Credit evaluation based on gene expression programming and clonal selection

Wei-hong Wang a* Yan-ye Du a Qu Li a, b and Zhao-lin Fang a

a College of Computer Science, Zhejiang University of Technology, Hangzhou310023, China
b College of Computer Science, Zhejiang University, Hangzhou310023, China

Abstract Commercial banks’ credit evaluation and risk management have been one of the main issues in financial fields. As Gene Expression Programming (GEP) has a powerful search capability, a new algorithm for credit evaluation called Clonal-GEP-DDAG is proposed. The algorithm is designed on the classification algorithm DDAG, while using GEP for encoding and clonal selection for evolutionary mechanism. The algorithm has got good experimental results on the Australian credit data and German credit data, which show the effectiveness and efficiency of this algorithm on credit evaluation problems.

© 2011 Published by Elsevier Ltd. Selection and/or peer-review under responsibility of [CEIS 2011]

Keywords: Credit evaluation, Gene expression evaluation(GEP), Clonal selection, classification

1. Introduction

Credit evaluation is essentially a pattern recognition classification algorithm, that is, lenders will be divided into “good” customers and “bad” customers. According to the characteristics of credit evaluation problems, we apply Platt’s Decision Directed Acyclic Graph (DDAG)[1] classification algorithm to credit evaluation, and propose a new credit evaluation algorithm called Clonal-GEP-DDAG based on Gene Expression Programming (GEP) and clonal selection evolutionary mechanism.

2. Related work

2.1 DDAG

A Directed Acyclic Graph (DAG) is a graph whose edges have an orientation and no cycles. A Rooted DAG has a unique node such that it is the only node which has no arcs pointing into it. A Rooted Binary DAG has nodes which have either 0 or 2 arcs leaving them. We use Rooted Binary DAGs in order to define a class of functions to be used in classification tasks. The class of functions computed by Rooted Binary DAGs is formally defined as follows[1].

Definition 1: Decision DAGs (DDAGs). Given a space X and a set of Boolean functions \( F = \{ f : X \rightarrow \{0,1\} \} \), the class \( DDAG(F) \) of Decision DAGs on N classes over F are functions which can...
be implemented using a rooted binary DAG with \( N \) leaves labeled by the classes where each of the \( K = N(N - 1)/2 \) internal nodes is labeled with an element of \( F \). The nodes are arranged in a triangle with the single root node at the top, two nodes in the second layer and so on until the final layer of \( N \) leaves. The \( i-th \) node in layer \( j < N \) is connected to the \( i-th \) and \( (i+1)-st \) node in the \( (j+1)-st \) layer (see [1]).

To evaluate a particular DDAG \( G \) on input \( x \in X \), starting at the root node, the binary function at a node is evaluated. The node is then exited via the left edge, if the binary function is zero; or the right edge, if the binary function is one. The next node’s binary function is then evaluated. The value of the decision function \( D(x) \) is the value associated with the final leaf node (see Fig. 1).

![DDAG for finding the best class out of four classes][1]

2.2 GEP

Gene Expression Programming (GEP)[2] is first created by the Portuguese scholar Candida Ferreira, based on the Genetic Algorithm (GA) and the Genetic Programming (GP). A GEP gene is the basic unit of a GEP genome and consists of head and tail parts. The gene is then mapped into an Expression Tree (ET) by following a width-first fashion. And the ET is easy to be converted into a mathematical expression[3].

For example, let \( F=\{Q,+, -, *, /\} \) be the function symbol set, and \( T=\{a, b\} \) be the terminal symbol set, where \( Q \) is the square root function. A GEP gene with head length of 10 could be as follows:

\[
012345678901234567890\\n*_{b^a-aQ_{ab}+babbabaasaba}
\]

![Translation of a GEP gene into an ET][2]

By translating the ET, it’s easy to find that the corresponding mathematical expression is \( b * (a + (a - \sqrt{a})) \).

