A Support Vector Machine Classifier from a Bit–Constrained, Sparse and Localized Hypothesis Space

Davide Anguita, Senior Member, IEEE, Alessandro Ghio, Member, IEEE, Luca Oneto, Student Member, IEEE, Sandro Ridella, Member, IEEE

Abstract—Choosing an appropriate hypothesis space in classification applications, according to the Structural Risk Minimization (SRM) principle, is of paramount importance to train effective models: in fact, properly selecting the space complexity allows to optimize the learned functions performance. This selection is not straightforward, especially (though not solely) when few samples are available for deriving an effective model (e.g. in bioinformatics applications). In this paper, by exploiting a bit-based definition for Support Vector Machine (SVM) classifiers, selected from an hypothesis space described according to sparsity and locality principles, we show how the complexity of the corresponding space of functions can be effectively tuned through the number of bits used for the function representation. Real world datasets are exploited to show how the number of bits and the degree of sparsity/locality imposed to define the hypothesis space affect the complexity of the space of classifiers and, consequently, the performance of the model, picked up from this set.

I. INTRODUCTION

The learning process to train a classifier consists of two phases, according to the Structural Risk Minimization (SRM) principle [1]. As a first step, an appropriate hypothesis space is chosen during the Model Selection (MS) phase, by estimating the size (and, thus, the complexity) of the hypothesis space [1], [2], [3] and, consequently, the generalization ability of the class of models. Secondly, the classifier characterized by the best trade-off between underfitting and overfitting [1] is picked up from the chosen space of functions during the training (TR) step.

In the case of Support Vector Machine (SVM) classifiers [1], [4], the TR process consists in finding a set of parameters by solving a Convex Constrained Quadratic Programming (CCQP) problem, for which many effective techniques have been proposed [5]. The MS phase comprises the tuning of a set of additional variables (hyperparameters), which is not a trivial task and represents an open research problem [6], [7], [8], [9], [10], [11]. This last assertion is particularly valid in the small-sample setting, i.e. when few high-dimensional data are available (e.g. in microarray classification applications): in these cases, traditional MS approaches result, in fact, to be unreliable [3], [12], [13], [7], [14].

The key factor for the success of MS procedures is the ability of identifying the smallest hypothesis space which includes the best possible model, which is obviously unknown. In this paper, we target this objective by focussing on the SVM MS phase and showing how benefits on the learning process of SVM classifiers can be obtained by exploiting a bit-based definition for the hypothesis spaces, i.e. by using classes of functions where models are described through a fixed, and properly tuned, number of bits.

Bit-based classifiers have been studied in the past in the framework of Machine Learning (ML) (e.g. [15], [16], [17], [18], [19], [20], [21]), usually motivated by application-specific requirements, such as the need to accelerate the process of learning with dedicated hardware or the energy-sparing requirements of applications based on mobile stand-alone devices (e.g. smartphones [22]). In this paper, we completely change this perspective, as the motivation for expressing a classifier in a bit-based way is not derived from applicative necessities but is strictly linked to the possibility of properly shrinking the size of the SVM hypothesis space, with resulting positive consequences on the reduction of the complexity of the class of functions. In this framework we also show how encapsulating the notions of sparsity [23] and local complexity [11] in the description of the hypothesis space can lead to further improvements in the estimation of the generalization ability of classification models.

The paper is organized as follows: in Section II we introduce the learning framework and the notation; in Section III we recall some measures of complexity in order to estimate the size of an hypothesis space; then, in Section IV we introduce a new approach for bit-based SVM classifiers and hypothesis spaces definition, in accordance with sparsity and locality principles. Section V describes the algorithm, properly modified for SVM learning purposes in the new defined framework. Finally, in Section VI, we show the results obtained with the proposed approach on a series of real-world data, in particular belonging to the small-sample setting, being it the most challenging framework for testing new SVM MS methods.

II. THE LEARNING FRAMEWORK

In the framework of supervised learning the goal is to approximate a relationship between examples from a set \( X \) and outputs from a set \( Y \): as we are targeting binary classification problems, we assume here \( X \subset \mathbb{R}^d \) and \( Y \subset \{ \pm 1 \} \). The relationship between examples and outputs is encapsulated by a fixed, but unknown, probability measure \( P \) on \( Z = X \times Y \). We denote as \( \mathcal{F} \) a class functions on \( X \).

A training set \( D_n = \{(x_1,y_1),\ldots,(x_n,y_n)\} \) is sampled according to \( P \). The learning algorithm maps \( D_n \) to \( f \in \mathcal{F} \) and the accuracy in representing the hidden relationship \( P \)
is measured with reference to a loss function $\ell(f(x), y)$. The goal is to find the class of function $F$ and then the function $f \in F$ that minimizes the generalization error $L(f) = \mathbb{E}_{x,y}[\ell(f(x), y)]$.

$L(f)$ cannot be explicitly computed as $\mathcal{P}$ is unknown; however, according to the well-known Structural Risk Minimization (SRM) principle [1], we can resort to defining a series of hypothesis spaces of increasing size $\{F_1 \subseteq F_2 \subseteq F_3 \ldots \}$. For each of these spaces, we can study the supremum of the uniform deviation $S(F_j) = \sup_{f \in F_j} \left| L(f) - \hat{L}(f) \right|$, which can be seen as a complexity term [1], [2], [3], [7] and where $\hat{L}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)$ is the empirical error. Then, we can upper-bound the generalization error through the following expression, which holds with probability $(1 - \delta)$:

$$L(f) \leq \hat{L}(f) + S(F_i) + \varphi(\delta) \quad (1)$$

where $\varphi(\delta)$ is the confidence term [24], [7], independent of both the hypothesis space and the model. In practice, the Model Selection (MS) phase consists in identifying the most suitable class of functions $F_j$ and the training (TR) phase consists in identifying the best classifier in it $f \in F_j$, so to minimize the estimated generalization error of Eq. (1).

