High dimensional Data visualization and clustering using Self Organizing Maps

Oliver Bernhardt
oliver.bernhardt@fh-hagenberg.at

Department of Medical- and Bioinformatics
Upper Austria University of Applied Sciences
Softwarepark 11, 4232 Hagenberg, Austria

May 26, 2010
Overview

<table>
<thead>
<tr>
<th>Overview</th>
<th>About SOMs</th>
<th>Introduction to SOMs</th>
<th>SOMTHING App.</th>
<th>Benchmark</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>About Self Organizing Maps (SOMs)</td>
<td>Introduction to SOMs</td>
<td>SOMTHING Application</td>
<td>Benchmark Analysis</td>
<td>Conclusion</td>
</tr>
<tr>
<td></td>
<td>Topology</td>
<td>Basic Algorithm</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- About Self Organizing Maps (SOMs)
- Introduction to SOMs
  - Topology
  - Basic Algorithm
- SOMTHING Application
- Benchmark Analysis
About SOMs

What are Self Organizing Maps

- Invented by Dr. Teuvo Kohonen
- Unsupervised Learning Process
- Inspired by the Human Brian
- Grid of Neurons trained by Stimuli
- Visualises High Dimensional Data as a 2D Map.

"one of the most significant inventions in computational science"

(IEEE.org 2010)
Introduction

Data Types

- **Input Data Entry**
  - An Item of the \(d\)-Dimensional Input Data Space
  - Represented by an Input Data Vector of Size \(d\).

- **Neuron**
  - A Node in a Grid connected to a specified amount of Neighbours.
  - Containing a Weight Vector of Size \(d\).
  - Representing any point of the \(d\)-Dimensional Input Data Space.
Introduction

Data Types

- **Input Data Entry**
  - An Item of the $d$-Dimensional Input Data Space
  - Represented by an Input Data Vector of Size $d$.

- **Neuron**
  - A Node in a Grid connected to a specified amount of Neighbours.
  - Containing a Weight Vector of Size $d$.
  - Representing any point of the $d$-Dimensional Input Data Space.
Introduction

Grid Topology

- Rectangular Grid
  - Simple Implementation
  - Satisfying Data Structure Preservation

- Hexagonal Grid
  - Complex Implementation
  - Good Data Structure Preservation
Introduction

Grid Topology

- Rectangular Grid
  - Simple Implementation
  - Satisfying Data Structure Preservation

- Hexagonal Grid
  - Complex Implementation
  - Good Data Structure Preservation
Training

Initialisation

Step 1 Create a Grid of $n \cdot m$ Nodes (Neurons).
Step 2 Initialise Random Weight Vectors for each Neuron.

Training

Step 1 Select an Entry of the Input Data Space by Chance.
Step 2 Determine the Best Matching Unit (BMU).
Step 3 Adjust Weight Vectors of the BMU and its Neighbours inside a certain Radius $r$.
Step 4 Decrease the Radius $r$ and the Learning Rate $\eta$.
Step 5 Go Back to Step 1 until Training is done.
Training

Initialisation

Step 1 Create a Grid of \( n \cdot m \) Nodes (Neurons).
Step 2 Initialise Random Weight Vectors for each Neuron.

Training

Step 1 Select an Entry of the Input Data Space by Chance.
Step 2 Determine the Best Matching Unit (BMU).
Step 3 Adjust Weight Vectors of the BMU and its Neighbours inside a certain Radius \( r \).
Step 4 Decrease the Radius \( r \) and the Learning Rate \( l \).
Step 5 Go Back to Step 1 until Training is done.
Training

**Initialisation**

Step 1 Create a Grid of $n \cdot m$ Nodes (Neurons).

Step 2 Initialise Random Weight Vectors for each Neuron.

**Training**

Step 1 Select an Entry of the Input Data Space by Chance.

Step 2 Determine the Best Matching Unit (BMU).

Step 3 Adjust Weight Vectors of the BMU and its Neighbours inside a certain Radius $r$.

Step 4 Decrease the Radius $r$ and the Learning Rate $l$.

Step 5 Go Back to Step 1 until Training is done.
Training

Initialisation

Step 1  Create a Grid of $n \cdot m$ Nodes (Neurons).
Step 2  Initialise Random Weight Vectors for each Neuron.

Training

Step 1  Select an Entry of the Input Data Space by Chance.
Step 2  Determine the Best Matching Unit (BMU).
Step 3  Adjust Weight Vectors of the BMU and its Neighbours inside a certain Radius $r$.
Step 4  Decrease the Radius $r$ and the Learning Rate $l$.
Step 5  Go Back to Step 1 until Training is done.
Training

Initialisation

Step 1 Create a Grid of \( n \cdot m \) Nodes (Neurons).
Step 2 Initialise Random Weight Vectors for each Neuron.

