Noname manuscript No.
(will be inserted by the editor)

Learning Deep Kernels in the Space of Dot Product Polynomials

Michele Donini · Fabio Aiolli

Received: date / Accepted: date

Abstract Recent literature has shown the merits of having deep representations in the context of neural networks. An emerging challenge in kernel learning is the definition of similar deep representations. In this paper, we propose a general methodology to define a hierarchy of base kernels with increasing expressiveness and combine them via Multiple Kernel Learning (MKL) with the aim to generate overall deeper kernels. As a leading example, this methodology is applied to learning the kernel in the space of Dot-Product Polynomials (DPPs), that is a positive combination of homogeneous polynomial kernels (HPKs). We show theoretical properties about the expressiveness of HPKs that make their combination empirically very effective. This can also be seen as learning the coefficients of the Maclaurin expansion of any definite positive dot product kernel thus making our proposed method generally applicable. We empirically show the merits of our approach comparing the effectiveness of the kernel generated by our method against baseline kernels (including homogeneous and non homogeneous polynomials, RBF, etc...) and against another hierarchical approach on several benchmark datasets.

1 Introduction

Kernel methods have become a standard paradigm in machine learning and applied in a multitude of different learning tasks. Their fortune is mainly due their ability to perform well on different domains provided that ad-hoc kernels tailored to that domain can be designed. Given the crucial importance of the kernel adopted for the performance of a kernel machine, researchers are investigating on the automatic learning of kernels, also known as kernel learning.

Multiple Kernel Learning (MKL) is one of the most popular paradigms to learn a kernel (see [10] for a recent survey) which has been adopted already in many real world applications, including [4, 25, 23, 8, 24, 5]. The main goal of MKL is to alleviate the user's effort on designing a good kernel for a given problem.

M. Donini and F. Aiolli
University of Padova - Department of Mathematics
Via Trieste, 63, 35121 Padova - Italy
E-mail: {mdonini,aiolli}@math.unipd.it

The kernel generated by a MKL method is a combination of *base* kernel functions $k_0, ..., k_R$. In the simplest case, it consists of a linear and non-negative combination of base kernels, that is,

$$k_{\boldsymbol{\eta}}(\mathbf{x}, \mathbf{z}) = \sum_{r=0}^{R} \eta_r k_r(\mathbf{x}, \mathbf{z}) = \sum_{r=0}^{R} \eta_r \langle \boldsymbol{\phi}_r(\mathbf{x}), \boldsymbol{\phi}_r(\mathbf{z}) \rangle, \qquad \eta_r \geq 0,$$

thus basically performing a re-weighting of groups of features in a compounded feature space $\boldsymbol{\phi}(\mathbf{x}) = (\boldsymbol{\phi}_0(\mathbf{x}), \ldots, \boldsymbol{\phi}_R(\mathbf{x}))$. Note that, doing MKL on such features can be seen as non-linear feature weighting. For this, often some regularization enforcing sparsity of the parameters $\boldsymbol{\eta}$ is also provided.

MKL algorithms are supported by several theoretical results bounding the difference between the true error and the empirical margin error (i.e. *estimation error*). These bounds limit the *Empirical Rademacher Complexity* (ERC) of the combination of kernels [7, 13, 6]. However, empirical studies on MKL are giving conflicting outcomes concerning the real effectiveness of MKL. For example, doing better than the simple average (or sum) of base kernels seems surprisingly challenging [22]. This can be due to two main reasons: (i) standard MKL algorithms are typically applied with base kernels which are not so dissimilar to each other and (ii) the combined shallow kernels do not have structural differences, e.g. they have the same degree of abstraction, thus producing shallow representations.

Up to now, MKL research has been mainly focused on the learning of the combination weights. In this work, we take a different perspective of the MKL problem investigating on principled ways to design base kernels such to make their supervised combination really effective. Specifically, aiming at building *deeper* kernels, a hierarchy of features of different degrees of abstraction is considered. Features at the top of the hierarchy will be more general and less expressive features, while features at the bottom of the hierarchy will be more specific and expressive features. Features are then grouped based on a general-to-specific ordering (their level in the hierarchy) and base kernels built according to this grouping, in a way that the supervised MKL algorithm can detect the most effective level of abstraction for any given task. Similarly to the hierarchical kernel learning (HKL) approach in [2], features that can be embedded in a DAG will be considered.

As a further contribution of this paper, we give a characterization of the specificity of a representation (kernel function). Intuitively, more general representations correspond to kernels constructed on *simpler* features (e.g. the single variable x_i), at the top layers of the DAG, while, more specific representations correspond to kernels defined on elaborated features (e.g. high degree product of variables $\prod_j x_j$), at the bottom layers of the DAG. The characterization is based on the *spectral ratio* of the kernel matrices obtained in the target representation. We also prove relationships between the spectral ratio of a kernel matrix with its rank, with the radius of the Minimum Enclosing Ball (MEB) of examples in feature space, and with the ERC of linear functions using that representation.

Although the idea presented above is quite general and applicable in many different contexts which include ANOVA kernels, kernels for structures, and convolution kernels in general, here we exemplify the approach focusing on features which are a special kind of monomials (that is products of powers of variables with non-negative integer exponents, possibly multiplied by a constant). See Figure 2 for an example. In this case, base kernels will consist of Homogeneous Polynomial Kernels (HPK) of different degree and their combination to a Dot-Product

Polynomial (DPP) with form $K(\mathbf{x}, \mathbf{z}) = \sum_{s=0}^{R} a_s(\mathbf{x} \cdot \mathbf{z})^s$. Exploiting the result in [18] that any dot-product kernel of the form $K(\mathbf{x}, \mathbf{z}) = f(\mathbf{x} \cdot \mathbf{z})$ can be seen as a DPP, $K(\mathbf{x}, \mathbf{z}) = \sum_{s=0}^{+\infty} a_s(\mathbf{x} \cdot \mathbf{z})^s$, it turns out that proposing a method to learn the coefficients of a general DPP means giving a method that virtually can learn any dot-product kernel, including RBF and non-homogeneous polynomials.

A related but different idea is exploited in deep neural networks. for example defining families of neural networks with polynomial activation functions, as it is done in [16]. In this approach the polynomial features are learned as non-linear combinations of the original variables.

Similarly, another example is described in [15], where the authors present an efficient deep learning algorithm (with polynomial computational time). The layers of this deep architecture are created one-by-one and the final predictors of this algorithm are polynomial functions over the input space. Specifically, they create higher-and-higher levels of representation of the data generating a good approximation of all the possible values obtained by using polynomial functions with bounded degree over the training set. The final linear combination (in the output layer) is a combination of sets of the polynomial functions that depend on the coefficient of the previous layers.

We can easily note that the feature space on which these methods work is completely different from ours. In our case, the hierarchy of features intrinsic in the polynomial kernels is used. This allows us to apply the results of this paper to other kernel functions besides the dot-product kernels and polynomials, including most of the kernels for structures.

Summarizing, the paper contribution is three-fold:

- We propose a simple to compute qualitative measure of expressiveness of a kernel defined in terms of the trace (or nuclear) and Frobenius norms of the kernel matrix generated using that kernel and we show connections with the rank of the matrix, with the radius of the MEB, and with the ERC of linear functions defined in that feature space;
- We propose a MKL based approach to learn the coefficients of general DPPs and we support the proposal by showing empirically that this approach outperforms the baselines, including RBF, homogeneous and non-homogeneous polynomials (often significantly) against several benchmark datasets in terms of classification performance. Interestingly, the method is very robust to overfitting even when many base HPK are used, which permits to spare the tedious step of validation of the kernel hyper-parameters;
- Finally, we present empirical evidence that building base kernels exploiting the structure of the features and their dependencies, makes the combined kernel improve upon alternatives which do not exploit the same structure. In particular, a comparison with the HKL approach [2, 11] on the same DPP learning task shows the advantages of our method in terms of effectiveness and efficiency.

2 Notation, Background and Related Work

In this section, we present the notation and briefly discuss some background and related work useful for the comprehension of the remainder of the paper.

Throughout the paper we consider a binary classification problem with training examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_L, y_L)\}$, $\mathbf{x}_i \in \mathbb{R}^m$, $||\mathbf{x}_i||_2 = 1$, $y_i \in \{-1, +1\}$. $\mathbf{X} \in \mathbb{R}^{L \times m}$ will denote the matrix where examples are arranged in rows and $\mathbf{y} \in \mathbb{R}^L$ the corresponding vector of labels. The symbol $\mathbf{I}_L$ will indicate the $L \times L$ identity matrix and $\mathbf{1}_L$ the L-dimensional vector with all entries equal to 1. A generic entry of a matrix $\mathbf{M}$ will be indicated by $\mathbf{M}_{i,j}$ and $\mathbf{M}_{:,j}$ corresponds to the j-th column vector of the matrix. When not differently indicated, the norm $||\cdot||$ will refer to the 2-norm of vectors, while $||\cdot||_F$ and $||\cdot||_T$ will refer to the Frobenius and trace matrix norms, respectively. $\mathbf{B}(\mathbf{0},1)$ will denote the unitary ball centered in the origin. Finally, $\mathbb{R}_+$ will denote the set of non-negative reals.

2.1 EasyMKL

EasyMKL [1] is a recent MKL algorithm able to combine sets of base kernels by solving a simple quadratic problem. Besides its proved empirical effectiveness, a clear advantage of EasyMKL compared to other MKL methods is its high scalability with respect to the number of kernels to be combined. Specifically, its computational complexity is constant in memory and linear in time.

EasyMKL finds the coefficients $\boldsymbol{\eta}$ that maximize the margin on the training set, where the margin is computed as the distance between the convex hulls of positive and negative examples. In particular, the general problem EasyMKL tries to optimize is the following:

$$\max_{\boldsymbol{\eta}:||\boldsymbol{\eta}||=1} \min_{\boldsymbol{\gamma} \in \Gamma} \boldsymbol{\gamma}^\top \mathbf{Y} (\sum_{s=0}^{R} \eta_s \mathbf{K}_s) \mathbf{Y} \boldsymbol{\gamma} + \Lambda ||\boldsymbol{\gamma}||^2.$$

where $\mathbf{K}_s$ is the kernel matrix obtained applying the s-th kernel function k_s on training pairs, $\mathbf{Y}$ is a diagonal matrix with training labels on the diagonal, and Λ is a regularization hyper-parameter. The domain Γ represents two probability distributions over the set of positive and negative examples of the training set, that is $\Gamma = \{\boldsymbol{\gamma} \in \mathbb{R}_+^L \ | \sum_{y_i=+1} \boldsymbol{\gamma}_i = 1, \sum_{y_i=-1} \boldsymbol{\gamma}_i = 1\}$. At the solution, the first term of the objective function represents the obtained margin, that is the (squared) distance between a point in the convex hull of positive examples and a point in the convex hull of negative examples, in the compounded feature space.

