ON-THE-FLY ENSEMBLE CLASSIFIER PRUNING IN EVOLVING 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
Sanem Elbaşı
September 2019

ON-THE-FLY ENSEMBLE CLASSIFIER PRUNING IN EVOLVING DATA STREAMS

By Sanem Elbaşı

September 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

ON-THE-FLY ENSEMBLE CLASSIFIER PRUNING IN EVOLVING DATA STREAMS

Sanem Elbaşı
M.S. in Computer Engineering
Advisor: Fazlı Can
September 2019

Ensemble pruning is the process of selecting a subset of component classifiers from an ensemble which performs at least as well as the original ensemble while reducing storage and computational costs. Ensemble pruning in data streams is a largely unexplored area of research. It requires analysis of ensemble components as they are running on the stream and differentiation of useful classifiers from redundant ones. We present two on-the-fly ensemble pruning methods; Class-wise Component Ranking-based Pruner (CCRP) and Cover Coefficient-based Pruner (CCP). CCRP aims that the resulting pruned ensemble contains the best performing classifier for each target class and hence, reduces the effects of class imbalance. On the other hand, CCP aims to select components that make misclassification errors on different instances. The conducted experiments on real-world and synthetic data streams demonstrate that different types of ensembles that integrate pruners consume significantly less memory and perform significantly faster without hurting the predictive performance.

Keywords: Ensemble learning, Ensemble pruning, Ensemble efficiency, Concept drift.

ÖZET

EVRİLEN VERİ AKIŞLARINDA HEYET SINIFLANDIRICILARIN ANINDA BUDANMASI

Sanem Elbaşı
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Danışmanı: Fazlı Can
Eylül 2019

Heyet yaklaşımlı sınıflandırıcılarda budama, bellek kullanımını ve işlem zamanını azaltırken en azından orijinal heyet kadar başarılı bir alt küme seçme işlemidir. Heyet yaklaşımlı sınıflandırıcılarda budama konusu veri akışlarında üzerine düşülmemiş bir araştırma konusudur. Heyet bileşenlerinin veri akışı devam ederken analiz edilmesini ve faydalı bileşenlerin gereksiz olanlardan ayrıştırılmasını gerektirir. Bu tezde iki anında heyet budama metodu önerilmiştir; CCRP ve CCP. CCRP seçilen alt kümede her bir hedef sınıf için en iyi sınıflandırmayı yapan bileşenlerin bulunmasını hedefliyerek veri akışındaki sınıf dağılımlarının dengesizliğinin etkilerini azaltır. Öte yandan, CCP farklı veri noktalarında hatalar yapan bileşenleri seçer. Gerçek ve sentetik veri akışları üzerinde farklı sınıflandırıcılar ile yapılan deneyler, budama işleminin kaynaştırıldığı heyet sınıflandırıcılarının bellek kullanımını ve harcadıkları zamanı istatistiksel açıdan önemli derecede azaltırken sınıfların tahminsel başarına zarar vermediğini göstermiştir.

Anahtar sözcükler: Heyet sınıflandırıcılar, Heyet yaklaşımlı sınıflandırıcılarda budama, Heyet verimliliği, Kavram sürüklenmesi.

Acknowledgement

First and foremost, I would like to express my gratitude for my advisor Prof. Fazlı Can for his patience, guidance, and support. I would like to thank my previous advisor Prof. Hakan Ferhatosmanoğlu for his guidance. I would also like to thank my jury members Selim Aksoy and İsmail Sengör Altıngövde.

I would like to express my most intimate thanks to my beloved parents, Aydan and Mehmet, and by lovely brother Atakan for their encouragement and endless support.

I would like to sincerely thank Alican Büyükçakır for his support and contribution.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the problems addressed in this thesis and gives an overview of subsequent chapters. Section 1.1 introduces the ensemble systems by highlighting their advantages over single model learners and the importance of the diversity among the models within an ensemble. In section 1.2, we discuss the challenges of stream classification and significance of handling concept drifts in data streams. Section 1.3 describes the motivation behind this study. Section 1.4 summarizes our contributions.

1.1 Ensemble Systems

Ensemble systems consist of several machine learning models and operate by combining the decisions of these models. Generalization ability of a single machine learning model is bounded by the noises, outliers and overlapping data distributions. By combining decisions of multiple models, ensemble systems are able to achieve higher generalization over the data [3].

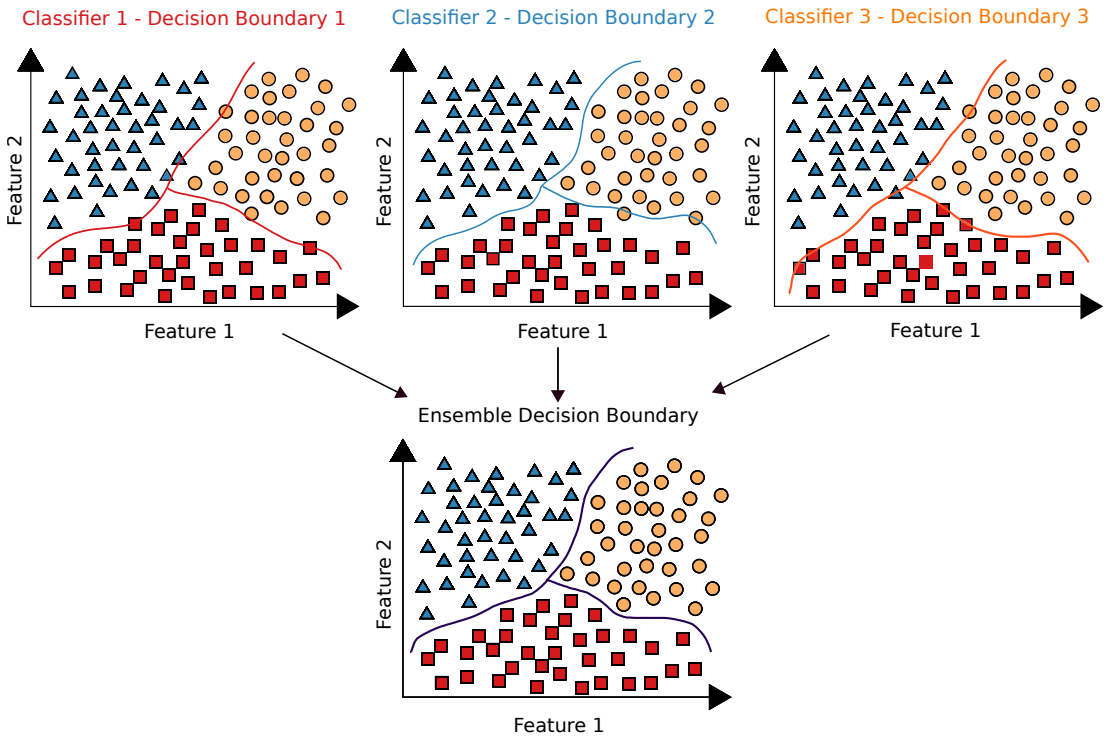

Figure 1.1: Ensemble Systems. Combining decision boundaries of several classifiers to achieve a better decision boundary over the dataset. Revised from [1]

Since it is almost impossible to train a classifier that classifies each data point correctly, we can only hope for a classifier that has learned a region of the dataset and is able to correctly classify data points within that region. By using ensemble models, we are able to improve the overall performance of the system. The main idea behind the ensembles is that if a model did not learn a region on the dataset, another model might be able to cover that region. In other words, if a model misclassifies a data point, another model on the ensemble can correctly classify that data point. In order to obtain an ensemble with high accuracy, ideally we need to have distinct classifiers within the ensemble so that no two models in the ensemble misclassify the same data point [1].

