Mining Case Bases for Action Recommendation

Qiang Yang and Hong Cheng
Department of Computer Science
Hong Kong University of Science and Technology
Clearwater Bay, Kowloon, Hong Kong, China
(qyang, csch)@cs.ust.hk

Abstract

Corporations and institutions are often interested in deriving marketing strategies from corporate data and providing informed advice for their customers or employees. For example, a financial institution may derive marketing strategies for turning their reluctant customers into active ones and a telecommunications company may plan actions to stop their valuable customers from leaving. In data mining terms, these advice and action plans are aimed at converting individuals from an undesirable class to a desirable one, or to help devising a direct-marketing plan in order to increase the profit for the institution. In this paper, we present an approach to use 'role models' for generating such advice and plans. These role models are typical cases that form a case base and can be used for customer advice generation. For each new customer seeking advice, a nearest-neighbor algorithm is used to find a cost-effective and highly probable plan for switching a customer to the most desirable role models. In this paper, we explore the tradeoff among time, space and quality of computation in this case-based reasoning framework. We demonstrate the effectiveness of the methods through empirical results.

Keywords: case base mining, AI contributions to data mining (case-based reasoning), actionable data mining, financial applications of data mining.

1. Introduction

Data mining has traditionally focused on studying how to build statistical models from large databases. These models can be used to classify a given customer into a most probable class. Using these models, managers in corporations and institutions can decide whether to accept or reject a customer into a certain class membership. In this work, we take one step further: we not only use statistical models to make decisions on new customers, we also produce advice for the failed customers in the form of actions or plans. For example, a cell-phone company may decide to reduce the monthly fee for a subgroup of its customers who are both highly valuable and likely to leave the company for its competitors. Likewise, in a university scenario, instead of rejecting a graduate-school applicant with only a "no" answer, it helps to suggest steps that might be taken by the applicant in the future to increase his/her chance of being admitted the next time around. Similarly, these plans can be used to give advice to the customers who fall short of a loan application and to corporate managers on the strategies in marketing campaigns.

As an example, consider a customer-loan database on customer information and past loan in Table 1. Suppose that we are interested in providing advice for Steve (the last row) who failed to apply for a bank loan. Obviously, there are many candidate actions that one can advise Steve to take in order to succeed in his next loan application. These actions are designed based on other successful applicants who serve as positive examples in the same database. In this example, we have two such positive cases: John and Mary. For Steve, we can advise him to find another job with a salary close to 80K and increase his car number from one to three; this will make him look more like John. Alternatively, we can advise Steve to take up a mortgage from the bank worth at least 300K. This will make Steve look more like Mary. In either situation, Steve might have a higher chance of succeeding than before, but the actions come with different costs. The prescribed actions for Steve are shown in Table 2.

This example also introduced a number of interesting aspects for the problem. First, for each advisee such as Steve, there are potentially many possible actions that we can provide. Each action comes with an inherent cost associated with it. For example, it may be more costly for Steve to buy two more cars. Thus, the action for buying cars may be so prohibitive that the advice should not be given to Steve. Second, not all attributes can be acted on; some attributes cannot be changed.

Table 1. An example customer database. The last attribute is the class attribute.

Customer	Salary	Cars	Mortgage	Loan Approved?
John	80K	3	None	Y
Mary	40K	1	300K	Y
…	…	…	…	…
Steve	40K	1	None	N

Table 2. Prescribed alternative actions for Steve.

Advice for Steve	Salary	Cars	Mortgage
Plan 1	40K→80K	1→3	
Plan 2			0→300K

These correspond to *non-actionable attributes*. For example, it would be impossible to change the salary of Steve to that of John by taking a simple action, or to change the gender of a person. This impossibility can be modeled through prohibited high cost measures. Third, some attributes may not be relevant to the problem at hand. For example, the *address* attribute (not shown in the table) may be such an irrelevant attribute. Finally, the actions are not guaranteed to succeed 100% of the time; some actions may have higher probability of success than others. Our task is to choose high-utility actions that increase the probability of success while reducing costs.