The problem above is a min-max problem that can be reduced to a simple quadratic problem with a technical derivation described in [1]. Specifically, let γ^* be the unique solution of the following quadratic optimization problem:

$$\gamma^* = \arg\min_{\boldsymbol{\gamma} \in \Gamma} \boldsymbol{\gamma}^\top \mathbf{Y} \bar{\mathbf{K}} \mathbf{Y} \boldsymbol{\gamma} + \Lambda ||\boldsymbol{\gamma}||_2^2, \tag{1}$$

where $\bar{\mathbf{K}} = \sum_{s=0}^{R} \mathbf{K}_s$ is the simple sum of base kernels evaluated on training data, then the optimal vector of weights $\boldsymbol{\eta}^*$ has a simple analytic solution:

$$\boldsymbol{\eta}^* = \frac{\mathbf{d}(\boldsymbol{\gamma}^*)}{||\mathbf{d}(\boldsymbol{\gamma}^*)||_2}, \tag{2}$$

where the components of $\mathbf{d}(\boldsymbol{\gamma}^*)$ are $d_s(\boldsymbol{\gamma}^*) = \boldsymbol{\gamma}^{*\top}\mathbf{Y}\mathbf{K}_s\mathbf{Y}\boldsymbol{\gamma}^*$, $s \in \{0, \ldots, R\}$. Note that, the s-th entry of the vector $\mathbf{d}(\boldsymbol{\gamma}^*)$ represents the contribution given by the s-th kernel to the distance between the representative points in the convex hulls.

In the following, with no loss in generality, we consider coefficients of unitary 1-norm (i.e. re-scaled such that $||\boldsymbol{\eta}^*||_1 = 1$) and normalized base kernels. Finally, the output kernel will be computed by $k_{MKL}(\mathbf{x}, \mathbf{z}) = \sum_{s=0}^{R} \eta_s^* k_s(\mathbf{x}, \mathbf{z})$ which, in this case, it can be easily shown to be a normalized kernel as well.

2.2 Hierarchical Kernel Learning

Hierarchical Kernel Learning (HKL) [2] is a generalization of the MKL framework. The idea is to design a hierarchy over the given base kernels/features. In particular, base kernels are embedded in a DAG each one defined on a single feature. An ℓ_1/ℓ_2 block-norm regularization is then added in a way to induce a group sparsity pattern. This implies that the prediction function will involve very few kernels. Also, the condition of the kernels being strictly positive, makes this hierarchical penalization inducing a strong sparsity pattern [11], that is if a kernel/feature k_s is not selected, then none of the kernels associated with its descendants in the DAG are selected. Also, the weight η_s assigned to a kernel associated to a specific DAG node is always greater than the weight of the kernels associated with its descendants, basically giving a bias toward more general features. Interestingly, even if the DAG is exponentially large, the proposed HKL optimization algorithm is able to work with polynomial complexity.

As noted in [11], the sparsity pattern enforced by HKL can lead to the selection of many redundant features, namely the ones at the top of the DAG. For this, in the same work, a variant of the HKL, called generalized HKL (gHKL), is presented that partially overcomes this problem. The gHKL framework has a more flexible kernel selection pattern by using a ℓ_1/ℓ_p regularizer, with $p \in (1, 2]$, and maintains the polynomial complexity of the original method.

As stated in the original paper, this generic regularizer enables the gHKL formulation to be employed in the Rule Ensemble Learning (REL) where the goal is to construct an ensemble of conjunctive propositional rules. From this point of view, the task of gHKL is slightly different from ours (i.e. classification).

2.3 Dot-product Polynomial Kernels

A *generalized* polynomial kernel can be built on the top of any other valid base kernel as in $k(\mathbf{x}, \mathbf{z}) = p(k_0(\mathbf{x}, \mathbf{z}))$, where the base kernel k_0 is a valid kernel and $p : \mathbb{R} \longrightarrow \mathbb{R}$ is a polynomial function with non-negative coefficients, that is $p(x) = \sum_{s=0}^{d} a_s x^s, a_s \geq 0$. In this paper, we focus on *Dot-Product Polynomials* (using the acronym DPP) which is the class of generalized polynomial kernels where the simple dot product is used as base kernel, that is $k(\mathbf{x}, \mathbf{z}) = p(\mathbf{x} \cdot \mathbf{z})$.

A well known result from harmonic theory [18, 12] gives us an interesting connection between DPP and general non-linear dot-product kernels.

Theorem 1 *[12] A function $f : \mathbb{R} \to \mathbb{R}$ defines a positive definite kernel $k : \mathbf{B}(\mathbf{0}, 1) \times \mathbf{B}(\mathbf{0}, 1) \to \mathbb{R}$ as $k : (\mathbf{x}, \mathbf{z}) \to f(\mathbf{x} \cdot \mathbf{z})$ iff f is an analytic function admitting a Maclaurin expansion with non-negative coefficients, $f(x) = \sum_{s=0}^{\infty} a_s x^s, a_s \geq 0$.*

Kernel	Definition	DPP coefficients a_s
Polynomial ($K_{D,c}$)	$(\mathbf{x}\cdot\mathbf{z}+c)^D$	$\binom{D}{s}c^{D-s}, \forall s \in \{0,\dots,D\}$
RBF (K^{γ}_{RBF})	$e^{-\gamma\|\|\mathbf{x}-\mathbf{z}\|\|^2}$	$e^{-2\gamma}\frac{(2\gamma)^{2s}}{s!}, \forall s$
Rational Quadratic	$1-\frac{\|\mathbf{x}-\mathbf{z}\|_2^2}{\|\mathbf{x}-\mathbf{z}\|_2^2+c}$	$\left(-\frac{2\prod_{j=1}^{s}2+(j-1)}{(2+c)^{s+1}}+\frac{\prod_{j=1}^{s}2+(j-1)}{(2+c)^{s}}\right)\frac{1}{s!}, \forall s$
Cauchy	$(1+\frac{\|\mathbf{x}-\mathbf{z}\|_2^2}{\gamma})^{-1}$	$\frac{s!!}{3^{s+1}\gamma^s}\frac{1}{s!}, \forall s$

Table 1: Classical dot-product kernels formulated as DPPs with coefficients a_s (the symbol !! denotes the semifactorial of a number).

The theorem above guarantees that any dot product kernel of the form $k(\mathbf{x},\mathbf{z}) = f(\mathbf{x}\cdot\mathbf{z})$, $\mathbf{x}$ and $\mathbf{z}$ defined in the unitary ball, can be characterized by its Maclaurin expansion with non-negative coefficients, that is a DPP in the form $k(\mathbf{x},\mathbf{z}) = \sum_{s=0}^{\infty} a_s(\mathbf{x}\cdot\mathbf{z})^s$ (some examples of vector dot-product kernels and their DPP coefficients are presented in Table 1). On the other hand, any choice of non negative coefficients of a DPP induces a valid kernel. In this paper, we exploit this second implication proposing a method for the supervised learning of DPP coefficients.

3 Learning the kernel from a hierarchy of features

In this section, the principal idea of the paper is described. Similarly to the HKL approach of Section 2.2, we also consider a hierarchical set of features which can be mapped into a DAG. However, differently from HKL, we propose to group the features layer-wise, that is a different kernel k_s is built for each layer of the DAG. Kernels defined on bottom layers of the DAG will be more expressive leading to sparser kernel matrices, while kernels defined on top layers will be broader and their kernel matrices denser. In this way a hierarchy of representations of different levels of abstraction is created similarly to what happens in deep learning.

We will give a more formal definition of a measure of expressiveness for a kernel in Section 3.1. Generally speaking, we expect that the kernel expressiveness will increase going toward lower layers of the DAG. We also show the connection between our measure of expressiveness with the ERC and the rank of the kernel matrices induced by the kernel function.

Note that the procedure described above to construct base kernels is completely unsupervised and can be considered a sort of pre-training. The rationale of this construction is that too general or too specific features tend not to be useful in general. In particular, too general features are likely to be unable to discriminate since they tend to emphasize similarities between examples, while, using too specific features only, diversity is emphasized instead as examples are represented in such a way that distances are the same for every pair. It is important to note here that there is a difference between expressiveness of a kernel (which does not depend from the concept to learn) and informativeness of a kernel (which says how good the features of the kernel are on discriminating a given concept). Our intuition here is that different tasks defined on a same set of examples may need of feature spaces of different expressiveness. Given a binary task, these different

representations are aggregated using a maximum margin based MKL algorithm, for example EasyMKL, as presented in Section 3.2.

3.1 Complexity and expressiveness of kernel functions

Now, we propose a methodology to compare different representations on the basis of the complexity of the hypotheses space induced by the associated kernel function. Representations inducing lower complexity hypotheses will correspond to more abstract or general representations.

Kernel learning methods typically impose some regularization over the combined kernels to limit their expressiveness with the hope to limit over-fitting of the hypotheses constructed using that kernels. In the simplest case, the trace of the produced kernel can be used. However, the trace might not be the best choice of a measure of expressiveness for a kernel. For example, the identity matrix $\mathbf{I}_L \in \mathbb{R}^{L\times L}$ and the constant matrix $\mathbf{1}_L\mathbf{1}_L^\top \in \mathbb{R}^{L\times L}$ have the same trace but it is clear that the associated kernel functions have different expressiveness. In the first case, the examples are orthogonal in feature space and the expressiveness is maximal while, in the second case, they overlap and the expressiveness is minimal.

The expressiveness of a kernel function, that is the number of dichotomies that can be realized by a linear separator in that feature space, is more captured by the rank of the kernel matrices it produces. This can be motivated in several ways. A quite intuitive one can be given using the following theorem.

Theorem 2 *Let $\mathbf{K} \in \mathbb{R}^{L\times L}$ be a kernel matrix over a set of L examples. Let rank($\mathbf{K}$) be the rank of $\mathbf{K}$. Then, there exists at least one subset of examples of size rank($\mathbf{K}$) that can be shattered by a linear function.*

Proof Let be given a diagonal matrix $\mathbf{Y} \in \{-1,+1\}^{L\times L}$ of binary labels for the examples (i.e. a diagonal matrix with labels on the diagonal), then we can see that the squared distance between the convex hulls of positive and the convex hull of negative examples can be written as $\rho^2 = \min_{\boldsymbol{\gamma}\in\Gamma} \boldsymbol{\gamma}^\top\mathbf{YKY}\boldsymbol{\gamma}$ where $\Gamma = \{\boldsymbol{\gamma} \in \mathbb{R}^L_+ \mid \sum_{y_i=+1}\gamma_i = 1, \sum_{y_i=-1}\gamma_i = 1\}$. If the kernel matrix has maximal rank L, then using the Courant-Fisher theorem (see [19]) we have that $\frac{\boldsymbol{\gamma}^\top\mathbf{YKY}\boldsymbol{\gamma}}{||\mathbf{Y}\boldsymbol{\gamma}||^2} \geq \lambda_L > 0$ where λ_L is the minimum eigenvalue, for any $\boldsymbol{\gamma} \in \Gamma$. Let L_+ and L_- be the number of positive and negative examples, then we have $\boldsymbol{\gamma}^\top\mathbf{YKY}\boldsymbol{\gamma} \geq \lambda_L||\mathbf{Y}\boldsymbol{\gamma}||^2 \geq (L_+^{-1} + L_-^{-1})\lambda_L > 0$ for any $\boldsymbol{\gamma} \in \Gamma$. This implies $\rho^2 > 0$ (the set can be linearly separated using that feature space) for any possible labeling of the examples, that is any choice of the matrix $\mathbf{Y}$. Now, suppose rank($\mathbf{K}$) $< L$, then it will exist a minor of $\mathbf{K}$ of order rank($\mathbf{K}$) with maximal rank and this corresponds to select a subset of k examples which can be linearly shattered. □

In this section we propose a new, simple to compute, expressiveness measure for kernel matrices, namely the spectral ratio. Next, we will show that this measure is strongly related to the rank of the matrix, to the radius of the MEB, and to the ERC of the hypotheses space associated with that representation.

