Towards Addressing Problem-Distribution Drift with Case Discovery

David Leake and Brian Schack

Luddy School of Informatics, Computing, and Engineering, Indiana University
Bloomington IN 47408, USA
{leake,schackb}@indiana.edu

Abstract. Case-based reasoning (CBR) is a problem-solving and learning methodology that applies records of past experiences, captured as cases, to solve new problems. The performance of CBR depends on retrieving cases relevant to each new problem that the reasoner encounters. In real-world applications, the distribution of problems can change over time, which can cause an issue for the competence and efficiency of CBR systems. This paper proposes addressing this issue through *predictive case discovery*, which involves predicting cases expected to be useful for future problems to acquire them in advance. It presents an overview of case discovery for problem-distribution drift, including the challenges involved, proposed strategies, and future research directions. It illustrates with a case study evaluating a clustering-based case discovery strategy in a path planning domain across four scenarios: no drift, non-cyclical drift, cyclical drift, and drift from obsolescence.

Key words: adversarial drift, case-base maintenance, case discovery, data drift, problem-distribution drift, problem-distribution regularity, representativeness assumption

1 Introduction

Case-based reasoning (CBR) solves new problems by adapting solutions from similar past experiences to fit new circumstances (e.g. [16]). It is well known that the effectiveness of any CBR system depends on two types of regularity, which have been formalized as problem-solution regularity and problem-distribution regularity [14]. *Problem-solution regularity* can be informally characterized as, "Similar problems have similar solutions," and is needed in order for the adaptation of the solutions to retrieved cases to be useful in similar situations. *Problem-distribution regularity* can be informally be characterized as, "Future problems will resemble past problems." This property is necessary so that learned cases will tend to be useful in the future. In addition to these types of regularity, another regularity is required by machine learning systems more generally: the regularity that learned concepts tend to remain valid over time.

The effectiveness of CBR applications suggests that a combination of careful system design and suitable problem environment can provide these properties in practical situations. However, they are not guaranteed. As CBR is used in long-lived systems, developments over time may diminish any of these regularities, presenting difficulties for those systems. When concepts change over time, the result is concept drift [32], which makes prior cases no longer apply and requires case base updating [3]. When similarity criteria no longer reflect similarity for system needs, for example due to changes in case adaptation knowledge, performance may degrade [12]. When the distribution of problems changes over time, the quality of case base coverage may degrade, contravening problem-distribution regularity and requiring maintenance to ensure that the system has the cases that it needs going into the future.

This paper first examines each of these types of regularity, and then focuses on problem-distribution regularity, which to the authors' knowledge, has received little prior attention in CBR. To mitigate problem-distribution regularity failures, it proposes *predictive case discovery:* developing methods that identify and anticipate problem-distribution drift to guide the selection of cases to request from an external source.

This paper identifies general requirements and classes of detection methods that may be used to guide case discovery. It then illustrates with a clustering-based method aimed at identifying "hot spots" in the problem space on which to focus discovery. It presents an evaluation of this sample method in a path planning domain across four scenarios: no drift, non-cyclical drift, cyclical drift, and drift from obsolescence. The results support the general effectiveness of the strategy and also illustrate its limitations. The paper concludes with future opportunities.

2 Regularities Underpinning CBR

Problem-Solution Regularity: The effectiveness of reuse of past cases depends on *problem-solution regularity* — the property that solutions to similar problems will provide a useful starting point for solving a new problem. Often in the CBR literature, the assumption is that adapting the solution to a similar problem should reduce solution generation cost compared to reasoning from scratch, for generating an acceptable solution. Systems achieving this type of problem-solution regularity have been demonstrated in multiple scenarios (e.g. [24, 31]).

Problem-Distribution Regularity: The benefit to the CBR process of storing past cases depends on *problem-distribution regularity* — the property that the distribution of future problems will tend to reflect that of past problems, such that accumulated stored cases from past episodes tend to provide useful information for future problems.

Formalizing the Properties: Leake and Wilson [14] provide a formalization of these properties. They define problem-solution regularity as depending on:

1. The retrieval function the system uses to map problems to cases in the case base

2. A definition of the goals to be satisfied by retrieval — what would make a case a good starting point for solving a new problem.
3. The initial set of cases available to the system
4. The problems that the system is called upon to solve

Retrieval goals are often defined in terms of a predefined similarity measure, such as the semantic similarity between a target problem and the problem part of stored cases. However, they can be based on other criteria, e.g. that the solution to the retrieved case should be inexpensive to adapt to generate a correct solution to the new problem [24]. Another possible criterion is that the retrieved case should be adaptable to generate a result within a certain accuracy.