### III. Complexity Measures

Given a hypothesis space $F$, in order to compute $S(F)$, both data independent [1] (e.g. the Vapnik-Chervonenkis Dimension) and data dependent measures of complexity [24], [2] (e.g. Maximal Discrepancy or Rademacher Complexity) can be exploited. In this paper, we opt for the Maximal Discrepancy (MD) for two main reasons: data dependent measures of complexity give better insight on the process of learning; secondly, MD is easier to calculate with respect to the Rademacher Complexity [25].

We can compute the empirical MD $\hat{M}_n(F)$ (or its expected value $\mathcal{M}_n(F) = \mathbb{E}_F \hat{M}_n(F)$) by exploiting a bounded loss function $\ell_B \in [0, 1]$, and we have:

$$\hat{M}_n(F) = \min_{f \in F} \frac{2}{n} \sum_{i=1}^{n} \sigma_i \ell_B(f(x_i), y_i) \quad (2)$$

where $\sigma$ are independent, uniformly distributed and $\{\pm 1\}$-valued random variables with $\sum_{i=1}^{n} \sigma_i = 0$. Usually, in order to avoid unlucky realizations of $\sigma$, $\hat{M}_n(F)$ is averaged over some (typically $\approx 30$) different realizations; however, rigorously speaking, one realization is enough [2], [26].

Computing the minimum in Eq. (2) over the whole hypothesis space using a bounded loss function is clearly computationally difficult to handle: branch-and-bound approaches result to be effective for this purpose in the small sample setting (e.g. the approach of [27], implemented in CPLEX [28]). Instead, when the cardinality of the set exceeds few patterns (some tens up to very few hundreds), algorithms allowing to find approximated (non-optimal) solutions must be exploited; in these cases, we can always generalize the formulation of Eq. (2) by approximating the MD of the hypothesis space with the MD computed using a generic algorithm $\mathcal{A}$ [29]:

$$\hat{M}_n(\mathcal{A}) = \min_{A} \frac{2}{n} \sum_{i=1}^{n} \sigma_i \ell_B(f(x_i), y_i) \quad (3)$$

i.e. we hypothesize $\hat{M}_n(F) \approx \hat{M}_n(\mathcal{A})$, analogously to what done for other complexity measures in [1], [24], [11].

### IV. Bit-Constrained, Sparse and Localized SVM

In the following we consider only the class of functions of linear classifiers in the original input space $\mathcal{X}$ where the bias is nullified, i.e. $f(x) = w \cdot x$. Note that this simplifying hypothesis can be safely carried out if: the dimensionality of the data is much larger than the cardinality of the learning set $D_n$ (i.e. $d \gg n$) [30], [31], since the Vapnik-Chervonenkis dimension of the models $d_{VC}$ is larger than $n$ [1], which is the typical configuration for small-sample problems mostly targeted by this work; we exploit Radial Basis Function (RBF) kernels [30], for which we target the extension of the proposed approaches in the Appendix.

Since we are dealing with classification problems and we have to choose a bounded loss function, the hard loss $\ell_H(f(x), y) = \text{sign} \left( y f(x) \right)$, where $\text{sign}(\cdot) = +1$ if $\cdot \leq 0$ and $\text{sign}(\cdot) = 0$ if $\cdot > 0$, should be the most natural choice, as it counts the number of misclassified samples. However, the use of a hard loss function makes the problem of finding the optimal $f$ computationally hard. Therefore, the conventional SVM algorithm makes use of the well-known hinge loss function, which can be expressed as $\ell_h(f(x), y) = [1 - y f(x)]_+$ where $[,]_+ = \max (0, \cdot)$. Unfortunately $\ell_h$ is unbounded and this complicates the problem of predicting the generalization ability of $f$. In order to address this issue we exploit the trimmed hinge loss $\ell_T$ [32], [14] that is a bounded and smooth upper bound of $\ell_H$:

$$\ell_T(f(x), y) = \begin{cases} \ell_H(f(x), y) & \text{if } y f(x) \leq 0 \\ \ell_h(f(x), y) & \text{otherwise}. \end{cases} \quad (4)$$

By introducing a constraint $\|w\|_2 \leq w_{\text{MAX}}$, which allows to control the size of the hypothesis space $F$ thanks to the exploitation of an Ivanov regularization problem [1], the training optimization formulation to identify the best $f \in F$ can be defined as:

$$\min_{w} \sum_{i=1}^{n} \ell_T(f(x_i), y_i) \quad (5)$$

s.t. $\|w\|_2 \leq w_{\text{MAX}}, \quad w \in \mathbb{R}^d$.

In this paper we propose to introduce a bit-based regularization term in Problem (5) in order to further shrink the hypothesis space, so to define smaller and (generally) more efficient classes of functions. We can thus switch from the conventional representation of $w_{j \in \{1, \ldots, d\}} \in \mathbb{R}$ to a bit-based representation of the main quantities in Problem (5): $\tilde{w}_{j \in \{1, \ldots, d\}} \in \{+\frac{1}{\kappa} \ldots -\frac{1}{\kappa}, \ldots, -\frac{1}{\kappa}, \ldots, -2^{\kappa} + 1, \ldots, 2^{\kappa} - 1\}$. $\kappa$ is the number of bits needed for representing $f(x)$ and influences the complexity of the space: if we use more bits we can
represent more functions and then we have a more complex space. Consequently Problem (5) becomes:

\[
\begin{align*}
\min_{\mathbf{w}} & \quad \sum_{i=1}^{n} \ell_{T}(f(\mathbf{x}_i), y_i) \\
\text{s.t.} & \quad \|\mathbf{w}\|_{2}^{2} \leq \omega_{\text{MAX}}^{2} \\
& \quad w_{j}\in\{1,\ldots,d\} \in \frac{\omega_{\text{MAX}}}{2^{\kappa} - 1} \{ -2^{\kappa} + 1, \ldots, 2^{\kappa} - 1 \}
\end{align*}
\]

where \( \kappa \in \{1, 2, \ldots\} \).

