Reframing on Relational Data

Chowdhury Farhan Ahmed, Clément Charnay,
Nicolas Lachiche(✉), and Agnès Braud

ICube Laboratory, University of Strasbourg, Strasbourg, France
{cfahmed,charnay,nicolas.lachiche,agnes.braud}@unistra.fr

Abstract. Construction of aggregates is a crucial task to discover knowledge from relational data and hence becomes a very important research issue in relational data mining. However, in a real-life scenario, dataset shift may occur between the training and deployment environments. Therefore, adaptation of aggregates among several deployment contexts is a useful and challenging task. Unfortunately, the existing aggregate construction algorithms are not capable of tackling dataset shift. In this paper, we propose a new approach called reframing to handle dataset shift in relational data. The main objective of reframing is to build a model once and make it workable in many deployment contexts without retraining. We propose an efficient reframing algorithm to learn optimal shift parameter values using only a small amount of labelled data available in the deployment. The algorithm can deal with both simple and complex aggregates. Our experimental results demonstrate the efficiency and effectiveness of the proposed approach.

Keywords: Relational data mining · Aggregates · Dataset shift · Supervised learning · Stochastic optimization

1 Introduction

Data mining discovers hidden knowledge from databases. Relational data mining [7] is an important field of data mining which aims at extracting interesting and useful knowledge from relational databases. In a relational database, data are stored in multiple relations/tables and connected through some common keys/fields. One to many relationships are a special kind of link where each tuple of a primary table may be linked to several keys of a secondary table. An example is *Department* table and *Student* table. Here *Department* is a primary table and one department has many students in the *Student* table which implies a one to many relationship. This type of relationship is very useful for representing several real-life scenarios for example customers and purchases in market basket databases, urban blocks and buildings in geographical databases, molecules and atoms in chemical databases, phone numbers and call records in telecommunication databases, and so on.

Aggregates [4,8,12,19,20] are used to carry information from the secondary to the primary table. Consider a given customer has many transactions in the

J. Davis and J. Ramon (Eds.): ILP 2014, LNAI 9046, pp. 1–15, 2015.
DOI: 10.1007/978-3-319-23708-4_1

Purchase table and we want to add this information in the primary table *Customer*. Then the sum aggregate can be used to accumulate it as an attribute *total_purchase* in the primary table. Complex aggregates [4,19,20] may have multiple conditions and/or thresholds inside them. They are very useful for propositionalization [8,12,13]. Consider the following complex aggregate CA_1. It can be used to define a threshold of two classes of urban blocks in a city. For example, urban blocks satisfying CA_1 fall into class X (e.g., individual housing area), and otherwise into class Y (e.g., apartment housing area).

$$CA_1 : count(buildings\ such\ that\ area < 200) > 30$$

On the other hand, reuse of learnt knowledge is of critical importance in the majority of knowledge-intensive application areas, particularly because the operating context can be expected to vary from training to deployment. Dataset shift [14] is a crucial example of this phenomenon where training and testing datasets follow different distributions. Consider the above complex aggregate CA_1 is used to train a classifier in City 1 and we need to classify the data of City 2 and City 3 while the average area of individual houses in City 2 is comparatively smaller than City 1 and housing in a given area is more compact in City 3 than City 1. Suppose CA_2 and CA_3 represent the appropriate complex aggregates to classify the data of City 2 and City 3, respectively.

$$CA_2 : count(buildings\ such\ that\ area < 100) > 30$$
$$CA_3 : count(buildings\ such\ that\ area < 200) > 50$$

Existing relational data mining algorithms use retraining (building a new model in the deployment from the very beginning) to classify the data of City 2 or City 3. Can we use the existing model of City 1 and avoid this costly retraining? This motivates us to design a new approach of reframing on relational data. The contributions of our approach are described as follows

- We develop a new idea of reframing on relational data, and design an efficient algorithm, called Reframing Aggregates with Stochastic Hill Climbing (RASHC) for reframing both the aggregate condition and the aggregate output.
- It can handle both simple and complex aggregates.
- Both classification and regression tasks can be addressed.
- Our approach has the capability of tackling different changes in data distributions and decision functions and make the existing model workable in different deployment environments.
- It can discover optimal shift parameter values although a small amount of labelled deployment data are available.
- We experimentally show the efficiency and effectiveness of the proposed algorithm using synthetic and real-life datasets. In particular, we present the existence of dataset shift problem in a real-life dataset [1,9] and capability of our algorithm to approximate these unknown real-life shifts accurately.

The remainder of this paper is organized as follows. Section 2 discusses related work. In Sect. 3, we describe our proposed reframing approach for relational data. In Sect. 4, our experimental results are presented and analyzed. Finally, conclusions are drawn in Sect. 5.

