GOOWE-ML: A NOVEL ONLINE STACKED ENSEMBLE FOR MULTI-LABEL CLASSIFICATION IN DATA STREAMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

By
Alican Büyükçakır
July 2019

GOOWE-ML: A NOVEL ONLINE STACKED ENSEMBLE FOR MULTI-LABEL CLASSIFICATION IN DATA STREAMS

By Alican Büyükçakır

July 2019

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Fazlı Can(Advisor)

Selim Aksoy

İsmail Sengör Altıngövde

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

ABSTRACT

GOOWE-ML: A NOVEL ONLINE STACKED ENSEMBLE FOR MULTI-LABEL CLASSIFICATION IN DATA STREAMS

Alican Büyükçakır
M.S. in Computer Engineering
Advisor: Fazlı Can
July 2019

As data streams become more prevalent, the necessity for online algorithms that mine this transient and dynamic data becomes clearer. Multi-label data stream classification is a supervised learning problem where each instance in the data stream is classified into one or more pre-defined sets of labels. Many methods have been proposed to tackle this problem, including but not limited to ensemble-based methods. Some of these ensemble-based methods are specifically designed to work with certain multi-label base classifiers; some others employ online bagging schemes to build their ensembles. In this study, we introduce a novel online and dynamically-weighted stacked ensemble for multi-label classification, called GOOWE-ML, that utilizes spatial modeling to assign optimal weights to its component classifiers. Our model can be used with any existing incremental multi-label classification algorithm as its base classifier. We conduct experiments with 4 GOOWE-ML-based multi-label ensembles and 7 baseline models on 7 real-world datasets from diverse areas of interest. Our experiments show that GOOWE-ML ensembles yield consistently better results in terms of predictive performance in almost all of the datasets, with respect to the other prominent ensemble models.

Keywords: Multi-label, data stream, supervised learning, classification, ensemble learning, stacking.

ÖZET

GOOWE-ML: VERİ AKIŞLARINDA ÇOK-ETİKETLİ SINIFLANDIRMA İÇİN YENİ BİR ÜST-ÖĞRENİCİLİ ÇOKLU-SINIFLANDIRICI

Alican Büyükçakır
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Danışmanı: Fazlı Can
Temmuz 2019

Veri akışları yaygınlaştıkça, bu geçişken ve dinamik verilerin madenciliği için çevrimiçi algoritmalara gereksinim gittikçe daha belirgin hale gelmektedir. Çok-etiketli veri akışı sınıflandırması, veri akışındaki her bir veri örneğinin etiket kümesindeki bir ya da birden fazla etiketle sınıflandırıldığı bir gözetimli sınıflandırma problemidir. Bu problemin çözümü için içinde çoklu-sınıflandırıcıların da bulunduğu birçok yöntem geliştirilip öne sürülmüştür. Bu çoklu-sınıflandırıcılardan bazıları yalnızca belirli birtakım çok-etiketli temel sınıflandırıcılarla çalışabilecek şekilde tasarlanmış, diğerleri ise çevrimiçi *bagging* gibi yöntemlerle çoklu-sınıflandırıcılarını meydana getiren sınıflandırıcılarını seçmiştir. Bu çalışmada, çok-etiketli sınıflandırma problemi için GOOWE-ML adında yeni bir çevrimiçi, dinamik-ağırlıklı bir çoklu-sınıflandırıcı sunulmuştur. GOOWE-ML, uzamsal modelleme kullanarak içindeki sınıflandırıcılara en-iyileştirilmiş (optimal) ağırlıklar atayabilmektedir ve artımlı herhangi bir çok-etiketli sınıflandırıcıyı kendisi için bir temel sınıflandırıcı olarak kullanılabilecek niteliktedir. Bu çalışmada, 4 adet GOOWE-ML-bazlı çoklu-sınıflandırıcı ile, 7 adet rakip modele karşı çeşitli alanlardan 7 veri kümesi üzerinde deneyler yapılmıştır. Bu deneyler, GOOWE-ML-bazlı çoklu-sınıflandırıcıların neredeyse tüm veri kümelerinde, tahmin performansı bakımından rakip çoklu-sınıflandırıcılardan istikrarlı bir biçimde daha iyi sonuçlar verdiğini göstermektedir.

Anahtar sözcükler: çok-etiketli, veri akışı, gözetimli öğrenme, sınıflandırma, çoklu-sınıflandırıcılar, üst-öğrenici.

Acknowledgement

First and foremost, I would like to thank my advisor Prof. Fazlı Can for his trust in me, never-ending enthusiasm, continuous support and invaluable contributions to this thesis, and my personal and professional development. Doing research can be overwhelming from time to time. Yet, he made this difficult process bearable, if not enjoyable for me. Besides my advisor, I would like to thank the rest of my thesis committee, Assoc. Prof. Selim Aksoy and Assoc. Prof. İ. Sengör Altıngövde, for their valuable feedbacks.

I must express my gratitude to my beloved, Ezgi, who comforted me, cheered me up and supported me when I feel down, distressed or anxious. She was always by my side, and helped me grow mentally, spiritually, and emotionally.

In addition, very special thanks to my office mates at EA507 and fellows at BilIR (Bilkent Information Retrieval Group). Thanks to you, the office felt like home, and I had a reason to commute to the office almost everyday.

I would like to thank TÜBİTAK, since I was financially supported by TÜBİTAK's 2211 Domestic Graduate Scholarship Program (2211 Yurt İçi Lisansüstü Burs Programı) throughout the two years of my master's degree. Also, I would like to thank Bilkent University Computer Engineering Department for their financial support on my accommodation, my conference travel to Turin for CIKM 2018, and everything else they have done for me. Here, our department secretary, Ebru Ateş, deserves a special mention.

Finally, I must express my sincerest and most profound gratitude to my parents, Hatice and Şenol, and my brother, Eşref. I am forever indebted to them for their incredible love, kindness and company. Without their sacrifices, I would not be who I am today.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

An earlier version of this thesis is published as a conference paper [1] in ACM CIKM 2018. The title of the thesis, *"GOOWE-ML: A Novel Online Stacked Ensemble for Multi-label Classification in Data Streams,"* indicates that the presented study involves the combination of different approaches and learning paradigms, namely: (1) data streams, (2) multi-label learning, and (3) ensemble learning. In this section, each of these are briefly discussed to provide preliminaries for the proposed method.

1.1 Data Streams

Data streams are possibly infinite sequences of data that continuously and rapidly grow over time [2]. Over the past decade, data stream mining has become one of the most prevalent and fruitful subfields of data mining, mostly due to the ever-increasing number of data stream generators in our lives such as sensors, mobile phones, IoT devices and so on. However, data stream mining is a challenging task due to the complexities posed by The Three Vs of Big Data: Volume, Variety and Velocity [3]. *Volume* is the sheer amount of data, *Velocity* is the speed of which data is generated by its sources, and *Variety* refers to differences / variation in

the types of data (e.g. sensor data, text, image, audio or video)[1].

On top of these, there is a temporal dimensionality of data streams as well, i.e. data distribution from which the data stream generates its instances may not be stationary but dynamic / evolving. In such cases, changes in the distribution of the data are called concept drifts [7]. In such dynamic environments, the concepts that are learned by models can become obsolete over time. This causes models to misclassify the instances from the new concept, and deteriorates their predictive performance [8].

Considering difficulties that are discussed above, designing a learning system for data stream mining requires the following conditions [9]:

1. The learning system can see and process a data instance once and only once. Since the flow of the stream is fast, the system needs to be done with a given instance quickly and proceed to the next one.

2. Memory that is utilized by the learning system cannot grow indefinitely with the possibly-infinite stream of data. There has to be a memory constraint.

3. The learning system should be robust against concept drifts, i.e. it should be able to capture changes in the distribution of the data over time and adapt itself accordingly.

4. The learning system should be ready to respond and give prediction results of a given query at any time.

[1]Even though the aforementioned 3Vs are widely accepted, there is no general consensus on the number of Vs of Big Data. It varies across articles and contexts. Some of the other Vs are as follows. *Veracity* [4], indicating the reliability / trustworthiness of data which may be relevant for social network mining and systems that are prone to bias, noise and other contingencies. *Value* [5], indicating importance of collected data and the value generated from it. Variability [6], referring to the expansion in the ranges of values of collected data.

1.2 Multi-label Learning Paradigm

The traditional supervised learning task is single-label, i.e. a data instance is classified into one label λ among a disjoint set of labels $\mathcal{L}$. However, this may not be the case for some real-world data. For instance, *The Big Lebowski* can simultaneously be classified as a *crime*, *comedy*, and *cult* movie. In such settings where an instance can be classified into a subset of labels, $L^* \subseteq \mathcal{L}$, the learning paradigm is called Multi-label Learning (MLL).

As of 2017, it is estimated that around 4.9 billion connected devices are generating data and this number is expected to rise to 25 billion by 2020 [10]. With such a rate of increase in the number of data in the form of streams, it becomes more and more important to extract meaningful information from seemingly chaotic data. Some of these data streams are multi-label, which led to MLL algorithms that can cope with streaming settings (i.e. with time and memory constraints, as well as changes in the distribution of the data over time) being developed.

MLL algorithms have drawn considerable attention over the last decades by accomplishing strong results in diverse areas including bioinformatics [11, 12], text classification [13] and image scene classification [14].

1.3 Ensemble Learning

Clearly, learning systems that handle such complexities are required to have robustness / resilience against the inherent dynamism involved in the task. That is one of the reasons why the proposed models are often times ensemble models. Ensembles in data streams are studied extensively in the literature [15, 16] and preferred over single classifiers in dynamic environments thanks to their adaptive nature. Ensembles perform consistently better than their individual component classifiers, even though the ensembling techniques (whether bagging [17], boosting [18] or stacking [19]), ensemble maintenance strategies and vote combination

schemes differ among them. Different classifiers in the ensemble focus on learning different concepts that exist in the data. Some ensembles may create their components in a temporal order [20, 21], so that more recent concepts in the stream are captured by more recent classifiers in the ensemble. Some others may remove old or useless classifiers from the ensemble to unlearn the previously acquired concepts [22]. These kind of dynamic mechanisms of ensembles makes them preferable for multi-label stream classification task as well.

1.4 Contributions of this Work

A considerable number of MLL algorithms resort to ensemble methods to increase their predictive performances [23, 24, 25, 26]. However, these methods usually employ online bagging schemes for ensemble construction (where some of them utilized change detection mechanisms [27] as an upgrade to these bagging-based ensembles). To the best of our knowledge, there are a very few stacked ensembles for multi-label stream classification, and most of the stacked ensembles in the literature are designed for and can only work with specific types of MLL algorithms. In this paper, we propose a novel stacked ensemble that is agnostic of the type of multi-label classifier that is used within the ensemble.

The main contributions of this thesis are as follows: We

- introduce a batch-incremental, online stacked ensemble for multi-label stream classification, GOOWE-ML, that can work with any incremental multi-label classifier as its component classifiers,
- construct an $|\mathcal{L}|$ dimensional space to represent the relevance scores of classifiers of the ensemble, and utilize this construction to assign optimum weights to the model's component classifiers,
- conduct experiments on 7 real-world datasets to compare GOOWE-ML with 7 state-of-the-art ensemble methods, and apply statistical tests to show that our model outperforms the state-of-the-art multi-label ensemble models.

