

Random Forest Kernel for High-Dimension Low Sample Size Classification

Lucca Portes Cavalheiro^{1,2*}, Simon Bernard¹, Jean Paul
Barddal² and Laurent Heutte^{1,2}

¹LITIS (UR 4108), Université de Rouen Normandie, F-76000
Rouen, France.

²Graduate Program in Informatics (PPGIA), Pontificia
Universidade Católica do Paraná (PUCPR), R. Imaculada
Conceição 1155, Curitiba, 80215-901, Paraná, Brazil.

*Corresponding author(s). E-mail(s): lucca54@gmail.com;
Contributing authors: simon.bernard@univ-rouen.fr;
jean.barddal@ppgia.pucpr.br; laurent.heutte@univ-rouen.fr;

Abstract

High dimension, low sample size (HDLSS) problems are numerous among real-world applications of machine learning. From medical images to text processing, traditional machine learning algorithms are usually unsuccessful in learning the best possible concept from such data. In a previous work, we proposed a dissimilarity-based approach for multi-view classification, the Random Forest Dissimilarity (RFD), that performs state-of-the-art results for such problems. In this work, we transpose the core principle of this approach to solving HDLSS classification problems, by using the RF similarity measure as a learned precomputed SVM kernel (RFSVM). We show that such a learned similarity measure is particularly suited and accurate for this classification context. Experiments conducted on 40 public HDLSS classification datasets, supported by rigorous statistical analyses, show that the RFSVM method outperforms existing methods for the majority of HDLSS problems and remains at the same time very competitive for low or non-HDLSS problems.

Keywords: High Dimension Low Sample Size, Classification, Random Forest, Similarity learning, SVM, Kernel

1 Introduction

In many modern machine learning problems, data are made available as small samples while being described in high dimensions. These datasets are generally referred to as “High Dimension, Low Sample Size” (HDLSS) datasets. These situations occur when the data are intrinsically complex and need to be described with many features, and when at the same time they are available only in potentially very limited quantities. Formally, a dataset T composed of $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\}$ instances, where \mathbf{x}_i is the vector of descriptive features belonging to \mathbb{R}^m and y_i is its corresponding class label, is considered to be a HDLSS dataset when $m \gg n$ [1, 2]. However, it should be noted that there is no consensus on the threshold to apply to the ratio between m and n to unambiguously decide whether a dataset is pertaining to an HDLSS learning problem [1–3].

HDLSS datasets are recurring in many real-world applications, including, but not limited to, medical imaging, DNA microarrays, and text processing [1, 4]. For instance, for pattern recognition tasks in medical imaging, numerical image representations are known to be high dimensional, whether they are built with hand-crafted features or with automatically learned deep features [5]. This is combined with the fact that acquiring samples for medical-related pattern recognition applications is not an easy task due to health data privacy policies, political concerns, or even the homogenization of medical protocols. In general, this means dealing with particularly small-sized datasets.

HDLSS datasets pose a number of challenges to general-purpose machine learning algorithms. These datasets usually embed very complex concepts due to a large number of relevant features, while they generally do not contain a sufficient number of instances for learning these concepts. This leads to several machine learning issues, well known to the community and often referred to as the curse of dimensionality. For example, for the many methods based on distance metrics, like the k -Nearest Neighbors methods, the notion of “neighborhood” becomes progressively ill-defined in high dimensions [6]. This is because the distances between instances become more and more similar when the dimension increases [7]. Another example of difficulties encountered in learning HDLSS data is the presence of outliers. Some methods are known to be particularly sensitive to outliers [8], and such data are quite common in high-dimensional feature spaces [6]. A last example of well-known learning problems often encountered in the HDLSS context is overfitting. In this context, many general-purpose machine learning techniques are likely to overfit. It is the case of SVM-based methods for example, for which the so-called data-piling phenomenon is particularly salient in the HDLSS context and leads to strong overfitting [9].

The most common approach in the literature for dealing with HDLSS learning tasks is based on dimensionality reduction. The key idea is to reduce the dimension to transform a high dimensional problem into a learning problem where $m \approx n$. However, these methods can be considered as workarounds

rather than true solutions for learning HDLSS datasets. Moreover, they often suffer from two major drawbacks:

- Using dimensionality reduction often result in a significant loss of information if the features are mostly relevant and poorly correlated [10, 11].
- Selecting a small subset of the most relevant features may not yield good results if the number of data is too small [11, 12].

In contrast, few methods have been proposed in the literature specifically to handle HDLSS learning tasks, which we review in Section 2. The most efficient ones are derived from the SVM principle, with a modified formulation of the underlying optimization problem in order to better adapt to the HDLSS specificities [11, 13]. However, we think that another promising approach is to use a HDLSS-compliant similarity measure as a kernel for SVM, instead of relying on a specific problem formulation. This idea has been recently applied to multi-view learning as in this context, similarity representations allow to easily merge the different views together [10]. This approach, named Random Forest Dissimilarity (RFD), leans on Random Forest classifiers to build dissimilarity representations that are then used as pre-computed kernels in SVM classifiers. We show in this paper that this principle can be straightforwardly and efficiently applied to HDLSS classification problems, for which we think it presents real assets: (i) Learning based on (dis)similarities between instances is a good way to deal with particularly small-sized datasets and (ii) the Random Forest (dis)similarity measure is known to be particularly robust to high dimensions. Therefore, this work presents:

- a transposition of the RFD method to HDLSS classification tasks with an emphasis on its strength to face the HDLSS challenges.
- a rigorous experimental validation, including comparisons with several state-of-the-art HDLSS learning methods on 40 real-world problems, along with a thorough statistical analysis of the results.

Note that for simplicity, we focus only on classification problems in this study. However, our proposal is straightforwardly applicable to regression tasks, as well as all the other methods used for comparison.

The remainder of this paper is organized as follows. Section 2 reviews the main state-of-the-art approaches for HDLSS learning. Section 3 presents the Random Forest SVM method and discuss its assets fro HDLSS classification. Section 4 describes the experimental setting, followed by the presentation and analysis of the results in Section 5. Finally, Section 6 gives our conclusion and future works.

2 Related Work

This section focuses on solutions to address the challenges posed by HDLSS learning tasks and provides an overview of the leading solutions that can

be found in the literature, with the exception of dimensionality reduction approaches, not included for reasons explained in the introduction.

2.1 Limitations of traditional methods for dealing with HDLSS problems

Traditional general-purpose machine learning techniques like Discriminant Analysis [14], k -Nearest Neighbors [15], and Support Vector Machines [9] usually fail to handle HDLSS datasets mainly due to the fact that such a context leads to ill-posed problems or to unsuitable learning conditions. For example, Linear Discriminant Analysis (LDA) suffers from a well-known problem when the dimension is larger than the number of training instances, i.e., $m \gg n$. The underlying principle of LDA is to find a projection of the data in which the between-class separability is maximized while the within-class variability is minimized. To do so, it uses a within-class scatter matrix that is known to be singular when $m \gg n$. This is an important problem since the non-singularity of this matrix is required to find the LDA basis vectors. The usual ways to circumvent this situation is either to perform dimensionality reduction beforehand (which is beyond the scope of this work as previously explained), or to use regularization techniques, as in the Regularized Discriminant Analysis [14] which has been successfully used for real-world HDLSS problems [16].

Distance-based classification techniques, like the k -Nearest Neighbors family of methods, are classifiers that usually perform well when $n > m$. However, they are also known to suffer from the curse of dimensionality in the opposite situation because the pairwise distances between all observations concentrate around a single value in this case [17]. Similarly as before, a large part of the solutions proposed in the literature are based on dimensionality reduction techniques, e.g., [18]. A few others, however, propose alternative mechanisms for dealing with HDLSS data like weighted voting scheme [19], fuzzy neighborhood [20], or new proximity measurements [21] to name a few.

Nevertheless, most state-of-the-art methods tailored for HDLSS classification in the literature are based on adaptations of SVM classifiers. SVM are known to be particularly prone to overfitting in the HDLSS context, which is often illustrated through the data-piling phenomenon [9]. The following section details this phenomenon as well as the different solutions that have been proposed in the literature.

2.2 SVM-inspired methods

In binary classification, the underlying principle of Support Vector Machines (SVM) [22] is to find a hyperplane that best separates the instances according to their classes by maximizing the distance (called the margin) to its closest instances (called the support vectors). The solution is the hyperplane whose

parameters that minimizes the following problem:

$$\begin{aligned}
 \min_{\mathbf{w}, b, \{\xi_i\}} & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=0}^n \xi_i & (1) \\
 \text{s.t.} & y_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, i = 1, \dots, n \\
 & \xi_i \geq 0, i = 1, \dots, n
 \end{aligned}$$

where \mathbf{w} is the normal vector of the hyperplane, b its intercept term, and where C is a regularization hyperparameter to control a trade-off between maximizing the margin and taking into account some training errors when the problem is not strictly linearly separable.

