Available online at www.sciencedirect.com
ScienceDirect
ELSEVIER
Artificial Intelligence ••• (••••) •••–•••
Artificial Intelligence
www.elsevier.com/locate/artint

Mining competent case bases for case-based reasoning

Rong Pan *,1, Qiang Yang 1, Sinno Jialin Pan 1

Department of Computer Science and Engineering, Hong Kong University of Science and Technology,
Clear Water Bay, Kowloon, Hong Kong, China

Received 29 September 2006; received in revised form 22 April 2007; accepted 30 April 2007

Abstract

Case-based reasoning relies heavily on the availability of a highly competent case base to make high-quality decisions. However, good case bases are difficult to come by. In this paper, we present a novel algorithm for automatically mining a high-quality case base from a raw case set that can preserve and sometimes even improve the competence of case-based reasoning. In this paper, we analyze two major problems in previous case-mining algorithms. The first problem is caused by noisy cases such that the nearest neighbor cases of a problem may not provide correct solutions. The second problem is caused by uneven case distribution, such that similar problems may have dissimilar solutions. To solve these problems, we develop a theoretical framework for the error bound in case-based reasoning, and propose a novel case-base mining algorithm guided by the theoretical results that returns a high-quality case base from raw data efficiently. We support our theory and algorithm with extensive empirical evaluation using different benchmark data sets.

Keywords: Case-based reasoning; Case-base mining; Competence; KGCM

1. Introduction

Case-based reasoning (CBR) is a problem solving strategy that uses previous cases to solve new problems [23,24,46]. Over the years, CBR has enjoyed tremendous success in solving problems related to knowledge reuse. A major open issue in CBR is how to automatically mine a quality case base from a raw case set. A quality case base must have high "competence" in the sense that most future problems can be matched to some cases in the case base that have a solution that can be adapted to solve the problem [40]. In addition, the mined case bases must be compact; they should not contain too much redundant case knowledge in order to ensure the efficiency of case-similarity computation. However, traditionally, case bases are either assumed to be in existence already, or mined manually by human experts. It remains a critical task how to automatically mine quality case bases from the raw case sets.

We refer to the process of automatically mining a compact and high-quality case base from a raw case set as *case-base mining*. We observe that case-base mining is a different problem from case-base maintenance tasks in that case-base mining takes input from a raw case set whereas case-base maintenance takes input from an existing case

[*] Corresponding author.
E-mail addresses: panrong@cse.ust.hk (R. Pan), qyang@cse.ust.hk (Q. Yang), sinnopan@cse.ust.hk (S.J. Pan).

[1] We would like to thank the support of Hong Kong RGC 621606.

0004-3702/$ – see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.artint.2007.04.018

base to improve its performance. These two problems are also related; as we see in the related work section where there are many case-base editing algorithms that can be applied for case-base mining, although these algorithms may not produce results with the best quality and efficiency.

In case-base mining, a natural approach would be to directly extend the case-base maintenance works by Smyth and Keane [40] and by Yang and Zhu [52,54] in order to find a quality case base from a raw case set. However, in doing so, we notice several difficulties:

(1) Case-base mining should be a global optimization problem. As such, there is a need for providing theoretical guidance as to what kind of cases to extract and how many cases to extract in order to obtain the best performance. While the case-categorization provided in [40] addresses one aspect (i.e., case coverage) of this problem, the general issue of what and how many cases to select still remain open. It is our aim to provide a precise error bound to compare the case-base mining algorithms with the quality of the resultant case bases.
(2) In a typical real-world case set such as a Web browsing log file, the cases can be stochastic in nature or may contain mistakes. Stochastic cases are those that deviate from a certain mean according to some laws of nature, whereas the mistakes in cases occur when the problems or solutions are described with some kind of inherent error. In this paper, we refer to both types of cases as noisy cases. Thus a key problem is to recognize and handle the noisy cases. In this paper we provide a general framework for case-feature selection and case mining in order to remove the noisy features and cases. We observe that in the past, other solutions to handling noise have been attempted. For example, some CBR systems allow $k > 1$ in k-NN search to improve their robustness.
(3) The previous works on case-base maintenance paid little attention to the distribution of future problems to be solved. Some previous works are coverage-based algorithms in that the categorization of useful cases often correspond to outlier or rare cases. In addition, we observe that in many data sets, most of the cases have approximately the same coverage. Thus, the coverage-based case-base mining algorithms may sometimes not provide a good basis for selecting cases into the final case base.

In this paper, we present a case-base mining framework that includes an error-bound theorem that serves as the theoretical foundation, and a case-base mining algorithm. This framework is aimed at solving the three challenges listed above. We first present a theoretical result that shows (a) the cases that are selected for a high-quality case base must be accurate, highly representative and diverse in future-problem distribution; (b) there is an optimal number of such cases for each problem domain and the size of an optimal case base is bounded. We then present our algorithm called *Kernel-based Greedy Case-base Mining* (KGCM), which extends our earlier solutions in [31] to solve the above problems. In the KGCM algorithm, we first map the problem set from the original problem space to a feature space, where the features can be obtained through a kernel transformation. Then we present a Fisher Discriminant Analysis (FDA) based feature-extraction method to help remove noise and extract the highly predictive features. Finally, in the KGCM algorithm, we take into account the "diversity" of the selected cases in terms of the coverage of future problems, which is a direct result of our theoretical framework. We have done extensive experiments to verify our theory and algorithm. Our experiments confirm that the KGCM algorithm outperforms many of the previous coverage-based algorithms in building high-quality case bases while preserving and sometimes enhancing the competence of the case bases.

The rest of the paper is organized as follows. The next section discusses related works. Section 3 builds a theoretical model for case-base mining. Section 4 presents the KGCM algorithm. Section 5 presents the experimental results for the evaluation of KGCM. Section 6 concludes with a discussion of future work.

2. Related work on case-base mining

In this paper, a case set refers to a set of initial input cases that are collected in practice. We define a case as a problem-solution pair. That is, each case c in a case set is a pair $c = (\vec{x}, s)$, where $s \in S$ is a solution corresponding to a problem feature vector $\vec{x}$, and S is a set of solutions. Given a training case set T containing a set of problem-solution pairs: $T = \{(\vec{x}_i, s_i), i = 1, 2, \ldots, N\}$, our objective is to select a subset, which is called a case base, $CB = \{(\vec{x}_{i_j}, s_{i_j}), j = 1, 2, \ldots, M\}$ of the training case set, such that the competence of similarity-based problem solving based on CB is sufficiently high for future problems. We formally define competence in Section 5, but here we can understand this concept as a measure of the collective problem solving effectiveness of a case base. This problem is

Table 1
Notations used in the paper

Notation	Meaning
T	Raw case set
CB	Case base
T	A general case set, which can refer to T or CB
X	Problem set
$\mathfrak{X}$	Transformed problem set
S	Solution set
$\vec{x}$	Problem feature vector
c	A case, $c = (\vec{x}, s)$
$\vec{\mathfrak{s}}$	Solution vector of $\vec{x}$
N	Size of T
M	Size of the CB
l	Size of the solution set
d	Dimension of problem feature vector $\vec{x}$
$E_{\vec{x},\vec{\mathfrak{s}}}[\cdot]$	Expectation over $\vec{x}$ and $\vec{\mathfrak{s}}$ with respect to their probabilities
CRS	Candidate Reachability Set of a Problem
CCS	Candidate Coverage Set of a Case
Coverage	See Definition 1
Reachability	See Definition 2
GroupCoverage	See Definition 4
RelativeCoverage	See Definition 3

referred to as *case-base mining*. The major concepts that we use in this paper are listed in Table 1. We also put a more complete notation list in Appendix B.

Generally speaking, case-base maintenance aims at updating an existing case base (*CB*) in order to maintain problem solving competence. In case-base mining, the input is not an existing case base, but a raw case set T, where T can be much larger and noisier. Furthermore, case-base mining stresses scalability in that algorithms for case-base mining must be applicable to large case sets. Other researchers used such terms as *case editing* when they refer to the process of updating a given case base through adding, deleting and combining cases.

There are several different ways of categorizing existing case-base mining algorithms. We classify these algorithms in three dimensions: *search direction, order sensitivity, and evaluation criteria* (see, for example, [48,49]):

Search Direction There are two search directions: incremental search and decremental search. The incremental search process begins with an empty subset *CB* and incrementally adds instances from T to *CB* where *CB* is the mined case base, as long as the new cases satisfy some predefined optimality criteria.

The decremental approach begins a search for a case base with a mined case base $CB = T$. It then finds case instances to remove from the *CB*. The advantage of this approach is that at all times, it has a global view of the remaining case base.

Order Sensitivity Associated with both of the above search directions, another dimension in categorization is to classify case-base mining algorithms by means of *the order* in which cases are added or deleted. When adding instances into a case base *CB* or removing instances from it, some case-base mining algorithms need to select and evaluate each of the existing instances one by one. These algorithms are *sensitive* to the order in which the instances are selected for addition or deletion. Otherwise, the algorithms are known as *order insensitive*.

Evaluation Criteria Yet another aspect that distinguishes different case-base mining algorithms is the *evaluation criteria* for an instance or a set of instances to be considered as final cases. Some criteria are *local* measures in which only the neighboring instances of the evaluated ones are involved. Others are *global* in that the tested cases are evaluated in the entire set of the original training instances.

We now review the previous case-base mining algorithms along the search dimension. At the end of the section, we categorize the algorithms along three different dimensions as given above, in Table 2. An early system is Hart's Condensed Nearest Neighbor algorithm (CNN) [19], which was an early attempt at finding a minimally *consistent* subset of the training case set incrementally. Here consistent sets of cases are defined as any subset of T that offers the

Please cite this article in press as: R. Pan et al., Mining competent case bases for case-based reasoning, Artificial Intelligence (2007), doi:10.1016/j.artint.2007.04.018

Table 2
Summary of the related works of case-base mining algorithms

	Search Direction		Order Sensitive		Evaluation Criteria	
	Incremental	Decremental	Yes	No	Local	Global
CNN [19]	√		√		√	
RNN [18]		√	√		√	
ENN [47]		√	√		√	
SNN [34]	√		√		√	
All k-NN [44]		√	√		√	
Prototypes [10]		√	√		√	
Deletion Policy		√	√		√	
IB1 [2,3]	√		√		√	
IB2 [2,3]	√		√		√	
Shrink (subtractive) [22]		√	√		√	
IB3[2,3]	√		√		√	
MCS [8]		√	√		√	
TIBL [53]	√		√			√
Random mutation hill climbing [39]	√		√			√
Encoding length [9]	√		√			√
ICF [7]		√	√		√	
Addition Policy [54]	√		√		√	
RISE [13]		√	√		√	
EACH (NGE) [37]	√		√		√	
RT Family [48]		√	√		√	
DROP (RT) Family [48,49]		√	√		√	
COV, COV-FP [25,26,43]	√			√	√	
RC, RC-FP [25,26,43]	√			√	√	
RFC, RFC-FP [25,26,43]	√			√	√	

same classification accuracy as T itself. As a addition strategy, CNN starts from a randomly selected initial subset CB of the original case set T. Then it iteratively inserts cases into CB that are misclassified by the current case base CB. This process continues until all cases in T are classified correctly by the case base CB. As such, CNN is also order sensitive.

Also in the incremental category, Smyth and McKenna [25,26,43] proposed a series of effective coverage-based algorithms such as COV-FP (MCOV), RFC-FP and RC-FP with different case-sorting techniques. Coverage and reachability are defined as follows.

Definition 1 *(Coverage [40]).* $Coverage(c) = \{c' \in CB\colon\ Adaptable(c, c')\}$, where $Adaptable(c, c')$ means that c can be adapted to solve c''s problem within a given cost limit.

Definition 2 *(Reachability [40]).* Given a case base $CB = \{c_1, \ldots, c_n\}$, $\forall c \in CB$,

$$Reachability(c) = \left\{c' \in CB\colon\ Adaptable(c', c)\right\}.$$

Based on these definitions, algorithm COV-FP sorts cases into descending order of the size of their *coverage* set. Algorithm RFC-FP sorts cases into ascending order of the size of their *reachability* set (see Definition 2). In addition, algorithm RC-FP sorts cases into descending order of their *relative coverage* values, as defined as follows:

Definition 3 *(Relative Coverage [42]).*

$$RelativeCoverage(c) = \sum_{c' \in Coverarge(c)} \frac{1}{|Reachability(c')|},$$

where $Coverage(c)$ and $Reachability(c')$ are as defined in Definitions 1 and 2.

Please cite this article in press as: R. Pan et al., Mining competent case bases for case-based reasoning, Artificial Intelligence (2007), doi:10.1016/j.artint.2007.04.018

These algorithms adds large-coverage cases before other cases so that they can be added into a case base in order of importance.

Zhang [53] presented an approach called *Typical Instance Based Learning* (TIBL) which keeps the case instances near the center of clusters rather than at the border when mining a new case base. The reason is that these cases are more robust and stable when used in case-based reasoning. Ritter et al. [34] extended presented the Selective Nearest Neighbor algorithm (SNN). In this algorithm, every case in the original training case base is required to have a nearer neighbor of a correct case in the new case base as compared to any other cases that belong to different solution categories. If this condition is satisfied by a new case, then it is added to a case base.

Aha et al. [2,3] presented a series of instance-based learning algorithms, which are in the incremental direction. IB1 is the standard one-nearest-neighbor (1-NN) algorithm. IB2 decides whether to add new training cases into a case base in terms of the following procedure: if a new case to be added can already be classified correctly on the basis of the current case base, which starts from an empty set, then the new case is discarded. In this way, only the cases where the learner cannot classify correctly, in the order given, are stored. Thus this method is also *order sensitive*. The problem with IB2 is that it can be easily affected by noisy cases, a problem that is overcome by the IB3 algorithm [2,3].

Unlike the linearly ordered case-base mining algorithms, Skalak presented a method called *random-mutation hill climbing* [39] to add instances to a case base. The method begins with m randomly selected instances in the training case set T as starting cases, where m is a parameter that is supplied by the user. Then in each iteration, a randomly selected instance in T is replaced by another randomly selected instance in $T - CB$, where CB is the current case base. If this replacement strictly improves the classification accuracy of the cases in the training case set T, the decision to change will be held. Otherwise, the change is reversed. This process is repeated for n iterations, where n is another parameter provided by the user.

Rather than evaluating each case individually to tell its effectiveness in the resultant case bases, Cameron-Jones [9] used an *encoding-length heuristic* to determine how good the subset CB is in describing the training set T. The basic algorithm begins with a case-base-growing phase that takes each instance in T and adds it to the case base CB if the addition results in a *lower cost* than not adding it, where a cost measure is defined according to the encoding length heuristic.

The relation between cases can also be modeled geometrically, and case bases are evaluated based on the *generalization ability*. Salzberg [37] introduced a *Nested Generalized Exemplar* algorithm that considers each case a hyper-rectangle conceptually. The case base is initialized with several randomly selected cases in the training case set T. It then iteratively examines the instances to update the distance to a case which in turn updates the corresponding hyper-rectangle. When a new case having the same solution as its nearest neighbors appears, the old case is *generalized* so that it covers the new cases as well. Similarly, Domingos [13] introduced an algorithm called *RISE* that mines a case base from the training case set T where each case is treated as an if-then rule. For each case c in the case base, *RISE* finds the nearest example case c' in T to c that is not covered by c, and then generalizes a new case by combining c and c'. This process repeats until no more rules are generalized during an entire pass.

