Weight tuning and pattern classification by self organizing map using genetic algorithm

Masahiro Tanaka
Okayama University

Yasuyuki Furukawa
Okayama University

Tetsuzo Tanino
Okayama University
Weight Tuning and Pattern Classification by Self Organizing Map Using Genetic Algorithm

Masahiro Tanaka
Dept. of Information Technology
Okayama University
Okayama 700, Japan
tanaka@mathpro.it.okayama-u.ac.jp

Yasuyuki Furukawa
Dept. of Information Technology
Okayama University
Okayama 700, Japan
tanino@mathpro.it.okayama-u.ac.jp

Tetsuzo Tanino
Dept. of Information Technology
Okayama University
Okayama 700, Japan

Abstract—This paper deals with the supervised learning. In many problems, the training data contains only the final judgment information in conjunction with the input data, but in some problems, more information needs to be extracted from the training data. A typical example is a medical diagnosis. The objective of this paper is to give the user internal information contained in the data by using only the binary class-information data. Self organizing map (SOM) is used as the main tool for this purpose. Our method is to tune the weight of the elements of the data so that the data of the same category tend to be mapped in the near points on SOM, and separation of different category can be well done. Genetic algorithm (GA) is used for the tuning of the weight coefficients. After the learning, we can obtain the feature map, as well as the weight coefficients of the elements that indicate the importance for the categorization for the current data.

I. INTRODUCTION

There are various kinds of neural networks to be used in pattern classification problems, e.g. backpropagation neural network [6], radial basis function network [4], k-nearest neighbor classifier, probabilistic neural network by Bayes classification [7], etc. The function of pattern classification networks is usually limited to make a judgment to which class an input vector belongs.

On the other hand, there are also many kinds of neural networks that are used for unsupervised learning. The primary functions of this sort of algorithms are to cluster the input data or to find information of the density that are embedded in the data.

Like the k-means clustering, SOM learns the centers of the clusters, but SOM is not limited to that. The topological relation of the input data space is well preserved in the low-dimensional SOM. This feature was utilized in the phoneme typewriter [2], where the relative similarity between the phonemes are easily seen in two dimensional SOM space.

There are works where GA was used for SOM, for example, Polani and Uthmann [5] used GA for the topological determination of SOM, but the objective of using GA is different from ours. The formation of SOM heavily depends on the data scaling. If the critical elements are emphasized, it is more likely that, on the SOM, different class of data are separated and mapped to different units and same class data tend to form a concentrated mass. Thus, an appropriate scaling is necessary for this purpose, but it is a very difficult problem, as Kohonen suggests to do it by methods like “try and error” [2].

We will modify the canonical SOM in this paper as follows.

1. Category information is attached to the training vector with a fixed parameter \( \rho \).

2. Elements of training vector are also weighted, and these weights are determined by GA. The evaluation of the set of weights is done by classification ability of the corresponding SOM.

II. SOM

Here we briefly explain the structure and the learning method of SOM. We use two-dimensional SOM, with which the result can be displayed visually. SOM is defined on two-dimensional discrete space \( \Omega \), and at each grid point \( r \in \Omega \), SOM has a codebook vector \( m_r \in \mathbb{R}^n \).

SOM's objective function to be minimized is

\[
J = \sum_n J(n) = \sum_n \sum_{r \in \Omega} \Phi(||r - r^*(x(n))||)||x(n) - m_r||^2
\]

where \( \Phi(p) \geq 0 \) is a monotonically decreasing function that takes

\[
\Phi(0) = 1, \quad \Phi(\infty) = 0
\]

and \( r^*(x(n)) \) is the point whose codebook vector is most similar to \( x(n) \); i.e.,

\[
||x(n) - m_r^*(x(n))|| \leq ||x(n) - m_r|| \quad \forall r \in \Omega
\]

Here \( \Phi(p) \) is defined as

\[
\Phi(p) = \exp\left(-\frac{p^2}{2\sigma^2}\right)
\]

We can minimize this objective function by modifying the codebook vectors against the gradient vector \( \partial J(n)/\partial x(n) \), that is

\[
m_r(n) = m_r(n-1) + \lambda_r \Phi(||r - r^*(x(n))||) \\
\times (x(n) - m_r(n-1))
\]

When the number of data is not large, the data are repeatedly used.

