Learning Through Non-linearly Supervised Dimensionality Reduction

Josif Grabocka and Lars Schmidt-Thieme

Information Systems and Machine Learning Lab
Samelsonplatz 22, 31141 Hildesheim, Germany
{josif,schmidt-thieme}@ismll.uni-hildesheim.de

Abstract. Dimensionality reduction is a crucial ingredient of machine learning and data mining, boosting classification accuracy through the isolation of patterns via omission of noise. Nevertheless, recent studies have shown that dimensionality reduction can benefit from label information, via a joint estimation of predictors and target variables from a low-rank representation. In the light of such inspiration, we propose a novel dimensionality reduction which simultaneously reconstructs the predictors using matrix factorization and estimates the target variable via a dual-form maximum margin classifier from the latent space. Compared to existing studies which conduct the decomposition via linearly supervision of targets, our method reconstructs the labels using non-linear functions. If the hyper-plane separating the class regions in the original data space is non-linear, then a nonlinear dimensionality reduction helps improving the generalization over the test instances. The joint optimization function is learned through a coordinate descent algorithm via stochastic updates. Empirical results demonstrate the superiority of the proposed method compared to both classification in the original space (no reduction), classification after unsupervised reduction, and classification using linearly supervised projection.

Keywords: Machine Learning; Dimensionality Reduction; Feature Extraction; Matrix Factorization; Supervised Dimensionality Reduction

1 Introduction

Dimensionality reduction is an important ingredient of machine learning and data mining. The benefits of projecting data to latent spaces constitute in (i) converting large dimensionality datasets into feasible dimensions, but also (ii) improving the classification accuracy of small and medium datasets [1]. The scope of this work lies on improving prediction accuracy rather than ensuring scalability. There exists a trade-off between accurate and scalable methods, concretely a plain unsupervised dimensionality reduction is often advised for scalability (fast classification) purposes [2]. Via carefully tuned dimensionality reduction (aka feature extraction) we are able to retrieve the necessary patterns from the datasets, by leaving out the noise. Traditional dimensionality reduction (described in Section 2.1) has been focused on extracting features prior to classification. Such a mentality has been recently found to perform non-optimal [3, 4], since the features are not directly extracted/optimized for boosting classification. This discrepancy is created because the objective function (loss) of the unsupervised decomposition process is different from the one used during the evaluation of the prediction accuracy. Typically the L2 (Euclidean) error is used in approximating the original predictor values from the low-rank data, while a logistic, or hinge loss function, for the classification of targets. In order to solve this challenge, there have been attempts to incorporate class supervision into feature extraction, (mentioned in Section 2.3), such that the latent features are guided to enforce the discernment/separation of instances belonging to opposite classes in the reduced space.

Throughout this work we propose a principle, (details in Section 3.1), according to which dimensionality reduction should optimize the latent features through the same optimization function as the final classification method, thereby ensuring that the classification accuracy in the latent space is optimized. Inspired by the accuracy success of Support Vector Machines (SVM) which is largely credited to the kernel trick approach, we propose a novel supervised dimensionality reduction that incorporates kernel-based classification in the reduced dimension (Section 3). The novelty relies on defining a joint dimensionality reduction via matrix factorization, in parallel to a dual-form kernel-based maximum margin classification in the latent space. The reduced data is simultaneously updated in a coordinate descent fashion in order to optimize both loss terms. Experimental results (Section 4) demonstrate the superiority of the proposed method compared to both unsupervised dimensionality reduction and classification in the original space. The main contribution of this work are:

1. Defined a supervised dimensionality reduction with a kernel-based target variable estimation
2. Reviewed and elaborated the state of the art in supervised dimensionality reduction
3. Derived a coordinate descent algorithm which simultaneously learns the latent factors for the reconstruction of predictors and the accuracy over target
4. Compared the paradigms of linearly versus non-linearly supervised dimensionality reduction
5. Provided empirical results to demonstrate the superiority of the proposed method

2 Related Work

2.1 Dimensionality Reduction

Dimensionality reduction is a field of computer science that focuses on extracting lower dimensionality features from datasets [1]. Numerous techniques exist for extracting features. Principal Component Analysis (PCA) is a famous approach involving orthogonal transformations and selecting the topmost principal components, which preserve necessary variance [5]. Alternatively, Singular Value

Decomposition decomposes a dataset into latent unitary, nonnegative diagonal and conjugate transpose unitary matrices [1].

Further elaborations of dimensionality reductions involve nonlinear decomposition of data [6]. For instance kernel PCA replaces the linear operations of PCA through nonlinear mappings in a Reproducing Kernel Hilbert Space [7]. The whole subfield of manifold learning elaborates, as well, on nonlinear projections. Specifically, Sammon's mappings preserves the structure of instance distances in the reduced space [8], while principal curves embed manifolds using standard geometric projections [9]. More nonlinear dimensionality algorithms are described in [10]. In addition, temporal dimensionality reduction have been proposed in scenarios where the time difference of observations is not evenly spaced [11]. The field of Gaussian Processes have been extended to dimensionality reduction through the Gaussian Processes Latent Variable Models (GPLVM) [12, 13]. In comparison to PCA, the GPLVM models define nonlinear approximative functions for the predictor values [12].

2.2 Matrix Factorization

Matrix factorization refers to a family of decompositions which approximate a dataset as a product of latent matrices of typically lower dimensions. A generalization and categorization of the various proposed factorization models is elaborated in [14], where factorizations are seen as applications of the Bregman Divergence paradigm. The learning of the decomposition is typically conducted by defining a L2-norm and updating the latent matrices via a stochastic gradient descent algorithm [15]. Matrix factorization has been applied in a range of domains, ranging from recommender systems where decomposition focuses on collaborative filtering of sparse user-item ratings [16], up to time series dimensionality reduction [17]. The Matrix Factorization approach is a special instance of a probabilistic PCA, while it extends the functionality of a PCA by adding bias terms [15]. In terms of similarities, a factorization can be also characterized as a biased probabilistic SVD. In a broader sense, the linear approximation of predictors can be also interpreted as an instance of the GPLVM models for a linear (polynomial of degree one) kernel [12].

2.3 Supervised Dimensionality Reduction

In addition to the standard dimensionality reduction and Matrix Factorization, there has been attempts to utilize the labels information, therefore dictating a supervised projection. Fisher's linear discriminant analysis is a popular supervised projection method [18]. The classification accuracy loss objective functions occurring in literature vary from label least square regression [19], to generalized linear models [20], linear logistic regression [3], up to hinge loss [4, 21]. Another study aimed at describing the target variable as being conditionally dependent on the features [22]. Other families of supervisions strive for preserving the neighborhood structure of intra-class instances [23], or links in a semi supervised scenarios [24]. The GPLVM models have been, as well, adopted for discriminative classification [25]. A self-contained description of the state of the art in linearly supervised dimensionality reduction is offered in Section 3.4. In comparison to the aforementioned methods, we propose a supervised dimensionality reduction with a kernel-based classifier that directly optimizes the dual formulation in the projected space.

3 Proposed Method

3.1 Principle

The method proposed in this study relies on the principle that feature extraction, analogously referred also as dimensionality reduction, should not be conducted "ad-hoc" or via particular heuristics. Most of the classification tasks have a unifying objective, which is to improve classification accuracy. In that context we are referring as "ad-hoc" to the family of feature extraction techniques that don't directly optimize their loss functions for classification accuracy. Unsupervised projection is not optimized for the same loss function which is used during the evaluation of the target variable. Techniques such as SVD or Matrix Factorization focus solely on approximating the predictors of the original data. Unfortunately, unsupervised decompositions pose the risk of losing the signal relevant to the prediction accuracy. While such approaches approximate the original data, they become vulnerable to the noise present in the observed predictors' values. Stated else-wise, we believe that instance labels should guide the feature extraction, such that the utilization of the extracted features improves accuracy. In that perspective, we propose a feature extraction method which operates by optimizing a joint objective function composed of the feature extraction term and also the classification accuracy term. In comparison with similar feature extraction ideas reviewed in Section 2.3, which use linear classifiers in the optimization, we propose a novel method which learns a nonlinear SVM over the projected space via jointly optimizing a dual form together with dimensionality reduction. Further details will be covered throughout Section 3 which is organized progressively. Initially the unsupervised dimensionality reduction is explained and then the state of the art in linearly supervised decomposition. Finally the stage is ready for introducing our novel method on non-linearly supervised dimensionality reduction.

3.2 Introduction to Supervised Dimensionality Reduction

Unsupervised dimensionality reduction (e.g.: matrix factorization described in Section 3.3), is guided only by the reconstruction loss. Such an approach does not take into consideration the classification accuracy impact of the extracted features, therefore the produced reduced dimensionality data is not optimized to improve accuracy. In order to overcome such a drawback, the so called supervised dimensionality reduction has been proposed by various authors (see Section 2.3).

The key commonalities of those supervised dimensionality methods rely on defining a joint optimization function, consisting of the reconstruction loss terms and the classification accuracy terms.

