Aging and rejuvenating strategies for fading windows in multi-label classification on data streams

Martha Roseberry
Virginia Commonwealth University
Richmond, Virginia
mroseberry@vcu.edu

Sašo Džeroski
Jožef Stefan Institute
Jožef Stefan International Postgraduate School
Ljubljana, Slovenia
saso.dzeroski@ijs.si

Albert Bifet
AI Institute, University of Waikato
Hamilton, New Zealand
LTCI, Télécom Paris, IP Paris
Paris, France
abifet@waikato.ac.nz

Alberto Cano
Virginia Commonwealth University
Richmond, Virginia
acano@vcu.edu

ABSTRACT

Combining the challenges of streaming data and multi-label learning, the task of mining a drifting, multi-label data stream requires methods that can accurately predict labelsets, adapt to various types of concept drift and run fast enough to process each data point before the next arrives. To achieve greater accuracy, many multi-label algorithms use computationally expensive techniques, such as multiple adaptive windows, with little concern for runtime and memory complexity. We present Aging and Rejuvenating kNN (ARkNN) which uses simple resources and efficient strategies to weight instances based on age, predictive performance, and similarity to the incoming data. We break down ARkNN into its component strategies to show the impact of each and experimentally compare ARkNN to seven state-of-the-art methods for learning from multi-label data streams. We demonstrate that it is possible to achieve competitive performance in multi-label classification on streams without sacrificing runtime and memory use, and without using complex and computationally expensive dual memory strategies.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning algorithms**;

KEYWORDS

Machine Learning, Data Streams, Multi-label Learning

ACM Reference Format:
Martha Roseberry, Sašo Džeroski, Albert Bifet, and Alberto Cano. 2023. Aging and rejuvenating strategies for fading windows in multi-label classification on data streams. In *The 38th ACM/SIGAPP Symposium on Applied Computing (SAC '23), March 27 - March 31, 2023, Tallinn, Estonia.* ACM, Tallinn, Estonia, Article 4, 8 pages. https://doi.org/10.1145/3555776.3577625

SAC '23, March 27 - March 31, 2023, Tallinn, Estonia

ACM ISBN 978-1-4503-9517-5/23/03.
https://doi.org/10.1145/3555776.3577625

1 INTRODUCTION

As the amount of continuously generated data produced by everyday systems and devices increases, so does the need to accurately and efficiently mine data streams. Not only must models be able to make predictions for incoming data in real time, but they must also be able to incorporate new data and update in response to the constantly evolving nature of the stream [3]. Many methods have been proposed for responding to concept drift, including sliding windows, adaptive windows and evolving ensembles [9]. To improve accuracy, these methods often sacrifice simplicity and require substantial resources to run but resources are limited and as the data increases in complexity, their resource use becomes prohibitive. Multi-label streaming data [28] is a good example of complex data for which many proposed solutions require significant resources. It is common for multi-label classification methods to rely on multiple classifiers or multiple windows of short-term and long-term data. The complexity of the methods is dependent on the number of labels, which may be high [28].

In this paper, we present a series of resource efficient techniques to age and rejuvenate instances in a single data window. Each technique builds on the previous, with the final combined method, Aging and Rejuvenating kNN (ARkNN), weighting each data instance within the window based on its age, past performance, and similarity to the incoming data. Aging techniques allow us to decrease the importance of older and worse performing data, allowing ARkNN to respond to concept drift and prune data that contributes to error. Rejuvenation techniques allow us to keep beneficial instances without the complexity of multiple windows. We experimentally compare each added modification to demonstrate the impact of each technique on performance, runtime and memory use. The final ARkNN method is then experimentally compared to seven state-of-the-art multi-label streaming methods. We demonstrate that performance of ARkNN is better or comparable to that of state-of-the-art methods, while the average runtime of ARkNN is one fifth the next fastest model and its average memory use is 1/50th of the time of its nearest competitor.

The main contributions of this paper are:

- ARkNN: a fast and memory efficient algorithm for multi-label classification on data streams using instance aging and rejuvenation.
- A detailed analysis of the impact of six techniques to age, rejuvenate and prune data to respond to concept drift and improve performance with minimal resource cost.
- A thorough experimental study comparing the predictive performance, runtime, and memory use of ARkNN against seven state-of-the-art algorithms for multi-label classification on data streams across 37 datasets.

The rest of the paper is organized as follows. Background on data streams, multi-label learning and related windowing, aging and rejuvenation techniques is given in Section 2. A description of each of the six proposed techniques and how they build upon one another is detailed in Section 3. Section 4 describes the experimental setup, with experimental results and discussion presented in Section 5. Concluding remarks and future work are presented in Section 6.

2 BACKGROUND

Resource Efficient Data Mining

Most machine learning literature focuses on the accuracy and predictive power of machine learning algorithms. In many cases, however, computing resources are limited. As more data is collected and available to be mined, this problem is unlikely to diminish. In recent years, attention has been drawn to energy efficient data mining, particularly energy efficient deep learning. Others have suggested techniques for measuring energy and resources with the goal of making it easier for researchers to evaluate resource consumption when comparing methods. Mining streaming data is one scenario where resource efficiency is highly important, as the speed of the data can quickly make resources scant. However, less research works in this area focus on minimizing resource consumption [10].

Streaming Data

Streaming data refers to the situation where data is arriving as a potentially unbounded sequence. Formally, a data stream is an ordered sequence of data $S = \{s_1, s_2, \ldots\ s_t, \ldots\}$, where the data instance s_t arrives at time t. Streaming data presents both the problem of never having all the data available and having a limited amount of time to process instance s_t before instance s_{t+1} arrives and creates a backlog. Further complicating data streams, it is generally assumed that the stream is not static.

