Sequential Data Classification in the Space of Liquid State Machines

Yang Li, Junyuan Hong, and Huanhuan Chen(✉)

UBRI, School of Computer Science and Technology,
University of Science and Technology of China, Hefei 230027, Anhui, China
{csly,jyhong}@mail.ustc.edu.cn, hchen@ustc.edu.cn

Abstract. This paper proposes a novel classification approach to carrying out sequential data classification. In this approach, each sequence in a data stream is approximated and represented by one state space model – liquid state machine. Each sequence is mapped into the state space of the approximating model. Instead of carrying out classification on the sequences directly, we discuss measuring the dissimilarity between models under different hypotheses. The classification experiment on binary synthetic data demonstrates robustness using appropriate measurement. The classifications on benchmark univariate and multivariate data confirm the advantages of the proposed approach compared with several common algorithms. The software related to this paper is available at https://github.com/jyhong836/LSMModelSpace.

Keywords: Sequential learning · Classification · Learning in the model space

1 Introduction

Sequential data classification is a fundamental problem in the machine learning community. In the classification, the degree of dissimilarity between sequences needs to be quantified. If the sequential data are of equal length, it is sufficient to use conventional machine learning methods by treating sequences as numerical vectors. Kernel methods could be efficient and might achieve satisfying performances [18], provided that the length of sequence is not long. However, in reality, large amount of sequential data are variable-length.

To deal with sequential data that are variable-length and possibly long, plenty of algorithms, e.g. dynamic time warping [1], autoregressive kernel [8], spectral analysis [11], are proposed.

Searching for a global alignment between variable-length sequences is a way to handle variable-length data. This methodology of non-linear warping and matching segments of two sequences is exemplified by *dynamic time warping* (DTW) [21]. However, due to non-linear warping, the triangular inequality, one of the requisites for the validity of a metric, is not satisfied. The measurement in DTW is not a metric actually, lacking geometric interpretation to the experimental result [9].

P. Frasconi et al. (Eds.): ECML PKDD 2016, Part I, LNAI 9851, pp. 313–328, 2016.
DOI: 10.1007/978-3-319-46128-1_20

Fisher Kernel [12] fits one single generative model (Hidden Markov Model) to sequences and compares how much new incoming sequence "stretches" the average model trained with past sequences. *Fisher Kernel* defines *Fisher Score* as gradients of log-likelihood, $\log p(\boldsymbol{x}|\theta)$, with regard to hidden parameters. As *Fisher Kernel* train the generative model under maximum likelihood principle, it may lead to sub-optimal results. Since a generative model that fits data well may easily get stuck in the local minimum of its log-likelihood, where the gradient representation of data is (nearly) zero [17].

The computation of *Fisher Kernel* of sequences $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$ is defined as:

$$\nabla_{\boldsymbol{\theta}}^T p(\boldsymbol{x}|\theta)\mathcal{I}^{-1}\nabla_{\boldsymbol{\theta}} p(\boldsymbol{x}|\theta) \tag{1}$$

where $\mathcal{I}$ is the *Fisher information matrix*. Computation of *Fisher Kernel* involves the inverse of *Fisher information matrix*. This procedure could be time-consuming. A routinely adopted way to bypass this difficulty is to replace the *Fisher information matrix* with identity matrix, at the cost of losing some precision in the approximation [22].

Fisher Kernel learning [17] leverages the label information so that the objective functions in the same class have similar gradients. It applies idea from metric learning to improve its performance. Both methods show effectiveness but low efficiency in obtaining the representations to data, as more computation is involved in computing gradients, even when the *Fisher information matrix* is assumed to be identity matrix.

Autoregressive Kernel (AR) [8] employs a likelihood profile as features for sequences. The likelihood profile is generated by a *Vector Autoregressive Model* under different parametric settings. The dissimilarity between sequences is computed with Bayesian method. It can be verified that this measurement is a valid Hilbertian metric [8]. AR relaxes the constraint of using a single generative model to explain the whole data as did in *Fisher Kernel* and *Fisher Kernel learning*. However, AR does not use the timestamps in a sequence to improve the prediction [20].

Chen *et al.* approximated time series via echo state networks (ESN) [4,5], and demonstrated that readout weights in ESNs could offer discriminant features for sequences. Under the representation provided by topologically fixed reservoir for the whole data, the readout weights, the only trained part, covers the uniqueness of a specific sequence, bringing in more versatility and flexibility. It was demonstrated that ESN is able to handle continuous sequences in complicated scenario [6]. In addition, a co-learning strategy was devised to strengthen its representation capability on continuous sequences [3]. In this paper, we further extend this methodology to process binary data, and demonstrate the improvement on performance by using liquid state machine (LSM). In LSM, individual node (neuron) has its own "state memory", and responds from its own history and current input signal, while nodes in ESN give responses based on merely their current state. The replacement brings enhancement "memory" to the reservoir, and demonstrates to be beneficial by experiments.

