Kernels and Dynamical Systems ⋆

Alexander J. Smola [a,b], René Vidal [c], S.V.N. Vishy Vishwanathan [b]

[a] *Australian National University, RSISE, Machine Learning Group, Canberra, 0200 ACT, Australia*

[b] *National ICT Australia, Machine Learning Program, Canberra, 0200 ACT, Australia*

[c] *Center for Imaging Science, Johns Hopkins University, Baltimore, MD 21218, USA*

Abstract

In this paper we study Hilbert space embeddings *of* dynamical systems and embeddings generated *via* dynamical systems. This is achieved by following the behavioral framework invented by Willems, namely by comparing trajectories of states. By a simple application of the Binet-Cauchy theorem we are able to provide a unifying framework for all kernels on dynamical systems currently used in machine learning. As important special cases we recover and extend graph kernels, we provide novel kernels on matrices and on Markov Models. Secondly we show how this relates to kernels based on the cepstrum coefficients of a linear time-invariant system. This leads to filter-invariant kernels.

Besides their theoretical appeal, we show that these kernels can be used efficiently in the comparison of video sequences via their embedding in linear time-invariant systems. The advantage of our setting over the one by Doretto and coworkers is that we are able to take the initial conditions of such dynamical systems into account. Experimental evidence shows superior performance of our kernel.

Key words: Support Vectors, kernels, Hilbert Space, dynamical systems, ARMA model, Sylvester equation, machine learning, estimation

1 Introduction

The past few years have witnessed an increasing interest in the application of system-theoretic techniques to the modeling of visual dynamical processes. For instance, Doretto *et al.* [15,41] have proposed to model the appearance of dynamic textures, such as video sequences of water, foliage, hair, etc. as the output of a Gaussian linear dynamical model. Similarly, Saisan *et al.* [35] have used linear dynamical models in the modeling of human gaits such as walking, running, jumping, etc.

However, when modeling visual dynamical processes one may be interested not only in obtaining a *model* for the process (identification problem), but also in determining whether two video sequences correspond to the same process (classification problem) or which process is being observed in a given video sequence (recognition problem). Since the space of models for dynamical processes typically has a non Euclidean structure, [1] the above problems have naturally raised the issue of how to do estimation, classification and recognition on such spaces.

The study of classification and recognition problems has been the mainstream areas of research in machine learning for the past decades. Among the various methods for nonparametric estimation that that have been developed, *kernel methods* have become one of the mainstays as witnessed by a large number of books [43,44,46,23,26,38,10]. However, not much of the existing literature has addressed the design of kernels in the context of dynamical systems. To the best of our knowledge, the metric for ARMA models based on comparing their cepstrum coefficients [31] is one of the first papers to address this problem. De Cook and De Moor [11] extended this concept to ARMA models in state-space representation by considering the subspace

⋆ Parts of this paper were presented at SYSID 2003. Corresponding author A. J. Smola, Tel. +61-410-457686, Fax +61-2-6125-8651.

Email addresses: Alex.Smola@anu.edu.au (Alexander J. Smola), rvidal@cis.jhu.edu (René Vidal), vishy@axiom.anu.edu.au (S.V.N. Vishy Vishwanathan).

[1] For instance, the model parameters of a linear dynamical model are determined only up to a change of basis, hence the space of models has the structure of a Stiefel manifold.

angles between the observability subspaces. Recently, Wolf and Shashua [51] demonstrated how subspace angles can be efficiently computed by using kernel methods.

1.1 Paper contributions

In this paper we attempt to bridge the gap between nonparametric estimation methods and dynamical systems. More specifically, we build on previous work using explicit embedding constructions and regularization theory [39,51,40] to propose a family of kernels explicitly designed for dynamical systems. These comparisons are carried out by comparing the trajectories arising from dynamical systems.

We show that the two strategies which have been used in the past are special cases of one unified framework. It works by computing traces over compound matrices. More specifically, we show that one popular approach is based on the computation of traces, whereas the other relies on the computation of determinants over matrix products.

- The first family is based on the idea of extracting coefficients of a linear time-invariant (LTI) system (ARMA or linear differential equation) and defining a scalar product in terms of these quantities. This construction will allow us to state important properties regarding invariance of the systems with respect to filtering directly by using results from Martin [31]. The results of [11,15] allow us to connect these kernels with the recent work of Wolf and Shashua of kernels via subspace angles [51].

  The downside of the above approach is that it that the resulting kernel is insensitive to initial conditions. For instance, not every initial configuration of the joints of the human body is appropriate for modeling human gaits.
- The second family is based directly on the time-evolution properties of dynamical systems [39,51,40]. We show that the diffusion kernels of [28], the graph kernels of [18], and similarity measures between graphs [8] can be found as special cases of our framework. It has the option to take the initial conditions of the systems into account explicitly.

As disparate as both methods seem, we show that both arise from a common family of kernels which can be defined in terms of traces over compound matrices of order q. The Binet-Cauchy theorem [1] is then invoked to show that both families of kernels arise simply by suitable preprocessing of the trajectory matrices and by a corresponding choice of the order q.

Finally, we show how the proposed kernels can be used to classify dynamic textures via their embedding in LTI systems. Experimental evidence shows superior performance of our kernel.

1.2 Paper outline

After a brief overview of kernel methods in Section 2 and an outline of the basic concepts involved in computing kernels on dynamical systems in Section 3 we derive explicit formulae for so-called trajectory kernels in Sections 4 and 4.3. Special cases are discussed in Section 6. Experiments and a discussion conclude in Sections 8 and 9.

2 Kernel Methods

In this section we give a brief overview of binary classification with kernels. For extensions such as multi-class settings, the ν-parameterization, loss functions, etc. we refer the reader to [38,23,44] and the references therein.

Assume we have m observations (x_i, y_i) drawn iid (independently and identically distributed) from a distribution over $\mathcal{X} \times \mathcal{Y}$, where in our case $\mathcal{X}$ is the set of images and $\mathcal{Y} = \{\pm 1\}$ are the labels, that can represent for instance water versus foliage in dynamic textures. It is our goal to find a function $f : \mathcal{X} \to \mathcal{Y}$ which classifies observations $x \in \mathcal{X}$ into classes $+1$ and -1.

For the moment assume that $\mathcal{X}$ is a dot product space (we will relax this condition subsequently via the introduction of kernels) and f be given by

$$f(x) = \operatorname{sgn}(\langle w, x \rangle + b). \tag{1}$$

Here the sets $\{x | f(x) \geq 0\}$ and $\{x | f(x) \leq 0\}$ denote half-spaces and the separating hyperplane between them is given by $H(w, b) := \{x | \langle w, x \rangle + b = 0\}$. In the case of linearly separable datasets, that is, if we can find (w, b) such that all (x_i, y_i) pairs satisfy $y_i f(x_i) = 1$, the *optimal separating hyperplane* is given by the (w, b) pair for which all x_i have maximum distance from $H(w, b)$. In other words, it is the hyperplane which separates the two sets with the largest possible margin.

2.1 Linearly Nonseparable Problems

Unfortunately, such separation is not always possible and we need to allow for slack in the separation of the two sets. Without going into details (which can be found in [38]) this leads to the optimization problem:

$$\begin{aligned} \underset{w,b,\xi}{\text{minimize}} \quad & \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{m} \xi_i \\ \text{subject to} \quad & y_i \left(\langle w, x_i \rangle + b\right) \geq 1 - \xi_i \quad \forall 1 \leq i \leq m \\ & \xi_i \geq 0 \end{aligned} \tag{2}$$

Here the constraint $y_i \left(\langle w, x_i \rangle + b\right) \geq 1$ ensures that each (x_i, y_i) pair is classified correctly. The slack variable ξ_i relaxes this condition at penalty $C\xi_i$. Finally,

minimization of $\|w\|^2$ ensures maximization of the margin by seeking the smallest $\|w\|$ for which the condition $y_i\left(\langle w, x_i\rangle + b\right) \geq 1$ is still satisfied.

Computing the dual of (2) leads to a quadratic optimization problem

$$\begin{array}{ll} \underset{\alpha}{\text{minimize}} & \frac{1}{2}\alpha^\top K\alpha - \vec{1}^\top\alpha \\ \text{subject to} & \vec{y}^\top\alpha = 0 \text{ and } \alpha_i \in [0, C] \end{array} \tag{3}$$

where $K_{ij} = y_i y_j \langle x_i, x_j\rangle$ and $\vec{1}$ denotes the vector of ones. The primal variable w can be found as a linear combination of the Lagrange multipliers α via $w = \sum_i y_i \alpha_i x_i$.

2.2 Kernel Expansion

To obtain a nonlinear classifier, one simply replaces the observations x_i by $\Phi(x_i)$. That is, we extract *features* $\Phi(x_i)$ from x_i and compute a linear classifier in terms of the features. Note that there is no need to compute $\Phi(x_i)$ explicitly, since Φ only appears in terms of dot products:

- $\langle \Phi(x), w\rangle$ can be computed by exploiting the linearity of the scalar product, which leads to $\sum_i \alpha_i y_i \langle \Phi(x), \Phi(x_i)\rangle$.
- Likewise $\|w\|^2$ can be expanded in terms of a linear combination scalar products by exploiting the linearity of scalar products twice to obtain $\sum_{i,j} \alpha_i \alpha_j y_i y_j \langle \Phi(x_i), \Phi(x_j)\rangle$.