- Additionally, we discuss how and why some multi-label classifiers yield poor Hamming Scores while performing considerably well on the rest of the performance metrics (e.g. accuracy, F1 Score).

All in all, we argue that GOOWE-ML is well-suited for the multi-label stream classification task, and it is a valuable addition to the present day models.

The rest of the thesis is organized as follows: Chapter 2 defines the problem of multi-label stream classification and gives preliminaries. Chapter 3 introduces the most widely used multi-label algorithms and ensemble techniques in the literature. In Chapter 4, our ensemble, GOOWE-ML, is described. Experimental setup, evaluation metrics and datasets are given in the first half of Chapter 5, and the results are presented and discussed in its second half. Lastly, the thesis is concluded with insights and possible future work in Chapter 6.

Chapter 2

Problem Definition and Notation

MLL is considered to be a hard task by nature, as the output space increases exponentially with the number of labels, since there are 2^L possible outcomes of classification for the labelset of size L. The high dimensionality of the label space causes increased computational cost, execution time and memory consumption. *Multi-label stream classification* (MLSC [26]) is the version of this task that takes place on data streams. See Figure 2.1 for a multi-label learner in a stream environment. The two time steps that are depicted in that figure shows (1) how training and testing are done in data streams, and (2) why the learner needs to adapt the concept drifts.

According to this definition of the problem, below are listed all the notation that are used throughout this work.

Data stream $\mathcal{D}$ is the set of data that has a temporal dimension and is possibly infinite. $\mathcal{D} = d_0, d_1, .., d_t, .., d_N$ where d_t is the data point in time t and d_N is the lastly seen data point in the data stream. The knowledge of lastly seen data point d_N is not known a priori, and it is only there to indicate the end of the processed data instances for evaluation purposes. Each data point d_t is of form $d_t = (x, y)$ where x is a data instance and y is its labelset (label relevance vector). The data instance x is a vector represented as $x =< x_1, x_2, .., x_i, .., x_M >$, and

Table 2.1: Symbols and Notation for Multi-Label Stream Classification [1]

Symbol	**Meaning**
M	Number of attributes in a data instance
L	Number of labels in the labelset of a data instance
N	Number of instances in the data stream
$\mathcal{X}$	Input attribute space. $\mathcal{X} = \mathbb{R}^M$
x	A data instance. $x =< x_1, x_2, .., x_i, .., x_M > \in \mathcal{X}$
$\mathcal{L}$	Set of all possible labels. $\mathcal{L} = \{\lambda_1, \lambda_2, .., \lambda_L\}$
y	Label relevance vector. $y =< y_1, y_2, .., y_j, .., y_L >= \{0, 1\}^L$
$\hat{y}$	Predicted relevance vector. $\hat{y} = h(x) =< \hat{y}_1, \hat{y}_2, .., \hat{y}_j, .., \hat{y}_L >= [0, 1]^L$
$d_t = (x^t, y^t)$	The data point that arrives at time t
$\mathcal{D}$	Possibly infinite data stream. $\mathcal{D} = d_0, d_1, .., d_t, .., d_N$

each $x_i \in \mathcal{X}$. The labelset y is a vector represented as $y =< y_1, y_2, .., y_j, .., y_L >$, and each $y_j \in \{0, 1\}$. Here, $y_j = 1$ means the jth label is relevant, and 0 otherwise. A prediction (hypothesis) of a multi-label classifier is $\hat{y} = h(x)$ that is of form $\hat{y} =< \hat{y}_1, \hat{y}_2, .., \hat{y}_j, .., \hat{y}_L >$ and $\hat{y} \in [0, 1]^L$ meaning that the prediction vector consists of relevance probabilities (relevance scores) for each label. For the final decision of classification and evaluation, the prediction vector is sent to *de-fuzzification*, typically done by thresholding the relevance scores [28].

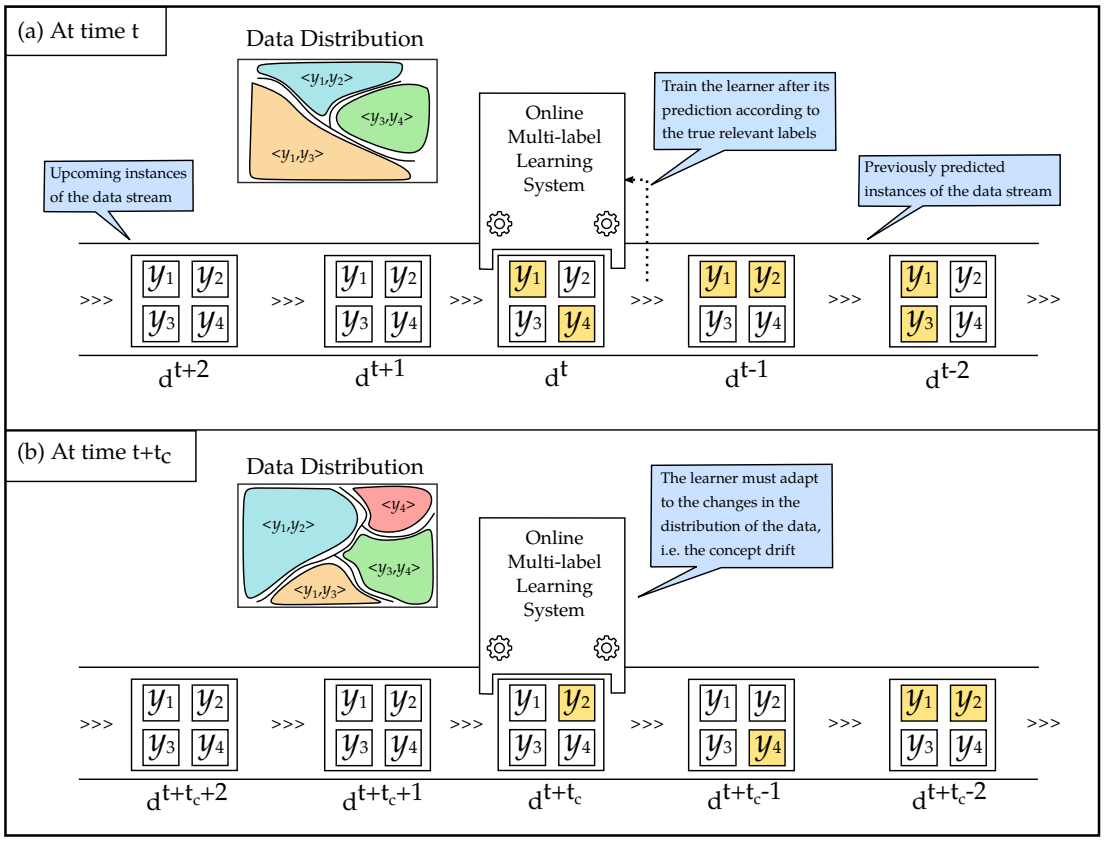

Figure 2.1: Multi-label Stream Classification. (a) A multi-label learner that performs classification on a data stream with $L = 4$ is depicted. Labels that are predicted as relevant in the past by the learner are filled with yellow. The learner is trained using interleaved-test-then-train approach (for details, see Section 5.1.3). (b) t_c units of time later, a concept drift happens. Now, the learner must modify itself according to the changes in the distribution of the data.

Chapter 3

Related Work

Comprehensive reviews on multi-label learning can be found in [28, 29, 30], on ensemble learning for data streams in [16, 31] and ensemble of multi-label classifiers in [15]. In this thesis, we discuss the state-of-the-art multi-label classification methods, and focus on how these methods are used in ensemble learners for data streams.

3.1 Multi-label Methods

As a widely accepted taxonomy in the field of MLL, there are two general methods [29] of tackling a multi-label classification problem:

3.1.1 Problem Transformation

In *Problem Transformation*, the multi-label problem is transformed into more well-understood and simpler problems.

The most widely used Problem Transformation method is the Binary Relevance (BR) [32] where the multi-label problem is transformed into $|\mathcal{L}|$ distinct

binary classification problems. After the transformation is applied to the dataset, any off-the-shelf binary classification algorithm can be utilized to get individual outputs corresponding to each binary problem. It scales linearly with respect to the number of labels, which makes it an efficient choice for practical purposes. However, it is discussed [29, 23] that BR inherently fails to capture label-wise interdependencies.

To capture dependencies among labels and overcome this weakness of BR, some other BR-based methods are developed; most notably Classifier Chains (CC) [23] where BR classifiers are randomly permuted and linked in a chain-like manner in which each BR classifier yields its output to its connected neighbor classifier as an attribute. It is claimed that this helps the classifiers to capture the label dependencies, as each classifier in the chain learns not only the data itself, but also the label associations of every previous classifier in the chain. Derivatives of this method are generated by modifying the underlying structure of its information feeding network among the classifiers (Classifier Trellises (CT) [33]), by introducing Bayesian Risk Minimization (Probabilistic Classifier Chains (PCC) [34]), or by utilizing Monte Carlo methods to find a good chain (Monte Carlo Classifier Chains (MCC) [35]).

Another common method of Problem Transformation is Label Powerset (LP) [32] method where each possible subset of labels is treated as a single label to translate the initial problem into single-label classification task with a bigger set of labels (hence, having multi-class problem of size $2^{|\mathcal{L}|}$). Pruned Sets (PS) [24] is an LP-based technique where the instances with infrequent label sets are pruned from the dataset. This allows only the instances with the most important subsets of labels to be considered for classification. Afterwards, the pruned instances are recycled back into an auxiliary dataset for another phase of classification; but for every subset of their relevant labels, instead of their initial relevant labels.

3.1.2 Algorithm Adaptation

In *Algorithm Adaptation*, existing classification algorithms (such as decision trees, nearest neighbor classifiers and so on) are modified to be compatible with the multi-label setting.

In ML-KNN [36], the k-Nearest Neighbor algorithm is modified by counting the number of relevant labels for each neighboring instance to acquire posterior relevance probabilities for labels.

In ML-DT [11], the split criterion of C4.5 decision trees is modified by introducing the concept of multi-label entropy. In streaming environments, however, Hoeffding Trees are the common choice for decision trees. Hoeffding Trees [37] are incremental decision trees that have a theoretical guarantee that their output will become asymptotically identical to that of a regular decision tree as more and more data instances arrive. Modifying the split criterion of Hoeffding Trees for multi-label entropy, Multi-label Hoeffding Trees [38] are developed. More recently, a novel decision tree based method, iSOUP-Trees (incremental Structured OUtput Prediction Tree) [39] are proposed where adaptive perceptrons are placed in the leaves of incremental trees and the perceptrons' weights are used in producing a prediction that is a linear combination of the input's attributes.

In a nutshell, "in Problem Transformation, data is modified to make it suitable for algorithms; whereas in Algorithm Adaptation, algorithms are modified to make them suitable for data" [29].