The *spectral ratio* (SR) for a positive semi-definite matrix $\mathbf{K}$ is defined as the ratio between the 1-norm and the 2-norm of its eigenvalues, or equivalently, as the

ratio between its trace norm $||\mathbf{K}||_T$ and its Frobenius norm $||\mathbf{K}||_F$:

$$\mathcal{C}(\mathbf{K}) = \frac{\sum_{i=1}^{L} \lambda_i}{\sqrt{\sum_{i=1}^{L} \lambda_i^2}} = \frac{||\mathbf{K}||_T}{||\mathbf{K}||_F} = \frac{\sum_i \mathbf{K}_{ii}}{\sqrt{\sum_{ij} \mathbf{K}_{ij}^2}}. \tag{3}$$

Note that, compared to the rank of a matrix, the above measure has the advantage that it does not require the decomposition of the matrix.

An equivalent standardized version of the spectral ratio with values in $[0, 1]$ can also be defined as follows:

$$\bar{\mathcal{C}}(\mathbf{K}) = \frac{\mathcal{C}(\mathbf{K}) - 1}{\sqrt{L} - 1} \in [0, 1]. \tag{4}$$

A plethora of different measures from other fields also exploit the trace and the rank of a matrix in its formulation. For example, in the multilinear algebra regularizers [17], the rank is used to control the *degree of freedom* of the final model, or in quantum information theory [21] where the, so called, *trace-distance* is fundamental to discriminate between two different states of a system.

Now, we are ready to give a qualitative measure of expressiveness of kernel functions, in terms of specificity and generality as it follows:

Definition 1 Let be given k_i, k_j, two kernel functions. We say that k_i *is more general (or less expressive) than* k_j ($k_i \geq_G k_j$) or equivalently that k_j *is more specific (or more expressive) than* k_i ($k_j \leq_G k_i$) whenever for any possible dataset $\mathbf{X}$, we have $\mathcal{C}(\mathbf{K}_{\mathbf{X}}^{(i)}) \leq \mathcal{C}(\mathbf{K}_{\mathbf{X}}^{(j)})$ with $\mathbf{K}_{\mathbf{X}}^{(i)}$ the kernel matrix evaluated on data $\mathbf{X}$ using the kernel function k_i.

3.1.1 Connection between SR and the rank of a kernel matrix

The (squared) spectral ratio can be seen as an (efficient) strict approximation of the rank of a matrix. In fact, using a result of [20], namely:

$$||\mathbf{K}||_F \leq ||\mathbf{K}||_T \leq \sqrt{\text{rank}(\mathbf{K})}||\mathbf{K}||_F,$$

we can easily derive the following strict bounds:

$$1 \leq \mathcal{C}(\mathbf{K}) \leq \sqrt{\text{rank}(\mathbf{K})}.$$

The spectral ratio $\mathcal{C}(\mathbf{K})$ has the following additional nice properties:

- the identity matrix $\mathbf{I}_L$ having rank equal to L has the maximal spectral ratio with $\mathcal{C}(\mathbf{I}_L) = \sqrt{L}$ and $\bar{\mathcal{C}}(\mathbf{I}_L) = 1$;
- the kernel $\mathbf{K} = \mathbf{1}_L \mathbf{1}_L^\top$ having rank equal to 1 has the minimal spectral ratio with $\mathcal{C}(\mathbf{1}_L \mathbf{1}_L^\top) = 1$ and $\bar{\mathcal{C}}(\mathbf{1}_L \mathbf{1}_L^\top) = 0$;
- it is invariant to multiplication with a positive scalar as $\mathcal{C}(\alpha\mathbf{K}) = \mathcal{C}(\mathbf{K}), \forall \alpha > 0$.

3.1.2 Connection between SR and ERC

The spectral ratio can also be related to the ERC. In order to show this, we consider the set of vectors with fixed 1-norm

$$\Psi = \{\boldsymbol{\alpha} \in \mathbb{R}^L \text{ s.t. } \|\boldsymbol{\alpha}\|_1 = 1\}. \tag{5}$$

Then, we focus on the class of linear functions with bounded norm $\mathcal{F}_B = \{\mathbf{x}_j \rightarrow \sum_{i=1}^{L} \alpha_i \mathbf{K}_{i,j} : \boldsymbol{\alpha}^T \mathbf{K} \boldsymbol{\alpha} \leq B^2, \boldsymbol{\alpha} \in \Psi\} \subseteq \{\mathbf{x}_j \rightarrow \mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_j) : \|\mathbf{w}\|_2 \leq B\}$ where $\boldsymbol{\phi}$ is a feature mapping associated to the kernel $\mathbf{K}$. It is well known that the following result bounding the complexity of $\mathcal{F}_B$ holds:

Theorem 3 *([19], Theorem 4.12). Given a kernel matrix* $\mathbf{K}$*, evaluated over a set of points* $\mathcal{X}$*, the ERC of the class* $\mathcal{F}_B$ *satisfies*

$$\hat{\mathcal{R}}(\mathcal{F}_B) \leq \frac{2B}{L}\sqrt{||\mathbf{K}||_T}. \tag{6}$$

Equation 6 gives a bound on the ERC dependent on the trace of the kernel.

Now, we can observe that, for a general kernel $\mathbf{K}$, the value of $\boldsymbol{\alpha}^T \mathbf{K} \boldsymbol{\alpha}$ can be bounded by the Frobenius norm of $\mathbf{K}$, that is:

Proposition 1 *Let* $\mathbf{K}$ *be a kernel matrix in* $\mathbb{R}^{L \times L}$ *with eigenvalues* $\lambda_1 \geq \cdots \geq \lambda_L \geq 0$*, then*

$$\forall \boldsymbol{\alpha} \in \Psi, \quad \boldsymbol{\alpha}^T \mathbf{K} \boldsymbol{\alpha} \leq \lambda_1 \leq \|\mathbf{K}\|_F. \tag{7}$$

Proof We can exploit the spectral decomposition of the matrix and rewrite $\boldsymbol{\alpha}^T \mathbf{K} \boldsymbol{\alpha} = \sum_{i=1}^{L} \lambda_i (\boldsymbol{\alpha}^T \mathbf{u}_i)^2$, where $\mathbf{u}_i$ is the eigenvector with eigenvalue λ_i. Then, it is easy to see that: $(\boldsymbol{\alpha}^T \mathbf{u}_i)^2 = \cos(\theta_{\boldsymbol{\alpha},\mathbf{u}_i})^2 \|\boldsymbol{\alpha}\|_2^2$, where $\theta_{\boldsymbol{\alpha},\mathbf{u}_i}$ is the angle between the vector $\boldsymbol{\alpha}$ and the eigenvector $\mathbf{u}_i$. Using the properties of the norms ($\|\boldsymbol{\alpha}\|_2^2 \leq \|\boldsymbol{\alpha}\|_1^2 = 1$) and the fact that the eigenvectors $\mathbf{u}_i$ are an orthonormal base, we can obtain the final result:

$$\boldsymbol{\alpha}^T \mathbf{K} \boldsymbol{\alpha} \leq \sum_{i=i}^{L} \lambda_i \cos(\theta_{\boldsymbol{\alpha},\mathbf{u}_i})^2 \|\boldsymbol{\alpha}\|_1^2 \leq \lambda_1 \leq \sqrt{\sum_{i=1}^{L} \lambda_i^2} = \|\mathbf{K}\|_F.$$

□

Finally, we are ready to prove the following theorem that gives us a connection between the spectral ratio of a kernel matrix and the complexity of the induced hypotheses space:

Theorem 4 *Given a kernel* $\mathbf{K}$ *evaluated over a set of points* $\mathcal{X}$*, the ERC* $\hat{\mathcal{R}}$ *of the class of functions* $\mathcal{F} = \{\mathbf{x}_j \rightarrow \sum_{i=1}^{L} \alpha_i \frac{\mathbf{K}_{i,j}}{\|\mathbf{K}\|_F}, \boldsymbol{\alpha} \in \Psi\}$ *satisfies*

$$\hat{\mathcal{R}}(\mathcal{F}) \leq \frac{2}{L}\sqrt{\mathcal{C}(\mathbf{K})}. \tag{8}$$

Proof Applying Theorem 3 to the matrix $\frac{\mathbf{K}}{||\mathbf{K}||_F}$ we can derive

$$\hat{\mathcal{R}}(\mathcal{F}_{\mathcal{B}}) \leq \frac{2B}{L}\sqrt{\left\|\frac{\mathbf{K}}{\|\mathbf{K}\|_F}\right\|_T} = \frac{2B}{L}\sqrt{\mathcal{C}(\mathbf{K})}.$$

The result is then obtained by using Proposition 1 and setting $B = 1$. □

3.1.3 Connection between SR and the radius of the MEB

The subject of this section is to show that the SR, and hence the degree of sparsity of the kernel matrices, are related to the radius of the Minimum Enclosing Ball (MEB). Given a dataset embedded in a feature space, the MEB is a the smallest hypersphere containing all the data. We can show that the radius increases with the SR of a kernel. In fact, see for example [19], when considering a normalized kernel $\mathbf{K} \in \mathbb{R}^{L \times L}$, the radius of the MEB of the examples in feature space can be computed by $r^*(\mathbf{K}) = 1 - \min_{\boldsymbol{\alpha} \in A} \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha}$, where $A = \{\boldsymbol{\alpha} \in \mathbb{R}^L_+, \sum_i \alpha_i = 1\}$. A nice approximation of the radius can be computed as $\tilde{r}(\mathbf{K}) = 1 - \bar{k}$ where $\bar{k}$ is the average of the entries in the matrix $\mathbf{K}$. This much simpler formula is exact in the two extreme cases, as $\tilde{r}(\mathbf{1}_L \mathbf{1}_L^\top) = r^*(\mathbf{1}_L \mathbf{1}_L^\top) = 0$ and $\tilde{r}(\mathbf{I}_L) = r^*(\mathbf{I}_L) = 1 - 1/L$. In general, the approximation is a lower bound of the radius, that is $\tilde{r}(\mathbf{K}) \leq r^*(\mathbf{K})$, since $\tilde{r}(\mathbf{K})$ can be obtained using a sub-optimal $\boldsymbol{\alpha} = \frac{1}{L}\mathbf{1}_L$. In Figure 1, the value of the MEB radius and the average approximation has been plotted for kernels of increasing expressiveness over two different datasets of example.

