Model-based Kernel for Efficient Time Series Analysis

Huanhuan Chen[§†], Fengzhen Tang[†], Peter Tino[†] and Xin Yao[†]
[§]UBRI, School of Computer Science and Technology, University of Science and Technology of China
[†]CERCIA, School of Computer Science, University of Birmingham, United Kingdom
hchen@ustc.edu.cn, {fxt126,p.tino,x.yao}@cs.bham.ac.uk

ABSTRACT

We present novel, efficient, model based kernels for time series data rooted in the reservoir computation framework. The kernels are implemented by fitting reservoir models sharing the same fixed deterministically constructed state transition part to individual time series. The proposed kernels can naturally handle time series of different length without the need to specify a parametric model class for the time series. Compared with most time series kernels, our kernels are computationally efficient. We show how the model distances used in the kernel can be calculated analytically or efficiently estimated. The experimental results on synthetic and benchmark time series classification tasks confirm the efficiency of the proposed kernel in terms of both generalization accuracy and computational speed. This paper also investigates on-line reservoir kernel construction for extremely long time series.

Categories and Subject Descriptors

G.3 [**Probability And Statistics**]: Time series analysis; H.2.8 [**Database Applications**]: Data mining

Keywords

Time Series, Reservoir Computing, Kernel Methods

1. INTRODUCTION

Kernel methods have received considerable attention in the machine learning community dealing with structured data, such as image, graphs, texts or voice signals. However, as an important ubiquitous data type in science and engineering, time series have received relatively less research in the kernel literature [5].

There has been active research on quantification of the 'similarity' or the 'distance' between time series. However, these measures are not always applicable for kernel approaches as many of such similarity measures are not positive definite, which is a necessary basis for the reproducing kernel Hilbert space to be valid.

A simple way to distinguish between two time series of the same length is to treat the time series as vectors and simply employ a linear kernel or Radial basis kernel. This method can be simple and efficient provided the time series are short and of equal length. However, in many real-world applications, the time series of interest are of variable-length and can be quite long. It is therefore desirable to construct kernels capable of handling possibly long time series of variable length. For example, in dynamic time warping [1] time series similarity is quantified through finding an alignment between variable-length multivariate time series.

Another possibility is to use a generative probabilistic model of the time series data and then define the time series kernel through model parameters corresponding to different sequences, e.g. probability product kernel [13], Kullback-Leibler (KL) divergence based kernels [19, 2] and Autoregressive kernel [5]. These approaches depend on the particular parametric model class. For example, *Fisher* kernel [11] maps individual time series into score functions of the single generative model that is assumed to be able to 'explain' most of the data. Often a Hidden Markov Model (HMM) with a fixed number of states is employed. In some situations the assumption of the particular generative model underlying the data can be too strong. In addition, *Fisher* kernels [11] are computationally expensive because of the calculation of metric tensor (inverse of *Fisher* information matrix) in the tangent space of the generative model manifold. The 'practical' *Fisher* kernel used in most of the time replaces the metric tensor with an identity matrix. This can result in a loss of valuable information in the data [25].

The requirement of using a single generative model in kernel calculations is relaxed e.g. in the Autoregressive kernel [5]. Sequences are judged to be similar/dissimilar according to the corresponding likelihood profile of a Vector Autoregressive Model (VAR) under a variety of parameter settings (controlled by the prior). In this case it is less crucial that the VAR model is a faithful model of the data since the base VAR model class is used as a 'feature extractor'.

Due to the requirements of many time series applications, the kernel evaluation should happen in real-time. Therefore, computational complexity of kernel construction and evaluation can play a critical role in applying kernel methods to time series data. However, many of the existing time series kernels are computationally demanding. For example Auto-correlation Operators (DACO) kernel [9] proposed recently by Gaidon et al. for action recognition, compares the dynamic aspects of two time series by using the difference between their auto-correlations. The kernelized DACO inevitably needs to invert a matrix of size related to the time series length. Thus the kernel can be used for relatively short time series only.

To address the problems mentioned above, we propose novel general time series kernels that can naturally and efficiently handle long time series data of variable length. The core idea is to transform the time series into a higher dimensional "dynamical feature space" via reservoir computation models [18] and then represent varying aspects of the signal through variation in the linear readout models trained in such dynamical feature spaces. In this way each time series will be represented by the corresponding readout model of the same *fixed* reservoir. Hence, unlike in the *Fisher* kernel, there will be a different dynamic model for each time series, but all such models will share the same dynamical reservoir. The sequence-specific dynamic models will differ only in the corresponding linear readout models from the reservoir. The intuition is that while the general fixed dynamic reservoir provides a unique and rich pool of dynamic features for the whole data set, the individual readout models bring enough flexibility to represent specifics of different time series, thus providing a platform for wide applicability across time series of different characteristics and origins.

One can, of course, argue that our approach is yet another variation on model-based kernel construction for time series based on a particular class of dynamic (reservoir) models. However, unlike parametric time series models of a particular from, reservoir models have been extensively shown to be 'generic' in the sense that they are able to represent a wide variety of dynamical features of the input signals, so that given a task at hand only the linear readout on top of the reservoir needs to be retrained [18]. As stated above, in our formulation, the underlying dynamic reservoir will be the *same* for all time series - the differences in the signal characteristics in different time series will be captured solely by the linear readout models and will be quantified in the function space of such models.

There are several advantages of such reservoir based time series kernels:

1. The proposed kernels can naturally handle time series of different length;

2. General reservoir model is flexible enough so that it can be used for a variety of data types without the need to specify a particular parametric model class for the time series;

3. Since only the linear readout on top of the reservoir needs to be trained, compared with most time series kernels, our kernels are computationally very efficient;

4. With recursive least squares algorithm to train readout mapping of reservoir models, our kernels can be operating in an on-line fashion, with the ability to efficiently handle extremely long time series;

5. Under some assumptions, the model distances between linear readouts can be formulated analytically.

The rest of this paper is organized as follows. Section 2 reviews the related work on kernels for time series. Section 3 introduces deterministic reservoir computing and proposes time series kernels based on reservoir models. The experimental results and analysis are reported in Section 4. Section 5 studies on-line reservoir kernel construction for extremely long time series. Finally, Section 6 discusses and concludes the paper.

2. BACKGROUND

In this section, we will review some of the related work on time series kernels.