3.2 Ensembles in MLL and MLSC

One of the most commonly used ensemble methods is Bagging where each classifier in an ensemble is trained with a bootstrap sample (a data sample that has the same size with the dataset, but each data point is randomly drawn with replacement). This assumes that the whole dataset is available, which is not the

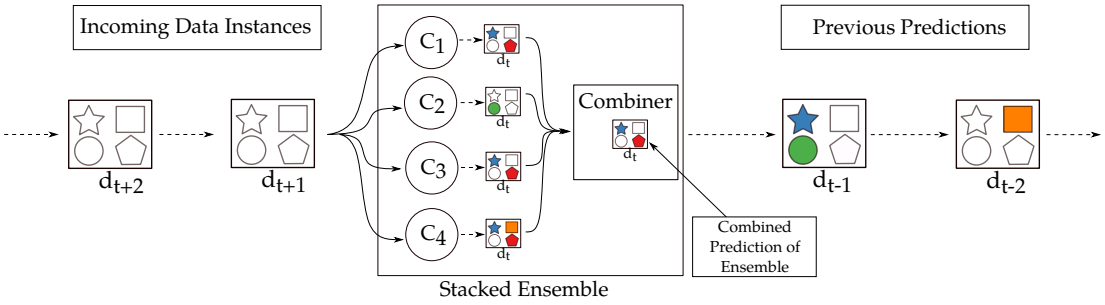

Figure 3.1: A stacked multi-label ensemble for stream classification. For each data instance, associated labels are shown with the geometric shapes (□, ◯, and so on). A shape is colored if that label is relevant. Component classifiers (C_1, C_2, C_3, C_4) generate their own predictions, and these predictions are combined by the combiner algorithm of the ensemble [1].

case for data stream environments. However, observing that the probability of having K many of a certain data point in a bootstrap sample is approximately `Poisson(1)` for big datasets, each incoming data instance in a data stream can be weighted proportional to `Poisson(1)` distribution to mimic bootstrapping in an online setting [40]. This is called Online Bagging, or OzaBagging, and it has been widely used in MLSC. In fact, the phrase *'Ensemble of'* in the field usually means that it is the OzaBagged version of the base classifier that is mentioned. EBR [23], ECC [23], EPS [24] and EBRT [39] (Ensembles of BR, CC, PS and iSOUP Regression Trees respectively) are examples of this convention.

Additionally, it is common for the ensembles that use OzaBagging to also use a concept change detection mechanism called ADWIN (Adaptive Windowing) [27]. ADWIN keeps a variable-length window of the most recent items in the data stream to detect diversions from the average of some statistics on the window. Therefore, whenever ADWIN detects a change, the worst classifier in the OzaBag is reset. This is called ADWIN Bagging [38].

To the best of our knowledge, stacked ensemble models in the field of MLSC are very rare. A general scheme for a stacked ensemble for MLSC is given in Figure 3.1. Predictions of the component classifiers of an ensemble should be combined by a function (a meta-classifier) which will generate the final prediction of the ensemble. One can use either raw confidence scores of the labels for each instance, or predictions for each label and their counts (majority voting scheme) as the

contributions of each component. How to optimally combine the contributions of each classifier is still a question in MLSC.

Stacked ensembles that are proposed in the field are as follows:

SWMEC [26] is a weighted ensemble that is designed for ML-KNN as its base classifier. Its weight adjustment scheme utilizes distances in ML-KNN to obtain a confidence coefficient. IBR (Improved BR) [25] employs a feature extension mechanism in which the outputs of a BR classifier is firstly weighted by the accuracy of that classifier, and then added as a new feature to the data instance. New BR classifiers are trained from the data with extended feature spaces. Multiple Windows (MW) [41] is another extension to BR where two sliding windows are used instead of one, and the relevant and the non-relevant instances are evaluated in different windows. This allows MW to handle class imbalance, too. SMART [42] is an ensemble of random trees where each tree node collects statistics about estimated relevance of each label and estimated number of relevant labels (label cardinality) for each instance. The functionalities of these models involve algorithm-specific properties and therefore cannot be extended to any other base classifier. Such models are constrained by the success of their base classifiers.

In [43], the authors followed an unorthodox approach and created a *label-based* ensemble instead of a chunk-based one, which tackled the class imbalance problem that exists in the multi-label datasets as well as concept drifts. Recently, in ML-AMRules [44], multi-label classification task is interpreted as a rule learning task and the rule learners are combined in an ensemble that uses online bagging (called ML-Random Rules).

All in all, ensemble models in MLSC are not explored thoroughly. Base multi-label classifiers are either combined with Online Bagging or ADWIN Bagging, or with stacked combination schemes that depends on the type of the base classifier. There is a lack of online ensembles in the field that can work with any type of multi-label base classifier which also involve a smart combination scheme. Our method, GOOWE-ML (described in Chapter 4), addresses this inadequacy.

Chapter 4

Proposed Method: GOOWE-ML

We propose GOOWE-ML (**G**eometrically **O**ptimum **O**nline **W**eighted **E**nsemble for **M**ulti-**L**abel Classification): a batch-incremental (chunk-based) and dynamically-weighted online ensemble that can be used with any incremental multi-label learner that yields confidence outputs for predicting relevant labels for an incoming data instance.

Let the multi-label classifiers in the ensemble be $\{C_1, C_2, \ldots, C_K\}$. For each incoming data instance, each classifier C_k generates relevance score vector s_k, which consists of the relevance scores of each label for that instance, i.e. $s_k =< S_{k1}, S_{k2}, \ldots, S_{kL} >$. The relevance score vectors of classifiers for each instance is stored in the rows of matrix S, which will be used to populate elements of the matrix A and the vector d (see Eqn.4.4 and 4.5, and Alg.2:7-8).

4.1 Ensemble Maintenance

Let ξ denote the ensemble that is initially empty. A new classifier is trained at each incoming data chunk, as well as the existing ones (if any). The ensemble grows as the new classifiers from incoming data chunks are introduced, until the maximum ensemble size is reached (i.e. ensemble is full). Then, the newly trained

Table 4.1: Additional Symbols and Notation for GOOWE-ML [1]

Symbol	Meaning
K	Number of component classifiers in the ensemble, i.e. ensemble size
h	Number of data points in the instance window I
n	Maximum capacity of a data chunk DC
C_k	kth component classifier in the ensemble. $1 \leq k \leq K$
ξ	Ensemble of classifiers. $\xi = \{C_1, C_2, .., C_k, .., C_K\}$
w	Weight vector for the ensemble ξ. $w =< W_1, W_2, .., W_k, .., W_K >$
s_k^i	Relevance scores for the kth classifier for ith instance in the ensemble. $s_k^i =< S_{k1}^i, S_{k2}^i, .., S_{kj}^i, .., S_{kL}^i >$
S	Relevance scores matrix. Each relevance score s_{kj} is an element in this matrix. $S \in \mathbb{R}^{K \times L}$
I	Instance window of size n, having latest n data instances. $I = d_1, d_2, .., d_h$
DC	Data chunk that consists of the latest h data points. $DC = d_1, d_2, .., d_n$

classifier replaces one of the old classifiers in the ensemble. This replacement is often times done by removing the temporally oldest component or the most poorly performed component with respect to some metric [45]. In GOOWE-ML, this replacement is done by re-weighting the component classifiers and removing the component with the lowest weight (Alg 1:7-12). Analogous model management systems are employed in both ensembles for single-label classification such as Accuracy Weighted Ensemble (AWE) [46] and Accuracy Updated Ensemble (AUE2) [21]; and for multi-label classification such as SWMEC [26].

Having a fixed number of base classifiers in the ensemble prevents the model to swell in terms of memory usage. Also, training new classifiers from each data chunk allows the ensemble to notice new trends in the distribution of the data, and thus, be more robust against concept drifts.

In addition to fixed-sized data chunks, GOOWE-ML also uses a sliding window for stream evaluation purposes, which consists of the most recently seen h instances. Size of the instance window can be smaller than the size of each data

chunk, i.e. $h \leq n$, so that higher resolution can be obtained for the prequential evaluation. Prequential evaluation is discussed in more detail in 5.1.2.

4.2 Weight Assignment and Update

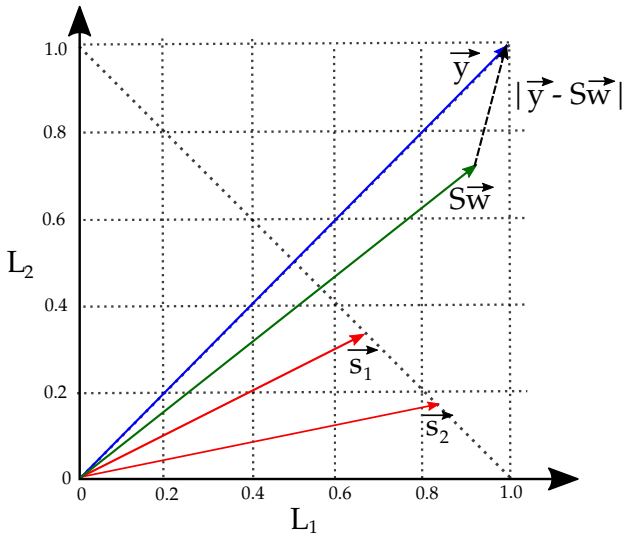

Figure 4.1: Transformation into label space in GOOWE-ML [1]. Relevance scores of the components (red): $S_1 =< 0.65, 0.35 >$ and $S_2 =< 0.82, 0.18 >$. The optimal vector $\vec{y}$ (blue): $y =< 1, 1 >$, generated from the ground truth. Weighted prediction of the ensemble: $S\vec{w}$ (green). The distance between $\vec{y}$ and $S\vec{w}$ is minimized.

In our geometric framework, we represent the relevance scores s_k of each component classifier in our ensemble as vectors in an L-dimensional space. Previously, Tai & Lin [47] used a similar approach, which they called Principal Label-Space Transformation, to interpret the existing multi-label algorithms in a geometrical setting and reduce the high dimensionality of the multi-label data. Bonab & Can [48] adapted an analogous setting to investigate the optimal ensemble size for single-label data stream classification. In GOOWE-ML, this spatial modeling is used to assign optimal weights for component classifiers in the ensemble.

Geometrically, an intuitive explanation of our spatial modeling and weighting scheme is shown in Figure 4.1 for the 2-dimensional case, i.e. when $L = 2$.