This result confirms that the SR can be used as a measure of the (intrinsic) complexity of the feature space. At the end of Section 3.2, a result linking the radius of the MEB with the leave-one-out error of the proposed algorithm will be given. In other words, limiting the SR of the combined kernels, and hence the radius of the MEB, while maximizing the margin on labeled data, gives a principled strategy to pursue to learn effectively.

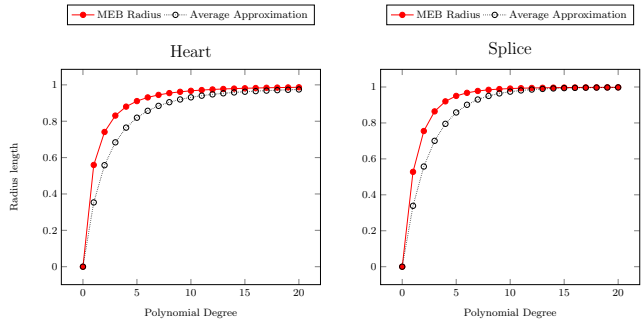

Fig. 1: The values of the radius of the minimum enclosing ball of patterns in feature space and its *average kernel value* approximation are reported for kernels of increasing expressiveness over two different datasets (Heart and Splice). It is interesting to note how the radius nicely increases with the SR of the kernel matrix.

3.2 EasyMKL for learning over a hierarchy of feature spaces

In this section, the general approach proposed in this paper is summarized and its generalization ability is briefly discussed.

Given a hierarchical set of features F (see Figure 2 for an example of polynomial features)[1], the approach proposed in this paper consists of the following steps:

Learning over a hierarchy of feature spaces: the algorithm

1. Consider a partition of the features
$$P = \{F_0, \dots, F_R\}, F_i \cap F_j = \emptyset, \bigcup_{s=0}^{R} F_s = F$$
and construct normalized kernels $k_0, .., k_R$ associated to those sets of features in such a way to obtain a set of kernels of increasing expressiveness, that is $k_0 \geq_G k_1 \geq_G \cdots \geq_G k_R$;
2. Apply EasyMKL on kernels $\{k_0, \cdots, k_R\}$ to learn the coefficients $\boldsymbol{\eta} \in \mathbb{R}_+^{R+1}$ such that $\sum_{s=0}^{R} \eta_s = 1$ and define $k_{MKL}(\mathbf{x}, \mathbf{z}) = \sum_{s=0}^{R} \eta_s k_s(\mathbf{x}, \mathbf{z})$.

Note that, the rationale of our method is quite different from the one of HKL. In that case, the combination weights are determined in such a way that kernels higher in the hierarchy get higher weights. As we already discussed, this may not be always the best choice. As we will see in the experimental part (see Section 5.4), higher margin kernels are often obtained combining base kernels with intermediate expressiveness. In particular, when the supervised task is simple and labeled data can be more easily separated, then it is expected that more general base kernels will get large weights from the MKL algorithm and the converse should happen when the task turns out to be particularly difficult. In Section 5.5 we dedicated a set of experiments to a detailed analysis of this particular situation.

Finally, we briefly discuss about the generalization ability of the method. In particular, we can use the result presented in Section 3.1.3 to give a radius-margin bound on the generalization error of the general method described above. Specifically, since the base kernels have increasing sparsity and expressiveness, then the radius of the enclosing ball is proved to increase. Then, let $r_{\boldsymbol{\eta}}$ be the MEB radius of the produced kernel k_{MKL}, then it can be proved that $r_{\boldsymbol{\eta}} \leq \max_s(r_s) = r_R$ where r_s is the radius of the MEB of the examples when the s-th feature space (k_s) is used [9]. Hence, looking for the EasyMKL solution $\boldsymbol{\eta}$ maximizing the margin $\rho_{\boldsymbol{\eta}}$ can be understood as trying to minimize the radius-margin bound of the expected leave-one-out error, namely $\frac{1}{L} r_R^2 / \rho_{\boldsymbol{\eta}}^2$.

4 Learning the kernel in the space of DPPs

As we have seen in Section 2.3, any choice of non negative coefficients of a DPP gives a valid kernel. In particular, under mild condition, any dot-product kernel $k(\mathbf{x}, \mathbf{z}) = f(\mathbf{x} \cdot \mathbf{z})$ can be decomposed as a (possibly infinite) non negative combi-

[1] Note that, besides the running example of monomials used in this paper, other possibilities are available, including ANOVA features, subtrees of different length for trees, substrings of different length for strings, etc.

nation of homogeneous polynomial kernels (HPK), that is:

$$k(\mathbf{x},\mathbf{z}) = f(\mathbf{x}\cdot\mathbf{z}) = \sum_{s=0}^{\infty} a_s k_s(\mathbf{x},\mathbf{z})$$

where $k_s(\mathbf{x}\cdot\mathbf{z}) = (\mathbf{x}\cdot\mathbf{z})^s$. Here we propose to learn the DPP weights using the EasyMKL algorithm. In this section, we will show that the base HPKs of the combination have increasing expressiveness and hence the proposed solution is an instance of the general methodology proposed in Section 3.

4.1 Structure of Homogeneous Polynomial Kernels

It is well known that the feature space of a d-degree HPK corresponds to all possible monomials of degree d, that is $\phi_j(\mathbf{x}) = \prod_{i=1}^{d} x_{j_i}$ where $j \in \{1,\dots,m\}^d$ enumerates all possible d-combinations with repetitions from m variables, that is $j_i \in \{1,\dots,m\}$. Note that, there is a clear dependence between features of higher order HPKs and features of lower order HPKs.

For example, the value of the feature $x_1x_4x_5x_9$ in the 4-degree HPK gives us some information about the values of the features x_1, x_4, x_5 and x_9 in the 1-degree HPK and viceversa. An illustration of this kind of dependencies is depicted in Figure 2. In general, we expect that the higher the order of the HPK, the sparser the kernel matrix produced. We will prove this is true at least when the HPKs are normalized. Specifically, the following theorem shows that the exponent d of a HPK of the form $K(\mathbf{x},\mathbf{z}) = (\mathbf{x}\cdot\mathbf{z})^d$ induces an order of expressiveness in the kernel functions.

Proposition 2 *For any choice $D \in \mathbb{N}$, the family of kernels $\mathcal{K}_D = \{k_0,\dots,k_D\}$, with $k_d(\mathbf{x},\mathbf{z}) = (\frac{\mathbf{x}\cdot\mathbf{z}}{||\mathbf{x}||||\mathbf{z}||})^d$ the d-degree normalized homogeneous polynomial kernel, has monotonically increasing expressiveness, that is $k_i \geq_G k_j$ when $i \leq j$.*

Proof Let i, j be two indexes such that $0 \leq i < j \leq D$, we need to prove that $\mathcal{C}(\mathbf{K}_{\mathbf{X}}^{(i)}) \leq \mathcal{C}(\mathbf{K}_{\mathbf{X}}^{(j)})$ for any dataset $\mathbf{X}$ of any size L. Since the kernels are normalized, then $||\mathbf{K}_{\mathbf{X}}^{(i)}||_T = L$, and all we need to prove is that $||\mathbf{K}_{\mathbf{X}}^{(i)}||_F \geq ||\mathbf{K}_{\mathbf{X}}^{(j)}||_F$ or equivalently $||\mathbf{K}_{\mathbf{X}}^{(i)}||_F^2 \geq ||\mathbf{K}_{\mathbf{X}}^{(j)}||_F^2$ which is easy to show, as:

$$||\mathbf{K}_{\mathbf{X}}^{(i)}||_F^2 = \sum_{i,j}^{L,L} \left(\frac{\mathbf{x}_i\cdot\mathbf{x}_j}{||\mathbf{x}_i||||\mathbf{x}_j||}\right)^{2i} \geq \sum_{i,j}^{L,L} \left(\frac{\mathbf{x}_i\cdot\mathbf{x}_j}{||\mathbf{x}_i||||\mathbf{x}_j||}\right)^{2j} = ||\mathbf{K}_{\mathbf{X}}^{(j)}||_F^2$$

where we used the fact that $\left|\frac{\mathbf{x}_i\cdot\mathbf{x}_j}{||\mathbf{x}_i||||\mathbf{x}_j||}\right| \leq 1$ and $z^i \geq z^j$ when $i < j$ and $|z| \leq 1$. □

Note that, as far as we are concerned with normalized HPK, the Frobenius norm of the combination kernel $\mathbf{K}_{MKL} = \sum_{s=0}^{D} \eta_s \mathbf{K}_s$ is

$$||\sum_{s=0}^{D} \eta_s \mathbf{K}_s||_F^2 = \sum_{s=0,t=0}^{D,D} \eta_s\eta_t \mathbf{C}_{s,t}, \text{ where } \mathbf{C}_{s,t} = \sum_{i=0,j=0}^{L,L} k_s(\mathbf{x}_i,\mathbf{x}_j)k_t(\mathbf{x}_i,\mathbf{x}_j).$$

It means that the Frobenius norm of the computed kernel is a convex combination of values $\mathbf{C}_{s,t}$ with weights $\eta_s\eta_t$. Interestingly, the matrix $\mathbf{C}$ is a sort of correlation

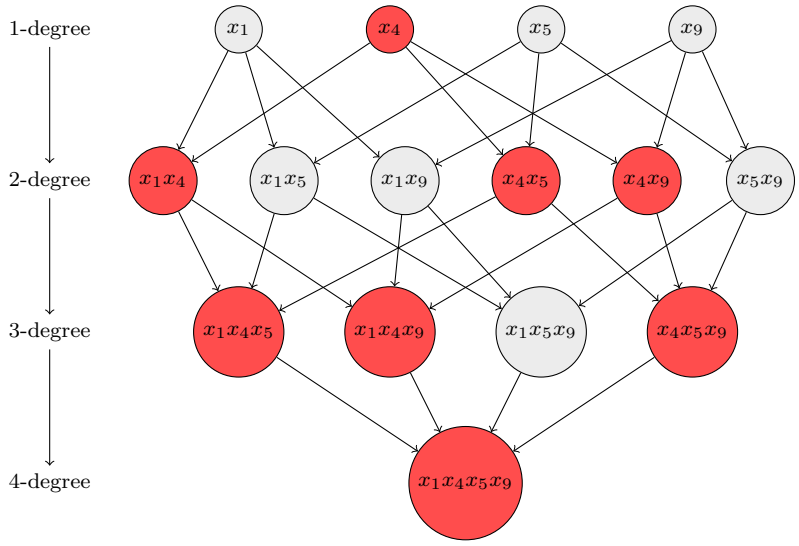

Fig. 2: Example of dependencies between features. The arrows represent the dependencies between features of different degrees. The nodes in red, starting from the top, represent the diffusion of the zeros (i.e. the sparsity): if the value of x_4 is zero then the value of all the dependent features is also zero. Conversely, if the value of the feature $x_1x_4x_5x_9$ is different from zero then all the features in the given graph must have values different from zero.

matrix between base kernels, containing the squared Frobenius norms of individual kernels in the diagonal. It is easy to see that $||\mathbf{K}_D||_F \leq ||\mathbf{K}_{MKL}||_F \leq ||\mathbf{K}_0||_F$ holds for any setting of the parameters $\boldsymbol{\eta}$.