The behavior of this SVM method for HDLSS classification has been extensively analyzed and discussed in [9]. The authors show that in such a setting, SVM classifiers face the so-called *data-piling* problem. This problem arises specifically with HDLSS datasets because, in this case, a large proportion of the training instances are support vectors, i.e. they lie on the margin boundaries resulting from the minimization of Equation 5. When projected in the discriminant direction (i.e. to the normal vector \mathbf{w} obtained by minimizing Equation 5), all these support vectors "pile up on top of each other", i.e. they are projected onto exactly two points, one for each class. In this discriminant projection, the separating hyperplane is exactly halfway between these two points. Nevertheless, this usually reflects severe overfitting since independent test instances may not be projected the same way in the discriminant direction. As a result, while the resulting linear classifier perfectly fit (most of) the training instances, there is a strong risk that it will not generalize well to new data points. This phenomenon has been widely illustrated and analyzed in the literature and we refer the reader to [2, 9, 23] for further details.

Variations have been proposed to solve this problem, mainly by modifying the underlying optimization problem. For example, the Distance Weighted Discrimination (DWD) [9] leans on a minimization problem for which the best separating hyperplane is the one that maximizes the harmonic mean of all distances to the hyperplane:

$$\begin{aligned}
 \min_{\mathbf{w}, b, \{\xi_i\}} & \sum_{i=1}^n \left(\frac{1}{r_i} + C\xi_i \right) & (2) \\
 \text{s.t.} & r_i = y_i (\mathbf{w}^\top \mathbf{x}_i + b) + \xi_i, \\
 & r_i \geq 0, \xi_i \geq 0, \|\mathbf{w}\|^2 \leq 1
 \end{aligned}$$

In this process all training instances are taken into account to find the hyperplane, instead of relying only on the support vectors. A variant of this principle, named Weighted Distance Weighted Discrimination (wDWD) [23], has been proposed to allow for more flexibility and robustness, as the method is known to be sensitive to imbalanced classes and quickly become computationally expensive as the number of training instances increases [2].

Another SVM-inspired method is proposed in [2] named the Population-Guided Large Margin Classification (PGLMC) method, that aims to take up the idea of DWD while improving its computational performance. Equation 5 is modified to take into account the distance between the centroids of the classes (instead of all the training instances) to ensure the training instances of both classes are as far as possible along the projecting direction \mathbf{w} :

$$\min_{\mathbf{w}, b, \{\xi_i\}} \frac{\|\mathbf{w}\|^2}{(m_1 - m_2)^\top \mathbf{w}} + C \sum_{i=0}^n \xi_i \quad (3)$$

$$\begin{aligned} \text{s.t. } & y_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, i = 1, \dots, n \\ & (m_1 - m_2)^\top \mathbf{w} \geq 2 \\ & \xi_i \geq 0, i = 1, \dots, n \end{aligned} \quad (4)$$

where m_1 (resp. m_2) is the centroid of the first class (resp. the second class). Following a similar principle, a slightly different formulation is proposed in [11] that still maximize the distance between classes but that also strives to distribute the points as much as possible in the projection direction to avoid data-piling. The resulting method is named No-separated Data Maximum Dispersion classifier (NPDMD).

2.3 SVM with an HDLSS-robust kernel

The solutions mentioned above are all based on different formulations of the underlying optimization problem, whose solution leads to the final linear classifier. However, SVMs are known to allow the learning of non-linear classifiers using the kernel trick. Intuitively, it consists in applying a non-linear projection of the data into an *implicit* feature space in which a separating hyperplane exists. The key feature of this projection is that it does not require coordinates to be calculated explicitly, as all operations are performed through scalar products in this space, calculated with a kernel function.

Formally, this trick is based on the Lagrangian dual form of the SVM optimization problem:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \quad (5)$$

$$\text{s.c. } \alpha_i \geq 0, \quad i = 1, \dots, n \quad (6)$$

$$\sum_{i=1}^n \alpha_i y_i = 0 \quad (7)$$

where the α_i are the Lagrange multipliers. This dual form can be efficiently solved by quadratic programming algorithm, with the notable advantage of searching for n parameters (the Lagrange multipliers) instead of $m + 1$ parameters with the primal version (the m values of \mathbf{w} and b). This makes it

particularly suitable for HDLSS problems where $m \gg n$. This also allows to express the resulting classifier as a function of the support vectors:

$$h(\mathbf{x}) = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i^\top \mathbf{x} + b = \sum_{i=1}^n \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j) + b \quad (8)$$

As a consequence, all instances are only accessed through scalar product, allowing the use of any kernel functions K , as in the right-hand expression of Equation 8. We refer the reader to [24] for further details about the kernel trick, and kernel methods in general.

There are several popular kernels for SVM in the literature, but the best performing one for a wide range of real-world problems is the radial basis function (RBF) kernel, defined as:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2) \quad (9)$$

where $\mathbf{x}_i, \mathbf{x}_j$ is any pair of instances, and where γ is a hyperparameter. It is important to note that the behavior of the resulting model is highly sensitive to the value of γ and that it must be tuned in conjunction with the regularization parameter C . The traditionally recommended protocol for tuning these hyperparameters is detailed in Section 4.

The asymptotic behaviors of SVM with RBF kernel in the HDLSS context are further investigated in [3]. This study shows that nonlinear SVM classifiers with RBF kernels are highly biased in the HDLSS context, especially with imbalanced classes, and we therefore believe that using an HDLSS-robust kernel would be more relevant. For this, one can rely on the fact that the kernels can be directly interpreted as a similarity measurement [25]. In fact, any similarity measure can be used as kernel provided that it fulfills specific mathematical conditions [10, 26]. Therefore, using SVM for tackling the HDLSS classification challenges could be addressed by proposing a suitable similarity measure instead of modifying the underlying SVM optimization problem. This is the main idea of the method we propose to evaluate in this work and which is described in the next section.

3 The Random Forest-SVM method (RFSVM)

3.1 Using Random Forest as a kernel

Random Forest is a very versatile general-purpose learning method, that have shown to be accurate for many real-world problems [27]. These methods are also known to provide a number of mechanisms for analysis and interpretability, such as a similarity (or proximity) measure [10].

For computing the RF similarity between any pair of instances, one must have a previously trained RF classifier noted $H(\mathbf{x}) = \{h_k(\mathbf{x}) \mid 1 \leq k \leq M\}$, made up with M decision trees h_k . Any RF learning algorithm can be used for that purpose, as the similarity measurement leans on the final ensemble of

decision trees grown during learning. Note that for all experiments of this work, the Breiman’s Random Forest method [28] has been used, via the implementation proposed in the Scikit-learn python library [29]. The similarity between two instances $(\mathbf{x}_i, \mathbf{x}_j)$ is inferred by the forest by comparing the descending paths followed by both instances in each tree: let \mathcal{L}_k denote the set of leaves of h_k , and let $l_k(\mathbf{x})$ denote a function from the input domain \mathcal{X} to \mathcal{L}_k , that returns the leaf of h_k where \mathbf{x} lands when one wants to predict its class. The similarity measure s_k is defined as in Equation 10: if the two instances \mathbf{x}_i and \mathbf{x}_j land in the same leaf of h_k , then the similarity between both instances is set to 1, else it is equal to 0.

$$s_k(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} 1, & \text{if } l_k(\mathbf{x}_i) = l_k(\mathbf{x}_j) \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

The RF similarity measure $s_H(\mathbf{x}_i, \mathbf{x}_j)$ derived from the whole forest H consists in calculating s_k for each tree h_k , and in averaging the resulting values over the M trees:

$$s_H(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{M} \sum_{k=1}^M s_k(\mathbf{x}_i, \mathbf{x}_j) \quad (11)$$

A remarkable property of this similarity measure is that it can be assimilated to a positive semi-definite (p.s.d.) kernel function [10]. Similarly, one can show that the similarity matrix \mathbf{S}_H :

$$\mathbf{S}_H = \begin{bmatrix} s_H(x_1, x_1) & s_H(x_1, x_2) & \dots & s_H(x_1, x_n) \\ s_H(x_2, x_1) & s_H(x_2, x_2) & \dots & s_H(x_2, x_n) \\ \dots & \dots & \dots & \dots \\ s_H(x_n, x_1) & s_H(x_n, x_2) & \dots & s_H(x_n, x_n) \end{bmatrix} \quad (12)$$

whose elements are the RF similarity measures between each pair of training instances, is a positive semi-definite matrix (see the proof in Appendix A of [10]). As a consequence, such a matrix can be used in a kernel method as a pre-computed kernel.

In a nutshell, using such a pre-computed kernel with SVM classifier, noted RFSVM in the following, consists of:

1. Training a Random Forest classifier H on the training set T ;
2. Building a similarity matrix \mathbf{S}_H from H ;
3. Feeding an SVM classifier with \mathbf{S}_H as a pre-computed kernel.

As for the prediction phase, once the RFSVM classifier is trained this way, an unseen testing instance \mathbf{x} can be predicted as follows:

1. The similarity values between \mathbf{x} and all the n training instances from T are computed, leading to a n -sized vector of $s_H(\mathbf{x}, \mathbf{x}_i), i = 1, \dots, n$;
2. This vector is given as input to the RFSVM classifier for prediction.

The whole learning procedure is further detailed in Algorithm 1. Note that the Random Tree building method used in this work is the CART-based implementation from the Scikit-learn library [29] and the SVM solver used in this work is the LIBSVM solver that implements the sequential minimal optimization (SMO) algorithm for kernelized SVM [30].

