Case-Base Reduction Using Learned Local Feature Weights

Eric C.C. Tsang, Simon C.K. Shiu, X.Z. Wang, and Keith Ho
Department of Computing, The Hong Kong Polytechnic University
Emails: csetsang, csckshiu, csxzwang@comp.polyu.edu.hk

Abstract
Case-base reasoning (CBR) systems making use of previous cases to solve new, unseen and different problems have drawn great attention in recent years. It is true that the number of cases stored in the case library of a CBR system is directly related to the retrieval efficiency. Although more cases in the library can improve the coverage of the problem space, the system performance will be downgraded if the size of the library grows to an unacceptable level. This paper addresses the problem of case base maintenance by developing a new method to reduce the size of large case libraries so as to improve the efficiency while maintaining the accuracy of the CBR System. To achieve this, we adopt the *local feature weights approach*. This approach consists of three phases. The first phase involves partitioning the case-base into different clusters. The second phase involves learning the optimal local feature weights for each case and the final phase involves reducing the case-base based on the optimal local weights. This paper focuses on the last two phases. To justify the usefulness of the method, we perform an experiment which uses efficiency, competence, and ability to solve new problems as the benchmark to verify our design.

1. Introduction

The CBR System is a type of problem solving technology [1] which tries to find a suitable solution to a new problem based on the past experience accumulated in the case library. The major components of CBR include *Indexing Component, Retrieval Engines of Cases, Adaptation, Evaluation of the New, and Memory Update for New Cases*. All these components are inter-related which work together to give life to a CBR system.

Recently the Case-Base Maintenance (CBM) issues have drawn much attention and many researchers start to find a better way to improve the performance of CBR systems. The key researchers include Smyth et al. [2], and Leake et al. [3]. As time passes, more and more experience can be accumulated through problem solving cycles and reserved in the case-base for future reuse. The size of a case-base is directly related to the retrieval efficiency of a CBR system. Although a case-base with small size offers higher potential efficiency benefits, it suffers from insufficient coverage of the problem space. In most of the large-scale CBR systems, the tradeoff between the number of cases stored in the case library and the retrieval efficiency becomes a critical issue. As more and more large-scale and critical CBR systems have emerged in the market, the importance of CBM has drawn great attention of the CBR community.

To tackle the performance problem, one realizes that the **competence** and **efficiency** of the CBR systems are the two main factors contributing to their performance. Striking an optimal balance between these two factors is definitely an important key to improve the performance of any case-bases.

In the area of performance optimisation, removing redundant cases from the case library is one of the effective solutions. However, designing a good deletion strategy is not an easy task. In order to preserve the competence of a case-base after reduction process, we need to understand the characteristics of a case-base and know what cases need to be deleted from the case-base. Among different methods to solve this problem, **learning local feature weights** proposed by the authors is a promising approach. This paper is organized into different sections. In section 2 we present a method on how to learn local feature weights, while in section 3 a method is proposed to reduce the case-base by learning the optimal local feature weights. Section 4 presents an experiment to demonstrate our method and its experimental results. Finally, a conclusion is mentioned in section 5.

2. Local Feature Weights Learning

In this section we present the presentation and construction of case-base. We look into how local feature weights could be used and applied to case-base. Similarity measure between two cases and an algorithm to learn the optimal local feature weights are presented.

2.1 Case-Base Construction And Representation

Assuming that a case library contains N cases. Formally, we can write **case library** or **case-base** as $CL = \{e_1, e_2, \cdots, e_N\}$ in which each case can be identified by a set of problem descriptors or features In a case library, if the number of feature is *j*, we can formally use a collection of features $\{F_j\ (j = 1, \cdots, n)\}$ to index the cases and use a variable *v* to denote the solution. Each feature, $F_j\ (1 \le j \le n)$, and the solution, *v*, take value in a specified metric space.

In vector notation, the *i-th* case e_i in the library is represented as a *n*+1 dimensional vector,

$$e_i = (x_{i1}, x_{i2}, \cdots, x_{in}, v_i)$$

where x_{ij} corresponds to the value of feature $F_j (1 \le j \le n)$ and v_i corresponds to the solution $(i = 1, \cdots, N)$.