5 Experimental Work

In this section we present the extensive experimental work we have done. First of all, we demonstrate empirically that $\mathcal{K}_D$ as defined in Section 4.1 is effectively a good choice in practice as family of base kernels in MKL showing state-of-the-art classification performances on several UCI datasets (Section 5.1) and the large MNIST handwritten multi-classification task (Section 5.8) compared to common baselines. Second, we show the importance of the structure of the feature partition $\mathcal{K}_D$ comparing it with possible alternatives, namely the random partition (Section 5.2) and the partition used by the HKL method (Section 5.7). Third, we offer a deeper analysis reporting the spectral ratio (Section 5.3) and a study of the weights returned by EasyMKL when treating increasingly noisy tasks (Section 5.4 and Section 5.5). Finally, an analysis of the computational complexity of our method compared to the traditional SVM with the RBF kernel is presented (Section 5.6). Our implementation of EasyMKL is available at `https://github.com/jmikko/EasyMKL`. Further details about the datasets are summarized in Table 2.

Data set	Source	Features	Examples
Haberman	*UCI* [3]	3	306
Liver	*UCI*	6	345
Diabetes	*UCI*	8	768
Abalone	*UCI*	8	4177
Australian	*UCI*	14	690
Pendigits	*UCI*	16	4000
Heart	*UCI*	22	267
German	*Statlog*	24	1000
Ionosphere	*UCI*	34	351
Splice	*UCI*	60	1000
Sonar	*UCI*	60	208
MNIST	[14]	784	70000
Colon	*UCI*	2000	62
Gisette	*NIPS*	5000	4000

Table 2: Datasets information: name, source, number of features and number of examples.

5.1 MKL for learning DPP on UCI datasets

In this section we describe the experiments we performed to test the accuracy in terms of AUC of the kernel generated by learning the coefficients of a dot-product polynomial using $\mathcal{K}_D = \{k_0, \ldots, k_D\}$ as base HPKs as defined in Section 4.1, and varying the value of D. This method is indicated with $\mathbf{K}_{MKL}$ in the following.

The AUC results are obtained using a stratified nested 10-fold cross validation. Specifically we used the following procedure:

- Each dataset is divided in 10 folds $\mathbf{f}_1, \ldots, \mathbf{f}_{10}$ respecting the distribution of the labels, where $\mathbf{f}_i$ contains the list of indexes of the examples in the i-th fold;
- One fold $\mathbf{f}_j$ is selected as test set;
- The remaining nine out of ten folds $\mathbf{v}_j = \bigcup_{i=1, i\neq j}^{10} \mathbf{f}_i$ are then used as validation set for the choice of the hyper-parameters. In particular, another 10-fold cross validation over $\mathbf{v}_j$ is performed;
- The set $\mathbf{v}_j$ is selected as training set to generate a model (using the validated hyper-parameters);
- The test fold $\mathbf{f}_j$ is used as test set to evaluate the performance of the model;
- The reported results are the averages (with standard deviations) obtained repeating the steps above over all the 10 possible test sets $\mathbf{f}_j$ (i.e. for each j in $\{1, \ldots, 10\}$).

For each D, we compared our algorithm against other DPP fixed weighting rules:

- $\mathbf{K}_D$: the weight η_D is set to 1 (and all the other weights are set to 0);
- $\mathbf{K}_{sum}$: the weight is set uniformly over the base kernels, that is $\eta_s = \frac{1}{D+1}$ for $s \in \{0, 1, \ldots, D\}$ (as pointed before, this is generally a strong baseline);
- $\mathbf{K}_{D,c}$: the weights are assigned using the polynomial kernel rule (see Table 1):

$$\eta_s \propto \binom{D}{s} c^{D-s}, \quad s \in \{0, \ldots, D\}. \tag{9}$$

In this case, the value c is selected *optimistically* as the one from the set $\{0.5, 1, 2, 3\}$ which obtained the best AUC on the test set.

- $\mathbf{K}^{\gamma}_{RBF}$: the weights are assigned according to the *truncated* RBF rule (see Table 1):

$$\eta_s \propto \frac{(2\gamma)^{2s}}{s!}, \quad s \in \{0, \ldots, D\}. \tag{10}$$

Again, the value for γ is selected *optimistically* as the one from the set $\{2^i : i \in \{-5, -4, \ldots, 0, 1\}\}$ which obtained the best AUC on the test set.

Note that, the results depicted in the following for $\mathbf{K}_{D,c}$ and $\mathbf{K}^{\gamma}_{RBF}$ are optimistic estimates of the real performance because of the selection *a posteriori* of the best parameters c and γ, respectively.

In all the cases above, we performed a stratified nested 10-fold cross validation to select the optimal EasyMKL parameter Λ from the set of values $\{\frac{v}{1-v} : v \in \{0.0, 0.1, \ldots, 0.9, 1.0\}\}$.

The AUC results of these experiments are reported in Figures 3, 4 and 5 for all datasets. As the reader can see from the figure, our method consistently and significantly outperforms both the single base kernel solution $\mathbf{K}_D$ and the average solution $\mathbf{K}_{sum}$, especially for high polynomial degrees where $\mathbf{K}_D$ and $\mathbf{K}_{sum}$ tend to overfit. Moreover, our solution is at least comparable and often better than the optimistic AUC performances of $\mathbf{K}_{D,c}$ and $\mathbf{K}^{\gamma}_{RBF}$ weighting rules.

5.2 Is the deep structure important?

In this section, we show empirically that the structure present in HPKs is indeed useful in order to obtain good results using our MKL approach. With this aim, we built two alternative sets of base kernel matrices $\mathcal{Q}_D = \{\mathbf{K}^{(Q)}_0, \ldots, \mathbf{K}^{(Q)}_D\}$ and $\mathcal{R}_D = \{\mathbf{K}^{(R)}_0, \ldots, \mathbf{K}^{(R)}_D\}$. Specifically, we considered the same set of features (monomials with degrees less or equal to D) for both families of base kernels, but the features are assigned to base kernels in a different way. When generating $\mathcal{Q}_D$, the features are assigned to the kernels according to the degree rule, that is features of degree d are assigned to the kernel $\mathbf{K}^{(Q)}_d$. On the other hand, when generating $\mathcal{R}_D$, the features are assigned randomly to one of its base kernels.

It is well known that the number of possible combinations with repetition of $d \in \{0, \ldots, D\}$ features, picked from a set of m variables, is equal to $N_d = m^d$. This fact can be exploited to define a distribution over the $D + 1$ degrees, that is:

$$\pi(d) = \frac{N_d}{\sum_{j=0}^{D} N_j}, \quad d \in \{0, \ldots, D\}. \tag{11}$$

Algorithm 1 gives an effective procedure to generate families of $D + 1$ base kernels using monomials of degrees less or equal to a given D. Basically, the algorithm draws a random feature from the feature space, using Eq. 11 to draw its degree and then draws uniformly at random among all monomials of that degree. After that, the feature is assigned to a base kernel selected by the function $\mathcal{S}$ which is a parameter of the algorithm. Specifically, the algorithm will be invoked with $\mathcal{S}(d) = d$ for the family $\mathcal{Q}_D$ and with $\mathcal{S}(d) = \text{random}(0, \ldots, D)$ for the family $\mathcal{R}_D$. Finally, all the base kernels generated will be normalized.

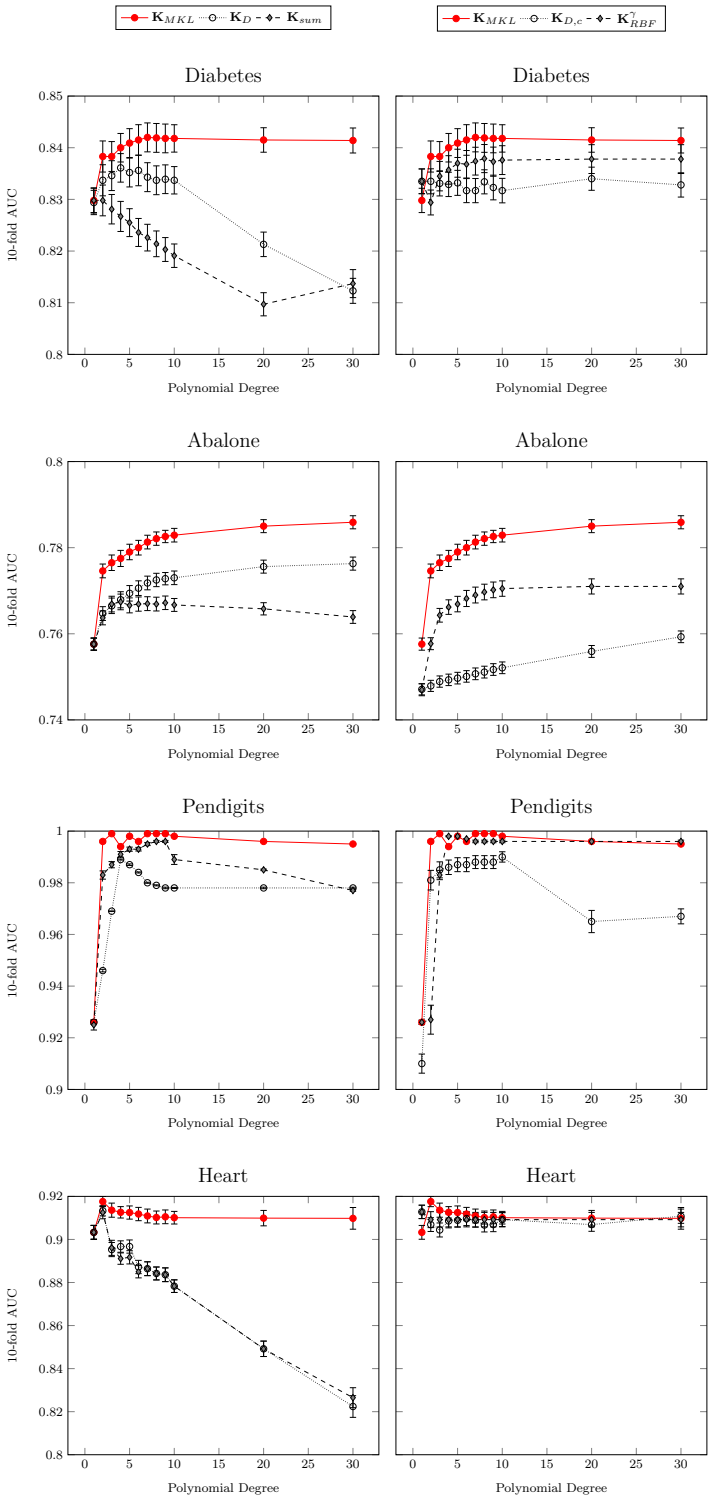

Fig. 3: AUC with standard deviation for different values of D. Our proposed solution $\mathbf{K}_{MKL}$ (solid red line) has been compared against the baselines $\mathbf{K}_D$ and $\mathbf{K}_{sum}$ and the optimistic AUC results of $\mathbf{K}_{D,c}$ and $\mathbf{K}^{\gamma}_{RBF}$ where *a posteriori* optimal $c \in \{0.5, 1, 2, 3\}$ and optimal $\gamma \in \{2^i : i = -5, -4, \ldots, 1\}$ have been selected, respectively.

