Proceedings of Machine Learning Research 94:23–37, 2018 LIDTA 2018

Multi-label kNN Classifier with Self Adjusting Memory for Drifting Data Streams

Martha Roseberry MROSEBERRY@VCU.EDU
Alberto Cano ACANO@VCU.EDU
Department of Computer Science
Virginia Commonwealth University
Richmond, VA, USA

Editors: Luís Torgo, Stan Matwin, Nathalie Japkowicz, Bartosz Krawczyk, Nuno Moniz, and Paula Branco

Abstract

Multi-label data streams is a highly challenging task involving drifts in features and labels. Classifiers must automatically adapt to changes while keeping a competitive accuracy in a real-time dynamic environment where the frequencies of the labelsets are non-stationary and highly imbalanced. This paper presents a multi-label k Nearest Neighbor (kNN) with Self Adjusting Memory (SAM) for drifting data streams (ML-SAM-kNN). It exploits short- and long-term memories to predict the current and evolving states of the data stream. The experimental study compares the proposal with eight other multi-label classifiers for data streams on 23 datasets on six multi-label metrics, evaluation time, and memory consumption. Non-parametric statistical analysis of the results shows the superiority of ML-SAM-kNN, including when compared with ML-kNN.

Keywords: Multi-label classification, Nearest neighbor, Data stream, Concept drift

1. Introduction

The increasing number of real world intelligent systems that continuously generate data has spurred on much recent research into data stream mining. Concerned with the velocity of the data, as much as with the volume, such algorithms strive to achieve a high accuracy while maintaining a computational complexity low enough to handle the rapidly incoming data. Many methods have been developed to help solve this problem (Gaber, 2012; Gomes et al., 2017), the most recent often concerned with evolving data streams, where the stream's data distribution is assumed to be dynamic and drifting (Gama et al., 2014; Ditzler et al., 2015; Krawczyk et al., 2017; Khamassi et al., 2018).

On a different tack, another challenge for contemporary machine learning is multi-label learning where each instance, rather than being associated with a single label, may simultaneously belong to many. Multi-label classification is relevant to many real world scenarios, including, but not limited to, text classification, scene classification, and bioinformatics. Still more challenging is the intersection of the two problems, where multi-label data is arriving as a stream. Although not entirely unexplored (Read et al., 2012; Osojnik et al., 2017; Sousa and Gama, 2018), there is a need for higher performing and efficient multi-label algorithms for evolving data streams.

One recent method proposed for drifting data streams is Self Adjusting Memory k Nearest Neighbor (SAM-kNN) (Losing et al., 2016, 2018). In this method, a window of the most recent instances is used as a short-term memory, while information from older instances are kept in a compressed form in a long-term memory. Both memories automatically adjust as the stream progresses. The use of two memories allows SAM-kNN to adapt to varying forms of concept drift. Using simple multi-class kNN classifiers with each of its memories, SAM-kNN outperforms other state-of-the-art algorithms for data stream mining, demonstrating a robust ability to adapt to varying concept drifts.

In this paper we present the Multi-label Self Adjusting Memory k Nearest Neighbors algorithm (ML-SAM-kNN), a multi-label classifier explicitly designed for data streams experiencing concept drift. Multi-label kNN classifiers are used with a memory architecture inspired by SAM-kNN (Losing et al., 2016, 2018) that distinguishes between the current and past concepts of the data stream, allowing the method to adapt to change without forgetting information from earlier in the stream. We show experimentally that ML-SAM-kNN attains higher quality predictive results than other multi-label data stream algorithms, consistently and across all multi-label evaluation metrics.

The main contributions of this paper are:

- ML-SAM-kNN: a new classification algorithm for multi-label drifting data streams
- A multi-label self-adjusting memory architecture using short and long-term memories to adapt to evolving data streams
- A comprehensive experimental study comparing ML-SAM-kNN to eight state-of-the-art multi-label data stream algorithms using 23 data benchmarks.

The rest of the paper is organized as follows. In Section 2 we present a background in data stream mining and multi-label learning. Section 3 describes the proposed algorithm. Section 4 details the experimental study and discusses the results. Concluding remarks are given in Section 5.

2. Background

A data stream is a sequence $S = \{s_1, s_2, \dots\ s_t, \dots\}$ of instances that arrive continuously. Each $s_i = (\mathbf{x}_i, y_i)$, where y is the corresponding label given to the input vector $\mathbf{x}$, is generated by the distribution $P_i(\mathbf{x}, y)$. Data streams are assumed to be potentially infinite, with instances arriving in some order and in rapid succession (Khamassi et al., 2018). Because of this, a classifier has a limited amount of time in which to process each instance and will never have access to all instances simultaneously. In addition, while the instances in a data stream may be generated from a fixed distribution, in real world scenarios it is likely that a data stream will evolve over time, experiencing concept drift (Khamassi et al., 2018). If at any times t and $t + \Delta$, the distributions $P_t(\mathbf{x}, y) \neq P_{t+\Delta}(\mathbf{x}, y)$, a concept drift has occurred (Krawczyk et al., 2017). There are a variety of types of concept drift, depending on the speed and nature of the change, including (Khamassi et al., 2018; Krawczyk et al., 2017):

- real drift, where the posterior probabilities $P_t(y|\mathbf{x})$ are changing.
- virtual drift, where the prior probabilities of the classes $P_t(y)$ are changing.

- abrupt drift, where distribution $P_t(\mathbf{x}, y)$ is suddenly replaced by $P_{t+1}(\mathbf{x}, y)$.
- gradual drift, where the concept shifts slowly and instances are decreasingly less likely to be generated by $P_t(\mathbf{x}, y)$ and increasingly more likely to be generated by $P_{t+1}(\mathbf{x}, y)$.
- incremental drift, where $P_t(\mathbf{x}, y)$ slowly morphs into $P_{t+1}(\mathbf{x}, y)$ and instances are generated by a series of intermediate distributions during the transition.
- recurring drift, where a previous concept reappears, either cyclically or not.

Both real and virtual drift may be abrupt, gradual or incremental, and any combination may be recurring. Thus, real data streams may experience a mixture of many types of drift.

