Credit Scoring Analysis Using B-Cell Algorithm and K-Nearest Neighbor Classifiers*

Cheng-An Li

College of Economics and Commerce
South China University of Technology, Guangzhou, 510006, P.R. China
chanli@scut.edu.cn

Abstract. This paper applies B-Cell algorithm (BCA) for credit scoring analysis problems. The proposed BCA-based method is combined with k-nearest neighbor (kNN) classifiers. In the algorithm, BCA is introduced to select the optimal feature subsets and kNNs are used to classify the investors in different groups representing different levels of credit in the classification phase. Experiments employing the benchmark data sets from UCI databases will be used to measure the performance of the algorithm. Its comparison with genetic algorithm, particle swarm optimization and ant colony optimization will be shown.

Keywords: Classification, Credit Scoring, K-Nearest Neighbor Classifiers, B-Cell Algorithm.

1 Introduction

Credit scoring is a major issue for financial institutions, for firms that grant credit to their customers, as well as for institutional and individual investors. One of the key decisions financial institutions have to make is to decide whether or not to grant a loan to a customer. The decision basically boils down to a binary classification problem so as to distinguish customers with low credit risk from customers with high credit risk [1], [2].

Selecting the right sets of features for classification is one of the most important problems in designing a good classifier [1]. The objective of feature selection is to search through the space of feature subsets to identify the optimal or near-optimal one with respect to a selected performance measure. In the literature, many feature selection algorithms have been proposed [2]. These algorithms can be classified into two categories, that is, filter methods that select variables by ranking them with correlation coefficients and wrapper methods that assess subsets of variables according to their usefulness to a given predictor.

Recently, randomized metaheuristic search algorithms, including simulated annealing, genetic algorithms [3], particle swarm optimizations (PSO) [4]and ant

* This paper is supported by the Guangdong Planning Office of Philosophy and Social Science (Grant No. GD10CGL11).

colony optimizations (ACO) [5] are of great interest because they often yield high accuracy and are much faster, and have been used to find the optimal feature subsets in many real-world cases, including credit scoring analysis. However, the complexity of the methods in implementation made them difficult to solve some real problems.

The B-Cell algorithm (BCA) [6] is one of artificial immune algorithms, which is built on the clonal selection principle and uses an alternative model called somatic contiguous hypermutation for the mutation probability in the affinity mutation process. It is very important that BCA is very simple and has a small number of parameters [7]. Thus, in this paper, we propose a novel approach to solve the feature subset selection problem using B-Cell algorithm. In the classification phase of the proposal algorithm, the nearest neighbor classification methods are used. The proposed BCA-based algorithm is tested using the data from UCI benchmark databases in order to classify the investors in different groups representing different levels of credit risk. In addition, the other algorithms are compared with the proposed algorithms in order to validate their efficiency.

This paper is organized as follows. Section 2 summarizes the principle of BCA briefly. Section 3 addresses the BCA based classification method. Section 4 shows its application and experimental results. Finally, Section 5 concludes the paper and discusses briefly.

2 B-Cell Algorithm

2.1 B-Cell Algorithm

The B-Cell algorithm (BCA) [6] is inspired by the clonal selection process. It implements non-deterministic iterated process of exploring multidimensional search space and works with a population of tentative solutions called B-cells.

An important feature of BCA is its use of a unique mutation operator, known as contiguous somatic hypermutation. Here, a contiguous region of the point representation is randomly selected and each position is mutated with a certain probability. The general principles of the BCA method are outlined as follows.

In BCA, each cell is a candidate solution that has an associated performance index that allows it to be compared with the other cells. Each cell, represented as $x_i = (x_{i1}, x_{i2}, \cdots x_{iN})$, is composed of the binary variable and decides on {1} or {0}, where N is dimension number. Computationally, the implementation of the algorithm is quite simple and could be performed using the following steps:

--First, generate an initially randomized population of individuals P over the search space.

--Second, evaluate each individual. Clone and place in clonal pool C.

--Third, for each clone, apply the contiguous somatic hypermutation operator.