2 Related Work

A way of mining relational databases is propositionalization [8,12,13] which transforms the data of primary and secondary tables into a single attribute value table. There are some existing methods of propositionalization such as Relaggs [12], Cardinalization and Quantiles [8]. TILDE [3] introduced a logical representation for relational decision trees and has been extended to handle complex aggregates. A method called FORF (first order random forests) [19] has been proposed for learning relational classifiers with complex aggregates. Another research work was done to refine aggregate conditions in relational learning [20]. Charnay et al. [4] proposed a method for incremental construction of complex aggregates.

On the other hand, several methods [5,11,14,18,22] have been proposed to handle the situation where training and testing environments are different. Input variable shift is most often known as covariate shift [14] in the data mining and machine learning literature. Some research works have been done to tackle covariate shift [2,14,18] where shifts in input variables are considered over different deployment contexts. For example, a new variant of cross validation, called Importance Weighted Cross Validation (IWCV), is proposed by assuming that the ratio of the test and training input densities is known at training time [18]. According to this ratio, it applies different weights to each input during cross validation to handle covariate shift. Another method, called Integrated Optimization Problem (IOP), finds contribution of each training instance to the optimization problem ideally needs to be weighted with its test-to-training density ratio [2]. It uses a discriminative model that characterizes the possibility of an instance to occur in the test set rather than in the training set. Instantiating the general optimization problem leads to a kernel logistic regression and an exponential model classifier for covariate shift [2]. In order to cope with covariate shift, Kernel Mean Matching (KMM) approach [10] reweighs the training data such that the means of the training and test data in a reproducing kernel Hilbert space (i.e., in a high dimensional feature space) are close.

Recently, an input transformation based approach, called GP-RFD (Genetic Programming-based feature extraction for the Repairing of Fractures between Data), has been proposed which can handle general dataset shift [16]. At first, it creates a classifier C for a training dataset A and considers B as a test dataset with a different distribution. To discover optimal transformation, it applies several genetic operators (e.g., selection, crossover, mutation) on B and creates dataset S. Subsequently, it applies C on dataset S and calculates the accuracy in order to determine the best transformation. This approach is computationally expensive and needs a large amount of deployment data to be labelled. In

addition, another related research area is transfer learning [6,17] which learns a new model for the deployment data reusing knowledge from the base model, but it often (re)trains a new model rather than adapting the original.

However, the existing dataset shift adapting algorithms need retraining to be applied in a different deployment environment and hence are not suitable for reframing which aims at building a model once and make it (i.e., the same model) workable in several deployment environments without retraining. Moreover, they are not capable of tackling relational data. Therefore, we propose a new approach called reframing to handle dataset shift on relational data.

3 Our Proposed Approach

The main reason behind changing the condition and/or output of an aggregate is the change in data distribution. Suppose M is a model built with a base learner C using some labelled training data T_{tr}; and CA_{tr} and CA_d are complex aggregates to classify the training and deployment data, respectively. In this example, we have shown one aggregate condition for simplicity. But, there may be multiple aggregate conditions in a complex aggregate. However, if we want to use M to classify the deployment data, we have to transform the shifted parameter values of CA_d to the appropriate parameter values of CA_{tr}. By using a transformation function T and a few labelled deployment data T_d, our approach will find optimal shift parameter values for this transformation. Thus, it will be able to use the existing model that has been built with the training data. In the first and second subsections, we demonstrate the concept of reframing aggregate condition and output, respectively. Finally, our proposed reframing algorithm is presented.

$$CA_{tr} : AggF(buildings\ such\ that\ attVal_{tr} < aggCond_{tr}) > aggOutCond_{tr}$$
$$CA_d : AggF(buildings\ such\ that\ attVal_d < aggCond_d) > aggOutCond_d$$

3.1 Reframing Aggregate Condition

For reframing aggregate condition, T transforms the input attribute value $attVal_d$ of deployment to the appropriate (approximate) corresponding input attribute value $attVal_{\widehat{tr}}$ of training by an equation e.g., Eq. 1 so that they can properly be classified.

$$attVal_{\widehat{tr}} = \alpha(attVal_d) + \beta \tag{1}$$

Hence, for reframing the aggregate condition, our reframing model M evaluates the following

$$\begin{aligned} AggF(T(attVal_d) < aggCond_{tr}) &= AggF((\alpha(attVal_d) + \beta) < aggCond_{tr}) \\ &= AggF(attVal_{\widehat{tr}} < aggCond_{tr}) \end{aligned}$$

