A New Frontier of Kernel Design for Structured Data

Kilho Shin YSHIN@AI.U-HYOGO.AC.JP
University of Hyogo, 7-1-28 Minatojima-Minami, Kobe 6500047, Japan

Abstract

Many kernels for discretely structured data in the literature are designed within the framework of the convolution kernel and its generalization, the mapping kernel. The two most important advantages to use this framework are an easy-to-check criteria of positive definiteness of the resulting kernels and efficient computation based on the dynamic programming methodology. On the other hand, the recent theory of partitionable kernels reveals that the known kernels only take advantage of a very small portion of the potential of the framework. In fact, we have good opportunities to find novel and important kernels in the unexplored area. In this paper, we shed light on a novel important class of kernels within the framework: We give a mathematical characterization of the class, and then, based on the characterization, we show a parametric method to optimize kernels of the class to specific problems. Also, we present some experimental results that show that the new kernels are promising in both accuracy and efficiency.

1. Introduction

When applying the kernel method to specific problems, the choice of the kernel to use is critical. First, the kernel must be positive definite (Berg et al., 1984). This condition is mandatory not only for a reproducing kernel Hilbert space to exist but for performing kernel multivariate analysis, such as SVM. Secondly, an efficient algorithm to compute the kernel must exist. Even if these two conditions are met, the choice of the kernel still impact the results.

Assume that data objects are (row) vectors of real values of the same dimension, for example. They are in a common space, and the kernel $K(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{x}\boldsymbol{y}^{\mathsf{T}}$ is canonically associated. Although this kernel is positive definite and efficiently computed, whether we can obtain good results with this kernel is not certain. We need a latitude to *tune* kernels, and the polynomial and Gaussian kernels provide methods. The polynomial kernel is defined as $K(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x}\boldsymbol{y}^{\mathsf{T}}+c)^d$ with the adjustable parameters c and d, and Gaussian kernel as $K(\boldsymbol{x},\boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{2\sigma^2}\right)$ with the adjustable parameter σ. We can optimize these parameters by means of cross validation and grid search, for example.

Also, the kernel method is good at dealing with data structured in the form of strings, trees and graphs. In fact, we have an effective framework to incorporate structural information of data objects into the design of kernels. To illustrate, we first see the R-convolution kernel that Haussler (1999) introduced as follows:

$$K(x,y) = \sum_{(\boldsymbol{x}',\boldsymbol{y}')\in R^{-1}(x)\times R^{-1}(y)} \prod_{d=1}^{D} \kappa_d(x'_d, y'_d).$$

$R \subseteqq \chi'_1\times\cdots\times\chi'_D\times\chi$ is a relation to relate a data object $x\in\chi$ to a D-dimensional vector $\boldsymbol{x}'\in\chi'_1\times\cdots\times\chi'_D$, and $\kappa_d : \chi'_d\times\chi'_d\to\mathbb{R}$ are kernels. Haussler has proved that K is positive definite, if κ_d's are positive definite.

The spectrum kernel (Leslie et al., 2002) is a good example of the R-convolution kernel. Given a pair of strings (x,y), the kernel counts the pairs $(\boldsymbol{x}',\boldsymbol{y}')$ of D-character-long substrings of x and y with $\boldsymbol{x}'=\boldsymbol{y}'$. As an R-convolution kernel, we define it so that

$$R = \{(\boldsymbol{x}', x) \mid x\in\Sigma^*, \boldsymbol{x}'\in\Sigma^D, \boldsymbol{x}'\subseteqq x\}$$

and $\kappa_d(x',y') = \delta_{x',y'}$ with Kronecker's delta function.

Shin & Kuboyama (2008) have generalized the R-convolution kernel to introduce the mapping kernel:

$$K(x,y) = \sum_{(\boldsymbol{x}',\boldsymbol{y}')\in M_{x,y}} \prod_{d=1}^{|\boldsymbol{x}'|} \kappa(x'_d, y'_d).$$

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

The Cartesian product $R^{-1}(x) \times R^{-1}(y)$ is replaced with arbitrary $M_{x,y}$. Also, K is positive definite, if $M_{x,y}$ is symmetric ($(\boldsymbol{x}', \boldsymbol{y}') \in M_{x,y} \Leftrightarrow (\boldsymbol{y}', \boldsymbol{x}') \in M_{y,x}$) and transitive ($(\boldsymbol{x}', \boldsymbol{y}') \in M_{x,y} \wedge (\boldsymbol{y}', \boldsymbol{z}') \in M_{y,z} \Rightarrow (\boldsymbol{x}', \boldsymbol{z}') \in M_{x,z}$), and κ is positive definite.

The *all-subsequence kernel* is an example of the mapping kernel. The kernel counts the identical sparse substring pairs $(\boldsymbol{x}', \boldsymbol{y}')$ of variable length with the weights of $\lambda^{|\boldsymbol{x}'|}$. To define this formally, we let

$$M_{x,y} = \{(\boldsymbol{x}', \boldsymbol{y}') \mid |\boldsymbol{x}'| = |\boldsymbol{y}'|, \boldsymbol{x}' \subseteqq x, \boldsymbol{y}' \subseteqq y\} \quad (1)$$

and $\kappa(x', y') = \lambda \delta_{x',y'}$. Apparently, the family of $M_{x,y}$ is symmetric and transitive, and κ is positive definite. Therefore, this kernel is positive definite.

Another interesting example of the mapping kernel is derived from the generic framework of edit distance. Let $d(x, y) = \min_{\sigma: x \to y} \gamma(\sigma)$ be an edit distance between objects x and y, where $\sigma : x \to y$ and $\gamma(\sigma)$ denote an edit script from x to y and its cost. As Cortes et al. (2004) proved for Levenshtein edit distance, $e^{-\lambda d(x,y)}$ is not always positive definite. By contrast, when we define a parameterized family of distances by $d_\lambda(x, y) = -\frac{1}{\lambda} \log \left(\sum_{\sigma: x \to y} e^{-\lambda \gamma(\sigma)} \right)$ for $\lambda > 0$, $e^{-\lambda d_\lambda(x,y)}$ is not only positive definite in most of cases, but $\lim_{\lambda \to \infty} d_\lambda(x, y) = d(x, y)$ also holds. Hence, we define mapping kernels K_λ by

$$K_\lambda(x, y) = \frac{e^{-\lambda d_\lambda(x,y)}}{e^{-\lambda(|x|+|y|)}} = \sum_{(\boldsymbol{x}', \boldsymbol{y}') \in M_{\boldsymbol{x}, \boldsymbol{y}}} \prod_{i=1}^{|\boldsymbol{x}'|} e^{-\lambda \gamma(x'_i \to y'_i) + 2\lambda}.$$

$M_{x,y}$ is the set of mappings (traces) across all of the edit scripts that transform x into y (Wagner & Fischer, 1974; Taï, 1979). In the case of Levenshtein distance, $M_{x,y}$ is determined by Formula (1). Also, $\gamma(x' \to y')$ is the cost of substituting y' for x', and we assume that the costs of deletion and insertion are 1. For Levenshtein distance, and for Taï (Taï, 1979), constrained (Zhang, 1995) and Lu (Lu, 1979) distances for trees, the family of $M_{x,y}$ is symmetric and transitive, and hence, the derived mapping kernels are positive definite, while this does not hold true for the less-constrained distance for trees (Kuboyama et al., 2006).

Many other important kernels in the literature are also shown to be mapping kernels, including the gap-sensitive string kernel (Lodhi et al., 2001), the parse tree kernel (Collins & Duffy, 2001) and the elastic tree kernel (Kashima & Koyanagi, 2002), and furthermore, Shin & Kuboyama (2010) reported that 18 of the 19 tree kernels surveyed were mapping kernels.

In the original paper (Shin & Kuboyama, 2008), the mapping kernel is defined in the more general form of

$$K(x, y) = \sum_{(\boldsymbol{x}', \boldsymbol{y}') \in M_{x,y}} k(\boldsymbol{x}', \boldsymbol{y}'), \quad (2)$$

and the following interesting and useful theorem holds.

