2014 IEEE 26th International Conference on Tools with Artificial Intelligence

# Reframing Continuous Input Attributes

Chowdhury Farhan Ahmed, Nicolas Lachiche, Clément Charnay, Agnès Braud
*ICube Laboratory, University of Strasbourg, France*
{*cfahmed, nicolas.lachiche, charnay, agnes.braud*}*@unistra.fr*

***Abstract*—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 is a crucial example of this where training and testing datasets follow different distributions. However, most of the existing dataset shift solving algorithms need costly retraining operation and are not suitable to use the existing model. In this paper, we propose a new approach called reframing to handle dataset shift. The main objective of reframing is to build a model once and make it workable without retraining. We propose two efficient reframing algorithms to learn the optimal shift parameter values using only a small amount of labelled data available in the deployment. Thus, they can transform the shifted input attributes with the optimal parameter values and use the same existing model in several deployment environments without retraining. We have addressed supervised learning tasks both for classification and regression. Extensive experimental results demonstrate the efficiency and effectiveness of our approach compared to the existing solutions. In particular, we report the existence of dataset shift in two real-life datasets. These real-life unknown shifts can also be accurately modeled by our algorithms.**

***Keywords*-machine learning; classification; regression; dataset shift; adaptation; continuous attributes**

## I. Introduction

It is natural in real-life scenarios that data distributions and decision functions may change from one location to another location. But retraining, i.e., building a model (classifier) in the deployment from the very beginning, will not be feasible in most of the cases because of the lack of availability of labelled data and time. Therefore, it has become a challenging problem in machine learning to build a model once and make it workable in several deployment environments without retraining. The main objective of reframing is to solve this problem.

Several methods [1], [2], [10], [11] have been proposed to use the learnt model in different deployment environments by adjusting the output values. However, an alternative is to transform the input values before using the existing model. Consider a real-life scenario where we are building a model using training data taken from City 1. We have discovered from the training data that once the temperature is greater than $18°C$, most of the people buy an ice-cream. Hence, we build a classifier based on the knowledge of the continuous attribute temperature. Afterwards, we move to City 2 and observe the same event happens when the temperature is greater than $25°C$. Now if we rescale this data by subtracting a numeric value of 7, we can easily use our trained model for City 1 without any modification and be able to predict whether a person in City 2 will buy an ice-cream or not. This is a very simple example of reframing continuous input attributes which we want to propose in this paper. Hence, reframing is needed when the decision function is different in the deployment conditions than in the training conditions and reframing is very effective when a few labelled data are available in the deployment so that it is not feasible to retrain the model or retraining is computationally costly.

In the data mining and machine learning literature, input variable shift is most often known as covariate shift [3]. Accordingly, covariate shift is the situation when the training and test data follow different distributions while the conditional probability distribution remains the same. Besides covariate shift, there exists some other important cases of dataset shifts such as concept drift where data distribution remains the same but the decision function changes [3]. Some research works [4], [5], [6] have been done to tackle covariate shift. However, these approaches need retraining to be applied in a different deployment environment and hence are not suitable for reframing.

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 [7]. But, this approach is computationally expensive and needs a large amount of deployment data to be labelled and can almost be referred to as retraining. Its accuracy may fall below the base model's accuracy (the classifier trained with the training data and used directly in the deployment) if it cannot learn the optimal parameter values with its random genetic programming trials [8]. Moreover, it can only be applied for classification, not for regression.

Can we design a method that learns the optimal parameter values with a very few labelled data in different deployment contexts within a short period of time while ensuring that its performance does not fall below the performance of the base model? It gives us the motivation to propose this approach. The contributions of our approach are described as follows

- Developing a new idea of reframing continuous input attributes.

1082-3409/14 $31.00 © 2014 IEEE
DOI 10.1109/ICTAI.2014.16

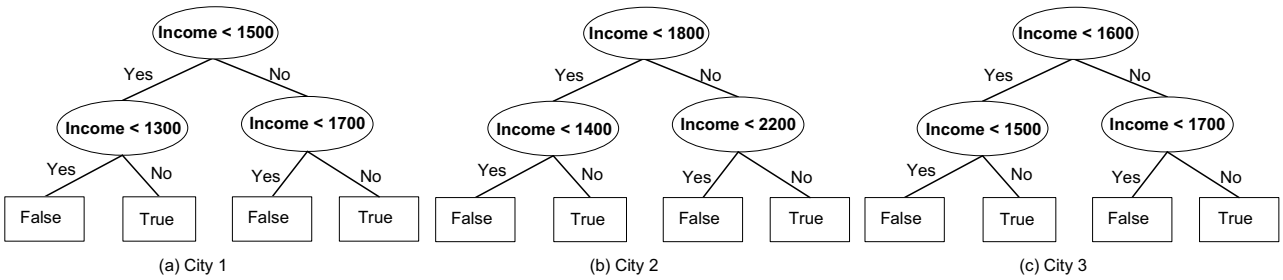


Figure 1. Decision functions for three different cities.