There are many different strategies for data mining with drifting data streams, often separated into active strategies and passive strategies (Gama et al., 2014; Losing et al., 2016; Gomes et al., 2017). In active strategies, a classifier employs some mechanism to detect concept drift. Upon detecting drift, the classifier updates itself. The adaptive windowing used in ADWIN is an example of an active strategy (Bifet and Gavaldà, 2007). Passive strategies don't detect drift explicitly, but rather continuously update themselves as new information arrives, gradually forgetting outdated information. Many ensemble methods used for data streams employ a passive strategy (Gomes et al., 2017). Passive strategies adapt well to incremental and gradual concept drift, but may react slowly to abrupt drift (Gomes et al., 2017; Losing et al., 2018). Conversely, active strategies often adapt quickly in cases of abrupt drift, but a slow, incremental drift may not trigger the drift detector (Losing et al., 2018). Both strategies typically forget previous concepts, meaning that recurring concepts will be treated as new.

Self Adjusting Memory k Nearest Neighbor (SAM-kNN) is a recent, biologically inspired proposal by Losing et al. (2016, 2018) that updates and adapts a short-term memory so it contains only the current concept, while also retaining a record of all past concepts in a long-term memory. New instances are added to the short-term memory (STM), which reduces in size whenever the current concept changes. This is done by evaluating multiple window sizes and retaining the one with the minimum interleaved test-train error. Instances discarded from the STM are transferred to the long-term memory (LTM). Here, older instances are not forgotten when newer instances arrive. Rather, whenever a maximum size is reached, clustering is used to evenly thin instances allowing the LTM to be compressed while still retaining knowledge from all previous concepts. Only instances that conflict with the current concept in the STM are thrown out. Both the STM and the LTM induce a classifier, as does the union of the two, the combined memory. The final prediction is made by the memory with the current highest average accuracy. Using the two memories, this method seeks to take advantage of the STM's sliding window to react quickly to abrupt drift, while also enabling the classifier to fall back on the LTM in cases of recurring drift.

Although the short- and long-term memories are broadly applicable, Losing et al. (2018) use the memories with the simple kNN classifier. At odds with data stream mining's need for quick processing, each prediction using kNN requires the costly operation of finding the k nearest neighbors (Zhang et al., 2011). On the other hand, as a lazy learner, kNN incrementally updates with the addition of each new instance, a convenient feature for data streams with concept drift. Another advantage of kNN is that it can and has been adapted for the multi-label learning environment (Zhang and Zhou, 2007; Gonzalez et al., 2018).

In single-label classification, each instance is associated with a single class y from a set of classes L (Tsoumakas and Katakis, 2007). This can be either binary, where $|L| = 2$, or multi-class, where $|L| > 2$. Imbalance measures the relative ratio among classes. In multi-label classification, each instance is associated with a set of labels Y where $Y \subseteq L$. The cardinality and density measure the imbalance of the labels in the labelset (Charte et al., 2015; Cano et al., 2016). Methods for multi-label learning can be categorized as either problem transformation methods, algorithm adaption methods or ensemble methods (Madjarov et al., 2012; Gibaja and Ventura, 2015). Problem transformation methods transform a multi-label problem into one or more single-label problems, which can then be solved using existing single-label classifiers. Examples of problem transformation methods include binary relevance (BR) (Tsoumakas and Katakis, 2007) and classifier chains (CC) (Read et al., 2011). Algorithm adaption methods, including ML-C4.5 (Clare and King, 2001) and ML-kNN (Zhang and Zhou, 2007; Skryjomski et al., 2018), adapt existing classifiers to work directly with multi-label data. Ensemble methods have also been developed as multi-label classifiers (Gonzalez et al., 2017), a popular example being the RAKEL method (Tsoumakas and Vlahavas, 2007).

Despite the large number of research works on multi-class classifiers for data streams, few works address multi-label data streams. Those that are inherently updatable, like classifier chains (CC), can be used with streaming data. A few methods, such as iSOUP-Tree (Osojnik et al., 2017) and ML-AMRules (Sousa and Gama, 2018), have been specifically designed for multi-label data streams. However, their performance is far from their counterparts for static data. Therefore, there is a need to develop new multi-label data stream classifiers.

3. ML-SAM-kNN

The basic concept of ML-SAM-kNN was to create a simple, yet effective, classifier for multi-label data streams using a self-adjusting memory, inspired by Losing et al. (2018), and a majority-vote kNN adapted for multi-label. ML-SAM-kNN uses the short-term memory (STM) and long-term memory (LTM), both updated and maintained in a method similar to SAM-kNN, but adapted for multi-label data. These two memories are especially effective for multi-label data streams given the dynamic frequencies and imbalance of the labelsets. The short-term memory classifies the most frequent and recent labelsets whereas the long-term remembers the infrequent and recurrent labelsets. Figure 1 shows the architecture of ML-SAM-kNN.

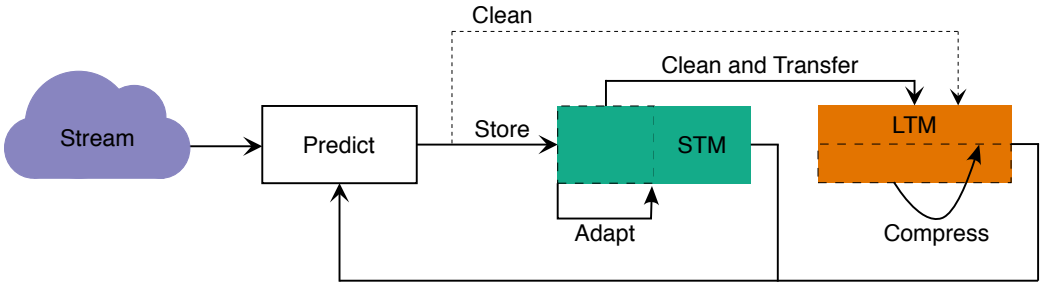


Figure 1: ML-SAM-kNN with short- and long-term memories.

3.1. Adaptation of the STM

The STM is a sliding window that at time t contains the most recent m instances, formally:

$$M_{ST_m} = \{(\mathbf{x}_{t-m+1}, Y_{t-m+1}), ..., (\mathbf{x}_t, Y_t)\}$$

where $Y = \{y_0, \dots y_L\}$ is the set of labels associated with the input vector $\mathbf{x}$. With the addition of each instance, different sized windows, $M_{ST_{m'}}$ are evaluated to ensure that the STM contains only the most current concept, where $m' \in \{m, m/2, m/4 \dots\}$ and $m' \geq m_{min}$, the minimum size of the STM. For the single-label SAM-kNN, evaluation was done using the interleaved test-train error, a measure of prediction accuracy. With multi-label data, where it is possible for a prediction to be partially correct, it is typical for a variety of metrics to be used for evaluation, often the Hamming score, subset accuracy, accuracy, precision, recall and F-measure (Madjarov et al., 2012). We evaluated each of the STM windows based on their Hamming score, a measure of per-label accuracy defined as:

$$Hamming\ score = \frac{1}{NL} \sum_{i=0}^{N} \sum_{l=0}^{L} \mathbb{1} \mid y_l = z_l,\ y_l \in Y_i,\ z_l \in Z_i$$

where N is the total number of instances, L is the number of labels, Y_i is the true labelset and Z_i is the predicted labelset. The window $M_{m'}$ with the highest Hamming score is used at time $t+1$. The set of instances discarded from the STM, $O = M_{ST_m} \backslash M_{ST_{m'}}$, are cleaned and transferred to the LTM.

