Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)

Model Metric Co-Learning for Time Series Classification

Huanhuan Chen
School of Computer Science, Univ. of Sci. & Tech. of China
Hefei, Anhui, China
hchen@ustc.edu.cn

Fengzhen Tang, Peter Tino
School of Computer Science
University of Birmingham
Birmingham, B15 2TT, UK
fxt126,pxt@cs.bham.ac.uk

Anthony G Cohn
School of Computing
University of Leeds
Leeds, LS2 9JT, UK
A.G.Cohn@leeds.ac.uk

Xin Yao
School of Computer Science
University of Birmingham
Birmingham, B15 2TT, UK
X.Yao@cs.bham.ac.uk

Abstract

We present a novel model-metric co-learning (MMCL) methodology for sequence classification which learns in the model space – each data item (sequence) is represented by a predictive model from a carefully designed model class. MMCL learning encourages sequences from the same class to be represented by 'close' model representations, well separated from those for different classes. Existing approaches to the problem either fit a single model to all the data, or a (predominantly linear) model on each sequence. We introduce a novel hybrid approach spanning the two extremes. The model class we use is a special form of adaptive high-dimensional non-linear state space model with a highly constrained and simple dynamic part. The dynamic part is identical for all data items and acts as a temporal filter providing a rich pool of dynamic features that can be selectively extracted by individual (static) linear readout mappings representing the sequences. Alongside learning the dynamic part, we also learn the global metric in the model readout space. Experiments on synthetic and benchmark data sets confirm the effectiveness of the algorithm compared to a variety of alternative methods.

1 Introduction

Classification of structured data, such as time series, is an important problem in many application domains [Sha and Saul, 2007; Ghanem and Ahuja, 2010]. One popular approach, termed Dynamic time warping (DTW), measures 'similarity' between two sequences of variable-length by 'warping' the time axis of one (or both) sequences to achieve a better alignment [Berndt and Clifford, 1994]. *DTW* can sometimes generate unintuitive alignments (e.g. mapping a single point in one time series onto a large subsection of another time series), leading to inferior results [Keogh and Pazzani, 2001]. Since the *DTW* similarity measure is not essentially positive definite it cannot be directly used in a kernel machine. A time series kernel based on global alignment, motivated by DTW, has been proposed in [Cuturi *et al.*, 2007], with an efficient version presented in [Cuturi, 2011].

In this context, a new trend has emerged in the machine learning community, using models that are fitted on parts of data as more stable and parsimonious data representations. Learning is then performed directly in the model space, instead of the original data space. For example, Brodersen et al. [Brodersen *et al.*, 2011] used a generative model of brain imaging data to represent fMRI measurements of different subjects through subject-specific models. They subsequently employed SVM on the models' parameters to distinguish aphasic patients from healthy controls.

Prior to these developments, some aspects of learning in the model space occurred in different forms in the machine learning community. For example, the use of generative kernels for classification (e.g. the P-kernel [Shawe-Taylor and Cristinanini, 2004] or *Fisher* kernel [Jaakkola and Haussler, 1999]) can be viewed as a form of learning in a model induced feature space (see e.g. [Jebara *et al.*, 2004; Bosch *et al.*, 2008]). The *Fisher* kernel maps individual time series into score functions of a single generative model that is assumed to be able to 'explain' most of the data. The generative model employed in a *Fisher* kernel is often a Hidden Markov Model (HMM) with a fixed number of states. In some situations the assumption of the particular generative model underlying the data can be too strong. In addition, *Fisher* kernels are computationally expensive because of the calculation of metric tensor (inverse of the *Fisher* information matrix) in the tangent space of the generative model manifold. The often used 'practical' *Fisher* kernel replaces the metric tensor with an identity matrix. This can result in a loss of valuable information in the data [Van der Maaten, 2011]. Yet another approach based on an *Autoregressive kernel* [Cuturi and Doucet, 2011] uses a vector autoregressive (VAR) model of a given order to generate an infinite family of features from the time series. Each time series is represented by its likelihood profile of VAR across all possible parameter settings (under a matrix normal-inverse Wishart prior). The kernel is then defined as the dot product of the corresponding likelihoods.

In the context of time series classification, several approaches have used a generative probabilistic model of time series to define a time series kernel through models corresponding to each of the sequences (as opposed to a single model for all sequences employed in the *Fisher* kernel), e.g. the probability product kernel [Jebara *et al.*, 2004] or Kullback-Leibler (KL) divergence based kernels [Moreno *et al.*, 2004]. [Chan and Vasconcelos, 2005] proposed a probabilistic kernel based on KL divergence between spatio-temporal linear systems fitted on video sequences. [Saisan *et al.*, 2001], while also using linear systems to model individual image sequences, suggested three novel distances between the models, in particular: principal angles between specific subspaces derived from the models, Martin and geodesic distances. Binet-Cauchy kernels [Vishwanathan and Smola, 2006] formed a general family of kernels on linear systems subsuming several existing time series kernels. Vishwanathan et al. derived explicit formulae for efficient kernel computation. Staying within the linear dynamics domain, [Chan and Vasconcelos, 2007] allowed for a non-linear readout (observation function) from the (hidden) states.

