Working Paper Series
ISSN 1177-777X

Efficient Multi-label Classification for Evolving Data Streams

Jesse Read, Albert Bifet, Geoff Holmes and Bernhard Pfahringer

Working Paper: 04/2010
May 2010

Efficient Multi-label Classification for Evolving Data Streams

Jesse Read
University of Waikato
Hamilton, New Zealand
jmr30@cs.waikato.ac.nz

Albert Bifet
University of Waikato
Hamilton, New Zealand
abifet@cs.waikato.ac.nz

Geoff Holmes
University of Waikato
Hamilton, New Zealand
geoff@cs.waikato.ac.nz

Bernhard Pfahringer
University of Waikato
Hamilton, New Zealand
bernhard@cs.waikato.ac.nz

ABSTRACT

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.

This paper proposes a new experimental framework for studying multi-label evolving stream classification, and new efficient methods that combine the best practices in streaming scenarios with the best practices in multi-label classification. We present a Multi-label Hoeffding Tree with multi-label classifiers at the leaves as a base classifier. We obtain fast and accurate methods, that are well suited for this challenging multi-label classification streaming task. Using the new experimental framework, we test our methodology by performing an evaluation study on synthetic and real-world datasets. In comparison to well-known batch multi-label methods, we obtain encouraging results.

Categories and Subject Descriptors

H.2.8 [**Database applications**]: Database Applications—*Data Mining*

General Terms

Algorithms

Keywords

Data streams, ensemble methods, concept drift, decision trees

1. INTRODUCTION

Real-time analysis of data streams is becoming a key area of data mining research as the number of applications demanding such processing increases. 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 exploring, manufacturing processes, call detail records, email, blogging, twitter posts, and other sources.

In the traditional supervised classification task, each example is associated with a single class label. A classifier learns to associate each new unseen example with exactly one of these known class labels. When each example may be associated with *multiple* labels, then this is called *multi-label classification*. Hence multi-label classification is simply the classification task where each example may be associated with multiple labels.

A common approach to multi-label classification is *problem transformation*, whereby a multi-label problem is transformed into one or more single-label problems. In this fashion, a single-label classifier can be employed to make single-label classifications, and these can then be transformed back into multi-label predictions. The alternative to problem transformation is *algorithm adaption*; to modify an existing single-label algorithm directly for the purpose of multi-label classification.

A data stream environment has different requirements from the traditional batch learning setting. The most significant are the following: process one example at a time, use a limited amount of memory, work in a limited amount of time, and be ready to predict at any time. Therefore, data streams pose several challenges for data mining algorithm design. Algorithms must make use of limited resources (time and memory), and they must deal with data whose nature or distribution changes over time.

A *multi-label data stream* is a data stream with the same properties as multi-label data. Multi-label learning problems have received considerable attention in the machine learning literature, but prior work focusses almost exclusively on a batch learning environment with train-test or cross-validation scenarios. To the best of our knowledge this is the first work on multi-label classification within the constraints of a data stream context with evolving data.

In Section 2 we review related work. Section 3 presents a novel framework for generating synthetic multi-label streams. Section 4 presents new methods for multi-label data stream classification, and Section 5 shows a first comprehensive cross-method comparison. We summarise and draw conclusions in Section 6.

Source code and datasets will be made available at `http://sourceforge.net/projects/moa-datastream`.

2. RELATED WORK

Before embarking on an empirical evaluation of the methods presented in this paper, let us review existing work on multi-label learning and data stream mining.

The most well-known and widely documented problem transformation method is the *binary relevance* method (`BR`) [18]. `BR` transforms any multi-label problem into multiple binary problems; one for each label. Each binary classifier is responsible for predicting the association of a single label. `MLkNN` [21] is a well-known `BR`-based lazy-classification scheme. An improved lazy approach, `IBLR`, has recently been presented in [4].

`BR` has often been sidelined in the literature under the consensus view that it is crucial to take into account label correlations during the classification process, which `BR` fails to do by default [9, 19, 14]. However there are simple ways to combat this problem without leaving the `BR`-scheme. Examples include stacking `BR` classification outputs [9]. In [16], we presented an efficient chaining scheme which passes label-correlation information between binary classifiers. We also showed that bagging `BR` in an ensemble produces good results, especially for larger datasets.

An alternative paradigm to `BR` is the *label combination* or *label powerset* method (`LC`). `LC` treats all label sets as atomic (single) labels to form a single-label problem in which the set of possible single labels represents all distinct label subsets in the original multi-label representation. In other words, each label set becomes a single class-label within a single-label problem.

`LC` is well recognised as facing computational complexity problems [19, 14], as well as issues with over-fitting [14]. Several works have addressed these issues. Perhaps the most well-known is the `RAkEL` system [19]. `RAkEL` draws a random label subsets from the label set and trains `LC` classifiers on each in an ensemble scheme. In more recent work, we presented `PS` [14], which uses pruning to reduce the computational complexity of `LC`. This method proved to be competitive in terms of efficiency, while retaining the advantages of an `LC` scheme.

When binary classifiers are used for every possible *pair* of labels, multi-label learning becomes *pairwise* classification (`PW`). A good example of this approach is `CLR`, presented in [7]. While having showed considerable success in small-dimensional problems `PW`-methods face the complexity of $(N \times (N-1)/2)$ classifiers for N labels, which becomes infeasible where relatively large numbers of labels are involved, especially in a streaming environment problems.

A *Hoeffding tree* [6] is an incremental, anytime decision tree induction algorithm that is capable of learning from massive data streams, assuming that the distribution generating examples does not change over time. Hoeffding trees exploit the fact that a small sample can often be enough to choose an optimal splitting attribute. This idea is supported mathematically by the Hoeffding bound, which quantifies the number of observations (in our case, examples) needed to estimate some statistics within a prescribed precision (in our case, the information gain of an attribute). More precisely, the Hoeffding bound states that with probability $1-\delta$, the true mean of a random variable of range R will not differ from the estimated mean after n independent observations by more than:

$$\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}.$$

A theoretically appealing feature of Hoeffding Trees not shared by many other incremental decision tree learners is that it has sound theoretical guarantees of performance. Using the Hoeffding bound one can show that the output of a Hoeffding tree is asymptotically nearly identical to that of a non-incremental learner using infinitely many examples. See [6] for details.