III. WEIGHT TUNING BY GA

In the canonical SOM, all the elements of data are used with same weight, that is, the distance between the input vector and the code vector is measured by the Euclidian distance. To extract useful information that reflects the output category, it is a useful method to scale the vector element wise. If an element is uncorrelated to the...
output (category), the element seems to be the noise. In statistical field, the discrimination analysis is to find the optimal hyperplane that divides the data space into two categories. Our method is similar to that, but not only that. Our method is aimed at making discrimination on the SOM grids. It may be possible to compute the weights by correlation analysis, but it does not use the same criterion for SOM formation.

Here we decide the weights by using the genetic algorithm, where one SOM corresponds to an individual. The weight vector $w = (w_1, w_2, \ldots, w_n)$ and the weighted norm are defined by

$$||x - m||_w := \sqrt{\sum_i w_i^2 (x_i - m_i)^2} \tag{2}$$

where $0 \leq w_i \leq 1$.

IV. SUPERVISED SOM WITH WEIGHT

It is easy to form SOM as a supervised learning by using an augmented vector

$$x = [u^T, \rho y^T]^T$$

where the superscript $T$ denotes the matrix transpose, $u$ is the input vector, $y$ is the output (training) data that expresses the class ($y_i = 0$ or $1$), and $x$ is the new input vector to SOM. If $\rho = 0$, this reduces to the canonical SOM, and as $\rho$ becomes large, the vector $x$ tends to contain the class information only. It is possible to tune the classification ability and the feature mapping ability which SOM originally has.

The algorithm is summarized as follows.

Initialization: Generate $\{m_r(0), r \in \Omega\}$ randomly.

Repeat: For $t = 1, 2, \ldots, L$; $k = 1, 2, \ldots, N$:

A-2-1

$$n = (t - 1)N + k$$

$$r^* = \arg\min_{r \in \Omega} ||m_r - x(k)||_w$$

A-2-2

$$m_r(n) = m_r(n - 1) + \alpha(n) \times N_e(||x(k) - m_r||; s)||x(k) - m_r(n - 1)||$$

$\alpha(n)$ is a pre-defined monotonically decreasing function, e.g. $\alpha(t) = 1/n$, and $N_e(p; s)$ is a Gaussian function defined by

$$N_e(p; s) = \exp \left( -\frac{p^2}{s^2} \right)$$

where $s$ is a parameter defined as

$$s = \exp(-t/50)$$

V. GA OPERATIONS

Here we will define the whole GA structure, evaluation, crossover and mutation operations.

A. GA code

The code (chromosome) used for GA is defined as

$$I_t = (w_1^t w_2^t \cdots w_n^t)$$

where $w_i^t$ is defined by a 3-bit code. Thus totally a chromosome is defined by a 3n-bit code as

$$I_t = (c_1^t c_2^t \cdots c_{3n}^t)$$

Also,

<table>
<thead>
<tr>
<th>genotype</th>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
</tr>
</thead>
<tbody>
<tr>
<td>phenotype</td>
<td>0/7</td>
<td>1/7</td>
<td>2/7</td>
<td>3/7</td>
</tr>
<tr>
<td>genotype</td>
<td>100</td>
<td>101</td>
<td>110</td>
<td>111</td>
</tr>
<tr>
<td>phenotype</td>
<td>4/7</td>
<td>5/7</td>
<td>6/7</td>
<td>7/7</td>
</tr>
</tbody>
</table>

B. Flow of GA

The genetic algorithm operation used here is the following.

1. Initialization

(a) Randomly generate $N$ individuals.

(b) Compute SOM using the training data, where all the data are cyclically used for $T$ times.

2. Iteration Iterate the following for a fixed times, where the best one is copied to the next generations.

(a) Evaluate the individuals.

(b) Reproduction Based on the evaluation, new population is generated by reproduction.

(c) Crossover The crossover is carried out for all the individuals.

(d) Mutation Some bits are inverted by the mutation operation.

C. Evaluation

The evaluation of individuals is given a value in $(0, 1]$ by

$$EV(I_t) = 1 - \sum_{k=1}^{N_1} \sum_{l=1}^{N_2} c(k,l)/(N_1N_2)$$

where $c(k,l) = \min(a(k,l), b(k,l))/\max(a(k,l), b(k,l))$

and $(a(k,l), b(k,l))$ denotes the number of instants of classes 0 and 1 that fell onto the SOM grid $(k,l)$.