After representing relevance scores in the label space, GOOWE-ML minimizes the Euclidean distance between the combined relevance vector, $\hat{y}$, and the ideal vector that represents the ground truth, y, in the label space. Analogously, Wu & Crestani [49] utilized this approach in the field of Data Fusion to optimally combine the query results, and Bonab & Can [20] in the field of single-label data stream classification, both with successful results. This is equivalent to the following *linear least squares problem* [50].

$$\min_{\vec{w}} ||\vec{y} - S\vec{w}||_2^2 \tag{4.1}$$

Here, S is the relevance scores matrix, w is the weight vector which is to be determined, and y is the vector representing the ground truth for a given data point. In other words, our objective function to be minimized is the following:

$$f(W_1, W_2, .., W_K) = \sum_{i=1}^{n} \sum_{j=1}^{L} \left(\sum_{k=1}^{K} (W_k S_{kj}^i - y_j^i) \right)^2 \tag{4.2}$$

Taking a partial derivative of W and setting the gradient to zero, i.e. $\nabla f = 0$, we get:

$$\sum_{k=1}^{K} W_k \left(\sum_{i=1}^{n} \sum_{j=1}^{L} S_{qj}^i S_{kj}^i \right) = \sum_{i=1}^{n} \sum_{j=1}^{L} y_j^i S_{qj}^i \tag{4.3}$$

Equation 4.3 is of the form $Aw = d$ where A is a square matrix of size $K \times K$ with elements:

$$a_{qk}^i = \sum_{i=1}^{n} \sum_{j=1}^{L} S_{qj}^i S_{kj}^i \qquad (1 \leq q, k \leq K) \tag{4.4}$$

and d is the remainder vector of size K with elements:

$$d_q^i = \sum_{i=1}^{n} \sum_{j=1}^{L} y_j^i S_{qj}^i \qquad (1 \leq q \leq K) \tag{4.5}$$

Therefore, solving the equation $Aw = d$, for w, gives us the optimally adjusted weight vector. The weight vector w is updated at the end of each data chunk, where the components in the ensemble are trained from the instances in the data chunk, as well. Also, notice that this update operation resembles the Batch Gradient Descent [51] in a way that w is updated at the end of each batch, having trained from the instances in the batch. However, unlike Batch Gradient Descent, this weight update scheme does not take steps towards better weights iteratively, but rather finds the optimal weights directly after solving the linear system $Aw = d$. As a consequence, the updated weights do not depend on the previous values in w, they only depend on the performance of the components on the latest chunk. This allows the ensemble to capture sudden changes in the distribution of the data.

Here, the linear least squares solution to the system $Aw = d$ does not necessarily produce weights that are *bounded* within specified upper and lower bounds, or *all non-negative*. To overcome this, we apply min-max normalization to the resulting vector w and acquire the ensemble component weights that are all within the interval $[0, 1]$.

Instead of ordinary least squares, *non-negative least squares (NNLS)* [50] can be used to acquire all non-negative weights directly. However, it is reported that the computation of pseudoinverse step in NNLS causes slow runtime in practice [52]. Another, more complex approach that allows the weights to be found directly within specified bounds is *bounded-variable least squares (BVLS)* [52], which requires number of iterations comparable to the number of variables to work with.

4.3 Multi-label Prediction

The ensemble's prediction for the ith example, $\hat{y}^i$, is the weighted sum of its components' relevance scores, s_k.

$$\hat{y}^i_j(\xi) = \sum_{k=1}^{K} w_k S^i_{kj} \qquad (1 \leq j \leq L) \tag{4.6}$$

Here, each relevance score, S^i_{kj}, is normalized beforehand into the range of $[0, 1]$ by the following normalization [1]:

$$S^i_{kj} \leftarrow \frac{S^i_{kj}}{\sum_{j=1}^{L} S^i_{kj}} \qquad (1 \leq j \leq L) \tag{4.7}$$

After normalization, the relevance scores sum up to 1. The final prediction of the classifier is obtained by thresholding the relevance scores by $(1/L)$, which is the expected prior relevance probability of a label of a data instance.

$$\hat{y}^i_j \leftarrow \begin{cases} 1, & \text{if } \hat{y}^i_j > \dfrac{1}{L} \\ 0, & otherwise \end{cases} \qquad 1 \leq j \leq L \tag{4.8}$$

These three operations (Weighted Voting (Eqn.4.6), Normalization (Eqn.4.7) and Thresholding (Eqn.4.8)) are done consecutively and can be considered as one atomic operation in the algorithm, shown as *predict()* in the pseudocode (see Alg.1:5).

[1] We implemented other normalization methods such as Softmax for this purpose, but found out that this kind of linear normalization works better in multi-label contexts.

Algorithm 1 GOOWE-ML: Geometrically Optimum Online Weighted Ensemble for Multi-Label Classification

Require: $\mathcal{D}$: data stream, DC: latest data chunk, K: maximum number of classifiers in the ensemble, C: a multi-label classifier in the ensemble,

Ensure: ξ: ensemble of weighted classifiers, $\hat{y}$: multi-label prediction of the ensemble as combined score vector.

$\xi \leftarrow \emptyset$;
$A, d \leftarrow null, null$
while $\mathcal{D}$ has more instances **do**
  $d_i \leftarrow$ current data instance
  $\hat{y} \leftarrow$predict(d_i, ξ) {Eqn.4.6, 4.7 and 4.8}
  if DC is full **then**
    $C_{in} \leftarrow$ new component classifier built on DC;
    if ξ has K classifiers **then**
      $A', d' \leftarrow$TrainOptimumWeights$(DC, \xi, null, null)$
      $w \leftarrow$ solve$(A'w = d')$;
      $C_{out} \leftarrow$ classifier C_k with minimum w_k
      $\xi \leftarrow \xi - C_{out}$
    end if
    $\xi \leftarrow \xi \cup C_{in}$
    Train all classifiers $C \in \xi - C_{in}$ with DC
  end if
end while

Algorithm 2 GOOWE-ML: Train Optimum Weights

Require: DC: one or more data instances, ξ: Ensemble of Classifiers, A: square matrix, d: remainder vector

Ensure: The matrix A and the vector d, ready for optimum weight assignment

if A is null or d is null **then**
  Initialize square matrix A of size $K \times K$
  Initialize remainder vector d of size K
end if
for all instances $x^t \in DC$ **do**
  $y^t \leftarrow$ true relevance vector of x^t {To be used in Eqn.4.5}
  $A \leftarrow A + A^t$; {Eqn.4.4}
  $d \leftarrow d + d^t$; {Eqn.4.5}
end for

4.4 Complexity Analysis

Let the prediction of a component classifier in the ensemble take $O(c)$ time. Also, notice that the ensemble size is of order $O(K)$, since ensemble is not fully formed only for the first K chunks and the size is always K afterwards.

For a data chunk, each component classifier predicts each data instance, which takes $O(nKc)$ time, as the size of each data chunk in the stream are the same and n. At the same time, the square matrix A and the remainder vector d are filled using each pair of relevance scores of the components for each label and each instance, which takes $O(nK^2L)$ time. Then, the linear system $Aw = d$ is solved, where A is of size K. Solving this linear system with no complex optimization methods take at most $O(K^3)$ time [53] (where there are more complex but asymptotically better methods). This loop continues for N/n many chunks. Thus, the whole process has the complexity of:

$$O\left(\frac{N}{n}\Big((nKc + nK^2L) + K^3\Big)\right) = O\left(N\Big(Kc + K^2L + \frac{K^3}{n}\Big)\right) \tag{4.9}$$

Here, the term with (Kc), (K^2L) and (K^3/n) represents the time complexity of prediction, training and optimal weight assignment respectively.

c is generally small, since most of the models use Hoeffding Trees and its derivatives as their base classifiers. Therefore, the term with (K^2L) dominates the sum there in the Eqn. 4.9. When the terms (K^2L) and (K^3/n) are compared, it can be also noticed that the former always dominates the latter: n's magnitude is of hundreds or thousands, whereas K's magnitude is generally of tens. As a result, L (a whole number) will be always higher than (K/n) (a fraction that is < 1). As a consequence, the algorithm has overall $O(NK^2L)$ complexity.

Chapter 5

Experiments and Results

5.1 Experimental Design

5.1.1 Datasets

To understand how densely multi-labeled a dataset is, *Label Cardinality* and *Label Density* are used. Label Cardinality is the average number of relevant labels of the instances in $\mathcal{D}$; Label Density is the Label Cardinality per number of labels [54], indicating the percentage of labels that are relevant on average.

$$LC(\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N}|y^i| \qquad LD(\mathcal{D}) = \frac{LC(\mathcal{D})}{L} = \frac{1}{LN}\sum_{i=1}^{N}|y^i|$$

Our experiments are conducted on 7 datasets[1] that are from diverse application domains (genes, newspapers, aviation safety reports and so on), given in Table 5.1. These datasets are extensively used in the literature [39, 44, 38].

[1]Datasets are downloaded from MEKA's webpage. Available at: `https://sourceforge.net/projects/meka/files/Datasets/`.

Table 5.1: Multi-Label Datasets [1]. The superscripts after the name of the dataset indicates that the features in that dataset is binary (^{b}) or numeric (^{n}).

Source $\mathcal{D}$	**Domain**	**N**	**M**	**L**	**LC($\mathcal{D}$)**	**LD($\mathcal{D}$)**
20NG ^{b}	Text	19,300	1,006	20	1.020	0.051
Yeast ^{n}	Biology	2,417	103	14	4.237	0.303
Ohsumed ^{b}	Text	13,529	1,002	23	1.660	0.072
Slashdot ^{b}	Text	3,782	1,079	22	1.180	0.053
Reuters ^{n}	Text	6,000	500	101	2.880	0.028
IMDB ^{b}	Text	120,919	1,001	28	2.000	0.071
TMC2007 ^{b}	Text	28,596	500	22	2.160	0.098

5.1.2 Evaluating Multi-label Learners

Multi-label evaluation metrics that are widely used throughout the studies in the field are divided into two groups [29]: (1) *Instance-Based Metrics*, (2) *Label-Based Metrics*. These two metrics indicate how well the algorithms perform. In addition to these, efficiency of the performing algorithms can be measured, which indicates how much resources they consume. Hence, (3) *Efficiency Metrics* is added to the evaluation. In the Tables 5.2, 5.3, 5.4, 5.5, 5.6 and 5.7; $\uparrow$ ($\downarrow$) next to the metric indicates that the corresponding metric's score is to be maximized (minimized).

5.1.2.1 Instance-Based Metrics

Instance-based metrics are evaluated for every instance and averaged over the whole dataset. Exact Match, Hamming Score, and Instance-Based {Accuracy, Precision, Recall, F1-Score} [29] are used in this study.

Exact Match is the fraction of strictly correctly classified examples in the

dataset [29].

$$\text{Exact Match (EM)} = \frac{1}{N}\sum_{i=1}^{N}[\![y^i = \hat{y}^i]\!]$$

Hamming Score is the fraction of correctly classified labels (i.e. bitwise similarity) over all examples and each label [18].

$$\text{Hamming Score (HS)} = \frac{1}{LN}\sum_{i=1}^{N}\sum_{j=1}^{L}[\![y_j^i = \hat{y}_j^i]\!]$$

Example-based Accuracy, Precision, Recall and F1-Score are defined as follows [29]:

$$\text{Acc}_{ex} = \frac{1}{N}\sum_{i=1}^{N}\frac{|y^i \bigcap \hat{y}^i|}{|y^i \bigcup \hat{y}^i|} \qquad \text{Pr}_{ex} = \frac{1}{N}\sum_{i=1}^{N}\frac{|y^i \bigcap \hat{y}^i|}{|\hat{y}^i|}$$

$$\text{Re}_{ex} = \frac{1}{N}\sum_{i=1}^{N}\frac{|y^i \bigcap \hat{y}^i|}{|y^i|} \qquad \text{F1}_{ex} = \frac{2 * \text{Pr}_{ex} * \text{Re}_{ex}}{\text{Pr}_{ex} + \text{Re}_{ex}}$$

Example. To illustrate, assume a labelset with $L = 5$ labels, and for a given instance, let the prediction be $\hat{y} =< 1, 1, 0, 0, 0 >$ and the ground truth be $y =< 0, 1, 1, 1, 0 >$. According to these definitions of metrics,

- $EM = 0$, since two vectors are not completely the same.
- $HS = 2/5$, since 2 bits are matching between $\hat{y}$ and y.
- $Acc_{ex} = 1/4$, since 1 relevant bit is mutual among the relevant bits of $\hat{y}$ and y.
- $Pr_{ex} = 1/2$, since 1 relevant bit is correct among 2 predicted relevant bits.
- $Re_{ex} = 1/3$, since 1 relevant bit is correct among 3 correct bits from the ground truth.