The typical classification accuracy loss term focuses on defining a classifier in the latent space, i.e. $U \in \mathbb{R}^{(n+n')\times d}$, via a hyperplane defined by the weights vector $W \in \mathbb{R}^d$, such that the weights can correctly classify the training instances of U in order to match observed label $Y \in \mathbb{R}^n$. Equation 1 defines a cumulative joint optimization function using a reconstruction term for the predictors, denoted $F_R(X, U, V)$, and a classification accuracy term, denoted $F_{CA}(Y, U, W)$. The trick of such a joint optimization constitutes on updating the low-rank data U simultaneously, in order to minimize both F_R and F_{CA} via gradient descent on both loss terms. The hyper parameter β is a switch which balances the impact of reconstruction vs classification accuracy. Throughout this paper we evaluate the binary classification problem, even though the explained methods could be trivially transferred to multi-nominal target variables by employing the one-vs-all technique. Should that be needed, we would have to build as many classifiers as there are categories in the target variable, while each classifier would treat one category value as the positive class and all the remaining categories as the negative class. In addition to the reconstruction F_R and the classification accuracy F_Ca loss terms, the model has additive regularization terms parametrized by coefficients $\lambda_U, \lambda_V, \lambda_W$. Such a regularization helps the model avoid overfitting and enables a better generalization over the test instances.

$$F(X, Y, U, V, W) = \beta\, F_R(X, U, V) + (1 - \beta)\, F_{CA}(Y, U, W) \tag{1}$$

3.3 Matrix Factorization as Dimensionality Reduction

Matrix factorization is a dimensionality reduction technique which decomposes a dataset $X \in \mathbb{R}^{(n+n')\times m}$ matrix of n training instances and n' testing instances, per m features, into two smaller matrices of dimensions $U \in \mathbb{R}^{(n+n')\times d}$ and $V \in \mathbb{R}^{d\times m}$ [15]. The latent/reduced projection of the original data X is the latent matrix U, where d is the dimensionality of the projected space. Typically d is much smaller than m, meaning that the dimensionality is reduced. In case $d < m$, then U is nominated as the low-rank representation of X. Otherwise, if $d > m$ a *non-grata* inflation phenomenon is achieved. Such decomposition is expressed in a form of a regularized reconstruction loss, denoted $F_R(X, U, V)$ and depicted in Equation 2. The optimization of such a function aims at computing latent matrices U, V such that their dot product approximates the original matrix X via an Euclidean distance ($L2$ norm) loss. In addition to the $L2$ reconstruction norm, we also add $L2$ regularization terms weighted by factors λ_U, λ_V in order to avoid over-fitting.

$$\underset{U,V}{\operatorname{argmin}}\ F_R(X, U, V) = ||X - UV||^2 + \lambda_U ||U||^2 + \lambda_V ||V||^2 \tag{2}$$

Bias terms, $B_U \in \mathbb{R}^{(n+n')\times 1}, B_V \in \mathbb{R}^{1\times m}$ are added to the reconstruction loss [15], such that each element of B_U incorporates the prior belief value of the respective instance, while each element of B_V the prior belief value of the respective feature. More concretely the loss can be expanded as a reconstruction of each cell $X_{i,j}$ as depicted by Equation 3.

$$\begin{aligned} \underset{U,V,B_U,B_V}{\operatorname{argmin}} \; F_R(X,U,V) &= \sum_{i=1}^{n+n'} \sum_{j=1}^{m} \left(X_{i,j} - \left(\sum_{k=1}^{d} U_{i,k} V_{k,j} + B_{U_i} + B_{V_j} \right) \right)^2 \\ &+ \lambda_U \sum_{i=1}^{n+n'} \sum_{k=1}^{d} U_{i,k}^2 + \lambda_V \sum_{k=1}^{d} \sum_{j=1}^{m} V_{k,j}^2 \end{aligned} \tag{3}$$

In order to learn the Matrix Factorization defined in Equation 3 we need to define the gradients to be used for updating our latent matrices. Stochastic Gradient Descent is a fast optimization technique for factorizations [15] and operates by reducing the approximation error of each cell (i,j) of X. Therefore, we can represent the reconstruction loss F_R as sum of smaller loss terms $F_{R i,j}$, per each cell (i,j) of the original dataset X. Such a decomposition will later enable the stochastic gradient descent to optimize for each small loss term stochastically, i.e. the indices (i,j) will be visited randomly.

$$F_R(X,U,V) = \sum_{i=1}^{n+n'} \sum_{j=1}^{m} F_R(X,U,V)_{i,j} \tag{4}$$

$$\begin{aligned} F_R(X,U,V)_{i,j} &= \beta \left(X_{i,j} - \left(\sum_{k=1}^{d} U_{i,k} V_{k,j} + B_{U_i} + B_{V_j} \right) \right)^2 + \\ &\lambda_U \frac{1}{m} \sum_{k=1}^{d} U_{i,k}^2 + \lambda_V \frac{1}{n+n'} \sum_{k=1}^{d} V_{k,j}^2 \end{aligned} \tag{5}$$

The gradients of the latent data U, V with respect to the reconstruction loss are computed as the first derivative of the loss. The error in approximating a cell $X_{i,j}$ is defined as $e_{i,j}$ and can be pre-computed for scalability. As can be observed from the gradients of Equations 6-9, the pre-computed error term $e_{i,j}$ is used in all gradients.

$$e_{i,j} = X_{i,j} - \sum_{k=1}^{d} U_{i,k} V_{k,j} - B_{U_i} - B_{V_j} \tag{6}$$

$$\frac{\partial F_R(X,U,V)_{i,j}}{\partial U_{i,k}} = -2\beta \, e_{i,j} \, V_{k,j} + 2\lambda_U \frac{1}{m} U_{i,k} \tag{7}$$

$$\frac{\partial F_R(X,U,V)_{i,j}}{\partial V_{k,j}} = -2\beta \, e_{i,j} \, U_{i,k} + 2\lambda_V \frac{1}{n+n'} V_{k,j} \tag{8}$$

$$\frac{\partial F_R(X,U,V)_{i,j}}{\partial B_{U_i}} = \frac{\partial F_R(X,U,V)_{i,j}}{\partial B_{V_j}} = -2\beta \, e_{i,j} \tag{9}$$

3.4 Linearly Supervised Dimensionality Reduction

The linear supervision of the dimensionality reduction refers to the inclusion of a linear classification loss term to the objective function, expressed as F_{CA} in Equation 1. The addition of the linear classification loss term enforces the instances of different classes to be linearly separable in the low-rank space. Various loss terms have been proposed depending on the utilized linear classifier. Before explaining the different losses, we introduce the predicted value of instance i as $\hat{Y}_i$ and defined in Equation 10. The predicted value is the dot product of the instance values $U_{i,:} \in \mathbb{R}^d$ and linear weights $W \in \mathbb{R}^d$. In addition, the bias term for the instance $B_{U_i} \in \mathbb{R}$ and the bias of the classification weight vector $W_0 \in \mathbb{R}$ are summed up.

$$\hat{Y}_i = B_{U_i} + W_0 + \sum_{k=1}^{d} U_{i,k} W_k, \quad \forall i \in \mathbb{N}_i^n \tag{10}$$

Loss terms quantify the degree of violation that a classifier exhibits from the desired (perfect) prediction accuracy. Concretely the least square loss measures the L2 distance between the true targets Y and predicted vales $\hat{Y}$. In the context of linearly supervised reduction [19], the least-squares loss term can be defined as shown in Equation 11. Similar to the regression case, least squares is adopted for classification by treating the target values as $Y \in \{-1, 1\}^n$, while predicted positive values $\hat{Y}$ indicate a positive class and vice versa.

$$F_{CA}(Y, U, W)^{LS} = \sum_{i=1}^{n} \left(Y_i - \hat{Y}_i\right)^2 + Reg(U, W), \quad \forall i \in \mathbb{N}_i^n \tag{11}$$

The logistic loss has been applied to guide the decomposition by minimizing the target prediction error along a sigmoid curve [3]. Equation 12 presents the loss, while the target values are expected to be in the range $Y \in \{0, 1\}^n$. Please note that the sigmoid function is defined as: $\text{sigmoid}(\hat{Y}) = \frac{1}{1+e^{-\hat{Y}}}$.

$$F_{CA}(Y, U, W)^{LO} = \sum_{i=1}^{n} -Y_i \log(\text{sigmoid}(\hat{Y}_i)) - (1 - Y_i) \times \tag{12}$$

$$\log\left(1 - \text{sigmoid}(\hat{Y}_i)\right) + Reg(U, W) \tag{13}$$

Another strong linear classifier is the hinge loss, which represents the underlying foundation of the Support Vector Machines is depicted in Equation 14. The hinge loss has been also applied to supervised reduction [4]. The hinge loss is also called a maximum margin loss because it tries to find a margin of unit size between the hyperplane W and the region of each class.

$$F_{CA}(Y, U, W)^{HI} = \sum_{i=1}^{n} \max(0, 1 - Y_i \hat{Y}_i) + Reg(U, W), \quad \forall i \in \mathbb{N}_i^n \tag{14}$$

Unfortunately the hinge loss is not differentiable at $Y\hat{Y} = 1$, therefore a smoothed variant of the hinge loss [21] is preferred in cases where a gradient based optimization is needed (Equation 15).

$$F_{CA}(Y,U,W)^{SH} = \sum_{i=1}^{n} \left(\begin{cases} 1 - Y_i\hat{Y}_i & Y_i\hat{Y}_i < 0 \\ \frac{1}{2}\left(1 - Y_i\hat{Y}_i\right)^2 & 0 \leq Y_i\hat{Y}_i < 1 \\ 0 & Y_i\hat{Y}_i \geq 1 \end{cases} \right) + Reg(U,W) \quad (15)$$

The regularization term is a L2 norm and defined in Equation 16. The regularization parameters λ_U, λ_W control the complexity of the model and avoid over-fitting.

$$Reg(U,W) = \sum_{i=1}^{n+n'} \sum_{k=1}^{d} {U_{i,k}}^2 + \sum_{k=1}^{d} {W_k}^2 \quad (16)$$

The Advantage of Supervised Decomposition relies on using the label information to guide the projection. In that way, any noise which is present in the observed data can be eliminated in the low-rank representation.

