Distance Based Neural Networks – 2

Contents of this lecture

• Kohonen net for data compression
• SOFM: Self-organizing feature map
• LVQ: Learning vector quantization
• R4-rule: Structural learning of NN-MLP

Data compression with Kohonen net: reducing the number of data

• The Kohonen neural network can be used for finding the representatives of clusters.
• This is useful for selecting important data from the training set.
• This is a kind of data compression – Reducing the number of data.
• Another well known algorithm is k-means, which is the off-line version of WTA learning.

Data compression with Kohonen net: reducing the data length

• This is well-known as vector quantization (VQ).
• Suppose that there are \( N \) n-dimensional patterns.
• The problem is to find \( M \) codes minimizing the approximation error

\[
MSE = \sum_{i=1}^{N} \sum_{k=1}^{n} (x_{ik}^i - c_{ik}^{opt})^2
\]

where \( c_{ik}^{opt} \) is the \( k \)-th element of the closest code (or winner for the current input pattern).
Data compression with Kohonen net: reducing the data length (cont.)

• If the error is sufficiently small, the $N$ patterns can be represented by the $M$ code words.
• Each pattern can be represented by an index with $\log_2 M$ bits.
• Many algorithms have been proposed to solve this problem.
  – LBG, ISODATA, K-means, etc.
  – Winner-take-all is a on-line learning algorithm.

SOFM: Another way for reducing the data length

• The Kohonen network can also be extended to reducing the dimensionality of the pattern space.
• It is useful for visualizing the pattern space.
• A Kohonen network used for this purpose is called SOFM (self-organizing feature map).
• SOFM can reduce the dimensionality and may also preserve the topologic (neighborhood relation) structure of the problem space.

Structure of SOFM

• The neurons are usually arranged in a 2-dimensional planar array with hexagonal neighborhoods (see the figure in the next slide).
• During learning, the weights of all neurons in the neighborhood of the winner are updated.
• The amount of modification is inversely proportional to the distance between the neuron to be updated and the winner.
• The size of the neighborhood is reduced during learning.
The learning rule

• For any training example \( x \), find the winner.
• Suppose that the winner is the \( m \)-th neuron, the weight vectors of all neurons close to the winner are updated as follows

\[
W_i = W_i + \alpha(x - W_i), \text{ for } i \in N_m
\]

- \( N_m \) is the neighborhood of the \( m \)-th neuron.
- The size of \( N_m \) is decreased as the training progresses.

The learning rate

• In general, the learning rate \( \alpha \) is not a constant, it depends both on the training time and the distance between the neuron to be updated and the winner. For example

\[
\alpha = \alpha(t)e^{-r(t)/\sigma(t)}
\]

where \( \alpha(t) \) and \( \sigma(t) \) are decreasing functions of the learning time \( t \), and \( r \) is the distance between the current neuron and the winner.

Calibration: assign labels to the neurons

• After learning, each labeled pattern is provided to the neuron array.
• The winner is found.
• The winner is assigned with the same class label as the current input pattern

- Example: when B is given as the input, the upper left neuron is the winner, and thus this neuron is labeled “B”.

Example 1
(from “The self-organizing map” written by T. Kohonen, 1990)

• There are 32 different 5-dimensional patterns labeled from A through 6
• We want to reduce the dimension from 5 to 2, and see the relation between patterns

|      | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| \( x_1 \) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| \( x_2 \) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| \( x_3 \) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| \( x_4 \) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| \( x_5 \) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
Structure of the neural network

- The rectangular array consists of 70 neurons.
- Each neuron has 5 inputs.
- The neurons are trained using the 32 patterns selected each time at random.

The learning parameters

- The learning rate
  - During the first 1,000 learning cycle, the learning rate decreases linearly with time from 0.5 to 0.04.
  - During the subsequent 10,000 learning cycles, the learning rate decreases from 0.04 to 0 linearly with time.

- The neighborhood size
  - During the first 1,000 learning cycles, the neighborhood size decreased from 6 to one linearly with time.
  - The neighborhood size is kept the same during the subsequent learning cycles.