Consequently there is no need to compute $\Phi(x)$ explicitly. Furthermore, if we define

$$k(x, x') := \langle \Phi(x), \Phi(x')\rangle, \tag{4}$$

we may use $k(x, x')$ wherever $\langle x, x'\rangle$ occurs. We only need to modify (2) marginally to suit our needs: replace the old definition of K with

$$K_{ij} = y_i y_j k(x_i, x_j). \tag{5}$$

The resulting hyperplane (now in feature space) is

$$f(x) = \langle \Phi(x), w\rangle + b = \sum_{i=1}^{m} \alpha_i y_i k(x_i, x). \tag{6}$$

This is also known as the *kernel expansion* and it stems from a deep fact about Reproducing Kernel Hilbert Spaces. See [38] for details on the Representer Theorem.

Finally, it is worthwhile mentioning that the particular setting of the optimization problem (2) typically results in many coefficients α_i being 0. This considerably speeds up the estimation step, since these terms can be dropped from (6). The nonzero terms are commonly referred to as Support Vectors, since they support the optimal separating hyperplane. Efficient codes exist for the solution of (2) [25,16,45,27].

2.3 Kernels on Euclidean Spaces

There are many practical situations in which one is given a similarity measure $k(x, x')$ which is constructed without necessarily having an embedding $\Phi(x)$ is mind. In such cases one may ask if the given k is indeed a kernel, *i.e.,* whether it can be seen as a scalar product in some space as in (4). The following theorem provides us with a tool to analyze such k.

Theorem 1 (Mercer [32,29]) *Suppose $k \in L_\infty(\mathcal{X}^2)$ (i.e. k is square integrable on $\mathcal{X}^2$) is a symmetric real-valued function such that the integral operator*

$$\begin{aligned} T_k &: L_2(\mathcal{X}) \to L_2(\mathcal{X}) \\ (T_k f)(x) &:= \int_{\mathcal{X}} k(x, x') f(x')\, d\mu(x') \end{aligned} \tag{7}$$

is positive definite*; that is, for all $f \in L_2(\mathcal{X})$, we have*

$$\int_{\mathcal{X}^2} k(x, x') f(x) f(x')\, d\mu(x) d\mu(x') \geq 0. \tag{8}$$

Let $\psi_j \in L_2(\mathcal{X})$ be the normalized orthogonal eigenfunctions of T_k associated with the eigenvalues $\lambda_j > 0$, sorted in non-increasing order. Then

(1) the series of eigenvalues is absolutely convergent and the eigenfunctions are uniformly bounded almost everywhere.
(2) $k(x, x') = \sum_{j=1}^{N_\mathcal{H}} \lambda_j \psi_j(x) \psi_j(x')$ holds for almost all (x, x'). Either $N_\mathcal{H} \in \mathbb{N}$, or $N_\mathcal{H} = \infty$; in the latter case, the series converges absolutely and uniformly for almost all (x, x').

This means that for a kernel satisfying Mercer's condition, we can find the feature space associated with k by studying the eigensystem of the integral operator T_k. In the following we will call such kernels *admissible.*

Two of the most popular kernels include the Gaussian kernel (9) introduced in [2] and the polynomial kernel (10) proposed in [33]:

$$k(x, x') = \exp\left(\lambda \|x - x'\|^2\right) \text{ for } \lambda > 0 \tag{9}$$

$$k(x, x') = \left(\langle x, x'\rangle + r\right)^d \text{ where } r \geq 0, d \in \mathbb{N}. \tag{10}$$

Many further kernels exist, e.g., kernels for sets, multisets, strings, or probability distributions. One of the

attractive properties of kernels is that they can be composed to form more complex objects in a simple fashion. For instance

$$k(x,x') = k_1(x,x') + k_2(x,x')$$
$$k(x,x') = k_1(x,x')k_2(x,x')$$
$$k(x,x') = \kappa(\langle\Phi(x),\Phi(x')\rangle) \text{ if } \kappa(\langle x,x'\rangle) \text{ is a kernel}$$
$$k(x,x') = \kappa(\|\Phi(x)-\Phi(x')\|) \text{ if } \kappa(\|x-x'\|) \text{ is a kernel}$$

are all kernels. Similarly a convex combination of kernels is also a kernel. All these properties allow us to build structured kernels in an efficient fashion.

2.4 Metric Methods

Metric methods rely on the distances between objects for estimation. For instance, a Nearest Neighbor classifier will classify an unseen object as belonging to the class of its nearest neighbor. Variations on this theme, such as using k-nearest neighbors for classification via a majority vote have been studied in great detail and rates of convergence exist. See for instance [14] and the references therein. For the computation of distances in feature space one simply uses

$$\begin{aligned} d(x,x') &:= \|\phi(x)-\phi(x')\| \\ &= \sqrt{k(x,x)+k(x',x')-2k(x,x')}. \end{aligned} \tag{11}$$

Likewise distance-based clustering operations such as k-means [30] can easily be extended to kernels [37]: simply replace all distance computations by kernel functions.

3 Kernels on Dynamical Systems

3.1 Dynamical Systems

We begin with some definitions. For the remainder of the paper we assume that the state space $\mathcal{X}$, with $x \in \mathcal{X}$, is a Hilbert space. This is not a major restriction, since e.g., any countable set S can be made into a Hilbert space by mapping it into ℓ_2^S. Similar choices can be made for non-countable sets. [2]

Moreover we assume that the time evolution of x is determined by

$$x_{\mathbf{A}}(t) := \mathbf{A}(t)x \text{ for } \mathbf{A} \in \mathcal{A}, \tag{12}$$

where $t \in \mathcal{T}$ is the time of the measurement and $\mathcal{A}$ denotes the set of time evolution operators. We will choose $\mathcal{T} = \mathbb{N}_0$ or $\mathcal{T} = \mathbb{R}_0^+$, depending on whether we wish to deal with discrete-time or continuous-time systems.

Note that $\mathbf{A}$ may be a *nonlinear* operator and that both x and $\mathbf{A}$ may be random variables rather than deterministic variables. In those cases, we assume that both x and $\mathbf{A}$ are endowed with suitable probability measures. For instance, we may want to consider initial conditions corrupted by additional noise or LTI systems with additive noise.

Also note that $x_{\mathbf{A}}(t)$ need not be the only variable involved in the time evolution process. For instance, for partially observable models, we may only see $y_{\mathbf{A}}(t)$ which depends on the evolution of a hidden state $x_{\mathbf{A}}(t)$. These cases will be discussed in more detail in Section 4.

3.2 Trajectories

When dealing with dynamical systems, one may compare their similarities by checking whether they satisfy similar functional dependencies. For instance, for LTI systems one may want to compare the the transfer functions, the system matrices, the poles and/or zeros, etc. This is indeed useful in determining when systems are similar whenever suitable parameterizations exist. However, it is not difficult to find rather different functional dependencies, which, nonetheless, behave almost identically, e.g., as long as the domain of initial conditions is sufficiently restricted. For instance, consider the maps

$$x \leftarrow a(x) = |x|^p \text{ and } x \leftarrow b(x) = \min(|x|^p, |x|) \tag{13}$$

for $p > 1$. While a and b clearly differ, the two systems behave identically for all initial conditions satisfying $|x| \leq 1$. This example may seem contrived and it is quite obvious from (13) that identical behavior will occur in this case, yet for more complex maps and higher dimensional spaces such statements are not quite as easily formulated.

One way to amend this situation is to compare *trajectories* of dynamical systems and derive measures of similarity from them. The advantage of this approach is that it is independent of the parameterization of the system. This approach is in spirit similar to the behavioral framework of [48,49,50], which identifies systems by identifying trajectories. However, it has the disadvantage that one needs efficient mechanisms to compute the trajectories, which may or may not always be available. We will show later that in the case of LTI models such computations can be performed efficiently by solving Sylvester-like equations.

In keeping with the spirit of the behavioral framework we will consider the pairs $(x, \mathbf{A})$ only in terms of the trajectories which they generate. In other words, we only

[2] Should further generality be required, one may use algebraic semi-ring methods proposed by [12], albeit at a significant technical complication. This leads to rational series and functions on them.

study the result of the map

$$\mathrm{Traj}\,\mathcal{X}\times\mathcal{A}\to\mathcal{X}^{\mathcal{T}} \text{ where } \mathrm{Traj}(x,\mathbf{A})(t)=x_{\mathbf{A}}(t). \quad (14)$$

This means that we can identify $\mathrm{Traj}(x,\mathbf{A})$ with a linear operator mapping from $\mathcal{X}$ to $\mathbb{R}^{\mathcal{T}}$, simply by computing scalar products between its argument and $x_{\mathbf{A}}(t)$ for all $t\in\mathcal{T}$.