Theorem 1 (Shin & Kuboyama (2008)) *For Formula (2), the following are equivalent.*

1. *$\{M_{x,y} \mid x, y \in \chi\}$ is symmetric and transitive.*
2. *K is positive definite for any positive definite k.*

Nevertheless, the examples in the literature all employ the restricted setting of $k(\boldsymbol{x}', \boldsymbol{y}') = \prod \kappa(x'_i, y'_i)$. Is this just a coincidence? What if $\sum \kappa(x'_d, y'_d)$, $\sum \kappa(x'_d, y'_d)^2$ or $\prod \kappa_1(x'_d, y'_d) \cdot (\sum \kappa_2(x'_d, y'_d))$ substitutes for $\prod \kappa(x'_d, y'_d)$? Answering these questions was the initial motivation of our research.

The answer to the first question is negative: The setting of $k(\boldsymbol{x}', \boldsymbol{y}') = \prod \kappa(x'_i, y'_i)$ is important for computational feasibility of the resulting mapping kernels. For example, for the all-subsequence kernel and the kernel derived from Levenshtein distance, Formula (3) holds thanks to $k(\boldsymbol{x}', \boldsymbol{y}') = \prod \kappa(x'_i, y'_i)$, and the formula yields an efficient dynamic-programming-based algorithm to compute the respective mapping kernels.

$$\begin{aligned} K(x, y) =& (\kappa(x_1, y_1) - 1) K(x \setminus \{x_1\}, y \setminus \{y_1\}) \\ &+ K(x \setminus \{x_1\}, y) + K(x, y \setminus \{y_1\}) \end{aligned} \quad (3)$$

To answer the second question, we need the theory of *partitionable kernels* that Shin (2011) has developed. The theory defines partitionable kernels so that, if a partitionable kernel is specified for the inner kernel k of a mapping kernel, and if $M_{x,y}$ meets a certain condition, the mapping kernel has a set of recurrence formulas to compute itself. All of $\prod \kappa(x'_d, y'_d)$, $\sum \kappa(x'_d, y'_d)$, $\sum \kappa(x'_d, y'_d)^2$ and $\prod \kappa_1(x'_d, y'_d) \cdot (\sum \kappa_2(x'_d, y'_d))$ are examples of the partitionable kernel.

Furthermore, each partitionable kernel is associated with a unique non-negative integer, named *hidden degree*, and a partitionable kernel of hidden degree h is defined through association with other $h - 1$ partitionable kernels (Definition 2). In other words, if $h > 1$, whether a given string kernel is partitionable is not always an easy question to answer, since it requires to find the remaining (hidden) associated kernels. Contrarily, a kernel k is a partitionable kernel of hidden degree one, if, and only if, it is of the form of $k(\boldsymbol{x}', \boldsymbol{y}') = \lambda^{1-|\boldsymbol{x}'|} \prod_{d=1}^{|\boldsymbol{x}'|} \kappa(x'_d, y'_d)$. The hidden degree

also relates to computational complexity of mapping kernels: Approximately, a mapping kernel defined with a partitionable kernel of hidden degree h has h times greater complexity than when $h = 1$.

We have two findings from the above. First, all of the examples of mapping kernels in the literature merely take advantage of the easiest portion of the entire partitionable kernels, that is, the class of partitionable kernels of hidden degree one.

Secondly, the research of partitionable kernels of hidden degree two is significantly attractive for two reasons: It is an exploration to a new frontier of kernel design, and we have good opportunities to discover novel important kernels; Mapping kernels for hidden degree two will be merely twice slower than the mapping kernels known so far in the literature.

Nevertheless, it is also a fact that we have a problem to solve: The space of partitionable kernels of hidden degree two might be too broad to find *good* kernels. Since partitionable kernels of hidden degree one are characterized to be $k(\boldsymbol{x}', \boldsymbol{y}') = \lambda^{1-|\boldsymbol{x}'|} \prod_{d=1}^{|\boldsymbol{x}'|} \kappa(x'_d, y'_d)$, we can perform an efficient linear search on λ to find optimal kernels. By contrast, no such a characterization is known for hidden degree two.

To solve this problem, we start with giving a mathematical characterization to the partitionable kernels of hidden degree two. In fact, our main theorem (Theorem 4) asserts: Any partitionable kernel κ of hidden degree two belongs to one of three types; Furthermore, a normal form pair $(\tilde{\kappa}_1, \tilde{\kappa}_2)$ of partitionable kernels is determined according to the type to which κ belongs, and $\kappa = a\tilde{\kappa}_1 + b\tilde{\kappa}_2$ holds for some $a, b \in \mathbb{R}$; $(\tilde{\kappa}_1, \tilde{\kappa}_2)$ includes at most one adjustable parameter β.

Interestingly, this theorem presents an efficient method to tune partitionable kernels of hidden degree two: We first divide the entire space for hidden degree two into three subspaces, each of which corresponds to one of the three types, and then evaluate cross validation scores for $t\tilde{\kappa}_1 + (1-t)\tilde{\kappa}_2$ changing t in each subspace. Thus, if $(\tilde{\kappa}_1, \tilde{\kappa}_2)$ includes the parameter β, we can find optimal kernels by means of the two-dimensional grid search on t and β, for example. Otherwise, the faster linear search on t can apply.

2. The theory of partitionable kernels

In this section, we outline the theory introduced by Shin (2011), while Lemma 1 is a new result.

Like many other string kernels in the literature, a partitionable kernel compares two strings of the same length. Such a string kernel, called an *integral kernel* for convenience, is represented as a series of kernels $\kappa^{[*]} = \{\kappa^{[i]} : \Sigma^i \times \Sigma^i \longrightarrow \mathbb{R} \mid i \in \{0\} \cup \mathbb{N}\}$. Σ is an alphabet, and Σ^0 is the singleton set of the null string $\{\boldsymbol{\emptyset}\}$. A partitionable kernel is an integral kernel that meets the condition specified in Definition 2, based on the notion of *partitions* of string pairs (Definition 1).

Definition 1 *Let $\boldsymbol{x} = x_1 \dots x_k \in \Sigma^k$ and $\boldsymbol{y} = y_1 \dots y_k \in \Sigma^k$. A* two-partition *of $(\boldsymbol{x}, \boldsymbol{y})$ is a pair of string pairs $((\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}), (\boldsymbol{x}^{\mathsf{R}}, \boldsymbol{y}^{\mathsf{R}}))$ such that $\boldsymbol{x}^{\mathsf{L}} = x_1 \dots x_j$, $\boldsymbol{y}^{\mathsf{L}} = y_1 \dots y_j$, $\boldsymbol{x}^{\mathsf{R}} = x_{j+1} \dots x_k$ and $\boldsymbol{y}^{\mathsf{R}} = y_{j+1} \dots y_k$ for some $0 \le j \le k$. If $j = 0$, $\boldsymbol{x}^{\mathsf{L}} = \boldsymbol{y}^{\mathsf{L}} = \boldsymbol{\emptyset}$; If $j = k$, $\boldsymbol{x}^{\mathsf{R}} = \boldsymbol{y}^{\mathsf{R}} = \boldsymbol{\emptyset}$.*

For a family of integral kernels $\boldsymbol{\kappa} = (\kappa_1^{[*]}, \dots \kappa_n^{[*]})$, we let $\boldsymbol{\kappa}^{[k]}(\boldsymbol{x}, \boldsymbol{y})$ denote the n-dimensional row vector of kernels $\left(\kappa_1^{[k]}(\boldsymbol{x}, \boldsymbol{y}), \dots, \kappa_n^{[k]}(\boldsymbol{x}, \boldsymbol{y})\right)$.