In this paper, we propose a novel approach to representing sequences, which might be of different lengths and of different characteristics, in a higher dimen-

sional space. In this approach, each sequence is represented by a LSM, which gives approximation to the conditional probability of likelihood of the sequence. After obtaining models, the classification is conducted on the models, rather than on the sequences directly. In this paper, we discuss measurements under different assumptions on the "model distributions". The model set, along with the defined measurements, offers a novel space for classification and other possible learning tasks. This space is referred as a model space for a certain data set in this paper.

2 Discriminant Learning in the Model Space

LSM incorporates time into the model of neural network to enhance the level of realism in the simulation, emerging as a new computational model [16]. A LSM consists of two parts (apart from input layer) in its framework. A large collection of nodes that are randomly connected to each other make up the reservoir part. Each node receives inputs from input layer as well as from other nodes. The spatio-temporal pattern of the activations in nodes is read out by the final layer as linear combinations in performing certain tasks. The final layer is the only part that needs training.

We illustrate the scheme diagram of model space and LSM in Fig. 1. In the figure, LSMs are used to give approximations to sequences and in turn the set of LSMs is considered in the learning algorithms.

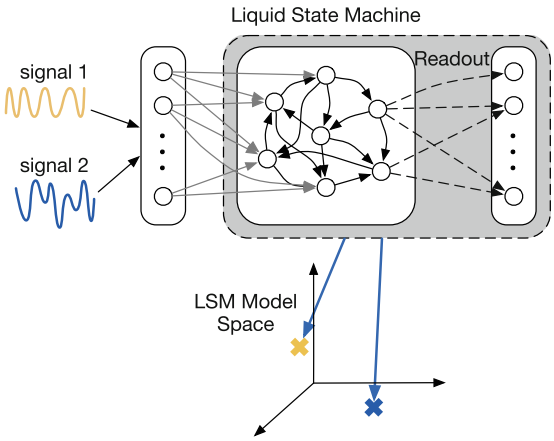

Fig. 1. The schematic diagram for LSM and model space. LSMs provide representations for two sequences. The model space is seen as a high dimensional space, in which the readout weights of LSMs are assembled.

The form of the LSM [14] is generalized as follows:

$$\begin{cases} \boldsymbol{x}(t) = Q(R\boldsymbol{x}(t-1) + V\boldsymbol{s}(t)) \\ \boldsymbol{y}(t) = f(\boldsymbol{x}(t)) = W\boldsymbol{x} \end{cases} \tag{2}$$

where $\boldsymbol{x}(t) \in \Re^n$ is the state vector defined in the real domain. Subscript n is the number of reservoir nodes. Input $\boldsymbol{s}(t) \in \Re^{d+1}$ is input which has been augmented by adding bias as one of its components. R and V are the appropriately defined coefficient matrices. $\boldsymbol{y}(t) \in \Re^{n'}$ and W denote output and readout weights respectively. Superscript n' is the dimensionality of output. $Q(\cdot)$ is the response function defined on the internal nodes.

A LSM is trained by making use of past values and predicting the present value. Readout weights $W \in \Re^{n' \times n}$ are trained by adjusting W in order for $W\boldsymbol{x}(\boldsymbol{t}) = \boldsymbol{s}(t+1)$. The dimensionality n' satisfies $n' = d$ in this scenario.

We consider an arbitrary sequence $\boldsymbol{s} = \{s_0, s_1, \cdots, s_n\} \in \Re^d$, where d is the dimensionality of the sequence. We also use $\boldsymbol{s}(t)$ to denote a sequence which is indexed by t. We assume that the index starts from 0 unless otherwise stated.

The likelihood of a sequence $\boldsymbol{s}$ is expressed as:

$$\ell(\boldsymbol{s}) = \ell(\{s_0, s_1, \cdots, s_n\})$$

which can be further factorized into

$$\ell(\boldsymbol{s}) = \mathcal{P}_0(s_0)\mathcal{P}_1(s_1|s_0)\mathcal{P}_2(s_2|s_1, s_0) \cdots\cdots \mathcal{P}_n(s_n|s_{n-1}, \cdots, s_0)$$

where $\mathcal{P}_i(s_i|s_{i-1} \cdots s_0), i = 0 \cdots n$ is the conditional probability.

In most cases, the assumption is too strong that the conditional probability $\mathcal{P}_i(\cdot|\cdot)$ of a sequence can be generalized and formulated explicitly. Assumptions on the form of $\mathcal{P}_i(\cdot|\cdot)$ might lead to sub-optimal results.

In our approach, we make use of the universal approximating ability [16] of LSM under a weak assumption on the conditional probability distribution, assuming $\mathcal{P}_i(\cdot|\cdot)$ is time-invariant, i.e. $\mathcal{P}_i(\cdot|\cdot) = \mathcal{P}(\cdot|\cdot)$. The universal approximating ability states that, given enough variety in the interior nodes, nonlinear input-output mappings could be approximated by LSM under training of sufficiently long input sequences. Our approach bases the approximation to $\mathcal{P}(\cdot|\cdot)$ on this ability and therefore uses models rather than simplified formulations in the classification algorithm.