5.1.2.2 Label-Based Metrics

Label-based metrics are evaluated for every label and averaged over examples within each individual label. Macro and Micro-Averaged Precision, Recall and F1 Score [30] are used in this study.

{Precision, Recall, F1-Score} $\in M$. M is a function of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) for each label, i.e.

$$M := M(TP_\lambda, TN_\lambda, FP_\lambda, FN_\lambda)$$

Then, the macro and micro-averaged evaluation metrics are defined as follows [30]:

$$M_{macro} = \frac{1}{L}\sum_{j=1}^{L} M(\mathrm{TP}_j, \mathrm{TN}_j, \mathrm{FP}_j, \mathrm{FN}_j)$$

$$M_{micro} = M\Big(\sum_{j=1}^{L}\mathrm{TP}_j, \sum_{j=1}^{L}\mathrm{TN}_j, \sum_{j=1}^{L}\mathrm{FP}_j, \sum_{j=1}^{L}\mathrm{FN}_j\Big)$$

5.1.2.3 Efficiency Metrics

Finally, to measure the efficiency of the algorithms, the execution time and memory consumption of each algorithm are monitored.

5.1.3 Experimental Setup

Experiments are implemented in MOA [55], utilizing multi-label methods in MEKA [56]. The evaluation of each algorithm is *prequential* [57]. An incoming data instance is first tested by classifiers (see Alg.1:5); evaluation measures corresponding the prediction are recorded, and then, that data instance is used to

train classifiers, as well as the updated weighting scheme (see Alg.1:9,15). This is also called *Interleaved-Test-Then-Train (ITTT)* approach and is widely common in algorithms in streaming settings.

If an ensemble is batch-incremental, then the ensemble is trained at the end of each batch (i.e. whenever a data chunk is filled). The evaluation of ensembles are started after the first learner in the ensemble is formed. We used fixed number of 10 classifiers as the ensemble size, mimicking the previously conducted experiments to enable comparison [38, 39]. For incremental evaluation of the classifiers, we used window-based evaluation with the window size $\{100, 250, 500, 1000\}$ according to the size of datasets. The results can be reproduced using the aforementioned datasets. The program outputs the predictive performance measures, time and memory consumption, as well as incremental evaluations for each window.

All experiments are conducted on a machine with an Intel Xeon E3-1200 v3 @ 3.40GHz processor and 128GB DDR3 RAM.

We experimented with 4 GOOWE-ML models (referred with their abbreviations from now onward):

- **GOBR:** the components use BR (Binary Relevance) Transformation.
- **GOCC:** the components use CC (Classifier Chains) Transformation.
- **GOPS:** the components use PS (Pruned Sets) Transformation.
- **GORT:** the components use iSOUP Regression Trees.

We have 7 baseline models. Four of the baselines use fixed-sized windows with no concept drift detecting mechanism: **EBR** [23], **ECC** [23], **EPS** [24], **EBRT** [39], whereas 3 of them use ADWIN as their concept drift detector: **EaBR** [38], **EaCC** [38], and **EaPS** [38]). In all models, BR and CC transformations use a Hoeffding Tree classifier whereas the PS transformation uses a Naive Bayes classifier.

5.1.4 Evaluation of Statistical Significance

We evaluated the aforementioned algorithms using multi-label example-based, and label-based evaluation metrics, as well as efficiency metrics. To check the statistical significance among the algorithms, we used *Friedman test with Nemenyi post-hoc analysis* [58]. We applied the Friedman test with $\alpha = 0.05$ where the null hypothesis is that all of the measurements come from the same distribution. If the null hypothesis is failed, then Nemenyi post-hoc analysis is applied to see which algorithms that performed statistically significantly better than which others.

The result of Friedman-Nemenyi Test can be seen in the *Critical Distance Diagram*s where each algorithm is sorted according to their average ranks for a given metric on a number line, and the algorithms that are within the critical distance of each other (that are not statistically significantly better than each other) are linked with a line. These diagrams compactly show Nemenyi Significance. Better models have lower average rank, and therefore on the right side of a Critical Distance Diagram. The Critical Distance for Nemenyi Significance is calculated as follows [58]:

$$CD = q_{\alpha,m}\sqrt{\frac{m(m+1)}{6|\mathcal{D}|}} \tag{5.1}$$

where m is the number of models that are being compared, and $|\mathcal{D}|$ number of datasets that are experimented on. Plugging in $m = 11$, $q_{\alpha=0.05,m=11} = 3.219$ (from Critical Values Table for Two-Tailed Nemenyi Test [2]) and $|\mathcal{D}| = 7$, we get $CD = 5.707$ as our Critical Distance.

[2] Available at: `http://www.cin.ufpe.br/~fatc/AM/Nemenyi_critval.pdf`

5.2 Results

5.2.1 Predictive Performance

Example-Based F1 Score, Micro-Averaged F1 Score, Hamming Score and Example-Based Accuracy are given in Tables 5.2, 5.3, 5.4, and 5.5 for each model on each dataset. Winner models of each dataset for the metrics are shown in bold in the table. Precision and Recall scores are omitted, since we report the F1 Scores, which is calculated as the harmonic mean of the two. Exact Match scores are omitted, since it is a very strict metric and the scores tend to be near zero for each algorithm especially when $|\mathcal{L}|$ is large.

Before starting to analyze models individually, let us look at the big picture: It is apparent that the predictive performance of a streaming multi-label model highly depends on the dataset. Looking at the results, no single model is clearly better than the rest, regardless of the dataset that it has run on. For instance, PS transformation-based ensembles (GOPS, EPS and EaPS) did relatively better in the Slashdot, Reuters and IMDB datasets; whereas the ensembles with BR and CC transformations were clearly superior in the Yeast, Ohsumed and TMC2007 datasets.

It can be observed in the Table 5.2, 5.3, and 5.4; and their corresponding critical distance diagrams (Figure 5.1, 5.2, and 5.3), GOOWE-ML-based classifiers performed better than Online Bagging and ADWIN Bagging models consistently over all datasets. Especially **GOCC** and **GOPS** placed 1st and 2nd respectively, in every performance metric, except Hamming Score. More detailed discussion on the Hamming Score and its relation to the Precision and Recall scores of the models are provided below in a separate section.

Read et al. [38] previously claimed that instance-incremental methods are better than batch-incremental methods in the MLSC task. However, our experimental evidence shows that our batch-incremental ensemble performs better than the state-of-the-art instance-incremental models in almost every performance metric

Table 5.2: Predictive Performance: Example-based F1 Score [1]

	20NG	Yeast	Ohsumed	Slashdot	Reuters	IMDB	TMC7	
Example-Based F1 Score ($F1_{ex}$) ↑								**Avg. Rank**
GOBR	0.364	0.650	0.307	0.189	0.076	0.283	0.623	4.00
GOCC	**0.442**	**0.652**	**0.352**	0.028	0.145	0.221	**0.668**	2.57
GOPS	0.224	0.644	0.331	**0.405**	**0.252**	**0.333**	0.485	3.00
GOBRT	0.196	0.607	0.297	0.189	0.078	0.283	0.452	5.71
EBR	0.365	0.638	0.230	0.023	0.106	0.075	0.654	4.71
ECC	0.349	0.632	0.217	0.020	0.098	0.016	0.643	6.43
EPS	0.096	0.584	0.213	0.269	0.148	0.133	0.330	6.71
EBRT	0.100	0.509	0.056	0.001	0.000	0.001	0.008	10.57
EaBR	0.341	0.638	0.202	0.018	0.059	0.031	0.661	6.57
EaCC	0.156	0.633	0.005	0.020	0.004	0.001	0.646	8.14
EaPS	0.109	0.578	0.200	0.258	0.183	0.104	0.384	6.85

Table 5.3: Predictive Performance: Example-based Accuracy [1]

	20NG	Yeast	Ohsumed	Slashdot	Reuters	IMDB	TMC7	
Example-Based Accuracy (Acc_{ex}) ↑								**Avg. Rank**
GOBR	0.239	0.508	0.184	0.106	0.040	0.164	0.457	4.57
GOCC	**0.391**	**0.509**	**0.277**	0.025	0.120	0.138	0.515	<u>3.00</u>
GOPS	0.137	0.504	0.211	**0.299**	0.160	**0.204**	0.327	3.29
GOBRT	0.115	0.454	0.178	0.107	0.040	0.164	0.298	6.71
EBR	0.352	0.502	0.191	0.020	0.098	0.055	0.520	4.29
ECC	0.337	0.493	0.180	0.018	0.093	0.012	0.511	6.14
EPS	0.094	0.460	0.180	0.260	0.143	0.105	0.246	6.29
EBRT	0.100	0.372	0.049	0.001	0.000	0.001	0.007	10.57
EaBR	0.330	0.502	0.169	0.016	0.056	0.024	**0.529**	6.14
EaCC	0.152	0.495	0.004	0.018	0.004	0.001	0.516	7.71
EaPS	0.108	0.455	0.170	0.250	**0.179**	0.083	0.290	6.43

Table 5.4: Predictive Performance: Micro-Averaged F1 Score [1]

	20NG	Yeast	Ohsumed	Slashdot	Reuters	IMDB	TMC7	
Micro-Averaged F1 Score ($F1_{micro}$) ↑								**Avg. Rank**
GOBR	0.237	0.638	0.291	0.187	0.076	0.276	0.584	4.86
GOCC	**0.516**	**0.640**	**0.410**	0.050	0.196	0.228	0.634	2.71
GOPS	0.206	0.629	0.298	**0.315**	**0.210**	**0.314**	0.447	3.43
GOBRT	0.153	0.598	0.270	0.187	0.077	0.277	0.439	6.57
EBR	0.499	0.631	0.294	0.041	0.141	0.099	0.638	4.29
ECC	0.486	0.625	0.280	0.037	0.134	0.025	0.631	6.14
EPS	0.115	0.584	0.216	0.286	0.162	0.138	0.342	7.00
EBRT	0.174	0.519	0.076	0.001	0.000	0.001	0.008	10.58
EaBR	0.477	0.632	0.266	0.033	0.081	0.041	**0.640**	5.71
EaCC	0.262	0.627	0.007	0.037	0.007	0.002	0.632	7.71
EaPS	0.180	0.580	0.205	0.278	0.200	0.118	0.378	6.71

Table 5.5: Predictive Performance: Hamming Score [1]