Definition 2 *A family of integral kernels $\boldsymbol{\kappa}$ is said to be* partitionable, *if, and only if, there exist n-dimensional square matrices $\mathbf{Q}_1, \dots, \mathbf{Q}_n$ such that*

$$\kappa_i^{[k]}(\boldsymbol{x}, \boldsymbol{y}) = \left(\boldsymbol{\kappa}^{[|\boldsymbol{x}^{\mathsf{L}}|]}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})\right) \mathbf{Q}_i \left(\boldsymbol{\kappa}^{[|\boldsymbol{x}^{\mathsf{R}}|]}(\boldsymbol{x}^{\mathsf{R}}, \boldsymbol{y}^{\mathsf{R}})\right)^{\mathsf{T}}$$

for $\forall i = 1, \dots, n$, $\forall k \ge 0$, $\forall \boldsymbol{x} \in \Sigma^k$, $\forall \boldsymbol{y} \in \Sigma^k$ and $\forall((\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}), (\boldsymbol{x}^{\mathsf{R}}, \boldsymbol{y}^{\mathsf{R}}))$, a two-partition of $(\boldsymbol{x}, \boldsymbol{y})$.

When an integral kernel $\kappa^{[*]}$ is a member of a partitionable family of integral kernels, we simply say that $\kappa^{[*]}$ is partitionable. Furthermore, if $\kappa^{[*]}$ is partitionable, we define its *hidden degree*, denoted by $\mathrm{hd}(\kappa^{[*]})$, as the minimum of n such that $\kappa^{[*]}$ is a member of a partitionable kernel family $(\kappa_1^{[*]}, \kappa_2^{[*]} \dots, \kappa_n^{[*]})$. For the constant function $\kappa^{[*]} \equiv 0$, we define $\mathrm{hd}(\kappa^{[*]}) = 0$.

The partitionable kernel class is abundant, that is, includes an infinite number of instances.

Example 1 *$e_0^{[*]}, e_1^{[*]}, \dots, e_\infty^{[*]}$ defined below are partitionable, where $\kappa : \Sigma \times \Sigma \to \mathbb{R}$ is an arbitrary kernel.*

$$e_d^{[k]}(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1, & \text{if } d = 0, \\ 0, & \text{if } d > k, \\ \displaystyle\sum_{1 \le i_1 < \dots < i_d \le k} \prod_{j=1}^{d} \kappa(x_{i_j}, y_{i_j}), & \text{if } 0 < d \le k, \end{cases}$$

$$e_\infty^{[k]}(\boldsymbol{x}, \boldsymbol{y}) = \prod_{i=1}^{k} \kappa(x_i, y_i).$$

Evidently, $e_d^{[k]}$ is the elementary symmetric polynomial of degree d over $\kappa(x_1, y_1), \dots, \kappa(x_k, y_k)$, and $e_d^{[*]}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i+j=d} e_i^{[*]}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}) \cdot e_j^{[*]}(\boldsymbol{x}^{\mathsf{R}}, \boldsymbol{y}^{\mathsf{R}})$ and $e_\infty^{[*]}(\boldsymbol{x}, \boldsymbol{y}) = e_\infty^{[*]}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}) \cdot e_\infty^{[*]}(\boldsymbol{x}^{\mathsf{R}}, \boldsymbol{y}^{\mathsf{R}})$ hold. Thus, we have $\mathrm{hd}(e_d^{[*]}) = d + 1$ and $\mathrm{hd}(e_\infty^{[*]}) = 1$.

Example 2 illustrates the relation between partitionable kernels and computation of mapping kernels.

Example 2 *We determine $M_{x,y}$ by Formula (1), and let $K_d(x,y) = \sum_{(\boldsymbol{x}',\boldsymbol{y}')\in M_{x,y}} e_d^{[*]}(\boldsymbol{x}',\boldsymbol{y}')$. Then,*

$$K_d(x,y) = \sum_{(\boldsymbol{x}',\boldsymbol{y}')\in M^\circ_{x,y}} e_d^{[*]}(\boldsymbol{x}',\boldsymbol{y}') + K_d(x\setminus\{x_1\},y) + K_d(x,y\setminus\{y_1\}) - K_d(x\setminus\{x_1\},y\setminus\{y_1\})$$

holds, for the first characters x_1 and y_1 of x and y, and $M^\circ_{x,y} = \{(x_1\boldsymbol{x}', y_1\boldsymbol{y}') \mid (\boldsymbol{x}',\boldsymbol{y}') \in M_{x\setminus\{x_1\},y\setminus\{y_1\}}\}$. To obtain recurrence formulas to compute K_d, we need to evaluate the first term. When $d=\infty$,

$$\sum_{(\boldsymbol{x}',\boldsymbol{y}')\in M^\circ_{x,y}} e_\infty^{[*]}(\boldsymbol{x}',\boldsymbol{y}') = \kappa(x_1,y_1)K_\infty(x\setminus\{x_1\},y\setminus\{y_1\})$$

follows from $e_\infty^{[*]}(x_1\boldsymbol{x}', y_1\boldsymbol{y}') = \kappa(x_1,y_1)e_\infty^{[*]}(\boldsymbol{x}',\boldsymbol{y}')$. Thus, we obtain Formula (3). For $d=0$,

$$\sum_{(\boldsymbol{x}',\boldsymbol{y}')\in M^\circ_{x,y}} e_0^{[*]}(\boldsymbol{x}',\boldsymbol{y}') = K_0(x\setminus\{x_1\},y\setminus\{y_1\}).$$

apparently holds. For $d>0$,

$$\sum_{(\boldsymbol{x}',\boldsymbol{y}')\in M^\circ_{x,y}} e_d^{[*]}(\boldsymbol{x}',\boldsymbol{y}') = K_d(x\setminus\{x_1\},y\setminus\{y_1\}) + \kappa(x_1,y_1)K_{d-1}(x\setminus\{x_1\},y\setminus\{y_1\}).$$

follows from $e_d^{[*]}(x_1\boldsymbol{x}', y_1\boldsymbol{y}') = e_0^{[1]}(x_1,y_1)e_d^{[*]}(\boldsymbol{x}',\boldsymbol{y}') + e_1^{[1]}(x_1,y_1)e_{d-1}^{[*]}(\boldsymbol{x}',\boldsymbol{y}')$. Thus, to compute K_d, we have to compute $K_0,\dots,K_{d-1}$ together.

Theorem 2 generalizes the results of Example 2.

Theorem 2 (Shin (2011, Theorem 3)) *Let $M_{x,y}$ be pretty decomposable and $(\kappa_1^{[*]},\dots,\kappa_n^{[*]})$ be a partitionable kernel family. There exists a set of recurrence formulas that reduces the computation of*

$$K_i(x,y) = \sum_{(\boldsymbol{x}',\boldsymbol{y}')\in M_{x,y}} \kappa_i^{[*]}(\boldsymbol{x}',\boldsymbol{y}') \quad \text{for } i=1,\dots,n$$

to the computation of $K_{i_j}(x_j,y_j)$ with $j=1,\dots,N$ and $i_j\in\{1,\dots,n\}$, such that at least one of x_j and y_j is smaller than x or y.

Intuitively speaking, if a family of $M_{x,y}$ is *pretty decomposable*, any member $M_{x,y}$ can be decomposed into one or more M_{x_j,y_j} such that at least one of x_j and y_j is smaller than x or y, and M_{x_j,y_j}'s are combined with each other by means of the set union $\cup$ and the string concatenation $\|$: For two string $\boldsymbol{x}'$ and $\boldsymbol{x}''$, $\boldsymbol{x}'\|\boldsymbol{x}''$ means the canonical concatenation of $\boldsymbol{x}'$ and $\boldsymbol{x}''$; For two sets S' and S'' of strings, we let $S'\|S'' = \{\boldsymbol{x}'\|\boldsymbol{x}'' \mid \boldsymbol{x}'\in S', \boldsymbol{x}''\in S''\}$. For example, $M_{x,y}$ by Formula (1) can be decomposed as

$$M_{x,y} = \left(M_{x_1,y_1}\|M_{x\setminus\{x_1\},y\setminus\{y_1\}}\right)\cup M_{x,y\setminus\{y_1\}}\cup M_{x\setminus\{x_1\},y}.$$

For the formal definition, we ask the reader to refer to Shin (2011, Definition 6). Furthermore, the following are important to note: Almost all of the kernels in the literature categorized as mapping kernels have recurrence formulas due to Theorem 2, and the recursive formulas can be explicitly derived from $\mathbf{Q}_i$ (Shin, 2011, Lemma 1); If $(\kappa_1^{[*]},\dots,\kappa_n^{[*]})$ is a minimal partitionable family, we need to compute $K_1,\dots,K_n$ simultaneously, where K_i is determined by $\kappa_i^{[*]}$; Hence, $\mathrm{hd}(\kappa^{[*]})$ is a factor to determine the complexity to compute the mapping kernel K determined by $\kappa^{[*]}$.