Concept Drift

A concept drift is a change in the distribution of the incoming instances that may cause a change in the decision boundaries [12]. A wide variety of strategies for dealing with concept drift have been proposed, usually broken into two categories - continuously evolving methods and concept drift detectors [9]. Drift detectors use an outside method to monitor the performance of the learner. As the stream continues, if the detector encounters a sufficient loss in performance, it will trigger the learner to update or re-train based on newer data. A benefit of concept drift detectors is that the model only re-trains upon the detection of drift, saving resources when the model is stable. A disadvantage is that drift detectors are most effective for abrupt drift, where the data distribution changes suddenly.

However, concept drift may also appear as gradual or incremental drift. During gradual drift, instances from both the old concept and the new concept appear for some time together, with instances from the new concept slowly becoming dominant. In incremental drift, the old concept slowly transitions into the new concept. Both take place more slowly than abrupt drift, without a sharp divide between the two concepts. This makes gradual and incremental drifts difficult for drift detectors to detect. To handle slower drifts, many models incorporate a method for continuously evolving.

Sliding, Fading and Adaptive Windows

Continuous methods typically involve mechanisms to update the classification model as new information arrives and forget older concepts [9]. One of the most basic methods is to use a first-in, first-out sliding window. In its most basic form, a sliding window is a window of size m in which each data instance is stored. Once the window is full, as the newest instance arrives at time s_t, the oldest instance in the window s_{t-m} is discarded. Using this mechanism, the model trained on the window slowly evolves with the stream, forgetting older concepts.

Sliding windows are an efficient method for forgetting older information, but while data is still in the window, all instances are valued equally. However, for a drifting stream, more recent information is often seen as more valuable. This leads to the definition of damped windows or fading windows [9]. In fading windows, a fading factor or decay function is used to weight older information, such that newer information has a higher weight and has a greater impact on the predictions. In this way, older instances in the window are not yet forgotten, but if the concept has shifted, the classifier will update more quickly because of the higher weight of new data. Fading windows are popular in pattern mining and finding frequent itemsets [7, 15, 26]. Fading or aging is also a common technique used in weighting ensembles of classifiers [11, 13] and fading windows can also be used to detect drift or evaluate classifiers [8, 20].

Forgetting mechanisms and drift detectors can also be used together. Adaptive size windows act as traditional sliding or fading windows, but monitor the incoming data to change the window size and abruptly forget information in the presence of abrupt concept drift [9]. Other methods use multiple windows. The popular adaptive windowing technique ADWIN monitors the data over two windows [5]. When a change is detected, the older window is abruptly dropped. Another mechanism uses two windows as short and long-term memories, summarizing past data into the long-term memory whenever the short-term memory is reduced due to a drift [14]. Multiple window methods can be very effective, but by requiring that multiple windows be maintained and updated, they often suffer from time and memory complexity.

Multi-Label Streaming Data

While complexity is a concern for all data stream mining, it is a particular challenge for multi-label classification of streaming data. Multi-label data refers to the situation where each data instance can be classified as belonging to multiple classes simultaneously. The classes are not mutually exclusive. In contrast to multi-class classification, each instance has the form $s = (\mathbf{x}, y)$ where y is the single relevant class, in multi-label classification $s = (\mathbf{x}, Y)$ where

$Y \in \{0,1\}^{|L|}$ and L is the set of all possible labels. Just as the number of instances and the number of features in traditional single-label classification are assumed to be large, the number of possible labels, $|L|$, may be huge, increasing the complexity of the problem.

Methods for multi-label learning are split into two categories - algorithm adaption methods and problem transformation methods. Problem transformation methods convert a multi-label problem into multiple single-label problems. On the other hand, algorithm adaptation methods adapt existing techniques for single-label data to multi-label data. Ensembles of classifiers are also popular for multi-label classification.

Most methods for multi-label classification on data streams prioritize predictive performance rather than minimizing complexity. Many of the state-of-the-art multi-label streaming classifiers use ensembles or multiple windows of data, which are resource-intensive [2, 16, 17, 19, 22]. However, when multi-label data arrives as a stream, it has all the issues associated with the velocity of the stream that single-label data has. Each arriving instance must be processed prior to the arrival of the next and in real-world scenarios, resources are likely to be limited.

Related Work

Beringer and Hüllermeier proposed an instance method for data stream mining that used changes to the window of stored data to update a classifier in response to concept drift [4]. While they consider the similarity and age of instances, their method is concerned with determining which instances should be removed from the window, not with the weighting of instances. Using weights to adapt classifiers is more common for ensemble methods. Woźniak et al. use accuracy weighted ensembles to adapt to concept drift [24]. Rejuvenation methods are much less common than aging and accuracy-based weighting. A few proposals have analyzed rejuvenation techniques for weighting base classifiers in ensembles [24, 25], but no work has proposed rejuvenating the instances themselves.

For multi-label data, Wang et al. proposed an ensemble of MLkNN classifiers using confidence, time, and distance to weight the base classifiers [21]. Roseberry et al. used a punitive method to remove instances with poor historic accuracy [18]. This was followed by with a method to enable and disable specific labels based on the most recent past performance [19].

3 PROPOSED TECHNIQUES

Sliding window model - SLI

As a baseline for comparing our proposed aging and rejuvenation techniques, the first technique uses a simple first-in, first-out sliding window. This model is denoted as SLI. For this model, a simple sliding window of size m is used. For the first m instances of the model, the window fills. As shown in Figure 1, once the window is full, as each new instance s_t arrives, the instance s_{t-m} is removed. Within the window, all instances are weighted equally.

