non-empty, the predicted output M_{nm} for $\langle n, m \rangle$ is the average of the true output o_i over the training patterns located inside this square:

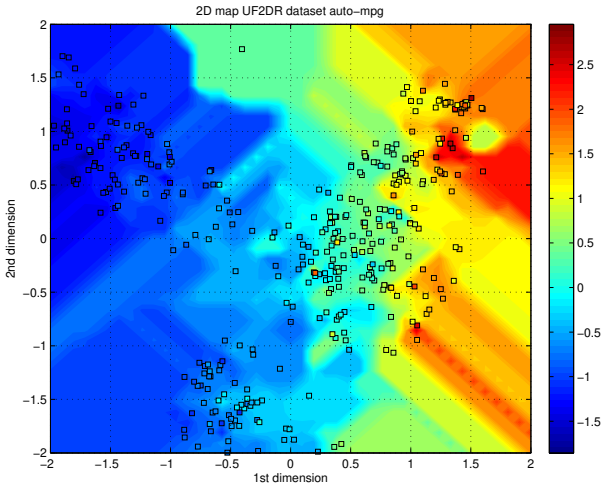

Fig. 2 Regression map defined by UF2DR and 2D training patterns for dataset `auto-mpg`.

$$M_{nm} = \frac{O_{nm}}{P_{nm}} \tag{11}$$

With empty squares ($P_{nm} = O_{nm} = 0$), the output M_{nm} is the average of O_{rv} over the values of r, v inside the mask of size $2k+1$ centered in $\langle n, m \rangle$. The limits r_{nk}, R_{nk}, v_{mk} and V_{mk}, for the square indices inside the mask are defined as in subsection 2.3. The mask semi-size k is the smallest value in $\{1, \ldots, s\}$ for which the mask has non-empty squares:

$$k = \arg\min_{l=1,\ldots,s} \left\{ \exists r_{nl} \leq r \leq R_{nl}, v_{ml} \leq v \leq V_{ml} : P_{rv} > 0 \right\} \tag{12}$$

and M_{nm} is given by:

$$M_{nm} = \frac{\sum_{r=r_{nk}}^{R_{nk}} \sum_{v=v_{mk}}^{V_{mk}} O_{rv}}{\sum_{r=r_{nk}}^{R_{nk}} \sum_{v=v_{mk}}^{V_{mk}} P_{rv}} \tag{13}$$

The output predicted by UF2DR for a test pattern $\mathbf{y}_t = (y_{t1}, y_{t2})$ is $z_t = M_{nm}$, where $n = u_{t1}$ and $m = u_{t2}$ and u_{tq} are calculated as function of y_{tq} for $q = 1, 2$, similarly to UF2DC (see the definition of u_{iq} in eq. 4). The median filtering of the $\mathbf{P}$ matrix is no longer required in UF2DR because the averaging already filters the output values in the $\mathbf{O}$ matrix. Similarly, the resampled map $\mathbf{M}'$ and image $\mathbf{I}$ matrices are no longer necessary because the

higher η value leads to a matrix $\mathbf{M}$ of size larger than in the classification case, that can be used for visualization. Appendix A in the suplementary material lists the whole algorithm of UF2DC and UF2DR. Figure 2 shows an example of regression map defined by UF2DR for the UCI dataset *Auto MPG* (see subsection 3.2), alongside with the training patterns (small squares), whose color codifies its true output value according to the color bar on the right. Note that the majority of the squares (2D mapped training patterns) have the same color as the background below them, which means a good agreement between the predicted and true output. Thus, the 2D regression map created by UF2DR is a meaningful and understandable 2D representation of the function to be predicted and of the training pattern distribution. This representation may help to understand: 1) what regions on this map correspond to low, medium and high values of this function; 2) in what directions the function raises or decreases more fastly; 3) in what directions it remains relatively constant; 4) the relations among the spatial locations of the patterns with values of the function similar or in a given range. The prediction is performed based on, and being coherent with, the 2D spatial location of the mapped training patterns, saving the time required to train any machine learning algorithm on the 2D patterns, a method that is very different to the existing approaches in the literature.

3 Results and discussion

The proposed methods, UF2DC and UF2DR, are evaluated on a collection of classification and regression datasets, described in the following subsections. We used 4-fold cross-validation in all the experiments, with 2, 1 and 1 folds in each trial for training, validation and test, respectively, in the hyper-parameter tuning. All the programs are executed in Matlab R2012a under a Linux Kubuntu 18.04 (64 bit) computer with Intel®Core™ i7-4790K CPU at 4GHz, equipped with 32GB of RAM. The Matlab code is available online[1]. The classifiers and regressors are allowed a maximum run time of 192 hours (8 days).

3.1 Classification

Figure 3 (left panel) plots the simple 2D artificial dataset multiple-spirals-appart, generated by us with $N = 1,206$ patterns in $C = 6$ classes (spirals) with $N_j = 200$ patterns/class for $j = 1, \ldots, C$. The i-th point in the j-th spiral (x_{ji}, y_{ji}), with $j = 1, \ldots, C$ and $i = 0, \ldots, N_j$, is given by $x_{ji} = \rho_{ji} \cos \theta_{ji}$ and $y_{ji} = \rho_{ji} \sin \theta_{ji}$, with $\rho_{ji} = 2\pi i / N_j$ and $\theta_{ji} = \rho_{ji} + j$. The right panel shows the classification map defined by UF2DC, where the 6 classes are clearly separated and each 2D mapped pattern (small square) is placed in the region of the right class.

[1] `persoal.citius.usc.es/manuel.fernandez.delgado/papers/uf2dcr`

Table 1 List of the 22 UCI classification datasets.

