A Fuzzy Kernel *c*-Means Clustering Model for Handling Concept Drift in Regression

Yiliao Song, Guangquan Zhang, Jie Lu, Haiyan Lu
Decision System and e-Service Intelligence (DeSI) Lab,
Centre for Artificial Intelligence, Faculty of Engineering and Information Technology
University of Technology, Sydney
Sydney, Australia
Yiliao.Song@student.uts.edu.au; {Guangquan.Zhang, Jie.Lu, Haiyan.Lu}@uts.edu.au

***Abstract*—Concept drift, given the huge volume of high-speed data streams, requires traditional machine learning models to be self-adaptive. Techniques to handle drift are especially needed in regression cases for a wide range of applications in the real world. There is, however, a shortage of research on drift adaptation for regression cases in the literature. One of the main obstacles to further research is the resulting model complexity when regression methods and drift handling techniques are combined. This paper proposes a self-adaptive algorithm, based on a fuzzy kernel *c*-means clustering approach and a lazy learning algorithm, called FKLL, to handle drift in regression learning. Using FKLL, drift adaptation first updates the learning set using lazy learning, then fuzzy kernel *c*-means clustering is used to determine the most relevant learning set. Experiments show that the FKLL algorithm is better able to respond to drift as soon as the learning sets are updated, and is also suitable for dealing with reoccurring drift, when compared to the original lazy learning algorithm and other state-of-the-art regression methods.**

Keywords—Lazy learning; concept drift; kernel; fuzzy systems; c-mean cluster;

I. INTRODUCTION

With the rapid advancement of Information and Communication Technology (ICT), internet and data sensor technology, data has been generated with a great speed and a huge volume of high-speed data streams are being created at a remarkably rapid pace. In the big data era, datasets are showing a large number of data features in terms of both quantity and complexity. Due to these new features, novel data processing applications are urgently needed to represent, process, and analyze big data to generate predictions and support decision-making. However, traditional machine learning models have limited ability to handle time-changing problem because they are based on a statistical assumption that the probability distribution of the attributes and the target is stationary [1]. Historically, data arrived slowly and therefore was considered to be stationary. High-speed data streams hardly satisfy this requirement, so it is inappropriate to apply traditional statistical or machine learning methods to handle time-changing problems. Variety in data streams has been recognized as concept drift issue, and is described by a range of characteristics in both the attribute set and the set's relationship to the target variable. Research interest in this issue is increasing and has been applied to many real-world prediction applications, such as weather, customer preferences, stocks, sales, bankruptcy potential, and clinical decision-making.

Generally, concept drift is defined as a change in the probability distribution of a dataset over time: $\boldsymbol{p}_{t_1}(\boldsymbol{X},\boldsymbol{Y}) \neq \boldsymbol{p}_{t_2}(\boldsymbol{X},\boldsymbol{Y})$, where $\boldsymbol{p}_t(\boldsymbol{X},\boldsymbol{Y})$ is the joint probability distribution of the attribute vector X and the target variable $\boldsymbol{Y}$ [2]. A series of self-adaptive models have been proposed to overcome concept drift in data [2]. Usually, methods that detect concept drift require the model to forget data that is no longer relevant. Although drift is defined in the form of a probability distribution, it is difficult to quantify its precise value because a statistic is required to measure variance in the distribution. This can be difficult, especially in high-dimensional cases. Training errors or a structured distance mainly act as this statistic in prevailing drift detection methods. In general, techniques to handle drift are separated into three categories: instance selection, instance weighting, and ensemble learning. Instance selection only uses the most relevant instances to the current concept to estimate the output of new instances. Instance weighting gives each instance a weight to represent the decreasing relevance of existing training examples. The ensemble learning methods utilize multiple models by voting or selecting the most relevant ones to construct a proper predictive model.

