Chart Pattern Recognition using the Bees Algorithm

Afshin Ghanbarzadeh¹

Mechanical Engineering Department, Faculty of Engineering, Shahid Chamran University

Ghanbarzadeh.A@scu.ac.ir

Abstract

Control charts are employed in manufacturing industry for statistical process control (SPC). It is possible to detect incipient problems and prevent a process from going out of control by identifying the type of patterns displayed by the control charts. Various techniques have been applied to this control chart pattern recognition task. This paper presents the use of radial basis function networks for recognising patterns in control charts in order to determine if the process being monitored is operating normally or if it shows gradual changes (trends), sudden changes (shifts) or periodic changes (cycles). The radial basis function networks were trained, not by applying standard training algorithms, but by employing a new optimisation algorithm developed by the authors. The algorithm, called the Bees Algorithm, is inspired by the food foraging behaviour of honey bees. The paper briefly explains the Bees Algorithm and gives the results obtained.

Keywords: Bees Algorithm, Neural Network, Pattern Recognition, Control Charts, Swarm Intelligence.

Introduction

Statistical Process Control (SPC) employs statistical means such as control charts to show how consistently a process is performing and whether it should be adjusted [1]. SPC control charts enable a manufacturing engineer to compare the actual performance of a process with customer specifications and provide a process capability index to guide and assess quality improvement efforts. By means of simple rules, it is possible to determine if a process is out of control and needs corrective action. However, incipient problems could be detected before the process goes out of control from the type of patterns displayed by the control charts. Various techniques have been applied to this control chart pattern recognition task [2, 3].

This paper presents the use of Radial Basis Function (RBF) networks for recognising patterns in control charts in order to determine if the process being monitored is operating normally or if it shows gradual changes (trends), sudden changes (shifts) or periodic changes (cycles) (Figure 1). The RBF

¹ Assistant Professor
networks were trained, not by applying standard training algorithms, but by employing a new optimisation algorithm developed by the authors. The algorithm, called the Bees Algorithm, is inspired by the behaviour of honey bees [4, 5].

The paper is organised as follows. Section 2 outlines the Bees Algorithm. Section 3 explains the RBF network and both the standard RBF training method and the training procedure based on the Bees Algorithm. Section 4 presents the results of control chart pattern recognition experiments with RBF networks trained using the Bees Algorithm and the conventional RBF procedure.

The Bees Algorithm
A.Bees in nature
A colony of honey bees can be seen as a diffuse creature which can extend itself over long distances in multiple directions in order to exploit a large number of food sources at the same time [6, 7]. In principle, flower patches with plentiful amounts of nectar or pollen that can be collected with less effort should be visited by more bees, whereas patches with less nectar or pollen should receive fewer bees [7, 8].

The foraging process begins in a colony by scout bees being sent to search for promising flower patches. Scout bees search randomly from one patch to another. During the harvesting season, a colony continues its exploration, keeping a percentage of the population as scout bees [6].

When they return to the hive, those scout bees that found a patch which is rated above a certain threshold (measured as a combination of some constituents, such as sugar content) deposit their nectar or pollen and go to the “dance floor” to perform a dance known as the “waggle dance” [7]. This dance is essential for colony communication, and contains three vital pieces of information regarding a flower patch: the direction in which it will be found, its distance from the hive and its quality rating (or fitness) [6, 9]. This information helps the bees to find the flower patches precisely, without using guides or maps. Each individual’s knowledge of the outside environment is gleaned solely from the waggle dance. This dance enables the colony to evaluate the relative merit of different patches according to both the quality of the food they provide and the amount of energy needed to harvest it [9]. After waggle dancing on the dance floor, the dancer bee (i.e. the scout bee) goes back to the flower patch with follower bees that were waiting inside the hive. The number of follower bees assigned to a patch depends on the overall quality of the patch. This allows the colony to gather food quickly and efficiently.
While harvesting from a patch, the bees monitor its food level. This is necessary to decide upon the next waggle dance when they return to the hive [9]. If the patch is still good enough as a food source, then it will be advertised in the waggle dance and more bees will be recruited to that source.