1.2 Concept Drifts in Data Streams

An ordered sequence of items that arrive in a timely fashion is called a *data stream*. Main challenges of data stream classifications are huge volume of the data, time and memory constraints and concept drifts that occur over time. Data classification models learn relations amongst input features and target classes; however, in data streams, there is always an uncertainty about the future data instances [4]. Over time, the relationship between the input features and the target classes may change and the same features may lead to different target classes [5]. Such changes in the relation between data points and their corresponding classes over time are called *concept drifts* [6]. Concept drifts are categorized under two main subjects; real and virtual concept drifts [7]. In real concept drift, the true decision boundary of the problem changes, which leads to misclassification of data points. Where in virtual concept drifts the true decision boundary stays the same but the data distributions are shifted [8]. Even though the true decision boundary stays the same, the change in the data distribution can affect the predictive performance of the models. When concept drifts arise either real or virtual, the previously learned relationship between the input data point and the target classes may lead to misclassification.

Figure 1.2 demonstrates the four types of real concept drifts; sudden, incremental, gradual and reoccurring drifts [2, 5, 9, 10]. Sudden or abrupt drifts occur instantly and the changes in the target classes are constant. Incremental drifts occur slowly in a timely manner. Gradual drifts occur when the change in data example involves the class distribution of various data [11]. When the change in the concept reverts over time, it is called reoccurring drift. There are also other types of changes in the data that may be observed over time, outliers and noise, which should not be considered as concept drifts and the models should not take action when they occur.

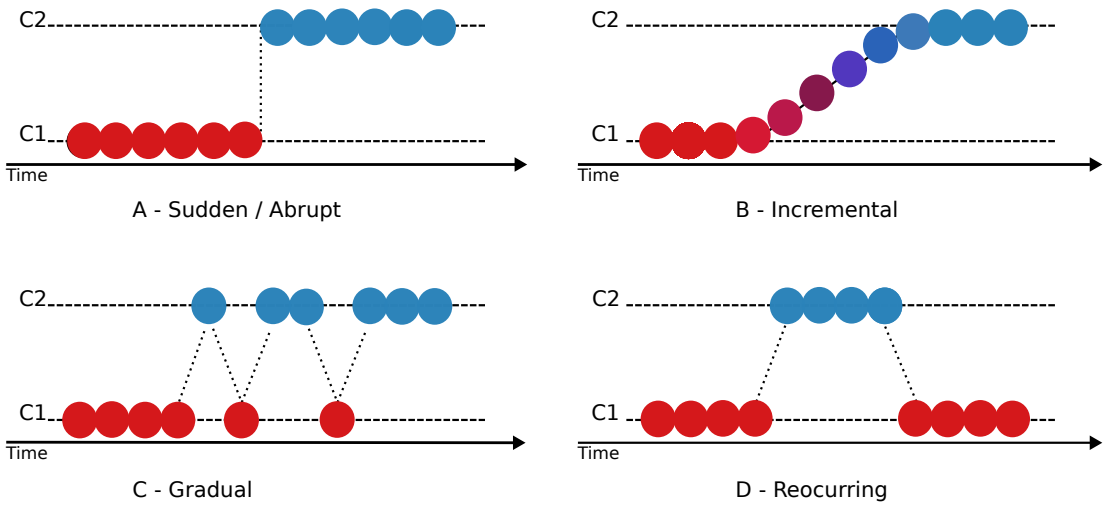

Figure 1.2: Four patterns of real concept drift over time. Revised from [2]

1.3 Motivation

Data streams are environments such as network event logs, video streams, call records and transaction records where a vast amount of data is generated at high speed [12]. Classification models in data streams have to work under strict time and memory constraints and be able to adapt changes in the distribution of data over time [5]. Robustness against concept drifts is a topic that is extensively studied in the literature [13, 14]. As data continuously arrive, learned information from the past instances become irrelevant under concept drifts. Classification models need to adapt to the new concepts by not only learning the new concepts but also forgetting the now-obsolete ones [14].

Ensembles are common choices for models in stream environments, as they often improve predictive performance while providing robustness against concept-drifting and time-evolving data [5]. Ensembles employ different approaches for dealing with concept drifts, such as replacing the weakest / lowest weighted classifier with a new one that is trained with the more recent data [13], or updating ensemble weights regularly [13, 15]. Despite the popularity and prevalence of ensemble learning in data streams, ensemble pruning is still a vastly undiscovered area of research in the stream mining community.

1.4 Contributions

In this study, we tackle the task of ensemble pruning in data streams for multi-class classification. Our contributions are as follows. We,

- introduce 2 explicit pruning techniques that can be integrated into any type of streaming ensemble and called whenever pruning is requested. Our methods result in pruned ensembles that are more efficient in terms of memory consumption and execution time.
- provide to the best of our knowledge, the first large-scale[1] on-the-fly ensemble pruning methods for streaming data.
- propose CCRP[2], a class imbalance-aware pruning method, where a pruned ensemble does not lose its ability to classify rare or less-frequent classes, as a result of our class-wise component prediction analysis and ranking.
- present CCP[2], a misclassified instances aware pruning method, where we select classifiers such that selected classifiers are as unique as possible with respect to the misclassified instances.

[1]The previous studies [16, 17] experimented on datasets with number of instances in the scales of hundreds or thousands

[2]Full phrases are given in Chapter 4

Chapter 2

Problem Definition and Notation

In data stream classification task, main challenges are computational cost, execution time, memory consumption and handling concept drifts. Figure 2.1 demonstrates the effects of the concept drifts in the stream environment and the importance of handling concept drifts. Ensemble classifiers are shown to be better performing with drifting environments than single classifiers [18, 19, 20]. However, ensembles with a large number of classifiers entail large memory consumption and execution time which does not comply with the constraints of the data stream classification task. Selecting complementary classifiers from the ensemble and prune the rest of the ensemble copes with the execution time and memory consumption constraints[21]. On top of everything, a pruned ensemble with complementary classifiers can achieve higher accuracy results and cope with the concept drifts by removing the classifiers that learned now-obsolete information [22, 23, 24, 25]. One major concern with integrating a pruning method to a data stream classification model is the computational cost.

We consider the problem of ensemble pruning within the context of supervised classification in data streams. Data stream $\mathcal{D}$ is a (possibly infinite) sequence of time-ordered data that consists of pairs $(X^{(t)}, y^{(t)})$ where t denotes the arrival time. In classification, the target $y^{(t)}$ has L-many classes where $L > 1$. An ensemble model ξ is a group of K component classifiers, i.e. $\xi = \{C_1, C_2, \ldots, C_K\}$

where the prediction of the model for an instance, $H(x)$, is a combination of individual hypotheses ($h_k(x)$) of its component classifiers where k denotes the classifier index.

The aim of ensemble pruning is selecting a subset of classifiers $\xi' \subset \xi$ such that the ensemble's both efficiency and predictive performance are improved. Here, let us denote the size of the pruned ensemble as $|\xi'| = \varphi$ where $\varphi < K$. 2.1 gives a summary of the symbols and notations used in this thesis.

Table 2.1: Symbols and Notation used in this thesis

Symbols	Meaning
K	Number of components in the original ensemble
N	Size of the sliding window
L	Number of classes in the target space
φ	Number of components in the pruned ensemble
$\mathcal{X}$	Input attribute space.
x	A data instance. $x =< x_1, x_2, .., x_i, .., x_N > \in \mathcal{X}$
ξ	Original ensemble of components $\xi = C_1, C_2, ..., C_K$
ξ'	Pruned ensemble where $\xi' \subset \xi$ and $\varphi = \|\xi'\|$
$h_k(x)$	Hypotheses of component C_k for an instance x
$H(x)$	Hypotheses of the ensemble for an instance