The key to overcome drift is to find the most relevant instances or models to the new concept fast and accurately. There are two gaps in the existing drift handling models. One is that much effort has been focused on concept drift problems in classification, yet limited attention has been given to regression problems despite its wide applications in real world practice. The main reason is that most machine learning models, such as support vector regression, neural networks, and decision trees, have complex expressions in regression problems; therefore, models are very expensive to update for concept drift. The other is how to respond to the drift fast. The drift-detection based methods, such as FIMT-DD[3] and AMRules[4] updated their models based on the drift detection results from the past time interval. These models are suitable for the instances from the past time interval but uncertain for the new arrivals. They are outdated when the drift significantly changes in the next time interval.

To solve the above problem, this study proposes a novel approach to concept drift in regression learning, FKLL. The method introduces a fuzzy clustering technique to update the learning set of a local regression algorithm so that it can self-

adapt to concept drift when needed. The local regression algorithm predicts the output of query points by integrating the information of its *k*-nearest neighbors from the learning set. It handles drift problems by updating the learning set using a forgetting policy. The challenge is how to choose useless instances and keep the most relevant data instances in the learning set. To accomplish this, a fuzzy kernel clustering technique is introduced to find the best learning set. Compared to existing adaptive models for drift in regression learning, FKLL handles drift without delay because the learning set is updated as soon as new data arrives. Additionally, FKLL also overcomes reoccurring drift because the neighbors selected for the learning set are the most relevant instances of the query point.

The remainder of this paper is organized as follows. Section II provides a brief literature review of related areas. Section III presents the fuzzy kernel-lazy learning (FKLL) algorithm along with a detailed explanation. The experiments are described in Section IV. Section V discusses the strengths and limitations of FKLL, and Section VI concludes the paper.

II. Related works and preliminaries

At present, self-adaptive models aimed at handling concept drift can generally be classified into three categories: instance selection, instance weighting, and ensemble learning [5]. The main idea of instance selection is to choose the instances most relevant to the current concept [6]. Instead of discarding the irrelevant instances, instance weighting methods give each instance a weight according to their relevance to the current concept [7]. Ensemble learning combines multiple models together by voting or selecting relevant ones to construct a proper predictive model [8].

Although the study of concept drift has increased over the past decade, research into regression problems is limited. Ikonomovska proposed a batch learning tree model, FIMT-DD, to learn regression patterns from possibly unbounded, high-speed, and time-changing data streams. To the best of our knowledge, this was one of the earliest algorithms to perform explicit drift detection and informed adaptation [3]. In FIMT-DD, rather than proposing any other advanced concept drift detection skills, they directly introduced and embedded a PH-test as an alarm signal for the occurrence of drift. The PH-test is a sequential adaptation of the detection of an abrupt change in the average of a Gaussian signal. Its main principle is to test whether prediction errors in the existing model are increasing. If this increase of errors lasts for a while, a drift is likely to occur in subsequent data streams, because the volatility brought on by random noise is limited and momentary. The advantage of their model is its ability to solve complex problems by recursively fitting different functions into each subspace of an instance space using less assumptions than the original data. The drawback of decision trees, however, as highlighted in [4], is that even a slight drift in the target function may trigger several changes in the model, severely compromising learning efficiency. Moreover, like most detection-based adaptive models, time-lag is a significant consideration because they use part of the data to detect change and adjust the model for the upcoming data stream. PH-tests also limit the method's applications in detecting abrupt drift.

Another successful trial in regression drift data was undertaken in [4], where a random adaptive model rules algorithm based on an ensemble adaptive model improved the original massive online analysis method. Rather than using an adaptive model to specifically solve the concept drift problem in high-speed data streams, their model was also designed to handle other issues in time-evolving streams, such as outlier detection. Further research is needed to demonstrate which of their improvements contributed to better accuracy and to what extent. Still, their model has proven to be effective in different regression problems compared to FIMT-DD and instance-based learner on streams. Time-lag and the limitations that abrupt drift brings have not yet been solved in FIMT-DD because of the way the PH-test combines with the algorithm and the drift detection method does not update.

Dong [9] proposed an online sequential learning algorithm, FP-ELM, for online learning environments. This model is able to reduce the weight of previous training data that negatively affects current performance using a forgetting parameter. A latent assumption of the forgetting parameter is that the newest data contains the most accurate information. This is true in most drift cases, but cannot be applied in reoccurring drift situations, as old patterns will show up repeatedly in the future.