To the best of our knowledge, previous model-based time series kernel formulations have been based on linear dynamical systems, predominantly with linear observation functions. [Vishwanathan and Smola, 2006] discussed extension to non-linear dynamics, but only for the case of fully observable states. Direct extension of the previous work to non-linear dynamics may be highly non-trivial and computationally expensive, perhaps requiring approximation techniques.

Existing approaches to model based sequence classification either use a single model fitted on the full data/individual classes, or fit a full linear dynamic model on each sequence. Models fitted on the full data/classes can be more complex than those fitted on individual sequences. The price is to be paid is the potential inadequacy of such a model to capture individual variations among the data items. Hence on the other hand, individual sequence models have to be relatively simple (e.g. with linear dynamics) to keep the model estimation stable. We propose a new hybrid approach spanning the two extremes in an attempt to keep the best of both worlds. As the core model class used to represent individual sequences we use a special form of adaptive high-dimensional non-linear state space model with a highly constrained dynamic part. The dynamic part is the same for all data items and acts as a temporal filter providing a rich pool of dynamic features that can be selectively extracted by individual (static) linear readout mappings representing the sequences. Alongside learning the dynamic part, we also learn the global metric in the readout model space. The overall goal is to adapt the shared dynamical system and metric tensor on the readout maps (standing for the sequences) so that sequences from the same class are represented by 'close' readouts, while readouts of sequences from different classes are well-separated. We term our methodology *model-metric co-learning* (MMCL).

The MMCL framework for sequence classification builds on ideas of model based representation for sequences [Chen *et al.*, 2013; Chan and Vasconcelos, 2005] and discriminative learning [Brodersen *et al.*, 2011; Van der Maaten, 2011] but differs mainly in three aspects: First, the final representation of the sequences is a linear readout mapping, but the full underlying model is non-linear dynamic system. In this way our methodology reduces the computational demands without the loss of the computational ability of non-linear models. Second, it treats the model parameters adaptation and model distance definition jointly rather than adapting the model parameters first and defining the model distance afterwards in two independent steps. Third, it has a similar motivation as other discriminative kernels [Van der Maaten, 2011], but differs in its goal of utilizing models that both represent the time series well and at the same time best separate the time series classes.

The paper has the following organization: We introduce our methodology in the following section, then report the experimental results and analysis. Finally, the last section discusses and concludes the paper.

2 Model Metric Co-learning

As the core model class for sequence representation we use a high-dimensional parameterized non-linear state space model rooted in Echo State Networks (ESN) with a simple deterministically constructed dynamic coupling structure [Rodan and Tiňo, 2012]. As explained above, each sequence is represented by the corresponding linear readout mapping acting on top of the common shared non-linear dynamical system. The difference between two sequences is measured through the distance of their linear readout weights weighted by a metric tensor. The shared dynamical system, as well as the metric tensor in the readout space are learned simultaneously.

The N-dimensional dynamical model we employ has the following form:

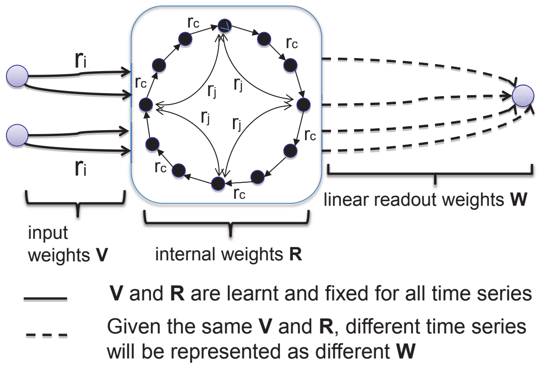
$$\boldsymbol{x}(t) = \tanh(R\,\boldsymbol{x}(t-1) + V\,\boldsymbol{s}(t)), \tag{1}$$
$$\boldsymbol{f}(t) = W\boldsymbol{x}(t) \tag{2}$$

where $\boldsymbol{x}(t) \in \Re^N$, $\boldsymbol{s}(t) \in \Re^O$ and $\boldsymbol{f}(t) \in \Re^O$ are the state vector, input vector and output at time t, respectively; R is a $(N \times N)$ dynamic coupling matrix; $V \in \Re^{N\times O}$and $W \in \Re^{O\times N}$ are the input and output weight matrices[1], respectively. We refer to (1) as the state transition mapping (STM).

Provided the non-autonomous dynamics (1) is able to represent a rich set of features of the given time series, a particular time series $\boldsymbol{s}$ can be represented by the linear readout mapping parameterized by W (eq. (2)) operating on reservoir activations $\boldsymbol{x}$, specifically fitted to $\boldsymbol{s}$ on the next-item prediction task $\boldsymbol{f}(t) \sim \boldsymbol{s}(t+1)$ by minimizing the mean squared error (MSE) between the model predictions $\boldsymbol{f}(t)$ and targets $\boldsymbol{s}(t+1)$.

