Learning from Temporal Data through Learning in the Space of Dynamical Systems (non-parametric case)

Peter Tino
School of Computer Science
University of Birmingham
UK
Special thanks to ...

- Yuan Shen
- Huanhuan Chen
- Barbara Hammer
- Krasimira Tsaneva-Atanasova
- Ali Rodan
- Jochen Steil
- Nick Gianniotis
- Kerstin Bunte
- ...

...
"Complex" data, e.g.
- multivariate time series of variable length
- fMRI data measured on normalized cognitive tasks
- sensor signals...

We kind-of know what to do for static fix-dimensional vectorial data, but how to represent such complex data in order to usefully "learn" from it?

The critical notion of "smoothness" when learning from historic data - "close" things should lead to the same response "closeness" - distance, dot product, ...
Each data item (e.g. time series) is represented by a model that "explains" it.

Learning is formulated in the space of models (function space)

Similarity quantifications between the models not their (arbitrary) parametrizations!

(Non-parametric) inferential base model class
- flexible enough to represent variety of data items
- sufficiently constrained to avoid overfitting
NARMA task - illustrative example

NARMA sequences - orders 10, 20 and 30 - represented as state space models
Constructing the base model - state space model

\[
\begin{align*}
1 & \xrightarrow{a} 2 \\
1 & \xleftarrow{a} 2 \\
2 & \xrightarrow{b} 1 \\
2 & \xleftarrow{b} 1
\end{align*}
\]
State space model

State space models - a very general and powerful model class capturing temporal structure through the notion of information processing state (IPS).

IPS at time $t$ codes for the entire history of time series items seen up to time $t$.

IPS at time $t + 1$ codes for the entire history of time series items we have seen up to time $t + 1$. This is equivalent to IPS at time $t$ and input at time $t + 1$.

Recursive updates of IPSs
Discrete time state space model

States of the DS form IPS

These states code the entire history of sequence elements we have seen so far.
Picking the Right Base Model

In the finite-state example - the state space model was perfectly tailored for its intended purpose!

In general, we may not know (and typically this is the case!) the right state space structure that extracts from time series what is needed for the particular task:
- what are the appropriate IPS?
- What is their transition structure?

Since we are ML people, the answer seems to be obvious: Learn everything with our parametrized state space model!
... easier said than done ...
Non-Parametric Base Model, Long Time Series Data

Parametrized state space model - RNN

- $N$-dimensional continuous state space $[-1, 1]^N$

\[
\begin{align*}
\mathbf{x}(t) &= f(\mathbf{x}(t-1), \mathbf{u}(t)) \\
&= \sigma(R\mathbf{x}(t-1) + V\mathbf{u}(t) + \mathbf{b}), \\
\mathbf{y}(t) &= h(\mathbf{x}(t)) \\
&= W\mathbf{x}(t) + \mathbf{a}
\end{align*}
\]

- $\mathbf{x}(t) \in [-1, 1]^N$ - state vector at time $t$
- $\mathbf{u}(t)$ - input time series element at time $t$
- $f(\cdot)$ - state-transition function
- $\mathbf{y}(t)$ - output of the linear readout from the state space.
- $h(\cdot)$ - output readout function
Problems...

Problems with fitting the model to individual time series - information latching problem.

Even if we managed perfect model fitting so that each sequence is nicely represented
- how to define distance/similarity between state space models - driven dynamical systems?

Highly non-trivial - e.g. work in machine vision and video processing communities (low-dim (mostly linear) dynamical systems).
Simplified "general purpose" state space model

\[ \mathbf{x}(t) = \sigma(R\mathbf{x}(t-1) + \mathbf{V}\mathbf{u}(t) + \mathbf{b}), \]
\[ \mathbf{y}(t) = \mathbf{W}\mathbf{x}(t) + \mathbf{a} \]

Fix the DS part to a high-dimensional DS, but with very few free parameters (only 3!).

General purpose state space organization (dynamical filter) that allows for rich and redundant representations of input histories.

Learn the (static) linear readout only. Can be done efficiently. No problem with model distance calculations.
Problems solved

No information latching problem - we don’t need to learn the dynamical coupling.

No problem with calculating model distance - dynamical parts for all time series will be the same.

Model fitting can be done efficiently - linear regression for readout fitting.
Too good to be true?

Inspiration from information theory ("general purpose" information sources) - Observable Markovian IPS.