III. Methodology

The proposed algorithm, fuzzy kernel-lazy learning (FKLL) is based on a lazy learning (LL) algorithm, which uses information from the neighborhood of the query point to estimate its output. Its effectiveness is strongly affected by how the neighbors are selected. Rather than a distance-based neighborhood, this paper introduces a fuzzy kernel *c*-means (FKCM) clustering method to limit the range of nearest neighbors. This accurately constrains the instances selected in forecasting model and, as a result, improves prediction accuracy. This section explains FKLL by first briefly introducing FKCM and LL.

A. Fuzzy kernel c-means clustering (FKCM) algorithm

Fuzzy *c*-means clustering was proposed by Bezdek, Ehrlich & Full [10] to replace the hard partitions in data with fuzzy partitions in order to classify indistinct or aberrant data. Let $\boldsymbol{X} = \{\boldsymbol{X_1}, \boldsymbol{X_2}, \cdots, \boldsymbol{X_n}\} \subset \boldsymbol{R^p}$ be a sample in *p*-dimensional Euclidean space. A fuzzy c-partition is defined in (1), where $\boldsymbol{u_{ik}}$ represents the grades of membership of the kth sample in the fuzzy subset $\boldsymbol{u_i}$ of $\boldsymbol{X}$.

$$M_{fc} = \{\boldsymbol{U}_{c\times N} | u_{ik} \in [0,1], \forall\, i,k; \\ \textstyle\sum u_{ik} > 0, \forall\, i;\ \sum u_{ik} = 1, \forall k\} \quad (1)$$

To identify the optimal fuzzy c-partitions in X, the most popular method is to minimize the following cost function,

$$J_m(\boldsymbol{U}, v) = \textstyle\sum_{k=1}^{n}\sum_{i=1}^{c}(u_{ik})^m \cdot \|x_k - v_i\|^2 \quad (2)$$

$$c = number\ of\ clusters\ in\ \boldsymbol{X} \quad (3)$$

$$m = weighting\ exponent \quad (4)$$

$$v = (v_1, v_2, \cdots, v_c) = vectors\ of\ centers \quad (5)$$

$$\|\cdot\| = any\ inner\ product\ norm\ on\ R^p \quad (6)$$

In high-dimensional feature space, direct computation of $\boldsymbol{u_{ik}}$ consumes too much time. Therefore, a kernel function is introduced to improve its effectiveness. We refer readers to [11] for details of the steps.

B. Lazy learning (LL) algorithm

Lazy learning, also referred to as memory-based learning, finds relevant data in the database to answer a particular query (i.e., the output of a target point)[12]. Relevance is often measured using a distance function, with nearby instances having higher relevance. In this paper, predictions are calculated using an average of the k-values of its nearest neighbors $\{\boldsymbol{y_1}, \boldsymbol{y_2}, \cdots, \boldsymbol{y_k}\}$:

$$\hat{\boldsymbol{y}} = \frac{\sum \boldsymbol{y_i}}{\boldsymbol{k}} \tag{7}$$

This estimate minimizes the criterion:

$$\boldsymbol{C} = \sum(\hat{\boldsymbol{y}} - \boldsymbol{y_i})^2 \tag{8}$$

Two factors contribute to LL's accuracy; one is an appropriate statistic to measure relevance, the other is the members of the active learning set. Neighbors are selected from the learning set. The learning set in the original LL is the same as the training set and this became a subset of the training set when FKCM evolved because, at that time, neighbors could only be selected from some special instances in the same cluster as the query point. A larger learning set could possibly contain a greater number of related instances; however, this would lead to a computational cost problem. In static environments, the learning set usually contains historical data or a proportion of historical data, and the set will not change once it is established. As a result, the set may be out of date when drift occurs. The FKCM method solves this problem.