3. The new credit evaluation algorithm Clonal-GEP-DDAG based on GEP and clonal selection

3.1 Antibody-Antigen binding

The binding between antigens and antibodies is depicted in Fig. 3. Each antibody is a GEP
encoded rule, which translates to an expression tree. Each antigen represents a single class and is a collection of data records belongs to this class. The binding between antibodies and antigens depends on how well an antibody classifies the class instances of an antigen. The values on the arrows connecting antibodies and antigens denote the affinity of the two entities[4].

3.2 Flow of Clonal-GEP-DDAG algorithm

(1) Initialization
Randomly generate initial individuals to form the initial antibody population \( A \), which is composed of memory pool \( A_m \) and remaining pool \( M \), and calculate their affinities.

(2) Antibody selection & proliferation/cloning
The best \( n_b \) antibodies in terms of their affinities are selected and form the set \( B \). Each antibody of the set \( B \) is cloned according to its affinity. Antibodies with higher affinities produce more clones. This set of clones is called \( c \). The clone rule is as follows: in order to control the proliferation of the best antibodies, it first sorts the set \( B \) of the best antibodies in descending order, and then applies the formula

\[
 n_i = \text{round}\left( \frac{B \cdot n_b}{i} \right), \quad 1 \leq i \leq n_b \quad (2)
\]

to compute the number of clones that each antibody will produce. In this formula, \text{round}(.) is the rounding function, \( B \) is a constant, called clone factor, \( i \) is the rank of each selected antibody in the ordered set \( B \) [4].

(3) Hyper-mutation
Each clone \( c_j \) of set \( c \) is hyper-mutated[4]. The mutated clones form the set \( c_m \). Hyper-mutation is controlled through the exponential function

\[
 h(s) = h_{\text{max}} e^{-\rho \cdot s}, \quad h_{\text{max}} \leq 1 \quad (3)
\]

Where \( h_{\text{max}} \) is the mutation rate, \( \rho \) is a decay factor, and \( s \) is the affinity normalized in the interval \([0,1]\).

(4) Clone selection & memory update
The best \( n_m \) mutated clones are selected according to their affinities to update the memory pool.

(5) Remaining pool refresh
At this step, the remaining pool is refreshed in order to preserve its diversity. The \( n_d \) worst cells in terms of their affinities are replaced by completely new one, which are randomly generated individuals.

4. Experiments and analysis

4.1 Settings of the experimental parameters
We use the Australian credit data and German credit as two experimental data sets, which can be downloaded from [5]. For the German credit data, it has 1000 credit samples, each sample contains 20 attribute variables and one class variable, which are depicted in Table 1.

