Piecewise 1-Dim Self Organizing Map

P1D-SOM

A fast Competition Approach for Multivariate Data Applications

Alaa Sagheer\textsuperscript{1} \quad Naoyuki Tsuruta\textsuperscript{2}

Rin Ichiro Taniguchi\textsuperscript{1}

MLSS2006-Taipei

\textsuperscript{1}Kyushu University - \textsuperscript{2}Fukuoka University - Japan
AGENDA

- Introduction
  - Why/When Fast Competition SOM?
- Conventional SOM
  - Features and Advantages
  - Drawback and Challenge
  - Motivation
- Previous Works
- P1D-SOM Algorithm
  - Training Phase
  - Recognition Phase
  - Example
- Experimental Results
- Conclusion
- Future Work

MLSS2006-Taipei
Introduction

....In Pattern Recognition domain..

- It is necessary to achieve Fast Feature Extraction

- Then you need Dimensionality Reduction

  Principal Components Analysis

  1. Unsupervised Linear Mapping
  2. Minimize Information Loss
  3. Stability with Low Cost

MLSS2006-Taipei
Principal Components Analysis
Since PCA uses *Mean* and *Covariance Matrix*... Then it is accurate only if the data distribution is *Gaussian.*
Curved Subspace

Distribution of 2000 Lip Image

MLSS2006-Taipei
Curved Subspace

Winding!

First PC

PCA

Third PC
**Motivation -1**

Then...Our Target is...

A *Non-Linear Mapping* can be used for *Feature Extraction* and *Dimensionality Reduction* in *Multivariate Data Applications*
Kernel PCA*

- Non Linear Mapping capable to construct the ordered principal components.

.....However

- No adequate way for exploiting such nonlinear mapping**.

- Much pre-experiments computations**, 
  - Solve Eigen-equation for the covariance matrix of input, 
  - Calculation to a kernel function between “objective data” and “all training data” to get a principal component score.

** Saegusa et al., Neurocomputing, vol.61, 2004
Kohonen SOM

- **SOM** is an *Unsupervised Competitive* learning algorithm.
- It capable to map a multi-dimensional input space into a NN map (called codebook) includes a lot of nodes (neurons)...
- Its **Codebook** has two characteristics:
  1. Its PDF is a good approximation for training data.
  2. The *Topographical Order* of training data is *preserved* in the codebook.
- It can be used in
  - **Visualization** and
  - **Cluster Analysis**
Kohonen SOM

- **Input Data:** \( X = \{ x_i, 1 < i < M \} \) s.t. \( x_i = (x_i^{(l)})_{1<l<n} \in \mathbb{R}^n \)

- **Codebook:** \( W = \{ w_j, 1 < j < N \} \) s.t. \( N < M. \)

- **In “each” training step, “two” steps are repeated:**
  1. **Using the winner-take-all rule to find the winner** \( c \)
     \[ \| x_i - w_c \| = \min_u (\| x_i - w_u \|) \]
  2. **Update the codebook**
     \[ w_u(t+1) = w_u(t) + h_{cu}(t)[x_i(t) - w_u(t)] \]
...HOWEVER!
Practically...For large map size...For Multivariate Data
SOM Computation is Complex!

This is because, “each” learning pass requires a computation of the distance of the “current” sample to “all” nodes in the map*

Please give me more Evidence?

<table>
<thead>
<tr>
<th># Dim</th>
<th>Time</th>
<th>Training Data</th>
<th>Testing Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Word</td>
<td>Sent</td>
</tr>
<tr>
<td>2**</td>
<td>92.4</td>
<td>76.9</td>
<td>61.1</td>
</tr>
<tr>
<td>3***</td>
<td>656.7</td>
<td>88.5</td>
<td>74.1</td>
</tr>
<tr>
<td>4</td>
<td>5312.8</td>
<td>92.3</td>
<td>85.2</td>
</tr>
</tbody>
</table>

(9x8), (9x8x7) and (9x8x7x6)

* Lawrence et al., *IEEE Trans. on Neural Networks*, vol 8, no. 1, 1997.
Motivation-2

Our target now became... Giving a large-size high-dimensional data space...

