The Vector Clustering Based on the Recursive Particle Swarm Optimization with Radial Basis Function Networks Modeling System

Xue-ming JIA
College of Information Security, Yunnan Police Officer Academy, Kunming, 650223, China

Keywords: Recursive particle swarm optimization, Recursive estimation, Dynamic optimization, Radial basis function networks modeling system.

Abstract. The vector clustering plays an important role to the applications of information science, such as information compression. The particle swarm optimization is used to implement the clustering operation. In this paper, a Recursive Particle Swarm Optimization (RPSO) is proposed to solve dynamic optimization problems where the data is obtained not once but one by one. The position of each particle swarm is updated recursively based on the continuous data and the historical knowledge. The experiment results indicate that RPSO-based radial basis function networks needs fewer radial basis functions and gives more accurate results than traditional PSO in solving dynamic problems. Then the proposed RPSO is suggested to cluster statistic counting vectors in order to enhance the performance of the data compression.

Introduction

The vector clustering plays an important role to the applications of information science, such as information compression. For a lot of entropy coding technologies, the probability distributions used to code is estimated by using the corresponding counting vectors. Sometimes, the number of the counting data is less to obtain good estimation. In this case, the clustering operation is executed to merge some counting vectors into one to increase the number of data so that the better estimation could be achieved. Actually, the Particle Swarm Optimization is one of efficient clustering algorithms. During the past decade, the Particle Swarm Optimization (PSO) is widely used in various regions of the optimizing application[1]. However, PSO is suitable for the offline optimization, but not good for solving the dynamic optimization problems. In the recursive estimation procedure, the data is not obtained by only once but is achieved sequently. For instance, in the network optimization, the optimal parameter of the network is obtained adaptively with the change of the current states of the network. The recursive Particle Swarm Optimization (RPSO), which could achieve online optimization by tracking the observed information and the data obtained, takes the result that the range of each parameter could be known in the application.

In PSO procedure, each particle modify its location by trending to the location corresponding to its past optimal result and the location related to the optimal results of whole swarm. However, this mechanism could not utilize the solution space of the prior knowledge fully[2]. Meanwhile, it is impossible to obtain the data by only once in the linear optimization application. Furthermore, only these nearest data could perform importance for the optimization[3-4]. In order to resolve the online recursive optimization, the recursive particle swarm optimization (R-PSO) is proposed in this paper. The difference between PSO and R-PSO is that R-PSO search the solution space by using the data obtained successively. Then the prior knowledge of the solution space is introduced into the optimizing procedure. Meanwhile, the locations of these particles are modified with the fitness value of these particles and the scale of the solution space. After iterations, the optimal result could achieve adaptively. The experiments results indicate that the R-PSO proposed could obtain better solution space searching result than the result by PSO.

On the other hand, the vector clustering is influencing the development of the information science. As an example, the digital signal compression relies on the optimized cluster to obtain the best coding probability to drive the encoders. In those entropy based coding methods, the probability is estimated...
by using the corresponding counting vectors. If the number of the data which are counted is not sufficient, the probability estimated will become the uniform distribution that the maximum codelength will be obtained. One method to tackle this problem is to merge some counting vectors into one. In this time, the clustering algorithm will be used to implement this merging operation. In[8,9], the K-means and the genetic algorithm were suggested to merge those similar counting vectors into various classes respectively. Actually, the PSO is also used to implement the clustering operation. In this paper, we improve the PSO algorithm firstly, then the proposed algorithm is employed to merge counting vectors to enhance the compression efficiency.