- Designing two efficient algorithms, called Reframing with Stochastic Hill Climbing (RSHC) and Reframing with Randomized Search (RRS), for reframing continuous input attributes.
- Having the capability to tackle different changes in data distributions and decision functions and make the existing model workable in different deployment environments.
- Discovering the optimal shift parameter values even though a very few labelled data are available in the deployment while ensuring a performance that does not fall below the performance of the base model.
- Capable of handling both classification and regression.
- Experimentally showing the efficiency and effectiveness of the proposed algorithms compared to the existing solutions.
- Presenting the existence of dataset shift problem in two real-life datasets in UCI [9] and capability of our algorithms to approximate these unknown real-life shifts accurately.

The remainder of this paper is organized as follows. In Section II, we describe our proposed reframing approach for both classification and regression. In Section III, our experimental results are presented and analyzed. Section IV discusses related work. Finally, conclusions are drawn in Section V.

## II. Our Proposed Approach

In this section, we present our reframing concept to tackle the shift for continuous input attributes and propose two efficient algorithms to learn the optimal parameter values of the shifted attributes for classification and regression. The input attributes of the deployment will be transformed by these parameter values and then applied to the original learnt model with the training database.

Let $M$ be a model built with a base classifier $C$ using some labelled training data $T_{tr}$. We have a few labelled deployment data $T_d$ and several unlabelled test data $T_{tst}$. Suppose $T$ is a transformation function which transforms the values of input attributes of each tuple $X\{x_1, x_2, ....., x_n\} \in T_d$ to $\acute{X}$ by an equation e.g., Equation 1. Our reframing approach evaluate $M(T(X))$, i.e., $M(\acute{X})$ to find the optimal parameter values (values of $\alpha, \beta$ in this case) so that it can transform the deployment data with these values and then use the existing model $M$. Thus, it enables $M$ to be built once and used over several deployment contexts without any modification.

$$\acute{x} = \alpha x + \beta \tag{1}$$

Consider the ice-cream buying example given in Section I. The optimal parameter values of this dataset shift can be $\alpha = 1$ and $\beta = -7$ for the transformation function of Equation 1, i.e., the correct transformed input value $\acute{x}$ can be found by using these parameter values. Thus, every input value of the deployment dataset will be transformed to its corresponding appropriate value of the training dataset and will properly be classified by the existing model.

### A. Reframing Algorithms for Classification

Suppose the decision function becomes more complex as shown in Fig. 1 which represents whether a customer will buy a product $P$ or not. If we build a model $M$ with the data of City 1, and want to use it for the data of City 2 and City 3, then we can also use the transformation function of Equation 1. If we want to classify the data of City 2 using $M$, then we have to use $\alpha = 0.5$ and $\beta = 600$ as optimal parameter values. For example, $1400$ will be converted to $1400 * 0.5 + 600 = 1300$. For City 3, the optimal parameter values are $\alpha = 2$ and $\beta = -1700$. Here the main challenge is to learn these optimal parameter values for this transformation function. We propose the following two efficient algorithms to solve this problem. They are capable of learning these optimal parameter values even though a very few labelled data are available in the deployment.

The first algorithm, called Reframing with Stochastic Hill Climbing (RSHC), (presented in Fig. 2) and the second algorithm, called Reframing with Randomized Search (RRS), use a stochastic hill climbing search and randomized search, respectively to find the optimal parameter values. Indeed, the solution of our problem space is non-convex, and according to [15], [16] these techniques provide the optimal solutions in most of the times of this situation. Therefore, we

have chosen these techniques for discovering our reframing solutions.

The RSHC algorithm (Fig. 2) builds a classification model $M$ using the training labelled data (line 2). The transformation function uses the few available labelled data of deployment to perform the transformation (line 3). The loop described in lines 6 to 17 learns the global optimal parameter values. In the nested loop (lines 7 to 12), it learns a set of local optimal parameter values using the HillClimbing procedure shown in lines 20 to 41. The shouldRestart procedure (invoked in line 13) updates $bestSft$ with the value of current $Sft$ and makes the $count$ value equal to zero, if the current $Sft$ produces better performance than the current $bestSft$. Otherwise, $count$ is incremented by one. It returns True if the $count$ is less than 10, and False (i.e., $bestSft$ remains unchanged for the last 10 times) otherwise. Accordingly, we need to start with another $Sft$ value if the returned value is True. This part is performed in lines 14 to 16. Finally, the learnt optimal parameter values of $bestSft$ are applied to transform the input values of $T_{tst}$ and $M$ is used to predict the class labels (line 18).