Ensemble methods are combinations of several models whose individual predictions are combined in some manner (e.g., averaging or voting) to form a final prediction. Ensemble learning classifiers often have better accuracy and they are easier to scale and parallelize than single classifier methods. In [11] Oza and Russell developed online versions of bagging and boosting for data streams. They show how the process of sampling bootstrap replicates from training data can be simulated in a data stream context.

In [2] two new state-of-the-art bagging methods were presented: ASHT Bagging using trees of different sizes, and `ADWIN` Bagging using a change detector to decide when to discard underperforming ensemble members.

A first approach to multi-label data stream classification is reported in [13], however the empirical evaluation is done using WEKA, with non streaming classifiers.

3. A FRAMEWORK FOR GENERATING SYNTHETIC DATA STREAMS

Despite the ubiquitous presence of multi-label data streams in the real world, they can rarely be easily assimilated on a large scale with both labels and timestamps intact and there may be issues with sensitive data – for example with e-mail, and medical text corpora. In many cases, in-depth domain knowledge may be necessary to determine and pinpoint changes to the concepts represented by the data.

Hence the motivation to generate synthetic multi-label data streams is to 1) increase the pool of multi-label stream data and thereby also the depth of analysis and conclusions which can be drawn in respect to the performance of various algorithms; 2) allow for theoretically infinite data streams; and 3) help conduct specific analysis of incremental multi-label algorithms, such as how they respond to concept drift.

In [15], we described a novel problem transformation-inspired approach for generating synthetic multi-label data streams. Here we present an improved version of that work, which is able to take into account label sets as opposed to just label pairs, is more efficient, more theoretically grounded, and is configured by fewer parameters, but is based upon the same principles. Next we review prior work related to this task, followed by an in depth presentation of our framework.

3.1 Prior Work

Generating single-label synthetic data streams has been common practice for some time. The work in [10] provides the software environment **M**assive **O**nline **A**nalysis (MOA) for implementing algorithms and running experiments for online learning from data streams. This framework (software available at `http://www.cs.waikato.ac.nz/~abifet/MOA/`)

contains a variety of methods for generating single-label data. This is expanded in [2] which additionally considers concept drift, as opposed to simply an incremental context.

Methods for generating synthetic multi-label data are much less developed. The authors of [20] generate a multi-label synthetic dataset where the examples pertaining to certain labels are associated with certain Gaussian distributions. In [3], a tree structure is used with random weight vectors generated for each node.[12] uses a set of pairwise constraints, and generates random permutations which satisfy this set.

Overall, prior methods produce data which usually contains very few attributes and labels (as few as two to three in the works just mentioned), relatively few examples, and were never intended for large scale multi-label evaluation, rather mainly for highlighting certain characteristics of the algorithms that the authors present. Furthermore, none of these data generation techniques are for creating data stream contexts.

3.2 A Generator for Multi-label Data Streams

It has already been well established that multi-label data can be transformed into single-label data via the process of problem transformation [18]. Our claim is that the reverse transformation is also possible: single-label data can be transformed into multi-label data. This allows for a general and powerful framework which can create a multi-label synthetic data stream by using off-the-shelf single-label data generators. Thus the production of a multi-label stream is independent of the actual data-generation process.

We mentioned the `MOA` framework, which already provides state-of-the-art functionality for generating single-label synthetic data streams under a variety of schemes. Our framework deals with the task of composing a multi-label data stream from any such single-label data generation scheme.

We define the following notation.

- Let $\mathcal{X}$ denote the input attribute space, where $\mathcal{X} \subset \mathbb{R}^d$
- Let $x \in \mathcal{X}$ be an *instance*, i.e. *feature vector* $x = ([x]_1, \cdots, [x]_d)^d$
- Let $\mathcal{L} = \{l_1, l_2, \cdots, l_N\}$ denote the finite label set of N labels
- Let $l \in \mathcal{L}$ be a single label
- Let (x, l) be a *single-label example*
- Let $S \subseteq \mathcal{L}$ be a *label subset*; representable as a feature vector $S = (l_1, l_2, \cdots, l_N) \in \{0, 1\}^N$ where:

$$S[j] = \begin{cases} 1 & \text{if } l_j \in S \\ 0 & \text{if } l_j \notin S \end{cases}$$

- Let (x, S) be a *multi-label example*
- Let $\mathcal{D} = \{(x_1, S_1), (x_2, S_2), \cdots, (x_t, S_t), \cdots\}$ be a multi-label data stream where (x_t, S_t) is the current example

As in single-label generation, our framework must supply the number of class-labels $|\mathcal{L}|$ as a parameter. Additionally, there are several essential elements which relate specifically to multi-label data.

Primarily, each example may have multiple labels, and hence an average number of z labels over the stream (where z is supplied as a parameter).

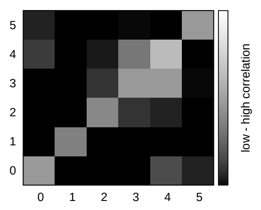

Figure 1: Label relationships displayed in the form of a heatmap where lighter shades indicate higher probabilities. Prior probabilities are displayed in the diagonal.

Importantly, in multi-label data, relationships exist between the labels. In the absence of label relationships, multi-label data is uninteresting, since this would mean we could simply treat each label a separate binary problem, without any loss of information. Aside from generally finding that $P(l_j|l_k) \approx P(l_j)$, which we expect in the absence of any strong relationship (as implied by Bayes' rule), we note strong domain-dependent relationships between labels. Figure 1 shows a representation of the relationships between labels in the *Scene* dataset (mentioned later). The domain dependent relationships are seen clearly, for example labels 4 and 0.