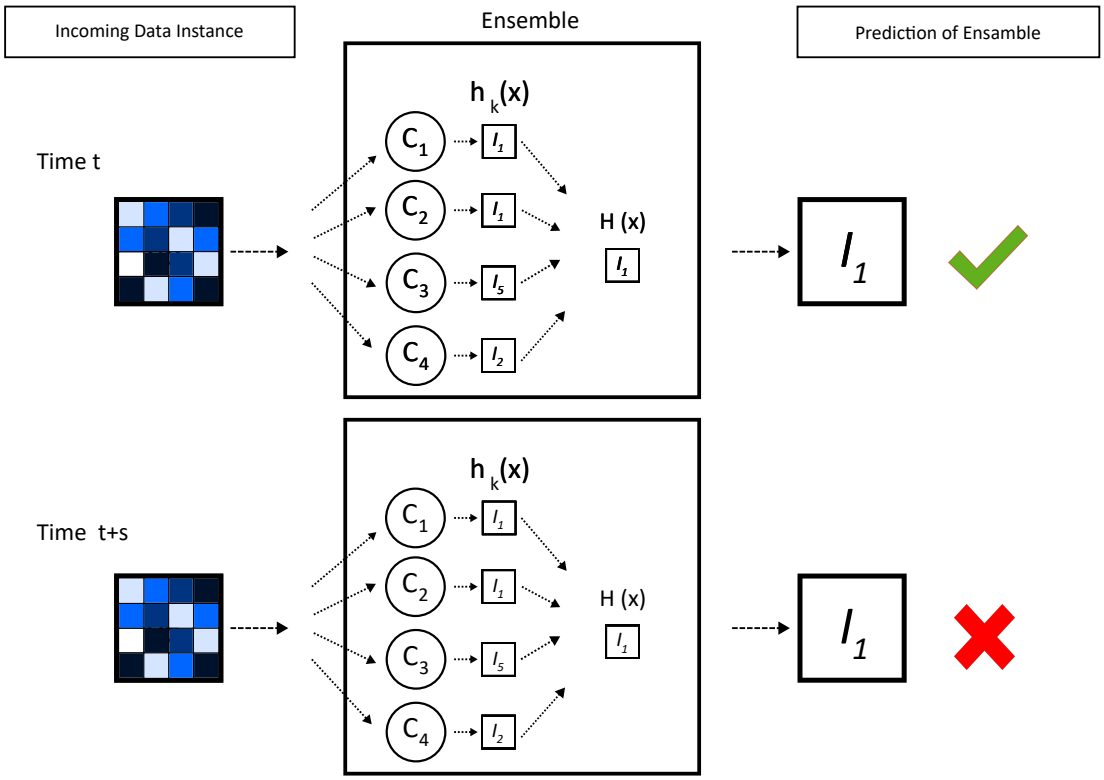

Figure 2.1: Stream data classification under concept drift. Feature-wise the same input data instances are classified by the ensemble at 2 different time points. Ensemble makes the same prediction for the input data; however, the prediction is correct for time t, it is incorrect for time $t+s$. If this behaviour is consistent for a time period, we can say that a concept drift occurred during the time interval $[t, t+s]$. Misprediction at time $t+s$, shows that the learned information by the components of the ensemble became obsolete.

Chapter 3

Related Work

3.1 Ensemble Pruning Methods

Even though ensembles with an immense number of classifiers may increase the generalization accuracy of the system, in real-world problems the memory and time constraints may not comply with this approach. Empirical studies have shown that even a fraction of the original ensemble can perform the same classification task without a significant decrease in accuracy, in fact, they may increase the accuracy [22, 26, 27, 28]. In [29, 30] it is theoretically proven that the ideal ensemble size is equal to the number of target classes and throughout the experiments, it is shown that the optimal ensemble size is closer to the number of target classes than infinite. However, selecting a subset of classifiers that achieves the best performance when combined is a computationally expensive process. For a given performance measure, finding the optimal subset of classifiers has exponential complexity, dependent on the size of the original set of classifiers, since it requires a combinatorial search through the classifiers.

In this section, we discuss some of the notable pruning approaches.

3.1.1 Ordering-based Pruning

In ordering based pruning methods, components are sorted by a defined criterion metric and only the components that surpass a boundary are kept. In [21] they show that with an appropriate metric, ensembles that are pruned based on the ordering achieves minimum error with intermediate φ and generally they achieve lower error rates compared to the original ensemble.

KL-divergence Pruning. This approach is proposed in [26] with three other ordering-based pruning methods. In KL-divergence authors start with the assumption that all of the components in the ensemble have similar error rates. They use Kullback-Leibler Divergence [31] as a measure of diversity between two components' probability distribution. Let p_i and p_j be the probability distributions of C_i and C_j where $C_i, C_j \in \xi$; KL Divergence, $D(p_i||p_j)$, is given by;

$$D(p_i||p_j) = \sum_{x \in X} p_i(x) \log \frac{p_i(x)}{p_j(x)}. \tag{3.1}$$

The goal is to find a subset $\xi' \in \xi$ such that, the sum of the pair-wise KL-Divergence of the components, $J(\xi')$, is minimized.

$$J(\xi') = \sum_{C_i, C_j \in \xi'} D(p_i||p_j) \tag{3.2}$$

Since finding the optimal subset is a computationally expensive problem, they use a greedy algorithm. They start by adding the first created component in the ξ' and iterate through $\xi \setminus \xi'$ to find the C_i that achieves minimum $J(\xi')$ when added. They repeat the process until ξ' has φ components.

Kappa Pruning. This approach is also proposed in [26]. This time, they consider the divergence of the classification decisions of the components. Kappa statistic [32], κ, is used to measure the divergence of the component decisions. For each $C_i, C_j \in \xi$, they create a matrix R with size L x L where R_{lm} denotes the

number of instances C_i predicted l while C_j predicted m. They define agreement of two components Θ as;

$$\Theta = \frac{\sum_{n=0}^{L} R_{nn}}{m} \tag{3.3}$$

where m denotes the number of classified instances. Then, by calculating the agreement of two components by chance, they determine the κ score. With this approach $\kappa = 0$ denotes that the agreement of two classifiers is equal to the agreement by chance and $\kappa = 1$ denotes that two components are identical based on their predictions. They use a greedy algorithm to construct ξ' by iterating through all of the components in ξ.

In [25], the authors propose a reverse approach to Kappa pruning. They start with ξ and iteratively remove components that are reducing the average κ score of the ensemble. In [21], they improve Kappa pruning proposed in [26] by considering the Kappa scores of the components $C_i \in \xi \setminus \xi'$ with respect to the components in ξ' where $\setminus$ denotes the set difference operator. Another improvement on Kappa pruning is proposed in [26] which is called Kappa-Error Convex Hull pruning. With this approach, they incorporate diversity and accuracy by plotting the pair-wise kappa scores and the pairs average error which they call Kappa-Error diagram. After the diagram is generated they apply Convex hull and ξ' is constructed by the components that appear on the pair-wise points on the convex hull. With this approach they select components that are most accurate and most diverse; however, the components selected by this approach does not necessarily exhibit both of these features at the same time.

Reduce Error Pruning. This approach is first introduced in [33] as a decision tree pruning method, then [26] revised this method to apply in ensembles. In this method ξ' starts with the component that achieves the highest accuracy, then it iteratively adds a component from $\xi \setminus \xi'$ that estimates minimum generalization error. They define their measure, T_u, for C_u as;

$$T_u = \sum I(H_{\xi' \cup C_u}(x) = y) \tag{3.4}$$

where $H_{\xi'}(x)$ denotes the hypothesis of the ξ' if the component C_u is selected. In [26], after φ components are selected, they apply a backfitting for a predefined number of iterations to approximate even better-performing subensemble. During backfitting phase, they replace a component C_i from ξ' with C_j from the eliminated components if the resulting ensembles estimated accuracy is higher than the current ξ'. In later works backfitting phase is abandoned since it dramatically increases the execution time [24, 25, 34, 35]. Besides, it has been shown that backfitting does not significantly reduce the generalization error of ξ' [35]. In a more recent study, Reverse Reduce Error (RRE) Pruning is proposed [36]. RRE starts identical as Reduce Error Pruning but instead of applying backfitting, they select the Worst Single Model (WSM) from ξ and subtract its prediction from the prediction of ξ' based on the assumption that WSM generally mispredicts.

