Online Multi-Label Classification with Adaptive Model Rules

Ricardo Sousa[1] and João Gama[1,2]

[1] LIAAD/INESC TEC, Universidade do Porto, Portugal
rtsousa@inesctec.pt
[2] Faculdade de Economia, Universidade do Porto, Portugal
jgama@fep.up.pt

Abstract. The interest on online classification has been increasing due to data streams systems growth and the need for Multi-Label Classification applications have followed the same trend. However, most of classification methods are not performed on-line. Moreover, data streams produce huge amounts of data and the available processing resources may not be sufficient. This work-in-progress paper proposes an algorithm for Multi-Label Classification applications in data streams scenarios. The proposed method is derived from multi-target structured regressor AMRules that produces models using subsets of output attributes(output specialization strategy). Performance tests were conducted where the operation modes global, local and subset approaches of the proposed method were compared to each other and to others online multi-label classifiers described in the literature. Three datasets of real scenarios were used for evaluation. The results indicate that the subset specialization mode is competitive in comparison to local and global approaches and to other online multi-label classifiers.

Keywords: $Multi-Label \cdot Classification \cdot AMRules \cdot DataStreams$

1 Introduction

Nowadays, data streams systems are very common (sensors systems, network monitoring logs, video streams, ...) [1]. These systems produce data unlimitedly in real time at high rates. Collected data can not be all stored and processed in just one procedure but one example at a time. Moreover, the data may present changes over time [2]. Therefore, systems such as regressors and classifiers need to perform training and prediction operations dynamically through online systems [3]. Some classification problems require that more than one class label should be assigned to an example. Two different examples may have a different number of class labels assigned [4]. The process that tries to solve this problem is called Multi-Label Classification(MLC) [2]. Formally, representing $\mathcal{D} = \{..., (\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), ..., (\mathbf{x}_i, \mathbf{y}_i), ...\}$ as an unbounded data stream, where $\mathbf{x}_i = [x_{i,1} \cdots x_{i,j} \cdots x_{i,M}]$ is a M-dimensional vector containing the data descriptive variables $x_{i,j}$ (input attributes) of the i^{th} example (considering an example

as a reference) and $\mathbf{y}_i$ corresponds to the response (output attributes) that consists of subset of nominal labels λ_k such that $\mathbf{y}_i \subseteq \{\lambda_1, ..., \lambda_k, ..., \lambda_L\}$, where L is the number of possible labels. The objective of MLC is to learn a function $f(\mathbf{x}_i) \rightarrow \mathbf{y}_i$ that maps the input values of $\mathbf{x}_i$ into the output values of $\mathbf{y}_i$. Online MLC is used in several domains such as Biology(gene and protein function classification) [5], Engineering (Network Monitoring and sensor applications) [6], Economics (online stock market data) [7], Social Sciences (social networks evolution) [8], Library and Information Science (text categorization) [7] and Multimedia (image, video and music categorization and annotation) [4]. Among classification techniques, structured classifiers present the advantage of selecting the most discriminative features implicitly, without requiring variables scaling. Moreover, these classifiers are resilient to outliers and the produced models are easily interpreted [8]. From structured classifiers, Rules Learning algorithms presents high modularity due to the fact that each rule can be interpreted individually [9]. The rule learning is independent which is an advantage when compared to tree-based algorithms. Modularity of rule sets can be explored to overcome the global and local methods thought rule specialization on a subset of output variables [10]. This work suggests a solution for MLC on data streams based on the algorithm AMRules and inspired on a regression approach [10]. This method was evaluated through a comparison of performance measures against other methods found in the literature. In addiction, the local and global operation mode were also compared to the subset approach. This paper presents the following structure. Section 2 summarizes related work(small presentation of online multi-label classifiers found in literature) and Section 3 describes the proposed rule-based algorithm for online MLC. The performance tests are described in Section 4. The results are discussed on Section 5 and the conclusions are remarked in Section 6.