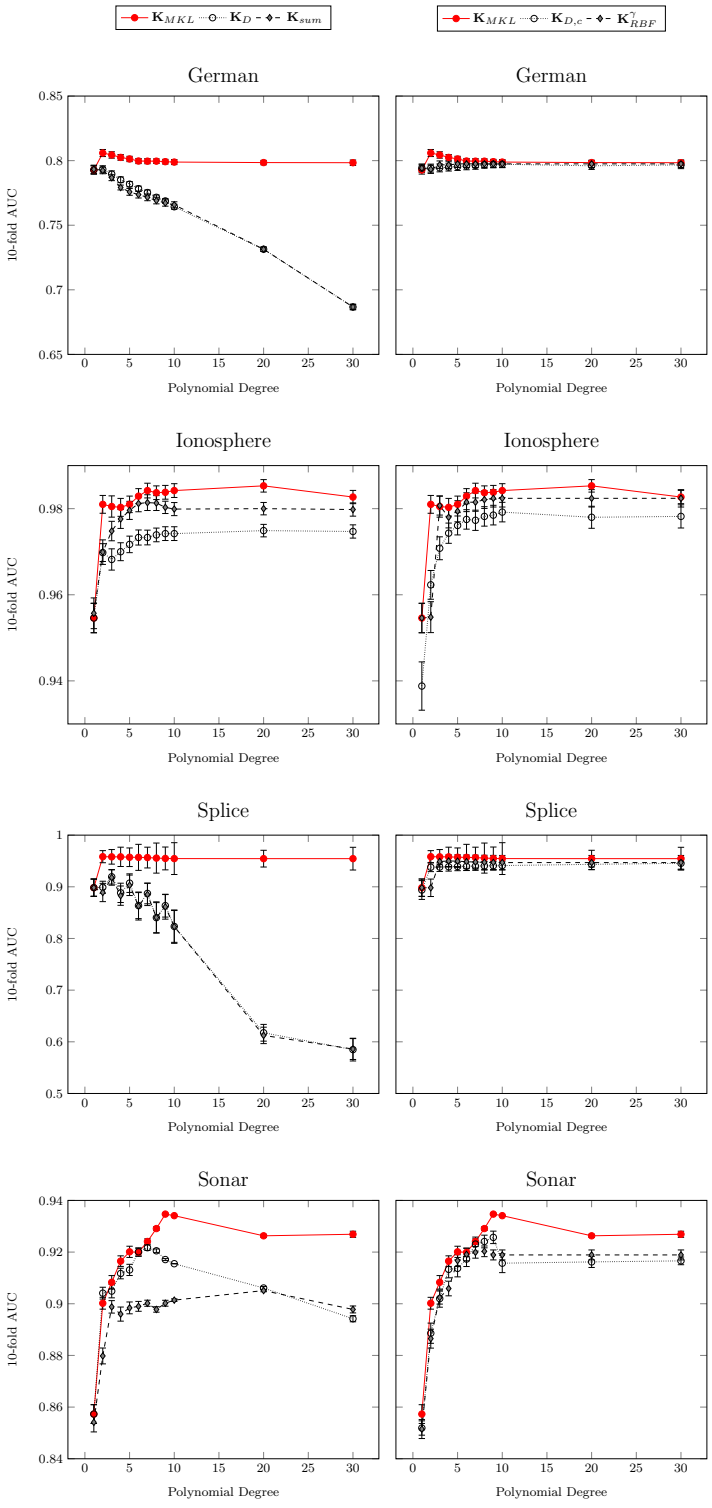

Fig. 4: AUC with standard deviation for different values of D. Our proposed solution $\mathbf{K}_{MKL}$ (solid red line) has been compared against the baselines $\mathbf{K}_D$ and $\mathbf{K}_{sum}$ and the optimistic AUC results of $\mathbf{K}_{D,c}$ and $\mathbf{K}^{\gamma}_{RBF}$ where *a posteriori* optimal $c \in \{0.5, 1, 2, 3\}$ and optimal $\gamma \in \{2^i : i = -5, -4, \ldots, 1\}$ have been selected, respectively.

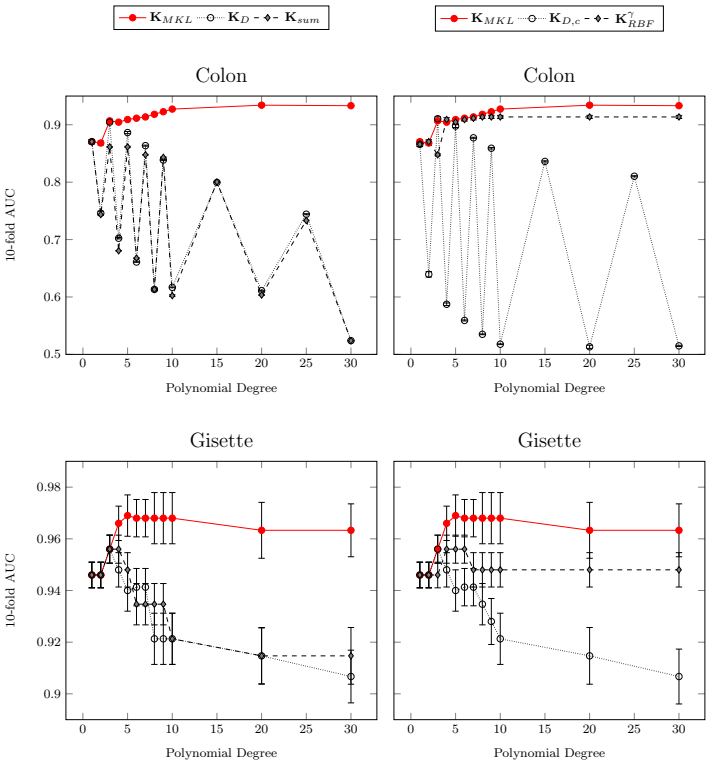

Fig. 5: AUC with standard deviation for different values of D. Our proposed solution $\mathbf{K}_{MKL}$ (solid red line) has been compared against the baselines $\mathbf{K}_D$ and $\mathbf{K}_{sum}$ and the optimistic AUC results of $\mathbf{K}_{D,c}$ and $\mathbf{K}^{\gamma}_{RBF}$ where *a posteriori* optimal $c \in \{0.5, 1, 2, 3\}$ and optimal $\gamma \in \{2^i : i = -5, -4, \ldots, 1\}$ have been selected, respectively.

We generated families $\mathcal{Q}_D$ and $\mathcal{R}_D$ for different values of D, with number of steps fixed to $50,000$ for two different datasets. The stratified nested 10-fold cross validation results for Abalone and Ionosphere datasets are reported in Figure 6.

These results seem to confirm the importance of the deep structure imposed in $\mathcal{Q}_D$ to obtain good results with EasyMKL. Interestingly, we noticed that the weights assigned by EasyMKL when the family $\mathcal{R}_D$ was used were almost uniform thus generating a solution near to the one of the average kernel $\mathbf{K}_{sum}$.

5.3 Analysis of the spectral ratio

Here, the study we have performed about the spectral ratio of $\mathbf{K}_{MKL}$, $\mathbf{K}_D$ and $\mathbf{K}_{sum}$ on the benchmark datasets is presented. Figure 7 summarizes these results for six benchmark datasets. Interestingly, for all the values of D, $\mathbf{K}_{MKL}$ have shown a spectral ratio trapped between the spectral ratio of $\mathbf{K}_D$ and the one of $\mathbf{K}_{sum}$. While the first fact was theoretically expected, the second can be surprising. Considering the discussion we made in Section 4.1, this can be due to the very

Algorithm 1 Random generation of a family of *base* kernels.
The symbol $\bigodot$ stands for the entry-wise multiplication among vectors of the same dimension.

Require: $\mathbf{X}, D, steps, \mathcal{S} : \{0, \dots, D\} \to \{0, \dots, D\}$
Ensure: A kernel family $\mathcal{K}_D = \{\mathbf{K}_0, \dots, \mathbf{K}_D\}$.
  for $s = 0$ **to** D **do**
    $\mathbf{K}_s = \mathbb{O}$
  end for
  for $i = 1$ **to** *steps* **do**
    pick $d \in \{0, \dots, D\}$ according to the distribution $\pi(d)$ (see Eq. 11)
    if $d = 0$ **then**
      $\mathbf{H} = \mathbf{1}\mathbf{1}^\top$
    else
      set $\mathbf{L}$ a list of d random values in $[1, \dots, m]$ (with replica)
      $\mathbf{z} = \bigodot_{j \in \mathbf{L}} \mathbf{X}_{:,j}$
      $\mathbf{H} = \mathbf{z} \cdot \mathbf{z}^T$
    end if
    set $s = \mathcal{S}(d) \in \{0, \dots, D\}$ (apply the selector)
    $\mathbf{K}_s = \mathbf{K}_s + \mathbf{H}$
  end for
  for $s = 0$ **to** D **do**
    Normalize the kernel matrix $\mathbf{K}_s$
  end for
  return $\{\mathbf{K}_0, \dots, \mathbf{K}_D\}$

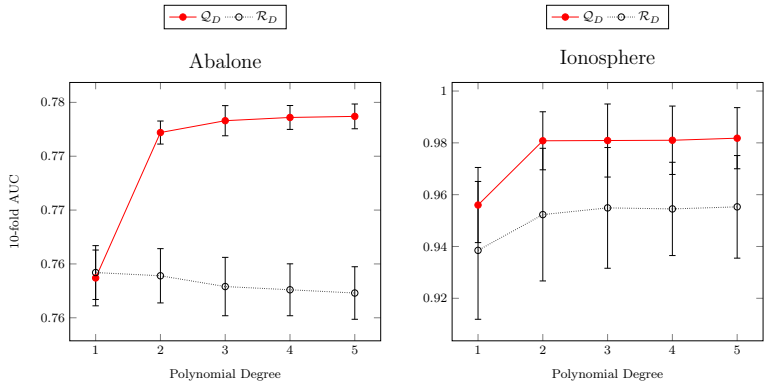

Fig. 6: Nested 10-fold AUC with standard deviation using EasyMKL with $\mathcal{Q}_D$ and $\mathcal{R}_D$ with different values of D.

low SR (high Frobenius norm) of low degree polynomial kernels which strongly influences the final SR of the $\mathbf{K}_{sum}$ kernel. These results also confirm the theoretical finding about the monotonicity of the spectral ratio for base kernels in $\mathcal{K}_D$.

