Binary Transformation Method for Multi-Label Stream Classification

No Author Given

No Institute Given

Abstract. Data streams produce extensive data with high throughput from various domains and require copious amounts of computational resources and energy. Many data streams are generated as multi-labeled and classifying this data is computationally demanding. Some of the most well-known methods for Multi-Label Stream Classification are Problem Transformation schemes; however, previous work on this area does not satisfy the efficiency demands of multi-label data streams. In this study, we propose a novel Problem Transformation method for Multi-Label Stream Classification called Binary Transformation, which utilizes regression algorithms by remodeling the data stream. We compare our method against three of the leading problem transformation methods using eight datasets. Our results show that Binary Transformation achieves statistically similar effectiveness and provides much higher level of efficiency.

Keywords: Data stream classification · Multi-label · Problem transformation

1 Introduction

Data streams carry information that is of extensive amounts and arrive at extremely high speeds, such as audio and video samples, emails, online articles, and social media updates [11,15,22]. Data stream classification, is the affiliation of data instances in data streams with their associated labels. Since data streams provide continuous and potentially infinite data, stream classification algorithms must be able to keep up with the speed and storage requirements of the data [11].

A great majority of the data stream classifiers focus on the problem of single-label (multiclass) classification. As far as single-label classification is concerned, each data instance is associated with only one class label from the collection of individual labels [27]. Multi-Label Stream Classification (MLSC) differs from single-label classification in the sense that it handles the problem of one sample being associated with multiple labels at the same time [22]. It is reported to be used in various fields such as text categorization, image and video classification, and biology [26]. Common approaches for MLSC are Problem Transformation (PT), which is converting the problem into a single-label classification problem, Algorithm Adaptation, which is modifying the classification algorithm to handle

multiple labels, or Ensemble, which is using a combination of MLSC techniques together.

Problem Transformation (PT) techniques include Binary Relevance (BR), Classifier Chains (CC), and Label Powerset (LP) [26,34,16,30]. These are some of the most prominent methods for MLSC and are widely used [24]. BR and CC are based on transforming the data into individual labels, whereas LP deals with converting combinations of labels into new slack labels. All of these methods have the benefit of being conceptually simple since they utilize already existing classification algorithms that handle disjoint labels. However, as the size of the label set increases, BR and CC scale linearly, and LP scales exponentially in the solution domain. This not only declares that the current PT techniques inefficient in terms of time and space, but also shows that they require generous amounts of energy and computational resources [23]. The properties make them unsuitable for MLSC in real-life problems.

In this study, we propose a novel PT method for multi-label classification that changes the label vectors into continuous values, which are then used for classification through regression. Our method provides a highly scalable and efficient procedure that is more suitable to process data streams with high throughput.

In this work, our main contributions are the following. We,

- Propose a novel and efficient, problem transformation-based multi-label classification method.
- Perform experimental and statistical evaluation of our method by comparing it against three of the most prevalent PT methods using eight datasets with varying domains and properties.
- Compare the efficiency of our method against the baseline methodologies in terms of execution time through experimental and statistical testing.

In the remainder of this paper, we first formally define MLSC and introduce the previous work in the literature. Then, we formally describe our proposed method and explain our evaluation and experimental setup. Finally, we present our effectiveness and efficiency results alongside discussions about them.

2 Related Works

2.1 Multi-label Stream Classification

The following symbols and notation will be used for the rest of the paper:

- N: Number of data instances in a data stream
- M: Number of features in a data item
- n: Number of class labels in a data stream
- $\mathcal{X}$: Input feature space where $\mathcal{X} = \mathbb{R}$
- λ: A single class label. $\lambda \in \{0, 1\}$
- $\mathcal{L}$: Set of all possible labels where $\mathcal{L} = \{\lambda_1, \lambda_2, ..., \lambda_n\}$
- $\mathcal{C}$: Set of all possible values of a transformed label where $\mathcal{C} \in [0, 2^n - 1]$
- x: Features of a data instance. $x =< x_1, x_2, ..., x_M >\in \mathcal{X}$