The main challenging task is to find optimal shift parameter values (values of α, β in this case) so that it can transform the deployment data with these values and able to predict $attVal_{\widehat{tr}}$ very close to $attVal_{tr}$. Our method can

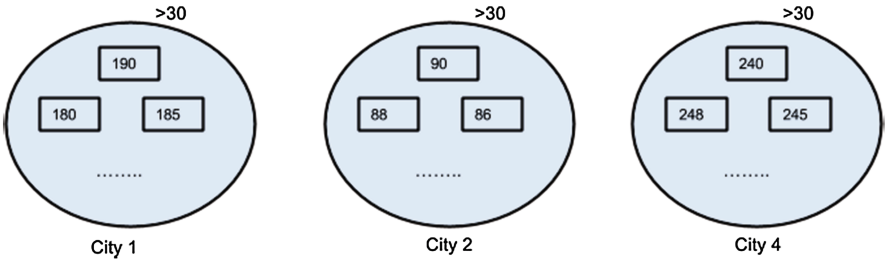

Fig. 1. Reframing aggregate condition.

perform this task properly. In addition, it can learn multiple $\{\alpha, \beta\}$ pairs if there are multiple aggregate conditions in a complex aggregate. Hence, it enables M to be built once and used over several deployment contexts without any modification. On the other hand, retraining means building a new model M_{new} for every deployment context using the few labelled data available over there while not using any knowledge of M at all. And, using a base model to other contexts means that applying model M to other deployment contexts directly, i.e., without any reframing or retraining. Accordingly, reframing will be useful to handle most of the cases of dataset shift compared to apply the base model or retraining.

$$CA_4 : count(buildings\ such\ that\ area < 250) > 30$$

Now, consider the scenario in Fig. 1 where a training environment is City 1 and deployment environments are City 2 and City 4. Complex aggregates of these cities are described in CA_1, CA_2 and CA_4, respectively. An individual housing block in City 1 contains at least 30 buildings with area less than 200. But in City 2, buildings are comparatively smaller (half compared to the building area of City 1) and in City 4 they are comparatively larger(larger than 50 units compared to the building area of City 1). However, their cardinality (aggregate output) is same like City 1 (i.e., 30). We can use the following transformation to re-scale the deployment data to fit in our existing model. For City 2, we can use $\alpha = 2$ and $\beta = 0$ (Eq. 1), so that a building area of 100 in City 2 can be mapped to its appropriate value of 200 $(100 \times 2 + 0)$ in City 1 and properly be classified by the existing model of City 1. Similarly, we can use $\alpha = 1$ and $\beta = -50$ for reframing the data of City 4.

3.2 Reframing Aggregate Output

Here, we have to transform the aggregate output with T in a similar way as follows

$$\begin{aligned} T(AggF(attVal_d < aggCond_d)) &= T(aggOut_d) \\ &= \alpha(aggOut_d) + \beta \\ &= aggOut_{\widehat{tr}} \end{aligned} \tag{2}$$

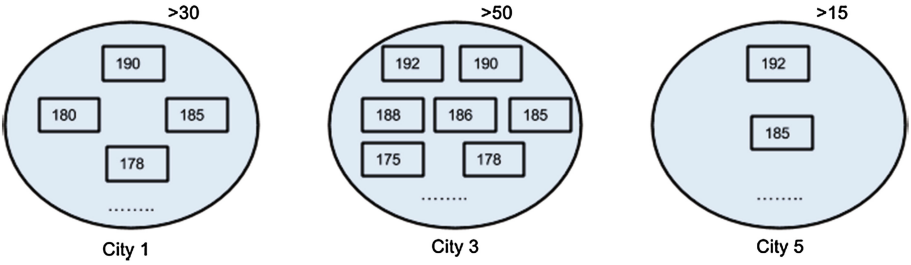

Fig. 2. Reframing aggregate output.

Our reframing approach tries to make $aggOut_{\widehat{tr}}$ very close to $aggOut_{tr}$ so that it can properly be classified by $aggOutCond_{tr}$. Consider the scenario in Fig. 2 where a training environment is City 1 and deployment environments are City 3 and City 5. Complex aggregates of these cities are described in CA_1, CA_3 and CA_5, respectively. A residential housing block in City 1 contains at least 30 buildings with area less than 200. But in City 3, even though buildings have the same size, the blocks are very crowded (too many buildings in a block) and contain at least 50 buildings. On the other hand, the reverse situation occurs in City 5, i.e., even though buildings have the same size, the blocks contain fewer buildings compared to City 1 (at least 15 buildings in a block).

$$CA_5 : count(buildings\ such\ that\ area < 200) > 15$$

For City 3, we can use $\alpha = 1$ and $\beta = -20$ (Eq. 2), so that an aggregate output of 50 in City 3 can be mapped to its appropriate value of 30 ($50 \times 1 - 20$) in City 1 and properly be classified by the existing model of City 1. Similarly, we can use $\alpha = 2$ and $\beta = 0$ for reframing the aggregate output of City 5.

