Pattern Recognition 82 (2018) 130–146

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Data-independent Random Projections from the feature-space of the homogeneous polynomial kernel

Daniel López-Sánchez [a,*], Angélica González Arrieta [a], Juan M. Corchado [a,b]

[a] University of Salamanca, BISITE Research Group, Spain
[b] Osaka Institute of Technology, Japan

ARTICLE INFO

Article history:
Received 4 July 2017
Revised 9 March 2018
Accepted 3 May 2018
Available online 8 May 2018

Keywords:
Random Projection
Homogeneous polynomial kernel
Nonlinear dimensionality reduction

ABSTRACT

Performing a Random Projection from the feature space associated to a kernel function may be important for two main reasons. (1) As a consequence of the Johnson–Lindestrauss lemma, the resulting low-dimensional representation will preserve most of the structure of data in the kernel feature space and (2) an efficient linear classifier trained on transformed data might approximate the accuracy of its nonlinear counterparts. In this paper, we present a novel method to perform Random Projections from the feature space of homogeneous polynomial kernels. As opposed to other kernelized Random Projection proposals, our method focuses on a specific kernel family to preserve some of the beneficial properties of the original Random Projection algorithm (e.g. data independence and efficiency). Our extensive experimental results evidence that the proposed method efficiently approximates a Random Projection from the kernel feature space, preserving pairwise distances and enabling a boost on linear classification accuracies.

1. Introduction

The Random Projection method [1] is an extremely simple and widely used linear dimensionality reduction method [2–5]. As opposed to other approaches, Random Projection computes the projection matrix from a random distribution, thus being a data-independent method. In spite of its simplicity, Random Projection has strong theoretical foundations. The main theoretical basis that underpins Random Projection is the Johnson–Lindestrauss (JL) lemma, which states that a small set of points in a high-dimensional space can be embedded into a space of much lower dimension in such a way that distances between the points are nearly preserved. Formally, for any $0 < \epsilon < 1$ and $x_1, x_2, \ldots, x_n \in \mathbb{R}^d$ there is a map $f : \mathbb{R}^d \to \mathbb{R}^k$ for $k = \mathcal{O}(\epsilon^{-2}\log(n))$ such that:

$$(1-\epsilon)||x_i - x_j||^2 \leqslant ||f(x_i) - f(x_j)||^2 \leqslant (1+\epsilon)||x_i - x_j||^2 \quad \forall i, j \tag{1}$$

Furthermore, this map can be found in randomized polynomial time.[1] In the original version of the algorithm, the map f consisted of projecting the points from $\mathbb{R}^d$ to $\mathbb{R}^k$ by means of a $d \times k$ projection matrix, whose elements were drawn from a standard normal distribution. Once the $d \times k$ matrix R had been populated, an arbitrary set of n points represented as an $n \times d$ matrix X could be projected from $\mathbb{R}^d$ to $\mathbb{R}^k$ according to Eq. (2).

$$X'_{n\times k} = \frac{1}{\sqrt{k}} X_{n\times d} R_{d\times k} \tag{2}$$

However, Achlioptas showed that the projection matrix can instead be drawn from a much simpler distribution [7]. Specifically, he showed that if the projection matrix is drawn from the distribution defined by Eq. (3) (where he used $s = 1, 3$), then the JL-lemma will be satisfied [8].

$$r_{ij} = \sqrt{s}\begin{cases} 1 & \text{with prob. } 1/2s \\ 0 & \text{with prob. } 1 - 1/s \\ -1 & \text{with prob. } 1/2s \end{cases} \tag{3}$$

Moreover, Achlioptas proved that as long as the elements of the projection matrix are independent and identically distributed random variables with zero mean and unit variance, pairwise distances will be approximately preserved. Using the distribution proposed by Achlioptas reduces the computational cost of the projection. In fact, if the multiplication by $\sqrt{s}$ present in Eq. (3) is delayed, the computation of the projection itself reduces to aggregate evaluation (i.e. summation and subtraction but no multiplication), which can be efficiently performed in database environments using standard SQL primitives.

More recently, a non-linear variant of the Random Projection algorithm has been proposed in the literature [9,10]. In this con-

* Corresponding author.
E-mail address: lope@usal.es (D. López-Sánchez).

[1] A particularly simple proof of this lemma was introduced in [6].

https://doi.org/10.1016/j.patcog.2018.05.003
0031-3203/

text, the authors try to perform a Random Projection from an extended non-linear feature space. Specifically, their method is able to perform a Random Projection from the *feature space*[2] of an arbitrary kernel function. This can be of interest because (1) the low-dimensional projected points preserve most of the structure from the kernel feature-space and (2) after the random projection, a classification problem may become more linearly solvable, and the classification accuracy of scalable linear classifiers may increase. In spite of being compatible with any kernel function, this kernelized version of Random Projection sacrificed some of the advantages of the original Random Projection method, namely data-independence and computational efficiency. In addition, it is unclear whether this algorithm is compatible with the database-friendly distribution proposed by Achlioptas.

In this paper, we propose an novel method to efficiently perform Random Projections from the feature space of homogeneous polynomial kernels of arbitrary degree. By focusing specifically on the family of homogeneous polynomial kernels, our approach manages to preserve the data-independence and efficiency of the original Random Projection method, as compared to previous Kernelization attempts which work with arbitrary kernels but sacrifice these beneficial properties [9,10]. Although less popular than the Radial Basis Function kernel (RBF), polynomial kernels have been found to be very effective in some cases [11,12]. In essence, the method proposed in this paper can be used to efficiently capture the structure of data in the feature-space of homogeneous polynomial kernels, condensing this information in a low-dimensional representation of data. In addition, our method is compatible with the database-friendly distribution proposed by Achlioptas. Our experimental results evidence that the proposed method outperforms alternative approaches in terms of pairwise-distance preservation, while requiring significantly less computational resources. We also present results evidencing that the generated feature representations can be used for a higher linear classification accuracy, approximating the effectiveness of nonlinear classifiers in some datasets.

The rest of this manuscript is structured as follows. Section 2 reviews some of the most prominent works that study the possible kernelization of the Random Projection technique. Section 3 introduces our proposed algorithm and analyzes its compatibility with the database-friendly distribution proposed by Achlioptas. It also contains a detailed analysis of the computational complexity of our algorithm and other alternative approaches. Section 4 compiles the results of an extensive empirical, which evidences the properties of our kernelized variant of Random Projection. Finally, in Section 5 we present the conclusions of this work and propose some promising future lines of research.

2. Related work

As previously said, the problem of developing a kernelized variant of the original Random Projection algorithm has already been addressed in the literature. The interest in this attempt is motivated by two main reasons:

1. A kernelized variant of the Random Projection algorithm would provide a means to generate low dimensional representations where relative distances between data points would be approximately equal to those in the kernel feature space. This could have applications in tasks such as clustering and information retrieval.
2. The question of whether Random Projections preserve inner products has recently been an object of controversy [13]. However, it has been shown that angles between data samples and separability margins are approximately preserved after a Random Projection [14]. As a consequence, an efficient technique which performs a Random Projection from kernel feature spaces could be used as a representation generator to learn a linear classifier. Such linear classifier would benefit from the non-linearity of the feature space and approximate the accuracy of non-linear classifiers, while being significantly more efficient in both training and test stages [15].

Motivated by these possibilities, the authors of [14,16] analyzed whether it would be possible to formulate an algorithm capable of performing a Random Projection from the feature space of an arbitrary kernel function, by just having black-box access to the kernel function but no unlabeled training samples (i.e. without access to the distribution of input data). Unfortunately, their results were negative, and the authors proved that this is not possible for an arbitrary black-box kernel. However, they left the question open of whether such methods could be developed for specific kernel functions such as the polynomial kernel.

Years later, Alavi et al. [9] and Zhao et al. [10] proposed a general method to perform Random Projections from arbitrary kernel feature spaces. Their findings did not contradict the result described in the previous paragraph since the method they proposed required access to a number of unlabeled training samples to work. Interestingly, their algorithm was based on an approach developed to solve a different problem, namely the Kulis–Grauman approach [17]. This technique, originally developed to perform a kernelized variant of Locally Sensitive Hashing, can be used to generate a set of nearly Gaussian hyperplanes in an arbitrary kernel implicit feature space, without any computation of the explicit embedding $\phi : \mathbb{R}^d \to \mathcal{H}$. Despite its success, this approach has a major drawback inherent to its core idea: the approximately Gaussian hyperplanes in the implicit kernel space are built as a weighted sum of a subset of the database items, thus making the method data-dependent. On their side, the authors of [9,10] showed how, with minimal modifications, the Kulis–Grauman approach can be used to perform a Random Projection from an arbitrary kernel's feature space. From now on we will refer to this kernelized variant as the Kulis–Grauman Random Projection (KG-RP). As previously explained, KG-RP is a data-dependent method. As a consequence, the quality of the embeddings it produces depends on the amount of data available and its variability. In addition, most of the computational efficiency of the original Random Projection method is lost in this version. For example, the training phase in the original method only involves the population of a projection matrix from a random distribution. Unfortunately, the training phase in KG-RP entails expensive computations over training samples[3]

Following a diametrically opposite approach, Chang et al. proposed explicitly computing the feature map of low-rank polynomial kernels and using these to train efficient linear classifiers [11]. They exploited the fact that, as opposed to other popular kernel functions, the feature space associated to polynomial kernels is known and of finite dimension. They also took advantage of the sparse nature of some datasets to reduce the time and storage requirements of explicitly computing the kernel feature-map. Although their results evidenced the potential of polynomial kernels, this approach is prohibitively demanding in terms of storage and computation. This is especially true when working with polynomial degrees greater than two, as in the case of polynomial kernels the dimension of the feature space grows exponentially

[2] In the context of kernel functions, the term *feature space* refers to the Hilbert space $\mathcal{H}$ associated to a given positive definite kernel function such that $K(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$.

[3] The computational complexities of alternative methods are analyzed in more detail in Section 3.4.

with the degree. More recent work on low-degree polynomial kernel approximation has shown that rather than explicitly computing the explicit feature space, it is possible to implicitly perform a Random Projection from it. Particularly, a data-independent algorithm named P-RP has recently been proposed to perform Random Projections from the feature space of degree-two homogeneous polynomial kernels [18]. Unfortunately, the applicability of this method is severely limited by the exclusive compatibility with the second degree homogeneous polynomial kernel. In addition, it requires populating a number of complete Random Projection matrices, thus incurring in significant computational overheads. Furthermore, this method is outperformed in most of the experiments presented in Section 4, while being almost one order of magnitude slower than the method presented in this paper.

Finally, it is worth noticing that during the past decade a lot of effort has been put into designing methods to efficiently approximate different kernels' feature spaces. Formally, given a kernel function $K(\cdot, \cdot)$, the goal of such methods is to find an approximated feature map $h(\cdot)$ such that:

$$K(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}} \approx \langle h(x), h(y) \rangle_{\mathbb{R}^k} \tag{4}$$

where $h(\cdot)$ can be computed efficiently and the feature space it generates is sufficiently low-dimensional or sparse [19]. The main motivation behind this effort is that, while Support Vector Machines (SVM) using the kernel trick scale poorly [20], linear SVMs can be learned in linear time with respect to the number of samples available at training time [21]. As a consequence, approximate kernel feature spaces can serve as a basis for training efficient linear SVMs which achieve accuracy rates similar to those of their kernelized counterparts. Note that these methods are designed to approximate dot-products between samples rather than Euclidean distances (i.e. they are not directly related to Random Projection or the JL-lemma). Nevertheless, we chose one of the most popular and generally applicable methods of this class, namely the Nyström method [22] and included it in our experimental comparisons.

3. Kernelized Random Projection with homogeneous polynomial kernels

In this section, we introduce the proposed method and provide a simple pseudo-code description to ease its implementation and increase the reproducibility of our results. Afterwards, the compatibility of our technique with the database-friendly distribution proposed by Achlioptas [7] is explored. Finally, the computational complexity of our algorithm is analyzed in both train and test phases, and compared to alternative approaches.