3.2. Cleaning and Transfer

Instances in the LTM, or those being transferred to the LTM, are cleaned with respect to the STM to ensure that they are not in direct conflict with the current concept. To clean a set A with respect to an instance $s = (\mathbf{x}, Y)$ in the STM, a threshold θ_l for each label $l \in L$ is defined as:

$$\theta_l = max\{d(s, s_i) \mid s_i \in N_k(STM),\ y_l(s_i) = y_l(s)\}$$

where $N_k(STM)$ is the k nearest neighbors of s in the STM. If $y_l(s_i) \neq y_l(s)$ for all $s_i \in N_k(STM)$, then $\theta_l = -1$. Instances s_j in A that are inconsistent with s are added to the set of instances to be removed, $\hat{A}$ such that:

$$\hat{A} = \{s_j \mid s_j \in N_k(A),\ \exists l \text{ such that } d(s, s_j) \leq \theta_l,\ y_l(s_j) \neq y_l(s)\}$$

where $N_k(A)$ is the k nearest neighbors of s in A. The final cleaned set $A_{clean} = A \backslash \hat{A}$.

Whenever the STM is shrunk, the set O of discarded instances is cleaned with respect to every instance s in the STM and the resulting cleaned set is added to the LTM. To keep the LTM consistent with the STM, with the addition of each new instance the LTM is cleaned with respect to only the latest instance s_t.

3.3. Compression of the LTM

To retain information from past concepts, instances in the LTM are not faded out or discarded with time. To keep the LTM from growing indefinitely, it is compressed whenever the

size of the combined memories exceeds a maximum C_{max}. Similarly to single-label SAM-kNN, compression was done using kMeans++ clustering. However, as each multi-label instance is associated with more than one class, instances were grouped prior to clustering by labelset Y, rather than by class. Formally,

$$M_{LT_Y} = \{s_i \mid s_i \in M_{LT},\, Y_i = Y\}$$

where M_{LT} is the set of instances in the LTM and M_{LT_Y} is the set of instances in the LTM with labelset Y. For each unique labelset Y, $|M_{LT_Y}|/2$ clusters were found and a set of instances $\tilde{M}_{LT_Y}$ was created from the instances $\tilde{s}_i = (\tilde{\mathbf{x}}_i, Y)$, where $\tilde{\mathbf{x}}_i$ was a cluster centroid. After compression, the LTM consists of the union of all $\tilde{M}_{LT_Y}$.

Using the unique labelsets allows us to compress the LTM evenly without favoring any specific labels or labelsets. It is worth noting that the number of possible labelsets will grow exponentially with the number of labels. In extreme situations where the possible number of labelsets nears or reaches the number of instances in the LTM, the compression will become ineffective. However, in most cases the number of actual unique labelsets is very small compared to the number of possible unique labelsets, making this an unlikely problem to encounter in practice.

3.4. Prediction

To predict the labelset for each incoming instance, each of the LTM, the STM and combined union of the two induce a simple multi-label kNN classifier that uses a majority-vote among the k nearest neighbors for each label independently. The classifiers were ranked by their Hamming score, the final prediction being made by the classifier with the highest.

4. Experimental study

4.1. Experimental set-up

Table 1 shows the information of the 23 multi-label datasets, including the cardinality and density of the labels. Datasets with low density have a predominance of negative labels, then being highly imbalanced. They were selected from the KDIS multi-label dataset repository[1]. Table 2 shows the algorithms and their main parameters, implemented in MOA (Bifet et al., 2010). Experiments were run on an Intel Xeon CPU E5-2690v4 with 128 GB of memory.

4.2. Results and discussion

Tables 3 and 4 show the subset accuracy and F-measure results of all algorithms on the 23 datasets, respectively. Due to the limited space, results for all metrics including Hamming score, accuracy, precision, and recall are available on the web[2]. For multi-label instances, accuracy, defined as

$$Accuracy = \frac{1}{N}\sum_{i=0}^{N}\frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|}$$

[1] KDIS multi-label dataset repository: http://www.uco.es/kdis/mllresources

[2] ML-SAM-kNN website for code and additional results: http://people.vcu.edu/~acano/ML-SAM-kNN/

Table 1: Datasets and their characteristics.

Dataset	Instances	Attributes	Labels	Cardinality	Density
20NG	19300	1006	20	1.029	0.051
Bibtex	7395	1836	159	2.402	0.015
Birds	645	260	19	1.014	0.053
Bookmarks	87860	2150	208	2.028	0.010
Corel16k001	13770	500	153	2.859	0.019
Corel5k	5000	499	374	3.522	0.009
Emotions	593	72	6	1.868	0.311
Enron	1702	1001	53	3.378	0.064
EukaryotePseAAC	7766	440	22	1.146	0.052
Eurlex-sm	19350	5000	201	2.213	0.011
Flags	194	19	7	3.392	0.485
Genbase	662	1186	27	1.252	0.046
Imdb	120900	1001	28	2.000	0.071
Mediamill	43910	120	101	4.376	0.043
Medical	978	1449	45	1.245	0.028
Nuswide_cVLAD	269600	129	81	1.869	0.023
PlantPseAAC	978	440	12	1.079	0.090
Reuters-K500	6000	500	103	1.462	0.014
Scene	2407	294	6	1.074	0.179
Tmc2007-500	28600	500	22	2.220	0.101
VirusGO	207	749	6	1.217	0.203
Water-quality	1060	16	14	5.073	0.362
Yeast	2417	103	14	4.237	0.303

Table 2: Algorithms and their parameters.