2 Related work

In this section, some existing online MLC approaches are briefly described. Typically, most approaches are based on problem transformation [8]. The output set of labels $\mathbf{y}_i$ are transformed into a vector of outputs variables $[y_{i,1} \cdots y_{i,k} \cdots y_{i,L}]$, where $y_{i,k} \in \{0, 1\}$ are binary. If label λ_k is assigned to the i^{th} example then $y_{i,k} = 1$, otherwise $y_{i,k} = 0$. Here, the outputs variables are redefined as $\mathbf{y}_i = [y_{i,1} \cdots y_{i,k} \cdots y_{i,L}]$. The Binary Relevance (BR) is a simple multi-label classifier that uses directly the problem transformation. An online binary classifier trains and predicts for the k^{th} output variable only. The prediction procedure is represented by $\hat{\mathbf{y}}_i = [f_1(\mathbf{x}_i), ..., f_k(\mathbf{x}_i), ..., f_L(\mathbf{x}_i)]$, where f_k represent the classifier of the k^{th} output variable. This classifier is used as a baseline in the performance tests. Classifier Chains (CC) also uses the problem transformation like the BR method [11]. The L outputs variables indexes are shuffled in a sequence. Then, a classifier k is used to model the inputs and the first $(k-1)$ outputs variables. The prediction can be expressed as $\hat{\mathbf{y}}_i = [f_1(\mathbf{x}_i), ..., f_k(\mathbf{x}_i, \hat{y}_{i,1}, ..., \hat{y}_{i,k-1}), ..., f_L(\mathbf{x}_i, \hat{y}_{i,1}, ..., \hat{y}_{i,L-1})]$ (posteriorly reordered as before shuffling). Multi-Label

Hoeffding Trees (MHT) is an online structured classifier based on a decision tree that uses the Hoeffding bound criterion in the induction. The algorithm uses the information gain in the split decision and multi-label classifiers at the tree leaves [2]. The process can be modelled as $\hat{\mathbf{y}}_i = f_n(\mathbf{x}_i)$, where f_n is a basic online multi-label classifier of n leaf.

3 Multi-Label AMRules for Classification

In this section, Multi-Label AMRules (ML-AMRules) algorithm and its underlying principles are presented. As main principle, this algorithm is based on the adaptation of the multi-target AMRules regressor to the MLC problem through problem transformation [10]. This section also presents the underlying Rule Learning theory, the description of ML-AMRules training and prediction (multi-target adaptation to the MLC problem) and the description of the local and global modes.

3.1 Rule Learning

Rule R is defined as $\mathcal{A} \Rightarrow \mathcal{C}$ implication where the antecedent $\mathcal{A}$ is a conjunction of conditions (called literals) of the input variables $\mathbf{x}_i$, and the consequent $\mathcal{C}$ is a predicting function (in this context, it is a basic online multi-label classifier). For numerical data, literals may present the forms $(X_j \leq v)$ and $(X_j > v)$, where X_j represents the j^{th} input variable, meaning that $x_{i,j}$ must be less or equal to v, and $x_{i,j}$ must be greater than v, respectively. Regarding nominal data, literals may present forms $(X_j = v)$ expressing that $x_{i,j}$ must be equal to v or $(X_j \neq v)$ indicating that $x_{i,j}$ must be different than v. R is said to cover $\mathbf{x}_i$ if, and only if, $\mathbf{x}_i$ satisfies all the literals in $\mathcal{A}$. The support of the input variables of an example, $S(\mathbf{x}_i)$, corresponds to a set of rules that cover $\mathbf{x}_i$. Function (the basic classifier) in $\mathcal{C}$ returns a prediction $\hat{\mathbf{y}}_i$ if a rule R_r covers the example input variables $\mathbf{x}_i$. Data structure $\mathcal{L}_r$ containing the necessary statistics(about the rule and the examples) to the algorithm training and prediction (expand the rule, detect changes and identify anomalies,...) is associated to each rule R_r. A particular rule D, called default rule, exists for initial conditions and for the case of none of the current rules covers the example $(S(\mathbf{x}_i) = \emptyset)$. The antecedent of D and its statistics $\mathcal{L}_D$ start as an empty set. Rule set is formed by a set of U learned rules defined as $\mathcal{R} = \{R_1, \cdots, R_r, \cdots, R_U\}$ and a default rule D as depicted in Figure 1. In summary, Rule Learning allows to create partitions on the input variables space and build a model on each partition. Consequently, the linear model can fit more easily to data.