Additionally to relationships between labels, there also exist relationships between labels and attributes. Clearly, simply more adding labels to a single-label example will not create a realistic multi-label example.

We demonstrate some of these relationships using text data, since it is both intuitive to examine, and also typical to multi-label data streams (although we have discovered similar effects in other kinds of data). Tables 1 and 2 relate to two text corpora which we worked with in [15]. They show the most frequent words for labels occurring *exclusively* of each other, together *in combination* with each other, and found *globally* across the dataset. Table 3 shows the average and standard deviations for specific word-features taken from the tables (in terms of predicting specific labels). In reference to these samples, influences between features and labels can be observed.

An attribute may identify a certain label: $x_a \rightarrow l_j$, where $x_a \in \mathcal{X}$ and $l_j \in \mathcal{L}$. An intuitive example is the word feature 'linux' (see Tables 1, and 3) which pertains strongly to the label `Linux` – occurring in over half of all documents associated with this label.

An attribute may identify a *combination* of labels; $x_a \rightarrow S$, where $x_a \in \mathcal{X}$ and $S \subseteq \mathcal{L}$; i.e. several labels co-occurring together, but *not* necessarily either of the labels occurring individually. This is the case in the *20 Newsgroups* dataset for the word 'arms' (see Tables 2 and 3), which tends to occur frequently only when the newsgroup post is also posted to *both* `politics.guns` *and* `misc.religion`.

There are also various *random effects* or *non-effects*. Words like 'anonymous' in *Slashdot* are generic and do not strongly indicate the presence or absence of either labels or combinations of labels. Such features are not helpful to classification and should already arguably have been removed by efficient feature selection. Therefore we need not consider them in a synthetic stream. Surprisingly, the case where an attribute value is near the average of the attribute of a combination; i.e. $P(x_a|\{l_1, l_2\}) \approx (P(x_a|l_1)+P(x_a|l_2))/2$, is *not* common.

Table 1: *Slashdot.* Most frequent words for labels `Linux` and `Mobile`

Global	`Linux`	`Mobile`	`{Linux,Mobile}`
anonymous	linux	mobile	linux
reader	ubuntu	iphone	open
game	source	anonymous	windows
story	open	reader	phone
reports	released	phone	netbook
world	anonymous	android	source
years	kernel	apple	mobile
released	software	phones	free

Table 2: *20 Newsgroups.* Most frequent words for labels `politics.guns` and `religion.misc`

Global	`politics.guns`	`religion.misc`	`{politics.guns, religion.misc}`
don	people	don	jews
1	don	people	arms
2	gun	christian	bear
people	time	god	don
time	government	years	koresh
good	fbi	good	fbi
make	guns	time	people
3	waco	make	news

Aside from parameters $|\mathcal{L}|$ and z, our framework only requires a single-label binary generator g. A prime advantage of our framework is that any single-label stream generator can be used for g. The initialisation process is as follows.

Prior probabilities are generated for all labels, i.e. $P(l_j) \in [0.0, 1.0]$ for all $j = 1 \cdots |\mathcal{L}|$. These probabilities are scaled according to parameter z so as to approximate to the desired average number of labels. Following this, a $|\mathcal{L}| \times |\mathcal{L}|$ probability matrix m (where each $m[j][k] = P(l_j|l_k)$) is filled for $\forall m[j][k] : 0 < j < k \leq |\mathcal{L}|$ with $P(l_j|l_k) \approx P(l_j)$. We override some of these values with random probabilities (within the constraints of probability laws) to simulate the domain-dependent relationships. Thereafter, the remaining half of the matrix $\forall m[j][k] : 0 < k < j \leq |\mathcal{L}|$ can be calculate according to Bayes' rule:

$$P(l_j|l_k) = \frac{P(l_k|l_j) \cdot P(l_j)}{P(l_k)}$$

Using the resulting matrix we can calculate the top n most-likely combinations $S_1 \cdots S_n$ where each $S_i \subseteq \mathcal{L}$. We use $n = \frac{|\mathcal{L}|}{2}$. These include single-labels, i.e. $|S_i| \geq 1$. From this list we create an attribute-label mapping ζ of size d where each attribute influences the presence or absence of a either a single label ($|S_i| = 1$) or combination of labels ($|S_i| > 1$), i.e. $\zeta[a] \rightarrow S_{a \mod n}$ for each $x[a]$.

Finally, the binary generator is initialised, and the generation process can begin. Figure 2 illustrates the overall process for generating a multi-label example. An initial label is chosen at random from the distribution of prior probabilities. Labels may then be added to this label to form a label set. A multi-label instance space is formed for these labels

Table 3: Distributions of word frequencies for certain labels both individually and in combination.

Slashdot for 'linux'	
$P(\texttt{Linux}\|\text{'}\mathit{linux}\text{'})$	0.60 ± 0.49
$P(\texttt{Mobile}\|\text{'}\mathit{linux}\text{'})$	0.02 ± 0.15
$P(\{\texttt{Linux},\texttt{Mobile}\}\|\text{'}\mathit{linux}\text{'})$	0.61 ± 0.51
20 Newsgroups for 'arms'	
$P(\texttt{politics.guns}\|\text{'}\mathit{arms}\text{'})$	0.14 ± 0.34
$P(\texttt{religion.misc}\|\text{'}\mathit{arms}\text{'})$	0.01 ± 0.10
$P(\{\texttt{politics.guns},\texttt{religion.misc}\}\|\text{'}\mathit{arms}\text{'})$	0.48 ± 0.50

```
nextInstance()
▷ randomly pick an initial label (index) for this example
$S = \{l \leftarrow \text{pick}(\text{norm}([P(l_1), \cdots, P(l_{|\mathcal{L}|})]))\}$
▷ add labels
while $l \geq 0$
    do $S \leftarrow S \cup l$
       $l \leftarrow \text{addLabel}(S)$
▷ generate an instance space for these labels
$x \leftarrow \text{genML}(S)$
return $(x, S)$
```

Figure 2: Algorithm for generating a multi-label example.

```
genML($S$)
▷ Create an empty instance
$x_m = (\cdot, \cdot, \cdots, \cdot)^d$
▷ Generate two binary examples (positive; negative)
$x_{+1} \leftarrow g.\text{genSL}(+1)$
$x_{-1} \leftarrow g.\text{genSL}(-1)$
▷ Fill the instance space $x_m$ according to $x_{-1}, x_{+1}$ and $\zeta$
for $a \leftarrow 1 \ldots d$
    do
        if $\zeta[a] \subseteq S$
           then $x_m[a] \leftarrow x_{+1}[a]$
           else $x_m[a] \leftarrow x_{-1}[a]$
return $x_m$
```

Figure 3: Generating a multi-label instance to fit a given label set.

according to the feature-label mapping ζ. Finally, instance and label-set are returned together as a newly generated multi-label example.