- y: Ground truth labels of a data instance. $y =< y_1, y_2, ..., y_n >= \{0, 1\}^n$
- d_t: Data instance that arrives at time t where $d_t = (x^t, y^t)$
- $\hat{y}^t$: Predicted labels of a data instance d_t. $\hat{y} =< \hat{y}_1, \hat{y}_2, ..., \hat{y}_n >= \{0, 1\}^n$
- $\hat{C}_t$: Predicted regression output of a data instance d_t where $\hat{C}_t \in \mathbb{R}$
- C_t: Transformed value of a ground truth label y^t where $C_t \in \mathbb{R}$
- $\mathcal{D}$: Data stream where $\mathcal{D} = d_0, d_1, ..., d_N$
- Y: The ground truth labels of a data stream $\mathcal{D}$ where $Y = y^1, y^2, ..., y^N$

In traditional single label approaches, each data instance in a data stream $\mathcal{D}$ is affiliated with only one class λ from a set of all possible labels $\mathcal{L} = \{\lambda_1, \lambda_2, ..., \lambda_n\}$, where $n > 1$. In binary classification, $n = 2$, and in multiclass classification, $n > 2$ [28]. This means that, in a multiclass environment, given that $d_t = (x^t, y^t)$:

$$\sum_{i=1}^{n} y_i^t = 1 \tag{1}$$

In Multi-Label Stream Classification, every data instance d_t is associated with multiple labels at the same time, meaning that:

$$\sum_{i=1}^{n} y_i^t \geqslant 1 \tag{2}$$

A multi-label classifier is concerned with the prediction of a set of relevant labels to associate with a new data instance [7].

2.2 Problem Transformation Methods

Problem Transformation [3,25,26,34] methods utilize the presence of numerous established single-label learning methods by converting the data from multiple labels into single labels and therefore transforming the multi-label problem into separate binary classification problems. For transformation, PT techniques can use a single label, a pair of labels, or multiple labels at a time, and are categorized accordingly. The most common transformation methods include Binary Relevance [26,32] for single-label transformation, Classifier Chains [18,26,30,34] and Label Powerset [10,26,30,34,27] for multiple label transformation.

Binary Relevance [26,32] is one of the most prevalent Problem Transformation methods. In a multi-label setting, BR creates respective classifiers for each label to deal with each class independently. The results of all classifiers are then taken into account for the classification. This label by label approach makes BR fairly simple to use, and the complexity of the classifier model scales linearly with the number of the labels. Furthermore, BR is not limited to a single learning method, since any technique for dealing with individual labels can be preferred. One weakness of BR is that the interrelationship between the labels is overlooked since it handles each label one by one.

Classifier Chains [18,26,30,34], which is based on BR, constructs n binary classifiers following the sequence of the labels. Each classifier takes into account the predictions for the previous labels, thus creating a chain of classifiers. The

labels can be permuted using a variety of techniques. CC preserves the advantages of BR and overcomes the problem of losing the correlations between labels. However, the predictions of CC are highly dependent on the arrangement of labels, therefore accuracy is easily affected by differences between label sequences. Furthermore, the dependence on the previous classifiers prevents the process from being parallelized, thus increasing the time complexity.

Label Powerset [10,26,30,34,27] transforms each combination of labels into a single class, converting the problem into a multiclass problem. LP is a simple method that allows the usage of any multiclass learning technique for classification. Unlike BR, LP takes the interrelationship between labels into account. The drawback of LP is that the number of slack classes grows exponentially with the number of label combinations, which causes the computation to be costly when dealing with a large number of them. The fact that each combination is converted into a different class can cause some classes to appear significantly less often than others, which may hinder accuracy. Finally, LP cannot predict label combinations it has not seen before.

3 Binary Transformation Method

In a streaming environment, the high throughput of incoming data requires the classification algorithms to be efficient. However, many of the previously proposed methods fall short in terms of execution time. To solve this problem, we propose the Binary Transformation method, an efficient PT technique for classification that utilizes regression algorithms.

