Learning deep kernels in the space of monotone conjunctive polynomials

Pattern Recognition Letters
journal homepage: www.elsevier.com

Learning deep kernels in the space of monotone Conjunctive Polynomials

Ivano Lauriola[a,b,**], Mirko Polato[b], Fabio Aiolli[b]

[a] *Fondazione Bruno Kessler, via Sommarive 18, Trento 38123, Italy*
[b] *University of Padova - dept of Mathematics, via Trieste 63, Padova 35121, Italy*

ABSTRACT

Dot-product kernels is a large family of kernel functions based on dot-product between examples. A recent result states that any dot-product kernel can be decomposed as a non-negative linear combination of homogeneous polynomial kernels of different degrees, and it is possible to learn the coefficients of the combination by exploiting the Multiple Kernel Learning (MKL) paradigm. In this paper it is proved that, under mild conditions, any homogeneous polynomial kernel defined on binary valued data can be decomposed in a parametrized finite linear non-negative combination of monotone conjunctive kernels. MKL has been employed to learn the parameters of the combination. Furthermore, we show that our solution produces a deep kernel whose feature space consists of hierarchically organized features of increasing complexity. We also emphasize the connection between our solution and existing deep kernel learning frameworks. A wide empirical assessment is presented to evaluate the proposed framework, and to compare it against the baselines on several categorical and binary datasets.

1. Introduction

Kernel methods (Shawe-Taylor et al., 2004) are general purpose algorithms widely used in machine learning problems, of which the Support Vector Machine (SVM) is the best known member. Kernel methods are composed by two modules: an algorithm that exploits dot-products between training examples, and a kernel function. The kernel is a symmetric positive semi-definite function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ corresponding to a dot-product in a Reproducing Kernel Hilbert Space (RKHS). In other words, there exists a function $\phi : \mathcal{X} \to \mathcal{K}$ which maps data from an input space $\mathcal{X}$ to a kernel (or feature) space $\mathcal{K}$, such that $\kappa(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$, where $\mathbf{x}, \mathbf{z} \in \mathcal{X}$. The choice of the kernel function defines the (implicit) representation of the data, and it is a key step for building good predictors.

Amongst others, Dot-Product Kernels (DPK) are a large family of kernel functions that can be expressed as a function of the dot product between examples, i.e. $\kappa(\mathbf{x}, \mathbf{z}) = f(\langle \mathbf{x}, \mathbf{z} \rangle)$. A recent result in the literature (Donini and Aiolli, 2016; Schoenberg, 1942) states that any DPK can be decomposed as a Dot-Product Polynomial (DPP), that is a linear non-negative combination of Homogeneous Polynomial Kernels (HPK), i.e., $\kappa(\mathbf{x}, \mathbf{z}) = \sum_{d=0}^{+\infty} a_d \langle \mathbf{x}, \mathbf{z} \rangle^d$, with appropriate coefficients $a_d \geq 0$. Moreover, in both the binary and the multi-class contexts, the Multiple Kernel Learning (MKL) framework (Donini and Aiolli, 2016; Lauriola et al., 2017) allows to generalize any DPK by making this combination non-parametric via the optimization of the coefficients a_d directly from data. In short, MKL algorithms combine a set of base kernels into a single one, by learning the combination coefficients α_d which maximize a quality criterion, such as the margin between classes. In the case of DPP optimization, base kernels are HPKs. Authors of the aforementioned work also showed that the DPPs rely on a deep feature space described by a hierarchical arrangement of features of increasing complexity. Specifically, polynomial features can be grouped by their complexity and they can be fully described by HPK of increasing degree. The combination of such groups of features, i.e. the combination of HPKs of increasing complexity, produces a deep kernel, that is the DPP. Authors also exposed the benefits of the deep feature space that characterizes DPPs, showing that this structure is much more expressive and powerful than shallow kernel learning solutions, where the hierarchical organization of features is not taken into account.

The main contribution of this work is the extension of the above-mentioned result in the case of binary valued data, i.e., $\mathbf{x}, \mathbf{z} \in \{0, 1\}^n$, showing that any DPK defined on Boolean vectors

**Corresponding author:
e-mail: ivano.lauriola@phd.unipd.it (Ivano Lauriola)

can be seen as a non-negative linear combination of monotone Conjunctive Kernels (mC-kernels) of different degrees. The mC-kernel of degree d counts the number of d-degree conjunctions that are satisfied in both the input vectors, and can be easily computed by the binomial coefficient $\kappa_\wedge^d(\mathbf{x},\mathbf{z}) = \binom{\langle \mathbf{x},\mathbf{z}\rangle}{d}$ (Polato et al., 2017). Throughout this paper, the combination of such kernels is called monotone Conjunctive Kernel Polynomial (mCKP). Besides, an empirical evaluation in terms of AUC score surrounds the theoretical analysis, showing the benefits of mCKP against different baselines. The evaluation has been carried on several categorical and binary-valued small- and medium-sized datasets. Furthermore, we emphasize the connection between the mCKP and the DPP, showing a relevant extension of the deep kernel learning framework proposed by Donini and Aiolli (2016).

The remainder of this paper is organized as follows. Section 2 defines the notation used throughout the paper and provides an introduction to the classification problem, kernels, and MKL. In Section 3 the proof of the decomposition of DPKs in a non-negative linear combination of mC-kernels (a mCKP) is provided together with additional theoretical results. Section 4 shows properties of the resulting feature space and the deep kernel learning framework. Then, in Section 5 all the performed experiments are reported and discussed. Finally, Section 6 concludes the paper.

2. Notation and background

In this work we consider binary classification tasks where training instances are binary vectors. Specifically, the n-dimensional training examples $\mathbf{X} \in \{0,1\}^{l\times n}$ are arranged in l rows, and $\mathbf{y} \in \{+1,-1\}^l$ is the vector of labels associated with the examples, where $+1$ and -1 represent the positive and negative class, respectively. The d-degree mC-kernel will be denoted by $\kappa_\wedge^d(\mathbf{x},\mathbf{z}) = \binom{\langle \mathbf{x},\mathbf{z}\rangle}{d}$, while the d-degree HPK will be indicated by $\kappa_{\text{HP}}^d(\mathbf{x},\mathbf{z}) = \langle \mathbf{x},\mathbf{z}\rangle^d$. The notation $\widetilde{\kappa}$ indicates the normalized version of a kernel function κ, defined as

$$\widetilde{\kappa}(\mathbf{x},\mathbf{z}) = \frac{\kappa(\mathbf{x},\mathbf{z})}{\sqrt{\kappa(\mathbf{x},\mathbf{x})\kappa(\mathbf{z},\mathbf{z})}}.$$

2.1. The Multiple Kernel Learning framework

When dealing with kernel machines, the choice of the kernel function is a key step to build good predictors. Typically, during the learning phase, a validation procedure is performed in which the user chooses the kernel that achieves the best validation score on the training data.