The auxiliary function $\text{pick}(R)$ simply returns j with probability $R[j]$, and -1 with probability $1.0 - \sum_j^{|R|} R[j]$. Equation 1 defines the function $\text{addLabel}(S)$ which takes a label set S and returns a label likely to be added to this set. Note that $\prod_{k=1}^{|S|} P(l_j|S[k]) = 0$ whenever $l_j \in S$ (a label cannot be added twice). A null label is possible, in which case $\text{addLabel}(S) = -1$ returns and the process of adding labels must halt. The process of forming a multi-label instance for a label set using ζ is outlined in Figure 3, and exemplified in Figure 4.

$$\text{addLabel}(S): \quad \textbf{return } \text{pick}\left(\prod_{k=1}^{|S|} P(l_1|S[k]), \cdots, \prod_{k=1}^{|S|} P(l_N|S[k])\right) \tag{1}$$

3.3 Adding Concept Drift

A new experimental framework for concept drift in streaming data was presented in [2]. The main goal of this framework is to introduce artificial drift to data stream generators in a straightforward way.

Considering data streams as data generated from pure distributions, we can model a concept drift event as a weighted combination of two pure distributions that characterizes the target concepts before and after the drift. This framework

Figure 4: A small illustration of how two binary instances are combined to form a multi-label instance.

attribute	$\mathcal{X}[1]$	$\mathcal{X}[2]$	$\mathcal{X}[3]$	$\mathcal{X}[4]$	$\mathcal{X}[5]$
mapping ζ	$\{l_1\}$	$\{l_2\}$	$\{l_3\}$	$\{l_2, l_3\}$	$\{l_1\}$

label(s)	instance (attribute space)						
	x_{-1} =(	0.9	0.8	0.2	0.9	-0.1	)
	x_{+1} =(	0.1	0.7	-0.1	0.8	0.2	)
$S = \{l_1, l_3\}$	x_m =(	0.9	0.7	0.2	0.8	-0.1	)

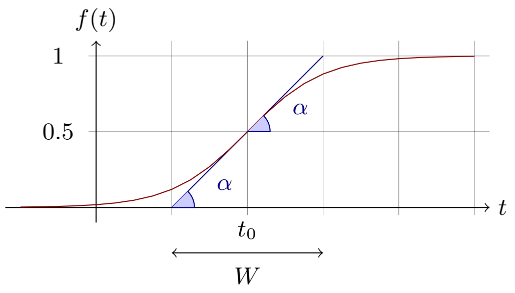

Figure 5: A sigmoid function $f(t) = 1/(1 + e^{-s(t-t_0)})$.

defines the probability that every new instance of the stream belongs to the new concept after the drift. It uses the sigmoid function, as an elegant and practical solution.

We see from Figure 5 that the sigmoid function

$$f(t) = 1/(1 + e^{-s(t-t_0)})$$

has a derivative at the point t_0 equal to $f'(t_0) = s/4$. The tangent of angle α is equal to this derivative, $\tan\alpha = s/4$. We observe that $\tan\alpha = 1/W$, and as $s = 4\tan\alpha$ then $s = 4/W$. So the parameter s in the sigmoid gives the length of W and the angle α. In this sigmoid model we only need to specify two parameters : t_0 the point of change, and W the length of change.

Definition 1. *Given two data streams a, b, we define $c = a \oplus^W_{t_0} b$ as the data stream built joining the two data streams a and b, where t_0 is the point of change, W is the length of change and*

- $\Pr[c(t) = a(t)] = e^{-4(t-t_0)/W}/(1 + e^{-4(t-t_0)/W})$
- $\Pr[c(t) = b(t)] = 1/(1 + e^{-4(t-t_0)/W})$.

In order to create a data stream with multiple concept changes, we can build new data streams joining different concept drifts:

$$(((a \oplus^{W_0}_{t_0} b) \oplus^{W_1}_{t_1} c) \oplus^{W_2}_{t_2} d) \ldots$$

Multi-label concept changes can be formed using the same method, where a, b, c, etc. are simply multi-label streams as defined by our framework.

4. MULTI-LABEL HOEFFDING TREES

We extend the Hoeffding Tree to deal with multi-label streams since the Hoeffding Tree is the state-of-the-art classifier for single-label data streams. A *Multi-label Hoeffding Tree* is an incremental decision tree classifier for multi-label data streams that it is based on the use of the Hoeffding bound as a criterion to decide whether to split nodes.

Clare and King [5] adapted C4.5 to multi-label data classification. We use the same strategy to develop a decision tree for multi-label data streams. We present two main extensions: the use of a new definition of entropy to compute information gain, and the use of multi-label classifiers at the leaves.