C. Our fuzzy kernel -lazy learning algorithm

The proposed FKLL algorithm is based on an LL algorithm, which uses information from the neighborhood of the query point to estimate its output. To clarify how the original LL has been changed, (7) has been re-written by a product of the parameter vector $\boldsymbol{A}$ and the training output vector $\boldsymbol{y}$ in (9):

$$\hat{y} = \boldsymbol{A}' \cdot \boldsymbol{y} \tag{9}$$

$\boldsymbol{A} = [a_1, a_2, \cdots, a_n]'$ consists of 0 or 1, where 1 represents the ith sample and is one of the neighbors of the query point. We define $\boldsymbol{d_i} = \boldsymbol{D}(\boldsymbol{x_i}, \boldsymbol{x_q})$ as the distance between ith training sample and the query point for each element in $\boldsymbol{A}$, as defined in (10), where $\boldsymbol{d_{(k)}}$ is the kth nearest neighbor of the query point.

$$F: \boldsymbol{x} \rightarrow \boldsymbol{A}$$

$$a_i = F(d_i) = \begin{cases} 1 & d_i \le d_{(k)} \\ 0 & others \end{cases} \tag{10}$$

The FKLL algorithm is an improvement over the original in two aspects: it selects the neighbors more accurately and updates $\boldsymbol{y}$ periodically. Instead of computing the distance between the samples and the query point for all instances, they are separated into clusters based on the fuzzy distance as computed by FKCM. The fuzzy distance $\tilde{d}_i$ is defined in (11). In the original distance, the features are considered equally, as the discrepancy between the samples and the query point are simply added together. However, given real data, the attributes will affect the target variable to different extents, and sometimes an attribute even causes fluctuations in other attributes. In these cases, FKCM also solves the problem. By clustering data into groups, the neighbors of the query point are limited to the range of instances that share similar properties to the query point. A point that is near, but in a different cluster to the query point, will be excluded from the database.

$$\tilde{d}_i = \begin{cases} D(x_i, x_q) & x_i, x_q \ in \ the \ same \ cluster \\ \infty & else \end{cases} \tag{11}$$

To update $\boldsymbol{y}$, a sliding window size is set at the beginning of the algorithm, for example 200 data instances. As the data stream arrives, the oldest 200 data instances are abandoned from the learning set and the newest ones are added. This naive forgetting policy has drawbacks. As highlighted in [12], it is not suitable for learning tasks with high complexity due to insufficient samples. Additionally, the performance of the method also depends on how rapidly and in which way the data changes. For example, this forgetting policy will be inferior to the original method if old patterns reappear, because those patterns will have been forgotten. A safe strategy is to add new data points into the learning set after every sliding window of incoming data. However, the process will become more and more time-consuming as more distances to the new query point need to be re-calculated. Fortunately, a forgetting policy is suitable for FKLL because the level of complexity is represented by high-dimensional distances in FKCM's kernel function. When old patterns reappear, FKCM simply looks for the most similar instances of the pattern for the learning set. The distance will be computed within a subset of $\boldsymbol{X}$ to save time when searching for its nearest neighbors.

During the experiments, FKCM could not find the optimal fuzzy c-partition $\boldsymbol{U}$ because the values of some attributes were too concentrated, and it was difficult to cluster the data consistently within the matrix using the norm function ($\|\cdot\|$) and these cases are potentially irreversible. Therefore, a Kurtosis test has been incorporated prior to clustering. A large Kurtosis value means all the instances are similar in this attribute; therefore, it no longer needs to be included in the FKCM process.

The pseudocode of the FKLL algorithm is shown in Algorithm 1 below:

Algorithm 1: FKLL algorithm

Input:

Input $\boldsymbol{x}$ and output $\boldsymbol{y}$ of the training set
Input of the query points $\boldsymbol{x_q}$

Output:

Estimations of output $\boldsymbol{y_q}$

Parameter:

L: the length of the training set;
window: the size of window;
C: number of clusters;
k: number of neighbors of x_q;
$maxK$: maximum acceptable value of kurtosis