Items (3) and (4) reflect that problem-solution regularity must be measured in terms of the problems the system has to solve. Items (1), (2), and (3) are under the control of a system designer. For example, to further problem-solution regularity, a retrieval function aimed at retrieving adaptable cases could be hand-designed, or it could be learned (e.g. [15]). However, new cases received by the system, as referenced in (4), may cause changes in problem-solution regularity if the retrieval function is ill-suited to judging similarity for those cases.

Problem-distribution regularity reflects the correlation between the distribution of cases in the case base and the distribution of future problems. Even a case base evenly distributed across the problem space may have low problem-distribution regularity if upcoming problems are not evenly distributed.

Problem-distribution regularity has been formalized in terms of the long-term behavior of a CBR system: The probability that, at a given point in processing a stream of problems, the system will be able to retrieve a case sufficiently close to an input problem [14]. The assumption of problem-distribution regularity relates to the influential *representativeness assumption* of case-base maintenance, "The case base is a representative sample of the target problem space," [25]. This property is important for assessing case competence for the possible range of future problems. However, problem-distribution regularity only measures whether eventually the system will have a sufficient probability of containing the cases needed to cover new problems; it measures (after the fact) whether it was possible to cover most problems actually received by the system, rather than predicting whether the case base provides a good sample of possible future problems.

3 How Regularity May Degrade: Types of Drift

Even in suitable domains, regularities may not always hold. Existing cases may become obsolete [14], space or time requirements may necessitate deletion [28, 26] with corresponding competence loss [29], or the system may simply not have been provided with sufficient initial cases (or the experiences to acquire cases) to address current problems. The following subsections consider how drift may affect concept regularity and problem-distribution regularity.

3.1 Concept Drift

Concept drift, a widely explored phenomenon in machine learning [17], refers to the situation where the underlying concepts or relationships between the problem and solution change over time. For example, in a system for property valuation, inflation could cause prices to increase over time, providing a gradual transition, or a chemical spill could cause an abrupt decrease in valuations in a particular region. Or in a system for sentiment analysis, the evolution of slang could change the interpretation of the sentiment of online messages. By making cases inaccurate, concept drift can degrade the performance of a formerly accurate CBR system. When inaccuracy is associated with the time passed since acquiring the case, then this is characterized as *case obsolescence.*

3.2 Problem-Distribution Drift

Another issue arises in domains for which the problem distribution changes over time. For a disaster management system, climate change can lead to shifts in weather patterns, reducing the usefulness of a case base containing response plans for historical weather patterns. For a travel agency recommender system, certain areas may become “hot” destinations — resulting in a different range of necessary coverage. For a real estate appraisal system, developers may build newer properties with different characteristics from older properties and beyond the scope of reliable adaptation.

Problem-distribution drift refers to the change in the distribution of problems that the CBR system must solve. If a customer support system has cases for certain problems and the types of problems customers encounter change, the existing case base may become less useful, and the CBR system may need to acquire new cases that are relevant to the new problem distribution. Because problem-distribution regularity is defined in terms of the limit of case base growth, even when a domain satisfies problem-distribution regularity in the long term, practical issues can still arise in the short term.

Causes of Problem-Distribution Drift Problem-distribution drift can be caused by various factors, including changes in the environment, changes in user preferences, changes in technology, or changes in the problem space itself.

Environmental changes The environment in which a CBR system operates can change over time, leading to changes in the distribution of problems that the system solves. For example, in a medical diagnosis system, changes in the prevalence of certain diseases can lead to changes in the distribution of medical problems that the system needs to diagnose and treat.

User preferences User desires or the problems that they wish to address may change. For example, a recommender system for clothing or travel packages needs to change seasonally as the problem distribution changes with user preferences. In some scenarios, changes in fashion may also render prior cases obsolete.

Technological changes As technology advances, new problems can arise that were not present before. For example, in a software support system, upgrades in the software can incorporate new features and associated bugs for which the system needs to provide support.

Changes in the problem space Changes in the underlying physics or changes in the social or economic context of a domain may affect problem distributions. For example, in a financial prediction system, changes in the market conditions or regulations can lead to changes in the distribution of financial problems that the system needs to predict.

3.3 Adversarial Drift

Adversarial drift refers to data drift in which a reasoner responds to cases presented by an adversary who presents characteristically different cases over time with the intention of degrading performance [10]. Adversarial drift could involve concept drift, problem-solution regularity drift, or problem-distribution regularity drift. Adversarial drift can be observed in imperfect information games such as poker, in which players benefit from associating their opponent's bets and "tells" to their strategic position or the strength thereof. In such games, the opponent may bluff their bets or fake their tells, intentionally breaking regularity to gain an advantage. Delany et al. [5] tracked and mitigated adversarial drift as spammers adapted their techniques to circumvent spam filters.