Reference	Acronym	Algorithm	Parameters
Bifet et al. (2010)	MLS	Majority Labelset	none
Read et al. (2016)	BRU	Binary Relevance Updateable	learner: HoeffdingTree
Read et al. (2011)	CCU	Classifier Chains Updateable	learner: HoeffdingTree
Read et al. (2008)	PSU	Pruned Sets Updateable	learner: HoeffdingTree
Read et al. (2016)	RTU	Ranking Threshold Updateable	learner: HoeffdingTree
Oza (2005)	BMLU	On-line bagging of Oza and Russell	learner: HoeffdingTree components: 10
Osojnik et al. (2017)	ISOUPT	Structured Output Prediction Tree	default
Zhang and Zhou (2007)	ML-kNN	Multi-label k Nearest Neighbor	k: 10 window: 1000
	ML-SAM-kNN	Multi-label Self Adjusting Memory k Nearest Neighbor	k: 5 maxSTM: 400 maxLTM: 600

uses the Jaccard index to measure the similarity between the true labelset Y_i an the predicted labelset Z_i, whereas subset accuracy is a very strict metric and allows us to evaluate how frequently the whole labelset is predicted correctly, formally

$$Subset\ accuracy = \frac{1}{N}\sum_{i=0}^{N} \mathbb{1} \mid Y_i = Z_i$$

Table 3: Results for subset accuracy.

Subset Accuracy	MLS	BRU	CCU	PSU	RTU	BMLU	ISOUPT	ML-kNN	ML-SAM-kNN
20NG	0.2682	0.2694	0.2694	0.2186	0.2664	0.2694	**0.4130**	0.3783	0.3249
Bibtex	0.0632	0.0920	0.0974	0.1091	**0.1304**	0.0990	0.0011	0.0270	0.0172
Birds	0.4550	0.4348	0.4379	0.4550	0.3478	0.4332	0.4441	0.4720	**0.4752**
Bookmarks	0.0692	0.1142	0.1264	**0.1802**	0.1670	0.0966	0.1061	0.1344	0.1316
Corel16k001	0.0174	0.0214	0.0291	0.0052	0.0045	0.0221	0.0067	0.0097	**0.0713**
Corel5k	0.0092	0.0088	0.0096	0.0114	0.0002	0.0086	0.0002	0.0046	**0.0200**
Emotions	0.1216	0.1909	0.1959	0.1166	0.1368	0.1993	0.0811	0.2483	**0.2753**
Enron	0.2510	0.2510	0.2510	0.2510	0.2040	0.2510	0.2540	0.2681	**0.2792**
EukaryotePseAAC	0.2027	0.4129	0.4392	0.1552	0.0979	0.4030	0.2900	0.2968	**0.6945**
Eurlex-sm	0.0537	0.1041	0.1146	**0.1520**	0.0411	0.0082	0.0006	0.0617	0.0423
Flags	**0.1295**	0.0984	0.1036	0.1244	0.0000	0.0933	0.0311	0.0829	0.0725
Genbase	0.2481	0.4947	0.4932	0.2481	0.1150	0.3918	0.0272	0.8079	**0.8744**
Imdb	0.4041	0.4041	0.4041	0.0825	0.4040	0.4041	0.4051	**0.4063**	0.4055
Mediamill	0.0537	0.0575	0.0829	0.0542	0.0394	0.0708	0.0779	0.0879	**0.1476**
Medical	0.1576	0.3460	0.3367	0.1556	0.2958	0.3183	0.0041	**0.3982**	0.3961
Nuswide_cVLAD	0.2236	0.2057	0.2325	0.1264	0.2124	0.2439	0.2253	0.2367	**0.2818**
PlantPseAAC	0.2805	0.1904	0.2129	0.2805	0.2129	0.1965	0.2528	0.1648	**0.3971**
Reuters-K500	**1.0000**	**1.0000**	**1.0000**	0.8913	**1.0000**	**1.0000**	0.9998	0.9998	**1.0000**
Scene	0.1663	0.3013	0.3803	0.3437	0.6475	0.3408	0.1500	0.5191	**0.8184**
Tmc2007-500	0.0866	0.2173	0.2349	0.2340	0.1310	**0.2425**	0.1174	0.1730	0.1575
VirusGO	0.2573	0.2767	0.2767	0.2524	0.2184	0.3641	0.2136	0.4951	**0.5146**
Water-quality	0.0085	0.0047	0.0038	0.0085	0.0057	0.0038	**0.0151**	0.0123	0.0142
Yeast	0.0902	0.0844	0.1502	**0.1618**	0.0000	0.1395	0.0459	0.1366	0.1428
Average	0.2007	0.2426	0.2558	0.2008	0.2034	0.2435	0.1810	0.2792	**0.3284**

Precision and recall are defined as

$$Precision = \frac{1}{N}\sum_{i=0}^{N}\frac{|Y_i \cap Z_i|}{|Y_i|}$$

$$Recall = \frac{1}{N}\sum_{i=0}^{N}\frac{|Y_i \cap Z_i|}{|Z_i|}$$

F-measure represents the harmonic mean of precision and recall, formally

$$F\text{-}measure = \frac{1}{N}\sum_{i=0}^{N}\frac{2 \times |Y_i \cap Z_i|}{|Y_i| + |Z_i|}$$

ML-SAM-kNN obtains the best subset accuracy and F-measure on 13 of the 23 datasets, achieving averaged results significantly better than the compared algorithms. ML-kNN obtains the second best averaged results whereas ISOUPT obtains the worst average subset accuracy, and RTU the worst average F-measure.

Table 5 shows the average results for all metrics and algorithms. ML-SAM-kNN also obtains the best results for Hamming score, accuracy, precision, and recall. However, it does so at the cost of a higher computational complexity and therefore it achieves runtimes comparable to BRU, CCU, ISOUPT, and ML-kNN. Nevertheless, it is faster than the bagging ensemble BMLU, which performs slowest. MLS is the fastest method as it does not

Table 4: Results for F-Measure.