1 Training set = the original training set $[\boldsymbol{x}\ \boldsymbol{y}]$

For i = 1 : *window*
  For each attribute
    if Kurtosis > $maxK$
    Delete this attribute in the FKCM process
  $\boldsymbol{U_{(L+1)\times C} = KFCM([x\ x_q])}$;
  find $c \in \{1,2,\cdots,C\}$
    s.t. $u_{L+1,c} = \max\{u_{L+1,1},\cdots,u_{L+1,C}\}$
  find $\tilde{\boldsymbol{x}} \subset \boldsymbol{x}$
    s.t. $u_{\sim,c} = \max\{u_{\sim,1}, u_{\sim,2},\cdots,u_{\sim,C}\}$
  Compute $\tilde{\boldsymbol{d}} = \boldsymbol{D}(\tilde{\boldsymbol{x}}, \boldsymbol{x_q})$
  Order $\tilde{\boldsymbol{d}}$ increasingly
  Compute $\boldsymbol{A}$
  $\hat{y} = \boldsymbol{A}' \cdot \boldsymbol{y}$
Update $[\boldsymbol{x}\ \boldsymbol{y}]$ by forgetting policy

IV. Experimental evaluation

A. Experimental setup

The experiments in this paper were designed to test the following scenarios: 1) Can FKCM improve the prediction accuracy of the original LL algorithm? How much does FKCM improve performance? 2) How does FKLL compare to other state-of-the-art regression algorithms designed for adaption in drift environments?

As the drift issue is rarely discussed in the pervious resaerch, there is no widely accepted datasets to test in this area. Another big obstacle for selecting testing datasets is the drift is invisible in the real data stream as they are assumed to be determined by potential causes. Therefore, this paper chooses the testing datasets in the previous literatures. Four regression datasets were used to test these scenerios, which can be downloaded from OpenML [13]. They are briefly described in Table I. All four datasets contain real data with the potential for different kinds of drift.

TABLE I. Datasets description

Datasets	*Instances*	*Attributes*	*Details*
Ailerons	13750	40	A control problem of flying an F16 aircraft
CalHousing	20640	8	Median house value in California
House8L	22784	8	Median house value in the US in 1990
House16H	22784	16	Median house value in the US in 1990

The above four datasets were tested by first separating L instances as the training set; the remainder were used as incoming data. Drift handling was performed by FKCM and the learning set was updated after every window of instances.

The parameters of the FKLL algorithm were set at the beginning of prediction process as follows: 1) length of the training set: $L = 2000$; 2) size of window: $window = 200$; 3) number of clusters: $C = 3$; 4) number of neighbors: $k = 24$; and 5) acceptable maximum value of Kurtosis: $maxK = 30$.

Two evaluation metrics were used: MAE and $RMSE$, as defined in (12) and (13) respectively. Additionally, we used normal variance to assess the model's robustness. As the clustering results are affected by our pseudo-random initial guess of the fuzzy partition, the prediction results are not always the same. Therefore, each experiment was repeated 10 times and the mean and variance of MAE and $RMSE$ are provided as the evaluation metrics.

$$MAE = \sum_{window} \frac{|y_q - \hat{y}_q|}{window} \tag{12}$$

$$RMSE = \sqrt{\frac{\sum_{window}(y_q - \hat{y}_q)^2}{window}} \tag{13}$$

For example, the Ailerons dataset contains 13,750 instances. The first 2000 instances, considered to be historical, were selected as the training set. Assume that the real value of the query point is available immediately after its estimation is computed. The prediction results and the 2001st-2200th errors were computed one by one, based on the training set. From the 2201st instance, the previous training set was considered to be outdated. The oldest 200 instances were discarded from the training set and the 2001st-2200th instances were included to create a new training set. The remaining 11,750 instances have 58 windows, therefore 58 pairs of MAE and $RMSE$ values were computed. The average value of these 58 pairs represent the final result of each repetition of the experiment.

B. Experimental results

This section is separated into two parts. The first section compares FKCM to the original LL algorithm. The second compares FKLL with AMRules, FIMT-DD, and Perceptron.

Table II shows the comparison between the original LL and the FKLL algorithm. The FKLL algorithm was tested on two models. One is the mean model, described in Algorithm 1; the other is a linear model where $\hat{y} = \theta' \times \tilde{x}$. The first column is the dataset column, followed by the MAE of LL, FKLL-mean and FKLL-linear. The last three columns show the $RMSE$ of LL, FKLL-mean and FKLL-linear. LL does not contain any random factors, so computing the average values of the repetition is not required. In other words, the value of LL can be interpreted as an average value of the repetitions with a variance equal to 0.