	20NG	Yeast	Ohsumed	Slashdot	Reuters	IMDB	TMC7	
Hamming Score ↑								**Avg. Rank**
GOBR	0.749	0.769	0.738	0.625	0.707	0.727	0.886	9.86
GOCC	0.952	0.771	0.932	0.946	0.984	0.887	0.916	5.57
GOPS	0.769	0.754	0.830	0.872	0.956	0.836	0.854	9.29
GOBRT	0.624	0.716	0.730	0.644	0.720	0.732	0.815	10.57
EBR	**0.961**	0.786	**0.936**	0.946	**0.986**	0.925	0.934	2.14
ECC	**0.961**	0.786	**0.936**	**0.947**	**0.986**	0.928	0.934	1.57
EPS	0.924	0.764	0.918	0.937	0.985	0.919	0.911	7.29
EBRT	0.952	0.773	0.930	0.946	**0.986**	**0.929**	0.902	4.00
EaBR	**0.961**	0.786	0.935	0.946	**0.986**	0.928	**0.935**	2.00
EaCC	0.955	**0.787**	0.928	**0.947**	**0.986**	**0.929**	0.934	2.29
EaPS	0.950	0.767	0.918	0.937	0.985	0.924	0.913	6.71

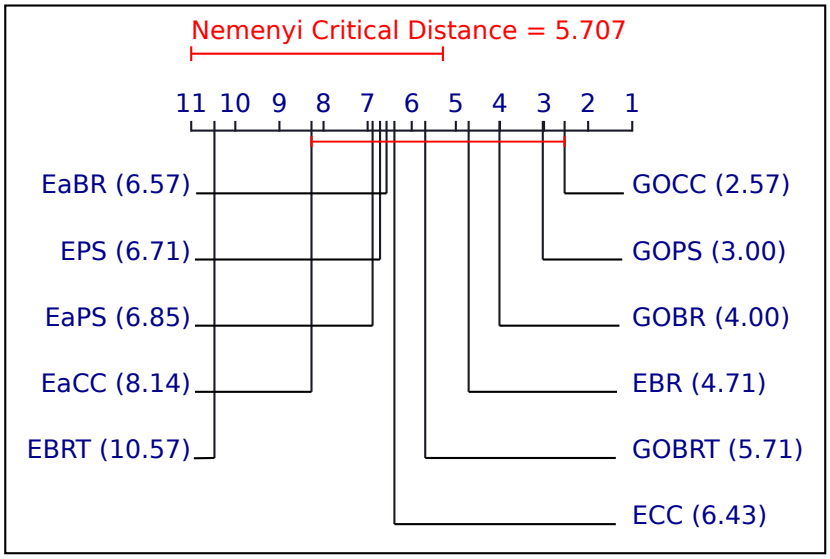

Figure 5.1: Critical Distance Diagram for Example-based F1 Score [1] (given in Table 5.2).

(again, except Hamming Score).

5.2.2 Efficiency

Results for the Execution Time and Memory Consumption of the models, and the corresponding Critical Distance Diagrams are given in Table 5.6 and 5.7; and Figure 5.5 and 5.6, respectively. It is clear that time and memory efficiency of an MLSC ensemble is highly correlated with the problem transformation method that its component classifiers use. Ensembles that used PS Transformation (GOPS, EPS, EaPS) are ranked consistently higher in terms of both time and memory efficiency. Indeed, as it can be seen in Figure 5.5 and 5.6, EPS and GOPS are among the top 3 for both of the metrics.

Models with iSOUP Tree are among the fastest, but their memory consumption is significantly high compared to the PS-based ensembles. Considering GORT and EBRT's relatively underwhelming predictive performance (see Figure 5.1 and 5.3), PS-based ensembles should be preferable over ensembles of iSOUP Regression Trees.

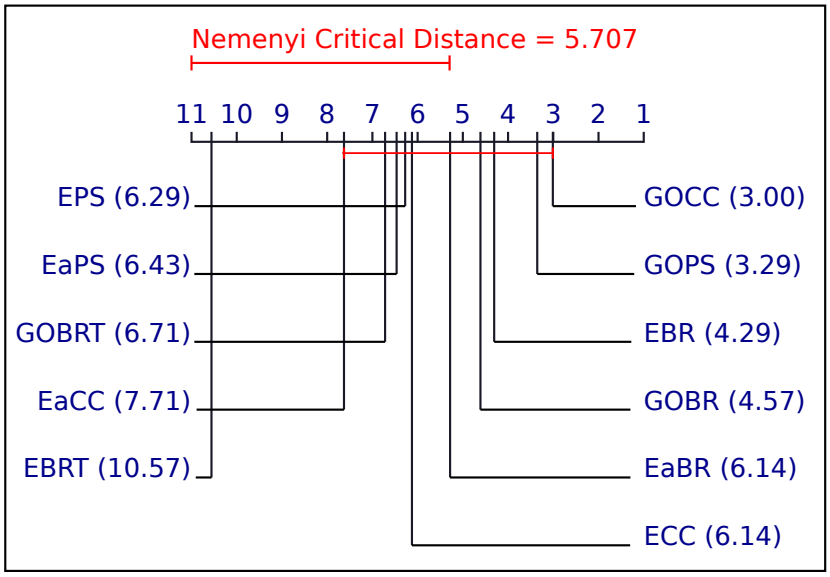

Figure 5.2: Critical Distance Diagram for Example-based Accuracy [1] (given in Table 5.3).

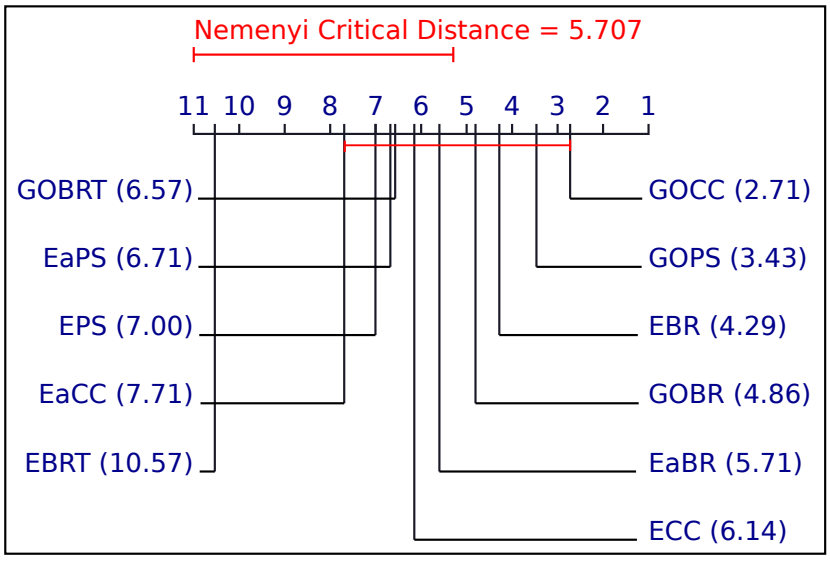

Figure 5.3: Critical Distance Diagram for Micro-averaged F1 Score [1] (given in Table 5.4).

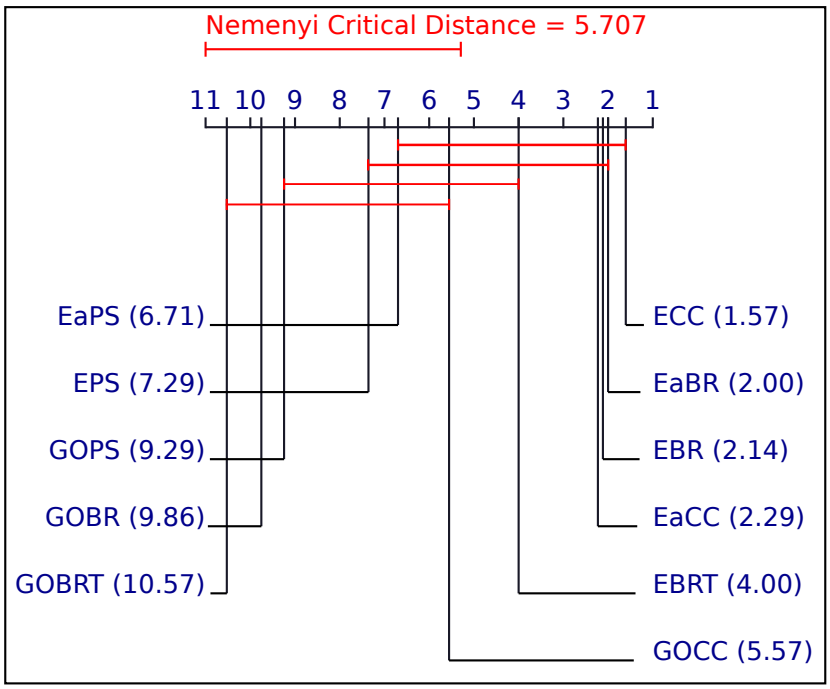

Figure 5.4: Critical Distance Diagram for Hamming Score [1] (given in Table 5.5).

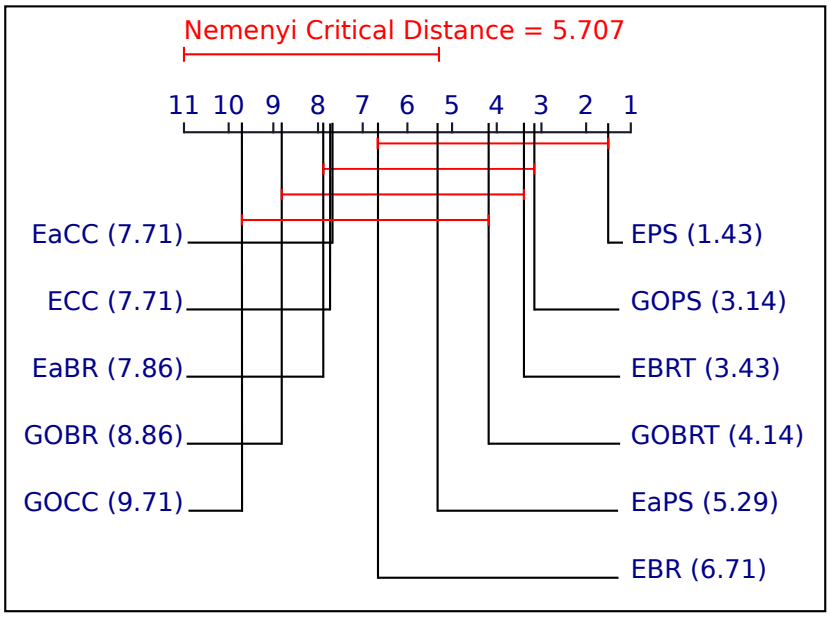

Figure 5.5: Critical Distance Diagram for Time Consumption [1] (given in Table 5.6).