--Fourth, evaluate each clone; update the individual if a clone has higher affinity than its parent B-cell.

Stopping Criterion:

BCA is usually terminated with a maximal number of iterations or the entire population cannot be improved further after a sufficiently large number of iterations.

2.2 The Contiguous Somatic Hypermutation Operators

The immune-inspired mutation operator is one of the main characteristic of the B-Cell algorithm. The operator is called the somatic contiguous hypermutaion operator [8]. It decides randomly about a contiguous region of the bit string and flips each bit within this region with a given probability $r \in [0,1]$. In this paper, the somatic contiguous hypermutaion which has wrapping around has been used in the BCA shown in the following.

Definition 1 Somatic Contiguous Hypermutation (CSM) Wrapping Around [9]). *CHM mutate $x \in \{0,1\}^n$, given a parameter $r \in [0,1]$.*

```
1. Select $p \in \{0, 1, \ldots, n-1\}$ uniformly at random.
2. Select $l \in \{0, 1, \ldots, n\}$ uniformly at random.
3. For $i=p$ to $l-1$ do
4. With probability $r$ set $x[(p+1) \bmod n]=1- x[(p+1) \bmod n]$.
```

2.3 B-Cell Algorithm for Classification

In this paper, we introduce BCA into classification, in particular, credit scoring problem. To the best of our knowledge, no related work about BCA for classification problems has been reported in the previous literatures. A new algorithm is proposed by utilizing the k-Nearest Neighbor Classifiers and BCA for credit scoring problems. In the proposed method, BCA is employed to find the optimal feature subset and an independent classifier to evaluate the quality of the subset.

3 The Proposed B -Cell Algorithm

3.1 The Method

In this paper, an algorithm for the solution to the feature selection problem based on the B-Cell Algorithm is presented. We apply B-Cell Algorithm to select optimal feature subsets. This algorithm is combined with k-Nearest Neighbor classifiers which are used to evaluate the quality of the selected feature subsets. A pseudo-code of the proposed BCA -based method is presented in Table 1.

Table 1. B-Cell Algorithm

```
Begin
 Randomly Initialize the population P
 Do until the maximum number of solutions has been reached:
   Select randomly a number of features to activate
Enddo
Selection of the maximum number of generations
While (number of iterations, or the termination criterion is not met)
   Classify given data based each B-cell in the population
   For each $x_i \in P$
     Evaluate $x_i$ with a fitness function $f(x)$
     Clone $x_i$, and place in clonal pool C
     Randomly select a clone $c \in C$, randomise the vector
     Each $c \in C$, apply the contiguous somatic hypermutation operator
     Evaluate each clone c with the fitness function $f(x)$
   If a clone has higher fitness than its parent $x_i$ then
      $x_i = c$
   endif
 next $x_i$
 Update the optimal cell with new population P
next generation until termination criterion
 Return the best cell ( best solution)
End
```

In each iteration, the algorithm records and updates the global optimal solution.

3.2 K-Nearest Neighbor Classifiers

In this paper, the classic k-Nearest Neighbor (*k*NN) methods are used for classification after a number of features were selected in each iteration. For each sample of the test set, the Euclidean Distance from each sample in the training set is calculated. The Euclidean Distance is calculated as follows:

$$D_{ij} = \sqrt{\sum_{l=1}^{d} | x_{il} - x_{jl} |^2} \tag{1}$$

where D_{ij} is the distance between the test sample $i=1,\ldots, M_{test}$ (M_{test} is the number of test samples) and the training sample $j=1,\ldots, M_{train}$ (M_{train} is the number of train samples), and $l=1,\ldots, d$ is the number of activated features in each iteration.

In the classic k-Nearest neighbor methods, every member among the k nearest neighbors has an equal weight in the vote. As described in Ref. [5], it is nature more weight to those members that are closer to the test sample. A method, Weighted k Nearest Neighbor (wkNN), is proposed [5]. In the wkNN method, the most distant neighbor from the test sample has the smallest weight while the nearest neighbor has the largest weight. The weight of th *i*th neighbor is set:

$$w_i = \frac{i}{\sum_{i=1}^{k} i} \tag{2}$$

3.3 Population

A state vector of all features is denoted as a component or cell. In feature selection problems, we represent the cell by binary bit strings of length N, where N is the total number of attributes. Every bit represents an attribute, the value '1' means the corresponding attribute is selected while '0' not selected. Each vector is an attribute subset.

