The Self-Organizing Map

Teuvo Kohonen
(1990)

Presented By Fatma Faruq
For the Deep Learning Class
Modeling The Brain

- Feedback Networks
- Feedforward Networks
- Competitive, Unsupervised or Self Organizing Learning
  - Self-Organizing Maps
Competitive Learning

- $x$ – set of input vectors
- $m$ – set of variable reference vectors
- What is the Best Matching Reference Vector
Competitive Learning

- $x$ – set of input vectors
- $m$ – set of variable reference vectors
- What is the Best Matching Reference Vector
Competitive Learning

- $x$ – set of input vectors
- $m$ – set of variable reference vectors
- What is the Best Matching Reference Vector
Competitive Learning

- The idea is to:
  - Find a **winner** \( m_c \)
  - Update

\[
d(x, m_c) = \min_i d(x, m_i)
\]

\[
m_c(t+1) = m_c(t) + a(t) [x(t) - m_c(t)]
\]

\[
m_i(t+1) = m_i(t) \quad i \neq c
\]

- 'a' is a variable that decreases monotonically with time
- Note: m could be thought of as weight vectors
Self-Organizing Map

- Each input vector is connected to all cells
- The cells are not connected to each other
- Perform Competitive Learning
  - Find the best Neighborhood (N_c) of cells that matches the input vector

Since N should decrease over time 'a' could be replaced by a kernel function h_{ci}

\[
m_c(t+1) = m_c(t) + h_{ci}(t)[x(t) - m_c(t)]
\]
Hints for Application

- **Weight vectors**
  - Arbitrarily initialized
  - Spread over in time
  - Number of Iterations is important

- Number of iterations = 500 x number of network units
- Number of components in input has no effect
- Recycle samples if needed
- \( a(t) \) should start close to unity and decrease monotonically
  - \( a(t) = 0.9(1-t/1000) \)
  - Weights are ordered in the beginning then fine tuned
- Start with a large enough neighborhood
- Matching and updating laws should be mutually compatible
Fine Tuning the Map

• Type One Learning Vector Quantization:
  - Start with values of $m$ that correspond to the original density function of the input
  - Determine the number of clusters (codebook vectors) by assigning cells to different classes using majority voting (known classification)

\[
\begin{align*}
  m_c(t + 1) &= m_c(t) + \alpha(t)[x(t) - m_c(t)] \\
  &\quad \text{if } x \text{ is classified correctly,} \\
  m_c(t + 1) &= m_c(t) - \alpha(t)[x(t) - m_c(t)] \\
  &\quad \text{if the classification of } x \text{ is incorrect,} \\
  m_i(t + 1) &= m_i(t) \text{ for } i \neq c.
\end{align*}
\]
Fine Tuning the Map

• Type One Learning Vector Quantization (LVQ1):
  - Start with values of $m$ that correspond to the original density function of the input
  - Determine the number of clusters (codebook vectors) by assigning cells to different classes using majority voting (known classification)

\[
\begin{align*}
m_c(t+1) &= m_c(t) + \alpha(t)[x(t) - m_c(t)] \\
& \quad \text{if } x \text{ is classified correctly}, \\
m_c(t+1) &= m_c(t) - \alpha(t)[x(t) - m_c(t)] \\
& \quad \text{if the classification of } x \text{ is incorrect}, \\
m_i(t+1) &= m_i(t) \text{ for } i \neq c. \quad (11)
\end{align*}
\]
Fine Tuning the Map contd

- Type Two Learning Vector Quantization (LVQ2):
  - Define a window of nonzero width when two vectors intersect
  - Correct only if input falls within this window
  - Optimal size of window depends on the number of training samples
  - Improves accuracy only in the beginning so apply for short period of time

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{window.png}
\caption{Illustration of the “window” used in the LVQ2 and LVQ3 algorithms. The curves represent class distributions of \( x \) samples.}
\end{figure}

\begin{align*}
  m_i(t + 1) &= m_i(t) - \alpha(t)[x(t) - m_i(t)], \\
  m_j(t + 1) &= m_j(t) + \alpha(t)[x(t) - m_j(t)],
\end{align*}

if \( C_i \) is the nearest class, but \( x \) belongs to \( C_j \neq C_i \) where \( C_j \) is the next-to-nearest class (“runner-up”); furthermore \( x \) must fall into the “window”. In all the other cases,

\begin{equation}
  m_k(t + 1) = m_k(t).
\end{equation}
Fine Tuning the Map contd

• Problems with LVQ2:
  - The correction on the correct class is larger than that on the wrong class
  - If the process is continued may lead to non-optimal distribution

• LVQ3:
  - Corrections to ensure that class distributions are approximated (shown right)
  - The optimal value of epsilon depends on the size of the window (smaller for narrower windows)

\[
\begin{align*}
  m_i(t + 1) &= m_i(t) - \alpha(t)[x(t) - m_i(t)], \\
  m_j(t + 1) &= m_j(t) + \alpha(t)[x(t) - m_j(t)],
\end{align*}
\]

where \( m_i \) and \( m_j \) are the two closest codebook vectors to \( x \), and \( x \) and \( m_i \) belong to the same class, while \( x \) and \( m_j \) belong to different classes; furthermore \( x \) must fall into the “window”;

\[
\begin{align*}
  m_k(t + 1) &= m_k(t) + \epsilon\alpha(t)[x(t) - m_k(t)],
\end{align*}
\]

for \( k \in \{i, j\} \), if \( x \), \( m_i \), and \( m_j \) belong to the same class. (14)
Applications – An Example

- Applied SOM to
  - Speech Recognition: creating map of Phonemes

![Phoneme Map](image)

**Fig. 10.** An example of a *phoneme map*. Natural Finnish speech was processed by a model of the inner ear which performs its frequency analysis. The resulting signals were then connected to an artificial neural network, the cells of which are shown in this picture as circles. The cells were tuned automatically, without any supervision or extra information given, to the acoustic units of speech known as *phonemes*. The cells are labeled by the symbols of those phonemes to which they “learned” to give responses; most cells give a unique answer, whereas the double labels show which cells respond to two phonemes.
Applications contd

- Comparison of algorithms

### Table 2: Speech Recognition Experiments with Error Percentages for Independent Test Data

<table>
<thead>
<tr>
<th></th>
<th>Parametric Bayes</th>
<th>kNN</th>
<th>LVQ1</th>
<th>LVQ2</th>
<th>LVQ3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 1</td>
<td>12.1</td>
<td>12.0</td>
<td>10.2</td>
<td>9.8</td>
<td>9.6</td>
</tr>
<tr>
<td>Test 2</td>
<td>13.8</td>
<td>12.1</td>
<td>13.2</td>
<td>12.0</td>
<td>11.5</td>
</tr>
</tbody>
</table>
Thank You