4 Addressing Problem-Distribution Drift with Guided Case Discovery

Problem-distribution drift may be addressed in a two-step process. First, the CBR system can detect and track drift. And second, it can extrapolate to anticipate the path of the drift and request cases in the path of the drift. For example, in a CBR system to recommend travel packages, if a new destination is published in a major magazine, then the number of requests for packages for that region may increase. If the trend of having more requests in that area can be detected, then the system could request additional cases in that area to better prepare for future requests.

4.1 Prior Work on Drift Detection

Drift detection algorithms generally fall into four categories: error rate-based drift detection, data distribution-based drift detection, multiple hypothesis-based drift detection, and competence-modeling strategies.

Error rate-based strategies compare the error rate of the model before and after a certain time period. These algorithms are commonly used in classification tasks, where the error rate can be calculated by comparing the predicted class labels to the true class labels. Increased error rate over time can indicate data drift. One common error rate-based algorithm is the ADWIN algorithm which adapts the window size based on observed changes in the error rate [2].

Data distribution-based strategies compare the data distribution before and after a certain time frame. These algorithms can detect gradual changes in the data distribution which may not be reflected in the error rate. One popular algorithm is the Kullback-Leibler (KL) divergence-based method which measures the difference between two probability distributions. If the divergence exceeds a threshold, then this can indicate data drift [4].

Multiple hypothesis-based strategies test multiple hypotheses simultaneously, making it possible to detect complex drift patterns. These algorithms are useful when the data drift is not well understood or cannot be modeled using a simple statistical model. One example of a multiple hypothesis-based algorithm is the Just-in-Time adaptive classifiers (JIT) algorithm which uses a sequence of hypotheses tests to detect changes in the data distribution [1].

Competence-modeling strategies, such as proposed by Lu et al. [18] detect drift based on changes in competence.

Depending on the strategy, the curse of dimensionality can also impact the detection of data drift. The *curse of dimensionality* refers to the problem of high dimensionality in a problem space leading to sparsity and high computational costs [11]. As the number of dimensions increases, cases can become widely dispersed in the high-dimensional space making changes in their distribution difficult to detect. One approach to addressing the curse of dimensionality is to reduce the dimensionality of the dataset by selecting relevant features before applying drift detection algorithms.

4.2 Case Discovery

When problem-distribution regularity decreases, a potential repair is to add cases to the case base. Case discovery can fill gaps in the distribution of the cases in the case base to cover upcoming problems that would otherwise fall into those gaps. In systems including a generative component capable of solving problems from scratch, discovery can be done by calling upon that component. In that situation, generating the solution in advance does not increase competence but provides speed-up learning by avoiding the need to generate a solution from scratch at run time (e.g., as in Prodigy/Analogy [31]). In other scenarios, discovery may be done by requesting cases beyond system competence from an external source such as a domain expert.

Which Cases to Discover: As the cost of case solicitation may be high, effectiveness of discovery depends on targeting [19–23]. For example, McKenna and Smyth [20] propose a case discovery strategy which identifies competence holes in a case base to fill by discovering spanning cases. Such approaches can be combined with problem prediction to further focus on the regions of the problem space likely to be relevant to incoming problems.

Let

- CB: the case base,
- *part*: a partition function,
- *active-part*: a function selecting one of the subsets of a partition,
- *rep*: a function generating a case to discover given a set of cases

```
Generate-target-case(CB, part, active-part, rep)

1. parts $\leftarrow$ part(CB)
2. chosen-part $\leftarrow$ active-part(parts)
3. target-case $\leftarrow$ rep(chosen-part)
4. Return target-case
```

Fig. 1: General discovery procedure

Let d be a distance metric, N the number of desired clusters, and CB the case base:

```
1. Divide the training cases into $N$ clusters using the k-means algorithm for distance $d$.
2. Randomly choose a cluster $CL$ of training cases.
3. Find $cr$ = centroid case of $CL$.
4. Alter the value of a single feature of $cr$ to yield a variation to request for discovery.
```

Fig. 2: K-means discovery algorithm

Additional methods could be brought to bear from outside CBR. For example, the SMOTE oversampling algorithm [6] mitigates class imbalance by generating synthetic instances by interpolating between neighboring minority instances. Extensions to SMOTE can handle concept drift on time series data [8]. Case discovery and oversampling can also be seen as similar in spirit to data augmentation for neural networks [9]. Applying a similar spirit to case-based reasoning, adaptation rules provide a knowledge-rich source of transformations that go beyond interpolation, yielding "ghost cases" that tend to preserve case cohesion [13]. Ghost cases generated by adaptation can improve efficiency but generally would not be expected to increase competence.