3.3 Kernels

In order to define kernels on $\mathcal{X}^{\mathcal{T}}$ we need to introduce the notion of compound matrices. Their definition can be extended to operators as long as we can reduce the problem to finite dimensional objects by projections. The motivation for introducing compound matrices is that they allow us to use the Binet-Cauchy theorem for the definition of kernels.

Definition 2 (Compound Matrix) *Let $A\in\mathbb{R}^{n\times k}$. The compound matrix $C_q(A)$ is the matrix of minors of order q of A. That is, the (i,j)-th element of $C_q(A)$ is the determinant of a submatrix of A formed by choosing a subset of rows indexed by i and a subset of columns indexed by j, with the subsets arranged in lexicographical order.*

Theorem 3 (Binet-Cauchy [1, pp. 93]) *For any $A\in\mathbb{R}^{k\times n}, B\in\mathbb{R}^{n\times m}$ and $1\leq q\leq\min(k,m,n)$ we have*

$$C_q(AB)=C_q(A)C_q(B) \quad (15)$$

This theorem allows us to state a strengthened version of [51, Corollary 5] by exploiting the fact that $\mathrm{tr}A^\top B$ is a scalar product between the entries of A and B:

Corollary 4 *For any matrices $A,B\in\mathbb{R}^{n\times k}$ and $1\leq q\leq\min(n,k)$ the function*

$$k_q(A,B):=\mathrm{tr}C_q(A^\top B) \quad (16)$$

is a kernel. We have the following special cases

$$k_1(A,B)=\mathrm{tr}C_1(A)^\top C_1(B)=\mathrm{tr}A^\top B \quad (17)$$
$$k_n(A,B)=\det A^\top B. \quad (18)$$

Moreover, when defining kernels over $\mathcal{X}^{\mathcal{T}}$ we can modify (17) and (18) to include a positive semidefinite weighting matrix W over the time domain, such as exponential downweighting. This leads to the kernel functions $\mathrm{tr}A^\top WB$ and $\det A^\top WB$.

To establish a connection to dynamical systems we only need to realize that the trajectories $\mathrm{Traj}(x,\mathbf{A})$ are essentially matrices. As we shall see, (17) and some of its refinements lead to kernels which carry out comparisons of state pairs. Eq. (18), on the other hand, can be used to define kernels via subspace angles. In this case, one whitens the trajectories before computing their determinant. We give technical details in Section 4.3.

3.4 Trace Kernels

Computing $\mathrm{tr}A^\top B$ means taking the sum over scalar products between the rows of A and B. For trajectories this amounts to summing over $\langle x_{\mathbf{A}}(t),x'_{\mathbf{A}'}(t)\rangle$ with respect to t. There are two potential problems arising from this strategy:

- The sum over t need not converge. This is easily amended by using a probability measure μ over the domain $\mathcal{T}$. The particular choice of μ gives rise to a number of popular and novel kernels for graphs and similar discrete objects [28,18]. The exponential discounting schemes

$$\mu(t)=\lambda^{-1}e^{-\lambda t} \text{ for } \mathcal{T}=\mathbb{R}_0^+ \quad (19)$$
$$\mu(t)=\frac{1}{1-e^{-\lambda}}e^{-\lambda t} \text{ for } \mathcal{T}=\mathbb{N}_0 \quad (20)$$

  are popular choice in reinforcement learning [42,6] and control theory. Another possible measure is

$$\mu(t)=\delta_\tau(t) \quad (21)$$

  where δ_τ corresponds to the Kronecker-δ for $\mathcal{T}=\mathbb{N}_0$ and to the Dirac's δ-distribution for $\mathcal{T}=\mathbb{R}_0^+$.
- $x_{\mathbf{A}}(t)$ by itself need not lie in a scalar product space. This can be addressed by the assumption that there exists some kernel $k(x,x')$ on the domain $\mathcal{X}\times\mathcal{X}$.

The above considerations allow us to extend (17) to obtain the following kernel

$$k((x,\mathbf{A}),(x',\mathbf{A}')):=\mathop{\mathbf{E}}_{t\sim\mu(t)}\left[k\left(x_{\mathbf{A}}(t),x'_{\mathbf{A}'}(t)\right)\right] \quad (22)$$

on both initial conditions and dynamical systems.

Kernels on dynamical systems: We can specialize (22) to kernels on initial conditions or dynamical systems only, simply by taking expectations over a distribution of them. This means that we can define

$$k(\mathbf{A},\mathbf{A}'):=\mathop{\mathbf{E}}_{x,x'}\left[k\left((x,\mathbf{A}),(x',\mathbf{A}')\right)\right]. \quad (23)$$

However, we need to show that (23) actually satisfies Mercer's condition.

Theorem 5 *Assume that $\{x,x'\}$ are drawn from an infinitely extendable and exchangeable probability distribution. Then (23) satisfies Mercer's condition.*

Proof. By De Finetti's theorem [13], infinitely exchangeable distributions arise from conditionally independent random variables. In practice this means that

$$p(x, x') = \int p(x|c)p(x'|c)dp(c) \tag{24}$$

for some $p(x|c)$ and a measure $p(c)$. Hence we can rearrange the integral in (23) to obtain

$$k(\mathbf{A}, \mathbf{A}') = \int \left[k\left((x, \mathbf{A}), (x', \mathbf{A}')\right) dp(x|c)dp(x'|c)\right] dp(c).$$

Here the result of the inner integral is a kernel by the decomposition admitted by Mercer's theorem. Taking a convex combination of such kernels preserves Mercer's condition, hence $k(\mathbf{A}, \mathbf{A}')$ is a kernel. ■

In practice, one typically uses either $p(x, x') = p(x)p(x')$ if independent averaging over the initial conditions is desired, or $p(x, x') = \delta(x - x')p(x)$ whenever the averaging is assumed to occur synchronously.

Kernels via dynamical systems: By the same strategy, we can also define kernels exclusively on initial conditions x, x' simply by averaging over the dynamical systems they should be subjected to:

$$k(x, x') := \mathop{\mathbf{E}}_{\mathbf{A},\mathbf{A}'} \left[k\left((x, \mathbf{A}), (x', \mathbf{A}')\right)\right]. \tag{25}$$

As before, whenever $p(\mathbf{A}, \mathbf{A}')$ is infinitely exchangeable in $\mathbf{A}, \mathbf{A}'$, (25) corresponds to a proper kernel. [3] Note that in this fashion the metric imposed on initial conditions is one that follows naturally from the particular dynamical system under consideration. For instance, differences in directions which are rapidly contracting carry less weight than a discrepancy in a divergent direction of the system.

3.5 Determinant Kernels

Instead of computing traces of $\mathrm{Traj}(x, \mathbf{A})^\top \mathrm{Traj}(x', \mathbf{A}')$ we can follow (18) and compute determinants of such expressions. As before, the following extensions are useful:

- We allow for the introduction of a kernel to compare states. This means that the scalar products in $\mathrm{Traj}(x, \mathbf{A})^\top \mathrm{Traj}(x', \mathbf{A}')$ are replaced by $k(x_{\mathbf{A}}(t), x'_{\mathbf{A}'}(t))$.
- Moreover, as before we insert a positive semidefinite matrix weighting matrix W over the time domain. For the sake of computational tractability one typically chooses a diagonal matrix with a finite number of zeros.

[3] Note that we made no specific requirements on the parameterization of $\mathbf{A}, \mathbf{A}'$. For instance, for certain ARMA models the space of parameters has the structure of a manifold. The joint probability distribution, by its definition, has to take such facts into account. Often the averaging simply takes additive noise of the dynamical system into account.

This means that we are computing the determinant of a kernel matrix, which in turn is then treated as a kernel function. This allows us to give an information-theoretic interpretation to the so-defined kernel function. Indeed, [20,3] show that such determinants can be seen as measures of the independence between sequences. This means that independent sequences can now be viewed as orthogonal in some feature space, whereas a large overlap indicates statistical correlation.

4 Kernels on Linear Dynamical Models

A special yet important class of dynamical systems are LTI systems of the form

$$\begin{aligned} y_t &= Cx_t + w_t \qquad \text{where } w_t \sim \mathcal{N}(0, R), \\ x_{t+1} &= Ax_t + v_t \qquad \text{where } v_t \sim \mathcal{N}(0, Q), \end{aligned} \tag{26}$$

where the driving noise w_t, v_t consists of iid normal random variables with zero mean, y_t are the observed random variables at time t, x_t are the latent variables, and the matrices A, C are the parameters of the dynamical system.

In this section, we show how to efficiently compute an expression for the trace kernel between two LTI systems (x_0, A, C) and (x'_0, A', C'), where x_0 and x'_0 are the initial conditions. We also establish a connection between the determinant kernel, the Martin kernel [31] and the kernels based on subspace angles [11] .

4.1 Trace kernels on LTI systems

If one assumes an exponential discounting $\mu(t) = e^{-\lambda t}$ with rate $\lambda > 0$, then the trace kernel for LTI systems is

$$k((x_0, A, C), (x'_0, A', C')) := \mathop{\mathbf{E}}_{v,w} \left[\sum_{t=1}^{\infty} e^{-\lambda t} y_t^\top W y'_t\right], \tag{27}$$

where W is a user-specified positive semidefinite matrix W specifying the kernel in $\mathcal{Y}$ as $k(y, y') = y^\top W y'$ — by default, one may choose $W = \mathbf{1}$, which leads to the standard Euclidean scalar product between y_t and y'_t.