Dataset	Name	#patterns	p	C
Susy	susy	5,000,000	18	2
MiniBooNE particle identification	miniboone	130,064	50	2
Connect-4	connect-4	67,557	42	3
Adult	adult	48,842	14	2
MAGIC Gamma Telescope	magic	19,020	10	2
Nursery	nursery	12,958	8	4
Abalone	abalone	4,177	9	3
Wireless indoor localization	wifi	2,000	7	4
Banknote authentication	banknote	1,372	4	2
South German credit	south	1,000	20	2
Tic-tac-toe endgame	tictac	958	18	2
Annealing	annealing	886	52	5
Energy efficiency	energy-heat	768	7	3
Pima indians diabetes	pima	768	8	2
Australian credit approval	australian	690	35	2
Mammographic mass	mammograph	642	15	2
German credit	german	366	44	2
Ionosphere	ionosphere	350	33	2
US congressional voting records	voting	342	32	2
Statlog heart	heart	270	20	2
Seeds	seeds	210	7	3
Wine recognition	wine	178	13	3
Hepatitis domain	hepatitis	155	24	2

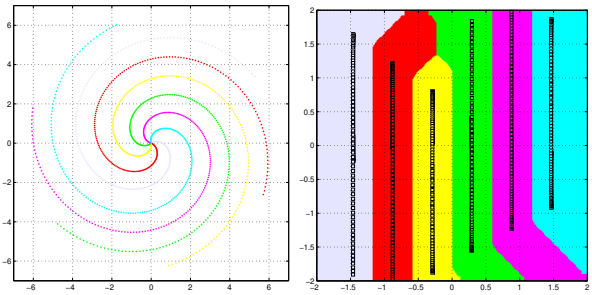

Fig. 3 Data set multiple spirals appart (left) and classification map defined by UF2DC (right).

Table 1 compiles the 22 real classification datasets from the UCI machine learning repository [6], sorted by decreasing populations, that we used to evaluate the proposed approach for classification problems. Figure 4 shows an example of classification map defined by UF2DC in the binary dataset `tic-tac`. The 2D mapped training patterns of both classes are very separated and they are located inside the region of the classification map associated to its right class, so that UF2DC discriminates perfectly between both classes. In binary classification problems such as this, the LDA normally projects to $C-1=1$ dimension by selecting the main eigenvector of matrix $(\mathbf{W}+\alpha\mathbf{I})^{-1}\mathbf{B}$. Since in our case we are interested in a 2D map of the classification problem, we use the two dimensions defined by the two main eigenvectors, and both classes are well separated, as in Figure 4. The first (horizontal) dimension, corresponding to the main eigenvector, discriminates well between classes. Although the sec-

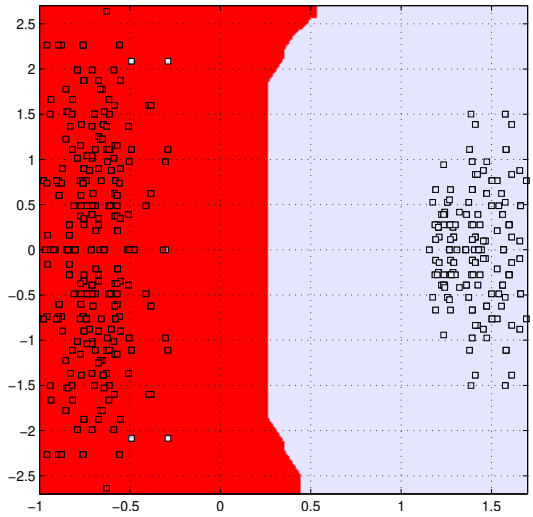

Fig. 4 Classification map defined by UF2DC and 2D training patterns for dataset `tic-tac` (2 classes).

ond (vertical) dimension does not discriminate both classes, we can use both projections to create a self-explaining 2D map of the classification problem.

The performance of UF2DC is compared with other four well-known classifiers operating on the original p-dimensional patterns.

1. **GSVC**: Gaussian kernel support vector machine classifier (LibSVM [5] library v. 3.22), a state-of-the-art classifier [8], although its training speed, tunable hyper-parameters (regularization parameter, with 11 values in the set $\{2^i\}_{-5}^{15}$, and inverse of the Gaussian kernel spread, with 10 values in the set $\{2^i\}_{-15}^{3}$) and memory requirements may hinder its use with large datasets.
2. **LSVC**: large-scale linear kernel support vector machine (LibLinear [7] library v. 2.3.20). It is selected due to its speed and appliability to large datasets, although its performance is expected to be suboptimal compared to GSVC. The regularization parameter is tuned as GSVC.
3. **WLSVC**: a linear SVC combined with DR in a wrapper approach [29], performed by a radial basis function (RBF) neural network trained to optimize the performance of the linear SVC (LibSVM v. 3.18) applied on the 2D patterns[2]. The spread of the RBF network is tuned with values $\{2^{2i-7}\}_{i=0}^{7}$ and the regularization parameter of LSVC is tuned as GSVC.
4. **PNN**: probabilistic neural network [13], implemented by the `newpnn` function in the Matlab Neural Network Toolbox. The PNN was selected due to its speed, low memory requirements and simple tuning (only the spread of the Gaussian functions must be tuned with 19 values between 0.001 and 10), which allows to classify large datasets.