A number of cells are used. Each cell begins from a random initialization in the feature vector and changes according to a mutation operator.

3.4 Fitness Function

To effectually evaluate the quality of the produced members of the population, the following fitness function is adopted in our method with the classification accuracy of good group and bad group and the number of selected features. Thus, for each cell, the classifiers are called and the produced number of selected features and overall classification accuracy are used to generate the fitness function.

Accuracy of the good group is:

$$AC_1 = \frac{T_1}{T_1 + F_1} \tag{3}$$

Accuracy of the bad group is:

$$AC_2 = \frac{T_2}{T_2 + F_2} \tag{4}$$

where T_1 and F_1 are correct and error classification for good group respectively, and T_2 and F_2 are correct and error classification for bad group respectively.

Hence, the fitness function can be defined as:

$$fitness = \alpha * AC_1 * AC_2 + \beta * (1/(5 + Fs)) \tag{5}$$

where α and β are constant factors used to weight each term in (5), $0 \le \alpha \le 1$ and $\alpha + \beta = 1$. Fs is denoted as the number of the selected features.

4 Experiments

This section describes the experimental setup used to test the performance of the proposed approach. We compare the proposed BCA-based algorithm with the GA-based, PSO-based, ACO-based algorithm in credit scoring by using two public credit card datasets (Australian, and German credit datasets) from the UCI KDD Archive [10].

4.1 Data Sets and Parameter Setting

The data are obtained from UCI Repository of Machine Learning Databases [10] illustrated in Table 2. The Australian credit data consists of 307 instances of creditworthy applicants and 383 instances where credit is not creditworthy. This dataset has a good mixture of attributes: continuous, nominal with small numbers of values, and nominal with larger numbers of values, along with missing attributes in 5% of the samples. To protect the confidentiality of data, the attributes names and values have been changed to meaningless symbolic data. The dataset are preprocessed by fitting suitable values into the missing values for the original data set. The German credit data are more unbalanced, and it consists of 700 instances of creditworthy applicants and 300 instances where credit should not be extended. This data set only consists of numeric attributes. In implementation, the input variables in each dataset were normalized independently in each dimension into the range [0, 1].

Table 2. The data sets from the UCI Repository

No	Names	# Instances	Nominal features	Numeric features	# Class
1	Australian	690	6	9	2
2	German	1000	0	24	2

These data have been classified into two classes: creditworthy, denoted as good group, and not creditworthy, denoted bad group. Hence, the credit scoring task will be to discriminate between these two groups of data. The algorithm was implemented in Matlab 7.10 and Intel at 3.3 GHZ.

The parameters for all algorithms are shown in Table 3.

Table 3. Parameter setting for differen algorithms

Methods	Parameter setting
BCA	Generation: 50, B cells: 5, Clone cells: 5, mutation probability: 0.1
GA	Generation: 100, population size: 50, crossover: 0.9, Mutation: 0.01
PSO	Generation: 50, particles: 25, $w_{\max}$ =0.9, $w_{\min}$ =0.1, c1=2.0, c2=2.0, $V_{\max}$ =6
ACO	*ants*=25, *tmax*=50, *r*=20, r_1=5, *q*=0.5, θ_a =0.25, θ_b =0.95 [5]

In addition, α and β in fitness function are set to 0.8 and 0.2 respectively.

4.2 Evaluation Functions

Seven measures are used to validate the proposed method: number of selected features, the Root Mean Squared Error (RMSE), accuracy of good group, accuracy of bad group, average accuracy, overall accuracy and computational time.

RMSE can be calculated from the formula:

$$RMSE = \sqrt{\sum_{i=1}^{M} | y_i - y_{ei} |^2 / M} \tag{6}$$

where M is the number of samples in the data set, y_{ei} is the classifier model output and y_i is the true class of the test sample *i*.