For the prediction phase of SLI, a simple multi-label k-nearest neighbor classifier was used. To minimize computational complexity, our kNN uses a simple majority vote [1]. As an instance arrives, the k-nearest neighbors are found using the cosine distance, $D_C(s,n) = 1-S_C(s,n)$ where S_C is the cosine similarity of the instance s and the neighbor n. For each label l independently, the algorithms predicts

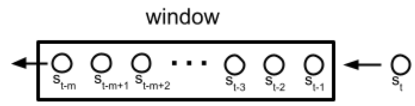

Figure 1: Sliding window.

label l is relevant if a majority of the nearest neighbors labelsets contain l. Formally, the relative frequency (rf) of the label among the set of nearest neighbors NN_k is defined as

$$rf = \frac{1}{k}\sum_{i=0}^{k} \mathbb{1} \mid (n_i \in NN_k,\ y_{i_l} = 1)$$

where y_{i_l} is the lth label of the ith nearest neighbor. Each predicted label z_l is defined as

$$z_l = \begin{cases} 1, & \text{if } rf \geq 0.5 \\ 0 & otherwise \end{cases}$$

The full algorithms for SLI is shown in Algorithm 1.

Algorithm 1: Sliding window model (SLI)

```
Require: k, m
while stream.hasMoreInstances() do
    s ← stream.nextInstance()
    votes[] ← {∅}^L
    for n ∈ window.neighbors(s, k) do
        for label l ∈ L do
            votes[l].add(n.l)
    for label l ∈ L do
        z(l) ← argmax(votes[l])
    window.add(s)
    if window.size() > m then
        window.removeOldest()
```

As a lazy learner, kNN is a good method to test the effectiveness of the techniques we are applying to the stored data. An advantage of the mechanisms presented here is that they act upon the data, and can be applied in many scenarios using different classifiers, which makes them flexible and applicable to different situations.

Fading window model - FAD

Our first mechanism for responding to concept drift is a fading window model, denoted FAD. Here, each instance is added to the window with weight $w = 1$. With the arrival of each new instances, the weight of each instance is multiplied by a constant fading factor $\delta \in [0,1]$. Using this simple mechanism, when the prediction for instance s_t is made, the oldest instance in the window, s_{t-m}, has the lowest importance and the newest, s_{t-1}, has the highest. As with the basic sliding window, when the window is full, the oldest instance is removed. Figure 2 illustrates the fading window, with instance weight depicted using shades of gray.

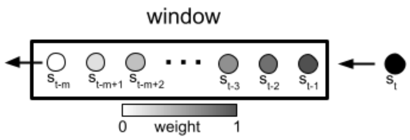

Figure 2: Fading window.

The prediction for FAD is performed by a kNN similar to that used with SLI, but here the weights of each instance are taken into consideration using weighted voting, such that

$$rf = \frac{1}{k}\sum_{i=0}^{k} w_{n_i} * \mathbb{1} \mid (n_i \in NN,\ y_{i_l} = 1)$$

where w_{n_i} is the weight of the ith nearest neighbor. The fading window mechanisms within ARkNN are shown in Algorithm 2, labeled *FAD*.

Query-based rejuvenation model - QBR

The query-based rejuvenation model, QBR, implements the first mechanism to rejuvenate instances. For both the sliding window model and the fading window model, an instance will always be removed from the window once m more recent instances were added. Aging instances out in this manner promotes more recent information and allows a model to adapt to concept drift. However, it is not always the case that the instance that has been in the window longest is the least-valuable instance.

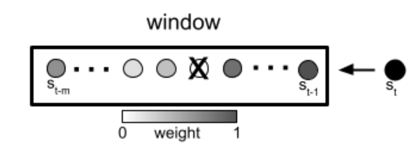

Figure 3: Rejuvenating window.

The QBR model uses a naive mechanism to rejuvenate the weight of any instance that is queried as a nearest neighbor while making a prediction. As in the fading window model, the weight of all instances slowly decreases over time, but the rejuvenation allows instances that have been utilized to reset their weights, making these instances more important to the model. In addition, rather than removing the instance that has been in the window longest when the window is full, QBR removes the instance with the lowest weight from the window. Figure 3 depicts a rejuvenating window. Rather than removing the oldest instance s_{t-m} on the left, the instance with the lowest weight is removed, regardless of its position in the window. Contributions for QBR in ARkNN are shown in Algorithm 2.

Accuracy-based rejuvenation model - ABR

Rejuvenating instances allows us to keep valuable information that is positively contributing to the model. However, just because an instance was queried as a neighbor, does not mean that it had a positive contribution. It's quite possible that one or more of the nearest neighbors had negative impact. This might be because the concept had shifted or that some or all of the nearest neighbors were no longer relevant. It might also happen that an instance is simply noise, in which case it will negatively impact the classifier as long as it is in the window. While we want to keep positively contributing instances in the window longer, we also want to remove instances from old concepts and noisy instances as quickly as possible.

The window used for ABR looks like the QBR window in Figure 3, except that the method of rejuvenation has changed. Rather than just rejuvenate an instance that has been queried, ABR uses the predictive performance of each of the queried instances to rejuvenate them. After having predicted the labelset of the test instance, we conduct a posterior analysis of the contribution of the queried instances. When an instance n is queried as a nearest neighbor, its labelset Y_n is checked against the labelset Y_s of the arriving instance s_t. For any label l, the neighbor n is contributing true values if $y_{l_n} \in Y_n$ equals $y_{l_s} \in Y_s$, such that the total number of true labels is:

$$true_n = \sum_{l=0}^{L} \mathbb{1} \mid (y_{l_n} = y_{l_s},\ y_{l_n} \in Y_n,\ y_{l_s} \in Y_s)$$

The accuracy of the neighbor n is defined as $accuracy_n = true_n/L$, where L is the number of possible labels.