In order to show the advantage of the supervised decomposition, we present the experiment of Figure 1. A 2-dimensional synthetic dataset of ten instances, belonging to two classes (red, blue) is depicted in sub-figure a). Please note that the original data are **linearly separable** by a hyperplane. Then, we added a random variable X_3 (shown in b)) of uniform random values between $[-1,1]$. The experiment aims at reducing the 3-dimensional noisy data back to 2-dimensions using both unsupervised and supervised dimensionality reductions. As can be observed, the unsupervised projection is affected by the added noise and the resulting 2-dimensional data in c) is not anymore linearly separable. In contrast, the linearly supervised decomposition can benefit from a linear classification accuracy loss term to separate instances by label. A smooth hinge loss supervised decomposition was applied to the decomposition of d). Please note that the resulting 2-dimensional projection depicted in d) is linearly separable as the original data. The experiment demonstrates that a supervised decomposition has stronger immunity towards the presence of noise in the data. For the sake of reproducibility, the parameters used during the experiment are provided in the caption note.

Learning the Linearly Supervised Decomposition is carried on through optimizing the latent weights U and W by taking a step in the first derivative of the classification accuracy term F_{CA}. In comparison to full gradient approaches, stochastic techniques operate by eliminating the error of a random single instance i, i.e. optimizing for $F_{CA i}$. Since the gradient computations for each instance are much simpler than for the full dataset, the stochastic gradient descent computes faster than the full gradient learning.

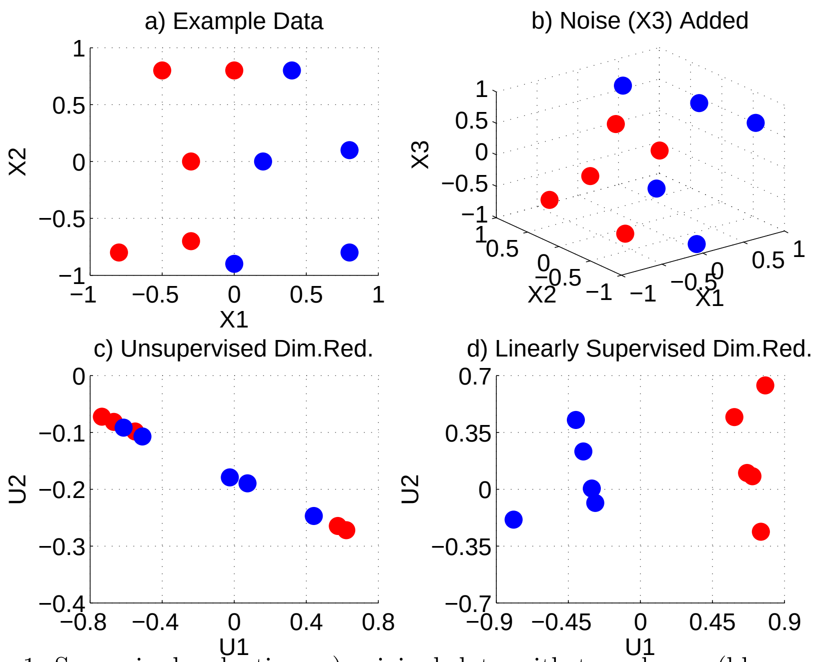

Fig. 1: Supervised reduction: a) original data with two classes (blue and red); b) random noise variable (X3) added; c) unsupervised dimensionality reduction through matrix factorization (from b) to c)); d) linearly supervised maximum margin dimensionality reduction (from b) to d)). Parameters: $\eta_R = \eta_{CA} = 0.001$, $\lambda_U = \lambda_V = 0.0001$, $\lambda_V =$ and $\beta = 0.4$.

More specifically, the gradients of the least-squares loss term are shown in Equations 17-19 and are the result of the first derivative with respect to each cell of U, W and the biases B_{U_i}, W_0.

$$\frac{\partial F_{CA}(Y,U,W)_i^{LS}}{\partial U_{i,k}} = -2\left(Y_i - \hat{Y}_i\right) W_k + 2\frac{\lambda_U}{m} U_{i,k} \tag{17}$$

$$\frac{\partial F_{CA}(Y,U,W)_i^{LS}}{\partial W_k} = -2\left(Y_i - \hat{Y}_i\right) U_{i,k} + 2\frac{\lambda_W}{n} W_k \tag{18}$$

$$\frac{\partial F_{CA}(Y,U,W)_i^{LS}}{\partial B_{U_i}} = \frac{\partial F_{CA}(Y,U,W)_i^{LS}}{\partial W_0} = -2\left(Y_i - \hat{Y}_i\right) \tag{19}$$

The logistic loss has derivatives similar to the least square loss with the difference being the inclusion of the sigmoid of the predicted target value $\hat{Y}$. The detailed gradients for the latent weights, with respect to the logistic loss term $F_{CA}(Y,U,W)^{LO}$ are presented in Equations 20-22.

$$\frac{\partial F_{CA}(Y,U,W)_i^{LO}}{\partial U_{i,k}} = -2\left(Y_i - \text{sigmoid}(\hat{Y}_i)\right) W_k + 2\frac{\lambda_U}{m} U_{i,k} \quad (20)$$

$$\frac{\partial F_{CA}(Y,U,W)_i^{LO}}{\partial W_k} = -2\left(Y_i - \text{sigmoid}(\hat{Y}_i)\right) U_{i,k} + 2\frac{\lambda_W}{n} W_k \quad (21)$$

$$\frac{\partial F_{CA}(Y,U,W)_i^{LO}}{\partial B_{U_i}} = \frac{\partial F_{CA}(Y,U,W)_i^{LS}}{\partial W_0} = -2\left(Y_i - \text{sigmoid}(\hat{Y}_i)\right) \quad (22)$$

The optimization of the smooth hinge loss requires the optimization of the three conditional steps present in the loss function, for the regions $Y\hat{Y} < 0$, $0 \leq Y\hat{Y} < 1$ and $Y\hat{Y} < 1$. The computation of the gradients follows a similar practice as the other aforementioned loss types. First derivatives of the smooth hinge loss term F_{CA}^{SH} are computed per each latent weight U, W, B_U, W_0. The derived gradients are shown in Equations 23-25.

$$\frac{\partial F_{CA}(Y,U,W)_i^{SH}}{\partial U_{i,k}} = \left(\begin{cases} -Y_i\hat{Y}_i W_k & Y_i\hat{Y}_i < 0 \\ -\left(1 - Y_i\hat{Y}_i\right) W_k & 0 \leq Y_i\hat{Y}_i < 1 \\ 0 & Y_i\hat{Y}_i \geq 1 \end{cases}\right) + 2\frac{\lambda_U}{m} U_{i,k} \quad (23)$$

$$\frac{\partial F_{CA}(Y,U,W)_i^{SH}}{\partial W_k} = \left(\begin{cases} -Y_i\hat{Y}_i U_{i,k} & Y_i\hat{Y}_i < 0 \\ -\left(1 - Y_i\hat{Y}_i\right) U_{i,k} & 0 \leq Y_i\hat{Y}_i < 1 \\ 0 & Y_i\hat{Y}_i \geq 1 \end{cases}\right) + 2\frac{\lambda_W}{n} W_k \quad (24)$$

$$\frac{\partial F_{CA}(Y,U,W)_i^{SH}}{\partial B_{U_i}} = \frac{\partial F_{CA}(Y,U,W)_i^{SH}}{\partial W_0} = \begin{cases} -Y_i\hat{Y}_i & Y_i\hat{Y}_i < 0 \\ -\left(1 - Y_i\hat{Y}_i\right) & 0 \leq Y_i\hat{Y}_i < 1 \\ 0 & Y_i\hat{Y}_i \geq 1 \end{cases} \quad (25)$$

A Final Learning Algorithm is constructed by applying the defined gradients in a stochastic gradient descent approach over the reconstruction and accuracy loss terms. Algorithm 1 concatenates all the pieces of the learning process. The learning process is separated into two main sections, namely (i) the updates with respect to the reconstruction loss and (ii) the updates with respect to the classification accuracy loss terms. The first loop iterates over all cells of X indexed by row-column pairs (i, j), and also updates all the cells of U according to the error present in approximating $X_{i,j}$. Similarly, the second loop iterates over the train targets Y_i and corrects the classification errors. The name of the loss term (LT) is a generic placeholder and aforementioned gradients of each loss (least-squares, logistic and hinge) can be directly plugged in.