Recently, [Rodan and Tiňo, 2012] proposed a highly constrained and simple dynamic coupling (cycle reservoir with jumps - CRJ) with dynamic representational capabilities comparable to those of the traditional randomized ESN models. Viewing (1) as a recurrent neural network, the structure of R in CRJ has the following form: state units are connected in a uni-directional cycle with bi-directional shortcuts (jumps). All cyclic connections have the same weight $r_c > 0$. Likewise, all jumps share the same weight $r_j > 0$. The input weight matrix is also highly constrained: the input connections have the same absolute value $r_i > 0$; the sign pattern needs to be aperiodic, see [Rodan and Tiňo, 2011] for details. The non-linear state space model used in this paper is illustrated in Figure 1.

[1]As usual, the state vector $\boldsymbol{x}(t)$ can be extended with a constant element (e.g. 1) and W with a column to account for the bias term.



r_i
r_i
r_c
r_c
r_c
r_c
r_c
r_c
r_j
r_j
r_j
r_j
input weights **V**
internal weights **R**
linear readout weights **W**
V and **R** are learnt and fixed for all time series
Given the same **V** and **R**, different time series will be represented as different **W**

Figure 1: Illustration of the parameterized state space models used in this work. The parameterized state space model has a fixed topology, i.e. a uni-directional cycle with bi-directional jumps. There are only three parameters in this model: the cyclic connection weight $r_c > 0$; the jumps weight $r_j > 0$; and the input weight $r_i > 0$. The input weight matrix V only contains r_i, and the state transition matrix R contains r_c and r_j.

In the original CRJ, the weights r_c, r_j, r_i were determined by cross-validation using grid search, which is computationally extensive in practice. In this contribution, given a sequence classification task, we aim to learn these parameters (common for the whole data set) so that **(1)** a faithful modelling of individual sequences is achieved (by allowing individualized readout mappings from the common state transition mapping), **(2)** the sequence classes in the space of readout models are well separated. We will learn the weights (r_c, r_j, r_i) using Real-Time Recurrent Learning [Williams and Zipser, 1989] and a metric tensor[2] on an appropriate cost functional reflecting **(1)** and **(2)** above.

2.1 Cost Functional

We vectorize[3] parameters W of the readout mapping (eq. (2)) into the parameter vector $\boldsymbol{w}$. Assume we are given a set of M labelled sequences $\{\boldsymbol{s}_m, y_m\}_{m=1}^{M}$. Using the fixed common state transition mapping (eq. (1)), each sequence $\boldsymbol{s}_m$ will be represented by the linear readout mapping parameterized by $\boldsymbol{w}_m$. The readout is trained on the next item prediction on $\boldsymbol{s}_m$ via ridge regression. The readout weights $\boldsymbol{w}_m$ are thus determined by the model parameters $\boldsymbol{r} = (r_i, r_c, r_j)$, as well as the sequence $\boldsymbol{s}_m$. To simplify the notation, this dependence[4] will not be made explicit.

Representability: A good non-linear state space model parameterized by $\boldsymbol{r} = (r_i, r_c, r_j)$ should minimize the representability cost with respect to all (i.e. M) training sequences:

$$Q_p(\boldsymbol{r}) = \sum_{m=1}^{M} (\sum_{t=1}^{L_m-1} \|\boldsymbol{f}_m(t) - \boldsymbol{s}_m(t+1)\|^2 + \eta ||W_m||^2), \quad (3)$$

where L_m is the length of sequence $\boldsymbol{s}_m$, and $\boldsymbol{f}_m$ is the output mapping (eq. (2)) determined by $\boldsymbol{w}_m$. This equation aims to minimize the difference between actual output $\boldsymbol{f}(t)$ and desired output $\boldsymbol{s}(t+1)$.

Separability: We aim to learn a global metric A such that sequences from the same class are close to each other in the readout model space and sequences in different classes are far away in the readout model space. Given a training sequence $\boldsymbol{s}_m$, following [Globerson and Roweis, 2006], we introduce a conditional distribution over training sequence indexes $q \neq m$:

$$p_A(q|m) = \frac{e^{-d_A(m,q)}}{\sum_{k \neq m} e^{-d_A(m,k)}}, m \neq q,$$

where

$$d_A(m,q) = (\boldsymbol{w}_m - \boldsymbol{w}_q)^T A (\boldsymbol{w}_m - \boldsymbol{w}_q) \quad (4)$$

is the squared distance between readout parameter $\boldsymbol{w}_m$ and $\boldsymbol{w}_q$ under the (global) positive semi-definite metric A. If all readouts in the same class were mapped to a single point and infinitely far from points in different classes, we would have the ideal distribution [Globerson and Roweis, 2006]:

$$p_0(q|m) \propto \begin{cases} 1 & \text{if } y_m = y_q \\ 0 & \text{if } y_m \neq y_q \end{cases} .$$

We optimize the non-linear state space model (eq. (1)) and the metric A such that $p_A(q|m)$ is as close as possible to $p_0(q|m)$ with respect to KL divergence:

$$\min_A \sum_{m=1}^{M} D_{KL} \left[p_0(q|m) \| p_A(q|m) \right],$$
$$s.t. A \text{ is positive semi-definite.}$$

