Super-sparse Learning in Similarity Spaces

Ambra Demontis *Student Member, IEEE,* Marco Melis *Student Member, IEEE,*
Battista Biggio *Member, IEEE,* Giorgio Fumera *Member, IEEE,* and Fabio Roli *Fellow, IEEE*
Department of Electrical and Electronic Engineering,
University of Cagliari, Piazza d'Armi, 09123 Cagliari, Italy

Abstract—**In several applications, input samples are more naturally represented in terms of similarities between each other, rather than in terms of feature vectors. In these settings, machine-learning algorithms can become very computationally demanding, as they may require matching the test samples against a very large set of reference prototypes. To mitigate this issue, different approaches have been developed to reduce the number of required reference prototypes. Current reduction approaches select a small subset of representative prototypes in the space induced by the similarity measure, and then separately train the classification function on the reduced subset. However, decoupling these two steps may not allow reducing the number of prototypes effectively without compromising accuracy. We overcome this limitation by jointly learning the classification function along with an optimal set of virtual prototypes, whose number can be either fixed a priori or optimized according to application-specific criteria. Creating a super-sparse set of virtual prototypes provides much sparser solutions, drastically reducing complexity at test time, at the expense of a slightly increased complexity during training. A much smaller set of prototypes also results in easier-to-interpret decisions. We empirically show that our approach can reduce up to ten times the complexity of Support Vector Machines, LASSO and ridge regression at test time, without almost affecting their classification accuracy.**

I. Introduction

In a growing number of applications, including computer vision, biometrics, text categorization and information retrieval, samples are often represented more naturally in terms of similarities between each other, rather than in an explicit feature vector space [1], [2]. Traditional machine-learning algorithms can still be used to learn over similarity-based representations; *e.g.*, linear classification algorithms like Support Vector Machines (SVMs) [3], [4] can be trained in the space implicitly induced by the similarity measure (*i.e.*, the kernel function) to learn nonlinear functions in input space. However, the main drawback of similarity-based techniques is their high computational complexity at test time, since computing their classification function often requires matching the input sample against a large set of reference prototypes, and evaluating such similarity measures is usually computationally demanding. Even SVMs, that induce sparsity in the number of required prototypes (the so-called support vectors, SVs) may not provide solutions that are sparse enough, as the number of prototypes (*i.e.*, SVs) grows linearly with respect to the number of training samples [5], [6]. To reduce the number of reference prototypes, several state-of-the-art approaches select them from the training data, and then separately train the classification function using the reduced set of prototypes. However, decoupling these two steps may not effectively reduce the number of prototypes, without significantly affecting classification accuracy [2], [7], [8].

In this work, we first discuss the relationship between current prototype-selection methods and our recently-proposed approach for learning super-sparse machines on similarity-based representations [9]–[11]. We then show that our approach can successfully tackle this issue by jointly learning the classification function along with an optimal set of virtual prototypes. The number of prototypes required by our approach can be either fixed a priori, or optimized through a carefully-designed, incremental cross-validation (CV) procedure. Creating a super-sparse set of virtual prototypes allows us to learn much sparser solutions, drastically reducing the computational complexity at test time, at the expense of a slightly increased computational complexity during training. A much smaller set of reference prototypes also provides decisions that are easier to interpret. We validate our approach on two application examples, including biometric face verification and age estimation from faces. Our approach does not almost affect the generalization capability of SVMs, LASSO [12], and ridge regression [13], while being capable of reducing their complexity of more than ten times, overcoming the performance of other reduction methods. We conclude this paper with a discussion of future research directions.

II. Learning in Similarity Spaces

Two main approaches can be used to learn classification and regression functions in similarity spaces, *i.e.*, functions that depend only on similarities between samples, and not on their features [1], [2]. The first one consists of computing an explicit representation of samples in a vector space that aims to preserve the similarity values between the original samples. A well-known example is related to kernel functions (*i.e.*, positive semi-definite similarity functions), as one may exploit the eigenvalue or Cholesky decomposition of the kernel matrix to represent samples in an explicit vector space, called the empirical kernel mapping [1], [2], [7], [8], [19]. Their underlying idea is to decompose the kernel (similarity) matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ computed on n training samples as

A. Demontis: e-mail ambra.demontis@diee.unica.it, phone +39 070 675 5776

M. Melis: e-mail marco.melis@diee.unica.it, phone +39 070 675 5776

B. Biggio (corresponding author): e-mail battista.biggio@diee.unica.it, phone +39 070 675 5776

G. Fumera: e-mail fumera@diee.unica.it, phone +39 070 675 5754

F. Roli: e-mail roli@diee.unica.it, phone +39 070 675 5779, fax (shared) +39 070 675 5782

$\mathbf{K} = \mathbf{X}^\top \mathbf{X}$, where $\mathbf{X} \in \mathbb{R}^{n \times n}$ represents the training samples in the empirical kernel space. Then, at test time, never-before-seen samples should be mapped onto the same space, to be classified in a consistent manner, and this often requires matching them against all training samples [1], [2]. Clearly, the aforementioned decompositions are only possible for positive semi-definite similarities (*i.e.*, kernel matrices). For indefinite similarities different techniques can be exploited to account for the negative eigenvalues of $\mathbf{K}$, and potentially exploit the same decompositions, or adapt the corresponding classification functions [1], [2], [19]. They include: spectrum clip, in which the negative eigenvalues are set to zero; spectrum flip, in which their absolute values are used; spectrum shift, in which they are increased by a quantity equal to the minimum eigenvalue, such that the smallest one becomes exactly zero; and spectrum square, in which the eigenvalues are squared [1]. Notably, spectrum flip amounts to mapping the input data onto a pseudo-Euclidean space. This space consists of two Euclidean spaces: one for which the inner product is positive definite, and one for which it is negative definite. This enables computing quadratic forms (*e.g.*, vector norms and inner products) as the difference of two positive-definite norms. Pseudo-Euclidean spaces are a particular case of finite-dimensional Kreĭn spaces consisting of two real subspaces. In general, indefinite kernels and similarities allow a consistent (*e.g.*, infinite-dimensional) representation in a Kreĭn space [1], [2], [19].

The second approach consists of learning classifiers directly in the similarity space, *i.e.*, exploiting similarities computed against a set of reference prototypes as feature values. This is equivalent to the former approach, if the spectrum-square technique is used [1], [2] (see the example in Fig. 1). Worth remarking, some learning algorithms have been explicitly modified to deal with similarities, instead of adapting the similarity-based representation to existing learning algorithms. Examples can be found in the area of relational fuzzy clustering [20], and relational lexical variant generation [21], [22].

Feature- and similarity-based representations may be thought as two facets of the same coin: modifying the similarity measure amounts to modifying the implicit feature space in which the linear decision function operates, and vice versa. This means that, to achieve good generalization capabilities, it is necessary to properly define this space on the basis of the given learning algorithm. When one is given a similarity-based representation, this space can be manipulated essentially in two ways, *i.e.*, by either modifying the similarity measure or the prototypes. Several approaches have been proposed to manipulate the similarity measure, including multiple kernel learning and similarity learning [14], [15]. They exploit a parametric similarity or distance measure, whose parameters are tuned by running a learning algorithm on the training data. In particular, in the case of multiple kernel learning, the goal is to learn the coefficients of a convex linear combination of a set of given kernels [14]. Conversely, only few works have addressed the problem of selecting the reference prototypes to reduce complexity of similarity-based classifiers [7], especially in the context of structured inputs like graphs and strings [16], [17]. To this end, it is also worth remarking that the Nyström approximation can be exploited to approximate $n \times n$ similarity matrices based on a subset of $m \ll n$ randomly-chosen prototypes, reducing the complexity of computing all pairwise similarities from $\mathcal{O}(n^2)$ to $\mathcal{O}(nm^2)$. This approximation also works for indefinite kernels and similarities [18].