Given a multi-label data stream $\mathcal{D}$, each data instance d_t has labels $y^t =< y_1^t, y_2^t, ..., y_n^t >$. The singular label y_i^t $(1 \leq i \leq n)$ represents whether this instance d_t belongs to a specific class and a classification algorithm predicts the entirety of the output vector $\hat{y}_t$. However, y^t vector can also be seen a as a binary encoding of a continuous integer.

During the training process, a BT classifier transforms each ground truth label y^t into an integer. This transformation is done by:

$$C_t = \sum_{i=1}^{n} y_i^t * 2^{i-1} \tag{3}$$

Through continuous transformation of each of the training labels, we reconstruct our data stream into Equation 4, which we use to train a base regression model R to be able to predict the transformed integer as a continuous value.

$$\mathcal{D}' = d'_1, ..., d'_N \quad , \quad d'_t = (x^t, C_t) \tag{4}$$

Following the training of the regression model, given a data instance d_t and its features x^t, a BT classifier predicts by first receiving the regression output:

$$\hat{C}_t = \lfloor R(x^t) \rceil \tag{5}$$

Then, it solves Equation 6 for each predicted label $\hat{y}_i^t$ where i is the i-th label of a prediction $\hat{y}^t$:

$$\hat{C}_t = \hat{y}_1^t \cdot 2^0 + \hat{y}_2^t \cdot 2^1 + ... + \hat{y}_n^t \cdot 2^{n-1} \tag{6}$$

Finally, it obtains the prediction vector:

$$\hat{y}^t =< \hat{y}_1^t, \hat{y}_2^t, ..., \hat{y}_n^t > \tag{7}$$

A well trained regression algorithm (optimized for minimal loss), can make close enough predictions, which results in accurate label vectors after transformation that allows the method to be effective. Furthermore, BT preserves the information about common labels e.g, $< 1, 0, 1 >$ and $< 1, 1, 0 >$ are seen as two independent classes by LP even though they share a common label. However, our approach utilizes this dependency since it directly transforms the label vector into an integer, which produces values in close proximity (5 and 6 in the previous example). This allows the regression model to associate the data features to this value range which increases per-label effectiveness.

Furthermore, this approach allows BT to be scalable with an increasing number of labels since only a single regression algorithm is employed, thus enabling fast execution. Furthermore, since we only predict one continuous value, BT scales well with the increasing number of label combinations unlike LP. The combinations of these properties makes BT scale in O(1) time in terms of execution time, with regards to n. Figure 1 exhibits a simple illustration of BT.

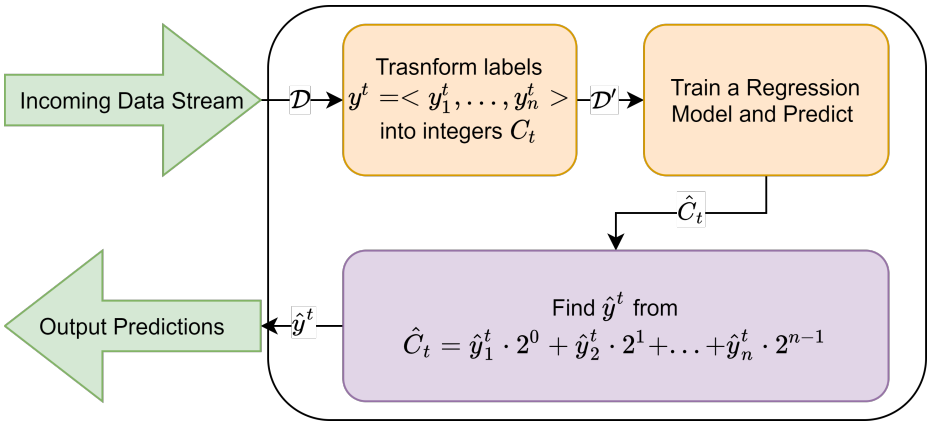


Fig. 1: General workflow of the Binary Transformation method.

4 Experimental Setup and Evaluation

We use eight real-world datasets with varying properties and from different problem domains which are in the MEKA [19] format[1]. Their properties are displayed

[1] The datasets can be accessed from: `http://www.uco.es/kdis/mllresources/`

in Table 1. The chosen datasets are transformed into data streams in which the samples arrive in the order they originally have.