With this structure, CBR systems can solve new problems by remembering previous cases and retrieving the solutions to similar prior problems based on their degree of similarity. However, how can we make this work? The key is the similarity function.

2.2 Similarity Function And Local Feature Weight Concept

The core part of the matching process is based on the **similarity function** of the CBR system. Most case-based systems represent cases by features and employ a similarity function to compute the degree of match of an old case in the case library to a new one.

To measure the difference between the features of two cases, we need the metric, ρ which is a **distance function** to measure the actual distance between two features. For two features, say, *x* and *y*, we will have either one of the following cases:

1. $\rho(x, y) \ge 0$,
2. $\rho(x, y) = 0$ if and only if $x = y$, $\rho(x, y) = \rho(y, x)$, and $\rho(x, y) \le \rho(x, z) + \rho(z, y)$.

As the two values are very close, the value of $\rho(x, y)$ should approach zero or very small value. However, in some cases, the similarity value between two cases in the case-base is not so obvious. In order to disambiguate this fuzziness, we use the concept of local feature weights to learn the concepts encapsulated in each cluster.

Local feature weights are the values assigned to every feature of each case to indicate the degree of importance of this feature in the case. We will use a vector representation *w* = (*w*$_1$,*w*$_2$,...,*w*$_n$) to denote the value of weight sets.

(problem domain) and has an associated solution.

Suppose that, for a given case $e_p\ (1 \le p \le N)$ and each $j\ (1 \le j \le n)$, a local weight $w_j(p)\left(w_j(p) \in [0,1]\right)$ has been assigned to the *j-th* feature to indicate the degree of importance of the feature. Then, for any e_q in the library, a weighted metric can be defined as

$$d_p^{(w)}(q) = \left(\sum_{j=1}^{n} w_j^2(p) \cdot \rho^2(x_{pj}, x_{qj}) \right)^{1/2} \qquad (1)$$

By using the weighted distance, a similarity measure between two cases, $SM_p^{(w)}(q)$, can be defined as follows:

$$SM_p^{(w)}(q) = \frac{1}{1 + \alpha \cdot d_p^{(w)}(q)} \qquad (2)$$

where α *is a positive parameter.*

2.3 Cluster Concepts

Assuming that there are *m* clusters produced in phase 1, say, they are namely, $L_1, \cdots, L_m$. Within each cluster, all solutions are considered to be relatively similar with respect to a metric. The problem domain of each fixed cluster is assumed to be composed of several concepts. Even different cases with the same solution, they can have different concepts. The so-called concept here refers to the unique knowledge the case encapsulates. All cases belonging to a concept in a cluster form a **competence group**. For cases with the same concept, one representative can be selected to cover the whole set. All other cases with the same concept are considered as redundant and can be removed from the cluster.

In order to find all the concepts in a case-base, we can use a **neural-fuzzy technique**, with the assumption that each concept can be characterized by one representative case with local weights. The membership degree of a case to a concept is ambiguous. After training the local weights, the fuzziness of the membership degree closes to zero or one.

After obtaining the optimal local weights for each case in the case library, we start to select the representative for each concept. If the local feature weights are well-tuned, there should be only one case left for each concept in a cluster. Therefore, the number of cases in a cluster can greatly be reduced.

2.4 Index Evaluation Function

Initially, we generate concepts for a case-base by assigning random local feature weights to each case. These concepts are fuzzy and their fuzziness depends on the local weights of the cases. To make selecting representatives easier, we need to train the local weights such that the fuzziness of these concepts becomes as smaller as possible. If the difference between two cases is high, it closes to zero. If the similarity between two cases is high, it closes to one. To find out the quality of the concept represented by a case, say, e_p, we need a **feature evaluation function, *E***, to assist us finding the optimal local weights.

$$E(e_p) = E_1 + E_2$$
$$= \sum_{q \in L_i (q \neq p)} \left(SM_p^{(w)}(q)(1 - SM_p^{(1)}(q)) + SM_p^{(1)}(q)(1 - SM_p^{(w)}(q)) \right)$$
$$+ \sum_{q \in L_j (j \neq i,\, q \neq p)} SM_p^{(w)}(q) \qquad (3)$$

in which e_p *is the selected case and* $SM_p^{(w)}(q)$, $SM_p^{(1)}(q)$ *are defined by equations (2) and the SM when w=1 respectively.*

The smaller the evaluation value, the better the corresponding local feature weights for the case. Thus, we would like to find the local weights such that the evaluatuion function attains its minimum. The task of minimizing the evaluation function with respect to the local feature weights is performed by using a **gradient-descent technique**.