The latter approach, based on learning classifiers directly in the similarity space, includes the solution we propose in this paper and it is of particular interest in applications where the similarity metric is: (i) defined and can not be modified, (ii) not given in analytical terms, and (iii) not necessarily positive semi-definite. For instance, in fingerprint recognition, the similarity measure is often defined a priori, as it encodes the knowledge of domain experts, and in most of the cases it is not positive semi-definite (it does not obey the triangle inequality). In addition, it is usually computed by a physical device, called matcher, and it is not even analytically defined. Modifying the similarity measure in these cases is not possible. The only way of manipulating the space induced by such a measure consists of modifying the prototypes. The main limit of the corresponding state-of-the-art approaches is however intrinsic to the fact that they separately select the prototypes and then learn the classification function [7]. The approach advocated in the next section, instead, is based on the idea of jointly optimizing the prototypes and the parameters of the classification function, to outperform existing reduction methods in similarity spaces. Furthermore, with respect to methods devoted to reduce the number of SVs in SVMs, our approach requires neither the similarity to be a positive semi-definite kernel, nor the learning algorithm to be an SVM [9]. It can be applied, in principle, to reduce the complexity of any similarity-based learning algorithm.

III. Super-Sparse Virtual Vector Machines

We present here our approach to learn super-sparse machines, inspired from [9]–[11]. The underlying idea is to reduce complexity of similarity-based learning algorithms by employing a very small set of virtual prototypes. The virtual prototypes obtained by our learning algorithm are not necessarily part of the training data, but are specifically created with the goal of retaining a very high generalization capability.

Let us denote with $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$ the training data, where $\boldsymbol{x}_i \in \mathcal{X}$ are the input samples and $y_i \in \mathcal{Y}$ are their labels. We consider here a vectorial representation of the input data $\boldsymbol{x}$, *i.e.*, we assume that $\mathcal{X}$ is a vector space. The output space $\mathcal{Y}$ depends on whether we are considering a regression or classification problem. For regression, we consider $\mathcal{Y} \subseteq \mathbb{R}$, whereas for two-class classification, we set $\mathcal{Y} = \{-1, +1\}$. The set of virtual prototypes is denoted with $\boldsymbol{z} = \{\boldsymbol{z}_j\}_{j=1}^m$, where $m \ll n$ to obtain much sparser solutions. The value of m can be either fixed a priori, depending on application-specific constraints (*e.g.*, specific requirements on storage and time complexity, as in match-on-board biometric verification systems), or it can be optimized through a well-crafted CV procedure, as defined at the end of this section. Note that the virtual prototypes $\boldsymbol{z}_j$ belong to the same input space $\mathcal{X}$ as the training samples. We finally denote the similarity function with $s : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$. It can be any symmetric similarity function (not necessarily positive semi-definite).

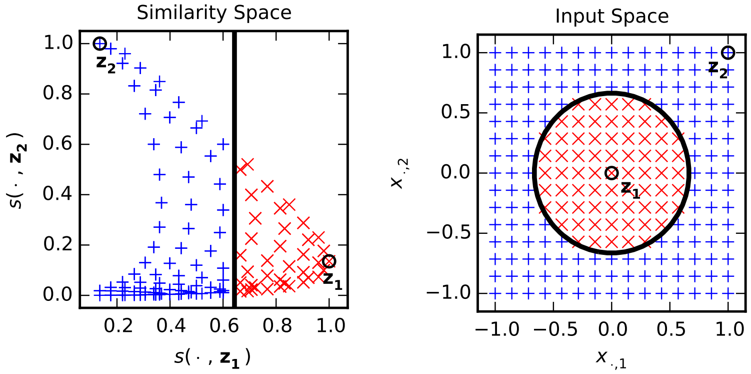

Fig. 1: *Left:* A linear SVM classifier trained in a two-dimensional similarity space, using similarities as features, $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ as the reference prototypes, and the RBF kernel $s(\boldsymbol{x},\boldsymbol{z}) = \exp\left(-\|\boldsymbol{x}-\boldsymbol{z}\|^2\right)$ as the similarity measure. Blue '+' and red '×' represent the training data, while the black solid line denotes the classification boundary. *Right:* The corresponding nonlinear decision function in the input space.

Our goal is to learn a discriminant function $g(\boldsymbol{x})$ consisting of a very sparse linear combination of similarities computed against the virtual prototypes:

$$g(\boldsymbol{x}) = \textstyle\sum_{j=1}^{m} \beta_j \phi_j(\boldsymbol{x}) + b = \sum_{j=1}^{m} \beta_j s(\boldsymbol{x}, \boldsymbol{z}_j) + b \ , \qquad (1)$$

where $\boldsymbol{\beta} = (\beta_1, \dots, \beta_m) \in \mathbb{R}^m$ is the vector of coefficients (one per virtual prototype), and b is the bias. Our approach is specifically designed to work in the similarity space, *i.e.*, our basis functions $\phi_j(\boldsymbol{x})$ are similarity functions $s(\boldsymbol{x}, \boldsymbol{z}_j)$. The reason is that we aim to optimize the virtual prototypes $\boldsymbol{z}_j$ without changing s, as in several applications such a function can not be changed; *e.g.*, in biometric verification, one is often given a matching function which is neither customizable nor even known analytically.

As in [8], [9], we consider a regression problem in which the goal is to minimize the distance between the target variables y and g on the training points, with respect to the parameters of g, *i.e.*, the coefficients $\boldsymbol{\beta}$, b and the set of virtual prototypes $\boldsymbol{z}$. We do not constrain the prototypes $\boldsymbol{z}$ to be in $\mathcal{D}$, but enable the creation of novel (virtual) prototypes. This allows our algorithm to achieve a better trade-off between accuracy and the number of required prototypes. The problem can be thus formulated as:

$$\min_{\boldsymbol{\beta},b,\boldsymbol{z}} \ \textstyle\sum_{i=1}^{n} u_i \left(g(\boldsymbol{x}_i) - y_i\right)^2 + \lambda \boldsymbol{\beta}^\top \boldsymbol{\beta} \ , \qquad (2)$$

where the scalars $u_1, \dots, u_n$ balance the contribution of each training sample $\boldsymbol{x}_i$ to the loss function (which may be useful when training classes are imbalanced), the quadratic regularizer $\boldsymbol{\beta}^\top \boldsymbol{\beta}$ controls overfitting, and λ is a regularization parameter. Note that sparsity is not induced here by a sparse regularizer on $\boldsymbol{\beta}$, but rather by setting m to a small value. This approach is clearly linear in the space induced by the similarity function, but not necessarily in the input space $\mathcal{X}$, *i.e.*, the similarity mapping can be used to induce nonlinearity as in kernel methods. The objective function in Problem (2) can be rewritten in matrix form:

$$\Omega\left(\boldsymbol{\beta}, b, \boldsymbol{z}\right) = \left(\boldsymbol{g}^\top \mathbf{U} \boldsymbol{g} - 2\boldsymbol{y}^\top \mathbf{U} \boldsymbol{g} + \boldsymbol{y}^\top \mathbf{U} \boldsymbol{y}\right) + \lambda \boldsymbol{\beta}^\top \boldsymbol{\beta} \ , \quad (3)$$

where the column vectors $\boldsymbol{g} = \mathbf{S}_{\boldsymbol{xz}} \boldsymbol{\beta} + b \in \mathbb{R}^n$ and $\boldsymbol{y} \in \mathbb{R}^n$ consist of the values of g and y for the training data, $\mathbf{U} \in \mathbb{R}^{n \times n}$ is a diagonal matrix such that $\operatorname{diag}(\mathbf{U}) = (u_1, \dots, u_n)$, and $\mathbf{S}_{\boldsymbol{xz}} \in \mathbb{R}^{n \times m}$ is the similarity matrix computed between $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ and the prototypes $\boldsymbol{z}$.

The objective function in Eq. (3) can be iteratively minimized by modifying $\boldsymbol{\beta}$, b and $\boldsymbol{z}$. First, we randomly initialize the prototypes $\{\boldsymbol{z}_j^{(0)}\}_{j=1}^m$ with m training samples from $\mathcal{D}$, and then iteratively repeat the two steps described below.

(1) **$\boldsymbol{\beta}$-step.** The optimal coefficients $\boldsymbol{\beta}$ are computed while keeping the prototypes $\boldsymbol{z}$ fixed. This amounts to solving a standard ridge regression problem, whose analytical solution is given by deriving Eq. (3) with respect to $\boldsymbol{\beta}$ and b (with $\boldsymbol{z}$ constant), and then setting the corresponding gradients to zero:

$$\underbrace{\begin{bmatrix} \mathbf{S}_{\boldsymbol{xz}}^\top \mathbf{U} \mathbf{S}_{\boldsymbol{xz}} + \lambda \mathbb{I} & \mathbf{S}_{\boldsymbol{xz}}^\top \mathbf{U} \mathbf{1} \\ \mathbf{1}^\top \mathbf{U} \mathbf{S}_{\boldsymbol{xz}} & \mathbf{1}^\top \mathbf{U} \mathbf{1} \end{bmatrix}}_{\mathbf{M}} \begin{bmatrix} \boldsymbol{\beta} \\ b \end{bmatrix} = \begin{bmatrix} \mathbf{S}_{\boldsymbol{xz}}^\top \\ \mathbf{1}^\top \end{bmatrix} \mathbf{U} \boldsymbol{y} \ , \qquad (4)$$

where $\mathbb{I} \in \mathbb{R}^{m \times m}$ denotes the identity matrix. Note that the system given by Eq. (4) can be iteratively solved without necessarily inverting $\mathbf{M}$, *e.g.*, using stochastic gradient descent [23].

(2) **$\boldsymbol{z}$-step.** If the similarity function is differentiable, Eq. (3) can be minimized through gradient descent (as no closed-form solution exists for this problem). Deriving with respect to a given $\boldsymbol{z}_j$, we obtain:

$$\frac{\partial \Omega}{\partial \boldsymbol{z}_j} = 2\left(\boldsymbol{g} - \boldsymbol{y}\right)^\top \mathbf{U} \left(\beta_j \frac{\partial \mathbf{S}_{\boldsymbol{x}\boldsymbol{z}_j}}{\partial \boldsymbol{z}_j} + \mathbf{S}_{\boldsymbol{xz}} \frac{\partial \boldsymbol{\beta}}{\partial \boldsymbol{z}_j} + \mathbf{1} \frac{\partial b}{\partial \boldsymbol{z}_j}\right) + 2\lambda \boldsymbol{\beta}^\top \frac{\partial \boldsymbol{\beta}}{\partial \boldsymbol{z}_j} \ , \qquad (5)$$

where $\mathbf{S}_{\boldsymbol{x}\boldsymbol{z}_j}$ is the j^{th} column of $\mathbf{S}_{\boldsymbol{xz}}$. Note that all the derivatives computed with respect to $\boldsymbol{z}_j$ here are vectors or matrices with the same number of columns as the dimensionality of $\boldsymbol{z}_j$. To compute $\frac{\partial \boldsymbol{\beta}}{\partial \boldsymbol{z}_j}$ and $\frac{\partial b}{\partial \boldsymbol{z}_j}$, as required by Eq. (5), we derive Eq. (4) with respect to $\boldsymbol{z}_j$ and solve for the required quantities:

$$\begin{bmatrix} \frac{\partial \boldsymbol{\beta}}{\partial \boldsymbol{z}_j} \\ \frac{\partial b}{\partial \boldsymbol{z}_j} \end{bmatrix} = -\mathbf{M}^{-1} \left(\beta_j \begin{bmatrix} \mathbf{S}_{\boldsymbol{xz}}^\top \\ \mathbf{1}^\top \end{bmatrix} + \begin{bmatrix} \mathbf{V}^\top \\ \mathbf{0}^\top \end{bmatrix}\right) \mathbf{U} \frac{\partial \mathbf{S}_{\boldsymbol{x}\boldsymbol{z}_j}}{\partial \boldsymbol{z}_j} \ , \qquad (6)$$

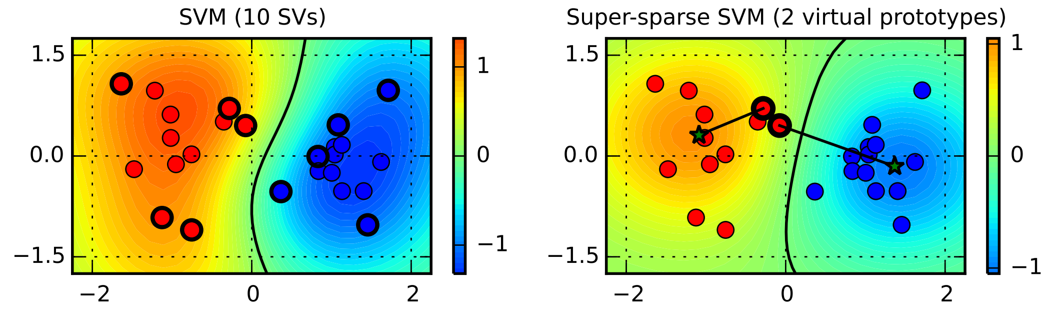

Fig. 2: A two-dimensional example showing the values of the discriminant function (in colors) of an SVM with the RBF kernel (*left*) trained on 25 samples (red and blue points in the left plot), and of $g(\boldsymbol{x})$ for our super-sparse learning algorithm (*right*). Our approach almost exactly replicates the SVM's decision function (solid black line) using only 2 virtual prototypes instead of 10 SVs (highlighted with black circles in the left plot). The initial and final positions of the two virtual prototypes optimized by our approach are also shown in the right plot, respectively as black circles and green stars.

where $\mathbf{V} \in \mathbb{R}^{n \times m}$ is a matrix that consists of all zeros except for the j^{th} column, which is equal to $(\boldsymbol{g} - \boldsymbol{y})$, and $\mathbf{0}, \mathbf{1} \in \mathbb{R}^n$ are column vectors of respectively n zeros and n ones.

