JMLR: Workshop and Conference Proceedings 17 (2011) 19–25 2nd Workshop on Applications of Pattern Analysis

Streaming Multi-label Classification

Jesse Read JMR30@CS.WAIKATO.AC.NZ
Albert Bifet ABIFET@CS.WAIKATO.AC.NZ
Geoff Holmes GEOFF@CS.WAIKATO.AC.NZ
Bernhard Pfahringer BERNHARD@CS.WAIKATO.AC.NZ
Department of Computer Science
University of Waikato, Hamilton, New Zealand

Editor: Tom Diethe, José L. Balcázar, John Shawe-Taylor, and Cristina Tîrnăucă

Abstract

This paper presents a new experimental framework for studying multi-label evolving stream classification, with efficient methods that combine the best practices in streaming scenarios with the best practices in multi-label classification. Many real world problems involve data which can be considered as multi-label data streams. Efficient methods exist for multi-label classification in non streaming scenarios. However, learning in evolving streaming scenarios is more challenging, as the learners must be able to adapt to change using limited time and memory. We present a new experimental software that extends the MOA framework. **M**assive **O**nline **A**nalysis (MOA) is a software environment for implementing algorithms and running experiments for online learning from evolving data streams. It is released under the GNU GPL license.

1. Introduction

Nowadays, data is generated at an increasing rate from sensor applications, measurements in network monitoring and traffic management, log records or click-streams in web exploration, manufacturing processes, call-detail records, email, blogs, RSS feeds, social networks, and other sources. Real-time analysis of these data streams is becoming a key area of data mining research as the number of applications demanding such processing increases.

Data streams pose several challenges for data mining algorithm design. First, they must make use of limited resources (time and memory). Second, they must deal with data whose nature or distribution changes over time.

We present a software system specifically designed for the task of *streaming multi-label classification*; the generalisation of the traditional multi-class (single-label) task. In multi-label classification, instead of a single class-label, each example can be associated with *multiple* labels. Multi-label classification has seen considerable development in recent years, but so far most of this work has been carried out in the static context of *batch learning* where train-then-test scenarios or cross-fold validation evaluations are typical. Despite the fact that most of the example data stream applications above can be applied to multi-label contexts, very few authors have looked explicitly at the task in a data stream context. It is a new context, and a software framework for multi-label streaming was needed to have benchmarks to compare to.

2. A Framework For Multi-label Data Stream Mining

We use the following notation. $\mathcal{X} = \mathbb{R}^M$ is the input attribute space. $\boldsymbol{x} \in \mathcal{X}$ is an *instance*, which can be represented as an M-vector $\boldsymbol{x} = [x_1, \ldots, x_M]$. $\mathcal{L} = \{1, \ldots, L\}$ is a set of L possible labels. A *label set*, i.e. label relevances for a particular instance, is represented by an L-vector $\boldsymbol{y} = [y_1, \ldots, y_L] = \{0,1\}^L$ where $y_j = 1$ iff the jth label is relevant (otherwise $y_j = 0$). $\sum \boldsymbol{y}$ indicates the number of positive label relevances indicated in $\boldsymbol{y}$. $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ is the ith *example* in a theoretically infinite stream of instances where y^i_j is the binary relevance of the jth label with respect to the ith instance, and x^i_a is the value of the ath attribute of the ith instance. In the data stream prequential evaluation setting (Gama et al., 2009) a classifier h makes a prediction $\hat{\boldsymbol{y}}_i = h(\boldsymbol{x}_i)$, which is evaluated against $\boldsymbol{y}_i$, then $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ becomes a new training example.

Typically, a classifier in a real-world incremental context interleaves classification and training, where new examples are classified (i.e. labelled) automatically as they become available, and can be used for training as soon as their label assignments are confirmed or corrected. It may be a human or community of humans doing the checking, or checking might be automatic: For example, a robot learns multiple actions to complete a task, and each time it attempts the task, it can auto-evaluate its success.

A complete framework for mining multi-label instances from evolving data streams must have the following components:

- Generators of evolving multi-label streams
- Multi-label adaptive classifiers
- An incremental streaming evaluation component

Despite the ubiquitous presence of multi-label data streams in the real world, assimilating and storing them on a large scale with both labels and time-order intact has so far proved largely impractical. Furthermore, in-depth domain knowledge may be necessary to determine and pinpoint changes to the concepts represented by the data, making drift-analysis difficult. This fact provides strong motivation for generating synthetic data.