3.2 ML-AMRules Training(Rule Induction)

Algorithm 1 illustrates the pseudo-code for the ML-AMRules training. The algorithm initializes the statistics $\mathcal{L}_D$ of the default rule and starts the rule set $\mathcal{R}$ out empty. When an example $(\mathbf{x}_i, \mathbf{y}_i)$ is received, the algorithm searches for rules

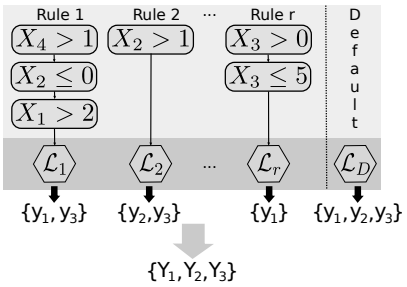

Fig. 1. Multi-Label AMRules based on subsets specialization.

that covers the example input variables $\mathbf{x}_i$. Considering one rule $R_r \in S(\mathbf{x}_i)$, the example input variables $\mathbf{x}_i$ are submitted to anomaly ($\texttt{isAnomaly}(\mathcal{L}_r, \mathbf{x}_i)$) and change ($\texttt{changeDetected}(\mathcal{L}_r, \mathbf{x}_i)$) detection in order to prune the examples. For change detection, the Page-Hinkley (PH) is used [12]. For anomaly detection, a method based on probability of example occurrence was used [10]. In case of anomaly, the example is simply rejected and in case of change detection, R_r is removed from the rule set(the rule is outdated). Otherwise, the statistics $\mathcal{L}_r$ are updated($\texttt{update}(\mathcal{L}_r)$).

Rule expansion (addition of new literal) is attempted and in an affirmative case($\texttt{expand}(R_r)$), specialization of the rule on the output subset and rule addiction to $\mathcal{R}$ are performed (Section 3.4). This specialization leads to more accurate predictions and it increases the speed of processing. The example input variables $\mathbf{x}_i$ may not be covered by any rule. Consequently, the statistics of the default rule $\mathcal{L}_D$ are updated and the expansion is attempted. If an expansion occur, the default rule D is added to the rule set $\mathcal{R}$ and a new default rule is initialized. The training process also involves the computation of a weight parameter for the case of more than one rule covers the example, in the prediction operations. The parameter is the mean error with a fading factor, for the rule r and output variable k, $e_{r,k} = \frac{T_{r,k}}{W_r}$. $T_{r,k}$ is the accumulated error and W_r is the number of examples observed since the last expansion, both affected by a fading factor $0 < \alpha < 1$. These parameters are computed as

$$T_{r,k} \leftarrow \alpha T_{r,k} + |\hat{y}_{i,k} - y_{i,k}|,\ W_r \leftarrow \alpha W_r + 1, \tag{1}$$

where and $y_{i,k}$ is the true value and $\hat{y}_{i,k}$ is post-training prediction. Each output variable under the rule R_r is associated to a linear and to an output mean predictors. The output-mean predictor is simply defined as $\hat{y}^r_{i,k} = \frac{1}{n}\sum_{u=1}^{n} y_{u,k}$, where n is the number of examples seen since last expansion. The purpose is to allow fast training convergence of the linear predictor. The error $e_{r,k}$ is computed for each predictor.