Simplest construction of IPS - based on concentrating on the very recent (finite) past, e.g. definite memory machines, finite memory machines, Markov models

Example:
Only the last 5 input symbols matter when reasoning about what symbol comes next

```
... 1 2 1 1 2 3 2 1 2 1 2 4 3 2 1 1
... 2 1 1 1 1 3 4 4 4 4 4 1 4 3 2 1 1
... 3 3 2 2 1 2 1 2 2 1 3 4 3 2 1 1
```

All three sequences belong to the same IPS “43211”
Variable memory length MM (VLMM)

Sophisticated implementation of potentially high-order MM

Take advantage of subsequence structure in data

Save resources - make memory depth context dependent

Use deep memory only when it is needed

Natural representation of IPS in form of Prediction Suffix Trees

Closely linked to the idea of universal simulation of information sources.
Prediction suffix tree (PST)

IFS are organized on nodes of PST
Traverse the tree from root towards leaves in reversed order
Complex and potentially time-consuming training

Each node $i$ has associated next-symbol probability distribution $P(\cdot|\ i\ )$
Contractive DS lead to Markovian IPS
Contractive DS lead to Markovian IPS

Mapping symbolic sequences into Euclidean space

Coding strategy is Markovian

Sequences with shared histories of recently observed symbols have close state space representations

The longer is the common suffix of two sequences, the closer are they mapped to each other

Add a simple output-generating readout on top of the fixed state dynamics!
More Formally...

Given an input symbol $s \in A$ at time $t$, and activations of recurrent units from the previous step, $x(t-1)$,

$$x(t) = f_s(x(t-1)).$$

This notation can be extended to sequences over $A$ the recurrent activations after $t$ time steps are

$$x(t) = f_{s_1...s_t}(x(0)).$$

If maps $f_s$ are contractions with Lip. constant $0 < C < 1$,

$$\|f_{vq}(x) - f_{wq}(x)\| \leq C|q| \cdot \|f_v(x) - f_w(x)\| \leq C|q| \cdot \text{diam}(\mathcal{X}).$$
Theoretical grounding

Theorem (Hammer & Tino):

- Recurrent networks with contractive transition function can be approximated arbitrarily well on input sequences of unbounded length by a definite memory machine. Conversely,

- every definite memory machine can be simulated by a recurrent network with contractive transition function.

Hence, contractive RNNs (e.g. initialized with small weights) have Markovian architectural bias.
Learnability (PAC framework)

The architectural bias emphasizes one possible region of the weight space where generalization ability can be formally proved.

Standard RNN are not distribution independent learnable in the PAC sense if arbitrary precision and inputs are considered.

Theorem (Hammer & Tino):

- Recurrent networks with contractive transition function with a fixed contraction parameter fulfill the distribution independent UCED property and so

- unlike general recurrent networks, are distribution independent PAC-learnable.
State space complexity $\leftrightarrow$ sequence complexity

Natural ”general purpose” state space organization:
complexity of the driving input time series (topological entropy) is directly reflected in the complexity of the state space activations (fractal dimension).

Theorem (Tino & Hammer):
Recurrent activations inside contractive RNN form fractal clusters the dimension of which can be bounded by the scaled entropy of the underlying driving source. The scaling factors are fixed and are given by the RNN parameters.

Can be extended to entropy spectra and multifractal spectra.
Fix the DS to a contractive system!

Fix the DS part to a randomly constructed contraction.

Only train the simple linear readout output map. Can be done efficiently.

Reservoir computation models.
Cycle reservoir with regular jumps

Still a simple deterministic construction - only 3 free parameters for the high-dim dynamical system!
Reservoir characterizations - memory capacity

Memory capacity - [Jaeger] Measures richness of the DS state space representations.
How much information about the past is stored in the states?

Make the memory task difficult: DS is driven by a univariate i.i.d. input signal $u(t)$.

For a given delay $k > 0$, find optimal readout for recalling $u(t - k)$ after seeing the input stream $...u(t - 1)u(t)$ up to time $t$.

Goodness of recall - squared correlation coefficient between the desired output $u(t - k)$ and the observed network output $y(t)$
Memory capacity of dynamical systems

\[ MC_k = \frac{Cov^2(u(t - k), y(t))}{Var(u(t)) \ Var(y(t))}, \]

where \( Cov \) denotes the covariance and \( Var \) the variance operators. Memory Capacity is then given by:

\[ MC = \sum_{k=1}^{\infty} MC_k. \]

Theorem (Jaeger): MC of every smooth \( N \)-dimensional DS is \( \leq N \).
Memory capacity of SCR

MC of linear $N$-dimensional Simple Cycle Reservoir can be made arbitrarily close to $N$.