The rejuvenation is computed using the accuracy of the neighbor n relative to the global accuracy of the window. The latter is computed prequentially as each prediction is made, comparing the predicted labelset to the true labelset. To rejuvenate n, its weight is first updated as follows

$$weight_n += accuracy_n - accuracy_{window}$$

then set to zero if negative, i.e., $weight_n = max(0, min(weight_n, 1))$, so that $weight_n \in [0, 1]$.

In this way, instances that are performing better than the window overall are rejuvenated. Their weight is increased and they will contribute more to future predictions and remain in the window longer. Conversely, instances that are preforming worse that the window overall will age. Their weight will decrease so they have less impact on future predictions. They will be removed from the window if they continue to negatively impact predictions, allowing the learner to respond to concept drift faster. The ABR sections of ARkNN are shown in Algorithm 2, denoted with ABR.

Pruning instances model- PRU

Lowering the weight of an instance reduces how much impact that instance has on future predictions. Despite the low weight, however, these instances are still in the window, using resources and potentially acting as nearest neighbors. The pruning instance model, PRU, adds a mechanism to prune the worst performing instances. As each instance arrives, the predicted labelset is determined and the weights of the neighbors are updated as in the ABR model. After the weight of each instance is faded, any instance with a weight below a given fitness factor f is removed from the window. If most instances are preforming well and pruning does not occur, the instance with lowest weight is removed when the window exceeds size m. Figure 4 illustrates a pruned window, with weights depicted as shades of gray.

The fading and accuracy-based rejuvenation mechanism ensure that the weights of older and more poorly performing instances will drop quickly. Pruning allows us to remove obsolete or noisy instances faster, helping to more quickly adapt the model to concept drift. In addition, pruning reduces the overall size of the window,

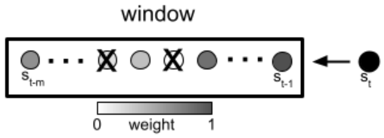

Figure 4: Pruned Window.

resulting in fewer distance computations and comparisons when making predictions and training the model, improving the runtime and memory use of the model. Pruning contributions to ARkNN are shown in Algorithm 2, noted as PRU.

Distance-weighted voting - DWV

The last mechanism added to ARkNN is distance weighted voting, which is a well-known tactic. When predicting the labelset for each incoming instance s, each vote is weighted not only by the weight of the nearest neighbor, computed as in ABR, but also by the distance of that neighbor from s. Thus, the vote takes into consideration not just the age and performance of the neighbor, but also its similarity to s. The distances between the neighbor n and the incoming instance s is computed as the cosine difference, $D_C(s,n) = 1 - S_C(s,n)$ where S_C is the cosine similarity and $D_C \in [0,2]$. We want to give closer neighbors a higher weight, so the weight is adjusted by a factor of $2 - D_C(s,n)$, such that

$$rf = \frac{1}{k}\sum_{i=0}^{k} w_{n_i} * (2 - D_C(s,n)) * \mathbb{1} \mid (n_i \in NN,\ y_{i_l} = 1)$$

The distance weighting contributions to ARkNN are shown in Algoritm 2, denoted DWV.

Aging and Rejuvenating kNN - ARkNN

The final ARkNN algorithm is given in Algorithm 2. Taking mechanisms from each iteration presented in Section 3, ARkNN includes the following mechanisms:

- a fading window reducing the weight of each instance over time by a fading factor δ as in FAD.
- accuracy-based rejuvenation of instances based on their accuracy relative to the window accuracy as in ABR.
- distance weighted voting as in DWV.
- instance pruning of all instances with a weight below a fitness threshold f as in PRU.
- removal of the instance with the lowest weight to maintain a window of size m as in QBR.

The time-complexity for computing the k-nearest neighbors is $O(mkd)$ where m is the maximum size of the window, k is the number of neighbor used, and d is the dimensionality of the data. To make a multi-label prediction using the weighted votes from the k neighbors takes $O(k|L|)$ time, where $|L|$ is the number of labels. The combined the complexity of ARkNN is $O(mkd + k|L|)$.

This is the same complexity as the baseline sliding window model (SLI) shown in Algorithm 1. While ARkNN employs multiple mechanisms to improve prediction capabilities and the classifiers ability to react to concept drift, all of these strategies are simple strategies that act on a single window of stored instances and are very resource efficient. Additionally, since these strategies act on the window of instances, they are highly flexible and could easily be incorporated into models using different base classifiers other than kNN.

Algorithm 2: Aging and Rejuvenating kNN - ARkNN

```
Require: k : number of neighbors, m : window size,
         δ : fading factor, f : fitness threshold
while stream.hasMoreInstances() do
    s ← stream.nextInstance()
    votes[] ← {∅}^L
    true[] ← {∅}^k                                   ▷ ABR
    σ[] ← {∅}^k                                      ▷ DWV
    for n ∈ window.neighbors(s, k) do
        σ[n] = 2 − distance(n, s)                    ▷ DWV
        for label l ∈ L do
            votes[l].add(n.l∗n.w∗σ[n])               ▷ FAD, DWV
            if n.l = s.l then true[n] + = 1;         ▷ ABR
        relativeAcc = true[n]/L − windowAcc          ▷ ABR
        n.w = n.w + relativeAcc; 0 ≤ w ≤ 1           ▷ ABR
    for label l ∈ L do
        z(l) ← argmax(votes[l])
    windowAcc.update()                               ▷ ABR
    for i ∈ window do
        i.w ∗ = δ                                    ▷ FAD
        if i.w < f then window.remove(i);            ▷ PRU
    window.add(s)
    if window.size() > m then
        worst ← argmin(window.getWeight)             ▷ QBR
        window.remove(worst)                         ▷ QBR
```