3.3 Rule Expansion

Rule R_r expansion consists of adding a new literal to the antecedent A_r. The new literal is determined by finding the input variables and by computing the

Algorithm 1 Adaptive Model Rules training

$\mathcal{R} \leftarrow \emptyset,\ D \leftarrow 0$
for all $(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}$ **do**
  for all $R_r \in S(\mathbf{x}_i)$ **do**
    if $\neg$`isAnomaly`$(\mathcal{L}_r, \mathbf{x}_i)$ **then**
      if `changeDetected`$(\mathcal{L}_r, \mathbf{x}_i)$ **then**
        $\mathcal{R} \leftarrow \mathcal{R} \setminus \{R_r\}$
      else
        $R_c \leftarrow R_r$
        `update`$(\mathcal{L}_r)$
        expanded $\leftarrow$ `expand`(R_r)
        if *expanded* = `TRUE` **then**
          Compute $\mathcal{O}'_c$
          $\mathcal{O}_c \leftarrow \mathcal{O}'_c$
          $\mathcal{R} \leftarrow \mathcal{R} \cup \{R_c\}$
  if $S(\mathbf{x}_i) = \emptyset$ **then**
    `update`$(\mathcal{L}_D)$
    expanded $\leftarrow$ `expand`(D)
    if *expanded* = `TRUE` **then**
      $\mathcal{R} \leftarrow \mathcal{R} \cup \{D\}$
      $D \leftarrow 0$

split-points that maximize the uniformity of two groups of values, divided by v. This procedure uses an Extended Binary Search Tree(E-BST) with limited depth that keeps data statistics [13]. Mean Information Gain(MIG) is the maximizing function for splitting an input X_j given the split-point v with respect to the output variables. MIG is defined as

$$MIG(X_j, v) = \frac{1}{|\mathcal{O}_r|} \sum_{u \in \mathcal{O}_r} IG_u(X_j, v), \tag{2}$$

where $IG_u(X_j, v)$ is the Information Gain of splitting X_j given v considering the output variable Y_u, and $\mathcal{O}_r$ is the set of output variables indexes currently being considered by the rule R_r. The Information Gain(IG) is defined as

$$IG_u(X_j, v) = H_u(E) - \frac{|E_L|}{|E|}\frac{H_u(E_L)}{H_u(E)} - \frac{|E_R|}{|E|}\frac{H_u(E_R)}{H_u(E)}, \tag{3}$$

where,

$$H_u(E) = -[p \log(p) + (1-p)\log(1-p)],$$

is the entropy, p is the probability of $Y_u = 1$ and E denotes the set of examples processed since the last expansion. If the input variables are numerical, $E_L = \{\mathbf{x}_i \in E : x_{i,j} \leq v\}$ and $E_R = \{\mathbf{x}_i \in E : x_{i,j} > v\}$. Considering nominal input variables, $E_L = \{\mathbf{x}_i \in E : x_{i,j} = v\}$ and $E_R = \{\mathbf{x}_i \in E : x_{i,j} \neq v\}$. The rule expansion procedure uses the Hoeffding bound [14] to determine the minimum

number of examples n required to expand, which states that the true mean of a random variable β, with range P, will not differ from the sample mean more than ϵ with probability $1-\delta$. The Hoeffding bound is defined as $\epsilon = \sqrt{\frac{P^2 \ln(1/\delta)}{2n}}$. This procedure suggests several candidates for splitting $[MIG(X_j, v_1)...MIG(X_j, v_c)]$ that are organized in decreasing order. A comparison between the two best splits is performed using the difference: $\beta = MIG(X_j, v_1) - MIG(X_j, v_2)$. The range of β is $[0, 1]$, therefore $P = 1$. In case of $\beta > \epsilon$, $MIG(X_j, v_1)$ is the best split with the probability of $1 - \delta$. Threshold τ is defined to limit ϵ for numerical instabilities. If $\epsilon < \tau$ is met, the split with higher $MIG(X_j, v_1)$ is selected and the expansion takes place. The relation sign is determine by finding the set of example(E_L or E_R) with the lowest $H(E)$. This incremental algorithm presents $\mathcal{O}(n)$ complexity which makes it suitable for online scenarios.