2.1. Evaluation Methodology

Unlike in single-label evaluation, a simple example-*accuracy* metric does not suffice to give a clear picture of performance, due to the extra dimension of the label space. Treating label combinations as examples is too harsh, since combinations must match exactly to be correct; whereas treating individual labels as examples can be overly lenient and ignores the importance of label combinations. Other measures of predictive performance are needed.

We use *accuracy* as defined in (Tsoumakas and Katakis, 2007), for a window of N instances, as:

$$\text{Accuracy} = \frac{1}{N} \sum_{i=1}^{N} \frac{\sum_{j=1}^{L} y^i_j \wedge \hat{y}^i_j}{\sum_{j=1}^{L} y^i_j \vee \hat{y}^i_j}$$

As argued in (Read et al., 2009a), it is essential to include several evaluation measures in any multi-label experiment. Given the extra label dimension, it is otherwise possible to optimise for certain evaluation measures. Thus, we include two other, contrasting, measures; *F1 macro averaged by label*, and *log-loss*.

The F-measure is the harmonic mean between precision and recall, common to information retrieval. It can be calculated from the true positives, true negatives, false positives, and false negatives of predictions $\boldsymbol{y}$ as compared to actual values y. While accuracy is averaged over examples, this F-measure is averaged over labels:

$$\text{F-measure}^{\times L} = \frac{1}{L}\sum_{j=1}^{L} \text{F1}(\{\hat{y}_j^1, \ldots, \hat{y}_j^N\})$$

Finally, we use *log loss*, introduced in (Read et al., 2009a), distinct from other measures because it punishes prediction confidences rather than 0/1 relevances, and worse errors are punished exponentially more harshly. Log loss thus provides a good contrast to other measures. If $\boldsymbol{w}_j \in \mathbb{R}$ is the prediction confidence for the jth label, then:

$$\text{LLoss} = \frac{1}{NL}\sum_{i=1}^{N}\sum_{j=1}^{L} \min(-\text{lloss}(\hat{w}_j^i, y_j^i), \ln(N))$$

where $\text{lloss}(\hat{w}, y) = \ln(\hat{w})y + \ln(1-\hat{w})(1-y)$

We have used a dataset-dependent maximum of $\ln(N)$ to limit the magnitudes of the penalty, where N is the size of the evaluation window. Such a limit, as explained in (Schapire and Singer, 1999), serves to smooth the values and to prevent a small subset of poorly predicted labels from greatly distorting the overall error. Note that, as a loss metric, the best possible score for log loss is 0.0.

In data stream mining, the most frequently used measure for evaluating predictive accuracy of a classifier is prequential accuracy. Gama et al. (Gama et al., 2009) propose to use a forgetting mechanism for estimating holdout accuracy using prequential accuracy: either a sliding window of size w with the most recent observations can be used, or fading factors that weigh observations using a decay factor α. The output of the two mechanisms is very similar, as every window of size w_0 can be approximated by some decay factor α_0.

3. Generating Synthetic Multi-label Data

Despite the ubiquitous presence of multi-label data streams in the real world, assimilating and storing them on a large scale with both labels and time-order intact has so far proved largely impractical. Furthermore, in-depth domain knowledge may be necessary to determine and pinpoint changes to the concepts represented by the data, making drift-analysis difficult. This fact provides strong motivation for generating synthetic data.

There are several existing methods for generating synthetic multi-label data, for example (Park and Fürnkranz, 2008; Qu et al., 2009), but most are task-specific and not sufficient for general use. Of these, only (Qu et al., 2009) introduces concept shift, by changing generation parameters, but is far too simple, involving only two labels, to be considered as a good approximation of real-world data. Overall, prior methods produce data which contains very few attributes and labels, as few as two to three, and are therefore not a generally good real-world approximation, even though they can be useful for analysing or highlighting particular characteristics of certain algorithms. Our framework contributes a general method which can simulate co-occurrence patterns and dependencies between labels as found in real data, as well as any number and type of attributes in the attribute space, and their relationships to the label space. See (Read et al., 2009b) for details.

4. Multi-label Classification Learners

We review some of the base multi-label approaches on the literature implemented in our framework, and then discuss their applications to data stream contexts.

A simple base-line method for multi-label classification is the *binary relevance* method (BR). BR transforms a multi-label problem into multiple binary problems, such that binary models can be employed to learn and predict the relevance of each label. BR has often been overlooked in the literature because it fails to take into account label correlations directly during the classification process (Godbole and Sarawagi, 2004; Tsoumakas and Vlahavas, 2007; Read et al., 2008), although there are several methods that overcome this limitation e.g. (Cheng and Hüllermeier, 2009; Godbole and Sarawagi, 2004; Read et al., 2009a). BR can be applied directly to data streams by using incremental binary base models.