Partitionable string kernels have many other good properties: The space of partitionable kernels is closed under addition, scalar multiplication and multiplication; In particular, $\mathrm{hd}(\kappa^{[*]}+\lambda^{[*]}) \le \mathrm{hd}(\kappa^{[*]})+\mathrm{hd}(\lambda^{[*]})$, $\mathrm{hd}(c\kappa^{[*]}) = \mathrm{hd}(\kappa^{[*]})$ and $\mathrm{hd}(\kappa^{[*]}\cdot\lambda^{[*]}) \le \mathrm{hd}(\kappa^{[*]})\cdot\mathrm{hd}(\lambda^{[*]})$ hold; If $\kappa_1^{[1]},\dots,\kappa_n^{[1]}$ are all positive definite and $\mathbf{Q}_i$'s are all non-negative, $\kappa_1^{[*]},\dots,\kappa_n^{[*]}$ are all positive definite; If $\mathbf{Q}_i$'s are all symmetric, $\kappa_1^{[*]},\dots,\kappa_n^{[*]}$ are all symmetric, that is, $\kappa_i^{[k]}(x_1\dots x_k, y_1\dots y_k) = \kappa_i^{[k]}(x_{\sigma(1)}\dots x_{\sigma(k)}, y_{\sigma(1)}\dots y_{\sigma(k)})$ holds for arbitrary k and permutation $\sigma\in\mathfrak{S}_k$.

In addition, Theorem 3 characterizes the partitionable kernels of hidden degree one.

Theorem 3 (Theorem 2, (Shin, 2011)) *$\mathrm{hd}(\kappa^{[*]}) = 1$, if, and only if, $\kappa^{[k]}(\boldsymbol{x},\boldsymbol{y}) = (\kappa^{[0]})^{1-k}\prod_{i=1}^k \kappa^{[1]}(x_i,y_i)$.*

Lemma 1 is a new result, and asserts that $\kappa_i^{[k]}(\boldsymbol{x},\boldsymbol{y})$ is a homogeneous polynomial in $\kappa_j^{[1]}(x_\ell,y_\ell)$.

Lemma 1 *For a partitionable family $(\kappa_1^{[*]},\dots,\kappa_n^{[*]})$ with associated matrices $\mathbf{Q}_1,\dots,\mathbf{Q}_n$, the following holds for $k>2$, where $\mathbf{Q}_i[a,b]$ denotes the (a,b)-element of $\mathbf{Q}_i$.*

$$\kappa_i^{[k]}(\boldsymbol{x},\boldsymbol{y}) = \sum_{j_1=1}^n\cdots\sum_{j_k=1}^n\left[\prod_{b=1}^k \kappa_{j_b}^{[1]}(x_b,y_b)\cdot\left(\sum_{\ell_1=1}^n\cdots\sum_{\ell_{k-2}=1}^n \mathbf{Q}_i[j_1,\ell_1]\cdot\prod_{a=2}^{k-2}\mathbf{Q}_{\ell_{a-1}}[j_a,\ell_a]\cdot\mathbf{Q}_{\ell_{k-2}}[j_{k-1},j_k]\right)\right]$$

Proof. Follows from $\kappa_i^{[k]} = \boldsymbol{\kappa}^{[1]}\mathbf{Q}_i\left(\boldsymbol{\kappa}^{[k-1]}\right)^\mathsf{T}$ and the mathematical induction on k. □

Table 1. Definition of $\tilde{\kappa}_1$ and $\tilde{\kappa}_2$

TYPE I

$$c_{i:j_1,\ldots,j_k} = \begin{cases} 1, & \text{if } k = 0; \\ 1, & \text{if } k \geq 1 \text{ and } j_1 = \cdots = j_k = i; \\ 0, & \text{otherwise.} \end{cases}$$

TYPE II

$$c_{1:j_1,\ldots,j_k} = \begin{cases} 1, & \text{if } k = 0; \\ \beta^{\frac{o_2(j_1,\ldots,j_k)}{2}}, & \text{if } k \geq 1 \text{ and } 2 \mid o_2(j_1,\ldots,j_k); \\ 0, & \text{if } k \geq 1 \text{ and } 2 \nmid o_2(j_1,\ldots,j_k). \end{cases}$$

$$c_{2:j_1,\ldots,j_k} = \begin{cases} \beta^{\frac{o_2(j_1,\ldots,j_k)-1}{2}}, & \text{if } k \geq 1 \text{ and } 2 \nmid o_2(j_1,\ldots,j_k); \\ 0, & \text{otherwise.} \end{cases}$$

TYPE III

$c_{1:j_1,\ldots,j_k} = 1$ for $o_2(j_1,\ldots,j_k) = 0$;
$c_{1:j_1,\ldots,j_k} = \beta F_{o_2(j_1,\ldots,j_k)-2}(\beta)$ for $o_2(j_1,\ldots,j_k) > 0$.
$c_{2:j_1,\ldots,j_k} = F_{o_2(j_1,\ldots,j_k)-1}(\beta)$.

$F_{-2}(x) = \frac{1}{x}, F_{-1}(x) = 0, F_n(x) = F_{n-1}(x) + xF_{n-2}(x)$

Table 2. $\mathbf{Q}_1$ and $\mathbf{Q}_2$ for Type I, II and III

	TYPE I	TYPE II	TYPE III
$\mathbf{Q}_1 =$	$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & \beta \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & \beta \end{bmatrix}$
$\mathbf{Q}_2 =$	$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$

3. Characterization of partitionable kernels with hidden degree two

3.1. The main theorem

Although partitionable kernels with hidden degree higher than one will definitely produce novel mapping kernels, too high hidden degrees will result in computationally unfeasible mapping kernels. This motivates us to study the case of hidden degree two.

Theorem 4 *If $\kappa^{[*]}$ is partitionable with hidden degree two, $\kappa^{[*]}$ is a linear combination of a kernel pair $(\tilde{\kappa}_1^{[*]}, \tilde{\kappa}_2^{[*]})$ of Type I, Type II or Type III determined by Table 1 and 2. We use the following notation:*

$$\tilde{\kappa}_i^{[k]}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{(j_1,\ldots,j_k)\in\{1,2\}^k} c_{i:j_1,\ldots,j_k} \tilde{\kappa}_{j_1}^{[1]}(x_1, y_1) \cdots \tilde{\kappa}_{j_k}^{[1]}(x_k, y_k)$$

$$o_2(j_1,\ldots,j_k) = |\{\ell \mid j_\ell = 2, \ell = 1,\ldots,k\}| \quad \textit{and}$$

$$F_{-2}(x) = \frac{1}{x}, F_{-1}(x) = 0,$$
$$F_n(x) = F_{n-1}(x) + xF_{n-2}(x) \quad \textit{for } n \geq 0.$$

Example 3 *The kernel $e_1^{[*]}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{*} \kappa(x_i, y_i)$ is a partitionable kernel of hidden degree two, and belongs to Type II. In fact, if we let $\beta = 0$, $\tilde{\kappa}_1^{[1]} = 1$ and $\tilde{\kappa}_2^{[1]} = \kappa$, we have $\tilde{\kappa}_1^{[*]} = e_0^{[*]}$ and $\tilde{\kappa}_2^{[*]} = e_1^{[*]}$.*

Example 4 *The kernel $(e_\infty^{[*]} \cdot e_1^{[*]})(\boldsymbol{x}, \boldsymbol{y}) = \prod_{i=1}^{*} \kappa_1(x_i, y_i) \left(\sum_{i=1}^{*} \kappa_2(x_i, y_i)\right)$ is also a partitionable kernel of hidden degree two, and belongs to Type II. In fact, if we let $\beta = 0$, $\tilde{\kappa}_1^{[1]} = \kappa_1$ and $\tilde{\kappa}_2^{[1]} = \kappa_1\kappa_2$, we have $\tilde{\kappa}_1^{[*]} = e_\infty^{[*]}$ and $\tilde{\kappa}_2^{[*]} = e_\infty^{[*]} \cdot e_1^{[*]}$.*

3.2. Preliminaries

We show some preliminary results that we will use to prove Theorem 4.