Dynamic time warping (DTW) tries to "warp" the time axis of one (or both) sequences to achieve a better alignment [1]. *DTW* has been successfully used in many applications. However, *DTW* can generate un-intuitive alignments by mapping a single point on one time series onto a large subsection of another time series, leading to inferior results [15]. A time series kernel based on global alignment, motivated by DTW, has been proposed in [7], with an efficient version presented in [6].

Autoregressive kernel [5] is another probabilistic kernel for time series. In autoregressive kernel, VAR model class of a given order is used to generate an infinite family of features from the time series. For a given time series $\boldsymbol{s}$, the likelihood profile $p_\theta(\boldsymbol{s})$ across all possible parameter setting (under a matrix normal-inverse Wishart prior $\omega(\boldsymbol{\theta})$) forms a representation of $\boldsymbol{s}$. Given two time series $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$, the kernel is defined as the dot product of the corresponding sequence representations:

$$\mathcal{K}_{AR}(\boldsymbol{s}_i, \boldsymbol{s}_j) = \int_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\boldsymbol{s}_i) p_{\boldsymbol{\theta}}(\boldsymbol{s}_j) \omega(d\boldsymbol{\theta}).$$

Fisher kernel [11] was proposed to combine the power of generative modelling with discriminant classifiers such as Support Vector Machines. It has been successfully used in numerous applications. *Fisher* kernel assumes that the generative model $p(\boldsymbol{s}|\boldsymbol{\theta})$ can explain all the data. The *Fisher* kernel maps each individual data into a vector in the gradient log-likelihood space specified by this generative model. The feature vector (*Fisher* score) $U_{\boldsymbol{s}}$ is the gradient of the log-likelihood of the generative model (fit on the data set) for the time series $\boldsymbol{s}$:

$$U_{\boldsymbol{s}} = \nabla_{\boldsymbol{\theta}} \log P(\boldsymbol{s}|\boldsymbol{\theta}).$$

The *Fisher* kernel is then defined as follows:

$$\mathcal{K}(\boldsymbol{s}_i, \boldsymbol{s}_j) = U_{\boldsymbol{s}_i}^T \mathcal{I}^{-1} U_{\boldsymbol{s}_j},$$

where $\mathcal{I}$ is the *Fisher* information matrix. As mentioned above, calculation of $\mathcal{I}^{-1}$ can be computationally expensive. A routinely used practical 'trick' is to use the identity matrix in place of $\mathcal{I}$ [23], which speeds up the computation at the cost of losing some important information [23].

3. RESERVOIR BASED KERNELS

In this paper we will introduce new time series kernels based on a general "temporal filter" implemented by Echo State Network (ESN) with a simple deterministically constructed reservoir architecture. Reservoir models [18] have been extensively shown to be able to successfully process and model time series of a surprisingly wide variety of types (from deeper memory deterministic chaotic systems, to shorter memory stochastic sequences) [22, 24].

The ESN reservoir model with N reservoir (state) units represents a parameterized input driven state space model

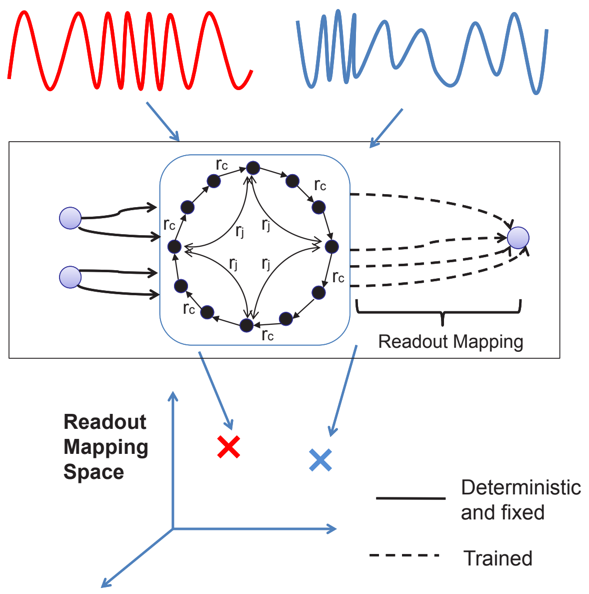


Figure 1: Illustration of the time series kernel in the deterministic reservoir model space. The first stage is to train the readout mapping of reservoir models using time series, i.e. generate individual points in the model space to represent time series. The second stage is to construct the kernel by investigating the model distance.

formulated as:

$$\boldsymbol{x}(t) = g(R\,\boldsymbol{x}(t-1) + V\boldsymbol{s}(t)), \tag{1}$$

$$\boldsymbol{y}(t) = W\boldsymbol{x}(t) + \boldsymbol{a} = f(\boldsymbol{x}(t)), \tag{2}$$

where $\boldsymbol{x}(t) = [x_1(t), \cdots, x_N(t)]^T \in \Re^N$ is the state vector of reservoir activations, $\boldsymbol{s}(t)$ is the input time series element at time t, R is the reservoir weight matrix $(N \times N)$, V is the input weight matrix $(N \times O)$ to the reservoir, O is the dimensionality of the time series, $g(\cdot)$ is element-wise application of the *tanh* transfer function and $\boldsymbol{y}(t)$ is the output of the linear readout from the reservoir. The state transition and output parts of the state space model are described by eqs. (1) and (2), respectively.

The main idea is that, provided the reservoir is able to represent a rich set of features of the input time series, the model-based representation of a particular time series $\boldsymbol{s}$ will be given by the linear readout mapping $f(\boldsymbol{x})$ (eq. (2)) operating on reservoir activations $\boldsymbol{x}$, specifically fitted to $\boldsymbol{s}$ on the next-item prediction task $\boldsymbol{y}(t) \sim \boldsymbol{s}(t+1)$ by minimizing the normalized mean square error (NMSE) between the model predictions $\boldsymbol{y}(t)$ and targets $\boldsymbol{s}(t+1)$. We will denote the readout $f(\boldsymbol{x})$ fitted to sequence $\boldsymbol{s}$ by $h(\boldsymbol{x}; \boldsymbol{s})$.

The kernel between a pair of time series $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$ will then be calculated using a model distance $d(h(\boldsymbol{x}; \boldsymbol{s}_i), h(\boldsymbol{x}; \boldsymbol{s}_j))$ between the corresponding readouts $h(\boldsymbol{x}; \boldsymbol{s}_i)$ and $h(\boldsymbol{x}; \boldsymbol{s}_j)$ from the same 'fixed' general reservoir (eq. (1)).