4 EXPERIMENTAL SETUP

This section introduces the experimental setup used to compare the proposed methods with the state of the art. The experiments are designed to answer the following research questions:

- **RQ1:** Does instance aging improve the predictions of the nearest neighbor classifier?
- **RQ2:** Do instance rejuvenation strategies allow us to retain relevant concepts in the stream?
- **RQ3:** Do instance pruning and distance-weighted voting improve the classifier's predictions?
- **RQ4:** Are the proposed strategies competitive against other state of the art methods, particularly short and long-term memory based classifiers?

Algorithms. Table 1 enumerates the strategies proposed and the state of the art algorithms. The source code for all methods is available at https://github.com/canoalberto/ARkNN to facilitate the reproducibility of the experiments. All window-based methods are evaluated with a window size of 1,000 instances.

Datasets. Table 2 shows the 37 multi-label datasets evaluated and their properties. These include the number of instances, features, labels, cardinality, and density. The lower the density the more sparse positive labels are in the dataset.

Table 1: Algorithms compared in the experiments.

	Ref	Acronym	Algorithm
Proposed methods	This paper	SLI	Sliding window
		FAD	Fading window with instance aging
		QBR	Query-based instance rejuvenation
		ABR	Accuracy-based instance rejuvenation
		PRU	ABR with instance pruning
		DWV	PRU with distance-weighted voting
		ARkNN	Aging and Rejuvenating kNN
State of the art	[27]	MLkNN	Multi-label kNN
	[17]	MLSAMkNN	ML Self-Adjusting Memory kNN
	[18]	MLSAMPkNN	ML Self-Adjusting Memory Punitive kNN
	[19]	MLSAkNN	ML Self-Adjusting kNN
	[2]	AESAKNNS	Adaptive Ensemble of SAkNN Subspaces
	[22]	OMK	Online Memory k-Means
	[23]	ODM	Online Dual Memory

Table 2: Multi-label datasets and their properties.

Dataset	Instances	Features	Labels	Cardinality	Density
20NG	19,300	1,006	20	1.0289	0.0514
Bibtex	7,395	1,836	159	2.4019	0.0151
Birds	645	260	19	1.0140	0.0534
Bookmarks	87,856	2,150	208	2.0281	0.0098
CAL500	502	68	174	26.0438	0.1497
CHD49	555	49	6	2.5802	0.4300
Corel16k	13,766	500	153	2.8587	0.0187
Corel5k	5,000	499	374	3.5220	0.0094
Emotions	593	72	6	1.8685	0.3114
Enron	1,702	1,001	53	3.3784	0.0637
Eukaryote	7,766	440	22	1.1456	0.0521
Eurlex-sm	19,348	5,000	201	2.2133	0.0110
Flags	194	19	7	3.3918	0.4845
Genbase	662	1,186	27	1.2523	0.0464
GnegativePseAAC	1,392	440	8	1.0460	0.1307
HumanPseAAC	3,106	440	14	1.1851	0.0847
Hypercube	100,000	100	10	1.0002	0.1000
Hypersphere	100,000	100	10	2.3138	0.2314
Imdb	120,919	1,001	28	1.9997	0.0714
Langlog	1,460	1,004	75	1.1801	0.0157
Mediamill	43,907	120	101	4.3756	0.0433
Medical	978	1,449	45	1.2454	0.0277
Nuswide-BoW	269,648	500	81	1.8685	0.0231
Nuswide-cVLAD	269,648	128	81	1.8685	0.0231
Ohsumed	13,929	1,002	23	1.6631	0.0723
PlantPseAAC	978	440	12	1.0787	0.0899
Reuters-K500	6,000	500	103	1.4622	0.0142
Scene	2,407	294	6	1.0740	0.1790
Slashdot	3,782	1,079	22	1.1809	0.0537
Stackex-chess	1,675	585	227	2.4113	0.0106
Tmc2007	28,596	500	22	2.2196	0.1009
VirusGO	207	749	6	1.2174	0.2029
Water-Quality	1,060	16	14	5.0726	0.3623
Yahoo-Society	14,512	31,802	27	1.6704	0.0619
Yahoo-Computers	12,444	34,096	33	1.5072	0.0457
Yeast	2,417	103	14	4.2371	0.3026
Yelp	10,806	671	5	1.6383	0.3277

Metrics. Dozens of metrics are used to evaluate the performance of multi-label classifiers [6]. The most representative metrics are subset accuracy, accuracy, and F-Measure. Given n instances and L labels, a true labelset $Y_i = \{y_{i1} \dots y_{iL}\}$ and a predicted labelset $Z_i = \{z_{i1} \dots z_{iL}\}$, the example-based metrics are defined as:

$$\textit{Subset accuracy} = \frac{1}{n}\sum_{i=0}^{n} \mathbb{1} \mid (Y_i = Z_i)$$

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

$$\textit{F-Measure} = \frac{1}{n}\sum_{i=0}^{n} \frac{2 \times |Y_i \cap Z_i|}{|Y_i| + |Z_i|}$$

Moreover, data stream algorithms are expected to be of low computational and memory complexity. Therefore, we must jointly assess the classification metrics, the runtime (seconds), and the memory consumption (RAM-Hours). All methods are run on an AMD 5950X 16-core CPU with 64 GB RAM and Ubuntu 22.04.