The RRS algorithm has a main loop which randomly generates $K$ sets of shift values to find the global optimal parameter values. This algorithm is useful if the search space is very bumpy and has many local optimal values. Then, RSHC algorithm may miss the area of the global optimal parameter values at an extreme case. Hence, the $K$ value of RRS algorithm is normally set larger (around 5 to 10 times) than the $count$ value of RSHC algorithm.

**Input**: A base classifier $C$, e.g., Decision Tree J48; Labelled training data $T_{tr}$; Few labelled data in deployment $T_d$; Unlabelled test data $T_{tst}$; Precision of adjustment $p > 0$.

**Output**: Optimal parameter values to reframe continuous input attributes and the predicted class labels.

```
1  begin
2      Train a model M by using C from T_tr;
3      Let T be a transformation function which shifts the
       attributes of each tuple X{x_1, x_2, ....., x_n} ∈ T_d by
       α_1 x_1 + β_1, ....., α_n x_n + β_n;
4      Sft = (α_1, β_1, ....., α_n, β_n) = (1, 0, ....., 1, 0);
5      bestSft = Sft; Continue = True; p = 0.1; count = 0;
6      while Continue=True do
7          repeat
8              Sft_old = Sft;
9              for i = 1 to 2n do
10                 HillClimbing(Sft, i, p);
11             end
12         until Sft = Sft_old ;
13         Continue = shouldRestart(Sft, count);
14         if Continue = True then
15             Select randomly another Sft value;
16         end
17     end
18     Apply bestSft to T_tst and Output the class labels by
       using M;
19 end
20 Procedure HillClimbing(Sft, i, p)
21 begin
22     Sft+ = Sft− = Sft;
23     Sft+[i] = Sft[i] + p;   Sft−[i] = Sft[i] − p;
24     if Acc(M(T(Sft+))) > Acc(M(T(Sft))) then
25         δ = 1 ;
26     end
27     else if Acc(M(T(Sft−))) > Acc(M(T(Sft))) then
28         δ = −1 ;
29     end
30     else
31         return;
32     end
33     Sft_prev = Sft;
34     repeat
35         Incr = 1;    Sft = Sft_prev;
36         repeat
37             Sft_next = Sft_prev = Sft;
               Sft[i] = Sft[i] + (Incr ∗ p ∗ δ);
38             Incr = Incr ∗ 2;
               Sft_next[i] = Sft[i] + (Incr ∗ p ∗ δ);
39         until Acc(M(T(Sft_next))) < Acc(M(T(Sft))) ;
40     until Incr ≤ 2 ;
41 end
```

Figure 2. The RSHC algorithm.

### *B. Reframing Algorithms for Regression*

Our two reframing algorithms presented in the previous subsection can also be used for regression by changing the base model and evaluation measure. For the sake of clear presentation, we assign the names of reframing algorithms for regression as Reframing with Stochastic Hill Climbing for Regression (RSHCR), and Reframing with Randomized Search for Regression (RRSR). Their base models $M$ have been built with a regression model, e.g., REPTree. For RSHCR, parameter $p$ value is significantly smaller (e.g., 0.001) than the classification problem.

We have used Mean Absolute Error (MAE) measure to compare the output of $M$. For example, in lines 24, 27 and 39 of Fig. 2, MAE measure is used instead of accuracy. In line 24, a positive direction ($\delta = 1$) is chosen when the MAE value of $Sft^+$ is less than the MAE value of current shift value. Similar comparison is done for the negative direction in line 27. In line 39, the loop stops when further shift produces larger MAE value. For RRSR, similar changes should be performed. For both algorithms, $M$ predicts numeric outputs instead of class labels (e.g., line 18 in Fig. 2). We report both MAE and RMSE (root mean square error) values as the output of $M$.

## III. Experimental Results

We have performed several experiments on synthetic and real-life datasets to show the efficiency and effectiveness of our approach. In the first subsection, we show the significance of reframing with respect to retraining and the base model. Subsequently, we compare our approach with the existing GP-RFD method in the second subsection. However, it is shown in [8] that decision tree gives best performance for GP-RFD in most of the cases. Hence, similar to the existing work, we have used decision tree J48 from the Weka