Updates are applied to all the cells of U, V, W and the biases B_U, B_V, W_0 in a stochastic gradient fashion, i.e. visited randomly. The random updates speed up the learning process because the continuous update of columns from a single row is avoided. An update relies on decrementing the value of a cell in the direction of the aforementioned gradients. The magnitude of the decrement step is controlled

Algorithm 1 Learning Algorithm: Linearly Supervised Dim. Red.

```
Input: Dataset matrix $X \in \mathbb{R}^{(n+n')\times m}$, Labels vector $Y \in \mathbb{R}^n$, Parameters:
  {Optimization switch $\beta$, Latent dimensions $d$, Learning rates $\eta_R, \eta_{CA}$, Regularizations $\lambda_U, \lambda_V, \lambda_W$ }
Output: $U, V, B_U, B_V, W, W_0$
  Initialize randomly $U \in \mathbb{R}^{(n+n')\times d}$, $V \in \mathbb{R}^{d\times m}$, $W \in \mathbb{R}^d$, $B_U \in \mathbb{R}^{(n+n')\times 1}$, $B_V \in \mathbb{R}^{1\times m}$, $W_0 \in \mathbb{R}$
  while $F_R + F_{CA}$ not reached an optimum do
    for $\forall(i,j,k) \in (\{1...(n+n')\}, \{1...m\}, \{1...d\})$ in random order do
      $U_{i,k} \leftarrow U_{i,k} - \eta_R \frac{\partial F_R(X,U,V)_{i,j}}{\partial U_{i,k}}$
      $V_{k,j} \leftarrow V_{k,j} - \eta_R \frac{\partial F_R(X,U,V)_{i,j}}{\partial V_{k,j}}$
      $B_{U_i} \leftarrow B_{U_i} - \eta_R \frac{\partial F_R(X,U,V)_{i,j}}{\partial B_{U_i}}$
      $B_{V_j} \leftarrow B_{V_j} - \eta_R \frac{\partial F_R(X,U,V)_{i,j}}{\partial B_{V_j}}$
    end for
    $LT \leftarrow \{LS, LO, SH\}$ {LT stands for 'Loss Type'}
    for $\forall(i,k) \in (\{1...n\}, \{1...d\})$ in random order do
      $U_{i,k} \leftarrow U_{i,k} - \eta_{CA} \frac{\partial F_{CA}(Y,U,W)_i^{LT}}{\partial U_{i,k}}$
      $W_k \leftarrow W_k - \eta_{CA} \frac{\partial F_{CA}(Y,U,W)_i^{LT}}{\partial W_k}$
      $B_{U_i} \leftarrow B_{U_i} - \eta_{CA} \frac{\partial F_{CA}(Y,U,W)_i^{LT}}{\partial B_{U_i}}$
      $W_0 \leftarrow W_0 - \eta_{CA} \frac{\partial F_{CA}(Y,U,W)_i^{LT}}{\partial W_0}$
    end for
  end while
  return $U, V, B_U, B_V, W, W_0$
```

using a learning rate parameters. Technically, there are two learning rates for each relation η_R and η_{CA}, which can also have equal values. The second cycle iterates over all the low-rank instances $U_{i,:}$ and the classification weights vector W_k.

$$\hat{Y}_t^{LS,SH} = \text{sign}(W_0 + B_{U_t} + \sum_{k=1}^{d} U_{t,k} W_k), \qquad t \in \mathbb{N}_{n+1}^{n'} \quad (26)$$

$$\hat{Y}_t^{LO} = \text{sigmoid}(W_0 + B_{U_t} + \sum_{k=1}^{d} U_{t,k} W_k) > \frac{1}{2} \; ? \; 1 \; : \; 0, \qquad t \in \mathbb{N}_{n+1}^{n'} \quad (27)$$

Once the weights are learned, then the prediction of the test instances can be produced as shown in Equations 26-27. For the least squares and the smooth hinge loss the sign function defines the prediction, i.e.: the positive class for positive predictive values, otherwise the negative class. On the other side, the logistic loss is defined in Equation 27 as one of $\{0, 1\}$ based on the value threshold of 0.5.

3.5 Nonlinearly Supervised Dimensionality Reduction

In comparison to previous approaches that propose linear models, in this study we propose a kernel-based **binary** classifier approach in the latent space U. Let us initially define the classification accuracy loss term, denoted $F_{CA}(Y,U,W)$, in Equation 28, in form of a maximum margin soft SVMs with hinge loss [26]. Such form of the SVMs is called the primal form. The parameter C scales the penalization of the instances violating the distances from the maximum margin. Please note that W_0 is the intercept bias term of the hyperplane weights vector W.

$$\begin{aligned} \underset{U,W}{\operatorname{argmin}} \; & F_{CA}(Y,U,W) = \frac{1}{2}||W||^2 + C\sum_{i=1}^{n}\xi_i \qquad (28) \\ \text{s.t: } & Y_i(\langle W, U_i\rangle + B_{U_i} + W_0) \geq 1 - \xi_i, \quad i = 1,...,n \\ & \xi_i \geq 0, \quad i = 1,...,n \end{aligned}$$

Unfortunately the primal form doesn't support kernels, therefore we have to convert the optimization functions into the dual form equation 29. In order to get rid of of the inequality constraint we apply Lagrange multipliers to include the inequalities by introducing dual variables α_i per instance and adding $\alpha_i\,(y_i(\langle W, U_i\rangle + W_0))$ to the optimization function for all instance i. Then we solve the objective function for W and W_0 by equating the first derivative to zero. Putting the derived expressions of W and W_0 to the objective function, we obtain the so-called dual representation optimization:

$$\underset{U,\alpha}{\operatorname{argmin}} F_{CA}(Y,U,\alpha) = \frac{1}{2}\sum_{i=1}^{n}\sum_{l=1}^{n}\alpha_i\alpha_l Y_i Y_l \langle [U_{i,*}, B_{U_i}], [U_{t,*}, B_{U_l}]\rangle - \sum_{i=1}^{n}\alpha_i \quad (29)$$

$$\text{s.t: } 0 \leq \alpha_i \leq C, \quad i = 1,...,n; \qquad \text{and} \qquad \sum_{i=1}^{n}\alpha_i Y_i = 0$$

Once the optimization model is build any new test instance U_t can be classified in terms of learned α as shown in Equation 30.

$$\hat{Y}_t = \text{sign}\left(\sum_{i=1}^{n}\alpha_i Y_i \langle [U_{i,*}, B_{U_i}], [U_{t,*}, B_{U_l}]\rangle + W_0\right) \qquad (30)$$

The dot product, found in the dual formulation, between the instance vectors appears both in the optimization function 29 and the classification function 30. Such a dot product can be replaced by the so called kernel functions [26]. Various kernel representations exists, however in this study, for the sake of clarity and generality, we are going to prove the concept of the method using polynomial kernels, defined in Equation 31, which are known to be successful off-the-shelf kernels [26].

$$K([U_{i,*}, B_{U_i}], [U_{t,*}, B_{U_l}]) = \left(B_{U_i}B_{U_l} + \sum_{k=1}^{d} U_{i,k}U_{l,k} + 1\right)^p \qquad (31)$$

The ultimate objective function that defines nonlinear supervised dimensionality reduction is presented in Equation 32. *This model, in cooperation with the forthcoming learning algorithm, are the main contributions of our paper.*

$$\begin{aligned}
\underset{U,V,\alpha,B_U,B_V}{\operatorname{argmin}} \quad F(X,Y,U,V,\alpha) &= \beta \sum_{i=1}^{n+n'} \sum_{j=1}^{m} \left(X_{i,j} - \left(\sum_{k=1}^{d} U_{i,k} V_{k,j} + B_{U_i} + B_{V_j} \right) \right)^2 \\
&+ (1-\beta) \left(\frac{1}{2} \sum_{i=1}^{n} \sum_{l=1}^{n} \alpha_i \alpha_l Y_i Y_l \, K([U_{i,*}, B_{U_i}], [U_{t,*}, B_{U_l}]) - \sum_{i=1}^{n} \alpha_i \right) \\
&+ \lambda_U \sum_{i=1}^{n+n'} \sum_{k=1}^{d} U_{i,k}^2 + \lambda_V \sum_{k=1}^{d} \sum_{j=1}^{m} V_{k,j}^2 \qquad (32) \\
\text{s.t:} \quad & 0 \le \alpha_i \le C, \;\; i = 1, ..., n \\
& \sum_{i=1}^{n} \alpha_i Y_i = 0
\end{aligned}$$

Meanwhile the classification of a test instance U_t using kernels and the learned U, α, resulting from the solution of the dual joint optimization is shown in Equation 33.

$$Y_t = \operatorname{sign} \left(\sum_{i=1}^{n} \alpha_i Y_i \, K([U_{i,*}, B_{U_i}], [U_{t,*}, B_{U_l}]) + W_0 \right) \qquad (33)$$

The Benefit of Non-linear Supervision is the ability to preserve both the reconstruction and the classification accuracy. This dual objective is achieved best if there is no sacrifice in terms of reconstruction. More concretely, let us assume the original data is non-linearly separable. Then, a linearly supervised decomposition cannot easily minimize both F_R and F_{CA}. The handicap is created due to trying to classify the low-rank data linearly, even though the original data is non-linear. As a consequence, the structure of the data cannot be accurately preserved and the reconstruction is poor, i.e. high F_R error. Unable to preserve the structure of the data, a linearly supervised decomposition struggles to achieve a competitive generalization of prediction accuracy over the test instances.