As a sparse representation of the solution is desirable to further improve the classifier performance on new samples generated from \( \mathcal{P} \) [23], we introduce another hyperparameter, that is the number of \( w_{j}\in\{1,\ldots,d\} \) different from zero. In order to include this constraint, Problem (6) must be reformulated as:

\[
\begin{align*}
\min_{\mathbf{w}} & \quad \sum_{i=1}^{n} \ell_{T}(f(\mathbf{x}_i), y_i) \\
\text{s.t.} & \quad \|\mathbf{w}\|_{2}^{2} \leq \omega_{\text{MAX}}^{2} \\
& \quad w_{j}\in\{1,\ldots,d\} \in \frac{\omega_{\text{MAX}}}{2^{\kappa} - 1} \{ -2^{\kappa} + 1, \ldots, 2^{\kappa} - 1 \} \\
& \quad \sum_{j=1}^{d} |w_{j}| \neq 0 \leq d\zeta
\end{align*}
\]

where \( \zeta \in [0, 1] \) represents the percentage of \( w_{j}\in\{1,\ldots,d\} \) different from zero.

According to the ideas of [11], [3], [33], we can further shrink the hypothesis space: let \( D'_{n} = \{(x'_1, y'_1), \ldots, (x'_n, y'_n)\} \) be another set of data consisting of \( n' \) samples (originated by \( \mathcal{P} \) but independent of \( D_{n} \)). Then we can choose only those functions which are characterized by an error rate on this set below a predetermined threshold: in fact, these classifiers will be most likely chosen by the optimization procedure, being the models able to combine a small misclassification rate on \( D_{n} \) and a good generalization performance on the set \( D'_{n} \). By reformulating Problem (7), we have:

\[
\begin{align*}
\min_{\mathbf{w}} & \quad \sum_{i=1}^{n} \ell_{T}(f(\mathbf{x}_i), y_i) \\
\text{s.t.} & \quad \|\mathbf{w}\|_{2}^{2} \leq \omega_{\text{MAX}}^{2} \\
& \quad w_{j}\in\{1,\ldots,d\} \in \frac{\omega_{\text{MAX}}}{2^{\kappa} - 1} \{ -2^{\kappa} + 1, \ldots, 2^{\kappa} - 1 \} \\
& \quad \sum_{j=1}^{d} |w_{j}| \neq 0 \leq d\zeta \\
& \quad \sum_{k=1}^{n'} \ell_{\xi}(f(\mathbf{x}'_k), y'_k) \leq 2n' \varepsilon
\end{align*}
\]

where \( \varepsilon \in [\varepsilon_{\text{MIN}}, \infty) \) is the percentage of misclassifications that we accept to perform on \( D'_{n} \). Note, moreover, that we exploited \( \ell_{\xi} \) instead of \( \ell_{T} \) in the constraint of Eq. (12) in order to avoid introducing another non-convexity: we can safely use \( \ell_{\xi} \) as the hinge loss is a looser upper-bound of the number of errors than the trimmed loss. \( \varepsilon_{\text{MIN}} \) can be found by solving:

\[
\begin{align*}
\varepsilon_{\text{MIN}} = \min_{\mathbf{w}} & \quad \sum_{i=1}^{n'} \ell_{\xi}(f(\mathbf{x}'_k), y'_k) \\
\text{s.t.} & \quad \text{Eqns. (9), (10) and (11)}
\end{align*}
\]

with the same \( \omega_{\text{MAX}}, \kappa \) and \( \zeta \) of Problem (8). The previous problem defines the search for a linear SVM-like classifier in a bit-based, sparse and local (thanks to the exploitation of \( D'_{n} \)) class of functions. Note that \( D'_{n} \) could not be available: however, part of the samples of \( D_{n} \) can be reserved for this purpose, analogously to the approach proposed in [3].

In order to further enhance the sparsity of the solution, we can switch from the L2-regularization term of Eq. (9) to an L1-regularization term, which is known to allow reducing the number of \( w_{j}\in\{1,\ldots,d\} \) different from zero [34], [35], [36] (as graphically shown by the geometry of the constraints in Figure 1). Broadly speaking, the original framework of SVM is totally lost and solutions based on L1-regularization are usually characterized by worse performance than L2-regularization ones. However, we will show in the following sections that the L1 alternative formulation to Problem (8) can be useful for enhancing the sparsity of the solution of the L2-formulation:

\[
\begin{align*}
\min_{\mathbf{w}} & \quad \sum_{i=1}^{n} \ell_{T}(f(\mathbf{x}_i), y_i) \\
\text{s.t.} & \quad \|\mathbf{w}\|_{1} \leq \omega_{\text{MAX}}, \text{Eqns. (10), (11) and (12)}
\end{align*}
\]

V. SOLVING THE OPTIMIZATION PROBLEMS

Let us define the following quantities with reference to \( D_{n} \): \( X = [x_1, x_2, \cdots, x_n]^T, y = [y_1, y_2, \cdots, y_n]^T, Y = \text{diag}(y) \) (i.e. \( Y \) is a diagonal matrix where the elements on the diagonal are the labels of the training data). Let also \( \alpha_p \) be a vector of \( p \) element all assuming value \( a \). The quantities \( X', y' \) and \( Y' \) can be analogously defined for \( D'_{n} \).

As the trimmed hinge loss, exploited in Problems (8) and (14), is bounded, we can exploit the results of the minimization for computing the MD term of Eq. (3). For this purpose, and according to the quantities defined above,
the objective of this section is to derive an algorithm to solve the following re-formulation of Problem (8):

$$\min_{w, \xi} \sigma^T \min (1_n, \xi)$$

s.t. $$w^T w \leq w_{\text{MAX}}$$

$$YXw \geq 1_n - \xi, \quad \xi \geq 0_n$$

$$Y'X'w \geq 1_n - \xi', \quad \xi' \geq 0_n,$$

$$1^T w \xi \leq 2n' \xi,$$

$$w_j \in \{1, \ldots, d\} \in \frac{w_{\text{MAX}}}{2^n - 1} \{-2^n + 1, \ldots, 2^n - 1\}$$

$$\sum_{j=1}^{d} [w_j \neq 0] \leq d \xi,$$

Equations (16), (17) and (18).

