The Self-Organizing Map

Jenny Hamer

University of California, San Diego

Paper by Teuvo Kohonen, IEEE 1990

November 8, 2018
Overview

1. Competitive learning

2. Learning vector quantization (LVQ) methods
   - Unsupervised vector quantization
   - Supervised LVQ

3. Topological neighborhood

4. Self-organizing map algorithm

5. Applications and results
Let \( \mathbf{x} \in \mathbb{R}^n \) be an input vector. Let \( d(\mathbf{a}, \mathbf{b}) \) be a similarity metric between two vectorial inputs, for example the Euclidean distance between \( \mathbf{a}, \mathbf{b} \).

1. Initialize a set of “reference” vectors \( M = \{ \mathbf{m}_i \mid \mathbf{m}_i \in \mathbb{R}^n, i = 1, \ldots, k \} \) (e.g. randomly)

2. Simultaneously compare \( \mathbf{x} \) with each \( \mathbf{m}_i \); compute the similarity between \( \mathbf{x}, \mathbf{m}_i \): \( d(\mathbf{x}, \mathbf{m}_i) \)

3. The best-matching \( \mathbf{m}_i \) is declared the “winner”; let \( \mathbf{m}_c \) be this reference vector

4. Tune \( \mathbf{m}_c \) such that \( d(\mathbf{x}, \mathbf{m}_c) \) decreases while all other \( \mathbf{m}_i \) are kept unchanged.

Supposing that the \( \mathbf{x} \) follow some distribution with probability density function \( p \), then the tuned \( \mathbf{m}_i \) form an approximation of \( p \).
The unit \( u_j \) in the layer \( F_2 \) is the "winner" for input pattern \( I \): input \( I \)'s cluster is determined by this unit.
Unsupervised Vector Quantization: k-means

Goal: learn a finite, discrete approximation $M$ of continuous probability density function of vectorial data $X = \{x \mid x \in \mathbb{R}^n\}$.

As in competitive learning, randomly initialize the codebook $M$.

For each input vector $x$:

1. Assign $x$ to the cluster of its “winner” $m_c$ satisfying $\min_i \{||x - m_i||\}$ (in general, w.r.t. $L^p$ norm: we’ll use $p = 2$ or Euclidean distance)

2. Tune the codebook vectors:
   
   $m_c(t + 1) = m_c(t) + \alpha(t)[x(t) - m_c(t)]$ where $m_c$ was a “winner”
   
   $m_i(t + 1) = m_i(t)$ for all $i \neq c$

Originally used in signal processing and approximation:

- often more economical to use a batched method: match and assign numerous $x(t)$, then update in a single step.
Supervised LVQ

Suppose some set of the input data $\mathbf{x}$ have known class labels: assign several $\mathbf{m}_i \in M$ as representatives for each class (e.g. randomly). The $\mathbf{m}_i$ closest to $\mathbf{x}$ determines $\mathbf{x}$’s class.

1. Assign class of $\mathbf{m}_c$ to $\mathbf{x}$
2. Update the $\mathbf{m}_i$ as follows:

If $\mathbf{x}$ is correctly classified:

$$\mathbf{m}_c(t + 1) = \mathbf{m}_c(t) + \alpha(t)[\mathbf{x} - \mathbf{m}_c(t)]$$

If $\mathbf{x}$ is incorrectly classified:

$$\mathbf{m}_c(t + 1) = \mathbf{m}_c(t) - \alpha(t)[\mathbf{x} - \mathbf{m}_c(t)]$$

leaving all weights $i \neq c$ unchanged: $\mathbf{m}_i(t + 1) = \mathbf{m}_i(t)$

Here $\alpha(t)$, $0 < \alpha(t) < 1$, is a scalar gain which should decrease monotonically over time.
Goal: partition the cells of the map such that each responds to a discrete set of features of the data. Like competitive learning, but with *neighborhood-dependent* updates of cells in $N_c$ to learn a spatial ordering.

![Diagram of topological neighborhood](image)

**Fig. 2.** Examples of topological neighborhood $N_c(t)$, where $t_1 < t_2 < t_3$.

This algorithm allows for high-dimensional data to be visualized in lower dimensional (often 2D) space.
SOM architecture

- Network: (most commonly) $n \times m$ cells positioned on a grid or lattice, where each cell $\boldsymbol{m}_i \in \mathbb{R}^n$ (the red nodes)

- $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$ are the input vectors or patterns $\in \mathbb{R}^n$ (the yellow nodes)

- Each input $\boldsymbol{x}_i$ is “connected” to every $\boldsymbol{m}_i$ in the network (so that each $\boldsymbol{x}_i$ can be compared with all $\boldsymbol{m}_i$)

Figure: An example of a $3 \times 3$ network

(Snapshot from https://youtu.be/-Euwc9fWBJw)
SOM Learning algorithm

Forming the topological order of the map: begin with a large $N_c$, roughly $1/2$ number of neurons in network.