A major difficulty in computing the above kernel, is that it involves computing an infinite sum. As it turns out, one can avoid computing such a sum by solving a couple of Sylvester equations. Before proceeding further, we need the following technical lemma.

Lemma 6 *Denote by A, B linear operators on $\mathcal{X}$. Then for all λ such that $e^{-\lambda}\|A\|\|B\| < 1$ and for all linear operators $W : \mathcal{X} \to \mathcal{X}$ the series*

$$M := \sum_{t=1}^{\infty} e^{-\lambda t} A^t W B^t \tag{28}$$

converges and M can be computed by solving the Sylvester equation $e^{-\lambda}AMB+e^{-\lambda}AWB = M$. If the series starts at a $t = 0$, then M is the solution of $e^{-\lambda}AMB+W = M$.

Note that Sylvester equations of type $AXB + CXD = E$ can be readily solved at $O(n^3)$ time [17] with freely available code ($A, B, C, D \in \mathbb{R}^{n\times n}$).

Proof. To show that M is well defined we use the triangle inequality, leading to

$$\begin{aligned}\|M\| &= \left\|\sum_{t=1}^{\infty} e^{-\lambda t}A^tWB^t\right\| \leq \sum_{t=1}^{\infty}\left\|e^{-\lambda t}A^tWB^t\right\| \\ &\leq \sum_{t=0}^{\infty}\left(e^{-\lambda}\|A\|\|B\|\right)^t\|W\| = \frac{\|W\|}{1-e^{-\lambda}\|A\|\|B\|}.\end{aligned}$$

Next we decompose the sum in M to obtain

$$\begin{aligned}M &= e^{-\lambda}AWB + \sum_{t=1}^{\infty} e^{-\lambda t}A^tWB^t \\ &= e^{-\lambda}AWB + e^{-\lambda}A\left[\sum_{t=1}^{\infty} e^{-\lambda t}A^tWB^t\right]B \\ &= e^{-\lambda}AWB + e^{-\lambda}AMB.\end{aligned}$$

Similarly if the series starts at $t = 0$. ∎

We now have all the ingredients to derive a closed form expression for computing the trace kernel (27).

Theorem 7 *If $e^{-\lambda}\|A\|\|A'\| < 1$, then the kernel of (27) is given by*

$$k = x_0^\top Mx_0' + \left(e^\lambda - 1\right)^{-1}\operatorname{tr}\left[Q\tilde{M} + WR\right], \tag{29}$$

where $M, \tilde{M}$ satisfy

$$M = e^{-\lambda}A^\top C^\top WC'A' + e^{-\lambda}A^\top MA' \tag{30}$$
$$\tilde{M} = C^\top WC' + e^{-\lambda}A^\top\tilde{M}A'. \tag{31}$$

Proof. By repeated substitution of (26) we see that

$$y_t = C\left[A^tx_0 + \sum_{i=0}^{t-1}A^iv_{t-i}\right] + w_t. \tag{32}$$

Hence in order to compute k we need to take expectations and sums over 9 different terms for every $y_t^\top My_t'$. Fortunately, terms involving v_i alone, w_i alone, and the mixed terms involving v_i, w_j for any i, j, and the mixed terms involving v_i, v_j for $i \neq j$ vanish since all the random variables are zero mean and independent. Next note that

$$\mathop{\mathbf{E}}_{w_t}\left[w_t^\top Ww_t\right] = \operatorname{tr} WR, \tag{33}$$

where R is the covariance matrix of w_t, as specified in (26). Taking sums over $t \in \mathbb{N}$ yields

$$\sum_{t=1}^{\infty} e^{-\lambda t}\operatorname{tr} WR = \left(e^\lambda - 1\right)^{-1}\operatorname{tr} WR. \tag{34}$$

Next we address the terms depending only on x_0, x_0'. Define

$$\bar{W} := C^\top WC'. \tag{35}$$

Then we have

$$\begin{aligned}&\sum_{t=1}^{\infty} e^{-\lambda t}(CA^tx_0)^\top W(C'A'^tx_0') && (36)\\ =&x_0^\top\left[\sum_{t=1}^{\infty} e^{-\lambda t}(A^t)^\top\bar{W}A'^t\right]x_0' && (37)\\ =&x_0^\top Mx_0', && (38)\end{aligned}$$

where M is the solution of a Sylvester equation

$$e^{-\lambda}A^\top C^\top WC'A' + e^{-\lambda}A^\top MA' = M. \tag{39}$$

The last terms to be considered are those depending on v_i. Using the fact that for $i \neq j$ the random variables v_i, v_j are independent, we have

$$\begin{aligned}&\sum_{t=1}^{\infty}\sum_{j=0}^{t-1} e^{-\lambda t}(CA^jv_{t-j})^\top W(C'A'^jv_{t-j}) && (40)\\ =&\operatorname{tr}Q\left[\sum_{t=1}^{\infty} e^{-\lambda t}\sum_{j=0}^{t-1}(A^j)^\top\bar{W}A'^j\right] && (41)\\ =&\operatorname{tr}Q\left[\sum_{j=0}^{\infty}\frac{1}{e^\lambda - 1}e^{-\lambda j}(A^j)^\top\bar{W}A'^j\right] && (42)\\ =&\left(e^\lambda - 1\right)^{-1}\operatorname{tr}Q\tilde{M}, && (43)\end{aligned}$$

where $\tilde{M}$ is the solution of the Sylvester equation

$$C^\top WC' + e^{-\lambda}A^\top\tilde{M}A' = \tilde{M}. \tag{44}$$

Here (42) follows from rearranging the sums, which is permissible because $e^{-\lambda}\|A\|\|A'\| < 1$, hence the series is absolutely convergent. Combining (33), (38), and (43) yields the result. ∎

Therefore, in the case of LTI systems, the trace kernel can be computed efficiently by solving two Sylvester equations. This can be done at a cost $O(m^3)$ where m is

the dimensionality of x [17], the same order as a matrix-matrix multiplication between the characteristic matrices of the systems.

In case we wish to be independent of the initial conditions, we can simply take the expectation over x_0, x_0'. Since the only dependency of (29) on the initial conditions manifests itself in the first term of the sum, the kernel on the dynamical systems (A, C) and (A', C') is

$$k((A,C),(A',C')) := \mathop{\mathbf{E}}_{x_0,x_0'}\left[k((x_0,A,C),(x_0',A',C'))\right] = \mathrm{tr}\Sigma M + \left(e^{\lambda}-1\right)^{-1}\mathrm{tr}\left[Q\tilde{M}+WR\right],$$

where Σ is the covariance matrix of the initial conditions x_0, x_0' (assuming that we have zero mean).

An important special case are fully observable LTI systems, *i.e.,* systems with $C = \mathbf{1}$ and $R = \mathbf{0}$. In this case, the expression for the trace kernel reduces to

$$k = x_0^\top M x_0' + \left(e^{\lambda}-1\right)^{-1}\mathrm{tr}(Q\tilde{M}), \tag{45}$$

where $M, \tilde{M}$ satisfy

$$M = e^{-\lambda}A^\top W A' + e^{-\lambda}A^\top M A' \tag{46}$$
$$\tilde{M} = W + e^{-\lambda}A^\top \tilde{M} A'. \tag{47}$$

This special kernels were first derived in [40], though with the summation starting at $t = 0$.

4.2 Determinant kernels on LTI systems

As before, we consider an exponential discounting $\mu(t) = e^{-\lambda t}$. We thus obtain the following expression for the determinant kernel on LTI systems:

$$k((x_0,A,C),(x_0',A',C')) := \mathop{\mathbf{E}}_{v,w}\det\left[\sum_{t=1}^{\infty} e^{-\lambda t} y_t y_t'^\top\right]. \tag{48}$$

Notice that in this case the effect of the weight matrix W is just a scaling of k by $\det(W)$, thus we assume $W = \mathbf{1}$ w.l.o.g.

Also for the sake of simplicity of the calculations and in order to compare with other kernels we assume an LTI system with no noise, *i.e.,* $v_t = 0$ and $w_t = 0$. Then we have the following:

Theorem 8 *If $e^{-\lambda}\|A\|\|A'\| < 1$, then the kernel of (48) is given by*

$$k = \det CMC'^\top, \tag{49}$$

where M satisfies

$$M = e^{-\lambda}Ax_0x_0'^\top A'^\top + e^{-\lambda}AMA'^\top. \tag{50}$$

Proof. We simply have

$$k = \det\sum_{t=1}^{\infty} e^{-\lambda t}CA^t x_0 x_0'^\top A'^\top C'^\top = \det CMC'^\top$$

where

$$M = \sum_{t=1}^{\infty} e^{-\lambda t}A^t x_0 x_0'^\top A'^\top = e^{-\lambda}Ax_0x_0'^\top A'^\top + e^{-\lambda}AMA'^\top.$$

■

As pointed out in[51], the determinant is *not* invariant to permutations of the columns of $\mathrm{Traj}(x, \mathbf{A})$ and $\mathrm{Traj}(x', \mathbf{A}')$. Since different columns or linear combinations of them are determined by the choice of the initial conditions x_0 and x_0', this means, as obvious from the formula for M, that the determinant kernel does depend on the initial conditions.