How can we get a Well-Ordered Feature Map using SOM

1. Without consuming much Time and Efforts, and

2. Preserving the SOM Performance and Quality
Previous Works!

.....Three Main Categories

• Tree SOMs Structure,
• Hierarchical SOMs Structure, and
• Randomized Structure SOM
**Tree* Structure**

- **Advantages**
  - Ability to handle Complex Data Sets*

- **Disadvantages**
  - Repeating input Data,
  - Bias brought by the tree structure,
  - The lateral search destroys the interlayer topology

---


** Jin et al., *Information Science*, vol.163,2004 and Xu et al., *ISCAS04*, vol. 5,2004
Hierarchical Structure

• Advantages
  - Doubling Size Periodically During Training
  - Accuracy Improvement

• Disadvantages
  - Assuming Optimal Topological Order before Doubling*,
  - The Hierarchy Map loses Original Feature Map Properties**

* Rauber et al., *The IEEE Trans on Neural Networks*, vol.13, no 6, Nov. 2002

Randomized Structure

• Advantages
  - Real Time Recognition,

• Disadvantages
  - The Random Mapping is Linear*
  - Parameterized method**

** Tobely et al., *ICONIP00*, vol.2, Nov. 2000
P1D-SOM

ALGORITHM
Original SOM

\[ u = \left\{ u_{d_1,d_2,...,d_N} \mid d_j = 1, 2, ..., D_j, \ j = 1, ..., N \right\} \]

Each neuron \( u_{d_1,d_2,...,d_N} \) has a codebook vector \( W_{d_1,d_2,...,d_N} \)

- The **Winner** neuron is decided according to **winner-take-all** rule:

\[ \| x_i - w_{c_1,c_2,...,c_N} \| = \min_{d_j} \left( \| x_i - w_{d_1,d_2,...,d_N} \| \right) \]

**Computational Complexity of Recognition is**

\[ O (D_1 \cdot D_2 \cdot D_3 \cdot \ldots \cdot D_N) \]
**P1D-SOM Algorithm**

- Greatest variance of the data comes to lie on the low-order axes
- Low order axes contain the most important aspects of feature space

- The new SOM consists of $N$-sequence of 1-dim SOMs

\[
    u_1 = \{ u_{1, d_1} \mid d_1 = 1, 2, \ldots, D_1 \} \\
    u_2 = \{ u_{2, d_1, d_2} \mid d_2 = 1, 2, \ldots, D_2 \} \\
    \quad \vdots \\
    u_N = \{ u_{N, d_1, \ldots, d_{N-1}} \mid d_N = 1, 2, \ldots, D_N \} \\
\]

...Such that $u_1$ matches first PC, $u_2$ matches second PC and so on
I. Learning Phase

The learning process is described as a recursive function call \( \text{Learn}(n, u_{n,d_n}, C) \)

\[
\begin{align*}
\text{If } n &= N \quad \{ \text{Train } u_{N,d_n} \}. \quad \text{That is, for each } (d_N = 1, 2, \ldots, D_N), \text{ apply winner-take-all}\n\end{align*}
\]

\[
\| x_i - w_{N,d_N}(t) \|
\]

\[
\}
\]

else {

1- Train \((N-n+1)\) dimensional original SOM

\[
u = \{ u_{d_n,d_{n+1},\ldots,d_N} \mid d_i = 1, 2, \ldots, D_i, i = n, n+1, \ldots, N \}\]

2- Regard to the center column \( p_n = \{ u_{d_n,D_{n+1}/2,\ldots,D_N/2} \mid d_n = 1, 2, \ldots, D_n \}\)

as the principal component at the winners list \( C \) and copy it into \( u_{n,d_n} \)

3- For each \( d_{n+1} \), do

\[
\text{Learn}(n+1, u_{n+1,d_{n+1}}, C) \}
\]

MLSS2006-Taipei
I. Learning Phase

\[
\text{Learn}\left(n, u_{n,d_n}, C\right)
\]

\[
\text{Learn}\left(n + 1, u_{n+1,d_{n+1}}, C\right)
\]

\[D_n\text{ times}\]

The computational complexity to train \(u_{n+1}\) is \(1/D_n\) of \(u_n\)

Total Computation Complexity

\[
\text{CC} \left( 1 + 1/D_1 + 1/D_1D_2 + \cdots + 1/\prod_i D_i \right) \approx \text{CC}
\]
II. Recognition Phase

\[ \|x_i - w_{1,c_1}\| = \min_{d_1} \left( \|x_i - w_{1,d_1}\| \right) \]

\[ \|x_i - w_{2,c_1,c_2}\| = \min_{d_2} \left( \|x_i - w_{2,c_1,d_2}\| \right) \]

\[ \vdots \]

\[ \|x_i - w_{N,c_1,c_2,\ldots,c_N}\| = \min_{d_N} \left( \|x_i - w_{N,c_1,c_2,\ldots,c_{N-1},d_N}\| \right) \]

**Computational Complexity**

\[ O(D_1 + \cdots + D_N) \]

MLSS2006-Taipei
Example
Example

Consider $D_1=30$, $D_2=20$ and $D_3=10$

I- Training Phase

1- For each input sample $x_i$, train the normal SOM in 3-dim using

$$\left\| x_i - w_{3,c_1,c_2,c_3} \right\| = \min_{d_i} \left( \left\| x_i - w_{3,d_1,d_2,d_3} \right\| \right)$$