Training

Step 1 Select an Entry of the Input Data Space by Chance.
Step 2 Determine the Best Matching Unit (BMU).
Step 3 Adjust Weight Vectors of the BMU and its Neighbours inside a certain Radius \( r \).
Step 4 Decrease the Radius \( r \) and the Learning Rate \( l \).
Step 5 Go Back to Step 1 until Training is done.
# Training

## Initialisation

1. Create a Grid of \( n \cdot m \) Nodes (Neurons).
2. Initialise Random Weight Vectors for each Neuron.

## Training

1. Select an Entry of the Input Data Space by Chance.
2. Determine the Best Matching Unit (BMU).
3. Adjust Weight Vectors of the BMU and its Neighbours inside a certain Radius \( r \).
4. Decrease the Radius \( r \) and the Learning Rate \( l \).
5. Go Back to Step 1 until Training is done.
Training

**Initialisation**

- **Step 1** Create a Grid of $n \cdot m$ Nodes (Neurons).
- **Step 2** Initialise Random Weight Vectors for each Neuron.

**Training**

- **Step 1** Select an Entry of the Input Data Space by Chance.
- **Step 2** Determine the Best Matching Unit (BMU).
- **Step 3** Adjust Weight Vectors of the BMU and its Neighbours inside a certain Radius $r$.
- **Step 4** Decrease the Radius $r$ and the Learning Rate $l$.
- **Step 5** Go Back to Step 1 until Training is done.
Training Example

(a) Input Data
(b) Map Training
(c) Final Projection
Example

Training Example

(a) Input Data

(b) Map Training

(c) Final Projection
Example

Training Example

(a) Input Data  
(b) Map Training  
(c) Final Projection
Example

Training Example

(a) Input Data

(b) Map Training

(c) Final Projection
SOMTHING Application Main Window

Self Organised Mapping Tool using Hexagonal Interlaced Neuron Grids
SOMTHING

Features

- SOM Visualisation Methods
  - U-Matrix
  - P-Matrix
  - U*-Matrix
  - Component Planes
  - Hit Histogram

- Clustering
  - Hierarchical Clustering
  - SOM Clustering
Features

- SOM Visualisation Methods
  - U-Matrix
  - P-Matrix
  - U*-Matrix
  - Component Planes
  - Hit Histogram

- Clustering
  - Hierarchical Clustering
  - SOM Clustering
U-Matrix

U-Height = Total Euclidean Distances between a Neuron’s Weight Vector to its Neighbours.

- **Local Distances**
  - Bright Colors
    - Low U-Height
    - Similar to Neighbours
    - Cluster Centres
  - Dark Colors
    - High U-Height
    - Different from Neighbours
    - Cluster Borders
U-Matrix

U-Height = Total Euclidean Distances between a Neuron’s Weight Vector to its Neighbours.

- Local Distances
- Bright Colors
  - Low U-Height
  - Similar to Neighbours
  - Cluster Centres
- Dark Colors
  - High U-Height
  - Different from Neighbours
  - Cluster Borders
U-Matrix

U-Height = Total Euclidean Distances between a Neuron’s Weight Vector to its Neighbours.

- Local Distances
- Bright Colors
  - Low U-Height
  - Similar to Neighbours
  - Cluster Centres
- Dark Colors
  - High U-Height
  - Different from Neighbours
  - Cluster Borders
P-Matrix

P-Height = Pareto Density Estimation at the Neuron’s representative point in the Input Data Space.

- Data Density Estimation
  - Bright Colors
    - Low P-Height
    - Low Density
    - Outliers
  - Dark Colors
    - High P-Height
    - High Density
    - Cluster Centres
P-Matrix

P-Height = Pareto Density Estimation at the Neuron’s representative point in the Input Data Space.

- Data Density Estimation
- Bright Colors
  - Low P-Height
  - Low Density
  - Outliers
- Dark Colors
  - High P-Height
  - High Density
  - Cluster Centres
**P-Matrix**

P-Height = Pareto Density Estimation at the Neuron’s representative point in the Input Data Space.

- **Data Density Estimation**
- **Bright Colors**
  - Low P-Height
  - Low Density
  - Outliers
- **Dark Colors**
  - High P-Height
  - High Density
  - Cluster Centres
Benchmark

Chainlink Dataset

- Common Clustering Benchmark
- 2 intertwined 3D Rings
- 500 Data Points per Ring
- unsolvable with K-Means or Hierarchical Clustering

image taken from www.ifs.tuwien.ac.at
Benchmark

Learning Process

(a) Iteration 0  (b) Iteration 100  (c) Iteration 1000
(d) Iteration 3000  (e) Iteration 4000  (f) Iteration 6000
Benchmark

Results

(a) X-Z Axis  
(b) Y-Z Axis  
(c) Y-X Axis

Automatic Clustering Result
Not one Sample misclassified!
Benchmark

Results

(a) X-Z Axis
(b) Y-Z Axis
(c) Y-X Axis

Automatic Clustering Result
Not one Sample misclassified!
Conclusion

Self Organising Maps are ...

- an unsupervised learning method
- a powerful approach to visualise very high dimensional data.
- an interesting alternative to usual Clustering Methods.
Questions?