4.3 Clustering-Based Case Discovery

This paper proposes a general case discovery strategy of dividing the problem space into parts, predicting the most active part, selecting a point in that part, and then requesting discovery of that case. This is illustrated in Figure 1. As an illustration and for empirical evaluation, the paper applies this strategy in a simple clustering-based approach that we call *k-means discovery*.

K-means discovery uses k-means clustering to divide the problem space into N regions for a predefined value N. It then selects a random cluster from the N regions, meaning that N determines the balance between exploration (for large N, resulting in small regions) and exploitation (for small N, resulting in large regions). Alternative methods could, for example, favor "up and coming" clusters with recent activity.

After a cluster is selected, k-means discovery finds the case at the centroid of that cluster. Then it generates a variation on the centroid case by altering that case. In our simple demonstration, it does so by altering the value of a single feature. For example, in the path planning domain, the variation could be a path where one endpoint is the same as in the centroid path, and the other endpoint is randomly chosen from the entire graph. Richer methods could be used, such as adaptation strategies changing several features of the problem description in concert. (See Leake and Schack [13] for a discussion of adaptation of both problem and solution parts of cases to generate ghost cases.) The variation on the centroid case becomes the case to discover.

This paper uses clustering-based case discovery as an illustration because it is simple and requires minimal domain knowledge, making it suitable for domains where knowledge is scarce or costly to obtain. And because, by choosing the case at the centroid of the cluster, the strategy tends to discover a variant of a case representative of that cluster — which the authors hypothesized would reflect 'hot spots' because less representative cases would tend towards the edges. As needed, k-means could be replaced with other methods. For example, spherical k-means, using cosine similarity, could be used for textual cases, or affinity propagation could be used for clustering based on precomputed distances and without predetermining a number of clusters.

5 Managing Multiple Strategies

Drift detection and case discovery strategies may have different strengths and weaknesses depending on the characteristics of the problem domain and the data drift (if any). Additionally, each strategy may have parameters (such as window size or number of clusters) that need to be tuned to achieve optimal performance. Choosing the "right" strategies and parameters can impact the accuracy and efficiency of drift detection and case discovery.

A potential approach to dealing with this issue is to develop a library of strategies and select which to apply. Bandit meta-strategies provide a potential approach. If strategies are initially selected at random, and the choice between strategies is weighted based on the number of problems for which the strategies' cases were successfully used in the past, choices could be refined to favor successful strategies. However, for a rapidly-changing distribution, this information could quickly become obsolete. This is a subject for future study.

6 Evaluation

The experiments evaluate the proposed clustering-based case discovery strategy, examining the following question:

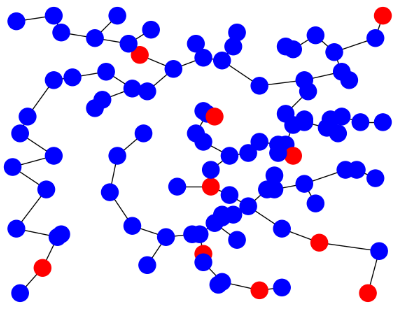


Fig. 3: An example of a graph modeling a randomly constructed transit network. Frequently visited nodes are red, and infrequently visited nodes are blue.

What effect does the k-means discovery *strategy have on the adaptation cost of retrieved cases compared to random case discovery across four scenarios: no drift, non-cyclical drift, cyclical drift, and obsolescence?*

6.1 Testing Scenario

The experiments are conducted in a simulated path planning domain. This is an established domain for evaluating CBR systems (e.g. PathFinder [27]), and it has practical applications such as for mobile robots [7] and autonomous underwater vehicles [30]. The intuitive motivation for the scenario is that an agent needs to travel from place to place. Cases are modeled with a problem part and a solution part. The problem part is the starting and ending points of the travel path, and the solution part is a list of nodes along the path from the starting point to the ending point.

Some routes and destinations will occur more frequently than others and the scenario can change over time due to moving homes, changing jobs, closing roads, and so on, providing problem distribution drift.

The experiments model the road and transit network as a graph in which each node is a destination. Variations on the scenario explore obsolescence of cases and cyclical and non-cyclical problem-distribution drift. Obsolescence is modeled by assigning an expiration age to cases; cases are no longer available after a certain number of problems have been solved. The following paragraphs in this section describe the details of the testing scenario, and code for replicating the evaluation is available in a public GitHub repository.[1]

Constructing the Graphs Each iteration was done on a different graph generated with 100 nodes. Each node was positioned randomly in two-dimensional space. The edges were constructed with k-edge augmentation where $k = 1$. The edge

[1] https://github.com/schackbrian2012/ICCBR-2023

augmentation ensures that the graph cannot be disconnected unless k or more edges are removed. Although the graphs were unweighted, the edge augmentation was weighted by the Euclidean distance between nodes. The number of edges varied from graph to graph.

