Topology Preservation - part I
Self-organising maps

Dušan Sovilj

23.10.2007
Outline

• Distance vs Topology
• Topology-preserving methods
• Self-organising map in dimensionality reduction
  – embedding of data set
  – embedding of test set
  – example
  – summary
Distance vs Topology

- **Distance**
  - simple to understand
  - easy to compute
  - rigid in dimensionality reduction

- **Topology**
  - can fully describe manifolds
  - hard to represent with few points
  - without a good description of topology, dimensionality reduction is impossible
Topology-preserving methods

- Very ofted they use the discrete mapping model
  - the topology is then also discrete, called lattice
- Topology can be discretely represented with a graph
- Classified according to type of topology used in embedding space
  - predefined lattice
  - data-driven lattice
'Predefined lattice' methods

- Methods
  - self-organising map
  - generative topographic mapping
- Application very limited (few manifolds fit predefined shape)
- Good way for visualization of labeled data
SOM in DR

- Can be interpreted as non-linear & discrete PCA
  - fitting of hyperplane inside the data cloud
  - hyperplane is described with discrete points
  - fitting of a 'fishing net' around an object
- SOM uses a mix of vector quantization and topology preservation
Classic VQ & SOM's VQ

- Vector quantization in SOMs are mandatory
- Classic VQ doesn't take into account the relations between the prototypes/representatives
- SOM tries to 'preserve' the lattice in data space by moving prototypes in neighbouring groups
SOM definition

- Set $\mathcal{C}$ containing the prototypes in the data space for vector quantization. Prototypes are $D$-dimensional points $\mathbf{c}(r)$
- Distance function $d_g(r, s)$ between the prototypes in the lattice
- $r$ & $s$ are indices of prototypes
Embedding of data points in SOM

- Lattice plays the role of embedding space
  - $d_g(r, s)$ cannot be any distance function
  - $d_g(r, s)$ is often defined as $d_g(g(r), g(s))$, $g(r), g(s) \in G \subset \mathbb{R}^P$
    $P$ is dimensionality of embedding space
- Prototypes have coordinates both in data and embedding space
  - coordinates in embedding space are known (predefined lattice)
  - coordinates in data space are unknown, and SOM has to find them
- Embedding of data points $y(i)$
  $x(i) = g(r)$
  $r = \arg \min_s d(y(i), c(s))$
  where $d$ is Euclidean distance
Prototypes' coordinates

- Iteratively by going through all data points in $\mathbf{Y}$ (epochs)
- For every point $y(i)$
  1. Find closest prototype
     
     $$ r = \underset{s}{\arg \min} d(y(i), c(s)) $$
     
     2. Update coordinates of all prototypes
     
     $$ c(s) \leftarrow c(s) + \alpha \nu_{\lambda}(r, s) (y(i) - c(s)) $$

     $\alpha$ learning rate, $0 \leq \alpha \leq 1$ decreases with epochs
     $\nu_{\lambda}$ neighbourhood function
Neighbourhood functions

- 'Bubble'

\[ \nu_\lambda(r, s) = \begin{cases} 
0 & \text{if } d_g(r, s) > \lambda \\
1 & \text{if } d_g(r, s) \leq \lambda 
\end{cases} \]

\[ d_g(r, s) = L_1, L_2, L_\infty \]

- Gaussian like

\[ \nu_\lambda(r, s) = \exp\left(-\frac{d_g^2(r, s)}{2\lambda^2}\right) \]

\[ d_g(r, s) = L_2 \]
Shape of lattice

- In most implementations the points in the lattice are equally spaced in the plane which forces the embedding space to be two-dimensional
  - square structure (8 neighbours)
  - hexagonal struture (6 neighbours)
- For higher dimension lattices the hyper-cubic neighbourhhoods are mostly used
SOM batch algorithm

1. Define the lattice by assigning the low-dimensional coordinates $g(r)$ of the prototypes in the embedding space

2. Initialize the coordinates $c(r)$ of prototypes in the data space

3. Give $\alpha$ and $\nu$ their scheduled values for epoch $q$

4. For all points $y(i)$ in the data set, find it's closest prototype and update coordinates of all prototypes

5. Return the step 3 until convergence is reached (the updates of prototypes become negligible)
Embedding of test points

- Calculation is done exactly as for data points
  \[ c(s) \leftarrow c(s) + \alpha \nu_\lambda(r, s) (y(i) - c(s)) \]
- This gives rough estimate of exact coordinates of test points in embedding space, since there is fixed number of outputs
- There exist precise interpolation methods for finding these coordinates
Example
Example (cont.)
SOM summary (1)

- Vector quantization is mandatory
- Models data in nonlinear and discrete way
- Mappings is explicit (prototypes only)
SOM summary (2)

- **Advantages**
  - Easy to understand
  - Nice visualization method
  - Applicable to various fields

- **Drawbacks**
  - Low dimensionality of embedding space (usually 1 or 2 dimensions)
  - Vector quantization
  - Shape of lattice predefined (often arbitrary)
  - No objective function or error criterion
  - What are good values for parameters $\alpha$ and $\nu$
SOM variants

- Extend the lattice by adding rows and columns
  - GG (Growing Grid) & GSOM (Growing SOM)
- Change the shape appropriately to fit data
  - GCS (Growing Cell Structure)