3.3 RASHC: Our Proposed Algorithm

In this subsection, we present our reframing algorithm on relational data called RASHC (Reframing Aggregates with Stochastic Hill Climbing). We have considered a relational decision tree learner [4] to build the classification model M. For reframing, we consider there are n conditions in M, where each condition represents either an aggregate condition c or an aggregate output o. Consider the models of three cities given in Fig. 3 where City 1 is the training context and City 2 and City 3 are the deployment contexts. Please note that City 2 has different parameter values in the root node while leaf nodes have the same parameter values. On the other hand, City 3 has different parameter values in all the nodes (six parameters in three nodes). As described earlier, our method will learn optimal shift parameter values for these aggregates with a small amount of available labelled deployment data. In this example, it will learn total six (α, β) pairs. If there is no change, it can safely learn $\alpha = 1, \beta = 0$ values.

The RASHC algorithm is presented in Fig. 4. It uses a stochastic hill climbing search to find optimal shift parameter values. Indeed, the solution of our

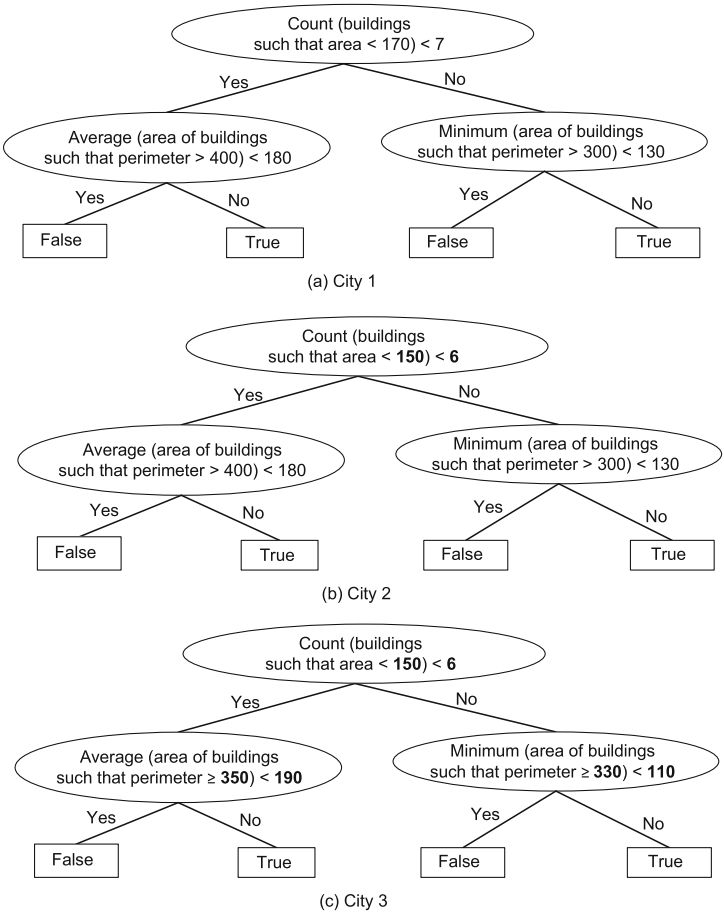

Fig. 3. Relational decision trees for three different cities.

problem space is non-convex, and according to [21] this technique provides optimal solution in most of the cases of this situation. Therefore, we have chosen this technique for discovering our reframing solutions. The RASHC algorithm (Fig. 4) builds a classification model M using the training labelled data (line 2). The transformation function T uses the few available labelled data of deployment (T_d) to perform the transformation (line 4). The loop described in lines 7 to 18 learns global optimal shift parameter values. In the nested loop (lines 8 to 13), it learns a set of local optimal shift parameter value using the HillClimbing procedure shown in lines 21 to 41. The shouldRestart procedure (invoked in line 14) updates $bestShift$ with the value of current Sft and makes the *count* value equal to zero, if the current Sft produces better performance than the current $bestShift$. Otherwise, *count* is incremented by one. It returns True if the *count* is less than 10, and False (i.e., $bestShift$ remains unchanged for the last 10 times)

Input: A base learner C, e.g., relational decision tree learner;
Labelled training data T_{tr}; Few labelled data of the deployment T_d;
Unlabelled test data T_{tst}; Precision of adjustment $p > 0$.
Output: Optimal shift parameter values and the predicted class labels.