Table 1: The table of multi-label datasets used in the experiments. LC($\mathcal{D}$) is the label cardinality (Average number of true labels for the samples in a data stream $\mathcal{D}$) and LD($\mathcal{D}$) is the label density (Average label cardinality per label).

Dataset	**Domain**	N	M	n	**LC**($\mathcal{D}$)	**LD**($\mathcal{D}$)
20NG [12]	Text	19,300	1,006	20	1.029	0.051
EukaryotePseAAC [31]	Biology	7,766	440	22	1.146	0.052
Imdb [17]	Text	120,900	1,001	28	2.000	0.071
Mediamill [21]	Video	43,910	120	101	4.376	0.043
Reuters-K500 [29]	Text	6,000	500	103	1.462	0.014
Scene [1]	Image	2,407	294	6	1.074	0.179
Slashdot [17]	Text	3,782	1,079	22	1.181	0.054
Yelp [20]	Text	10,810	671	5	1.638	0.328

For our evaluation, we use subset accuracy, Hamming score, micro-averaged F1 score, and macro-averaged F1 score to measure the effectiveness of the methods [33]. The former two metrics allow us to gauge the complete and partial correctness of the predictions, and the latter two allow us to assess the sample-based and class-based effectiveness. Additionally, we evaluate the efficiency of our algorithm by measuring the execution time (in seconds). The formal definitions of the metrics are presented below. In the equations, TP_i, FP_i, FN_i represent the true positive, false positive, and false negative predictions where i denotes the *i-th* sample in the stream. Moreover, Δ is the symmetric difference between the prediction and the ground truth.

$$SubsetAccuracy = \frac{1}{N}\sum_{i=1}^{N}[\![\hat{y}^i = y^i]\!] \tag{8}$$

$$HammingScore = 1 - \frac{1}{N}\sum_{i=1}^{N}\frac{1}{n}|\hat{y}^i \Delta y^i| \tag{9}$$

$$F1_{micro} = F1\left(\sum_{i=1}^{n}TP_i, \sum_{i=1}^{n}FP_i, \sum_{i=1}^{n}FN_i\right)^{2} \tag{10}$$

$$F1_{macro} = \frac{1}{n}\sum_{i=1}^{n}F1(TP_i, FP_i, FN_i) \tag{11}$$

The experiments are performed using the River framework[3] [13]. We compare our algorithm against BR, CC and LP using a Hoeffding Tree (HT) [9] as

[2] $F1(TP, FP, FN) = \frac{2 \cdot TP}{2 \cdot TP + FN + FP}$

[3] The source code of BT will be available after the review process.

a base classifier through the interleaved-test-then-train evaluation (prequential evaluation) [2]. Likewise, we use an HT regressor as the base model for our algorithm. The HT classifiers employed are tested using default parameters, where they use adaptive Naive Bayes classifiers on the leaf nodes. The HT regressor for our algorithm similarly utilizes linear regression nodes on the leaves.

Furthermore, although many classifiers are adapted to multi-label learning, we chose to compare BT against other PT techniques to focus the scope of this work since it is also a transformation-based approach.

5 Results and Discussions

In the following sections, we first present and discuss our experiments on effectiveness. Then, we examine the efficiency of the tested methods. Finally, we perform experiments on BT using different base regression algorithms. In all of the tables presented in this section, the best results are highlighted in bold.

5.1 Effectiveness Analysis

Table 2 displays the effectiveness results of the tested algorithms. The experimental results show that, on average, BT achieves the best results on F1 scores and remains close behind on the other two metrics. From the experiments, we can deduce that the main detriment of BT, seems to be large number of labels in terms of effectiveness. This is due to the rapid increase in the size of the solution set. However, from the macro-averaged F1 scores, we can see that BT performs well based on per-label effectiveness since macro-averaged F1 score is calculated using the individual F1 scores of the classes. This indicates that, although BT performs comparatively worse on streams with large number of labels per-sample, it still shows adequate effectiveness on partial predictions label-wise.