We formulate the above problem as a case-based reasoning problem [13], where the key issue is to look for actionable plans on a case-by-case basis. Our approach is to first identify typical positive cases to form a small and highly representative case base, and then use the case base as "role models" to formulate the marketing actions that adapt each incoming problem to its nearest neighbor in the case base. Our focus in this paper is to identify the case bases automatically; we leave the planning issues in our future work.

More specifically, we first classify the training data into two classes: the "good" or "positive" data set contains data that belong to customers who have already been accepted into the good class and the "bad" or "negative" set for those who have not. Given this labeled dataset, our second step is to perform an analysis on the positive data to find out a number of representative cases of customers that can be "role models" for the rest. Finally, the negative cases are converted to positive ones by formulating marketing actions. Figure 1 (a) and (b) illustrate two distributions of the positive and negative classes in a two-attribute dataset. We will study the effect of the data distributions on our case-mining algorithms.

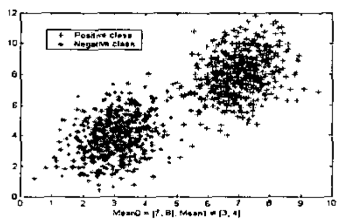

Figure 1(a). A demonstration of well-separated positive class and negative class distributions.

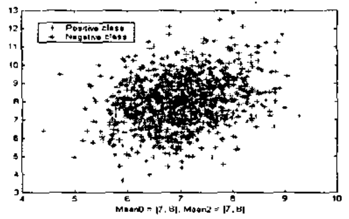

Figure 1(b). A demonstration of mixed positive class and negative class distributions.

Typically, real customer data are highly unbalanced and quite large. For example, the 1998 KDDCUP dataset [4] contains only 5% positive data and 95% negative data. Given such a dataset, a naïve application of classification such as decision tree would result in no useful information. In dealing with the unbalanced data problem, our case-mining algorithm will focus on the positive cases in the selection of role models, thus avoiding overwhelming the data distribution by negative cases. We consider three case-mining approaches. The most naïve one is to simply use the original database as the case base. While this model allows the creation of optimal actions from the past data, this approach is highly inefficient. The second approach constructs clusters from the database, and takes the centroids of the clusters as the potential cases for the case base. This approach can be very efficient, but the quality of the cases is still not optimized. This is because in creating role models for the positive class, it is more desirable to find cases that are "close" to the majority of the negative instances. These cases are often located near the "boundary" of the distributions of these classes. In order to find these boundary cases, our third approach is to apply a support-vector-machine (SVM) learning algorithm for extracting the support vectors as cases. These cases can give rise to more cost-effective plans.

In data mining area, researchers are interested in building statistical models of the database for classification and data analysis [2, 5, 17]. A typical statistical model partitions the test data into different classes according the trained model learned from the training data. Some recent work has specifically targeted the issue of marketing strategies in data mining [14, 8]. A major difference between our work and the majority of data mining work is that we do not stop at classification of the data into different classes; instead, we propose actions to switch customers to more desirable classes. Another related area is case based reasoning, in particular case-base maintenance [6, 18] and case-transformation [12, 13]. In this area, a new problem is solved by consulting a case base of past solutions. The new solution is formed by deriving the difference between a past solution and the new problem. In this area, the problem of identifying concise, high-quality case bases remains an open question.

2. Mining Case Bases

We formulate the problem formally. Given a database of customer records, we assume that each customer record is labeled as either a positive or negative class. Multiple class generalization is possible but will be considered separately in future work. In this section, we discuss three progressive more sophisticated solutions, each having its own computational advantages. The three methods are, respectively, instance-based, cluster-based and support-vector-based.