**B. Bees Algorithm**

As mentioned above, the Bees Algorithm is an optimisation algorithm inspired by the natural foraging behaviour of honey bees [5]. Figure 2 shows the pseudo code for the algorithm in its simplest form. The algorithm requires a number of parameters to be set, namely: number of scout bees (n), number of points selected out of n visited points (m), number of elite points out of m selected points (e), number of bees recruited for the best e points (nep), number of bees recruited for the other (m-e) selected points (nsp), initial size of patches (ngh) which includes point and its neighbourhood, and stopping criterion. The algorithm starts with the n scout bees being placed randomly in the search space. The fitnesses of the points visited by the scout bees are evaluated in step 2.

1. Initialise population with random solutions.
2. Evaluate fitness of the population.
3. While (stopping criterion not met) //Forming new population.
   4. Select points for neighbourhood search.
   5. Recruit bees for selected points (more bees for the best e points) and evaluate fitnesses.
   6. Select the fittest bee from each patch.
   7. Assign remaining bees to search randomly and evaluate their fitnesses.
   8. End While.

**Figure 2: Pseudo code of the basic Bees Algorithm.**

In step 4, bees that have the highest fitnesses are designated as “selected bees” and points visited by them are chosen for neighbourhood search. Then, in steps 5 and 6, the algorithm conducts searches in the neighbourhood of the selected points, assigning more bees to search near to the best e points. The bees can be chosen directly according to the fitnesses associated with the points they are visiting. Alternatively, the fitness values are used to determine the probability of the bees being selected. Searches in the neighbourhood of the best e points which represent more promising solutions are made more detailed by recruiting more bees to follow selected bees than the other selected bees. Together with scouting, this differential recruitment is a key operation of the Bees Algorithm.

In step 6, for each patch only the bee with the highest fitness will be selected to form the next bee population. In nature, there is no such a restriction. This constraint is introduced here to reduce the number of points to be explored. In step 7, the remaining bees in the population are assigned randomly around the search space scouting for new potential solutions. These steps are repeated until a stopping criterion is met. At the end of each iteration, the colony will have two parts to its new population – representatives from each selected patch and other scout bees assigned to conduct random searches.

**Radial Basis Function (RBF) Network**

**A. Network structure**

The radial basis function (RBF) network is a popular type of network that is very useful for pattern classification problems [10]. Figure 3 shows the structure of a RBF network which consists of three layers of neurons.

The input layer neurons receive the input pattern \((x_1 \text{ to } x_N)\). The hidden layer neurons provide a set of activation functions that constitute an arbitrary “basis” for the input patterns in the input space to be expanded into the hidden space by way of non-linear transformation. At the input of each hidden neuron, the distance between the centre of each activation or basis function and the input vector is calculated. Applying the basis function to this distance produces the output of the hidden neuron. The
RBF network outputs $y_1$ to $y_p$ are formed by the neurons in the output layer as weighted sums of the hidden layer neuron activations.

The basis function is generally chosen to be a standard function which is positive at its centre $x=0$, and then decreases uniformly to zero on either side. A common choice is the Gaussian distribution function:

$$K(x) = \exp\left(-\frac{x^2}{2}\right)$$  \hspace{1cm} (1)

This function can be shifted to an arbitrary centre, $x=c$, and stretched by varying its spread $\sigma$ as follows:

$$K\left(\frac{x-c}{\sigma}\right) = \exp\left(-\frac{(x-c)^2}{2\sigma^2}\right)$$  \hspace{1cm} (2)

The outputs of the RBF network $y_j$ are given by:

$$y_j = \sum_{i=1}^{p} w_{ji} K\left(\frac{x-c}{\sigma_i}\right)$$  \hspace{1cm} (3)

where $w_{ji}$ is the weight of the hidden neuron $i$ to output $j$, $c_i$ the centre of basis function $i$ and $\sigma_i$ the spread of the function. $\|x-c_i\|$ is the norm of $(x-c_i)$. There are various ways to calculate the norm. The most common is the Euclidean norm given by:

$$\|x-c\| = \sqrt{(x_1-c_1)^2 + (x_2-c_2)^2 + ... + (x_N-c_N)^2}$$  \hspace{1cm} (4)
This norm gives the distance between the two points \( x \) and \( c_i \) in \( N \)-dimensional space. All points \( x \) that are the same radial distance from \( c_i \) give the same value of the norm. The purpose of training an RBF network is to determine the neuron weights \( w_{ji} \), RBF centres \( c_i \) and spreads \( \sigma_i \) that enable the network to produce the correct outputs \( y_j \) corresponding to the input patterns \( x \).