3.1. Homogeneous polynomial kernels

As outlined before, our method is specifically designed to efficiently perform Random Projections from the feature space of homogeneous polynomial kernels. We focused on this family of kernel functions due to their simplicity, proven power [11] and special properties, which will allow us to perform the Random Projection efficiently and without any knowledge of the distribution of data to be projected. Formally, polynomial kernels are computed as follows:

$$K(x, y) = (\langle x, y \rangle + c)^g \tag{5}$$

Homogeneous polynomial kernels are uniquely those polynomial kernels with $c = 0$. Given that homogeneous polynomial kernels are positive-definite, there is a feature map $\phi : \mathbb{R}^d \to \mathcal{H}$ such that:

$$K(x, y) = \langle x, y \rangle^g = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}} \tag{6}$$

In fact, in the case of homogeneous polynomial kernels the mapping function $\phi(\cdot)$ is known and produces a finite-dimensional

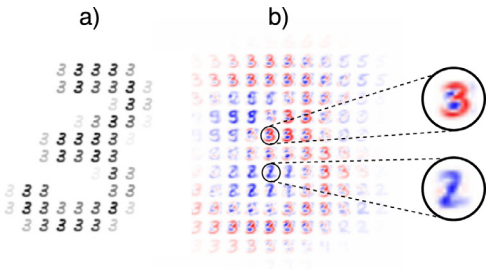

Fig. 1. Visualization of (a) A digit from MNIST in the feature space of the homogeneous polynomial kernel of degree two, and (b) The weights learned by a simple gradient descent linear classifier on that feature space. Positive weights are depicted in red and negative weights in blue. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

representation [23]. Formally, the feature space associated to the polynomial kernel of dimension g is computed as follows:

$$\phi(x) = \bigotimes_{i=1}^{g} x \tag{7}$$

where $\otimes$ denotes the Kronecker product. For example, if $g = 2$ and $x \in \mathbb{R}^d$, then:

$$\begin{aligned} \phi_{g=2}(x) &= x \otimes x \\ &= [x_1 x_1, \cdots, x_1 x_d, x_2 x_1, \cdots, x_d x_d] \in \mathcal{H} \end{aligned}$$

The intuition behind this kernel family is that it is often useful to construct new features as the product of original features. Note that the polynomial degree g in Eq. (7) determines the order of monomials composing the feature space. To provide some intuition on the nature of homogeneous polynomial kernels, we generated the explicit feature-space representation for a number of 14×14 resized digit images from the MNIST dataset.[4] Then, a simple gradient-descent linear classifier was trained on them to distinguish the digit "3" from all the others:

$$y = tanh(\langle \phi(x), w \rangle + b) \tag{8}$$

Fig. 1 shows one of those samples in the kernel feature space and the weight vector learned by the linear classifier. As we can see, it uses different "templates" to emit a prediction, depending on the presence/absence of intensity in the different regions of the original digit image. This exemplifies how linear classification can benefit form polynomial features as they allow them to account for interactions between features.

Unfortunately, the dimension of $\phi(x) \in \mathcal{H}$ grows exponentially with the polynomial degree. In particular, the dimension of the feature space $\mathcal{H}$ for the g-dimensional homogeneous polynomial kernel is d^g. As a consequence, any algorithm using the explicit feature space of homogeneous polynomial kernels will rapidly become intractable as the original dimension of samples d or the polynomial degree g grow.

3.2. Kernelized Random Projection algorithm

Our goal is to perform a Random Projection form the kernel feature space $\mathcal{H}$ onto a lower-dimensional Euclidean space $\mathbb{R}^k$, while avoiding any explicit computation of the feature-mapping $\phi(\cdot)$. In this regard, each output component must be generated as the inner product between the mapped data point and a random hyperplane whose elements are drawn from a valid JL-distribution (e.g., a standard normal distribution):

$$\langle \phi(x), r \rangle_{\mathcal{H}} \quad \text{where } r \sim \mathcal{N}(0, I) \tag{9}$$

[4] http://yann.lecun.com/exdb/mnist/.

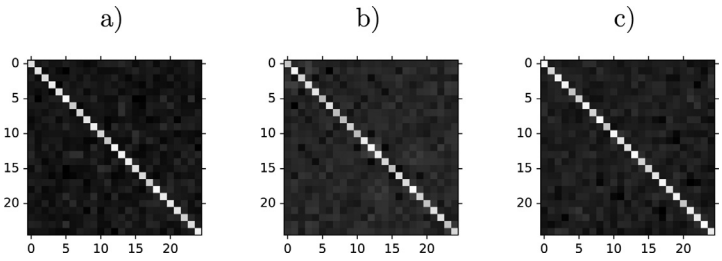

Fig. 2. Covariance matrices computed over 500 d^2-dimensional samples generated by (a) directly sampling $\mathcal{N}_{d^2}(0, I)$, (b) computing $\bigotimes_{i=1}^{2} r_i$ where $r_i \sim \mathcal{N}_d(0, I)$ to generate each sample and (c) computing $\sum_{j=1}^{t} \bigotimes_{i=1}^{2} r_{ij}$ where $r_{ij} \sim \mathcal{N}_d(0, I)$ to generate each sample. In this case d was set to 5.

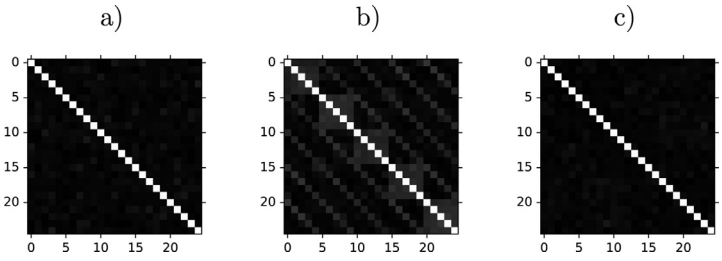

Fig. 3. Distance-correlation matrices [26] computed over 500 d^2-dimensional samples generated by (a) directly sampling $\mathcal{N}_{d^2}(0, I)$, (b) computing $\bigotimes_{i=1}^{2} r_i$ where $r_i \sim \mathcal{N}_d(0, I)$ to generate each sample and (c) computing $\sum_{j=1}^{t} \bigotimes_{i=1}^{2} r_{ij}$ where $r_{ij} \sim \mathcal{N}_d(0, I)$ to generate each sample. In this case d was set to 5.

To achieve this, our method uses the following property of homogeneous polynomial kernels[5]:

$$\prod_{i=1}^{g} \langle x, r_i \rangle_{\mathbb{R}^d} = \langle \phi(x), r\prime \rangle_{\mathcal{H}} \quad \text{where } r\prime = \bigotimes_{i=1}^{g} r_i \tag{10}$$

If $r_1, \cdots, r_g$ are drawn from $\mathcal{N}(0, I)$, the elements of r' will follow a symmetric distribution with zero mean and unit variance (the normal product distribution [24,25] in particular). As mentioned before, a necessary condition for a given projection hyperplane to satisfy the JL-lemma is that its elements must be independently drawn from a distribution with zero mean and unit variance. When analyzed individually, the elements of r' exhibit the desired zero mean and unit variance. However, due to the manner in which they are computed, they are not strictly independent. As a consequence, we cannot ensure that Eq. (10) performs a valid Random Projection.

To provide insight into the dependence of features in r', we generated a number of such random hyperplanes by explicitly computing $r\prime = \bigotimes_{i=1}^{g} r_i$. To keep the computations tractable, we considered a polynomial degree of two ($g = 2$) and an input feature space of dimension 5 ($r_i \sim \mathcal{N}_5(0, I)$). Hence, the dimension of the hyperplanes generated in this manner was 25. For comparison, we also generated a set of random hyperplanes by directly sampling $\mathcal{N}_{25}(0, I)$. Fig. 2(a) and (b) show the covariance matrices for the hyperplanes generated by either method. Interestingly, covariance matrices of both sets of hyperplanes look quite similar. This suggest that the dependence between the features of hyperplanes generated by means of Eq. (10) is not a mere linear correlation. To actually visualize this phenomenon, we need to use a more sophisticated measure of statistical dependence, namely the correlation-distance[6] [26]. Fig. 3(a) and (b) show the distance-correlation matrices for the random hyperplanes generated by directly sampling $\mathcal{N}_{25}(0, I)$ and using $r\prime = \bigotimes_{i=1}^{g} r_i$ respectively. In this case, the matrix corresponding to the hyperplanes generated by using $r\prime = \bigotimes_{i=1}^{g} r_i$ shows a clear non-random deviation with respect to the identity matrix. This indicates that, as expected, a certain degree of dependence exists among features.

Our proposed method applies the Central Limit Theorem (CLT) [27] to overcome the problem of feature dependence in projection hyperplanes. This classical result states that the sum of independent random variables with finite, non-null variance is approximately distributed according to a normal distribution. In this regard, our method will compute the projection hyperplanes as the sum of a set of t random hyperplanes generated according to Eq. (10). Formally, the final projection hyperplanes will take the form of:

$$\sum_{i=1}^{t} \left(\frac{1}{\sqrt{t}} \bigotimes_{j=1}^{g} r_{i,j} \right) \quad \text{where } r_{i,j} \sim \mathcal{N}_d(0, I) \tag{11}$$

Applying the CLT we can ensure that, for a sufficiently large t value, these hyperplanes will be approximately normal with zero mean and unit variance. Note that a correction factor of $1/\sqrt{t}$ was introduced to obtain the desired unit variance. Again, we explicitly generated a number of such projection hyperplanes to empirically asses the independence of their features (note that the final version of the algorithm will never compute these hyperplanes explicitly). Figs. 2(c) and 3(c) show the covariance and distance-correlation matrices of the generated hyperplanes. As desired, the distance-correlation matrix approximates the identity except for some random noise. This indicates that the use of the CLT efficiently mitigated the dependence among the elements of the projection hyperplanes. As a consequence, these random hyperplanes are valid for performing a Random Projection, ensuring that the JL-lemma will be satisfied.

At this point, we can present how our proposed method computes each component of the k-dimensional output representation for a given data point $x \in \mathbb{R}^d$. For computational reasons, instead of creating gtk unique random hyperplanes, our method generates a set of p hyperplanes and uses random subsets sampled from it for each output component. Formally, let S be a set of p random vectors drawn from $\mathcal{N}_d(0, I)$. Then, for each output component we

[5] Note that, although $r\prime \in \mathcal{H}$, an r such that $\phi(r) = r\prime$ does not exist in most cases.

[6] Distance-correlation is a statistical measure of dependence between random variables. As opposed to Pearson's correlation coefficient, distance-correlation takes a value of zero if and only if the variables are statistically independent.

form $S_l = \{r_1, \cdots, r_{tg}\}$, a set of $t \cdot g$ vectors chosen at random[7] from S (i.e. $S_l \subset S$). Afterwards, the l-th component in the output feature space is computed as follows:

$$f_l(x) = \left\langle \phi(x),\ \sum_{i=0}^{t-1} \left(\frac{1}{\sqrt{t}} \bigotimes_{j=1}^{g} r_{gi+j} \right) \right\rangle_{\mathcal{H}} \tag{12}$$

which corresponds to the projection of the mapped data point $\phi(x)$ onto a random hyperplane drawn from a standard normal distribution. Conveniently, Eq. (12) can be rewritten to avoid any explicit computation of the feature space or the Kronecker product:

$$\begin{aligned} f_l(x) &= \left\langle \phi(x), \sum_{i=0}^{t-1} \left(\frac{1}{\sqrt{t}} \bigotimes_{j=1}^{g} r_{gi+t} \right) \right\rangle_{\mathcal{H}} \\ &= \sum_{i=0}^{t-1} \left(\frac{1}{\sqrt{t}} \left\langle \phi(x),\ \bigotimes_{j=1}^{g} r_{gi+j} \right\rangle_{\mathcal{H}} \right) \\ &= \sum_{i=0}^{t-1} \left(\frac{1}{\sqrt{t}} \prod_{j=1}^{g} \langle x, r_{gi+j} \rangle \right) \end{aligned} \tag{13}$$

Then, the output representation for a given sample is formed by concatenating the k components and multiplying them by a scaling factor (as in [7]):

$$f(x) = \frac{1}{\sqrt{k}} \left[f_1(x), \cdots, f_k(x) \right] \tag{14}$$

In practice, the most effective strategy for transforming a sample involves pre-computing the projection of that sample over the p random hyperplanes in S. By so doing, Eq. (13) can be evaluated without any further projection operation (i.e. with a computational complexity independent of d). The computations involved in initializing and transforming a number of samples with the proposed method are summarized in Algorithm 1.

3.3. Sparse kernelized Random Projections

Thus far, we have assumed that the projection hyperplanes used in our method had to be drawn from a standard Gaussian distribution. However, as mentioned before, the hyperplanes used in the classical Random Projection algorithm can be drawn from the much simpler distribution proposed by Achlioptas [7]. When using this discrete distribution, the projection of data samples over the projection hyperplanes can be done in terms of aggregate evaluation. Also, if the hyperparameter s from Eq. (3) is set to values greater than one, only a fraction of each sample's components need to be evaluated when performing the projection, and fast sparse-matrix multiplication routines can be applied.