F-Measure	MLS	BRU	CCU	PSU	RTU	BMLU	ISOUPT	ML-kNN	ML-SAM-kNN
20NG	0.0332	0.0000	0.0000	**0.3892**	0.0026	0.0000	0.3084	0.2921	0.1994
Bibtex	0.0669	0.2236	0.2248	0.1441	**0.2884**	0.2264	0.0360	0.1052	0.1157
Birds	0.0000	0.0021	0.0020	0.0000	0.0517	0.0133	0.0102	0.0767	**0.0874**
Bookmarks	0.0730	0.1307	0.1354	0.1994	**0.2180**	0.1006	0.1065	0.1472	0.1414
Corel16k001	0.0585	0.1197	0.1273	0.0691	0.0464	0.1047	0.0191	0.0457	**0.2157**
Corel5k	0.0420	0.1160	0.1285	0.0587	0.0741	0.1064	0.0146	0.0481	**0.1948**
Emotions	0.2914	0.5460	0.5480	0.2837	0.4378	0.5436	0.2636	0.5435	**0.5972**
Enron	0.0000	0.0000	0.0000	0.0000	0.0075	0.0000	0.0446	**0.1391**	0.1109
EukaryotePseAAC	0.2607	0.5229	0.5159	0.2160	0.1159	0.4996	0.3182	0.3549	**0.7808**
Eurlex-sm	0.0823	**0.2701**	0.2636	0.2364	0.1551	0.0228	0.0010	0.1889	0.1521
Flags	0.6171	**0.6941**	0.6919	0.5960	0.0056	0.6605	0.6427	0.6349	0.6263
Genbase	0.2499	0.5186	0.5139	0.2499	0.1157	0.4088	0.0272	0.8676	**0.9163**
Imdb	0.0000	0.0000	0.0000	**0.1036**	0.0000	0.0000	0.0055	0.0017	0.0032
Mediamill	0.4277	0.4881	0.4651	0.4259	0.0000	0.4987	0.5080	0.5063	**0.5579**
Medical	0.2317	0.4252	0.4160	0.2296	0.4004	0.4074	0.0048	**0.5033**	0.4937
Nuswide_cVLAD	0.0029	0.1326	0.1242	0.2557	0.0181	0.1177	0.0084	0.1494	**0.2698**
PlantPseAAC	0.2900	0.2409	0.2606	0.2886	0.2197	0.2311	0.2610	0.1755	**0.4195**
Reuters-K500	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Scene	0.1850	0.4260	0.4896	0.4101	0.6888	0.4747	0.1616	0.5765	**0.8630**
Tmc2007-500	0.2167	0.6120	0.6147	0.5697	0.4158	**0.6298**	0.4304	0.5190	0.5041
VirusGO	0.3390	0.4600	0.4529	0.3341	0.2888	0.6514	0.2694	0.6348	**0.6883**
Water-quality	0.2453	0.4267	0.4252	0.2219	0.0000	0.4390	0.4816	0.4835	**0.5041**
Yeast	0.5196	0.5937	**0.6131**	0.5572	0.0000	0.6094	0.5061	0.5918	0.5627
Average	0.1840	0.3021	0.3049	0.2539	0.1544	0.2933	0.1926	0.3298	**0.3915**

Table 5: Average results of the algorithms for all metrics.

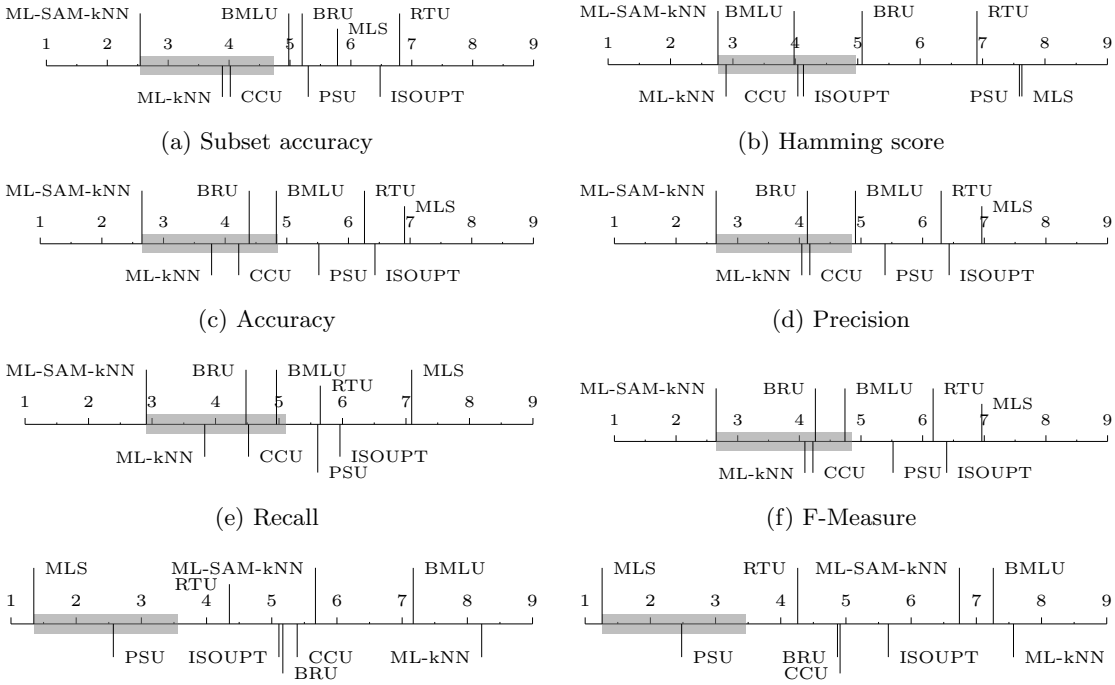
Average	MLS	BRU	CCU	PSU	RTU	BMLU	ISOUPT	ML-kNN	ML-SAM-kNN
Subset accuracy	0.2007	0.2426	0.2558	0.2008	0.2034	0.2435	0.1810	0.2792	**0.3284**
Hamming score	0.8787	0.9105	0.9113	0.8842	0.8936	0.9116	0.9084	0.9192	**0.9224**
Accuracy	0.1574	0.2586	0.2636	0.2195	0.1380	0.2513	0.1602	0.2889	**0.3478**
Precision	0.1760	0.2999	0.3018	0.2430	0.1382	0.2918	0.1786	0.3182	**0.3845**
Recall	0.2103	0.3412	0.3424	0.2927	0.1977	0.3313	0.2391	0.3788	**0.4350**
F-Measure	0.1840	0.3021	0.3049	0.2539	0.1544	0.2933	0.1926	0.3298	**0.3915**
Evaluation time (s)	**64**	10071	12008	95	2404	25445	17184	17404	16394
Model cost (RAM-Hours)	**2.0E-5**	8.5E-1	1.4E+0	2.5E-4	1.3E-1	2.7E+0	2.1E+0	4.9E-1	8.8E-1

infer any model and simply predicts the most frequent labelset. Table 6 shows the ranks of the algorithms according to Friedman, i.e., the best performing algorithm for a dataset is given rank 1, the next best given rank 2, and so on, and ranks are averaged across all datasets. This allows us to evaluate the relative performance of algorithms when compared to each other. Finally, the meta-rank averages the ranks across all metrics and provides an overall idea of how good each algorithm is, taking into account both the quality and performance metrics.

Figure 2 shows the Bonferroni–Dunn test diagrams for all metrics. The test identifies statistically significant differences in a multiple-algorithm comparison. It assumes that two classifiers are significantly different if their rank differs by at least some critical distance. The critical distance for $\alpha = 0.05$ is 2.20. The figure highlights the critical distance (in gray) between the best algorithm and the others. Algorithms that fall out of this area are

Table 6: Ranks of the algorithms for all metrics.