This results in a separability cost[5] $Q_s(\boldsymbol{r}, A)$ equal to

$$\sum_{\{(m,q): y_q = y_m\}} d_A(m,q) + \sum_{m=1}^{M} \log \sum_{k \neq m} e^{-d_A(m,k)}. \quad (5)$$

Overall cost functional:
Using the representation (3) and separation costs (5), we construct the cost functional to be minimized by the state space model with parameters $\boldsymbol{r}$ and metric A:

$$Q(\boldsymbol{r}, A) = Q_s(\boldsymbol{r}, A) + \lambda Q_p(\boldsymbol{r}) \quad (6)$$

where parameter $\lambda > 0$ controls the tradeoff between the *representability* Q_p and the *separability* Q_s. W_m is obtained by minimizing $Q_p(r)$. However, since $Q_p(r)$ is quadratic with respect to W_m, enforcing the derivatives of $Q_p(r)$ with respect to W_m to be 0 will give an closed-form solution of W_m as presented in Eq(9).

In practice we first minimize $Q(\boldsymbol{r}, A)$ (via gradient descent) and then project the resulting A onto the space of positive semi-definite matrices (minimizing L_2 norm) [Globerson

[2]This can be viewed as metric learning [Xing *et al.*, 2002] in the readout model space.

[3]For multivariate time series with dimensionality O, the linear readout weight matrix W is a $O \times N$ matrix. In this case, we vectorize the matrix W into a column vector.

[4]The readout weight $\boldsymbol{w}_m$ is a function of the model parameters $\boldsymbol{r}$ and the sequence $\boldsymbol{s}_m$ as the variables.

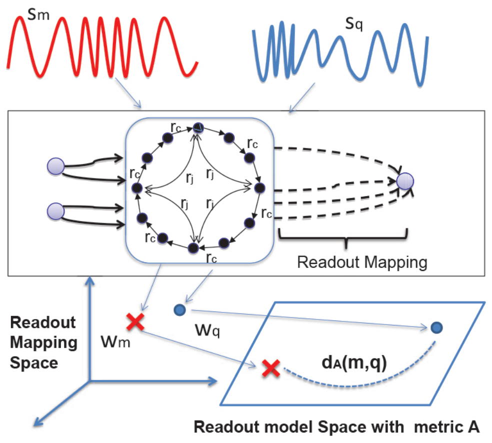


Figure 2: Outline of the MMCL framework.

and Roweis, 2006]. The gradients of Q with respect to STM parameters $\boldsymbol{r}$ and metric tensor A are obtained as (θ stands for r_i, r_c or r_j):

$$\begin{aligned}\frac{\partial Q}{\partial \theta} &= \sum_{m,q}^{M} (p_0(q|m) - p_A(q|m)) (\boldsymbol{w}_q - \boldsymbol{w}_m)^T \quad (7)\\ &\cdot (A + A^T) \left(\frac{\partial \boldsymbol{w}_q}{\partial \theta} - \frac{\partial \boldsymbol{w}_m}{\partial \theta} \right)\\ &+ \lambda \sum_{m=1}^{M} \sum_{t=1}^{L_m - 1} 2 \left(\boldsymbol{f}_m(t) - \boldsymbol{s}_m(t+1)\right)^T \frac{\partial \boldsymbol{f}_m(t)}{\partial \theta}\end{aligned}$$

$$\begin{aligned}\frac{\partial Q}{\partial A} &= \sum_{m,q}^{M} (p_0(q|m) - p_A(q|m)) \cdot\\ &(\boldsymbol{w}_q - \boldsymbol{w}_m)(\boldsymbol{w}_q - \boldsymbol{w}_m)^T . \quad (8)\end{aligned}$$

Using ridge regression, the readout parameters W_m ($\boldsymbol{w}_m = vec(W_m)$) representing sequence $\boldsymbol{s}_m$ are obtained:

$$W_m = \tilde{\boldsymbol{s}}_m \, X_m^T (X_m \, X_m^T + \eta I)^{-1}. \quad (9)$$

where $X_m = [\boldsymbol{x}_m(1), \cdots, \boldsymbol{x}_m(L_m - 1)]$ is the $(N \times (L_m - 1))$ STM state matrix storing state activations obtained while processing $\boldsymbol{s}_m$ as columns, $\tilde{\boldsymbol{s}}_m = [\boldsymbol{s}_m(2), \cdots, \boldsymbol{s}_m(L_m)]$ is the target matrix for the next-item prediction task and $\eta > 0$ is a regularization parameter. To ease the presentation, we omit the sequence index m in the following derivations. We have

[5]assuming equally probable classes and ignoring constant terms.