The edge augmentation method served three purposes: First, some path should exist between any pair of nodes so that path planning is possible. Second, most paths should be longer than a single edge so that path planning is non-trivial. And third, edges should be more common between nearby pairs of nodes than between distant pairs of nodes, for a correlation between Euclidean distance (for retrieval) and graph distance (for adaptation). Figure 3 illustrates a graph constructed by this method.

Populating the Case Base For each test, the case base was populated by first randomly choosing 10 distinct nodes from the graph, to serve as the frequently visited nodes. These were unequally weighted from most frequent (10) to least frequent (1). A node was randomly chosen from the frequently visited nodes, weighted by node weights. Another node was randomly chosen from the whole graph, giving each node an equal probability. Either a departing path (from a frequently visited node to a random node) or a returning path (from a random node to a frequently visited node) was constructed. The process was repeated to generate 1,000 paths.

This method of populating the case base served three purposes: First, because the nodes are randomly selected from the graph, the paths in the case base cover different parts of the graph. This diversity is important for evaluating the ability of the case discovery strategy to adapt to changes in the problem distribution. Second, by giving higher weights to frequently visited nodes, the method accounts for the fact that some parts of the graph may be more important than others, a realistic assumption for the path planning domain. Third, 1,000 paths is large enough to capture the variability in the problem distribution and the ability of the case discovery strategy to adapt to it.

Discovery Methods The evaluation compared three discovery methods. The *No Discovery* method is a baseline that does not discover any cases. The *Random Discovery* and *Clustering-Based Discovery* methods discover one case at each time step. The Random Discovery method selects random values for each feature of the problem part of the case for discovery. In the path planning domain, the starting and ending points of the path for discovery are two nodes randomly chosen from the entire graph. The Clustering-Based Discovery method uses k-means discovery with eight clusters and a categorical distance metric. Exploratory analysis of different numbers of clusters showed that eight was effective for this task. The categorical distance metric treats each node as a category (instead of a two-dimensional coordinate) and compares nodes by exact match.

Distance Metric for Retrieval The retrieval process yields the most similar training or discovery case to the testing problem measured according to the Euclidean distance between two four-dimensional vectors — the x- and y-coordinates of the source and target nodes of each path — resolving ties arbitrarily.

Performance Assessment Methods The three discovery methods were compared using two performance assessment methods: Leave One Out and Time Series Cross-Validation. For both evaluation methods, at each time step, the reasoner is presented with the problem part of a different testing case — making the number of time steps equal to the number of folds. The order of testing is the same as the order of generation described in the previous "Populating the Case Base" subsection and Section 6.2. For the Leave One Out method, the training cases include all cases other than the testing case. For the Time Series Cross-Validation method, the training cases include all cases encountered prior to the testing case. The experiment ran for 10 iterations with different graphs and case bases in each iteration.

The Leave One Out method tests generalizability by iteratively training on all but one data point, and then testing on the left-out data point, to prevent over-fitting. The Time Series Cross-Validation method is suited for temporal data, as it evaluates the performance of the model on future time points based on the training data available up to that point, simulating a real-world scenario where the model has to predict future events based on past data.

Distance Metric for Evaluation The distance metric for evaluating the adaptation cost of a solution sums of the number of edges in the shortest path to adapt the starting and ending points of a training case to the starting and ending points of the testing case. Unlike the distance metric for retrieval, the distance metric for evaluation ignores Euclidean distance. For an exact match, the distance metric returns zero.

6.2 Variations on the Testing Scenario

The experiments evaluated four variations on the testing scenario: No Drift, Non-Cyclical Drift, Cyclical Drift, and Obsolescence.

No Drift Scenario The frequently visited nodes remain the same throughout the time series. The No Drift Scenario serves as a baseline for comparison with the other scenarios. It tests the ability of the case discovery strategy to handle a stable problem distribution where the frequently visited nodes remain the same throughout the time series.

Non-Cyclical Drift Scenario Halfway through the time series, at time step 500, the frequently visited nodes are changed to a different set of frequently visited nodes constructed by the same random sampling of the nodes. This in turn abruptly changes the paths constructed from the frequently visited nodes. The Non-Cyclical Drift Scenario tests the ability of the case discovery strategy to handle abrupt changes in the problem distribution.

Cyclical Drift Scenario This scenario alternates between two sets of frequently visited nodes for two equal-length phases of each set. It tests the ability of the case discovery strategy to handle cyclic changes in the problem distribution, which can occur due to seasonal changes or recurring patterns in the user behavior. Specifically, this scenario tests the ability of the case discovery strategy to adapt to two different sets of frequently visited nodes and construct paths that switch between the two sets.