The complete algorithm is given as Algorithm 1. Note that we invert the $\boldsymbol{z}$- and $\boldsymbol{\beta}$-step detailed above for the sake of compactness. In fact, the coefficients $\boldsymbol{\beta}$ and b should always be updated after changing the prototypes. Our method can also be used to reduce the number of prototypes used by kernel-based or prototype-based classifiers, like the SVM, by setting the target variables y to the values of the discriminant function of the target classifier for each training point. An example is reported in Fig. 2.[1] Worth remarking, the virtual prototypes found by our algorithm are quite different from the SVs found by the SVM. In fact, the latter are close to the boundary of the discriminant function, whereas our prototypes are found approximately at the centers of small clusters of training points.

Gradient of $s(\boldsymbol{x}_i, \boldsymbol{z}_j)$. In Eq. (6), the computation of the derivative of $s(\boldsymbol{x}_1, \boldsymbol{z}_j)$, $\ldots$, $s(\boldsymbol{x}_n, \boldsymbol{z}_j)$ with respect to the corresponding $\boldsymbol{z}_j$, depends on the given similarity measure s. If s has an analytical representation, like in the case of kernels, the derivative can be easily computed; *e.g.*, for the RBF kernel, $s(\boldsymbol{x}_i, \boldsymbol{z}_j) = \exp(-\gamma ||\boldsymbol{x}_i - \boldsymbol{z}_j||^2)$, and $\frac{\partial s(\boldsymbol{x}_i, \boldsymbol{z}_j)}{\partial \boldsymbol{z}_j} = 2\gamma \exp(-\gamma ||\boldsymbol{x}_i - \boldsymbol{z}_j||^2)(\boldsymbol{x}_i - \boldsymbol{z}_j)$. Otherwise, the gradient can be only approximated numerically, by querying $s(\cdot, \boldsymbol{z}_j)$ in a neighborhood of $\boldsymbol{z}_j$. This is computationally costly, especially if $\boldsymbol{z}_j$ is high dimensional. In the case of images, we have found that the similarity tends to increase while shifting $\boldsymbol{z}_j$ towards $\boldsymbol{x}_i$, even if this shift is operated linearly in the space of the pixel values (*e.g.*, by computing a convex combination of the two images) [9]–[11]. This amounts to approximating the gradient as

$$\frac{\partial s(\boldsymbol{x}_i, \boldsymbol{z}_j)}{\partial \boldsymbol{z}_j} = s(\boldsymbol{x}_i, \boldsymbol{z}_j)\,(\boldsymbol{x}_i - \boldsymbol{z}_j) \quad . \tag{7}$$

Although using this heuristic to approximate the gradient of s may affect, in general, the convergence of our algorithm, in the next section we show that it works quite well even when the similarity function is not analytically given (*i.e.*, for a graph-based face matching algorithm). Clearly, different heuristics may be considered, if the proposed one turns out to be not suited to the task at hand.

Algorithm 1 Super-sparse Learning Machine

```
Input: The training data $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$; the similarity function $s(\cdot,\cdot)$; the regularization parameter $\lambda$; the initial coefficients $(\boldsymbol{\beta}^{(0)}, b^{(0)})$ and vectors $\{\boldsymbol{z}_j^{(0)}\}_{j=1}^m$; the gradient step size $\eta$; a small number $\varepsilon$.
Output: The learned coefficients $(\boldsymbol{\beta}, b)$ and vectors $\{\boldsymbol{z}_j\}_{j=1}^m$.
Set the iteration count $t \leftarrow 0$.
repeat
    Set $j \leftarrow \mathrm{mod}(t, m) + 1$ to index a virtual prototype.
    Increase the iteration count $t \leftarrow t + 1$
    z-step. Set $\boldsymbol{z}_j^{(t)} \leftarrow \boldsymbol{z}_j^{(t-1)} - \eta \frac{\partial \Omega}{\partial \boldsymbol{z}_j^{(t-1)}} - \frac{1}{t^2} \frac{\partial \mathbf{S}_{\boldsymbol{zz}_j}}{\partial \boldsymbol{z}_j^{(t-1)}}$ (Eq. 5 with penalty for close prototypes).
    Project $\boldsymbol{z}_j^{(t)}$ onto the feasible domain, if any (e.g., a box constraint).
    Set $\boldsymbol{z}_i^{(t)} = \boldsymbol{z}_i^{(t-1)}$, $\forall i \neq j$.
    β-step. Compute $(\boldsymbol{\beta}^{(t)}, b^{(t)})$ using $\boldsymbol{z}_1^{(t)}, \ldots, \boldsymbol{z}_m^{(t)}$ (Eq. 4).
until $\left| \Omega(\boldsymbol{\beta}^{(t)}, b^{(t)}, \boldsymbol{z}^{(t)}) - \Omega(\boldsymbol{\beta}^{(t-1)}, b^{(t-1)}, \boldsymbol{z}^{(t-1)}) \right| < \varepsilon$
return: $\boldsymbol{\beta} = \boldsymbol{\beta}^{(t)}$, $b = b^{(t)}$ and $\boldsymbol{z} = \boldsymbol{z}^{(t)}$.
```

Prototype Initialization. Our approach might suffer from the intrinsic nature of the non-convex optimization problem faced in the $\boldsymbol{z}$-step, *i.e.*, when optimizing the virtual prototypes. In fact, due to the presence of multiple local minima (in which some prototypes may be too close to each other), our algorithm turns out to be quite sensible to the initialization of the virtual prototypes [9]–[11]. To overcome this limitation, we propose the following strategy. Instead of running the algorithm multiple times with different initializations (which

[1] Exploiting the values of a classifier's discriminant function as the target variables y in our approach usually works better than using the true classification labels y directly. The reason is that our approach uses the ℓ_2 loss, which is best suited to regression tasks. For classification, one should indeed exploit a loss function tailored to classification approaches, *e.g.*, the hinge loss.

would increase the overall computational complexity of our approach), we modify the gradient defined in Eq. (5) to account for a penalty term. It aims to reduce similarities between virtual prototypes, avoiding them to converge towards the same points. The penalty that we add to the gradient of Eq. (5) is simply $\frac{\partial \mathbf{S}_{\boldsymbol{z}\boldsymbol{z}_j}}{\partial \boldsymbol{z}_j}$. Since this term achieves large values for prototypes that are closer to $\boldsymbol{z}_j$, it is clear that $\boldsymbol{z}_j$ will be shifted away from them during optimization. We further multiply the penalty term by a decaying coefficient (*e.g.*, t^{-2}, being t the iteration count) to distance the prototypes sufficiently during the first iterations of the algorithm, without affecting convergence of our algorithm.

Selecting the number of virtual prototypes. Another important issue in our method regards the selection of m. As already discussed, m can be directly defined from specific constraints; otherwise, when some some degree of flexibility is allowed, it is possible to tune m by minimizing the number of prototypes without significantly compromising accuracy. To this end, we define an objective function characterizing this trade-off as

$$\mathcal{L}(m) = \ell(m) + \rho \cdot m \,, \tag{8}$$

and set $m = \arg\min_{m'} \mathcal{L}(m')$. In the above objective, ℓ is a loss (error) measure to be evaluated on a validation set, and ρ is a trade-off parameter. For higher ρ values, fewer prototypes are selected, at the expense of higher error rates.