Smyth and McKenna [41] extended the previous footprint-based models [40] of case-base competence from individual cases to groups of cases. They first defined the *group coverage* of a group of cases as a measure that is proportional to the group size and inversely proportional to the average density of cases. More formally,

Definition 4 *(Group Coverage [41]).*

$$GroupCoverage(G) = 1 + |G| * \bigl(1 - GroupDensity(G)\bigr),$$

where the group density is defined as the average similarity to other cases in the group.

The concept of a group is defined by an equivalence relation induced by mutual coverage of other cases in the case base. In this manner, the competence of a case base is the sum of group-coverage values for all groups in the case base. A case base is then constructed in order to increase the group coverage of a case base. Experiments on benchmark domains showed that the concept of group coverage closely matches the competence of a case base.

Also along the incremental direction, McSherry presented the CaseMaker algorithm [27] to identify the "most useful cases" to be added to a case base. In this method, the best case to add to a developing case-base is selected based on an evaluation of the additional coverage that it provides. The case which provides the most additional coverage to

Please cite this article in press as: R. Pan et al., Mining competent case bases for case-based reasoning, Artificial Intelligence (2007), doi:10.1016/j.artint.2007.04.018

the case-base is added. This method can be given greater autonomy in the construction of a case base, where the role of a user is simply to provide solutions for cases selected by CaseMaker.

Yang and Zhu took a more theoretical approach in the incremental search direction when mining a case base. Given a training case set T, the problem of finding an optimal case base of a smaller size M is shown in [52,54] to be NP-complete [17]. Yang and Zhu designed heuristics to find near-optimal case bases under a uniform distribution. In their case-addition algorithm, the new case base *CB* is initially empty. They repeatedly selected the cases from T that, of those remaining cases, have the maximal *coverage* (see Definition 1) to be added to the new case base. Yang and Zhu [52,54] proved that its case coverage is at least 63% of the optimal case base, for any fixed case-base size M under a uniform distribution of the cases. This would make the algorithm near-optimal under uniform problem distribution in the future.

Along the *decremental search direction*, there has also been a large number of previous works. A method known as MC was introduced by Brodley [8], which keeps track of how many times a case is one of the k nearest neighbors of another case in the final case base. This method allows the case distribution information to play a role, which is one of our major motivations for our proposed method. In [8], if the number of mistakes the algorithm makes on a validation case set is greater than the number of times the reasoning is correct, then the case is removed from the case base.

The Reduced Nearest Neighbor algorithm, or RNN [18] for short, is an extension of the CNN algorithm. This algorithm can additionally remove noisy instances and so-called *internal* instances that are located inside the border of a case cluster, the rationale being that these internal instances offer less benefit as compared to the boundary cases. The resultant case base consists of cases that are scattered around the boundary of the case base. Along the decremental direction, Wilson [47] developed the Edited Nearest Neighbor algorithm (ENN) that improves the previous methods including CNN and RNN by keeping a smoother decision boundary around the clusters of cases. Tomek [44] further extended Wilson's algorithm with his All-k-NN method, by repeating the algorithm for an increasing number of cases. A problem with the All-k-NN method is that it is relatively space inefficient. A slightly different decremental algorithm is Chang's algorithm [10], which reduces the case base size decrementally by repeatedly *merging* two cases into a new case, thus reducing the case base sizes. These new cases are called the *prototypes*, which we can view as the *virtual cases* that are similar to the concept of centroids in a K-means clustering algorithm.

Kibler and Aha [22] presented an algorithm called *Shrink* that decrementally updates case bases starting from the training data. The main criterion used to remove a case is similar to that in the RNN method. While *Shrink* tests if the removed instance is still classified correctly by the remaining cases, RNN considers whether the classification results of *other* instances are accurate.

Smyth and Keane [40] presented a case-deletion-based approach. The premise of this approach is that each case in the case base should be categorized according to its competence. These categorizations are made according to two key concepts: *coverage* and *reachability* (see Definitions 1 and 2). They then defined four categories of cases: pivotal case, support case, spanning case and auxiliary case. A case is a *pivotal* case if its reachability set includes just the case itself. A case c_1 is an *auxiliary* case if it is completely subsumed by another case c_2 in the case base; that is, the coverage set of c_2 includes that of c_1 as a subset, and that c_1 and c_2 are different. In this instance, the case c_1 is auxiliary and is thus considered less important than pivotal cases. In between the pivotal and auxiliary case categories are the *spanning* cases that link together coverage sets of some other cases, and the *support* cases that exist in groups in support of a problem to be solved.

From these definitions, we see that pivotal problems are the most unique, spanning problems are less unique, but auxiliary problems are not unique at all. Based on this classification, Smyth and Keane's footprint deletion (FD) and footprint-utility deletion (FUD) policy delete auxiliary cases first, followed by support cases, then spanning cases, and finally pivotal cases [40]. The difference between FD and FUD lies in that when there is more than one case belonging to the same type of case, auxiliary, spanning, support or pivotal, FUD will delete the case with the lowest utility instead of FD's deletion of the case with the lowest *coverage* or largest *reachability* [40]. This approach was shown to be better than a random deletion policy for preserving competence.

The RT1, RT2 and RT3 algorithms [48], developed by Wilson and Martinez, are also deletion-based algorithms. RT1 is built on the relationship between the nearest neighbors set and the *associates* of each case, where the associates of a case are cases whose nearest neighbors include the case itself. The RT1 algorithm removes cases that can be solved using the remaining cases after its deletion. RT2 sorts the training set by the distances from their nearest *enemy*, which is a case with a different solution. It then follows the same case-removal process as RT1. The difference between RT3 and RT2 is that RT3 uses an additional noise-filtering process to remove instances that are misclassified by their k nearest neighbors, while RT2 does not perform this filtering step.

Brighton and Mellish introduced an *Iterative Case Filtering* Algorithm (ICF) [7] that iteratively removes a case whose absence produces better results as compared to retaining it. This algorithm uses *coverage* and *reachability* (see Definitions 1 and 2) as the selective criteria. It repeatedly uses a deletion rule that removes cases whose reachability size is greater than that of the coverage until the conditions of the rule are not satisfied. That is to say, ICF deletes a case c if c is solved by more cases than those it can solve itself.

In order to keep both inaccuracy and redundancy low, two objectives that are often at odds with each other, Delany and Cunningham [11] presented an interesting competence-based framework that included two phases to both remove the noise by deleting incorrect cases and reduce redundancy in a case base. One objective is to reduce the *damage* that certain cases are causing in misclassification. The approach was evaluated in the domain of spam email filtering. They reported that too aggressive removal strategies can result in some loss of generalization accuracy due to overfitting.

We summarize the above related algorithms in Table 2 in terms of the three issues: search direction, order sensitivity and evaluation criteria. We can see that the past literature on the case-base mining topic has large overlap, and yet there are still several unfulfilled objectives in the previous research. Our proposed research (see Section 4) is aimed at integrating the advantages of these previous works in a general case-base mining framework.

Some works in case-index-learning are also related to case-base mining. For example, in similarity function learning, Aamodt et al. [1] presented learning algorithms for similarity functions used in case retrieval. In learning indexing structures, Bonzano et al. [6] explored how to learn the index of a case base when there is already a case base. Patterson et al. [32] extended the use of K-means clustering algorithm for regression-based numerical value prediction.

As we discuss in Section 4, case-base mining is also closely related to feature selection and transformation in machine learning. In the field of machine learning, there has been much research in feature-space transformation with *kernel functions*, which is a technique that we will exploit (see Section 4). In this area, some examples are *Kernel Principle Component Analysis* (Kernel PCA) and *Kernel Independent Component Analysis* (Kernel ICA) [4,38]. Recently, some researchers have proposed to use kernel transformations for case-based reasoning. For example, Corchado and Fyfe et al. explored case-base mining with a unsupervised kernel method [16]. Pan et al. [30] proposed to use nonlinear similarity measures for case retrieval and presented some preliminary results in unsupervised frameworks. In case-based reasoning, it is important to relate the input features and the target solutions. However, the previous works that rely on an unsupervised framework do not address this issue satisfactorily due to the lack of global guidance provided by the distribution of cases and their solutions. In order to use this relationship fully, in our work, as we introduce in Section 4, we apply *Kernel Fisher Discriminant Analysis* (KFDA) [29,35] to a case base by taking into account both the similarity measures and at the same time the solution distribution of cases in the kernel space.

3. A theoretical model for case-base mining

In this section, we wish to relate case-base competence with the choice of cases in a case base, in order to ensure the competence of a case base. The competence concept is directly related to the potential "loss" that are made in solving future problems. We first illustrate a motivating example. Then we define the notion of loss to denote the collective mistakes made by a case-based reasoning system for solving problems. After that, we focus on the expected loss of a mined case base and explore the relationship between the expected loss and the individual component case bases.

Many previous algorithms are based on categorizing and ranking cases according to how many other cases can cover them. However, this coverage-based approach alone may not handle noisy cases well. To understand this better, consider an example where the distribution of cases is as shown in Fig. 1. In this figure, each circle or star represents a case. Suppose that we use a one-nearest-neighbor algorithm (1-NN) to find the solution for new cases. Assume that each case can be covered by one of its nearest neighbors, if it is *of the same shape* (a circle or a star). As a result, we can see that the dark circle and star are two cases that are unique; that is, no other cases can cover them. However, if they are selected in the final case base, even though other cases might be selected as well, the white circles around the black star will be misclassified as a star, and similarly the white star will be classified as a circle. These two noisy cases may cause errors depending on what future problems are input. Thus, from this example, we see that ranking cases by coverage only might mislead a case-base mining algorithm to choose the noisy and outlier cases. In this paper, we study how to filter out these noisy cases through feature selection and case selection.

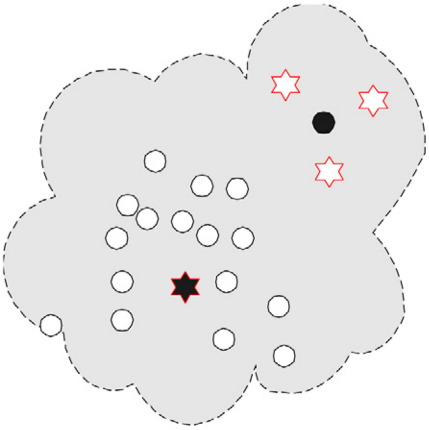

Fig. 1. Illustrating noisy cases.

In general, there is a need for some theoretical guidance as to how to select cases into a high competence case base. In addition, a feature transformation may be necessary to allow better correlation between the problem descriptions of case clusters and their solutions. This is done so that the true coverage of cases can be maximized in a transformed feature space, as we see in Section 4 after the model building for case-base mining in the next subsection.

We model the ranking process of case-based reasoning as follows. Given an input problem $\vec{x}$, we search a case base *CB* for k most similar cases. Among the k cases, one or more solutions will be chosen for adaptation. Some case-based reasoning algorithms use the one-nearest neighbor as the final solution for adaptation, others allow all k cases to vote and select a majority solution. Still some others integrate a subset of the k solutions into a final one. These situations can be jointly represented using a solution vector defined as follows.

Definition 5 *(Solution Vector).* Let l be the size of the solution set S in the domain. For a given input problem, a solution vector $\vec{\mathfrak{s}}$ is defined as an l-dimensional vector $\vec{\mathfrak{s}} = (\mathfrak{s}_1, \mathfrak{s}_2, \ldots, \mathfrak{s}_l)$, where $\mathfrak{s}_i \in [0, 1]$. Each element in a solution vector represents a probabilistic vote on how likely that solution is the final solution. The sum of all elements in a solution vector is one.

Therefore, for example, in a binary classification problem, the vector length $l = 2$. In general, any solution can be represented by a solution vector. For example, a solution $s_2 \in S$, where S is a size l set, can be represented as $(0, 1, 0, \ldots, 0)$, where only the second element is 1, and the rest are 0. A solution vector can be used to represent a chosen solution to a case-based reasoning problem in a probabilistic sense as well. For instance, suppose that there are three different solutions in the domain and we use a 5-NN algorithm. If three nearest neighbors vote for the first solution s_1, whose distances to the new problem are 0.4, 0.5 and 0.5 respectively, two nearest neighbors vote for the second solution s_2, whose distances to the new problem are 0.4 and 1 respectively, and there is no vote for the third one, the solution vector should be the weighted sum vector $(0.65, 0.35, 0)$. Here, for example, 0.65 is computed by $(1/0.4 + 1/0.5 + 1/0.5)/(1/0.4 + 1/0.5 + 1/0.5 + 1/0.4 + 1/1)$. According to this solution vector, we return the first solution s_1 for the problem, as the proportion of it is 0.65.

Using the solution vector defined above, we can now treat the difference between a solution found by a CBR system and the true solution vector as a vector of difference. For example, using the above example, the difference vector should be $(-0.65, 0.65, 0)$ if the true solution is s_2. The L^2-norm of this difference is then taken as the error or loss value for the problem we are trying to solve. An important effect of using the concept of a solution vector is that the error can be defined for both real value and discrete type solutions. They are treated in a similar way using this formalism, and thus we can include in this formalism both regression type of inference and logical and plan-based inference using CBR, which we pursue in the future.

We now consider the effect of choosing a subset of the training case set as a case base, and measure the quality of the expected solutions by using the case base to solve problems. The operation of a case base reasoning algorithm can be formalized using a mapping function f that maps from a problem $\vec{x}$ and a case base, which is also a case set, to a solution vector. We call f a k-NN estimator.

Please cite this article in press as: R. Pan et al., Mining competent case bases for case-based reasoning, Artificial Intelligence (2007), doi:10.1016/j.artint.2007.04.018

Definition 6 *(k-NN Estimator).* Given a problem feature vector $\vec{x}$, a case set T, and some similarity-function parameter set Θ (Θ is a collection of parameters that specify the form of a similarity function), a parameterized k-NN Estimator $f(\vec{x}; \mathsf{T}, k, \Theta)$ maps $\vec{x}$ to a solution vector $\vec{\mathfrak{s}} \in [0,1]^l$, where $\vec{\mathfrak{s}} = (\mathfrak{s}_1, \mathfrak{s}_2, \ldots, \mathfrak{s}_l)$.

The parameter set Θ can specify the form of the similarity function and its parameters. Note that in Definition 6, $\mathfrak{s}_i$ $(i = 1, \ldots, l)$ can be viewed as the weight of the ith solution s_i estimated by f. Additionally, for simplification, we will omit some parameters of f. That is, when there is no ambiguity, instead of $f(\vec{x}; \mathsf{T}, k, \Theta)$, we write simply $f(\vec{x})$ or f to express the estimation of problem $\vec{x}$.

For a problem $\vec{x}$, we define the k nearest neighbor cases of the problem as a set of candidate cases by similarity. These cases are called the candidate reachability set of the problem.

Definition 7 *(Candidate Reachability Set of a Problem).* $\mathrm{CRS}(\vec{x}; \mathsf{T}, k, \Theta)$ is defined as the set of k nearest neighbors of a problem $\vec{x}$ given a case set T and similarity-function parameter set Θ.