**B. RBF network training procedure**

The training of an RBF network involves the minimisation of an error function. The error function defines the total difference between the actual output and the desired output of the network over a set of training patterns [11]. Training proceeds by presenting to the network a pattern of known class taken from the training set. The error component associated with that pattern is the sum of the squared differences between the desired and actual outputs of the network corresponding to the presented pattern. The procedure is repeated for all the patterns in the training set and the error components for all the patterns are summed to yield the value of the error function for an RBF network with a given set of basis function centres, spreads and neuron connection weights.

**Standard RBF network training procedure**

With the standard procedure for training RBF networks, after the number of hidden neurons \( h \) has been decided, the following steps will be taken:

1. Choose the RBF centres \( c_i \); centre selection could be performed by trial and error, self-organised or supervised.
2. Choose spreads \( \sigma_i \); several heuristic methods are available. A popular method is to set \( \sigma_i \) equal to the distance to the centre nearest to \( c_i \).

Calculate neuron weights \( w_{ji} \); when \( c_i \) and \( w_{ji} \) are known, the outputs of hidden neurons \( (K_1, \ldots, K_h)^T \) can be calculated for any pattern of inputs \( x = (x_1, \ldots, x_N) \). Assuming there are \( s \) input patterns \( x \) in the training set, there will be \( s \) sets of hidden neuron outputs that can be calculated. These can be assembled into a \( h \times s \) matrix:

\[
K = \begin{bmatrix}
  k_1^1 & k_1^2 & \ldots & k_1^s \\
  k_2^1 & k_2^2 & \ldots & k_2^s \\
  \vdots & \ddots & \vdots & \vdots \\
  k_h^1 & k_h^2 & \ldots & k_h^s \\
\end{bmatrix}
\]

The output of the RBF network \( (y) \) is given by equation (6).

\[
y = K^T w^T
\]

where

\[
w^T = \begin{bmatrix}
w_{11} & w_{12} & \ldots & w_{1p} \\
w_{21} & w_{22} & \ldots & w_{2p} \\
\vdots & \ddots & \vdots & \vdots \\
w_{h1} & w_{h2} & \ldots & w_{hp}
\end{bmatrix}
\]

\( y \) is the matrix of actual outputs corresponding to the training inputs \( x \). Ideally, \( y \) should be equal to \( d \), the desired or target outputs. Unknown coefficients \( w_{ji} \) can be calculated from equation (8) in order to minimise the sum of the squared differences between \( y \) and \( d \).
\[ w^r = (KK^T)^{-1}Kd \]  

**RBF network training using the Bees Algorithm**

In terms of the Bees Algorithm, each bee represents an RBF network with a particular set of basis function centres, spreads and weight vectors. The aim of the algorithm is to find the bee producing the smallest value of the error function.

The RBF network training procedure using the Bees Algorithm thus comprises the following steps:

1. Generate an initial population of bees.
2. Apply the training data set to determine the value of the error function associated with each bee.
3. Based on the error value obtained in step 2, create a new population of bees comprising the best bees in the selected neighbourhoods and randomly placed scout bees.
4. Stop if the value of the error function has fallen below a predetermined threshold or after completing a set number of iterations.
5. Else, return to step 2.