2.1 Measurement of Dissimilarity Between Models in the Model Space

The dissimilarity of two sequences is judged from the divergence between two fitting LSMs. Given two sequences $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$, a general measurement of dissimilarity is formulated as follows:

$$\begin{aligned} \mathcal{D}(\boldsymbol{s}_i, \boldsymbol{s}_j) &= \Big(\int_{\boldsymbol{x} \in \mathcal{I}} ||f_i - f_j||^2 d\mu(\boldsymbol{x})\Big)^{1/2} \\ &= \left(\int_{\boldsymbol{x} \in \mathcal{I}} (W_i\boldsymbol{x} - W_j\boldsymbol{x})^T (W_i\boldsymbol{x} - W_j\boldsymbol{x}) d\mu(\boldsymbol{x})\right)^{1/2} \end{aligned} \tag{3}$$

$||\cdot||$ is the norm which calculates the disagreement between two model outputs. $\mathcal{I}$ is the change interval for model vector $\boldsymbol{x}$. $\mu(\boldsymbol{x})$ is the probability distribution for $\boldsymbol{x}$.

Uniform distribution over $\boldsymbol{x}$ considers the simplest case, in which the probability distribution $\mu(\boldsymbol{x})$ is assumed to be only dependent on the interval $\mathcal{I}$. Later, this assumption will be relaxed and more general cases will be discussed.

Under the assumption of the uniform distribution, the dissimilarity between sequences $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$ is simplified into

$$\begin{aligned}\mathcal{D}(\boldsymbol{s}_i, \boldsymbol{s}_j) &= \left(\int (W_i\boldsymbol{x} - W_j\boldsymbol{x})^T (W_i\boldsymbol{x} - W_j\boldsymbol{x}) d\mu(\boldsymbol{x}) \right)^{1/2} \\ &= \mathcal{C}||W_i\boldsymbol{x} - W_j\boldsymbol{x}||.\end{aligned} \tag{4}$$

where the irrelevant terms in last formula of Eq. (4) are generalized into constant $\mathcal{C}$.

In more general cases where $\boldsymbol{x}$ is not evenly distributed, but not changes dramatically, we use Gaussian mixture model to approximate the probability distribution $\mu(\boldsymbol{x})$. It fits the probability distribution $\mu(\boldsymbol{x})$ with a mixture of finite Gaussian distributions.

$$\mu(\boldsymbol{x}) = \sum \alpha_i N(\theta_i, \Sigma_i)$$

where α_i are the mixture coefficients for i-th Gaussian distribution. All α_i sum up to 1, $\sum \alpha_i = 1$. Parameters θ_i and Σ_i are mean and variance in i-th Gaussian distribution.

Substitute $\mu(\boldsymbol{x})$ with Gaussian mixture model, the dissimilarity between two sequences is formulated as:

$$\mathcal{D}(\boldsymbol{s}_i, \boldsymbol{s}_j) = \sum_k \alpha_k trace(W_i^T W_j \Sigma_k) + \theta_k^T W_i^T W_j \theta_k \tag{5}$$

Sampling, as a natural alternative to the above approximation method, makes no assumptions on the form of $\mu(\cdot)$. An asymptotic optimal estimation for a probability distribution $\mu(\cdot)$ is guaranteed from the law of large numbers. This estimation may lead to more robust result, if no prior information on $\mu(\cdot)$ exists. Applying sampling to Eq. (3) is straightforward.

$$\mathcal{D}(\boldsymbol{s}_i, \boldsymbol{s}_j) \approx \frac{1}{m} \sum_k ||W_i\boldsymbol{x}_k - W_j\boldsymbol{x}_k|| \tag{6}$$

where m denotes the amount of sampling points.

Assume the deviation $\varepsilon(t)$ between the output of a LSM $\boldsymbol{y}(t) = W\boldsymbol{x}$ and the desired output $\boldsymbol{s}(t+1)$ follows a zero-mean Gaussian distribution $\varepsilon(t) = \mathcal{N}(0, \delta^2 I)$. When the methodology of *Fisher Kernel* is applied, the conditional probability of observing $\boldsymbol{s}(t+1)$ given past values is formulated as:

$$\mathcal{P}((\boldsymbol{s}(t+1)|\boldsymbol{s}(1\cdots t)) = (2\pi\delta^2)^{-d/2} exp\Big(- \frac{||\boldsymbol{s}(t+1) - W\boldsymbol{x}(t)||}{2\delta^2} \Big)$$

The *Fisher score* U between $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$ takes the form of inner product of two derivatives with regard to the hidden parameters. The derivative quantifies how the model adjusts its current parametric setting in order to fit a new sequence. The derivative of probability $\mathcal{P}(\cdot|\cdot)$ in terms of W gives rise to:

$$\begin{aligned} U &= \frac{\partial \log \mathcal{P}(\boldsymbol{s}(1 \cdots l))}{\partial W} \\ &= \sum_{t=1}^{l} \frac{\boldsymbol{s}(t)\boldsymbol{x}(t-1)^T - W\boldsymbol{x}(t-1)\boldsymbol{x}(t-1)}{\delta^2} \end{aligned}$$

The dissimilarity between $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$ is expressed as:

$$\mathcal{D}(\boldsymbol{s}_i, \boldsymbol{s}_j) = \mathbf{1}U_i. * U_j\mathbf{1}^T \tag{7}$$

where $.*$ denotes element-wise multiplication and $\mathbf{1}$ is the all-one vector.