$$\begin{aligned}\frac{\partial W}{\partial \theta} &= \tilde{\boldsymbol{s}} \frac{\partial X^T}{\partial \theta} (XX^T + \eta I)^{-1} + \tilde{\boldsymbol{s}} X^T \cdot\\ &\quad \frac{\partial (XX^T + \eta I)^{-1}}{\partial \theta}\\ &= \tilde{\boldsymbol{s}} \left(\frac{\partial X}{\partial \theta}\right)^T (XX^T + \eta I)^{-1} - \tilde{\boldsymbol{s}} X^T\\ &\quad (XX^T + \eta I)^{-1} \frac{\partial (XX^T + \eta I)}{\partial \theta} (XX^T + \eta I)^{-1}\\ &= \tilde{\boldsymbol{s}} \left(\frac{\partial X}{\partial \theta}\right)^T (XX^T + \eta I)^{-1} - \tilde{\boldsymbol{s}} X^T\\ &\quad (XX^T + \eta I)^{-1} \left(\frac{\partial X}{\partial \theta} X^T + X \left(\frac{\partial X}{\partial \theta}\right)^T \right) \cdot\\ &\quad (XX^T + \eta I)^{-1} \quad (10)\end{aligned}$$

where

$$\frac{\partial X}{\partial \theta} = \left[\frac{\partial \boldsymbol{x}(1)}{\partial \theta}, \frac{\partial \boldsymbol{x}(2)}{\partial \theta}, \ldots, \frac{\partial \boldsymbol{x}(L-1)}{\partial \theta} \right]$$

Adopting real time recurrent learning, we have:

$$\begin{aligned}\frac{\partial \boldsymbol{x}(t)}{\partial \theta} &= sech^2(R\boldsymbol{x}(t-1) + V s(t))\\ &\quad .* \left(\frac{\partial V}{\partial \theta} s(t) + R \frac{\partial \boldsymbol{x}(t-1)}{\partial \theta} \right), \quad (11)\end{aligned}$$

where $.*$ stands for element-wise matrix multiplication (MATLAB notation). Finally,

$$\frac{\partial \boldsymbol{f}(t)}{\partial \theta} = W \frac{\partial \boldsymbol{x}(t)}{\partial \theta} + \frac{\partial W}{\partial \theta} \boldsymbol{x}(t), \quad (12)$$

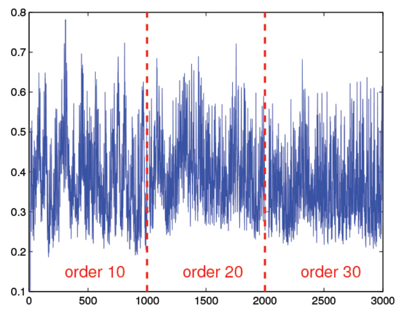
and $\frac{\partial W}{\partial \theta}$ can be obtained by combining Equations (10–12). Hence, state space model learning (eq. (7)) can be obtained by combining eq. (12) and eq. (10). MMCL can be achieved by alternating between performing state space model learning (eq. (7)) and metric learning in the readout model space (eq. (8)) to obtain a tradeoff between *representability* and *separability*.

2.2 Sequence Classification

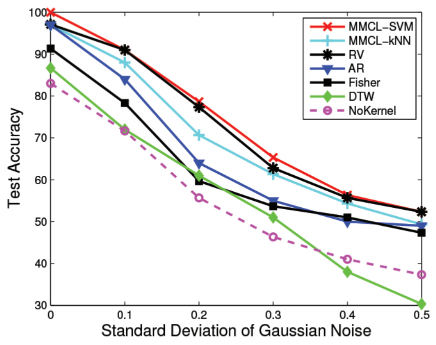
Having determined the appropriate STM and global metric tensor on the readout parameters, we can use any convenient distance-based classifier, e.g. k-Nearest Neighbour (kNN). The (squared) distance between two sequences $\boldsymbol{s}_m$ and $\boldsymbol{s}_q$ is simply $d_A(m, q)$ in eq. (4) , where, as explained above, the feature vector $\boldsymbol{w}_m$ of a sequence $\boldsymbol{s}_m$ is obtained by ridge regression from the dynamical system generated using the learned parameters r_i, r_c and r_j with the task of predicting the next item of the sequence. Of course, one can also adopt a kernel classification framework (e.g. SVM) using a sequence kernel

$$\mathcal{K}(\boldsymbol{s}_m, \boldsymbol{s}_q) = \exp\left\{-\gamma \, d_A(m, q)\right\}, \quad (13)$$

where $\gamma > 0$ is a scale parameter. In our experiments we employ both kNN operating in the readout space and SVM with the kernel defined in (13).



(a)



(b)

	Parameters	Parameter range
AR	ξ, p	$\xi \in \{0.1, 0.2, \cdots, 0.9\}$, $p \in \{1, 2, \cdots, 10\}$
Fisher	$\#states$	$\#states \in \{1, 2, \cdots, 10\}$
RV	r_i,r_c,r_j, η	r_i,r_c,$r_j \in \{0.01, 0.05, \cdots, 1\}$ $\eta \in \{10^{-5}, 10^{-4}, \cdots, 10^{1}\}$
MMCL	η, λ	$\eta \in \{10^{-5}, 10^{-4}, \cdots, 10^{1}\}$ $\lambda \in \{0, 10^{-1}, \cdots, 10\}$

(c)

Figure 3: (a) illustration of 3 NARMA sequences. (b) the classification accuracies on the synthetic (test) data sets. (c) Parameter search ranges for methods employed in the paper. p is the order of the vector autoregressive model, ξ is an *AR* kernel parameter [Cuturi and Doucet, 2011], $\#states$ is the number of states for HMM in *Fisher* kernel, η is the ridge regression regularization parameter, λ is the trade-off parameter. r_i, r_c and r_j are parameters of the deterministic state space mode used in *RV* kernel.