5 RESULTS

Comparison of the proposed techniques

First, we evaluate each the proposed techniques to analyze their effectiveness, starting with instance aging using the fading window. Table 3 shows the average and the rank for the subset accuracy, accuracy, f-measure, runtime and RAM-hours for each of the six techniques detailed in Section 3 across all 37 evaluated datasets.

To evaluate the impact of instance aging using a fading window, we compare the fading window model (FAD) with the baseline sliding window model (SLI). Experiments show that weighting using a fading window does increase all subset accuracy, accuracy, and F-measure. It has worse runtime and RAM-hours, but the impact here is very minimal. This demonstrates that although the fading mechanism is very simple, it does result in improved predictions at very low cost.

Comparing the results for the query-based rejuvenation model (QBR) to those for FAD, we actually see a decrease in performance. QBR preforms worse than FAD, and actually worse than SLI, across all metrics. It is clear that this naive method for rejuvenation is not beneficial and that instances should not be rejuvenated solely for being queried as a nearest neighbor. This reflects the importance of the adaptation to concept drift as the most similar instances in the window do no longer necessarily provide meaningful information to the classifier. Looking at the accuracy-based rejuvenation model (ABR), however, we see that rejuvenation of instances improves the classification performance. Here, instances are rejuvenated based on the accuracy of their contributions to predictions. Across subset accuracy, accuracy and F-measure, ABR consistently performs better than FAD. Therefore, instances and concepts strengthened by the accuracy-based rejuvenation strategy are relevant and beneficial to the classification model.

Looking at subset accuracy, accuracy and F-measure for the pruning model (PRU) and the distance-weighted voting model (DWV) we also see incremental improvements in predicting. Pruning all instances below a fitness threshold provides slight gains in these metrics and distance-weighted voting still more. Interestingly, the average runtime and RAM-hours of PRU are the worst of the six models, we don't see improvements in resource use until we add in the distance-weighted voting. This demonstrates the interconnectivity of the strategies. The aging, rejuvenation and distance-weighting strategies improve the accuracy of the classification

Table 3: Comparison of the proposed aging and rejuvenating strategies over the average performance on 37 multi-label datasets. The last row averages the ranks across all metrics.

Avg. Perf.	SLI	FAD	QBR	ABR	PRU	DWV
Subset acc.	0.2790	0.2943	0.2642	0.3221	0.3345	**0.3353**
Accuracy	0.3904	0.4138	0.3724	0.4364	0.4477	**0.4486**
F-Measure	0.4817	0.4963	0.4624	0.5173	0.5275	**0.5285**
Runtime	**0.0740**	0.0745	0.0755	0.0760	0.0798	0.0757
RAM-Hours	**6.13E-3**	6.51E-3	7.17E-3	7.79E-3	8.55E-3	6.87E-3
Rank	SLI	FAD	QBR	ABR	PRU	DWV
Subset acc.	4.2162	4.2703	4.1892	3.1622	2.7973	**2.3649**
Accuracy	4.2838	3.8243	4.4054	3.1622	2.9865	**2.3378**
F-Measure	3.3378	3.7973	3.8649	3.8243	3.4730	**2.7027**
Runtime	2.8784	3.5135	3.3919	4.0270	4.9865	**2.2027**
RAM-Hours	3.1000	3.7571	3.6571	4.2000	4.4286	**1.8571**
Avg. Rank	3.5632	3.8325	3.9017	3.6751	3.7344	**2.2931**

model. Consequently, the pruning mechanism and the accuracy-based rejuvenation, which rejuvenates based on instance accuracy as compared to the whole window accuracy, can more easily identify poorly performing instances for removal, reducing the window size and improving performance. The combined DWV model, with all the techniques working together, shows the best average rank across all five metrics, demonstrating that pruning and distance-weighted voting improve the classifier's predictions and that the six proposed techniques complement each other.

Comparison with other Multi-Label kNN classifiers

Our second experiment was to compare ARkNN against state-of-the-art multi-label classifiers. Focusing on algorithms that also use a k-nearest neighbors method, we chose seven recent multi-label streaming classifiers as reference algorithms. Most of the compared algorithms use self-adjusting or dual windows as techniques for adapting to concept drift. Table 4 shows the average performance and the corresponding ranks for the subset accuracy, accuracy, F-measure, runtime, and RAM-hours for ARkNN and each of the seven compared algorithms across all 37 datasets.

Looking at just the predictive performance (subset accuracy, accuracy, and F-measure), ARkNN does not achieve the best results. AESAKNNS has the highest average for all three metrics and the highest rank for two of the three. This is mainly because AESAKNNS is an ensemble method. However, ARkNN is a close contender. When looking at the subset accuracy, accuracy and F-measure, only AESAKNNS and ODM consistently outperform ARkNN and the differences are small. Looking at the runtime and memory use, ARkNN is clearly superior. The average runtime for ARkNN is 1/5th of the average runtime of AESAKNNS, the next fastest competitor. In addition, the ranks show that AESAKNNS is on average the second slowest algorithm, whereas ARkNN is consistently fast, on average ranking as the fastest. Similarly, the average RAM-hours for ARkNN are 1/50th of the RAM-hours for OMK, and ARkNN also achieves that best rank for RAM-hours. Looking at the overall average rank across all five metrics, ARkNN clearly out-performs the other methods. The Nemenyi critical value is 1.7261 for $\alpha = 0.05$ indicating statistical differences between ARkNN and all methods expect MLSAkNN and ODM for the average rank.