Algorithm 1: The RFSVM learning method

```

input :  $T$ , a training set composed of  $n$  instances  $(\mathbf{x}_i, y_i)$ 
input :  $M$ , the number of trees in the random forest
input :  $\Theta$ , the random tree learning hyperparameter values
input :  $C$ , the SVM regularization hyperparameter value
output:  $SVM_H$ , a RFSVM model trained on  $T$ 

begin
    // 1. Train a random forest composed of  $M$  random trees
     $H \leftarrow \emptyset$ 
    for  $k = 0$  to  $M$  do
         $T_k \leftarrow \text{BootstrapSampling}(T)$ 
         $h_k \leftarrow \text{RandomTree}(T_k, \Theta)$ 
         $H \leftarrow H \cup h_k$ 
    end
    // Note: the bootstrap sampling method and the random tree method used in
    // this work are those of the random forest implementation of the Scikit-learn
    // library [29]

    // 2. Compute the similarity matrix from the random forest
     $S_H \leftarrow I_n$  /* init.  $S_H$  to a  $n \times n$  identity matrix */
    for  $i = 1$  to  $n - 1$  do /* for each pair  $\mathbf{x}_i, \mathbf{x}_j \in T$  */
        for  $j = i + 1$  to  $n$  do /* for each tree  $h_k \in H$  */
            for  $k = 1$  to  $M$  do
                 $l_i \leftarrow \text{getLeaf}(h_k, \mathbf{x}_i)$  /* get the leaf of  $h_k$  where  $\mathbf{x}_i$  lands */
                 $l_j \leftarrow \text{getLeaf}(h_k, \mathbf{x}_j)$  /* get the leaf of  $h_k$  where  $\mathbf{x}_j$  lands */
                if  $l_i = l_j$  then
                    |  $S_H(i, j) \leftarrow S_H(i, j) + 1$ 
                end
            end
             $S_H(i, j) \leftarrow S_H(i, j) / M$ 
             $S_H(j, i) \leftarrow S_H(i, j)$ 
        end
    end
    // 3. Use  $S_H$  as a precomputed kernel for the SVM learning
     $SVM_H \leftarrow \text{LIBSVM Solver}(S_H, C)$ 
    // Note: the LIBSVM solver used in this work implements the sequential minimal
    // optimization (SMO) algorithm [30]
end
    
```

3.2 Discussion

It is worth noting that the RFSVM procedure can be applied with any similarity measure provided that the resulting similarity matrix is p.s.d. For example, the well-known cosine measure could be used to replace the Random Forest similarity measure. However, we would like to stress that the Random Forest similarity measure is particularly suitable to do so for classification tasks. The reason is that it is computed in such a way it can well reflect the class belonging of the instances. According to this measure, two instances that belong to

the same class are more likely to be similar than two instances from different classes. This is due to the fact that the Random Forest classifier is built beforehand by taking the class into account so that the leaves of all the trees are expected to gather instances from the same class.

Similarly, metric learning methods could replace the RF classifier for inferring the similarity values. However, we argue that the state-of-the-art metric learning methods require a formulation of the metric beforehand, which is not the case of RF methods, and more importantly, they are known to be sensitive to high dimensions. For example, most of the state-of-the-art metric learning methods that are based on the Mahalanobis distance suffer from two important drawbacks for HDLSS classification:

1. They are computationally intractable in high dimensions since the number of parameters to learn (the covariance matrix elements) is $\mathcal{O}(m^2)$.
2. They face a strong risk of overfitting phenomenon since the number of training instances used to estimate these parameters is very low in the HDLSS context.

Consequently, we argue that classical metric learning methods are not suitable for HDLSS problems, contrary to RF classifiers, known to be very robust to high dimensions without the need for a proportionally large training set. To support this statement, the most popular metric learning method, namely Large Margin Nearest Neighbors (LMNN) [31], has been included in the experiments as an alternative to RF classifiers for building a pre-computed kernel in SVM. The resulting method is named LMNNSVM in the following.

The RFSVM method explained in the previous section has been successfully applied to multi-view classification problems [10, 32, 33]. These studies have mainly focused on using RF (dis)similarity matrices on each view and fuse them in order to take benefit from the complementarity between the different views. However, the extent to which the success of this approach depends on how well it exploits complementarities in multi-view learning or on the use of the RF similarity measure itself has not yet been studied. Therefore, the main contribution of this work is to extend the experimental study and validate this approach to regular HDLSS single-view classification problems.

4 Experiments

This section details the experimental protocol, as well as its underlying goal of evaluating how the proposed method compares to general-purpose machine learning methods, similarity-based methods, and HDLSS-specific methods for a variety of datasets that exhibit different levels of HDLSS.

4.1 Datasets

Our experimentation encompasses 40 public datasets, among which 21 datasets were acquired from the OpenML repository [34], 3 text processing datasets from the UCI repository [35], and 16 medical datasets from [36].

In the literature, the only consensus to formally define an HDLSS problem is “ $m \gg n$ ” [1–3]. Therefore, *HDLSSness* could be measured by the ratio between the number of instances and the number of features in the associated dataset. However, this does not take into account the number and imbalance of classes, which can have a major impact on the difficulties arising from HDLSS learning. Therefore, we propose to quantify a level of HDLSS for each dataset, with the measure defined as:

$$\Omega = \frac{1}{m} \times \frac{\sum_{j=1}^c n_j}{c}, \quad (13)$$

where c is the number of classes, m is the number of features, and n_j is the number of instances from the j -th class in a dataset. This Ω measure corresponds to the average number of instances per class divided by the number of features in a dataset. Consequently, the smaller the value of Ω , the more HDLSS a dataset is. Using the average number of instances per class, instead of the total number of instances in the dataset, allows to limit the bias introduced by strong class imbalances, such as those found in some of the datasets we selected in our experiments.

Table 1 gives a description of all the datasets used in our experiments, with the number of instances, the number of features, the imbalance ratio (IR), the number of classes, and the Ω value. The IR is computed by dividing the number of instances from the majority class by the number of instances from the minority class. In this table, datasets are sorted by increasing value of Ω . One can observe that as Ω increases, the number of instances also increases, and the dimensionality decreases. This table also contains a separation between HDLSS datasets in the upper part and non-HDLSS datasets in the lower part, corresponding to values of $\Omega = 1$. This allows to highlight that these datasets have been chosen in a wide range of cases, including datasets corresponding to traditional classification problems. This will also allow us to show that the RFD method remains competitive in a more classical learning context.

4.2 Methods and parametrization

In addition to the RFSVM method, several learning methods were selected for comparison, in three families of methods: general-purpose methods, SVM variants, and similarity-based methods. In the first group, the Random Forest classifier and the Extreme Gradient Boosting (XGBoost) method were selected because of their state-of-the-art performance on various ML problems [27, 28, 37]. Regarding SVM variants, the regular method with RBF kernel [22] has been retained, as well as the DWD method presented in Section 2. The reasons why the DWD methods has been retained in our experiments instead of the other SVM variants listed in Section 2 are (i) because the results obtained in [11] from the experimental comparison between all these methods show very similar results without any statistical test of significance supporting the superiority of one method over the others and (ii) because it is the only one

Table 1 Datasets description with the number of instances, the number of features, the imbalance ratio (IR), the number of classes, and the Ω value. Datasets with $\Omega < 1$ are considered to be HDLSS datasets, whereas $\Omega \geq 1$ are non-HDLSS datasets.

Name	Instances	Features	IR	Classes	Ω
UMIST_Faces_Cropped [34]	575	10304	2.526	20	0.003
leukemia [34]	72	7129	1.88	2	0.005
alizadeh-2000-v3 [36]	62	2091	2.333	4	0.007
tr45.wc [34]	690	8261	11.429	10	0.008
laiho-2007 [36]	37	2202	3.625	2	0.008
bittner-2000 [36]	38	2201	1.0	2	0.009
arcene [34]	200	10000	1.273	2	0.010
ramaswamy-2001 [36]	190	1363	3.0	14	0.010
armstrong-2002-v2 [36]	72	2194	1.4	3	0.011
su-2001 [36]	174	1571	4.667	10	0.011
lapointe-2004-v2 [36]	110	2496	3.727	4	0.011
golub-1999-v2 [36]	72	1868	4.222	3	0.013
Dexter [34]	600	20000	1.0	2	0.015
yeoh-2002-v2 [36]	248	2526	5.267	6	0.016
tomlins-2006-v2 [36]	92	1288	2.462	4	0.018
khan-2001 [36]	83	1069	2.636	4	0.019
west-2001 [36]	49	1198	1.042	2	0.020
eating [34]	945	6373	1.176	7	0.021
bhattacharjee-2001 [36]	203	1543	23.167	5	0.026
micro-mass [34]	360	1300	1.0	10	0.028
oh15.wc [34]	913	3100	2.962	10	0.029
oh10.wc [34]	1050	3238	3.173	10	0.032
shipp-2002-v1 [36]	77	798	3.053	2	0.048
cnae-9-half [34]	540	856	1.283	9	0.070
OVA_Colon [34]	1545	10935	4.402	2	0.071
OVA_Breast [34]	1545	10935	3.491	2	0.071
imdb [35]	748	3047	1.066	2	0.123
cnae-9 [34]	1080	856	1.0	9	0.140
lsvt [34]	126	310	2.0	2	0.203
yelp [35]	1000	2033	1.0	2	0.246
amazon [35]	1000	1847	1.0	2	0.271
chowdary-2006 [36]	104	182	1.476	2	0.286
chen-2002 [36]	179	85	1.387	2	1.053
gina [34]	3153	970	1.034	2	1.625
madelon [34]	2600	500	1.0	2	2.600
scene [34]	2407	299	4.585	2	4.025
wdbc [34]	569	30	1.684	2	9.483
led24 [34]	3200	24	1.139	10	13.333
segment [34]	2310	19	1.0	7	17.368
spambase [34]	4601	57	1.538	2	40.360

with a freely available implementations. As for similarity-based methods, the cosine distance and the LMNN variant of the Mahalanobis distance were used in the same way as with the RFSVM method, i.e. as a precomputed kernel, to support the claims given in the discussion subsection of the Section 3.