D. Reproduction

The evaluated value itself doesn’t have a special meaning. Rather, the ranking of the individuals is more important. Thus, here we employ a stochastic ranking reproduction, where probability of reproduction is linearly decreasing along the ranking axis. Elitist strategy is also adopted, where the best one is kept in the next generation.
E. Crossover

Uniform crossover [8] is adopted here. This operation is as follows.
1. Pick up two individuals $I_i, I_j$ from the population.
2. For $k = 1, \ldots, n$, exchange the $w^k_i$ with $w^k_j$ with probability $p$.

F. Mutation

Every $c_k$ is inverted with probability $p_m$.

VI. NUMERICAL EXAMPLE

A. Data Description

The database “Pima Indians Diabetes” from UCI Repository[3] is used here. The diagnostic, binary-valued variable investigated is whether the patient shows signs of diabetes according to World Health Organization criteria (i.e., if the 2 hour post-load plasma glucose was at least 200 mg/dl at any survey examination or if found during routine medical care). All patients here are females at least 21 years old of Pima Indian heritage, and the population lives near Phoenix, Arizona, USA.

The number of Instances is 768, but there exist many incomplete data. The attributes are all numeric-valued, and the input variables are
1. Number of times pregnant
2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (\( \mu U/ml \))
6. Body mass index (weight in kg/(height in m)^2)
7. Diabetes pedigree function
8. Age (years)

The output value is 0 (for sick) and 1 (for healthy). The class Distribution is shown in Table 1.

<table>
<thead>
<tr>
<th>Class</th>
<th>Complete data</th>
<th>Training data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>130</td>
<td>65</td>
</tr>
<tr>
<td>1</td>
<td>262</td>
<td>131</td>
</tr>
</tbody>
</table>

B. Classification by Traditional Methods

We briefly show the classification results by using the normalized (i.e. scaled to the range \([0,1]\)) data without rescaling (the same weights (=1)).

It is well known that the Perceptron can classify only linearly separable data. The correct judgment for the training data and the checking data were 66.6% and 69.1%, respectively.

By the discriminant analysis, the correct judgment was 80.6% for the training data and 77.5% for the test data.

Next we used the backpropagation network. The network is with one hidden layer. Correct judgments in some typical experimental results are 84.7%, 84.2% and 84.2% for the training data and 78.6%, 79.6% and 79.6% for the checking data by the network with hidden units 5, 10 and 15, respectively.

C. Effect of $\rho$

We will show the classification results by using SOM hereafter. As the weight of the output $\rho$ increases, the classification performance for the training data monotonically increases, but it is not so for the checking data. In our experiment, $\rho = 0.07$ showed the best classification result.

D. Proposed Method

Genetic algorithm used here is as follows.

The population was 20, and the iteration times was 100. The crossover probability was set to $p_c = 0.8$, and for each pair of individuals, the uniform crossover was adopted with exchange rate 0.5. The mutation probability $p_m = 0.02$ for each bit.

For the data of all the elements with the same weight (= 1), \((n_0, n_1)\) in Table 2 shows the number of training data that are mapped to each grid for categories, where $n_0$ shows the ones with category 0, and $n_1$ for category 1, respectively.

Table 4 shows the one after the GA learning.

In Table 3, the obtained weight coefficients are shown.

<table>
<thead>
<tr>
<th>Class</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete data</td>
<td>(n_0)</td>
<td>(n_1)</td>
<td>(n_0)</td>
<td>(n_1)</td>
<td>(n_0)</td>
<td>(n_1)</td>
<td>(n_0)</td>
<td>(n_1)</td>
<td>(n_0)</td>
</tr>
</tbody>
</table>

Table 2. Supervised SOM without weighting

\[
\begin{array}{ccc}
9 & 6 & (0, 10) \\
11 & 2 & (4, 2) \\
2 & 1 & (6, 3) \\
3 & 5 & (7, 6) \\
\end{array}
\]

Table 3. Weight coefficients of the best individual at 100th generation

\[
\begin{array}{cccccccc}
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | \\
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11</td>
<td>110</td>
<td>001</td>
<td>000</td>
<td>000</td>
<td>000</td>
<td>001</td>
<td>011</td>
</tr>
</tbody>
</table>
\end{array}
\]

Table 4. Classification after learning

\[
\begin{array}{cccc}
(9,2) & (7,3) & (13,2) & (7,0) \\
(3,17) & (0,3) & (4,8) & (2,7) \\
(10,0) & (1,12) & (2,10) & (0,9) \\
(2,20) & (2,12) & (1,14) & (2,12) \\
\end{array}
\]

In Table 5, the codebook vectors are shown. The position with * are assumed to belong to class 0 (sick).