Unfortunately, validation works well when the number of hyperparameters is small, and the selection may depend on the user's choices and its a-priori knowledge. A recent approach to automatically learn the data representation, i.e., the kernel function, is known as Kernel Learning (KL), where the goal is to learn the best kernel function for a given problem, for example, by combining different base kernels as in the case of Multiple Kernel Learning (MKL) (Gönen and Alpaydın, 2011).

In short, the Multiple Kernel Learning framework takes in input a list of P base (or weak) kernels and combines them into a single one as follows:

$$\kappa_\mu(\mathbf{x},\mathbf{z}) = f\left(\{\kappa_r(\mathbf{x},\mathbf{z})\}_{r=1}^P, \boldsymbol{\mu}\right),$$

where $\kappa_r(\mathbf{x},\mathbf{z}) = \langle \phi_r(\mathbf{x}), \phi_r(\mathbf{z})\rangle$ is the r-th base kernel, based on the mapping function ϕ_r, and $\boldsymbol{\mu}$ is the weights vector which parametrizes the combination function f. Several functional forms exist in the literature for performing such combination. In this paper, linear non-negative combinations of P base kernels are considered, that is,

$$\kappa(\mathbf{x},\mathbf{z}) = \sum_{r=1}^P \mu_r \langle \phi_r(\mathbf{x}), \phi_r(\mathbf{z})\rangle = \sum_{r=1}^P \mu_r \kappa_r(\mathbf{x},\mathbf{z}),$$

where $\forall\ r \in [1,P], \mu_r \geq 0$. MKL algorithms learn the combination of base kernels optimizing a quality criterion of the resulting representation. Supported by several empirical results (Gönen and Alpaydın, 2011; Aiolli and Donini, 2015; Xu et al., 2013; Cortes et al., 2013b; Rakotomamonjy et al., 2008), the majority of MKL methods try to maximize the margin between classes. However, the margin is not the only key element (Chapelle et al., 2002) for a good representation. In fact, it has been proved how the radius of the Minimum Enclosing Ball (MEB) of data in feature space is also an important criterion to take into consideration. To this end, many algorithms based on the optimization of the ratio between the radius of the MEB and the margin have been recently proposed (Chung et al., 2003; Kalousis and Do, 2013).

The MKL framework has been applied in many applications, such as sentiment analysis (Poria et al., 2017), cancer subtype discovery (Speicher and Pfeifer, 2015), biological networks (Tsuda and Noble, 2004), malware detection (Anderson et al., 2012), Alzheimer's prediction (Liu et al., 2015), and it has been used to semplify the validation process (Lauriola et al., 2017; Massimo et al., 2016). Other examples of Kernel Learning applications are (Cortes et al., 2013a, 2010). For more information about the Multiple Kernel Learning framework, please refer to (Gönen and Alpaydın, 2011; Bach et al., 2004; Sonnenburg et al., 2006) for detailed surveys.

In this work two MKL algorithms have been used to learn the coefficients of the mCKP. These algorithms are EasyMKL (Aiolli and Donini, 2015) and R-MKL (Do et al., 2009). On one hand, EasyMKL is a scalable and time-efficient MKL algorithm which optimizes a simplification of the maximum-margin kernels combination. On the other hand, R-MKL optimizes a stronger quality criterion, i.e. the radius-margin ratio, sacrificing efficiency.

2.2. Deep kernel learning

An emerging challenge in kernel learning is to create kernels which reflect the deep structure that characterizes neural networks. Different definitions and examples of deep kernels exist in the literature. As a prime notable result, Cho and Saul (2009) analyzed deep kernels defined as successive embeddings of a mapping function ϕ, that is $k(\mathbf{x},\mathbf{z}) = \langle \phi(\phi(\dots\phi(\mathbf{x}))), \phi(\phi(\dots\phi(\mathbf{z})))\rangle$. Authors also showed that, in the case of arc-cosine embeddings, the resulting kernel mimics the

computation of large multilayer threshold networks. More recently, Donini and Aiolli (2016) showed a general framework based on MKL which rely on a sequence of kernels, HPKs in their case, of increasing expressiveness. Furthermore, authors empirically show the importance and the benefits of the deep structure w.r.t. shallow kernels. The same concept has been applied in the case of graph structured data (Massimo et al., 2016).

Differently, Yanardag and Vishwanathan (2015) presented a general framework for graph kernels inspired by latest advancements in natural language processing and deep learning. In that work, substructures of graphs are codified as words, and the similarities can be learned by means of word embeddings.

Moreover, ensembles which combine and aggregate the benefits from the two paradigms, deep neural networks and kernel methods, have been recently proposed. Some relevant examples consist of the application of kernelized layers on top of a deep neural network, as is the case of Deep Fried convnet (Yang et al., 2015), or the method proposed in (Wilson et al., 2016). A further newsworthy approach exploits kernels to pre-train neural networks (Tran et al., 2018; Navarin et al., 2018). The relevant question then becomes not which paradigm (e.g., kernel methods or neural networks) replaces the other, but whether we can combine the advantages of both approaches Wilson et al. (2016).

The union between deep networks and kernels has also been explored for performing two-sample testing Kirchler et al. (2020); Liu et al. (2020). In these works, the term deep kernel is used to describe the application of a kernel function on top of a deep neural network. Specifically, the deep network learns a new representation for the input data that is then fed into the kernel. Similarly, Li et al. Li et al. (2019) define a combination of different Gaussian kernels in which the training procedure simultaneously learns the parameter of the deep network, the parameters of the kernels and the combination parameters. All these "deep kernels" approaches are deeply different from what we propose here. In the just mentioned related works, the "deep" attribute refers to the use non-shallow neural network to learn a new input representation for the kernels, while in this paper we refer to the deep structure of the kernel's features themselves.

In particular, in this work we adopt the definition of deep kernels described by Donini and Aiolli (2016), i.e. kernels whose feature space can be described by a hierarchical arrangement of features of increasing complexity. We also show a strong connection between mCKPs defined as a linear non-negative combination of mC-kernels of increasing complexity and deep kernels.