In order to make it independent from initial conditions, as before we can take expectations over x_0 and x_0'. Unfortunately, although M is linear in on both x_0 and x_0', $k = \det CMC'^\top$ is multilinear. Therefore, the kernel depends not only on the covariance Σ on the initial conditions, but also on higher order statistics. Only in the case of single-output systems, we obtain a simple expression

$$k((A,C),(A',C')) = CMC'^\top, \tag{51}$$

where

$$M = \mathrm{e}^{-\lambda}A\Sigma A'^\top + e^{-\lambda}AMA'^\top, \tag{52}$$

for the kernel on dynamical systems only.

4.3 Kernel via subspace angles and Martin kernel

Another way of defining a kernel on dynamical systems only is by using subspace angles among observability subspaces, as proposed in [31,11,51]. Such a kernel is defined as

$$-\log\prod_{i=1}^{n}\cos^2(\theta_i) \tag{53}$$

where θ_i is the $i-th$ subspace angle between the column spaces of the extended observability matrices $\mathcal{O} = [C^\top\ A^\top C^\top\ \cdots]^\top$ and $\mathcal{O}' = [C'^\top\ A'^\top C'^\top\ \cdots]^\top$. The above kernel can be efficiently computed as $\det(Q^\top Q')^2$, where $\mathcal{O} = QR$ and $\mathcal{O}' = Q'R'$ are the QR decompositions of $\mathcal{O}$ and $\mathcal{O}'$, respectively. Therefore, the kernel based on subspace angles is essentially the square of a

determinant kernel [4] formed from a whitened version of the outputs (via the QR decomposition) rather than from the outputs directly, as with the determinant kernel in (49).

Yet another way of defining a kernel on dynamical systems only is via cepstrum coefficients, as proposed by Martin [31]. More specifically, if $H(z)$ is the transfer function of the ARMA model described in (26), then the cepstrum coefficients are defined via the Laplace transformation of the logarithm of the ARMA power spectrum,

$$\log H(z)H^*(z^{-1}) = \sum_{n\in\mathbb{Z}} c_n z^{-n}. \tag{54}$$

The Martin kernel between (A, C) and (A', C') with corresponding cepstrum coefficients c'_n is defined as

$$k((A, C), (A', C')) := \sum_{n=1}^{\infty} n c_n^* c'_n. \tag{55}$$

As it turns out, the Martin kernel and the kernel based on subspace angles are the same as show in [11].

5 Extension to Nonlinear Dynamical Systems

We now extend our results to the case of nonlinear dynamical models of the form

$$x_{t+1} = f(x_t). \tag{56}$$

5.1 Linear in Feature Space

For such systems, the simplest possible extension would be to seek a bijective transformation $x \mapsto z = \Phi(x)$ such that in the new coordinates we obtain a linear system

$$z_{t+1} = Az_t. \tag{57}$$

Then, we can define a kernel on the nonlinear models with vectors fields f and f' as

$$k_{nonlinear}((x_0, f), (x'_0, f')) = k_{linear}((z_0, A), (z'_0, A'))$$

where k_{linear} is any of the kernels for linear models defined in the previous sections.

The above construction can be immediately applied whenever the vector fields f and f' are feedback-linearizable. Conditions for f to be feedback-linearizable as well as an algorithm for computing Φ, hence A, from f can be found in [24].

[4] Recall that the square of a kernel is a kernel thanks to the product property.

However, a given f is in general not feedback-linearizable. In such cases, we propose to find an injection Φ such that

$$\Phi(x_{t+1}) = A\Phi(x_t) + v_t \tag{58}$$

for a fully observable model and $\Phi'(y_t) = C\Phi(x_t) + w_t$ for a partially observable model. In the following we only deal with the fully observable case. The key in these transformations is that $\Phi(x)$ and x have the same dimensionality. Consequently we can define kernels on the so-defined dynamical systems, simply by studying trajectories in $\phi(x_t)$ rather than x_t.

Unfortunately, such a transformation need not always exist. Moreover, for the purpose of comparing trajectories it is not necessary that the map Φ be bijective. In fact, injectivity is all we need. This means that as long as we can find a linear system such that (58) holds we can extend the tools of the previous sections to nonlinear models. In essence this is what was proposed in [34,4] in the context of time-series prediction. In the following, we only consider the fully observable case.

The problem with (58) is that once we spell it out in feature space using Φ the matrix A turns into an operator. However, since we have only a finite number of observations at our disposition we need to impose further restrictions on the operators for a useful result.

5.2 Solution by Restriction

The first restriction that we impose is that the image of A be contained in the span of the $\Phi(x_i)$. This means that we can find an equivalent condition to (58) via

$$k(x_i, x_{t+1}) = \underbrace{\Phi(x_i)^\top A\Phi(x_t)}_{:=\tilde{A}_{it}} + \langle \Phi(x_i), v_t\rangle. \tag{59}$$

For a perfect solution we "only" need to find an operator A for which

$$KT_+ = \tilde{A}, \tag{60}$$

where $T_+ \in \mathbb{R}^{m\times m-1}$ is the shift operator in $\mathbb{R}^m$, that is $(T_+)_{ij} = \delta_{i-1,j}$. Moreover K is the kernel matrix $k(x_i, x_j)$. For a large number of kernel matrices K with full rank, such a solution always exists regardless of the dynamics, which leads to overfitting problems. Consequently we need to restrict A further.

A computationally attractive option is to restrict the rank of A further so that

$$A := \sum_{i,j\in S} \alpha_{ij}\Phi(x_i)\Phi(x_j)^\top \tag{61}$$

for some subset S. We choose the same basis in both terms such that the image of A and of its adjoint operator

$A^\top$ lie in the same subspace. Using $\tilde{K} \in \mathbb{R}^{m \times |S|}$ with $\tilde{K}_{ij} = k(x_i, x_j)$ (where $j \in S$) we obtain the following condition:

$$KT_+ = \tilde{K}\alpha\tilde{K}T_-. \tag{62}$$

Here $T_- \in \mathbb{R}^{m \times m-1}$ is the inverse shift operator, that is $(T_-)_{ij} = \delta_{ij}$. One option to solve (62) is to use pseudo-inverses. This yields

$$\alpha = \tilde{K}^\dagger (KT_+)(\tilde{K}T_-)^\dagger. \tag{63}$$

5.3 Sylvester Equations in Feature Space

Finally, one needs to solve the Sylvester equations, or QR factorizations or determinants in feature space. We will only deal with the Sylvester equation below. Using derivations in [51] it is easy to obtain analogous expressions for the latter cases.

We begin with (46). Without loss of generality we assume that $W = \mathbf{1}$ (other cases can be easily incorporated into the scalar product directly, hence we omit them). Moreover we assume that A, A' have been expanded using the same basis $\Phi(x_i)$ with $i \in S$.

The RHS of (46) has the same expansion as A, hence we can identify M uniquely by observing the action of M on the subspace spanned by $\Phi(x_i)$. Using $M = \sum_{i,j \in S} \eta_{ij} \Phi(x_i)\Phi(x_j)^\top$ and $\bar{K}_{ij} := k(x_i, x_j)$ we obtain

$$\begin{aligned} \Phi(x_i)^\top M \Phi(x_j) =& [\bar{K}\eta\bar{K}]_{ij} \\ =& e^{-\lambda} \left[\bar{K}\alpha^\top \bar{K}\alpha'\bar{K}\right]_{ij} + \\ & e^{-\lambda} \left[\bar{K}\alpha^\top \bar{K}\eta\bar{K}\alpha'\bar{K}\right]_{ij} \end{aligned} \tag{64}$$

Assuming that $\tilde{K}$ has full rank (which is a reasonable assumption for a set of vectors used to span the image of an operator, we can eliminate $\bar{K}$ on both sides of the equation and we have the following new Sylvester equation:

$$\eta = e^{-\lambda}\alpha^\top \bar{K}\alpha' + e^{-\lambda}\alpha^\top \bar{K}\eta\bar{K}\alpha'. \tag{65}$$

In the same fashion, (47) can be expressed in terms of a feature space representation. This yields

$$\tilde{\eta} = \tilde{K} + e^{-\lambda}\alpha^\top \tilde{K}\tilde{\eta}\tilde{K}\alpha'. \tag{66}$$

Finally, computing traces over finite rank operators can be done simply by noting that

$$\operatorname{tr} \sum_{i,j} \alpha_{ij} \Phi(x_i)\Phi(x_j)^\top = \operatorname{tr} \sum_{i,j} \alpha_{ij} \Phi(x_j)^\top \Phi(x_i) = \operatorname{tr} \tilde{K}\alpha. \tag{67}$$

6 Markov Processes

Many discrete objects can be interpreted as describing a Markov process. This connection is then used to define kernels on them. One such example are graphs, which we will discuss in Section 6.2. The combination with induced feature spaces then leads to efficient similarity measures between discrete structures (Section 6.3).