In this section we show that, surprisingly, the method presented in the previous section is directly compatible with the sparse distribution proposed by Achlioptas [7]. Let us consider Eq. (10) again: we want to analyze the distribution of r' when the random hyperplanes $r_1, \cdots, r_g \in \mathbb{R}^d$ used in this equation are drawn from the database-friendly distribution of Eq. (3). As $r\prime = \bigotimes_{i=1}^{g} r_i$, its elements are indeed the product of g discrete random variables drawn from Eq. (3). As a consequence, the distribution of the elements in r' can be determined using the following property of discrete random variables. Formally, given g independent and identically distributed random variables $V_1, \cdots, V_g$ with support $\mathcal{V}$ (i.e. the set of realizations that have a strictly positive probability of being observed), the distribution of $P(V_1 \cdots V_2 = c)$ can be computed as follows:

$$P(V_1 \cdots V_2 = c) = \sum_{\substack{v_1, \cdots, v_g \in \mathcal{V} \\ v_1 \cdots v_g = c}} P(V_1 = v_1) \cdots P(V_g = v_g) \tag{15}$$

Looking at Eq. (3) we can see that, in our case, the support is $\mathcal{V} = \{-1, 0, 1\}$, with associated probabilities $\frac{1}{2s}$, $1 - \frac{1}{s}$ and $\frac{1}{2s}$. Applying Eq. (15) we get that the elements of r' are distributed according to:

$$r\prime_i = \sqrt{s^g} \begin{cases} 1 & \text{with prob.} \quad 1/2s^g \\ 0 & \text{with prob.} \quad 1 - 1/s^g \\ -1 & \text{with prob.} \quad 1/2s^g \end{cases} \tag{16}$$

which is a valid JL-distribution according to Achlioptas' work (simply substitute s by s^g in Eq. (3)). However, as in the case of using the normal distribution, the elements in r' are not completely independent of each other. Fortunately, as the above distribution has a zero mean and unit variance, the CLT can be applied just like in the Gaussian case. As a consequence, the method proposed in the previous section is directly compatible with the discrete distribution proposed by Achlioptas. In fact, one might draw the projection hyperplanes of Eq. (13) from Achliptas' distribution and the result would still approximate a JL-projection from the kernel feature space. This claim is also supported by the experimental results presented in Section 4.

To use this sparse variant of the projection hyperplanes with our method, it suffices to modify step 1 of Algorithm 1. Instead of generating the projection hyperplanes in S by sampling $\mathcal{N}_d(0, I)$, they can be populated following the sparse distribution described in Eq. (3). Formally, step 1 of Algorithm 1 becomes:

$$S \leftarrow \{r_1, \cdots, r_p\}, \quad \text{where the entries of } r_i \in \mathbb{R}^d \text{ are drawn from } P(R)$$

$$P(R = x) = \begin{cases} 1/2s, & x = \sqrt{s} \cdot 1 \\ 1 - 1/s, & x = 0 \\ 1/2s, & x = \sqrt{s} \cdot -1 \end{cases} \tag{17}$$

Where the hyperparameter s controls the sparsity level of the hyperplanes. Conveniently, The following steps of the algorithm remain exactly the same. Also note that, apart from the sparseness of the hyperplanes, a major advantage of Achlioptas' distribution

Algorithm 1 Kernelized Random Projection for homogeneous polynomial kernels.

Require: A set of points $\{x_1, \cdots, x_N\}$ from $\mathbb{R}^d$, the desired degree g for the polynomial kernel, the total number of hyperplanes p, the number of hyperplanes t used for each output component and the desired output dimension k.

Ensure: Returns a set of output samples $\{x'_1, \cdots, x'_N\}$ in $\mathbb{R}^k$ such that pairwise distances between these samples are approximately equal to those of input data samples mapped on the feature space of the homogeneous polynomial kernel of degree g.

1: $S \leftarrow \{r_1, \cdots, r_p\}$ where $r_i \sim \mathcal{N}_d(0, I)$
2: Sample S to form $S_1, \cdots, S_k \subset S$, where $|S_i| = gt$
3: **for** n = 1; n ≤ N; n++ **do**
4: **for** l = 1; l ≤ k; l++ **do**
5: $x'_n[l] \leftarrow 0$
6: **for** i = 0; i < t; i++ **do**
7: $temp \leftarrow \frac{1}{\sqrt{t}}$
8: **for** j = 1; j ≤ g; j++ **do**
9: $temp \leftarrow temp \cdot \langle x_n, S_l[gi + j] \rangle$
10: $x'_n[l] \leftarrow x'_n[l] + temp$
11: $x'_n[l] \leftarrow \frac{1}{\sqrt{k}} \cdot x'_n[l]$
12: **return** $\{x'_1, \cdots, x'_N\}$

[7] In practice, in order to save storage resources, the subsets $S_1, \cdots, S_k$ store the indexes to the selected hyperplanes of S, rather than duplicated copies of them.

is the fact that the projection of samples onto the random hyperplanes (step 9 of Algorithm 1) can be implemented solely in terms of aggregate evaluation (i.e., summations and subtractions) by delaying the multiplication by $\sqrt{s}$ present in Achlioptas' distribution. This implementation trick can be an advantage in structured database environments, as the projections can be implemented with standard SQL primitives.

3.4. Computational complexity analysis

This section compares the computational complexities of the different methods both in training and test phases. First, we analyze the computational complexity of the KG-RP method. Most of the computations involved in this method correspond to the calculations performed in the Kulis–Grauman approach [17]. However, the computational complexities reported vary slightly due to different optimizations applied. Let us analyze step by step the computations performed by taking the pseudo-code implementation presented in [10] as reference. In the training stage we have to:

1. Compute the $p \times p$ kernel Gram matrix K_S among the p selected training points. Assuming the kernel computation takes $\mathcal{O}(d)$ for samples in $\mathbb{R}^d$, this step requires $\mathcal{O}(dp^2)$ time.
2. Compute $K_S^{-1/2}$ by means of eigendecomposition, which requires $\mathcal{O}(p^3)$ time.
3. Form the weight vector for each output component $w_1, \cdots, w_k$. Since each vector is computed as $w_i = \sqrt{\frac{p-1}{t}} K_S^{-1/2} e_S$, and e_S is a p-dimensional vector, this step has a complexity of $\mathcal{O}(kp^2)$.

By combining the different steps, the obtained complexity of training KG-RP is $\mathcal{O}(dp^2 + p^3 + kp^2)$. Note that, in the case of KG-RP, p is a hyper-parameter which controls the number of training samples used to estimate the mean and covariance matrix of the population in the implicit kernel space. The authors suggest a heuristic rule to select this value, namely using $p = \mathcal{O}(\sqrt{n})$, where n is the number of available training samples. Regarding the test phase, the following computations must be performed to transform a single sample:

1. Compute the kernel Gram matrix K between the test sample and the p points in $\mathcal{S}$. Assuming the kernel computation takes $\mathcal{O}(d)$ for samples in $\mathbb{R}^d$, this step requires $\mathcal{O}(pd)$ operations.
2. Generate the final representation of the test sample as KW, where $W = [w_1, \cdots, w_k]$. This can be done at the cost of $\mathcal{O}(pk)$ time.

Therefore, transforming a single test sample with KG-RP has a complexity of $\mathcal{O}(pd + pk)$.

Now we analyze the proposed kernelized Random Projection version. The computations needed to initialize/train the algorithm (steps 1–2 from Algorithm 1) are the following:

1. The set S is populated with p random hyperplanes drawn from $\mathcal{N}_d(0, I)$ (or alternatively using Achlioptas' distribution as described in Section 3.3), where d is the dimension of data samples. This can be done in $\mathcal{O}(pd)$ time.
2. The set S is sampled at random to form $S_1, \cdots, S_k \subset S$, each with cardinality gt. This takes $\mathcal{O}(gtk)$ time, where $gt < p$.

This shows that the training stage of the proposed method has a computational complexity of $\mathcal{O}(pd + gtk)$. To project a test sample, each output component is computed by using Eq. (12) or equivalently, executing the steps 3–12 of Algorithm 1. In any case, this computation requires a time of $\mathcal{O}(gtkd)$. As mentioned before, this complexity can be reduced by pre-computing the inner products between the test sample and the p hyperplanes in S. By so doing, the computational complexity of transforming a sample by means of the proposed method ends up being $\mathcal{O}(pd + gtk)$. It is also worth comparing the complexity of our method with that of P-RP, presented in [18]. As explained in the original paper, P-RP uses a number m of $d \times k$ projection matrices, and a good approximation of the degree-two homogeneous polynomial kernel can be achieved by using $m = 30$. From the analysis presented in the original paper, populating the projection matrices for P-RP takes $\mathcal{O}(dmk)$ time, and transforming one sample requires $\mathcal{O}(dmk)$ operations. As evidenced by our experimental results, this multiple projection matrix approach is highly inefficient, often leading to computing times one order of magnitude higher than our approach, while exhibiting an equal or worse performance. Also note that the complexity of P-RP is independent of the polynomial degree g, because this method is only compatible with $g = 2$.

Finally, the Nyström method works by generating a low-rank approximation of the kernel matrix by sampling a number of columns [22]. Although some alternative sampling methods have been studied, the original method, where a fixed random distribution is used to select the columns from the kernel matrix, continues being the fastest and one of the most widely used approaches [28]. For our analysis and experiments, we focus on the standard Nyström algorithm as implemented in [29]. The computations involved in training this algorithm are the following:

1. Compute the $k \times k$ reduced kernel Gram matrix W among the samples corresponding to the k selected columns from the full kernel matrix. Assuming the kernel computation takes $\mathcal{O}(d)$ for samples in $\mathbb{R}^d$, this step takes $\mathcal{O}(k^2 d)$.
2. Compute $W^{-1/2}$ by means of Singular Value Decomposition,[8] which requires $\mathcal{O}(k^3)$ time.

In summary, the training stage of Nyström requieres a time of $\mathcal{O}(dk^2 + k^3)$. To transform each sample, the following operations are performed in the test phase:

1. Compute the kernel Gram matrix K between the test sample and the k samples selected during training. Assuming the kernel computation takes $\mathcal{O}(d)$ for samples in $\mathbb{R}^d$, this step requires $\mathcal{O}(dk)$ operations.
2. Generate the output representation for the test sample as $KW^{-1/2}$. Since $W^{-1/2}$ is of size $k \times k$, this can be done in $\mathcal{O}(k^2)$.

From the combination of these complexities we obtain that transforming a test sample by means of Nyström has as a time complexity of $\mathcal{O}(dk + k^2)$.

Our analysis shows that the proposed algorithm exhibits a better computational complexity than the alternative methods. Concerning the training phase, the time required by KG-RP increases as the cube of p, and also requires p^2 evaluations of the kernel function. Similarly, Nyström's training time grows as the cube of k, and involves k^2 kernel evaluations. For its part, our proposed method has a training time which grows linearly with respect to p and k. In addition, thanks to its data-independent nature, it requires no evaluation of the kernel function in training time. Our method is also very competitive in terms of testing-time complexity. Provided that $tg < p$, the complexity of our method is lower than that of KG-RP.

The train and test computational complexities of the different methods analyzed in this section are summarized in Table 1.

4. Experimental results

This section presents extensive experimental results validating the ability of the proposed method to (1) generate a low-dimensional representation where the distances between points

[8] In particular, this is performed by using LAPACK's implementation of SVD (see http://www.netlib.org/lapack/).

Table 1
Computational complexities of different methods. (*) Complexity obtained by pre-computing the inner product between test samples and the p random hyperplanes in S.

Method	Train	Transform
Proposed KRP	$\mathcal{O}(pd + gtk)$	$\mathcal{O}(dtgk)$
Proposed KRP*	$\mathcal{O}(pd + gtk)$	$\mathcal{O}(pd + gtk)$
KG-RP [9]	$\mathcal{O}(p^2d + p^3 + kp^2)$	$\mathcal{O}(pd + pk)$
Nyström [22]	$\mathcal{O}(dk^2 + k^3)$	$\mathcal{O}(dk + k^2)$
P-RP [18]	$\mathcal{O}(dtk)$	$\mathcal{O}(dtk)$
$\phi(\cdot)$+RP	$\mathcal{O}(d^g k)$	$\mathcal{O}(d^g k)$

are approximately equal to the pairwise distances in the homogeneous polynomial kernel's feature space; and (2) boost the accuracy of linear classifiers by generating a data representation where linear classifiers can approximate the accuracy of their non-linear counterparts.

The methods evaluated are our proposed KRP version, KG-RP [9] and Nyström [22] (approximating the homogeneous polynomial kernel). We also compare these methods with the explicit approach $\phi(\cdot) + RP$, which involves explicitly transforming data with the feature mapping $\phi(\cdot)$ followed by linear RP. Of course, this approach is highly inefficient, but we use it to measure how well KG-RP and the proposed KRP approximate a Random Projection from the kernel feature space.