The FKLL-linear model shows the best performance on all four datasets. In comparing the performance of FKLL-mean with FKLL-linear, FKLL-linear is more accurate but less robust. Although the linear model is superior to the mean model in these four special cases, it is still difficult to select the best model for all datasets because the optimal model for different datasets varies substantially. A more accurate conclusion is that the mean model is more robust than the linear model. This is because the mean model gives equal weight to neighbor information, and changes in each neighbor will be weakened by others.

TABLE II. COMPARISON BETWEEN LL AND FKLL

DATASETS	*LL MAE*	*FKLL-mean MAE(Variance)*	*FKLL-linear MAE(Variance)*	*LL RMSE*	*FKLL-mean RMSE(Variance)*	*FKLL-linear RMSE(Variance)*
Ailerons	3.10E-04	2.76E-04(3.90E-13)	1.31E-04(8.81E-14)	3.86E-04	3.48E-04(6.07E-13)	1.75E-04(1.72E-12)
CalHousing	8.30E+04	7.78E+04(5.19E+03)	6.57E+04(4.74E+08)	1.00E+05	9.46E+04(1.30E+04)	7.72E+04(3.33E+05)
House8L	2.84E+04	2.54E+04(2.38E+03)	2.32E+04(7.92E+04)	4.90E+04	4.59E+04(4.03E+03)	4.15E+04(1.36E+07)
House16H	2.87E+04	2.70E+04(3.44E+03)	2.60E+04(4.53E+03)	4.88E+04	4.77E+04(2.43E+03)	4.56E+04(5.56E+04)

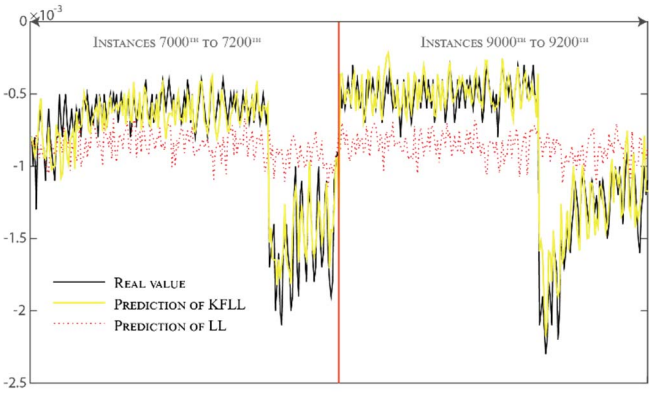

Fig. 1. *Predictions of the 26th and 36th window in Ailerons*

In the linear model, however, if the change happens to be on an attribute whose coefficient is much bigger than the others, the prediction may change greatly. Experimental results support the above argument. Variances in the mean model are smaller than the linear model in terms of both MAE and RMSE for the four datasets, with the exception of the MAE on the Ailerons dataset. The RMSE is similar to the MAE in that FKLL-mean and FKLL-linear both improve accuracy but to different degrees.

In terms of the MAE, fuzzy clustering improves the original LL as shown by the average MAE for FKLL-mean, which is smaller than LL's on the Ailerons, CalHousing, House8L, and House16H datasets. Comparing the performance of FKLL-mean and FKLL-linear models, FKLL-linear model is more accurate than FKLL-mean model but it is less robust. Although the linear model is prior to mean model for these special four cases, it is still difficult to select the best model for all datasets because the optimal model for different datasets varies a lot.

Fig. 1 shows the real data and the predicted value of the 26th and 36th window of the Ailerons dataset. It compares the forecast results of the 7000th to 7200th and the 9000th to 9200th instances. The black line represents the real value of these instances; the pink dotted line represents the predicted value of the original LL; and the yellow line denotes FKLL's evaluation. These data points are separated into two sections by a red line. The points on the left are all from the 26th window and the points on the right are all from the 36th window. The reoccurring drift in the 36th window shows a similar trend to the 26th window. The original LL cannot handle drift issues, so the prediction is almost at the same level. Using FKLL, although the instances in the 26th window are no longer in the learning set, given the 10-window forgetting policy (i.e., the number of forgetting instances equals exactly 2000), the algorithm can still find the most relevant instances in the learning set to handle the recursive drift.