[2] faculty.ucmerced.edu/mcarreira-perpinan/research/software/ ldsvm.tar.gz

Table 2 reports the Cohen kappa statistic [4], averaged over the 4 folds (standard deviations are not reported to save space and increase readability), achieved by the previous classifiers in each dataset. The last line reports the average kappa values over the datasets where no clasifier fails. Kappa is used instead of accuracy because it discards the agreement by change between predicted and true class when classes are unbalanced. Dashed (—) cells correspond to datasets where the corresponding classifier did not finish within 8 days (which will be referred as time errors) or crashed due to lack of memory (referred as memory errors). This happens in the 3 largest datasets (`susy`, `miniboone` and `connect-4`) for GSVC (time errors); in the same datasets plus `adult` for WLSVC (time errors); in dataset `susy` for PNN (memory errors).

Table 2 Kappa achieved by UF2DC, GSVC, LSVC, WLSVC and PNN, average values and p-value of a Wilcoxon ranksum test over datasets where no classifier fails.

Dataset	UF2DC	GSVC	LSVC	WLSVC	PNN
wine	0.983	0.958	0.983	0.974	0.932
tictac	0.960	0.977	0.960	0.975	0.970
annealing	0.866	0.970	0.956	0.764	0.906
seeds	0.951	0.907	0.943	0.865	0.894
banknote	0.954	1.000	0.941	1.000	0.997
nursery	0.836	1.000	0.866	0.732	0.917
voting	0.839	0.885	0.862	0.876	0.786
wifi	0.961	0.976	0.838	0.914	0.974
energy-heat	0.856	0.918	0.787	0.815	0.862
miniboone	0.761	—	0.724	—	0.818
heart	0.617	0.686	0.692	0.667	0.713
australian	0.690	0.719	0.675	0.685	0.555
mammograph	0.510	0.579	0.568	0.613	0.551
adult	0.503	0.556	0.556	—	0.488
susy	0.517	—	0.544	—	—
magic	0.592	0.708	0.524	0.710	0.621
pima	0.438	0.476	0.469	0.460	0.394
abalone	0.442	0.454	0.454	0.487	0.442
connect-4	0.408	—	0.422	—	0.428
hepatitis	0.373	0.489	0.348	0.500	0.454
south	0.327	0.360	0.327	0.362	0.225
german	0.286	0.281	0.313	0.179	0.143
Average	0.693	0.741	0.695	0.699	0.685
p-value	—	0.384	0.975	0.912	0.912

The UF2DC achieves average kappa (0.693) very near to GSVC (0.741), although slighly lower. This is expectable because GSVC is a state-of-the-art classifier running on the original high-dimensional data, while UF2DC operates on the 2D mapped data in order to achieve an explainable classification map. Thus, some loss of information may always happen in the projection, that may reduce the performance. However, the difference between UF2DC and GSVC is not very high: 0.048 in average, and below 0.16 in all the datasets. The p-value of a two-sided Wilcoxon ranksum test comparing UF2DC and GSVC over the 18 datasets without fails is 0.38 (see last line of Table 2), so the difference between both is very far from statistical significance. Thus, the use of two eigenvalues for 2D visualization in UF2DC causes a loss in performance (in kappa) with respect to the original high-dimensional data

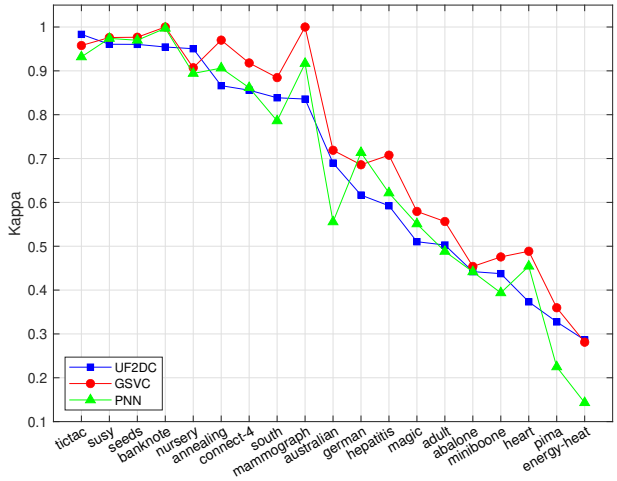

Fig. 5 Values of kappa achieved using UF2DC, GSVC and PNN on datasets without fails.

(GSVC) that is not very significant for the majority of the datasets. Comparing UF2DC with LSVC, the average performance of the former (0.693) is only slightly below the latter (0.695), despite that it runs on the original data, with a high p-value (0.975). Considering individual datasets, the maximum difference between both is 0.089 in dataset `annealing`, and is below 0.05 in 18 of 22 datasets. Comparing UF2DC and WLSVC, the average difference is also low (0.006) with high p-value (0.912), and the difference only overcomes 0.1 in 3 of 18 datasets where WLSVC does not fail, while UF2DC also overcomes WLSVC in other 3 datasets by more than 0.1. The UF2DC outperforms PNN (0.693 vs. 0.685), despite the former uses the 2D mapped patterns while the latter runs on the high-dimensional space. Figure 5 shows that UF2DC achieves kappa values near to GSVC, excepting dataset `mammograph`, and slightly better than PNN, with small differences excepting datasets `australian`, where UF2DC outperforms PNN, and `heart`, where PNN outperforms UF2DC. It is remarkable that UF2DC, operating on the 2D space, is able to follow quite closely the results of both classifers, that work on the original multi-dimensional space.