Table 5.6: Efficiency: Time Consumption [1]

	20NG	Yeast	Ohsumed	Slashdot	Reuters	IMDB	TMC7	
(a) Execution Time (seconds) ↓								**Avg. Rank**
GOBR	2,631	28	2,310	537	2,366	31,769	1,942	8.86
GOCC	2,591	33	2,314	544	2,555	34,348	1,990	9.71
GOPS	670	8	522	129	115	5,098	181	3.14
GOBRT	390	47	435	68	412	3,719	333	4.14
EBR	2,246	25	1,934	488	1,917	101,243	1,769	6.71
ECC	2,270	29	1,958	495	2,057	48,325	1,789	7.71
EPS	383	5	299	99	46	2,168	109	<u>1.43</u>
EBRT	338	63	404	64	389	3,919	264	3.43
EaBR	2,376	35	1,997	488	1,968	20,675	2,220	7.86
EaCC	2,041	40	1,622	503	2,062	17,148	2,292	7.71
EaPS	2,393	24	1,862	363	402	14,361	574	5.29

Table 5.7: Efficiency: Memory Consumption [1]

	20NG	Yeast	Ohsumed	Slashdot	Reuters	IMDB	TMC7	
(b) Memory Consumption (MB) ↓								**Avg. Rank**
GOBR	1,685.82	18.40	1,364.32	381.98	1,029.23	4,384.09	780.58	7.57
GOCC	1,429.26	24.85	1,229.32	351.74	1,261.34	6,284.62	748.33	7.43
GOPS	76.30	2.03	43.55	29.38	15.57	75.68	41.88	3.00
GOBRT	431.76	198.81	656.16	77.92	660.38	542.24	227.70	5.71
EBR	2,152.97	17.56	1,775.72	545.72	1,425.88	22,119.42	1,289.87	9.00
ECC	2,171.09	28.99	1,792.66	549.33	1539	22,380.89	1,305.07	10.57
EPS	8.40	0.97	8.53	10.38	3.67	7.55	6.34	1.29
EBRT	521.96	274.61	809.76	76.70	943.59	1,826.87	234.06	6.57
EaBR	1,997.59	17.57	1,678.32	399.99	1,330.31	3,522.14	93.60	7.29
EaCC	373.03	26.39	295.09	549.35	652.92	661.76	135.09	5.86
EaPS	6.25	1.50	15.80	12.15	8.05	13.53	2.56	1.71

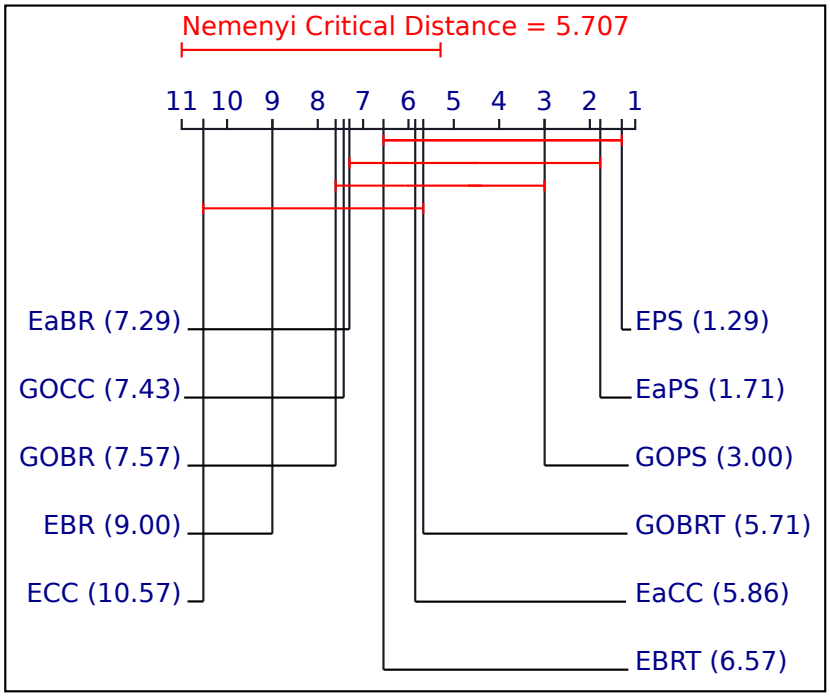

Figure 5.6: Critical Distance Diagram for Memory Consumption [1] (given in Table 5.7).

BR and CC Transformation-based models performed similarly within datasets across ensembling techniques. Their execution time and memory consumption is nearly identical with a few exceptions (where ADWIN Bagging models had significantly lower memory consumption due to resetting component classifiers many times). Having similar resource consumptions, GOCC can be preferred due to its greater predictive performance.

5.3 Discussion

5.3.1 On Hamming Scores in Datasets with Large Labelsets

Consider the prediction and the ground truth vector of a given data instance. Let TP, FP, FN and TN denote the number of true positives, false positives, false negatives and true negatives. For instance, FP is the number of labels that are predicted as relevant but are not. Then, Hamming Score for that instance can

be calculated as $\frac{TP+TN}{TP+FP+FN+TN}$.

For a multi-label dataset with a fairly large labelset and low label density, TN in the numerator and the denominator will dominate this score and Hamming Score will yield mis-interpretable results. Take IMDB dataset ($|\mathcal{L}| = 28$, $LD(\mathcal{D}) = 0.071$) for example: **GOPS** was the clear winner in terms of $F1_{ex}$, Acc_{ex} and $F1_{micro}$, yielding 20% better scores than its closest competitor (which was **GOBR**). Despite performing this well, GOPS had a considerably *low* Hamming Score (**0.836**) with respect to the Online Bagging-based models (all of them around **0.928**). Here, one can argue that perhaps the Hamming Score is the true indicator of success in MLL, and therefore Online Bagging-based models performed better. However, this hypothesis cannot be correct, since even a dummy classifier that predicts every single label as irrelevant (0) yields Hamming Score of **0.929**! Additionally, the reason why GOOWE-ML-based models have smaller Hamming Scores is that they have high FP (hence low TN) values in the contingency table. In other words, GOOWE-ML-based models are *Low Precision - High Recall* models. They eagerly predict labels as relevant. On the other hand, Online Bagging-based models are *High Precision - Low Recall* models. They predict few labels as relevant at each data instance. As a consequence, they are more confident about their predictions, but they miss many relevant labels due to being more conservative. This dichotomy is shown for 3 datasets with low label densities in Table 5.8, where the higher value among Precision and Recall is shown in bold for each model and dataset.

Observing these, we claim that *in datasets with large labelset and low label density, High Recall models may have considerably low Hamming Scores due to the nature of the metric. Hence, Hamming Score may not be the true indicator of the predictive performance while evaluating multi-label models.*

This hypothesis helps explaining why GOBR and GOCC performed poorly in terms of Hamming Scores in the datasets with relatively lower label densities (such as Slashdot, Reuters and IMDB datasets) whereas both of them were clear winners in the overall predictive performance.

Table 5.8: Micro Precision (Prec) vs Recall (Rec), and Their Effect on Hamming Score (HS) [1]. Two GOOWE-ML models and two Online Bagging Models with different problem transformation types are picked. Higher Precision and lower Recall resulted in better Hamming Scores consistently.

	20NG			**Ohsumed**			**Reuters**		
	Prec	Rec	HS	Prec	Rec	HS	Prec	Rec	HS
GOBR	0.140	**0.757**	0.749	0.181	**0.743**	0.738	0.040	**0.848**	0.707
GOPS	0.125	**0.580**	0.769	0.212	**0.500**	0.830	0.140	**0.418**	0.956
EBR	**0.753**	0.373	0.961	**0.713**	0.185	0.936	**0.510**	0.082	0.986
EPS	**0.142**	0.096	0.924	**0.348**	0.157	0.918	**0.361**	0.105	0.985

5.3.2 Window-Based Evaluation

Figure 5.7 presents window-based evaluation for two datasets and three models, where the sliding window size is equal to the chunk size, i.e. $n = h$. To this end, we evaluate each window's performance using $F1_{ex}$ measurement. For each group of models, the best performing strategy is chosen for the given dataset—e.g. in Reuters dataset, GOPS, EPS and EaPS performed the best among GOOWE-ML, Online Bagging and ADWIN Bagging models, respectively.

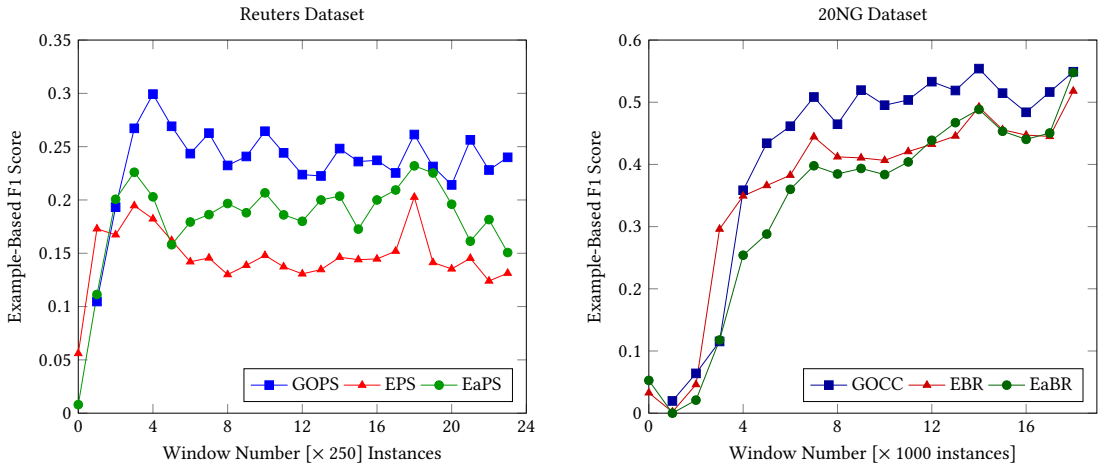

Figure 5.7: Window-Based Evaluation of Models: Example-Based F1 Score for Reuters and 20NG datasets [1].

It can be seen that GOOWE-ML-based models do not predict in the first evaluation window, since no training has been done while waiting the first chunk to be filled. In both of the datasets, we observe the optimal weight assignment strategy in effect: after the first few chunks, GOOWE-ML-based model's predictions continually yield better performance than its competitors.

Chapter 6

Conclusion and Future Work

We present an online batch-incremental multi-label stacked ensemble, GOOWE-ML, that constructs a spatial model for the relevance scores of its classifiers, and uses this model to assign optimal weights to its component classifiers. Our experiments show that GOOWE-ML models outperform the most prominent Online Bagging and ADWIN Bagging models. Two of the GOOWE-ML-based ensembles especially stand out: **GOCC** is the clear winner in terms of overall predictive performance, ranking first in Acc_{ex}, $F1_{ex}$ and $F1_{micro}$ scores. **GOPS**, on the other hand, is the best compromise between predictive performance and resource consumption among all models, yielding strong performance with very conservative time and memory requirements. In addition, we argue that Hamming Score can be deceptively low for models with low Precision and high Recall. We support this claim by experimental evidence.

Below is the list of possible extensions to the proposed work, and future research pointers:

- **Explicit Concept Drift Detector and Pruning:** As it is described in Chapter 4, GOOWE-ML handles concept drifts implicitly, by replacing one of its components at the end of each data chunk. Even though this scheme already outperforms its alternatives, it is possible to incorporate an explicit

drift detection mechanism into the proposed ensemble, as well. In that case, it would be possible to remove (prune) multiple ensemble components at once.