Ranks	MLS	BRU	CCU	PSU	RTU	BMLU	ISOUPT	ML-kNN	ML-SAM-kNN
Subset accuracy	5.78	5.20	4.02	5.30	6.80	4.98	6.48	3.89	**2.54**
Hamming score	7.63	5.07	4.04	7.59	6.91	3.98	4.13	2.89	**2.76**
Accuracy	6.91	4.39	4.22	5.52	6.26	4.83	6.43	3.78	**2.65**
Precision	6.96	4.13	4.17	5.39	6.30	4.91	6.43	4.04	**2.65**
Recall	7.09	4.48	4.52	5.61	5.65	4.96	5.96	3.83	**2.91**
F-Measure	6.96	4.26	4.22	5.52	6.17	4.74	6.39	4.09	**2.65**
Evaluation time (s)	**1.35**	5.17	5.39	2.57	4.35	7.17	5.11	8.22	5.67
Model cost (RAM-Hours)	**1.26**	4.87	4.91	2.48	4.26	7.26	5.65	7.57	6.74
Meta-Rank	5.49	4.70	4.44	5.00	5.84	5.35	5.82	4.79	**3.57**



(a) Subset accuracy (b) Hamming score

(c) Accuracy (d) Precision

(e) Recall (f) F-Measure

(g) Evaluation time (h) Model cost

Figure 2: Bonferroni–Dunn test for all metrics.

claimed to perform statistically worse than the control algorithm. The test indicates that only ML-kNN and CCU cannot be claimed as significantly different for all quality metrics, the others being statistically worse for all or some of the quality metrics.

On the other hand, the Wilcoxon test conducts pairwise comparisons of algorithms to identify significant differences. Table 7 shows the p-values of the Wilcoxon test when comparing ML-SAM-kNN vs the other algorithms. A p-value < 0.05 indicates significant differences between the two methods compared. According to this test, there are significant differences between ML-SAM-kNN and ML-kNN for all quality metrics except for Hamming score and recall, and there are differences with CCU for all metrics.

Table 7: Wilcoxon test for all metrics (p-values).

ML-SAM-kNN vs.	MLS	BRU	CCU	PSU	RTU	BMLU	ISOUPT	ML-kNN
Subset accuracy	7.34E-4	7.34E-4	2.30E-2	5.41E-3	5.79E-4	4.28E-3	6.03E-5	3.84E-2
Hamming score	6.41E-5	1.36E-2	3.65E-2	7.87E-6	7.41E-5	1.73E-2	2.42E-2	1.00E00
Accuracy	4.95E-5	1.94E-2	1.78E-2	1.03E-2	2.77E-4	3.48E-3	2.77E-4	4.41E-2
Precision	4.30E-5	1.36E-2	2.12E-2	1.13E-2	2.15E-4	3.86E-3	2.15E-4	2.51E-2
Recall	4.30E-5	1.36E-2	8.55E-3	8.55E-3	1.64E-3	4.74E-3	2.04E-3	2.42E-1
F-Measure	4.30E-5	1.13E-2	1.24E-2	8.55E-3	3.55E-4	2.82E-3	2.77E-4	3.77E-2
Evaluation time (s)	2.38E-7	6.65E-1	6.87E-1	4.77E-7	9.15E-3	2.59E-1	2.12E-2	5.22E-2
Model cost (RAM-Hours)	2.38E-7	2.54E-2	2.33E-2	2.38E-7	6.03E-5	7.49E-2	5.01E-1	9.41E-1

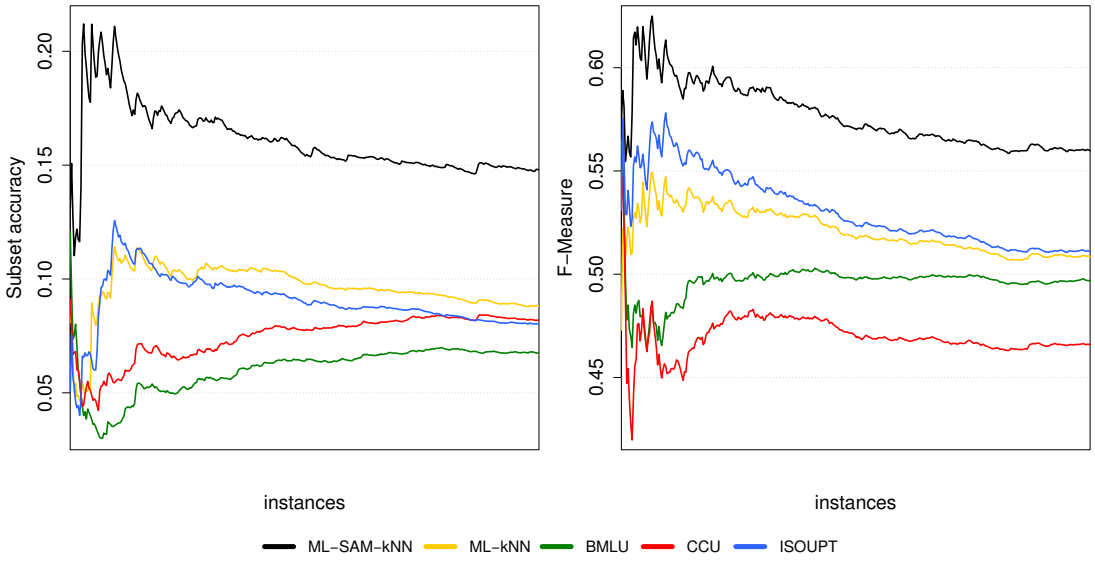


Figure 3: Subset accuracy and F-Measure results on the Mediamill dataset.

Figures 3 and 4 show the evolution of the subset accuracy and F-measure through the stream for two of the datasets, Mediamill and Nuswide_cVLAD, respectively. For simplification of the plot, only the top five algorithms with the best ranks are displayed. Detailed plots for all algorithms on all datasets are available at the referenced website[2]. In the beginning of the stream, having less data available makes for greater fluctuations in performance. With the progress of the stream, the performance stabilizes and bumps are due to drifts in the data distribution. In the long run, ML-SAM-kNN demonstrates significantly better performance than the other algorithms. The ever-decreasing performance of ISOUPT in the two datasets is noteworthy, showing that it is not capable of modeling the stream correctly and adapting to changes.