Extending to Binary Data. The sequential data recorded in binary digits $\{0, 1\}$ are more encountered in clinical research, e.g. heart beating signal, signals from neurons. In terms of binary or discrete data, the traditional ways that minimize mean square error (MSE) as did on numerical sequences are infeasible. The traditional ways rely on the gradient of objective function for inference of parameters, while MSE from binary data is non-smooth and thus no gradients exist. LSM is extended to process binary data by replacing MSE with exponential van Rossum metric [23].

A general exponential van Rossum metric $\psi(t, t_0)$ can be formulated as:

$$\psi(t, t_0) = \begin{cases} -(t - t_0)\dfrac{e^{-(t-t_0)/\tau}}{\tau} & 0 \leq t < \Delta t + t_0 \\ +\infty & \text{otherwise} \end{cases} \tag{8}$$

where index t_0 is the expected index. Δt is a threshold, restricting the comparison to the affinity of t_0. Argument τ is a penalty on the deviation.

3 Experimental Study

This section presents experiments conducted on synthetic binary data and classifications on benchmark univariate and multivariate data. For a given task, the topology (200 interior nodes) and interior weights between nodes were initialized and kept fixed. In this way, the randomness in LSM was controlled as an invariant factor for comparison purpose. The strategy of restart was adopted in experiments[1].

The implementation of LSM made use of a software simulating the microcircuits of neural network–CSIM [19]. The parameters were set referring the attached examples.

[1] The source code is available from https://github.com/jyhong836/LSMModelSpace.

Table 1. The parameters and search ranges

Name	Parameters	Search range
DTW	γ	$\gamma \in \{10^{-6}, 10^{-5}, \cdots, 10^{1}\}$
AR	γ, ξ, p	$\gamma \in \{10^{-6}, 10^{-5}, \cdots, 10^{1}\}, \xi \in \{0.1, 0.2, \cdots, 0.9\}, p \in \{1, 2, \cdots, 10\}$
FK	*state*	*state* $\in \{1, 2, 3, 10\}$
	λ,γ	$\lambda \in \{10^{-6}, 10^{-5}, \cdots, 10^{1}\}, \lambda \in \{10^{-5}, 10^{4}, \cdots, 10^{1}\}$

γ is the parameter in the kernel function.
ξ and p are the weight and the order of the negative definite kernel.
state is the number of states of hidden Markov model defined in Fisher kernel.
λ is the parameter used in ridge regression.

In the implementation of the Gaussian mixture model, the number of Gaussian distribution was auto-determined by the method proposed in [10]. In the sampling, since there existed training sequences that were not sufficiently long, circular block bootstrap was applied. The block length was auto-determined by the method proposed in [15].

LIBSVM [2] was adopted in the classification algorithm. Multi-class data were classified via its default strategy, one-against-one.

The proposed methods were compared with common methods, including *Dynamic Time Warping* (DTW), *Autoregressive Kernel* (AR), *Fast Fisher Kernel* (Fisher), and Reservoir model (RV) proposed in [4,5].

The parameters in the proposed algorithms (regression parameter λ), support vector machine (bandwidth θ and cost $\mathcal{C}$), and the comparison algorithms were tuned with 5-fold-cross-validation[2]. The search ranges for the parameters are detailed in Table 1.

Three classification methods defined with Eqs. (4)–(7) are named as LSM with L^2 norm (L^2-LSM), LSM with Gaussian mixture model (Gaussian-LSM), LSM with sampling method (Sampling-LSM), and LSM with Fisher methodology (Fisher-LSM).

3.1 Synthetic Data

Synthetic binary data were generated following Poisson distribution $p(t) = \frac{\lambda^t e^{-\lambda}}{t!}$. The merit of using Poisson distribution is that it makes the events (bars in Fig. 2) evenly distributed and ensures that no events happen at the same time. The synthetic data were labeled into three classes. Different classes were generated under a slightly changed parameter setting.

For each parametric setting, the simulation lasted 2 s with time unit 10^{-3} s, generating a 2000-length sequence. We generated 55 sequences for each class. In addition, all the sequences were corrupted with Gaussian white noise ($mean = 0, \Sigma = 0.02\boldsymbol{I}$). The Eq. (8) was adopted as cost function in the training algorithm. Figure 2 demonstrates parts of the binary sequences of three classes. From this figure, it is not easy to distinguish class labels.

[2] The procedure of cross-validation keeps identical to [4] for comparison.

Fig. 2. The parts of synthetic binary sequences. The data were generated following Poisson distribution and were corrupted with additive Gaussian white noise. Horizontal axis denotes the index. Different classes are drew in different colors, and are separated by a dash line.

The model space in this experiment, which is populated by readout weights of fitting models, is depicted in Fig. 3. In order to visualize the model space, *multidimensional scaling* (MDS) was used to reduce its dimensionality. MDS keeps the original between-objective distance faithfully in a lower dimensional space. Although it was hard to distinguish class labels in the binary data as depicted in Fig. 2, after representing the sequences in the model space, they became separable in Fig. 3.