Table 1. Customer attributes of the German credit data

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute1 Status of existing checking account</td>
<td>A11: ... &lt; 0 DM; A12: 0 &lt;= ... &lt; 200 DM; A13: ... &gt;= 200 DM / salary assignments for at least 1 year; A14: no checking account</td>
</tr>
<tr>
<td>Attribute2 Duration in month</td>
<td>numerical</td>
</tr>
<tr>
<td>Attribute3 Credit history</td>
<td>A30: no credits taken/ all credits paid back duly; A31: all credits at this bank paid back duly; A32: existing credits paid back duly till now; A33: delay in paying off in the past; A34: critical account/ other credits existing (not at this bank)</td>
</tr>
<tr>
<td>Attribute4 Purpose</td>
<td>A40: car (new); A41: car (used); A42: furniture/equipment; A43: radio/television; A44: domestic appliances; A45: repairs ; A46: education; A47: (vacation - does not exist?) ; A48: retraining; A49: business; A410: others</td>
</tr>
<tr>
<td>Attribute5 Credit amount</td>
<td>numerical</td>
</tr>
<tr>
<td>Attribute6 Savings account/bonds</td>
<td>A61: ... &lt; 100 DM; A62: 100 &lt;= ... &lt; 500 DM; A63 : 500 &lt;= ... &lt; 1000 DM; A64: .. &gt;= 1000 DM; A65 : unknown/ no savings account</td>
</tr>
<tr>
<td>Attribute7 Present employment since</td>
<td>A71: unemployed; A72: ... &lt; 1 year; A73: 1 &lt;= ... &lt; 4 years; A74: 4 &lt;= ... &lt; 7 years; A75:... &gt;= 7 years</td>
</tr>
<tr>
<td>Attribute8 Installment rate in percentage of disposable income</td>
<td>numerical</td>
</tr>
<tr>
<td>Attribute10 Other debtors / guarantors</td>
<td>A101: none; A102: co-applicant; A103: guarantor</td>
</tr>
<tr>
<td>Attribute11 Present residence since</td>
<td>numerical</td>
</tr>
<tr>
<td>Attribute12 Property</td>
<td>A121: real estate; A122: if not A121: building society savings agreement/ life insurance; A123: if not A121/A122 : car or other, not in attribute 6; A124: unknown / no property</td>
</tr>
<tr>
<td>Attribute13 Age in years</td>
<td>numerical</td>
</tr>
<tr>
<td>Attribute14 Other installment plans</td>
<td>A141: bank; A142: stores; A143: none</td>
</tr>
<tr>
<td>Attribute15 Housing</td>
<td>A151: rent ; A152: own; A153: for free</td>
</tr>
<tr>
<td>Attribute16 Number of existing credits at this bank</td>
<td>numerical</td>
</tr>
<tr>
<td>Attribute17 Job</td>
<td>A171: unemployed/ unskilled - non-resident; A172: unskilled – resident; A173: skilled employee / official ; A174: management/ self-employed/highly qualified employee/ officer</td>
</tr>
<tr>
<td>Attribute18 Number of people being liable to provide maintenance for</td>
<td>numerical</td>
</tr>
<tr>
<td>Attribute19 Telephone</td>
<td>A191: none; A192: yes, registered under the customers name</td>
</tr>
<tr>
<td>Attribute20 foreign worker</td>
<td>A201: yes; A202: no</td>
</tr>
</tbody>
</table>
4.2 Analysis of the experimental results

K-fold cross validation is a commonly used technology to evaluate the classification accuracy, and it is partitioned based on random sampling of the given data. We do 5 times 5-fold cross validation on these two data sets. For example, for the German credit data, it has a total of 1000 sample data, and we divide it into 5 disjoint subsets, each subset of 200 data, then, compose the training set and test set by 4:1 of these 5 subsets to do 5-fold cross validation. The experimental results are depicted in Table 3.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Average Accuracy (%)</th>
<th>Best Accuracy (%)</th>
<th>Average Time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Australian</td>
<td>84.03 ± 0.25</td>
<td>89.13 ± 0.15</td>
<td>2.55</td>
</tr>
<tr>
<td>German</td>
<td>71.56 ± 0.05</td>
<td>76.8 ± 0.33</td>
<td>3.79</td>
</tr>
</tbody>
</table>

The accuracy results are obtained from 5 times 5-fold cross validation in terms of their average accuracy and 95% confidence interval, which is proportional to the standard deviation. As can be seen from Table 3, the proposed new credit evaluation algorithm Clonal-GEP-DDAG get good experimental results both on the Australian credit data and German credit data.

5. Conclusion

Credit evaluation has great significance for commercial banks’ risk managements, however, traditional credit evaluation methods have some flaws. Compared with Genetic Algorithm (GA), GEP has shown more powerful search capabilities in a number of issues on the function optimization. To take full advantage of GEP, we proposed a new algorithm called Clonal-GEP-DDAG for credit evaluation based on GEP and clonal selection. We took experiments on the Australian credit data and German credit data, and got good results.

Acknowledgements

The work is supported by the National Natural Science Foundation under grant 60873033, Natural Science Foundation of Zhejiang Province under grant R1090569 and Project of Department of Science and Technology of Zhejiang Province under grant 2009C31108.

Reference