Figure 2 illustrates the benefit of the non-linear supervision with a concrete experiment. A 2-dimensional synthetic non-linearly separable dataset is created in sub-figure a). For experimental purposes we added noise through a new variable X_3 that contains random values between [-1,1]. The key aspect of the experiment is to project the noisy 3-dimensional data back to 2-dimensions using both linearly (*c)*) and non-linearly (*d)*) supervised reductions. The projection parameters are found in the caption comment. As can be seen from sub-figure c), the linear supervision cannot linearly separate all instances in the low-rank space under reasonable β values. On the contrary, a non-linear decomposition can achieve a 0% training error, because a non-linear arrangement of the data

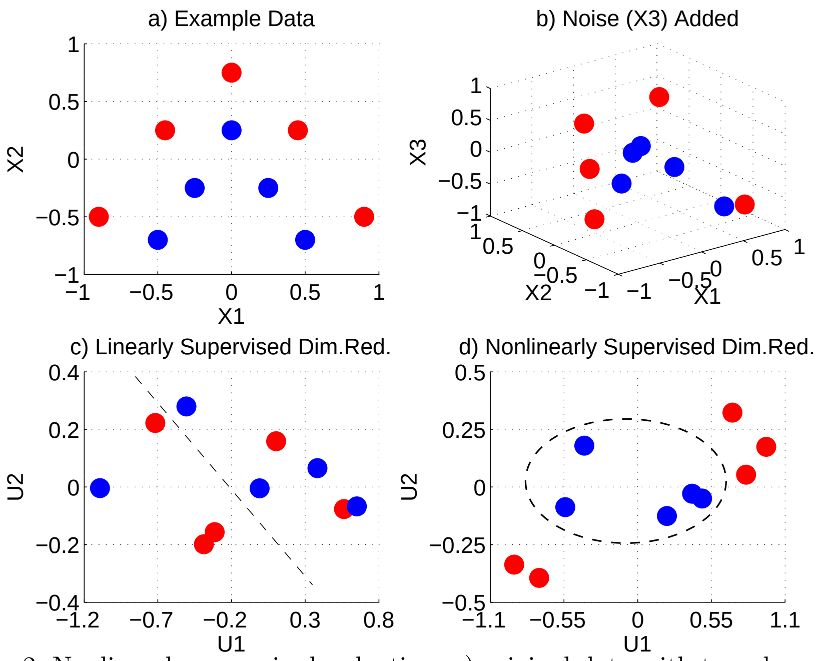

Fig. 2: Nonlinearly supervised reduction: a) original data with two classes (blue and red); b) random noise variable (X3) added; c) linearly supervised dimensionality reduction (from b) to c)); d) non-linearly supervised dimensionality reduction (from b) to d)). Parameters: $\eta_R = \eta_{CA} = 0.001$, $\lambda_U = \lambda_V = 0.01$, $\lambda_W = 2$, $C = 0.5$, $p = 3$ and $\beta = 0.7$ and 300 iterations.

is easily achieved in the low-rank space. Please note that reasonable switch parameter values are $\beta > 0$. In the absurd case of $\beta = 0$ no reconstruction loss updates will be applied and the classification loss term will create a low-rank arrangement of the training instances without preserving at all the structure of the original data. Such a classifier is destined to under-perform over the test data.

3.6 Algorithm for Learning the Non-linearly Supervised Dimensionality Reduction

The objective function of Equation 32 is a non-convex function in terms of U, V and W, which makes it challenging for optimization. However stochastic gradient descent is shown to perform efficiently in minimizing such non-convex functions [15]. The benefits of stochastic gradient descent rely on better convergence, because cells of X are randomly picked for optimization, thus updating different rows of U, instead of iterating through the all the features of the same instance.

On the other side, the classification accuracy terms of Equation 29 can be solved, in terms of α, by any standard SVMs dual solver method in case we consider U to be fixed. Thus, in an alternating fashion we solve the α-s by keeping U fixed. Then in the next step we update U using the learned α-s and V matrix, by taking a step in the negative direction of the overall loss w.r.t U. The update of V is performed as last step. Those three steps can be repeated until convergence as shown in the Algorithm 3.

Similarly, we can split up the classification accuracy loss term, F_{CA}, into smaller loss terms $F_{CA\,i,l}$, defined per each instance pair (i, l).

$$F_{CA}(Y,U,\alpha) = \sum_{i=1}^{n}\sum_{l=1}^{n} F_{CA}(Y,U,\alpha)_{i,l} \tag{34}$$

$$F_{CA}(Y,U,\alpha)_{i,l} = (1-\beta)\left(\frac{\alpha_i\alpha_l}{2}Y_iY_l\ K([U_{i,*},B_{U_i}],[U_{l,*},B_{U_l}]) - \frac{\alpha_i+\alpha_l}{n^2}\right) \tag{35}$$

Gradients:

$$\frac{\partial F_{CA}(Y,U,\alpha)_{i,l}}{\partial U_{i,k}} = \frac{\beta-1}{2}\alpha_i\alpha_lY_iY_l\ p\left(B_{U_i}B_{U_l}+\sum_{k=1}^{d}U_{i,k}U_{l,k}+1\right)^{p-1} U_{l,k} \tag{36}$$

$$\frac{\partial F_{CA}(Y,U,\alpha)_{i,l}}{\partial U_{l,k}} = \frac{\beta-1}{2}\alpha_i\alpha_lY_iY_l\ p\left(B_{U_i}B_{U_l}+\sum_{k=1}^{d}U_{i,k}U_{l,k}+1\right)^{p-1} U_{i,k} \tag{37}$$

$$\frac{\partial F_{CA}(Y,U,\alpha)_{i,l}}{\partial B_{U_i}} = \frac{\beta-1}{2}\alpha_i\alpha_lY_iY_l\ p\left(B_{U_i}B_{U_l}+\sum_{k=1}^{d}U_{i,k}U_{l,k}+1\right)^{p-1} B_{U_l} \tag{38}$$

$$\frac{\partial F_{CA}(Y,U,\alpha)_{i,l}}{\partial B_{U_l}} = \frac{\beta-1}{2}\alpha_i\alpha_lY_iY_l\ p\left(B_{U_i}B_{U_l}+\sum_{k=1}^{d}U_{i,k}U_{l,k}+1\right)^{p-1} B_{U_i} \tag{39}$$

The updates of α-s is carried through an algorithm which is a reduced version of the Sequential Minimal Optimization (SMO) [27]. Since the dual form optimization function contains the constraint $\sum_{i=1}^{n}\alpha_iY_i = 0$, then any update of an α_i will violate the constraint. Therefore SMO updates the α-s in pair, offering three heuristics which defines which subset of the pairs should be updates first, in order to speed up the algorithm.

In difference to the original algorithm, we have ignored the selection heuristic for the α pairs to update. The reason for omitting the heuristics is due to the fact that U instances are continuously updated/modified. For instance, let us consider an imaginary instance U_i far away from the decision boundary, which means $\alpha_i = 0$. However in the next iteration, the instance U_i might be updated and move close to the boundary, meaning that α_i becomes a candidate for being updated $(0 < \alpha_i \leq C)$, opposite to the functioning of SMO heuristic that would have avoided updating the instance, alluding that α_i is still 0.

The alpha updates rely on solving the function analytically for a pair of α-s at a step, until no $\alpha_i, \forall i$, violates the KKT [27] conditions described in Equation 40.