According to this technique, we need to find out the change of weight for each feature, Δw_j, using the following equation:

$$\Delta w_j = -\eta \frac{\partial E}{\partial w_j} = -\eta \left(\frac{\partial E_1}{\partial w_j} + \frac{\partial E_2}{\partial w_j} \right) \qquad (4)$$

for $j = 1, \cdots, n$*, where* η *is the learning rate.*

For the computation of $\frac{\partial E}{\partial w_j}$, the following expressions are used:

$$\frac{\partial E_1}{\partial w_j} = \sum_{q \in L_i (q \neq p)} \left(1 - 2 \cdot SM_p^{(1)}(q) \right) \cdot \frac{\partial SM_p^{(w)}(q)}{\partial d_p^{(w)}(q)} \cdot \frac{\partial d_p^{(w)}(q)}{\partial w_j} \qquad (5)$$

$$\frac{\partial E_2}{\partial w_j} = \sum_{q \in L_j (j \neq i,\, q \neq p)} \frac{\partial SM_p^{(w)}(q)}{\partial d_p^{(w)}(q)} \cdot \frac{\partial d_p^{(w)}(q)}{\partial w_j} \qquad (6)$$

$$\frac{\partial SM_p^{(w)}(q)}{\partial d_p^{(w)}(q)} = \frac{-\alpha}{(1 + \alpha \cdot d_p^{(w)}(q))^2} \qquad (7)$$

$$\frac{\partial d_p^{(w)}(q)}{\partial w_j} = w_j \chi_j^2 \Bigg/ \left(\sum_{j=1}^{n} w_j^2 \chi_j^2 \right)^{1/2} \qquad (8)$$

where $\chi_j^2 = (x_{pj} - x_{qj})^2$.

2.5 The Local Weights Training Algorithm

To obtain the optimal local weights, we need to minimise the index evaluation function by using the following training algorithm:

Step 1. Select the parameter α and the learning rate η.

Step 2. Initialize w_j with random values in [0, 1].

Step 3. Compute Δw_j for each *j* using equation (4).

Step 4. Update w_j with $w_j + \Delta w_j$ for each *j* if $w_j + \Delta w_j \in [0, 1]$.

Step 5. Repeat steps 3 & 4 until convergence, i.e., until the value of *E* is less than or equal to a given threshold, or the number of iterations exceeds a certain predefined number.

3. Reduction Algorithm

Once we obtain the trained local feature weights for each case, we proceed to find the representatives for each concept in every cluster. The algorithm for finding concept representatives is determined by **Coverage** and **Reachability**.

Coverage means the ability of a case to cover other cases within a cluster. If a case e_p can cover the case, e_q, the case e_q must be very similar to the case e_p. For each case in a cluster, there is a **coverage set** for it. The higher the ability of a case to cover other cases, the larger its coverage set. Consider a set of cases in a cluster, *C*, we can define its coverage set as follows:

$$\mathit{Coverage}(e_p) = \{ e_q \mid e_q \in C, SM_p^{(w)}(q) \geq 1 - \varepsilon_C \} \qquad (9)$$

The coverage set of e_p, *Coverage(e_p)*, contains all cases in *C* such that the similarity value of e_p and e_q using local weight of e_p should be larger than or equal to 1 minus error standard for coverage, ε_C. In general, the larger the coverage set, the higher the representativeness of the case for the concept it belongs.