This is more clearly shown in Figure 5. The height of each arc indicates the average rank across all datasets achieved by each algorithm. A greater height indicates a higher rank and better performance. Looking at the central circle of wedges indicating runtime, it clear that MLkNN, with its complex computation of prior and posterior probabilities, has the worst runtime and that ARkNN, with its simple instance-based strategies, is the fastest. Combined with the pink arc indicating memory use, it is clear that ARkNN is substantially more resource efficient than its competitors resulting in the best combined performance.

The complex strategies used by state-of-the-art algorithms to improve predictions and respond to concept drift regularly come at a steep cost in terms of resource use. MLSAMkNN, OMK and ODM all use dual short- and long-term memories. MLSAMPkNN and MLSAkNN use a resource-heavy self-adjusting window. AESAKNNS is an ensemble method that uses a collection of ADWIN detectors. These are all state-of-the-art techniques and AESAKNNS and ODM in particular have excellent predictive power. When classifying streaming multi-label data, however, resource use is important. In

Table 4: Comparison of the Aging and Rejuvenating kNN (ARkNN) with other multi-label kNN-based classifiers over the average performance on 37 multi-label datasets. The last row averages the ranks across all metrics.

Avg. Perf.	MLkNN	MLSAMkNN	MLSAMPkNN	MLSAkNN	AESAKNNS	OMK	ODM	ARkNN
Subset acc.	0.2171	0.2860	0.3074	0.3159	**0.3404**	0.2501	0.3477	0.3353
Accuracy	0.3023	0.3850	0.4106	0.4400	**0.4618**	0.3609	0.4565	0.4486
F-Measure	0.3824	0.4687	0.4920	0.5307	**0.5539**	0.4418	0.5351	0.5285
Runtime	3.2822	5.6077	0.5213	0.4533	0.3503	0.5167	0.4086	**0.0757**
RAM-Hours	9.67E-1	3.69E+0	1.09E+0	1.26E+0	6.14E-1	3.47E-1	3.52E-1	**6.87E-3**
Rank	MLkNN	MLSAMkNN	MLSAMPkNN	MLSAkNN	AESAKNNS	OMK	ODM	ARkNN
Subset acc.	6.7162	5.2568	4.4730	4.0676	3.6486	5.1486	**3.3243**	3.3649
Accuracy	6.7838	5.4595	4.6892	3.5946	**3.1622**	5.0811	3.6622	3.5676
F-Measure	6.7703	5.3243	4.7162	3.4054	**3.0270**	5.2703	3.6757	3.8108
Runtime	7.5946	5.0000	4.5405	4.6351	5.5135	4.0135	3.3243	**1.3784**
RAM-Hours	6.8000	4.9429	5.1143	5.0571	6.6571	3.5714	2.8000	**1.0571**
Avg. Rank	6.9330	5.1967	4.7066	4.1520	4.4017	4.6170	3.3573	**2.6358**

a real-world scenario it is unrealistic to assume that computing resources are infinite, and the streaming nature of the data requires that models can make predictions and update as quickly as the data is arriving. Our experiments demonstrate that ARkNN, using a collection of simple and highly resource-efficient techniques, can achieve predictive power competitive with state-of-the-art multi-label streaming algorithms using much fewer resources.

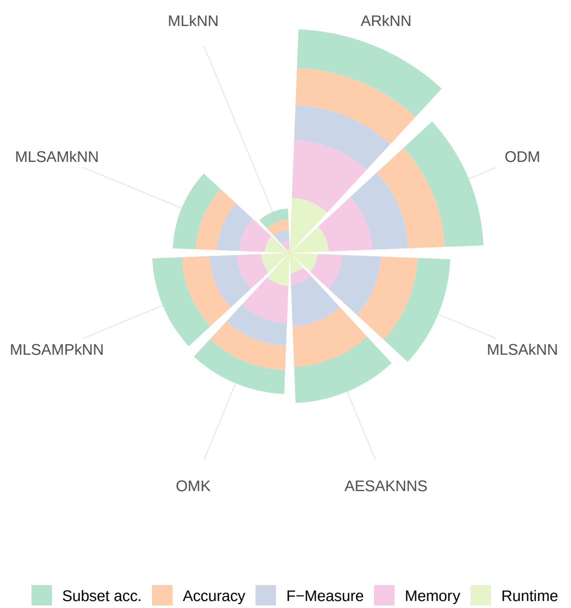

Figure 5: Stacked ranks of all performance metrics.

6 CONCLUSIONS

We have proposed ARkNN, a method using a combination of simple, resource efficient techniques to weight instances within a single updating window to classify streaming, multi-label data. ARkNN gives greater importance to neighbors that are closer, newer and better performing, allowing it respond to concept drift and quickly devalue and remove older concepts and noisy data, while retaining older information that consistently benefits the model. We have shown experimentally that each of the mechanisms used by ARkNN improves the predictive performance of the algorithm, without increasing its runtime or memory use. In a comprehensive study using real world data, we have shown that ARkNN has predictive capabilities that exceed many state-of-the-art methods and are closely comparable even to the most highly accurate algorithms. ARkNN achieves this predictive performance while using fewer resources than any of the compared methods. When evaluating runtime and memory use, as well as predictive performance, ARkNN outperforms all compared algorithms, making it an excellent choice for mining multi-label data streams, where resources are limited. ARkNN shows that it is possible to achieve excellent predictive power using methods that are resource efficient. Future works will also investigate the dynamic relevance of attributes (e.g. Relief family) in combination with dynamic instance weighting.