3.1 Deterministically Constructed Reservoir

This paper will focus on specific forms of ESN since they constitute one of the simplest, yet effective forms of RC. ESN has a "non-trainable" recurrent part ("the reservoir") (eq (1)) and a simple linear readout (eq (2)). Typically, the reservoir weights R and the input weights V to the reservoir are randomly generated so that the "Echo State Property" is satisfied. Loosely speaking, this means that the reservoir output would be independent of the initial conditions [12]. Training of ESN can be efficiently performed through linear regression. For more details we refer the interested reader to e.g. [18].

The downside of reservoir models is that their construction is largely driven by a series of randomized model building stages. Recently, Rodan et al. [22] proposed to use a simple deterministic constructed Cycle Reservoirs with regular Jumps (CRJ). This reservoir architecture has been shown to be comparable (or better) than the traditional ESN on a wide variety of time series modeling and prediction tasks [22]. In CRJ the reservoir nodes are connected in a uni-directional cycle with bi-directional shortcuts (jumps) (Figure 1). All cyclic reservoir weights r_c have the same value; all jumps r_j share the same weight and the input connections r_i have the same absolute value with an aperiodic sign pattern. This results in a sparse and deterministically constructed and simple coupling reservoir weight matrix R.

The reservoir forms a *fixed* non-linear high-dimensional non-autonomous dynamical system with fading memory that acts as a general temporal filter on top of which it is usually sufficient to train a linear readout mapping. As mentioned above, it is natural to represent individual time series by the linear readouts from the fixed dynamic filter that fits the series well.

3.1.1 *Distance in the Reservoir Model Space: Uniform State Distribution*

Using Euclidean metric on the readout parameters to calculate the distance between two readout mappings is not satisfying since one should be interested in the model distance in the function space of the readout models, rather than the distance between the model parameterizations.

We will use the L_2 distance in the model space [4], although our framework is general and can be applied to any appropriate function distance between the readouts. To simplify the notation, we will denote the readout $h(\boldsymbol{x}; \boldsymbol{s}_i)$ fitted to sequence $\boldsymbol{s}_i$ by $f_i(\boldsymbol{x})$. Consider two mappings $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$, $f_1, f_2 : \Re^N \to \Re^O$, where N is the number of reservoir units, O is the output dimensionality. Their L_2 distance is defined as:

$$L_2(f_1, f_2) = \left(\int_C \|f_1(\boldsymbol{x}) - f_2(\boldsymbol{x})\|^2 \, d\mu(\boldsymbol{x}) \right)^{1/2}, \tag{3}$$

where $\mu(x)$ is the probability density function on the input (reservoir) domain C. Recall that in (1) we use *tanh* transfer function and so $C = [-1, +1]^N$.

We will first assume that $\boldsymbol{x}$ is uniformly distributed. Later we will relax this assumption by considering non-uniform $\mu(\boldsymbol{x})$ to reflect the fact that state space activations $\boldsymbol{x}$ in the reservoir can follow a more complex distribution.

The readout model takes the form of an affine mapping:

$$f(\boldsymbol{x}) = W\boldsymbol{x} + \boldsymbol{a}, \tag{4}$$

where $\boldsymbol{x} = [x_1, \cdots, x_N]^T$ is the state vector, W is the parameter matrix $(O \times N)$ and $\boldsymbol{a} = [a_1, \cdots, a_o]^T$ is the bias vector.

Consider two readouts from the *same* reservoir

$$\begin{aligned} f_1(\boldsymbol{x}) &= W_1\boldsymbol{x} + \boldsymbol{a}_1, \\ f_2(\boldsymbol{x}) &= W_2\boldsymbol{x} + \boldsymbol{a}_2. \end{aligned}$$

Then,

$$L_2(f_1, f_2) = \left(\int_C \|W\boldsymbol{x}\|^2 + 2\boldsymbol{a}^T W\boldsymbol{x} + \|\boldsymbol{a}\|^2 \, d\boldsymbol{x} \right)^{1/2}$$

where $W = W_1 - W_2$, and $\boldsymbol{a} = \boldsymbol{a}_1 - \boldsymbol{a}_2$.

Note that since $C = [-1, 1]^N$, for any fixed $\boldsymbol{a}$ and W

$$\int_C \boldsymbol{a}^T W\boldsymbol{x} \; d\boldsymbol{x} = 0.$$

Therefore, it can be shown that

$$L_2(f_1, f_2) = \left(\frac{2^N}{3} \sum_{j=1}^{N} \sum_{i=1}^{O} w_{i,j}^2 + 2^N \|\boldsymbol{a}\|^2 \right)^{1/2} \qquad (5)$$

where $\boldsymbol{w}_i^T$ is the i-th row of W, $w_{i,j}$ is the (i, j)-th element of W.

Scaling of the squared model distance ($L_2^2(f_1, f_2)$) by 2^{-N} we obtain

$$\frac{1}{3} \sum_{j=1}^{N} \sum_{i=1}^{O} w_{i,j}^2 + \|\boldsymbol{a}\|^2 ,$$

which differs from the squared Euclidean distance on the readout parameters

$$\sum_{j=1}^{N} \sum_{i=1}^{O} w_{i,j}^2 + \|\boldsymbol{a}\|^2 ,$$

by the factor 1/3 applied to the differences in the linear part W of the affine readouts. Hence, more importance is given to the 'offset' than 'orientation' of the readout mapping.

3.2 Non-uniform State Distribution

In the above, we assumed that the distribution of reservoir states $\boldsymbol{x}$ is uniform in C. As mentioned before, it is likely that the state distribution $\mu(\boldsymbol{x})$ will be non-uniform. We will introduce two approaches for allowing general $\mu(\boldsymbol{x})$ - modelling of μ by a mixture of Gaussians and numerical approximation of the integral (3) by sampling using bootstrapped input series.

For non-uniform state distribution $\mu(\boldsymbol{x})$, a K-component Gaussian mixture model can be employed to approximate the distribution:

$$\begin{aligned} \mu(\boldsymbol{x}) &= \sum_{k=1}^{K} \alpha_k \; \mu_k(\boldsymbol{x}|\boldsymbol{\eta}_k, \Sigma_k), \\ \mu_k(\boldsymbol{x}|\boldsymbol{\eta}_k, \Sigma_k) &= \frac{\exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\eta}_k)^T \Sigma_k^{-1} (\boldsymbol{x} - \boldsymbol{\eta}_k)\right)}{(2\pi)^{N/2} |\Sigma_k|^{1/2}}, \end{aligned}$$

where α_k are mixture coefficients with $\sum_{k=1}^{K} \alpha_k = 1$.