An alternative paradigm to BR is the *label combination* or *label powerset* method (LC). LC transforms a multi-label problem into a single-label (multi-class) problem by treating all label combinations as atomic labels, i.e. each label set becomes a single class-label within a single-label problem. Thus, the set of single class-labels represents all distinct label subsets in the original multi-label representation. Disadvantages of LC include its worst-case computational complexity and tendency to over-fit the training data, although this problem has been largely overcome by newer methods (Tsoumakas and Vlahavas, 2007; Read et al., 2008).

Another multi-label approach is *pairwise classification* (PW), where binary models are used for every possible *pair* of labels(Fürnkranz et al., 2008). PW performs well in some contexts, but the complexity in terms of models ($(L \times (L-1)/2)$ for L labels) means that this approach is usually intractable on larger problems.

A less common approach is that of applying a threshold to the confidence outputs (e.g. posterior probabilities) of a single-label classifier, for each test instance, and treating all class labels falling above a threshold as the relevant labels. Multi-label training examples are duplicated at training time, so as to create one single-label example for each relevant label. We call this the *ranking and threshold* method (RT)(Tsoumakas and Katakis, 2007).

Note that these are all *problem transformation* methods, wherein a multi-label problem is transformed into one or more single-label problems, and any off-the-shelf multi-class classifier (or binary in the case of BR and PW) can be used. These methods are interesting generally due to their flexibility and general applicability. In fact, all methods in the literature use, mention or compare to them.

4.1. Using Problem Transformation Methods in Data Streams

An important advantage of problem transformation is the ability to use any off-the-shelf single-label base classifier to suit requirements. In this section we discuss using incremental classifiers to meet the requirements of data streams.

4.1.1. BR-based methods in data-stream settings

BR-based methods are composed of binary classifiers; one for each label. It is straightforward to apply BR to a data stream contexts by using an incremental classifier for the binary models. The advantages of BR methods, many of which are also particularly beneficial to data streams, are low time complexity and the ability to run in parallel, as well as a good resistance to label-set overfitting (since this method learns on a per-label basis). Class-label imbalance in BR schemes may become exacerbated by a huge numbers of training examples, therefore prior works has developed countermeasures, e.g. per-label thresholding methods or classifier weightings (Ráez et al., 2004).

In (Read et al., 2009a) we introduced ensembles of BR (EBR), and ensembles of *classifier-chains* (ECC). Both these methods can perform in a streaming fashion by simply employing incremental binary base classifiers.

4.1.2. LC-based methods in incremental settings

Using LC-based methods for incremental classification implies a single-label problem with the special case that the label space expands over time. This is because LC's single-label transformation treats every label combination as a single-label, but in a stream environment new combinations appear over time. One of two strategies can be employed: either an algorithm adaptation to be able to accommodate changes to the label space over time; or problem transformation, where a number of examples are initially buffered, so as to prime an LC model with label combinations, which can learn thereafter in an incremental fashion. During buffering, another incremental method can serve to adhere to the 'ready to predict at any point' requirement (for example, RT, see below).

A possible algorithm adaptation strategy is to alter label-priors in a Naive Bayes scheme to account for the introduction of new class labels over time. The potential rapid growth of the label space in this scenario would make time complexity difficult to control. The problem transformation strategy has lower time complexity bounds, although the LC model, once built, is forced to discard any training examples with new label combinations, since the model cannot account for them, and is therefore very sensitive to variations in the label space. The buffer may need to be rather large, or the model may need to be restarted on a regular basis.

The PS method we presented in (Read et al., 2008) is actually well suited to the buffering scheme. Since rare label sets are sub-sampled for common label sets, far fewer sets need to be discarded during classification. As a result, a PS model is much more robust against variation in the label-space, and needs to be restarted less frequently (or can be primed with a smaller buffer). The ensemble setting we presented (EPS) further mitigates the negative effects of buffering by reducing overfitting on the original label space.

4.1.3. PW-based methods in data-stream settings

PW-based methods can be used with incremental classifiers, and this has been done in the past (Fürnkranz et al., 2008). These methods can be used only if the number of labels considered is very limited. As in this paper, we would like to consider the case where the number of labels is not restricted, we don't use PW-based methods in our experiments.