Complementarity Measure Pruning. This approach is quite similar to Reduced Error pruning and first proposed in [24]. This approach starts ξ' with the most accurate component from the ξ than iteratively adds a component from $\xi \setminus \xi$ that complements ξ' the most. They define the complementariness measure of component C_u, T_u, is defined as;

$$T_u = \sum I(y = h_u(x) and H_{\xi'}(x) \neq y) \tag{3.5}$$

where $H_{\xi'}(x)$ denotes the hypothesis of the ξ'. At each iteration, the component that achieves highest complementarity score is added to ξ' until the size of ξ' reaches to φ. The difference between this approach and the Reduce Error pruning is that complementarity measure favors the components that lower the confidence of the ξ' when the hypothesis of the ensemble is incorrect.

Concurrency thinning [25] takes a similar approach, they iterate through ξ and remove the component with the smallest complementarity measure that they define. In concurrency thinning they reward component by increasing its score

by 2 if the $h_u(x) = y$ where $H_\xi(x) \neq y$ and if $h_u(x) = y$ where $H_\xi(x) = y$ the component is rewarded by 1. Finally, component is penalized by decreasing its score by 2 if $h_u(x) \neq y$ where $H_\xi(x) \neq y$.

Margin Distance Minimization [24]. In this method, authors define a *signature vector* c_i for each $C_i \in \xi$ where the dimension of the vector is equal to the number of classified instances, V. If C_i correctly classifies instance t; the related index of the C_i is equal to 1, otherwise -1. The performance of the ensemble, $\bar{c}$, is estimated by averaging signature vectors of all components in ξ. If all of the instances are correctly classified then $\bar{c}$ would be on the first quadrant in V-dimensional space. The goal is to select ξ' such that the distance between $\bar{c}$ and a selected target vector from the first quadrant of V-dimensional space is minimized.

3.1.2 Clustering-based Pruning

In clustering-based Pruning methods, a clustering method is applied to the components in the original ensemble in order to find a subset that is representative of the original set and diverse. After clustering, components are selected with different strategies.

In [37], authors use agglomerative clustering algorithm [38]. In the first phase, in which they define the distance metric for a pair of clusters as the compound error probability. After the clustering, from each cluster component that has the maximum average distance from the other clusters is chosen in order to achieve higher diversity.

k-means clustering is used in [39] by using the predictions of the components as features. The optimal k is found by gradually increasing k until the diversity (disagreement) between the cluster components starts to deteriorate. For the optimal k diversity within clusters is low and more components can be pruned from the clusters. In the component selection phase, for each cluster, components are sorted according to their accuracy results on the training set. If a components'

diversity from the most accurate component in the cluster is higher than the pre-defined threshold, that component is removed from the ensemble. Also, if the accuracy of the ensemble increases after removing a component, that component is also pruned.

3.1.3 Optimization-based Pruning

In optimization-based pruning, selecting the subemsemble problem is approached as an optimization problem by maximizing or minimizing the objective function. Three main approaches are adopted for optimization-based pruning; heuristic, mathematical programming and probabilistic pruning.

Heuristic optimization pruning first introduced by GASEN [40]. Weights are assigned to each component which represents the component's importance for the ensemble. They start with randomly assigned weights and use a genetic algorithm to obtain the optimal weights on a test dataset. After optimum weights are assigned components with weights that are below a pre-defined threshold are pruned from the ensemble. [34] used greedy hill climbing for ensemble selection from a large set of models and [41] use collaborative web search based on the similarity of search cases.

Mathematical programming optimization pruning has a better theoretical foundation with respect to heuristic optimization pruning. [42] proposed using semi-defined programming to find a better approximate solution for ensemble pruning problem. [43] considers the ensemble selection problem under regularized framework and proposed and reduces the ensemble selection problem to a quadratic problem.

3.1.4 Pruning in Data Streams

A related topic to on-the-fly ensemble pruning is Dynamic Ensemble Selection (DES) [44]. In DES, a subset of components from a pool is selected for each test

instance. DES assigns estimated levels of competence for each classifier in the pool so that a selection method selects the classifiers to be assigned to incoming instances. Note that, unlike ensemble pruning, selection of a subset of classifiers for every instance does not reduce the size of the pool of classifiers.

Lastly, FIRE-DES++ [17] incorporates a pre-selection stage where only the classifiers that can correctly classify at least a pair of instances with different classes can proceed to the next stage of classification. Again, the pre-selection stage in FIRE-DES++ does not change the size of the pool of classifiers.

Chapter 4

Proposed Methods

Here we propose; class-wise component ranking-based and cover coefficient-based pruning methods for on-the-fly ensemble pruning for data stream classification. Both methods can be integrated to work with both dynamic and static multi-class ensembles to improve the performance and memory consumption and can be performed at any point in time on the stream.

First we introduce the notations used. Let classifiers in the original ensemble, ξ, be $\{C_1, C_2, ..., C_K\}$. Let $\mathcal{L}$ be the set of all possible target classes where $\mathcal{L} = \{l_1, l_2, .., l_L\}$. y denotes the ground truth of the instances in the sliding window where y_n denotes the ground truth class of the nth instance in the sliding window. Here, $\hat{y}_{k,n}$ denotes the predicted classes by the C_k for the last nth instances i.e. $\hat{y}_{k,n} = h_k(x_n)$. Let $\hat{\rho}_{k,l}$ be the record of the prediction of kth component for the lth class on a sliding window containing the latest N instances. Similarly, let ρ_l be the record of the ground truth information for the lth class on the sliding window.

The aim of ensemble pruning is selecting a subset of classifiers $\xi' \subset \xi$ such that the ensemble's both efficiency and predictive performance are improved. Here, let us denote the size of the pruned ensemble as $|\xi'| = \varphi$ where $\varphi < K$.

4.1 Class-wise Component Ranking-based Pruner

We introduce, CCRP, **C**lass-wise **C**omponent **R**anking-based **P**runer for multi-class online ensembles. The proposed method consists of three phases: ***Preliminary Phase***: Recording Component Predictions on the Sliding Window, ***Phase I***: On-the-Fly Performance Analysis and Class-wise Ranking of Components, and ***Phase II***: Fusion of Rankings and Component Selection (see Figure 4.1).

Preliminary Phase: Recording Component Predictions on the Sliding Window

This phase only applies to the ensembles that are not already recording the component predictions on the sliding window (e.g. OzaBagging [45]). In this phase, for the latest N data instances, predictions of classifiers and the ground truth are recorded as ρ and $\hat{\rho}$ respectively (Alg.1 line 4). Since only the latest N instances are kept in the records, the required memory for this process is fixed, despite being proportional to the number of classes.

Phase I: On-the-Fly Performance Analysis and Class-wise Ranking of Components

This is the initial phase of CCRP in which per class performances of classifiers are measured. In this phase, for each class l, predictions of classifiers and the ground truth are extracted as ρ_l and $\hat{\rho}_{k,l}$ respectively (Alg.1 line 7). Then, scores of classifiers are calculated using Mean Square Error (Eqn. 4.1) for each class (Alg.1 line 11).

$$\mathcal{L}_{k,l}(X) = \sum_{i=1}^{N} \left(\hat{\rho}_{k,l}^{(i)} - \rho_l^{(i)}\right)^2, \qquad 1 \leq k \leq K,\ 1 \leq l \leq L \tag{4.1}$$

Phase II: Fusion of Rankings and Component Selection

In the final phase, CCRP generates the overall ranking of the classifiers, using a modified version Borda Count [46] rank fusion method (Alg.1 line 15). In Modified Borda Count (MBC, hereafter), CCRP assigns $K \times L$ points (instead of K points in the regular Borda Count [46]) to the highest ranking components in the class-wise rankings. This ensures the winning component for each class to appear at the top L places in the overall ranking. Afterwards, the second classifier gets $K - 1$ points and each proceeding classifier gets 1 point less than its successor. Then, the top φ classifiers from the overall ranking are selected as the members of the pruned ensemble. It is recommended [29] that the pruned ensemble size should be at least equal to the number of classes. Taking this into account, CCRP guarantees that the best performing classifier for each class is included in the pruned ensemble when $\varphi \geq L$.