Then, the distance $L_2(f_1, f_2)$ can be obtained as follows:

$$L_2(f_1, f_2) = \left(\int_C \boldsymbol{x}^T W^T W \boldsymbol{x} + 2\boldsymbol{a}^T W\boldsymbol{x} + \boldsymbol{a}^T \boldsymbol{a} \; d\mu(\boldsymbol{x}) \right)^{1/2}$$

According to [20] (page 42), for a Gaussian variable $\boldsymbol{x} \sim N(\boldsymbol{\eta}, \Sigma)$,

$$E(\boldsymbol{x}^T W^T W \boldsymbol{x}) = trace(W^T W \Sigma) + \boldsymbol{\eta}^T W^T W \boldsymbol{\eta}.$$

Therefore, the distance can be obtained as follows:

$$L_2^2(f_1, f_2) = \sum_{k=1}^{K} \alpha_k \left\{ \begin{array}{c} trace(W^T W \Sigma_k) + \boldsymbol{a}^T \boldsymbol{a} \\ + \; \boldsymbol{\eta}_k^T W^T W \boldsymbol{\eta}_k + 2\boldsymbol{a}^T W \boldsymbol{\eta}_k \end{array} \right\}. \qquad (6)$$

We employed the mixture model construction proposed by Figueiredo et. al [8] that automatically selects the appropriate number of mixture components in a top-down manner.

Alternatively, the integral can be numerically approximated by using reservoir activations collected while processing the input time series. Assume that for a given time series $\boldsymbol{s}$, after the initial wash-out [12], m state activations are collected $\boldsymbol{x}(1), \cdots, \boldsymbol{x}(m)$. Then,

$$L_2^2(f_1, f_2) \approx \frac{1}{m} \sum_{t=1}^{m} \|f_1(\boldsymbol{x}(t)) - f_2(\boldsymbol{x}(t))\|^2 . \qquad (7)$$

However, in some applications the length of the time series is not sufficient to yield a good approximation. We therefore adopted the circular block bootstrap for time series [17] to construct sufficiently long input series. The block length in bootstrapping was automatically determined following [21].

In the above, the function distance between readout mappings of reservoir models is formulated. Therefore, the three kernels can be defined as follows:

$$\mathcal{K}(\boldsymbol{s}_i, \boldsymbol{s}_j) = \exp\left\{-\gamma \cdot L_2^2(f_i, f_j)\right\},$$

where $L_2^2(f_i, f_j)$ can be Equations (5), (6) and (7) as reservoir kernel (*RV*), Gaussian mixture model based reservoir kernel (*GMMRV*), and sampling based reservoir kernel (*SamplingRV*). The main algorithm is summarized below:

Algorithm 1 Model based Kernel Algorithm (*RV, GMMRV, SamplingRV*)

```
1: Input: Set of sequences $\boldsymbol{s}_1, \cdots, \boldsymbol{s}_M$; parameters (number of reservoir units $N$; CRJ weights $(r_c, r_j, r_i)$; ridge regression parameter $\lambda$; kernel scale parameter $\gamma$;
2: Output: Kernel (Gram) matrix $\mathcal{K}$.
3: for each time series $\boldsymbol{s}_i$, $i = 1, \cdots, M$ do
4:     Drive the reservoir state evolution with the input sequence $\boldsymbol{s}_i$ (eq. (1)).
5:     Fit the linear readout $f_i$ using ridge regression for the next item prediction task on $\boldsymbol{s}_i$ (eq. (2)).
6: end for
7: Calculate the pairwise model distance matrix $L_2(f_i, f_j)$ $i, j = 1, \cdots, M$, via eqs. (5) (RV), (6) (GMMRV), or (7) (SamplingRV) - Sections 3.1 and 3.2.
8: Calculate the kernel matrix as $\mathcal{K}(\boldsymbol{s}_i, \boldsymbol{s}_j) = exp\{-\gamma \cdot L_2^2(f_i, f_j)\}$.
```

3.3 Fisher Kernel Based on Reservoir Model

Besides the model distance based kernels introduced above, we also considered the *Fisher* kernel obtained with the reservoir model (*FisherRV*).

Endowing the readout with a noise model yields a generative time series model of the form:

$$\begin{aligned} \boldsymbol{x}(t) &= g(R\, \boldsymbol{x}(t-1) + V\boldsymbol{s}(t)), \\ \boldsymbol{s}(t+1) &= W\boldsymbol{x}(t) + \boldsymbol{a} + \boldsymbol{\varepsilon}(t), \end{aligned}$$

Assume the i.i.d. noise model $\boldsymbol{\varepsilon}(t)$ follows a Gaussian distribution,

$$\boldsymbol{\varepsilon}(t) = \mathcal{N}(0, \sigma^2 I).$$

Table 1: Parameters for all kernels. γ is the parameter in RBF function, ξ in *AR* kernel is the weight of the negative definite kernel [5], p is the order of the vector autoregressive model, *state* is the number of states for HMM in *Fisher* kernel, λ is the ridge regression parameter.

Kernel	Parameters	Parameter range
DTW	γ	$\gamma \in \{10^{-6}, 10^{-5}, \cdots, 10^{1}\}$,
AR	γ, ξ, p	$\gamma \in \{10^{-6}, 10^{-5}, \cdots, 10^{1}\}$, $\xi \in \{0.1, 0.2, \cdots, 0.9\}$, $p \in \{1, 2, \cdots, 10\}$
Fisher	*state*	*state* $\in \{1, 2, \cdots, 10\}$
RV, *FisherRV*, *GMMRV*, *SamplingRV*	γ, λ	$\gamma \in \{10^{-6}, 10^{-5}, \cdots, 10^{1}\}$, $\lambda \in \{10^{-5}, 10^{-4}, \cdots, 10^{1}\}$

Then,

$$\begin{aligned} & P((\boldsymbol{s}(t+1) \mid \boldsymbol{s}(1..t)) = P((\boldsymbol{s}(t+1) \mid \boldsymbol{x}(t)) \\ = \;& (2\pi\sigma^2)^{-O/2} \exp\left\{-\frac{\|\boldsymbol{s}(t+1) - W\boldsymbol{x}(t) - \boldsymbol{a}\|^2}{2\sigma^2}\right\}, \end{aligned}$$

where $\boldsymbol{s}(1..t)$ denotes the time series $\boldsymbol{s}(1), \boldsymbol{s}(2), \cdots, \boldsymbol{s}(t)$.