Iteratively:

1. Find cell $c$ s.t. $||x - m_c|| = \min_i \{||x - m_i||\}$
2. Update the $m_i$:

If cell $i \in N_c(t)$

$$m_i(t+1) = m_i(t) + \alpha(t)[x(t) - m_i(t)]$$

The adaption gain scalar $\alpha$, $0 < \alpha < 1$ is often replaced with a scalar “kernel function” $h_{ci}(t)$, most commonly a Gaussian neighborhood:

$$h_{ci}(t) = h_0(t) \exp(-||r_i - r_c||^2/\sigma(t)^2)$$

where $h_0(t), \sigma(t)$ decrease over time, and $r_i, r_c$ denote cells $c$ and $i$. 
Simple demonstration with colors

Using 15 samples of RGB values, this $20 \times 30$ cell network learns a classification of the colors.
Simple demonstration with colors

Using 15 samples of RGB values, this $20 \times 30$ cell network learns a classification of the colors.
Simple demonstration with colors

Using 15 samples of RGB values, this $20 \times 30$ cell network learns a classification of the colors.
Simple demonstration with colors

Using 15 samples of RGB values, this $20 \times 30$ cell network learns a classification of the colors.
Simple demonstration with colors

Using 15 samples of RGB values, this $20 \times 30$ cell network learns a classification of the colors.
Effect of decrease in network size

Alternative clustering of the same 15 RGB vectors with a $10 \times 15$ network:
Effect of decrease in network size

Alternative clustering of the same 15 RGB vectors with a $10 \times 15$ network:
Effect of decrease in network size

Alternative clustering of the same 15 RGB vectors with a $10 \times 15$ network:
Effect of decrease in network size

Alternative clustering of the same 15 RGB vectors with a $10 \times 15$ network:
Effect of decrease in network size

Alternative clustering of the same 15 RGB vectors with a $10 \times 15$ network:
Effect of decrease in network size

Now, with a $50 \times 60$ network:
Effect of decrease in network size

Now, with a $50 \times 60$ network:
Effect of decrease in network size

Now, with a $50 \times 60$ network:
Effect of decrease in network size

Now, with a $50 \times 60$ network:
Effect of decrease in network size

Now, with a $50 \times 60$ network:
Practical tips in applications

- **Initialization of** $m_i(0)$:
  - May be arbitrarily or randomly chosen
  - Only restriction is that they are distinct

- **Number of iterations**:
  - “Rule of thumb”: $500 \times$ number of network cells

- **Learning rate**: $\alpha$
  - Use a high learning rate (close to 1) for the first 1000 iterations, then monotonically decrease $\alpha$ during fine-tuning

- **Neighborhood size**: $N_c$
  - If $N_c$ too small $\rightarrow$ map will not learn global ordering
  - If $N_c$ too large $\rightarrow$ map will be very coarse-grained in clustering
  - Choose are generous $N_c$ at start of training:
    - $N_c(0)$ can be > than half the size of the network;
    - Reduce the size during training

Note that the updates to each cell’s weights tend to smooth out over learning and **weight vectors tend to become ordered in value along the axes of the network.**
Theory behind the SOM

- Extremely difficult to capture the dynamics of the SOM in mathematical theorems and analyses

- Many strict analyses have been attempted under simplifying constraints, though these are beyond the scope of this paper
  - Cottrell and Fort (1987) have proven that the SOM will converge to a globally ordered state in some constrained low-dimensional cases
  - Ritter, Martinez, and Schulten (1992) have shown that under the assumption the $\mathbf{x}$ are discrete-valued, local order can be achieve for more general dimensionalities of the vectors

- Several meaningful applications have been simulated using the SOM to demonstrate its utility and efficacy
Building a “phonetic typewriter” to identify and recognize phonemes: represent continuous Finnish speech as spectral vectors, $\mathbf{x}$

- $\mathbf{x} \in \mathbb{R}^{15}$ spectral vectors computed every 9.83ms using 256-pt FFT

- No segmentation of speech and fully unsupervised: naturally occurring features contributed to self-organization

Figure: 8 × 12 Finnish phoneme map

The 21 phonemes of Finnish have been learned: most cells respond to a unique phoneme, but some cells do to two.
Application: Semantic mapping

**Goal:** extract abstract logically similar, semantic information from symbolic data.

- Meaning of a symbolic encoding requires consideration of the conditional probabilities of its occurrences (contexts) with other encodings.

- Represent symbolic encoding as a vector $x_s$ of symbols and their context $x_c$:

$$x = \begin{bmatrix} x_s \\ x_c \end{bmatrix} = \begin{bmatrix} x_s \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ x_c \end{bmatrix}$$

**Figure:** Vocabulary used in experiment
Semantic meaning should be inferred only from context in which words occur: Each word encoded as a random, 7-dim. unit vector.
Application: semantic maps

Presented 2,000 word-context-pairs derived from 10,000 random sentences

**Figure:** $15 \times 10$-cell learned semantic map

![Semantic Map Diagram]
Concluding remarks

- The SOM is a powerful neural-model capable of creating localized, structured, clustered representations of input data.

- On its own, the SOM is not well suited for classification tasks unless fine-tuning methods are used to increase decision accuracy.

- Initialization of the map affects the locality of its learned responses and ordering.

- Particularly useful for high-dimensional data: reduces dimensionality of the data and the topological mapping allows underlying properties of data to be visualized.
References

Tuevo Kohonen
The Self-Organizing Map

Tuevo Kohonen
Essentials of the self-organizing map
Any questions?

1. Competitive learning

2. Learning vector quantization (LVQ) methods
   - Unsupervised vector quantization
   - Supervised LVQ

3. Topological neighborhood

4. Self-organizing map algorithm

5. Applications and results