For each dataset, all classifiers had their hyperparameters tuned using a 3-fold cross-validation procedure. The tuning process was performed using the `hyperopt` library [38]. This library requires as input a search space for each hyperparameter and the number of evaluations for the tuning process. In the following experiments, the number of evaluations was set to 100, and the optimization criterion was accuracy maximization. Internally, `hyperopt` conducts hyperparameter optimization by converting the search process into a generative process using Tree-structured Parzen Trees (TPEs) [39]. We refer the reader to [38] for more details about its functioning.

Regarding the Random Forest method, the maximum tree depth¹, the maximum number of features assessed for a split decision, the minimum number of samples at leaf nodes, the minimum number of samples for a split, and the number of trees have been tuned following the values given in the upper part of Table 2. The search space for XGBoost is given in the middle part of the

¹The `None` option stands for fully grown trees, i.e., the trees are grown until all leaf nodes have pure class distributions.

same table. XGBoost search space follows the suggestion given in [40], where the maximum tree depth, instance subsample ratio, column sample by tree portion, regularization lambda, and the maximum number of iterations were tuned. Finally, the search space for SVM with the gaussian kernel is depicted in the bottom part of Table 2, where the hyperparameter C and γ were optimized. For all other SVM-based methods, i.e. RFSVM, DWD, COSSVM and LMNNSVM, only C needs to be optimized, and it has also been done following the values in Table 2.

Table 2 Hyperparameter search space for Random Forest, XGBoost, SVM, and other SVM-based methods (RFSVM, COSSVM, LMNNSVM, and DWD).

Random Forest	
Max. Depth	$\{10^i i = 1, \dots, 10\}$ and None
Max. Features	$\{1\%, 5\%, 10\%, 20\%, 30\%\}$
Min. Samples Leaf	$\{1, 2, 4\}$
Min. Samples Split	$\{2, 5, 10\}$
Number of trees	500
XGBoost	
Max. Depth	$\{x x \in \mathbb{N}, 4 \leq x \leq 15\}$
Subsample	$[0.8, 1]$
Column sample by tree	$[0.5, 1]$
Regularization Lambda	$[0, 1]$
Max. number of iterations	500
SVM	
C	$\{10^i i = -2, \dots, 4\}$
γ	$\{10^i i = -4, \dots, 2\}$
RFSVM, COSSVM, LMNNSVM, and DWD	
C	$\{10^i i = -2, \dots, 4\}$

4.3 Validation protocol and implementation details

For each dataset used in this experimental comparison, all the methods were tested 10 times, each time with a random half of the dataset for training and the remaining half for test. The hyperparameter setting procedure was performed on the training set and the performance was measured with the traditional accuracy measurement.

Two different statistical tests of significance have been used to analyze the differences in performance between the methods. The tests used are (i) the Friedman test along with the Nemenyi post-hoc test [41], and (ii) the Bayesian sign test [42]. The Friedman/Nemenyi test is classically used to assess the statistical significance of a comparison of several methods over multiple datasets, based on the average ranks of the methods. We refer the reader to [41] for more details about its functioning and the reasons why this test is advised for this type of comparisons. In contrast, the Bayesian sign test is a pairwise test based on the difference in performance of the two methods over multiple datasets. Unlike frequentist null-hypothesis tests such as the Friedman test, Bayesian analysis can provide more insight than simply rejecting the hypothesis that the two classifiers are of equivalent performance. In particular, it outputs probabilities of one classifier a to be practically better than another classifier b , based on a set of results from multiple datasets. It also allows to

integrate in the analysis a “region of practical equivalence” (*rope*) defined by an average performance gap at which the classifier a is considered practically better than the classifier b . For instance, with $rope = 0.05$, if the mean accuracy of classifier a is 0.980 and the mean accuracy of classifier b is 0.976, both classifier will be considered to be practically equivalent in the Bayesian analysis because the difference $0.980 - 0.976 = 0.04$ is less than the *rope*. In our experiments, we considered two values for *rope*, 0.005 and 0.01.

Finally, all the experiments were executed in Python. Random Forest (RF), SVM, and cosine distance implementations were those available in the *scikit-learn* library [29] version 0.23.2. The implementation of DWD was found in the IDC9 repository², and the PYLMNN library [43] version 1.6.4 was used for computing the LMNN distance. XGBoost was performed using the implementation provided in its original paper (version 1.1.1).

5 Results and Discussion

Table 3 presents the mean accuracy rates (and standard deviations) obtained by the seven methods across all 40 datasets. The values in bold are the best mean accuracy rates obtained for each of the datasets. In this table, the datasets are separated into three groups: the very-HDLSS datasets at the top (when $\Omega < 0.015$), the mid-HDLSS datasets in the middle (when $0.015 \leq \Omega < 1$), and non-HDLSS datasets at the bottom. In this section, we analyze these results first globally and then in more detail for each of these three groups.

5.1 Analysis of overall results

The first observation that can be made from Table 3 is that most of the values in bold, the best mean accuracy rates, are obtained by the methods on the right-hand side of the table, that is to say the kernel based SVM. More precisely, as shown in the last row of Table 3, RFSVM is the method that achieves the best performance on most of the datasets (on 16 of the 40 datasets), followed by COSSVM (on 7 of the 40 datasets), and LMNNSVM (on 5 of the 40 datasets, *ex aequo* with Random Forest).

For a more precise analysis of these global results, it is necessary to look at the results of post-hoc statistical tests. Figure 1 shows the Critical Difference diagram [41] drawn from the results of the Friedman/Nemenyi test. This diagram sorts the seven methods by their average rank across the 40 datasets and indicates whether the difference between them is statistically significant: methods that are connected by a thick line are the one for which the statistical test does not reject the null-hypothesis that the classifiers perform equally well. RFSVM and COSSVM are the two best methods on average but closely followed by Random Forest. Considering statistical significance, RFSVM is the

²<https://github.com/idc9/dwd>

Table 3 Mean accuracy rates (along with standard deviations) obtained by the seven methods on all the 40 datasets. Bold values are the best mean accuracy rates obtained on each dataset.