Figures 5 and 6 analyze the performance of ML-SAM-kNN, ML-kNN (sliding window), and the individual predictions of the short-term (STM), long-term (LTM), and combined memories (CM). Results indicate that the STM is adapting quickly to changes in the stream achieving the best results, whereas the LTM provides significantly worse and outdated

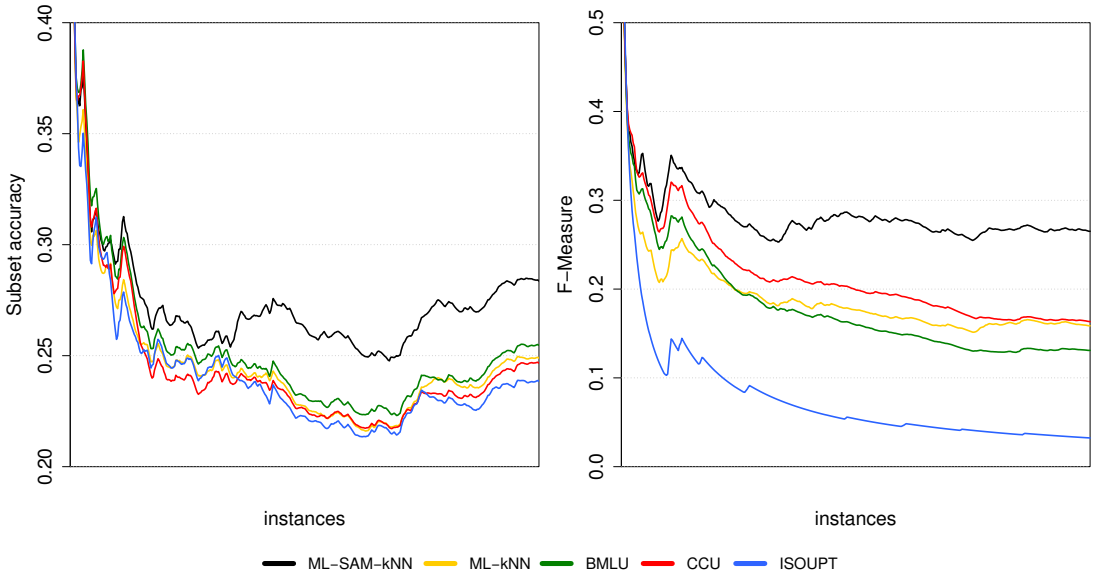


Figure 4: Subset accuracy and F-Measure results on the Nuswide_cVLAD dataset.

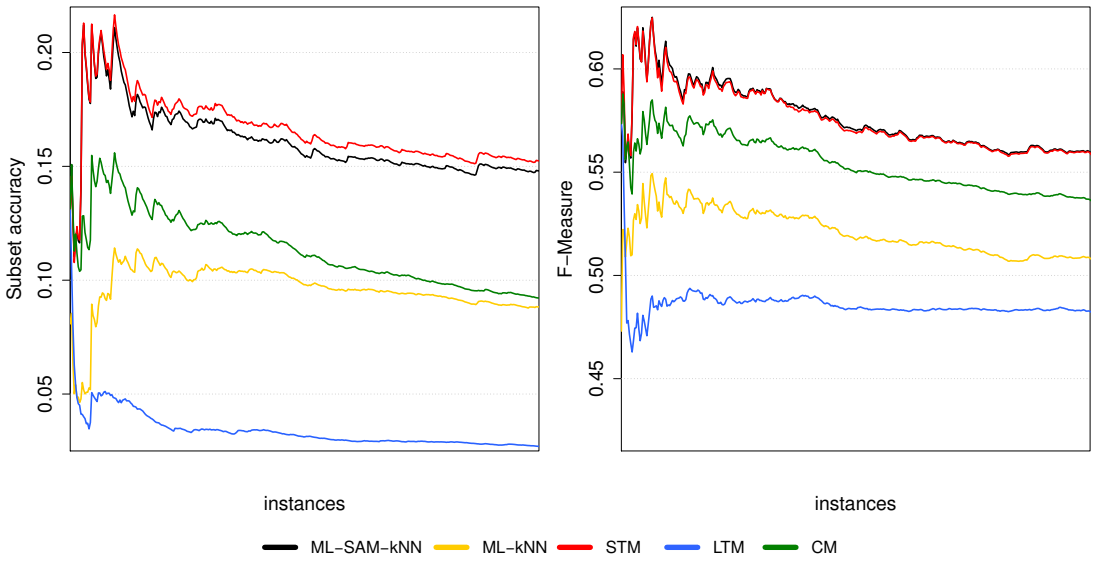


Figure 5: Performance of STM, LTM, and CM memories on the Mediamill dataset.

predictions. Results from the combined memory do not significantly improve upon those from the LTM. Therefore, ML-SAM-kNN is highly influenced by the speed of the drift and the STM effectively handles such rapid changes. However, the poor performance of the LTM as compared with the sliding window of ML-kNN indicates that the compression and clustering strategy employed to compact instances is not appropriate for multi-label data and further analysis on compacting multi-label instances for the LTM is required.

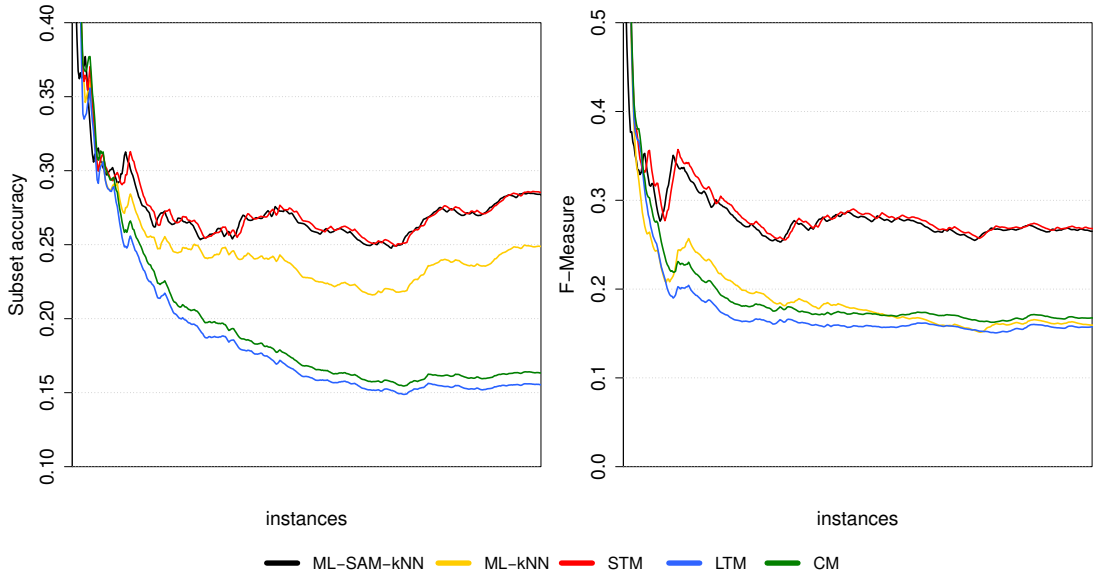


Figure 6: Performance of STM, LTM, and CM memories on the Nuswide_cVLAD dataset.