Algorithm 2 UpdateAlphaPair

```
Input: First alpha index $i$, Second alpha index $j$
Output: Updated $\alpha$ and $W_0$
    $(\alpha_i^{old}, \alpha_j^{old}) \leftarrow (\alpha_i, \alpha_j)$
    Let $s \leftarrow Y_i Y_j$
    $(L, H) \leftarrow \big(\max(0, \alpha_j^{old} + s\alpha_i^{old} - \frac{s+1}{2}C), \min(C, \alpha_j^{old} + s\alpha_i^{old} - \frac{s-1}{2}C)\big)$
    $E_k \leftarrow \big(\sum_{l=0}^{n} Y_l \alpha_l K(U_{l,*}, U_{k,*}) + W_0\big) - Y_k, \forall k \in \{i, j\}$
    $\alpha_j^{new} \leftarrow \alpha_j^{old} - \frac{Y_j(E_i - E_j)}{2K(U_{i,*},U_{j,*}) - K(U_{i,*},U_{i,*}) - K(U_{j,*},U_{j,*})}^{1}$
    $\alpha_j^{new,clipped} = \begin{cases} L, & \text{if } \alpha_j^{new} < L \\ \alpha_j^{new}, & \text{if } L < \alpha_j^{new} < H \\ H, & \text{if } \alpha_j^{new} > H \end{cases}$
    $\alpha_i^{new} \leftarrow \alpha_i^{old} + s(\alpha_j^{new,clipped} - \alpha_j^{old})$
    $b_i \leftarrow E_i + y_i(\alpha_i^{new} - \alpha_i^{old})K(U_{i,*}, U_{i,*}) + Y_2(\alpha_j^{new,clipped} - \alpha_j^{old})K(U_{i,*}, U_{j,*}) + W_0$
    $b_j \leftarrow E_j + y_i(\alpha_i^{new} - \alpha_i^{old})K(U_{i,*}, U_{i,*}) + Y_2(\alpha_j^{new,clipped} - \alpha_j^{old})K(U_{i,*}, U_{j,*}) + W_0$
    $\mathbf{W_0} \leftarrow \frac{\mathbf{b_i + b_j}}{\mathbf{2}}$, $(\alpha_\mathbf{j}, \alpha_\mathbf{i}) \leftarrow \left(\alpha_\mathbf{j}^{\mathbf{new,clipped}}, \alpha_\mathbf{i}^{\mathbf{new}}\right)$
    return $\alpha, W_0$
```

$$
\begin{aligned}
\text{Let } \hat{Y}_i &= \operatorname{sign}\left(\sum_{j=1}^{n} \alpha_j Y_j \, K([U_{i,*}, B_{U_i}], [U_{t,*}, B_{U_l}]) + W_0\right) \\
\alpha_i = 0 &\rightarrow Y_i \hat{Y}_i \geq 1 \\
0 < \alpha_i < C &\rightarrow Y_i \hat{Y}_i = 1 \\
\alpha_i = C &\rightarrow Y_i \hat{Y}_i \leq 1
\end{aligned} \tag{40}
$$

Therefore the learning algorithm will update all the pairs of α-s in each iteration. The SMO-like update of each pair of alphas is shown in the Algorithm 2, with more details in [27]. Please note that the algorithm also updates the hyperplane intercept W_0, which is used for classification of latent instances.

Having defined the gradients for updating latent matrices U, V with respect to the optimization loss and also the update rules for α-s, we can derive a final learning algorithm based on coordinate gradient descent. Algorithm 3 shows the learning algorithm in full terms. The updates of each cell of U, V, B_U, B_V, as response to the reconstruction loss F_R and the classification accuracy loss F_{CA}, are conducted in the negative direction of the gradients scaled by hyperparameter learning rates η_R, η_{CA}. The convergence is guaranteed by selecting small values for the learning rates. The stopping criteria is when the final loss from Equation 32 reaches an optimum, meaning it doesn't get further minimized.

The Convergence of Learning is guaranteed because both steps (i) the learning of the latent data U, V for the reconstruction loss, (ii) updates of U for the classification loss and the Lagrangian multipliers, are both steps of the Expec-

Algorithm 3 Learning Algorithm: Nonlinearly Supervised Dim. Red.

```
Input: Dataset matrix $X \in \mathbb{R}^{(n+n')\times m}$, Labels vector $Y \in \mathbb{R}^n$, Parameters: { Box
  constraint $C$, Optimization switch $\beta$, Latent dimensions $d$, Learning rates $\eta_R, \eta_{CA}$,
  Regularizations $\lambda_U, \lambda_V$, Kernel degree $p$ }
Output: $U, V, B_U, B_V, \alpha, W_0$
  Initialize $U \in \mathbb{R}^{(n+n')\times d}$, $V \in \mathbb{R}^{d\times m}$, $B_U \in \mathbb{R}^{(n+n')\times 1}$, $B_V \in \mathbb{R}^{1\times m}$ randomly
  Initialize $\alpha \leftarrow \{0\}^n$, $W_0 \leftarrow 0$
  while $F$ not reached an optimum do
    for $\forall (i,j,k) \in (\{1...(n+n')\}, \{1...m\}, \{1...d\})$ in random order do
      $U_{i,k} \leftarrow U_{i,k} - \eta_R \frac{\partial F_R(X,U,V)_{i,j}}{\partial U_{i,k}}$
      $V_{k,j} \leftarrow V_{k,j} - \eta_R \frac{\partial F_R(X,U,V)_{i,j}}{\partial V_{k,j}}$
      $B_{U_i} \leftarrow B_{U_i} - \eta_R \frac{\partial F_R(X,U,V)_{i,j}}{\partial B_{U_i}}$
      $B_{V_j} \leftarrow B_{V_j} - \eta_R \frac{\partial F_R(X,U,V)_{i,j}}{\partial B_{V_j}}$
    end for
    for $\forall (i,l,k) \in (\{1...n\}, \{1...n\}, \{1...d\})$ in random order do
      $U_{i,k} \leftarrow U_{i,k} - \eta_{CA} \frac{\partial F_{CA}(Y,U,\alpha)_{i,l}}{\partial U_{i,k}}$
      $U_{l,k} \leftarrow U_{l,k} - \eta_{CA} \frac{\partial F_{CA}(Y,U,\alpha)_{i,l}}{\partial U_{l,k}}$
      $B_{U_i} \leftarrow B_{U_i} - \eta_{CA} \frac{\partial F_{CA}(Y,U,\alpha)_{i,l}}{\partial B_{U_i}}$
      $B_{U_l} \leftarrow B_{U_l} - \eta_{CA} \frac{\partial F_{CA}(Y,U,\alpha)_{i,l}}{\partial B_{U_l}}$
    end for
    for $\forall i \in \{1 \ldots n\}$ do
      if $\alpha_i$ violates KKT of Equation 40 then
        for $\forall j \in \{1 \ldots n\}$ in random order do
          $(\alpha, W_0) \leftarrow$ UpdateAlphaPair$(i,j)$, from Algorithm 2
        end for
      end if
    end for
  end while
  return $U, V, B_U, B_V, \alpha, W_0$
```

tation Maximization algorithm. Figure 3 specifically illustrate the convergence of our algorithm for the Ionosphere dataset.

The reconstruction loss (F_R) and the classification accuracy loss (F_{CA}) converge smoothly as depicted in the left plot of Figure 3. The learning algorithm updates the latent data with respect to the objective function of Equation 1, therefore the decrease of the values of loss terms (shown in sub-figure *a)*) is an indication that our algorithm converge as expected. On the right, sub-figure *b)* demonstrates the consequence that a minimization of the classification loss has towards decreasing the error rate on both training and testing data. There is no significant gap between the train and test errors, which indicates that the hyper-plane (α) learned over training instances generalizes accurately on the unobserved test data.

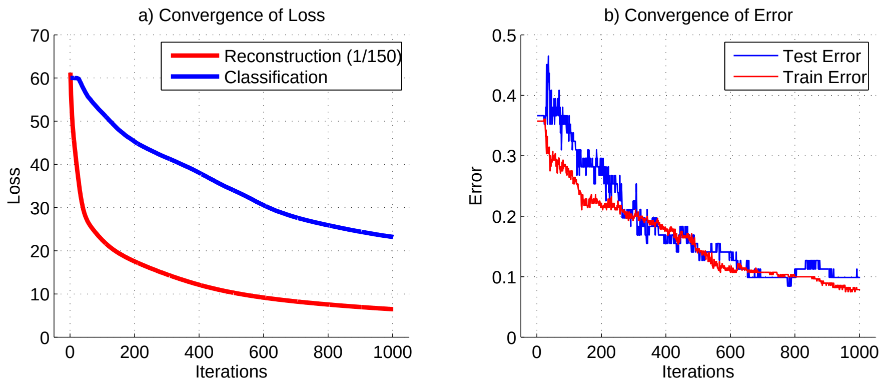

Fig. 3: Convergence of the loss terms and errors belonging to the Ionosphere dataset; Parameters $\beta = 0.7$, $C = 1, p = 1$, $\lambda_U = \lambda_V = 10^{-5}$, $\eta_R = 0.001, \eta_{CA} = 0.0001$ and $d = 25$.