Update $w_{3,d_1,d_2,d_3}$

2- For each $d_1$, Put

$$U_{1,d_1} = U_{3,d_1,d_2/2,d_3/2}$$

Then

$$\left\| x_i - w_{1,c_1} \right\| = \min_{d_i} \left( \left\| x_i - w_{1,d_1} \right\| \right)$$

and

$$\left\| x_i - w_{3,c_1,c_2,c_3} \right\| = \min_{d_i} \left( \left\| x_i - w_{3,c_1,d_2,d_3} \right\| \right)$$

Update

MLSS2006-Taipei
3- For each $d_1$ and $d_2$, Put

$$u_{2,d_1,d_2} = u_{3,d_1,d_2,d_3}/2$$

Then

$$\|x_i - w_{1,c_1}\| = \min_{d_1}(\|x_i - w_{1,d_1}\|)$$

$$\|x_i - w_{2,c_1,c_2}\| = \min_{d_2}(\|x_i - w_{2,c_1,d_2}\|)$$

$$\|x_i - w_{3,c_1,c_2,c_3}\| = \min_{d_3}(\|x_i - w_{3,c_1,c_2,d_3}\|)$$
Example

II- Recognition Phase

1- First winner “c” is picked from first SOM (or first PC)

\[ u_{1,c} = \min_{l} \left( \| x_i - w_{1,l} \| \right) \]

2- Second winner “d” is picked from second SOM (or second PC)

\[ u_{2,c,d} = \min_{j} \left( \| x_i - w_{2,c,j} \| \right) \]

3- Third winner “m” is picked from third SOM (or third PC)

\[ u_{3,c,d,m} = \min_{k} \left( \| x_i - w_{3,c,d,k} \| \right) \]
II- Recognition Phase

• Using Original SOM

\[30 \times 20 \times 10 \approx 6000\]

• Using P1D-SOM

\[30 + 20 + 10 \approx 60\]

It is 1%
EXPERIMENTAL RESULTS

LIP-READING

MLSS2006-Taipei
Database

- Arabic (6480) Japanese (5670) image,
- 9 subjects (6 for training, 3 for test),
- Input image size is 160x120 pixels,
**Feature Extraction Time** (second)

- **SOM vs. P1D-SOM**

### Arabic Database

<table>
<thead>
<tr>
<th># Dim</th>
<th>SOM</th>
<th>P1D-SOM</th>
<th>SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>92.4</td>
<td>15.1</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>656.7</td>
<td>36</td>
<td>18</td>
</tr>
<tr>
<td>4</td>
<td>5312.8</td>
<td>101.9</td>
<td>52</td>
</tr>
</tbody>
</table>

(9x8), (9x8x7) and (9x8x7x6)

### Japanese Database

<table>
<thead>
<tr>
<th># Dim</th>
<th>SOM</th>
<th>P1D-SOM</th>
<th>SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>144.7</td>
<td>25.2</td>
<td>5.7</td>
</tr>
<tr>
<td>3</td>
<td>713.6</td>
<td>51.4</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>6652.1</td>
<td>154.7</td>
<td>43</td>
</tr>
</tbody>
</table>

(11x9), (11x9x7) and (11x9x7x5)

SR: Speedup Ratio

MLSS2006-Taipei
**Feature Extraction Time** (second)

### FDCT vs. P1D-SOM

<table>
<thead>
<tr>
<th>Data Base</th>
<th>FDCT (second)</th>
<th>P1D-SOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Japanese</td>
<td>302.8</td>
<td>47.2</td>
</tr>
<tr>
<td>Arabic</td>
<td>374.4</td>
<td>58</td>
</tr>
</tbody>
</table>

Fast DCT is a fast known algorithm used for image compression. We already showed that FDCT is faster than SOM and HCM in [1]

Feature Extraction Time (second)

• For one Image!

160x120 gray image

<table>
<thead>
<tr>
<th># Dim</th>
<th>SOM</th>
<th>P1D-SOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.104</td>
<td>0.005</td>
</tr>
<tr>
<td>4</td>
<td>0.785</td>
<td>0.065</td>
</tr>
</tbody>
</table>
## Recognition Accuracy

### Arabic Database (%)

<table>
<thead>
<tr>
<th>Model</th>
<th>Training Data</th>
<th>Testing Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Word</td>
<td>Sent</td>
</tr>
<tr>
<td>SOM</td>
<td>82.7</td>
<td>64.8</td>
</tr>
<tr>
<td>P1D-SOM</td>
<td>85.3</td>
<td>70.4</td>
</tr>
</tbody>
</table>