In order to solve Problem (15) we can adopt the approach of [37], [38], which consists in momentarily neglecting the constraints (19) and (20):

$$\min_{w, \xi} \sigma^T \min (1_n, \xi)$$

s.t. $$w^T w \leq w_{\text{MAX}}^2.$$  Equations (16), (17) and (18).

Equation (19) can be subsequently satisfied by searching for the nearest solution that meets the constraint; on the contrary, finding a solution that satisfies Eq. (20) is not straightforward and we will have to resort to L1-regularization. As remarked above, in fact, L1-based reformulation can be useful to enhance sparsity of the hypothesis space. Thus, we can also reformulate Problem (14):

$$\min_{w^+, w^-} \sigma^T \min (1_n, \xi)$$

s.t. $$1^T (w^+ + w^-) \leq w_{\text{MAX}}$$

$$w^+, w^- \geq 0_d,$$  Equations (16), (17) and (18).

where $$w = w^+ - w^-$$ and the constraints of Equations (19) and (20) are neglected.

A. Solving Problems (21) and (22)

Problems (21) and (22) are based on Ivanov regularization, and can be solved accordingly to the procedure proposed in [39] by deriving a Tikhonov regularization based reformulation, which is usually easier to solve. The Tikhonov formulation for Problem (21) is the following [39]:

$$\min_{w, \xi} C^{\text{L2}} : \frac{1}{2} w^T w + C \sigma^T \min (1_n, \xi)$$

s.t.  Equations (16), (17) and (18)

while for the L1-regularization formulation of Problem (22) we have:

$$\min_{w^+, w^-} C^{\text{L1}} : 1^T (w^+ + w^-) + C \sigma^T \min (1_n, \xi)$$

s.t.  $$w^+, w^- \geq 0_d,$$  Equations (16), (17) and (18).

The two previous formulations are both non-convex, thus we have to resort to the Convex ConCave Constrained Programming (CCCP) method [40], [14] for finding a good (though approximate) solution. The first step of the procedure consists in splitting the cost function in its concave $$J_{\text{concave}}(w)$$ and convex $$J_{\text{convex}}(w)$$ parts, computing the derivative of $$J_{\text{concave}}(w)$$ and finally apply the method depicted in Algorithm 1.

In order to separate the convex and concave parts in the cost functions of Problems (23) and (24), we note that the two regularization terms $$1^T d (w^+ + w^-)$$ and $$w^T w$$ in $$C^{\text{L1}}$$ and $$C^{\text{L2}}$$ are both convex; consequently, we concentrate on the term $$C \sigma^T \min (1_n, \xi).$$ First we define $$I^+$$ as the set of indexes $$I^+ = \{i : \sigma_i = +1, \forall i \in \{1, \ldots, n\}\}$$ and $$I^-$$ as the set of indexes $$I^- = \{i : \sigma_i = -1, \forall i \in \{1, \ldots, n\}\}.$$

Then we have:

$$C \sigma^T \min (1_n, \xi) = C \sigma^T [\xi - \max (0_n, \xi - 1_n)]$$

$$= \begin{cases} J_{\text{concave}}(w) & \text{if } i \in I^+: \frac{\partial J_{\text{concave}}(w)}{\partial w} I_{\text{concave}}(w) \\ \text{max}(C \xi_i + \xi_i - 1) & \text{if } i \in I^-: C \xi_i - C \max (0, \xi_i - 1) \end{cases}$$

(25)

Then we obtain:

$$w^* = \sum_{i=1}^{n} \frac{d J_{\text{concave}}(w)}{dw} \frac{\partial J_{\text{concave}}(w)}{\partial w(i)} = \sum_{i=1}^{n} \Delta_i^{(t)} y_i w^T x_i$$

$$= \Delta^{(t)} Y X w$$

(26)

where $$\Delta_i^{(t)}$$ is computed as:

$$\Delta_i^{(t)} = \begin{cases} C & \text{if } y_i w^{(t)}(i, x_i) \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

(27)

Consequently we reformulate Problems (23) and (24) so that they can be solved with the CCCP procedure. In particular, for Problem (23) we have:

$$\min_{w, \xi} \frac{1}{2} w^T w + C 1^T d (w^+ + w^-) + \Delta^{(t)} Y X w$$

s.t.  $$YXw \geq T - \xi, \quad \xi \geq 0_n$$

$$Y'X'w \geq 1_n - \xi', \quad \xi' \geq 0_n,$$

$$1^T e \xi \leq 2n' \xi,$$

(28)

$$\text{where } T_i = 1, \forall i \in I^+ \text{ and } T_i = 0, \forall i \in I^-.$$  For Problem (24), instead, we have:

$$\min_{w^+, w^-} 1^T d (w^+ + w^-) + C \sigma^T \min (1_n, \xi)$$

$$+ \Delta^{(t)} Y X (w^+ - w^-)$$

(29)

Algorithm 1: CCCP procedure

Data: $$J_{\text{convex}}(w)$$ and $$J_{\text{concave}}(w)$$

Result: $$w^*$$

Initialize $$w^{0}$$.

while True do

$$w^{t+1} = \arg \min_{w} J_{\text{convex}}(w) + \frac{d J_{\text{concave}}(w)}{dw} w;$$

if $$\|w^{t+1} - w^{t}\|_2 \leq \epsilon$$ then

break

return $$w^* = w^{t+1}$$
Since Problem (32) is a standard Linear Programming (LP) problem that can be solved with many effective tools like CPLEX [28], the solution of Problem (24) and, obviously, of Problem (22) by using the approach of [39]) is pretty straightforward to derive.