begin
  Train a model M by using C from T_{tr};
  Let n be the no. of conditions in M; where each condition represents either an aggregate condition c or an aggregate output o;
  Let T be a transformation function which shifts the attribute values $\forall c \in M$ and the output values $\forall o \in M$ by Eq. 1 and Eq. 2, respectively;
  $Sft = (\alpha_1, \beta_1, ..., \alpha_n, \beta_n) = (1, 0, ..., 1, 0)$;
  $bestSft = Sft$; $Continue = True$; $p = 1$; $count = 0$;
  while *Continue=True* **do**
    repeat
      $Sft_{old} = Sft$;
      for $i = 1$ *to* $2n$ **do**
        HillClimbing(Sft, i, p);
      end
    until $Sft = Sft_{old}$;
    $Continue$ = shouldRestart$(Sft, count)$;
    if $Continue = True$ **then**
      Select randomly another Sft value;
    end
  end
  Apply $bestSft$ to T_{tst} and **Output** the class labels by using M;
end
Procedure HillClimbing(Sft, i, p)
begin
  $Sft^+ = Sft^- = Sft$; $Sft^+[i] = Sft[i] + p$; $Sft^-[i] = Sft[i] - p$;
  if $Acc(M(T(Sft^+))) > Acc(M(T(Sft)))$ **then**
    $\delta = 1$;
  end
  else if $Acc(M(T(Sft^-))) > Acc(M(T(Sft)))$ **then**
    $\delta = -1$;
  end
  else
    return;
  end
  $Sft_{prev} = Sft$;
  repeat
    $Incr = 1$; $Sft = Sft_{prev}$;
    repeat
      $Sft_{next} = Sft_{prev} = Sft$; $Sft[i] = Sft[i] + (Incr * p * \delta)$;
      $Incr = Incr * 2$; $Sft_{next}[i] = Sft[i] + (Incr * p * \delta)$;
    until $Acc(M(T(Sft_{next}))) < Acc(M(T(Sft)))$;
  until $Incr \leq 2$;
end

Fig. 4. The RASHC algorithm.

otherwise. Accordingly, we need to start with another Sft value if the returned value is True. This part is performed in lines 15 to 17. Finally, learnt optimal parameter values of $bestShift$ are applied to transform the input values of T_{tst} and M is used to predict the class labels (line 19).

4 Experimental Results

We have performed several experiments on synthetic and real-life datasets to show the efficiency and effectiveness of our approach. We show the performance of reframing with respect to retraining and the base model. We have used a relational decision tree [4] as the base learner.

4.1 Performance in Synthetic Datasets

Three synthetic relational datasets have been generated to represent building blocks data of three different cities. The decision functions have been taken from the example given in Fig. 3. For each city, there are two tables. The primary and the secondary tables are called as *Block*(***Block_Id***, *Class*) and *Building*(***Building_Id***, *Block_Id*, *Building_area*, *Building_perimeter*), respectively. We have generated 500 data for blocks of a particular city. For each block, the number of buildings is taken from a Geometric distribution with a mean of 10. In addition, Gaussian distributions have been used to generate the data of area ($\mu = 180$, $\sigma = 40$) and perimeter ($\mu = 450$, $\sigma = 50$) of a building.

At first, we report the accuracies of the base models of these three cities in Table 1. Now, we assume training data are available for City 1 and only a small amount of labelled data in City 2 and City 3. The number of labelled data available in the deployment is denoted as N, and we have considered $N = 10\,\%$ here. As mentioned in the previous sections, we have two straightforward options in this situation. Firstly, we can use the base model of City 1 directly in City 2

Table 1. Performance of the Base models of different cities.

Base	Accuracy (%)
City 1	98.4
City 2	99.8
City 3	86.4

Table 2. Performance (accuracy (%)) of the reframing algorithm.

Deployment $N = 10\,\%$	RASHC	Retraining	Base (City-1)
City 2	97.78	43.56	88.89
City 3	82.67	59.11	72.67

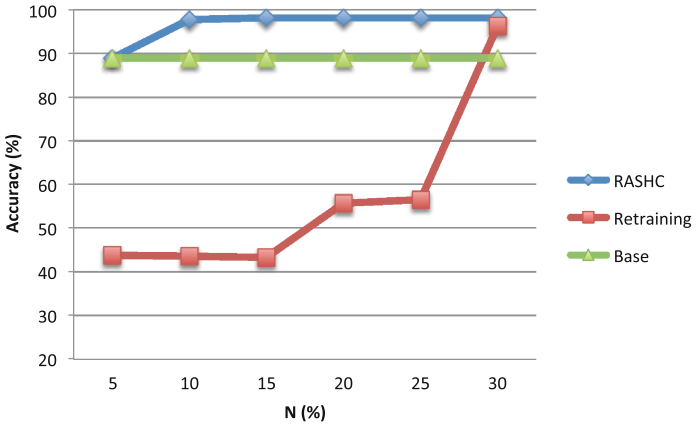

Fig. 5. Learning curves for reframing, retraining and base model (City 1) considering City 1 as training and City 2 as deployment.