5. Conclusions

This paper introduced ML-SAM-kNN, a multi-label classifier using a self-adjusting memory for drifting data streams. A short-term memory is employed to maintain the most recent examples, while a long-term memory remembers historical and infrequent data. Pairing this memory architecture with a simple multi-label kNN classifier, ML-SAM-kNN is designed to cope with multi-label data streams undergoing various and mixed concept drift.

ML-SAM-kNN has been experimentally compared with eight multi-label classifiers on 23 datasets. It performs demonstrably better than its peers, exhibiting statistically better results against all compared algorithms for nearly all metrics and achieving the highest average result and average rank for all quality metrics. While this performance has come at the expense of runtime, ML-SAM-kNN performs comparably with the top ranked algorithms.

In future, we plan to develop ML-SAM-kNN to explicitly handle label imbalance and adjust the memory architecture to account for label dependencies when selecting the best instances to preserve, as well as seeking ways to improve the speed and modify the LTM compression so that it is suitable for imbalanced multi-label data.

Acknowledgments

This research was partially supported by the 2018 VCU Presidential Research Quest Fund and an Amazon AWS Machine Learning Research award.

References

Albert Bifet and Ricard Gavaldà. Learning from time-changing data with adaptive windowing. In *SIAM Int. Conf. on Data Mining*, pages 443–448, 2007.

Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. MOA: massive online analysis. *Journal of Machine Learning Research*, 11:1601–1604, 2010.

Alberto Cano, Jose Maria Luna, Eva Gibaja, and Sebastian Ventura. LAIM discretization for multi-label data. *Information Sciences*, 330:370–384, 2016.

Francisco Charte, Antonio J Rivera, María J del Jesus, and Francisco Herrera. Addressing imbalance in multilabel classification: Measures and random resampling algorithms. *Neurocomputing*, 163:3–16, 2015.

Amanda Clare and Ross D King. Knowledge discovery in multi-label phenotype data. In *Eur. Conf. on Principles of Data Mining and Knowledge Discovery*, pages 42–53, 2001.

Gregory Ditzler, Manuel Roveri, Cesare Alippi, and Robi Polikar. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25, 2015.

Mohamed Medhat Gaber. Advances in data stream mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):79–85, 2012.

João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4): 44:1–44:37, 2014.

Eva Gibaja and Sebastian Ventura. A tutorial on multilabel learning. *ACM Computing Surveys*, 47(3):52:1–52:38, 2015.

Heitor Murilo Gomes, Jean Paul Barddal, Fabrício Enembreck, and Albert Bifet. A survey on ensemble learning for data stream classification. *ACM Computing Surveys*, 50(2): 23:1–23:36, 2017.

Jorge Gonzalez, Alberto Cano, and Sebastian Ventura. Large-scale multi-label ensemble learning on spark. In *IEEE Trustcom/BigDataSE/ICESS*, pages 893–900, 2017.

Jorge Gonzalez, Sebastian Ventura, and Alberto Cano. Distributed Nearest Neighbor Classification for Large-Scale Multi-label Data on Spark. *Future Generation Computer Systems*, 87:66–82, 2018.

Imen Khamassi, Moamar Sayed-Mouchaweh, Moez Hammami, and Khaled Ghedira. Discussion and review on evolving data streams and concept drift adapting. *Evolving Systems*, 9(1):1–23, 2018.

Bartosz Krawczyk, Leandro L. Minku, João Gama, Jerzy Stefanowski, and Michał Woźniak. Ensemble learning for data stream analysis: A survey. *Inf. Fusion*, 37:132–156, 2017.

Viktor Losing, Barbara Hammer, and Heiko Wersing. KNN classifier with self adjusting memory for heterogeneous concept drift. In *IEEE Int. Conf. on Data Mining*, pages 291–300, 2016.

Viktor Losing, Barbara Hammer, and Heiko Wersing. Tackling heterogeneous concept drift with the Self-Adjusting Memory (SAM). *Knowledge and Inf. Sys.*, 54(1):171–201, 2018.

Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and Sašo Džeroski. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, 45(9): 3084–3104, 2012.

Aljaž Osojnik, Panče Panov, and Sašo Džeroski. Multi-label classification via multi-target regression on data streams. *Machine Learning*, 106(6):745–770, 2017.

Nikunj C Oza. Online bagging and boosting. In *IEEE Int. Conf. on Systems, Man and Cybernetics*, volume 3, pages 2340–2345, 2005.

Jesse Read, Bernhard Pfahringer, and Geoff Holmes. Multi-label classification using ensembles of pruned sets. In *IEEE Int. Conf. on Data Mining*, pages 995–1000, 2008.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine learning*, 85(3):333–359, 2011.

Jesse Read, Albert Bifet, Geoff Holmes, and Bernhard Pfahringer. Scalable and efficient multi-label classification for evolving data streams. *Mach. Learn.*, 88(1-2):243–272, 2012.

Jesse Read, Peter Reutemann, Bernhard Pfahringer, and Geoff Holmes. MEKA: A multi-label/multi-target extension to Weka. *J. of Machine Learning Research*, 17(21):1–5, 2016.

Przemyslaw Skryjomski, Bartosz Krawczyk, and Alberto Cano. Speeding up k-Nearest Neighbors Classifier for Large-Scale Multi-Label Learning on GPUs. *Neurocomputing*, In press, 2018.

Ricardo Sousa and João Gama. Multi-label classification from high-speed data streams with adaptive model rules and random rules. *Progress in Artificial Int.*, pages 1–11, 2018.

Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *Int. Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.

Grigorios Tsoumakas and Ioannis Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In *Eur. Conf. on Machine Learning*, pages 406–417, 2007.

Min-Ling Zhang and Zhi-Hua Zhou. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, 2007.

Peng Zhang, Byron J. Gao, Xingquan Zhu, and Li Guo. Enabling fast lazy learning for data streams. In *IEEE Int. Conf. on Data Mining*, pages 932–941, 2011.