To evaluate the first property (i.e. pairwise distance preservation), we compare the squared Euclidean distance between two transformed data samples to their squared Euclidean distance in the kernel feature space. Formally, let x, y be a couple of data samples from $\mathbb{R}^d$ and let $f(x), f(y)$ be their k-dimensional representation generated by any of the methods described in this paper, then:

$$dist_{x,y} = \frac{abs(||f(x) - f(y)||^2 - ||\phi(x) - \phi(y)||^2)}{||\phi(x) - \phi(y)||^2} \tag{18}$$

This measure can be interpreted easily. For example, if $dist_{x,y} = 0.11$ we can conclude that the distance between both samples in $\mathcal{H}$ suffered a 11% distortion (i.e. an increase or decrease) in the resulting feature space. Conveniently, the computation of $||\phi(x) - \phi(y)||^2$ can be done without any explicit evaluation of $\phi(\cdot)$, via the kernel function [30]:

$$||\phi(x) - \phi(y)||^2 = -2K(x, y) + K(x, x) + K(y, y) \tag{19}$$

To measure the distortion induced by a given method while transforming a set of n samples, the average distortion among all the $\binom{n}{2}$ possible pairs of different samples is computed. For this reason, we used the average distortion measure to compare the different approaches described in this paper in terms of distance preservation. To evaluate them, the different methods were first provided with the corresponding training set. Next, 500 samples were selected at random from the test-subset of each dataset and transformed by means of each competing method. The induced distortion was then computed and averaged for the $\binom{500}{2}$ possible pairs of test samples.

As stated above, we also evaluated to what extent the different methods can be used to boost the accuracy of linear classifiers by generating a data representation where linear classifiers can approximate the accuracy of their non-linear counterparts. To this extent, each method was trained on the corresponding training set. Next, it was used to transform both the training set and the complete test set. A linear SVM was then trained[9] on that representation and its classification accuracy was evaluated. To evaluate the improvement in the classification accuracy, we also provide the resulting accuracy of training the linear SVM directly on the original features of each dataset.

To mitigate the non-deterministic nature of some of the evaluated methods, which might negatively affect the significance of our results, the above described evaluation protocol was run ten times. As a consequence, all the results reported in this section consist of the average and standard deviation of the corresponding metric over those ten runs. For a fair comparison, all the experiments in this paper were carried out on the same machine, equipped with an Intel i7-6700K processor and 16GB of DDR4 RAM. It is also worth noticing that, to ease the visualization of results in tables, each cell is colored according to the reported score (lighter is better in all tables).

4.1. MNIST dataset

The database used for the first set of experiments is MNIST [32]. This database consists of a collection of images of handwritten digits and has been extensively used in optical character recognition and machine learning research. It contains a total of 70,000 images, each of 28 × 28 dimension. The digits are size-normalized and centered on the center of gravity of the intensity in the image. A predefined train/test split is usually used with 60,000 images for training and 10,000 for testing.

4.1.1. Distance preservation on MNIST

First, we evaluate the different algorithms in terms of pairwise distance preservation. We do so for the two most frequently used polynomial degrees, namely $g = 2$ and $g = 3$. We also measured the time required to train each algorithm and transform 500 test samples, reporting the average time required by each method. For both KG-RP and our method, the hyperparameter p must be manually selected. Recall that, for KG-RP, hyperparameter p controls the number of samples used by the underlaying Kulis–Grauman method to estimate the mean and covariance matrix of data in the kernel feature space (see [17]). Meanwhile, in our method p controls the number of random hyperplanes used to populate S (see Section 3.2). The reason for comparing KG-RP and our method with equal p values while they have different meanings is that, for both algorithms, the value of p determines the number of evaluations of inner products involving the d-dimensional data samples during test phase (see Section 3.4). Furthermore, in both cases p controls the accuracy/efficiency trade-off of the algorithm. Due to the way these algorithms were designed, we know that higher p values will always yield better results at the expense of higher processing times. For this reason, we empirically evaluated the accuracy/efficiency trade-of that occurs when different p values are chosen. In particular, we experimented with various p values following the heuristic criterion proposed in [17]. Here, the authors advise using a $p = \mathcal{O}(\sqrt{n})$, where n is the number of training samples available. Accordingly, we experimented with $p = \frac{1}{2}\sqrt{n}, \sqrt{n}, 2\sqrt{n}$ and $4\sqrt{n}$. The hyperparameter t, which controls the number of samples used by the CLT, was set to the recommended value of 30 (see [17]). The results for the polynomial degrees 2 and 3 can be found in Tables 2 and 3 respectively.

Note that the method involving the explicit computation of $\phi(\cdot)$ for each test sample was not evaluated for $g = 3$. In the case of the MNIST dataset, storing the explicit form of test samples in the kernel feature space for $g = 2$ required approximately 1.172GB of free memory.[10] Doing so for the homogeneous poly-

[9] We used the linear SVM implementation of Liblinear [31]. An appropriate C value for the SVMs was determined by performing Cross-Validation over the training set on each case.

[10] Since $x \in \mathbb{R}^{784}$, $\phi(x) \in \mathcal{H}$ is 784^g-dimensional. In the case of the homogeneous polynomial kernel of degree 2, $\mathcal{H}$ is 614656-dimensional. As a consequence, the storage of 500 samples, assuming that a 4-byte float format is used, takes 1171MB of memory.

Table 2
Results on distance preservation from the homogeneous polynomial kernel of degree two ($g = 2$) for 500 samples from **MNIST**.

Method	Parameters	200 output dim.		500 output dim.		1000 output dim.	
		avg. dist.	time	avg. dist.	time	avg. dist.	time
$\phi(\cdot)$+RP	Gaussian	0.079±0.002	4.416s	0.050±0.002	10.124s	0.035±0.001	19.724s
$\phi(\cdot)$+RP	Sparse, s=1	0.080±0.003	4.314s	0.051±0.002	9.893s	0.036±0.001	19.338s
$\phi(\cdot)$+RP	Sparse, s=3	0.080±0.002	3.616s	0.050±0.002	8.899s	0.036±0.001	19.495s
Nyström [22]	-	0.259±0.002	0.064s	0.155±0.001	0.102s	0.101±0.001	0.309s
P-RP [18]	s=1	0.085±0.002	0.121s	0.054±0.002	0.311s	0.038±0.001	0.651s
Proposed KRP	Gaussian, p=16000	0.082±0.004	0.583s	0.053±0.002	0.611s	0.038±0.002	0.609s
Proposed KRP	Gaussian, p=8000	0.083±0.003	0.299s	0.055±0.003	0.315s	0.040±0.002	0.334s
Proposed KRP	Gaussian, p=3000	0.087±0.003	0.122s	0.056±0.002	0.134s	0.046±0.005	0.152s
Proposed KRP	Gaussian, p=976	0.092±0.007	0.039s	0.079±0.018	0.054s	0.059±0.007	0.079s
Proposed KRP	Sparse, s=1, p=976	0.094±0.004	0.038s	0.068±0.008	0.050s	0.059±0.008	0.075s
Proposed KRP	Sparse, s=3, p=976	0.098±0.007	0.038s	0.072±0.009	0.051s	0.060±0.008	0.074s
KG-RP [10]	p=976	0.141±0.013	7.525s	0.127±0.011	8.602s	0.130±0.006	10.509s
Proposed KRP	Gaussian, p=488	0.106±0.008	0.025s	0.102±0.029	0.036s	0.095±0.038	0.058s
Proposed KRP	Sparse, s=1, p=488	0.101±0.004	0.023s	0.088±0.012	0.037s	0.082±0.017	0.058s
Proposed KRP	Sparse, s=3, p=488	0.113±0.015	0.022s	0.095±0.019	0.037s	0.092±0.021	0.061s
KG-RP [10]	p=488	0.207±0.019	1.957s	0.211±0.005	2.378s	0.210±0.004	3.136s
Proposed KRP	Gaussian, p=244	0.120±0.006	0.015s	0.106±0.018	0.028s	0.112±0.024	0.053s
Proposed KRP	Sparse, s=1, p=244	0.132±0.018	0.015s	0.126±0.028	0.027s	0.101±0.020	0.050s
Proposed KRP	Sparse, s=3, p=244	0.122±0.010	0.016s	0.116±0.027	0.030s	0.118±0.027	0.055s
KG-RP [10]	p=244	0.329±0.014	0.546s	0.329±0.006	0.714s	0.328±0.005	1.018s
Proposed KRP	Gaussian, p=122	0.177±0.052	0.011s	0.151±0.027	0.026s	0.135±0.014	0.051s
Proposed KRP	Sparse, s=1, p=122	0.152±0.028	0.011s	0.136±0.013	0.024s	0.159±0.038	0.045s
Proposed KRP	Sparse, s=3, p=122	0.177±0.046	0.012s	0.150±0.033	0.024s	0.159±0.027	0.045s
KG-RP [10]	p=122	0.503±0.012	0.180s	0.497±0.008	0.268s	0.499±0.005	0.427s

Table 3
Results on distance preservation from the homogeneous polynomial kernel of degree three ($g = 3$) for 500 samples from **MNIST**.

Method	Parameters	200 output dim.		500 output dim.		1000 output dim.	
		avg. dist.	time	avg. dist.	time	avg. dist.	time
Nyström [22]	-	0.372±0.002	0.065s	0.255±0.001	0.116s	0.187±0.001	0.334s
Proposed KRP	Gaussian, p=976	0.119±0.019	0.045s	0.095±0.023	0.060s	0.092±0.031	0.089s
Proposed KRP	Sparse, s=1, p=976	0.109±0.009	0.040s	0.092±0.018	0.055s	0.080±0.016	0.088s
Proposed KRP	Sparse, s=3, p=976	0.114±0.020	0.039s	0.117±0.057	0.057s	0.088±0.032	0.082s
KG-RP [10]	p=976	0.215±0.008	7.578s	0.212±0.008	8.708s	0.214±0.006	10.576s
Proposed KRP	Gaussian, p=488	0.153±0.031	0.029s	0.131±0.040	0.043s	0.108±0.046	0.072s
Proposed KRP	Sparse, s=1, p=488	0.148±0.035	0.024s	0.127±0.041	0.042s	0.110±0.024	0.071s
Proposed KRP	Sparse, s=3, p=488	0.134±0.031	0.027s	0.124±0.026	0.041s	0.131±0.055	0.067s
KG-RP [10]	p=488	0.298±0.012	2.001s	0.306±0.006	2.415s	0.305±0.008	3.145s
Proposed KRP	Gaussian, p=244	0.164±0.025	0.018s	0.141±0.036	0.034s	0.168±0.066	0.062s
Proposed KRP	Sparse, s=1, p=244	0.192±0.075	0.018s	0.162±0.053	0.033s	0.159±0.053	0.063s
Proposed KRP	Sparse, s=3, p=244	0.209±0.097	0.018s	0.169±0.040	0.034s	0.149±0.049	0.061s
KG-RP [10]	p=244	0.421±0.011	0.560s	0.424±0.004	0.731s	0.427±0.007	1.020s
Proposed KRP	Gaussian, p=122	0.249±0.064	0.013s	0.250±0.112	0.032s	0.241±0.083	0.059s
Proposed KRP	Sparse, s=1, p=122	0.253±0.080	0.013s	0.247±0.119	0.030s	0.241±0.090	0.059s
Proposed KRP	Sparse, s=3, p=122	0.232±0.093	0.013s	0.213±0.038	0.029s	0.271±0.172	0.057s
KG-RP [10]	p=122	0.587±0.009	0.180s	0.587±0.009	0.277s	0.581±0.008	0.424s

nomial kernel of degree 3 would have required almost a terabyte of main memory, which is nearly intractable even for specialized high-performance computing systems.

Not surprisingly, the results obtained using $\phi(\cdot)$+RP are the best in all cases. However, the explicit computation of the feature mapping comes at a great cost. Looking at the computation times of this approach we see that, in all cases, transforming 500 MNIST samples took longer than one minute. In addition, as previously explained, this approach becomes intractable for polynomial degrees greater than two. For its part, Nyström aims to preserve inner products rather than pairwise distances. However, a close relation between both operators exists. As a consequence, our experiments show that Nyström was able to approximately preserve the pairwise distances, but induced a significantly higher distortion than the other methods. In addition, the distance preservation capabilities of Nyström seem to be highly dependent on the output dimension, which is an important drawback since Nyström's time complexity scales polynomially with this hyperparameter.

Finally, KG-RP and our proposed KRP method try to approximate $\phi(\cdot)$+RP while avoiding the expensive computation of the

Table 4
Classification accuracies on **MNIST** obtained by a linear SVM trained on the representations generated by different methods to approximate the feature space of the homogeneous polynomial kernel of degree two.