Notice the difference between $\mathrm{CRS}(\vec{x}; \mathsf{T}, k, \Theta)$ and the reachability set of a problem as defined in Definition 2. The reachability set is the set of cases whose solutions can actually solve $\vec{x}$, whereas the candidate reachability set is the set of cases that are considered as candidate cases for solving $\vec{x}$ due to their close distances. Some cases in CRS $(\vec{x}; \mathsf{T}, k, \Theta)$ actually give incorrect solutions.

Similar to CRS, we define the candidate coverage set of a case as the set of problems that a case c is similar to; thus the case c can be a candidate for solving these problems. However, c may not always be the correct solution.

Definition 8 *(Candidate Coverage Set of a Case).* $\mathrm{CCS}(c; \mathsf{T}, k, \Theta) \triangleq \{\vec{x} \mid c \in \mathrm{CRS}(\vec{x}; \mathsf{T}, k, \Theta)\}$.

Similar to the difference between $\mathrm{CRS}(\vec{x}; \mathsf{T}, k, \Theta)$ and reachability set of a problem, the candidate coverage set of a case and the coverage set of a case as defined in Definition 1 are also different. The difference lies in that some problems in the CCS cannot actually be solved using the case c, because the latter is purely computed using similarity.

We formally define a loss function to denote the amount of error incurred in solving a problem $\vec{x}$.

Definition 9 *(Loss).* Consider the predicted solution $f(\vec{x})$ and the true solution vector $\vec{\mathfrak{s}}$. Their difference is captured using the *loss* function $L : [0,1]^l \times [0,1]^l \to \mathbb{R}$. The L^2-norm loss function is defined as

$$L\big(f(\vec{x}), \vec{\mathfrak{s}}\big) = \big\| f(\vec{x}) - \vec{\mathfrak{s}} \big\|^2. \tag{1}$$

The expected loss (over the raw case set T) of the k-NN estimator f is

$$E_{\vec{x},\vec{\mathfrak{s}}}\big[L\big(f(\vec{x}), \vec{\mathfrak{s}}\big)\big] = \int\limits_T \big\| f(\vec{x}) - \vec{\mathfrak{s}} \big\|^2 \, \mathrm{d}\vec{x} \, \mathrm{d}\vec{\mathfrak{s}}. \tag{2}$$

When we deal with discrete cases and solutions, we replace the integration with the summation. The goal of case-base mining is to search for the optimal $CB \subseteq T$, k and Θ that maximize the expected competence of CB, which is equivalent to minimizing the $E_{\vec{x},\vec{\mathfrak{s}}}[L(f(\vec{x}; CB, k, \Theta), \vec{\mathfrak{s}})]$. That is,

$$\begin{align}
(CB^*, k^*, \Theta^*) &= \mathop{\arg\min}_{CB,k,\Theta} E_{\vec{x},\vec{\mathfrak{s}}}\big[L\big(f(\vec{x}; CB, k, \Theta), \vec{\mathfrak{s}}\big)\big] \tag{3}\\
&= \mathop{\arg\min}_{CB,k,\Theta} \int\limits_T \big\| f(\vec{x}; CB, k, \Theta) - \vec{\mathfrak{s}} \big\|^2 \, \mathrm{d}\vec{x} \, \mathrm{d}\vec{\mathfrak{s}}. \tag{4}
\end{align}$$

We now show how to decompose the expected loss into its components so that we can infer some general criteria for reducing the expected loss function. First, since case-based reasoning is based on finding the k nearest neighbors of a problem, and each individual case defines its own corresponding estimator, a k-NN estimator $f(\cdot)$ can be decomposed into the sum of weighted nearest neighbor components as follows:

$$f(\vec{x}; T, k, \Theta) = \frac{1}{\sum_{i=1}^{k} \omega_i} \sum_{c_i \in \mathrm{CRS}(\vec{x}; T, k, \Theta)} \omega_i f\big(\vec{x}; \{c_i\}, 1, \Theta\big). \tag{5}$$

Please cite this article in press as: R. Pan et al., Mining competent case bases for case-based reasoning, Artificial Intelligence (2007), doi:10.1016/j.artint.2007.04.018

The weights ω_i in Eq. (5) can be defined in various ways. One method is to define it by similarity such that the weight is proportional to the similarity measure. An alternative is to define it probabilistically. In either case, the weight should reflect the importance of a case as accurately as possible.

Theorem 3.1. *Let*

$$g(\vec{x}; c_i) \triangleq \frac{\omega_i}{\sum_{n=1}^{k} \omega_n}\bigl(f\bigl(\vec{x}; \{c_i\}, 1, \Theta\bigr) - E_{\vec{\mathfrak{s}}|\vec{x}}[\vec{\mathfrak{s}}]\bigr).$$

The expected loss function can be decomposed as

$$E_{\vec{x},\vec{\mathfrak{s}}}\bigl[L\bigl(f(\vec{x}; CB, k, \omega), \vec{\mathfrak{s}}\bigr)\bigr] = \mathit{Bias} + \mathit{Covariance} + \mathit{Noise}, \tag{6}$$

where

$$\mathit{Bias} = \sum_{i=1}^{M} E_{\vec{x}\sim \mathrm{CCS}(c_i)}\bigl[\bigl\| g(\vec{x}; c_i)\bigr\|^2\bigr], \tag{7}$$

$$\mathit{Covariance} = \sum_{i=1}^{M} E_{\vec{x}\sim \mathrm{CCS}(c_i)}\Biggl[\sum_{\substack{c_j \in \mathrm{CRS}(\vec{x}; CB, k, \Theta)\\ i \neq j}} \bigl\langle g(\vec{x}; c_i), g(\vec{x}; c_j)\bigr\rangle\Biggr], \tag{8}$$

$$\mathit{Noise} = E_{\vec{x},\vec{\mathfrak{s}}}\bigl[\bigl\| E_{\vec{\mathfrak{s}}|\vec{x}}[\vec{\mathfrak{s}}] - \vec{\mathfrak{s}}\bigr\|^2\bigr]. \tag{9}$$

The proof of this theorem is given in Appendix A. Theorem 3.1 underlies several important points in case-base mining, in terms of minimizing the three terms of Eq. (6). We summarize the implications as follows.

- First, to reduce the expected loss of Eq. (6), we need to minimize the three terms of the right-hand side of Eq. (6) simultaneously. Note that the "noise" term in Eq. (9) reflects of the consistency of solutions in the case base, and does not depend on the choice of the estimator. Thus, if we can select a good set of features while reducing the potential noise, we can reduce the noise term in the loss function. We see that this can be achieved by performing feature selection as well as feature transformation using kernels.
- Second, to reduce the "Bias" term (Eq. (7)), we should make each case c_i able to solve most of the problems close to it in terms of the relative similarity $\omega_i / \sum_{n=1}^{k} \omega_n$. This observation supports the idea that cases selected into a case base should have as much coverage as possible.
- Third, to reduce the term "*Covariance*" (Eq. (8)), we can lower the term $\omega_i / \sum_{n=1}^{k} \omega_n$ or $\omega_j / \sum_{n=1}^{k} \omega_n$, as $(f(\vec{x}; \{c_i\}, k, \Theta) - E_{\vec{\mathfrak{s}}|\vec{x}}[\vec{\mathfrak{s}}])$ is bounded. This can be done by increasing the *dissimilarity* between different cases c_i and c_j, so that when one weight is large, another weight is small. Moreover, the "*Covariance*" term can be reduced if $f(\vec{x}; \{c_i\}, k, \Theta)$ and $f(\vec{x}; \{c_j\}, k, \Theta)$ are orthogonal, since this will reduce the inner product

  $$\bigl\langle f\bigl(\vec{x}; \{c_i\}, k, \Theta\bigr) - E_{\vec{\mathfrak{s}}|\vec{x}}[\vec{\mathfrak{s}}], f\bigl(\vec{x}; \{c_j\}, k, \Theta\bigr) - E_{\vec{\mathfrak{s}}|\vec{x}}[\vec{\mathfrak{s}}]\bigr\rangle. \tag{10}$$

  This again means that we should strive to make the case base as *diversified* as possible so that the dissimilarity between all pairs of cases are large.
- In addition, we can see that the similarity parameter set Θ can also affect the "Bias" and "Covariance" terms. In Section 4.4, we introduce a kernel transformation for this purpose, as we can have a large number of both linear and nonlinear functions that can be used to model the similarity of two vectors. For instance, a linear kernel function is equivalent to the Euclidean distance function that is commonly used.

4. Case-base mining algorithms

From the analysis in Section 3, we consider the following guidelines for discovering a "good" case base from a case set T:

- Each case should cover as much of the problem space as possible to reduce the potential bias, and
- The cases should be as diverse as possible to reduce co-variance in producing errors.

Please cite this article in press as: R. Pan et al., Mining competent case bases for case-based reasoning, Artificial Intelligence (2007), doi:10.1016/j.artint.2007.04.018

When designing our case-base mining algorithm, we consider both these conditions. In this section, we show how to enlarge the coverage of a case by applying the *Fisher Discriminant Analysis* (FDA) [15] to select features and remove noise. When selecting the next case to be added to a case base, we try to maximize the diversity of the case base through the introduction of a diversity measurement.

4.1. Feature extraction in the original problem space

Case-base mining aims to extract a case base from a training case set where cases can cover the future problem space. In this subsection, we perform the Fisher Discriminant Analysis to find candidate features. In the next subsection, we discuss the case-base mining problem.

FDA was first developed by Fisher [15] for linear data analysis. Classical linear discriminant analysis is a statistical method that attempts to project data vectors that belong to different solution categories into a lower dimensional space where data points belonging to similar solutions become closer and those with different labels become more separable. For simplicity, below we assume that the case solutions have been partitioned into l different solution categories or classes.

Assume that we are given a case set $T = \{c_1, c_2, \ldots, c_N\}$ where c_i denotes a case which is a pair of problem features and a solution $(\vec{x}_i, s_i)$, where $\vec{x}_i \in \mathbb{R}^d$ and s_i is the solution part of the case, after we have transformed all symbolic or nominal features to numeric ones. Various methods for such transformation have been designed. For example, a three-valued symbolic attribute can be transformed to three binary valued numerical attributes, each being of the form: for each symbolic attribute A_i with values v_i, $i = 1, 2, 3$, we invent a new attribute $(A_i = v_i)$ corresponding to each value v_i (see Chapter 2 of [50]). Suppose that $X_i = \{\vec{x}^i_1, \vec{x}^i_2, \ldots, \vec{x}^i_{l_i}\}$ is the set of cases with the same solution category s_i. The centroid $\vec{m}_i$ of X_i can be calculated as $\vec{m}_i = l_i^{-1} \sum_{j=1}^{l_i} (\vec{x}^i_j)$, and l_i is the size of X_i. Then the total difference of the cases corresponding to the same solution s_i can be expressed as $\mathcal{S}_W = \sum_{i=1}^{l} \sum_{j=1}^{l_i} (\vec{x}^i_j - \vec{m}_i)(\vec{x}^i_j - \vec{m}_i)^{\mathsf{T}}$, where $^{\mathsf{T}}$ is the transpose operator, and the sum of distances between any two centroids of different solutions can be expressed as $\mathcal{S}_B = \sum_{i,j=1}^{l} (\vec{m}_i - \vec{m}_j)(\vec{m}_i - \vec{m}_j)^{\mathsf{T}}$. FDA attempts to find a new direction $\vec{\mathbf{w}}$ such that the projection of $\mathcal{S}_B$ can be maximized while the projection of $\mathcal{S}_W$ can be minimized. That is, given a vector $\vec{\mathbf{w}}$, we need to maximize the following formula

$$J(\vec{\mathbf{w}}) = \frac{\vec{\mathbf{w}}^{\mathsf{T}} \mathcal{S}_B \vec{\mathbf{w}}}{\vec{\mathbf{w}}^{\mathsf{T}} \mathcal{S}_W \vec{\mathbf{w}}}. \tag{11}$$

The computational details of FDA can be found in [15,20,21].

4.2. Case-base mining

With the eigenvectors found by solving Eq. (11), we wish to then find a case base of size M, where M is an empirically computed value. Suppose that we have found the most correlating direction $\vec{\mathbf{w}}$ extracted by FDA. Consider how to mine a new case base that has a size smaller than the raw case set T for efficient case retrieval. To achieve this, we need to find the most representative cases for the same solutions. We define a *global diversity* among the mined case base with respect to the raw case set T as the spread of the cases among the data which we wish to maximize.

We define an evaluation function for cases that can combine two factors: the importance of a case in terms of its projection onto an eigenvector $\vec{\mathbf{w}}$, and its diversity score with respect to the cases that are mined already. These functions are defined below. In the functions, *CB* is a case base.

$$\mathit{Eval}(\mathit{CB}) = \mathit{weight}(\mathit{CB}) * \mathit{globaldiv}(\mathit{CB}), \quad \text{where} \tag{12}$$

$$\mathit{weight}(\mathit{CB}) = \prod_{c_i \in \mathit{CB}} (\mathbf{w}_i), \quad \text{and} \tag{13}$$

$$\mathit{globaldiv}(\mathit{CB}) = \min_{c_i, c_j \in \mathit{CB}} \mathit{dissimilarity}(c_i, c_j). \tag{14}$$

In Eq. (14), *globaldiv*(*CB*) is the global diversity measure of a case base *CB*. *dissimilarity*(., .) represents the dissimilarity of two cases, which can be defined as the Euclidean distance between the two cases. We can replace the Euclidean distance with other measures of case distances.

Table 3
Linear greedy case-base mining (LGCM) algorithm

```
     LGCM($T$,$M$)
     $T$ is the case set, $M$ is the maximum size of case base to be mined.
1    Run FDA on $T$ to find the direction $\vec{\mathbf{w}}$
2    $CB = \emptyset$
3    $U = T$
     % Steps 4 to 7 find the initial seeds for the case base.
4    for every $s_i \in S$, $T_i = \{(\vec{x}^i_j, s_i) \mid (\vec{x}^i_j, s_i) \in T\}$
5        Compute the projection $\mathbf{w}^i_j$ of each data record $\vec{x}^i_j$ onto $\vec{\mathbf{w}}$
6        Select the case $c$ with largest $\mathbf{w}^i_j$, $CB = CB \cup \{c\}$, $U = U - \{c\}$
7    end for
8    while $U \neq \emptyset$ and $|CB| < M$
9        Select the case $c$ with largest evaluation value: $Eval(CB \cup \{c\})$
10       $CB = CB \cup \{c\}$, $U = U - \{c\}$
11   end while
12   return $CB$
```

Based on these functions, we present a case-base mining algorithm in Table 3.

In this algorithm, T and M are the input parameters representing the case set and the size of the resulting case base, respectively. M can be determined by increasing its values and selecting the value for M that gives an optimal competence level. In Table 3, after performing FDA in Step 1, the *LGCM* algorithm first finds a case c with the largest projection onto a direction $\vec{\mathbf{w}}$ for each solution s_i in Step 5. These cases are added to the case base in Step 6. Then, we find subsequent cases whose diversity is the largest (Step 9). We add this case to the case base in Step 10. We continue until there are M cases selected or all the cases in T are selected.