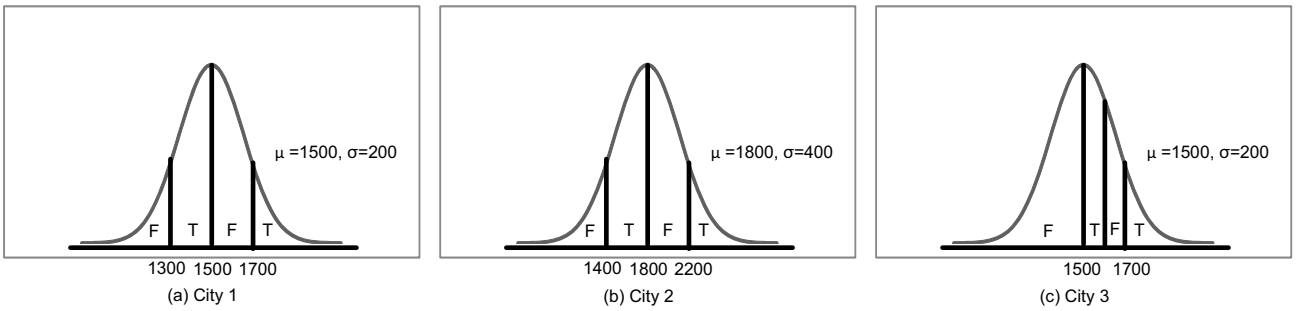


Figure 3. Input data distributions with decision functions for three different cities.

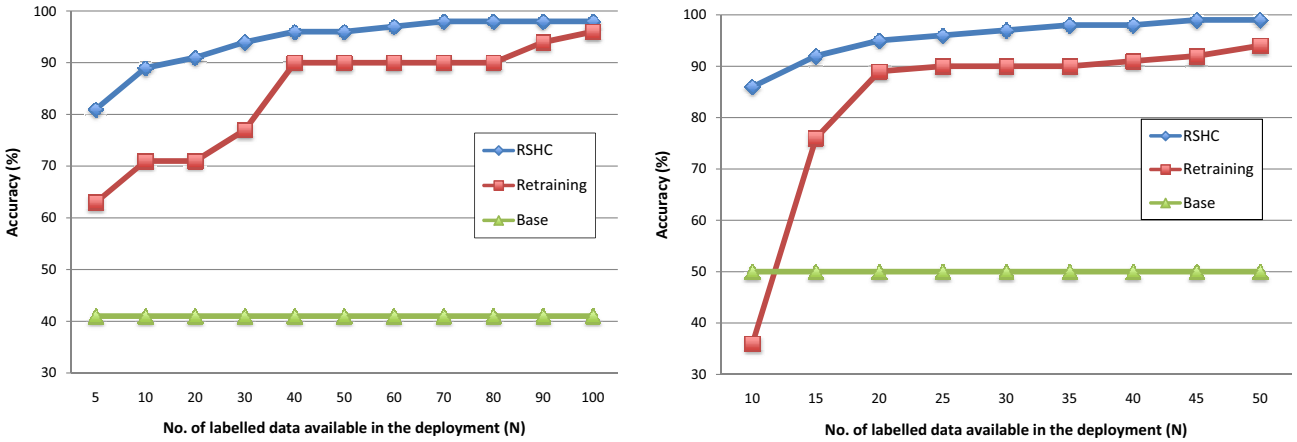


Figure 4. Learning curves for reframing, retraining and base model considering City 1 as training and (a) City 2 (b) City 3 as deployments.

data mining tool [17] as base classifier. Finally, we present the performance of our algorithms in real-life datasets from UCI machine learning repository [9] for regression. We have used REPTree (regression tree) and SMOreg (support vector machine for regression) from Weka [17] as base models to perform regression.

### A. *Performance Analysis*

Three synthetic datasets have been generated using Gaussian distribution to represent income data of the people of three different cities as shown in Fig 3. The decision functions have been taken from the example given in Fig. 1. We have generated 1000 training data for City 1, 1000 testing data for City 2 and City 3 and a few labelled data for learning the distributions of the data of City 2 and City 3. The learning curves for reframing versus retraining with the position of base classifier are shown in Fig. 4.

At first, we have built a model (classifier) with the training data of City 1. Consider we have a few labelled data of City 2 and several unlabelled data for testing. The learning curves for reframing (RSHC), retraining and the base model are shown in Fig 4(a). Moreover, Fig. 3(a) and Fig. 3(b) show the data distributions of these two cities with the corresponding decision functions shown in Fig. 1(a) and Fig. 1(b), respectively. Now, we need to learn the optimal parameter values $\alpha = 0.5$ and $\beta = 600$, according to Equation 1. By using the base classifier, we can only get a very poor accuracy of $41\%$. Using RSHC we can learn very nearby values of optimal $\alpha, \beta$ when only 5 labelled deployment data are available and achieve more than $80\%$ accuracy. Retraining can get only $63\%$ accuracy in this situation. RSHC method achieves around $90\%$ and $95\%$ accuracy when 10 and 30 labelled data are available in the deployment, respectively. On the other hand, retraining needs 40 and almost 100 labelled data to reach $90\%$ and $95\%$ accuracy, respectively.