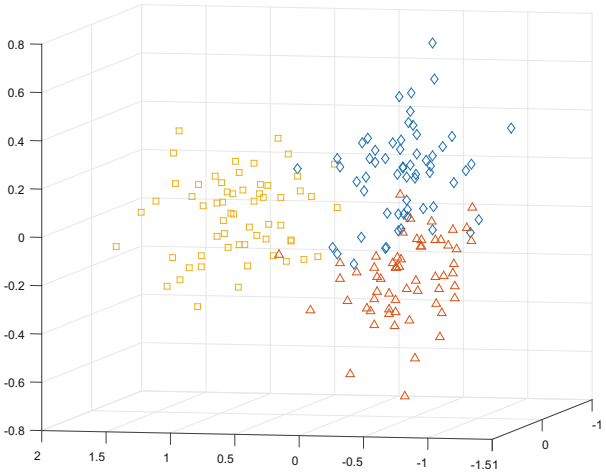

Fig. 3. The model space of synthetic binary data in a 3-dimensional coordinate. Parts of the data are depicted in Fig. 2. The model space was constructed by fitting LSMs to the binary data and extracting data-specific features, i.e. the readout weights, from LSMs. Each point offers representation to an individual binary sequence. Different classes are denoted with different markers.

The sensitivity of proposed method to the additive Gaussian noise was also investigated, in comparison with AR and *Fast Fisher Kernel* (Fisher)[3]. The

[3] The methodology of searching a global alignment is unsuitable for binary data, so the experiment of DTW was not reported.

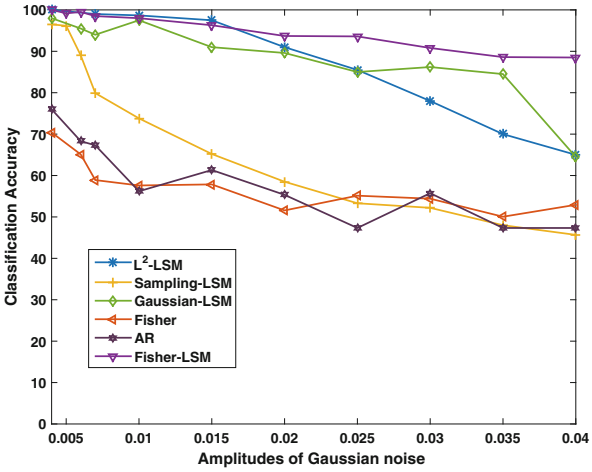

Fig. 4. The classification accuracy versus different amplitudes of Gaussian noise. The horizontal axis denotes the amplitude of noise, and the vertical axis is the classification accuracy on the synthetic data. Overall, the proposed methods demonstrate clear advantages in handling binary sequences in this experimental setting.

classifications were conducted on data with various amplitudes of Gaussian noise. The experimental results are depicted in Fig. 4.

An overall advantage can be observed from Fig. 4. Not surprisingly, Fisher-LSM has the best performance in terms of classification accuracy and robustness to the noise among all the methods. Fisher-LSM assumes that the deviation between observation and true value follows zero-mean Gaussian distribution, which coincides with the noise used in this experiment. Sampling-LSM shows to be less robust to the added noise. Its classification accuracy drops after corrupting data with noise. But as the amplitude of noise grows, its influence on the performance of Sampling-LSM decreases.

3.2 Benchmark Data

The benchmark data sets were obtained from UCR time series classification archive [7] and UCI machine learning repository[4]. Table 2 gives a summary of all data sets. In order to eliminate the influence of different units, all data sets were rescaled into interval $[-1, 1]$.

The experimental results of 5 runs are listed in Table 3. From this table, A general advantage of classifications carried out in the model space of LSM over comparison algorithms can be observed. Among all the proposed methods, learning based on sampling achieved the best performance. The better performance

[4] EEG data was obtained from UCI machine learning repository, https://archive.ics.uci.edu/ml/datasets/EEG+Database. And it was preprocessed via *Principle Component Analysis* to reduce its dimensionality.

Table 2. Summary description of univariate data sets.

Name	Instances	Length	Classes
Beef	60	470	6
Car	120	576	4
OSULeaf	220	240	6
Adiac	780	176	37
FISH	175	175	7
OliveOil	60	570	4
EEG	60	60	2

Table 3. Classification accuracy of Dynamic Time Warping (DTW), Autoregressive Kernel (AR), Fisher Kernel Learning (Fisher), Reservoir Model (RV), L^2-LSM, Gaussian-LSM, sampling-LSM and Fisher-LSM. The best results are marked in bold.