5.4 Analysis of the weights assigned to the base kernels

Here, we present an analysis of the weights assigned by EasyMKL to the base kernels in the family $\mathcal{K}_D$ for different values of D. Figure 8 reports the histograms for the weights for $D \in \{3, 5, 10\}$ and on two datasets: Heart and Splice. Note that these results show how the optimal distribution of the weights, learned by

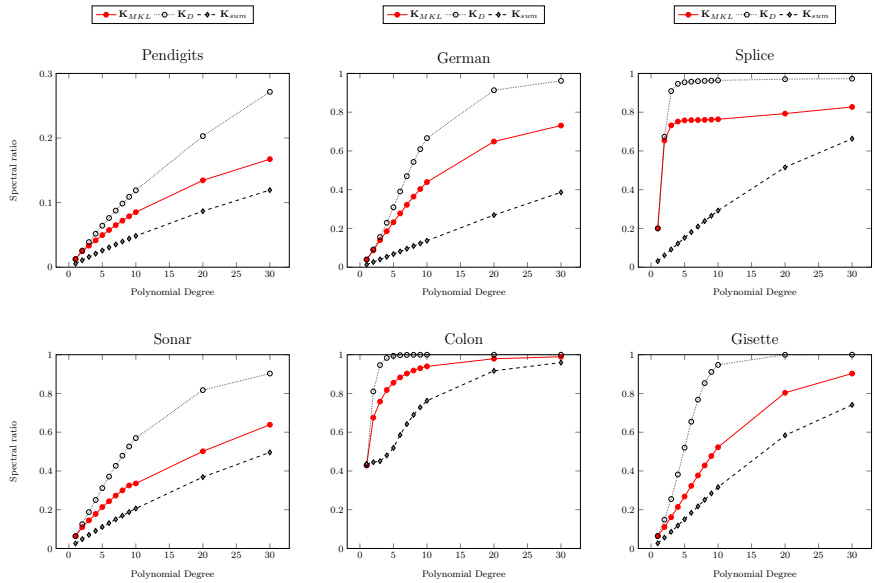

Fig. 7: Spectral ratio for different values of D. Our proposed solution $\mathbf{K}_{MKL}$ (solid red line) has been compared to the baselines $\mathbf{K}_D$ and $\mathbf{K}_{sum}$.

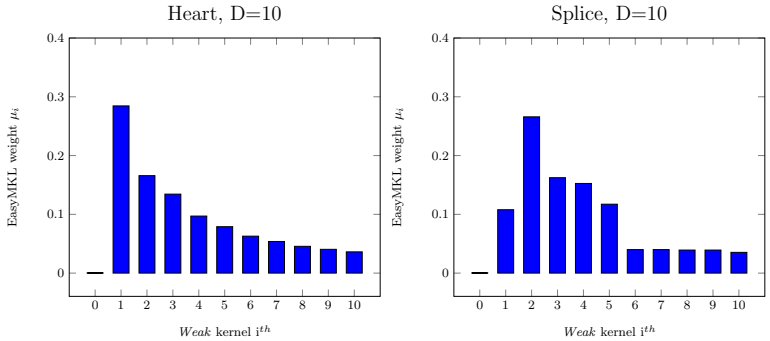

Fig. 8: The assigned weights η_i by using EasyMKL for the *weak* kernels in the families $\mathcal{K}_D$ for $D = 10$ of Heart and Splice datasets.

EasyMKL, is not the trivial choice of a single kernel but instead it is a combination of different kernels with similar expressiveness. In Heart, the weights are anti-correlated with respect to the degree of the base kernels. However, this behavior is rarely observed, in fact, for the Splice dataset, most of the total weight is shared among base kernels of degree in the range $[1, 5]$.

5.5 Catching the task complexity

In this section, we present experiments we have performed to demonstrate that, when base kernels of increasing expressiveness are given, then the weights com-

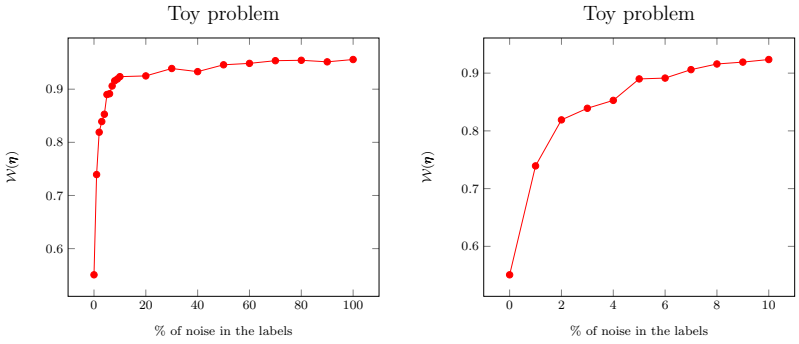

Fig. 9: The value of the function $\mathcal{W}$ with respect to the different solutions $\boldsymbol{\eta}$ of EasyMKL by using $\mathcal{K}_{10}$ as family of *base* kernels, for different percentages of swapped labels (i.e. noise).

puted by EasyMKL change increasing the complexity of the task giving more and more weight to more specific kernels.

For this, we generated a toy problem similar to the *Madelon* dataset used by Guyon[2]. To generate it, the same scikit-learn implementation for Python has been used. The task of the toy problem was a balanced binary classification task with 500 examples and 2 features. One of the features is informative, while the other is uncorrelated with the labels. The examples of different classes are initially arranged in two different clusters in the original space and then projected into the unit sphere (data was not linearly separable).

Starting from the original toy problem, noise is introduced in the task by swapping a fixed percentage of labels (randomly selected with replica). Then, models are trained by learning the coefficients of a DPP using $\mathcal{K}_D$ as base HPKs ($D = 10$). The hyper-parameter Λ has been fixed to 0.01 in this case.

We then observed how the center-of-mass of the list of assigned weights $\boldsymbol{\eta} = \{\eta_0, \ldots, \eta_D\}$ changed when increasing the complexity of the task. In particular, the center-of-mass is computed by $\mathcal{W}(\boldsymbol{\eta}) = \frac{1}{D}\sum_{s=0}^{D} s\ \eta_s$. This value is 0 whenever $\eta_0 = 1$ and $\eta_j = 0, \quad \forall j > 0$ and 1 whenever $\eta_D = 1$ and $\eta_j = 0, \quad \forall j = 1, \ldots, D-1$. $\mathcal{W}$ is higher if the weights are assigned to the most specific kernels.

The average values $\mathcal{W}(\boldsymbol{\eta})$ for 10 repetitions of this experiment are reported in Figure 9, for percentages of noise in $\{10i : i = 0, \ldots, 10\}$ (left) and $\{0, \ldots, 10\}$ (right). As expected, the increasing value of $\mathcal{W}$ with respect to the percentage of noise confirms that our method is able to catch the complexity of the problem and to distribute the weights to the base kernels consistently.

5.6 Analysis of the computational complexity

In this section we present an analysis of the computational complexity of our method (with $\mathcal{K}_{10}$, $\mathcal{K}_{20}$ and $\mathcal{K}_{30}$ as families of base kernels), compared to the

[2] http://clopinet.com/isabelle/Projects/NIPS2003/Slides/NIPS2003-Datasets.pdf

SVM with the Gaussian kernel[3]. The theoretical analysis of the complexity of EasyMKL, presented in [1], shows that EasyMKL has a linear increase of the computational complexity with respect to the number of base kernels. In fact, the optimization problem presented in Equation 1 has the same complexity of the standard SVM.

The difference in complexity between the two approaches is the evaluation of the base kernels and the computation of the weights, using the closed formula: $\eta_s = \frac{\mathbf{d}(\boldsymbol{\gamma}^*)}{\|\mathbf{d}(\boldsymbol{\gamma}^*)\|} \ \ \forall s = 1, \ldots, D$ (see Section 2.1).

This difference in complexity can be reduced evaluating the HPKs incrementally, noticing that if k_d is the HPK of degree d, then:

$$k_{d+1}(\mathbf{x}, \mathbf{z}) = k_d(\mathbf{x}, \mathbf{z}) k_1(\mathbf{x}, \mathbf{z}) \ \ \forall d = 1, \ldots, D-1. \tag{12}$$

Concerning the time in seconds, we performed an experiment using three benchmark datasets: Heart, Ionosphere and Splice. We trained the models using the same experimental framework presented in Section 5.1. The training times of the outer cross-validation cycle have been collected and divided for the number of repetitions. The computational times are evaluated using a CPU Intel Core i7-3632QM @ 2.20GHz. Finally, it is important to point out that using our method we are able to avoid the validation of the parameter γ of the Gaussian RBF kernel. For example, if the validation involves $V = 10$ different values of the hyper-parameter γ, then a fair comparison can be made by multiplying the $\mathbf{K}^{\gamma}_{RBF}$ column by 10.

	Training Time (average) in seconds			
Dataset	$\mathbf{K}^{\gamma}_{RBF}$	Our method $\mathcal{K}_{10}$	Our method $\mathcal{K}_{20}$	Our method $\mathcal{K}_{30}$
Heart	$0.016 \times V$	0.129	0.158	0.165
Ionosphere	$0.034 \times V$	0.243	0.276	0.341
Splice	$0.139 \times V$	2.092	2.400	2.882

Table 3: Computational time required by our method with three different families of base kernels ($\mathcal{K}_{10}$, $\mathcal{K}_{20}$ and $\mathcal{K}_{30}$) compared to the standard SVM with the Gaussian kernel using a validation set of parameters with cardinality V. The time is expressed in seconds and is the average of the training performed using a 10-fold cross-validation. It is important to highlight that in our method we are able to avoid the validation of the parameter γ of the Gaussian RBF kernel.

The results are summarized in Table 3. From these results we can notice that the complexity of our method is only one order of magnitude larger with respect to the simple SVM with a Gaussian kernel with fixed γ (i.e. with V=1). The difference is slightly higher when we use a larger amount of base kernels. As we noticed in the previous experimental results, 30 HPKs contain a sufficient level of complexity in order to learn effectively all the proposed tasks.

[3] For these experiments, the scikit-learn implementation of SVM at `http://scikit-learn.org/stable/modules/svm.html` has been used

Dataset	$\mathbf{K}_{MKL}$	$gHKL_{1.1}$	$gHKL_{1.5}$	$gHKL_{2.0}$
Haberman	$0.716_{\pm 0.014}$	$0.617_{\pm 0.166}$	$0.518_{\pm 0.110}$	$0.556_{\pm 0.070}$
Liver	$0.689_{\pm 0.056}$	$0.565_{\pm 0.109}$	$0.583_{\pm 0.110}$	$0.623_{\pm 0.038}$
Diabetes	$0.842_{\pm 0.027}$	$0.636_{\pm 0.118}$	$0.733_{\pm 0.058}$	$0.766_{\pm 0.046}$
Australian	$0.924_{\pm 0.081}$	$0.923_{\pm 0.101}$	$0.918_{\pm 0.049}$	$0.920_{\pm 0.045}$

Table 4: Nested 10-fold $\text{AUC}_{\pm std}$ using EasyMKL ($\mathbf{K}_{MKL}$) with $\mathcal{K}_{10}$ as *base* family compared to gHKL_ρ with $\rho \in \{1.1, 1.5, 2.0\}$.

5.7 A comparison with the Generalized Hierarchical Kernel Learning

In this set of experiments, the performance of the proposed method and the gHKL method presented in Section 2.2 are compared on a subset of UCI datasets. Unfortunately, gHKL is quite computational demanding and could only cope with very small datasets with few features. In these experiments, we used the implementation of the gHKL_ρ algorithm provided by the authors[4].