Another important concept for finding the concept representatives is reachability. We can think of coverage as a fan-out concept, and reachability as a fan-in concept. Reachability represents the degree of a case that can be reached or covered by other cases within the same cluster. If a case e_p can be covered by a case, e_q, the case e_q must be very similar to the case e_p. By definition,

$$Reachability(e_p) = \{e_q \mid e_q \in C, SM_q^{(w)}(p) \geq 1 - \varepsilon_R\} \quad (10)$$

The **reachability set** of e_p, *Reachability(e_p)*, contains all cases in the cluster, *C*, such that the similarity value of e_p and e_q using local feature weights of e_p should be larger than or equal to 1 minus error standard for reachability, ε_R. In general, the larger the reachability set, the lesser the representativeness of the case for the concept it belongs. Therefore, reachability can be used as a counter measure for the representativeness of a case to a concept.

3.1 Selecting Representatives Algorithm

Input: A case-base with tuned local weights.
Output: A reduced case-base with tuned local weights.
Let *CB* be the whole case-base.
Let $L_i (1 \leq i \leq n)$ be the cluster in the case-base containing *n* clusters.
Let $R_i (1 \leq i \leq n)$ be the reduced cluster for the corresponding cluster L_i.
Let ε_C be the error-standard for coverage.
Let ε_R be the error-standard for reachability.
Let e_r be the current representative case.

```
Begin
For each cluster $L_i \in \{L_1, L_2, ..., L_n\}$
  //Step1: Initialization
  Set the corresponding resultant cluster $R_i$ to empty
  // Step 2: Find Coverage
    For each case $e_p$ in $L_I$
      Determine the Coverage($e_p$)
    Next Case
  // Step 3: Find the set of cases with largest Coverage
      Insert all cases e* such that
      $|Coverage(e^*)| = Max_{e \in L} |Coverage(e)|$ into a
      new set $S_{MAX}$
  // Step 4: Find Reachability if required
      If number of cases in $S_{MAX} \geq 1$
        // Step 5: Find Reachability for all cases In $S_{MAX}$
            For each case $e_p$ in $S_{MAX}$
                Determine the Reachability($e_p$)
            Next Case
        // Step 6: Find the set of cases with smallest
        Reachability
        Insert all cases e** such that
      $|Reachability(e^{**})| = Min_{e \in S} |Reachability(e)|$
      into a new set $S_{MIN}$
        // Step 7: Select random cases as representative if
        unsolved
            If number of cases in $S_{MIN} \geq 1$
              Randomly select a case from $S_{MIN}$ as $e_r$
            Else $e_r$ = the solely case in $S_{MIN}$
            End-If
      Else $e_r$ = the solely case in $S_{MAX}$
      End-If
      // Step 8: Insert the case $e_r$ into $R_I$, $R_i = R_i \cup \{e_r\}$
      //Step 9: Remove all the cases covered by $e_r$ in $L_I$
            $L_i = L_i - Coverage(e_r)$
            If $L_i$ is not empty, Then
                goto Step 2
            Else
                // The set $R_I$ is regarded as the
      selected group of representative cases for
                // the cluster $L_I$
            End-If
      Next Cluster
End
```

For any case e_p and its local weight vector w, the Coverage and Reachability are defined as follows:

$Coverage(e_p) = \{e_q \mid e_q \in L, SM_p^{(w)}(q) \geq 1 - \varepsilon_C\}$ *and*

$Reachability(e_p) = \{e_q \mid e_q \in L, SM_q^{(w)}(p) \geq 1 - \varepsilon_R\}$.

It should be noted that in general $SM_p^{(w)}(q) = SM_q^{(w)}(p)$, or else p and q should be the same case.

4. An Experiment and Its Analysis

In our experiment, we will try to investigate (a) the relationship between the **error-standard for coverage and reachability** and the **reduction rate**, (b) the relationship between the **size** of the case-bases and the **time** required to find out solutions, and (c) the relationship between the **error-standard for coverage and reachability** and the **accuracy.**

In our experiments, we use the **reduction rate** (RR) and the **relative retrieval time** (RRT) as the measuring tools. Their definitions are as follows:

$$\text{RR} = 1 - \frac{\text{Total No. of Cases After Reduction}}{\text{Total No. of Cases Before Reduction}} \quad (10)$$

$$\text{RRT} = \frac{\text{Soln Seek Time for All Cases using the Reduced CB}}{\text{Soln Seek Time for All Cases using the Original CB}} \quad (11)$$

In Pima Indians Diabetes Database, there are totally 768 cases with 500 negative (Cluster 0) cases and 268 positive (Cluster 1) cases. According to the competence model proposed in [8], this case-base is in the adulthood or old-age stage. The objective of this experiment is to investigate the testing accuracy of the system. The testing accuracy is an indicator of the ability of the proposed system to solve new problems. The testing set and training case-base are two exclusive sets generated from the original case-base. To prepare the required case bases, we extract 100 cases from Cluster 0 and 50 cases from Cluster 1. The cases in the testing set are used as target problems. The remaining cases in each cluster are written to the training case-base and used as the training case-base.