4.1.4. RT-based methods in incremental settings

The RT method of classification can be seen as a special case of LC where each class label is a "combination", with the advantage over LC that it can be used directly as we know the number of labels, and therefore the number of posible combinations before classification begins. However, like BR and PW, RT does not explicitly model label combinations.

4.1.5. Multi-label Hoeffding Trees

The Hoeffding Tree is the state-of-the-art classifier for single-label data streams, and it performs prediction by choosing the majority class at each leaf. Predictive accuracy can be increased by adding naive Bayes models at the leaves of the trees. A *Multi-label Hoeffding Tree* is an extension of the Hoeffding Tree to deal with multi-label data

We emphasise that PS in this context meets all the requirements of a data stream: it always updates its model incrementally, does not store instances (only a constant number of distinct label combinations are stored at each leaf), and predicting and learning are carried out in real time.

4.1.6. Ensemble Methods

A popular way to achieve superior predictive performance, scalability and parallelization is to use ensemble learning methods, combining several models under a voting scheme to form a final prediction.

To allow adaption to concept drift we consider ADWIN Bagging (Bifet et al., 2009) i.e. bagging using ADWIN as a change detector. ADWIN is a change detector and estimator that keeps a variable-length window of recently seen items with theoretical guarantees on memory. When change is detected, under-performing (i.e. supposedly out-dated) ensemble members are discarded and replaced with fresh ones.

5. Conclusions

We presented a new experimental framework for multi-label data-stream classification, extension of MOA (Bifet et al., 2010). MOA can be found at: `http://moa.cs.waikato.ac.nz` Our experimental framework is the first of its kind involving instance-incremental multi-label stream classifiers, generating a range of synthetic multi-label data, and employing a variety of multi-label evaluation measures.

References

Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Richard Kirkby, and Ricard Gavaldà. New ensemble methods for evolving data streams. In *KDD '09*. ACM, 2009.

Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Philipp Kranen, Hardy Kremer, Timm Jansen, and Thomas Seidl. Moa: Massive online analysis, a framework for stream classification and clustering. *Journal of Machine Learning Research - Proceedings Track*, 11: 44–50, 2010.

Weiwei Cheng and Eyke Hüllermeier. Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, 76(2-3):211–225, 2009. doi: 10.1007/s10994-009-5127-5. URL `http://dx.doi.org/10.1007/s10994-009-5127-5`.

Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Mencía, and Klaus Brinker. Multilabel classification via calibrated label ranking. *Machine Learning*, 73(2):133–153, November 2008. ISSN 0885-6125. doi: 10.1007/s10994-008-5064-8. URL `http://dx.doi.org/10.1007/s10994-008-5064-8`.

João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. Issues in evaluation of stream learning algorithms. In *KDD '09*, pages 329–338, 2009.

Shantanu Godbole and Sunita Sarawagi. Discriminative methods for multi-labeled classification. In *PAKDD '04*, pages 22–30. Springer, 2004. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.1.3819`.

Sang-Hyeun Park and Johannes Fürnkranz. Multi-label classification with label constraints. Technical report, Knowledge Engineering Group, TU Darmstadt, 2008.

Wei Qu, Yang Zhang, Junping Zhu, and Qiang Qiu. Mining multi-label concept-drifting data streams using dynamic classifier ensemble. In *ACML*, 2009.

Arturo Montejo Ráez, Luis Alfonso Ureña López, and Ralf Steinberger. Adaptive selection of base classifiers in one-against-all learning for large multi-labeled collections. In *EsTAL*, pages 1–12, 2004.

Jesse Read, Bernhard Pfahringer, and Geoff Holmes. Multi-label classification using ensembles of pruned sets. In *ICDM'08*, pages 995–1000. IEEE, 2008.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. In *ECML '09*, pages 254–269. Springer-Verlag, 2009a.

Jesse Read, Bernhard Pfahringer, and Geoffrey Holmes. Generating synthetic multi-label data streams. In *MLD '09*, September 2009b.

Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, December 1999. URL `http://dx.doi.org/10.1023/A:1007614523901`.

Grigorios Tsoumakas and Ioannis Katakis. Multi label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3), 2007. URL `http://lpis.csd.auth.gr/paper\_details.asp?publicationID=219`.

Grigorios Tsoumakas and Ioannis P. Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In *ECML '07*, pages 406–417. Springer-Verlag, 2007. URL `http://mlkd.csd.auth.gr/publication\_details.asp?publicationID=241`.