Table II and Fig. 1 demonstrate that FKLL improves upon the original LL and is able to handle reoccurring drift issues well. The next step in this paper validates FKLL's competence against other regression models. Three models were involved in this evaluation: AMRules, FIMT-DD, and Perceptron.

The $RMSE$ and its variance are listed in Table III where figures in bold represent the most accurate method for four datasets and figures in red are the most robust models as they have smallest variance values. The average ranks for accuracy and robustness were also computed based on their RMSE and their corresponding variance.

TABLE III. COMPARISONS BETWEEN FKLL AND OTHER REGRESSION MODELS

DATASETS	*RMSE/ variance (accuracy rank/ robustness rank)*				
	FKLL-Mean	*FKLL-Linear*	*AMRules*	*FIMT-DD*	*Perceptron*
Ailerons	3.48E-04/ 1.26E-08 (2/ 2)	**1.75E-04**/ 3.44E-09 (1/ 1)	4.01E-04/ 9.87E-08 (3/ 3)	4.14E-02/ 1.36E-02 (4/ 5)	1.14E-03/ 3.66E-06 (5/ 4)
CalHousing	9.46E+04/ 2.51E+08 (4/ 1)	7.72E+04/ 2.89E+08 (2/ 2)	8.06E+04/ 2.98E+08 (3/ 3)	1.45E+05/ 5.33E+09 (5/ 5)	**7.51E+04**/ 3.09E+08 (1/ 4)
House8L	4.59E+04/ 4.71E+07 (5/ 2)	4.15E+04/ 4.94E+08 (2/ 5)	**4.12E+04**/ 6.42E+07 (1/ 3)	4.34E+04/ 4.36E+08 (4/ 4)	4.28E+04/ 5.31E+06 (3/ 1)
House16H	4.77E+04/ 4.54E+07 (3/ 2)	4.56E+04/ 5.14E+07 (2/ 4)	**4.37E+04**/ 3.83E+06 (1/ 1)	6.83E+04/ 5.12E+09 (5/ 5)	4.84E+04/3.75E+07 (4/ 2)
Average Rank(Accuracy)	3.5	1.75	2	4.5	3.25
Average Rank (Robustness)	2	3	2.5	4.75	2.75

On the Ailerons dataset, the FKLL-linear model performed the best as indicated by the lowest RMSE and variance. Similarly, AMRules performed best on the House16H dataset. There was no best model for CalHousing and the House8L dataset, as each model has its own drawbacks. For example, Perceptron shows the highest accuracy on CalHousing, but it is not as steady as the others. Perceptron is the most robust on House8L, but is not accurate enough. In practical applications, the optimal model would differ depending on the specific requirements – even on the same dataset. Sometimes precision is required and sometimes high stability is required. Therefore, selecting models like Perceptron in practical applications is not wise because it is difficult to anticipate its effectiveness, accuracy, or robustness in any given situation.

Average ranks are used to measure the overall performance of different models. It is clear that although FKLL is not always the best model, or the most robust model, it is considered the optimal model because it has the highest rank. In terms of accuracy, FKLL-linear is ranked highest with 1.75, and FKLL-mean is the steadiest with an average rank of 2 for robustness.

V. Discussion

Our experimental results demonstrate that the proposed FKLL shows significant improvement over LL in terms of accuracy, and it is able to overcome reoccurring drift. More analysis of the strengths of FKLL follow.

1) The FKLL algorithm responds quickly to incoming instances with drift. 2) Reoccurring drift does not present any problems because instances in the learning set are always selected from the most relevant points to the query point. 3) Over-fitting problems are less likely to appear because LL skips the process of constructing models. 4) Although not mentioned in the previous sections, FKLL is easy to combine with other drift detection techniques. So far, most drift adaption models combine global drift detection techniques, rather than local ones. FKLL can combine both global and local techniques simply by updating the learning set.