**Control Chart Pattern Recognition Experiments**

**A. Control chart patterns**

Each pattern used in the experiments was a time series comprising 60 points. The value \( y(t) \) at each point \( t \) was normalised to fall in the range \([0, +1]\) according to the following equation [12]:

\[
\bar{y}(t) = \frac{y(t) - y_{\min}}{y_{\max} - y_{\min}}
\]  

where

- \( \bar{y}(t) \) = scaled pattern value (in the range 0 to 1)
- \( y_{\min} \) = minimum allowed value (taken as 35)
- \( y_{\max} \) = maximum allowed value (taken as 125)

**B. Training and test data**

A total of 1500 patterns, 250 patterns in each of the six classes, were generated using the following equations:

1. **Normal patterns:**
   \[ y(t) = \mu + r(t) \sigma \]  

2. **Cyclic patterns:**
   \[ y(t) = \mu + r(t) \sigma + a \sin(2\pi t / T) \]  

3. **Increasing or decreasing trends:**
   \[ y(t) = \mu + r(t) \sigma \pm g t \]  

4. **Upwards or downwards shifts:**
   \[ y(t) = \mu + r(t) \sigma \pm k s \]  

where

- \( \mu \) = mean value of the process variable being monitored (taken as 80 in this work)
- \( \sigma \) = standard deviation of the process (taken as 5)
- \( a \) = amplitude of cyclic variations (taken as 15 or less)
- \( g \) = magnitude of the gradient of the trend (taken as being in the range 0.2 to 0.5)
\( k \) = parameter determining the shift position (0 before the shift position; 1 at the shift position and thereafter)

\( r \) = normally distributed random number (between \(-3\) and \(+3\))

\( s \) = magnitude of the shift (taken as being in the range 7.5 to 20)

\( t \) = discrete time at which the pattern is sampled (taken as being within the range 0 to 59)

\( T \) = period of a cycle (taken as being in the range 4 to 12 sampling intervals)

\( y(t) \) = sample value at time \( t \)

498 patterns (83 in each class) were used for training an RBF network and 1002 patterns (167 in each class) were employed for testing the trained network.

C. RBF network configuration

The RBF network configuration used involves three layers: an input layer, a hidden layer and an output layer. The input layer has 60 neurons, one for each point in a pattern. The hidden layer consists of 35 neurons. The output layer comprises 6 neurons, one for each of the six classes as shown in Table 1. Therefore, each bee would define a 2345-dimensional vector (60*35+6*35+35).

Table 1: Representation of the output categories.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Class</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>1</td>
<td>1 0 0 0 0 0</td>
</tr>
<tr>
<td>Increasing trends</td>
<td>2</td>
<td>0 1 0 0 0 0</td>
</tr>
<tr>
<td>Decreasing trends</td>
<td>3</td>
<td>0 0 1 0 0 0</td>
</tr>
<tr>
<td>Upwards shifts</td>
<td>4</td>
<td>0 0 0 1 0 0</td>
</tr>
<tr>
<td>Downwards shifts</td>
<td>5</td>
<td>0 0 0 0 1 0</td>
</tr>
<tr>
<td>Cyclic</td>
<td>6</td>
<td>0 0 0 0 0 1</td>
</tr>
</tbody>
</table>

D. Bees Algorithm parameters

Table 2 shows the parameter values adopted for the Bees Algorithm. The values were decided empirically.

Table 2: Parameters of the Bees Algorithm.

<table>
<thead>
<tr>
<th>Bees Algorithm parameters</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population</td>
<td>n</td>
<td>200</td>
</tr>
<tr>
<td>Number of selected points</td>
<td>m</td>
<td>10</td>
</tr>
<tr>
<td>Number of elite points out of m selected points</td>
<td>e</td>
<td>2</td>
</tr>
<tr>
<td>Initial patch size</td>
<td>ngh</td>
<td>0.1</td>
</tr>
<tr>
<td>Number bees for elite points</td>
<td>nep</td>
<td>80</td>
</tr>
<tr>
<td>Number of bees for other selected points</td>
<td>nsp</td>
<td>20</td>
</tr>
</tbody>
</table>

Result

Table 3 presents the classification results obtained for ten separate runs of the Bees Algorithm. The average for the ten runs is given in Table 4 against the classification results for RBF networks trained using the standard algorithm implemented in the MATLAB software. It can be seen that the test and training accuracies in the case of the Bees Algorithm are very close to those for the standard RBF procedure. In fact, the value of the error function (which is the optimisation criterion for the Bees
Algorithm) is smaller for a Bees-Algorithm-trained RBF network than for an RBF network created using the standard procedure and having five times as many hidden neurons.