Solving Problem (28) is not likewise easy, unfortunately. We have to compute the Lagrangian:
\[
L = \frac{1}{2} w^T w + C_1 n \xi + \Delta (t)^T Y X w \\
- \alpha^T (Y X w - T + \xi) - \mu^T \xi \\
- \alpha^T (Y' X' w - 1_n + \xi') - \mu^T \xi' \\
+ \tau (1_{n'} \xi' - 2 n' \varepsilon)
\]
and derive the following KKT and complementary conditions:
\[
\frac{\partial L}{\partial w} = w + X^T Y (\alpha - \Delta (t)) + X'^T Y' \alpha' = 0_n \\
\frac{\partial L}{\partial \xi} = C_1 n - \alpha - \mu = 0_n \Rightarrow \alpha \leq C_1 n \\
\frac{\partial L}{\partial \xi'} = \tau 1_n' - \alpha' - \mu' = 0_n' \Rightarrow \alpha' \leq \tau 1_n' \\
\gamma, \tau \geq 0, \alpha, \mu, \xi \geq 0, \alpha', \mu', \xi' \geq 0
\]
\[
\alpha^T (Y X w - T + \xi) = 0, \ Y X w \geq T - \xi \\
\mu^T \xi = 0, \xi \geq 0_n \\
\alpha'^T (Y' X' w - 1_n + \xi') = 0, \ Y' X' w \geq 1_n - \xi' \\
\mu'^T \xi' = 0, \xi' \geq 0_n' \\
\tau (1_{n'} \xi' - 2 n' \varepsilon) = 0, \ 1_{n'} \xi' \leq 2 n' \varepsilon
\]
Consequently the dual formulation of Problem (28) can be expressed as:
\[
\min_{\beta, \alpha', \tau} \frac{1}{2} \begin{bmatrix} \beta^T T \frac{Q}{Q'^T} Q' \beta' & \beta^T \alpha' \\
\alpha'^T T^{1_n'} \alpha' & \alpha'^T \alpha' & \tau \end{bmatrix} + \tau 2 n' \varepsilon \\
s.t. \begin{bmatrix} 0_n - \Delta (t) & 0_n' \end{bmatrix} \leq \begin{bmatrix} \beta' \\
\alpha' \end{bmatrix} \leq \begin{bmatrix} C_1 n - \Delta (t) \\
\tau 1_{n'} \end{bmatrix}
\]
where \( \Delta (t) = A - D \alpha' \)
\( Q = Y X Y^T Y, \ Q' = Y' X' Y' X' Y'^T \), \( Q'' = Y X Y^T Y \).
Since Problem (28) is convex, so will be its dual formulation (Problem (43)) [41].
In order to identify the best value \( \tau * \), we can apply a bisection iterative procedure [42], starting by fixing \( \tau = \tau_0 \) and solving:
\[
\min_{\beta, \alpha'} \text{Cost}(44) (\beta, \alpha', \tau_0) = \frac{1}{2} \begin{bmatrix} \beta^T T \frac{Q}{Q'^T} Q' \beta' & \beta^T \alpha' \\
\alpha'^T T^{1_n'} \alpha' & \alpha'^T \alpha' & \tau_0 2 n' \varepsilon \\
0_n - \Delta (t) & 0_n' \end{bmatrix} \leq \begin{bmatrix} \beta' \\
\alpha' \end{bmatrix} \leq \begin{bmatrix} C_1 n - \Delta (t) \\
\tau_0 1_{n'} \end{bmatrix}
\]
The solution of the previous problem can be easily found by exploiting the method proposed in [37], which slightly modifies the Sequential Minimal Optimization algorithm [5].

B. Contemplating the constraints of Eqns. (19) and (20)
In the previous subsection we showed how to find a solution to Problems (21) and (22), where the constraints of Eqns. (19) and (20) were neglected. Our objective is now to derive the final solution of Problem (15) by exploiting the methods proposed so far.

The constraint of Eq. (19) can be easily contemplated by exploiting the approach of [37], [38], i.e. we look in a neighborhood set of the solutions of Problem (32) for a new value of w satisfying the combinatorial constraint of Eq. (19). For what concerns, instead, Eq. (20), finding a sparse solution is not trivial. However, we can exploit the naive sparsity of L1-regularization solutions, as shown in Algorithm 3: in practice, we first find a sparse (though not very effective) solution according to the L1 procedure by solving Problem (22) as depicted in the previous subsection; then, we filter from the training set only the subset of \([d] \xi] \) features which are best ranked according to the L1 solution; finally, we exploit the filtered training set to find the best solution with the L2 approach (according to Algorithm 2 and the method proposed in [37], [38]). In this way, the approximated final solution of Problem (15) satisfies all the constraints and is effective, as we are able to combine the quality of sparseness of L1-regularization with the performance of L2-formulation solutions.

VI. EXPERIMENTAL RESULTS
To derive the experimental results, we focus on small-sample problems, i.e. datasets for which the dimensionality of the data is larger than the cardinality of the sets. In

Algorithm 2: Algorithm for solving Problem (43)

Data: \( D_n, D'_{n'} \), \( C \), \( \varepsilon \) and numerical precision \( \varepsilon \)
Result: \( w^* \)
\( \tau_{LOW} = 0 \)
Solve Problem (44) with \( \tau_0, C_{LOW} = \text{Cost}(44)(\beta, \alpha', \tau_0) \);
\( \tau_0 = 1 \);
while True do
\( \text{Solve Problem (44) with } \tau_0, C_{UP} = \text{Cost}(44)(\beta, \alpha', \tau_0); \)
if \( C_{UP} \geq C_{LOW} \) then
\( \tau_0 = \tau_0; \)
break
\( \tau_0 = 2 \tau_0; \)
while \( \left( \tau_{UP} - \tau_{LOW} \right) \geq \epsilon \) do
\( \tau_1 = \tau_{LOW} + \left( \tau_{UP} - \tau_{LOW} \right) / 2; \)
\( \tau_2 = \tau_{LOW} + 2 \left( \tau_{UP} - \tau_{LOW} \right) / 3; \)
Solve Problem (44) with \( \tau_0 = \tau_1, C_1 = \text{Cost}(44)(\beta, \alpha', \tau_0); \)
Solve Problem (44) with \( \tau_0 = \tau_2, C_2 = \text{Cost}(44)(\beta, \alpha', \tau_0); \)
if \( C_1 \geq C_2 \) then
\( \tau_{LOW} = \tau_1 \)
else
\( \tau_{UP} = \tau_2 \)
[\( \beta, \alpha' \) = Solve Problem (44) with \( \tau_0 = \left( \tau_{UP} + \tau_{LOW} \right) / 2; \)
return \( w^* = X^T Y \beta + X' T Y' \alpha' \)
Algorithm 3: Algorithm for solving Problem (15)