Name	DTW	AR	Fisher	RV [4]	L^2-LSM	Gaussian-LSM	Fisher-LSM	Sampling-LSM
Beef	66.67	56.67	58.00	**86.67**	60.00	46.67	53.3	76.67
Car	73.3	60.0	65.00	86.67	78.33	61.67	70.00	**90**
OSULeaf	62.15	73.33	54.96	64.59	72.31	69.00	69.01	**75.20**
Adiac	65.47	64.54	68.03	76.73	76.63	**78.10**	57.54	76.98
FISH	69.86	51.43	57.14	85.71	**89.71**	85.71	68.57	87.43
OliveOil	83.33	42.15	56.67	**90.00**	76.67	80.00	73.33	86.67
EEG	38.0	50.0	50.0	-	48.33	51.67	61.67	**63.33**

of sampling-LSM is largely contributed from the weak hypothesis it imposed on the probability distribution $\mu(\cdot)$. However, Fisher-LSM underperformed on all sequences. A possible reason for its deficiency lies in the pre-assumption over the deviation. When strong autocorrelation exists, the assumption of zero-mean Gaussian noise is unlikely to be true. Compared with good performance achieved on binary sequences, it is more encouraged to be used on binary or discrete sequences.

As the number of Gaussian distributions was auto-determined in Gaussian-LSM and bootstrap was adopted in Sampling-LSM, the computational complexities of proposed approaches are difficult to analyze. We adopted experiments to illustrate the actual time consumption on benchmark data. Experiments were conducted on a sequential data set PEMS[5]. By truncating the sequences and recording the time consumed in obtaining dissimilarities between pairwise sequences, we obtained tuples of time consumption[6] versus length of sequence. And the results are plotted in Fig. 7.

[5] PEMS was obtained from UCI machine learning repository. The sequences in PEMS were vectorized to be sufficiently long.

[6] The computational environment is Windows 7 with Intel Core i5 Duo 3.2 GHz CPU and 8 G RAM.

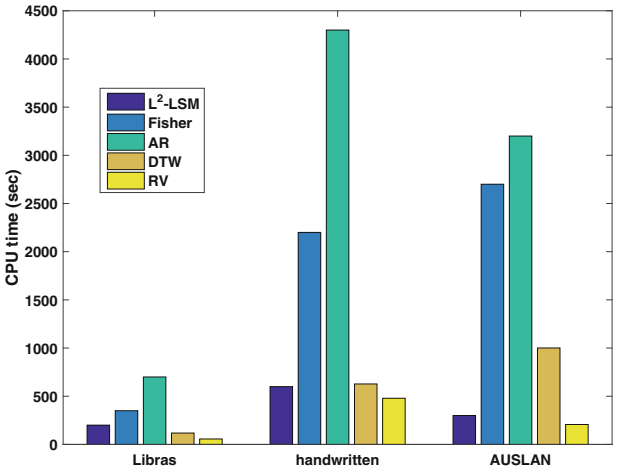

Fig. 5. The time consumptions on three multivariate data sets. The vertical axis denotes the time consumption. It is measured in the unit of CPU time (sec).

In the Fig. 7, the time consumptions of all proposed approaches grow slowly after the sequential length becomes large (beyond 1800). The lines of Gaussian-LSM and L^2-LSM grow in a similar pattern. However, Sampling-LSM maintains a (roughly) consistent time usage, even when the training sequences are short. The reason is, in order to compensate the approximation loss when the training sequences were not sufficiently long, more sampling had to been done. The computation of Fisher-LSM involves matrix multiplication, which makes it grow (roughly) linearly with the sequential length in our experiments (not shown).

In contrast, the time complexity of DTW is $O(m_i m_j)$, where m_i is the length of i-th sequence. An improved variation [13] speeds up DTW by using piece-wise line of length c to approximate the time series. It is reported to have time complexity $O(\frac{m_i m_j}{c^2})$. Autoregressive kernel [8] have time complexity $(m_i + m_j - 2p)^3$, where p is the order of employed model, far less than $min(m_i, m_j)$. So compared with the above algorithms, Gaussian-LSM and L^2-LSM show computational advantage.

Multivariate Sequences. The experiments of classifications on three multivariate data sets, *Brazilian sign language* (Libras), *handwritten* characters and Australian language of signs (AUSLAN) were conducted. Notably, *handwritten* and AUSLAN are also variable-length. The summary of three data sets is listed in Table 4.

In this experiment, we compared L^2-LSM against comparison algorithms in multivariate data. And the experimental results of 5 runs are plotted in Fig. 6.

From Fig. 6, L^2-LSM outperforms all comparison algorithms on *handwritten*. It also gains a slight advantage on data set *Libras*. On high dimensional data set AUSLAN, L^2-LSM only surpasses AR. The hypothesis of uniform distribu-

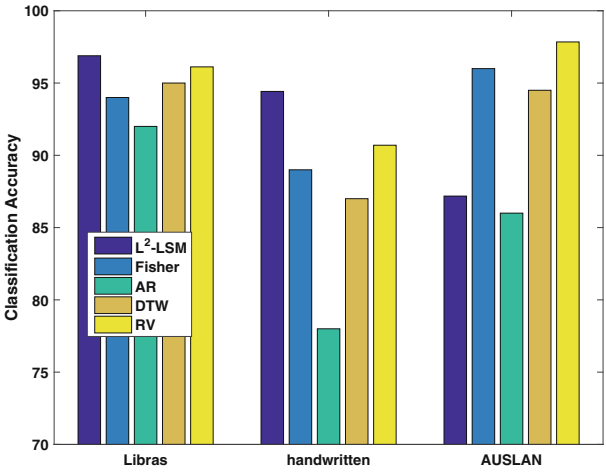

Fig. 6. The classification accuracies carried out on three multivariate data sets.