### Japanese Database (%)

<table>
<thead>
<tr>
<th>Model</th>
<th>Training Data</th>
<th>Testing Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Word</td>
<td>Sent</td>
</tr>
<tr>
<td>SOM</td>
<td>89.8</td>
<td>83.3</td>
</tr>
<tr>
<td>P1D-SOM</td>
<td>86.1</td>
<td>77.8</td>
</tr>
</tbody>
</table>

MLSS2006-Taipei
Conclusion

- However longer training time, the P1D-SOM competition time is shorter,
- P1D-SOM, at least will show same SOM performance,
- P1D-SOM is a good alternate for traditional SOM,
- P1D-SOM is applicable until N-dim feature map!
- Previous results are presented in WCCI06-IJCNN06, July – Vancouver.
- For Face Recognition using CMU-PIE database, it will present in ICPR06, Aug - Hong Kong.
Future Work

Stochastic Analysis for Randomized SOM
Randomized SOM*

- Input Image := \( x \in R^L \)
- Codebook := \( \{ w_i \mid w_i \in R^L, i = 1,2, .. , M \} \)
- Task: \( \text{winner} = \arg\min_i (D_i) \)

\[
D_i = \frac{\| x - w_i \|^2}{L} = \frac{1}{L} \sum_{j \in S} (x_j - w_{ij})^2,
\]

where \( S \) is a set of pixels, \( L \) is size of \( S \)

* By a colleagues!
Randomized SOM

1\textsuperscript{st} Step

\[ \text{win\_cand} = \arg \min_i (D'_i) \]

or \( D'_{\text{win\_cand} = a_1} \leq D'_2 \leq \cdots \leq D'_{a_M} \)

where \( D'_i = \frac{\|x - w_i\|^2}{L'} = \frac{1}{L'} \sum_{j \in S'} (x_j - w_{ij})^2 \),

\( S' \) is a subset of \( S \), \( L' \) is size of \( S' \)

\[ \text{PUR(Pixel Useage Ratio)} = \frac{L'}{L} \]
Randomized SOM

2nd Step

Assume that SOM can learn a (local) order

$$D_{b_1} \leftrightarrow D_{b_2} \leftrightarrow \cdots \leftrightarrow D_{b_M},$$ so that

when \( \text{winner} = b_k \),

then \( D_{b(k)} \leq D_{b(k+1)} \leq D_{b(k-1)} \leq D_{b(k+2)} \cdots \)

\( = D_{c_1} \leq D_{c_2} \leq D_{c_3} \leq \cdots \leq D_{c_m} \cdots \)

let \( m_{(NR)} << M \)

\( \text{winner}^' = \arg \min_{i=c_1, c_2, \ldots, c_m} (D_i) \)

\[ P_{SOM} (\text{winner} = \text{winner}^' \mid \text{PUR}, NR) = ? \]
Stochastic Analysis for RSOM

\[ D'_i = \frac{\|x - w_i\|^2}{L'} = \frac{1}{L'} \sum_{j \in S'} (x_j - w_{ij})^2 \]

\[ P(D'_i) \] is a normal distribution \( N\left(\mu_i, \frac{\sigma_i^2}{N}\right) \)

because of the central limit theory, where \( \mu_i \) and \( \sigma_i^2 \) are the average and the deviation of \( (x_j - w_{ij})^2 \), respectively.

How can we estimate the optimum of PUR and NR?
$P_i = P(D'_i)$ is a normal distribution $N\left(\mu_i, \frac{\sigma_i^2}{N}\right)$

\[ P(D_1) = P_{\text{win}=1} \]
\[ = \int_0^\infty P_1(x) \left\{ \prod_{i \neq 1} P_i(\xi > x) \right\} dx \]
\[ = \int_0^\infty P_1(x) \left\{ \prod_{i \neq 1} \int_0^\infty P_i(\xi) d\xi \right\} dx \]

\[ P(D_i) = P_{\text{win}=j} = \int_0^\infty P_j(x) \left\{ \prod_{i \neq j} \int_0^\infty P_i(\xi > x) d\xi \right\} dx \]
Monte Carlo Markov Chain

MCMC(?)

- According to the randomized SOM algorithm,
  - if we can know probabilities for any of $X$
    
    
    
    $P(d_1) \geq P(d_2) \geq \cdots \geq P(d_M),$
    
    and
    
    $P(d_1) + P(d_2) + \cdots + P(d_M) = 1,$
    
  - Then we can estimate
    
    $P_{SOM} = P(d_1) + P(d_2) + \cdots + P(d_m)$

Winner$=d_1$

MLSS2006-Taipei
Any Comment or Suggestion is Welcome! 😞