Subsequently, consider that we have been given a few labelled data of City 3 and several unlabelled data for testing. The learning curves for reframing (RSHC), retraining and the base model are shown in Fig 4(b). Fig. 3(a) and Fig. 3(c) show the data distributions with the corresponding decision functions shown in Fig. 1(a) and Fig. 1(c), respectively. Please note that City 1 and City 3 have same data distributions but different decision functions. We can get only $50\%$ accuracy using the base classifier. Retraining model (only $36\%$ accuracy) is worse than the base model if only

Table I
ACCURACY AND RUNTIME COMPARISON.

| Dataset | Shift | Accuracy (%) | | | | | Runtime (Sec.) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | RSHC | RRS | GP-RFD | Retraining | Base | RSHC | RRS | GP-RFD | Retraining | Base |
| *Heart* $N = 8$ | Low | 77.826 | 77.391 | 65.22 | 47.826 | 66.957 | 1.922 | 1.821 | 20.291 | 1.824 | 1.721 |
| | Medium | 75.652 | 76.957 | 62.173 | 47.826 | 63.043 | 2.017 | 1.935 | 21.727 | 1.843 | 1.726 |
| | High | 74.783 | 73.478 | 58.26 | 47.826 | 60.435 | 2.076 | 1.955 | 21.549 | 1.854 | 1.727 |
| | Extreme | 71.304 | 67.83 | 55.217 | 47.826 | 53.478 | 2.005 | 2.012 | 21.018 | 1.887 | 1.732 |
| | Average | 74.891 | 73.914 | 60.218 | 47.826 | 60.978 | 2.005 | 1.931 | 21.146 | 1.852 | 1.727 |
| *Breast-W* $N = 10$ | Low | 93.048 | 86.108 | 67.562 | 62.243 | 70.427 | 2.052 | 1.908 | 19.837 | 1.833 | 1.779 |
| | Medium | 87.17 | 85.129 | 64.921 | 62.243 | 57.515 | 1.993 | 1.879 | 19.197 | 1.813 | 1.785 |
| | High | 84.483 | 83.553 | 56.415 | 62.243 | 49.41 | 1.922 | 1.965 | 20.216 | 1.855 | 1.767 |
| | Extreme | 81.969 | 78.505 | 50.657 | 62.243 | 37.757 | 1.968 | 1.931 | 20.046 | 1.875 | 1.813 |
| | Average | 86.668 | 83.324 | 59.889 | 62.243 | 53.777 | 1.984 | 1.921 | 19.824 | 1.844 | 1.786 |

10 labelled data are available. On the other hand, RSHC gives $86\%$ accuracy using this 10 labelled data and shows the power and significance of reframing. To reach around $95\%$ accuracy, RSHC and retraining need 20 and 50 labelled data in the deployment, respectively. The RRS algorithm has similar learning curves like RSHC and hence are not shown here. The results presented in Fig 4 demonstrate that reframing is very useful and effective even though a very small amount of labelled data are available in the deployment. Furthermore, reframing is remarkably better than retraining and it can handle different kinds of dataset shift and changes in decision function.

### *B. Comparison with the Existing Algorithm*

We have compared our approach with the GP-RFD algorithm with respect to accuracy and runtime using benchmarks for dataset shift[1]. This dataset repository contains some real-life binary classification datasets in original (the original datasets are also available in UCI Machine Learning Repository [9]) and shifted format while shifts are generated randomly in one arbitrary attribute by using Equation 1. This type of shift is referred to as domain shift, and four variations of shifts called low, medium, high and extreme are generated. The performance of GP-RFD algorithm is shown in these datasets in [8]. But, they have used the full testing datasets to learn the best transformation.

We have taken two datasets, *Heart* (contains information about heart disease with 14 attributes and 270 instances) and *Breast-W* (contains Breast Cancer Wisconsin data with 10 attributes and 683 instances), from this benchmark. As mentioned above, the limitations of the benchmark are that it provides shift only in one attribute and this attribute is taken randomly. Hence, if the attribute is not an influential attribute then shifting operation on that attribute will not influence the base model even though the shift value is extreme. Moreover, in a real-life scenario, multiple attributes can be shifted. To overcome these problems, we have generated new shifted datasets by shifting three influential attributes. We have created a decision-tree with the original dataset to observe which attributes are influential on that dataset. According to the performance of the base model, we have indicated the shifts as low, medium, high and extreme. We have modified the source-code available on the benchmark website in order to generate these shifted datasets. Like datasets provided in the benchmark, we have generated 5-folds of the original dataset, used 4-folds as training and one fold as testing (after shifting). Therefore, our accuracy provides the average of the accuracies obtained in these 5-folds and the runtime reports the total time for obtaining these accuracies.

At first, we compare the accuracies of our algorithms with the GP-RFD algorithm in Table I (left). Accuracies of retraining and base models are also reported for better understanding. Number of labelled data available in the deployment is denoted as $N$. Accuracies of the base classifier demonstrate the severeness of a particular shift. The learning mechanism of GP-RFD algorithm is based on random crossover and mutation procedures of genetic algorithm and needs a large amount of labelled data in the deployment. It cannot ensure that its performance would not fall below the base model. Here, as a small amount of deployment data are available, its performance is not satisfactory. In the *Heart* dataset, it performs better than retraining but worse than the base model in the cases of low, medium and high shifts. It is only better than the base model in extreme shift. As a consequence, the overall accuracy of GP-RFD is worse than the base model. In the *Breast-W* dataset, even though its average accuracy is better than the base model, retraining performs better than it.