Algorithm 1 `CCRP`: Class-wise Component Ranking-based Pruner

Require: $\mathcal{D}$: data stream, ξ: ensemble, φ: pruned ensemble size
Ensure: ξ': pruned ensemble

Initialize ρ as an empty FIFO buffer of size N.
Initialize $\hat{\rho}$ as an empty FIFO buffer of size $K \times N$.
for $(X, y) \in \mathcal{D}$ **do**
  $\hat{\rho}_k$. `append`$(h_k(X))$ for $\forall k$ { Preliminary Phase }
  ρ. `append`(y)
  { Start CCRP }
  if *prune* **then**
    { Phase I }
    for $l \in L$ **do**
      for $k \in K$ **do**
        $scores$.`append`(`MSE`$(\rho_l, \hat{\rho}_{k,l})$) { Class-wise MSEs }
      end for
      $ranks[l]$ = `argsort`$(scores)$ { Class-wise order of components }
    end for
    $ccrp\_ranks$ = `ModifiedBorda`$(ranks)$ { Phase II }
    $\xi' \leftarrow$ top ranked φ components based on $ccrp\_ranks$
    $\xi \leftarrow \xi'$ { Pruned ensemble in effect }
  end if
end for

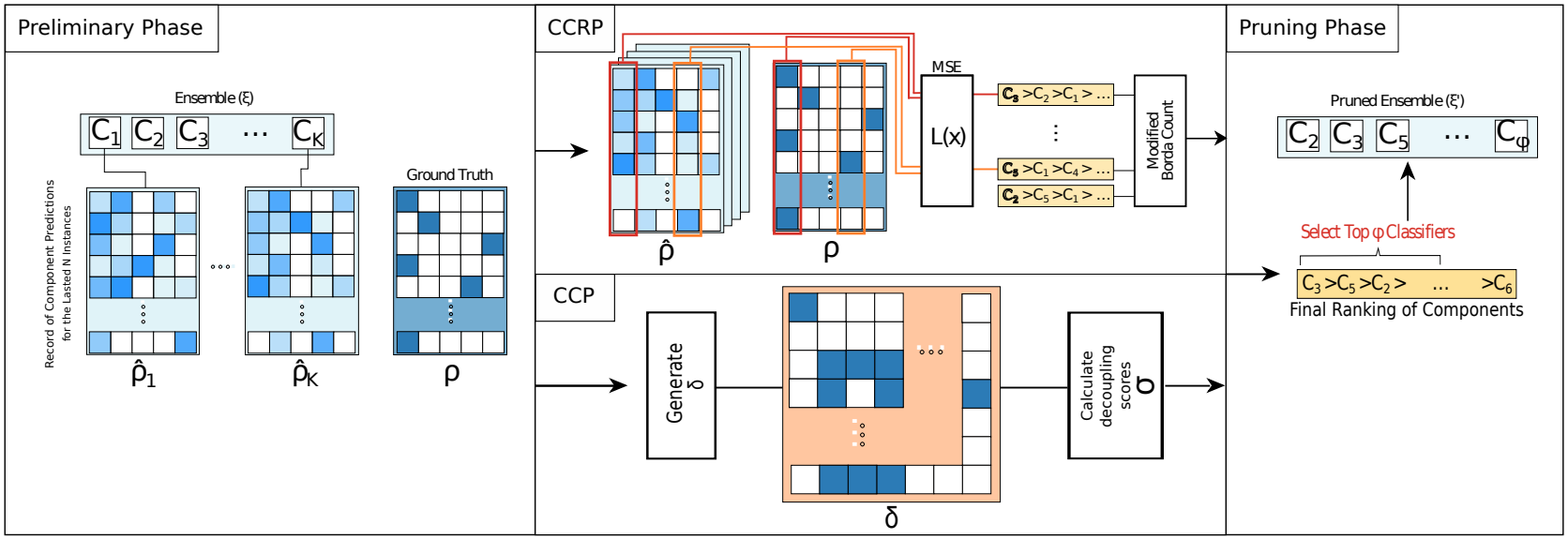

Figure 4.1: Operational pipeline of Proposed Methods. Both CCRP and CCP have preliminary and pruning phases in common. In **Preliminary Phase**, prediction history of each component in the ensemble are recorded. At each row of records, colored cells indicate relevance scores for corresponding classes where darker shades indicate higher scores. In **CCRP**, these records are used in calculating class-wise losses for each component, and class-wise rankings of those components. The acquired rankings are combined using a rank fusion algorithm (Modified Borda Count). In **CCP**, prediction histories are used to generate δ array. Colored instances in δ denotes the mispredicted instances. Decoupling scores of components, σ, is then calculated using δ. Finally, during **Pruning Phase**, the resulting top φ components ($\varphi < K$) are selected, and the rest is pruned.

4.2 Cover Coefficient-based Pruner

We introduce, CCP, **C**over **C**oefficient-based **P**runer for online classification ensembles, which is inspired by [47]; a cover coefficient-based clustering method for text clustering. CCP is ordered based pruning algorithm which can be categorized under reduce error pruning algorithms. We take components mispredictions into consideration and by using decoupling scores of each component and aim to find a subset of classifiers who make misclassification errors on different instances.

Cover Coefficient Concept

Before starting to describe the application of Cover Coefficient (CC) concept in our proposed method, it is important to introduce the CC concept as introduced in [47]. The CC concept was introduced for document clustering in databases. They define a document by term matrix where each row represents a document and each column represents a term. For each term in the document, related indices are set to 1 and the rest of the document row is set to 0. CC finds the relationship between documents by performing a *two-stage probability experiment* on the matrix and generate K by K CC matrix. In the CC matrix, each entry represents the relationship between the related documents i.e. $CC_{k,l}$ denotes coverage of kth document over lth document where $k \neq l$. In our case, our focus is on the $CC_{k,k}$ which denotes the decoupling value of kth document from the rest of the database.

Here, we adopt the CC concept to find the relationship between the components of ensemble based on the mispredicted instances. Therefore, we can distinguish components that make distinct errors so that the combined ensemble can make more accurate predictions with fewer components. We define δ as K by N matrix where each row represents a component and each column represents an instance in the sliding window (Alg.2 line 10). $\delta_{k,n}$ denotes the misprediction

of C_k at nth instance where;

$$\delta_{k,n} = \begin{cases} 1, & \texttt{if } h_k(x_n) \neq y_n \\ 0, & \texttt{if } h_k(x_n) = y_n. \end{cases}$$

After construction of δ, the CC algorithm continues with *two-stage probability experiment* which gives the probability of selecting the same mispredicted instance from two components i.e. coverage of components one another. First of all, we remove all-0 and all-1 columns from δ to simplify the calculations and save space (Alg.2 lines 12 and 13). In order to calculate the coverage of C_k over C_l, we start with C_k and look at the probability of selecting a randomly selected misclassified instance of C_k from C_l. Finally, decoupling score of C_k, σ_k, is the probability of selecting a randomly selected misclassified instance of C_k from C_k.

We give an example to clarify the *two-stage probability experiment* on a toy δ. Consider the δ in Figure 4.2, where each 1 represent misclassification error i.e. C_4 misclassified the instances $\{n_2, n_4, n_5\}$ and correctly classified $\{n_1, n_3\}$.

$$\delta = \begin{array}{c} \begin{array}{ccccc} n_1 & n_2 & n_3 & n_4 & n_5 \end{array} \\ \left[\begin{array}{ccccc} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{array}\right] \end{array} \begin{array}{c} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \end{array}$$

Figure 4.2: Example δ matrix where each column represents a component from the original ensemble, ξ, and each column represents a data instance in the last data chunk.