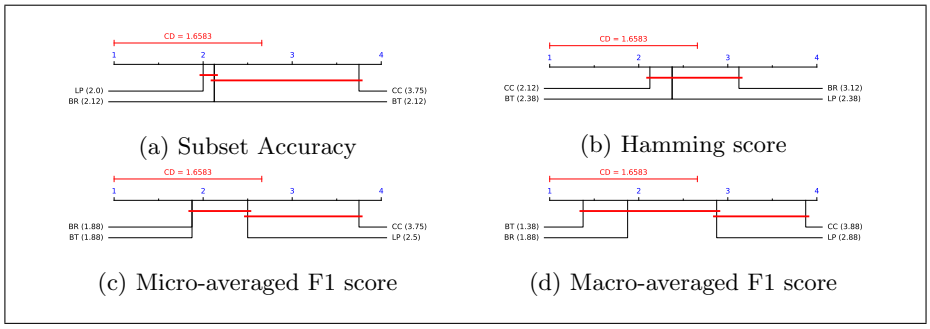
To further analyze the effectiveness of BT, we perform "Friedman test with Nemenyi post-hoc analysis" [14] where we investigate the statistical significance of our experimental results. Figure 2 illustrates our two-tailed Nemenyi statistical significance tests with critical distance $CD = 1.6583$. CD is calculated by Equation 12 where the $q_{\alpha,k}$ value is obtained from the "Critical Values Table" from Janez Demšar's work [6], while $\alpha = 0.05$, $k = 4$ (Number of tested methods), and $K = 8$ (Number of datasets).

$$CD = q_{\alpha,k}\sqrt{\frac{k(k+1)}{6K}} \tag{12}$$

Our statistical analysis shows that overall, BT demonstrates statistically insignificant effectiveness compared to the baselines for all metrics, except CC on F1 scores, which means that our method performs statistically similarly to the compared algorithms.

Table 2: Experimental results on all effectiveness metrics (higher is better). For some experiments, the estimated time for completion was infeasible which is denoted by TLC.

Exact Match (EM)	BT	BR	CC	PS	**Hamming Score**	BT	BR	CC	PS
20NG	0.187	0.248	0.040	**0.357**	20NG	**0.917**	**0.955**	0.912	0.928
EukaryotePseAAC	**0.776**	0.376	0.072	0.202	EukaryotePseAAC	**0.988**	0.953	0.911	0.925
Imdb	**0.054**	0.007	0.000	TLC	Imdb	0.889	0.925	**0.929**	TLC
Mediamill	**0.004**	**0.059**	0.054	0.054	Mediamill	**0.346**	**0.967**	0.958	0.958
Reuters-K500	0.018	0.062	0.000	**0.064**	Reuters-K500	0.820	0.977	**0.986**	**0.986**
Scene	**0.870**	0.301	0.153	0.562	Scene	**0.976**	0.841	0.713	0.861
Slashdot	**0.149**	0.021	0.000	0.141	Slashdot	**0.912**	**0.947**	0.946	0.915
Yelp	**0.593**	0.275	0.018	0.229	Yelp	**0.889**	0.753	0.502	0.700
Average Rank	2.125	2.125	3.750	**2.000**	Average Rank	2.375	3.125	**2.150**	2.375
Micro-averaged F1 Score					**Macro-averaged F1 Score**				
20NG	**0.192**	**0.428**	0.049	0.343	20NG	**0.192**	**0.418**	0.010	0.355
EukaryotePseAAC	**0.886**	0.513	0.083	0.287	EukaryotePseAAC	**0.795**	0.221	0.028	0.106
Imdb	**0.225**	0.041	0.000	TLC	Imdb	**0.129**	0.023	0.000	TLC
Mediamill	**0.093**	**0.507**	0.429	0.429	Mediamill	**0.051**	**0.079**	0.019	0.034
Reuters-K500	**0.025**	**0.059**	0.0004	0.045	Reuters-K500	**0.022**	0.002	0.0002	0.000
Scene	**0.932**	0.504	0.170	0.639	Scene	**0.938**	0.448	0.050	0.640
Slashdot	**0.179**	0.053	0.000	0.144	Slashdot	**0.086**	0.045	0.000	0.016
Yelp	**0.830**	0.607	0.067	0.514	Yelp	**0.858**	0.526	0.390	0.415
Average Rank	**1.875**	**1.875**	3.750	2.500	Average Rank	**1.375**	1.875	3.875	2.875