When kernels forming a partitionable family are linearly transformed, the conversion rules for the associated matrices are given as follows.

Lemma 2 *For a partitionable kernel family $(\kappa_1^{[*]}, \ldots, \kappa_n^{[*]})$, we replace $\kappa_1^{[*]}$ with $\tilde{\kappa}^{[*]} = \sum_{i=1}^{n} a_i \kappa_i^{[*]}$ such that $a_1 \neq 0$. Further, we let*

$$\mathbf{Q}_i'[1,1] = \frac{\mathbf{Q}_i[1,1]}{a_1^2}, \ \mathbf{Q}_i'[j,1] = \frac{\mathbf{Q}_i[j,1]}{a_1} - \frac{a_j \mathbf{Q}_i[1,1]}{a_1^2}$$

$$\mathbf{Q}_i'[1,k] = \frac{\mathbf{Q}_i[1,k]}{a_1} - \frac{a_k \mathbf{Q}_i[1,1]}{a_1^2} \quad \textit{and}$$

$$\mathbf{Q}_i'[j,k] = \mathbf{Q}_i[j,k] - \frac{a_j \mathbf{Q}_i[1,k]}{a_1} - \frac{a_k \mathbf{Q}_i[j,1]}{a_1} + \frac{a_j a_k \mathbf{Q}_i[1,1]}{a_1^2} \quad \textit{for } 2 \leq j,k \leq n.$$

Then, the kernel family $(\tilde{\kappa}^{[*]}, \kappa_2^{[*]}, \ldots, \kappa_n^{[*]})$ is also partitionable with associated matrices $\tilde{\mathbf{Q}}_1, \ldots, \tilde{\mathbf{Q}}_n$ determined by $\tilde{\mathbf{Q}}_1 = \sum_{i=0}^{n} a_i \mathbf{Q}_i'$ and $\tilde{\mathbf{Q}}_i = \mathbf{Q}_i'$ for $i = 2, \ldots, n$.

The following lemmas show important properties of $\mathbf{Q}_1$ and $\mathbf{Q}_2$ assuming $n = 2$. Their proofs are given in Appendix Appendix A and Appendix B.

Lemma 3 *If $\mathrm{hd}(\kappa^{[*]}) = 2$, $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are symmetric.*

Lemma 4 *If $\mathrm{hd}(\kappa^{[*]}) = 2$, there exists a partitionable kernel family $(\kappa_1^{[*]}, \kappa_2^{[*]})$ such that $\kappa^{[*]}$ is a linear combination of $\kappa_1^{[*]}$ and $\kappa_2^{[*]}$, and the associated matrix $\mathbf{Q}_1$ is diagonal.*

3.3. A proof to Theorem 4

By Lemma 4, $\kappa^{[*]}$ is a linear combination of some partitionable kernel family $(\kappa_1^{[*]}, \kappa_2^{[*]})$ with $\mathbf{Q}_1 = \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix}$ and $\mathbf{Q}_2 = \begin{bmatrix} \delta & \phi \\ \phi & \epsilon \end{bmatrix}$. First, $\alpha \neq 0$ holds, since $\mathrm{hd}(\kappa^{[*]}) =$

1 would follow from $\kappa_1^{[*]} = \kappa_2^{[0]}\beta\kappa_2^{[*]}$, otherwise. Secondly, we can assume $\alpha = 1$, since we can modify $\mathbf{Q}_1$ into $\mathbf{Q}_1 = \begin{bmatrix} 1 & 0 \\ 0 & \alpha\beta \end{bmatrix}$ by replacing $\kappa_1^{[*]}$ with $\alpha\kappa_1^{[*]}$. Assuming $\alpha = 1$, we evaluate $\kappa_1^{[3]}$ in two different ways as follows.

$$\begin{aligned}
\kappa_1^{[3]}(\boldsymbol{x},\boldsymbol{y}) &= \kappa_1^{[2]}((x_1,x_2),(y_1,y_2))\kappa_1^{[1]}(x_3,y_3) \\
&\quad + \beta\kappa_2^{[2]}((x_1,x_2),(y_1,y_2))\kappa_2^{[1]}(x_3,y_3) \\
&= \kappa_1^{[1]}(x_1,y_1)\kappa_1^{[1]}(x_2,y_2)\kappa_1^{[1]}(x_3,y_3) \\
&\quad + \beta\kappa_2^{[1]}(x_1,y_1)\kappa_2^{[1]}(x_2,y_2)\kappa_1^{[1]}(x_3,y_3) \\
&\quad + \beta\delta\kappa_1^{[1]}(x_1,y_1)\kappa_1^{[1]}(x_2,y_2)\kappa_2^{[1]}(x_3,y_3) \\
&\quad + \beta\phi\kappa_1^{[1]}(x_1,y_1)\kappa_2^{[1]}(x_2,y_2)\kappa_2^{[1]}(x_3,y_3) \\
&\quad + \beta\phi\kappa_2^{[1]}(x_1,y_1)\kappa_1^{[1]}(x_2,y_2)\kappa_2^{[1]}(x_3,y_3) \\
&\quad + \beta\epsilon\kappa_2^{[1]}(x_1,y_1)\kappa_2^{[1]}(x_2,y_2)\kappa_2^{[1]}(x_3,y_3) \\
\kappa_1^{[3]}(\boldsymbol{x},\boldsymbol{y}) &= \kappa_1^{[1]}(x_1,y_1)\kappa_1^{[2]}((x_2,x_3),(y_2,y_3)) \\
&\quad + \beta\kappa_2^{[1]}(x_1,y_1)\kappa_2^{[2]}((x_2,x_3),(y_2,y_3)) \\
&= \kappa_1^{[1]}(x_1,y_1)\kappa_1^{[1]}(x_2,y_2)\kappa_1^{[1]}(x_3,y_3) \\
&\quad + \beta\kappa_1^{[1]}(x_1,y_1)\kappa_2^{[1]}(x_2,y_2)\kappa_2^{[1]}(x_3,y_3) \\
&\quad + \beta\delta\kappa_2^{[1]}(x_1,y_1)\kappa_1^{[1]}(x_2,y_2)\kappa_1^{[1]}(x_3,y_3) \\
&\quad + \beta\phi\kappa_2^{[1]}(x_1,y_1)\kappa_1^{[1]}(x_2,y_2)\kappa_2^{[1]}(x_3,y_3) \\
&\quad + \beta\phi\kappa_2^{[1]}(x_1,y_1)\kappa_2^{[1]}(x_2,y_2)\kappa_1^{[1]}(x_3,y_3) \\
&\quad + \beta\epsilon\kappa_2^{[1]}(x_1,y_1)\kappa_2^{[1]}(x_2,y_2)\kappa_2^{[1]}(x_3,y_3)
\end{aligned}$$

By comparing the coefficients of the terms, we obtain $\beta = \beta\phi$ and $\beta\delta = 0$. In the same way, we can obtain $\beta\delta = 0$ and $\phi(\phi-1) = \epsilon\delta$, by evaluating $\kappa_2^{[3]}$. Finally, we have two cases to investigate: Case 1. $\beta = 0$ and $\phi(\phi-1) = \epsilon\delta$; Case 2. $\beta \neq 0$, $\delta = 0$ and $\phi = 1$. We investigate these cases.