3. DPK as linear combination of mC-kernels

The feature space of a HPK of a given degree d is formed by all the monomials of degree d each weighted by some coefficient. When the input vectors are binary, many of these monomials collide in a single one because the factors of the monomials x_i^p will have the same value for every $p \geq 1$. This observation allows us to give the following results concerning the relationship between HP-kernels and mC-kernels.

Theorem 1. *Given $\mathbf{x}, \mathbf{z} \in \{0,1\}^n$, then any HPK can be decomposed as a finite non-negative linear combination of mC-kernels (a mCKP) of the form:*

$$\kappa^d_{HP}(\mathbf{x}, \mathbf{z}) = \sum_{s=0}^{d} h(s,d)\ \kappa^s_\wedge(\mathbf{x}, \mathbf{z}), \qquad h(s,d) \geq 0.$$

Proof. Given $\mathbf{x}, \mathbf{z} \in \{0,1\}^n$, by definition:

$$\kappa^s_\wedge(\mathbf{x}, \mathbf{z}) = \binom{\langle \mathbf{x}, \mathbf{z} \rangle}{s} = \sum_{\mathbf{b} \in \mathbb{B}_s} \mathbf{x}^\mathbf{b} \mathbf{z}^\mathbf{b}$$

where $\mathbb{B}_s \equiv \{\mathbf{b} \in \{0,1\}^n \,\big|\, \|\mathbf{b}\|_1 = s\}$. Moreover:

$$\kappa^d_{\text{HP}}(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle^d = \left(\sum_{i=1}^{n} x_i z_i \right)^d = \sum_{\mathbf{p} \in \mathbb{P}_d} \underbrace{\left(d! \prod_{p_i \in \mathbf{p}} \frac{1}{p_i!} \right)}_{q(\mathbf{p},d)} \mathbf{x}^\mathbf{p} \mathbf{z}^\mathbf{p}$$
$$= \sum_{\mathbf{p} \in \mathbb{P}_d} q(\mathbf{p}, d) \mathbf{x}^\mathbf{p} \mathbf{z}^\mathbf{p}, \tag{1}$$

with $\mathbb{P}_d \equiv \{\mathbf{p} \in \mathbb{N}_0^n \,\big|\, \|\mathbf{p}\|_1 = d\}$, and $q(\mathbf{p}, d) \geq 0$. By partitioning the elements $\mathbf{p} \in \mathbb{P}_d$ so that vectors with the same number of non zero entries lie in the same set, Eq.1 can be rewritten as

$$\kappa^d_{\text{HP}}(\mathbf{x}, \mathbf{z}) = \sum_{s=0}^{d} \sum_{\mathbf{p} \in \mathbb{P}^s_d} q(\mathbf{p}, d) \mathbf{x}^\mathbf{p} \mathbf{z}^\mathbf{p}, \tag{2}$$

where $\mathbb{P}^s_d \equiv \{\mathbf{p} \in \mathbb{P}_d \,\big|\, \sum_{i=1}^n [\![p_i > 0]\!] = s\}$ and $[\![\cdot]\!]$ is the indicator function.

Let us now to further partition the set $\mathbb{P}^s_d$ in such a way that two vectors are in the same class of equivalence if and only if they share the same components greater than zero. Specifically, given $\mathbf{b} \in \mathbb{B}_s$, then $\mathbb{P}^s_d(\mathbf{b}) \equiv \{\mathbf{p} \in \mathbb{P}^s_d \mid \forall i : p_i > 0 \iff b_i = 1\}$. With this notation, the Eq. 2 can be rewritten as:

$$\kappa^d_{\text{HP}}(\mathbf{x}, \mathbf{z}) = \sum_{s=0}^{d} \sum_{\mathbf{b} \in \mathbb{B}_s} \mathbf{x}^\mathbf{b} \mathbf{z}^\mathbf{b} \sum_{\mathbf{p} \in \mathbb{P}^s_d(\mathbf{b})} q(\mathbf{p}, d). \tag{3}$$

Now, when s is fixed, then $\sum_{\mathbf{p} \in \mathbb{P}^s_d(\mathbf{b})} q(\mathbf{p}, d)$ is constant over the elements $\mathbf{b} \in \mathbb{B}_s$. This holds because the terms of the summations are the same. So, by taking any representative $\mathbf{b}_s \in \mathbb{B}_s$, the Eq. 3 can be rewritten as:

$$\kappa^d_{\text{HP}}(\mathbf{x}, \mathbf{z}) = \sum_{s=0}^{d} \underbrace{\left(\sum_{\mathbf{p} \in \mathbb{P}^s_d(\mathbf{b}_s)} q(\mathbf{p}, d) \right)}_{h(s,d)} \left(\sum_{\mathbf{b} \in \mathbb{B}_s} \mathbf{x}^\mathbf{b} \mathbf{z}^\mathbf{b} \right) = \sum_{s=0}^{d} h(s,d)\ \kappa^s_\wedge(\mathbf{x}, \mathbf{z}),$$

where $h(s,d) \geq 0$ by dfinition, and this ends the proof. □

In the following it is proved that, assuming Boolean input vectors with the same number of active variables, a similar result of Theorem 1 also holds when using normalized monotone conjunctive kernels.

Theorem 2. *Given* $\mathbf{x}, \mathbf{z} \in \{0,1\}^n$ *such that* $\|\mathbf{x}\|_1 = \|\mathbf{z}\|_1 = m$, $m > 0$, *then any HPK can be decomposed as a finite non-negative linear combination of normalized mC-kernels, that is:*

$$\kappa^d_{HP}(\mathbf{x}, \mathbf{z}) = \sum_{s=0}^{d} h(m, s, d)\, \widetilde{\kappa}^s_\wedge(\mathbf{x}, \mathbf{z}), \qquad h(m, s, d) \geq 0.$$

Proof. The normalized mC-kernel is defined as follows:

$$\widetilde{\kappa}^s_\wedge(\mathbf{x}, \mathbf{z}) = \frac{\binom{\langle \mathbf{x},\mathbf{z}\rangle}{s}}{\sqrt{\binom{\langle \mathbf{x},\mathbf{x}\rangle}{s}\binom{\langle \mathbf{z},\mathbf{z}\rangle}{s}}}.$$