Table 3 reports the times spent by UF2DC, GSVC, LSVC, WLSVC and PNN in each classification dataset, sorted by decreasing times of UF2DC. Values with asterisks are times estimated using the number of tuning trials finished by GSVC within 8 days, because it had time errors as shown in Table 2. Dashed cells mean that no tuning trial was finished due to time or memory errors, so the time could not be estimated. The times of UF2DC are really tiny compared to the remaining classifiers, whose values raise very fast with the dataset population, specially WLSVC, which is the slowest, and GSVC. The PNN is also much slower than UF2DC. Despite that LSVC is known to be very fast, it is 10-1000 times slower than UF2DC depending on the dataset size. Note that UF2DC only spends 1 and 3 minutes to classify large datasets as `miniboone` (130,000 patterns, 50 inputs) and `susy` (5 million patterns), where

pLSVC spends 6 and 32 hours, respectively. Remember that UF2DC requires to calculate the eigenvectors of the p-order square matrix $(\mathbf{W} + \alpha\mathbf{I}_p)^{-1}\mathbf{B}$, which may be time-consuming for high p. However, the UF2DC classifier is fairly fast (70 s.) on the dataset with the highest dimensionality (`miniboone`, p=50). Therefore it scales pretty well with the number of inputs compared to other classifiers, although it is oriented to data visualization and classification of general, and not specifically high-dimensional, datasets. The last row reports the average ratio of the times spent by GSVC, LSVC, WLSVC and PNN to the times spent by UF2DC, over the datasets where the first classifier does not fail. In average, GSVC is about 15,000 times slower than UF2DC, and WLSVC is about 392,000 times slower, while LSVC and PNN are 48 and 143 times slower. The second column (labeled Map) of Table 3 reports the times spent by UF2DC to define the map of the whole dataset including the resampling time (see subsection 2.3). These times only overcome 10 s. in the 4 datasets with more than 50,000 patterns, and only slightly overcome one minute in the largest dataset `susy`.

Table 3 Elapsed time spent by UF2DC and by its map definition, GSVC, LSVC, WLSVC and PNN.

Dataset	UF2DC	Map	GSVC	LSVC	WLSVC	PNN
susy	175.5	63.0	—	1.1E+05	—	—
miniboone	92.2	14.4	3.8E+06*	849.7	—	5.9E+04
connect-4	14.0	11.1	3.1E+06*	1698.2	3.8E+07*	1.7E+04
adult	15.8	8.2	5.2E+05	476.9	9.6E+06*	7073.8
magic	1.6	7.3	1.6E+04	22.9	7.7E+05*	127.5
nursery	1.6	10.0	3908.5	45.3	2.7E+05	89.6
abalone	0.5	4.3	10.6	10.7	1.6E+05	14.9
annealing	0.1	0.8	21.7	4.9	1.3E+05	5.4
south	0.2	1.9	14.4	1.4	4490.7	4.5
banknote	0.2	3.0	6.4	0.9	5899.2	4.2
wifi	0.2	3.6	17.1	3.5	2.0E+04	4.0
tictac	0.1	1.7	14.1	0.8	4.2E+04	2.5
australian	0.1	0.7	10.1	1.1	3.8E+04	2.5
energy-heat	0.1	0.8	6.3	1.1	2.7E+04	2.4
mammograph	0.1	0.7	11.0	0.8	1.6E+04	2.0
german	0.1	0.4	4.7	0.8	1.3E+04	2.0
pima	0.1	0.8	10.3	0.7	1.1E+04	2.0
voting	0.1	0.3	3.5	0.3	6781.9	1.9
heart	0.0	0.3	2.3	0.3	4506.2	1.7
hepatitis	0.0	0.2	1.3	0.3	3381.1	1.6
seeds	0.0	0.3	1.1	0.2	2025.2	1.5
wine	0.0	0.2	1.2	0.1	953.1	1.5
Avg. ratio	—	—	14638.9	47.8	391818.9	143.0

Figure 6 plots the times spent by UF2DC, GSVC, LSVC, WLSVC and PNN. In the smaller datasets (right end of the plot), UF2DC is in general two orders of magnitude faster than GSVC and PNN, 6 orders faster then WLSVC and one order faster than LSVC. The difference raises with the dataset size (left end of the plot) to five orders for GSVC (WLSVC has time errors) and three orders for LSVC, because UF2DC does not increase the size s of matrices $\mathbf{P}^j$ and $\mathbf{M}$, while the other classifiers strongly slow down with N and p.

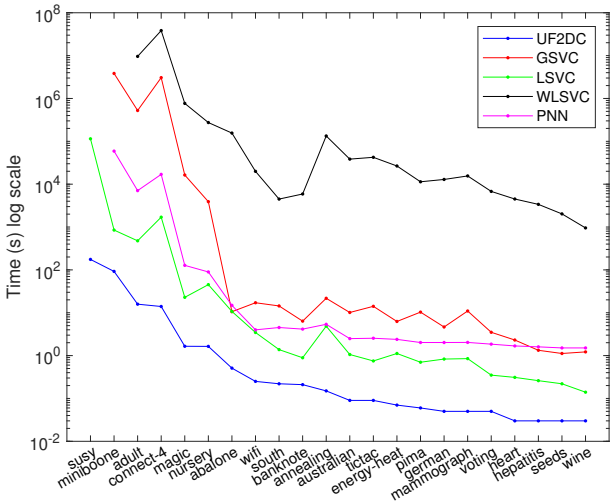

Fig. 6 Time (in a logarithmic scale) spent by UF2DC, GSVC, LSVC, WLSVC and PNN.