The value of m that minimizes Eq. (8) can be efficiently selected through an incremental CV procedure which uses a grid search on some pre-defined values $m \in \{m_j\}_{j=1}^{K}$, as described in what follows. Let us assume that $m_1 > \ldots > m_K$. We first learn our approach using the largest number of prototypes m_1. Then, to learn the solution with m_2 prototypes, we remove the $m_1 - m_2$ prototypes assigned to the smallest $\boldsymbol{\beta}$ coefficients (in absolute value), and update the remaining m_2 coefficients and prototypes by re-running the learning algorithm from the current solution (warm start). We iterate until the most compact solution with m_K prototypes is learned, and then select the value of m that minimizes Eq. (8). An example is given in Fig. 3.

Computational Complexity. As previously discussed, our learning algorithm consists of two steps. The $\boldsymbol{\beta}$-step is computationally lightweight, as it only requires solving a linear system involving $m+1$ variables. Furthermore, this system has to be solved from scratch only at the first iteration, while for subsequent iterations one can exploit the previous solution as a warm start, and use an iterative algorithm to converge to the solution very quickly. The most computationally-demanding part is instead the computation of $\mathbf{S}_{\boldsymbol{x}\boldsymbol{z}_j}$ during the $\boldsymbol{z}$-step, which has complexity of $\mathcal{O}(n)$, as we optimize one prototype at a time. Clearly this has to be repeated at each iteration. However, as our approach typically converges within 20 to 30 iterations, it is likely that it remains faster than other prototype-based learning algorithms. The reason is that the latter usually require computing the entire similarity matrix, which costs $\mathcal{O}(n^2)$ operations. We will nevertheless discuss some techniques to reduce the training complexity of our approach in Sect. VI. As for the complexity at test time, it is clear that our approach drastically reduces it to $\mathcal{O}(m)$ operations.

Structured Inputs. Worth remarking, if the prototypes are not represented in terms of feature vectors, but using more complex structures (*e.g.*, graphs or strings), then the $\boldsymbol{z}$-step of our algorithm can not be optimized through gradient descent. More generally, one may define a set of minimal modifications to each prototype (*e.g.*, adding or removing a vertex in a graph, or characters in a string), and select those that greedily minimize the objective in Eq. (3). Note however that this black-box optimization procedure is clearly computationally demanding, and further empirical investigations are required to validate its effectiveness and extend our approach to more generic, structured inputs.

IV. Application Examples

We report here two application examples related to face verification and age estimation from faces [9], [10]. The goal of these examples is to show how and to what extent our super-sparse reduction approach can improve computational efficiency at test time without almost affecting system performance.

A. Biometric Identity Verification from Faces

Face verification consists of validating if a client is claiming his/her real identity (genuine claim) or he/she is pretending to be someone else (impostor claim). To this end, his/her face is acquired through a camera and compared against the reference prototypes of the claimed identity. To save time and memory, few reference prototypes are used. The corresponding similarity values are then combined with heuristic schemes, separating prototype selection from the algorithm used to combine the similarities [9].

We train a one-vs-all SVM for each client. This automatically selects the optimal prototype gallery (*i.e.*, the SVs), which however is often too large. We show that the proposed algorithm can successfully reduce the number of SVs, without affecting the recognition performance. We use the benchmark AT&T[2] and BioID[3] face datasets, respectively consisting of 40 clients with 10 face images each, and of 1,521 face images belonging to 23 clients. We assume that half of the clients are enrolled into the system, while the remaining ones are used only as impostors at test time. For training we randomly select 5 face images per enrolled client, and simulate impostors using face images of enrolled clients that do not belong to the claimed identity. At test time, we simulate impostors using the non-enrolled clients, and genuine claims using all the remaining face images of the claimed identity not used in the training set. This guarantees that the impostors are different between training and test sets (a common practice when testing biometric systems). Results are averaged over five different training-test pairs and client splits.

Matching algorithms. We use two different matching algorithms to compute similarities between faces.

1) Eigenface-based RBF Kernel [24], [25]. This algorithm maps each face image onto a reduced d-dimensional feature

[2] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

[3] https://www.bioid.com/About/BioID-Face-Database

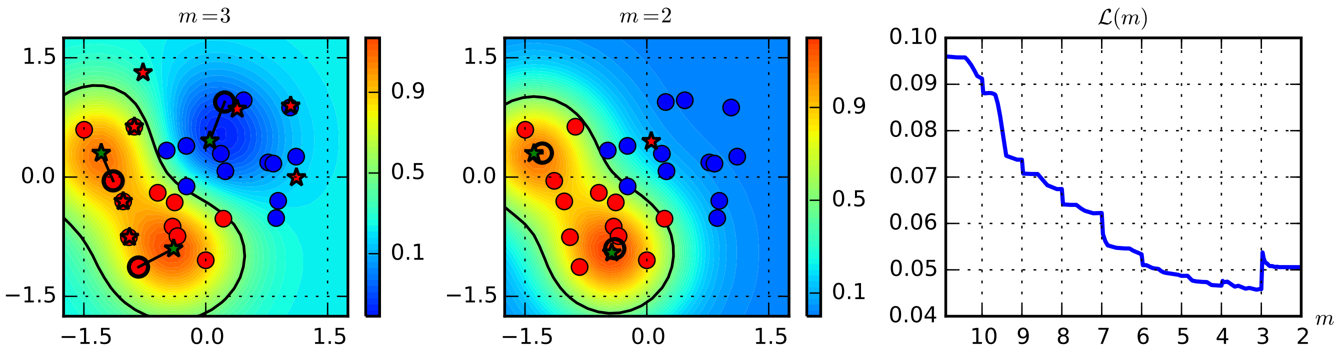

Fig. 3: An example of our incremental CV procedure to select m. We initially set $m = 10$, and iteratively remove one prototype at a time, up to $m = 2$. The discriminant function $g(\boldsymbol{x})$ of our super-sparse learning algorithm is shown (in colors) in the left and middle plot, for $m = 3$ and $m = 2$. The initial and final positions of the current prototypes are shown as black circles and green stars, respectively, while the prototypes removed at the previous iterations are reported as red stars. The minimum of $\mathcal{L}(m)$ (Eq. 8) is attained for $m = 3$ (using the mean squared error as the loss function $\ell(m)$, and $\rho = 10^{-3}$), as shown in the left plot. In fact, for $m = 2$, the set of blue points is not properly represented, and $\mathcal{L}(m)$ increases.

vector using principal component analysis (PCA). We select d so as to preserve 95% of the variance of the data. We then use the RBF kernel $s(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2)$ as the similarity measure, and set $\gamma = 1/d$ [26]. On average, we got $d = 56$ for AT&T, and $d = 41$ for BioID.

2) Elastic Bunch Graph Matching (EBGM) [27]. It extracts a bunch of image descriptors (*i.e.*, Gabor wavelets) at predefined landmark points in face images (*e.g.*, eyes and nose). These points correspond to nodes in a graph, which is elastically deformed to account for different face orientations and expressions during matching. In this case, $s(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is neither analytically given, nor positive semi-definite.