Since $\|\mathbf{x}\|_1 = \|\mathbf{z}\|_1 = m$, then the kernel can be simplified:

$$\widetilde{\kappa}^s_\wedge(\mathbf{x}, \mathbf{z}) = \frac{\binom{\langle \mathbf{x},\mathbf{z}\rangle}{s}}{\sqrt{\binom{m}{s}\binom{m}{s}}} = \frac{1}{\binom{m}{s}} \kappa^s_\wedge(\mathbf{x}, \mathbf{z}),$$

where it is used the fact that for binary vectors $\|\cdot\|_1 = \|\cdot\|_2^2$ always holds and thus, by exploiting the Theorem 1:

$$\kappa^d_{\mathrm{HP}}(\mathbf{x}, \mathbf{z}) = \sum_{s=0}^{d} \underbrace{h(s, d)\binom{m}{s}}_{h(m,s,d)} \widetilde{\kappa}^s_\wedge(\mathbf{x}, \mathbf{z}) = \sum_{s=0}^{d} h(m, s, d)\, \widetilde{\kappa}^s_\wedge(\mathbf{x}, \mathbf{z}),$$

where $h(m, s, d) \geq 0$ by definition, and this ends the proof. □

As discussed by Donini and Aiolli (2016), under mild conditions, any DPK of the form $\kappa(\mathbf{x}, \mathbf{z}) = f(\langle \mathbf{x}, \mathbf{z}\rangle)$ can be seen as a DPP, that is $\kappa(\mathbf{x}, \mathbf{z}) = \sum_{d=0}^{+\infty} a_d \kappa^d_{\mathrm{HP}}(\mathbf{x}, \mathbf{z})$. Exploiting this result and the previous theorems, the following corollary can be obtained.

Corollary 1. *Given* $\mathbf{x}, \mathbf{z} \in \{0,1\}^n$ *such that* $\|\mathbf{x}\|_1 = \|\mathbf{z}\|_1 = m$, $m > 0$, *then any DPK can be decomposed as a finite non-negative linear combination of normalized mCK:*

$$\kappa(\mathbf{x}, \mathbf{z}) = f(\langle \mathbf{x}, \mathbf{z}\rangle) = \sum_{s=0}^{m} g(m, s)\, \widetilde{\kappa}^s_\wedge(\mathbf{x}, \mathbf{z}), \qquad g(m, s) \geq 0.$$

Proof. By inserting Theorem 2 in the above mentioned result of Donini and Aiolli (2016) it holds that:

$$\kappa(\mathbf{x}, \mathbf{z}) = f(\langle \mathbf{x}, \mathbf{z}\rangle) = \sum_{d=0}^{+\infty} a_d \sum_{s=0}^{d} h(m, s, d)\, \widetilde{\kappa}^s_\wedge(\mathbf{x}, \mathbf{z}). \tag{4}$$

Recalling that anytime $s > m$ then $h(m, s, d) = 0$, the inner summation can be limited up to $\min(m, d)$ as in the following

$$f(\langle \mathbf{x}, \mathbf{z}\rangle) = \sum_{d=0}^{+\infty} a_d \sum_{s=0}^{\min(d,m)} h(m, s, d)\, \widetilde{\kappa}^s_\wedge(\mathbf{x}, \mathbf{z}).$$

Let now $\hat{h}$ be the function defined as

$$\hat{h}(m, s, d) = \begin{cases} h(m, s, d) & \text{if } s \leq d \\ 0 & \text{otherwise} \end{cases}.$$

By embedding it in Eq. (4), then

$$f(\langle \mathbf{x}, \mathbf{z}\rangle) = \sum_{d=0}^{+\infty} a_d \sum_{s=0}^{m} \hat{h}(m, s, d)\, \widetilde{\kappa}^s_\wedge(\mathbf{x}, \mathbf{z}), \tag{5}$$

and, since the inner summation does not depend on d anymore, the Eq. (5) can be rewritten as

$$f(\langle \mathbf{x}, \mathbf{z}\rangle) = \sum_{s=0}^{m} \underbrace{\sum_{d=0}^{+\infty} a_d \hat{h}(m, s, d)}_{g(m,s)}\, \widetilde{\kappa}^s_\wedge(\mathbf{x}, \mathbf{z}),$$

where both a_d and $\hat{h}(m, s, d)$ are non negatives, and hence $g(m, s) \geq 0$, which proves the theorem. □

4. Learning Deep Kernels in the space of mCKP

In their previous work, Donini and Aiolli (2016) presented a framework to learn deep kernels as linear non-negative combinations of base kernels. The framework rely on two key aspects, which are:

linear decomposability The framework is able to learn a kernel that can be decomposed as a linear non-negative combination of base kernels. The combination can be automatically learned via MKL.

hierarchical feature space Base kernels have a increasingly complex feature space, and their combination describes a hierarchy of (deep) features of increasing complexity. Each layer of the architecture can be described by a base kernel containing heterogeneous features with similar complexity.

In the same work, authors claim that (i) any DPK can be decomposed as a linear non-negative combination of HPKs and (ii) the complexity of HPKs is strictly related to their degree, showing a possible instance of the framework.

In the case of mC-kernels, Polato et al. (2017) showed that the complexity of the kernel increases with the arity of the conjunction. This result together with the formal proof in Section 3 allow us to learn (deep) kernels in the space of monotone conjunctive polynomials, showing a second instance of the deep MKL framework. The hierarchical feature space of the mCKP is depicted in Fig. 1. Each layer of the DAG gathers together conjunctions of increasing arity, and it can be fully described with a mC-kernel. The arrows represent the dependencies between features. The feature $x_1 \wedge x_4$ has positive value iff the two simpler features x_1 and x_4 are both active.

This framework has two main strengths. Firstly, the resulting kernel relies on a rich feature space which involves all possible features from the hierarchy. Last but most important, this structure allows the MKL algorithm to learn a solution which encapsulates the complexity of a given problem by playing with the combination weights. Indeed, complex solutions can be easily learned by assigning high weights to complex kernels.