3 Experimental Studies

We compare our MMCL framework with several state-of-the-art methods for sequence classification such as *Dynamic time warping (DTW)* [Berndt and Clifford, 1994], *Autoregressive kernel (AR)* [Cuturi and Doucet, 2011], the *Fisher kernel (Fisher)* [Jaakkola *et al.*, 1999] and the *Reservoir kernel (RV)* of [Chen *et al.*, 2013]. The RV kernel can be viewed essentially as the MMCL kernel, except for the dynamic parameters r_i, r_c, r_j are simply set by costly cross-validation without any regard for class separability and metric tensor learning. The implementation of the *AR* kernel is from Cuturi's website[6]. The *Fisher* kernel was obtained Maaten's website[7]. The *AR*, *Fisher*, *RV* and *DTW* based kernels were used in a SVM classifier.

In MMCL, the number of nodes was fixed to $N = 50$ and 10 jumps (making the jump length 5). The jump length could be set by cross-validation as in [Rodan and Tiňo, 2012], but to demonstrate the power of the framework, we simply fixed the topology for all experiments without pre-testing.

All (hyper) parameters, such as the MMCL trade-off parameter λ, order p in the *AR* kernel, number of hidden states in the HMM based *Fisher* kernel, regularization parameter η for ridge regression etc. have been set by 5-fold cross-validation on the training set. We employ a well-known, widely accepted and used implementation of SVM – LIBSVM [Chang and Lin, 2011]. In LIBSVM, we use cross validation to tune the slack-weight 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). The SVM parameters, kernel width γ in eq. (13) and C, were tuned in the following ranges: $\gamma \in \{10^{-6}, 10^{-5}, \cdots, 10^{1}\}$, $C \in \{10^{-3}, 10^{-2}, \cdots, 10^{3}\}$. We also tested our MMCL method using a k-NN classifier where $k \in \{1, 2, \cdots, 10\}$. We refer to SVM and kNN classifiers built within the MMCL framework as *MMCL-SVM* and *MMCL-kNN*, respectively. The search ranges for the parameters are detailed in Figure 3(c).

3.1 Synthetic Data

We employed 3 NARMA time series models of orders 10, 20 and 30, given by:

$$s_{t+1} = 0.3 s_t + 0.05 s_t \sum_{i=0}^{9} s_{t-i} + 1.5 u_{t-9} u_t + 0.1,$$

$$s_{t+1} = \tanh(0.3 s_t + 0.05 s_t \sum_{i=0}^{19} s_{t-i} + 1.5 u_{t-19} u_t + 0.01) + 0.2,$$

$$s_{t+1} = 0.2 s_t + 0.004 s_t \sum_{i=0}^{29} s_{t-i} + 1.5 u_{t-29} u_t + 0.201,$$

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 time series are challenging due to non-linearity and long memory (see Figure 3(a, b)).

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 a training set, and the other 100 time series form the test set. In order to study robustness of the kernels we also corrupt the time series with additive Gaussian noise (zero mean, standard derivation varies in [0.1,0.5]). Figure 3(b) shows the test set classification accuracy against the noise level. As a baseline, we also include results of SVM directly operating on the time series (300-dimensional inputs) - *NoKernel*. The *MMCL* based kernel method *MMCL-SVM* outperforms the baseline as well as all the other methods. Although the *RV* kernel achieves the second best performance in this noise data, it takes more time[8]

[6]http://www.iip.ist.i.kyoto-u.ac.jp/member/cuturi/AR.html

[7]http://homepage.tudelft.nl/19j49/Software.html

[8]The size of the search grid (for four parameters r_i, r_c, r_j, and η) is 109,375 in RV kernel, while it is 701 in the MMCL experimental setting.

Table 1: Description of the data sets (left) and the performances (best is boldfaced). Note that in 4 out of the 7 datasets (OSULeaf, Oliveoil, Fish and Adiac) both MMCL techniques outperform others. *Model* stands for *MMCL-SVM* with model learning only (without metric learning) and *Decouple* stands for *MMCL-SVM* with decoupled model and metric learning.