2.1. Instance-based Case-Base Mining

A first method to construct the case base is to use the entire positive population of data as the case base. Then for each negative instance we need to identify its nearest neighbor among the positive data instances. Thus the collection of all positive instances is taken as the case base. This method is purely memory based, and the learning is delayed until model application time. For this reason, it is also known as lazy evaluation or instance-based learning [1]. An advantage of this method is that for any given negative instance, it is guaranteed to find its nearest positive neighbor in terms of either the cost measure or utility. Thus, we also call this algorithm the *optimal* algorithm, because the customer-switching plan found will have optimal quality. However, the instance-based approach may suffer from computational inefficiency.

2.2. Cluster Centroid-based Case-Base Mining

A second idea is to compute clusters of positive instances and extract the centroids. The case-base mining algorithm thus constructed is described in Table 3. After feature selection, we divide the database into a positive class database DB+ and a negative class database DB-. The training database consists of the positive instances of the original database, whereas the testing data are the negative instances.

Table 3. Algorithm *Centroids-CBMine* (database DB, int *K*)

```
Steps  Begin
1          casebase= emptyset;
2          DB = FeatureSelection(DB);
3          Separate the DB into DB+ and DB-;
4          Clusters+ = ApplyKMedoids(DB+, K);
5          for each cluster in Clusters+, do
6                  C = findCentroid(cluster);
7                  Insert(C, casebase);
8          end for;
9        Return casebase;
         End
```

In the algorithm *Centroids-CBMine* in Table 3, the input database is DB. There are two classes in this database, where the positive class corresponds to population of desired cases and the negative class the unconverted cases.

Step 2 of the algorithm performs feature extraction by applying a feature filter to the database to remove all attributes that are considered low in information content or in classification power. In our implementation, subroutine *FeatureSelection* filters each attribute using an extended *OddLogRatio* algorithm [15] as the criterion for feature selection. *OddLogRatio* is designed for dataset with a highly unbalanced class distribution and asymmetric misclassification costs, which are exactly the characteristics of realistic dataset such as the KDDCUP'98 dataset. Using this criterion for feature selection on the KDDCUP'98 dataset, for example, we can reduce the number of attributes from 481 to 25. Subsequently, our case mining and planning activities is done on the selected subset of attributes. Using feature selection, we can avoid unnecessary changes on the "irrelevant" attributes, thus reducing overall customer switching costs.

Step 3 of the algorithm separates the training database into two partitions, a positive-class subset and a negative-class subset. Step 4 of the algorithm performs the *K*-medoids clustering on the positive-class sub-database [10]. Other good clustering algorithms can also be used here in place of *K*-medoids. For example, the density-based learning algorithm has been successfully applied to clustering on very large and complex data [9]. Step 6 of the algorithm finds centroids of the K clusters found in the previous step. These centroids are the bases of the case base constructed thus far, and are returned to the user. Finally, Step 9 returns the case base as the output.

Once the case base is built, it can then be applied to set of testing negative-class cases to see what the total cost would be for converting all the negative cases to positive ones. For each negative class case C1 in the test data set, a one-nearest neighbor algorithm is applied to the case base to find the most similar case C2. The difference between C1 and C2 are used to generate the switch plan.

Our notion of distance is based on a notion of cost of switching an attribute from one value to the next. For each attribute A and values v1 and v2 of A, there is a cost function: *cost(A, v1, v2)* which is a real value denoting the cost of changing from v1 to v2 on attribute A. In practice, such knowledge is not available in the dataset, but comes from domain expert. In our subsequent experiments, we set those values manually according to the semantic of attribute. For each nominal attribute, we have a cost matrix, each element of which denotes the cost changing from one value to the other. For each numeric attribute, we define a math function.

Consider the following example of the cost matrix. Suppose we have a nominal attribute which has three distinct values. We use 0, 1 and 2 to denote these three values. The cost matrix C is:

$$\begin{bmatrix} 0 & \infty & \infty \\ 200 & 0 & \infty \\ 500 & 300 & 0 \end{bmatrix}$$

In the cost matrix C, the value of C[1][0] is $200. That means the cost of changing from value 1 to value 0 is $200. All the elements in the diagonal are zero, which means that the cost of changing a value to itself is zero. The value of C[0][2] is infinite, which means that changing value 0 to value 2 is impossible or too prohibitive.