Slightly abusing mathematical notation, the model likelihood $p(\boldsymbol{s}(1..\ell))$ given the time series $\boldsymbol{s}$ of length ℓ can be written as follows:

$$\begin{aligned} p(\boldsymbol{s}(1..\ell)) &= \prod_{t=1}^{\ell} P(\boldsymbol{s}(t) \mid \boldsymbol{s}(1\cdots t-1)) \\ &= \prod_{t=1}^{\ell} (2\pi\sigma^2)^{-O/2} \exp\left\{-\frac{\|\boldsymbol{s}(t) - W\boldsymbol{x}(t-1) - \boldsymbol{a}\|^2}{2\sigma^2}\right\}. \end{aligned}$$

Therefore, the partial derivative of log likelihood $\log p(\boldsymbol{s}(1..\ell))$ can be obtained as

$$\begin{aligned} U &= \frac{\partial \log p(\boldsymbol{s}(1..\ell))}{\partial W} \\ &= \sum_{t=1}^{\ell} \frac{(\boldsymbol{s}(t) - \boldsymbol{a})\; \boldsymbol{x}(t-1)^T - W\boldsymbol{x}(t-1)\boldsymbol{x}(t-1)^T}{\sigma^2}. \end{aligned}$$

Note that the partial derivative U is an $(O \times N)$ matrix. The "practical" *Fisher* kernel for two time series $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$ with scores U_i and U_j, respectively, can be formulated as

$$\mathcal{K}(\boldsymbol{s}_i, \boldsymbol{s}_j) = \sum_{o=1}^{O}\sum_{n=1}^{N} (U_i \circ U_j)_{o,n},$$

where $\circ$ is Hadamard (element-wise) product. In practice, the noise variance σ^2 can be estimated from the original time series and the output of the fitted readout model.

4. EXPERIMENTAL STUDIES

This section presents experimental results of the proposed kernels, *RV*, *GMMRV*, *SamplingRV*, *FisherRV*, and other existing time series kernels, including autoregressive (*AR*) kernel, *Fisher* kernel with hidden Markov models (*Fisher*), and dynamic time warping based kernel (*DTW*).

All hyperparameters, such as the kernel width γ and order p in the *AR* kernel, number of hidden states in the HMM based *Fisher* kernel etc. have been set by 5-fold cross-validation on the training set. The search ranges for parameters of each algorithm are detailed in Table 1.

In the reservoir based kernels, we used a fixed topology reservoir (cycle with jumps) [22] for all data sets: $N = 100$, 15 jumps. The cycle weight r_c, jump weight r_j, input weight r_i and readout were obtained on the training set. The readout mapping was trained via ridge regression (hyperparameter λ tuned via cross-validation). To evaluate the readout model distance in the *SamplingRV* kernel, except for long time series in the PEMS data set (Section 5), the bootstrapped time series were 5 times longer than the original ones[1].

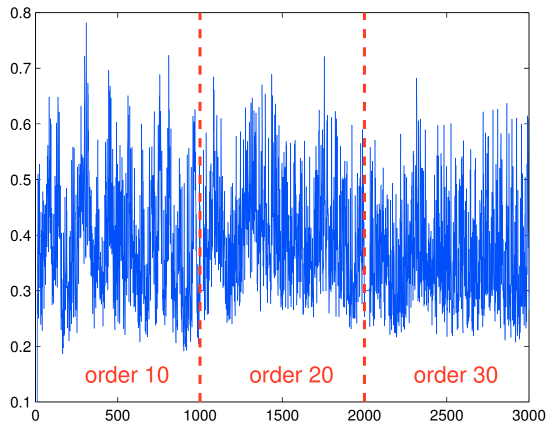


Figure 2: Illustration of three NARMA sequences with different orders (10, 20 and 30).

The implementation of *AR* kernel was obtained from Marco Cuturi's website[2]. *Fisher* kernel was obtained Maaten's website[3].

We employ a well-known, widely accepted and used implementation of SVM – LIBSVM [3]. In LIBSVM, we use cross validation to tune the regularization parameter C. After model selection using cross-validation on the training set, the selected model class representatives were retrained on the whole training set and were evaluated on the test set. Multi class classification is performed via the one-against-one strategy (default in LIBSVM).

4.1 Synthetic Data

We employed three NARMA time series models of orders 10, 20 and 30, given by:

$$s(t+1) = 0.3s(t) + 0.05s(t)\sum_{i=0}^{9} s(t-i) + 1.5u(t-9)u(t) + 0.1,$$

$$\begin{aligned} s(t+1) \;&=\; \tanh(0.3s(t) + 0.05s(t)\sum_{i=0}^{19} s(t-i) + \\ &\quad +1.5u(t-19)u(t) + 0.01) + 0.2, \end{aligned}$$

[1]$m = 5\;|\boldsymbol{s}|$, where $|\boldsymbol{s}|$ indicates the length of the time series.
[2]`http://www.iip.ist.i.kyoto-u.ac.jp/member/cuturi/AR.html`
[3]`http://homepage.tudelft.nl/19j49/Software.html`

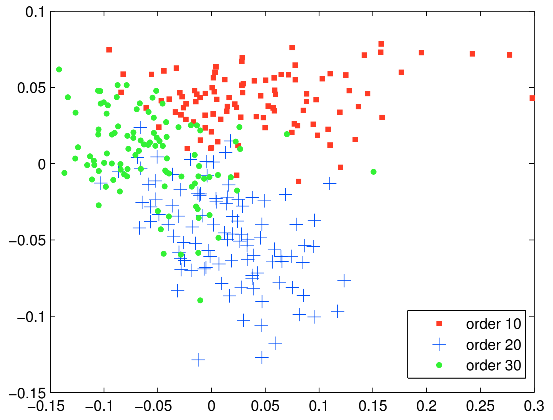


Figure 3: Illustration of MDS on the model distance among reservoir weights

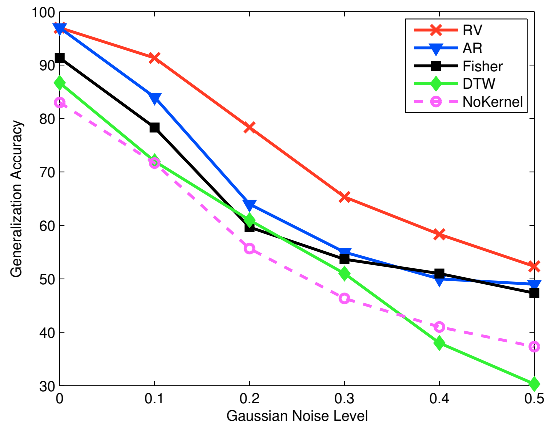


Figure 4: Illustration of the performance of compared kernels with different noise levels.

$$s(t+1) = 0.2s(t)+0.004s(t)\sum_{i=0}^{29} s(t-i)+1.5u(t-29)u(t)+0.201,$$

where $s(t)$ is the output at time t, $u(t)$ is the input at time t. The inputs $u(t)$ form an i.i.d stream generated uniformly in the interval $[0, 0.5)$. We use the same input stream for generating the three long NARMA time series (60,000 items), one for each order. The three sequences are illustrated in Figure 2. The time series are challenging due to non-linearity and long memory.