Dataset	XGB	RF	SVM	DWD	RFSVM	COSSVM	LMNNSVM
UMIST	0.928 ± 0.016	0.985 ± 0.013	0.365 ± 0.402	0.948 ± 0.009	0.988 ± 0.010	0.970 ± 0.011	0.989 ± 0.006
leukemia	0.944 ± 0.045	0.964 ± 0.031	0.950 ± 0.051	0.922 ± 0.057	0.969 ± 0.023	0.961 ± 0.045	0.956 ± 0.052
aliz	0.771 ± 0.071	0.877 ± 0.064	0.890 ± 0.060	0.897 ± 0.061	0.897 ± 0.054	0.906 ± 0.053	0.903 ± 0.050
tr45_wc	0.970 ± 0.008	0.949 ± 0.012	0.815 ± 0.070	0.665 ± 0.107	0.954 ± 0.007	0.925 ± 0.008	0.900 ± 0.046
laho-2007	0.805 ± 0.024	0.805 ± 0.034	0.874 ± 0.059	0.826 ± 0.041	0.858 ± 0.041	0.879 ± 0.058	0.911 ± 0.041
bittner-2000	0.716 ± 0.111	0.784 ± 0.044	0.753 ± 0.097	0.816 ± 0.035	0.784 ± 0.050	0.837 ± 0.044	0.800 ± 0.039
arcene	0.772 ± 0.026	0.767 ± 0.043	0.633 ± 0.116	0.820 ± 0.036	0.807 ± 0.038	0.835 ± 0.045	0.853 ± 0.042
ramas	0.705 ± 0.037	0.751 ± 0.027	0.635 ± 0.038	0.669 ± 0.035	0.773 ± 0.021	0.760 ± 0.042	0.617 ± 0.026
arms	0.942 ± 0.032	0.975 ± 0.015	0.928 ± 0.042	0.961 ± 0.025	0.972 ± 0.018	0.975 ± 0.015	0.942 ± 0.049
su-2001	0.895 ± 0.026	0.883 ± 0.027	0.897 ± 0.021	0.883 ± 0.025	0.920 ± 0.024	0.894 ± 0.025	0.897 ± 0.027
lapo	0.804 ± 0.044	0.809 ± 0.030	0.818 ± 0.047	0.815 ± 0.041	0.851 ± 0.039	0.833 ± 0.034	0.853 ± 0.043
gol	0.939 ± 0.055	0.931 ± 0.031	0.881 ± 0.071	0.864 ± 0.040	0.944 ± 0.030	0.903 ± 0.031	0.914 ± 0.019
Dexter	0.916 ± 0.015	0.925 ± 0.010	0.869 ± 0.030	0.916 ± 0.010	0.934 ± 0.013	0.936 ± 0.006	0.927 ± 0.013
yeoh	0.842 ± 0.018	0.809 ± 0.015	0.787 ± 0.018	0.715 ± 0.022	0.848 ± 0.024	0.718 ± 0.023	0.815 ± 0.031
toni	0.715 ± 0.052	0.730 ± 0.050	0.811 ± 0.071	0.798 ± 0.046	0.763 ± 0.037	0.837 ± 0.049	0.859 ± 0.044
khan-2001	0.960 ± 0.035	0.983 ± 0.015	0.955 ± 0.044	0.955 ± 0.034	0.979 ± 0.020	0.957 ± 0.030	0.974 ± 0.033
west-2001	0.860 ± 0.041	0.884 ± 0.045	0.860 ± 0.048	0.856 ± 0.057	0.880 ± 0.040	0.860 ± 0.057	0.864 ± 0.060
eating	0.560 ± 0.022	0.532 ± 0.025	0.148 ± 0.000	0.551 ± 0.027	0.556 ± 0.018	0.403 ± 0.206	0.300 ± 0.159
bhatt	0.946 ± 0.014	0.929 ± 0.016	0.937 ± 0.017	0.923 ± 0.022	0.957 ± 0.019	0.936 ± 0.018	0.934 ± 0.017
micro	0.929 ± 0.032	0.931 ± 0.023	0.429 ± 0.404	0.912 ± 0.040	0.937 ± 0.024	0.908 ± 0.020	0.904 ± 0.027
oh15_wc	0.824 ± 0.014	0.825 ± 0.008	0.742 ± 0.041	0.558 ± 0.170	0.835 ± 0.006	0.803 ± 0.025	0.763 ± 0.010
oh10_wc	0.831 ± 0.012	0.836 ± 0.013	0.759 ± 0.021	0.654 ± 0.103	0.835 ± 0.017	0.765 ± 0.019	0.736 ± 0.014
ship	0.841 ± 0.057	0.828 ± 0.051	0.890 ± 0.066	0.856 ± 0.071	0.882 ± 0.045	0.921 ± 0.044	0.910 ± 0.038
cnac-9H	0.879 ± 0.021	0.915 ± 0.021	0.888 ± 0.020	0.862 ± 0.034	0.893 ± 0.023	0.914 ± 0.013	0.818 ± 0.027
OVA_C	0.976 ± 0.003	0.974 ± 0.003	0.953 ± 0.046	0.968 ± 0.004	0.976 ± 0.003	0.966 ± 0.005	0.963 ± 0.003
OVA_B	0.972 ± 0.005	0.968 ± 0.006	0.816 ± 0.078	0.970 ± 0.003	0.972 ± 0.005	0.969 ± 0.004	0.964 ± 0.004
imdb	0.626 ± 0.019	0.701 ± 0.021	0.702 ± 0.025	0.633 ± 0.082	0.695 ± 0.027	0.710 ± 0.018	0.689 ± 0.025
cnac-9	0.908 ± 0.012	0.932 ± 0.009	0.917 ± 0.013	0.861 ± 0.020	0.909 ± 0.009	0.935 ± 0.010	0.861 ± 0.021
lsvt	0.832 ± 0.038	0.811 ± 0.056	0.689 ± 0.067	0.871 ± 0.017	0.833 ± 0.048	0.749 ± 0.103	0.660 ± 0.035
yselp	0.709 ± 0.018	0.765 ± 0.017	0.767 ± 0.027	0.755 ± 0.036	0.776 ± 0.022	0.771 ± 0.024	0.741 ± 0.018
amazon	0.725 ± 0.019	0.789 ± 0.011	0.802 ± 0.011	0.798 ± 0.023	0.786 ± 0.013	0.795 ± 0.014	0.771 ± 0.012
chow	0.948 ± 0.030	0.963 ± 0.013	0.963 ± 0.023	0.977 ± 0.021	0.969 ± 0.018	0.981 ± 0.012	0.915 ± 0.030
chen-2002	0.916 ± 0.027	0.937 ± 0.031	0.931 ± 0.030	0.938 ± 0.015	0.942 ± 0.035	0.928 ± 0.013	0.887 ± 0.031
gina	0.939 ± 0.005	0.925 ± 0.005	0.509 ± 0.000	0.852 ± 0.004	0.947 ± 0.004	0.860 ± 0.007	0.844 ± 0.030
madelon	0.760 ± 0.018	0.769 ± 0.017	0.672 ± 0.009	0.595 ± 0.008	0.799 ± 0.014	0.596 ± 0.015	0.600 ± 0.026
scene	0.980 ± 0.005	0.927 ± 0.008	0.990 ± 0.002	0.948 ± 0.042	0.954 ± 0.009	0.989 ± 0.002	0.973 ± 0.004
wdbc	0.968 ± 0.009	0.963 ± 0.011	0.937 ± 0.019	0.978 ± 0.004	0.966 ± 0.010	0.944 ± 0.012	0.956 ± 0.009
led24	0.701 ± 0.005	0.724 ± 0.007	0.717 ± 0.007	0.729 ± 0.007	0.714 ± 0.006	0.719 ± 0.006	0.719 ± 0.007
segment	0.974 ± 0.005	0.970 ± 0.004	0.958 ± 0.007	0.928 ± 0.006	0.976 ± 0.003	0.956 ± 0.004	0.973 ± 0.005
spam	0.952 ± 0.004	0.951 ± 0.003	0.915 ± 0.014	0.884 ± 0.007	0.954 ± 0.005	0.898 ± 0.022	0.939 ± 0.006
# of wins	4	5	2	3	16	7	5

method for which the difference is statistically significant with the most methods. The differences in performance between all other methods are globally not significant according to the Friedman/Nemenyi test.

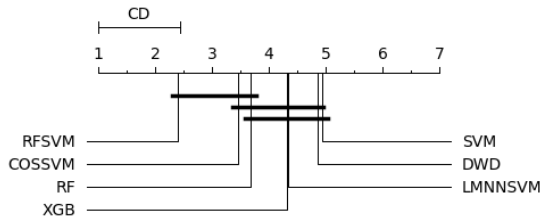


Fig. 1 Critical difference diagram from the Friedman/Nemenyi test results on all the 40 datasets.

These first overall results confirm that RFSVM is particularly relevant for HDLSS classification. However, this requires further confirmation, with careful analysis of the results obtained on the HDLSS datasets, which we give in the following section.

5.2 Analysis of the results on HDLSS datasets

Since this work focuses on HDLSS classification, we now deepen the analysis for the HDLSS datasets, that is to say, for the datasets with $\Omega < 1.0$. Figure 2 gives the Critical Difference diagram obtained by considering the HDLSS datasets only, i.e. 32 of the 40 datasets. The main difference one can observe from this diagram, compared to the one given in Figure 1, is that LMNNSVM performs slightly better. This supports the conclusion that the kernel based approach are generally effective for HDLSS classification. It can also be noted that, in contrast, the performance of DWD is surprisingly poor compared to general-purpose machine learning methods. On the other hand, in line with the analysis given in [9], SVM is the least successful method for these tasks.

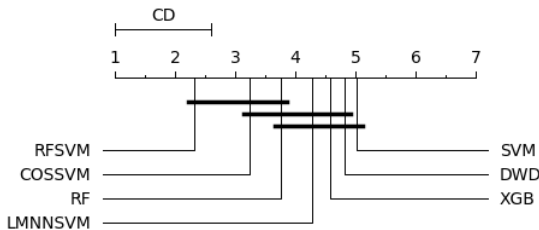


Fig. 2 Critical difference diagram from the Friedman/Nemenyi test results on the HDLSS datasets ($\Omega < 1.0$).

The results of the Bayesian test can now be used to make a more detailed comparison, by giving the probabilities that one classifier is more accurate than another. In the following, $p(a > b)$ denotes the probability for the classifier a to be more accurate than the classifier b according to the Bayesian test. Similarly, $p(a \sim b)$ denotes the probability that classifiers a and b perform equally well. In a nutshell, to estimate these probabilities, the Bayesian test uses the mean differences in accuracy between classifiers a and b on all the datasets, and deduces a distribution for $p(a > b)$, $p(a \sim b)$ and $p(b > a)$. We refer the reader to [42] for more details on how this distribution is obtained. Then M trinomial vectors of probabilities are drawn at random from this distribution. These vectors are typically represented as points in the simplex having vertices $\{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$. Three examples of such representations are given in Figure 3, for three pairwise comparisons, RFSVM vs RF, RFSVM vs COSSVM and SVM vs DWD, on the HDLSS datasets. These representations allow to observe the proportion of points falling in each of the three zones corresponding to each of the three situations. By considering the number of points that fall in the three regions, one can have an estimate of all three probabilities, $p(a > b)$, $p(a \sim b)$ and $p(b > a)$.