Obsolescence Scenario This scenario simulates case obsolescence by only reusing cases stored or discovered at less than 100 time steps before the testing problem that they solve (for the evaluation that allows use of all cases in the case stream, future cases are also only considered within 100 time steps). Evaluation penalizes the retrieval of an obsolete case by re-planning the entire path from the starting point to the ending point of the testing problem. The number of edges in the re-planned path is treated as the adaptation cost from the obsolete case to the testing problem. The distance metrics for retrieval and clustering do not consider obsolescence.

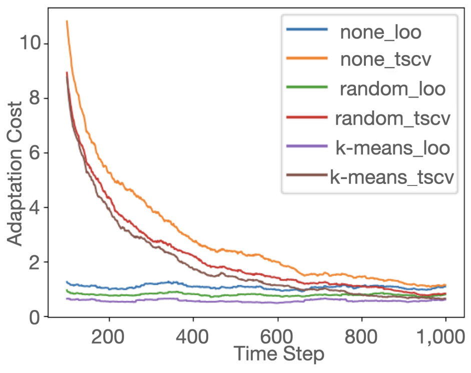
6.3 Experimental Results

Figure 4 shows the experimental results. The x-axis measures the time step of the testing case under evaluation. The y-axis measures the adaptation cost of the solution. Each graph presents the rolling average of adaptation cost over a window of 100 time steps.

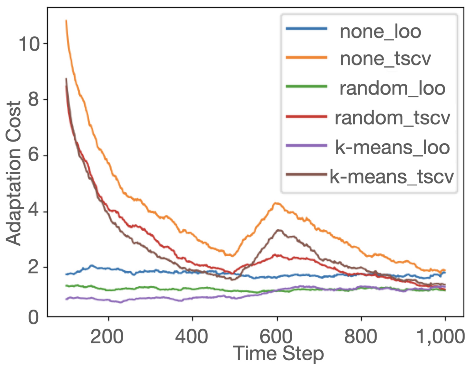
The none_loo and none_tscv series use the No Discovery strategy. The random_loo and random_tscv series use the Random Discovery strategy. And the k-means_loo and k-means_tscv series use the Clustering-Based Discovery strategy. The none_loo, random_loo, and k-means_loo series use the Leave One Out evaluation method; the none_tscv, random_tscv, and k-means_tscv series use the Time Series Cross-Validation evaluation method.

No Drift Scenario — Figure 4a The adaptation cost of Leave One Out evaluation remains steady over time because both past and future cases make up its training. The adaptation cost of Time Series Cross-Validation improves over time because the number of cases for training and discovery increase over time. The adaptation cost of each discovery method evaluated by Time Series Cross-Validation approaches the adaptation cost of the same discovery method evaluated by Leave One Out. Random Discovery outperforms No Discovery because discovery yields additional cases beyond the training cases. Clustering-Based Discovery outperforms Random Discovery because the cases discovered by the former tend to match the distribution of problems, and cases discovered by the latter tend towards an even distribution over the problem space.

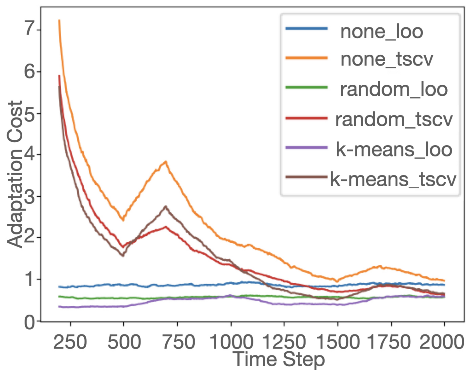
Non-Cyclical Drift Scenario — Figure 4b The earlier phase goes from time steps 0–500, and the later phase goes from time steps 500–1,000. The plot does not show time steps 0–100 because of the rolling window of 100 time steps. The



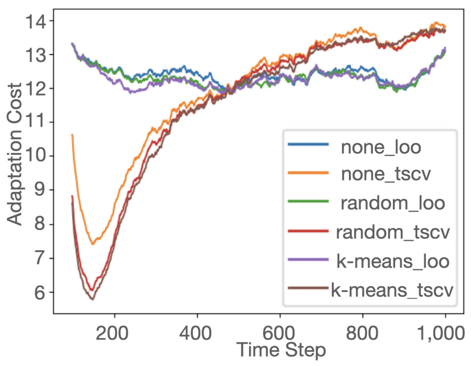
(a) No Drift



(b) Non-Cyclical Drift



(c) Cyclical Drift



(d) Obsolescence Scenario

Fig. 4: Evaluation of clustering-based case discovery across four scenarios.

earlier phase resembles the No Drift Scenario in Figure 4a because no drift has yet occurred. Halfway through, at time step 500, adaptation cost increases for the Time Series Cross-Validation evaluation method because the problem distribution of training cases from the earlier phase does not match the problem distribution of testing cases from the later phase.