Case 1. If $\epsilon \neq 0$, we replace $\kappa_2^{[*]}$ with $\phi\kappa_1^{[*]} + \epsilon\kappa_2^{[*]}$. Due to Lemma 2, $\mathbf{Q}_2$ is converted into $\mathbf{Q}_2 = \begin{bmatrix} \phi + \delta\epsilon - \phi^2 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$. This yields Type I.

If $\epsilon = 0$, $\phi = 1$ follows from $\phi(\phi-1) = \epsilon\delta$. We convert $\mathbf{Q}_2$ into $\mathbf{Q}_2 = \begin{bmatrix} \delta - 2\delta + \delta & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ by replacing $\kappa_2^{[*]}$ with $\delta\kappa_1^{[*]} + \kappa_2^{[*]}$. This is a special case of Type II.

Case 2. Since $\epsilon = 0$ indicates Type II, we investigate the case of $\epsilon \neq 0$. We can convert $\mathbf{Q}_2$ into $\mathbf{Q}_2 = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$ by replacing $\kappa_2^{[*]}$ with $\epsilon\kappa_2^{[*]}$. This yields Type III.

The assertion on $c_{i:j_1,\ldots,j_k}$ for Type I is a corollary to Theorem 3. On the other hand, the assertions for Type II and Type III can be verified by mathematical induction on k.

Type II. For $\boldsymbol{x}' = \boldsymbol{x}''x'_k$ and $\boldsymbol{y}' = \boldsymbol{y}''y'_k$, we have

$$\begin{aligned}
&\tilde{\kappa}_1^{[k]}(\boldsymbol{x}',\boldsymbol{y}') = \\
&\tilde{\kappa}_1^{[k-1]}(\boldsymbol{x}'',\boldsymbol{y}'')\tilde{\kappa}_1^{[1]}(x'_k,y'_k) + \beta\tilde{\kappa}_2^{[k-1]}(\boldsymbol{x}'',\boldsymbol{y}'')\tilde{\kappa}_2^{[1]}(x'_k,y'_k), \\
&\tilde{\kappa}_2^{[k]}(\boldsymbol{x}',\boldsymbol{y}') = \\
&\tilde{\kappa}_1^{[k-1]}(\boldsymbol{x}'',\boldsymbol{y}'')\tilde{\kappa}_2^{[1]}(x'_k,y'_k) + \tilde{\kappa}_2^{[k-1]}(\boldsymbol{x}'',\boldsymbol{y}'')\tilde{\kappa}_1^{[1]}(x'_k,y'_k).
\end{aligned}$$

If $j_k = 1$, $c_{i:j_1,\ldots,j_k} = c_{i:j_1,\ldots,j_{k-1}}$ and $o_2(j_1,\ldots,j_k) = o_2(j_1,\ldots,j_{k-1})$ hold, and, if $j_k = 2$, $c_{1:j_1,\ldots,j_k} = \beta c_{2:j_1,\ldots,j_{k-1}}$, $c_{2:j_1,\ldots,j_k} = c_{1:j_1,\ldots,j_{k-1}}$ and $o_2(j_1,\ldots,j_k) = o_2(j_1,\ldots,j_{k-1}) + 1$ hold. Hence, the assertion follows from the hypothesis of induction.

Type III. In the same way, we have

$$\begin{aligned}
&\tilde{\kappa}_1^{[k]}(\boldsymbol{x}',\boldsymbol{y}') = \\
&\tilde{\kappa}_1^{[k-1]}(\boldsymbol{x}'',\boldsymbol{y}'')\tilde{\kappa}_1^{[1]}(x'_k,y'_k) + \beta\tilde{\kappa}_2^{[k-1]}(\boldsymbol{x}'',\boldsymbol{y}'')\tilde{\kappa}_2^{[1]}(x'_k,y'_k), \\
&\tilde{\kappa}_2^{[k]}(\boldsymbol{x}',\boldsymbol{y}') = \tilde{\kappa}_1^{[k-1]}(\boldsymbol{x}'',\boldsymbol{y}'')\tilde{\kappa}_2^{[1]}(x'_k,y'_k) \\
&\quad + \tilde{\kappa}_2^{[k-1]}(\boldsymbol{x}'',\boldsymbol{y}'')\tilde{\kappa}_1^{[1]}(x'_k,y'_k) + \tilde{\kappa}_2^{[k-1]}(\boldsymbol{x}'',\boldsymbol{y}'')\tilde{\kappa}_2^{[1]}(x'_k,y'_k).
\end{aligned}$$

If $j_k = 1$, $c_{i:j_1,\ldots,j_k} = c_{i:j_1,\ldots,j_{k-1}}$ and $o_2(j_1,\ldots,j_k) = o_2(j_1,\ldots,j_{k-1})$ hold, and, if $j_k = 2$, $c_{1:j_1,\ldots,j_k} = \beta c_{2:j_1,\ldots,j_{k-1}}$, $c_{2:j_1,\ldots,j_k} = c_{1:j_1,\ldots,j_{k-1}} + c_{2:j_1,\ldots,j_{k-1}}$ and $o_2(j_1,\ldots,j_k) = o_2(j_1,\ldots,j_{k-1}) + 1$ hold. Hence, the assertion follows from the hypothesis of induction. In particular, if $j_k = 2$, we have the following by the recurrence formula with respect to $F_n(x)$.

$$\begin{aligned}
c_{2:j_1,\ldots,j_k} &= \beta F_{o_2(j_1,\ldots,j_k)-3}(\beta) + F_{o_2(j_1,\ldots,j_k)-2}(\beta) \\
&= F_{o_2(j_1,\ldots,j_k)-1}(\beta)
\end{aligned}$$

4. How to tune kernels

Theorem 4 asserts that a partitionable kernel $\kappa^{[*]}$ with hidden degree two is determined by the following parameters: Type, a pair of character-wise kernels $(\tilde{\kappa}_1^{[1]}, \tilde{\kappa}_2^{[1]})$, β to define $\tilde{\kappa}_1^{[*]}$ and $\tilde{\kappa}_2^{[*]}$, if Type II or Type III is chosen, and t of $\kappa^{[*]} = t\tilde{\kappa}_1^{[*]} + (1-t)\tilde{\kappa}_1^{[*]}$.

To tune kernels, we first select $\tilde{\kappa}_1^{[1]}$ and $\tilde{\kappa}_2^{[1]}$, and then to search optimal t and β.

4.1. Selecting $\tilde{\kappa}_1^{[1]}$ and $\tilde{\kappa}_2^{[1]}$

The character-wise (label) kernels that are used in the literature are limited: Constant functions λ, Kronecker's delta function $\delta_{x',y'}$ and their multiplication

$\lambda\delta_{x',y'}$ are the most common, and Gaussian function $e^{-\frac{|x'-y'|^2}{2\sigma^2}}$ can be used, when labels are real numbers. Since our target kernel is a linear combination of $\tilde{\kappa}_1^{[*]}$ and $\tilde{\kappa}_2^{[*]}$, it is natural to select $\tilde{\kappa}_1^{[1]}$ and $\tilde{\kappa}_2^{[1]}$ from these common candidates.