F. Extraction	Parameters	1500 output dim.			2000 output dim.		
		Acc. (%)	Train	Transform	Acc. (%)	Train	Transform
Nyström [22]	-	97.67±0.05	0.957s	3.510s	97.96±0.02	2.434s	5.394s
$\phi(\cdot)$+RP	Gaussian,	96.98±0.09	22.6s	4h 38m	97.40±0.07	34.8s	6h 10m
P-RP [18]	t=10	96.96±0.04	0.172s	18.65s	97.33±0.12	0.302s	45.19s
Proposed KRP	Gaussian, t=10, p=488	96.98±0.11	0.012s	3.162s	97.31±0.05	0.013s	4.073s
Proposed KRP	Sparse, s=1, t=10, p=488	96.97±0.11	0.010s	3.165s	97.37±0.07	0.012s	4.067s
Proposed KRP	Sparse, s=3, t=10, p=488	97.03±0.20	0.011s	3.165s	97.30±0.14	0.012s	4.067s
KG-RP [10]	t=10, p=488	96.15±0.04	3.918s	1.652s	96.27±0.11	4.577s	2.036s
Proposed KRP	Gaussian, t=1, p=488	96.95±0.12	0.012s	1.409s	97.31±0.13	0.011s	1.736s
Proposed KRP	Sparse, s=1, t=1, p=488	96.90±0.07	0.010s	1.390s	97.33±0.08	0.010s	1.738s
Proposed KRP	Sparse, s=3, t=1, p=488	97.00±0.08	0.011s	1.404s	97.23±0.08	0.010s	1.729s
KG-RP [10]	t=1, p=488	95.95±0.07	3.833s	1.652s	96.09±0.08	4.626s	2.018s
Proposed KRP	Gaussian, t=10, p=244	97.01±0.08	0.006s	2.950s	97.31±0.09	0.008s	3.859s
Proposed KRP	Sparse, s=1, t=10, p=244	96.91±0.11	0.006s	2.958s	97.30±0.06	0.006s	3.872s
Proposed KRP	Sparse, s=3, t=10, p=244	96.93±0.13	0.006s	2.956s	97.21±0.13	0.006s	3.861s
KG-RP [10]	t=10, p=244	94.19±0.26	1.295s	1.121s	94.44±0.23	1.586s	1.374s
Proposed KRP	Gaussian, t=1, p=244	96.84±0.09	0.006s	1.204s	97.20±0.09	0.006s	1.511s
Proposed KRP	Sparse, s=1, t=1, p=244	96.86±0.11	0.005s	1.186s	97.17±0.12	0.005s	1.513s
Proposed KRP	Sparse, s=3, t=1, p=244	96.85±0.16	0.006s	1.208s	97.24±0.06	0.005s	1.504s
KG-RP [10]	t=1, p=244	94.48±0.13	1.294s	1.122s	94.46±0.25	1.583s	1.372s
Proposed KRP	Gaussian, t=10, p=122	96.78±0.12	0.003s	2.869s	96.96±0.06	0.004s	3.750s
Proposed KRP	Sparse, s=1, t=10, p=122	96.70±0.17	0.003s	2.903s	97.00±0.20	0.003s	3.752s
Proposed KRP	Sparse, s=3, t=10, p=122	96.79±0.09	0.003s	2.834s	96.93±0.05	0.003s	3.752s
KG-RP [10]	t=10, p=122	91.96±0.17	0.572s	0.827s	92.10±0.10	0.714s	1.044s
Proposed KRP	Gaussian, t=1, p=122	96.65±0.11	0.004s	1.096s	96.82±0.14	0.003s	1.418s
Proposed KRP	Sparse, s=1, t=1, p=122	96.59±0.11	0.003s	1.073s	96.85±0.12	0.003s	1.422s
Proposed KRP	Sparse, s=3, t=1, p=122	96.59±0.10	0.003s	1.088s	96.91±0.11	0.003s	1.414s
KG-RP [10]	t=1, p=122	92.06±0.20	0.565s	0.815s	92.19±0.23	0.707s	1.051s

feature mapping. KG-RP exhibited a high dependence on the value of the hyperparameter p. Unfortunately, the computational cost of KG-RP grows fast with the value of this hyperparameter (see Section 3.4). Moreover, even if high p values are used, KG-RP's distance preservation results are significantly worse than those of our proposed method. The proposed KRP version provides the best approximation of $\phi(\cdot)$+RP with very low computational requirements. As expected, if a sufficiently high p is used, the proposed KRP method induces an average distortion in pairwise distances almost as small as the explicit approach. Conveniently, this approximation is achieved with a very small computational cost (e.g. it only takes 70 ms to train the algorithm and project 500 MNIST samples to $\mathbb{R}^{1000}$). In this case, P-RP seems to slightly outperform our proposed method when $p < 1000$, at the cost of much greater computation times. However, increasing p above that threshold enables our method to match the accuracy of P-RP, sacrificing some of its efficiency. Also note that P-RP was only evaluated for $g = 2$, as it is only compatible with second degree polynomial kernels.

4.1.2. Classification on MNIST

Here we evaluate to which extent the different methods can be used to boost the accuracy of linear classifiers. In this case, we experimented with different p hyperparameter values and also with different t values.[11] The resulting classification accuracies and their standard deviations can be found in Table 4, where we also provide the computation times required to train each method and to use it to transform the MNIST training set. It is worth noticing that the accuracy of a linear SVM classifier trained on the original MNIST samples is 91.81% (using the implementation of Liblinear [31] with $C = 0.5$) and the accuracy of a 2nd-degree polynomial-kernel SVM is 97.84% ($C = 0.5$). For completeness, we trained a SVM with the Gaussian kernel, which achieved a 98.56% accuracy ($C = 5$, $\gamma = 0.02$).

As we can see, the highest accuracies for both 1500 and 2000 output dimensions were achieved with Nyström. The explicit $\phi(\cdot)$+RP approach yielded slightly lower accuracies, with the gap being smaller when using 2000 output features. As expected, the computational time required by this approach was several orders of magnitude higher than that of the other methods. As both KRP and KG-RP try to approximate the computations performed in the explicit approach, we cannot expect them to outperform Nyström in this case. Regarding KRP, if a reasonably big p is used our method achieves the same classification accuracy as the explicit approach. The accuracies achieved by P-RP were similar to those of KRP, but again with a computational cost one order of magnitude larger.

We found that the impact of t on the classification accuracy is almost neglectable, and thus we recommend using a small value. In this regard, Fig. 4 shows the effect of varying t while keeping the remaining parameters fixed. As in the previous set of experiments, our method is the most efficient alternative, especially in the training phase where it is orders-of-magnitude faster than alternative methods. In this case, KG-RP fails to approximate the accuracies obtained by the explicit approach, even for the highest p values evaluated.

[11] Note that p and t hyperparameters are only used by two of the evaluated methods, namely KG-RP and our proposed KRP version (see Section 3 for more details).

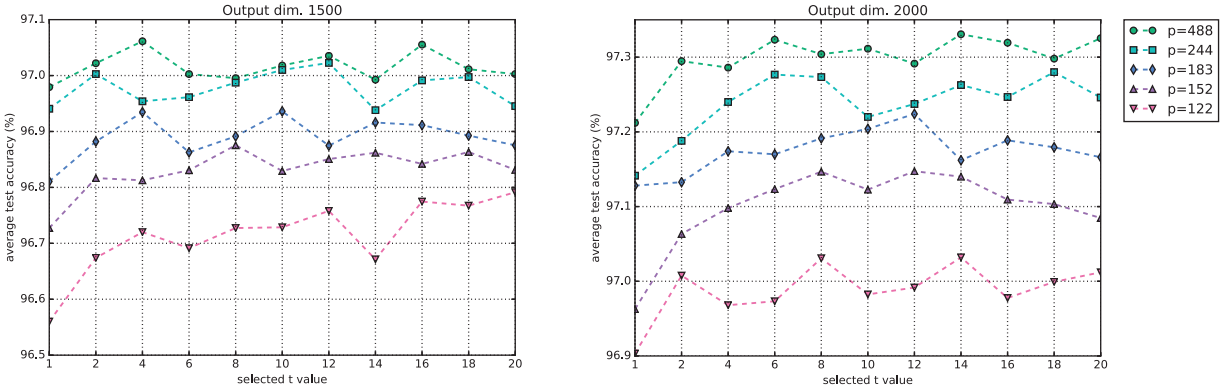

Fig. 4. Effect on the classification accuracy of varying the value of t while using a fixed output dimension and p hyperparameter value. The accuracies were obtained by applying the proposed method with $g = 2$ followed by a linear SVM on the **MNIST** dataset, averaging the resulting accuracies of each experiment over 15 runs.

Table 5
Results on distance preservation from the homogeneous polynomial kernel of degree two ($g = 2$) for 500 samples from **Webspam**.

Method	Parameters	200 output dim.		500 output dim.		1000 output dim.	
		avg. dist.	time	avg. dist.	time	avg. dist.	time
$\phi(\cdot)$+RP	Gaussian	0.078±0.007	7.358s	0.052±0.005	7.961s	0.036±0.004	8.991s
$\phi(\cdot)$+RP	Sparse, s=1	0.079±0.008	7.349s	0.049±0.004	7.915s	0.034±0.003	8.927s
$\phi(\cdot)$+RP	Sparse, s=3	0.084±0.007	7.265s	0.052±0.007	7.830s	0.036±0.004	8.798s
Nyström [22]	-	0.037±0.003	0.087s	0.014±0.000	0.124s	0.006±0.000	0.296s
P-RP [18]	s=1	0.170±0.016	0.038s	0.159±0.021	0.132s	0.145±0.008	0.288s
Proposed KRP	Gaussian, p=2116	0.082±0.011	0.034s	0.060±0.009	0.047s	0.051±0.015	0.068s
Proposed KRP	Sparse, s=1, p=2116	0.093±0.014	0.031s	0.057±0.005	0.046s	0.051±0.012	0.067s
Proposed KRP	Sparse, s=3, p=2116	0.086±0.010	0.034s	0.060±0.006	0.049s	0.049±0.005	0.069s
KG-RP [10]	p=2116	0.081±0.006	36.440s	0.054±0.005	41.336s	0.041±0.006	48.975s
Proposed KRP	Gaussian, p=1058	0.094±0.011	0.020s	0.067±0.011	0.034s	0.052±0.008	0.056s
Proposed KRP	Sparse, s=1, p=1058	0.088±0.007	0.019s	0.073±0.022	0.035s	0.054±0.009	0.055s
Proposed KRP	Sparse, s=3, p=1058	0.095±0.018	0.019s	0.066±0.004	0.035s	0.068±0.026	0.055s
KG-RP [10]	p=1058	0.085±0.006	8.683s	0.063±0.009	10.010s	0.047±0.006	12.305s
Proposed KRP	Gaussian, p=529	0.113±0.020	0.015s	0.086±0.013	0.027s	0.068±0.010	0.049s
Proposed KRP	Sparse, s=1, p=529	0.103±0.019	0.015s	0.075±0.012	0.028s	0.074±0.034	0.048s
Proposed KRP	Sparse, s=3, p=529	0.105±0.019	0.013s	0.085±0.018	0.027s	0.076±0.020	0.048s
KG-RP [10]	p=529	0.099±0.017	2.243s	0.082±0.011	2.754s	0.066±0.011	3.557s
Proposed KRP	Gaussian, p=264	0.114±0.016	0.010s	0.096±0.018	0.025s	0.088±0.012	0.047s
Proposed KRP	Sparse, s=1, p=264	0.113±0.044	0.011s	0.095±0.015	0.024s	0.080±0.009	0.045s
Proposed KRP	Sparse, s=3, p=264	0.138±0.034	0.011s	0.111±0.024	0.025s	0.104±0.025	0.045s
KG-RP [10]	p=264	0.140±0.013	0.620s	0.141±0.012	0.813s	0.131±0.013	1.121s

4.2. Webspam dataset

The *Webspam* dataset [33] compiles thousands of web pages categorized as spam or legitimate. The goal of its creators was to facilitate research on web spam detection algorithms by providing a large-scale, publicly available dataset. A refined version of this dataset, used in [11], can be found at the *LIBSVM tools* webpage [34]. This subset consists of uni-gram count features for 350,000 websites. Each sample was normalized to unit-length and the number of features for each sample is 254. As opposed to MNIST, this dataset does not come with predefined training and testing sets. For this reason, we used a 80/20 random split for training and testing, as done in [11]. Hence, the training and testing datasets consist of 280,000 and 70,000 samples respectively.

4.2.1. Distance preservation on Webspam

Tables 5 and 6 compile the results concerning the average distance distortion, obtained when transforming 500 samples from the Webspam dataset with the homogeneous polynomial kernels of degrees two and three respectively. A value of $t = 30$ was used for KG-RP and KRP in all cases.

Interestingly, in this case Nyström provided the best results in terms of pairwise distance preservation form the kernel feature space. It even outperformed the explicit approach of $\phi(\cdot)$ + RP. This result is certainly surprising because, by its mathematical formulation, Nyström seeks to preserve inner products rather than Euclidean distances. The success of this method when evaluated on the Webspam dataset contrasts with the results obtained in our experiments with other datasets, where Nyström never outperformed $\phi(\cdot)$ + RP or KRP. Nevertheless, the previously mentioned limitation regarding the scalability of Nyström holds, and the computational time needed to train the algorithm and transform 500 samples grows polynomially with the desired output dimension.

The results obtained using $\phi(\cdot)$+RP were as good as expected. However, once more computational costs render this approach impractical. Even with the relatively low original dimension of Webspam samples, the explicit approach consumes up to 9 s to initialize its projection matrices and transform 500 samples.