Data: $D_e$, $D_o$, $C, \kappa, \zeta$ and numerical precision $\epsilon$

Result: $w^*$

Solve Problem (22) and obtain $w^{L,1}$;

$X^{(d)}_{L} = []$;

$X^{(d)}_{h} = []$;

$w^{index} = []$;

for $j = 1, \ldots, |d|_{0}$ do

$[\text{value, index}] = \max ([|w_{j,1}^{l}|, \ldots, |w_{j,d}^{l}|])$

$w^{index} = w^{index} + \text{value}$

$X^{(d)}_{L} = X^{(d)}_{L} \cup X^{(d)}_{h}$

$X^{(d)}_{h} = X^{(d)}_{h} \setminus X^{(index)}$

$w_{j}^{index} = 0$

end for

Solve Problem (21) with $X^{(d)}_{L}, X^{(d)}_{h}$ and obtain $w^{L,2}$;

for $j = 1, \ldots, |d|_{0}$ do

$w_{j}^{L,2} = \text{nearest integer to } \frac{|w_{j}^{L,1}|}{2^{256}} \{ -2^{256} + 1, \ldots, 2^{256} - 1 \}$

$w_{j}^{*} = 0$

end for

for $j = 1, \ldots, |d|_{0}$ do

$w_{j}^{index} = w_{j}^{L,2}$

end for

return $w_{j}^{*}$

In these cases, linear classifiers can be effectively used and Model Selection is particularly challenging, as traditionally exploited approaches are known to be ineffective [3], [12], [13], [7], [14]. In these cases, properly tuning the hypothesis space is paramount to derive reliable and performing models: the purpose of this section is thus to show that we are able to remarkably reduce the complexity of the space without losing the possibility of representing the functions that are well-performing on the training and reference sets (as underlined in [11], [3]): these functions will be most likely chosen by the learning process and, then, there seem to be no reasons to search for more complex spaces. Moreover, note that few bits are required in order to represent these functions, thus contemplating an infinite precision space appears to be unmotivated by practical needs [37]. Results clearly show that the effect of a sparse and local hypothesis space on the measured complexity of the space itself opens rooms for further investigations, that deserve to be performed in future works. Broadly speaking, the approach is theoretically sound. In the SRM framework we have to search for the simplest hypothesis space that guaranties the best trade off between accuracy on the training set and complexity of the space. Then the introduction of a bit-based hypothesis space is also encouraged by the basic ML idea to search for the simplest class of functions capable of solving the problem under examination.

VII. CONCLUSIONS

In this paper we presented a procedure to define a bit-based hypothesis space, defined accordingly to locality and sparsity principles, and to train effective classifiers in this class of functions. The proposed approach clearly matches the main ideas which Structural Risk Minimization builds on, i.e. the idea of searching for the simplest hypothesis space that guarantees the best trade off between accuracy on the training set and complexity of the space.

Thank to the approach proposed as well as through the experimental results presented, we clearly show that the conventional approach of working with (virtually) infinite precision, using all the available features and contemplating the whole space of functions, is not just a waste of computational resources, but it is also a theoretically wrong approach to learning, as we have to look for the simplest solution to the problem under analysis.

APPENDIX

EXTENSION TO KERNELS

When we want to switch from linear separators to non-linear separators, the main trick consists in projecting, through proper mapping functions $\phi$, the input space $\mathbb{R}^d$ into another space $\mathbb{R}^D$ where usually $D \gg d$. Then by exploiting the well-known kernel trick $\phi(a) \cdot \phi(b) = K(a, b)$ [1], [8] we can avoid to compute or even know $\phi$, and we can simply work in the original space $\mathbb{R}^d$. 

In Table II we report the error on the reference set (RS) $L_{RS}$, performed by the chosen model, while in Table III we show the complexity of the space $\mathcal{M}_h(A)$ selected during the Model Selection phase by varying the hyperparameters $\kappa, \zeta$ and $\epsilon$. The set $D_e$ is generated by reserving 30% of the learning set analogously to [3]. 

In order to fairly compare the results we use the same realization of $\sigma$ when computing $\mathcal{M}(A)$. It is worth noting that we are able to remarkably reduce the complexity of the space without losing the possibility of representing the functions that are well-performing on the training and reference sets (as underlined in [11], [3]): these functions will be most likely chosen by the learning process and, then, there seem to be no reasons to search for more complex spaces. Moreover, note that few bits are required in order to represent these functions, thus contemplating an infinite precision space appears to be unmotivated by practical needs [37]. Results clearly show that the effect of a sparse and local hypothesis space on the measured complexity of the space itself opens rooms for further investigations, that deserve to be performed in future works. Broadly speaking, the approach is theoretically sound. In the SRM framework we have to search for the simplest hypothesis space that guaranties the best trade off between accuracy on the training set and complexity of the space. Then the introduction of a bit-based hypothesis space is also encouraged by the basic ML idea to search for the simplest class of functions capable of solving the problem under examination.

In this paper we presented a procedure to define a bit-based hypothesis space, defined accordingly to locality and sparsity principles, and to train effective classifiers in this class of functions. The proposed approach clearly matches the main ideas which Structural Risk Minimization builds on, i.e. the idea of searching for the simplest hypothesis space that guarantees the best trade off between accuracy on the training set and complexity of the space.