4.3. Kernelized greedy case-base mining algorithm

In the previous section, we have proposed a case-base mining algorithm using FDA. The assumption of FDA is that data points belonging to different solutions can be separable in the original problem space. This assumption is not necessarily true in many applications. In that case, FDA cannot perform well in noise removal. To solve this problem, in this section we apply a nonlinear FDA algorithm, which is called Kernel Fisher Discriminant Analysis (KFDA) and is proposed in [29,35]. Our aim is that through the transformation on the feature space, cases with similar solutions can be brought closer together even when they do not appear close to each other in the original problem description.

The main idea of kernel methods [29,35] is to build feature space mappings indirectly, by performing dot products on problem vectors which can be computed efficiently in high-dimensional spaces. Examples of this type of algorithm are Support Vector Machines [45] and Kernel Principal Component Analysis [38]. Using this "kernel trick", instead of mapping the data explicitly, we can seek a formulation of the algorithm that uses only dot-products $\langle \phi(x), \phi(y) \rangle$ of the problem features, where the function ϕ maps an input vector to a high dimensional feature space. As we are then able to compute these dot-products efficiently we can solve the original problem without ever mapping explicitly to the possibly much larger feature space $\mathcal{F}$. This can be achieved via a kernel function $k(\vec{x}_1, \vec{x}_2) = \langle \phi(\vec{x}_1), \phi(\vec{x}_2) \rangle$. Possible choices of the $k(.,.)$ functions include the Gaussian (RBF) kernel, where $k(\vec{x}_1, \vec{x}_2) = \exp(-\|\vec{x}_1 - \vec{x}_2\|^2/\sigma^2)$, or the polynomial kernels, where $k(\vec{x}_1, \vec{x}_2) = \langle \vec{x}_1, \vec{x}_2 \rangle^d$, for some positive constants σ^2 and d, respectively.

We wish to apply the kernel feature transformation to case-base mining. Assume that we are given a training case set $T = \{c_1, c_2, \ldots, c_N\}$ where c_i denotes a case which is a pair of problem attributes and solution $(\vec{x}_i, s_i)$ where $\vec{x}_i \in \mathbb{R}^d$. KFDA first maps the attribute part of the cases into a feature space with a function ϕ. The resulting set is $\mathfrak{X} = \{\phi(\vec{x}_1), \phi(\vec{x}_2), \ldots, \phi(\vec{x}_N)\}$. Suppose that $\mathfrak{X}_i = \{\phi(\vec{x}^i_1), \phi(\vec{x}^i_2), \ldots, \phi(\vec{x}^i_{l_i})\}$ is the set of cases with the same solution s_i in the feature space. The centroid $\vec{\mathfrak{m}}_i$ of $\mathfrak{X}_i$ can be calculated as $\vec{\mathfrak{m}}_i = l_i^{-1} \sum_{j=1}^{l_i} \phi(\vec{x}^i_j)$, and l_i is the size of $\mathfrak{X}_i$. Then the total differences of the cases corresponding to the same solution s_i can be expressed as $\mathbb{S}_w = \sum_{i=1}^{l} \sum_{j=1}^{l_i} (\phi(\vec{x}^i_j) - \vec{\mathfrak{m}}_i)(\phi(\vec{x}^i_j) - \vec{\mathfrak{m}}_i)^{\mathsf{T}}$, and the sum of distances between any two centroids of different solutions can be expressed $\mathbb{S}_B = \sum_{i,j=1}^{l} (\vec{\mathfrak{m}}_i - \vec{\mathfrak{m}}_j)(\vec{\mathfrak{m}}_i - \vec{\mathfrak{m}}_j)^{\mathsf{T}}$. KFDA attempts to find a new direction $\vec{\mathbf{w}}$ such that the projection

Please cite this article in press as: R. Pan et al., Mining competent case bases for case-based reasoning, Artificial Intelligence (2007), doi:10.1016/j.artint.2007.04.018

Table 4
Kernel greedy case-base mining (KGCM) algorithm

```
    KGCM($T$,$M$)
    $T$ is the training case set, $M$ is the maximum size of case base to be mined.
1   Run KFDA on $T$ to find the direction $\vec{\mathbf{w}}$
2   $CB = \emptyset$
3   $U = T$
    % Steps 4 to 7 find the initial seeds for the case base.
4   for every $s_i \in S$, $T_i = \{(\vec{x}^i_j, s_i) \mid (\vec{x}^i_j, s_i) \in T\}$
5       Compute the projection $\mathbf{w}^i_j$ of each data record $\vec{x}^i_j$ onto $\vec{\mathbf{w}}$
6       Select the case $c$ with largest $\mathbf{w}^i_j$, $CB = CB \cup \{c\}$, $U = U - \{c\}$
7   end for
8   while $U \neq \emptyset$ and $|CB| < M$
9       Select the case $c$ with largest evaluation value: $Eval(CB \cup \{c\})$
10      $CB = CB \cup \{c\}$, $U = U - \{c\}$
11  end while
12  return $CB$
```

of $\mathbb{S}_B$ can be maximized while the projection of $\mathbb{S}_W$ can be minimized. That is, given a vector $\vec{\mathbf{w}}$, we need to maximize the following formula

$$J(\vec{\mathbf{w}}) = \frac{\vec{\mathbf{w}}^\mathsf{T}\mathbb{S}_B\vec{\mathbf{w}}}{\vec{\mathbf{w}}^\mathsf{T}\mathbb{S}_W\vec{\mathbf{w}}}. \tag{15}$$

By the substitutions of

$$\vec{\mathbf{w}}^\mathsf{T}\mathbb{S}_B\vec{\mathbf{w}} = \vec{\alpha}^\mathsf{T}\mathbf{M}\vec{\alpha}, \tag{16}$$

and

$$\vec{\mathbf{w}}^\mathsf{T}\mathbb{S}_W\vec{\mathbf{w}} = \vec{\alpha}^\mathsf{T}\mathbf{N}\vec{\alpha}, \tag{17}$$

where $\mathbf{M} = (M_1 - M_2)(M_1 - M_2)^\mathsf{T}$, $(M_i)_j = \frac{1}{l_i}\sum_m^{l_i} k(x_j, x^i_m)$, $\mathbf{N} = \sum_{j=1,2} K_j(I - \mathbf{1}_{l_j})K_j^\mathsf{T}$, K_j is a $l \times l_j$ matrix with $(K_j)_{nm} = k(x_n, x^j_m)$, and I is the identity and $\mathbf{1}_{l_j}$ the matrix with all entries being $1/l_j$ [29], maximizing Eq. (15) is equivalent to maximizing

$$J(\vec{\alpha}) = \frac{\vec{\alpha}^\mathsf{T}\mathbf{M}\vec{\alpha}}{\vec{\alpha}^\mathsf{T}\mathbf{N}\vec{\alpha}}. \tag{18}$$

More computational details of KFDA can found in [29,35].

A kernelized case-base mining algorithm is given in Table 4. In this algorithm, we replace the FDA in LGCM with KFDA. The computational complexity of KGCM can be analyzed as follows. The cost of the KFDA computation is O(N^3) (N is the size of the training case set) [28]. Recently, more efficient algorithms are proposed [36,51]. The time complexity of [36,51] is O($N * d * l$), where d is the dimension of a training case set, and l is the number of solution categories. The time complexity for the rest of the KGCM algorithm is O($N * M$). In contrast, the time complexity of LGCM is O($d^3 + N * d * l + N * M$), which is smaller than that of KGCM when the dimensionality of the training case set is smaller than the size of the training case set.

4.4. An illustrative example

We now consider an example to illustrate our case-base mining algorithm KGCM. We use a case set that is an artificially generated spiral case set used to illustrate the effect of a nonlinear case base, where the raw case set is adapted from [14]. The solution set of this case set consists of two categories: positive and negative. The case set comprises 1000 cases in each category. In this example, we wish to extract a subset of M cases as a case base which can accurately classify a test case set into either a positive or a negative category. Therefore, for this example, the task for a CBR system is to classify problem feature vector into one of two classes, positive and negative.

We perform the KGCM algorithm on this case set (see Fig. 2), where "oripos" means the original data that belong to the solution "+1", and "orineg" means the data associated with the solution category "−1". Similarly, "poscase"

Please cite this article in press as: R. Pan et al., Mining competent case bases for case-based reasoning, Artificial Intelligence (2007), doi:10.1016/j.artint.2007.04.018

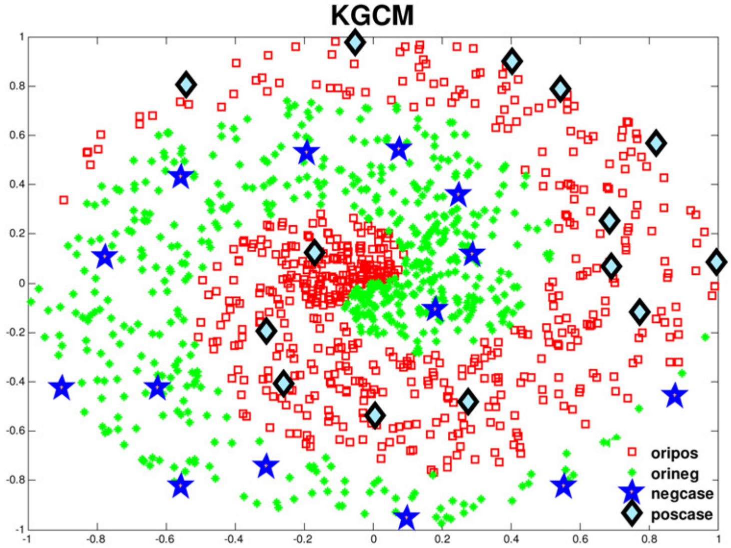

Fig. 2. Spiral data with 14 cases found for each solution by the KGCM algorithm.

corresponds to the data points that are associated with the solution category "+1", and similarly for solution "−1". Here, we use a Gaussian kernel with the $\sigma^2 = 4$. This example shows that when we mine cases from a training case set, we need to consider both the diversity and the representativeness of the cases. First, we can see that this problem space is nonlinear in nature, such that any naive method for selecting representative cases, such as a random selection method or a simple selection of cases on either side of the chart, may fail to find a good case base. Second, although the training case set in the original space is nonlinear, it can be transformed into a linearly separable space by means of a kernel transformation. In the feature space, it is then possible to perform a linear separation of the cases. Finally, when we select cases for this problem, we should avoid selecting cases that belong to a single region; instead, we should choose cases by maximizing the global diversity. The resultant case base shown in the figure, which consists of the stars and diamonds, are the output of the KGCM algorithm when the case base size $M = 14$. As we can see, this case base is of high quality due to the representativeness of the cases and the diversity of their distribution.

5. Experimental analysis

We wish to show that KGCM benefits from superior effectiveness and performance in terms of competence as compared to previous algorithms for case-base mining. We test this claim on a number of case bases in a comparison with case-deletion, case-addition, CNN-FP, COV-FP, RFC-FP, RC-FP and ICF algorithms. Our experiments are performed on a synthetic case set [14] and several publicly available case sets from UCI and Delve Data Repositories [5,33]. These case sets, covering a wide variety of problem domains, have been used as benchmark data. The ones that we use in our experiments include: `adult`, `australian`, `balance`, `banding`, `cars`, `chess`, `cleve`, `crx`, `diabetes`, `flare`, `german`, `heart`, `hepatitis`, `hungarian`, `hypothyroid`, `iris`, `led24`, `liver-disorder`, `monk1`, `monk2`, `monk3`, `mushroom`, `sick-euthyroid`, `sleep`, `tic-tac-toe`, `tokyo`, `vehicle`, `vote`, `waveform`, `wine`, `wdbc`, `optdigits`, `pendigits`, `shuttle`, `satellite`, `ionosphere`, `scale`, `sonar`, `ringnorm`, `twonorm`, and `spiral`.[2]

In all experiments we compare the competence of the case base against the *size ratio* of the mined case base to the training case set; we call this percentage the "ratio of case base size" in the experiments. Let X_t be a test set, $A_{\text{new}}(X_t)$

[2] The first 38 case sets are from the UCI machine learning repository [5]. The `ringnorm` and `twonorm` are case sets of Delve [33]. The `spiral` case set is from the classification toolbox in [14].

Please cite this article in press as: R. Pan et al., Mining competent case bases for case-based reasoning, Artificial Intelligence (2007), doi:10.1016/j.artint.2007.04.018

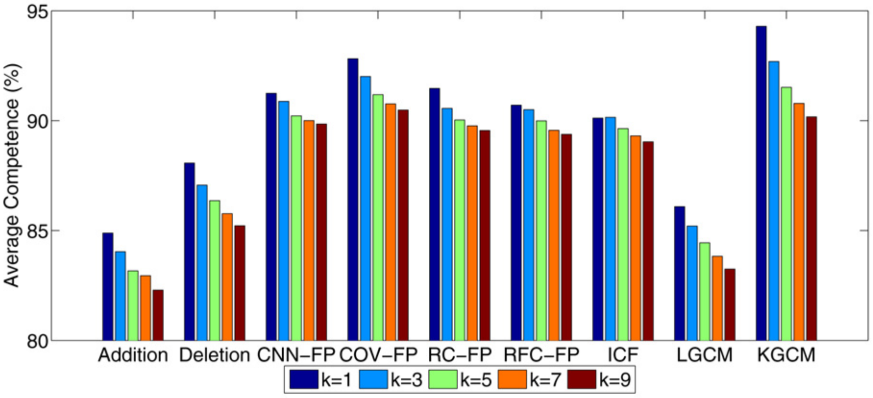

Fig. 3. Comparing the competence of case mining algorithms under different values of k in k-NN search.

be the percentage of new problems among the problem set X_t that can be solved by our case base, and $A_{\text{original}}(X_t)$ be that of the training case set. We define the competence of the new case base as

$$Comp(X_t) = \frac{A_{\text{new}}(X_t)}{A_{\text{original}}(X_t)}. \tag{19}$$

Note that in this definition, the competence $Comp(X_t)$ depends on the test set X_t. Note also that it is possible for the competence to be greater than 100%, in which case the case base found by an algorithm is better at solving future problems than the original case set. Section 5.2 presents the comparisons between KGCM and case-deletion (footprint deletion policy FD [40]) and case-addition [54] algorithms. And Section 5.3 presents the comparison between KGCM and CNN-FP, COV-FP, RFC-FP, RC-FP and ICF.

5.1. Impact of the number of nearest neighbors

The computation of competence depends on the search for similar cases in both the original case set and the mined case base. The number of nearest neighbors k is an important parameter in this computation. We wish to determine a proper value of k for the rest of the experiments. To do this, we run experiments on eight UCI case sets (*pendigits*, *satellite*, *sonar*, *scale*, *optdigits*, *wdbc*, *ionosphere*, *shuttle*), while varying k from 1 to 9 on every other value. For each case set, we randomly split the case set into 2/3 for training and 1/3 for testing, and we repeat the process 20 times. In each round, we vary the ratio of the case base size to the original case set size at 1, 3, 5, 10, 15, 20 and 25% for every fixed value of k. Fig. 3 shows the experimental results. We can see from this figure that the average competence when $k = 1$ is higher than other values of k for all algorithms. We believe that when the case base is sparser than the original case set, the distances between a new case and similar candidate solution cases when $k > 1$ can be much larger than that of $k = 1$. Therefore, we are able to obtain the highest competence when we use a one-nearest neighbor algorithm. In the remaining experiments in this paper, we will set $k = 1$ for the k-nearest neighbor search.