Information gain is a criterion used in leaf nodes to decide if it is worth splitting them or not. Information gain for an attribute A in a splitting node is the difference between the entropy of the training examples S at the node and the weighted sum of the entropy of the subsets S_v caused by partitioning on the values v of that attribute A.

$$\mathrm{Information\ Gain(S, A) = entropy(S)} - \sum_{\mathrm{v \in A}} \frac{|\mathrm{S_v}|}{|\mathrm{S}|} \mathrm{entropy(S_v)}$$

Hoeffding Trees expect that each example belongs to just one class. Entropy is used in C4.5 decision trees and single-label Hoeffding Trees for a set of examples S with N classes and probability $p(c_i)$ for each class c_i in the set S as

$$\mathrm{entropy_{SL}(S)} = -\sum_{\mathrm{i=1}}^{\mathrm{N}} \mathrm{p(c_i)\log(p(c_i))}$$

To deal with multi-label decision trees, we must use the following definition of entropy:

$$\mathrm{entropy_{ML}(S) = entropy_{SL}(S)} - \sum_{\mathrm{i=1}}^{\mathrm{N}} \mathrm{(1 - p(c_i))\log(1 - p(c_i))}$$

Entropy is a measure of the amount of uncertainty in the dataset. For each example, it is the information needed to describe all the classes it belongs to. In the case of multi-label examples, we need to add to the computation of the entropy the information needed to describe all the classes that it doesn't belong to. We do that by adding the term $(1 - p(c_i))\log(1 - p(c_i))$ for each class c_i.

The second important extension is the addition of multi-label classification at the leaves. We allow the insertion of any multi-label classifier, and use a majority-*label-set* classifier (the multi-label version of majority-class) as the default classifier.

In [14] we presented the pruned sets method `PS`. The motivation behind `PS` is to capitalise on the most important label relationships within a multi-label dataset. `PS` is based upon `LC`, but showed not only an improvement in predictive performance over `LC`, but also very significant gains in efficiency. By pruning away infrequently occurring label sets, much unnecessary and detrimental complexity is avoided. A post-pruning step breaks up the pruned sets into more frequently occurring subsets, and is able to reintroduce pruned instances into the data, ensuring minimal information loss. Its classification power comes from being able to take into account label combinations directly, and its efficiency makes it an ideal choice in the data stream problem. `LC`-based methods like `PS` are not naturally suited to data stream settings *on their own* because they focus around existing label sets, from which they create class-labels for an underlying single-label classifier. A `PS` classifier must be completely reset in order to take into account new combinations (an incremental version of `PS` is left for future work). In the leaves of a Hoeffding Tree, however, `PS` may be reset without affecting the core model structure, and thus becomes a viable solution. PS can either be primed with n instances and initialised with these, or reset whenever an ADWIN-monitor [1] detects change to the number of combinations being seen (PSa). PS requires parameters; we use pruning value $p = 1$, decomposition value $n = 1$, and Naïve Bayes as a single-label base classifier in all cases.

5. EXPERIMENTAL EVALUATION

The data stream evaluation framework and all algorithms evaluated in this paper were implemented in the Java programming language extending the MOA software. MOA includes a collection of offline and online methods as well as tools for evaluation.

5.1 Evaluation Measures and Methodology

Multi-label evaluation is not as straightforward as single-label evaluation, where the simple *accuracy* metric often suffices. In the single-label context, *accuracy* is simply the number of correctly labelled test instances relative to the total number of instances. However, this measure does not transfer well to the extra dimension of the label space in the multi-label context. If accuracy is *example-based*, then the label set must match exactly for each example to be considered correct, and the measure tends to be overly harsh. Other measures of predictive performance are needed.

We use the notation from Section 3.2, where Y_i is the *predicted* set for the ith example.

We use *subset accuracy* as defined in [18]:

$$\mathrm{SubsetAccuracy} = \frac{1}{|\mathcal{D}|}\sum_{\mathrm{i}=1}^{|\mathcal{D}|}\frac{|\mathrm{S_i} \cap \mathrm{Y_i}|}{|\mathrm{S_i} \cup \mathrm{Y_i}|}$$

Where Y_i is the predicted label set for the ith example, which is compared to S_i, the actual set. $|\mathcal{D}|$ is the number of examples that we are evaluating.

As we argued in [16], 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. For this reason we include two contrasting measures; macro-averaged F1, 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 (tp), true negatives (tn), false positives (fp) and false negatives (fn). While *subset accuracy* is averaged over examples, we use a *macro-average F-measure*; averaged over all labels:

$$F1_{Macro(\mathcal{L})} = \frac{1}{|\mathcal{L}|}\sum_{j=1}^{|\mathcal{L}|} F1(tp_j, fp_j, tn_j, fn_j) \qquad (2)$$

Finally we use *log loss*, which we introduced in [16], distinct from other measures because it punishes worse errors more harshly, and thus provides a good contrast to other measures. Rather than comparing predicted and actual sets, the prediction confidences of classifiers are evaluated, and the error is graded by the confidence at which it was predicted: predicting false positives with low confidence induces logarithmically less penalty than predicting with high confidence. If λ_j is the prediction confidence for the jth label, and $l_j \in \{0, 1\}$, then:

$$\begin{aligned}\mathrm{LogLoss} \;=\; & \frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|}\sum_{j=1}^{|\mathcal{L}|} -\max\Big(\log\frac{1}{|L|}, \\ & \log(\lambda_j)l_j + \log(1-\lambda_j)(1-l_j)\Big)\end{aligned}$$

We have used a dataset-dependent maximum of $log(\frac{1}{|\mathcal{L}|})$ to limit the magnitudes of the penalty. Such a limit, as explained in [17], 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.

Many multi-label algorithms, including most ensemble methods, initially result in a ranking, and require an extra process to separate relevant and irrelevant labels for each example to yield multi-label classifications. For log loss evaluation, we do not need to consider such a separation. For subset accuracy and F1-measure, we simply adjust a threshold over time according to label cardinality. If the predicted label cardinality becomes lower than the actual label cardinality, the threshold is adjusted upward, and adjusted downward in the case of the reverse. Obviously threshold adjustment is done posterior to each prediction.

In the analysis of running time we measure train time in seconds, and we measure memory use in terms of megabytes.

The evaluation methodology used was prequential [8], where every example was used for testing the model before using it to train. Results are averages of 10 runs.