Thank to the approach proposed as well as through the experimental results presented, we clearly show that the conventional approach of working with (virtually) infinite precision, using all the available features and contemplating the whole space of functions, is not just a waste of computational resources, but it is also a theoretically wrong approach to learning, as we have to look for the simplest solution to the problem under analysis.
Table I
HUMAN GENE EXPRESSION DATASETS: CHARACTERISTICS AND SOURCE OF THE DATASET, AND MAPPING OF THE MULTI-CLASS INTO TWO-CLASS PROBLEMS.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Abbr.</th>
<th>Ref.</th>
<th>d</th>
<th>n</th>
<th>Class +1</th>
<th>Class −1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brain Tumor 1</td>
<td>D1</td>
<td>[43]</td>
<td>5920</td>
<td>90</td>
<td>Medulloblastoma</td>
<td>Malignant glioma, AT/RT, Normal cerebellum, PNET</td>
</tr>
<tr>
<td>Brain Tumor 2</td>
<td>D2</td>
<td>[43]</td>
<td>10367</td>
<td>50</td>
<td>Classic Glioblastomas and Anaplastic Oligodendrogliomas</td>
<td>Non-classic Glioblastomas and Anaplastic Oligodendrogliomas</td>
</tr>
<tr>
<td>Colon Cancer 1</td>
<td>D3</td>
<td>[45]</td>
<td>22263</td>
<td>47</td>
<td>Already two-class</td>
<td></td>
</tr>
<tr>
<td>Colon Cancer 2</td>
<td>D4</td>
<td>[46]</td>
<td>2000</td>
<td>62</td>
<td>Already two-class</td>
<td></td>
</tr>
<tr>
<td>DLBCL</td>
<td>D5</td>
<td>[43]</td>
<td>5469</td>
<td>77</td>
<td>Already two-class</td>
<td></td>
</tr>
<tr>
<td>Duke Breast Cancer</td>
<td>D6</td>
<td>[47]</td>
<td>7129</td>
<td>44</td>
<td>Already two-class</td>
<td></td>
</tr>
<tr>
<td>Leukemia</td>
<td>D7</td>
<td>[44]</td>
<td>7129</td>
<td>72</td>
<td>Already two-class</td>
<td></td>
</tr>
<tr>
<td>Leukemia1</td>
<td>D8</td>
<td>[43]</td>
<td>5327</td>
<td>72</td>
<td>ALL B-cell</td>
<td>ALL T-cell, AML</td>
</tr>
<tr>
<td>Leukemia 2</td>
<td>D9</td>
<td>[43]</td>
<td>11225</td>
<td>72</td>
<td>ALL</td>
<td>AML, MLL</td>
</tr>
<tr>
<td>Lung Cancer</td>
<td>D10</td>
<td>[43]</td>
<td>12600</td>
<td>209</td>
<td>Adeno</td>
<td>Normal, Squamous, COID, SMCL</td>
</tr>
<tr>
<td>Myeloma</td>
<td>D11</td>
<td>[48]</td>
<td>2157</td>
<td>105</td>
<td>Already two-class</td>
<td></td>
</tr>
<tr>
<td>Prostate Tumor</td>
<td>D12</td>
<td>[43]</td>
<td>10509</td>
<td>102</td>
<td>Already two-class</td>
<td></td>
</tr>
<tr>
<td>SRBTC</td>
<td>D13</td>
<td>[43]</td>
<td>23088</td>
<td>83</td>
<td>EWS</td>
<td>RMS, BL, NB</td>
</tr>
</tbody>
</table>

The purpose here is to extend Problems (15) and (22) so to allow using kernels. Consequently, we exploit the representer theorem [50] for defining \( w = \sum_{i=1}^{n} \alpha_i x_i \) and \( w \in \mathbb{R}^D \). Then we project \( x_i \rightarrow \phi(x_i) \) and we obtain \( w = \sum_{i=1}^{n} \alpha_i \phi(x_i) \). Since we could be unable to explicitly express \( \phi \), we can define a new hypothesis space \( H \) based on \( \alpha \). Then we can formulate, analogously to Problem (15), the following learning problem for training classifiers in a kernelized, bit-based, sparse and localized hypothesis space:

\[
\min_{\alpha, \xi} \sum_{i=1}^{n} \xi_i \quad \text{s.t.} \quad \begin{align*}
\mathbf{y}^T X \phi(x) & \geq 1 - \xi, \quad \xi \geq 0, \\
\mathbf{y}^T \phi(x) & \geq 1 - \xi', \quad \xi' \geq 0
\end{align*}
\]

where, once again, \( \phi \) does not appear explicitly. Problem (45) can be solved with the approach presented in Section V, for which, anyhow, we also need to derive a kernelized version of Problem (22). The reformulation is analogous to the one used for Problem (45):

\[
\min_{\alpha, \xi} \mathbf{y}^T \mathbf{y} \quad \text{s.t.} \quad \begin{align*}
\sum_{i=1}^{n} \alpha_i \xi_i & \leq \epsilon \\
\sum_{i=1}^{n} \alpha_i \xi_i & \leq \epsilon'
\end{align*}
\]

Note that we only need to define a kernel function in this case as well, as \( \alpha = \alpha^+ - \alpha^- \) (see Eq. (51)). Problem (52) can be easily solved through a standard Linear Programming optimization tool, as shown in Section V.

REFERENCES


<table>
<thead>
<tr>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>D5</th>
<th>D6</th>
<th>D7</th>
<th>D8</th>
<th>D9</th>
<th>D10</th>
<th>D11</th>
<th>D12</th>
<th>D13</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2</td>
<td>2.0</td>
<td>3.5</td>
<td>4.0</td>
<td>4.5</td>
<td>5.0</td>
<td>5.5</td>
<td>6.0</td>
<td>6.5</td>
<td>7.0</td>
<td>7.5</td>
<td>8.0</td>
<td>8.5</td>
</tr>
</tbody>
</table>

### TABLE II

RESULTS ON HUMAN GENE EXPRESSION DATASETS: $\epsilon$ (IN PERCENTAGE).