(a) Subset Accuracy (b) Hamming score

(c) Micro-averaged F1 score (d) Macro-averaged F1 score

Fig. 2: Nemenyi critical distance diagrams for all effectiveness metrics. The number given within parentheses after the method name indicates the rank position of the method.

5.2 Efficiency Analysis

Table 3 presents the experiments we conducted where we measure the efficiency of the methods in terms of execution time over the datasets. Our results display that BT performs exceedingly better than the baselines. On average, our method exhibits 1,678.88% better performance with a minimum of 81.4% improvement on the Scene dataset which has low number of labels (6). We can see that BT achieves higher improvement margins on datasets with large number of labels or samples since the BR and CC scale linearly with the number of labels. Furthermore, although LP scales similarly to BT in terms of label count, the number of classes LP predicts increases exponentially, since it assigns a slack class for each

label combination, whereas BT only predicts one continuous value regardless of the number of label combinations.

Table 3: Execution time of the methods in seconds (lower is better). Average is calculated by calculating the mean execution time of the methods except BT. Improvement (%) is the increase in efficiency BT has over the average execution time. "*" denotes the estimated time for completion for TLC experiments.

Method	20NG	EukaryotePseAAC	Imdb	Mediamill	Reuters-K500	Scene	Slashdot	Yelp
BT	**214**	**237**	**1,403**	**594**	**38**	**18**	**40**	**79**
BR	2,357	491	22,413	4,046	1,041	36	510	280
CC	2,473	472	22,816	6,983	1,972	35	495	276
LP	2,097	579	180,000*	20,609	1,576	27	367	531
Average	2,309	514	75,076	10,546	1530	33	457	362
Improvement (%)	978.9	116.9	5251.1	1675.4	3925.4	81.4	1043.3	358.6

Figure 3 illustrates the two-tailed Nemenyi significance test for our efficiency experiments. It can be seen that BT displays statistically significantly better efficiency in time, while the baselines show similar performance.

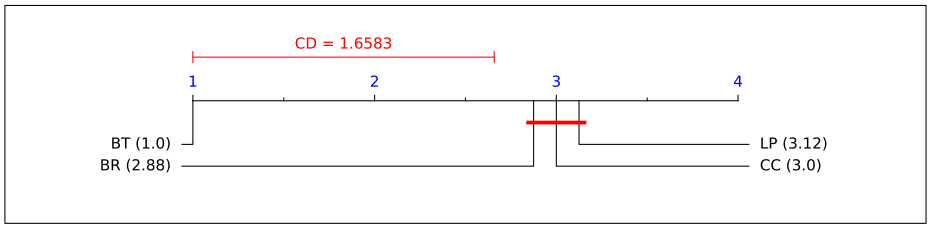


Fig. 3: Nemenyi critical distance diagram for execution time. The number given within parentheses after the method name indicates the rank position of the method.

Our combined results indicate that BT demonstrates similar effectiveness compared to our baselines in a much smaller time frame. Therefore, in streaming environments, BT is a more suitable PT method for MLSC.

5.3 Effects of Different Regression Algorithms on BT

We performed experiments on BT to evaluate the effect of different regression algorithms offered in the framework we utilize. Table 4 illustrates the subset accuracy and Hamming score results on these experiments. Micro and macro-averaged F1 scores are not included here as they show similar results.The experiments demonstrate that overall HT regressor and Linear Regression (LR)

provides the best effectiveness results. We observe that simple models such as Passive-Aggressive (PA) Regression are not suitable for streams with large number of labels in terms of subset accuracy; however, PA regressor displays much better effectiveness in such streams (e.g, Mediamill and Reuters-K500) in terms of Hamming score.

Table 4: Subset accuracy and Hamming scores for BT using different regressors (higher is better).