Accuracy of good group and bad group defined in (3) and (4) respectively.

The Average Classification Accuracy (ACA) is :

$$ACA = (AC_1 + AC_2)/2 . \tag{7}$$

The Overall Classification Accuracy (OCA) is:

$$OCA = \frac{T}{M}100 . \tag{8}$$

In (8), T is the number of the samples classified correctly. In fact, $T=T_1+T_2$.

In the experiments, the algorithms used the datasets with 10-fold cross validation. The mean values of number of selected features, RMSE and classification accuracy of 10-fold cross validation and computational time for three data sets are shown in the following tables Table 4 and 5.

4.3 Experimental Results

We implemented BCA and compared experiments with the other randomized search algorithms GA, PSO and ACO. We used 1NN and wkNN as base classifiers in each of the iterations. The results reported consist of the average number of selected features, RMSE, accuracy of good group, accuracy of bad group, average accuracy, overall accuracy and computational time. The classification results are presented in Table 4 and 5.

From Table 4 and 5, in the two base classifiers 1NN and wkNN, experimental results show that wkNN outperform 1NN except computational time for BCA, GA, PSO and ACO in general. Hence, we focused on the result analysis about wkNN as the base classifier in the following.

Table 4. Results of the proposed algorithm and the compared algorithms on Australian dataset

Methods	Average number of features	RMSE	Accuracy for good group (%)	Accuracy for bad group (%)	Average Accuracy (%)	Overall Accuracy (%)	Computational time(s)
BCA-1NN	6.4	0.2849	92.18	91.64	91.91	91.88	40.4199
BCA-wkNN	5.8	*0.2554*	*92.83*	93.99	*93.41*	*93.48*	186.9204
GA-1NN	7.4	0.2998	90.23	91.64	90.94	91.01	97.7806
GA-wkNN	7.2	0.2719	92.18	92.95	92.57	92.61	289.5219
PSO-1NN	6.9	0.2745	92.18	92.69	92.44	92.46	44.3350
PSO-wkNN	*5.6*	0.2638	92.83	93.21	93.02	93.04	147.0794
ACO-1NN	7.9	0.2973	89.58	92.43	91.00	91.16	*24.4885*
ACO-wkNN	7.4	0.2745	90.23	*94.20*	92.24	92.46	72.6886

It can be observed that RMSE, average accuracy, and overall accuracy of the BCA-wkNN (0.2554, 93.41% and 93.48%) are superior to those of GA-wkNN, PSO-wkNN and ACO-wkNN from Table4. Accuracy of good group of the BCA-wkNN (92.83%) is equivalent to that of PSO -wkNN, but better than GA-wkNN and ACO-wkNN. Accuracy of bad group of the BCA-wkNN is slightly inferior to that of ACO-wkNN, but better than GA-wkNN and PSO-wkNN. Average number of selected features of the proposed BCA-wkNN is slightly larger that of PSO -wkNN, but smaller than that of GA-wkNN and ACO-wkNN. It is also noticed that, for computational time, the proposed BCA -wkNN is superior to GA-wkNN, but slightly inferior to PSO-wkNN and ACO -wkNN.

Table 5. Results of the proposed algorithm and the compared algorithms on German dataset

Methods	Average number of features	RMSE	Accuracy for good group (%)	Accuracy for bad group (%)	Average Accuracy (%)	Overall Accuracy (%)	Computational time(s)
BCA-1NN	11.6	0.4266	85.71	*72.67*	79.19	81.80	*102.7111*
BCA-wkNN	13.1	*0.4123*	87.43	*72.67*	*80.05*	*83.00*	378.3960
GA-1NN	12.6	0.4658	82.43	68.67	75.55	78.30	260.6146
GA-wkNN	11.6	0.4393	86.86	66.33	76.60	80.70	657.7960
PSO-1NN	11.2	0.4722	80.71	70.67	75.69	77.70	139.4050
PSO-wkNN	*10.8*	0.4494	84.14	69.67	76.90	79.80	300.0934
ACO-1NN	12.6	0.4324	88.86	63.67	76.26	81.30	126.8113
ACO-wkNN	11.8	0.4159	*91.14*	63.00	77.07	82.70	334.0931