Considering the times spent by UF2DC in each stage of the map definition (column Map in Table 3), the calculation of the $\mathbf{P}^j$ matrices (subsection 2.3) spends a maximum time of 0.3 s. in the largest dataset `susy` (5 million patterns). The median filtering is also very fast, with times below 0.44 s. in all the datasets. The calculation of the $\mathbf{M}$ matrix also spends times below 1 s. except in the largest datasets `susy` (1.1 s.), `adult` (2.2 s.), `connect-4` (3.1 s.), `miniboone` (6.4 s.) and `mnist` (9.2 s). Remarkably, this time: 1) does not depend so much on N because the size of matrices $\mathbf{P}^j$ and $\mathbf{M}$ is upper bounded by S_U when $N > S_U^2/\eta$; and 2) it depends on the percentage of empty squares, i.e., on the pattern distribution. This explains that the calculation of the $\mathbf{M}$ matrix in UF2DC is faster in dataset `susy` than in the other four datasets, where N <130,000 patterns. Therefore, UF2DC scales well with N, being suitable for large-scale classification problems. It also scales well with p, which does not influence the elapsed time after the 2D projection, although the calculation of the covariance matrices $\mathbf{W}$ and $\mathbf{T}$, both of size p, might be slow for very high p.

3.2 Regression

Table 4 compiles the 16 UCI regression datasets used to evaluate UF2DR, with N and p, which are high for some datasets (up to four million patterns and almost 500 inputs). In all the experiments, UF2DR uses $\eta = 5$ (five times more squares in the map than patterns) and $\mathcal{L} = 10$ (number of output levels in the LDA mapping, see subsection 2.1). The UF2DR was compared to the following three regressors:

Table 4 List of the 16 UCI regression datasets.

Dataset	Name	N	p
Gas sensor array under dynamic gas	gas-co	4,178,504	17
Indiv. household electric power consum.	household	2,049,280	6
Dynamic features of virusshare execs.	dynamic	107,856	482
Relative location of CT slices	CT-slices	53,500	374
Combined cycle power plant	combined-cycle	9,527	4
Wine quality red	wine-quality	1,599	11
Air quality	air-quality-NOx	1,230	8
Geographical origin of music	geo-long	1,059	72
Energy efficiency	energy-cool	768	7
Bike sharing	bike-day	731	31
Student performance	student-por	649	30
Facebook performance metrics	fb-metrics	500	20
Auto MPG	auto-mpg	398	23
Yacht hydrodynamics	yacht-hydro	308	6
Servo	servo	167	15
Concrete slump test	slump	103	7

1. **GSVR**: Gaussian kernel ε-support vector regression (LibSVM library), selected by its good performance in the literature [9]. The loss function uses ε=0.1 and the regularization parameter and inverse of the Gaussian kernel spread are tuned as GSVC (see subsection 3.1).
2. **LSVR**: linear support vector regression (LibLinear library) with L2- regularized L2-loss SVR (primal) solver and $\varepsilon = 0.1$. The regularization parameter is tuned as pGSVR. This regressor is included due to its speed, which makes it suited for large datasets.
3. **GRNN**: generalized regression neural network (`newgrnn` function in the Matlab Neural Network Toolbox), selected by its speed [21] and by its easy hyperparameter tuning (only the Gaussian spread, with 25 values between 0.001 to 50).

Table 5 Squared correlation R^2 achieved by UF2DR, GSVR, LSVR and GRNN.

Dataset	UF2DR	GSVR	LSVR	GRNN
bike-day	0.989	0.998	0.997	0.745
fb-metrics	0.974	0.940	0.871	0.718
yacht-hydro	0.934	0.989	0.650	0.539
combined-cycle	0.911	0.947	0.928	0.943
CT-slices	0.908	—	0.861	1.000
energy-cool	0.901	0.965	0.883	0.903
auto-mpg	0.802	0.863	0.847	0.792
air-quality-NOx	0.796	0.872	0.800	0.856
gas-co	0.737	—	0.687	—
servo	0.692	0.776	0.706	0.334
slump	0.367	0.345	0.266	0.266
dynamic	0.353	—	0.014	0.486
household	0.243	—	0.248	—
wine-quality	0.234	0.384	0.356	0.350
student-por	0.144	0.298	0.270	0.258
geo-long	0.033	0.272	0.160	0.231
Average	0.648	0.721	0.644	0.578
p-value	—	0.436	0.840	0.436

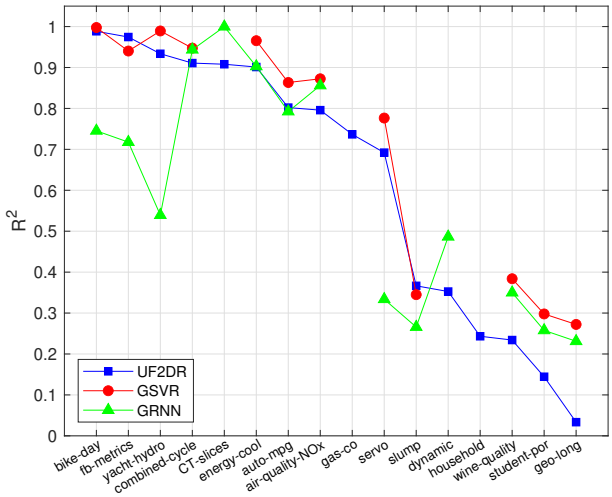

Fig. 7 Values of R^2 achieved by UF2DR, GSVR and GRNN.