REFERENCES

[1] David W Aha, Dennis Kibler, and Marc K Albert. 1991. Instance-based learning algorithms. *Machine learning* 6, 1 (1991), 37–66.
[2] Gavin Alberghini, Sylvio Barbon Junior, and Alberto Cano. 2022. Adaptive ensemble of self-adjusting nearest neighbor subspaces for multi-label drifting data streams. *Neurocomputing* 481 (2022), 228–248.
[3] Maroua Bahri, Albert Bifet, João Gama, Heitor Murilo Gomes, and Silviu Maniu. 2021. Data stream analysis: Foundations, major tasks and tools. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 11, 3 (2021).
[4] Jürgen Beringer and Eyke Hüllermeier. 2007. Efficient instance-based learning on data streams. *Intelligent Data Analysis* 11, 6 (2007), 627–650.
[5] Albert Bifet and Ricard Gavaldà. 2007. Learning from time-changing data with adaptive windowing. In *SIAM International Conference on Data Mining*. 443–448.
[6] Jasmin Bogatinovski, Ljupčo Todorovski, Sašo Džeroski, and Dragi Kocev. 2022. Comprehensive comparative study of multi-label classification methods. *Expert Systems with Applications* 203 (2022), 117215.
[7] Ling Chen and Qingling Mei. 2014. Mining frequent items in data stream using time fading model. *Information Sciences* 257 (2014), 54–69.
[8] João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. 2009. Issues in evaluation of stream learning algorithms. In *ACM SIGKDD*. 329–338.
[9] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)* 46, 4 (2014), 1–37.
[10] Eva Garcia-Martin, Albert Bifet, Niklas Lavesson, Rikard König, and Henrik Linusson. 2022. Green Accelerated Hoeffding Tree. *arXiv preprint arXiv:2205.03184* (2022).
[11] Heitor Murilo Gomes, Jean Paul Barddal, Fabrício Enembreck, and Albert Bifet. 2017. A survey on ensemble learning for data stream classification. *ACM Computing Surveys (CSUR)* 50, 2 (2017), 1–36.
[12] Imen Khamassi, Moamar Sayed-Mouchaweh, Moez Hammami, and Khaled Ghédira. 2018. Discussion and review on evolving data streams and concept drift adapting. *Evolving Systems* 9, 1 (2018), 1–23.
[13] Xiangnan Kong and S Yu Philip. 2011. An ensemble-based approach to fast classification of multi-label data streams. In *International Conference on Collaborative Computing: Networking, Applications and Worksharing*. 95–104.
[14] Viktor Losing, Barbara Hammer, and Heiko Wersing. 2016. KNN classifier with self adjusting memory for heterogeneous concept drift. In *IEEE International Conference on Data Mining*. 291–300.
[15] Xin Lu, Shaonan Jin, Xun Wang, Jiao Yuan, Kun Fu, and Ke Yang. 2020. A Mining Frequent Itemsets Algorithm in Stream Data Based on Sliding Time Decay Window. In *Artificial Intelligence and Pattern Recognition*. 18–24.
[16] Aljaž Osojnik, Panče Panov, and Sašo Džeroski. 2017. Multi-label classification via multi-target regression on data streams. *Machine Learning* 106, 6 (2017), 745–770.
[17] Martha Roseberry and Alberto Cano. 2018. Multi-label kNN classifier with self adjusting memory for drifting data streams. In *International Workshop on Learning with Imbalanced Domains: Theory and Applications, ECML-PKDD*, Vol. 94. 23–37.
[18] Martha Roseberry, Bartosz Krawczyk, and Alberto Cano. 2019. Multi-label punitive kNN with self-adjusting memory for drifting data streams. *ACM Transactions on Knowledge Discovery from Data* 13, 6 (2019), 1–31.
[19] Martha Roseberry, Bartosz Krawczyk, Youcef Djenouri, and Alberto Cano. 2021. Self-adjusting k nearest neighbors for continual learning from multi-label drifting data streams. *Neurocomputing* 442 (2021), 10–25.
[20] Raquel Sebastião, João Gama, and Teresa Mendonça. 2017. Fading histograms in detecting distribution and concept changes. *International Journal of Data Science and Analytics* 3, 3 (2017), 183–212.
[21] Lulu Wang, Hong Shen, and Hui Tian. 2017. Weighted ensemble classification of multi-label data streams. In *PAKDD*. 551–562.
[22] Xihui Wang, Pascale Kuntz, and Frank Meyer. 2021. Exploration des mémoires à court et long terme pour la classification multi-labels en flux. In *Conférence Extraction et Gestion des Connaissances*.
[23] Xihui Wang, Pascale Kuntz, Frank Meyer, and Vincent Lemaire. 2021. Multi-Label kNN classifier with Online Dual Memory on data stream. In *International Conference on Data Mining Workshops*. 405–413.
[24] Michał Woźniak, Andrzej Kasprzak, and Piotr Cal. 2013. Weighted aging classifier ensemble for the incremental drifted data streams. In *International Conference on Flexible Query Answering Systems*. Springer, 579–588.
[25] Michał Woźniak, Paweł Zyblewski, and Paweł Ksieniewicz. 2021. Active Weighted Aging Ensemble for Drifted Data Stream Classification. *arXiv preprint arXiv:2112.10150* (2021).
[26] Unil Yun, Donggyu Kim, Eunchul Yoon, and Hamido Fujita. 2018. Damped window based high average utility pattern mining over data streams. *Knowledge-Based Systems* 144 (2018), 188–205.
[27] Min-Ling Zhang and Zhi-Hua Zhou. 2007. ML-KNN: a lazy learning approach to multi-label learning. *Pattern Recognition* 40, 7 (2007), 2038–2048.
[28] Xiulin Zheng, Peipei Li, Zhe Chu, and Xuegang Hu. 2020. A survey on multi-label data stream classification. *IEEE Access* 8 (2020), 1249–1275.