The Algorithmic Complexity of our method both in terms of run-time and space depends on the size of the data. Concretely, the storage requirements are upper bounded to the size of the predictors, since for $d < m$ the storage of Y, U, V and α are all less than X. Therefore the space complexity of the method is $O((n+n') \times m)$. The running time depends on the number of iterations of Algorithm 3. If we denote the iterations as I, then the number of updates is proportional to $O(I \times (n+n') \times m \times d)$, since for every cell $X_{i,j}$ we update all the k-many row cells $U_{i,k}$ and column cells $V_{j,k}$. The updates of the classification accuracy loss term α-s are inferior in number ($O(I \times (n+n') \times m)$) and do not influence the upper bounding algorithmic complexity with respect to the reconstruction loss. Note that l, the number of target categories, is a small constant equals to two for binary problems. It is not possible to forecast exactly the number of iterations that a dataset will require until convergence, since it depends on the slope of the loss function's surface $(F_R + F_{CA})$ and the learning rate parameter. Large learning rates converge faster but increase both the risk of divergence and missing narrow local optima. On the other hand, small learning rates require more iterations to minimize the loss.

4 Experimental Results

4.1 Experimental Setup

In order to compare the classification accuracy of our method Nonlinearly Supervised Dimensionality Reduction (**NSDR**), we implemented and compared against three baselines:

- **PCA-SVMs:** Matching against the standard PCA dimensionality reduction and then SVMs classification will demonstrate the advantage of supervised decomposition against unsupervised decomposition (PCA).

Table 1: **Hyper-parameter Search Results**

DATASET	**LSDR**	**NSDR**	**PCA-SVMs**	**SVMs**
breast_canc.	$\lambda_U = 10^{-4}; \lambda_V = 10^{-5};$ $\eta_R = 10^{-3}; \eta_{CA} = 10^{-4}; d = 6;$ $\beta = 0.1, \lambda_W = 1$	$\lambda_U = 10^{-2}; \lambda_V = 1;$ $\eta_R = 10^{-3}; \eta_{CA} = 10^{-4}; d = 9;$ $\beta = 0.9; C = 10; p = 2$	$var = 1;$ $C = 10; p = 2$	$C = 10$ $p = 2$
ionosphere	$\lambda_U = 10^{-6}; \lambda_V = 10^{-5};$ $\eta_R = 10^{-3}; \eta_{CA} = 10^{-4}; d = 25;$ $\beta = 0.9, \lambda_W = 1$	$\lambda_U = 10^{-6}; \lambda_V = 10^{-6};$ $\eta_R = 10^{-3}; \eta_{CA} = 10^{-4}; d = 9;$ $\beta = 0.9; C = 10; p = 2$	$var = 1;$ $C = 1; p = 3$	$C = 0.1$ $p = 2$
pi-diabetes	$\lambda_U = 10^{-6}; \lambda_V = 10^{-3};$ $\eta_R = 10^{-3}; \eta_{CA} = 10^{-4}; d = 6;$ $\beta = 0.9, \lambda_W = 0.1$	$\lambda_U = 10^{-4}; \lambda_V = 1;$ $\eta_R = 10^{-3}; \eta_{CA} = 10^{-4}; d = 6;$ $\beta = 0.1; C = 0.1; p = 3$	$var = 1$ $C = 1; p = 3$	$C = 10$ $p = 3$
sonar	$\lambda_U = 10^{-6}; \lambda_V = 1;$ $\eta_R = 10^{-3}; \eta_{CA} = 10^{-4}; d = 45;$ $\beta = 0.5, \lambda_W = 10$	$\lambda_U = 10^{-2}; \lambda_V = 1;$ $\eta_R = 10^{-3}; \eta_{CA} = 10^{-4}; d = 60;$ $\beta = 0.1; C = 0.1; p = 2$	$var = 0.7$ $C = 10; p = 2$	$C = 0.1$ $p = 3$
spect	$\lambda_U = 10^{-2}; \lambda_V = 1;$ $\eta_R = 10^{-3}; \eta_{CA} = 10^{-4}; d = 16;$ $\beta = 0.5, \lambda_W = 0.1$	$\lambda_U = 10^{-2}; \lambda_V = 10^{-5};$ $\eta_R = 10^{-3}; \eta_{CA} = 10^{-4}; d = 22;$ $\beta = 0.5; C = 0.1; p = 3$	$var = 1$ $C = 1; p = 3$	$C = 0.1$ $p = 2$

- **SVMs:** Comparison against the default SVMs will provide insights on the advantages of dimensionality reduction.
- **LSDR:** The linearly supervised dimensionality reduction represents the state of the art in data projection. A comparison against this baseline clarifies the superiority position of our method [3, 4]. The smooth hinge loss is used due to its competitiveness and high prediction accuracy demonstrated together with SVMs.

The experiments were conducted using five folds cross validation, where the data was divided into five splits and each split was, in turn, the test and the other four the training data.

The hyper parameters of our method and the baselines was selected using a validation data split from the training data. The best grid-search combinations of hyper parameters that yielded the best accuracy was selected for being applied to the test split. The ranges of search for the LSDR and NSDR methods were $\lambda_U \in \{10^{-6}, 10^{-5}, \ldots, 10^0, 10^1\}$, $\lambda_V \in \{10^{-6}, 10^{-5}, \ldots, 10^0, 10^1\}$, $\eta_R \in \{10^{-4}, 10^{-3}\}, \eta_{CA} \in \{10^{-4}, 10^{-3}\}$, $d \in \{25\%, 50\%, 75\%, 100\%\}$ of m, $\beta \in \{0.1, 0.5, 0.9\}$, $C \in \{0.1, 1, 10\}, p \in \{1, 2, 3, 4\}$. For PCA-SVMs there is a variance parameter $var \in \{0.5, 0.7, 1.0\} \times 100\%$. The other SVMs parameters C, p for both PCA-SVMs and SVMs were searched in the same ranges as the ones reported for NSDR previously.

4.2 Results

In order to validate the proposed method we selected five popular binary datasets from the UCI repository, which cover a range of applications such as medicine (Breast Cancer, Pi-Diabetes and Spect), radar (Ionosphere) and undersea explorations (Sonar). The hyper-parameters of the models were fit in a **5-folds** cross-validation fashion among the aforementioned ranges. In order to promote experimental reproducibility, we present the exact values of parameters in Table 1 for our method and all the baselines.

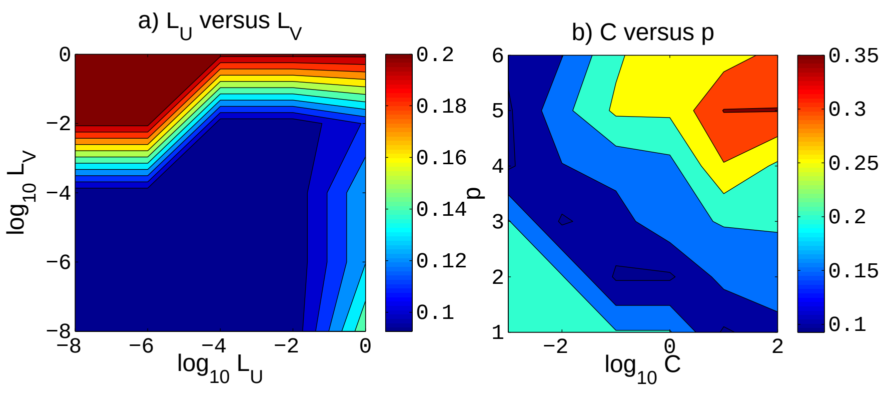

Fig. 4: Parameter Sensitivity Analysis on SPECT dataset; Parameters $\beta = 0.9$, $C = 0.1, p = 2$, $\lambda_U = 10^{-6}, \lambda_V = 10^{-5}$, $\eta_R = 0.001, \eta_{CA} = 0.0001$ and $d = 11$.

The Sensitivity of Parameters depends on the impact that a perturbation of the value of a parameter has over the error rate. The most sensitive parameters in a learning method are usually the regularization weights of the model complexity. In the case of NSDR, the learning rate, the number of iterations and the latent dimensions are less critical with respect to accuracy. The learning rate should be set small enough to avoid divergence and the iterations large enough to ensure convergence. In order to avoid under-fitting, the number of dimensions has to be set at a large value, e.g. 75% of m. Therefore, the most sensitive parameters are λ_U, λ_V for the reconstruction loss and C, p for the classification accuracy loss. Figure 4 illustrate the sensitivity relation among the complexity regularization parameters on the SPECT dataset. The left plot demonstrate the error rate heatmap (the smaller the less error) as a result of perturbing the values of λ_U versus λ_V. All the other parameters are kept constant at their optimal value (yielding smallest error) and are displayed in the figure caption. The ranges of the plot axis are displayed as the logarithm of the parameters values in order to have equidistant ticks. The plot on the right presents the sensitivity of the error rate with respect to the changes of the accuracy parameters C and p. As can be seen from both plots, the error fluctuates significantly in the case of C, p, which indicates that a practitioner should search for those parameters in a narrow grid of values. On the contrary, the search for λ_U, λ_V are less critical because there exists a large region of optimal values, which is expressed as a blue plateau.