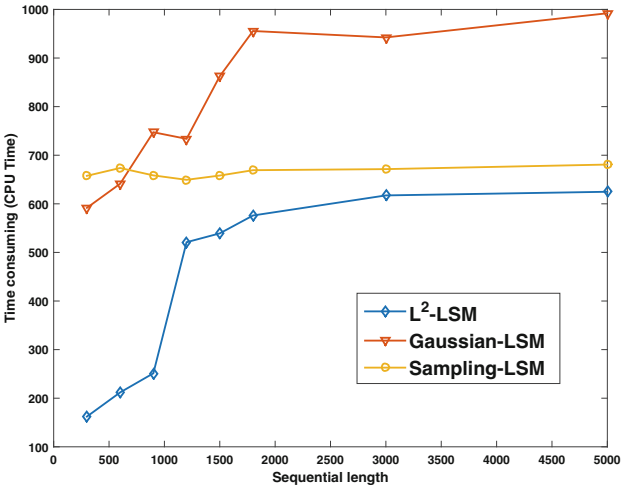

Fig. 7. The time consumptions under different length of sequential data. The vertical axis denotes the time consumption. It is measured in the unit of CPU time (sec). The horizontal axis denotes the sequential length.

tion fails to hold in the high dimensional data set AUSLAN, which leads to a suboptimal result for L^2-LSM.

The time consumption of L^2-LSM and comparison algorithms are plotted in Fig. 5. Generally, L^2-LSM demonstrates an advantage on its efficiency. It has comparative time usage with RV. On high dimensional data set AUSLAN, L^2-LSM and RV build a classifier using less time over other algorithms, and the difference within these two algorithms is not obvious.

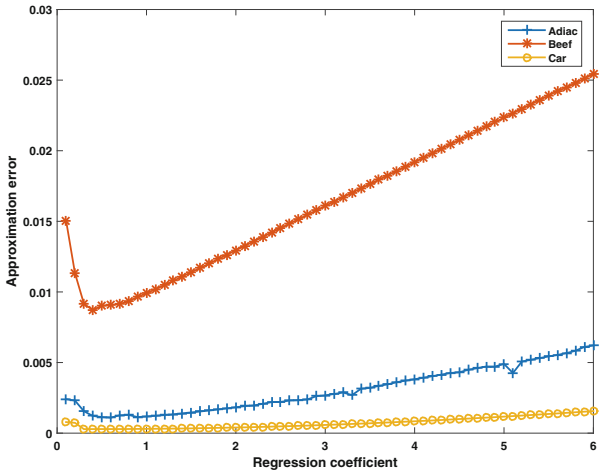

Fig. 8. The approximating errors under different regression coefficients ξ.

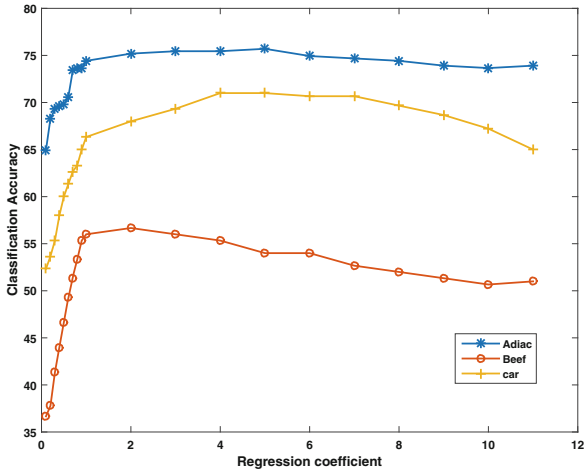

Fig. 9. The classification accuracies under different regression coefficients ξ.

Parametric Sensitivity Analysis. The performance achieved in the model space of LSM are jointly determined by two factors, i.e. the representations offered by LSMs to the sequences and the separation of LSMs in the corresponding space. An unsettled issue is the relationship between these two goals. In the approach, *regression coefficient* ξ is the parameter which needs careful tuning for a better trade-off between the approximation to the sequences and separation of LSMs in the model space.

In this experiment, three data sets were used as benchmark data sets. And experiments were conducted with different settings of parameter ξ. For simplicity,

Table 4. The description of multivariate data sets.

Name	Dim	Len	Class	Train	Test
Libras	2	45	15	360	585
handwritten	3	60-182	20	600	2268
AUSLAN	22	45-136	95	600	1865

we assume a uniform distribution for $\mu(\cdot)$. The experimental result in terms of classification accuracies versus ξ is plotted in Fig. 9, and the approximation errors versus ξ are plotted in Fig. 8.

Compare Figs. 8 and 9, we can observe a higher classification accuracy and a lower approximation error are likely to occur jointly, which suggests that two goals may not be conflicting objectives with regard to ξ. A joint optimization procedure for ξ may be feasible.