Finally, the cost of the model on an entire population of test data is the sum of all costs for all actions on each datum in the testing set. Assuming that the *j*th attribute for an *i*th customer is A_{ij}, Equation (1) shows the cost formula.

$$Cost = \sum_{i=1}^{|DB|} \sum_{j=1}^{l} \cos t(A_{ij}, v_1, v_2) \quad (1)$$

2.3. SVM-based Case-Base Mining

The centroid-based case-mining method extracts cases from the positive-class cluster centroids and takes into account only the positive class distribution. By considering the distribution of both the positive and negative class clusters, we can do better. This idea is as follows. Suppose C1 is the centroid of a positive-class cluster, while C2 is the centroid of a negative-class cluster. According to the *Centroids-CBMine* algorithm, C1 can be chosen as the corresponding positive case for the negative cases in the C2-cluster. However, if a positive case C3 that lies on the boundary of C1 cluster is nearer to the C2-cluster cases and we take C3 as the case instead of C1, the total cost of switching plans would be less. This is the intuition behind the SVM-based case mining.

The key issue then is to identify the positive cases on the *boundary* between the positive and negative cases, and select those cases as the final ones for the switching-plan generation. The cases along the boundary hyper-plane correspond to the support vectors found by an SVM classifier [16, 7]. These cases are the instances that are closest to the maximum margin hyper-plane in a hyperspace after an SVM system has discovered the classifier.

By exploiting the above idea, we have a different case-mining algorithm, *SVM-CBMine*. In the first step, we perform SVM learning on the database to locate the support vectors. Then we find the support vectors and insert them into the case base. This algorithm is illustrated in Table 4.

Compared with the *Centroids-CBMine* algorithm, the *SVM-CBMine* algorithm has several advantages. First, because the cases are the support vectors themselves, there is no need to specify the input parameter *K* as in the *Centroids-CBMine* algorithm; the parameter *K* is used to determine the number of clusters to be generated in *K*-medoids. Second, because the cases are themselves the boundary cases, they are naturally better examples for the entire negative-class members to switch to; the costs would be lower.

Table 4. Algorithm *SVM-CBMine* (Database *DB*, int *K*)

```
Steps  Begin
1          casebase = Emptyset;
2          Vectors = SVM(DB);
3          for each positive support vector C in
       Vectors do
4                  Insert(C, casebase);
5          end for
6          Return casebase;
       End
```

3. Switching Plan Generation

A straightforward plan-generation solution is to use a nearest neighbor approach. The plan used to advise a customer is one that is associated with the least cost. In our subsequent experiments, we call this the *MinCost* approach. While this approach is guaranteed to generate a cost effective plan, it is not guaranteed to generate a plan that will achieve its intended target all the time. In reality, the positive and negative cases are often distributed in a mixed manner. When executing a customer-switching plan, it is likely that the customer following the plan will land on a wrong target; it is wrong because it corresponds to an "unreliable" positive case whose neighborhood is dominated by negative instances, rendering the switching low probability of success. A more sensible method will consider not only the cost of switching, but also the probability of success of each switching. Together we have a notion of the utility of a plan, both in terms of success probability and costs [3, 11].

We can estimate the probability of success of switching to a certain target to be the probability density of positive instances around a target. More formally, let $p(+|t)$ be the probability density of an instance t, $cost(x, t)$ be the cost of switch from x to target case t, and $maxCost$ be the maximum value among the different costs of switching from x to every possible case y in the case base. The utility function we use for ranking cases in a case base is defined in Equation (2) below. The target case t with the maximum rank is chosen as the role model for switching-plan generation for customer x.

$$rank(x,t) = p(+|t) - \frac{Cost(x,t)}{maxCost} \qquad (2)$$

The algorithm for choosing the maximum utility plan for negative instances is called *MaxRank* algorithm. Once the role-model case is identified, the difference between the negative and positive cases can be taken and their differences are taken as recommended actions for the negative cases.