3.4 Specializing on subsets of the output variables

Let $\mathcal{O}_r$, E_{best} be the set of the current learning outputs indexes for rule R_r and the set of examples with the lowest $H(E)$ in the splitting, respectively. $\mathcal{O}'_r$ denotes the new learning outputs that consists of set of output variables indexes that reduce in entropy after the split:

$$\mathcal{O}'_r = \left\{ u : u \in \mathcal{O}_r \wedge \frac{H_u(E_{best})}{H_u(E)} < 1 \right\}. \tag{4}$$

A complementary rule R_c which contains the set of the pruned output variables is also added to the rule set in order to keep their information. The antecedents of R_c and R_r are equal before the expansion. R_c learns only from the output attributes $Y_u \in \mathcal{O}'_c$ in order to satisfy $\mathcal{O}'_c = \mathcal{O}_r \setminus \mathcal{O}'_r$.

3.5 ML-AMRules Prediction

In the prediction, the rules R_r that covers the example are considered $\Lambda = \{r : R_r \in \mathcal{S}(\mathbf{x}_i)\}$. The next step consists of choosing the predictor that presents lower error (the output-mean or linear predictor) to retrieve an estimation $\hat{y}^r_{i,k}$ for the output k. The final prediction is defined as

$$\hat{y}_{i,k} = \begin{cases} 1 & \text{if } m_{i,k} > 0.5 \\ 0 & \text{if } m_{i,k} \leq 0.5 \end{cases} \tag{5}$$

$$m_{i,k} = \sum_{u \in \Lambda} \theta_{u,k} \hat{y}^u_{i,k}, \quad \theta_{u,k} = \frac{(e_{u,k} + \varepsilon)^{-1}}{\sum_{t \in \Lambda} (e_{t,k} + \varepsilon)^{-1}}, \tag{6}$$

where ε is a small positive number used to prevent numerical instabilities. If the output attribute Y_u can not be predicted by any rule, the prediction is given by the default rule D.

3.6 Local and Global approaches

Two ML-AMRules operation modes are presented in this subsection according to local and global methods. The local approach is based on an instantiation of the ML-AMRules for each output variable. This operation mode resembles the BR approach. Each output variable has an independent rule set that models it. The final prediction is produced by combining the individual predictions. Considering the global approach, the rules are learned and predicted for all output attributes using one instantiation. For implementing the global algorithm the rule specialization is not performed (steps 11 and 12 in Algorithm 1).

4 Experimental Setup

This section presents the evaluation tests of the proposed algorithm described in Section 3. The proposed algorithm is compared to three online multi-label algorithms described in Section 2 in terms of their performance. The same comparison was performed for local and global operations mode described in Subsection 3.6. The algorithm CC is incorporated in the open source MEKA platform that includes both batch and online multi-label algorithms. The algorithms were implemented in JAVA programming language and are based on WEKA [2]. BR, MHT and the proposed methods ML-AMRules were implemented in the Massive Online Analysis (MOA) platform. Its an open source platform of Machine Learning and Data Mining algorithms applied to data streams. This platform was also implemented in JAVA programming language. Real scenarios datasets 20NG, mediamill and OHSUMED were used to simulate data streams. These datasets are described on literature [8] and some features are presented in Table 1.