It is noticeable that reframing algorithms have learnt the optimal parameter values in all the cases of these datasets and achieved better accuracy than any other methods. The performance of RSHC is better than or equal to RRS in almost all the cases as hill climbing can tune accurately to the optimal values described in Section II. RRS achieves slightly better accuracy than RSHC in the medium shift of *Heart* dataset. It indicates that at a rare occasion RSHC may not reach the region of optimal parameter values while RRS may reach there randomly if the $K$ value of RRS is large

[1] http://sci2s.ugr.es/dataset-shift/

Table II
PERFORMANCE OF BASE MODELS (DIFFERENT ORIGINS) OF THE *Auto MPG* DATASET.

| Measure | Origin-1 $\|D\| = 249$ | Origin-2 $\|D\| = 70$ | Origin-3 $\|D\| = 79$ |
|---|---|---|---|
| MAE | 2.434 | 4.36 | 3.575 |
| RMSE | 3.386 | 5.619 | 4.605 |

Table III
PERFORMANCE OF REFRAMING ALGORITHMS ON THE *Auto MPG* DATASET.

| Deployment $N = 8$ | Measure | RSHCR | RRSR | Re-training | Base (Org-1) |
|---|---|---|---|---|---|
| Origin-2 | MAE | 4.385 | 4.453 | 6.606 | 5.282 |
| | RMSE | 5.839 | 5.941 | 8.012 | 6.397 |
| Origin-3 | MAE | 3.945 | 3.983 | 6.052 | 4.592 |
| | RMSE | 4.815 | 4.852 | 7.328 | 5.552 |

enough compared to the restart controlling $count$ value of RSHC. Here, we have used 10 as the restart controlling $count$ of RSHC and 100 as the value of $K$ for RRS. 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 a few labelled data available in the deployment.

Afterwards, we compare the runtime of our algorithms with GP-RFD in Table I (right). Runtime of retraining and base model are also reported as usual. The base model obviously takes smallest amount of time as it does not perform any shift calculation in the deployment and directly apply the trained model. With a small amount of labelled data, reframing and retraining take slightly more time compared to the base model. However, GP-RFD takes significantly larger runtime compared to all the other methods due to the complex calculations of genetic programming.

These experimental results demonstrate that our reframing algorithms clearly outperform the existing GP-RFD algorithm with respect to both accuracy and runtime. Their performance is never worse than the performance of the base classifier which cannot be maintained by the existing algorithm.

### C. Adapting Real-Life Dataset Shift for Regression

We first show the existence of dataset shift problem in two real-life datasets, *Auto MPG* and *Bike Sharing* from UCI [9]. Afterwards, we demonstrate that our algorithms are able to handle these real-life dataset shifts.

The *Auto MPG* dataset [9] concerns city-cycle fuel consumption in miles per gallon (mpg) to be predicted (attribute 1). To build a regression model for this prediction, we have considered four continuous attributes displacement, horsepower, weight and acceleration. This dataset contains data from three different origins (values 1, 2 and 3 of the Origin attribute). REPTree is used as the base regression model here. Table II shows the performance of the base models of different origins using 10-fold cross validation. We have used the MAE (mean absolute error) and RMSE (root mean squared error) measures to represent the performance in regression. To show a real-life dataset shift, we deploy the regression model trained at Origin-1 into Origin-2 and Origin-3. Please notice that the base model of Origin-1 has a MAE value of 2.434 in Origin-1 (Table II), but MAE values of 5.282 and 4.592 in Origin-2 and Origin-3, respectively (Table III). While the base models of Origin-2 and Origin-3 have MAE values of 4.36 and 3.575 respectively (Table II). It clearly shows that dataset shift exists among these origins.

Now, we run our reframing algorithms on this dataset considering Origin-1 as training, Origin-2 and Origin-3 as deployments with only eight available labelled data in each deployment. Results (average of 10 runs) are reported in Table III. It is noticeable that the performance of our algorithms are very close to the base model of that particular deployment region. For example, the base model of Origin-2 has MAE and RMSE values of 4.36 and 5.619, respectively; while RSHCR algorithm has MAE and RMSE values of 4.385 and 5.839, respectively in this origin using only eight labelled data. Moreover, reframing algorithms are significantly better than retraining and the base model of Origin-1. As the GP-RFD method is not implemented for regression, we did not compare with it.