DATASET	LENGTH	CLASSES	# TRAIN	# TEST	*DTW*	*AR*	*Fisher*	*RV*	*Model*	*Decouple*	*MMCL-kNN*	*MMCL-SVM*
OSULEAF	427	6	200	242	74.79	56.61	54.96	69.83	70.25	77.69	**88.02**	85.12
OLIVEOIL	570	4	30	30	83.33	73.33	56.67	86.67	83.33	86.67	86.67	**93.33**
LIGHTNING2	637	2	60	61	64.10	**77.05**	64.10	**77.05**	75.41	75.41	70.49	75.41
BEEF	470	6	30	30	66.67	78.69	58.00	80.00	80.00	80.00	63.33	**82.67**
FISH	463	8	175	175	69.86	60.61	57.14	79.00	77.14	81.14	**88.57**	87.43
COFFEE	286	2	28	28	85.71	**100.00**	81.43	**100.00**	92.86	96.43	89.29	**100.00**
ADIAC	176	37	390	391	65.47	64.45	68.03	72.63	72.63	**73.40**	72.30	**73.40**

Table 2: Summary of multivariate (variable length) series classification problems (left) and the performances (best is boldfaced). For the handwritten and AUSLAN datasets both MMCL variants outperform others. *Model* stands for *MMCL-SVM* with model learning only (without metric learning) and *Decouple* stands for *MMCL-SVM* with decoupled model/metric learning.

DATASET	DIM	LENGTH	CLASSES	# TRAIN	# TEST	*DTW*	*AR*	*Fisher*	*RV*	*Model*	*Decouple*	*MMCL-k*NN	*MMCL*-SVM
Libras	2	45	15	360	585	94.02	91.79	**94.93**	93.25	92.82	94.19	94.19	94.87
handwritten	3	60-182	20	600	2258	88.67	73.30	87.52	89.41	89.02	90.79	91.31	**91.41**
AUSLAN	22	45-136	95	600	1865	97.00	76.53	94.74	96.00	96.00	96.41	97.05	**97.80**

to select the three parameters r_i, r_c and r_j by cross validation with grid search. In MMCL, the three parameters are learned based on data instead of selecting by grid search, which may achieve better performance and save the parameter tuning time.

3.2 Univariate Benchmark Data

We used 7 data sets from the UCR Time Series Repository [Keogh *et al.*, 2011]. Each data set has already been split into training and test sets (see Table 1). Table 1 reports performance of the studied methodologies on the benchmark data in terms of test set classification accuracy. The *MMCL* methods outperform the other algorithms on most of the data sets, except for being inferior to or on a par with *AR* kernel and *RV* kernel on Lightning2 and Coffee data sets, respectively, *MMCL-SVM* is superior to the simpler *MMCL-kNN* on 5 data sets.

In order to investigate MMCL further, we performed model learning only (without metric learning) using SVM in Tables 1 and 2, called *model*. We (as expected) observed similar performance levels to those of *RV* (model parameters optimized by cross validation). *RV* sometimes outperforms MMCL (model learning only) because the model learning may be trapped in the local optima. In Tables 1 and 2, we have also decoupled the model and metric learning in MMCL, termed *decouple*. This naturally resulted in an inferior performance. Independent learning might ignore possible coupling between the provider of dynamical features (dynamical system) and the static readout models.

As shown in Table 1, our methodology was slightly inferior to other algorithms on some datasets, e.g. Lightning2. Lightning2 contains time series of power spectra related lightning. The classes can be characterized by pronounced low-memory features of the power spectra series such as those corresponding to sharp turn-on of radiation, gradual increase in power series, or sudden spikes etc. Such time series features can be naturally captured by AR models and thus we hypothesize that the AR kernels provide a strong distinguishing platform for lightning classification.

3.3 Multivariate Time Series

The 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 variable length time series. Following [Cuturi and Doucet, 2011] (previous *AR* kernel study) we split the data into training and test sets (see Table 2).

Table 2 shows that the *MMCL* based *MMCL-SVM* is superior on two data sets with slightly worse, but comparable performance to *Fisher* kernel on the Libras dataset.

In terms of the learned parameters (r_i, r_c, r_j), we have checked their values on our data sets with and without metric learning. Interestingly enough, in general, metric learning lead to increased r_i, r_j. Cycle weights r_c differed by a comparatively smaller amount. The increased input loads r_i force the dynamical system to operate in saturation regimes of the tanh function (see eq. (1)). This can increase class separability of state space representations of sequences from different classes. Large jump weight r_j in effect broadens frequency spectrum of the non-autonomous dynamical system. While a cyclic coupling only [Rodan and Tiňo, 2011] can capture slow dynamical regimes, introduction of jumps leads to broadening the sensitivity of the system to faster changes in the input stream.

4 Discussion and Conclusion

This paper focuses on the *learning in the model space* approach where each data item (sequence) is represented by the corresponding model from a carefully designed model class retaining both reasonable flexibility to represent a wide variety of sequences and a relatively small number of free parameters. The model class used is a special form of state space model. The dynamic part of the model - state transition mapping - has only three free parameters that are learnt and then fixed for each data set. Individual sequences in the data set are then represented by the corresponding (static) linear readouts from the state space (fixed STM).

Given a particular data set, we learn both the appropriate state space model and metric in the readout model space. Since we consider sequence classification, learning is guided by two criteria: *representability* and *class separability* of the readout models.

On 11 sequence classification data sets the sequence kernels constructed using our methodology (and applied in SVM) were on a par or outperformed the other four studied methods - *Dynamic Time Warping* based kernels, *Autoregressive* kernel, the *Fisher* kernel based on *Hidden markov model*, and the *reservoir kernel* with *manually* chosen state transition parameters by cross validation using grid search and *without* metric learning in the readout space.