4. Experimental Results

In this section, we present the experimental results to test and compare the different algorithms we have proposed. Our experiments are aimed at finding out the tradeoff among the system execution time, which is the model-building time plus the model-application time on test cases, the size of the model (the number of cases) and the total cost and utility of switching plans for converting all negative examples into positive ones.

4.1. Artificial Dataset

Our first test uses an artificial dataset generated on a two-dimensional space (x, y), using a Gaussian distribution with different means and co-variance matrix for the positive (+) and the negative (–) classes. Our purpose is to demonstrate the effect of data distribution and case base size on the switching-plan quality and case mining efficiency. When the means of the two distributions are separated, we expect the class boundaries are easy to identify by the SVM-based method (see Figure 1 (a)). When the distributions are very close to each other, there will not be an easy-to-find boundary between the positive and negative classes; in this case the centroid-based method will perform better. The cost of switching a negative case to a positive one is defined as in this test as the Euclidean distance on the (x, y) plane. In the experiments with the artificial dataset, the cost matrix has a uniform cost value.

The mean for the positive class distribution is fixed to be *mean0*=(7, 8), with a co-variance matrix [(0.6, 0.3), (0.3, 1.8)]. For the negative class, the location of the mean moves from being far away from the *mean0* to being close to it to explore the effect of data distribution. In Artificial Data I, the mean for negative class *mean1*=(3, 4). The co-variance matrix for the negative class is defined as [(0.8, -0.5) (-0.5, 3.2)]. Figure 2 and Figure 3 show the test results comparing the case bases of different sizes and the resulting cost of switching plans. In Figure 2, we show the relative cost, as compared to the optimal cost, of switching all negative instances to positive cases for the corresponding case base size, using the minimal cost algorithm. In Figure 3, we show the same data for the MaxRank algorithm, which selects plans for each negative instance based on the utility formula (2). In both figures, SVM stands for the *SVM-CBMine* result. For example, SV=3 indicates that three support vectors were found to populate the case base. Parameter K indicates the number of clusters generated by K-medoids algorithm for the *Centroids-CBMine* system. "Optimal" indicates the cost for building and using the model using all positive examples in the original training data as the cases in the case base. As can be seen, the relative cost of switching from negative to positive cases decreases with the size of the case base. This is because as the case-base size increases, the choice in possible role models also increases.

To study the effect of different distribution of data, a second distribution is generated for the situation when the centers for the negative distribution are moved closer to that of the positive distributions (see Figure 1(b)). The result shown in Figures 4 and 5 – Artificial Data II with *mean2*=(7, 8), corresponds to the situation when there is no clear boundary between the two distributions.

As can be seen from the progression of the data distribution, as the two classes are distributed farther apart from each other (Figures 2, 3), the SVM-based method is a clear winner. This is because it uses far less time than the optimal method, and yet its total cost is nearly the same as that of the optimal method. As can be seen from the *K-medoids* based method, as the number K of clusters increases, the cost of switching plans also decreases. However, the time it takes to build and execute the model also increases with K (Figure 6). On the other hand, as the two distributions move close to each other such that there are no clear boundaries, as in the case of Figure 4, the SVM-based method selects nearly all the positive examples as cases in the case base, rendering it useless. Thus, its time expense is also very high (Figure 6). In this case, the *K-medoids-based* method is preferred.

Figures 7 and 8 show the success probability P(+|t) as a function of case base size. The *MaxRank* method for case retrieval obtains cases with higher costs, but much higher probability of success. For example, when K=100, there are 100 cases in the case base, the *MaxRank* method incurs a cost of 1583.5 as compared to the cost of 1243 for *MinCost,* but has a probability of success at 0.9 as compared to 0.3 for *MinCost*. It is also interesting to note that the as K increases from 10 to 100, the average probability of success decreases for *MinCost*, but not for *MaxRank* (Figure 7). This is because when the number of cases is large, it is more likely that the low probability cases will be selected as target cases for the plan generation, as compared with smaller case bases. However, as can also be seen from the figure, the *MaxRank* method for plan generation suffers less from this drawback, since in target-case selection, both the cost information and the probability information are taken into consideration.