Adaptation cost increases more steeply for the none_tscv and k-means_tscv series than the random_tscv series because the latter discovers cases unbiased by the problem distribution of the earlier phase. The adaptation cost of k-means_loo also increases because it discovers the same cases as k-means_tscv which are biased towards the problem distribution of the earlier phase. Around time step 600, adaptation cost for Time Series Cross-Validation begins to decrease again as training cases arrive from the later phase and the k-means clustering algorithm incorporates training cases from both phases. Like Figure 4a, approaching the end at time step 1,000, the adaptation cost of each discovery method evaluated by Time Series Cross-Validation approaches the adaptation cost of the same discovery method evaluated by Leave One Out.

Cyclical Drift Scenario -- Figure 4c The first phase, from time steps 0–500, resembles the No Drift Scenario in Figure 4a because no drift has occurred yet. The first two phases, from time steps 0–1,000, resemble the Non-Cyclical Drift Scenario in Figure 4b because the drift has not yet repeated an earlier phase. The first drift (from the first phase to the second phase) and the third drift (from the third phase to the fourth phase) impact adaptation cost more than the second drift (from the second phase to the third phase) because the problem distribution of the training cases and the Guided Discovery cases in the first phase matches the third phase.

Obsolescence Scenario -- Figure 4d Adaptation cost drops for the Time Series Cross-Validation evaluation method before time step 200 as training and discovery cases arrive to solve testing problems. Then the ratio of obsolete to contemporary training and discovery cases increases — causing an increase in the number of obsolete cases retrieved and an increase in the adaptation cost. Adaptation cost increases for the Leave One Out evaluation method at the start and end of the time series. The window of contemporary cases is 100 time steps before and after the test problem, but the start (resp. end) of the time series has fewer cases before (resp. after) the test problem.

7 Conclusion

This paper discusses the regularities required for successful case-based reasoning and potential issues arising from various types of drift, including concept drift, problem-distribution drift, and adversarial drift. Then it discusses different strategies for detecting drift, such as error rate-based, data distribution-based, and multiple hypothesis-based strategies. It illustrates a case discovery strategy, k-means discovery, guided by k-means clustering, and evaluates its effectiveness on synthetic time series data in a path planning domain across four different scenarios. The evaluation demonstrates that it outperforms baselines. However, because the effectiveness of discovery strategies depends on characteristics of the drift itself, there is no universal strategy. An important next step is to explore additional strategies for drift prediction and identifying cases to discover, including drawing on methods from outside CBR, and investigating ways to select the right strategy for the domain.

8 Acknowledgments

This work was funded in part by the Department of the Navy, Office of Naval Research (Award N00014-19-1-2655).

References

1. Alippi, C., Roveri, M.: Just-in-time adaptive classifiers—part I: Detecting nonstationary changes. IEEE Transactions on Neural Networks **19**(7), 1145–1153 (2008)