5.2 Datasets

Table 4 provides statistics for a collection of multi-label datasets. *Scene* and *Yeast* are well known datasets in the multi-label literature (see for example [18]), although they are unfortunately of insufficient size for a data-stream setting. Nevertheless, we display them to give an idea of typical multi-label dimensions. *TMC2007*[1] contains instances of aviation safety reports that document problems which occurred during certain flights. The labels represent the problems being described by these reports. We use a reduced version of this dataset with the top 500 features selected, as specified in [19]. *IMDB* comes from the Internet Movie Database `http://imdb.org` (we obtained the data from `http://www.imdb.com/interfaces#plain`). We used the movie plot text summaries labelled with the relevant genres. *MediaMill* originates from the 2005 NIST TRECVID challenge dataset, a competition[2] which contains video data annotated with various concepts. In the final row of the table we list the range of parameters we used to generate the synthetic data.

Several different schemes are used as base generators for single-label binary synthetic data streams as required by the multi-label generation framework. The Random Tree Generator is the generator proposed by Domingos and Hulten [6], producing concepts that in theory should favour decision tree learners. It constructs a decision tree by choosing attributes at random to split, and assigning a random class label to each leaf. Once the tree is built, new examples are generated by assigning uniformly distributed random values to attributes which then determine the class label via

[1]`http://www.cs.utk.edu/tmw07/`
[2]`http://www.science.uva.nl/research/mediamill/challenge/`

the tree. The generator has parameters to control the number of classes, attributes, nominal attribute labels, and the depth of the tree. For consistency between experiments, two random trees were generated and fixed as the base concepts for testing—one *simple* and the other *complex*, where complexity refers to the number of attributes involved and the size of the tree.

The RBF (Radial Basis Function) generator was devised to offer an alternate complex concept type that is not straightforward to approximate with a decision tree model. This generator effectively creates a normally distributed hypersphere of examples surrounding each central point with varying densities. Drift is introduced by moving the centroids with constant speed initialized by a drift parameter.

We use the following synthetic streams as the base generators (parameter g) in our multi-label framework:

- RTS: Simple random tree that has ten nominal attributes with five values each, and a tree depth of five, with leaves starting at level three and a 0.15 chance of leaves thereafter.
- RTC: Simple random tree that has one hundred nominal attributes with five values each, a tree depth of five, with leaves starting at level three and a 0.15 chance of leaves thereafter.
- RRBFS refers to a simple random RBF data set—50 centers and 10 attributes.
- RRBFC is more complex—50 centres, 100 attributes.
- SYNT is defined as

$$(((RTS_{z=1.5} \oplus^{W}_{t_0} RTS_{z=4}) \oplus^{W}_{2t_0} RTS_{z=2.5}) \oplus^{W}_{3t_0} RTS_{z=9.5})$$

For the multi-label generation process, we experiment with parameters $z = 1.5$ (approximate average number of labels per example) and $|\mathcal{L}| = 8$ (number of labels) for streams RTS and RRBFS, and $z = 5.0$ and $|\mathcal{L}| = 30$ for streams RTC and RRBFC.

5.3 Methods

We test the aforementioned data streams with the following classifiers:

- `HT`: Multi-label Hoeffding Tree
- `HT-PS`: Multi-label Hoeffding Tree with `PS` classsifier at the leaves, and using the first one thousand examples to compute the label combinations
- `HT-PSA`: Multi-label Hoeffding Tree with `PS` classsifier at the leaves, and using `ADWIN` monitoring the number of label combinations, and computing the label combinations every time a change is detected.
- `BBR`: Bagging of ten `BR` classifiers with Hoeffding Trees as the base classifier.
- `BAG HT-PSA`: `ADWIN` Bagging of ten decision trees, using HT-PSA as base classifier. `ADWIN` monitors the number of label combinations.

5.4 Results

The prequential evaluation procedure was carried out on one million examples from the RandomTree and RandomRBF datasets. Tables 5, 6 and 7 display the final subset accuracy, log-loss, and F-1-Macro measures respectively. Table 7 shows the culmative train time of the methods, and Table 8 the memory used. Additionally, the prequential learning curves with a window size of 10,000 for synthetic data, and 10% for real datasets, were plotted for for subset accuracy, log loss, and macro F1-Measure for SYNT on one million samples dataset are plotted in Figures 6, 7, and 8.

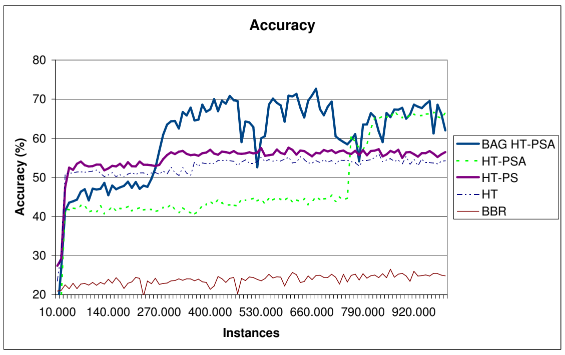

Figure 6: Subset accuracy on SYNT with three concept drifts.

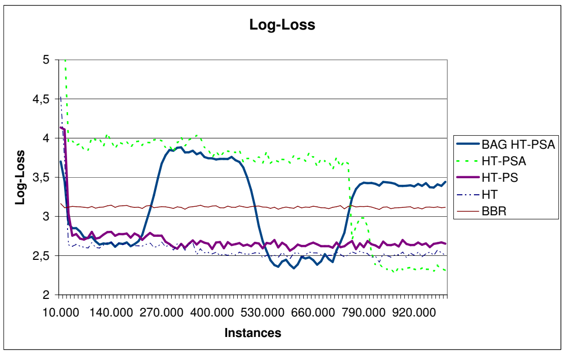

Figure 7: Log-loss on SYNT with three concept drifts.

5.5 Discussion

`HT-PS` and `HT-PSA` performed well in different situations. As evident from Tables 5 and 7, `HT-PSA` is the stronger method for adapting to concept drift because of the `ADWIN` change-monitor.