Table IV
PERFORMANCE OF BASE MODELS (DIFFERENT SEASONS) OF THE *Bike Sharing* DATASET.

| Model | Spring $\|D\| = 181$ | Summer $\|D\| = 184$ | Fall $\|D\| = 188$ | Winter $\|D\| = 178$ |
|---|---|---|---|---|
| REPTree | 893.799 | 1189.921 | 1213.936 | 1253.761 |
| | 1110.777 | 1449.981 | 1408.248 | 1506.756 |
| SMOreg | 820.54 | 1110.858 | 1237.773 | 1127.347 |
| | 1050.145 | 1328.08 | 1451.099 | 1342.523 |

The *Bike Sharing* Dataset [18], [9] contains usage log of a bike sharing system called Capital Bike Sharing (CBS) at Washington, D.C., USA. We have built regression models using four continuous attributes temp (temperature), atemp (feels like temperature), hum (humidity) and wind-speed to predict the numeric output cnt (count of total rental bikes in a day). This dataset contains data of different seasons (1:Spring, 2:Summer, 3:Fall, 4:Winter in the Season attribute). Both REPTree and SMOreg have been used as the base regression models here. Table IV shows the performance of the base models of different seasons using 10-fold cross validation. The first and second rows of each regression model represents MAE and RMSE values, respectively. Base model of the Spring season is deployed over the Summer, Fall and Winter seasons. Table V reports the performance

Table V
PERFORMANCE OF REFRAMING ALGORITHMS ON THE *Bike Sharing* DATASET.

| Deployment $N = 18$ | Measure | REPTree | | | | SMOreg | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | RSHCR | RRSR | Retraining | Base(Spr) | RSHCR | RRSR | Retraining | Base(Spr) |
| Summer | MAE | 1197.475 | 1199.118 | 1874.759 | 1213.755 | 1162.099 | 1184.354 | 1670.034 | 1227.468 |
| | RMSE | 1480.602 | 1484.346 | 2227.061 | 1516.177 | 1394.836 | 1420.633 | 2041.069 | 1475.085 |
| Fall | MAE | 1259.537 | 1259.537 | 1535.497 | 1275.993 | 1479.381 | 1486.776 | 1550.93 | 1885.664 |
| | RMSE | 1526.26 | 1526.26 | 1858.321 | 1563.39 | 1771.22 | 1781.765 | 1852.513 | 2299.89 |
| Winter | MAE | 1344.451 | 1356.276 | 1809.651 | 1519.932 | 1235.681 | 1279.772 | 1649.353 | 1322.544 |
| | RMSE | 1655.724 | 1669.64 | 2164.209 | 1839.349 | 1508.353 | 1556.549 | 1986.799 | 1608.638 |

(average of 10 runs) of the base model (Spring), reframing algorithms ($N = 18$) with this base model and retraining operation. It can clearly be observed from Tables IV and V that dataset shift occurs among these seasons. Reframing algorithms also perform very well in this dataset by achieving a closer performance to the corresponding season's base model and much better performance compared to the base model of Spring and retraining.

Experimental results in these two real-life datasets demonstrate that our approach is quite capable of learning the 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.

## IV. RELATED WORK

Some score-based methods have been proposed to handle classification problem in different deployment scenarios. For example, the binary classification algorithm of [1] generates scores of being positive vs. negative and define one threshold to divide these scores to predict the boundary between two classes. According to the deployment environment, typically a matrix of misclassification costs and the prior probability of the classes, this threshold can be tuned and the model is adapted for class prediction. Research has also been done to handle this problem for multi-class classification [1], [2].

It is also possible to adapt regression outputs when the cost function changes. A tuning method has been proposed to adjust the average misprediction cost of a cost-sensitive regression model [10] by assuming the cost function to be convex. This method adds/subtracts a constant shift to all the predicted values of the original regression model. A model of ROC analysis for regression is developed by [11]. Two axes have been marked with over and under estimations of predicted values. The ROC curve is drawn by considering several shift values with the original predictions. Adaptation relies on adding a shift to the original regression model. This shift is chosen according to the deployment conditions, typically the loss function and the prior distribution of instances in the deployment condition.

However, the above mentioned methods perform the adjustments on the model output. Research has been done to tackle covariate shift [3] where shifts in input variables are considered over different deployment contexts. 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 [4]. 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), derives a discriminative model for learning under different training and test distributions [5]. The contribution of each training instance to the optimization problem ideally needs to be weighted with its test-to-training density ratio. In order to cope with covariate shift, Kernel Mean Matching (KMM) approach [6] 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. However, these approaches need retraining to be applied in a different deployment environment and hence are not suitable for reframing.