- **Parameter Initialization:** For multi-class classification, the problem of optimal ensemble size is theoretically studied [48, 59]. Yet, to the best of our knowledge, the same problem for multi-label classification is not yet discussed in the literature. What should be the optimal ensemble size for multi-label ensembles? In addition, what should be the batch size for batch-incremental ensembles (such as GOOWE-ML)? How are these parameters related to the dimensionality of the feature set, number of labels, label cardinality and label density for a given multi-label dataset?

- **Synthetic Multi-label Data Generation:** There is a lack of huge real-world datasets in the field of MLSC. Hence, people resort to generating synthetic data streams. Apart from the obvious reliability issues of experimenting with synthetic data, the generation processes of such data are also problematic. Most of the time, synthetic datasets are generated according to [60], but concept drift behavior in such datasets are simulated by only considering changes in label cardinality. Embedding correlations and inverse correlations between labels into this framework, it can be possible to generate more realistic multi-label datasets.

- **Examining Inherent Structures in Multi-label Data:** Even though any subset of the labels can be relevant for a data instance in a multi-label dataset, it is generally the case that some labelsets are much more frequent than others. Existing studies exploit this fact by keeping statistics about the frequently occurring labels during pre-training [24]. If this *long-tail effect* is inherent in multi-label data, then a relationship similar to Power Law can be inferred, and that can be utilized to generate candidate frequent labelsets without a pre-training step.

Bibliography

[1] A. Büyükçakır, H. Bonab, and F. Can, "A novel online stacked ensemble for multi-label stream classification," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 1063–1072, ACM, 2018.

[2] C. C. Aggarwal, *Data streams: models and algorithms*, vol. 31. Springer Science & Business Media, 2007.

[3] P. Zikopoulos, C. Eaton, *et al.*, *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*. McGraw-Hill Osborne Media, 2011.

[4] K. Normandeau, "Beyond volume, variety and velocity is the issue of big data veracity," *Inside Big Data*, 2013.

[5] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171–209, 2014.

[6] J. Li, F. Tao, Y. Cheng, and L. Zhao, "Big data in product lifecycle management," *The International Journal of Advanced Manufacturing Technology*, vol. 81, no. 1-4, pp. 667–684, 2015.

[7] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys*, vol. 46, no. 4, p. 44, 2014.

[8] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: A survey," *IEEE Computational Intelligence Magazine*, vol. 10, no. 4, pp. 12–25, 2015.

[9] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà, "New ensemble methods for evolving data streams," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 139–148, ACM, 2009.

[10] C. Staff, "Big data," *Commun. ACM*, vol. 60, pp. 24–25, May 2017.

[11] A. Clare and R. D. King, "Knowledge discovery in multi-label phenotype data," in *European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 42–53, Springer, 2001.

[12] Z. Barutcuoglu, R. E. Schapire, and O. G. Troyanskaya, "Hierarchical multi-label prediction of gene function," *Bioinformatics*, vol. 22, no. 7, pp. 830–836, 2006.

[13] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "Rcv1: A new benchmark collection for text categorization research," *Journal of Machine Learning Research*, vol. 5, no. Apr, pp. 361–397, 2004.

[14] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern Recognition*, vol. 37, no. 9, pp. 1757–1771, 2004.

[15] J. M. Moyano, E. L. Gibaja, K. J. Cios, and S. Ventura, "Review of ensembles of multi-label classifiers: Models, experimental study and prospects," *Information Fusion*, vol. 44, pp. 33–45, 2018.

[16] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: A survey," *Information Fusion*, vol. 37, pp. 132 – 156, 2017.

[17] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[18] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37, no. 3, pp. 297–336, 1999.

[19] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.

[20] H. Bonab and F. Can, "GOOWE: Geometrically Optimum and Online-Weighted Ensemble classifier for evolving data streams," *ACM Transactions on Knowledge Discovery from Data*, vol. 12, no. 2, p. 25, 2018.

[21] D. Brzezinski and J. Stefanowski, "Reacting to different types of concept drift: The accuracy updated ensemble algorithm," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 81–94, 2014.

[22] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *Journal of Machine Learning Research*, vol. 8, no. Dec, pp. 2755–2790, 2007.

[23] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 254–269, Springer, 2009.

[24] J. Read, B. Pfahringer, and G. Holmes, "Multi-label classification using ensembles of pruned sets," in *The Eighth IEEE ICDM*, pp. 995–1000, IEEE, 2008.

[25] W. Qu, Y. Zhang, J. Zhu, and Q. Qiu, "Mining multi-label concept-drifting data streams using dynamic classifier ensemble," in *Asian Conference on Machine Learning*, pp. 308–321, Springer, 2009.

[26] L. Wang, H. Shen, and H. Tian, "Weighted ensemble classification of multi-label data streams," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 551–562, Springer, 2017.

[27] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proceedings of the 2007 SIAM International Conference on Data Mining*, pp. 443–448, SIAM, 2007.

[28] M. S. Sorower, "A literature survey on algorithms for multi-label learning," *Oregon State University, Corvallis*, vol. 73, 2010.

[29] M.-L. Zhang and Z.-H. Zhou, "A review on multi-label learning algorithms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 8, pp. 1819–1837, 2014.

[30] E. Gibaja and S. Ventura, "Multi-label learning: A review of the state of the art and ongoing research," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 4, no. 6, pp. 411–444, 2014.

[31] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet, "A survey on ensemble learning for data stream classification," *ACM Comput. Surv.*, vol. 50, pp. 23:1–23:36, Mar. 2017.

[32] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Random k-labelsets for multilabel classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 7, pp. 1079–1089, 2011.

[33] J. Read, L. Martino, P. M. Olmos, and D. Luengo, "Scalable multi-output label prediction: From classifier chains to classifier trellises," *Pattern Recognition*, vol. 48, no. 6, pp. 2096–2109, 2015.

[34] W. Cheng, E. Hüllermeier, and K. J. Dembczynski, "Bayes optimal multilabel classification via probabilistic classifier chains," in *Proceedings of the 27th International Conference on Machine Learning*, pp. 279–286, 2010.

[35] J. Read, L. Martino, and D. Luengo, "Efficient monte carlo methods for multi-dimensional learning with classifier chains," *Pattern Recognition*, vol. 47, no. 3, pp. 1535–1546, 2014.

[36] M.-L. Zhang and Z.-H. Zhou, "Ml-knn: A lazy learning approach to multi-label learning," *Pattern Recognition*, vol. 40, no. 7, pp. 2038–2048, 2007.

[37] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 71–80, ACM, 2000.

[38] J. Read, A. Bifet, G. Holmes, and B. Pfahringer, "Scalable and efficient multi-label classification for evolving data streams," *Machine Learning*, vol. 88, no. 1-2, pp. 243–272, 2012.

[39] A. Osojnik, P. Panov, and S. Džeroski, "Multi-label classification via multi-target regression on data streams," *Machine Learning*, vol. 106, no. 6, pp. 745–770, 2017.

[40] N. C. Oza, "Online bagging and boosting," in *2005 IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, pp. 2340–2345, IEEE, 2005.

[41] E. S. Xioufis, M. Spiliopoulou, G. Tsoumakas, and I. P. Vlahavas, "Dealing with concept drift and class imbalance in multi-label stream classification," in *IJCAI*, pp. 1583–1588, 2011.

[42] X. Kong and P. Yu, "An ensemble-based approach to fast classification of multi-label data streams," in *7th International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pp. 95–104, IEEE, 2011.

[43] P. Wang, P. Zhang, and L. Guo, "Mining multi-label data streams using ensemble-based active learning," in *Proceedings of the 2012 SIAM International Conference on Data Mining*, pp. 1131–1140, SIAM, 2012.

[44] R. Sousa and J. Gama, "Multi-label classification from high-speed data streams with adaptive model rules and random rules," *Progress in Artificial Intelligence*, pp. 1–11, 2018.

[45] L. I. Kuncheva, "Classifier ensembles for changing environments," in *International Workshop on Multiple Classifier Systems*, pp. 1–15, Springer, 2004.

[46] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 226–235, ACM, 2003.

[47] F. Tai and H.-T. Lin, "Multilabel classification with principal label space transformation," *Neural Computation*, vol. 24, no. 9, pp. 2508–2542, 2012.

[48] H. R. Bonab and F. Can, "A theoretical framework on the ideal number of classifiers for online ensembles in data streams," in *Proceedings of the 25th ACM CIKM International Conference on Information and Knowledge Management*, pp. 2053–2056, ACM, 2016.

[49] S. Wu and F. Crestani, "A geometric framework for data fusion in information retrieval," *Information Systems*, vol. 50, no. Supplement C, pp. 20 – 35, 2015.

[50] C. L. Lawson and R. J. Hanson, *Solving least squares problems*, vol. 15. SIAM, 1995.

[51] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[52] P. B. Stark and R. L. Parker, "Bounded-variable least-squares: an algorithm and applications," *Computational Statistics*, vol. 10, pp. 129–129, 1995.

[53] A. Bojańczyk, "Complexity of solving linear systems in different models of computation," *SIAM Journal on Numerical Analysis*, vol. 21, no. 3, pp. 591–603, 1984.

[54] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *International Journal of Data Warehousing and Mining*, vol. 3, no. 3, 2006.

[55] A. Bifet, G. Holmes, B. Pfahringer, J. Read, P. Kranen, H. Kremer, T. Jansen, and T. Seidl, "MOA: A real-time analytics open source framework," in *ECML PKDD*, pp. 617–620, Springer, 2011.

[56] J. Read, P. Reutemann, B. Pfahringer, and G. Holmes, "MEKA: A multi-label/multi-target extension to WEKA," *Journal of Machine Learning Research*, vol. 17, no. 21, pp. 1–5, 2016.

[57] J. Gama, R. Sebastião, and P. P. Rodrigues, "Issues in evaluation of stream learning algorithms," in *the 15th ACM SIGKDD*, pp. 329–338, ACM, 2009.

[58] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, no. Jan, pp. 1–30, 2006.

[59] H. Bonab and F. Can, "Less is more: a comprehensive framework for the number of components of ensemble classifiers," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[60] J. Read, B. Pfahringer, and G. Holmes, "Generating synthetic multi-label data streams," in *ECML/PKDD 2009 Workshop on Learning from Multi-label Data (MLD'09)*, pp. 69–84, 2009.

[61] J. Montiel, J. Read, A. Bifet, and T. Abdessalem, "Scikit-multiflow: a multi-output streaming framework," *The Journal of Machine Learning Research*, vol. 19, no. 1, pp. 2915–2919, 2018.

Appendix A

Code and Reproducibility

The proposed method and the experiments are initially written in Java and available at `https://github.com/abuyukcakir/gooweml`. Instructions on how to run the experiments and reproduce the results are also stated there.

After the release of scikit-multiflow [61], a Python-based alternative to MOA [55] framework, we extended GOOWE-ML (and its precursor, GOOWE [20]) to Python language with scikit-multiflow compatibility as well. Python version is available at `https://github.com/abuyukcakir/goowe-python`.