`BAG HT-PSA` performs the best overall, particularly under log loss. Strong performance was expected from a bagging

	HT	HT-PS	BBR	HT-PSA	BAG HT-PSA
RTS	0.52 ± 0.01	0.54 ± 0.00	0.24 ± 0.01	0.42 ± 0.01	**0.60** ± 0.07
RRBFS	0.33 ± 0.00	0.42 ± 0.00	0.48 ± 0.01	0.34 ± 0.01	**0.49** ± 0.03
RTC	**0.16** ± 0.03	0.08 ± 0.01	DNF	0.12 ± 0.03	0.12 ± 0.00
RRBFC	**0.16** ± 0.02	0.15 ± 0.02	DNF	0.14 ± 0.03	0.08 ± 0.00
RRBFS 0.0001	0.50 ± 0.05	0.51 ± 0.05	0.22 ± 0.01	0.58 ± 0.04	**0.69** ± 0.03
RRBFS 0.001	0.55 ± 0.04	0.56 ± 0.05	0.22 ± 0.00	0.58 ± 0.04	**0.69** ± 0.04
RRBFC 0.0001	0.15 ± 0.01	0.14 ± 0.01	DNF	0.14 ± 0.02	**0.20** ± 0.00
RRBFC 0.001	**0.15** ± 0.03	0.14 ± 0.03	DNF	0.13 ± 0.03	0.10 ± 0.00
IMDB	**0.20**	0.10	DNF	0.15	0.15
MMILL	**0.31**	0.23	0.16	0.05	0.21
TMC2007	0.17	**0.43**	0.39	0.26	0.24

Table 5: Comparison of methods. Subset accuracy is measured over the one million prequential evaluation. The best individual accuracies are indicated in boldface.

	HT	HT-PS	BBR	HT-PSA	BAG HT-PSA
RTS	2.61 ± 0.04	2.72 ± 0.02	3.12 ± 0.00	3.89 ± 0.08	**2.32** ± 0.18
RRBFS	3.53 ± 0.02	3.37 ± 0.02	**2.57** ± 0.01	3.92 ± 0.12	2.61 ± 0.10
RTC	14.59 ± 0.70	16.24 ± 0.59	DNF	14.74 ± 0.57	**8.41** ± 0.00
RRBFC	14.10 ± 0.40	13.39 ± 0.26	DNF	13.65 ± 0.45	**8.00** ± 0.00
RRBFS 0.0001	2.81 ± 0.30	2.91 ± 0.31	3.15 ± 0.00	2.48 ± 0.26	**1.96** ± 0.07
RRBFS 0.001	2.47 ± 0.25	2.66 ± 0.28	3.14 ± 0.00	2.52 ± 0.26	**1.95** ± 0.11
RRBFC 0.0001	14.27 ± 0.36	14.77 ± 0.24	DNF	13.64 ± 0.23	**7.64** ± 0.00
RRBFC 0.001	14.84 ± 0.61	14.73 ± 0.62	DNF	13.85 ± 0.51	**8.02** ± 0.00
IMDB	7.62	10.33	NF	10.39	**6.08**
MMILL	19.61	25.22	14.18	20.45	**13.52**
TMC2007	7.89	6.41	**4.90**	8.09	5.14

Table 6: Comparison of methods. LogLoss is measured as the final average over one million examples using prequential evaluation. The best individual accuracies are indicated in boldface.

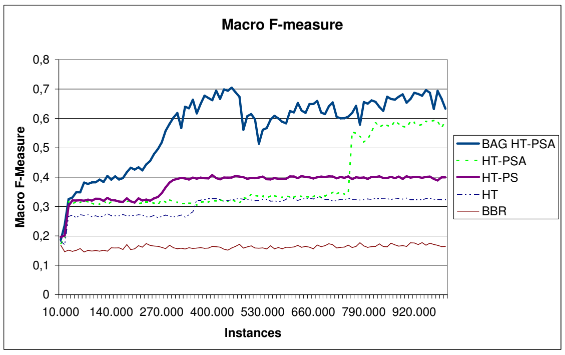

Figure 8: Macro F1-measure on SYNT with three concept drifts.

scheme, since ensembles are well known for increasing the performance of base models.

BBR ran into time and memory complexity issues. As a separate Hoeffding tree is needed for each single label in each bag. Surprisingly, BBR is not at all competitive overall, although is competitive in log loss in some situations, likely due to conservative prediction.

6. CONCLUSIONS AND FUTURE WORK

Table 4: A sample of multi-label datasets.

	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{L}\rvert$	$\lvert\mathcal{X}\rvert$	AVG($\lvert S\rvert$)
Scene	2407	6	$294n$	1.07
Yeast	2417	14	$103n$	4.24
TMC2007	28596	22	$500b$	2.16
MediaMill	43907	101	$120n$	4.38
IMDB	95424	28	$1001b$	1.92
Synthetic	1E6	{8,30}	{30,100}	{1.5,5.0}

n indicates numeric attributes, and b boolean.

We have presented an experimental framework for multi-label data stream classification, to help performing new experimental multi-label data stream benchmarks. The new methods we presented combine state-of-the-art approaches in both data stream and multi-label classification: multi-label Hoeffding trees with PS classifiers at the leaves, and additionally in an ADWIN-bagging ensemble framework.

We carried out an in depth experimental evaluation on both real and synthetic datasets using three multi-label evaluation measures, as well as measuring time and memory. We obtain satisfying results for the single multi-label Hoeffding tree, both in terms of runtime and memory consumption, and even better results under a bagging scheme which is able to adapt to concept drift.

As future work, we would like to build new incremental PS methods, and experiment with multi-label ensemble methods based on boosting.