Table 5 reports the squared Pearson correlation (R^2) between the true and predicted outputs achieved by the previous regressors and by UF2DR in each dataset, the average R^2 over all the datasets excepting errors, and the p-value of a Wilcoxon ranksum test comparing UF2DR and the other methods. We used the R^2 instead of the most frequent root mean squared error ($RMSE$) because R^2 is a performance (not error) measurement, and because $0 \leq R^2 \leq 1$ while $0 \leq RMSE < \infty$, so the former measures better how near the regressor is from the perfect prediction ($R^2 = 1$) in absolute terms. The dashed cells correspond to regressors and datasets which achieved memory (GRNN in `gas-co` and `household`) or time (GSVR in `CT-slices`, `gas-co`, `household` and `dynamic`) errors. Only UF2DR and LSVR can be executed over all the datasets. The average R^2 of UF2DR is near to GSVR (0.648 against 0.721, respectively), outperforming LSVR (0.644) and GRNN (0.578), so the 2D mapping in UF2DR does not reduce very much the performance. Comparing by datasets, the difference between UF2DR and GSVR is below 0.1 in 9 of 12 datasets where GSVR does not fail. The Wilcoxon comparing UF2DR and GSVR gives a high p-value (0.436), so their difference is far from being statistical significant.

Figure 7 compares R^2 of UF2DR, GSVR and GRNN, with missing symbols for GSVR in datasets `CT-slices`, `gas-co`, `dynamic` and `house- hold`, and for GRNN in datasets `gas-co` and `household`. The UF2DR follows GSVR in all the datasets, and the only dataset with high difference (0.23) is `geo-long`, where the R^2 of GSVR is already very low. The UF2DR outperforms GRNN by far in 4 datasets (`bike-day`, `fb-metrics`, `yatch-hydro` and `servo`), following the performance of GRNN in datasets with low R^2.

Figure 8 reports the average R^2 over all the datasets achieved by UF2DR using different numbers $\mathcal{L}$ of output levels (see subsection 2.1). Note that the

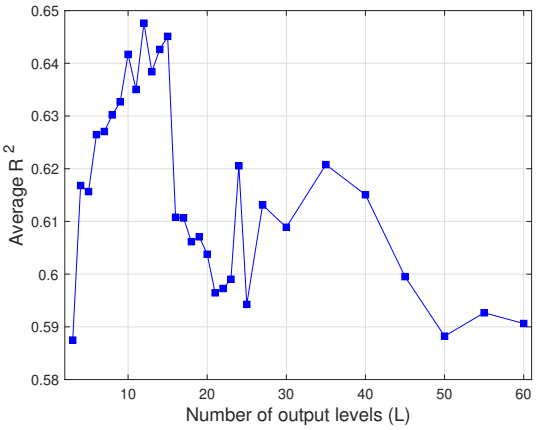

Fig. 8 Average R^2 achieved by UF2DR using different numbers $\mathcal{L}$ of levels in the LDA mapping.

vertical scale is very narrow (0.58–0.65), so the influence of $\mathcal{L}$ in the performance of UF2DR is low. Therefore, a default value can be used for $\mathcal{L}$ and a specific tuning is not required. With low $\mathcal{L}$ (about 3–10, left end of the plot), the average R^2 raises achieving the highest values in a narrow interval about $\mathcal{L}$=10-15 levels, so the selected default value $\mathcal{L} = 10$ is a good choice, whose performance is very near to the best one. The behavior fastly degrades for $\mathcal{L} > 16$ and never recovers, but the elapsed times increase because more covariance matrices must be calculated (eq. 2).

Table 6 reports the times spent by UF2DR, GSVR, LSVR and GRNN on the regression datasets. The times with asterisks correspond to datasets where GSVR did not finish within 8 days (time errors, see Table 5). These times are estimated considering the number of tuning trials executed by GSVR during 8 days. This estimation could not be performed in datasets `gas-co` and `household` (4 and 2 million patterns, respectively) because GSVR did not finish any tuning trial. The times of UF2DR are much lower than the other regressors, and the difference with respect to them raises fastly with N above dataset `combined-cycle` (9,527 patterns): 1-2 orders of magnitude with LSVR, and 2-5 orders of magnitude with the remaining regressors. Similarly to Table 3 in subsection 3, last row in Table 6 reports the average time ratios over the non-failing datasets, being GSVR about 36,000 times, GRNN 657 times and LSVR 17 times slower than UF2DR.

The second column (Map) in Table 6 reports the times spent by UF2DR to define the map of the whole dataset, with very low values which only overcome 10 s. in the two datasets with more than 2 million patterns (`gas-co` and `household`). These times are comparatively larger in some datasets (e.g. `auto-mpg`) with many empty squares. Compared to UF2DC, the map in UF2DR spends less time in the smallest datasets due to the lack of median filtering, map resampling and image smoothing. The two steps in the definition of the regression map are: 1) calculation of $\mathbf{P}$ and $\mathbf{O}$ matrices, which spend times

below 0.1 excepting `household` and `gas-co`, with the longest time, 0.381 s.; and 2) calculation of the **M** matrix, which is also fast with times below 1 s. excepting `household` and `CT-slices`, spending only 0.25 s. in the largest dataset `gas-co` despite its high N.

Table 6 Elapsed time spent by UF2DR, by the map definition in UF2DR, and by GSVR, LSVR and GRNN.