<table>
<thead>
<tr>
<th>$\epsilon$</th>
<th>$\epsilon$</th>
<th>$\epsilon$</th>
<th>$\epsilon$</th>
<th>$\epsilon$</th>
<th>$\epsilon$</th>
<th>$\epsilon$</th>
<th>$\epsilon$</th>
<th>$\epsilon$</th>
<th>$\epsilon$</th>
<th>$\epsilon$</th>
<th>$\epsilon$</th>
<th>$\epsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td>0.1</td>
<td>0.15</td>
<td>0.2</td>
<td>0.25</td>
<td>0.3</td>
<td>0.35</td>
<td>0.4</td>
<td>0.45</td>
<td>0.5</td>
<td>0.55</td>
<td>0.6</td>
<td>0.65</td>
</tr>
<tr>
<td>0.7</td>
<td>0.8</td>
<td>0.9</td>
<td>1.0</td>
<td>1.1</td>
<td>1.2</td>
<td>1.3</td>
<td>1.4</td>
<td>1.5</td>
<td>1.6</td>
<td>1.7</td>
<td>1.8</td>
<td>1.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$D_1$</th>
<th>$D_2$</th>
<th>$D_3$</th>
<th>$D_4$</th>
<th>$D_5$</th>
<th>$D_6$</th>
<th>$D_7$</th>
<th>$D_8$</th>
<th>$D_9$</th>
<th>$D_{10}$</th>
<th>$D_{11}$</th>
<th>$D_{12}$</th>
<th>$D_{13}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>1.5</td>
<td>2.0</td>
<td>2.5</td>
<td>3.0</td>
<td>3.5</td>
<td>4.0</td>
<td>4.5</td>
<td>5.0</td>
<td>5.5</td>
<td>6.0</td>
<td>6.5</td>
<td>7.0</td>
</tr>
</tbody>
</table>

$F$ = \text{function}$,
## TABLE III

### RESULTS ON HUMAN GENE EXPRESSION DATASETS: $\mathcal{M}_n(A)$ (IN PERCENTAGE).

<table>
<thead>
<tr>
<th>$D_n$</th>
<th>$\zeta \rightarrow \varepsilon$</th>
<th>$0.005$</th>
<th>$0.01$</th>
<th>$0.01$</th>
<th>$0.01$</th>
<th>$0.01$</th>
<th>$0.01$</th>
<th>$0.01$</th>
<th>$0.01$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$0.05$</td>
<td>$0.05$</td>
<td>$0.03$</td>
<td>$0.03$</td>
<td>$0.03$</td>
<td>$0.03$</td>
<td>$0.03$</td>
<td>$0.03$</td>
</tr>
<tr>
<td>D1</td>
<td>18.6</td>
<td>5.9</td>
<td>9.1</td>
<td>0.6</td>
<td>4.4</td>
<td>5.7</td>
<td>5.7</td>
<td>8.2</td>
<td>1.3</td>
</tr>
<tr>
<td>D2</td>
<td>20.2</td>
<td>1.8</td>
<td>1.7</td>
<td>29.3</td>
<td>1.6</td>
<td>1.6</td>
<td>0.8</td>
<td>1.5</td>
<td>0.9</td>
</tr>
<tr>
<td>D3</td>
<td>12.5</td>
<td>5.0</td>
<td>6.6</td>
<td>6.5</td>
<td>9.8</td>
<td>9.8</td>
<td>1.1</td>
<td>1.8</td>
<td>7.5</td>
</tr>
<tr>
<td>D4</td>
<td>7.5</td>
<td>5.0</td>
<td>5.3</td>
<td>34.4</td>
<td>25.5</td>
<td>1.3</td>
<td>6.7</td>
<td>8.7</td>
<td>4.5</td>
</tr>
<tr>
<td>D5</td>
<td>9.3</td>
<td>9.7</td>
<td>5.2</td>
<td>15.0</td>
<td>10.5</td>
<td>1.5</td>
<td>5.4</td>
<td>20.9</td>
<td>4.9</td>
</tr>
<tr>
<td>D6</td>
<td>7.5</td>
<td>9.9</td>
<td>4.3</td>
<td>6.6</td>
<td>5.7</td>
<td>2.2</td>
<td>9.3</td>
<td>7.5</td>
<td>1.2</td>
</tr>
<tr>
<td>D7</td>
<td>8.7</td>
<td>9.3</td>
<td>2.8</td>
<td>7.7</td>
<td>5.7</td>
<td>5.7</td>
<td>2.6</td>
<td>9.3</td>
<td>12.7</td>
</tr>
<tr>
<td>D8</td>
<td>20.9</td>
<td>3.1</td>
<td>10.7</td>
<td>10.6</td>
<td>8.0</td>
<td>1.6</td>
<td>9.4</td>
<td>9.8</td>
<td>5.7</td>
</tr>
<tr>
<td>D9</td>
<td>21.2</td>
<td>8.9</td>
<td>4.4</td>
<td>13.7</td>
<td>9.4</td>
<td>4.4</td>
<td>9.3</td>
<td>12.7</td>
<td>9.9</td>
</tr>
<tr>
<td>D10</td>
<td>21.1</td>
<td>8.9</td>
<td>4.4</td>
<td>13.7</td>
<td>9.4</td>
<td>4.4</td>
<td>9.3</td>
<td>12.7</td>
<td>9.9</td>
</tr>
<tr>
<td>D11</td>
<td>6.7</td>
<td>9.3</td>
<td>4.3</td>
<td>6.6</td>
<td>5.7</td>
<td>2.2</td>
<td>9.3</td>
<td>7.5</td>
<td>1.2</td>
</tr>
<tr>
<td>D12</td>
<td>6.0</td>
<td>9.9</td>
<td>4.3</td>
<td>6.6</td>
<td>5.7</td>
<td>2.2</td>
<td>9.3</td>
<td>7.5</td>
<td>1.2</td>
</tr>
</tbody>
</table>

... (Continued)