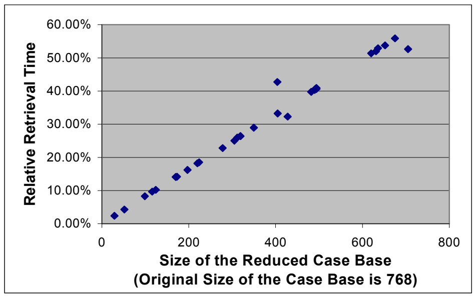

Fig. 1: Relative Retrieval Time versus Size of Reduced Case-Base.

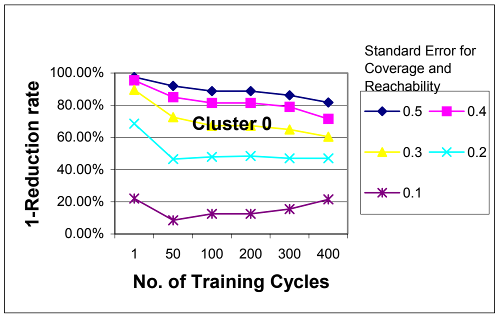

Fig. 2: Reduction Rate versus No. of Training Cycles in cluster 0

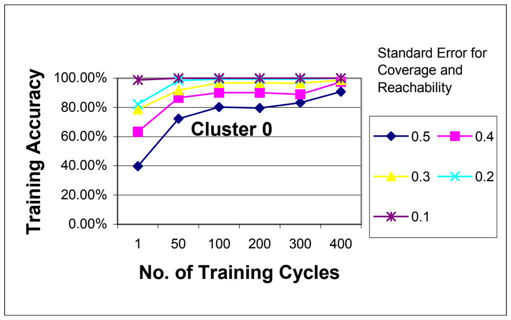

Fig 3: Training Accuracy versus No. of Training Cycles in cluster 0

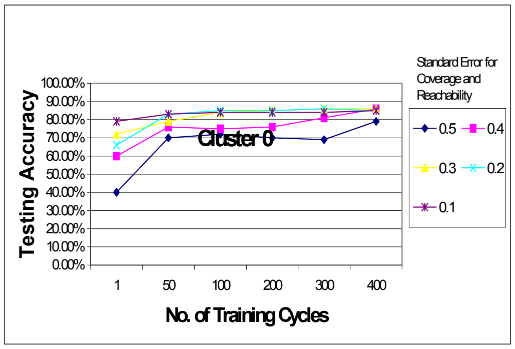

Fig 4: Testing Accuracy versus No. of Training Cycles in cluster 0

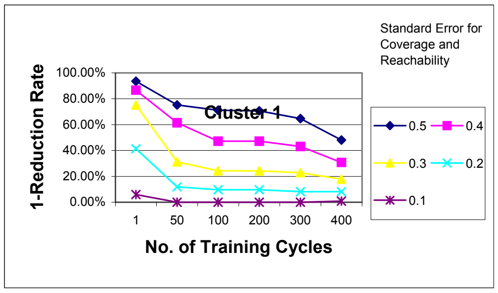

Fig 5: Reduction Rate versus No. of Training Cycles in cluster 1

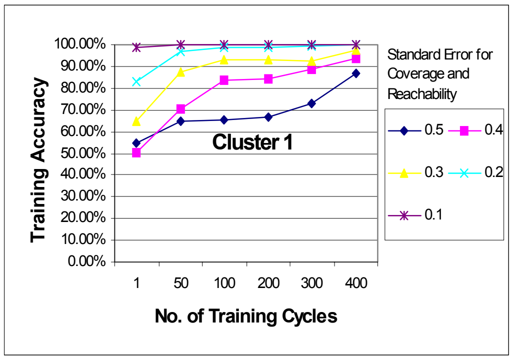

Fig 6: Training Accuracy versus No. of Training Cycles in cluster 1

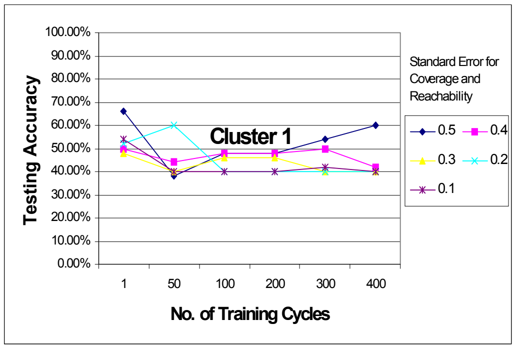

Fig 7: Testing Accuracy versus No. of Training Cycles in cluster 1

From Fig. 1, one may observe that the relative retrieval time is linearly proportional to the size of the reduced case-base. As the size of the reduced case-base is small, the required relative retrieval time is smaller and thus the efficiency of the system is higher.

From Figs. 2 & 5, one observes that the 1-reduction rate is quite high initially (over 90% for both clusters when error standard for coverage and reachability is 0.5). As the local feature weights are trained with longer duration (say, 400 cycles), the 1-reduction rate starts to decrease and reaches an optimal value. At this optimal point, we find that the cluster 0 and cluster 1 can attain over 80% and 50% training accuracy at maximum. This behaviour can be explained by the fact that the coverage set of a case is directly related to the error standard. Higher error standard means more chance that a case can cover another case in the same cluster. As a result, a selected representative can cover more cases and thus the more cases can be removed from the cluster. In our experiment, the results tell us that the cases in the cluster 0 are similar to each other and the cases in cluster 1 seem to be more unique and less redundant as it can be seen from Fig. 7. This explains why the reduction rate of cluster 0 is higher than that of cluster 1.