Regressor	20NG	EukaryotePseAAC	Imdb	Mediamill	Reuters-K500	Scene	Slashdot	Yelp
Subset Accuracy								
HT Regresor [9]	**0.187**	0.776	0.054	0.004	0.018	0.870	**0.149**	**0.593**
Linear Regression [8]	0.160	**0.784**	**0.079**	**0.005**	0.013	**0.875**	0.116	0.582
K-NN Regressor [4]	0.017	0.119	0.001	0.000	**0.042**	0.355	0.117	0.126
PassiveAggressive [5]	0.000	0.002	0.000	0.000	0.003	0.080	0.000	0.0322
Hamming Score								
HT Regresor	**0.917**	**0.988**	0.889	0.346	0.820	**0.976**	**0.912**	**0.889**
Linear Regression	0.914	0.886	**0.896**	0.346	0.827	0.914	0.907	0.887
K-NN Regressor	0.637	0.751	0.569	0.333	0.785	0.754	0.777	0.542
PassiveAggressive	0.700	0.651	0.684	**0.867**	**0.953**	0.606	0.752	0.498

The experiments show that BT is most suitable for streams in which partially correct predictions or label-wise accuracy is important. Since subset accuracy is a strict metric (as it requires exactly matching predictions), more complex regressors are needed for data streams with high number of classes. Furthermore, we found that with LR, the learning rate for the intercept (bias variable) needs to be high for it to provide accurate predictions. However, we were not able to discern the main cause behind this and left it for future work. For this reason, we did not use the default parameters given by the framework for LR(we used $intercept\_lr = 0.5$ which is selected through grid search).

6 Conclusion

In this paper, we propose Binary Transformation method for multi-label classification that employs regression algorithms by transforming the labels into a continuous value. Our method allows fast execution while exploiting label dependencies. We perform our evaluation on three of the most prevalent PT methods using eight datasets with varying problem domains and properties. Our results show that BT achieves statistically similar effectiveness while providing a much higher efficiency, an average of about 1,700%, in terms of execution time. The results demonstrate that BT is a better PT technique for multi-label data streams than the previous work in the field.

For future work, we plan to study the effectiveness of BT on ensemble methods and develop procedures to reduce the range of the solution domain of our algorithm for higher scalability in terms of number of labels.

References

1. Boutell, M.R., Luo, J., Shen, X., Brown, C.M.: Learning multi-label scene classification. Pattern recognition **37**(9), 1757–1771 (2004)
2. Büyükçakir, A., Bonab, H., Can, F.: A novel online stacked ensemble for multi-label stream classification. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management. pp. 1063–1072 (2018)
3. Cherman, E.A., Monard, M.C., Metz, J.: Multi-label problem transformation methods: a case study. CLEI Electronic Journal **14**(1), 4–4 (2011)
4. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE transactions on information theory **13**(1), 21–27 (1967)
5. Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., Singer, Y.: Online passive aggressive algorithms (2006)
6. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research **7**(Jan), 1–30 (2006)
7. Herrera, F., Charte, F., Rivera, A.J., Del Jesus, M.J.: Multilabel classification. In: Multilabel Classification, pp. 17–31. Springer (2016)
8. Hocking, R.R.: Developments in linear regression methodology: 1959–l982. Technometrics **25**(3), 219–230 (1983)
9. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 97–106 (2001)
10. Junior, J.C., Faria, E., Silva, J., Cerri, R.: Label powerset for multi-label data streams classification with concept drift. In: Proceedings of the 5th Symposium on Knowledge Discovery, Mining and Learning. pp. 97–104. Faculdade de Computação-Universidade Federal de Uberlândia (2017)
11. Kong, X., Philip, S.Y.: An ensemble-based approach to fast classification of multi-label data streams. In: 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom). pp. 95–104. IEEE (2011)
12. Lang, K.: The 20 newsgroup dataset. `http://people.csail.mit.edu/jrennie/20Newsgroups/` (visited on 14/08/2021) (2008), `http://people.csail.mit.edu/jrennie/20Newsgroups/`
13. Montiel, J., Halford, M., Mastelini, S.M., Bolmier, G., Sourty, R., Vaysse, R., Zouitine, A., Gomes, H.M., Read, J., Abdessalem, T., et al.: River: machine learning for streaming data in python (2021)
14. Nemenyi, P.B.: Distribution-free multiple comparisons. Princeton University (1963)
15. Nguyen, H.L., Woon, Y.K., Ng, W.K.: A survey on data stream clustering and classification. Knowledge and information systems **45**(3), 535–569 (2015)
16. Pant, P., Sabitha, A.S., Choudhury, T., Dhingra, P.: Multi-label classification trending challenges and approaches. In: Emerging Trends in Expert Applications and Security, pp. 433–444. Springer (2019)
17. Read, J.: Scalable multi-label classification. Ph.D. thesis, University of Waikato (2010)
18. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. Machine learning **85**(3), 333–359 (2011)
19. Read, J., Reutemann, P., Pfahringer, B., Holmes, G.: Meka: a multi-label/multi-target extension to weka. The Journal of Machine Learning Research **17**(1), 667–671 (2016)