However, the FKLL algorithm has some limitations. FKCM's clustering results are random, which means the forecasts are also random; an appropriate method is needed to determine the number of clusters.

VI. Conclusion and Further Study

This paper proposes a self-adaptive algorithm, FKLL, to solve drift in regression problems. FKLL is based on a lazy learning algorithm and uses the *k*-nearest neighbors of the query point to infer its attributes instead of constructing a global model. Compared to traditional forecasting models, LL costs less to model but requires more effort to choose the neighbors from the learning set. Additionally, in dynamic environments, updating the learning set is critical to solving drift problems. Therefore, the number of instances in the learning set affect both speed and accuracy. A fuzzy kernel *c*-means clustering method is introduced to determine the optimal learning set for LL.

Before computing the distance of all the instances to the query point, FKCM separates them into different categories. Only instances in the same category as the query point are selected for the learning set. The two advantages of clustering are a reduction in the estimation time and self-adaptation to changing environments. Compared to previous models, FKLL responds swiftly to potential incoming drift because the learning set is updated with every new window of instances. Additionally, FKLL is able to handle recursive drift by looking for the existing instances most relevant to incoming patterns. Four real-world datasets were used to validate the effectiveness of the proposed FKLL algorithm. These evaluations show that FKLL improves upon the original LL algorithm in terms of precision and robustness compared to other state-of-the-art drift handling methods in regression cases.

In future studies, the forgetting policy will be improved using other drift detection skills. FKLL's local model will also be further developed to handle special kinds of data.

Acknowledgment

The work presented in this paper was supported by the Australian Research Council (ARC) under discovery grant DP150101645.

References

[1] A. Tsymbal, "The problem of concept drift: definitions and related work," *Comput. Sci. Dep. Trinity Coll. Dublin*, vol. 4, no. C, pp. 2004–15, 2004.

[2] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *Comput. Surv.*, vol. 46, no. 4, pp. 1–37, 2014.

[3] E. Ikonomovska, J. Gama, and S. Dzeroski, "Learning model trees from evolving data streams," *Data Min. Knowl. Discov.*, vol. 23, no. 1, pp. 128–168, 2011.

[4] J. Duarte, João and Gama, "Adaptive Model Rules from High-Speed Data Streams," vol. 10, no. 3, 2016.

[5] N. Lu, J. Lu, G. Zhang, and R. Lopez De Mantaras, "A concept drift-tolerant case-base editing technique," *Artif. Intell.*, vol. 230, pp. 108–133, 2016.

[6] S. J. Delany, P. Cunningham, A. Tsymbal, and L. Coyle, "A case-based technique for tracking concept drift in spam filtering," *Knowledge-Based Syst.*, vol. 18, no. 4–5, pp. 187–195, 2005.

[7] P. Zhang, X. Zhu, and Y. Shi, "Categorizing and mining concept drifting data streams," *Int. Conf. Knowl. Discov. Data Min.*, p. 8, 2008.

[8] L. L. Minku, A. P. White, and X. Yao, "The impact of diversity on online ensemble learning in the presence of concept drift," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 5, pp. 730–742, 2010.

[9] D. Liu, Y. X. Wu, and H. Jiang, "FP-ELM: An online sequential learning algorithm for dealing with concept drift," *Neurocomputing*, vol. 207, pp. 322–334, 2015.

[10] J. C. Bezdek, R. Ehrlich, and W. Full, "FCM: The fuzzy c-means clustering algorithm," *Comput. Geosci.*, vol. 10, no. 2–3, pp. 191–203, 1984.

[11] X. Yang, G. Zhang, J. Lu, and J. Ma, "A kernel fuzzy c-Means clustering-based fuzzy support vector machine algorithm for classification problems with outliers or noises," *IEEE Trans. Fuzzy Syst.*, vol. 19, no. 1, pp. 105–115, 2011.

[12] D. W. Aha, *Lazy learning*. Kluwer academic publishers, 1997.

[13] "OpenML." [Online]. Available: http://www.openml.org/home.

[14] R. Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection," *Int. Jt. Conf. Artif. Intell.*, vol. 14, no. 12, pp. 1137–1143, 1995.