Verification methods. We compare our approach to learn super-sparse SVMs (SSVM) with the SVM-sel and SVM-red techniques for reducing the number of SVs [8] (see Sect. V), and also consider the standard SVM for comparison.

We set the SVM regularization parameter $C \in \{10^{-1}, 10^{0}, \ldots, 10^{3}\}$ by maximizing recognition accuracy through a 5-fold CV. As the two classes are highly unbalanced, we used a different C value for each class, multiplying it by the prior probability of the opposite one, estimated from training data. To ensure fast convergence of our Algorithm 1 to a stable solution, we run a set of preliminary experiments and set the gradient step $\eta = 0.5$ for both datasets; the regularization factor $\lambda = 10^{-6}$ for the Eigenface-based RBF Kernel, and $\lambda = 10^{-3}$ and $\lambda = 10^{-5}$ for the EBGM on the BioID and AT&T data, respectively. The gradients of $s(\boldsymbol{x}_i, \boldsymbol{z}_j)$ are analytically computable for the RBF Kernel; for the EBGM (which is not given analytically) we use the approximate gradient given in Eq. (7).

Results. Fig. 4 shows the fraction of incorrectly-rejected genuine claims (false rejection rate, FRR) vs. the fraction of incorrectly-accepted impostors (false acceptance rate, FAR) for each method, obtained by varying each client-specific threshold, and then by averaging over all clients and repetitions. The average number of matchings (prototypes) required by each method at test time is also reported. Except for SVM, this number has to be fixed in advance: for SVM-sel, SVM-red, and SSVM we respectively set it to 10, 2 and 2, when using the RBF Kernel, and to 5, for all methods, when using the EBGM.

Our SSVM achieves comparable performance as SVM but using only 2 and 5 virtual prototypes (instead of more than 20 and 15), respectively, for the RBF kernel and EBGM matching algorithms. Conversely, SVM-sel and SVM-red require a higher number of prototypes to achieve a comparable performance. This highlights that a principled approach may guarantee high recognition accuracy and efficiency using an extremely sparse set of virtual prototypes, without even knowing analytically the matching algorithm.

Interpretability. An example of the virtual prototypes found by our SSVM is shown in Fig. 5. We can appreciate that the genuine (virtual) prototypes ($\boldsymbol{\beta} > 0$) are obtained by merging genuine prototypes, preserving the aspect of the given client. Impostor prototypes ($\boldsymbol{\beta} < 0$) are instead the combination of faces of different identities, to compactly represent information about impostors. Although these prototypes do not correspond to any real user, interestingly they still resemble face images. This makes our SSVM approach interpretable, in the sense that a face image is considered genuine if it is sufficiently similar to the genuine prototypes found by our algorithm (and different from the impostor ones).

B. Age Estimation from Faces

The goal here is to predict the age of a person from a photograph of his/her face. We tackle this problem as a regression task, as most of the existing methods, and show that our approach can be helpful in this context to: (i) speed up age estimation at test time by dramatically reducing the number of reference prototypes; (ii) provide more interpretable decisions; and (iii) mitigate the risk of overfitting to specific face databases. The experimental setup is similar to the one defined in our previous work [10].

Datasets. We use two publicly-available benchmark face databases: Fg-Net Aging and FRGC. Fg-Net is the main

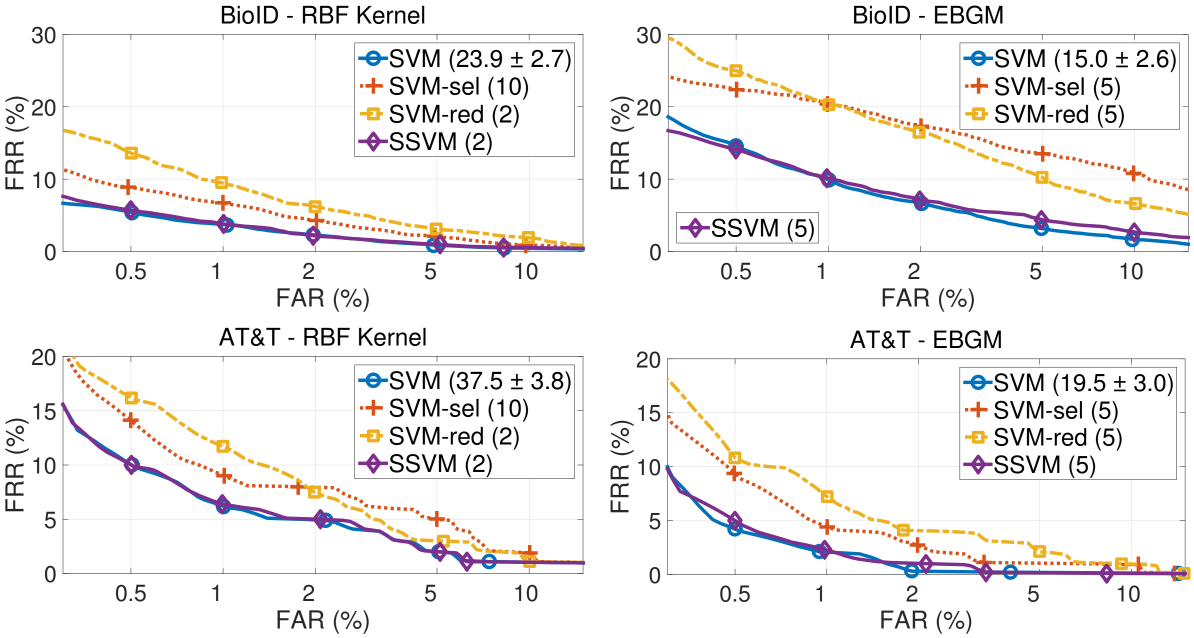

Fig. 4: Averaged FRR vs FAR values for the BioID (*top row*) and AT&T (*bottom row*) face datasets, and for the Eigenface-based RBF Kernel (*left column*) and EBGM (*right column*) algorithms. The average number of matchings required for verification and the std. dev. is reported in parentheses.

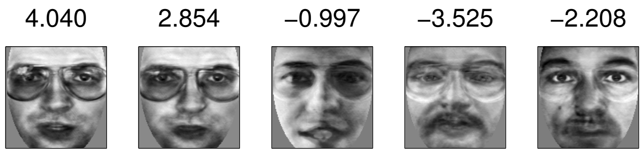

Fig. 5: Prototypes and $\boldsymbol{\beta}$ values learned by our SSVM for a client in the BioID dataset, using the EBGM matching algorithm.

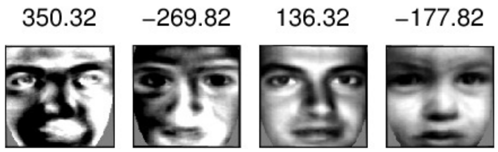

Fig. 6: Prototypes and $\boldsymbol{\beta}$ values learned by our super-sparse LASSO from the Fg-Net data.

database for this task. It includes about 1,000 images of 82 subjects acquired in a totally uncontrolled condition, which makes it particularly challenging. Many images are blurred, exhibit different resolutions and illumination conditions, and the number of subjects per age is not equally distributed. The age range for each subject varies from 0 to 69 years, although the majority of images belong to 20-year-old people. FRGC consists of about 50,000 face images acquired in different time sessions, belonging to about 500 people (about 200 females and 300 males) of different ethnicity, with ages spanning from 17 to 69 years. Face images were acquired in a controlled indoor environment, in frontal pose, to facilitate the recognition task. To keep the complexity of our experiments manageable, we restrict our analysis to a subset of about 5,000 images, randomly selected from this dataset. The age distributions of both datasets are shown in Fig. 7.