Theorem (Tino & Rodan): Consider a linear SCR network with reservoir weight $0 < r < 1$ and an input weight vector such that the reservoir activations are full rank. Then, the SCR MC is equal to

$$MC = N - (1 - r^{2N}).$$
Distance in the readout function space

- Two readouts $h_i$ and $h_j$ from the same $N$-dimensional DS obtained on two time series $u_i, u_j$. Squared $L_2$ norm:

$$d_2(h_i, h_j) = \int_C \| h_i(x) - h_j(x) \|^2 \, d\mu(x)$$

where $\mu(x)$ is the probability density measure on the input domain, and $C = [-1, +1]^N$.

- Time series kernel:

$$\mathcal{K}(u_i, u_j) = \exp \{-\gamma \cdot d_2(h_i, h_j)\}$$
Model distance with uniform measure

\[ h_1(x) = W_1 x + a_1, \]
\[ h_2(x) = W_2 x + a_2. \]

\[
d_2(h_1, h_2) = \int_{[-1,1]^N} \| h_1(x) - h_2(x) \|^2 \, dx
= \frac{2N}{3} \sum_{i,j} \omega_{i,j}^2 + 2^N \| a \|^2
\]

where \( \omega_{i,j} \) are the elements of \( W_1 - W_2 \), \( a = a_1 - a_2 \).

Note:

\[
\frac{1}{3} \sum_{i,j} \omega_{i,j}^2 + \| a \|^2
\]
Non-uniform state distribution

- For $K$-component Gaussian mixture model

$$\mu(x) = \sum_{i=1}^{K} \alpha_i \mu_i(x|\eta_i, \Sigma_i)$$

The model distance can be obtained as follows:

$$d_2(h_1, h_2) = \sum_{i=1}^{K} \alpha_i \left\{ \text{trace}(W^T W \Sigma_k) + a^T a + \eta_k^T W^T W \eta_k + 2a^T W \eta_k \right\}$$

- Sampling approximation

$$d_2(h_1, h_2) \approx \frac{1}{m} \sum_{i=1}^{m} \| h_1(x(i)) - h_2(x(i)) \|^2.$$
Three Time Series Kernels

- Uniform state distribution
  - Reservoir Kernel: RV

- Non-uniform state distribution
  - Reservoir kernel - sampling: SamplingRV
  - Reservoir kernel - Gaussian mixture models: GMMRV
## Benchmark Data

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Length</th>
<th>Classes</th>
<th>Train</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbols</td>
<td>398</td>
<td>6</td>
<td>25</td>
<td>995</td>
</tr>
<tr>
<td>OSULeaf</td>
<td>427</td>
<td>6</td>
<td>200</td>
<td>242</td>
</tr>
<tr>
<td>Oliveoil</td>
<td>570</td>
<td>4</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>Lighting2</td>
<td>637</td>
<td>2</td>
<td>60</td>
<td>61</td>
</tr>
<tr>
<td>Beef</td>
<td>470</td>
<td>6</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>Car</td>
<td>576</td>
<td>4</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>Fish</td>
<td>463</td>
<td>8</td>
<td>175</td>
<td>175</td>
</tr>
<tr>
<td>Coffee</td>
<td>286</td>
<td>2</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td>Adiac</td>
<td>176</td>
<td>37</td>
<td>390</td>
<td>391</td>
</tr>
</tbody>
</table>
## Results: Classification Accuracy

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

- Reservoir readouts can be trained in an on-line fashion, e.g. using Recursive Least Squares. This enables us to construct and refine reservoir kernels.

\[
\begin{align*}
\mathbf{w}_{opt} &= \mathbf{w}_0 + \mathbf{k}(\mathbf{b} - \mathbf{a}^T \mathbf{w}_0) \\
\mathbf{k} &= \frac{1}{\lambda + \mathbf{a}^T (\mathbf{A}_0^T \mathbf{A}_0)^{-1} \mathbf{a}} \left(\mathbf{A}_0^T \mathbf{A}_0\right)^{-1} \\
(\mathbf{A}_0^T \mathbf{A}_0)^{-1} &= [\lambda^{-1} \mathbf{I} - \mathbf{k} \mathbf{a}^T] (\mathbf{A}_0^T \mathbf{A}_0)^{-1}
\end{align*}
\]

\(\lambda\): forgetting factor
PEMS-SF is 15 months (Jan. 1st 2008 to Mar. 30th 2009) of daily data from the California Department of Transportation PEMS.

The data describes the occupancy rate of different car lanes of San Francisco bay area freeways.

PEMS-SF with 440 time series of length 138,672.
The computational complexity of our kernels is $O(l)$, where $l$ is the length of time series: Scale **lineally** with the length of time series.