For example, let $M_{x,y}$ be the one defined for the all-subsequence kernel (Section 1), and hence, $(\boldsymbol{x}', \boldsymbol{y}') \in M_{x,y}$ is a pair of substrings such that $|\boldsymbol{x}'| = |\boldsymbol{y}'| = k$. If we let $\tilde{\kappa}_2^{[1]}(x', y') = \delta_{x',y'}$ in Example 3, $\tilde{\kappa}_2^{[k]}(\boldsymbol{x}', \boldsymbol{y}') = \sum_{i=1}^{k} \delta_{x'_i,y'_i}$ gives the number of the identical character pairs between $\boldsymbol{x}'$ and $\boldsymbol{y}'$. Hence, the resulting mapping kernel counts the pairs of substring pairs of the same length using the number of identical character pairs as weights, whereas the all-subsequence kernel uses λ^k as weights. In the same way, if we let $\tilde{\kappa}_1^{[1]}(x', y') = \tilde{\kappa}_2^{[1]}(x', y') = \lambda\delta_{x',y'}$ in Example 4, $\tilde{\kappa}_2^{[k]}(\boldsymbol{x}', \boldsymbol{y}') = \lambda^k \cdot \prod_{i=1}^{k} \delta_{x'_i,y'_i} \cdot \left(\sum_{i=1}^{k} 1\right)$ holds, and hence, we have $\tilde{\kappa}_2^{[k]}(\boldsymbol{x}', \boldsymbol{y}') = \lambda^k \cdot k$, if $\boldsymbol{x}' = \boldsymbol{y}'$, and $\tilde{\kappa}_2^{[k]}(\boldsymbol{x}', \boldsymbol{y}') = 0$, otherwise.

4.2. Searching optimal t and β

After determining $\tilde{\kappa}_1$ and $\tilde{\kappa}_2$, the combination of cross validation and grid search will be effective to search optimal values for adjustable parameters (Chang & Lin, 2001).

For this purpose, it is desirable to calculate Gram matrices of $\tilde{\kappa}_1^{[1]}$ and $\tilde{\kappa}_2^{[1]}$ collectively, so that the elements are polynomials in β rather than simple numeric values. This significantly improves the efficiency of the search, since we can avoid recalculating kernel values with expensive dynamic-programming-based algorithms, whenever the value of β is changed.

To perform cross validation, we assume to use C-SVM in this paper. Thus, the adjustable parameters that we can change include t, β and the regularization parameter C of C-SVM. In addition, if we use adjustable character-wise kernels for $\tilde{\kappa}_i^{[1]}$, such as $\lambda\delta_{x',y'}$, the associated parameters can be included.

When the number of the adjustable parameters to examine is d, we perform the d-dimensional grid search. The most significant advantage of using the grid search consists in the fact that we can take advantage of parallel computation to accelerate the search.

5. Empirical result

It is important to answer when and how mapping kernels with the novel partitionable kernels that we study in this paper are effective. Although large-scale experiments are necessary to answer this question, we only ran a preliminary experiment, using three datasets retrieved from the KEGG/GLYCAN database (Hashimoto et al., 2006), which contain glycan structures, represented as rooted ordered trees and annotated relating to colon cancer, cystic fibrosis and leukemia cells. Table 3 describes these datasets. The size of a tree is the number of the vertices, the height is the length of the longest downward path, and the degree is the maximum number of child vertices.

We use five kernels for trees, referred to as COUNT, WEIGHTED, TYPE I, II and III. COUNT counts the identical pairs of substructures (forests) ($K_{\text{CNT}} = \sum_{(\boldsymbol{x}',\boldsymbol{y}')\in M_{x,y}} \prod_{i=1}^{|\boldsymbol{x}'|} \delta_{x'_i,y'_i}$), while WEIGHTED counts the same substructure pairs with the decay factor $\frac{1}{2}$ (Kuboyama et al., 2006) ($K_{\text{WGHT}} = \sum_{(\boldsymbol{x}',\boldsymbol{y}')\in M_{x,y}} \prod_{i=1}^{|\boldsymbol{x}'|} \frac{1}{2}\delta_{x'_i,y'_i}$). $M_{x,y}$ denotes the set of isomorphic substructure pairs. On the other hand, TYPE I ($tK_{\text{TI},1} + (1-t)K_{\text{TI},2}$), TYPE II ($tK_{\text{TII},1} + (1-t)K_{\text{TII},2}$) and TYPE III ($tK_{\text{TIII},1} + (1-t)K_{\text{TIII},2}$) are mapping kernels defined with partitionable kernels of hidden degree two, whose specification is given in Table 4. To make the experiment simple, we assume $\beta = 0$, instead of searching an optimal value for β.

The steps of the experiment are as follows: We randomly generate five pairs of training and test data subsets from each dataset, train C-SVM with the training subsets, and then measure AUC (Area Under Curve) of ROC (Receiver Operation Characteristic) curve with the test subsets. During training C-SVM, we optimize the regularization parameter C for C-SVM, and the parameter t for the TYPE I, II and III kernels, by means of cross validation and grid search.

Table 5 shows the results. We should note that TYPE II showed good performance for Cystic-Fibrosis, while the other kernels could show only poor performance. Cystic-Fibrosis is characterized by the highest average degree, and hence, contains trees with more complicated structures than the other two datasets.

The runtime scores with MacBook Air show that TYPE I, II and III were 5 times slower than the others on average: TYPE I, II and III commonly took around 20, 30 and 600 seconds to generate the Gram matrices for Colon-Cancer, Cystic-Fibrosis and Leukemia, respectively.

6. Future work

Characterizing the class of partitionable kernels with hidden degree three still interests us. Also, intensive experiments using a variety of datasets will justify the

Table 3. Features of the datasets used

Dataset	# of Examples	Average Size	Average Height	Average Degree
Colon-Cancer	134	8.4	5.6	1.46
Cystic-Fibrosis	160	8.3	5.0	1.52
Leukemia	442	13.5	7.4	1.42

Table 4. Definitions of the partitionable kernels used

Kernels	$\tilde{\kappa}_1^{[1]}$	$\tilde{\kappa}_2^{[1]}$	β	Details ($M_{x,y}$ is the set of isomorphic subforest pairs)
Type I	$\delta_{x',y'}$	1	N/A	$K_{\mathrm{TI},1}(x,y) = K_{\mathrm{CNT}}(x,y), K_{\mathrm{TI},2}(x,y) = \sum_{(\boldsymbol{x}',\boldsymbol{y}')\in M_{x,y}} 1$
Type II	$\delta_{x',y'}$	$\delta_{x',y'}$	0.0	$K_{\mathrm{TII},1}(x,y) = K_{\mathrm{CNT}}(x,y), K_{\mathrm{TII},2}(x,y) = \sum_{(\boldsymbol{x}',\boldsymbol{y}')\in M_{x,y}} \lvert\boldsymbol{x}'\rvert \cdot \prod_{i=1}^{\lvert\boldsymbol{x}'\rvert} \delta_{x'_i,y'_i}$
Type III	$\delta_{x',y'}$	$\delta_{x',y'}$	0.0	$K_{\mathrm{TIII},1}(x,y) = K_{\mathrm{CNT}}(x,y), K_{\mathrm{TIII},2}(x,y) = \sum_{(\boldsymbol{x}',\boldsymbol{y}')\in M_{x,y}} \left(2^{\lvert\boldsymbol{x}'\rvert}-1\right) \cdot \prod_{i=1}^{\lvert\boldsymbol{x}'\rvert} \delta_{x'_i,y'_i}$

Table 5. Average AUC-ROC

Dataset	Count	Weighted	Type I	Type II	Type III
Colon-Cancer	**0.941**	0.940	**0.941**	0.903	0.864
Cystic-Fibrosis	0.735	0.745	0.738	**0.796**	0.756
Leukemia	**0.937**	**0.937**	**0.937**	0.892	0.856

effectiveness of partitionable kernels.