2. Bifet, A., Gavalda, R.: Learning from time-changing data with adaptive windowing. In: Proceedings of the 2007 SIAM international conference on data mining. pp. 443–448. SIAM (2007)
3. Cunningham, P., Nowlan, N., Delany, S., Haahr, M.: A case-based approach to spam filtering that can track concept drift. Tech. Rep. TCD-CS-2003-16, Computer Science Department, Trinity College Dublin (2003)
4. Dasu, T., Krishnan, S., Venkatasubramanian, S., Yi, K.: An information-theoretic approach to detecting changes in multi-dimensional data streams. In: Proc. Symposium on the Interface of Statistics, Computing Science, and Applications (Interface) (2006)
5. Delany, S.J., Cunningham, P., Tsymbal, A., Coyle, L.: A case-based technique for tracking concept drift in spam filtering. In: Applications and Innovations in Intelligent Systems XII: Proceedings of AI-2004, the Twenty-fourth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence. pp. 3–16. Springer (2005)
6. Fernández, A., Garcia, S., Herrera, F., Chawla, N.V.: Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. Journal of artificial intelligence research **61**, 863–905 (2018)
7. Hodál, J., Dvorák, J.: Using case-based reasoning for mobile robot path planning. Engineering Mechanics **15**(3), 181–191 (2008)
8. Hoens, T.R., Polikar, R., Chawla, N.V.: Learning from streaming data with concept drift and imbalance: an overview. Progress in Artificial Intelligence **1**(1), 89–101 (2012)
9. Iwana, B.K., Uchida, S.: An empirical survey of data augmentation for time series classification with neural networks. PLOS one **16**(7), e0254841 (2021)
10. Kantchelian, A., Afroz, S., Huang, L., Islam, A.C., Miller, B., Tschantz, M.C., Greenstadt, R., Joseph, A.D., Tygar, J.: Approaches to adversarial drift. In: Proceedings of the 2013 ACM workshop on Artificial intelligence and security. pp. 99–110 (2013)
11. Köppen, M.: The curse of dimensionality. In: 5th online world conference on soft computing in industrial applications (WSC5). vol. 1, pp. 4–8 (2000)
12. Leake, D., Kinley, A., Wilson, D.: Learning to integrate multiple knowledge sources for case-based reasoning. In: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence. pp. 246–251. Morgan Kaufmann (1997)
13. Leake, D., Schack, B.: Exploration vs. exploitation in case-base maintenance: Leveraging competence-based deletion with ghost cases. In: Case-Based Reasoning Research and Development, ICCBR 2018. pp. 202–218. Springer, Berlin (2018)
14. Leake, D., Wilson, D.: When experience is wrong: Examining CBR for changing tasks and environments. In: Proceedings of the Third International Conference on Case-Based Reasoning. pp. 218–232. Springer Verlag, Berlin (1999)
15. Leake, D., Ye, X.: Harmonizing case retrieval and adaptation with alternating optimization. In: Case-Based Reasoning Research and Development, ICCBR 2021. pp. 125–139. Springer, Cham (2021)
16. López de Mántaras, R., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M., Cox, M., Forbus, K., Keane, M., Aamodt, A., Watson, I.: Retrieval, reuse, revision, and retention in CBR. Knowledge Engineering Review **20**(3) (2005)
17. Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., Zhang, G.: Learning under concept drift: A review. IEEE Transactions on Knowledge and Data Engineering **31**(12), 2346–2363 (2019)

18. Lu, N., Zhang, G., Lu, J.: Concept drift detection via competence models. Artificial Intelligence **209**, 11–28 (2014)
19. Massie, S., Craw, S., Wiratunga, N.: Complexity-guided case discovery for case based reasoning. In: AAAI'05: Proceedings of the 20th national conference on Artificial intelligence. pp. 216–221. AAAI Press (2005)
20. McKenna, E., Smyth, B.: Competence-guided case discovery. In: Research and Development in Intelligent Systems XVIII. pp. 97–108. Springer, London (2002)
21. McSherry, D.: Automating case selection in the construction of a case library. In: Research and Development in Intelligent Systems XVI: Proceedings of ES99, the Nineteenth SGES International Conference on Knowledge-Based Systems and Applied Artificial Intelligence, Cambridge, December 1999. pp. 163–177. Springer (2000)
22. McSherry, D.: Intelligent case-authoring support in casemaker-2. Computational Intelligence **17**(2), 331–345 (2001)
23. Mehdi Owrang O, M.: Case discovery in case-based reasoning systems. Information systems management **15**(1), 74–78 (1998)
24. Smyth, B., Keane, M.: Adaptation-guided retrieval: Questioning the similarity assumption in reasoning. Artificial Intelligence **102**(2), 249–293 (1998)
25. Smyth, B., McKenna, E.: Competence models and the maintenance problem. Computational Intelligence **17**, 235–249 (2001)
26. Smyth, B.: Case-base maintenance. In: Tasks and Methods in Applied Artificial Intelligence: 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems IEA-98-AIE Benicàssim, Castellón, Spain, June 1–4, 1998 Proceedings, Volume II 11. pp. 507–516. Springer (1998)
27. Smyth, B., Cunningham, P.: The utility problem analysed: A case-based reasoning perspective. In: Advances in Case-Based Reasoning: Third European Workshop EWCBR-96 Lausanne, Switzerland, November 14–16, 1996 Proceedings 3. pp. 392–399. Springer (1996)
28. Smyth, B., Keane, M.T.: Remembering to forget. In: Proceedings of the 14th international joint conference on Artificial intelligence. pp. 377–382. Citeseer (1995)
29. Smyth, B., McKenna, E.: Competence models and the maintenance problem. Computational Intelligence **17**(2), 235–249 (2001)
30. Vasudevan, C., Ganesan, K.: Case-based path planning for autonomous underwater vehicles. Autonomous Robots **3**(2-3), 79–89 (1996)
31. Veloso, M.: Planning and Learning by Analogical Reasoning. Springer Verlag, Berlin (1994)
32. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine Learning **23**(1), 69–101 (1996)