5.2. KGCM vs. case-deletion and case-addition algorithms

For each application domain, we validate our KGCM algorithm with 1-NN as the underlying CBR similarity retrieval function. In every experiment, we use 1-NN to classify the testing data with all the training data to obtain a baseline. To test the proposed algorithms, we apply three case-base mining algorithms that we discussed before on the training data and validate the mined case bases on the testing data. These three case-base mining algorithms are: our KGCM, LGCM and case deletion algorithms using the footprint deletion policy FD [40], where the sizes of the final case bases, varied in different comparisons, are set to be equal in each comparison. LGCM applies the FDA without using kernels, which serves as a baseline because we wish to test the effectiveness of using kernels separately.

In these experiments, cross validation is performed by randomly partitioning the raw case set into a training case set (2/3) and test case set (1/3). We first use each of the three algorithms to mine the case bases from the training

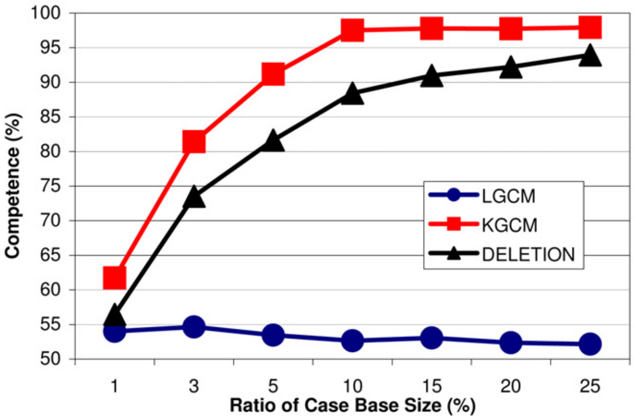

Fig. 4. Spiral case set: Competence of the case bases mined by KGCM and other case-base mining algorithms using 1-NN.

case set. Then we validate the mined cases on the test case set. We repeat this training and test process 20 times, and compute the average results as the performance of the algorithm.

For the `spiral` raw case set, we split the original 2000 raw cases into 1200 cases as the training case set and 800 cases as the test case set. Fig. 4 shows the comparison result. In the figure, we compare the case-base mining algorithms on the accuracy obtained from an 1-NN classifier. The experimental result indicates that for the classification problem, the KGCM algorithm has higher competence than the LGCM algorithm and case-deletion algorithm in the original space. We attribute this success to the result of using the diversity measure and the introduction of nonlinear transformation using Gaussian kernels. As the ratio of case base size increases from 1% to 5%, we can see from Fig. 4 that KGCM rises most rapidly towards the optimal competence level. Comparatively, the other two algorithms are rather slow to respond. When they all reach 25% in size ratio, the difference between the KGCM and case-deletion policy is very small.

In Section 2, we mentioned that the case-deletion policy may be sensitive to noise. In the following experiment, we compare the competence of the `spiral` case set against the percentage of the amount of noise added to the training case set; we call this percentage the "noise ratio" in this experiment. Fig. 5 shows the comparative performance between the case-deletion policy and the KGCM algorithm when there is some noise in the training case set. The figure shows the competence curves over different noise ratio values for the KGCM algorithm and case-deletion policy. We can see that the KGCM algorithm is more robust than the case-deletion policy when noise increases.

To further validate the KGCM algorithm, we conduct experiments on eight UCI case sets to compare the KGCM algorithm, case deletion policy (FD [40]), case addition policy [54] and LGCM (see Fig. 6). In most experiments, the KGCM algorithm has higher competence than the case-deletion policy. One reason for this high performance is that by applying the KFDA, we can find the most important features for the determination of case coverage, rather than using the full set of features given in the input. This makes it possible to focus on the most important features of the case in mining the case base.

Furthermore, when the coverage of each case is relatively small, the reachability of all cases are small. However, after a feature transformation, the coverage of cases can become much larger. This essentially makes it possible to use a smaller number of cases to achieve the same level competence after the transformation. In the experiments, we also compare the KGCM and LGCM algorithms. We also find that KGCM outperforms LGCM, which implies that using kernels in the case-base mining algorithm is important for most of these problems. In the following experiments, we only compare KGCM with the previous algorithms.

In addition, we conducted 32 more experiments on most UCI and Delve case sets that can be solved using k-NN algorithms, to compare the KGCM algorithm, case deletion policy and the case addition policy (see Figs. 7–10). Again, KGCM wins in all 32 domains under the tests. We notice that the case-addition algorithm performs poorly in some of the domains, which can be attributed to the fact that it only considers case-coverage as a quality measure, and does not consider case diversity as in KGCM.

In the experiments in Figs. 6–10, we also find another interesting phenomenon where the competence of KGCM sometimes increased above 100%. For instance, in the experiments on the `scale` and `wdbc` case sets (Fig. 6), we find

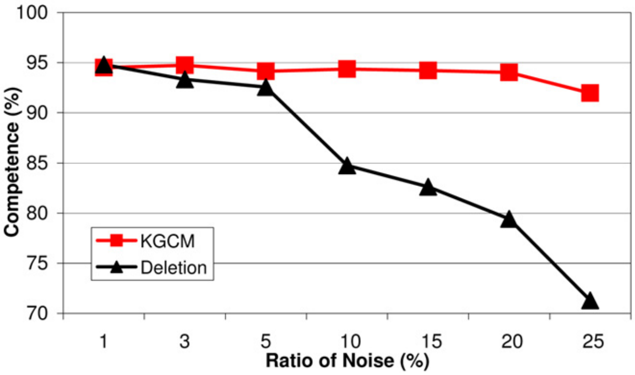

Fig. 5. Competence trends of KGCM and case-deletion policy with different noise ratio.

that the competence of KGCM increased above 100% when the ratio of case base size reached 3% and 25% respectively. This also happens in Figs. 7–10, including the `adult`, `australian`, `cleve`, `crx`, `diabetes`, `flare`, `hepatitis`, `hungarian`, `german`, `heart`, `hypothyroid`, `iris`, `led24`, `monk2`, `sick-euthyroid`, `tokyo`, `sleep`, `twonorm`, `vote`, and `waveform` case sets. However, this happens less frequently for case deletion and addition policies. This shows that, in some situations, the competence of the case base generated by KGCM algorithm are indeed of higher competence than the other algorithms.

5.3. KGCM versus CNN-FP, COV-FP, RFC-FP, RC-FP and ICF

In this section, we compare the KGCM algorithm with the classical case-base editing methods such as CNN-FP, COV-FP, RFC-FP, RC-FP [25,26,43] and ICF [7], which are reviewed in Section 2. Our experiments are performed on 24 publicly available raw case sets from the UCI Data Repository.

For each application domain, we validate every algorithm with 1-NN as the underlying similarity-retrieval function. In all the experiments, we randomly separated the raw case set into the training case set (2/3) and test case set (1/3). We first use the algorithm to mine the cases from a training case set. Then we validate the mined cases on the test data. We repeat this training and testing process 10 times, and compute the average result as the performance of the algorithm.

As mentioned in Section 2, CNN-FP is an order-sensitive approach. The algorithms COV-FP, RFC-FP and RC-FP represent three different case-sorting criteria in which important cases are considered before the less important ones. In other words, the earlier a case is selected by the case-base mining algorithm, the more important this case is. In Figs. 11–13, we compare KGCM with the other three algorithms when the ratio of case base size increases from 1% to 25%, respectively. In these figures, the notation 1% denotes the top 1% of the most important cases in the training case set. We find that KGCM outperforms the other three algorithms on many case sets.

Unlike the four algorithms mentioned above, the ICF algorithm uses the case-deletion policy. ICF removes the cases whose reachability size is greater than that of coverage. With this deletion policy, ICF can remove many low-quality cases efficiently and keep a few high quality cases as the resulting case base. Under this stop criterion of the ICF algorithm, it is difficult to control the ratio of case base size from 1% to 25% as in above experiments. In fact, in our experiments on ICF, it will stop at very different points of case base size (see Table 5). In the following experiments, we only compare the competence of KGCM and ICF at the same level of case base size. We conduct experiments to compare KGCM with ICF over the same 24 case sets (see Table 5). We find that KGCM outperforms the other algorithms over 23 case sets. We also performed a t-test to assess whether the mean values of expected competence in the test case set that are discovered by different algorithms are statistically significantly different from each other. We find that the mean of competence over the test case set based on ICF is 83.56 while the value based on KGCM is 100.62. With a t-value as 3.78 and a confidence level being 99.5%, this indicates that the difference in the mean values are significant. Furthermore, the KGCM algorithm results in higher competence in all the case sets that we experimented on even when the case-base sizes are very small. In contrast, ICF only has very low performance in most case sets. This shows that KGCM is more robust and steady as compared to the other algorithms.

Please cite this article in press as: R. Pan et al., Mining competent case bases for case-based reasoning, Artificial Intelligence (2007), doi:10.1016/j.artint.2007.04.018

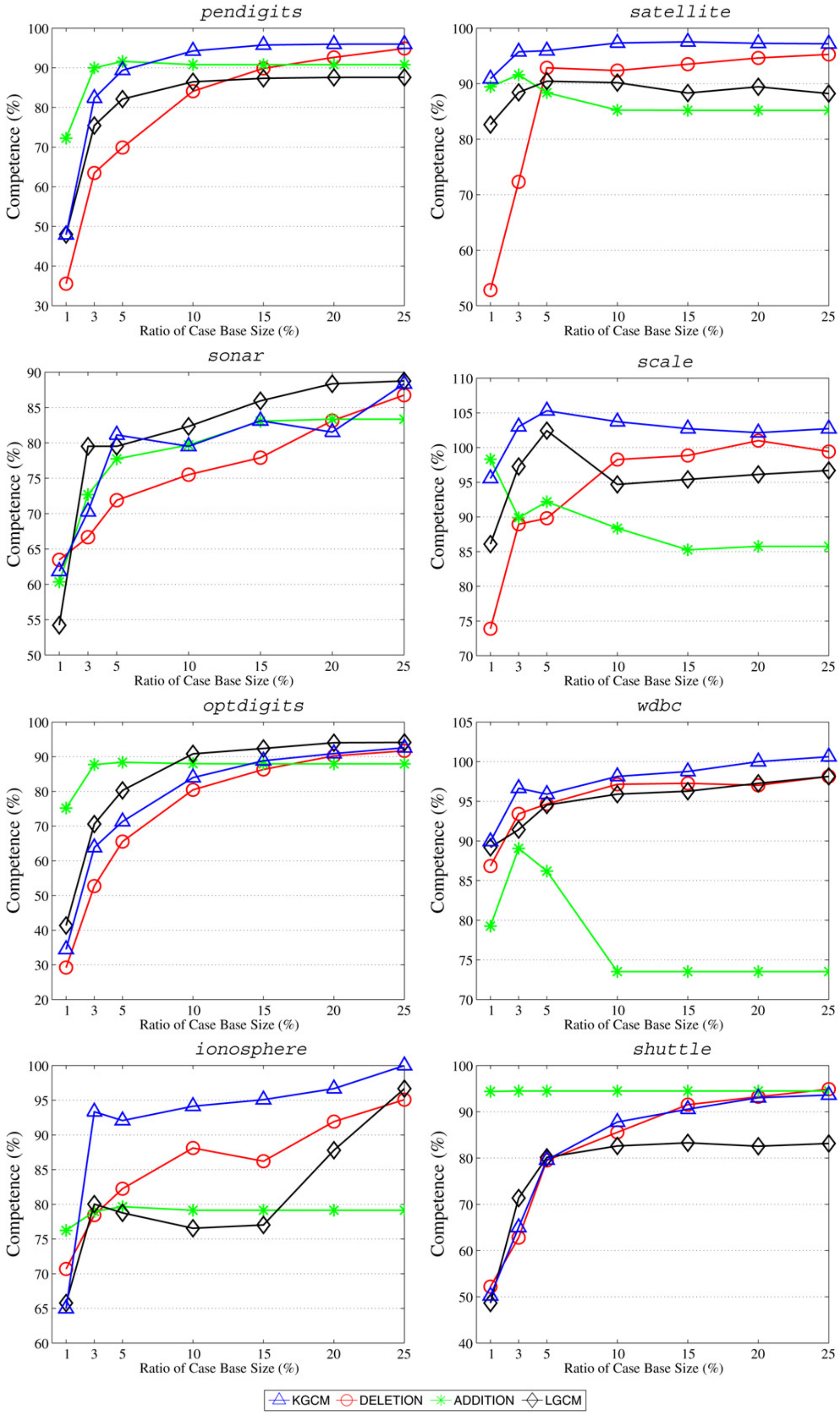

Fig. 6. Competence of the case bases mined by KGCM and other case-base mining algorithms using 1-NN.

Please cite this article in press as: R. Pan et al., Mining competent case bases for case-based reasoning, Artificial Intelligence (2007), doi:10.1016/j.artint.2007.04.018

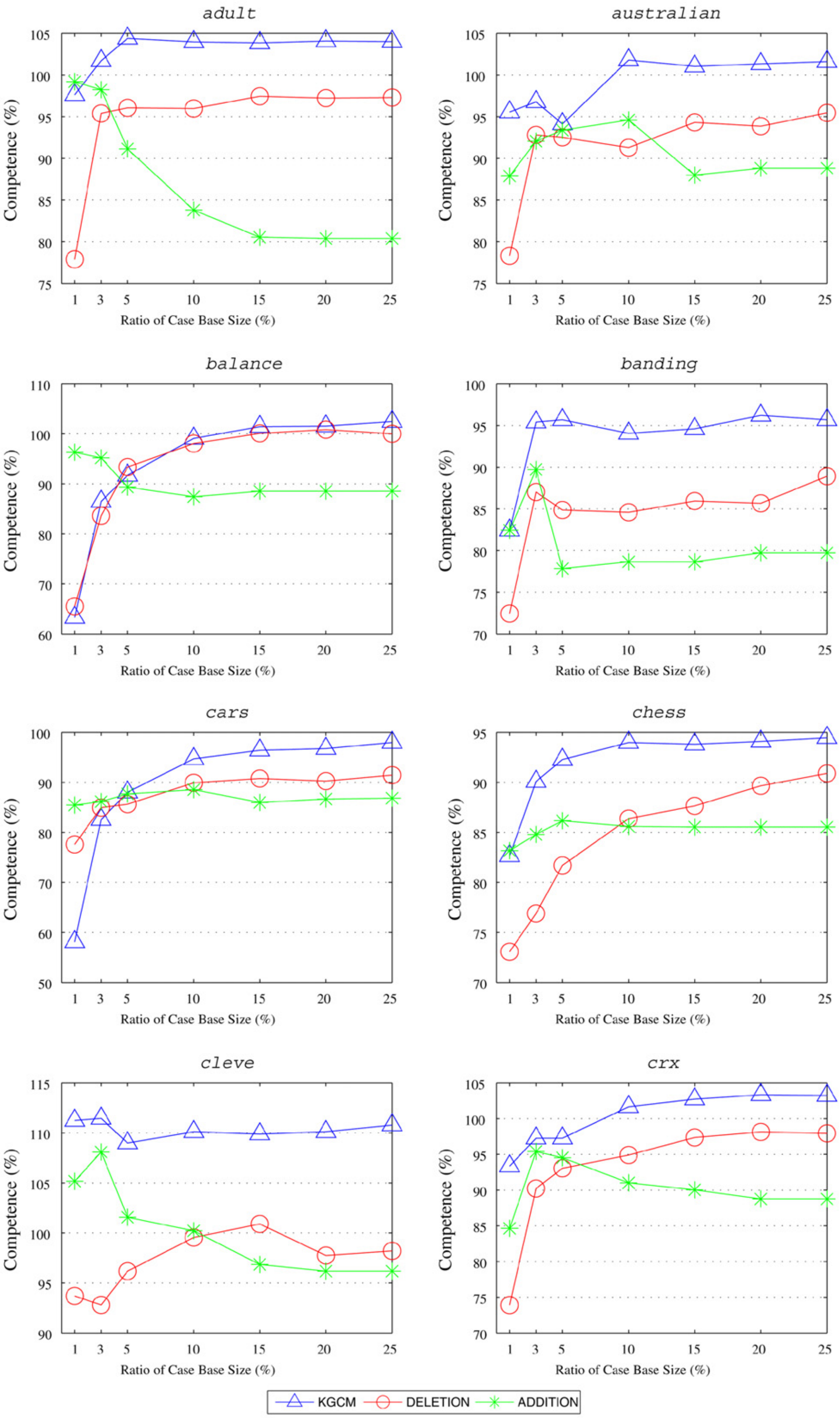

Fig. 7. Competence on the case base mined by KGCM and other case-base mining algorithms using 1-NN. (Part I.)