Table 6
Results on distance preservation from the homogeneous polynomial kernel of degree three ($g = 3$) for 500 samples from **Webspam**.

Method	Parameters	200 output dim.		500 output dim.		1000 output dim.	
		avg. dist.	time	avg. dist.	time	avg. dist.	time
Nyström [22]	-	0.067±0.001	0.097s	0.032±0.001	0.154s	0.018±0.001	0.379s
Proposed KRP	Gaussian, p=2116	0.098±0.014	0.040s	0.075±0.009	0.057s	0.064±0.016	0.083s
Proposed KRP	Sparse, s=1, p=2116	0.100±0.012	0.035s	0.086±0.044	0.055s	0.064±0.029	0.082s
Proposed KRP	Sparse, s=3, p=2116	0.096±0.011	0.039s	0.096±0.045	0.056s	0.081±0.032	0.082s
KG-RP [10]	p=2116	0.087±0.008	37.241s	0.061±0.013	41.357s	0.047±0.004	50.742s
Proposed KRP	Gaussian, p=1058	0.114±0.017	0.027s	0.079±0.020	0.042s	0.080±0.021	0.070s
Proposed KRP	Sparse, s=1, p=1058	0.102±0.014	0.023s	0.095±0.030	0.042s	0.085±0.036	0.070s
Proposed KRP	Sparse, s=3, p=1058	0.112±0.019	0.023s	0.082±0.013	0.039s	0.088±0.029	0.072s
KG-RP [10]	p=1058	0.092±0.009	8.867s	0.069±0.011	10.138s	0.057±0.004	12.319s
Proposed KRP	Gaussian, p=529	0.163±0.051	0.019s	0.108±0.026	0.037s	0.123±0.059	0.063s
Proposed KRP	Sparse, s=1, p=529	0.126±0.020	0.017s	0.141±0.076	0.034s	0.097±0.027	0.062s
Proposed KRP	Sparse, s=3, p=529	0.147±0.042	0.017s	0.094±0.024	0.035s	0.114±0.050	0.062s
KG-RP [10]	p=529	0.111±0.010	2.320s	0.086±0.011	2.778s	0.083±0.007	3.598s
Proposed KRP	Gaussian, p=264	0.198±0.132	0.013s	0.175±0.069	0.032s	0.189±0.098	0.062s
Proposed KRP	Sparse, s=1, p=264	0.166±0.077	0.014s	0.172±0.066	0.029s	0.137±0.056	0.061s
Proposed KRP	Sparse, s=3, p=264	0.209±0.067	0.013s	0.146±0.049	0.032s	0.120±0.033	0.062s
KG-RP [10]	p=264	0.172±0.017	0.658s	0.157±0.016	0.836s	0.162±0.009	1.141s

Regarding KG-RP and our proposed KRP method, our results suggest that they are evenly matched when it comes to approximating the pairwise-distance preservation capabilities of the explicit approach. However, if computational requirements are considered, our proposed approach offers a significantly better option, as in this case it can provide similar distance preservation results while keeping computation times under 80 ms (while KG-RP times range from half a second to almost one minute). In this case P-RP performed poorly, while also being significantly more computationally expensive than KRP.

4.2.2. Classification on Webspam

As with the previous dataset, we experimented with different p hyperparameter values and also with different t values. The resulting classification accuracies and their standard deviations are shown in Table 7. The table also provides the computation times required to train each method and to use it to transform the Webspam training set. Note that the accuracy of a linear SVM classifier trained on the original Webspam dataset is 92.55% (with $C = 4$) and the accuracy of a 2nd-degree polynomial-kernel SVM is 98.4% ($C = 512$). A SVM with the Gaussian kernel achieves a 99.23% accuracy ($C = 8,\ \gamma = 32$).

Again, the highest accuracies for both 1500 and 2000 output dimensions were achieved with Nyström. However, in this case the accuracy difference between $\phi(\cdot)$+RP and Nyström was almost neglectable ($\approx$0.09%). As expected, both KRP and KG-RP approximate the accuracy achieved by the explicit approach, which brings them very close to the accuracy obtained by the wining method, Nyström. For instance, using KRP with the Gaussian distribution, $k = 1500,\ p = 529$ and $t = 1$, one can achieve a linear classification accuracy of 97.80%, which is only 0.19% below the accuracy of Nyström for than same number of output dimensions. However, KRP achieves this with a training time two orders of magnitude lower and by using one-third the time to transform the samples. The accuracies obtained with KG-RP are slightly lower than those of KRP, and the difference increased when using low p values. Also, the computation times of KG-RP are significantly greater. P-RP performed comparably to KRP, but as in the previous experiments this performance came with a computational cost approximately ten times that of KPR.

4.3. w8a Dataset

The *w8a* dataset [35] is a widely used [36,37] web-classification dataset in the context of machine learning research. Conveniently, it is publicly available and can be downloaded from the *LIBSVM tools* web page [34]. Each sample in the dataset consist of a number of binary features which represent the presence/absence of a set of keywords in the web page associated to the sample. The dataset contains a total of 64,000 samples, with 300 features each. Predefined training and testing sets are usually used for evaluation with 49,749 and 14,951 samples respectively. As opposed to the other datasets used in this paper, *w8a* exhibits a significant imbalance in the distribution of class labels, which makes it much more challenging for algorithms which depend on correctly estimating the distribution of data for their proper operation.

4.3.1. Distance preservation on w8a

Tables 8 and 9 list the results concerning the average distance distortion obtained when transforming 500 samples from the *w8a* dataset with the homogeneous polynomial kernels of degrees two and three respectively. A value of $t = 30$ was used for KG-RP and KRP in all cases.

In this case, both the KG-RP and Nyström methods fail to preserve pairwise distances. This is probably due to the fact that both methods rely on estimating the distribution of the population of samples and the significant amount of class imbalance exhibited by the *w8a* dataset. One more time, $\phi(\cdot)$+RP produced the best results regarding distance preservation, at the cost of large processing times. Finally, the approach proposed in this paper approximated reasonably well the distance preservation properties of $\phi(\cdot)$+RP, while keeping computational times always below 70 ms. Again, P-RP performed poorly, while also being significantly more computationally expensive than KRP.

4.3.2. Classification on w8a

Finally, we present the classification results of the different methods on the *w8a* dataset. Again, we experimented with different p and t hyperparameter values. Due to the class imbalance in *w8a*, the accuracy is not the appropriate metric for measuring the performance of classification methods on this dataset. Instead, we

Table 7
Classification accuracies on **Webspam** obtained by a linear SVM trained on the representations generated by different methods to approximate the feature space of the homogeneous polynomial kernel of degree two.

F. Extraction	Parameters	1500 output dim.			2000 output dim.		
		Acc. (%)	Train	Transform	Acc. (%)	Train	Transform
Nyström [22]	-	97.99±0.04	3.071s	11.803s	98.23±0.02	3.745s	19.466s
$\phi(\cdot)$+RP	Gaussian	97.90±0.03	2.47s	2h 11m	98.15±0.03	3.17s	2h 54m
P-RP [18]	s=1,t=10	97.80±0.01	0.0631s	50.92s	98.08±0.01	0.080s	154.85s
Proposed KRP	Gaussian, t=10, p=1058	97.81±0.03	0.009s	8.232s	98.02±0.03	0.009s	10.656s
Proposed KRP	Sparse, s=1, t=10, p=1058	97.80±0.04	0.009s	8.204s	98.02±0.07	0.007s	10.652s
Proposed KRP	Sparse, s=3, t=10, p=1058	97.79±0.02	0.009s	8.215s	98.01±0.05	0.007s	10.678s
KG-RP [10]	t=10, p=1058	97.66±0.02	14.901s	11.321s	97.64±0.07	17.328s	14.485s
Proposed KRP	Gaussian, t=1, p=1058	97.79±0.06	0.009s	3.825s	97.98±0.03	0.009s	5.061s
Proposed KRP	Sparse, s=1, t=1, p=1058	97.70±0.09	0.009s	3.844s	98.04±0.06	0.007s	5.079s
Proposed KRP	Sparse, s=3, t=1, p=1058	97.83±0.06	0.009s	3.832s	98.03±0.04	0.007s	5.032s
KG-RP [10]	t=1, p=1058	97.27±0.13	14.915s	10.620s	97.42±0.07	17.844s	14.488s
Proposed KRP	Gaussian, t=10, p=529	97.82±0.05	0.006s	7.750s	98.03±0.06	0.007s	10.067s
Proposed KRP	Sparse, s=1, t=10, p=529	97.82±0.03	0.008s	7.692s	98.04±0.05	0.008s	10.071s
Proposed KRP	Sparse, s=3, t=10, p=529	97.79±0.03	0.007s	7.686s	98.01±0.04	0.008s	10.095s
KG-RP [10]	t=10, p=529	96.75±0.10	4.852s	6.404s	96.72±0.02	5.529s	9.058s
Proposed KRP	Gaussian, t=1, p=529	97.80±0.03	0.005s	3.319s	98.05±0.02	0.006s	4.251s
Proposed KRP	Sparse, s=1, t=1, p=529	97.77±0.04	0.006s	3.319s	98.01±0.06	0.007s	4.264s
Proposed KRP	Sparse, s=3, t=1, p=529	97.80±0.04	0.006s	3.295s	98.02±0.05	0.006s	4.234s
KG-RP [10]	t=1, p=529	96.63±0.13	4.860s	6.511s	96.69±0.09	5.486s	9.534s
Proposed KRP	Gaussian, t=10, p=264	97.81±0.03	0.005s	7.420s	98.00±0.05	0.006s	9.924s
Proposed KRP	Sparse, s=1, t=10, p=264	97.80±0.01	0.008s	7.426s	98.00±0.02	0.009s	9.891s
Proposed KRP	Sparse, s=3, t=10, p=264	97.79±0.06	0.008s	7.410s	98.02±0.02	0.009s	9.850s
KG-RP [10]	t=10, p=264	95.15±0.11	1.662s	5.047s	95.24±0.06	2.006s	6.603s
Proposed KRP	Gaussian, t=1, p=264	97.76±0.03	0.005s	3.070s	98.01±0.02	0.005s	3.988s
Proposed KRP	Sparse, s=1, t=1, p=264	97.75±0.10	0.007s	2.986s	97.94±0.06	0.009s	3.972s
Proposed KRP	Sparse, s=3, t=1, p=264	97.81±0.06	0.007s	3.007s	98.00±0.02	0.008s	3.961s
KG-RP [10]	t=1, p=264	95.24±0.07	1.677s	4.913s	95.21±0.13	1.968s	6.098s

Table 8
Results on distance preservation from the homogeneous polynomial kernel of degree two ($g = 2$) for 500 samples from **w8a**.

Method	Parameters	200 output dim.		500 output dim.		1000 output dim.	
		avg. dist.	time	avg. dist.	time	avg. dist.	time
$\phi(\cdot)$+RP	Gaussian	0.079±0.003	9.693s	0.051±0.002	10.532s	0.036±0.001	11.951s
$\phi(\cdot)$+RP	Sparse, s=1	0.077±0.002	9.684s	0.049±0.001	10.474s	0.035±0.001	11.890s
$\phi(\cdot)$+RP	Sparse, s=3	0.079±0.001	9.561s	0.050±0.001	10.351s	0.036±0.001	11.687s
Nyström [22]	-	0.772±0.005	0.024s	0.633±0.008	0.056s	0.507±0.006	0.200s
P-RP [18]	s=1	0.131±0.005	0.077s	0.112±0.003	0.334s	0.107±0.003	0.331s
Proposed KRP	Gaussian, p=892	0.105±0.005	0.021s	0.084±0.004	0.034s	0.078±0.002	0.054s
Proposed KRP	Sparse, s=1, p=892	0.103±0.002	0.019s	0.080±0.004	0.035s	0.071±0.005	0.055s
Proposed KRP	Sparse, s=3, p=892	0.107±0.003	0.021s	0.085±0.004	0.033s	0.079±0.005	0.053s
KG-RP [10]	p=892	0.554±0.009	6.744s	0.547±0.007	8.286s	0.548±0.007	10.891s
Proposed KRP	Gaussian, p=446	0.126±0.009	0.015s	0.109±0.005	0.029s	0.099±0.005	0.049s
Proposed KRP	Sparse, s=1, p=446	0.126±0.005	0.015s	0.104±0.005	0.026s	0.098±0.008	0.047s
Proposed KRP	Sparse, s=3, p=446	0.127±0.006	0.014s	0.107±0.006	0.027s	0.101±0.005	0.048s
KG-RP [10]	p=446	0.677±0.007	1.738s	0.676±0.007	2.241s	0.674±0.005	3.045s
Proposed KRP	Gaussian, p=223	0.157±0.008	0.012s	0.141±0.005	0.023s	0.142±0.008	0.045s
Proposed KRP	Sparse, s=1, p=223	0.152±0.009	0.010s	0.135±0.006	0.027s	0.132±0.010	0.047s
Proposed KRP	Sparse, s=3, p=223	0.154±0.008	0.010s	0.151±0.015	0.025s	0.138±0.005	0.047s
KG-RP [10]	p=223	0.791±0.007	0.485s	0.791±0.006	0.652s	0.788±0.006	0.950s
Proposed KRP	Gaussian, p=111	0.218±0.031	0.009s	0.203±0.016	0.024s	0.191±0.008	0.044s
Proposed KRP	Sparse, s=1, p=111	0.195±0.009	0.009s	0.183±0.008	0.021s	0.179±0.008	0.044s
Proposed KRP	Sparse, s=3, p=111	0.218±0.017	0.009s	0.194±0.014	0.022s	0.200±0.009	0.046s
KG-RP [10]	p=111	0.887±0.008	0.160s	0.883±0.007	0.251s	0.883±0.004	0.394s

Table 9
Results on distance preservation from the homogeneous polynomial kernel of degree three ($g = 3$) for 500 samples from **w8a**.