20. Sajnani, H., Saini, V., Kumar, K., Gabrielova, E., Choudary, P., Lopes, C.: Classifying yelp reviews into relevant categories (2012)
21. Snoek, C.G., Worring, M., Van Gemert, J.C., Geusebroek, J.M., Smeulders, A.W.: The challenge problem for automated detection of 101 semantic concepts in multimedia. In: Proceedings of the 14th ACM International Conference on Multimedia. pp. 421–430 (2006)
22. Sousa, R., Gama, J.: Multi-label classification from high-speed data streams with adaptive model rules and random rules. Progress in Artificial Intelligence **7**(3), 177–187 (2018)
23. Strubell, E., Ganesh, A., McCallum, A.: Energy and policy considerations for deep learning in nlp. arXiv preprint arXiv:1906.02243 (2019)
24. Szymanski, P., Kajdanowicz, T.: Scikit-multilearn: a scikit-based python environment for performing multi-label classification. The Journal of Machine Learning Research **20**(1), 209–230 (2019)
25. Tai, F., Lin, H.T.: Multilabel classification with principal label space transformation. Neural Computation **24**(9), 2508–2542 (2012)
26. Tidake, V.S., Sane, S.S.: Multi-label classification: a survey. International Journal of Engineering and Technology **7**(4.19), 1045–1054 (2018)
27. Tsoumakas, G., Katakis, I., Vlahavas, I.: Mining Multi-label Data, pp. 667–685. Springer US, Boston, MA (2010). https://doi.org/10.1007/978-0-387-09823-4$_3$4, https://doi.org/10.1007/978-0-387-09823-4_34
28. Tsoumakas, G., Katakis, I., Vlahavas, I.: Random k-labelsets for multilabel classification. IEEE transactions on knowledge and data engineering **23**(7), 1079–1089 (2010)
29. Tsoumakas, G., Vlahavas, I.: Random k-labelsets: An ensemble method for multilabel classification. In: European Conference on Machine Learning. pp. 406–417. Springer (2007)
30. Xu, D., Shi, Y., Tsang, I.W., Ong, Y.S., Gong, C., Shen, X.: Survey on multi-output learning. IEEE transactions on neural networks and learning systems **31**(7), 2409–2429 (2019)
31. Xu, J., Liu, J., Yin, J., Sun, C.: A multi-label feature extraction algorithm via maximizing feature variance and feature-label dependence simultaneously. Knowledge-Based Systems **98**, 172–184 (2016)
32. Zhang, M.L., Li, Y.K., Liu, X.Y., Geng, X.: Binary relevance for multi-label learning: an overview. Frontiers of Computer Science **12**(2), 191–202 (2018)
33. Zhang, M.L., Zhou, Z.H.: A review on multi-label learning algorithms. IEEE transactions on knowledge and data engineering **26**(8), 1819–1837 (2013)
34. Zheng, X., Li, P., Chu, Z., Hu, X.: A survey on multi-label data stream classification. IEEE Access **8**, 1249–1275 (2019)