Dataset	UF2DR	Map	GSVR	LSVR	GRNN
gas-co	135.8	45.1	—	1910.1	—
household	51.5	16.9	—	635.5	—
dynamic	31.3	10.0	9.2E+06*	5545.8	2.1E+05
CT-slices	25.9	7.6	4.6E+06*	380.8	5.7E+04
combined-cycle	1.1	0.5	4.0E+04	2.9	37.1
geo-long	0.2	0.1	42.7	1.4	6.3
wine-quality	0.3	—	324.5	0.7	4.6
air-quality-NOx	0.2	0.1	279.5	0.6	3.5
bike-day	0.1	0.1	8.0	0.5	3.1
student-por	0.1	0.0	17.1	0.5	2.9
energy-cool	0.1	0.1	93.6	0.3	2.5
fb-metrics	0.1	0.0	10.3	0.3	2.4
auto-mpg	0.1	2.7	7.2	0.3	2.3
yacht-hydro	0.0	0.0	7.0	0.1	2.0
servo	0.0	0.0	2.3	0.1	1.9
slump	0.0	0.0	1.3	0.1	1.8
Avg. ratio	—	—	36469.2	16.7	657.4

Figure 9 plots the times spent by UF2DR, GSVR, LSVR and GRNN (note that each tick represents 2 orders of magnitude). In the smallest datasets (right part of the plot), the time of UF2DR is one order lower than LSVR and 2 orders lower than GSVR and GRNN. The difference raises in the largest datasets (left part of the plot) to 6 orders for GSVR, 4 orders for GRNN, and 1-2 orders for LSVR. In the largest datasets `gas-co` and `household`, LSVR is 10-15 times slower than UF2DR, which only spends 1-2 minutes. The time spent by UF2DR scales so well with $N > S_U^2/\eta$ because the map size s is upper bounded by $S_U = 300$, and because the time sensitivity with p is low, as discussed in subsection 3.1.

4 Conclusions

The proposed methods, named ultra-fast 2D classifier (UF2DC) and regressor (UF2DR), define a 2D map which visualizes the high-dimensional problem: 1) using linear discriminant analysis to project the data in 2D, and dividing in a grid of squares the region where the standarized 2D patterns are placed; and 2) defining the 2D map as a square matrix where each element is the predicted value for the mapped patterns which are located in the square associated to that element. In UF2DC, the predicted output for a square is the label of the class with the highest relative population of patterns located in that square, in order to manage possible class imbalances. In UF2DR, the predicted output for a square is the average output over the training patterns inside that

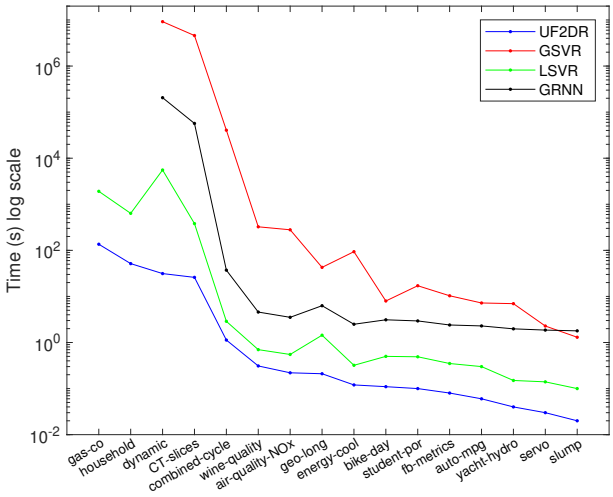

Fig. 9 Times (log. scale) spent by UF2DR, GSVR, LSVR and GRNN.

square. In empty squares, the voting and average are performed over the smallest non-empty squared region. The experimental work compares UF2DC and UF2DR with several well-known classifiers and regressors applied on the high-dimensional patterns with very large datasets (up to 5 million patterns and 791 inputs). The results show that: 1) The proposed methods perform an understandable 2D visualization of classification and regression datasets. 2) This visualization can be developed with a slight loss in performance. In classification, UF2DC outperforms PNN by 0.008 in terms of average kappa, and only losses 0.05, 0.002 and 0.003 compared to GSVC, LSVC and WLSVC, respectively. In regression, UF2DR outperforms LSVR and GRNN by 0.004 and 0.007, respectively, in terms of average R^2, and only loses 0.07 compared to GSVR. 3) The proposed methods are very fast, projecting datasets with 5 million patterns in just 1 minute when GSVC/GSVR, WLSVC and PNN/GRNN require huge amounts of memory or can not process large datasets in a reasonable time. In average, UF2DC/UF2DR are 5 orders of magnitude faster than GSVC/GSVR, 2 orders faster than PNN/GRNN, and 1 order faster than LSVC/LSVR.

5 Acknowledgments

This work has received financial support from the Consellería de Educación, Universidade e Formación Profesional, Xunta de Galicia (accreditation 2019-2022 ED431G-2019/04) and the European Regional Development Fund (ERDF), which acknowledges the CiTIUS - Centro Singular de Investigación en Tecnoloxías Intelixentes da Universidade de Santiago de Compostela as a Research Center of the Galician University System.

