PySOM: A Graph Based Implementation of Self-Organizing Maps

Charles R. Schmidt¹
Sergio J. Rey¹
André Skupin²

The GeoDa Center for Geospatial Analysis and Computation
Arizona State University¹
Department of Geography
San Diego State University²

November 21, 2008
Outline

1. Background
   - The Self-Organizing Map
   - Edge Effects in SOM

2. PySOM
   - Objectives
   - Design

3. Results
   - Status Report
   - Visualization

4. Conclusions
   - Limitations & Future Directions
The Self-Organizing Map

What is the SOM
- Artificial Neural Network
- High Dimension Input Data
- Unsupervised Competitive Learning Process
- Each Neuron Models of a Portion of the Input Space

Applications
- Data Visualization
- Data Reduction
- Clustering
SOM as Dimensionality Reduction

(Skupin 2008)
SOM as Clustering

(Skupin 2008)
SOM as Clustering

(Skupin 2008)
"translate data similarities into spatial relationship"

– Helge Ritter, 1999
Assignment and Updating

before training → first vector used → second vector used → third vector used → fourth vector used

(a) → (b) → (c) → (d) → (e)
Training

- Randomize input vectors
- Randomly Initialize the neurons
- Loop Until Map Converges
  - Grab an Input Vector
  - Find the Best Matching Neuron and its Neighborhood
  - Modify the Weights of the Neurons to Make them More Similar to the Input Vector
Training

Iterations

random initialization → 40000 training runs → 80000 training runs → 100000 training runs
Edge Effects in SOM

This diagram illustrates the edge effects in a Self-Organizing Map (SOM) applied to a map of the United States. The hexagonal grid represents the SOM, with each hexagon indicating the effect at that location. The shading varies from light to dark, indicating the intensity of the effect. States are labeled for reference.
Edge Effects

In SOM
- Higher density of observations mapped to edge neurons
- Edge neurons have less influence

In Spatial Analysis
- Inflated nearest neighbor distances
- Mask the true distribution

Components of Edge Effect
- True Boundary
Suggested Solutions

With Edges
- Hierarchical SOM
- Growing SOM
- Mathematical Weighting

Without Edges
- Spherical SOM
- Toroidal SOM
# Spherical SOM

<table>
<thead>
<tr>
<th>Geodesic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Most Common</td>
</tr>
<tr>
<td>Highly Regular</td>
</tr>
<tr>
<td>Limited Network Size</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rakhmanov (Spherical)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rejected in Literature</td>
</tr>
<tr>
<td>Less Regular</td>
</tr>
<tr>
<td>No Network Size Limitation</td>
</tr>
</tbody>
</table>
PySOM

Why build another SOM implementation

- Alternative topologies not available
- Difficult to extend

Objectives

- Open Source
- Support many topologies
- Testbed for different topologies

Design Components

- Python
- PySAL
- Graph Based Implementation
## Design

### Why Python
- Growing scientific community
- Extensive libraries
- Rapid development

### Python Spatial Analysis Library
- Collection of Spatial Analysis Tools
- Leveraging PySAL
- Extending PySAL
Graph Based Implementation

### Hard Coded Topology
- Fast
  - Optimized for each topology
- Difficult to extend
- Inflexible

### Graphs
- Topology independent
  - Can train using any graph
- Neighborhood construction
- Flexible
  - Can modify graph structure
Current Status

Open Source
- Public Repository

Core Library
- Limited API
- Training
- Mapping

Topologies
- Built-in
  - Rectangular, Hexagonal
- External
  - Rakhmanov (Spherical), Geodesic Sphere
Visualization

Visualizing the SOM

- Currently Handled by ESRI’s ArcMap
- Limited Assistance
  - txt-to-shp
- Component Mapping
  - DBF rewrite
Visualization

Geodesic
Visualization

Spherical
Limitations & Future Directions

Limitations

• Flexibility over speed
  • SOM_PAK: 90 seconds
  • PySOM: 45 minutes

• API/Interface

• Topology Support

• Visualization
Limitations & Future Directions

Extensions

- PySAL Style Distribution
  - Standalone Desktop Application
  - ArcMap Toolbox (Som Analyst)
  - Web Services
Back to Utah and Hawaii