For our experimental comparison, these estimates are given as two color maps (the left one for $rope = 0.005$ and the right one for $rope = 0.01$) in Figure 4. It should be read as follows: the value in the cell for the row of classifier a and the column of classifier b is $p(a > b)$ according to the Bayesian test. Note that

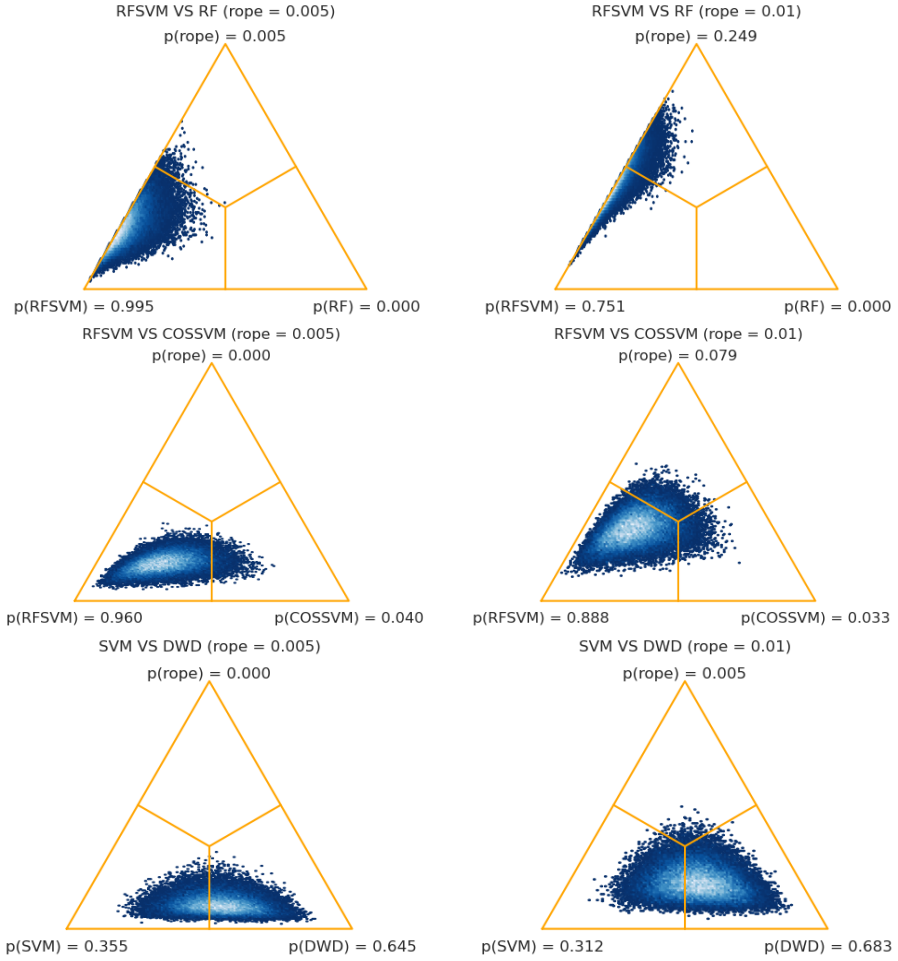


Fig. 3 Examples of trinomial vectors visual representations for the Bayesian analysis on the 32 HDLSS datasets.

the difference between $p(a > b)$ and $p(b > a)$ corresponds to $p(a \sim b)$, which is not given in these color maps. Therefore, smaller values of both $p(a > b)$ and $p(b > a)$ are expected with larger *rope* values. However, it should be noted that this analysis is subject to random draw, and so for cases where the differences in performance between a and b are larger than the *rope*, the opposite behavior may be observed, i.e., marginally larger probabilities occur with larger *rope* values.

In both color maps of Figure 4, the column corresponding to the RFSVM method is the one with the lowest values. This means that overall, it is the method with the lowest probabilities to be outperformed by any other method. Thus, although RFSVM, COSSVM, and RF are not significantly different

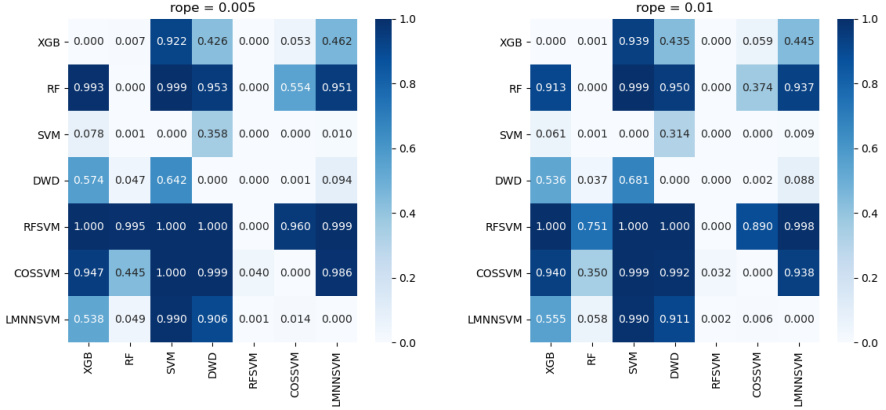


Fig. 4 Pairwise Bayesian analysis for HDLSS datasets. The value in each cell is $p(a > b)$, i.e. the probability that the row classifier a outperforms the column classifier b .

according to the Friedman/Nemenyi post-hoc test, the detailed pairwise analysis shows that both COSSVM and RF have very low probabilities of giving better results than RFSVM. Conversely, RFSVM has a high probability of being better than the cited methods. As for the comparison between SVM and DWD, these results confirm that the DWD method gives slightly better results on HDLSS datasets than the regular SVM method but with $p(\text{DWD} > \text{SVM})$ and $p(\text{SVM} > \text{DWD})$ being very close to 0.5 each.

Note that for $\text{rope} = 0.01$, most of the probabilities decrease as expected, as more of the pairwise comparisons now lie into the rope . However, the patterns observed did not have noticeable changes.

This analysis is given considering all the datasets with $\Omega < 1.000$, that is to say, the ones at the top and in the middle part of Table 3. It is now interesting to focus on the 13 very-HDLSS datasets, i.e., where $\Omega < 0.015$. Figure 5 gives the Critical Difference diagram for these datasets only. What is interesting to note here is that LMNNSVM is now a lot more competitive and that all three similarity-based methods are clearly in the lead. The difference in average rank between these three methods and the other methods is even greater than it was in the previous results. On the other hand, these differences are considered less statistically significant by the Friedman/Nemenyi test. Nevertheless, when looking at the bayesian analysis for these 13 very-HDLSS datasets in Figure 6, the probability of any similarity-based methods to be outperformed by XGBoost, RF, SVM or DWD is very low (see the upper right part of the color maps). This shows that in the most extreme cases of HDLSS classification, SVM with well chosen similarity measure as kernel are particularly relevant.

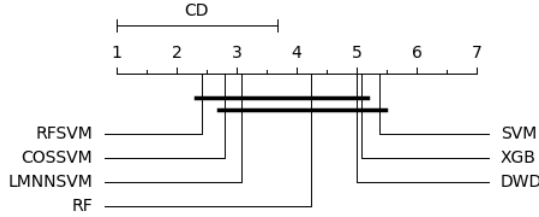


Fig. 5 Critical Difference diagram from the Friedman/Nemenyi test results on the very-HDLSS datasets ($\Omega < 0.015$).

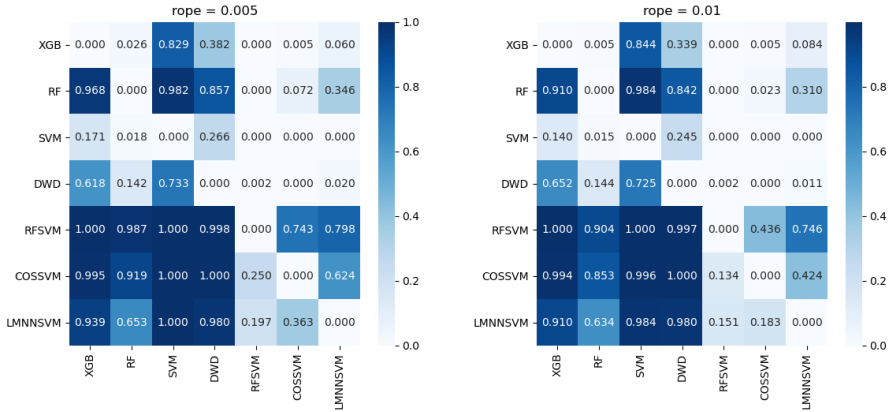


Fig. 6 Pairwise Bayesian analysis for very-HDLSS datasets. The value in each cell is $p(a > b)$, i.e. the probability that the row classifier a outperforms the column classifier b .