Method	Parameters	200 output dim.		500 output dim.		1000 output dim.	
		avg. dist.	time	avg. dist.	time	avg. dist.	time
Nyström [22]	-	0.915±0.006	0.025s	0.846±0.009	0.066s	0.773±0.005	0.226s
Proposed KRP	Gaussian, t=30, p=892	0.135±0.009	0.022s	0.127±0.013	0.039s	0.112±0.006	0.067s
Proposed KRP	Sparse, s=1, t=30, p=892	0.133±0.008	0.022s	0.114±0.005	0.040s	0.105±0.004	0.063s
Proposed KRP	Sparse, s=3, t=30, p=892	0.144±0.007	0.021s	0.121±0.006	0.040s	0.111±0.007	0.067s
KG-RP [10]	t=30, p=892	0.802±0.007	6.681s	0.798±0.009	8.239s	0.798±0.007	10.862s
Proposed KRP	Gaussian, t=30, p=446	0.170±0.009	0.016s	0.161±0.011	0.034s	0.158±0.008	0.061s
Proposed KRP	Sparse, s=1, t=30, p=446	0.161±0.009	0.016s	0.145±0.004	0.032s	0.140±0.009	0.060s
Proposed KRP	Sparse, s=3, t=30, p=446	0.171±0.010	0.016s	0.162±0.010	0.033s	0.150±0.009	0.061s
KG-RP [10]	t=30, p=446	0.871±0.007	1.720s	0.867±0.007	2.236s	0.871±0.007	3.031s
Proposed KRP	Gaussian, t=30, p=223	0.241±0.020	0.013s	0.224±0.026	0.032s	0.219±0.024	0.058s
Proposed KRP	Sparse, s=1, t=30, p=223	0.213±0.010	0.013s	0.209±0.014	0.029s	0.199±0.009	0.056s
Proposed KRP	Sparse, s=3, t=30, p=223	0.228±0.012	0.012s	0.221±0.012	0.030s	0.218±0.024	0.059s
KG-RP [10]	t=30, p=223	0.924±0.005	0.487s	0.923±0.007	0.640s	0.924±0.006	0.941s
Proposed KRP	Gaussian, t=30, p=111	0.320±0.030	0.012s	0.302±0.021	0.028s	0.310±0.028	0.056s
Proposed KRP	Sparse, s=1, t=30, p=111	0.290±0.020	0.013s	0.284±0.025	0.028s	0.298±0.040	0.057s
Proposed KRP	Sparse, s=3, t=30, p=111	0.336±0.033	0.011s	0.296±0.016	0.030s	0.294±0.014	0.056s
KG-RP [10]	t=30, p=111	0.959±0.004	0.166s	0.962±0.004	0.257s	0.959±0.004	0.414s

used the F1-score:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{20}$$

which can be interpreted as an average of the precision and recall with an equal contribution of both metrics. We also report the standard deviation in the F1-score over multiple runs of the experiments, and the computation times required to train each method and to use it to transform *w8a*'s entire training set (see Table 10). Note that the F1-score of a linear SVM classifier trained on the original *w8a* samples is 0.7343 (with $C = 1$), which corresponds to a classification accuracy of 98.66%. The classification F1-score of a 2nd-degree polynomial-kernel SVM is 0.9020 (with $C = 1$). For comparison, a SVM with the Gaussian kernel achieves a 0.9037 F1-score ($C = 200,\ \gamma = 0.05$).

As opposed to what happened in the experiments with MNIST and Webspam, the classification performance with Nyström was not the best. Instead, this approach is largely outperformed by $\phi(\cdot)$+RP, with the gap between their scores being lower when using 2000 output dimensions. Similarly, KG-RP performed poorly in this dataset, supporting our hypothesis that these methods are largely affected by the imbalance of the dataset.

For its part, KRP performed remarkably well in this dataset, achieving F1-scores similar to those of $\phi(\cdot)$+RP but with a computational-time lower by orders of magnitude. For instance, by using KRP with the Gaussian distribution, $k = 2000,\ p = 446$ and $t = 1$, one can achieve a linear classification F1-score of 0.8973, which is only 0.0018 below the score of $\phi(\cdot)$+RP for the same number of output dimensions. In this case P-RP performed poorly also for classification, in spite of being significantly more computationally expensive than KRP.

4.4. Polynomial kernel degree selection

As mentioned before, due to their semantic, increasing the hyperparameters t and p for our method will always result in higher accuracies at the expense of greater computational costs. However, determining the best polynomial degree g is not a straightforward task. While in some contexts the best performing polynomial kernel degree is known based on expert knowledge, experimentation is usually needed to determine the best value for this hyperparameter. In this subsection we show how the right polynomial degree for our method can be determined by using a standard hyper-parameter selection strategy. In particular, given a desired output dimension and the value of p, the most appropriate kernel degree can be determined by executing k-fold Cross-Validation on the training set with different polynomial kernel degrees. Then, the best performing value of g according to the Cross-Validation accuracies is selected and evaluated on the test set.

For the experiments in this section, we used the binary version of the Covertype dataset [38]. The task with this dataset is to predict the forest cover-type from cartographic variables (e.g., elevation, slope, soil type, etc.). In particular, we used the pre-processed version of the dataset available at the LIBSVM web page.[12] It contains a total of 581,012 samples each of dimension 54. Since no pre-defined train/test split exists for this dataset, for our experiments we randomly sampled 20% of the data to form the test set (116,202 samples) and 10% to form the training set (58,101 samples). Table 11 shows the 5-fold Cross-Validation accuracies and the corresponding test accuracies for different polynomial degrees, values of p and output dimensions on the Covertype dataset. Looking at the table we can see that, for each output dimension and selected p combination, the best performing polynomial degree in the Cross-Validation process over the training set matches the best performing kernel as evaluated on the test set. This suggests that the most appropriate kernel degree for a specific application can be successfully determined with the above described hyperparameter selection scheme. In addition, it must be noted that the best performing polynomial degree for our method need not be the same as the best polynomial degree for a conventional kernel-SVM using a polynomial kernel. Since our method is performing a Random Projection from the implicit kernel feature space, higher polynomial degrees might require a bigger output dimension to fully capture their discriminative information, as the dimension of the implicit kernel feature space grows with the degree. Therefore, the optimal polynomial degree to be used with our method depends on the selected output dimension. For instance, we can see that in this case our method performed best using $g = 3$ only when the output dimension was at least 2000.

[12] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html.

Table 10
Classification accuracies on **w8a** obtained by a linear SVM trained on the representations generated by different methods to approximate the feature space of the homogeneous polynomial kernel of degree two.

F. Extraction	Parameters	1500 output dim.			2000 output dim.		
		F1-score	Train	Transform	F1-score	Train	Transform
Nyström [22]	-	.7849±.015	0.743s	2.117s	.8596±.008	2.127s	3.480s
$\phi(\cdot)$+RP	Gaussian	.8738±.003	10m 2.47s	32m 34s	.8991±.002	3.17s	44m 48s
P-RP [18]	s=1, t=10	.7679±.005	0.068s	8.713s	.8496±.001	0.087s	11.935s
Proposed KRP	Gaussian, t=10, p=446	.8721±.011	0.004s	2.273s	.9003±.004	0.005s	2.927s
Proposed KRP	Sparse, s=1, t=10, p=446	.8702±.007	0.004s	2.255s	.8972±.003	0.004s	2.917s
Proposed KRP	Sparse, s=3, t=10, p=446	.8785±.012	0.004s	2.265s	.8988±.002	0.004s	2.938s
KG-RP [10]	t=10, p=446	.4223±.061	3.815s	1.054s	.4423±.048	4.692s	1.352s
Proposed KRP	Gaussian, t=1, p=446	.8667±.004	0.004s	0.795s	.8973±.002	0.004s	1.014s
Proposed KRP	Sparse, s=1, t=1, p=446	.8722±.005	0.003s	0.803s	.8999±.003	0.004s	1.016s
Proposed KRP	Sparse, s=3, t=1, p=446	.8695±.006	0.003s	0.805s	.8972±.002	0.004s	1.004s
KG-RP [10]	t=1, p=446	.4916±.030	3.819s	1.056s	.4865±.031	4.625s	1.336s
Proposed KRP	Gaussian, t=10, p=223	.8690±.002	0.003s	2.167s	.8982±.002	0.003s	2.879s
Proposed KRP	Sparse, s=1, t=10, p=223	.8715±.006	0.002s	2.180s	.8984±.001	0.002s	2.903s
Proposed KRP	Sparse, s=3, t=10, p=223	.8663±.005	0.002s	2.179s	.9008±.004	0.002s	2.888s
KG-RP [10]	t=10, p=223	.3070±.033	1.208s	0.724s	.3652±.031	1.498s	0.934s
Proposed KRP	Gaussian, t=1, p=223	.8637±.011	0.002s	0.717s	.8966±.004	0.002s	0.932s
Proposed KRP	Sparse, s=1, t=1, p=223	.8585±.006	0.002s	0.719s	.8960±.002	0.002s	0.923s
Proposed KRP	Sparse, s=3, t=1, p=223	.8650±.007	0.002s	0.728s	.8985±.003	0.002s	0.933s
KG-RP [10]	t=1, p=223	.2924±.048	1.210s	0.727s	.3337±.045	1.481s	0.949s
Proposed KRP	Gaussian, t=10, p=111	.8656±.006	0.002s	2.126s	.8978±.002	0.002s	2.828s
Proposed KRP	Sparse, s=1, t=10, p=111	.8622±.006	0.002s	2.133s	.8986±.003	0.002s	2.838s
Proposed KRP	Sparse, s=3, t=10, p=111	.8667±.011	0.002s	2.128s	.8983±.003	0.002s	2.823s
KG-RP [10]	t=10, p=111	.1242±.063	0.539s	0.566s	.1757±.044	0.679s	0.744s
Proposed KRP	Gaussian, t=1, p=111	.8434±.014	0.001s	0.670s	.8884±.005	0.001s	0.883s
Proposed KRP	Sparse, s=1, t=1, p=111	.8375±.009	0.002s	0.664s	.8863±.004	0.002s	0.888s
Proposed KRP	Sparse, s=3, t=1, p=111	.8361±.009	0.002s	0.664s	.8874±.004	0.001s	0.882s
KG-RP [10]	t=1, p=111	.1572±.037	0.542s	0.555s	.1542±.073	0.665s	0.746s

Table 11
Classification accuracies for 5-fold Cross-Validation on the training set and for the test set of **Covertype**. The results show that, given an output dimension and the desired value of p, the most suitable polynomial degree g for our method can be selected by using a standard hyper-parameter selection strategy.

Parameters	1000 output dim.		2000 output dim.		3000 output dim.	
	CV Acc. (%)	Test Acc. (%)	CV Acc. (%)	Test Acc. (%)	CV Acc. (%)	Test Acc.
g=2, p=500	79.21±0.28	79.55±0.03	79.33±0.25	79.56±0.03	79.37±0.21	79.57±0.03
g=3, p=500	78.27±0.25	79.17±0.26	79.85±0.34	80.45±0.15	80.45±0.17	81.17±0.09
g=4, p=500	69.34±0.71	70.39±0.51	72.08±0.23	73.10±0.25	73.64±0.70	74.25±0.42
g=2, p=1000	79.26±0.22	79.54±0.04	79.34±0.32	79.55±0.02	79.32±0.33	79.55±0.03
g=3, p=1000	78.51±0.14	78.93±0.28	79.82±0.11	80.63±0.13	80.34±0.16	81.13±0.11
g=4, p=1000	69.83±0.75	69.85±0.62	72.22±0.78	72.92±0.23	73.69±0.25	74.29±0.28

5. Discussion and future work

This paper proposes a novel method for perform Random Projections of data samples from the feature space of the homogeneous polynomial kernel. As opposed to previous kernelization attempts of the RP algorithm [10,39], our approach preserves the data-independence and low computational complexity of the original RP method. As a drawback, this was achieved by sacrificing the generality of the method, focusing on a specific kernel family. Nevertheless, the chosen kernel family, homogeneous polynomial kernels, is one of the most popular choices and has been applied to a wide range of classification and clustering problems, especially in the field of natural language processing.