We calculate the coverage of C_4 over C_3 by drawing a Bayesian Tree diagram as given below. Each edge represents the probability of randomly selecting below node from the above node. We start with C_4; since C_4 made 3 misclassification errors, we draw 3 edges to misclassified instances $\{n_2, n_4, n_5\}$ each of which having probability 1/3 of randomly being selected. We continue with randomly selecting components that misclassified the given instance. The only component that misclassified n_5 is C_4 so the probability that we randomly select C_4 over the components that misclassified n_5 is 1/1.

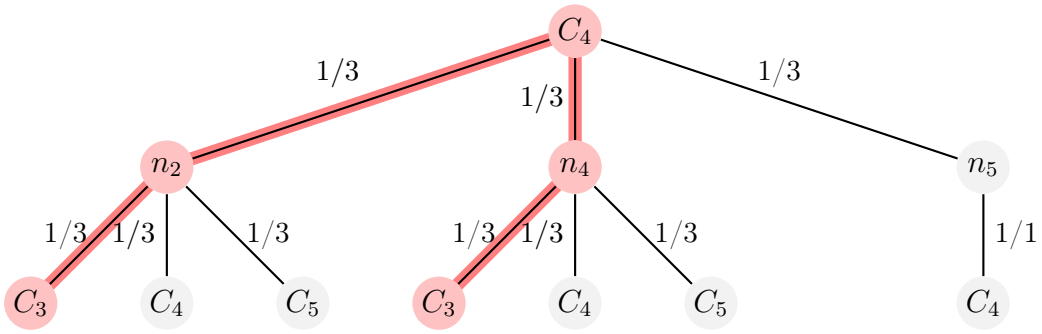

Figure 4.3: Bayesian Tree diagram for $CC_{4,3}$ where the routes to be followed are marked.

We calculate the coverage of C_4 over C_3, $CC_{4,3}$, by following all of the paths that starts with C_4 and ends with C_3.

$$CC_{4,3} = \frac{1}{3} * \frac{1}{3} + \frac{1}{3} * \frac{1}{3} = 0.22$$

In CCP, we only consider the decoupling scores of the components, since they indicate the uniqueness of the components' misclassification errors. For the decoupling scores, we follow the same routine but this time we follow the paths which end with the initial component. Figure 4.2 shows the Bayesian Tree diagram for the decoupling value of component C_4 i.e. σ_4.

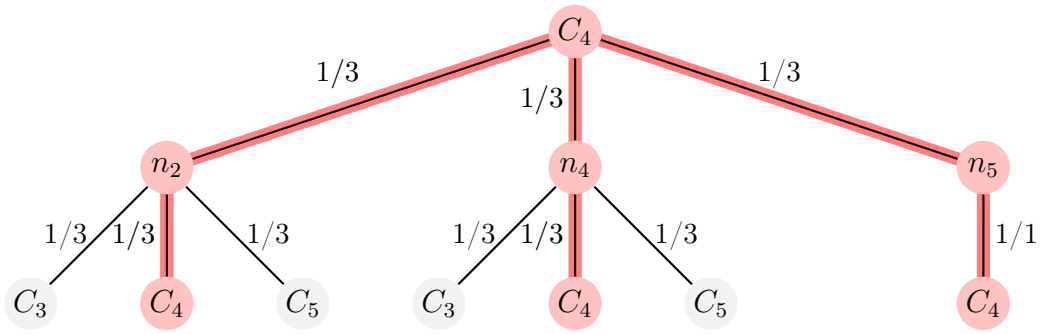

Figure 4.4: Bayesian Tree diagram for σ_4 where the routes to be followed are marked.

$$CC_{4,4} = \sigma_4 = \frac{1}{3} * \frac{1}{3} + \frac{1}{3} * \frac{1}{3} + \frac{1}{3} * \frac{1}{1} = 0.56$$

If a component correctly classified all of the instances, CC values of that component become 0. However, in ensemble pruning, we would like to give such components higher scores so that we guarantee a spot in the pruned ensemble. For that reason, we assign 1 for the decoupling scores of the perfect components.

Let us assume that we would like to prune this toy example with 5 components where $\varphi = 3$. After generating δ, we calculate each component's decoupling values.

$$\sigma_1 = 1$$

$$\sigma_2 = 1$$

$$\sigma_3 = 0.388$$

$$\sigma_4 = 0.611$$

$$\sigma_5 = 0.388$$

Finally, we select φ components with highest decoupling scores i.e. $\{C_1, C_2, C_4\}$. For the pruned ensemble ξ', δ is given in Figure 4.2 and it can be seen in the resulting ensemble components do not make misclassification error on the same instances.

$$\delta = \begin{array}{c} \begin{matrix} n_1 & n_2 & n_3 & n_4 & n_5 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix} \end{array} \begin{matrix} C_1 \\ C_2 \\ C_4 \end{matrix}$$

Figure 4.5: σ for the pruned ensemble ξ' where each column represents a component from the pruned ensemble and each column represents a data instance in the last data chunk.

Algorithm 2 `CCP`: Cover Coefficient-based Pruner

Require: $\mathcal{D}$: data stream, ξ: ensemble, φ: pruned ensemble size
Ensure: ξ': pruned ensemble

Initialize ρ as an empty FIFO buffer of size N.
Initialize $\hat{\rho}$ as an empty FIFO buffer of size $K \times N$.
for $(X, y) \in \mathcal{D}$ **do**
  $\hat{\rho}_k$. append $(h_k(X))$ for $\forall k$ { Preliminary Phase }
  ρ. `append` (y)
  if prune **then**
    { Generate δ }
    Initialize δ as a zero filled array of size K x N
    for $k \in K$ **do**
      $\delta_k[\hat{\rho}_k \neq \rho] = 1$
    end for
    Remove all-0 and all-1 columns from δ
    Initialize σ as zero filled array of size K
    { Calculate decoupling scores }
    $sum_r = sum(\delta, axis = 0)$ { Row-wise sum of δ }
    $sum_c = sum(\delta, axis = 1)$ { Column-wise sum of δ }
    for $k \in K$ **do**
      $mp\_instances = nonzero(\delta_k)$
      for $n \in mp\_instances$ **do**
        $\sigma_k += (1/sum_r[k]) * (1/(sum_c[n])$
      end for
    end for
    { Ranking and selecting the components }
    $ccp\_ranks = \texttt{argsort}(\sigma)$
    $\xi' \leftarrow$ top ranked φ components based on ccp_ranks
    $\xi \leftarrow \xi'$ { Pruned ensemble in effect }
  end if
end for

Chapter 5

Experiments and Results

5.1 Experimental Design

The proposed method and modifications to existing ensembles are integrated into scikit-multiflow [48] library. The experiments are evaluated prequentially [49]. We report prequential and overall accuracy as performance metrics as well as average memory consumption and execution time.

Throughout the experiments, we use two typical dynamic ensembles (AWE [15] and GOOWE [13]) with Hoeffding Trees [50] as their base classifiers. The models are run on three real-world and three synthetic well-known datasets with concept drifts [51, 52]. A summary of the dataset information is provided in Table 5.1.

All experiments are conducted on a machine with a 2.9 GHz dual-core Intel Core i5 processor and 8GB of 1866 MHz LPDDR3 RAM.

Table 5.1: Summary of datasets used in the experiments

	Name	# Features	# Classes	# Instances
Real	COVTYPE	54	7	581,012
	Poker Hand	10	10	829,201
	Rialto	27	10	82,250
Synth	Moving Squares	2	4	200,000
	Moving RBF	10	5	200,000
	Rotating Hyperplane	10	2	200,000

For memory consumption analysis in the experiments, we define *Mean Memory Consumption Ratio* (μ) which indicates the percentage of average memory the pruned ensemble uses with respect to the original ensemble (Eqn. 5.1).

$$\mu = \frac{\sum_{\forall \text{chunk}} Size(\xi')}{\sum_{\forall \text{chunk}} Size(\xi)} \tag{5.1}$$

In order to measure the statistical significance amongst the methods, we use the *Friedman test with Nemenyi post-hoc analysis* [53]. During statistical significance testing, we compare 3 methods; either AWE or GOOWE and their pruning versions with CCRP and CCP. For Friedman test [54], we choose $\alpha = 0.05$ where the null hypothesis holds, if all measurements are from the same distribution i.e. ${\mathcal{X}_r}^2$ (Eq. 5.2) is greater than selected critical value using α. If we reject the null hypothesis for Friedman test, Nemenyi post analysis is applied to compare the significance of the methods where *Critical Distance*, CD, is calculated according to Eq. 5.3.

$${\mathcal{X}_r}^2 = \frac{12}{\mathcal{D}m(m+1)} \sum R^2 - 3\mathcal{D}(m+1) \text{ where } R \text{ is sum of rankings for a model} \tag{5.2}$$

$$CD = q_{\alpha,m}\sqrt{\frac{m(m+1)}{6|\mathcal{D}|}} \tag{5.3}$$

For our experiments, CD is calculated as 1.353 with $\mathcal{D} = 6$, $m = 3$ and $q_{\alpha,m} = 2.344$.