A note on class imbalance

Given the results in Table 1, one can see that some of the datasets have quite high imbalanced ratios (high values in the ‘IR’ column of Table 1). In particular, three datasets show very high values in the ‘IR’ column of Table 1: the *tr45.wc* dataset, the *bhattacharjee-2001* dataset and the *yeoh-2002-v2* dataset, for which the imbalance ratio is superior to 5. In class imbalance scenarios, it is well-known that accuracy is not a suitable performance evaluation measure. To determine whether the high IR values affect the results presented so far, we give additional results for these three specific datasets.

Table 4 gives the results obtained by the seven methods on these three datasets in terms of F1 (top) and accuracy scores (bottom). F1 represents the harmonic mean between precision and recall, and it is widely applied in the assessment of imbalanced classification tasks [44]. When working with non-binary problems, we used the micro-average for F1, which computes the metric globally by counting the total number of true positives, false negatives and false positives per class. When comparing values in both parts of the table, one can note that most of them are comparable and that the rank of all seven methods is globally similar with respect to both evaluation measures. It means

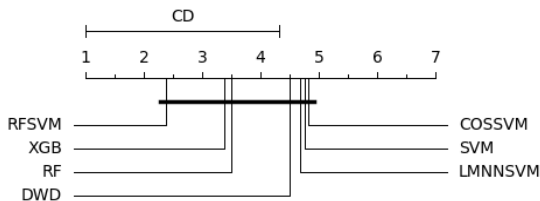
Table 4 F1 Score (top) and Accuracy (bottom) results obtained in imbalanced datasets.

dataset	XGB	RF	SVM	DWD	RFSVM	COSSVM	LMNNSVM
tr45.wc	0.972 ± 0.007	0.946 ± 0.001	0.745 ± 0.017	0.754 ± 0.017	0.954 ± 0.006	0.813 ± 0.019	0.784 ± 0.013
bhattacharjee-2001	0.949 ± 0.020	0.930 ± 0.016	0.937 ± 0.017	0.932 ± 0.025	0.959 ± 0.021	0.936 ± 0.018	0.934 ± 0.017
yeoh-2002-v2	0.848 ± 0.017	0.811 ± 0.017	0.787 ± 0.018	0.782 ± 0.029	0.849 ± 0.027	0.718 ± 0.023	0.817 ± 0.034
tr45.wc	0.970 ± 0.008	0.949 ± 0.012	0.815 ± 0.070	0.665 ± 0.107	0.954 ± 0.007	0.925 ± 0.008	0.900 ± 0.046
bhattacharjee-2001	0.946 ± 0.014	0.929 ± 0.016	0.937 ± 0.017	0.923 ± 0.022	0.957 ± 0.019	0.936 ± 0.018	0.934 ± 0.017
yeoh-2002-v2	0.842 ± 0.018	0.809 ± 0.015	0.787 ± 0.018	0.715 ± 0.022	0.848 ± 0.024	0.718 ± 0.023	0.815 ± 0.031

that the high IR values do not call into question the conclusions drawn earlier, including for the imbalanced HDLSS datasets.

5.3 Analysis of the results on non-HDLSS datasets

Finally, we present the results on non-HDLSS datasets in order to analyze whether the similarity-based approaches are still competitive for regular classification tasks, i.e., on datasets with $\Omega \geq 1$. For that purpose, Figure 7 gives the Critical Difference diagram on the non-HDLSS datasets, that is to say, the ones in the bottom part of Table 1.

**Fig. 7** Critical Difference diagram from the Friedman/Nemenyi test results on the non-HDLSS datasets ($\Omega \geq 1.0$).

From this diagram, one can see that the global ranking is quite different, with in particular COSSVM and LMNNSVM being a lot less competitive. In contrast, XGBoost is not surprisingly far more accurate on average on these datasets. Nevertheless, this time, none of the differences in rank is statistically significant according to the Friedman/Nemenyi post-hoc test.

However, we would like to emphasize that the RFSVM method is still ranked first on average and is particularly competitive with the state-of-the-art general-purpose classification methods, namely XGBoost and Random Forest. For a more detailed analysis, we also give the results of the Bayesian test in Figure 8. The competitiveness of RFSVM is confirmed by the fact that the probabilities of the RFSVM column in these color maps are still very low for these datasets.

6 Conclusion

HDLSS classification problems are unavoidable in many real-world pattern recognition problems, and having methods to provide a satisfactory solution

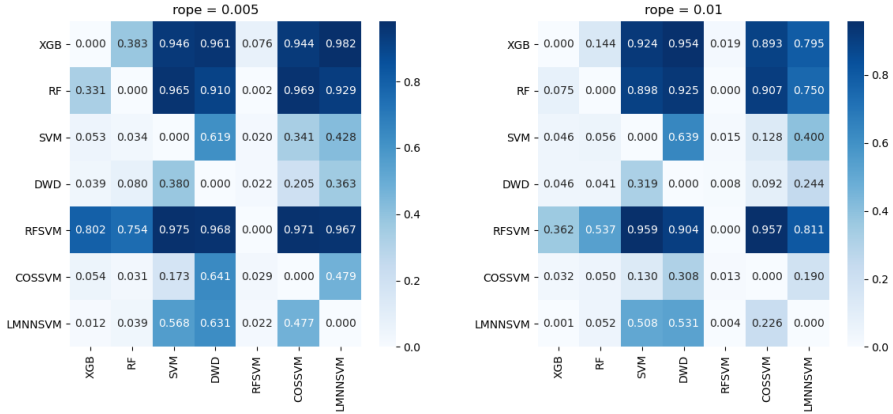


Fig. 8 Pairwise bayesian analysis for non-HDLSS datasets. The value in each cell is $p(a > b)$, i.e. the probability that the row classifier a outperforms the column classifier b .

to such problems is of crucial importance. Usually, it is faced with dimensionality reduction techniques and posterior induction of general-purpose machine learning models. However, in many situations, dimensionality reduction techniques give unsatisfactory results and genuine HDLSS learning methods are needed.

In this work, we show that one of these methods, RFSVM, is particularly efficient, regardless of the “degree of HDLSS” of the problem. This method, that has been designed in our previous works for multi-view learning, is based on the use of Random Forests to estimate the similarity between the training data and on the use of these similarities as a pre-computed kernel in an SVM classifier.

To show the suitability and efficiency of RFSVM for HDLSS data, we designed a rigorous experimental comparison with different state-of-the-art classification techniques, from general-purpose techniques to state-of-the-art HDLSS methods. This comparison has been conducted on 40 different datasets, with 32 HDLSS datasets and 8 regular classification datasets for control, all publicly available. The HDLSS datasets have been carefully selected to propose a wide variety of HDLSS levels, from “slightly HDLSS” to “very HDLSS”. Two statistical analyses of the results have been conducted to support the superiority of RFSVM over the others, a frequentist analysis with the Friedman statistical test along with the Nemenyi post-hoc test, and a bayesian analysis with a Bayesian sign test. Both statistical analyses confirm the superiority of RFSVM over its competitors, but with the interesting nuance that, more globally, similarity-based approaches are the most robust to extremely HDLSS learning conditions.

Based on the success of these similarity-based classification methods, we believe it is worthwhile to apply them to more specific tasks where HDLSS

datasets are known to be challenging, as anomaly/outlier detection, novelty detection, or data domain description. Detecting outliers, for example, is of particular importance in HDLSS problems since without much data for training, outliers may have an important influence on the result. Estimating distances between points, which is crucial for detecting outliers, is however difficult in the HDLSS context. We believe that the Random Forest Kernel approach could be an efficient alternative in this context.

Finally, many naturally HDLSS real-world applications have a characteristic that is usually a challenge in machine learning: data sparsity. While Random Forests are known to be robust to high dimensions, they are also known to suffer from data sparsity. However, these methods are frequently used on sparse HDLSS data as they embed efficient pre-processing and/or interpretability tools. For example, they may serve as feature selectors in a sparse data classification context, as in [45] where they are used for gene selection and classification of microarray data. For this reason, we believe it is relevant to further investigate the behavior of our method in the context of sparse data learning.

7 Acknowledgments

This work is part of the DAISI project, co-financed by the European Union with the European Regional Development Fund (ERDF) and by the Normandy Region.