Our work is highly related to the theoretical work of Balcan et al. [14,16], which demonstrated that performing a JL-valid Random Projection from the feature space of an arbitrary kernel is generally not possible, given only black-box access to the kernel function and without access to the distribution of data. However, they hypothesized that such methods could be developed for specific natural kernel families. The proposed method confirms their hypothesis, since it approximates a Random Projection from the feature space of the homogeneous polynomial kernel without ever computing the explicit form of the feature space nor considering the distribution of data samples being processed.

Our theoretical analysis of computational complexities showed that the time required by the proposed approach grows linearly with respect to the dimensionality of samples and the desired output dimension. Also, the training time of KRP is independent of the number of training samples as opposed to KG-RP, which requires $\mathcal{O}(p^3)$ training time where p must be set considering the number

of available training samples.[13] Our method also compares favorably to Nyström, whose training and testing time grow as $\mathcal{O}(k^3)$ and $\mathcal{O}(k^2)$ respectively. Our theoretical results regarding the complexity of the alternative methods are confirmed by the experimental results on computational time, where our approach consistently reported the lowest times.

The experimental results presented in Section 4 evidence the performance of the proposed method both in terms of distance preservation and generation of useful representations for linear classification. Regarding distance preservation, the proposed approach outperformed alternative methods in most of our experiments. In terms of classification accuracy, the proposed method showed its ability to approximate the results obtained with the explicit $\phi(\cdot)$+RP approach. Since our method is an approximation of the explicit approach, KRP was outperformed by Nyström on the datasets where this approach performed better than $\phi(\cdot)$+RP.

Apart from the above mentioned advantages of the proposed method, it is also worth noticing that it works in a completely data-independent manner. That is, the algorithm can be initialized without any training sample, and the properties of the algorithm do not depend on estimating the distribution of input data. As a consequence, our method is well suited to work in online/incremental learning scenarios [40], where data samples arrive in a sequential manner.

Lastly, we proved that the kernelization approach proposed in this paper is directly compatible with the database-friendly distribution proposed by Achlioptas [7]. This property can be used to ease the implementation of this algorithm in SQL environments, as the projection of data samples over the random hyperplanes can be done in terms of aggregate evaluation. In this regard, our experimental results show no significant accuracy loss when using Achlioptas' distribution instead of the standard Gaussian distribution.

As for future lines of research, we propose exploring the development of similar kernelized variants of Random Projection with different kernel families. While in this paper we have focused on the homogeneous polynomial kernel family, it would be interesting to compare different kernel feature-space approximation methods for various kernel families. In addition, we intend to investigate the applicability of the proposed approach in different clustering, classification and information retrieval tasks, especially in domains where a limited amount of computational power is available (e.g., embedded systems and Internet of Things).

Finally, we believe that a more in-depth theoretical and empirical analysis of the distance-preservation capabilities of the Nyström algorithm is required, as we obtained some unexpected results when analyzing this aspect on the Webspam dataset.

Acknowledgments

The research of Daniel López-Sánchez has been financed by the Ministry of Education, Culture and Sports of the Spanish Government (University Faculty Training (FPU) programme, reference number FPU15/02339).

References

[1] S.S. Vempala, The Random Projection Method, 65, American Mathematical Society, 2005.
[2] H. Foroughi, N. Ray, H. Zhang, Robust people counting using sparse representation and random projection, Pattern Recognit. 48 (10) (2015) 3038–3052.
[3] Â. Cardoso, A. Wichert, Iterative random projections for high-dimensional data clustering, Pattern Recognit. Lett. 33 (13) (2012) 1749–1755.
[4] L. Liu, P. Fieguth, D. Clausi, G. Kuang, Sorted random projections for robust rotation-invariant texture classification, Pattern Recognit. 45 (6) (2012) 2405–2418.
[5] Y. Gao, X. Shan, Z. Hu, D. Wang, Y. Li, X. Tian, Extended compressed tracking via random projection based on MERs and online LS-SVM learning, Pattern Recognit. 59 (2016) 245–254.
[6] S. Dasgupta, A. Gupta, An elementary proof of a theorem of Johnson and Lindenstrauss, Random Struct. Algorithm. 22 (1) (2003) 60–65.
[7] D. Achlioptas, Database-friendly random projections, in: Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, ACM, 2001, pp. 274–281.
[8] P. Li, T.J. Hastie, K.W. Church, Very sparse random projections, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2006, pp. 287–296.
[9] A. Alavi, A. Wiliem, K. Zhao, B.C. Lovell, C. Sanderson, Random projections on manifolds of symmetric positive definite matrices for image classification, in: Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on, IEEE, 2014, pp. 301–308.
[10] K. Zhao, A. Alavi, A. Wiliem, B.C. Lovell, Efficient clustering on Riemannian manifolds: akernelised random projection approach, Pattern Recognit. 51 (2016) 333–345.
[11] Y.-W. Chang, C.-J. Hsieh, K.-W. Chang, M. Ringgaard, C.-J. Lin, Training and testing low-degree polynomial data mappings via linear SVM, J. Mach. Learn. Res. 11 (April) (2010) 1471–1490.
[12] D. Chen, Z. Yuan, J. Wang, B. Chen, G. Hua, N. Zheng, Exemplar-guided similarity learning on polynomial kernel feature map for person re-identification, Int. J. Comput. Vis. 123 (3) (2017) 392–414.
[13] A. Kabán, Improved bounds on the dot product under random projection and random sign projection, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2015, pp. 487–496.
[14] M.-F. Balcan, A. Blum, S. Vempala, Kernels as features: on kernels, margins, and low-dimensional mappings, Mach. Learn. 65 (1) (2006) 79–94.
[15] G.-X. Yuan, C.-H. Ho, C.-J. Lin, Recent advances of large-scale linear classification, Proc. IEEE 100 (9) (2012) 2584–2603.
[16] A. Blum, Random projection, margins, kernels, and feature-selection, in: Subspace, Latent Structure and Feature Selection, Springer, 2006, pp. 52–68.
[17] B. Kulis, K. Grauman, Kernelized locality-sensitive hashing, IEEE Trans. Pattern Anal. Mach. Intell. 34 (6) (2012) 1092–1104.
[18] D. López-Sánchez, J.M. Corchado, A.G. Arrieta, Data-independent random projections from the feature-map of the homogeneous polynomial kernel of degree two, Inf. Sci. 436-437C (2018) 214–226.
[19] A. Vedaldi, A. Zisserman, Efficient additive kernels via explicit feature maps, IEEE Trans. Pattern Anal. Mach. Intell. 34 (3) (2012) 480–492.
[20] I. Steinwart, A. Christmann, Support Vector Machines, Springer Science & Business Media, 2008.
[21] T. Joachims, Training linear SVMs in linear time, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2006, pp. 217–226.
[22] C.K. Williams, M. Seeger, Using the Nyström method to speed up kernel machines, in: Proceedings of the 13th International Conference on Neural Information Processing Systems, MIT press, 2000, pp. 661–667.
[23] T. Gärtner, A survey of kernels for structured data, ACM SIGKDD Explor. Newsl. 5 (1) (2003) 49–58.
[24] R. Ware, F. Lad, Approximating the Distribution for Sums of Products of Normal Variables, Technical Report UCDMS 15, University of Canterbury, England, 2003, pp. 1–50.
[25] E.W. Weisstein, Normal Product Distribution, MathWorld, A Wolfram Web Resource., 2017. (http://mathworld.wolfram.com/NormalProductDistribution.html). Accessed: 2017-03-30
[26] G.J. Székely, M.L. Rizzo, N.K. Bakirov, et al., Measuring and testing dependence by correlation of distances, Ann. Stat. 35 (6) (2007) 2769–2794.
[27] O. Kallenberg, Foundations of Modern Probability, Springer Science & Business Media, 2006.
[28] S. Kumar, M. Mohri, A. Talwalkar, Sampling methods for the Nyström method, J. Mach. Learn. Res. 13 (April) (2012) 981–1006.
[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[30] J.M. Phillips, S. Venkatasubramanian, A gentle introduction to the kernel distance, arXiv:1103.1625(2011) 1–9.
[31] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, C.-J. Lin, LIBLINEAR: a library for large linear classification, J. Mach. Learn. Res. 9 (2008) 1871–1874.
[32] C.J.B. Yann LeCun, C. Cortes, The MNIST database of handwritten digits, 1998.
[33] S. Webb, J. Caverlee, C. Pu, Introducing the webb spam corpus: using email spam to identify web spam automatically, in: Proceedings of the 3rd Conference on Email and AntiSpam (CEAS), 2006.
[34] C.-C. Chang, C.-J. Lin, LIBSVM data: classification (binary class), 2017, (https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html). [Online; accessed 30-May-2017].
[35] J.C. Platt, Fast training of support vector machines using sequential minimal optimization, in: Advances in Kernel Methods, 1999, pp. 185–208.

[13] In the case of KG-RP, the p hyperparameter controls the number of training samples used by the underlying Kulis–Grauman method to estimate the distribution of data. As a consequence, the authors of this method [17] recommended setting $p = \mathcal{O}(\sqrt{n})$.

[36] D. Chu, C. Zhang, Q. Tao, A faster cutting plane algorithm with accelerated line search for linear SVM, Pattern Recognit. 67 (2017) 127–138.
[37] J.-X. Peng, S. Ferguson, K. Rafferty, V. Stewart, A sequential algorithm for sparse support vector classifiers, Pattern Recognit. 46 (4) (2013) 1195–1208.
[38] R. Collobert, S. Bengio, Y. Bengio, A parallel mixture of SVMs for very large scale problems, in: Advances in Neural Information Processing Systems, 2002, pp. 633–640.
[39] K. Zhao, A. Wiliem, B. Lovell, Kernelised orthonormal random projection on grassmann manifolds with applications to action and gait-based gender recognition, in: Identity, Security and Behavior Analysis (ISBA), 2015 IEEE International Conference on, IEEE, 2015, pp. 1–6.
[40] O. Fontenla-Romero, B. Guijarro-Berdiñas, D. Martinez-Rego, B. Pérez-Sánchez, D. Peteiro-Barral, Online machine learning, in: Efficiency and Scalability Methods for Computational Intellect, 27, 2013, pp. 27–55.

Daniel López-Sánchez obtained a Computer Science degree (highest GPA award) in 2015 and an M.Sc. in Artificial Intelligence in 2016 at the University of Salamanca (Spain). In 2016, he received the FPU grant from the Spanish Government and started pursuing a Ph.D. at the BISITE Research Group. His research interest focus in the field of machine learning, especially in the sub-fields of dimensionality reduction and kernel methods. He is also interested in developing novel applications of the Deep Learning paradigm. He has participated as a co-author in papers published in recognized international conferences and journals.

Angélica González-Arrieta received a Ph.D. in Computer Science from the University of Salamanca in 2000. She is currently a Lecturer in Salamancas University Department of Computer Science and has attended several Masters courses. She is further a professor and tutor for UNED (Universidad Española de Educación a Distancia, Spains Open University). In the past, she carried out relevant administrative tasks, such as Academic Secretary of the Science Faculty (19962000) and Chief of Staff for the University of Salamanca (20002003). From 1990, she has cooperated with the Home Ministry, and from 2008 with the Home and Justice Counsel of the local government (Junta de Castilla y León). She is a member of the research group BISITE (http://bisite.usal.es) and has lead several research projects sponsored by both public and private institutions in Spain. She is the coauthor of several works published in magazines, workshops, meetings, and symposia.

Juan M. Corchado received a Ph.D. in Computer Science from the University of Salamanca in 1998 and a Ph.D. in Artificial Intelligence (AI) from the University of Paisley, Glasgow (UK) in 2000. At present, he is Vice President for Research and Technology Transfer since December 2013 and a Full Professor with Chair at the University of Salamanca. He is the Director of the Science Park of the University of Salamanca and Director of the Doctoral School of the University. He has been elected twice as the Dean of the Faculty of Science at the University of Salamanca. He has led several Artificial Intelligence research projects sponsored by Spanish and European public and private sector institutions and has supervised seven Ph.D. students. He is the coauthor of over 230 books, book chapters, journal papers, technical reports, etc. Since January 2015, Juan Manuel Corchado is Visiting Professor at Osaka Institute of Technology. He is also member of the Advisory group on Online Terrorist Propaganda of the European Counter Terrorism Centre (EUROPOL). Juan Manuel is also editor and Editor-in-Chief of Specialized Journals like ADCAIJ (Advances in Distributed Computing and Artificial Intelligence Journal), IJDCA (International Journal of Digital Contents and Applications) and OJCST (Oriental Journal of Computer Science and Technology).