4.2. KDDCUP Datasets

Besides the above results, we also carried out experiments on some more realistic datasets, since real world dataset often has much more unbalanced distribution than the artificial data. The attributes in real data have their own semantics. Therefore, in the subsequent experiments, we define the cost matrix and functions for each attribute in the dataset. In the tests, we still use the case-base mining algorithm to provide advice for each failed customer. However, if a customer consulted all the role models in the case base and cannot find a finite-cost switching plan, we consider the customer a failed one. Only those advices with finite costs are considered successful. We define a *SwitchingRate* measure to be the proportion of negative instances in test data for which there exists a finite-cost role-model case in the case base. When the case-base size is small, this *SwitchingRate* will be low as well, since the majority of negative cases cannot be switched to a positive one. The switching rate should be the highest for the optimal case when the entire set of positive instances serve as the case base.

Figures 9 – 12 show the test result of the algorithms on the KDDCUP'98 dataset. The training dataset consists of over 90,000 records for persons who were approached to make donations to a certain charity. The characteristic of this dataset is that it is highly unbalanced and has 481 attributes. For this dataset, we first performed the *OddLogRatio* feature-selection algorithm on the attributes. This resulted in a total of 25 remaining attributes, for which we constructed a cost matrix. The test result on this dataset is shown in Figures 9 to 12.

For the KDDCUP'98 data, the SVM based method returned over one half of the entire positive instances as the case base (SV=2645). Thus it does not really save any time in the case base mining. In contrast, the *Centroid*-based method performed very well, resulting in decreasing costs when K increases. As can be seen from Figure 11, when the number of cases reaches 100, the success rate for switching the negative cases reaches nearly 80%. Thus, the cases selected are quite representative of the positive instances, and the probability of success for the case base is comparable to that of the optimal case base.

5. Conclusions and Future Work

In this paper, we described solutions for mining case bases for customer action recommendation. The central issue of the problem lies in the discovery of high-quality case bases from a large data set, a case-mining problem. We proposed two solutions for the problem. For the data distribution where the two classes are clearly separated, the SVM-CBMine algorithm, which is an SVM-based method, should be used. When the data distributions are not separated well by a boundary, the cluster-centroids based method is recommended. Furthermore, we compared the solutions where plan generation is done based on cost alone and the solution where the probability of success is also taken into account. It was shown that the solution with utility consideration is superior. In addition, the centroid-based method is shown to scale much better than the SVM-based method, demonstrating a quality-speed tradeoff.

In the future, we will continue to explore the planning aspect of the problem, and apply the approach to more business databases.

References

[1] Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning, 6,* 37-66.

[2] R. Agrawal and R. Srikant. 1994. *Fast algorithm for mining association rules.* Proceedings of the Twentieth International Conference on Very Large Databases. pp 487-499

[3] F. Bacchus and A. Grove. 1995. *Graphical Models for Preference and Utility,* Proceedings of the Uncertainties in AI.

[4] C. L. Blake, C.J. Merz. 1998. *UCI Repository of machine learning databases* Irvine, CA: University of California, Department of Information and Computer Science. http://www.ics.uci.edu/~mlearn/MLRepository.html

[5] P. S. Bradley and U. M. Fayyad. 1998. *Refining initial points for k-means clustering.* In Proceedings of the Fifteenth International Conference on Machine Learning (ICML '98), pages 91--99, San Francisco, CA. Morgan Kaufmann Publishers.

[6] *Computational Intelligence Journal, Special Issue on Case-base Maintenance.* 2001. Blackwell Publishers, Boston MA UK. Vol. 17, No. 2, May 2001. Editors: D. Leake, B. Smyth, D. Wilson and Q. Yang.