Post learning calibration

- After 10,000 training cycles, the neurons are calibrated by providing each data point once.
- The neighborhood relation between the patterns is visible in the 2-D space.

Learning vector quantization (LVQ)

- Kohonen network was originally proposed for self-organization learning.
- If the class labels (or teacher signals) of the training patterns are known, we can also use the same network for supervised learning.
- The basic idea is to find some representatives for each class, rather than for each cluster.
The LVQ learning rule

- In LVQ, each neuron is assigned a class label.
- For any input pattern $x$, find the winner.
- If the class label of the winner is the same as that of the input pattern, update the weight of the winner as follows:
  $$w_i = w_i + \alpha (x - w_i)$$

The LVQ learning rule

- If the class label of the winner is different from that of the input pattern, update the weight of the winner as follows:
  $$w_i = w_i - \alpha (x - w_i)$$

Varieties

- Update the weights only when the winner gives the correct answer.
- Update the weights only when the winner gives the wrong answer.
- Update the weights for both cases.

Problems in using LVQ

- LVQ itself does not tell us how many neurons should be used for a certain problem.
- If the number of neurons is too large, the compression ratio will be small.
- If the number of neurons is too small, the learning process may not converge.
  
  → A systematic way for determining the network structure is required
The R^4-rule


The first R: Recognition

• The first task of recognition is to find the recognition rate of the whole system.
• At the same time, the fitness of each neuron is also evaluated.
• The fitness of a neuron is high if it is a winning neuron for many training examples.

Definition of the winner

• For a given input pattern \( x \), a neuron is the winner if
  – It belongs to the same class as \( x \).
  – It fires before any neuron of different classes.
  – Its fitness is larger than that of all other neurons satisfying the first two conditions.

The second R: Remembrance

• The task is to remember an unknown example by inserting a new neuron to the current network.
• The weight vector of the new neuron equals to the unknown example.
• The example to be remembered is selected at random from all unknown examples.
The third R: Review

- The task of review is to re-adjust the weights of the current network using LVQ.
- In our research, we have adopted the DSM learning algorithm.
- That is, for any training example $x$, we update the weights of the related neurons **only if** the most active neuron has a different class label.

The fourth R: Reduction

- If the current network is good enough, we can select a neuron with a low fitness, and delete it from the network.
- In the reference, the neuron to be deleted is selected at random from all neurons whose fitness are smaller than a given threshold.
- We may also adopt some other selection methods proposed for genetic algorithms.

The whole learning process

- Step 1: Find the recognition rate and the fitness of each neuron using recognition.
- Step 2: If the recognition rate is smaller than the pre-specified rate, insert a new neuron using Remembrance; otherwise, delete a neuron using Reduction.
- Step 3: Re-adjust the network using Review.
- Step 4: Return to Step 1.

Examples

- The generalized XOR problem
- The straight line class boundaries problem
Examples (cont.)

<table>
<thead>
<tr>
<th>Methods</th>
<th>Size</th>
<th>Error rate</th>
<th>Size</th>
<th>Error rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>NNC</td>
<td>6400</td>
<td>0.91</td>
<td>6400</td>
<td>0.97</td>
</tr>
<tr>
<td>CNN</td>
<td>231</td>
<td>1.11</td>
<td>307</td>
<td>1.17</td>
</tr>
<tr>
<td>RNN</td>
<td>162</td>
<td>1.27</td>
<td>216</td>
<td>1.14</td>
</tr>
<tr>
<td>RCE</td>
<td>183</td>
<td>0.69</td>
<td>243</td>
<td>0.56</td>
</tr>
<tr>
<td>R^4-rule</td>
<td>4</td>
<td>0.13</td>
<td>10</td>
<td>0.48</td>
</tr>
</tbody>
</table>

Team Project V

- A program is given in the web for SOFM.
- Download this program, and check if it works for the example given in this lecture.
- Modify the program, and apply your program to the IRIS dataset.
- Try to find a set of learning parameters that can result in good results (a 2-Dimensional map of the feature space).