We begin with some elementary definitions: Markov processes have the property that their time evolution behavior depends only on their current state and the state transition properties of the model. Denote by S the set of discrete states, [5] . The evolution of the probabilities (denoted by $x \in [0,1]^S$) satisfies the Bellman equation. This means that

$$x(t+1) = Ax(t) \text{ or } \frac{d}{dt}x(t) = Ax(t) \tag{68}$$

for discrete-time and continuous time processes respectively. Here A is the state transition matrix, that is for discrete processes $A_{ij} = p(i|j)$ is the probability of reaching state i from state j.

6.1 General Properties

A kernel defined via A computes the average overlap between the states when originating from $x = e_i, \tilde{x} = e_j$ in $k(i,j)$. Here e_i are "pure" states, that is the system is guaranteed to be in state i rather than in a mixture of states.

Since A is a stochastic matrix (positive entries with row-sum 1), its eigenvalues are bounded by 1 and therefore, any discounting factor $\lambda > 0$ will lead to a well-defined kernel.

Note that the average overlap between state vectors originating from different initial states are used in the context of graph segmentation and clustering [47,21]. This means that $\mu(t)$ is nonzero only for some $t \leq t_0$, which is similar to heavy discounting.

Recall, however, if e^λ is much smaller than the mixing time, k will almost exclusively measure the overlap between the initial states $x, \tilde{x}$ and the transient distribution on the Markov process. The quantity of interest here will be the ratio between e^λ and the gap between 1 and the second largest eigenvalue of A [19].

An extension to *Continuous-Time Markov Chains* (CMTC) is straightforward. Again $x(t)$ corresponds to the state at time t and the matrix A (called the rate

[5] Extensions to the continuous case are straightforward, however for the sake of conciseness we omit them in this paper.

matrix in this context) denotes the differential change in the concentration of states.

When the CTMC reaches a state, it stays there for an exponentially distributed random time (called the state holding time) with a mean that depends only on the state. For instance, diffusion processes can be modelled in this fashion.

6.2 Graphs

In the present section we study an important special case of Markov processes, namely random walks on (directed) graphs. Here diffusion through each of the edges of the graph is constant (in the direction of the edge). This means that, given an adjacency matrix representation of a graph via D (here $D_{ij} = 1$ if an edge from j to i exists), we compute the Laplacian $L = D - \operatorname{diag}(D\mathbf{1})$ of the graph, and use the latter to define the diffusion process $\frac{d}{dt}x(t) = Lx(t)$.

- Using the measure $\mu(t) = \delta_\tau(t)$ of (21) we compute the overlap at time t, which yields

$$K = \exp(\tau L)^\top \exp(\tau L) \tag{69}$$

  as covariance matrix between the distributions over various states. K_{ij} therefore equals the probability that any other state l could have been reached jointly from i and j [28].
- The time τ chosen is user defined and consequently it tends to be debatable. On the other hand, we might as well average over a large range of τ, leading to a kernel matrix

$$K = \frac{1}{2}\left(A + \frac{\lambda}{2}\mathbf{1}\right)^{-1}. \tag{70}$$

  This yields a kernel whose inverse differs by $\frac{\lambda}{2}\mathbf{1}$ from the normalized graph Laplacian (another important quantity used for graph segmentation). The attraction of (70) is that its inverse is easy to come by and sparse, translating into significant computational savings for estimation.
- The kernel proposed by [18] can be recovered by setting W to have entries $\{0, 1\}$ according to whether vertices i and j bear the same label and considering a discrete time random walk rather than a continuous time diffusion processes (various measures $\mu(t)$ take care of the exponential and harmonic weights).
- Finally, discrete-time Markov processes can be treated in complete analogy, yielding either $K = (\mathbf{1} - A)^{-1}$ or similar variants, should snapshots be required.

[28] suggested to study diffusion on *undirected* graphs. As diffusion processes satisfy the Markov property, we get the kernels of Kondor and Lafferty as a special case of the above reasoning. In particular note that for undirected graphs $L = L^\top$ and consequently the kernel matrix K can simplify (69) to

$$K = \exp(\tau L)^\top \exp(\tau L) = \exp(2\tau L). \tag{71}$$

6.3 Inducing Feature Spaces

Burkhardt [8] uses features on graphs to derive invariants for the comparison of polygons. More specifically, he picks a set of feature functions, denoted by the vector $\Phi(x, p)$, defined on x with respect to the polygon p and he computes their value on the entire trajectory through the polygon. Here x is an index on the vertex of the polygon and the dynamical system simply performs the operation $x \to (x \bmod n) + 1$, where n is the number of vertices of the polygon.

Essentially, what happens is that one maps the polygon into the trajectory $(\Phi(1, p), \ldots, \Phi(n, p))$. For a fixed polygon, this already would allow us to compare vertices based on their trajectory. In order to obtain a method for comparing different polygons, one needs to rid oneself of the dependency on initial conditions: the first point is arbitrary. To do so, one simply assumes a uniform distribution over the pairs (x, x') of initial conditions. This distribution satisfies the conditions of de Finetti's theorem and we can therefore compute the kernel between two polygons via

$$k(p, p') = \frac{1}{nn'} \sum_{x,x'=1}^{n,n'} \langle \Phi(x, p), \Phi(x', p') \rangle. \tag{72}$$

Note that in this context the assumption of a uniform distribution amounts to computing the Haar integral over the cyclical group defined by the vertices of the polygon.

The key difference to [8] in (72) is that one is not limited to a small set of polynomials which need to be constructed explicitly. Instead, one can use any kernel function $k((x, p), (x', p'))$ for this purpose.

This is but a small example of how rich features on the states of a dynamical system can be used in the comparison of trajectories. What should be clear, though, is that a large number of the efficient computations has to be sacrificed in this case. Nonetheless, for discrete measures μ with a finite number of nonzero steps this can be an attractive alternative to a manual search for useful features.

7 Application to Video Sequences

Applications of kernel methods to computer vision have so far been largely restricted to methods which ana-

lyze a single image at a time, possibly with some post-processing to take advantage of the fact that images are ordered. Essentially there are two main approaches [22]: kernels which operate directly on the raw pixel data [36,7], such as the Haussdorff kernel which matches similar pixels in a neighborhood [5], and kernels which exploit the statistics of a small set of features on the image, such as intensity, texture, and color histograms [9].

However, when dealing with video sequences, such similarity measures are highly inadequate as they do not take the temporal structure of the data into account. The work of Doretto and coworkers [15,41] points out a means of dealing with dynamic textures by first approximating them with an autoregressive model, and subsequently computing the subspace angles between the models.

7.1 Setup

As in [15] we use the working assumption that dynamic textures are sequences of images of moving scenes which exhibit certain stationarity properties in time, which can be modeled by a stochastic process. The latter assumption has proven to be very reasonable, when applied to a large number of common phenomena such as smoke, fire, water flow, foliage, wind etc., as corroborated by the impressive results of Soatto and coworkers.

We briefly summarize the connection between ARMA models and dynamic textures. Let, $I_t \in \mathbb{R}^m$ for $t = 1, \ldots, \tau$ be a sequence of τ images and assume that at every time step we measure a noisy version $y_t = I_t + w_t$ where $w_t \in \mathbb{R}^m$ is Gaussian noise distributed as $\mathcal{N}(0, R)$. To model the sequences of observed images as a Gaussian ARMA model we will assume that there exists a positive integer $n \ll m$, a process $x_t \in \mathbb{R}^n$ and symmetric positive definite matrices $Q \in \mathbb{R}^{n \times n}$ and $R \in \mathbb{R}^{m \times m}$ such that (26) holds.

Without loss of generality, the scaling of the model is fixed by requiring

$$C^\top C = \mathbf{1} \tag{73}$$

Given a sequence of images the identification problem is to estimate A, B, Q and R. Exact solutions like the `n4sid` method in MATLAB exist but they are very expensive to compute for large images. Instead we use a sub-optimal closed form solution by computing the SVD of the matrix $Y := [y(1), \ldots, y(\tau)]$, as proposed in [15]. The latter is done for a) computational reasons and b) in order to be able to compare our results with previous work.

7.2 Parameters

The parameters of the ARMA model deserve some consideration. The initial conditions of a video clip are given by the x_0. This helps in distinguishing scenes which have a stationary background and dynamical texture part, e.g. a tree in the foreground with a lawn in the background and where the foreground shares similar dynamics, yet the stationary part differs.

In this case, if we use identical initial conditions for both systems, the similarity measure will focus on the dynamical part (the foreground). On the other hand, if we make use of x_0, we will be able to distinguish between scenes which only differ in their background.

Another important factor which influences the value of the kernel is the value of λ. If we want to provide more weightage to short range interactions due differences in initial conditions it might be desirable to use a large values of λ since it results in heavy attenuation of contributions as time t increases. On the other hand, when we want to identify samples of the same dynamical system it might be desirable to use small values of λ.