This approach cannot be considered *universal* since it can be instantiated only for HPKand mC-kernels. However, most well known and widely used kernels fall into these families of kernels thus the framework remains applicable in many contexts. Although the linear decomposability into simpler kernel functions is not mandatory and it is only a soft constraint, base kernels strongly need the hierarchical organization. Indeed, Donini

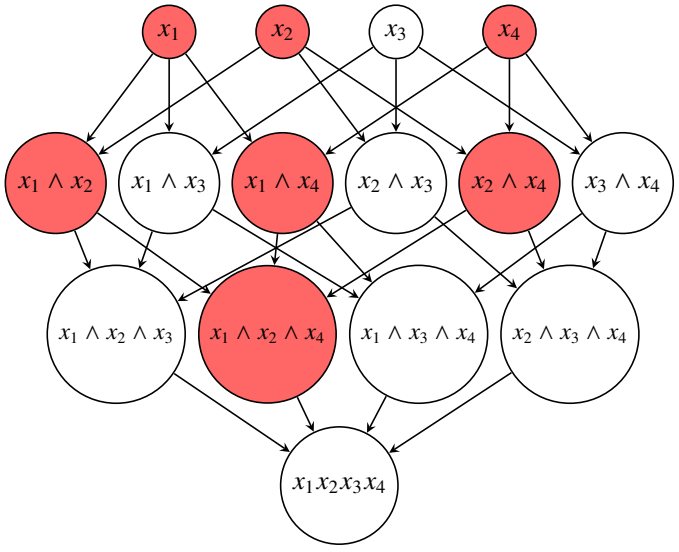

Fig. 1. A depiction of dependencies between conjunctive features with increasing complexity. The red nodes (i.e., active conjunctions) describe the diffusion of the information.

Table 1. Datasets description.

Dataset name	l	pos/neg (%)	n	$\|\cdot\|_1$
zoo	101	40/60	36	16
promoters	106	50/50	228	57
lymphography	148	45/55	44	15
house-votes	232	46/54	32	16
soybean	266	54/46	97	35
spect	267	79/21	44	22
breast	277	71/29	41	9
primary-tumor	339	41/59	34	15
crx	653	55/45	40	9
tic-tac-toe	958	65/35	27	9
flare	1066	49/51	41	11
car	1728	30/70	21	6
dna-bin	2000	47/53	180	47
splice	3175	48/52	240	60
kr-vs-kp	3196	52/48	73	36
mushroom	5644	62/38	98	22
HIV-1	6590	79/21	160	8
nursery	12960	36/64	27	8

and Aiolli (2016) empirically showed that the hierarchy plays a key role and the combination of base kernels that do not reflect a hierarchical feature space leads to a sub-optimal solution.

5. Experimental assessment

In this section the whole set of experiments is presented, including datasets description, data preprocessing, experimental methodology, and obtained results.

5.1. Datasets

Experiments have been performed using several binary-valued and categorical datasets obtained from the UCI Machine Learning Repository (Lichman, 2013) and the KEEL repository (Alcalá-Fdez et al., 2011). The datasets have different sizes and characteristics, that are reported in Table 1. A preprocessing phase has been performed to make all datasets binary. In detail, categorical and binary features have been mapped into binary ones by means of the *one-hot* encoding (Harris and Harris, 2013). Note that by one-hot encoding both binary and categorical features it is ensured that the number of active features m will be constant across the examples, allowing the application of the Corollary 1. Examples with missing values have been removed, and multiclass problems (`zoo`, `lymphography`, `primary-tumor`, `soybean`, `flare`, `car`, `dna`) have been transformed into binary ones by manually splitting the original classes in two balanced groups.

5.2. Evaluation and comparison

The base kernels used in our combinations are the identity matrix and normalized mC-kernels with degrees 1 up to m, where m is the number of non-zero features of the examples in a dataset. In this way, any Dot Product Kernel can be virtually learned, maximizing the expressiveness of the MKL algorithms according to the Corollary 1. The identity kernel allows to solve non-separable cases, where a couple of examples belonging to different classes have the same feature vector. These cases could easily occur in the case of binary-valued data. In order to evaluate the proposed framework, we considered kernels obtained by the following methods:

Linear : the linear kernel, which corresponds to the mC-kernel of arity 1;

Validation : a validation procedure is used to elect the single base kernel which performs best on the validation set;

Average : the average of base kernels, with weights $\forall r, \mu_r = \frac{1}{P}$ (even though this is a baseline, it is known to work fine);

EasyMKL : the MKL solution where coefficients $\boldsymbol{\mu}$ are computed by the EasyMKL method (Aiolli and Donini, 2015), a state-of-the-art maximum-margin MKL algorithm. EasyMKL has a regularization hyper-parameter λ which drives the kernels combination from maximum margin ($\lambda = 0$) to maximum centroids distance ($\lambda = 1$).

R-MKL : the MKL solution where coefficients $\boldsymbol{\mu}$ are computed by the R-MKL algorithm (Do et al., 2009). The algorithm has a regularization hyper-parameter C which defines a cost of training errors. This hyper-parameter is shared with the SVM.

A soft-margin SVM has been used with the selected/combined kernel for doing predictions. The available data have been split in training (60%), validation (20%) and test (20%) sets. While the training data has been used to train the models, the validation set has been used to select the best hyper-parameters configuration. The hyper-parameters are the arity of the mC-kernel in the case of the Validation baseline, with values in $\{1 \dots m\}$, and the $\lambda \in \{0, 0.1, 0.2 \dots 1\}$ for EasyMKL. The C value of the SVM (and R-MKL) has been selected with possible values in $\{10^i, i : -1 \dots 4\}$. After

Table 2. AUC score of MKL methods and baselines. Best scores are highlighted in bold characters.