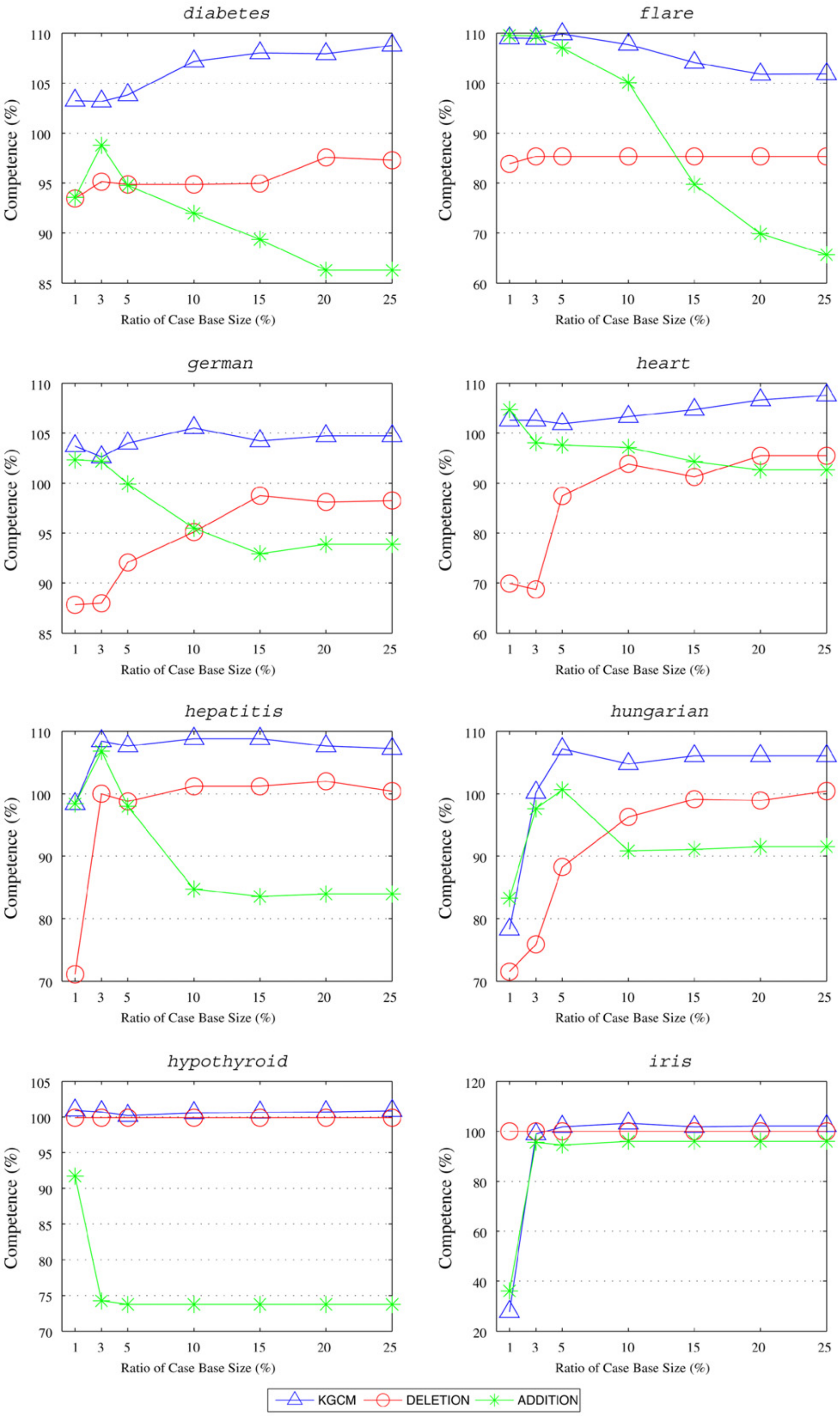

Fig. 8. Competence on the case base mined by KGCM and other case-base mining algorithms using 1-NN. (Part II.)

Please cite this article in press as: R. Pan et al., Mining competent case bases for case-based reasoning, Artificial Intelligence (2007), doi:10.1016/j.artint.2007.04.018

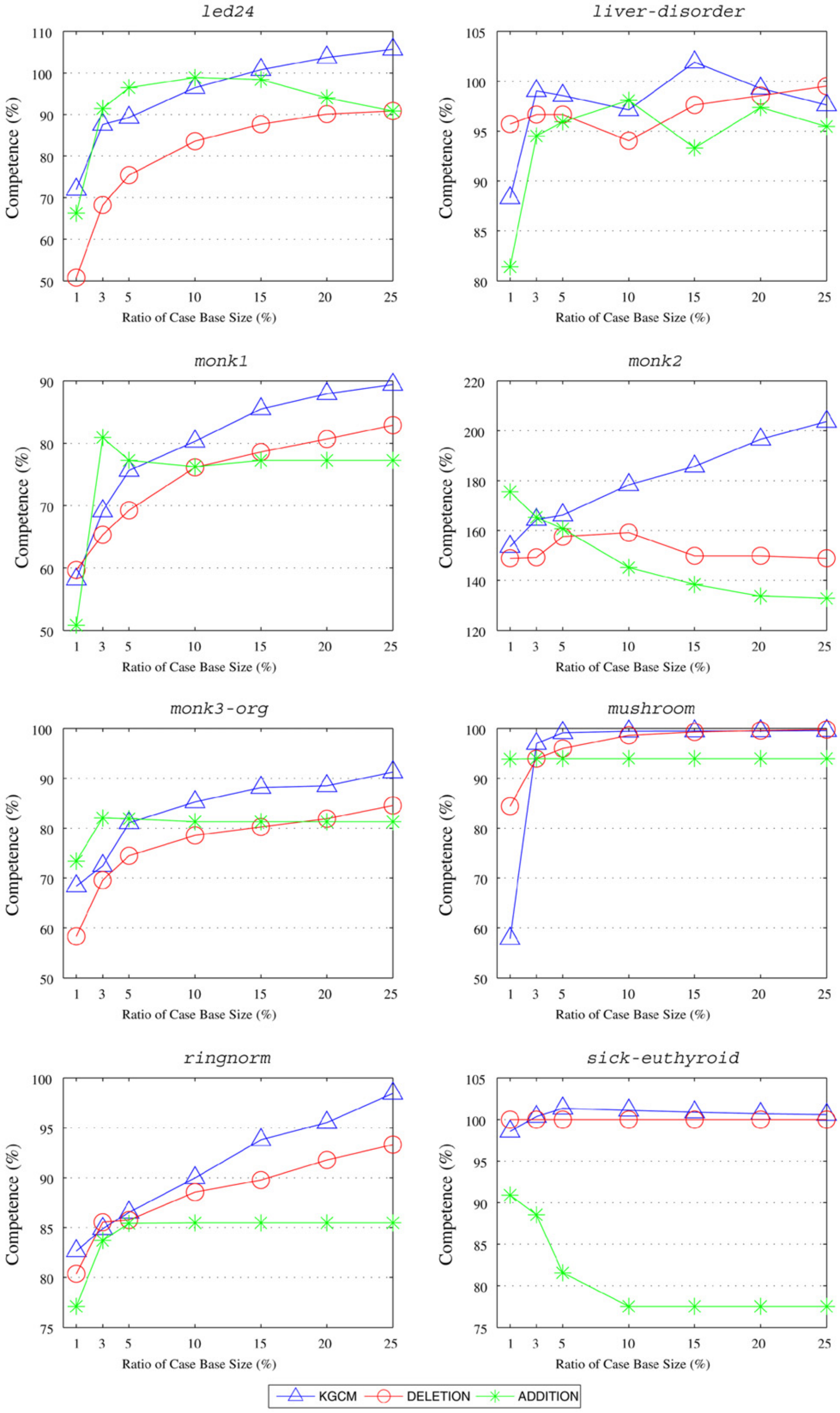

Fig. 9. Competence on the case base mined by KGCM and other case-base mining algorithms using 1-NN. (Part III.)

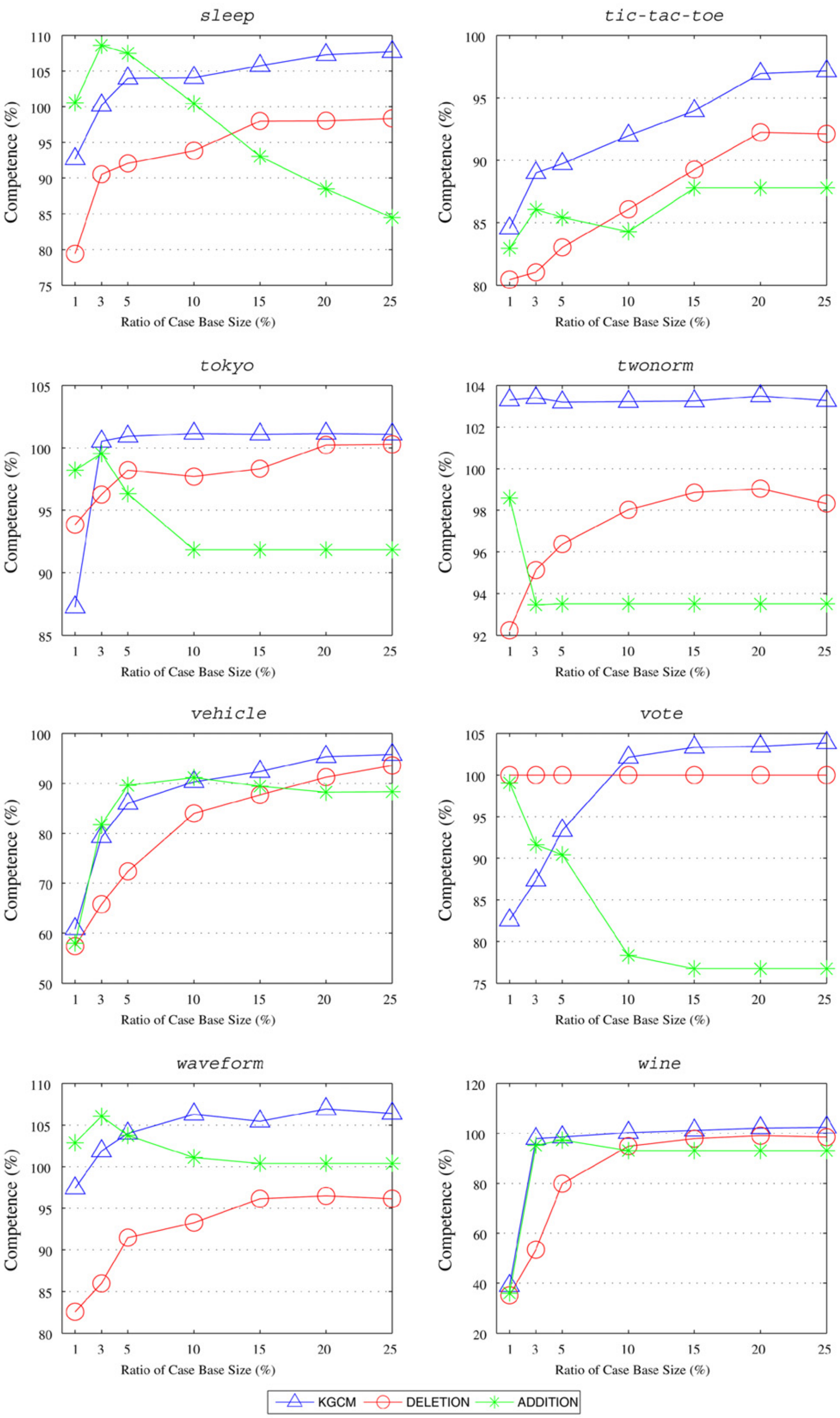

Fig. 10. Competence on the case base mined by KGCM and other case-base mining algorithms using 1-NN. (Part IV.)

Please cite this article in press as: R. Pan et al., Mining competent case bases for case-based reasoning, Artificial Intelligence (2007), doi:10.1016/j.artint.2007.04.018

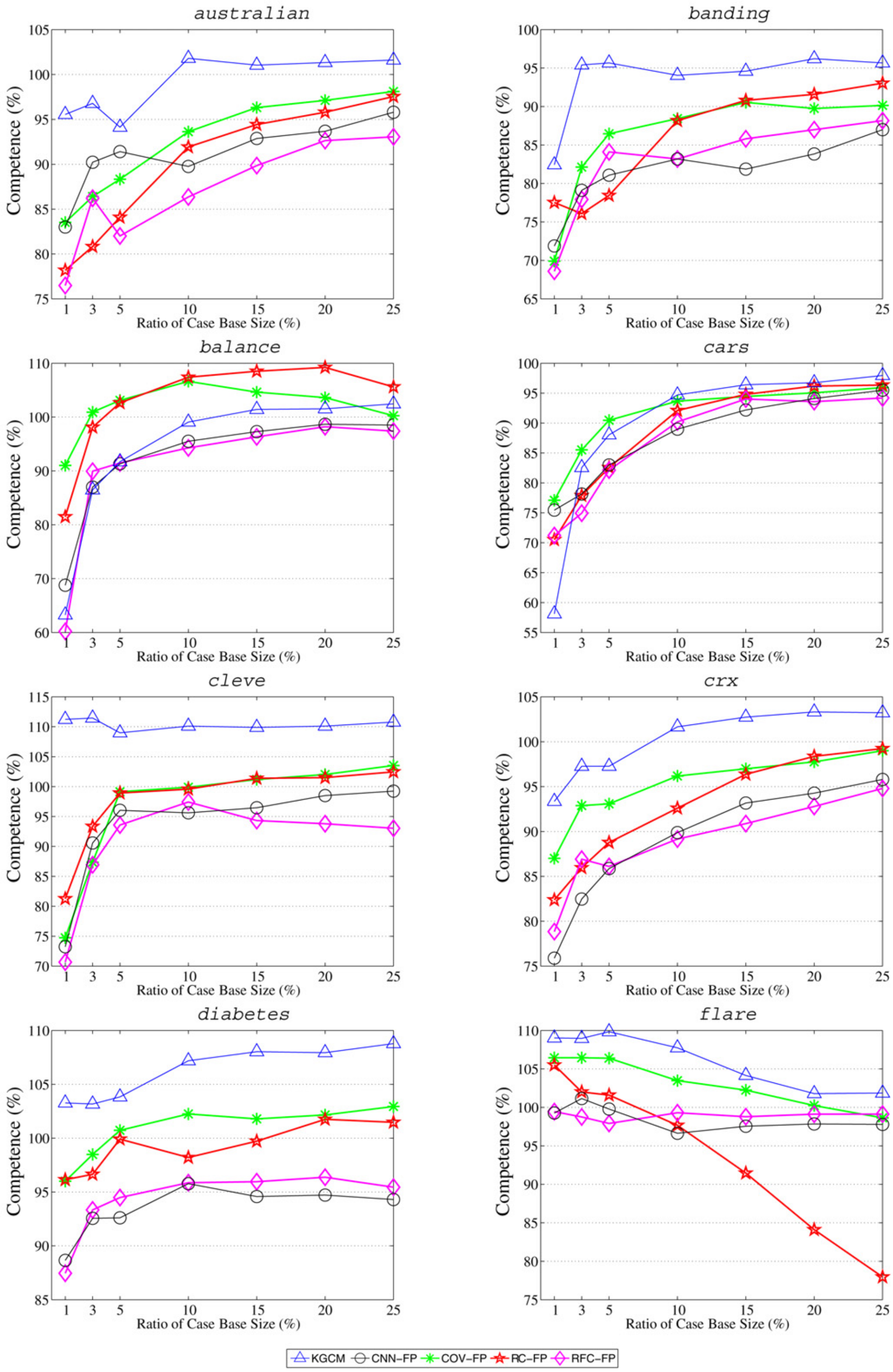

Fig. 11. Competence on the case base mined by KGCM, CNN-FP, COV-FP, RC-FP and RFC-FP using 1-NN. (Part I.)