Methods

Recursive Particle Swarm Optimization

In the execution procedure of PSO algorithm, the location and the speed of each particle are determined by its past optimal location and the optimal location of whole swarm. Suppose that there are \( M \) particles. Let \( X_i(k) (i=1,2,\ldots,M) \) denotes the location of \( i \)th particle after \( k \) iterations, \( V_i(k) (i=1,2,\ldots,M) \) denotes its current speed. \( p_i \) denotes the optimal location the \( i \)th particle searched after \( k \) iterations and \( p_g \) denotes the optimal location all particles searched in this time (Globle optimization ). Then the iteration of PSO obey the rule given by (1) and (2).

\[
V_i(k+1) = wV_i(k) + c_1 rand (P_i - X_i(k)) + c_2 rand (P_g - X_i(k))
\]  

(1)

And

\[
X_i(k+1) = X_i(k) + V_i(k+1)
\]

(2)

where \( c_1 \) and \( c_2 \) denote the accelerating parameters (learning parameters), \( rand \) denotes the random number with the range of \([0, 1]\). PSO actually use the knowledge of the solution space not any more. In order to utilize this important information of the optimal searching.

In R-PSO, the location of each particle is updated when the new data is obtained. The speed of the particle is changed according to the knowledge of the solution space. Based on this change, the optimization algorithm could achieve the important parameter adaptively. In next section, we use the recursive particle swarm optimization to optimize the design the radial basis function networks modeling system to improve the performance of the modeling system with the obtaining parameter adaptively.

Radial Basis Function Network

The structure of the radial basis function networks modeling system (RBFNMS) is given in figure 1.

![Figure 1. The structure of RBFNMS.](image)

From figure 1, it is easy to find that the RBFNMS holds multiple input with only one output. Meanwhile, RBFNMS is consisted of three layers which are input layer, implication layer and output layer respectively. In figure 1, the radial basis function is Gaussian function given in (3)
In (1), \( \| x - c_i \| \) denotes the distance (Euclidian distance normally) between the vector \( x \) and the center \( c_i \). \( \delta_i \) denotes the width of the \( i \)-th radial basis function. Then the output of each radial basis function could be calculated by (4)

\[
y = \frac{\sum_{i=1}^{n} HE_i(x) \times w_i}{\sum_{i=1}^{n} HE_i(x)}
\]

where \( n \) denotes the number of the radial basis function in the implication layer, \( w_i \) denotes the weight of the \( i \)-th radial basis function in the implication layer and the output layer. \( HE_i(x) \) denotes the output of the \( i \)-th implication layer which holds the matrix \( x = (x_1, x_2, ..., x_n) \) as the input. Then the radial basis function is determined by the parameters \( \{ w_i, c_i, \delta_i | i = 1, ..., n \} \).

Let \( e_i \) denotes the error of the \( i \)-th particle in RBFNMS, and it is calculated by (5)

\[
e_i = Y_{out} - Y_d
\]

where \( Y_{out} \) and \( Y_d \) denote the simulation value determined by the \( i \)-th particle in RBFNMS and the expected value of the result calculated by the simulation function respectively. When \( Y_{out} \) is calculated by (4), the size of the solution space (SSS) could be achieved by (6)

\[
SSS = S_{max} - S_{min}
\]

\( S_{max} \) denotes the maximum value of the solution space and \( S_{min} \) is the reverse. For RBFNMS, each matrix is consisted of \( n \) centers, \( n \) errors \( e_i \) and \( n \) connection weights \( w_i \). Then the location and the speed of the \( i \)-th particle could be modified by (7), (8) and (9)

\[
VS_i = SSS \times e_i \times or \times 0.001
\]

where \( VS_i \) denotes the matrix of the increment of the speed of \( i \)-th particle. The modification obey as:

\[
V_i(k+1) = (wV_i(k) + c_1rand(P_i - X_i(k)) + c_2rand(P_k - X_i(k))) \times VS_i^T
\]

And

\[
X_i(k+1) = X_i(k) + V_i(k+1)
\]

By this updating, the R-PSO could be used for the solution of the parameters of the radial basis function networks modeling system.

**Radial Basis Function Networks Modeling System Based on R-PSO**

Based on the discussions above, the algorithm proposed could be described by these steps:

**Step1:** Initializing the solution space. For RBFNMS, there are one maximum matrix \( S_{max} \) and one minimum matrix \( S_{min} \). Then initializing the matrix \( S \) .

**Step2:** Initializing the R-PSO, setting the objective function \( f \) , setting some important parameters including the rule \( c_1, c_2, w \) , the number of particles \( m \) , the speed vector and the location vector \( x \) .