and City 3. Secondly, we can develop a new model (retraining) in City 2 and City 3. The results are shown in Table 2. Please note that, the accuracy of the base model of City 1 is 98.4 %, but when it is applied in City 2 and City 3, it can achieve only 88.89 % and 72.67 % of accuracies, respectively. While accuracy of the base models of City 2 and City 3 are 99.8 % and 86.4 %, respectively. It clearly shows that dataset shift exists among these cities. Table 2 clearly indicates that the performance of retraining model with only $N = 10\,\%$ of labelled deployment data is bad. On the other hand, our reframing algorithm RASHC achieves 97.78 % and 82.67 % of accuracies in City 2 and City 3, respectively; which are very close to the accuracy value of the base model of the corresponding cities.

Subsequently, we show learning curves in Fig. 5 for reframing, retraining and base model (City 1) considering City 1 as training and City 2 as deployment in order to represent the learning capabilities of these models more elaborately. By using the base classifier, we can only get an accuracy of 88.89 % and it is constant for all N as it does not learn anything from the labelled deployment data. From Fig. 3, we can see that optimal shift parameter values from City 2 to City 1 would be $\beta = 20$ and $\beta = 1$ for aggregate condition and aggregate output, respectively in the root node while values of the other nodes should be kept unchanged. When $N = 5\,\%$, RASHC is equivalent to the base model. It also illustrates that the accuracy of RASHC does not fall below the base classifier. When $N = 10\,\%$, it achieves around 98 % of accuracy which is very close to the original base model of City 2. On the other hand, retraining can reach nearly RASHC when $N = 30\,\%$.

4.2 Performance in a Real-Life Dataset with Artificial Shifts

Here, we have used a real-life relational dataset, the PKDD'99 *Financial*[1], which contains some information of a bank regarding accounts, loans, transactions etc. The bank wants to distinguish their clients as good or bad before providing them loans according to their financial history. Therefore, we have taken two tables *Loan* (as primary table) and *Transactions* (as secondary table) for our experiment. However, the *Financial* dataset is highly imbalanced (around 90 % loans are positive) between the positive and negative classes, hence, we have taken 100 loans (50 positive and 50 negative classes) and their corresponding 5,755 transactions. This dataset does not have any dataset shift. Recently, research has been done [14–16] in non-relational domains to inject different kinds of artificial shift in real-life datasets because of the unavailability of shifts inside them. Similarly, we have injected three kinds of shift (low, medium and high) in this real-life relational dataset.

The accuracy of the base model is 91 % when no shift is used. Table 3 reports how the base model is affected by the extremeness of shift values. Like the previous experiments, we have used $N = 10\,\%$ here. The performance of retraining shows that it is not affected by the shifts as it rebuilds the model every time with the labelled data available in the deployment rather than learning the shifts. Furthermore, it does not use any knowledge from the training section. But obviously, it cannot rebuild a good new model with this few available labelled data. On the other hand, it is noticeable that our RASHC algorithm has learnt optimal shift parameter values in all the cases and achieved good accuracies.

These experimental results demonstrate that our approach is very useful and effective to learn optimal shift parameter values, using a small amount of available labelled data (around 10 %) in deployment. It is remarkably better than retraining and its performance does not fall below the performance of the base model. Furthermore, the capability of our approach is shown for handling different changes in decision functions. Therefore, it can be built in one training context and used over several deployment contexts.

Table 3. Accuracy (%) comparison in the real-life *Financial* dataset

Shift	RASHC	Retraining	Base
Low	90	45.56	81.11
Medium	87.78	45.56	73.33
High	83.33	45.56	55.56

4.3 Performance in a Real-Life Dataset with Real Shifts

A real-life dataset, called *Bike Sharing* dataset [1,9], has been used here to show the presence of real shifts and the capability of our method to handle these shifts

[1] http://lisp.vse.cz/pkdd99/Challenge/chall.htm.

Table 4. Performance of Base models (different seasons) of the *Bike Sharing* dataset.