Finally, A, C determine the dynamics. Note that there is no requirement that A, A' share the same dimensionality. Indeed, the only condition on the two systems is that the spaces of observations y_t, y'_t agree. This allows us, for instance, to determine the approximation qualities of systems with various detail of parameterization.

7.3 Change Detection within Sequences

A convenient side-effect of the kernels derived in the current context is that they allow us to detect changes within a video sequence (e.g. water starting to boil, changes in smoke patterns), simply by computing the ARMA model estimates at various points in time. Figure 2 shows how changes between video clips lead to increased distances, which can be used for change detection directly or via an online novelty detector [38].

8 Experiments

For our experiments we used some sequences from the *MIT temporal texture database*. We also constructed our own database of dynamic textures by capturing natural and artificial objects like trees, water bodies, water flow in kitchen sinks, etc. For instance we recorded the movement of the leaves of a tree at the same angle but under various wind conditions. Each sequence consists of 120 frames. We downsampled the images to use a grayscale colormap (256 levels) with each frame of size 115x170. Figure 1 shows some samples from our dataset.

We used the procedure outlined in [15] for estimating the model parameters A, B and C. Once the system parameters are estimated we compute distances between models using our trace kernel as well as the Martin distance described in [15]. We varied the value of the down-weighting parameter λ and report results for two values

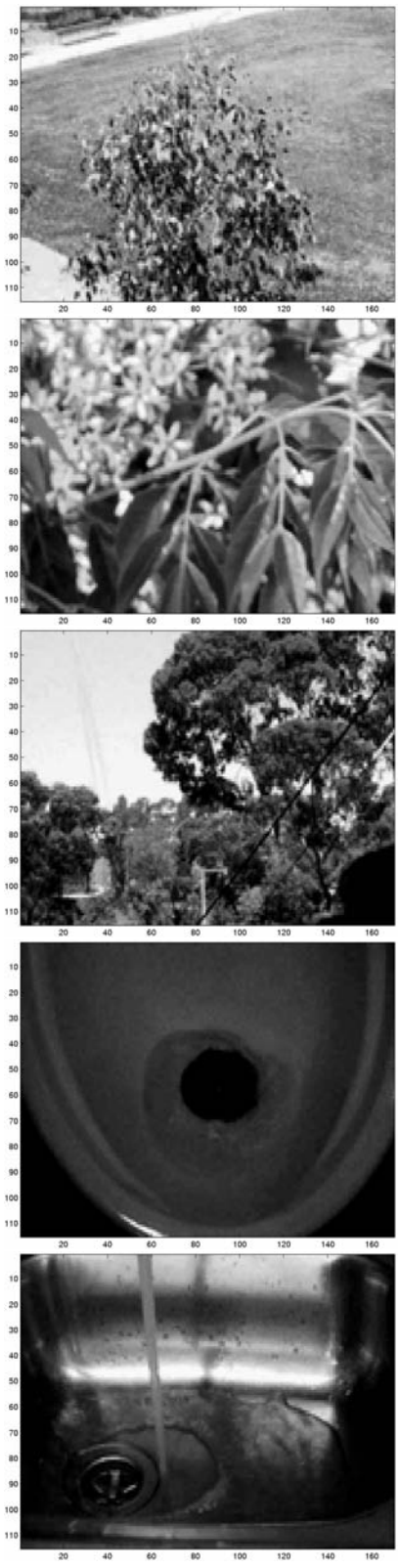

Fig. 1. Some sample textures from our dataset

$\lambda = 0.1$ and $\lambda = 0.9$. The distance matrix obtained using each of the above methods is shown in Figure 2.

In general, clips which are closer to each other on an axis are closely related (that is they are either from similar natural phenomenon or are extracted from the same master clip). Hence a perfect distance measure will produce a block diagonal matrix with a high degree of correlation between neighboring clips. As can be seen from our plots, the kernel using trajectories assigns a high similarity to clips extracted from the same master clip while the Martin distance fails to do so. Another interesting feature of our approach is that the value of the kernel seems to be fairly independent of λ. The reason for this might be because we consider long range interactions averaged out over infinite time steps. Two dynamic textures derived from the same source might exhibit very different short term behavior induced due to the differences in the initial conditions. But once these short range interactions are attenuated we expect the two systems to behave in a more or less similar fashion. Hence, an approach which uses only short range interactions might not be able to correctly identify these clips.

To further test the effectiveness of our method, we introduced some "corrupted" clips (clip numbers 65 - 80). These are random clips which clearly cannot be modeled as dynamic textures. For instance, we used shots of people and objects taken with a very shaky camera. From Figure 2 it is clear that our kernel is able to pick up such random clips as novel. This is because our kernel compares the similarity between each frame during each time step. Hence if two clips are very dissimilar our kernel can immediately flag this as novel.

9 Summary and Outlook

The current paper sets the stage for kernels on dynamical systems as they occur frequently in linear and affine systems. By using correlations between trajectories we were able to compare various systems on a behavioral level rather than a mere functional description. This allowed us to define similarity measures in a natural way.

While the larger domain of kernels on dynamical systems is still untested, special instances of the theory have proven to be useful in areas as varied as classification with categorical data [28,18] and speech processing [12]. This gives reason to believe that further useful applications will be found shortly.

For instance, we could use kernels in combination with novelty detection to determine unusual initial conditions, or likewise, to find unusual dynamics. In addition, we can use the kernels to find a metric between objects such as HMMs, e.g., to compare various estimation methods.

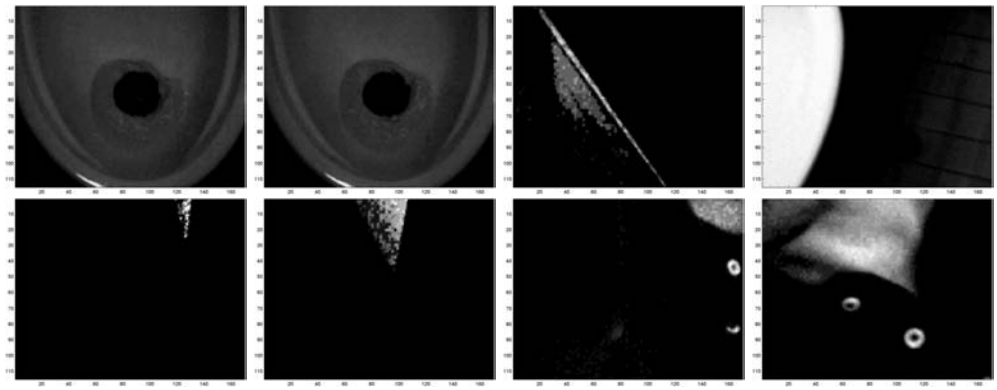

Fig. 3. Typical frames from a few samples are shown. The first two correspond to a flushing toilet, the remainder are corrupted sequences and do not correspond to a dynamical texture. The distance matrix shows that our kernel is able to pick out this anomaly.

In this paper, we show that the Martin distance used for dynamic texture recognition is a kernel. The main drawback of the Martin kernel is that it does not take into account the initial conditions and might be very expensive to compute. To overcome these drawbacks we introduced a kernel based on comparing the trajectory of a linear first-order ARMA process. Our kernel is simple to implement and compute in closed form. By appropriately choosing downweighting parameters we can concentrate of either short term or long term interactions. Future work will focus on applications of our kernels to system identification and computation of closed form solutions for higher order ARMA processes.

Acknowledgements

This work was supported by a grant of the ARC. We thank Frederik Schaffalitzky, Bob Williamson, Laurent El Ghaoui, Patrick Haffner, and Daniela Pucci de Farias for useful discussions and Gianfranco Doretto for providing code for the estimation of linear models.

References

[1] A.C. Aitken. *Determinants and Matrices*. Interscience Publishers, 4 edition, 1946.

[2] M. A. Aizerman, É. M. Braverman, and L. I. Rozonoér. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

[3] F. R. Bach and M. I. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.

[4] G. Bakir, J. Weston, and B. Schölkopf. Learning to find pre-images. In *Advances in Neural Processing Systems NIPS*. MIT Press, 2003.

[5] A. Barla, F. Odone, and A. Verri. Hausdorff kernel for 3D object acquisition and detection. In *European Conference on Computer Vision '02*, number 2353 in LNCS, page 20. Springer, 2002.

[6] J. Baxter and P.L. Bartlett. Direct gradient-based reinforcement learning: Gradient estimation algorithms. Technical report, Research School of Information, ANU Canberra, 1999.

[7] V. Blanz, B. Schölkopf, H. Bülthoff, C. Burges, V. Vapnik, and T. Vetter. Comparison of view-based object recognition algorithms using realistic 3D models. In C. von der Malsburg, W. von Seelen, J. C. Vorbrüggen, and B. Sendhoff, editors, *Artificial Neural Networks ICANN'96*, pages 251–256, Berlin, 1996. Springer Lecture Notes in Computer Science, Vol. 1112.

[8] H. Burkhardt. Invariants on skeletons and polyhedrals. Technical report, Universität Freiburg, 2004. in preparation.