4 Conclusion

This paper proposes model space learning for the sequential data on the basis of LSM. LSM is used as a universal approximating tool to fit the conditional probability of a sequence. The models offer representations for sequences of training data. As a result, the learning strategy is carried out in the model space instead of on the original data. From the experiments, the benefits brought by replacing the "memoryless" response function with node that has its own "history" are clear. Fisher-LSM is shown to be robust and effective on processing binary data. An overall improvement of classification accuracy on benchmark data has been observed via experiments. Sampling-LSM is encouraged when the dimensionality of training data is not high.

This paper also discusses measuring the dissimilarity between two LSMs in the model space. A set of models, instead of a single model, is used to give approximations to the training data. Learning in model space relaxes the requirement to use a single model to explain the whole data. The relationship between approximating capability to sequences and separation of LSMs is studied. The result shows the feasibility to implement joint optimization on two seemingly conflict targets.

In general, this paper proposes an approach to constructing data representation without need of assuming a parametric formulation. Its applications on lower dimensional data have been demonstrated to be effective. Promising future work includes improving the model space learning on high dimensional data without sacrificing its efficiency.

Acknowledgements. This work is supported by the National Ket Research and Development plan under Grant 2016YFB1000905, and the National Natural Science Foundation of China under Grants 91546116, 61511130083, 61673363. The authors would like to thank Dr. Hongfei Xing for her valuable comments.

References

1. Berndt, D.J., Clifford, J.: Using dynamic time warping to find patterns in time series. In: KDD Workshop, vol. 10, pp. 359–370 (1994)
2. Chang, C.C., Lin, C.J.: Libsvm: a library for support vector machines. ACM Trans. Intell. Syst. Technol. **2**(3), 27 (2011)
3. Chen, H., Tang, F., Tino, P., Cohn, A.G., Yao, X.: Model metric co-learning for time series classification. In: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, pp. 3387–3394. AAAI Press (2015)
4. Chen, H., Tang, F., Tino, P., Yao, X.: Model-based kernel for efficient time series analysis. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 392–400. ACM (2013)
5. Chen, H., Tino, P., Rodan, A., Yao, X.: Learning in the model space for cognitive fault diagnosis. IEEE Trans. Neural Netw. Learn. Syst. **25**(1), 124–136 (2014)
6. Chen, H., Tiňo, P., Yao, X.: Cognitive fault diagnosis in tennessee eastman process using learning in the model space. Comput. Chem. Eng. **67**, 33–42 (2014)
7. Chen, Y., Keogh, E., Hu, B., Begum, N., Bagnall, A., Mueen, A., Batista, G.: The UCR time series classification archive, July 2015. www.cs.ucr.edu/~eamonn/time_series_data/
8. Cuturi, M., Doucet, A.: Autoregressive kernels for time series. arXiv preprint arXiv:1101.0673 (2011)
9. Cuturi, M., Vert, J.P., Birkenes, O., Matsui, T.: A kernel for time series based on global alignments. In: IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 2, pp. 413–416 (2007)
10. Figueiredo, M.A.T., Jain, A.K.: Unsupervised learning of finite mixture models. IEEE Trans. Pattern Anal. Mach. Intell. **24**(3), 381–396 (2002)
11. Granger, C.W.J., Hatanaka, M., et al.: Spectral Analysis of Economic Time Series. Princeton University Press, Princeton (1964)
12. Jebara, T., Kondor, R., Howard, A.: Probability product kernels. J. Mach. Learn. Res. **5**, 819–844 (2004)
13. Keogh, E.J., Pazzani, M.J.: Scaling up dynamic time warping for datamining applications. In: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 285–289. ACM (2000)
14. Kitagawa, G.: A self-organizing state-space model. J. Am. Stat. Assoc. **93**, 1203–1215 (1998)
15. Lahiri, S.N.: Theoretical comparisons of block bootstrap methods. Ann. Stat. **27**, 386–404 (1999)
16. Maass, W., Natschläger, T., Markram, H.: Real-time computing without stable states: a new framework for neural computation based on perturbations. Neural Comput. **14**(11), 2531–2560 (2002)
17. Maaten, L.: Learning discriminative fisher kernels. In: Proceedings of the 28th International Conference on Machine Learning, pp. 217–224 (2011)
18. Müller, K.-R., Smola, A.J., Rätsch, G., Schölkopf, B., Kohlmorgen, J., Vapnik, V.: Predicting time series with support vector machines. In: Gerstner, W., Germond, A., Hasler, M., Nicoud, J.-D. (eds.) ICANN 1997. LNCS, vol. 1327, pp. 999–1004. Springer, Heidelberg (1997). doi:10.1007/BFb0020283
19. Natschläger, T., Markram, H., Maass, W.: Computer models and analysis tools for neural microcircuits. In: Kötter, R. (ed.) Neuroscience Databases, pp. 123–138. Springer, New York (2003)

20. Sahoo, D., Sharma, A., Hoi, S.C., Zhao, P.: Temporal kernel descriptors for learning with time-sensitive patterns. In: Proceedings of the First SIAM Conference on Data Mining (2016)
21. Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans. Acoust. Speech Sig. Process. **26**(1), 43–49 (1978)
22. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, New York (2004)
23. Van Rossum, M.C.: A novel spike distance. Neural Comput. **13**(4), 751–763 (2001)