Measure	Spring $\|Days\| = 181$ $\|Hours\| = 4242$	Summer $\|Days\| = 184$ $\|Hours\| = 4409$	Fall $\|Days\| = 188$ $\|Hours\| = 4496$	Winter $\|Days\| = 178$ $\|Hours\| = 4232$
MAE	1068.623	1216.842	1305.509	1272.703
RMSE	1354.019	1497.588	1570.728	1568.031

properly. The *Bike Sharing* dataset contains usage logs of a bike sharing system called Capital Bike Sharing (CBS) at Washington, D.C., USA for two years (2011 and 2012). It was prepared by Fanaee-T and Gama [9], and is publicly available in UCI Machine Learning Repository [1].

The dataset contains bike rental counts hourly and daily based on the environmental and seasonal settings. The *hour.csv* file [1] contains 17,379 records where the values of bike sharing counts have been aggregated on hourly basis. Similarly the *day.csv* file [1] aggregates the bike sharing counts on daily basis and contains 731 records. The input variables contain day, hour, season, workday/holiday and some weather information such as temperature, feels like temperature, humidity and wind speed. We have considered the day format as the primary table named *Day*(***Day_Id***, *Bike rental count*) and the hour format as secondary table named *Hour*(***Hour_Id***, *Day_Id*, *Temperature*, *Feels like temperature*, *Humidity*, *Windspeed*). Our goal is to solve a regression task in this relational database which predicts the daily bike rental counts based on hourly given weather information.

Since this dataset contains data of different seasons (1:Spring, 2:Summer, 3:Fall, 4:Winter in the Season attribute), we have divided the original dataset according to different seasons in order to observe dataset shift. Table 4 shows the performance of the base models of different seasons using 10-fold cross validation. We have used the MAE (mean absolute error) and RMSE (root mean square error) measures to represent the performance in regression. To show a real-life dataset shift, we deploy the regression model trained in Spring into Summer, Fall and Winter. Please notice that the base model of Spring has a MAE value of 1068.623 in Spring (Table 4), but MAE values of 1544.563, 3193.662 and 1658.313 in Summer, Fall and Winter, respectively (Table 5). While the base models of Summer, Fall and Winter have MAE values of 1216.842, 1305.509 and 1272.703, respectively (Table 4). It clearly shows that dataset shift exists among these seasons.

Now, we run our reframing algorithm RASHC on this dataset considering Spring as training; Summer, Fall and Winter as deployments with only 10 % available labelled data at each deployment. Results are reported in Table 5. It is noticeable that the performance of RASHC is very close to the base model of that particular deployment season. For example, the base model of Summer has MAE and RMSE values of 1216.842 and 1497.588, respectively; while RASHC algorithm has MAE and RMSE values of 1314.306 and 1609.793, respectively in

Table 5. Performance comparison in the *Bike Sharing* dataset.

Deployment $N = 10\,\%$	Measure	RASHC	Retraining	Base (Spring)
Summer	MAE	1314.306	2898.77	1544.563
	RMSE	1609.793	3181.496	1954.986
Fall	MAE	1366.234	1495.57	3193.662
	RMSE	1645.177	1849.431	3691.34
Winter	MAE	1446.568	1793.186	1658.313
	RMSE	1752.089	2155.191	1988.834

this season. Moreover, reframing algorithm RASHC is significantly better than retraining and the base model of Spring.

Experimental results in this real-life dataset demonstrate that our approach is quite capable of learning optimal shift parameter values, using a small amount of labelled data (around 10 %) at deployment, in a real-life environment where the nature of a shift is unknown from source to deployment. These results also reveal the applicability of linear shift in real-life domains by clearly expressing its strength to tackle these unknown dataset shifts between one training and different deployment contexts. In addition, applicability of our approach can also be observed in sequential time series data mining.

5 Conclusions

In this paper, we have proposed a new approach of reframing on relational data. We have designed an efficient algorithm to learn optimal shift parameter values for aggregate condition and output. It can deal with both simple and complex aggregates. Our approach has the capability of tackling different changes in data distributions and decision functions, and make the existing model workable in different deployment environments. Even though a small amount of labelled data are available in the deployment, it can discover desired optimal parameter values to handle the actual dataset shift. Using synthetic and real-life datasets, we have experimentally shown the efficiency and effectiveness of the proposed algorithm. In the worst case, our approach is equivalent to the base model which is an important property for dataset shift adaptation. Furthermore, it is quite suitable for those environments where retraining is not applicable to learn the decision functions. Finally, we have presented the existence of dataset shift in a real-life dataset by considering one season as training context and other seasons as deployment contexts. We have demonstrated the capability of our approach to approximate these unknown real-life dataset shifts accurately.