Dataset	Linear	Validation	Average	R-MKL	EasyMKL
zoo	$100.00_{\pm 0.00}$	$100.00_{\pm 0.00}$	$100.00_{\pm 0.00}$	$100.00_{\pm 0.00}$	$100.00_{\pm 0.00}$
promoters	$\mathbf{98.92}_{\pm 1.66}$	$97.98_{\pm 1.64}$	$97.98_{\pm 1.87}$	$97.98_{\pm 1.87}$	$97.98_{\pm 1.87}$
lymphography	$88.83_{\pm 4.58}$	$88.83_{\pm 4.58}$	$90.39_{\pm 3.53}$	$\mathbf{92.82}_{\pm 3.54}$	$89.84_{\pm 4.01}$
house-votes-84	$98.31_{\pm 1.67}$	$98.31_{\pm 1.67}$	$99.19_{\pm 1.09}$	$\mathbf{99.39}_{\pm 1.02}$	$99.12_{\pm 1.13}$
soybean	$99.27_{\pm 0.56}$	$99.27_{\pm 0.56}$	$\mathbf{99.46}_{\pm 0.58}$	$\mathbf{99.46}_{\pm 0.58}$	$99.45_{\pm 0.57}$
SPECT	$80.96_{\pm 7.22}$	$80.30_{\pm 8.13}$	$74.07_{\pm 11.44}$	$74.33_{\pm 10.88}$	$\mathbf{81.93}_{\pm 6.55}$
breast	$\mathbf{71.39}_{\pm 4.60}$	$\mathbf{71.39}_{\pm 4.60}$	$67.35_{\pm 5.51}$	$69.36_{\pm 5.98}$	$70.33_{\pm 5.95}$
primary-tumor	$68.75_{\pm 5.38}$	$67.14_{\pm 6.26}$	$65.72_{\pm 6.59}$	$65.52_{\pm 6.74}$	$\mathbf{71.71}_{\pm 5.69}$
crx	$91.11_{\pm 2.00}$	$91.30_{\pm 1.97}$	$89.42_{\pm 2.06}$	$89.42_{\pm 2.05}$	$\mathbf{91.63}_{\pm 2.03}$
tic-tac-toe	$99.93_{\pm 0.10}$	$\mathbf{99.97}_{\pm 0.06}$	$99.80_{\pm 0.17}$	$99.80_{\pm 0.17}$	$99.85_{\pm 0.13}$
flare	$93.53_{\pm 1.13}$	$93.53_{\pm 1.13}$	$92.38_{\pm 2.21}$	$92.36_{\pm 2.24}$	$\mathbf{94.38}_{\pm 1.51}$
car	$99.96_{\pm 0.10}$	$99.96_{\pm 0.10}$	$100.00_{\pm 0.01}$	$100.00_{\pm 0.01}$	$\mathbf{100.00}_{\pm 0.00}$
dna	$97.29_{\pm 0.70}$	$\mathbf{98.38}_{\pm 0.54}$	$97.83_{\pm 0.71}$	$97.83_{\pm 0.71}$	$98.34_{\pm 0.54}$
splice	$98.63_{\pm 0.39}$	$\mathbf{99.36}_{\pm 0.21}$	$98.91_{\pm 0.30}$	$98.91_{\pm 0.30}$	$99.08_{\pm 0.27}$
kr-vs-kp	$99.74_{\pm 0.09}$	$99.91_{\pm 0.12}$	$99.92_{\pm 0.03}$	$99.92_{\pm 0.03}$	$\mathbf{99.94}_{\pm 0.02}$
mushroom	$100.00_{\pm 0.00}$	$100.00_{\pm 0.00}$	$100.00_{\pm 0.00}$	$100.00_{\pm 0.00}$	$100.00_{\pm 0.00}$
HIV-1	$97.61_{\pm 0.39}$	$97.61_{\pm 0.39}$	$97.73_{\pm 0.38}$	$97.73_{\pm 0.38}$	$\mathbf{97.77}_{\pm 0.38}$
nursery	$100.00_{\pm 0.00}$	$100.00_{\pm 0.00}$	$100.00_{\pm 0.00}$	$100.00_{\pm 0.00}$	$100.00_{\pm 0.00}$
average rank	2.8	2.4	2.7	2.6	**1.7**

the model selection phase, the AUC score has been computed on the test sets. In order to improve the statistical significance of the results, for each method, 10 runs with different splits (the same for all the methods) have been performed.

5.3. Results

The average AUCs evaluated on the test sets and the standard deviations are reported in Table 2.

As described in the table, the classical validation baseline achieves the lowest score on average. We recall that the validation uses a single kernel, limiting the expressiveness of the resulting solution. The average of base kernels achieves slightly better results than the validation. This boost in performance is given by the richer feature space on which the average kernel relies, that includes all possible conjunctions of variables. However, features contribute equally to the final solution. The complexity of the main problem is not taken into account, and layers of the architecture could introduce more noise than useful information. In other words, the average of base kernels exploits only a portion of the information included in the hierarchy. However, what is evident from the table is that the two MKL algorithms, i.e. EasyMKL and R-MKL, provide different results. The former achieves, on average, better result compared to the baselines (1.7 of average rank), whereas the latter does not improve significantly the results w.r.t the average kernel. However, as discussed previously, R-MKL optimizes a more sophisticated and grounded quality criterion than EasyMKL. Hence, we hypothesize that the causes of these results could reside in the structure of the weights vector.

5.4. Weights evaluation

In order to better understand the behaviour and the differences of the MKL algorithms, Fig. 2 reports the weights distribution for EasyMKL and R-MKL computed on some datasets.

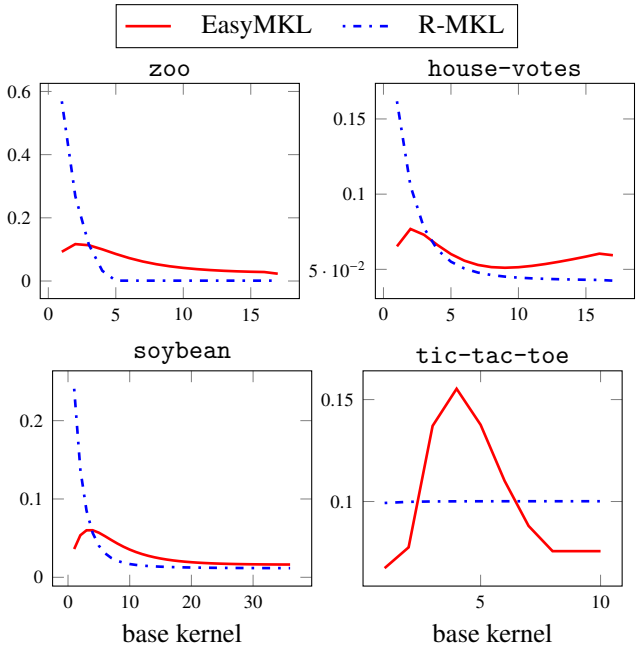

Fig. 2. Distribution of the weights when combining mC-kernels of degrees 1 to m.

From the figure it is self-evident that the two algorithms can give very different combinations. We can observe that the weight vectors learned by R-MKL are very sparse, and hence only a small subset of kernels are combined to form the final kernel. Differently, EasyMKL provides a dense solution, where each kernel, i.e. each layer of the features architecture, is involved in the resulting kernel. We hypothesize that the density of the solution plays a key role in the framework, and this is the reason why EasyMKL achieves often better results than R-MKL, even if the last one optimizes a better quality criterion.

Indeed, the sparsity prevents R-MKL from using the whole architecture of features, with the consequence that the learned kernel does not represent a deep feature space.