Experimental Setup. We normalize images as discussed in [10], and reduce the resulting set of 19,500 features (*i.e.*, pixel values) through linear discriminant analysis (LDA), retaining the maximum number of components (*i.e.*, the number of different age values minus one). We evaluate performance in terms of Mean Absolute Error: $\mathrm{MAE} = \frac{1}{r}\sum_{i=1}^{r} |g(\boldsymbol{x}_i) - y_i|$, where $g(\boldsymbol{x}_i)$ is the regression estimate of our approach for the i^{th} subject, whose true age is y_i, and r is the number of test images. We average results using a 5-fold CV procedure where each subject appears only in one fold. We use the RBF kernel as the similarity measure. We consider LASSO [12] and ridge [13] regression, and optimize their regularization parameter through CV. We compare our approach against the following prototype-selection methods [16], [17]: Random (PS-R), which randomly selects m prototypes from the training data; Border (PS-B), which selects m prototypes from the frontier of the training data; Spanning (PS-S), which selects the first prototype as the training-set median, and the remaining ones through an iterative procedure that maximizes the distance to the set of previously-selected prototypes; and k-medians (PS-KM), which runs k-means clustering to obtain m clusters from the training set, and then selects the m prototypes as their set medians (*i.e.*, the medians of the clusters). We optimize m according to the CV procedure defined in Sect. III, using the MAE as the loss function $\ell(m)$ and $\rho = 0.1$ (Eq. 8). We consider also a cross-database scenario in which training and test sets are drawn from different databases, to verify if prototype-selection methods are less prone to overfitting.

Results. As shown in Table I, both for the standard and cross-database evaluations, our algorithms exhibit almost the

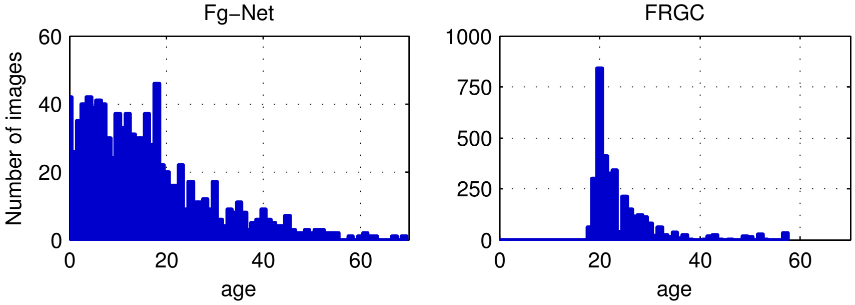

Fig. 7: Number of images per age (years) for Fg-Net (*left*) and FRGC (*right*).

TABLE I: Average MAE and the number of prototypes (in parentheses) selected by Ridge, LASSO, and the corresponding prototype-selection methods, for LDA-based features. Each column reports training/test sets, including cross-database evaluations; *e.g.*, Fg-Net/FRGC means that Fg-Net is used for training, and FRGC for testing. The lowest error values are highlighted in bold, for each configuration.

Method	**Fg-Net/Fg-Net**	**Fg-Net/FRGC**	**FRGC/Fg-Net**	**FRGC/FRGC**
Ridge	8.00 (781.6)	8.46 (781.6)	14.85 (2747.2)	4.53 (2747.2)
PS-R Ridge	9.93 (5.0)	9.11 (5.0)	**12.98** (4.0)	**4.10** (4.0)
PS-B Ridge	36.78 (5.0)	30.48 (5.0)	26.34 (4.0)	17.35 (3.2)
PS-S Ridge	11.13 (5.0)	10.36 (5.0)	15.29 (4.0)	4.85 (4.0)
PS-KM Ridge	10.45 (5.0)	13.94 (5.0)	13.62 (4.0)	4.04 (4.0)
SRidge	**9.06** (5.2)	**7.96** (4.6)	14.42 (4.4)	4.31 (4.2)
LASSO	7.92 (60.0)	8.99 (60.2)	14.71 (20.8)	4.67 (22.4)
PS-R LASSO	11.77 (5.0)	**7.81** (7.0)	14.54 (3.0)	5.40 (2.4)
PS-B LASSO	36.78 (5.2)	28.66 (7.0)	25.80 (2.6)	17.48 (2.2)
PS-S LASSO	10.54 (5.0)	10.34 (7.2)	**14.15** (3.0)	**4.48** (3.0)
PS-KM LASSO	12.54 (5.0)	9.96 (7.0)	15.37 (3.0)	5.14 (3.0)
SLASSO	**7.99** (6.2)	9.09 (7.6)	14.76 (3.6)	4.75 (4.0)

same performance as their non-sparse versions, despite using fewer prototypes. They often outperform also the competing prototype-selection methods, or achieve a comparable performance.

Interpretability. Interpretability of decisions is important to determine whether meaningful aging patterns are learned, *i.e.*, if the age of a subject can be correctly predicted from face images of different datasets. Fig. 6 shows a set of prototypes found by our super-sparse LASSO algorithm, which correctly assigns higher $\boldsymbol{\beta}$ values to older people.

V. Related Work

Besides work in similarity-based learning, discussed in Sects. I-II, another line of research related to super-sparse learning in similarity spaces is the one related to SVM reduction approaches, *i.e.*, approaches that aim at reducing the number of SVs of an SVM, or to learn SVM with reduced complexity directly [8], [28], [29]. Worth remarking, however, our approach is not specifically designed for SVMs and kernel machines, as it can be in principle applied to generic similarity functions and learning algorithms. Moreover, the first reported application example on face verification has also demonstrated that some of the existing methods for reduction of SVs in SVMs (*i.e.*, SVM-sel and SVM-red) can not achieve reduction rates and accuracy that are comparable with those achieved by our method. These methods in practice reduce the number of SVs by minimizing the ℓ_2 distance computed between the hyperplane normal of the given (unpruned) SVM and that of the reduced SVM in kernel space, as a function of the dual coefficients $\boldsymbol{\alpha}$ and the set of reduced prototypes [8]. While $\boldsymbol{\alpha}$ can be analytically found, as in our method, the choice of the reduced prototypes is different: SVM-sel eliminates one prototype at each iteration from the initial set using Kernel PCA to drive the selection; and SVM-red creates a new prototype at each iteration by minimizing the aforementioned ℓ_2-norm distance. Both approaches are thus greedy, as the reduced set of prototypes is constructed iteratively by removing or adding a prototype at a time, up to the desired number m. The reason of the superior performance exhibited by our algorithm is thus twofold: (i) SVM-sel and SVM-red require the matching algorithm to be a positive semi-definite kernel to *uniquely* find the coefficients $\boldsymbol{\alpha}$;[4] (ii) they do not modify the prototypes that are already part of the reduced expansion, and do not even reconsider the discarded ones. Our approach overcomes such limitations by optimizing a different objective (suited to non-positive semi-definite kernels too) and by iteratively modifying the virtual prototypes during the optimization. These may be common advantages also with respect to more recent reduction methods, but this deserves further investigation [28], [29].