5.2 Experimental Results

5.2.1 Effect of CCRP and CCP on Predictive Performance, Memory Consumption and Execution Time

These experiments demonstrate the effect of pruning on predictive performance, memory efficiency and execution time. Throughout these experiments, pruning is performed whenever the ensembles are full, i.e. when the ensemble size reaches K. After pruning, the ensembles continue to grow until they reach the maximum size K, where pruning takes place again. This process continues indefinitely. We perform experiments with AWE and GOOWE where $K = 30$ to investigate the effect of CCRP and CCP on the ensembles.

Since right up to the first pruning the components in the ensemble are same for both original method and the method integrated with pruning, we first investigate the change in behavior of the ensemble after the first pruning takes place. Figures 5.1 and 5.2 shows the effect of initial pruning on both real and synthetic dataset. By examining the figures, we can say that pruned ensembles' overall performance is better than the original ensemble for both of the pruning methods. By looking at the plots of AWE on Moving Squares dataset, it can be observed that CCP is more robust than CCRP based on the steep decrease in accuracy with CCRP between 3000th and 4000th instances.

Table 5.2, shows the effect of CCRP and CCP on the predictive performance of AWE and GOOWE. When we perform the Friedman test for accuracy results

(Eq. 5.2), we see that pruning does not hurt accuracy, for both pruning methods.

Table 5.3 shows the memory consumption ratio of pruning methods for the same experiment. Notice that CCRP uses at most 84% and at least 23% less memory where CCP reduces memory consumption by at most 90% and at least 21%. Figures 5.3 and 5.4 presents the results of Nemenyi post hoc analysis for memory consumption. While Nemenyi post hoc analysis shows no statistical significance for CCP with AWE and CCRP with GOOWE, at least 21% reduction on memory consumption is still notable for data stream classification.

Execution time ratios of pruning methods over original classification methods are given in Table 5.4. Pruning methods can reduce the execution time by up to 49%. For AWE, both pruning methods statistically significantly reduce the execution time (See Figure 5.2.1). Although, Nemenyi test shows there is no statistical difference in the execution time between GOOWE and GOOWE with CCRP, it should be noted that CCRP reduces the execution time at least by 32%.

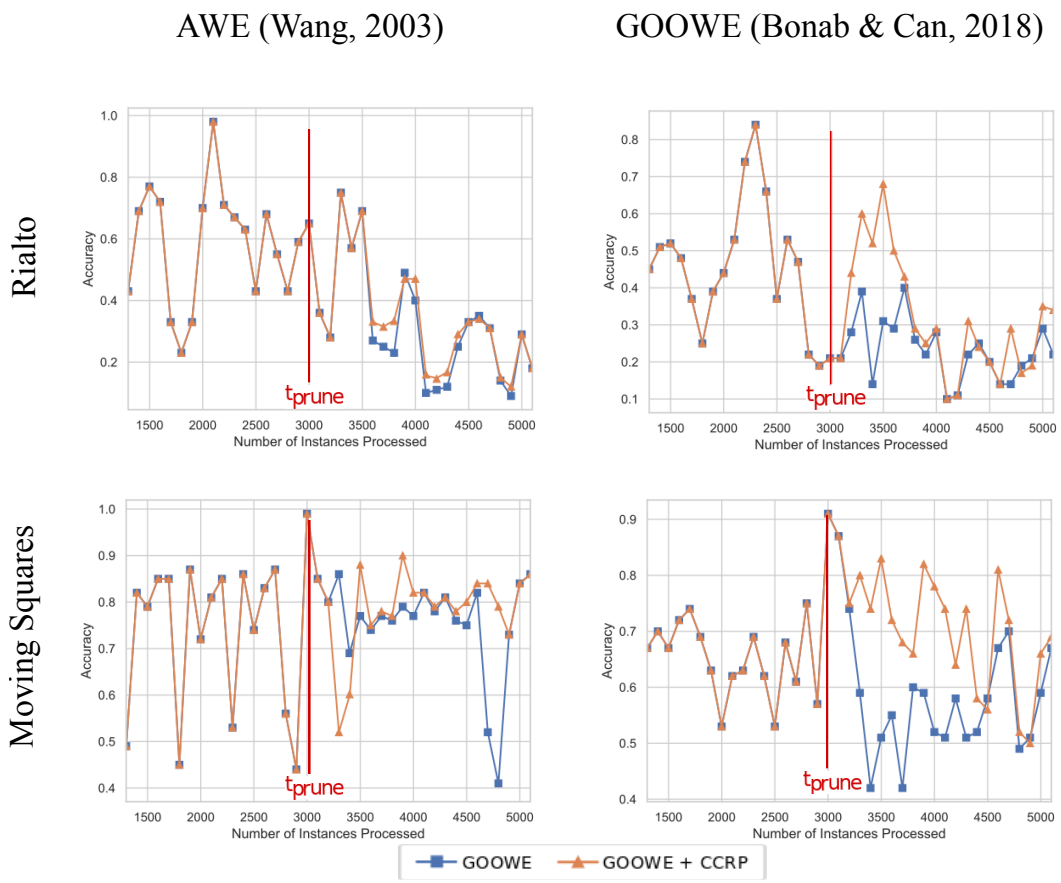

Figure 5.1: The impact of CCRP ($\varphi = L$) on the prequential accuracy over one real and one artificial dataset. Orange lines with triangles represent the CCRP applied models and blue lines with squares represents the models without CCRP. The first occurence of pruning is denoted with red vertical line which is labeled t_{prune}.