For each order, the series of 60,000 numbers is partitioned into 200 non-overlapping time series of length 300. The first 100 time series for each order are used as training set, and the other 100 time series form the test set.

As apparent from Figure 2, distinguishing the three NARMA models using the original time series may be challenging. However, when viewing the time series through the model space of fitted reservoir models, the three time series classes become separated, as illustrated in Figure 3 showing 2-dimensional multi-dimensional scaling[4] representation of the pair-wise readout model distances.

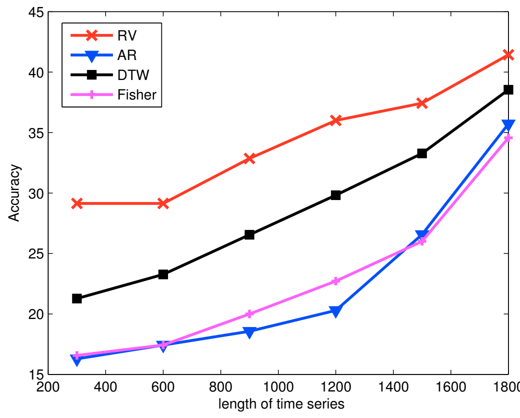
[4]Multidimensional scaling (MDS) aims to preserve the pair-wise distance between points, which is suitable to preserve the *model distance* for visualization.

In order to study robustness of the kernels we corrupt the time series with additive Gaussian noise (zero mean, standard derivation varies in [0.1,0.5]). Figure 4 shows the test set classification accuracy against the noise level. As a baseline we also include results by SVM operating on the time series directly (300-dimensional inputs) - *NoKernel*. The *RV* reservoir based kernel outperforms the baseline and the other time series kernels.

4.2 Benchmark Data

Table 2: Description of the data sets

Dataset	Length	Classes	Train	Test
Symbols	398	6	25	995
OSULeaf	427	6	200	242
Oliveoil	570	4	30	30
Lighting2	637	2	60	61
Beef	470	6	30	30
Car	576	4	60	60
Fish	463	8	175	175
Coffee	286	2	28	28
Adiac	176	37	390	391



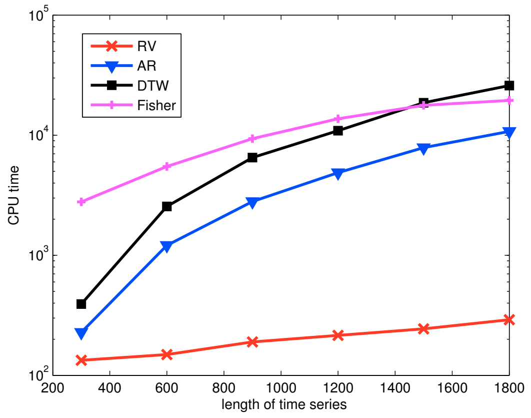


Figure 5: Comparison of generalization accuracy (top) and CPU time (bottom) in seconds of *RV*, *AR*, *DTW* and *Fisher* kernels on InlineSkate data set.

We used 9 data sets from UCR Time Series Repository [14]. Each data set has already been split into training and test sets (see Table 2).

Table 3: Comparison of *DTW*, *AR*, *Fisher* (with hidden Markov models), *RV*, *FisherRV*, *GMMRV*, and *SamplingRV* kernels on nine benchmark data sets by accuracy. The best performance for each data set has been boldfaced.

Dataset	*DTW*	*AR*	*Fisher*	*RV*	*FisherRV*	*GMMRV*	*SamplingRV*
Symbols	94.77	91.15	94.42	**98.08**	95.96	97.31	95.77
OSULeaf	**74.79**	56.61	54.96	69.83	64.59	56.55	63.33
Oliveoil	83.33	73.33	56.67	86.67	83.33	84.00	**90.00**
Lighting2	64.10	77.05	64.10	77.05	75.41	78.69	**80.33**
Beef	66.67	78.69	58.00	80.00	68.00	79.67	**86.67**
Car	58.85	60.00	65.00	76.67	72.33	78.33	**86.67**
Fish	69.86	60.61	57.14	79.00	74.29	78.00	**85.71**
Coffee	85.71	**100.00**	81.43	**100.00**	92.86	96.43	**100.00**
Adiac	65.47	64.45	68.03	72.63	71.61	74.94	**76.73**

Table 4: CPU Time (in seconds) of *DTW*, *AR*, *Fisher* (with hidden Markov models), *RV*, *FisherRV*, *GMMRV*, and *SamplingRV* kernels on nine benchmark data sets.

Dataset	*DTW*	*AR*	*Fisher*	*RV*	*FisherRV*	*GMMRV*	*SamplingRV*
Symbols	1,318	2,868	2,331	202	236	374	808
OSULeaf	6,030	1,375	3,264	98	111	186	447
Oliveoil	295	113	832	11	19	27	43
Lighting2	918	151	1,143	33	46	61	95
Beef	107	54	87	10	17	23	40
Car	679	442	902	27	42	50	84
Fish	3,353	495	1,998	81	96	159	286
Coffee	21	25	145	3	3	7	19
Adiac	550	8131	1,122	201	213	394	699

Table 3 reports performance of the time series kernels on the benchmark data in terms of test set classification accuracy. *SamplingRV* kernel outperforms the other kernels on 7 data sets; *RV* is superior on 2 data sets and *DTW* outperforms the other kernels on 1 data set. In terms of computation time[5], the reservoir kernels are clearly the most efficient. Table 4 shows the average CPU time taken to evaluate the kernels in seconds[6]. *SamplingRV* kernel is obviously the most expensive among the reservoir kernels. Still, it is faster than its state-of-art competitors.