Please cite this article in press as: R. Pan et al., Mining competent case bases for case-based reasoning, Artificial Intelligence (2007), doi:10.1016/j.artint.2007.04.018

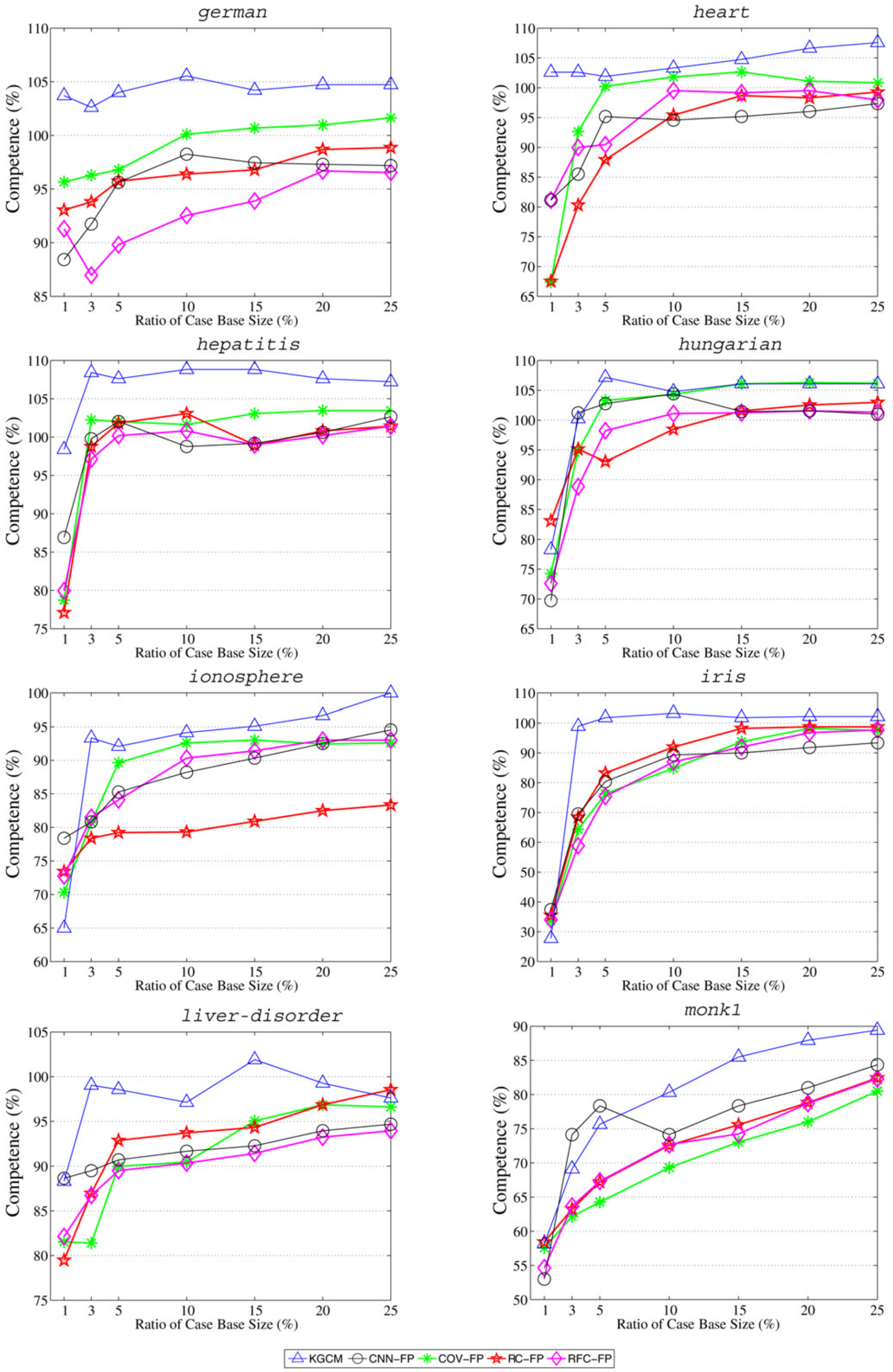

Fig. 12. Competence on the case base mined by KGCM, CNN-FP, COV-FP, RC-FP and RFC-FP using 1-NN. (Part II.)

Please cite this article in press as: R. Pan et al., Mining competent case bases for case-based reasoning, Artificial Intelligence (2007), doi:10.1016/j.artint.2007.04.018

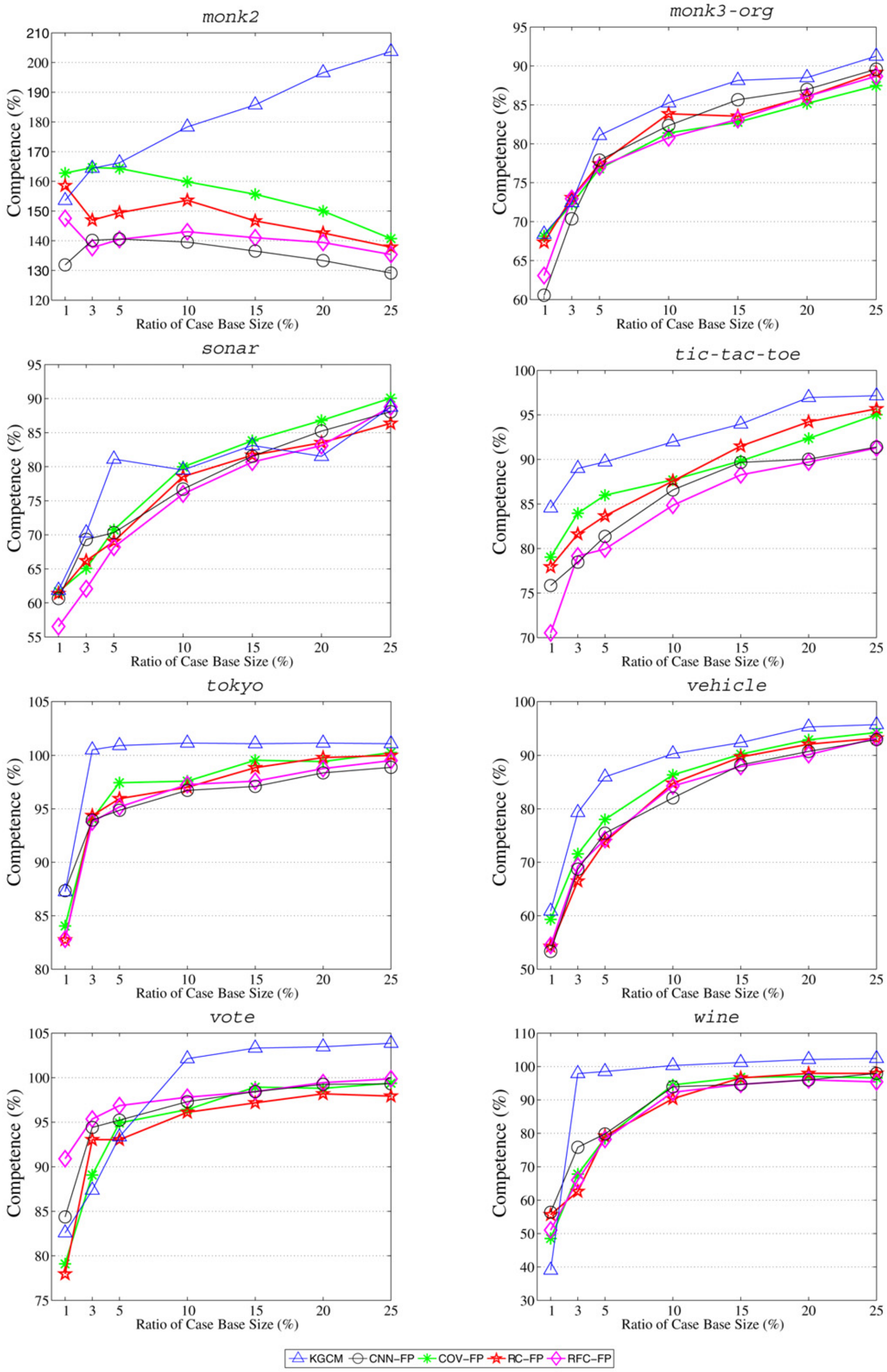

Fig. 13. Competence on the case base mined by KGCM, CNN-FP, COV-FP, RC-FP and RFC-FP using 1-NN. (Part III.)

Table 5
Competence and ratio of case base size for each domain. The value in parentheses denotes the ratio of case base size and the value outside the parentheses denotes the competence value of the resulting case base

Domain	ICF	KGCM
`australian`	75.19(5.39)	97.58(5.00)
`balance`	106.43(63.89)	102.46(25.00)
`banding`	70.53(12.80)	94.05(10.00)
`car`	82.93(14.53)	101.98(15.00)
`cleve`	87.53(5.55)	108.99(5.00)
`crx`	67.41(3.26)	97.26(3.00)
`diabetes`	83.88(3.60)	103.17(3.00)
`flare`	83.63(13.53)	104.15(15.00)
`german`	81.90(5.37)	104.01(5.00)
`heart`	87.85(5.86)	105.99(5.00)
`hepatitis`	98.34(4.95)	107.63(5.00)
`hungarian`	92.95(7.46)	104.78(10.00)
`ionosphere`	66.92(4.12)	92.08(5.00)
`iris`	88.73(12.67)	95.93(10.00)
`liver-disorder`	86.65(11.16)	97.14(10.00)
`monk1`	73.86(13.54)	85.486(15.00)
`monk2`	138.91(3.04)	164.4(3.00)
`monk3-org`	75.16(4.54)	81.078(5.00)
`sonar`	72.07(11.28)	82.33(10.00)
`tic-tac-toe`	79.82(4.19)	89.72(5.00)
`tokyo`	61.10(4.88)	100.92(5.00)
`vehicle`	85.62(16.01)	92.38(15.00)
`vote`	74.10(2.30)	102.78(3.00)
`wine`	84.06(6.45)	98.50(5.00)
Mean	*83.56*	*100.62*

5.4. Summary

We now summarize and explain the experimental results.

- We can see from our experiments that the competence of case bases computed by the KGCM algorithm generally increases with the case base sizes. In particular, while all algorithms perform increasingly better for larger case bases, KGCM generally performs the best among all baselines. An exception is the `flare` domain (see Fig. 8). In this domain, the competence of all algorithms that we tested tends to decrease when the case base size increases. We believe that this can be attributed to the fact that this problem domain contains much noise to start with, which makes it difficult to use a subset of cases to represent all cases. However, we can also observe that even in this domain, KGCM still outperforms the rest.
- For the `balance` domain (see Fig. 11), the algorithms RC-FP and COV-FP outperformed the KGCM algorithm. These two algorithms use the *relative coverage* and *coverage* (see Definitions 3 and 1) to evaluate the importance of each case. In contrast, KGCM uses a heuristic, which is the distance between a case and the class boundary, to select cases. We believe that this heuristic works better than many existing approaches, but as it is a heuristic, there may be situations when cases that are far from the boundary may not correspond to the high quality cases. In such a situation, RC-FP and COV-FP work better than KGCM , as in the `balance` domain.
- We can also observe the fact that sometimes, the competence of KGCM reaches above 100%. For instance, in the experiments on the `scale` and `wdbc` (Fig. 6) case sets, we find that the competence of KGCM increased above 100% when the ratio of case base size reached 3% and 25%, respectively. This also happens in the figures from Fig. 7 to Fig. 13, including the case sets `adult`, `australian`, `cleve`, `crx`, `diabetes`, `flare`, `hepatitis`, `hungarian`, `german`, `heart`, `hypothyroid`, `iris`, `led24`, `monk2`, `sick-euthyroid`, `tokyo`, `sleep`, `twonorm`, `vote`, and `waveform`. This can be explained by the fact that in these domains, the cases that are selected by the KGCM algorithm are indeed of very high quality and do not contain much noise. However, this happens less frequently for the other algorithms including some cited case deletion and case addition algorithms.

- Comparing with the LGCM algorithm, the time complexity of KGCM is higher. However, we can see that KGCM outperforms LGCM in terms of case-base competence, as shown in Figs. 3, 4 and 6. We attribute this improvement to the usage of kernels.

6. Conclusions and future work

In this paper we presented a new case-base mining framework that includes both the theory and the KGCM algorithm for mining a new case base from the raw case set. Our theorem provides guidance on how to select new cases. We have shown that it is better to select cases with high diversity after performing the KFDA-based feature selection operation. In our KGCM algorithm, we convert the problem feature vector set of a raw case set into a feature space through kernel transformation. Optimal feature selection is done in this space through a correlation analysis, and the best cases are chosen based on a global diversity constraint and a maximal correlation constraint. Although the time complexity of KGCM is slightly higher than that of LGCM, its performance is better. We also tested our KGCM algorithm on a large number of raw case sets against many previous algorithms.

In the future we wish to extend this work to include other kernel methods for mining of the case bases. Another important future work is to automatically find the best number of cases for a case base. This number should be controlled by the distribution of the raw case set. Determining the best value for the size of the case base might be an empirical question that is similar to the issue of determining the optimal number of clusters in a large data set. Finally, some case-based reasoning problems are inherently disjunctive in nature, where a concept can be defined according to different data groups or because of concept drift. This is the case with spam email detection, for which some researchers have shown that CBR methods outperform some traditional machine learning methods such as Naive Bayes [12]. We can extend KGCM to handle the disjunctive concepts by first performing a clustering operation on positive cases that correspond to spam emails, and then perform multi-class KFDA-based feature selection on each cluster. We plan to explore this issue in the future.