Table 1. Dataset description

Dataset	#Examples	#Outputs	#Inputs
20NG-F	19300	20	1006
mediamill	43907	101	120
OHSUMED-F	13929	23	1002

The examples of the datasets were replicated four times and shuffled due to the need of a significant number of examples by these algorithms. Performance example-based measures, Exact Match, Accuracy, Precision, Recall and F-measure were used [15]. This evaluation used the prequential mode where the algorithm starts by predicting the output values and the example-based measures. Posteriorly, it uses the example for training [16]. Datasets examples were divided into 100 windows and the above mentioned measures were computed for each window. Finally, the mean and the standard deviation of the measures of all windows were computed. Perceptron with a logistic activation function

was used as linear predictor by all algorithms due to its models simplicity, low computational cost and low error rates [17].

5 Results

In this section, the evaluation results are presented. The results are organized by performance measures. Tables 2 to 6 present the Accuracy, Exact Match, Precision, Recall and F-measure results of the online multi-label algorithms for each dataset. The ML-AMR(S), ML-AMR(G) and ML-AMR(L) correspond to the subset, global and local ML-AMRules operation modes, respectively.

Table 2. Accuracy. Mean and standard deviation values.

Dataset	ML-AMR(S)	ML-AMR(G)	ML-AMR(L)	BR	MHT	CC
20NG	0.65±0.07	**0.67±0.07**	0.63±0.06	0.65±0.07	0.66±0.07	0.64±0.05
mediamill	**0.37±0.01**	0.35±0.01	0.35±0.01	0.34±0.01	0.35±0.01	0.34±0.01
OSHUMED	0.46±0.06	0.46±0.06	0.44±0.05	**0.47±0.06**	**0.47±0.06**	0.44±0.05

Table 2 shows that the ML-AMRules approaches present values that have competitive accuracy. Among ML-AMRules approaches, the subset and global approaches seem to stand out.

Table 3. Exact Match. Mean and standard deviation values.

Dataset	ML-AMR(S)	ML-AMR(G)	ML-AMR(L)	BR	MHT	CC
20NG	0.62±0.06	**0.65±0.07**	0.61±0.06	0.64±0.07	0.64±0.07	0.62±0.06
mediamill	0.04±0.01	0.04±0.01	0.04±0.00	0.04±0.01	**0.04±0.01**	0.04±0.01
OSHUMED	0.30±0.04	0.30±0.05	0.29±0.04	**0.31±0.04**	0.31±0.05	0.30±0.05

Table 3 presents low values for mediamill dataset due to high number of possibles labels for all algorithms. In this aspect, the ML-AMRules approaches present lower values in comparison to other algorithm.

Table 4. Precision. Mean and standard deviation values.

Dataset	ML-AMR(S)	ML-AMR(G)	ML-AMR(L)	BR	MHT	CC
20NG	0.68±0.07	**0.69±0.07**	0.65±0.06	0.69±0.07	0.68±0.07	0.65±0.06
mediamill	0.40±0.02	0.41±0.02	**0.41±0.01**	0.41±0.01	0.41±0.01	0.40±0.01
OSHUMED	0.50±0.06	0.50±0.06	0.48±0.05	0.51±0.06	**0.51±0.06**	0.50±0.05

Table 4 displays favourable precision values for ML-AMRules approaches. Among ML-AMRules modes, global and local present better values.

Table 5. Recall. Mean and standard deviation values.

Dataset	ML-AMR(S)	ML-AMR(G)	ML-AMR(L)	BR	MHT	CC
20NG	0.66±0.07	**0.68±0.07**	0.64±0.06	0.67±0.07	0.67±0.07	0.65±0.07
mediamill	**0.68±0.02**	0.67±0.01	0.66±0.01	0.67±0.01	0.67±0.01	0.66±0.01
OSHUMED	0.59±0.06	0.59±0.06	0.57±0.05	0.59±0.06	**0.60±0.06**	0.58±0.05

Table 5 exhibits predominance of the ML-AMRules approaches. The global and subset approaches present better performance.

Table 6. F-Measure. Mean and standard deviation values.