1. RV kernel achieves 86.13% accuracy.
2. The best accuracy reported are 82%–83% for AR, global alignment, spline smoothing and Bag of vectors kernels.
Incremental One Class Learning

1. “Normal data” - apply reservoir model in the sliding window (size $m$) in the first $t$ steps

2. Train one-class SVM

3. If needed, incrementally create new clusters of models (signal ‘regimes’).
## Incremental One Class Learning - NARMA task

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Classes</th>
<th>Precision</th>
<th>Recall</th>
<th>Specificity</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBscan</td>
<td>4</td>
<td>0.6690*</td>
<td>0.7650*</td>
<td>0.8825*</td>
</tr>
<tr>
<td>ARMAX-OCS</td>
<td>5</td>
<td>0.9354*</td>
<td>0.9229*</td>
<td>0.9615*</td>
</tr>
<tr>
<td>RC-OCS</td>
<td>3</td>
<td>0.9637*</td>
<td>0.9615*</td>
<td>0.9808*</td>
</tr>
<tr>
<td>DRC-OCS(Sampling)</td>
<td>3</td>
<td>0.9683*</td>
<td>0.9692*</td>
<td>0.9914</td>
</tr>
<tr>
<td>DRC-OCS</td>
<td>3</td>
<td><strong>0.9861</strong></td>
<td><strong>0.9858</strong></td>
<td><strong>0.9929</strong></td>
</tr>
</tbody>
</table>
### Incremental One Class Learning - Van der Pol

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Classes</th>
<th>Precision</th>
<th>Recall</th>
<th>Specificity</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBscan</td>
<td>10</td>
<td>0.7629*</td>
<td>0.6842*</td>
<td>0.8018*</td>
</tr>
<tr>
<td>ARMAX-OCS</td>
<td>2</td>
<td>0.4309*</td>
<td>0.4880*</td>
<td>0.7868*</td>
</tr>
<tr>
<td>RC-OCS</td>
<td>6</td>
<td>0.9606*</td>
<td>0.9583*</td>
<td>0.9861</td>
</tr>
<tr>
<td>DRC-OCS (Sampling)</td>
<td>5</td>
<td>0.9617*</td>
<td>0.9726</td>
<td>0.9819*</td>
</tr>
<tr>
<td>DRC-OCS</td>
<td>5</td>
<td>0.9736</td>
<td>0.9731</td>
<td>0.9910</td>
</tr>
</tbody>
</table>

*Note: The asterisk (*) indicates statistical significance.*
# Incremental One Class Learning - Three tank

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Classes</th>
<th>Precision</th>
<th>Recall</th>
<th>Specificity</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBscan</td>
<td>14</td>
<td>0.8742*</td>
<td>0.7561*</td>
<td>0.9253*</td>
</tr>
<tr>
<td>ARMAX-OCS</td>
<td>5</td>
<td>0.9914*</td>
<td>0.9923</td>
<td>0.9984</td>
</tr>
<tr>
<td>RC-OCS</td>
<td>9</td>
<td>0.9182*</td>
<td>0.8788*</td>
<td>0.9596*</td>
</tr>
<tr>
<td>DRC-OCS(Sampling)</td>
<td>7</td>
<td><strong>0.9940</strong></td>
<td><strong>0.9949</strong></td>
<td><strong>0.9988</strong></td>
</tr>
<tr>
<td>DRC-OCS</td>
<td>10</td>
<td>0.9931</td>
<td>0.9931</td>
<td>0.9977</td>
</tr>
</tbody>
</table>

Tino et al. () Machine Learning in the Model Space
## Incremental One Class Learning - Barcelona Water System

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Classes</th>
<th>Precision</th>
<th>Recall</th>
<th>Specificity</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBscan</td>
<td>61</td>
<td>0.8019*</td>
<td>0.7326*</td>
<td>0.8654*</td>
</tr>
<tr>
<td>ARMAX-OCS</td>
<td>57</td>
<td>0.7826*</td>
<td>0.7419*</td>
<td>0.8237*</td>
</tr>
<tr>
<td>RC-OCS</td>
<td>44</td>
<td>0.8913*</td>
<td>0.8942*</td>
<td>0.9263*</td>
</tr>
<tr>
<td>DRC-OCS(Sampling)</td>
<td>39</td>
<td>0.9219*</td>
<td>0.9310*</td>
<td>0.9513*</td>
</tr>
<tr>
<td>DRC-OCS</td>
<td>48</td>
<td><strong>0.9538</strong></td>
<td><strong>0.9640</strong></td>
<td><strong>0.9871</strong></td>
</tr>
</tbody>
</table>
Interested?