From Fig. 4 one observes that the maximum testing accuracy for cluster 0 is 86%. This accuracy level is the convergence point for the curve with different error standard. The trend of the curve movement is similar to the training accuracy. The only difference is that it cannot attain 100% accuracy. This can be explained by the fact that as some cases are deleted out from the original case-base, some best representatives may have been taken out.

Assuming 90% is an acceptable accuracy, we can use the 0.4 error standard and 300 training cycles to obtain the optimal reduced case-base. With these parameters, we can achieve over 90% training accuracy in both cluster 0 and cluster 1 as can be seen from Figs. 3 & 6.

5. Conclusion

In this paper, we have described a local feature weight algorithm for maintaining the case-based reasoning system. Our experimental results show that the local feature weights are good indicators to find representatives for clusters in a case-base. Using gradient-descent technique, we can optimize these local feature weights such that the similar groups of cases become more separable and thus make it easier to find out the representatives during reduction process. In conclusion, our approach to reduce the size of case-bases is an effective one. It not only improves the efficiency of the CBR system at a significant level, but also maintains the competence of the case-base to enable it to solve problems accurately.

Acknowledgement

This work is supported by the Hong Kong Polytechnic University Central Research Grant A-PB88.

6. References

[1] A. Aamodt, and E. Plaza, *Case-Based Reasoning: Foundational Issues, Methodogical Variations, and System Approaches*, Artificial Intelligence Communications, Vol. 7, No. 1. 1994, pp 39-59.

[2] B. Smyth, and E. McKenna, *Building Compact Competent Case-Bases. In proceedings of the third International Conference on Case-based Reasoning,* Springer-Verlag, Munich, Germany, 1999, pp 329-342.

[3] D. B. Leake, and D. C. Wilson, *Remembering Why to Remember: Performance-Guided Case-Base Maintenance,* In Proceedings of 5th European Workshop EWCBR 2000, Springer Verlag, pp 161-172.

[4] B. Smyth, and E. McKenna, *Modeling the Competence of Case-Bases,* Lecture Notes in Artificial Intelligence, Springer Verlag, 1998.