The shown values are not scaled, but the scaled values were used in learning. These indicate the representative vector of each unit, respectively. There are various ways to use this table. For example, it is possible to know to which class a patient belongs to, and see whether it is close to the illness or completely healthy group. It may also be possible to trace patient's data, and explain at which stage the patient is.
Table 5. Codebook vectors

<table>
<thead>
<tr>
<th>Position</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,1)*</td>
<td>0.56</td>
<td>181</td>
<td>73.4</td>
<td>33.0</td>
<td>325</td>
<td>35.9</td>
<td>0.52</td>
<td>38.3</td>
</tr>
<tr>
<td>(1,2)*</td>
<td>2.97</td>
<td>174</td>
<td>73.0</td>
<td>29.4</td>
<td>249</td>
<td>34.7</td>
<td>0.66</td>
<td>28.7</td>
</tr>
<tr>
<td>(1,3)*</td>
<td>7.13</td>
<td>162</td>
<td>77.4</td>
<td>31.4</td>
<td>350</td>
<td>33.8</td>
<td>0.57</td>
<td>40.4</td>
</tr>
<tr>
<td>(1,4)*</td>
<td>12.20</td>
<td>130</td>
<td>77.0</td>
<td>30.3</td>
<td>184</td>
<td>34.9</td>
<td>0.50</td>
<td>48.6</td>
</tr>
<tr>
<td>(2,1)</td>
<td>0.40</td>
<td>140</td>
<td>72.2</td>
<td>30.9</td>
<td>252</td>
<td>34.8</td>
<td>0.49</td>
<td>22.7</td>
</tr>
<tr>
<td>(2,2)</td>
<td>2.31</td>
<td>137</td>
<td>88.0</td>
<td>27.1</td>
<td>107</td>
<td>25.9</td>
<td>0.57</td>
<td>24.5</td>
</tr>
<tr>
<td>(2,3)</td>
<td>3.67</td>
<td>132</td>
<td>73.8</td>
<td>28.5</td>
<td>174</td>
<td>32.3</td>
<td>0.52</td>
<td>34.7</td>
</tr>
<tr>
<td>(2,4)</td>
<td>8.42</td>
<td>112</td>
<td>79.2</td>
<td>33.5</td>
<td>164</td>
<td>32.2</td>
<td>0.42</td>
<td>49.3</td>
</tr>
<tr>
<td>(3,1)*</td>
<td>0.70</td>
<td>123</td>
<td>70.1</td>
<td>35.0</td>
<td>166</td>
<td>37.8</td>
<td>0.50</td>
<td>31.7</td>
</tr>
<tr>
<td>(3,2)</td>
<td>1.56</td>
<td>119</td>
<td>71.1</td>
<td>30.7</td>
<td>162</td>
<td>33.8</td>
<td>0.53</td>
<td>25.0</td>
</tr>
<tr>
<td>(3,3)</td>
<td>3.55</td>
<td>119</td>
<td>72.3</td>
<td>24.7</td>
<td>146</td>
<td>34.3</td>
<td>0.47</td>
<td>25.8</td>
</tr>
<tr>
<td>(3,4)</td>
<td>6.02</td>
<td>96</td>
<td>67.3</td>
<td>25.9</td>
<td>75</td>
<td>30.1</td>
<td>0.45</td>
<td>32.6</td>
</tr>
<tr>
<td>(3,5)</td>
<td>0.41</td>
<td>102</td>
<td>68.8</td>
<td>25.3</td>
<td>112</td>
<td>33.2</td>
<td>0.38</td>
<td>22.7</td>
</tr>
<tr>
<td>(3,6)</td>
<td>0.88</td>
<td>86</td>
<td>67.6</td>
<td>26.7</td>
<td>88</td>
<td>33.0</td>
<td>0.44</td>
<td>27.8</td>
</tr>
<tr>
<td>(3,7)</td>
<td>2.21</td>
<td>97</td>
<td>60.2</td>
<td>22.8</td>
<td>80</td>
<td>28.9</td>
<td>0.48</td>
<td>23.5</td>
</tr>
<tr>
<td>(3,8)</td>
<td>2.87</td>
<td>82</td>
<td>69.8</td>
<td>26.1</td>
<td>69</td>
<td>30.6</td>
<td>0.52</td>
<td>27.7</td>
</tr>
</tbody>
</table>