To further compare the computational effectiveness of the kernels, a relatively large data set, *InlineSkate* from UCR time series repository, has been employed. The data set contains 650 time series (100 training, 550 test) of length 1882, belonging to 7 classes. The influence of time series length on the classification performance and computational complexity was studied by considering from each training time series only the first ℓ elements, with ℓ growing from 300 to 1800 in increments of 300. The resulting accuracy and CPU times are shown in Figure 5. Relatively to the other kernels, the reservoir *RV* kernel has the lowest computational cost, while achieving competitive performance.

4.3 Multivariate Time Series

Data sets used so far involved univariate time series. In this section, we perform classification on three multivariate time series - Brazilian sign language (*Libras*), *handwritten* characters and Australian language of signs (*AUSLAN*). Unlike the other data sets, the *handwritten* characters and *AUSLAN* data sets contain time series of variable length. Following [5] (previous *AR* kernel study) we split the data sets into training and test sets as detailed in Table 5.

Table 5: Summary of multivariate (variable length) time series classification problems.

Dataset	dim	length	classes	train	test
Libras	2	45	15	360	585
handwritten	3	60-182	20	600	2258
AUSLAN	22	45-136	95	600	1865

The results are shown in Figure 6. *SamplingRV* is superior on all three data sets. *RV* kernel is outperformed by *DTW* and *AR* kernels on *Libras* and *AUSLAN* data sets, respectively. In terms of CPU time, *RV* kernel usually uses the least and *AR* consumes the most computation time.

5. ON-LINE RESERVOIR KERNEL

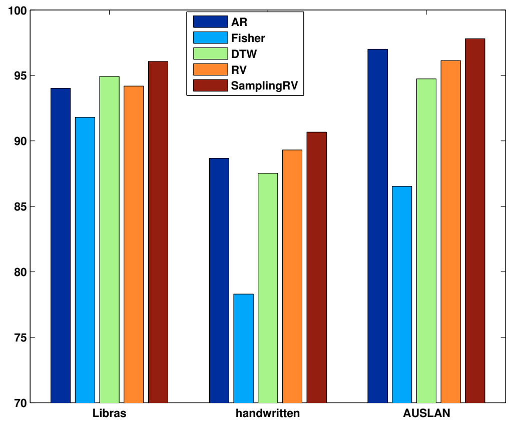
Reservoir readouts can be trained in an on-line fashion using Recursive Least Squares (RLS). In RLS, the readout weights W are recursively updated at every time step t:

$$\begin{aligned} \boldsymbol{z}(t) &= \frac{\Lambda(t-1)\ \boldsymbol{x}(t)}{\boldsymbol{x}^T(t)\ \Lambda(t-1)\ \boldsymbol{x}(t) + \varsigma} \\ \Lambda(t) &= \varsigma^{-1}(\Lambda(t-1) - \boldsymbol{z}(t)\ \boldsymbol{x}^T(t)\ \Lambda(t-1)) \\ W(t) &= W(t-1) + [\boldsymbol{s}(t+1) - \boldsymbol{y}(t)]\ \boldsymbol{z}^T(t), \end{aligned}$$

where $\boldsymbol{z}(t)$ stands for the innovation vector; $\boldsymbol{s}(t+1)$ and $\boldsymbol{y}(t)$ correspond to the desired (next-item prediction) and calculated (readout) output; $\Lambda(t)$ is the error covariance matrix.

[5]The computational environment is Windows XP with Intel Core 2 Duo 1.66G CPU and 4G RAM.

[6]We do not record the cross validation time for SVM.



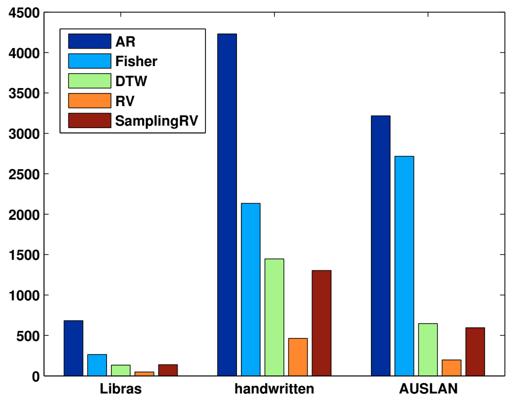


Figure 6: Comparison of generalization accuracy (top) and CPU time (bottom) in seconds of *AR*, *Fisher*, *DTW*, *RV* and *SamplingRV* kernels on 3 multivariate time series.

'Forgetting parameter' $0 < \varsigma < 1$ is usually set to a value close to 1.0. In this work ς is set by cross validation.

This enables us to construct and refine reservoir kernels on-line, as more and more data become available. This can be particularly convenient in situations where individual items to be classified (time series) are not fixed, but appear in an on-line manner.

We illustrate this approach on a set of long series *PEMS-SF* (UCI machine learning repository) with 440 time series of length 138,672. The data reports the occupancy rate of different car lanes of San Francisco freeways within 15 months. The generalization performance of on-line *RV* kernel is reported in Figure 7. As expected, the generalization improves monotonically with increasing amount of data. On full data *RV* kernel achieves 86.13% accuracy. This compares favorably with the best reported performance levels (82% $\sim$ 83%) [5] among a variety of time series kernels, such as *AR*, global alignment kernel [7], splines smoothing kernel [16] and Bag of vectors kernel [10].

6. DISCUSSION AND CONCLUSION

In this paper efficient kernels have been proposed to tackle the challenges in time series classification through kernel machines. Instead of constructing the kernel directly in the original data space, this paper introduces a "kernel in the deterministically constructed reservoir model space" that represents each time series as a reservoir model with the common dynamic part.

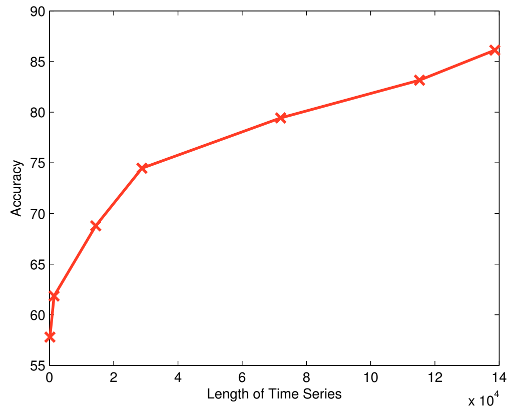


Figure 7: Generalization accuracy of on-line *RV* kernel on PEMS time series.