Acknowledgements. This work was supported by the REFRAME project granted by the European Coordinated Research on Long-term Challenges in Information and Communication Sciences & Technologies ERA-Net (CHIST-ERA).

References

1. Bache, K., Lichman, M.: UCI machine learning repository (2013) http://archive.ics.uci.edu/ml/
2. Bickel, S., Brückner, M., Scheffer, T.: Discriminative learning under covariate shift. J. Mach. Learn. Res. **10**, 2137–2155 (2009)
3. Blockeel, H., De Raedt, L.: Top-down induction of first-order logical decision trees. Artif. Intell. **101**(1–2), 285–297 (1998)
4. Charnay, C., Lachiche, N., Braud, A.: Incremental construction of complex aggregates: counting over a secondary table. In: Late Breaking Papers of the 23rd International Conference on Inductive Logic Programming (ILP), pp. 1–6 (2013)
5. Charnay, C., Lachiche, N., Braud, A.: Pairwise optimization of bayesian classifiers for multi-class cost-sensitive learning. In: Proceedings of the 25th IEEE International Conference on Tools with Artificial Intelligence (ICTAI), pp. 499–505 (2013)
6. Davis, J., Domingos, P.: Deep transfer via second-order markov logic. In: Proceedings of the 26th Annual International Conference on Machine Learning (ICML), pp. 217–224 (2009)
7. Dzeroski, S.: Relational data mining. In: Maimon, O., Rokach, L. (eds.) Data Mining and Knowledge Discovery Handbook, pp. 887–911. Springer, New York (2010)
8. El Jelali, S., Braud, A., Lachiche, N.: Propositionalisation of continuous attributes beyond simple aggregation. In: Riguzzi, F., Železný, F. (eds.) ILP 2012. LNCS, vol. 7842, pp. 32–44. Springer, Heidelberg (2013)
9. Fanaee-T, H., Gama, J.: Event labeling combining ensemble detectors and background knowledge. Prog. Artif. Intell. **2**(2–3), 113–127 (2014)
10. Gretton, A., Smola, A., Huang, J., Schmittfull, M., Borgwardt, K., Schölkopf, B.: Covariate shift by kernel mean matching. In: Dataset Shift in Machine Learning, pp. 131–160. MIT Press, Cambridge (2009)
11. Hernández-Orallo, J.: ROC curves for regression. Pattern Recogn. **46**(12), 3395–3411 (2013)
12. Krogel, M.-A., Wrobel, S.: Transformation-based learning using multirelational aggregation. In: Rouveirol, C., Sebag, M. (eds.) ILP 2001. LNCS (LNAI), vol. 2157, p. 142. Springer, Heidelberg (2001)
13. Lachiche, N.: Propositionalization. In: Sammut, C., Webb, G.I. (eds.) Encyclopedia of Machine Learning, pp. 812–817. Springer, New York (2010)
14. Moreno-Torres, J.G., Raeder, T., Alaíz-Rodríguez, R., Chawla, N.V., Herrera, F.: A unifying view on dataset shift in classification. Pattern Recogn. **45**(1), 521–530 (2012)
15. Moreno-Torres, J.G.: Dataset shift in classification: terminology, benchmarks and methods. Ph.D thesis (2013)
16. Moreno-Torres, J.G., Llorà, X., Goldberg, D.E., Bhargava, R.: Repairing fractures between data using genetic programming-based feature extraction: a case study in cancer diagnosis. Inf. Sci. **222**, 805–823 (2013)
17. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. Knowl. Data Eng. **22**(10), 1345–1359 (2010)
18. Sugiyama, M., Krauledat, M., Müller, K.R.: Covariate shift adaptation by importance weighted cross validation. J. Mach. Lear. Res. **8**, 985–1005 (2007)
19. Van Assche, A., Vens, C., Blockeel, H., Dzeroski, S.: First order random forests: learning relational classifiers with complex aggregates. Mach. Learn. **64**(1–3), 149–182 (2006)

20. Vens, C., Ramon, J., Blockeel, H.: Refining aggregate conditions in relational learning. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 383–394. Springer, Heidelberg (2006)
21. Weise, T.: Global optimization algorithms -theory and application, Second Edition (2009)
22. Zhao, H., Sinha, A.P., Bansal, G.: An extended tuning method for cost-sensitive regression and forecasting. Decis. Support Syst. **51**(3), 372–383 (2011)

http://www.springer.com/978-3-319-23707-7

Inductive Logic Programming
24th International Conference, ILP 2014, Nancy, France, September 14-16, 2014, Revised Selected Papers
Davis, J.; Ramon, J. (Eds.)
2015, X, 211 p. 62 illus. in color., Softcover
ISBN: 978-3-319-23707-7