We would like to point out that our non-linearly supervised dimensionality reduction (NSDR) is a generalization of the linearly supervised projection (LSDR). The linear case can be instantiated as a polynomial kernel of degree one. Since our method NSDR includes the functionality of LSDR, then every competitive result of LSDR is easily achieved by NSDR (with parameter $p = 1$). On the contrary, as our experiments show, a linear decomposition does not recover the inexpressive nature of its linear hyper-plane.

The accuracy results in terms of error ratios is presented in Table 2 with respect to five real-life datasets. The winning method per each dataset is shown in **bold**. As we can observe our proposed method outperforms the baselines in all the datasets.

Table 2: **Error Ratios on Real-Life Datasets**

DATASET	**NSDR**	**LSDR**	**PCA-SVMs**	**SVMs**
breast_cancer_w	**0.070 ± 0.018**	0.122 ± 0.013	0.082 ± 0.019	0.073 ± 0.021
ionosphere	**0.066 ± 0.008**	0.097 ± 0.016	0.091 ± 0.010	0.140 ± 0.018
pi-diabetes	**0.264 ± 0.023**	0.279 ± 0.016	0.280 ± 0.006	0.274 ± 0.030
sonar	**0.106 ± 0.041**	0.188 ± 0.042	0.226 ± 0.129	0.226 ± 0.056
spect	**0.138 ± 0.051**	0.142 ± 0.056	0.243 ± 0.103	0.206 ± 0.002

NSDR improves the classification on the *ionosphere* and *sonar* datasets with significant differences, while on the other datasets the gap to the second best is smaller. As can be deduced from the results, the linearly supervised decomposition is superior to the unsupervised decomposition (PCA-SVMs) in 3 out of 5 datasets. Furthermore the nonlinear supervision (NSDR) outperforms the linear method (LSDR) in all the datasets. The outcomes of the experiments validate the expectations of our paper and demonstrate the usefulness of non-linear supervision with respect to real-life data.

4.3 Run-time Disadvantage

While our proposed method (NSDR) achieves a better classification accuracy than the baseline, still it has a costly optimization procedure. Compared to faster classifiers like SVMs, our method has a joint factorization and classification loss term. The joint optimization requires significant time to compute, in particular because of the slower learning rates that are required to ensure a convergence. For instance, it takes only 0.415 seconds for the SVMs and 158.065 for NSDR to compute. Clearly, the run-time is a negative aspect of the paper with respect to SVMs. As a rule of thumb, we advice practitioners to use NSDR only if classification accuracy, not run-time, is the primary objective.

5 Conclusions

Throughout this study we presented a non-linearly supervised dimensionality reduction technique, which jointly combined a joint optimization on reconstruction and classification accuracy. Such an approach distances from traditional data mining that considered dimensionality reduction and classification as two disjoint, sequential processes. The supervised decomposition benefits from the knowledge on the target values of training instances, in order to both eliminate the noise present in the predictor values and also preserve the class segregation.

In our presented method, the reconstruction loss term is expressed as matrix factorization decomposition of latent matrices, while the classification accuracy as a dual form kernel maximum margin classifier. Consequently, the reduced dataset is learned via a coordinate descent algorithm which updates the reduced dimensionality dataset w.r.t to both loss terms simultaneously.

Existing state of the art methods in supervised decomposition incorporate linear classification terms in the objective function. In contrast, our method introduces a novel non-linear supervision of the dimensionality reduction process. We adopt a kernel based classification loss, which guides the low-rank data into being separated via a non-linear hyper-plane. A non-linear decomposition improves accuracy in cases where the original data is not linearly separable, because preserving the non-linear arrangement of instances does not deteriorate the reconstruction loss of predictor values. Since the linear supervision is a special instance of our methods for a polynomial kernel of degree, then our method offers a super-set of expressiveness.

Empirical results over five real-life datasets show that the proposed method outperforms the selected baselines in the majority of the datasets. Significant improvement is present against unsupervised techniques, which indicates the benefit of incorporating target value information into dimensionality reduction. In addition, experimental results validated the superiority of non-linearly guided supervision against the linearly supervised state of the art decomposition.

6 Acknowledgment

This study was funded by the Seventh Framework Programme (FP7) of the European Commission, through projects REDUCTION[2] and iTalk2Learn[3].

In addition, the authors express their gratitude to Lucas Rego Drumond (University of Hildesheim) for his assistance on formalizing the linearly supervised decomposition.

References

1. Samet, H.: Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2005)
2. Grabocka, J., Bedalli, E., Schmidt-Thieme, L.: Efficient classification of long time-series. In Markovski, S., Gusev, M., eds.: ICT Innovations 2012. Volume 207 of Advances in Intelligent Systems and Computing. Springer Berlin Heidelberg (2013) 47–57
3. Grabocka, J., Nanopoulos, A., Schmidt-Thieme, L.: Classification of sparse time series via supervised matrix factorization. In Hoffmann, J., Selman, B., eds.: AAAI, AAAI Press (2012)

[2] `www.reduction-project.eu`
[3] `www.italk2learn.eu`

4. Das Gupta, M., Xiao, J.: Non-negative matrix factorization as a feature selection tool for maximum margin classifiers. In: Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition. CVPR '11, Washington, DC, USA, IEEE Computer Society (2011) 2841–2848
5. Jolliffe, I.T.: Principal Component Analysis. Second edn. Springer (October 2002)
6. Wismüller, A., Verleysen, M., Aupetit, M., Lee, J.A.: Recent advances in nonlinear dimensionality reduction, manifold and topological learning. In: ESANN. (2010)
7. Hoffmann, H.: Kernel pca for novelty detection. Pattern Recogn. **40**(3) (March 2007) 863–874
8. Sun, J., Crowe, M., Fyfe, C.: Extending metric multidimensional scaling with bregman divergences. Pattern Recognition **44**(5) (2011) 1137 – 1154
9. Gorban, A.N., Zinovyev, A.Y.: Principal manifolds and graphs in practice: from molecular biology to dynamical systems. Int. J. Neural Syst. **20**(3) (2010) 219–232
10. Lee, J.A., Verleysen, M.: Nonlinear dimensionality reduction. Springer, New York; London (2007)
11. Gashler, M.S., Martinez, T.: Temporal nonlinear dimensionality reduction. In: Proceedings of the IEEE International Joint Conference on Neural Networks IJCNN'11. IEEE Press (2011) 1959–1966
12. Lawrence, N., Hyvrinen, A.: Probabilistic non-linear principal component analysis with gaussian process latent variable models. Journal of Machine Learning Research **6** (2005) 1783–1816
13. Lawrence, N.: Gaussian process latent variable models for visualisation of high dimensional data. In: In NIPS. (2003) 2004
14. Singh, A.P., Gordon, G.J.: A unified view of matrix factorization models. In: ECML/PKDD (2). (2008) 358–373
15. Koren, Y., Bell, R.M., Volinsky, C.: Matrix factorization techniques for recommender systems. IEEE Computer **42**(8) (2009) 30–37
16. Rendle, S., Schmidt-Thieme, L.: Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In Pu, P., Bridge, D.G., Mobasher, B., Ricci, F., eds.: RecSys, ACM (2008) 251–258
17. Cai, D., He, X., Han, J., Huang, T.S.: Graph regularized nonnegative matrix factorization for data representation. IEEE Trans. Pattern Anal. Mach. Intell. **33**(8) (2011) 1548–1560
18. Giannakopoulos, T., Petridis, S.: Fisher linear semi-discriminant analysis for speaker diarization. IEEE Transactions on Audio, Speech & Language Processing **20**(7) (2012) 1913–1922
19. Menon, A.K., Elkan, C.: Predicting labels for dyadic data. Data Min. Knowl. Discov. **21**(2) (2010) 327–343
20. Rish, I., Grabarnik, G., Cecchi, G., Pereira, F., Gordon, G.J.: Closed-form supervised dimensionality reduction with generalized linear models. In: ICML '08: Proceedings of the 25th international conference on Machine learning, New York, NY, USA, ACM (2008) 832–839
21. Rennie, J.D.M.: Loss functions for preference levels: Regression with discrete ordered labels. In: Proceedings of the IJCAI Multidisciplinary Workshop on Advances in Preference Handling. (2005) 180–186
22. Fukumizu, K., Bach, F.R., Jordan, M.I.: Dimensionality reduction for supervised learning with reproducing kernel hilbert spaces. Journal of Machine Learning Research **5** (2004) 73–99
23. Salakhutdinov, R., Hinton, G.: Learning a nonlinear embedding by preserving class neighbourhood structure. In: Proceedings of the International Conference on Artificial Intelligence and Statistics. Volume 11. (2007)

24. Zhang, D., hua Zhou Songcan Chen, Z.: Semi-supervised dimensionality reduction. In: In: Proceedings of the 7th SIAM International Conference on Data Mining. (2007) 11–393
25. Urtasun, R., Darrell, T.: Discriminative gaussian process latent variable models for classification. In: In International Conference in Machine Learning. (2007)
26. Scholkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA, USA (2001)
27. Platt, J.: Fast training of support vector machines using sequential minimal optimization. In Schoelkopf, B., Burges, C., Smola, A., eds.: Advances in Kernel Methods - Support Vector Learning. MIT Press (1998)