We demonstrated the application of the distance definition in the (function) model space of linear readout models. The model distance is different from the Euclidean distance of the readout parameters, indicating that more importance is given to the 'offset' than 'orientation' of the readout mapping. We also estimated the model distance by using either sampling methods or a Gaussian mixture model when the reservoir state distribution is non-uniform.

The proposed kernels were compared with other competitors on synthetic and benchmark data sets. The results confirm the effectiveness of reservoir based kernels. The on-line reservoir kernels proposed in Section 5 can process extremely long time series efficiently.

In general, the closed form simple reservoir (*RV*) kernel is the most efficient[7]. However, it is obtained under the (rather unrealistic) assumption of uniform state distribution and the tolerable increase in computational cost by the *SamplingRV* kernel is well offset by the increase in the classification accuracy. The *GMMRV* kernel can also be analytically obtained via approximating the state distribution by a Gaussian mixture. Of course, the quality of this kernel depends on how well the state distribution is captured by the Gaussian mixture model used.

It is interesting that the *Fisher* kernel based on the reservoir model achieves better performance than the *Fisher* kernel based on the HMM model with continuous (Gaussian distributed) emissions. The principal difference between the reservoir model and HMM is that in the reservoir model the state space is infinite (uncountable) with deterministic input-driven dynamics. In HMM the state space is finite and latent, with probabilistic state transitions.

In conclusion, reservoir based time series kernels can achieve superior performance in terms of both generalization accuracy and computational time, without the need for explicit specification of the parameterized model class for the time series data. This is potentially of great benefit in cases of very large data sets of long time series where the underlying parametric model is unknown. Reservoir kernels stand and fall on the ability of the particular dynamic reservoir to

[7]It is worth noting that there also exist fast implementations of non-kernelized variations of DACO and global alignment kernels.

generate a rich pool of dynamical features sufficiently representing the variety of time series occurring in a given task. If the echo state property - a cornerstone of reservoir modelling - is not an appropriate modelling assumption, the reservoir kernels cannot be expected to perform well. However, as has been demonstrated numerous times, for most real-world data the fading memory assumption (encapsulated in the echo state property) is appropriate.

7. ACKNOWLEDGMENTS

This work was supported by the European Union Seventh Framework Programme under Grant Agreement INSFOICT-270428 on "Making Sense of Nonsense (iSense)". The work of H. Chen was supported in part by the National Natural Science Foundation of China under Grants 61203292 and 61311130140, and the One Thousand Young Talents Program. The work of P. Tino was supported by the Biotechnology and Biological Sciences Research Council under Grant H012508/1. The work of X. Yao was supported by a Royal Society Wolfson Research Merit Award.

8. REFERENCES

[1] D. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370, 1994.

[2] A. B. Chan and N. Vasconcelos. Probabilistic kernels for the classification of auto-regressive visual process. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'05)*, pages 846–851, 2005.

[3] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27, 2011.

[4] H. Chen, P. Tino, A. Rodan, and X. Yao. Learning in the model space for cognitive fault diagnosis. *IEEE Transactions on Neural Networks and Learning Systems*, 2013. DOI: 10.1109/TNNLS.2013.2256797.

[5] M. Curturi and A. Doucet. Autoregressive kernels for time series. *ArXiv:1101.0673*, 2011.

[6] M. Cuturi. Fast global alignment kernels. In *Proceedings of the 28th International Conference on Machine Learning (ICML'11)*, pages 929–936, 2011.

[7] M. Cuturi, J.-P. Vert, O. Birkenes, and T. Matsui. A kernel for time series based on global alignments. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'07)*, volume 2, pages 413–416, 2007.

[8] M. A. T. Figueiredo and A. K. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):381–396, 2002.

[9] A. Gaidon, Z. Harchaoui, C. Schmid, et al. A time series kernel for action recognition. In *British Machine Vision Conference*, pages 63.1–63.11, 2011.

[10] M. M. Hein, M. Hein, and O. Bousquet. Hilbertian metrics and positive definite kernels on probability. In *Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 136–143, 2005.

[11] T. Jaakkola, M. Diekhans, and D. Haussler. Using the fisher kernel method to detect remote protein homologies. In *Proceedings of the International Conference on Intelligent Systems for Molecular Biology*, pages 149–158, 1999.

[12] H. Jaeger. The echo state approach to analysing and training recurrent neural networks. Technical report, German National Research Center for Information Technology, 2001.

[13] T. Jebara, R. Kondor, and A. Howard. Probability product kernels. *Journal of Machine Learning Research*, 5:819–844, 2004.

[14] E. Keogh, Q. Zhu, B. Hu, Y. Hao, X. Xi, L. Wei, and C. A. Ratanamahatana. The ucr time series classification/clustering, 2011. `http://www.cs.ucr.edu/~eamonn/time_series_data/`.

[15] E. J. Keogh and M. J. Pazzani. Derivative dynamic time warping. In *SIAM International Conference on Data Mining (SDM)*, 2001.

[16] K. Kumara, R. Agrawal, and C. Bhattacharyya. A large margin approach for writer independent online handwriting classification. *Pattern Recognition Letters*, 29(7):933–937, 2008.

[17] S. N. Lahiri. Theoretical comparisons of block bootstrap methods. *The Annals of Statistics*, 27(1):386–404, 1999.

[18] M. Lukoševičius and H. Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

[19] P. J. Moreno, P. P.Ho, and N. Vasconcelos. A Kullback-Leibler divergence based kernel for SVM classification in multimedia application. In *Advances in Neural Information Processing Systems*, 2004.

[20] K. Petersen and M. Pedersen. The matrix cookbook. Technical report, Technical University of Denmark, 2008.

[21] D. N. Politis and H. White. Automatic block-length selection for the dependent bootstrap. *Econometric Reviews*, 23(1):53–70, 2004.

[22] A. Rodan and P. Tiňo. Simple deterministically constructed cycle reservoirs with regular jumps. *Neural Computation*, 24(7):1822–1852, 2012.

[23] J. Shawe-Taylor and N. Cristinanini. *Kernel methods for patten analysis*. Cambridge University press, 2004.

[24] P. Tino, I. Farkaš, and J. van Mourik. Dynamics and topographic organization of recursive self-organizing maps. *Neural Computation*, 18(10):2529–2567, 2006.

[25] L. Van der Maaten. Learning discriminative fisher kernels. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.