Appendix A. A proof to Lemma 3

Since $\left(\kappa_1^{[0]}\kappa_2^{[1]} - \kappa_2^{[0]}\kappa_1^{[1]}\right)(\mathbf{Q}_i[1,2] - \mathbf{Q}_i[2,1]) = 0$ follows from $\kappa_i^{[1]} = \left(\kappa_1^{[1]},\kappa_2^{[1]}\right)\mathbf{Q}_i\left(\kappa_1^{[0]},\kappa_2^{[0]}\right)^{\mathsf{T}} = \left(\kappa_1^{[0]},\kappa_2^{[0]}\right)\mathbf{Q}_i\left(\kappa_1^{[1]},\kappa_2^{[1]}\right)^{\mathsf{T}}$, it suffices to prove the assertion assuming $\kappa_1^{[0]}\kappa_2^{[1]} - \kappa_2^{[0]}\kappa_1^{[1]} = 0$ and $\kappa^{[*]}0_1 \neq 0$. We claim $\kappa_1^{[0]}\kappa_2^{[k]} - \kappa_2^{[0]}\kappa_1^{[k]} = 0$, and prove it by mathematical induction on k: If this claim is true, $\kappa_1^{[*]}$ and $\kappa_2^{[*]}$ are linearly dependent, and we can let $\mathbf{Q}_i$ diagonal. By the hypothesis of induction we have

$$\begin{aligned}\kappa_i^{[k]} &= \left(\kappa_1^{[1]},\kappa_2^{[1]}\right)\mathbf{Q}_i\left(\kappa_1^{[k-1]},\kappa_2^{[k-1]}\right)^{\mathsf{T}} \\ &= \frac{\kappa_1^{[1]}}{\kappa_1^{[0]}}\cdot\left(\kappa_1^{[0]},\kappa_2^{[0]}\right)\mathbf{Q}_i\left(\kappa_0^{[1]},\kappa_2^{[0]}\right)^{\mathsf{T}}\cdot\frac{\kappa_1^{[k-1]}}{\kappa_1^{[0]}}\end{aligned}$$

The claim follows, as $(\kappa_1^{[0]},\kappa_2^{[0]})\mathbf{Q}_i(\kappa_1^{[0]},\kappa_2^{[0]})^{\mathsf{T}} = \kappa_i^{[0]}$.

Appendix B. A proof to Lemma 4

Let $(\kappa_1^{[*]} = \kappa^{[*]},\kappa_2^{[*]})$ be a partitionable kernel family with $\mathbf{Q}_1 = \begin{bmatrix}\alpha,\gamma\\ \gamma,\beta\end{bmatrix}$ and $\mathbf{Q}_2 = \begin{bmatrix}\delta,\phi\\ \phi,\epsilon\end{bmatrix}$.

If $\beta \neq 0$, by Lemma 2, we see that replacing $\kappa_2^{[*]}$ with $\frac{\gamma}{\beta}\kappa_1^{[*]} + \kappa_2^{[*]}$ transforms $\mathbf{Q}_1$ into

$$\mathbf{Q}_1 = \mathbf{Q}_1' = \begin{bmatrix}\alpha - 2\frac{\gamma}{\beta}\gamma + \frac{\gamma^2}{\beta^2}\beta & \gamma - \frac{\gamma}{\beta}\beta\\ \gamma - \frac{\gamma}{\beta}\beta & \beta\end{bmatrix} = \begin{bmatrix}\frac{\det\mathbf{Q}_1}{\beta} & 0\\ 0 & \beta\end{bmatrix}.$$

If $\beta = 0$, replacing $\kappa_1^{[*]}$ with $\kappa_1^{[*]} + x\kappa_2^{[*]}$ transforms $\mathbf{Q}_1$ into $\mathbf{Q}_1 = \mathbf{Q}_1' + x\mathbf{Q}_2'$, where

$$\mathbf{Q}_1' = \begin{bmatrix}\alpha & \gamma - x\alpha\\ \gamma - x\alpha & -2x\gamma + x^2\alpha\end{bmatrix} \quad \text{and}$$
$$\mathbf{Q}_2' = \begin{bmatrix}\delta & \phi - x\delta\\ \phi - x\delta & \epsilon - 2x\phi + x^2\delta\end{bmatrix}.$$

In particular, we have $\mathbf{Q}_1[2,2] = \delta x^3 + (\alpha - 2\phi)x^2 + (\epsilon - 2\gamma)x$. If $\delta^2 + (\alpha - 2\phi)^2 + (\epsilon - 2\gamma)^2 \neq 0$, we can choose x so that $\mathbf{Q}_1[2,2] \neq 0$, and we can reduce the current case to the case of $\beta \neq 0$. Otherwise, by replacing $\kappa_1^{[*]}$ with $\kappa_1^{[*]} + \frac{\gamma}{\phi}\kappa_2^{[*]}$ (if $\phi = 0$, there is nothing to prove), we obtain

$$\mathbf{Q}_1' = \begin{bmatrix}\alpha & \gamma - \frac{\alpha\gamma}{\phi}\\ \gamma - \frac{\alpha\gamma}{\phi} & -2\frac{\gamma^2}{\phi} + \frac{\alpha\gamma^2}{\phi^2}\end{bmatrix} = \begin{bmatrix}\alpha & -\gamma\\ -\gamma & 0\end{bmatrix} \quad \text{and}$$
$$\mathbf{Q}_2' = \begin{bmatrix}0 & \phi\\ \phi & \epsilon - 2\frac{\gamma\phi}{\phi}\end{bmatrix} = \begin{bmatrix}0 & \phi\\ \phi & 0\end{bmatrix}.$$

Hence, $\mathbf{Q}_1$ is modified to

$$\mathbf{Q}_1' + \frac{\gamma}{\phi}\mathbf{Q}_2' = \begin{bmatrix}\alpha & -\gamma + \frac{\phi\gamma}{\phi}\\ -\gamma + \frac{\phi\gamma}{\phi} & 0\end{bmatrix} = \begin{bmatrix}\alpha & 0\\ 0 & 0\end{bmatrix}.$$

References

Berg, C., Christensen, J. P. R., and Ressel, R. *Harmonic Analysis on semigroups. Theory of positive definite and related functions*. Springer, 1984.

Chang, C.-C. and Lin, C.-J. Libsvm: a library for support vector machines. http://www.csie.ntu.edu.tw/˜cjlin/libsvm/, 2001.

Collins, M. and Duffy, N. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001]*, pp. 625–632. MIT Press, 2001.

Cortes, C., Haffner, P., and Mohri, M. Rational kernels: Theory and algorithm. *Journal of Machine Learning Research*, 1:1–50, 2004.

Hashimoto, K., Goto, S., Kawano, S., Aoki-Kinoshita, K. F., and Ueda, N. Kegg as a glycome informatics resource. *Glycobiology*, 16:63R – 70R, 2006.

Haussler, D. Convolution kernels on discrete structures. UCSC-CRL 99-10, Dept. of Computer Science, University of California at Santa Cruz, 1999.

Kashima, H. and Koyanagi, T. Kernels for semi-structured data. In *the 9th International Conference on Machine Learning (ICML 2002)*, pp. 291–298, 2002.

Kuboyama, T., Shin, K., and Kashima, H. Flexible tree kernels based on counting the number of tree mappings. In *Proc. of Machine Learning with Graphs*, 2006.

Leslie, C. S., Eskin, E., and Stafford Noble, W. The spectrum kernel: A string kernel for SVM protein classification. In *Pacific Symposium on Biocomputing*, pp. 566–575, 2002.

Lodhi, H., Shawe-Taylor, J, Cristianini, N, and H., Watkins C. J. C. Text classification using string kernels. *Advances in Neural Information Processing Systems (NIPS 2000)*, 13, 2001.

Lu, S. Y. A tree-to-tree distance and its application to cluster analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 1:219—224, 1979.

Shin, K. and Kuboyama, T. A generalization of Haussler's convolution kernel – Mapping kernel. In *ICML 2008*, 2008.

Shin, K. and Kuboyama, T. A generalization of Haussler's convolution kernel – Mapping kernel and its application to tree kernels. *J. Comput. Sci. Technol*, 25(5)::1040–1054, 2010.

Shin, K. Partitionable kernels for mapping kernels. In *ICDM 2011*, pp. 645–654, 2011.

Taï, K. C. The tree-to-tree correction problem. *JACM*, 26(3):422–433, July 1979.

Wagner, R.A. and Fischer, M.J. The string-to-string correction problem. *JACM*, 21(1):168–173, 1974.

Zhang, K. Algorithms for the constrained editing distance between ordered labeled trees and related problems. *PR*, 28(3):463–474, March 1995.