References

- [1] Pappu, V., Pardalos, P.M.: High-dimensional data classification. In: Aleskerov, F., Goldengorin, B., Pardalos, P.M. (eds.) *Clusters, Orders, and Trees: Methods and Applications*, pp. 119–150. Springer, New York (2014). https://doi.org/10.1007/978-1-4939-0742-7_8
- [2] Yin, Q., Adeli, E., Shen, L., Shen, D.: Population-guided large margin classifier for high-dimension low-sample-size problems. *Pattern Recognition* **97**, 107030 (2020). <https://doi.org/10.1016/j.patcog.2019.107030>
- [3] Nakayama, Y., Yata, K., Aoshima, M.: Bias-corrected support vector machine with gaussian kernel in high-dimension, low-sample-size settings. *Annals of the Institute of Statistical Mathematics* **72**(5), 1257–1286 (2019). <https://doi.org/10.1007/s10463-019-00727-1>
- [4] Ghaddar, B., Naoum-Sawaya, J.: High dimensional data classification and feature selection using support vector machines. *European Journal of Operational Research* **265**(3), 993–1004 (2018). <https://doi.org/10.1016/j.ejor.2017.08.040>

- [5] Ma, J., Yuan, Y.: Dimension reduction of image deep feature using PCA. *Journal of Visual Communication and Image Representation* **63**, 102578 (2019). <https://doi.org/10.1016/j.jvcir.2019.102578>
- [6] Gunduz, N., Fokoue, E.: Robust Classification of High Dimension Low Sample Size Data. arXiv e-prints, 1501–00592 (2015) <https://arxiv.org/abs/1501.00592> [stat.AP]
- [7] Kouiroukidis, N., Evangelidis, G.: The effects of dimensionality curse in high dimensional knn search. In: 2011 15th Panhellenic Conference on Informatics, pp. 41–45 (2011). <https://doi.org/10.1109/PCI.2011.45>
- [8] Zhang, L., Lin, X.: Some considerations of classification for high dimension low-sample size data. *Statistical Methods in Medical Research* **22**(5), 537–550 (2011). <https://doi.org/10.1177/0962280211428387>
- [9] Marron, J.S., Todd, M.J., Ahn, J.: Distance-weighted discrimination. *Journal of the American Statistical Association* **102**(480), 1267–1271 (2007). <https://doi.org/10.1198/016214507000001120>
- [10] Cao, H., Bernard, S., Sabourin, R., Heutte, L.: Random forest dissimilarity based multi-view learning for radiomics application. *Pattern Recognition* **88**, 185–197 (2019). <https://doi.org/10.1016/j.patcog.2018.11.011>
- [11] Shen, L., Er, M.J., Yin, Q.: Classification for high-dimension low-sample size data. *Pattern Recognition* **130**, 108828 (2022). <https://doi.org/10.1016/j.patcog.2022.108828>
- [12] Kuncheva, L.I., Matthews, C.E., Arnaiz-González, A., Rodríguez, J.J.: Feature selection from high-dimensional data with very low sample size: A cautionary tale. arXiv e-prints, 2008–12025 (2020) <https://arxiv.org/abs/2008.12025> [cs.LG]
- [13] Shen, L., Yin, Q.: Data maximum dispersion classifier in projection space for high-dimension low-sample-size problems. *Knowledge-Based Systems* **193**, 105420 (2020). <https://doi.org/10.1016/j.knosys.2019.105420>
- [14] Friedman, J.H.: Regularized discriminant analysis. *Journal of the American Statistical Association* **84**(405), 165–175 (1989)
- [15] Dutta, S., Ghosh, A.K.: On some transformations of high dimension, low sample size data for nearest neighbor classification. *Machine Learning* **102**, 57–83 (2016). <https://doi.org/10.1007/s10994-015-5495-y>
- [16] Guo, Y., Hastie, T., Tibshirani, R.: Regularized linear discriminant analysis and its application in microarrays. *Biostatistics* **8**(1), 86–100 (2006).

<https://doi.org/10.1093/biostatistics/kxj035>

- [17] François, D., Wertz, V., Verleysen, M.: The concentration of fractional distances. *IEEE Transactions on Knowledge and Data Engineering* **19**, 873–886 (2007)
- [18] Deegalla, S., Bostrom, H.: Reducing high-dimensional data by principal component analysis vs. random projection for nearest neighbor classification. In: 2006 5th International Conference on Machine Learning and Applications (ICMLA'06), pp. 245–250 (2006). <https://doi.org/10.1109/ICMLA.2006.43>
- [19] Radovanović, M., Nanopoulos, A., Ivanović, M.: Nearest neighbors in high-dimensional data: The emergence and influence of hubs. In: Proceedings of the 26th Annual International Conference on Machine Learning. ICML '09, pp. 865–872. Association for Computing Machinery, New York, NY, USA (2009). <https://doi.org/10.1145/1553374.1553485>. <https://doi.org/10.1145/1553374.1553485>
- [20] Tomašev, N., Radovanović, M., Mladenić, D., Ivanović, M.: Hubness-based fuzzy measures for high-dimensional k-nearest neighbor classification. In: Proceedings of the 7th International Conference on Machine Learning and Data Mining in Pattern Recognition. MLDM'11, pp. 16–30. Springer, Berlin, Heidelberg (2011)
- [21] Pal, A.K., Mondal, P.K., Ghosh, A.K.: High dimensional nearest neighbor classification based on mean absolute differences of inter-point distances. *Pattern Recognition Letters* **74**(C), 1–8 (2016). <https://doi.org/10.1016/j.patrec.2016.01.018>
- [22] Cortes, C., Vapnik, V.: Support-vector networks. *Machine Learning* **20**(3), 273–297 (1995). <https://doi.org/10.1007/bf00994018>
- [23] Qiao, X., Zhang, H.H., Liu, Y., Todd, M.J., Marron, J.S.: Weighted distance weighted discrimination and its asymptotic properties. *Journal of the American Statistical Association* **105**(489), 401–414 (2010). <https://doi.org/10.1198/jasa.2010.tm08487>
- [24] Hofmann, T., Schölkopf, B., Smola, A.J.: Kernel methods in machine learning. *The Annals of Statistics* **36**(3), 1171–1220 (2008). <https://doi.org/10.1214/009053607000000677>
- [25] Vert, J.P., Tsuda, K., Schölkopf, B.: A primer on kernel methods. In: Kernel Methods in Computational Biology, pp. 35–70. MIT Press, Cambridge, MA, USA (2004). Max-Planck-Gesellschaft
- [26] Pekalska, E., Paclík, P., Duin, R.P.W.: A generalized kernel approach to

- dissimilarity-based classification. *Journal of Machine Learning Research* **2**, 175–211 (2002)
- [27] Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D.: Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research* **15**(90), 3133–3181 (2014)
- [28] Breiman, L.: Random forests. *Machine Learning* **45**(1), 5–32 (2001). <https://doi.org/10.1023/a:1010933404324>
- [29] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* **12**, 2825–2830 (2011)
- [30] Platt, J.: Sequential minimal optimization: A fast algorithm for training support vector machines (1998). <https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.43.4376>
- [31] Domeniconi, C., Peng, D.G.J.: Large margin nearest neighbor classifiers. *IEEE transactions on Neural Networks* **16**(4), 899–909 (2005)
- [32] Cao, H., Bernard, S., Heutte, L., Sabourin, R.: Dynamic voting in multi-view learning for radiomics applications. In: Bai, X., Hancock, E.R., Ho, T.K., Wilson, R.C., Biggio, B., Robles-Kelly, A. (eds.) *Structural, Syntactic, and Statistical Pattern Recognition*, pp. 32–41. Springer, Cham (2018)
- [33] Cao, H., Bernard, S., Sabourin, R., Heutte, L.: A novel random forest dissimilarity measure for multi-view learning. In: *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 1344–1351 (2021). <https://doi.org/10.1109/ICPR48806.2021.9412961>
- [34] Vanschoren, J., van Rijn, J.N., Bischl, B., Torgo, L.: Openml: Networked science in machine learning. *SIGKDD Explorations* **15**(2), 49–60 (2013). <https://doi.org/10.1145/2641190.2641198>
- [35] Dua, D., Graff, C.: *UCI Machine Learning Repository* (2017). <http://archive.ics.uci.edu/ml>
- [36] de Souto, M., Costa, I., Araujo, D., Ludermit, T., Schliep, A.: Clustering cancer gene expression data: A comparative study. *BMC Bioinformatics* **9** (2008). <https://doi.org/10.1186/1471-2105-9-497>
- [37] Chen, T., Guestrin, C.: XGBoost: A scalable tree boosting system. In: *Proceedings of the 22nd ACM SIGKDD International Conference on*

- Knowledge Discovery and Data Mining, pp. 785–794. ACM, New York, NY, USA (2016). <https://doi.org/10.1145/2939672.2939785>
- [38] Bergstra, J., Yamins, D., Cox, D.D.: Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In: Proceedings of the 30th International Conference on International Conference on Machine Learning. ICML'13, vol. 28, pp. 115–123 (2013)
- [39] Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyperparameter optimization. In: Proceedings of the 24th International Conference on Neural Information Processing Systems. NIPS'11, pp. 2546–2554. Curran Associates Inc., Red Hook, NY, USA (2011)
- [40] Putatunda, S., Rama, K.: A comparative analysis of hyperopt as against other approaches for hyper-parameter optimization of xgboost. In: Proceedings of the 2018 International Conference on Signal Processing and Machine Learning. SPML '18, pp. 6–10. Association for Computing Machinery, New York, NY, USA (2018). <https://doi.org/10.1145/3297067.3297080>. <https://doi.org/10.1145/3297067.3297080>
- [41] Demšar, J.: Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research* **7**(1), 1–30 (2006)
- [42] Benavoli, A., Corani, G., Demšar, J., Zaffalon, M.: Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. *Journal of Machine Learning Research* **18**(77), 1–36 (2017)
- [43] Weinberger, K.Q., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. *The Journal of Machine Learning Research* **10**, 207–244 (2009)
- [44] Forman, G., Scholz, M.: Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement. *SIGKDD Explor. Newsl.* **12**(1), 49–57 (2010). <https://doi.org/10.1145/1882471.1882479>
- [45] Diaz-Uriarte, R., de Andrés, S.A.: Gene selection and classification of microarray data using random forest. *BMC Bioinformatics* **7**(3) (2006)