[4] In fact, the notion of hyperplane exploited in their objective is only consistent for positive semi-definite kernels.

VI. Summary and Open Problems

The proposed approach aims to tackle computational complexity of similarity-based classifiers at test time. Our approach builds on [9]–[11], where we originally define our super-sparse learning machines. Here we have further extended our approach especially from an algorithmic viewpoint, by including a penalty term to reduce sensitivity to initialization of the virtual prototypes, and by designing a specific CV procedure to tune the number of prototypes m. We remark that we do not consider multi-class classification problems, but that an extension of our approach to deal with them has already been developed, exhibiting outstanding performance on image classification [11].

Our future research directions aim at designing very efficient (and interpretable) machines at test time. We will first explore different possibilities to overcome the computational bottleneck of our approach during training; *e.g.*, similarities between the virtual prototypes and the training samples can be only computed after a given number of iterations $p > 1$, instead of being computed at each shift of a virtual prototype. During the intermediate steps, the similarity values can be updated using a first-order approximation, which is provided by the computations performed at the previous iterations. Another interesting research direction consists of exploiting our super-sparse reduction approach to reduce complexity of other non-parametric estimators, like kernel density estimators and k-nearest neighbors. Finally, it would be interesting also to extend our approach to handle complex input structures like graphs and strings, considering efficient black-box optimization techniques.

References

[1] Y. Chen, E. K. Garcia, M. R. Gupta, A. Rahimi, and L. Cazzanti, “Similarity-based classification: Concepts and algorithms,” *J. Mach. Learn. Res.*, vol. 10, pp. 747–776, March 2009.

[2] E. Pękalska, P. Paclik, and R. P. W. Duin, “A generalized kernel approach to dissimilarity-based classification,” *J. Mach. Learn. Res.*, vol. 2, pp. 175–211, Dec. 2001.

[3] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer-Verlag New York, Inc., 1995.

[4] C. Cortes and V. Vapnik, “Support-vector networks,” *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sept. 1995.

[5] I. Steinwart, “Sparseness of support vector machines,” *J. Mach. Learn. Res.*, vol. 4, pp. 1071–1105, Nov. 2003.

[6] O. Chapelle, “Training a support vector machine in the primal,” *Neural Comput.*, vol. 19, no. 5, pp. 1155–1178, May 2007.

[7] E. Pękalska, R. P. W. Duin and P. Paclík, “Prototype selection for dissimilarity-based classifiers,” *Patt. Rec.*, vol. 39, no. 2, pp. 189–208, Feb. 2006.

[8] B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K.-R. Muller, G. Rätsch, and A. J. Smola, “Input space versus feature space in kernel-based methods," *IEEE Trans. on Neural Networks*, vol. 10, no. 5, pp. 1000–1017, Sept. 1999.

[9] B. Biggio, M. Melis, G. Fumera, and F. Roli, “Sparse support faces,” in *Proc. Int. Conf. Biometrics*, 2015, pp. 208–213.

[10] A. Demontis, B. Biggio, G. Fumera, and F. Roli, “Super-sparse regression for fast age estimation from faces at test time,” in *Proc. 18th Int. Conf. Image Analysis and Processing*, 2015, pp. 551–562.

[11] M. Melis, L. Piras, B. Biggio, G. Giacinto, G. Fumera, and F. Roli, “Fast image classification with reduced multiclass support vector machines,” in *Proc. 18th Int. Conf. Image Analysis and Processing*, 2015, pp. 78–88.

[12] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *J. Royal Stat. Soc. (Ser. B)*, vol. 58, no. 1, pp. 267–288, 1996.

[13] A. E. Hoerl and R. W. Kennard, “Ridge regression: Biased estimation for nonorthogonal problems,” *Technometrics*, vol. 12, no. 1, pp. 55–67, Feb. 1970.

[14] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf, “Large scale multiple kernel learning,” *J. Mach. Learn. Res.*, vol. 7, pp. 1531–1565, Dec. 2006.

[15] G. Chechik, V. Sharma, U. Shalit, and S. Bengio, “Large scale online learning of image similarity through ranking,” *J. Mach. Learn. Res.*, vol. 11, pp. 1109–1135, March 2010.

[16] K. Riesen, M. Neuhaus, and H. Bunke, “Graph embedding in vector spaces by means of prototype selection,” in *Proc. 6th Int. Conf. Graph-Based Repr. in Patt. Rec.*, 2007, pp. 383–393.

[17] B. Spillmann, M. Neuhaus, H. Bunke, E. Pekalska, and R. P. W. Duin, “Transforming strings to vector spaces using prototype selection,” in *Proc. Joint IAPR Int. Conf. Structural, Syntactic, and Stat. Patt. Rec.*, 2006, pp. 287–296.

[18] A. Gisbrecht, B. Mokbel, and B. Hammer, “The Nyström approximation for relational generative topographic mappings,” in *NIPS Workshop on Challenges of Data Visualization*, 2010.

[19] E. Pękalska and B. Haasdonk, “Kernel discriminant analysis for positive definite and indefinite kernels,” *IEEE Trans. Patt. An. and Mach. Intell.*, vol. 31, no. 6, pp. 1017–1032, June 2009.

[20] R. J. Hathaway and J. C. Bezdek, “Nerf c-means: Non-euclidean relational fuzzy clustering,” *Patt. Rec.*, vol. 27, no. 3, pp. 429–437, March 1994.

[21] D. O. Seaghdha and A. Copestake, “Using lexical and relational similarity to classify semantic relations,” in *Proc. 12th Conf. Assoc. for Computational Linguistics*, 2009, pp. 621–629.

[22] T. Pedersen, S. V. S. Pakhomov, S. Patwardhan, and C. G. Chute, “Measures of semantic similarity and relatedness in the biomedical domain,” *J. of Biomedical Informatics*, vol. 40, no. 3, pp. 288–299, June 2007.

[23] T. Zhang, “Solving large scale linear prediction problems using stochastic gradient descent algorithms,” in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, pp. 116–123.

[24] M. Turk and A. Pentland, “Eigenfaces for recognition,” *J. Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[25] K. Jonsson, J. Kittler, Y. Li, and J. Matas, “Support vector machines for face authentication,” *Image and Vision Comput.*, vol. 20, no. 5-6, pp. 369–375, April 2002.

[26] C.-C. Chang and C.-J. Lin, “LibSVM: A library for support vector machines,” *ACM Trans. Intelligent Systems and Technology*, vol. 2, no. 3, pp. 5412–5475, April 2011.

[27] J. Beveridge, D. Bolme, B. Draper, and M. Teixeira, “The CSU face identification evaluation system,” *Machine Vision and Applications*, vol. 16, no. 2, pp. 128–138, Feb. 2005.

[28] S. S. Keerthi, O. Chapelle, and D. DeCoste, “Building support vector machines with reduced classifier complexity,” *J. Mach. Learn. Res.*, vol. 7, pp. 1493–1515, July 2006.

[29] Z. Wang, K. Crammer, and S. Vucetic, “Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale SVM training,” in *J. Mach. Learn. Res.*, vol. 13, pp. 3103–3131, Oct. 2012.