From Table5, we can observe that RMSE, accuracy of bad group, average accuracy, and overall accuracy of the proposed BCA-wkNN (0.4123, 72.67%, 80.05% and 83%) is superior to those of GA-wkNN, PSO-wkNN and ACO-wkNN. Accuracy of good group of the BCA-wkNN is inferior to that of ACO -wkNN, but better than GA-wkNN and PSO-wkNN. For average number of selected features, the proposed BCA-wkNN performs slightly worse than GA-wkNN, PSO-wkNN and ACO-wkNN. Of course, for computational time, the proposed BCA-wkNN is superior to GA-wkNN, but slightly inferior to PSO-wkNN and ACO-wkNN.

In summary, the experimental results demonstrate that the proposed BCA-based approach is effective and has better classification performance than other optimization techniques such as GA, PSO and ACO for the credit scoring problems.

It is noted that the Australian and German datasets are most widely selected to test the performance of algorithms for financial crises [1]. Hence, in this paper, we also use these datasets to validate the proposed method so that future researchers can make direct and fair comparisons.

5 Conclusions and Discussion

This paper discussed the application of BCA for credit scoring. A novel approach has been proposed to solve the feature subset selection problem using B-Cell algorithm. In the approach, the nearest neighbor classification methods are used in the classification phase. The experimental results show that the proposed BCA-based approach outperforms the compared GA, PSO, and ACO-based approaches in the classification performance for credit scoring problems. In the future work, we shall discuss how to apply the proposed BCA algorithm to other applications such as optimal risk portfolios. In addition, we will use the proposed approach in some specific real-world problems.

References

1. Lin, W.-Y., Hu, Y.-H., Tsai, C.-F.: Machine Learning in Financial Crisis Prediction: A Survey. IEEE Transactions on Systems, Man and Cybernetics-Part C: Application and Reviews (2011), doi:10.1109/TSMCC.2011.2170420
2. Crook, J.N., Edelman, D.B., Thomas, L.C.: Recent developments in consumer credit risk assessment. European Journal of Operational Research 183, 1447–1465 (2007)
3. Vafaie, H., Imam, I.F.: Feature selection methods: genetic algorithms vs. greedy-like search. In: Proceedings of International Conference on Fuzzy and Intelligent Control Systems (1994)
4. Yang, C.-S., Chuang, L.-Y., Ke, C.-H., Yang, C.-H.: Boolean Binary Particle Swarm Optimization for Feature Selection. In: IEEE Congress on Evolutionary Computation, pp. 2093–2098 (2008)
5. Murinakis, Y., Marinaki, M.: Applicaton of Ant Colony Optimization to Credit Risk Assessment. New Mathematics and Natural Computation 4(1), 107–122 (2008)
6. Kelsey, J., Timmis, J.: Immune inspired somatic contiguous hypermutation for function optimisation. In: Cantu-Paz, E., et al. (eds.) GECCO 2003. LNCS, vol. 2724, pp. 207–218. Springer, Heidelberg (2003)
7. Zarges, C.: Theoretical Foundations of Artificial Immune Systems, The dissertation of Technical University Dortmund, Dortmund Germany (2011)
8. Jansen, T., Oliveto, P.S., Zarges, C.: On the Analysis of the Immune-Inspired B-Cell Algorithm for the Vertex Cover Problem. In: Liò, P., Nicosia, G., Stibor, T. (eds.) ICARIS 2011. LNCS, vol. 6825, pp. 117–131. Springer, Heidelberg (2011)
9. Jansen, T., Zarges, C.: Analyzing different variants of immune inspired somatic contiguous hypermutations. Theoretical Computer Science 412(6), 517–533 (2011)
10. Murphy, P.M., Aha, D.W.: UCI Repository of machine learning databases, Department of Information and Computer Science. University of California, Irvine (2001)