Table 3: RBF classification results.

<table>
<thead>
<tr>
<th>Number of runs</th>
<th>Training accuracy</th>
<th>Test accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>99.35%</td>
<td>98.79%</td>
</tr>
<tr>
<td>2</td>
<td>99.65%</td>
<td>99.15%</td>
</tr>
<tr>
<td>3</td>
<td>99.14%</td>
<td>98.51%</td>
</tr>
<tr>
<td>4</td>
<td>99.83%</td>
<td>99.46%</td>
</tr>
<tr>
<td>5</td>
<td>99.82%</td>
<td>99.44%</td>
</tr>
<tr>
<td>6</td>
<td>99.57%</td>
<td>98.99%</td>
</tr>
<tr>
<td>7</td>
<td>99.84%</td>
<td>99.43%</td>
</tr>
<tr>
<td>8</td>
<td>99.43%</td>
<td>98.84%</td>
</tr>
<tr>
<td>9</td>
<td>99.81%</td>
<td>99.45%</td>
</tr>
<tr>
<td>10</td>
<td>99.45%</td>
<td>98.95%</td>
</tr>
<tr>
<td>Max</td>
<td>99.84%</td>
<td>99.46%</td>
</tr>
<tr>
<td>Min</td>
<td>99.14%</td>
<td>98.51%</td>
</tr>
<tr>
<td>Mean</td>
<td>99.59%</td>
<td>99.10%</td>
</tr>
</tbody>
</table>

A typical plot of how classification accuracy evolves during training is shown in Figure 4.

![Figure 4: Typical plot of classification accuracy versus number of training iterations.](image)

Table 4: Comparison with conventional RBF training.

<table>
<thead>
<tr>
<th>Pattern recognition</th>
<th>No of hidden neurons</th>
<th>Error</th>
<th>Training Accuracy</th>
<th>Test accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBF (MTLAB)</td>
<td>35</td>
<td>32.6</td>
<td>100</td>
<td>99.6</td>
</tr>
<tr>
<td>RBF (MTLAB)</td>
<td>175</td>
<td>9.3</td>
<td>100</td>
<td>99.7</td>
</tr>
<tr>
<td>RBF (MTLAB)</td>
<td>498</td>
<td>0.02</td>
<td>100</td>
<td>99.8</td>
</tr>
<tr>
<td>RBF (Bees Algorithm)</td>
<td>35</td>
<td>8.9</td>
<td>99.6</td>
<td>99.1</td>
</tr>
</tbody>
</table>

Conclusion
This paper has described the use of the Bees Algorithm to train RBF networks for control chart pattern recognition. Despite the high dimensionality of the problem – each bee represented 2345 parameters
that had to be determined – the algorithm still succeeded to train very accurate classifiers. Although
the accuracy achieved is marginally lower than that obtained with conventionally-trained RBF
networks, the Bees Algorithm can solve a problem without any information apart from that needed
to evaluate fitnesses. In this respect, the Bees Algorithm shares the same advantage as global search
algorithms such as the Genetic Algorithm (GA). Also, comparing classification accuracies is strictly
not fair to the Bees Algorithm in this case because the optimisation criterion used is the total output
error value rather than classification accuracy. As the results obtained demonstrate, the Bees
Algorithm produced RBF networks with a lower error value than that obtained for an RBF network
trained conventionally even when the latter had five times as many hidden neurons.

References
Control Chart Pattern Recognition. in Proc of Institution of Mechanical Engineers.
Statistical Process Control. in Proc of Institution of Mechanical Engineers.
Novel Tool for Complex Optimisation Problems. in Proc 2nd Int Virtual Conf on Intelligent
Algorithm. Manufacturing Engineering Centre, Cardiff University: Cardiff.
N.Y.: Cornell University Press.
Cambridge, Massachusetts: Harvard University Press.