This is an empirical evidence concerning the benefits of a deep structure of features in the case of kernel methods.

5.5. The selection of the number of kernels

In the previous experiments, $m+1$ kernels have been used to learn the best Dot-Product Kernel for a given problem, where m is fixed and it depends on the number of non-zero features in each example. However, m may not be the best choice for a problem, and algorithms may overfit when this value increases. Moreover, from the Fig. 2 it is clear that the combination weights are focused more on simpler conjunctions. Additionally, for some datasets, the whole set of kernels used inside the MKL combination can be very expensive to compute and to handle in memory, such as `splice`, `promoters` or `kr-vs-kp`, where m and the number of examples are high. In other words, are features consisting of dozens of conjunctions relevant?

In order to better understand this aspect, an extension of the previous methodology have been analyzed, where only a subset of the whole set of m kernels has been considered. Formally, for each $d \in \{1, \ldots, m\}$, we consider combinations of $d+1$ kernels including mC-kernels with degrees $1, \ldots, d$ and the identity matrix. Note that, according to the theorems showed in Section 3, a combination of $d < m-1$ mC-kernels may limit the expressiveness of the algorithm, and the solution may not be the best DPK for the given problem but only an approximation. Only EasyMKL has been used in this experiment for two main reasons. Firstly, as discussed before, EasyMKL has a more interesting behavior if flanked by such features. Secondly, the time-complexity of R-MKL makes it hardly usable. The Fig. 3 shows the AUC score computed by EasyMKL while increasing the number d of mC-kernels on some datasets. In this experiment a single training/test split has been used.

Despite EasyMKL achieves better results compared to other baselines, it is clear that the number of kernels, i.e. the depth of the architecture of features, can be selected as hyper-parameter. Indeed, the performance of the system degrades in some cases while increasing the depth.

5.6. Is the deep structure important?

In the last part of our analysis, we show that the structure present in mC-kernels, and thus the deep architecture of mCKP, is useful to obtain good results with MKL optimization. Specifically, we consider two sets of base kernels based on conjunctive features with a different organization. The first set is aligned with previous experiments and it consists of d mC-kernels of increasing arity, where the r-th kernel contains all conjunctions of arity r. The same features are shuffled and spread over the second set of base kernels. In this case, conjunctions of arbitrary arity are randomly assigned to d base kernels. In our experiment, we combined both kernel sets with EasyMKL while increasing the maximum arity of the conjunction, corresponding to the number of kernels d in both sets.

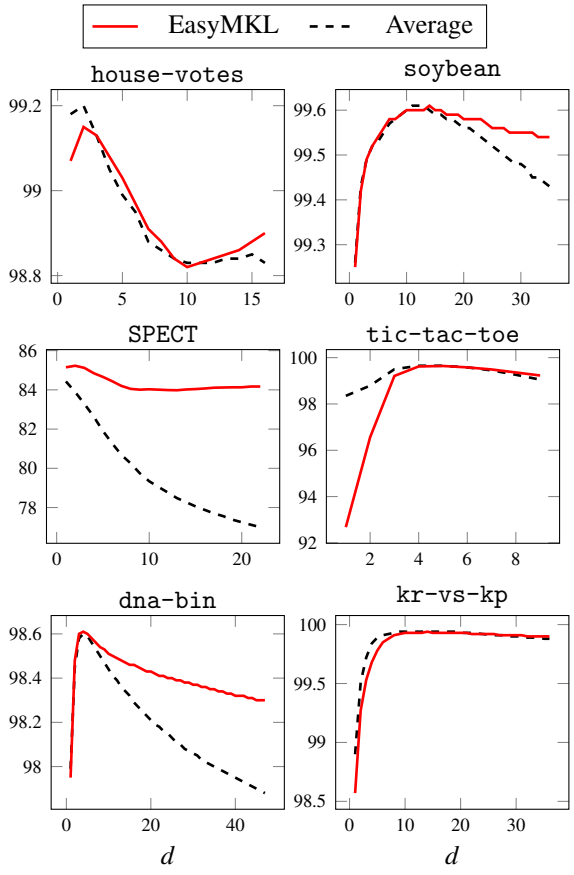

Fig. 3. AUC scores computed while increasing the maximum arity d of the mCKP from 1 up to m.

With this construction, both combinations leverage the same features and the same number of base kernels. The only difference is how those features are organized. Results of the comparison are exposed in Fig. 4. Clearly, the random organization always achieve worse results compared to our deep kernel based on a hierarchical distribution of features. Not surprisingly, the difference is highly pronounced when the number of kernels increases as we have even more shuffling (see `car`). This result suggests that mixing conjunctions of arity 2 with conjunctions of arity 10 is more dangerous than mixing conjunctions of arity 4 with conjunctions of arity 5.

6. Conclusions

In this paper we showed that, under mild conditions, any dot-product kernel can be decomposed as a linear non-negative combination of conjunctive kernels defined on binary-valued data. Moreover, inspired by the previous work of Donini and Aiolli (2016), we showed that the such combination produces a rich (and deep) feature space, and it can be directly optimized via Multiple Kernel Learning (MKL). The combination, here named monotone Conjunctive Polynomial, relies on a hierarchy of features of increasing expressiveness. A wide empirical assessment exposes the benefits of the proposed approach in terms of AUC.

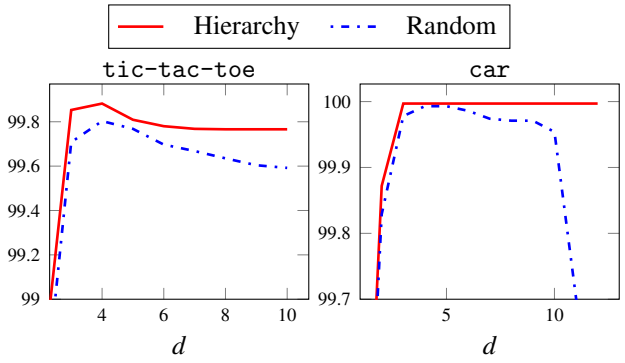

Fig. 4. Test AUC when using our proposed hierarchycal (and deep) structure and random features organization.

In the future, we plan to extend such framework towards two main directions. The first direction concerns the adaptation of such framework to other kernel families, including kernels for structures (strings, trees, graphs...) and convolution kernels. The idea is to produce a general and *universal* deep kernels framework leveraging the concepts described in this paper. Secondly, the combination strategy, i.e. the MKL part, needs to be improved. In this work we used two general MKL algorithms, which are EasyMKL and R-MKL. However, these algorithms may be not the most suitable for this type of kernels combination, and ad-hoc algorithms may even improve the solution in terms of accuracy and efficiency.