7. REFERENCES

[1] A. Bifet and R. Gavaldà. Mining adaptively frequent

	HT	HT-PS	BBR	HT-PSA	BAG HT-PSA
RTS	0.30 ± 0.02	0.37 ± 0.01	0.17 ± 0.01	0.32 ± 0.01	**0.59** ± 0.08
RRBFS	0.25 ± 0.00	0.35 ± 0.00	0.32 ± 0.01	0.34 ± 0.02	**0.46** ± 0.02
RTC	0.07 ± 0.03	0.05 ± 0.00	DNF	0.11 ± 0.02	**0.13** ± 0.00
RRBFC	0.05 ± 0.02	**0.15** ± 0.02	DNF	0.12 ± 0.02	0.12 ± 0.00
RRBFS 0.0001	0.36 ± 0.04	0.40 ± 0.03	0.16 ± 0.01	0.55 ± 0.04	**0.69** ± 0.04
RRBFS 0.001	0.36 ± 0.03	0.43 ± 0.04	0.15 ± 0.01	0.49 ± 0.03	**0.70** ± 0.05
RRBFC 0.0001	0.06 ± 0.02	0.09 ± 0.01	DNF	0.12 ± 0.01	**0.20** ± 0.00
RRBFC 0.001	0.06 ± 0.03	0.09 ± 0.02	DNF	0.12 ± 0.03	**0.13** ± 0.00
IMDB	0.06	**0.08**	DNF	0.08	0.07
MMILL	0.03	**0.07**	0.03	0.03	0.06
TMC2007	0.08	**0.38**	0.25	0.16	0.17

Table 7: Comparison of methods. Macro F1-Measure is measured as the final average over one million examples using prequential evaluation. The best individual accuracies are indicated in boldface.

	HT	HT-PS	BBR	HT-PSA	BAG HT-PSA
RTS	**4.51**	31.78	106.37	21.37	227.01
RRBFS	**0.96**	29.20	48.71	6.19	81.27
RTC	36.20	30.92	DNF	**36.15**	533.10
RRBFC	**9.46**	32.14	DNF	19.13	237.32
RRBFS 0.0001	**1.20**	31.25	48.19	6.87	86.60
RRBFS 0.001	**1.19**	31.63	49.30	7.07	85.10
RRBFC 0.0001	**11.79**	31.22	DNF	19.87	187.32
RRBFC 0.001	**12.20**	31.73	DNF	21.43	179.09
IMDB	**23.38**	142.59	DNF	24.64	179.65
MMILL	7.99	10.94	260.88	**2.37**	19.69
TMC2007	1.54	5.33	237.83	**1.44**	13.28

Table 8: Comparison of methods. Memory in megabytes.

	HT	HT-PS	BBR	HT-PSA	BAG HTPSA
RTS	**1.24**	13.97	133.55	5.12	40.73
RRBFS	**35.07**	239.73	209.08	56.39	244.76
RTC	**9.23**	745.94	DNF	115.84	1134.59
RRBFC	**324.75**	1758.66	DNF	537.22	4693.80
RRBFS 0.0001	258.71	360.19	**96.61**	359.41	4123.85
RRBFS 0.001	**8.30**	48.52	204.46	19.76	213.31
RRBFC 0.0001	**9.08**	965.32	DNF	125.16	1291.70
RRBFC 0.001	**273.99**	1228.54	DNF	472.93	5093.07
IMDB	**263.59**	2070.93	DNF	410.15	4927.07
MMILL	**139.43**	302.65	2552.27	226.90	2989.63
TMC2007	**64.56**	194.41	1176.92	86.27	829.46

Table 9: Comparison of methods. Time in seconds.

closed unlabeled rooted trees in data streams. In *KDD '08*, 2008.

[2] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà. New ensemble methods for evolving data streams. In *KDD '09*. ACM, 2009.

[3] L. Cai. *Multilabel Classification over Category Taxonomies.* PhD thesis, Department of Computer Science, Brown University, May 2008.

[4] W. Cheng and E. Hüllermeier. Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, 76(2-3):211–225, 2009.

[5] A. Clare and R. D. King. Knowledge discovery in multi-label phenotype data. In *PKDD '01*, pages 42–53, 2001.

[6] P. Domingos and G. Hulten. Mining high-speed data streams. In *KDD '00*, pages 71–80, 2000.

[7] J. Fürnkranz, E. Hüllermeier, E. Loza Mencía, and K. Brinker. Multilabel classification via calibrated label ranking. *Machine Learning*, 73(2):133–153, November 2008.

[8] J. Gama, R. Sebastião, and P. P. Rodrigues. Issues in evaluation of stream learning algorithms. In *KDD '09*, pages 329–338, 2009.

[9] S. Godbole and S. Sarawagi. Discriminative methods for multi-labeled classification. In *PAKDD '04*, pages 22–30. Springer, 2004.

[10] G. Holmes, R. Kirkby, and B. Pfahringer. MOA: Massive Online Analysis. `http://www.cs.waikato.ac.nz/ abifet/moa/`. 2007.

[11] N. Oza and S. Russell. Online bagging and boosting. In *Artificial Intelligence and Statistics 2001*, pages 105–112. Morgan Kaufmann, 2001.

[12] S.-H. Park and J. Fürnkranz. Multi-label classification with label constraints. Technical report, Knowledge Engineering Group, TU Darmstadt, 2008.

[13] W. Qu, Y. Zhang, J. Zhu, and Q. Qiu. Mining multi-label concept-drifting data streams using dynamic classifier ensemble. In *ACML*, 2009.

[14] J. Read, B. Pfahringer, and G. Holmes. Multi-label classification using ensembles of pruned sets. In *ICDM'08*, pages 995–1000. IEEE, 2008.

[15] J. Read, B. Pfahringer, and G. Holmes. Generating synthetic multi-label data streams. In *MLD '09*, September 2009.

[16] J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. In *ECML '09*, pages 254–269. Springer-Verlag, 2009.

[17] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, December 1999.

[18] G. Tsoumakas and I. Katakis. Multi label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3), 2007.

[19] G. Tsoumakas and I. P. Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In *ECML '07*, pages 406–417. Springer-Verlag, 2007.

[20] R. Yan, J. Tesic, and J. R. Smith. Model-shared subspace boosting for multi-label classification. In *KDD '07*, pages 834–843. ACM, 2007.

[21] M.-L. Zhang and Z.-H. Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recogn.*, 40(7):2038–2048, 2007.