Since in our framework each sequence is represented as a model in the readout model space with a learnt global metric tensor, any distance based classifier can be used. As a baseline we employed kNN. Except for the Coffee, Lightning2 and *Libras* data sets, kNN achieved comparable or better performance than the four alternatives.

Our results may seem surprising, especially given the rather simple structure of the state transition part. We emphasize that the STM employed does not necessarily have to provide a platform for the best sequence modelling. For that, the imposition of the same fixed STM for the whole data set may indeed be too restrictive. However, the task at hand is sequence classification and the requirements on the STM are of a different nature: allowing state space processing of sequences so that the corresponding linear readouts coincide as much as possible for sequences of the same class, while keeping the separation of the distinct classes. This, as demonstrated in our experiments, is indeed possible using a fixed STM. The optimal STM for the classification purposes is learnt alongside the metric tensor on linear readouts from the filter that enhances the within-class collapsed representations and between-class separation.

The superior performance of MMCL methodology comes at a price - relatively high computational complexity. In each iteration of gradient descent in MMCL, the complexity is $O(M^2d^2 + dML)$, where M is the number of sequences, d is the dimensionality of the time series, and L is the length of sequences. However, this tolerable computational cost is well offset by the high classification accuracy and the saving of effort for tuning parameters in the state space model using cross validation with grid search.

Future work includes experiments using lag-p model, L1 norm, the computational cost and the situation to deal with imbalanced data sets for MMCL.

Acknowledgments

Huanhuan Chen is supported by the National Natural Science Foundation of China under Grants 61203292, and the One Thousand Young Talents Program. The financial assistance of the Royal Society under International Exchange grant IE121685 is gratefully acknowledged.

References

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

[Bosch *et al.*, 2008] A. Bosch, A. Zisserman, and X. Muoz. Scene classification using a hybrid generative/discriminative approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(4):712–727, 2008.

[Brodersen *et al.*, 2011] K.H. Brodersen, T.M. Schofield, A.P. Leff, C.S. Ong, E.I. Lomakina, J.M. Buhmann, and K.E. Stephan. Generative embedding for model-based classification of fMRI data. *PLoS Computational Biology*, 7(6):e1002079, 2011.

[Chan and Vasconcelos, 2005] A. B. Chan and N. Vasconcelos. Probabilistic kernels for the classification of auto-regressive visual process. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 846–851, 2005.

[Chan and Vasconcelos, 2007] A. B. Chan and N. Vasconcelos. Classifying video with kernel dynamic textures. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[Chang and Lin, 2011] 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.

[Chen *et al.*, 2013] H. Chen, F. Tang, P. Tino, and X. Yao. Model-based kernel for efficient time series analysis. In *Proceedings of the 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD) (KDD'13)*, pages 392–400, Chicago, USA, 2013.

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

[Cuturi *et al.*, 2007] M. Cuturi, J-P Vert, O. Birkenes, and T. Matsui. A kernel for time series based on global alignments. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 413–416, 2007.

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

[Ghanem and Ahuja, 2010] B. Ghanem and N. Ahuja. Maximum margin distance learning for dynamic texture recognition. In *Proceedings of the 11th European Conference on Computer Vision (ECCV'10)*, pages 223–236. Springer Berlin Heidelberg, 2010.

[Globerson and Roweis, 2006] A. Globerson and S. Roweis. Metric learning by collapsing classes. In *Advances in neural information processing systems (NIPS)*, volume 18, pages 451–458, 2006.

[Jaakkola and Haussler, 1999] T.S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. *Advances in Neural Information Processing System (NIPS)*, pages 487–493, 1999.

[Jaakkola *et al.*, 1999] 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. AAAI Press, 1999.

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

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

[Keogh *et al.*, 2011] 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/.

[Moreno *et al.*, 2004] 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.

[Rodan and Tiňo, 2011] A. Rodan and P. Tiňo. Minimum complexity echo state network. *IEEE Transactions on Neural Networks*, 22(1):131–144, 2011.

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

[Saisan *et al.*, 2001] P. Saisan, G. Doretto, Y. Wu, and S. Soatto. Dynamic texture recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 58–63, 2001.

[Sha and Saul, 2007] F. Sha and L. K. Saul. Large margin hidden Markov models for automatic speech recognition. In *Advances in neural information processing systems (NIPS)*, 2007.

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

[Van der Maaten, 2011] L. Van der Maaten. Learning discriminative Fisher kernels. In *Proceedings of the 28th International Conference on Machine Learning (ICML'11)*, pages 217–224, 2011.

[Vishwanathan and Smola, 2006] S. V. N. Vishwanathan and A. J. Smola. Binet-Cauchy kernels on dynamic systems and its application to the analysis of dynamic scenes. *International Journal of Computer Vision*, 73(1):95–119, 2006.

[Williams and Zipser, 1989] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[Xing *et al.*, 2002] E. Xing, M. Jordan, S. Russell, and A. Ng. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pages 505–512, 2002.