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Alawadi, S., Fernández-Delgado, M., Mera, D., Barro, S.: Polynomial kernel discriminant analysis for 2D visualization of classification problems. Neural Computing & Applications **8**, 1–17 (2017)
2. Assi, K.C., Labelle, H., Cheriet, F.: Modified large margin nearest neighbor metric learning for regression. IEEE Signal Processing Letters **21**(3), 292–296 (2014)
3. Barshan, E., Ghodsi, A., Azimifar, Z., Jahromi, M.: Supervised principal component analysis: visualization, classification and regression on subspaces and submanifolds. Pattern Recognition **44**(7), 1357–1371 (2011)
4. Carletta, J.: Assessing agreement on classification tasks: the kappa statistic. Computational Linguistics **22**(2), 249–254 (1996)
5. Chang, C., Lin, C.: LIBSVM: a library for support vector machines. ACM Transactions on Inteligent Systems and Technologies **2**, 27:1–27:27 (2011)
6. Dua, D., Graff, C.: UCI machine learning repository (2017). URL `http://archive.ics.uci.edu/ml`
7. Fan, R., Chang, K., Hsieh, C., Wang, X., Lin, C.: LIBLINEAR: A library for large linear classification. Journal of Machine Learning Research **9**, 1871–1874 (2008)
8. Fernández-Delgado, M., Cernadas, E., Amorim, D., Barro, S.: Do we need hundreds of classifiers to solve real world classification problems? Journal of Machine Learning Research **15**, 3133–3181 (2014)
9. Fernández-Delgado, M., Sirsat, M., Cernadas, E., Alawadi, S., Barro, S., Febrero-Bande, M.: An extensive experimental survey of regression methods. Neural Networks **55**, 215–230 (2016)
10. Geng, X., Xuan, D., Zhou, Z.: Supervised nonlinear dimensionality reduction for visualization and classification. IEEE Transactions on Systems, Man and Cybernetics, part B **35**(6), 1098–1107 (2005)
11. Ghassabeh, Y., Rudzicz, F., Moghaddam, H.: Fast incremental LDA feature extraction. Pattern Recognition **48**, 1999–2012 (2015)
12. González, R., Woods, R., Eddins, S.: Digital image processing using Matlab. Prentice-Hall (2004)
13. Guan, S., Fang, Q., Guan, T.: Application of a novel PNN evaluation algorithm to a greenhouse monitoring system. IEEE Transactions on Instrumentation and Measurement **70**, 1–12 (2021)
14. Jalilvand, A., Salim, N.: Feature unionization: a novel approach for dimension reduction. Applied Soft Computing **52**, 1253–1261 (2017)
15. Li, C., Shao, Y., Yin, W., Liu, M.: Robust and sparse linear discriminant analysis via and alternative direction method of multipliers. IEEE Transactions on Neural Networks and Learning Systems **31**(3), 915–926 (2020)
16. Maaten, L.: An introduction to dimensionality reduction using Matlab. Tech. Rep. 2579-2605, Universiteit Maastricht (2007). URL `http://lvdmaaten.github.io`
17. Mai, Q.: A review of discriminant analysis in high dimensions. WIREs Computational Statistics **5**(3), 190–197 (2013)
18. Mendels, O., Stern, H., Berman, S.: User identification for home entertainment based on free-air hand motion signatures. IEEE Transactions on Systems, Man and Cybernetics, part A **44**(11), 1461–1473 (2014)
19. Nataraj, L., Karthikeyan, S., Jacob, G., Manjunath, B.: Malware images: visualization and automatic classification. In: Proceedings of the International Symposium on Visualization for Cyber Security, pp. 4–10 (2011)
20. Pasolli, E., Yang, H.L., Crawford, M.M.: Active-metric learning for classification of remotely sensed hyperspectral images. IEEE Transactions on Geoscience and Remote Sensing **54**(4), 1925–1939 (2016)
21. Shui, P., Shi, X., , Li, C., Feng, T., Xia, X., Y. Han, Y.: GRNN-based predictors of UHF-band sea clutter reflectivity at low grazing angle. IEEE Geoscience Remote Sensing Letters **19**, 1–5 (2022)
22. Stathakis, D., Perakis, K.: Feature evolution for classification of remotely sensed data. IEEE Geoscience and Remote Sensing Letters **4**(3), 354–358 (2007)

23. Tallón-Ballesteros, A., Riquelme, J., Ruiz, R.: Semi-wrapper feature subset selector for feed-forward neural networks: applications to binary and multi-class classification problems. Neurocomputing **353**, 28–44 (2019)
24. Tang, E., Suganthan, P., Yao, X., Qin, A.: Linear dimensionality reduction using relevance weighted LDA. Pattern Recognition **38**, 485–493 (2005)
25. Tao, C., Feng, J.: Canonical kernel dimensionality reduction. Computational Statistics & Data Analysis **107**, 131–148 (2017)
26. Turchetti, C., Falaschetti, L.: A manifold learning approach to dimensionality reduction for modeling data. Information Sciences **491**, 16–29 (2019)
27. Wang, A., An, N., Chen, G., Li, L., Alterovitz, G.: Accelerating wrapper-based feature selection with k-nearest-neighbor. Knowledge-Based Systems **83**, 81–91 (2015)
28. Wang, L., Mao, Q.: Probabilistic dimensionality reduction via structure learning. IEEE Transactions on Pattern Analysis and Machine Intelligence **41**(1), 205–219 (2019)
29. Wang, W., Carreira-Perpinan, M.: The role of dimensionality reduction in classification. In: Proceedings of the AAAI Conference on Artificial Inteligence, pp. 2128–2134 (2004). http://arxiv.org/abs/1405.6444
30. Wei, X., Shen, H., Li, Y., Tang, X., Wang, F., Kleinsteuber, M., Murphey, Y.L.: Reconstructible nonlinear dimensionality reduction via joint dictionary learning. IEEE Transactions on Neural Networls and Learning Systems **30**(1), 175–189 (2019)
31. Xiao, G., Li, J., Chen, Y., Li, K.: MalFCS: An effective malware classification framework with automated feature extraction based on deep convolutional neural networks. Journal of Parallel and Distributed Computing **141**, 49–58 (2020)
32. Yang, B., Xiang, M., Zhang, Y.: Multi-manifold discriminant Isomap for visualization and classification. Pattern Recognition **55**, 215–230 (2016)
33. Zhang, Z., Chow, T.W.S., Zhao, M.: Trace ratio optimization-based semi-supervised nonlinear dimensionality reduction for marginal manifold visualization. IEEE Transactions on Knowledge and Data Engineering **25**(5), 1148–1161 (2013)