[7] G. C. Cowley. 2000. *MATLAB Support Vector Machine Toolbox. v0.54B* University of East Anglia, School of Information Systems, Norwich, Norfolk, U.K. NR4 7TJ, 2000. http://theoval.sys.uea.ac.uk/~gcc/svm/toolbox

[8] P. Domingos and M. Richardson. 2001. *Mining the Network Value of Customers.* Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. August 2001. ACM. N.Y. N.Y. USA

[9] M. Ester, H.P. Kriegal, J. Sander and X. Xu (1996). A Denity-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In KDD 96 – Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining. 1996

[10] L. Kaufman and P.J. Rousseeuw. 1990. *Finding Groups in Data: An Introduction to Cluster Analysis.* New York: John Wiley & Sons.

[11] R. L. Keeney and H. Raiffa. 1976. Decisions with Multiple Objectives: Preferences and Value Trade-offs, Wiley, New York.

[12] J.L. Kolodner. 1993. *Case-based Reasoning.* Morgan Kauffman Publishers. San Mateo, CA, USA

[13] D. Leake. 1996. *Case-based Reasoning -- Experiences,* Lessons and Future Directions. AAAI Press/ The MIT Press.

[14] C. X. Ling and C. Li. 1998. *Data mining for direct marketing: Problems and solutions.* In Proceedings 4th International Conference on Knowledge Discovery in Databases (KDD-98), New York.

[15] Mlademnic, D., Grobelnik, M. 1999 *Feature selection for unbalanced class distribution and Naïve Bayes.* Machine Learning: Proceedings of the Sixteenth International Conference, pp.258-267.

[16] J. C. Platt. 1999. *Fast training of support vector machines using sequential minimal optimization,* in Advances in Kernel Methods - Support Vector Learning, (Eds) B. Scholkopf, C. Burges, and A. J. Smola, MIT Press, Cambridge, Massachusetts, chapter 12, pp 185-208.

[17] J. R. Quinlan. 1993. C4.5: *Programs for Machine Learning.* Morgan Kaufmann Publishers, Inc., San Mateo, CA.

[18] B. Smyth and M. T. Keane. 1995. *Remembering to forget: A competence--preserving deletion policy for case--based reasoning systems.* In Proceedings of the 14th International Joint Conference on Artificial Intelligence, pp 377—382.

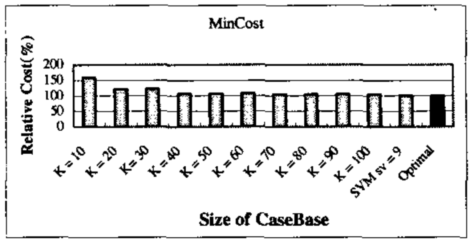

Figure 2. Relative cost vs. size of CB in *MinCost.* (Artificial data I, Mean 1 = [3, 4].)

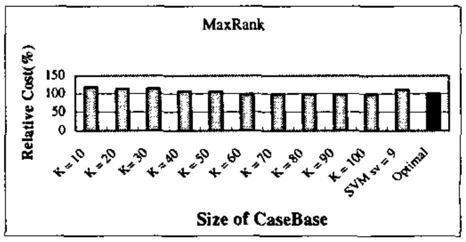

Figure 3. Relative cost vs. size of CB in MaxRank. (Artificial data I, Mean 1 = [3, 4].)

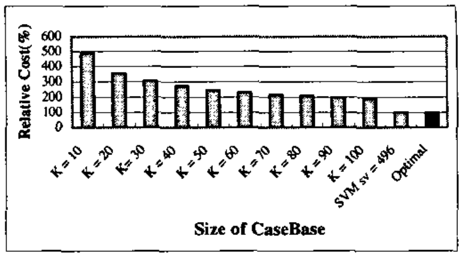

Figure 4. Relative cost vs. size of CB in MinCost. (Artificial data II, Mean 2 = [7, 8].)

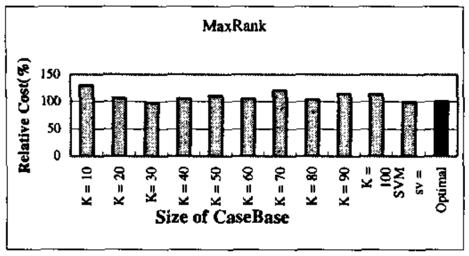

Figure 5. Relative cost vs. size of CB in MaxRank. (Artificial data II, Mean 2 = [7, 8].)