References

Aiolli, F., Donini, M., 2015. Easymkl: a scalable multiple kernel learning algorithm. Neurocomputing , 215–224.

Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., Herrera, F., 2011. Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. Journal of Multiple-Valued Logic & Soft Computing 17.

Anderson, B., Storlie, C., Lane, T., 2012. Multiple kernel learning clustering with an application to malware, in: Data Mining (ICDM), 2012 IEEE 12th International Conference on, IEEE. pp. 804–809.

Bach, F.R., Lanckriet, G.R., Jordan, M.I., 2004. Multiple kernel learning, conic duality, and the smo algorithm, in: Proceedings of the twenty-first international conference on Machine learning, ACM. p. 6.

Chapelle, O., Vapnik, V., Bousquet, O., Mukherjee, S., 2002. Choosing multiple parameters for support vector machines. Machine learning 46, 131–159.

Cho, Y., Saul, L.K., 2009. Kernel methods for deep learning, in: Advances in neural information processing systems, pp. 342–350.

Chung, K., Kao, W., Sun, C., Wang, L., Lin, C., 2003. Radius margin bounds for support vector machines with the RBF kernel. Neural Computation 15, 2643–2681.

Cortes, C., Kloft, M., Mohri, M., 2013a. Learning kernels using local rademacher complexity, in: Advances in neural information processing systems, pp. 2760–2768.

Cortes, C., Mohri, M., Rostamizadeh, A., 2010. Generalization bounds for learning kernels, in: Proceedings of the 27th International Conference on Machine Learning (ICML-10), pp. 247–254.

Cortes, C., Mohri, M., Rostamizadeh, A., 2013b. Multi-class classification with maximum margin multiple kernel, in: International Conference on Machine Learning, pp. 46–54.

Do, H., Kalousis, A., Woznica, A., Hilario, M., 2009. Margin and radius based multiple kernel learning. Machine Learning and Knowledge Discovery in Databases , 330–343.

Donini, M., Aiolli, F., 2016. Learning deep kernels in the space of dot product polynomials. Machine Learning , 1–25.

Gönen, M., Alpaydın, E., 2011. Multiple kernel learning algorithms. Journal of Machine Learning Research 12, 2211–2268.

Harris, D.M., Harris, S.L., 2013. Digital Design and Computer Architecture (Second Edition). second edition ed.. Morgan Kaufmann, Boston.

Kalousis, A., Do, H.T., 2013. Convex formulations of radius-margin based support vector machines, in: Proceedings of the 30th International Conference on Machine Learning, pp. 169–177.

Kirchler, M., Khorasani, S., Kloft, M., Lippert, C., 2020. Two-sample testing using deep learning, in: AISTATS.

Lauriola, I., Donini, M., Aiolli, F., 2017. Learning dot-product polynomials for multiclass problems, in: Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN).

Li, W., Sutherland, D.J., Strathmann, H., Gretton, A., 2019. Learning deep kernels for exponential family densities, in: Chaudhuri, K., Salakhutdinov, R. (Eds.), Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, PMLR. pp. 6737–6746. URL: `http://proceedings.mlr.press/v97/wenliang19a.html`.

Lichman, M., 2013. UCI machine learning repository. URL: `http://archive.ics.uci.edu/ml`.

Liu, F., Xu, W., Lu, J., Zhang, G., Gretton, A., Sutherland, D.J., 2020. Learning deep kernels for non-parametric two-sample tests.

Liu, X., Zhou, L., Wang, L., Zhang, J., Yin, J., Shen, D., 2015. An efficient radius-incorporated mkl algorithm for alzheimer s disease prediction. Pattern Recognition 48, 2141–2150.

Massimo, C.M., Navarin, N., Sperduti, A., 2016. Hyper-parameter tuning for graph kernels via multiple kernel learning, in: International Conference on Neural Information Processing, Springer. pp. 214–223.

Navarin, N., Tran, D.V., Sperduti, A., 2018. Pre-training graph neural networks with kernels.

Polato, M., Lauriola, I., Aiolli, F., 2017. Classification of categorical data in the feature space of monotone dnfs, in: 26th International Conference on Artificial Neural Networks (ICANN), pp. 279–286.

Poria, S., Peng, H., Hussain, A., Howard, N., Cambria, E., 2017. Ensemble application of convolutional neural networks and multiple kernel learning for multimodal sentiment analysis. Neurocomputing .

Rakotomamonjy, A., Bach, F.R., Canu, S., Grandvalet, Y., 2008. Simplemkl. Journal of Machine Learning Research 9, 2491–2521.

Schoenberg, I.J., 1942. Positive definite functions on spheres. Duke Math. J. 9, 96–108.

Shawe-Taylor, J., Cristianini, N., et al., 2004. Kernel methods for pattern analysis. Cambridge university press.

Sonnenburg, S., Rätsch, G., Schäfer, C., Schölkopf, B., 2006. Large scale multiple kernel learning. Journal of Machine Learning Research 7, 1531–1565.

Speicher, N.K., Pfeifer, N., 2015. Integrating different data types by regularized unsupervised multiple kernel learning with application to cancer subtype discovery. Bioinformatics 31, i268–i275.

Tran, D.V., Navarin, N., Sperduti, A., 2018. On filter size in graph convolutional networks, in: 2018 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1534–1541. doi:`10.1109/SSCI.2018.8628758`.

Tsuda, K., Noble, W.S., 2004. Learning kernels from biological networks by maximizing entropy. Bioinformatics 20, 326–333.

Wilson, A.G., Hu, Z., Salakhutdinov, R., Xing, E.P., 2016. Deep kernel learning, in: Artificial Intelligence and Statistics, pp. 370–378.

Xu, X., Tsang, I.W., Xu, D., 2013. Soft margin multiple kernel learning. IEEE transactions on neural networks and learning systems 24, 749–761.

Yanardag, P., Vishwanathan, S., 2015. Deep graph kernels, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM. pp. 1365–1374.

Yang, Z., Moczulski, M., Denil, M., de Freitas, N., Smola, A., Song, L., Wang, Z., 2015. Deep fried convnets, in: Proceedings of the IEEE International Conference on Computer Vision, pp. 1476–1483.