**Step3:** Updating the RBFNMS. Using the new observed data to calculate the simulation value.

**Step4:** Evaluating the healthy value of each particle for the new input data, updating the optimal location of this particle and the global optimal location.

**Step5:** Updating the information of the location and speed of the particle by (7), (8) and (9).
Step 6: Goto step 3 until the number of iteration reach the given number.

**Compression Application**

In this section, the proposed PSO algorithm is used to merge some counting vectors. The arithmetic encoder is used to assign the codewords for image sources. For clustering, the objective should be determined firstly. Similar with [8], the adaptive complementary relative entropy (10) is used as the similarity measure.

\[
A_{P,Q} = \sum_{i=0}^{I-1} \left( \log \frac{n_i}{m_i} \right) \ast (p(x = i) - q(x = i))
\]

where \(A_{P,Q}\) denotes the adaptive divergence of two conditional probability distributions, which is equal to the sum of two relative entropy \(D(P(X|c), P(X|c_1))\) and \(D(P(X|c), P(X|c_2))\). Where \(n_i\) denotes the counts of the symbol with the value \(i\) in the conditional probability distribution \(p(x|c)\) and \(m_i\) denotes the counts of the symbols with the value \(i\) in the conditional probability distribution \(q(X|c)\) and \(p(x=i)\) denotes the conditional probability of the symbol \(x\) with the value \(i\). The \(A_{P,Q}\) satisfies the spatial symmetry.

When these counting vectors are merged into their corresponding classes respective, each new counting vectors merged is used to estimate the corresponding probability distribution to drive the arithmetic encoder. Actually, the adaptive codelength given in [10] could be as the testify measure for arithmetic. Namely, the optimized objective is set as: To minimize the adaptive codelength (11)

\[
L_m = \sum_{s=0}^{S-1} \log(s + l\delta) - \sum_{s=0}^{S-1} \sum_{i=0}^{I-1} \log(i + \delta)
\]

Meanwhile, the formula (11) is also the function of the proposed PSO here.

Based on this algorithm above, the performance of the arithmetic encoder can be improved. The corresponding results will be given in next section.

**Experiments and Results**

Firstly, we construct experiments to testify the performance of the proposed PSO algorithm.

We use some experiments to testify the algorithm proposed. In our experiments, there are three modeling functions used. Let \(f_1, f_2\) and \(f_3\) denotes the Valley model, Rastrigin model and Michalewicz model respectively. The expressions of these functions are listed in Table 1:

<table>
<thead>
<tr>
<th>function</th>
<th>expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>(f_1)</td>
<td>(f_1 = \sin(\pi x_1)\sin(\pi x_2))</td>
</tr>
<tr>
<td>(f_2)</td>
<td>(f_2 = \sum_{i=1}^{2} (x_i^2 - 10\cos(2\pi x_i) + 10))</td>
</tr>
<tr>
<td>(f_3)</td>
<td>(f_3 = \sum_{i=1}^{2} \sin(x_i) \left[\sin\left(\frac{ix_i}{\pi}\right)\right]^{20})</td>
</tr>
</tbody>
</table>

The parameters for these functions are also given in the Table 2:
Table 2. The parameters used in our algorithm.

<table>
<thead>
<tr>
<th>function</th>
<th>parameters</th>
<th>or</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_1 )</td>
<td>((-3.0, -3.0, -3.0, 0.5, 0.5, 0.5, -1.1, -1.1))</td>
<td>2</td>
</tr>
<tr>
<td>( f_2 )</td>
<td>((-6.6, -6.6, -6.6, 0.5, 0.5, 0.5, 0.0, 0.0))</td>
<td>100</td>
</tr>
<tr>
<td>( f_3 )</td>
<td>((0.0, 0.0, 0.0, 0.5, 0.5, 0.5, -1.2, -1.2, -1.2))</td>
<td>2</td>
</tr>
</tbody>
</table>

In order to testify the dynamic performance of our algorithms, in our experiments, the data is inputted gradually. In figure 2, the corresponding test results are given. For comparison, the result by the algorithm AUTO-PSO+HPSO-RLS[7] is given in the image (d).