Appendix A

Proof of Theorem 3.1. According to Eq. (1), we have

$$\begin{aligned}
&E_{\vec{x},s}\big[L\big(f(\vec{x};CB,k,\Theta),\vec{\mathfrak{s}}\big)\big]\\
&\quad = E_{\vec{x},\vec{\mathfrak{s}}}\big[\big\|f(\vec{x};CB,k,\Theta)-\vec{\mathfrak{s}}\big\|^2\big]\\
&\quad = E_{\vec{x},\vec{\mathfrak{s}}}\big[\big\|f(\vec{x};CB,k,\Theta)-E_{\vec{\mathfrak{s}}|\vec{x}}[\vec{\mathfrak{s}}]+E_{\vec{\mathfrak{s}}|\vec{x}}[\vec{\mathfrak{s}}]-\vec{\mathfrak{s}}\big\|^2\big]\\
&\quad = E_{\vec{x}}\big[\big\|f(\vec{x};CB,k,\Theta)-E_{\vec{\mathfrak{s}}|\vec{x}}[\vec{\mathfrak{s}}]\big\|^2\big]+E_{\vec{x}}\big[\big\|E_{\vec{\mathfrak{s}}|\vec{x}}[\vec{\mathfrak{s}}]-\vec{\mathfrak{s}}\big\|^2\big],
\end{aligned} \tag{20}$$

where $E_{\vec{\mathfrak{s}}|\vec{x}}[\vec{\mathfrak{s}}]$ refers to the expectation taken with respect to all possible values of $\vec{\mathfrak{s}}$, weighted by their probabilities given $\vec{x}$.

$$\begin{aligned}
&E_{\vec{x}}\big[\big\|f(\vec{x};CB,k,\omega)-E_{\vec{\mathfrak{s}}|\vec{x}}[\vec{\mathfrak{s}}]\big\|^2\big]\\
&\quad = E_{\vec{x}}\left[\left\|\frac{1}{\sum_{i=1}^{k}\omega_i}\sum_{c_i\in\mathrm{CRS}(\vec{x};CB,k,\Theta)}\omega_i f\big(\vec{x};\{c_i\},1,\Theta\big)-E_{\vec{\mathfrak{s}}|\vec{x}}[\vec{\mathfrak{s}}]\right\|^2\right]\\
&\qquad \text{(in light of the definition of } g(\vec{x};c_i) \text{ in the theorem)}\\
&\quad = E_{\vec{x}}\left[\left\|\sum_{c_i\in\mathrm{CRS}(\vec{x};CB,k,\Theta)}g(\vec{x};c_i)\right\|^2\right]\\
&\quad = E_{\vec{x}}\left[\sum_{c_i\in\mathrm{CRS}(\vec{x};CB,k,\Theta)}\big\|g(\vec{x};c_i)\big\|^2+2\sum_{\substack{c_i,c_j\in\mathrm{CRS}(\vec{x};CB,k,\Theta)\\ i\neq j}}\big\langle g(\vec{x};c_i),g(\vec{x};c_j)\big\rangle\right]\\
&\quad = E_{\vec{x}}\left[\sum_{c_i\in\mathrm{CRS}(\vec{x};CB,k,\Theta)}\big\|g(\vec{x};c_i)\big\|^2\right]+2E_{\vec{x}}\left[\sum_{\substack{c_i,c_j\in\mathrm{CRS}(\vec{x};CB,k,\Theta)\\ i\neq j}}\big\langle g(\vec{x};c_i),g(\vec{x};c_j)\big\rangle\right]
\end{aligned}$$

Please cite this article in press as: R. Pan et al., Mining competent case bases for case-based reasoning, Artificial Intelligence (2007), doi:10.1016/j.artint.2007.04.018

(in light of Definition 8)

$$= \sum_{i=1}^{M} E_{\vec{x}\sim \mathrm{CCS}(c_i)}\Bigg[\big\|g(\vec{x};c_i)\big\|^2 + \sum_{\substack{c_j\in\mathrm{CRS}(\vec{x};CB,k,\Theta)\\ i\neq j}} \big\langle g(\vec{x};c_i), g(\vec{x};c_j)\big\rangle\Bigg]. \tag{21}$$

Then after plugging Eq. (21) into Eq. (20), we get the result of the theorem. □

Appendix B

The following is a complete list of the terms and notations used in this paper.

Notation	Meaning
T	Raw case set
CB	Case base
T	A general case set, which can refer to T or CB
X	Problem set
$\mathfrak{X}$	Transformed problem set
S	Solution set
$\vec{x}$	Problem feature vector
c	A case, $c = (\vec{x}, s)$
$\vec{\mathfrak{s}}$	Solution vector of $\vec{x}$
N	Size of T
M	Size of the CB
l	Size of the solution set
d	Dimension of problem feature vector $\vec{x}$
$E_{\vec{x},\vec{\mathfrak{s}}}[\cdot]$	Expectation over $\vec{x}$ and $\vec{\mathfrak{s}}$ with respect to their probabilities
CRS	Candidate Reachability Set of a Problem
CCS	Candidate Coverage Set of a Case
Coverage	See Definition 1
Reachability	See Definition 2
GroupCoverage	See Definition 4
RelativeCoverage	See Definition 3
ϕ	A transformation of problem feature vector
$\mathfrak{X}_i$	The set of cases with the same solution s_i in the feature space
$\vec{\mathfrak{m}}_i$	Centroid of $\mathfrak{X}_i$
$\mathbb{S}_B$	Scatter matrix between different solutions
$\mathbb{S}_W$	Scatter matrix within a solution
$\vec{\mathbf{w}}$	Eigenvectors of KFD

References

[1] A. Aamodt, H.A. Sandtorv, O.M. Winnem, Combining case based reasoning and data mining—a way of revealing and reusing rams experience, in: Proceedings of the International Conference on Safety and Reliability, ESREL'98, 1998, pp. 1345–1351.

[2] D.W. Aha, Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms, International Journal of Man-Machine Studies 36 (1992) 267–287.

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

[4] F.R. Bach, M.I. Jordan, Kernel independent component analysis, Journal of Machine Learning Research 3 (2002) 1–48.

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

[6] A. Bonzano, P. Cunningham, B. Smyth, Using introspective learning to improve retrieval in CBR: A case study in air traffic control, in: Proceedings of the 2nd International Conference on Case-Based Reasoning (ICCBR-97), July 25–27 1997, in: Lecture Notes Artif. Intell., Springer, 1997, pp. 291–302.

[7] H. Brighton, C. Mellish, On the consistency of information filters for lazy learning algorithms, in: Principles of Data Mining and Knowledge Discovery, Third European Conference, PKDD '99, Proceedings, Prague, Czech Republic, September 15–18, 1999, Springer, 1999.

Please cite this article in press as: R. Pan et al., Mining competent case bases for case-based reasoning, Artificial Intelligence (2007), doi:10.1016/j.artint.2007.04.018

[8] C.E. Brodley, Automatic algorithm/model class selection, in: ICML, 1993, pp. 17–24.
[9] R.M. Cameron-Jones, Instance selection by encoding length heuristic with random mutation hill climbing, in: Proceedings of the Eighth Australian Joint Conference on Artificial Intelligence, 1995, pp. 99–106.
[10] C.-L. Chang, Finding prototypes for nearest neighbor classifiers, IEEE Transactions on Computers 23 (1974) 1179–1184.
[11] S.J. Delany, P. Cunningham, An analysis of case-base editing in a spam filtering system, in: Proceedings of the European Case-based Reasoning Conference (ECCBR 2004), Springer, 2004, pp. 128–141.
[12] S.J. Delany, P. Cunningham, B. Smyth. Ecue: A spam filter that uses machine leaming to track concept drift, in: ECAI, 2006, p. 627.
[13] P. Domingos, Rule induction and instance-based learning: A unified approach, in: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal, Québec, Canada, 1995, pp. 1226–1232.
[14] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, second ed., Wiley-Interscience, 2000.
[15] R.A. Fisher, The use of multiple measurements in taxonomic problems, Annals of Eugenics 7 (1936) 179–188.
[16] C. Fyfe, J.M. Corchado, Automating the construction of cbr systems using kernel methods, International Journal of Intelligent Systems 16 (4) (2001) 571–586.
[17] M.R. Garey, D.S. Johnson, Computers and Intractability, A Guide to the Theory of NP-Completeness, W.H. Freeman, 1979.
[18] G.W. Gates, The reduced nearest neighbor rule, IEEE Transactions on Information Theory 18 (1972) 431–433.
[19] P.E. Hart, The condensed nearest neighbor rule, IEEE Transactions on Information Theory 14 (1968) 515–516.
[20] T. Hastie, A. Buja, R. Tibshirani, Penalized discriminant analysis, The Annals of Statistics 23 (1995) 73–102.
[21] T. Hastie, R. Tibshirani, A. Buja, Flexible discriminant analysis by optimal scoring, Journal of the American Statistical Association 89 (1994) 1255–1270.
[22] D.F. Kibler, D.W. Aha, Learning representative exemplars of concepts: an initial case study, in: P. Langley (Ed.), Proceedings of the Fourth International Workshop on Machine Learning, Irvine, 1987, Palo Alto, CA, 1987, pp. 24–30, MK.
[23] J. Kolodner, Case-Based Reasoning, Morgan Kaufmann, San Mateo, CA, 1993.
[24] D.B. Leake (Ed.), Case-Based Reasoning: Experiences, Lessons, and Future Directions, AAAI Press, Menlo Park, CA, 1996.
[25] E. McKenna, B. Smyth, Competence-guided case-base editing techniques, in: Proceedings of Advances in Case-Based Reasoning, 5th European Workshop, EWCBR 2000, Trento, Italy, 2000, pp. 186–197.
[26] E. McKenna, B. Smyth, Competence-guided editing methods for lazy learning, in: ECAI 2000, Proceedings of the 14th European Conference on Artificial Intelligence, Berlin, Germany, 2000, pp. 60–64.
[27] D. McSherry, Automating case selection in the construction of a case library, Knowledge-Based Systems 13 (2–3) (2000) 133–140.
[28] S. Mika, Kernel Fisher discriminants, PhD thesis, University of Technology, Berlin, 2002; http://ida.first.gmd.de/publications/Mik02.pdf.
[29] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, K.-R. Müller, Fisher discriminant analysis with kernels, in: Y.-H. Hu, J. Larsen, E. Wilson, S. Douglas (Eds.), Neural Networks for Signal Processing IX, IEEE, 1999, pp. 41–48.
[30] R. Pan, Q. Yang, L. Li, Case retrieval using nonlinear feature-space transformation, in: Advances in Case-Based Reasoning, 7th European Conference (ECCBR-04), 2004, pp. 361–374.
[31] R. Pan, Q. Yang, J. Junfeng Pan, L. Li, Competence driven case-base mining, in: Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, Pittsburgh, Pennsylvania, USA, 2005, pp. 228–233.
[32] D.W. Patterson, N. Rooney, M. Galushka, S.S. Anand, Towards dynamic maintenance of retrieval knowledge in cbr, in: Proceedings of the Sixteenth International Florida Artificial Intelligence Research Society Conference, 2002, pp. 126–131.
[33] C.E. Rasmussen, R.M. Neal, G. Hinton, D. van Camp, M. Revow, Z. Ghahramani, R. Kustra, R. Tibshirani, Delve data for evaluating learning in valid experiments, http://www.cs.toronto.edu/~delve/, 2003.
[34] G.L. Ritter, H.B. Woodruff, S.R. Lowry, T.L. Isenhour, An algorithm for a selective nearest neighbor decision rule, IEEE Transactions on Information Theory 21 (1975) 665–669.
[35] V. Roth, V. Steinhage, Nonlinear discriminant analysis using kernel functions, in: S.A. Solla, T.K. Leen, K.-R. Müller (Eds.), Advances in Neural Information Processing Systems, vol. 12, MIT Press, 1999, pp. 568–574.
[36] K. Saadi, N.L.C. Talbot, G.C. Cawley, Optimally regularised kernel fisher discriminant analysis, in: 17th International Conference on Pattern Recognition (ICPR (2)), 2004, pp. 427–430.
[37] S. Salzberg, A nearest hyperrectangle learning method, Machine Learning 6 (1991) 251–276.
[38] B. Schölkopf, A.J. Smola, K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Computation 10 (5) (1998) 1299–1319.
[39] D.B. Skalak, Prototype and feature selection by sampling and random mutation hill climbing algorithms, in: International Conference on Machine Learning, Morgan Kaufmann, 1994, pp. 293–301.
[40] B. Smyth, M.T. Keane, Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems, in: Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-95), August 1995, Morgan Kaufmann, San Francisco, CA, 1995, pp. 377–382.
[41] B. Smyth, M.T. Keane, Modelling the competence of case-bases, in: P. Cunningham, B. Smyth, M. Keane (Eds.), Proceedings of the Fourth European Workshop on Case-Based Reasoning, Springer, Berlin, 1998, pp. 208–220.
[42] B. Smyth, E. McKenna, Building compact competent case-bases, in: Proceedings of the Third International Conference on Case-Based Reasoning, Springer, Berlin, 1999, pp. 329–342.
[43] B. Smyth, E. McKenna, Competence models and the maintenance problem, Computational Intelligence 17 (2) (2001) 235–249.
[44] I. Tomek, An experiment with the edited nearest-neighbor rule, IEEE Transactions on Systems, Man, and Cybernetics 6 (1976) 448–452.
[45] V. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.
[46] I. Watson, Applying Case-Based Reasoning: Techniques for Enterprise Systems, Morgan Kaufmann, San Mateo, CA, 1997.

Please cite this article in press as: R. Pan et al., Mining competent case bases for case-based reasoning, Artificial Intelligence (2007), doi:10.1016/j.artint.2007.04.018

[47] D.L. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Transactions on Systems, Man, and Cybernetics 2 (1972) 408–421.

[48] D.R. Wilson, T.R. Martinez, Instance pruning techniques, in: Proceedings of the Fourteenth International Conference on Machine Learning (ICML 1997), Nashville, Tennessee, USA, 1997, pp. 403–411.

[49] D.R. Wilson, T.R. Martinez, Reduction techniques for instance-based learning algorithms, Machine Learning 38 (3) (2000) 257–286.

[50] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, second ed., Morgan Kaufmann, San Francisco, CA, 2005.

[51] T. Xiong, J. Ye, Q. Li, R. Janardan, V. Cherkassky, Efficient kernel discriminant analysis via qr decomposition, in: L.K. Saul, Y. Weiss, L. Bottou (Eds.), in: Advances in Neural Information Processing Systems, vol. 17, MIT Press, Cambridge, MA, 2005, pp. 1529–1536.

[52] Q. Yang, J. Zhu, A case-addition policy for case-base maintenance, Computational Intelligence 17 (2) (2001) 250–262.

[53] J. Zhang, Selecting typical instances in instance-based learning, in: Machine Learning: Proceedings of the Ninth International Conference ML92, Morgan Kaufmann, 1992, pp. 470–479.

[54] J. Zhu, Q. Yang, Remembering to add: Competence-preserving case-addition policies for case base maintenance, in: Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-99), Morgan Kaufmann, 1999, pp. 234–241.