We performed a 10-fold cross validation for the AUC evaluation, tuning the parameter C of the SVM for gHKL_ρ [11] with a 3-fold cross validation selecting C in $\{10^i : i = -3, \ldots, 3\}$. The same procedure has been repeated for $\rho \in \{1.1, 1.5, 2.0\}$. The number of base kernels is fixed to 2^m, where m is the number of features, as in the original paper [11]. It is important to point out that, with $\rho = 2$ the HKL formulation of Bach [2] is obtained.

For our algorithm, we fixed $D = 10$ (i.e. the family of base kernels is $\mathcal{K}_{10}$) and validated the parameter Λ of EasyMKL by using the same methodology (3-fold cross validation) with $\Lambda \in \{\frac{v}{1-v} : v \in \{0.0, 0.1, \ldots, 0.9, 1.0\}\}$.

In Table 4, the AUC results are presented. From these results, we can note how our solution ($\mathbf{K}_{MKL}$) outperforms the gHKL_ρ method in this task.

5.8 Experiments on the MNIST dataset

In this section we report on the performance of our method on the MNIST dataset [14]. The MNIST dataset of handwritten digits is a real-world benchmark dataset and it is widely used to evaluate the classification performance of pattern recognition algorithms. Digits are size-normalized and centered in a fixed-size image.

In our experiments, we have generated the family of base kernels $\mathcal{K}_D$ using different values of D, and considered two different tasks. Firstly, the *even-odd* task, where the goal was to correctly discriminate between even and odd digits. Specifically, even digits $(0, 2, 4, 6, 8)$ have been selected as positives and odd digits $(1, 3, 5, 7, 9)$ as negatives. The second task is the typical multi-class task of recognizing the label of a given handwritten digit.

For the *even-odd* task, we used EasyMKL obtaining new kernels $\mathbf{K}_{MKL}$ as combination of the base kernels in the families, one for each value of the parameter D. The single homogeneous polynomial kernels $\mathbf{K}_D$ and the average kernels $\mathbf{K}_{sum}$ are our baselines. Finally, the parameter D has been selected from the set $\{1, 2, 3, 4, 5, 10, 15, 20, 25, 30, 35, 40\}$. We exploited the selected kernels ($\mathbf{K}_{MKL}$, $\mathbf{K}_D$ and $\mathbf{K}_{sum}$) for each value of D using a standard SVM. The best parameter

[4] `http://www.cse.iitb.ac.in/~pratik.j/ghkl/`

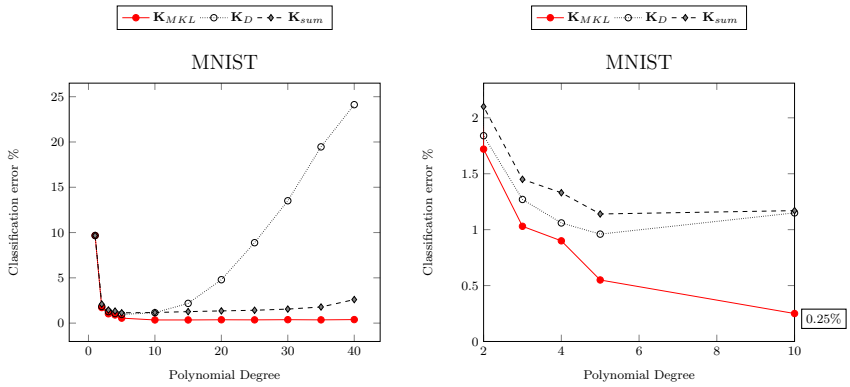

Fig. 10: Classification Error % of EasyMKL with $\mathcal{K}_D$ (K_{MKL}) compared to $\mathbf{K}_D$ and $\mathbf{K}_{sum}$ using different values of D.

C of the SVM has been selected from the set $\{2^n : n = 0, 1, 2, 3, 4\}$. A comparison of classification errors is depicted in Figure 10 for the *even-odd* task.

These results confirm the effectiveness in accuracy of our method. However, the average kernel $\mathbf{K}_{sum}$ represents a strong baseline in this case maintaining a good performance even when adding a large number of base kernels (i.e. all the HPKs $\mathcal{K}_D$ with $D > 10$).

For the experimental setting of the multi-class task an all-pairs approach has been used to cope with multi-class classification. In particular, 45 binary tasks, one for each possible pair of classes has been created. When a test example needs to be classified, each classifier is considered as a voter, and it votes for the class it predicts. Finally, the class with the highest number of votes is the predicted class of the algorithm.

The following steps have been performed to train the final model:

- Generation of the family of base HPK $\mathcal{K}_D$, with $D = 8$;
- Run of one EasyMKL for each binary task to learn a different kernel for each task ($\Lambda = \frac{0.01}{1-0.01}$);
- Training of the 45 binary SVM models using the kernels computed above (fixing $C = 4.0$).

In Table 5, the results of our experiments are summarized. In some cases, data has been deskwed in order to follow the current state-of-the-art results concerning SVMs (see [14]).

Also in this case, our methodology is able to create a model that outperforms the SVM with the optimal RBF kernel in terms of classification performance. Moreover, using deskewing, our method improves further its performance with an error of 0.8% (i.e. 80 erroneous digit classifications over $10,000$).

6 Conclusion and Future Work

Starting from a new perspective of the MKL problem, we have investigated on principled ways to design base kernels such to make their supervised combination

	RBF [14]	Our	Polynomial deskewed [14]	Best SVM deskewed [14]	Our deskewed
Classification Error %	1.4%	0.9%	1.1%	1.0%	0.8%

Table 5: Classification Error % of our method (Our) with and without data deskewing, with respect to the state-of-the-art results using the SVM with different kernels: RBF and Polynomial with optimal degree (i.e. 4). Moreover, we compared our results with respect to the best SVM result in literature (i.e. using the reduced set SVM with a polynomial kernel of degree 5).

really effective. Specifically, a hierarchy of features of different level of abstraction is considered. As a leading example of this methodology, a MKL approach is proposed to learn the kernel in the space of Dot-Product Polynomials (DPP), that is a positive combination of Homogeneous Polynomial Kernels (HPKs). We have given a deep theoretical analysis and empirically shown the merits of our approach comparing the effectiveness of the generated kernel against baseline kernels (including homogeneous and non homogeneous polynomials, RBF, etc...) and against the Hierarchical Kernel Learning (HKL) approach on many benchmark UCI/Statlog datasets and the large MNIST dataset. A deep experimental analysis has been also presented to get more insight of the method.
In the future, we want to investigate on extensions of the same methodology to general convolution kernels where the same type of hierarchy among features exist.

References

1. Aiolli, F., Donini, M.: Easymkl: a scalable multiple kernel learning algorithm. Neurocomputing **169**, 215–224 (2015). DOI 10.1016/j.neucom.2014.11.078
2. Bach, F.R.: Exploring large feature spaces with hierarchical multiple kernel learning. In: Advances in neural information processing systems, pp. 105–112 (2009)
3. Bache, K., Lichman, M.: Uci machine learning repository (2013)
4. Bucak, S.S., Jin, R., Jain, A.K.: Multiple kernel learning for visual object recognition: A review. Pattern Analysis and Machine Intelligence, IEEE Transactions on **36**(7), 1354–1369 (2014)
5. Castro, E., Gómez-Verdejo, V., Martínez-Ramón, M., Kiehl, K.A., Calhoun, V.D.: A multiple kernel learning approach to perform classification of groups from complex-valued fmri data analysis: Application to schizophrenia. NeuroImage **87**, 1–17 (2014)
6. Cortes, C., Kloft, M., Mohri, M.: Learning kernels using local rademacher complexity. In: C. Burges, L. Bottou, M. Welling, Z. Ghahramani, K. Weinberger (eds.) Advances in Neural Information Processing Systems 26, pp. 2760–2768. Curran Associates, Inc. (2013)
7. Cortes, C., Mohri, M., Rostamizadeh, A.: Generalization bounds for learning kernels. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel, pp. 247–254 (2010)
8. Damoulas, T., Girolami, M.A.: Probabilistic multi-class multi-kernel learning: on protein fold recognition and remote homology detection. Bioinformatics **24**(10), 1264–1270 (2008)
9. Do, H., Kalousis, A., Woznica, A., Hilario, M.: Margin and radius based multiple kernel learning. In: Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2009, Bled, Slovenia, September 7-11, 2009, Proceedings, Part I, pp. 330–343 (2009). DOI 10.1007/978-3-642-04180-8_39
10. Gönen, M., Alpaydin, E.: Multiple kernel learning algorithms. Journal of Machine Learning Research **12**, 2211–2268 (2011)
11. Jawanpuria, P., Nath, J.S., Ramakrishnan, G.: Generalized hierarchical kernel learning. Journal of Machine Learning Research **16**, 617–652 (2015)

12. Kar, P., Karnick, H.: Random feature maps for dot product kernels. In: Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2012, La Palma, Canary Islands, April 21-23, 2012, pp. 583–591 (2012)
13. Kloft, M., Blanchard, G.: The local rademacher complexity of lp-norm multiple kernel learning. In: Advances in Neural Information Processing Systems, pp. 2438–2446 (2011)
14. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2323 (1998). DOI 10.1109/5.726791
15. Livni, R., Shalev-Shwartz, S., Shamir, O.: An algorithm for training polynomial networks. arXiv preprint arXiv:1304.7045 (2013)
16. Livni, R., Shalev-Shwartz, S., Shamir, O.: On the computational efficiency of training neural networks. In: Advances in Neural Information Processing Systems, pp. 855–863 (2014)
17. Romera-Paredes, B., Aung, H., Bianchi-Berthouze, N., Pontil, M.: Multilinear multitask learning. In: Proceedings of the 30th International Conference on Machine Learning, pp. 1444–1452 (2013)
18. Schoenberg, I.J.: Positive definite functions on spheres. Duke Mathematical Journal **9(1)**, 96–108 (1942)
19. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press (2004)
20. Srebro, N.: Learning with matrix factorizations. Ph.D. thesis, Massachusetts Institute of Technology (2004)
21. Watrous, J.: Theory of quantum information. University of Waterloo Fall **128** (2011)
22. Xu, X., Tsang, I.W., Xu, D.: Soft margin multiple kernel learning. IEEE Transactions on Neural Networks and Learning Systems **24**(5), 749–761 (2013). DOI 10.1109/TNNLS.2012.2237183
23. Yang, J., Li, Y., Tian, Y., Duan, L., Gao, W.: Group-sensitive multiple kernel learning for object categorization. In: Computer Vision, 2009 IEEE 12th International Conference on, pp. 436–443. IEEE (2009)
24. Yu, S., Falck, T., Daemen, A., Tranchevent, L.C., Suykens, J.A., De Moor, B., Moreau, Y.: L2-norm multiple kernel learning and its application to biomedical data fusion. BMC bioinformatics **11**(1), 309 (2010)
25. Zien, A., Ong, C.S.: Multiclass multiple kernel learning. In: Proceedings of the 24th international conference on Machine learning, pp. 1191–1198. ACM (2007)