Some other related research areas such as theory revision [12], transfer learning [13] and domain adaptation [14] proposed solutions to the problem where training and test data follow different distributions. But, these techniques cannot serve the purpose of reframing which aims at building a model once and make it (i.e., the same model) workable in several deployment environments without retraining. Theory revision [12] assumes an explicit model (theory) and makes transformations on the model. In our approach the model does not change. We do not need an explicit model and any learning paradigm would work, e.g., SVM. Transfer learning [13] 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. Domain adaptation assumes that the input data follow some statistical distribution as most approaches of covariate shift or restricts to statistical learning models [14]. Our approach works with any learning paradigm and does not assume any distribution of the input data. Though it assumes that the change between the training and deployment data can be approximated by a linear model. Experiments on two real-life datasets showed

that such an approximation is effective. However, if the shift is more complex, we plan to learn piecewise linear adaptation to deal with non-uniform/non-linear shift. To perform this, we just need to modify our transformation function within the proposed framework. Instead of having a linear shift in the full range of an attribute, it may have different linear shifts in different intervals.

As discussed in Section I, an input transformation based approach, called GP-RFD, has been proposed recently to handle general dataset shift [7], [8]. 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 the 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 to determine the best transformation. This approach suffers from several problems including huge computation time, requirement of many labelled data in different deployment areas, having lower accuracy compared to the base model in some situations and inability to handle regression problem. Our proposed approach has solved the limitations of the existing methods.

## V. Conclusions

In this paper, we have proposed a new approach of reframing the values of continuous input attributes. Moreover, we have designed two efficient algorithms to learn the optimal parameter values for the shifted continuous input attributes. Our approach has the capability of tackling different changes in data distributions and decision functions, and making the existing model workable in different deployment environments. Even though a very few labelled data are available in the deployment, it can discover the desired optimal parameter values to handle the actual dataset shift. With extensive experimental results using benchmarks for dataset shift, we have shown that our approach is remarkably better than the existing input transformation based approach with respect to both accuracy and runtime. In the worst case, our approach is equivalent to the base model which property cannot be achieved by the existing method. 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 two real-life datasets by considering one place (resp., season) as training context and other places (resp., 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] N. Lachiche and P. A. Flach, "Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves," in *ICML*, 2003, pp. 416–423.

[2] C. Charnay, N. Lachiche, and A. Braud, "Pairwise optimization of bayesian classifiers for multi-class cost-sensitive learning," in *IEEE ICTAI*, 2013, pp. 499–505.

[3] J. G. Moreno-Torres, T. Raeder, R. Alaíz-Rodríguez, N. V. Chawla, and F. Herrera, "A unifying view on dataset shift in classification," *Pattern Recognition*, vol. 45, no. 1, pp. 521–530, 2012.

[4] M. Sugiyama, M. Krauledat, and K.-R. Müller, "Covariate shift adaptation by importance weighted cross validation," *Journal of Machine Learning Research*, vol. 8, pp. 985–1005, 2007.

[5] S. Bickel, M. Brückner, and T. Scheffer, "Discriminative learning under covariate shift," *Journal of Machine Learning Research*, vol. 10, pp. 2137–2155, 2009.

[6] A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, and B. Schölkopf, "Covariate shift by kernel mean matching," in *Dataset Shift in Machine Learning*. The MIT Press, 2009, pp. 131–160.

[7] J. G. Moreno-Torres, X. Llorà, D. E. Goldberg, and R. Bhargava, "Repairing fractures between data using genetic programming-based feature extraction: A case study in cancer diagnosis," *Information Sciences*, vol. 222, pp. 805–823, 2013.

[8] J. G. Moreno-Torres, "Dataset shift in classification: Terminology, benchmarks and methods," PhD Thesis, Available Online: http://sci2s.ugr.es/publications/ficheros/Thesis_JGMorenoTorres_2013.pdf, 2013.

[9] K. Bache and M. Lichman, "UCI machine learning repository," [http://archive.ics.uci.edu/ml/], 2013.

[10] H. Zhao, A. P. Sinha, and G. Bansal, "An extended tuning method for cost-sensitive regression and forecasting," *Decision Support Systems*, vol. 51, no. 3, pp. 372–383, 2011.

[11] J. Hernández-Orallo, "ROC curves for regression," *Pattern Recognition*, vol. 46, no. 12, pp. 3395–3411, 2013.

[12] D. Ourston and R. J. Mooney, "Theory refinement combining analytical and empirical methods," *Artif. Intell.*, vol. 66, no. 2, pp. 273–309, 1994.

[13] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2010.

[14] H. Daumé III and D. Marcu, "Domain adaptation for statistical classifiers," *J. Artif. Intell. Res.* , vol. 26, pp. 101–126, 2006.

[15] T. Weise, "Global optimization algorithms -theory and application-," Second Edition, Available Online: http://www.it-weise.de/projects/book.pdf, 2009.

[16] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.

[17] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, 2009.

[18] H. Fanaee-T and J. Gama, "Event labeling combining ensemble detectors and background knowledge," *Progress in Artificial Intelligence*, vol. 2, no. 2-3, pp. 113–127, 2014.