![Figure 2](image)

Figure 2. The comparison of the experiments results. (a): the Rastrigin function is used. (b): valley function is used (c): the Michalewicz function is used. (d): the result by AUTO-PSO+HPSO-RLS.

From Figure 2, RBFNMS based on R-PSO could achieve the degree of the accuracy better and better with the data input. Even the AUTO-PSO+HPSO-RLS perform better for the degree of the smooth, but the algorithm is not dynamic, which limit the algorithm used in the dynamic optimization application. Meanwhile, the degree of the accuracy by AUTO-PSO+HPSO-RLS is not satisfied, especially in the region of the edge. After comparison, RBFNMS based on R-PSO could perform better than AUTO-PSO+HPSO-RLS.

Then the proposed PSO algorithm is used to merge the counting vectors.

We quantize the pixel-value of 6 gray scale images (256 by 256) into 8 levels for lower simulation complexity. 3 images among the 6 quantized images are used for training the 4-order modeling context. The conditional probability distributions \( \Pr(x_1 | x_{-1}, x_{-2}, x_{-3}) \) are estimated, where \( x_{-1}, x_{-2}, x_{-3} \) have 512 possible combinations. The reminder 3 images are used as the source images for coding in our simulations. Meanwhile, in order to simplify the presentation, some abbreviations are used: NC denotes the number of the clusters and AL denotes the adaptive code length.

Table 3. The comparison of K-means and the proposed algorithm.

<table>
<thead>
<tr>
<th>Image</th>
<th>KMCQ</th>
<th>ALGORITHM BY [4]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NC</td>
<td>AL(bit)</td>
</tr>
<tr>
<td>barb</td>
<td>230</td>
<td>65213</td>
</tr>
<tr>
<td></td>
<td>231</td>
<td>65230</td>
</tr>
<tr>
<td></td>
<td>230</td>
<td>65199</td>
</tr>
</tbody>
</table>

In experiment 1, the K-means and the proposed algorithm are used to cluster the 512 counting vectors respectively and the corresponding adaptive codelength are calculated. For K-means, the
number of the classes should be given firstly. In order to obtain the best results, for each number of classes, we execute the K-means once and the best result is choice to list in Table 3. The image barb is used as the test image.

It is obviously that the proposed algorithm can achieve better result than the algorithm based on K-means slightly. But for lossless compression, the 0.2% improvement of the compression rate is also difficult. Actually, the improvement here comes from the PSO. Meanwhile, the number of classes will be stable under the proposed PSO algorithm.

Then, the proposed algorithm is suggested to compress other two images. In order to enhance the compression efficiency, the number of counting vectors increase to 4096, which is due to the theory that conditioning reduce the entropy. For comparison, the results from K-means based algorithm are also listed in Table 4:

<table>
<thead>
<tr>
<th>Image</th>
<th>KMCQ AL(bit)</th>
<th>ALGORITHM BY [4] NC AL(bit) Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>plane</td>
<td>713 63728 0.972</td>
<td>1011 63349 0.967</td>
</tr>
<tr>
<td>boat</td>
<td>726 63224 0.964</td>
<td>1213 62989 0.961</td>
</tr>
</tbody>
</table>

Apparently, with the improved PSO, the probability distribution estimated from the counting vectors merged can achieve better results than the results from the algorithms with K-means. Almost 0.003bit for each pixel is reduced.

Above all, the proposed improved PSD algorithm can obtain better clustering results and can lead to better compression performance. The design objective is achieved.

Conclusion

The Recursive Particle Swarm Optimization (RPSO) is proposed to solve dynamic optimization problems. The RPSO-based radial basis function networks needs fewer radial basis functions and gives more accurate results than traditional PSO in solving dynamic problems. Then RPSO is suggested to compress the image sources. The results indicate that the proposed PSO algorithm can achieve better compression performance than K-means.

References