Table 6. Codebook vectors of sickness

<table>
<thead>
<tr>
<th>Position</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,1)</td>
<td>0.56</td>
<td>181</td>
<td>73.4</td>
<td>33.0</td>
<td>325</td>
<td>35.9</td>
<td>0.52</td>
<td>38.3</td>
</tr>
<tr>
<td>(1,2)</td>
<td>2.97</td>
<td>174</td>
<td>73.0</td>
<td>29.4</td>
<td>249</td>
<td>34.7</td>
<td>0.66</td>
<td>28.7</td>
</tr>
<tr>
<td>(1,3)</td>
<td>7.13</td>
<td>162</td>
<td>77.4</td>
<td>31.4</td>
<td>350</td>
<td>33.8</td>
<td>0.57</td>
<td>40.4</td>
</tr>
<tr>
<td>(1,4)</td>
<td>12.20</td>
<td>130</td>
<td>77.0</td>
<td>30.3</td>
<td>184</td>
<td>34.9</td>
<td>0.50</td>
<td>48.6</td>
</tr>
<tr>
<td>(2,1)</td>
<td>0.40</td>
<td>140</td>
<td>72.2</td>
<td>30.9</td>
<td>252</td>
<td>34.8</td>
<td>0.49</td>
<td>22.7</td>
</tr>
<tr>
<td>(2,2)</td>
<td>2.31</td>
<td>137</td>
<td>88.0</td>
<td>27.1</td>
<td>107</td>
<td>25.9</td>
<td>0.57</td>
<td>24.5</td>
</tr>
<tr>
<td>(2,3)</td>
<td>3.67</td>
<td>132</td>
<td>73.8</td>
<td>28.5</td>
<td>174</td>
<td>32.3</td>
<td>0.52</td>
<td>34.7</td>
</tr>
<tr>
<td>(2,4)</td>
<td>8.42</td>
<td>112</td>
<td>79.2</td>
<td>33.5</td>
<td>164</td>
<td>32.2</td>
<td>0.42</td>
<td>49.3</td>
</tr>
</tbody>
</table>

Table 6 is the sub-table of Table 5, where only the codebook vectors that seem to belong to sickness are shown. Since Table 2 indicates that the elements 1, 2, 3, 7 and 8 are relatively important, we concentrate on them. Then the vector at (1,1) represents people with high glucose, (1,2) is of young women, (1,3) and (1,4) are of the relatively old women with many pregnant times, and (3,1) is of diabetes pedigree.

Fig. 1 shows the transition of the best individuals among the population for several trials. We can see that, in all the cases, the GA made a success.

Fig. 1. Evolution of best individual

E. Using Other Data

Fisher’s Iris data [1] is often used for the pattern classification problems. The data consists of three categories, each consists of 50 sets of 4-dimensional input.

Half of the data was used as the training data and the rest for the checking.

Table 7. Classification of Irises

<table>
<thead>
<tr>
<th>Training Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>(25,0.0)</td>
</tr>
<tr>
<td>(0,17.0)</td>
</tr>
<tr>
<td>(0,6.0)</td>
</tr>
<tr>
<td>(0,2.5)</td>
</tr>
<tr>
<td>(0,2.0)</td>
</tr>
</tbody>
</table>

Checking Data

| (25,0.0)      |
| (0,20.3)      |
| (0,5.0)       |
| (0,0.22)      |

The corresponding weights are

Table 8. Weight coefficients of the best individual at 100th generation

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>0010</td>
<td>1111</td>
<td>1001</td>
</tr>
</tbody>
</table>

VII. CONCLUSIONS

Under the framework of genetic algorithm, the weight tuning problem of the elements of data was treated. By setting the criterion to take high values if the training data were well separated on the SOM grids, consistent weight vectors were obtained. Since SOM is used as the classifier, various unsupervised classification of the data was obtained. A good representation of the obtained codebook vectors needs to be further explored.

REFERENCES