Dataset	ML-AMR(S)	ML-AMR(G)	ML-AMR(L)	BR	MHT	CC
20NG	0.66±0.07	**0.68±0.07**	0.64±0.06	0.67±0.07	0.67±0.07	0.64±0.06
mediamill	0.46±0.01	0.47±0.01	0.47±0.01	**0.47±0.01**	0.47±0.01	0.46±0.01
OSHUMED	0.51±0.06	**0.52±0.06**	0.50±0.05	0.50±0.06	0.51±0.06	0.50±0.05

Table 6 reveals the preponderance of the ML-AMRules approaches. The global and subset approaches present better performance. In general, the mean values present very close values due to the fact that all algorithms use the same linear predictor and due to datasets complexity.

6 Conclusions

This paper is the result of a preliminary work suggests a multi-target algorithm adaptation to the multi-label problems using Rule Learning methods. It can be concluded that the proposed approach is competitive when compared to online multi-label algorithms from the literature. The subset approach has shown to be competitive against local and global approaches. The experiments have shown that the datasets implicit models should be characterized in order to understand the data distribution.

7 Acknowledgments

This work was partly supported by the European Commission through MAESTRA (ICT-2013-612944) and the Project TEC4Growth - Pervasive Intelligence, Enhancers and Proofs of Concept with Industrial Impact/NORTE -01-0145-FEDER-000020 is financed by the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF).

References

1. Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Richard Kirkby, and Ricard Gavaldà. New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 139–148, New York, NY, USA, 2009. ACM.
2. Jesse Read, Albert Bifet, Geoff Holmes, and Bernhard Pfahringer. Scalable and efficient multi-label classification for evolving data streams. *Mach. Learn.*, 88(1-2):243–272, July 2012.
3. João Gama. *Knowledge Discovery from Data Streams*. Chapman and Hall / CRC Data Mining and Knowledge Discovery Series. CRC Press, 2010.
4. Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and SašO Deroski. An extensive experimental comparison of methods for multi-label learning. *Pattern Recogn.*, 45(9):3084–3104, September 2012.
5. Amanda Clare and Ross D. King. Knowledge discovery in multi-label phenotype data. In *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*, PKDD '01, pages 42–53, London, UK, UK, 2001. Springer-Verlag.
6. Charu C. Aggarwal. *Data Streams: Models and Algorithms (Advances in Database Systems)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
7. Xiangnan Kong and Philip S. Yu. *An ensemble-based approach to fast classification of multi-label data streams*, pages 95–104. 12 2011.
8. Aljaz Osojnik, Pance Panov, and Saso Dzeroski. Multi-label classification via multi-target regression on data streams. In *Discovery Science - 18th International Conference, DS 2015, Banff, AB, Canada, October 4-6, 2015, Proceedings*, pages 170–185, 2015.
9. Johannes Fürnkranz, Dragan Gamberger, and Nada Lavra. *Foundations of Rule Learning*. Springer, 2012.
10. João Duarte and João Gama. Multi-Target Regression from High-Speed Data Streams with Adaptive Model Rules. In *IEEE conference on Data Science and Advanced Analytics*, 2015.
11. Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II*, ECML PKDD '09, pages 254–269, Berlin, Heidelberg, 2009. Springer-Verlag.
12. E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.
13. Elena Ikonomovska, João Gama, and Saso Dzeroski. Learning model trees from evolving data streams. *Data Min. Knowl. Discov.*, 23(1):128–168, 2011.
14. Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.
15. M. S. Sorower. A literature survey on algorithms for multi-label learning.
16. João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. On evaluating stream learning algorithms. *Machine Learning*, 90(3):317–346, 2013.
17. Eneldo Loza Mencía and Johannes Fürnkranz. Pairwise learning of multilabel classifications with perceptrons. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2008, part of the IEEE World Congress on Computational Intelligence, WCCI 2008, Hong Kong, China, June 1-6, 2008*, pages 2899–2906, 2008.