[9] O. Chapelle, P. Haffner, and V. Vapnik. SVMs for histogram-based image classification. *IEEE Transactions on Neural Networks*, 10(5), 1999.

[10] V. Cherkassky and F. Mulier. *Learning from Data*. John Wiley and Sons, New York, 1998.

[11] K. De Cock and B. De Moor. Subspace angles between ARMA models. *Systems and Control Letter*, 46:265 – 270, 2002.

[12] C. Cortes, P. Haffner, and M. Mohri. Rational kernels. In *Proceedings of Neural Information Processing Systems 2002*, 2002. in press.

[13] B. de Finetti. *Theory of probability*, volume 1-2. John Wiley and Sons, 1990. reprint of the 1975 translation.

[14] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Number 31 in Applications of mathematics. Springer, New York, 1996.

[15] G. Doretto, A. Chiuso, Y.N. Wu, and S. Soatto. Dynamic textures. *International Journal of Computer Vision*, 51(2):91–109, 2003.

[16] S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, Dec 2001. http://www.jmlr.org.

[17] J. D. Gardiner, A. L. Laub, J. J. Amato, and C. B. Moler. Solution of the Sylvester matrix equation $AXB^\top + CXD^\top = E$. *ACM Transactions on Mathematical Software*, 18(2):223–231, 1992.

[18] T. Gärtner, P.A. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In B. Schölkopf and M. Warmuth, editors, *Sixteenth Annual Conference on Computational Learning Theory and Seventh Kernel Workshop, COLT*. Springer, 2003.

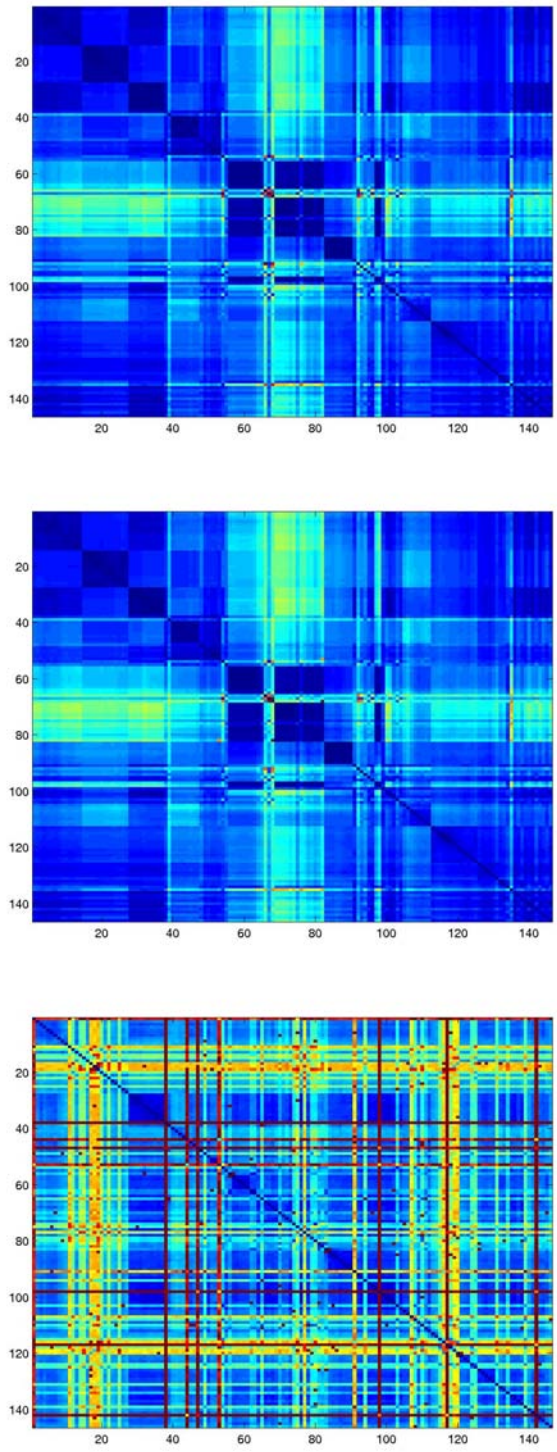

Fig. 2. Distance matrices obtained using the trace kernel and the Martin kernel. We used a value of $\lambda = 0.9$ for the first plot and a value of $\lambda = 0.1$ for the second plot. The matrix $W = \mathbf{1}$ was used for both plots. Clips which are closer to each other on an axis are closely related.

[19] F. C. Graham. Logarithmic Sobolev techniques for random walks on graphs. In D. A. Hejhal, J. Friedman, M. C. Gutzwiller, and A. M. Odlyzko, editors, *Emerging Applications of Number Theory*, number 109 in IMA Volumes in Mathematics and its Applications, pages 175–186. Springer, 1999. ISBN 0-387-98824-6.

[20] A. Gretton, R. Herbrich, and A.J. Smola. The kernel mutual information. In *Proceedings of ICASSP*, 2003.

[21] D. Harel and Y. Koren. Clustering spatial data using random walks. In *Knowledge Discovery and Data Mining (KDD01)*, pages 281–286, 2001.

[22] B. Heisele, P. Ho, and T. Poggio. Face recognition with support vector machines: Global versus component-based approach. In *Proceedings of the Eighth International Conference On Computer Vision (ICCV-01)*, pages 688–694, Los Alamitos, CA, 2001. IEEE Computer Society.

[23] R. Herbrich. *Learning Kernel Classifiers: Theory and Algorithms*. MIT Press, 2002.

[24] A. Isidori. *Nonlinear Control Systems*. Springer, 2 edition, 1989.

[25] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning*, pages 137–142, Berlin, 1998. Springer.

[26] T. Joachims. *Learning to Classify Text Using Support Vector Machines: Methods, Theory, and Algorithms*. The Kluwer International Series In Engineering And Computer Science. Kluwer Academic Publishers, Boston, May 2002. ISBN 0-7923-7679-X.

[27] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. A fast iterative nearest point algorithm for support vector machine classifier design. *IEEE Transactions on Neural Networks*, 11(1):124–136, January 2000.

[28] R. S. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the ICML*, 2002.

[29] H. König. *Eigenvalue Distribution of Compact Operators*. Birkhäuser, Basel, 1986.

[30] J. MacQueen. Some methods of classification and analysis of multivariate observations. In L. M. LeCam and J. Neyman, editors, *Proc. 5th Berkeley Symposium on Math., Stat., and Prob.*, page 281. U. California Press, Berkeley, CA, 1967.

[31] R.J. Martin. A metric for ARMA processes. *IEEE Transactions on Signal Processing*, 48(4):1164–1170, 2000.

[32] J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, London*, A 209:415–446, 1909.

[33] T. Poggio. On optimal nonlinear associative recall. *Biological Cybernetics*, 19:201–209, 1975.

[34] L. Ralaivola and F. d'Alché Buc. Dynamical modeling with kernels for nonlinear time series prediction. In *Advances in Neural Processing Systems NIPS*. MIT Press, 2003.

[35] P. Saisan, G. Doretto, Y.N. Wu, and S. Soatto. Dynamic texture recognition. In *Proceedings of CVPR*, volume 2, pages 58–63, 2001.

[36] B. Schölkopf. *Support Vector Learning*. R. Oldenbourg Verlag, München, 1997. Doktorarbeit, TU Berlin. Download: http://www.kernel-machines.org.

[37] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[38] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

[39] A.J. Smola and I.R. Kondor. Kernels and regularization on graphs. In B. Schölkopf and M. Warmuth, editors, *Proceedings of the Annual Conference on Computational Learning Theory*, Lecture Notes in Computer Science. Springer, 2003.

[40] A.J. Smola and S.V.N. Vishwanathan. Hilbert space embeddings in dynamical systems. In *Proceedings of the* 13^{th} *IFAC symposium on system identification*. IFAC, August 2003. In press.

[41] S. Soatto, G. Doretto, and Y.N. Wu. Dynamic textures. In *Proceedings of the Eighth International Conference On Computer Vision (ICCV-01)*, pages 439–446, Los Alamitos, CA, 2001. IEEE Computer Society.

[42] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[43] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

[44] V. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, New York, 1998.

[45] S. V. N. Vishwanathan, M. N. Murty, and A. J. Smola. SSVM: A simple SVM algorithm. In *Proceedings of the International Conference on Machine Learning*, 2003.

[46] G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, Philadelphia, 1990.

[47] Y. Weiss. Segmentation using eigenvectors: A unifying view. In *International Conference on Computer Vision ICCV*, pages 975–982, 1999.

[48] J. C. Willems. From time series to linear system. I. Finite-dimensional linear time invariant systems. *Automatica J. IFAC*, 22(5):561–580, 1986.

[49] J. C. Willems. From time series to linear system. II. Exact modelling. *Automatica J. IFAC*, 22(6):675–694, 1986.

[50] J. C. Willems. From time series to linear system. III. Approximate modelling. *Automatica J. IFAC*, 23(1):87–115, 1987.

[51] L. Wolf and A. Shashua. Learning oveer sets using kernel principal angles. *Jounal of Machine Learning Research*, 4:913–931, 2003.