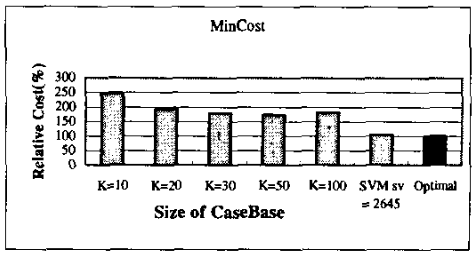

Figure 9 Relative cost vs. size of CB in MinCost. (KDD Cup98 data.)

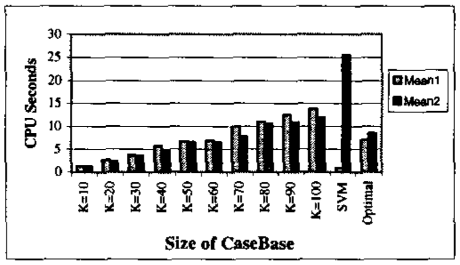

Figure 6. Comparison of CPU time for case-base mining for two different distributions Mean1 and Mean2 in the artificial dataset tests

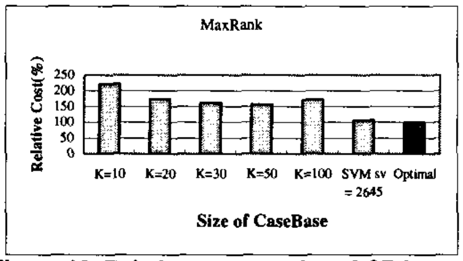

Figure 10. Relative cost vs. size of CB in MaxRank. (KDD Cup98 data.)

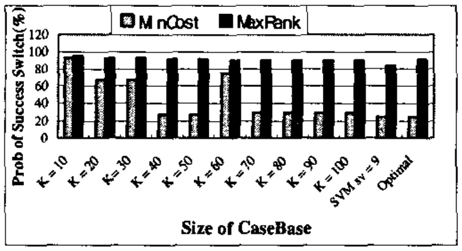

Figure 7. Success Prob P(+lt) vs. size of the CB. (Artificial data I, Mean 1 = [3, 4].)

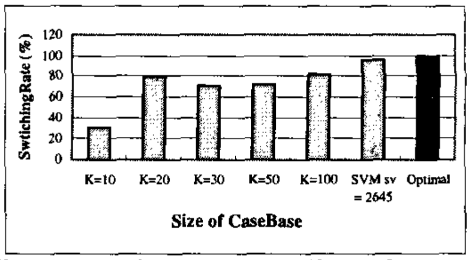

Figure 11. *SwitchingRate* vs. Size of CB for KDD Cup98 data.

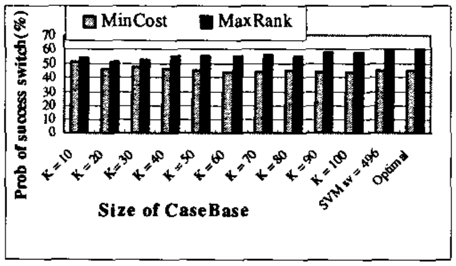

Figure 8.Success Probability P(+lt) vs. size of CB. (Artificial data II, Mean 2 = [7, 8].)

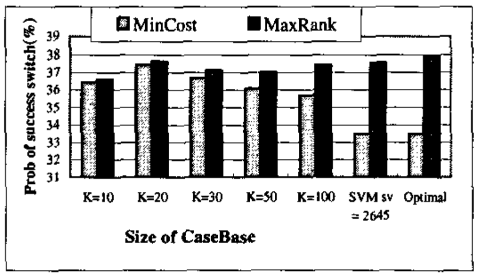

Figure 12. Success Prob P(+lt) vs. size of CB for KDD Cup98 data.