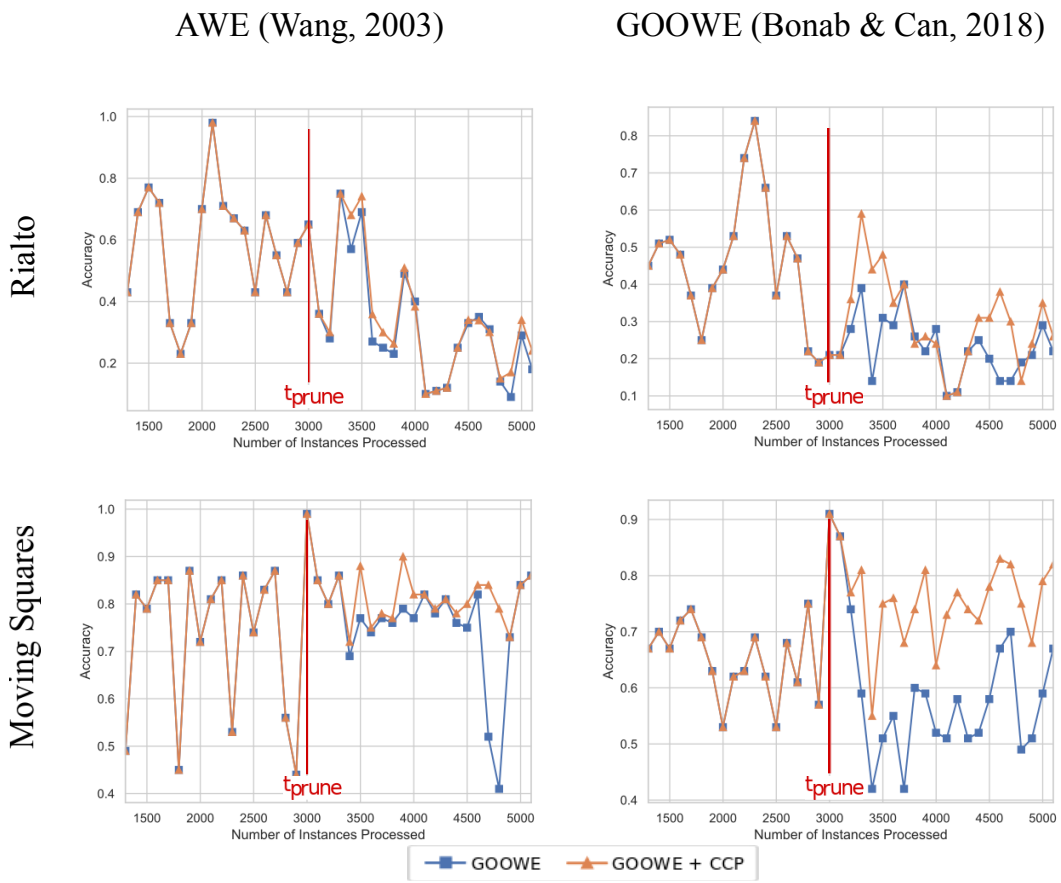

Figure 5.2: The impact of CCP ($\varphi = L$) on the prequential accuracy over one real and one artificial dataset. Orange lines with triangles represent the CCP applied models and blue lines with squares represents the models without CCP. The first occurence of pruning is denoted with red vertical line which is labeled t_{prune}.

Table 5.2: Predictive Performance

	AWE			GOOWE		
	Original	CCRP	CCP	Original	CCRP	CCP
COVTYPE	0.538	0.536	0.531	0.830	0.805	0.803
Poker Hand	0.542	0.608	0.549	0.688	0.672	0.671
Rialto	0.490	0.492	0.495	0.423	0.418	0.426
Moving Squares	0.782	0.767	0.778	0.639	0.692	0.716
Moving RBF	0.532	0.536	0.535	0.531	0.523	0.526
Rotating Hyperplane	0.893	0.890	0.888	0.884	0.889	0.886
Avg. Rank	2	1.834	2.167	1.667	2	2.334

Table 5.3: Mean Memory Consumption Ratio, μ, (Eqn. 5.1) of Pruning Methods (See Table A.1)

	AWE		GOOWE	
	CCRP	CCP	CCRP	CCP
COVTYPE	0.69	0.65	0.16	0.10
Poker Hand	0.69	0.79	0.45	0.32
Rialto	0.77	0.78	0.53	0.58
Moving Squares	0.50	0.51	0.47	0.44
Moving RBF	0.44	0.50	0.35	0.21
Rotating Hyperplane	0.62	0.68	0.34	0.24

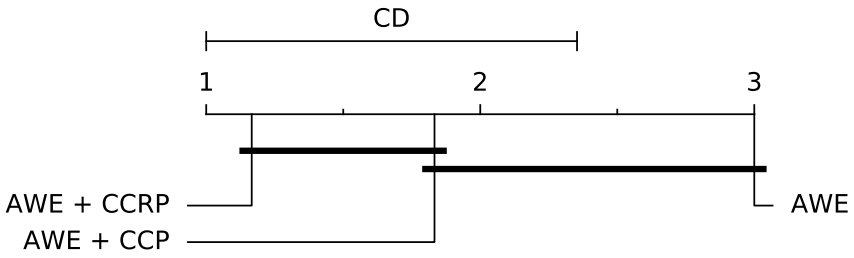

Figure 5.3: Critical distance diagram of average memory consumption of original AWE, AWE with CCRP and CCP (See Table A.1).

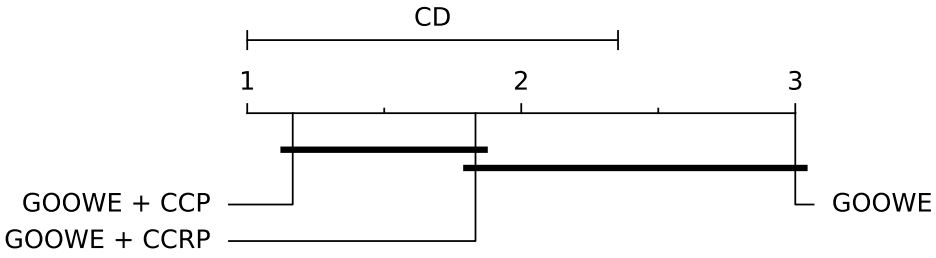

Figure 5.4: Critical distance diagram of average memory consumption of original GOOWE, GOOWE with CCRP and CCP (See Table A.1).

Table 5.4: Execution Time Ratio of Pruning Methods with Respect to Original Ensemble (See Table A.2)

	AWE		GOOWE	
	CCRP	CCP	CCRP	CCP
COVTYPE	0.70	0.61	0.70	0.68
Poker Hand	0.57	0.54	0.87	0.61
Rialto	0.72	0.76	0.65	0.64
Moving Squares	0.60	0.61	0.56	0.57
Moving RBF	0.63	0.66	0.61	0.61
Rotating Hyperplane	0.61	0.60	0.53	0.51

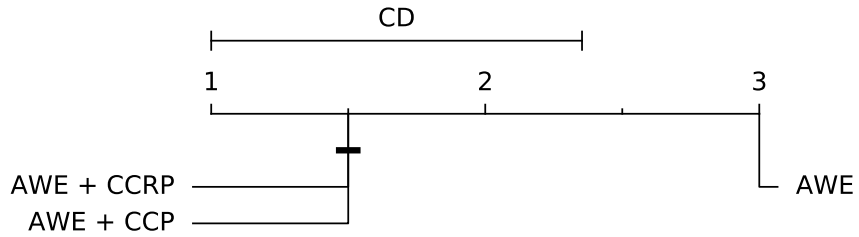

Figure 5.5: Critical distance diagram of execution time of original AWE, AWE with CCRP and CCP (See Table A.2).

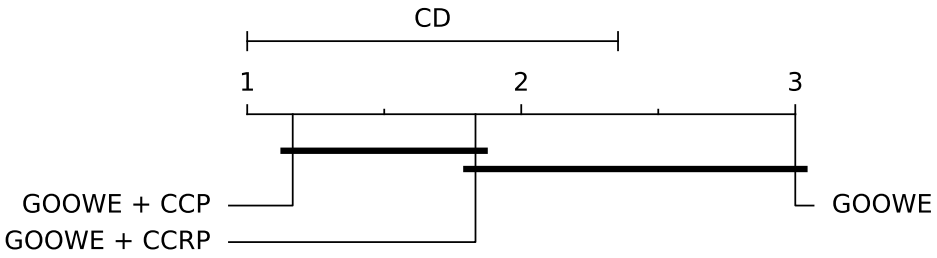

Figure 5.6: Critical distance diagram of execution time of original GOOWE, GOOWE with CCRP and CCP (See Table A.2).

5.2.2 Effect of Pruning with Concept Drift Detection

On previous experiments, pruning was performed when the ensemble reaches its maximum size ($K = 30$) which indicates a randomness. Here, we examine the effect of pruning when performed after a concept drift is detected. As for concept drift detection methods, we use ADWIN [55] and DDM [56]. Throughout the experiments, concept drifts detected by ADWIN and DDM are fed to the models with CCRP and CCP. Hence, we examined only the predictive performance of the models.

It can be observed from Tables 5.5 and 5.6 pruning performs better with concept drift detector. Even though Friedman Test shows no significant difference with ADWIN, pruned ensembles achieve the highest accuracy for 8 out of 12 cases and on par accuracy result for 1 case. Similarly pruned ensembles are better performing with DDM where they achieve the highest accuracy for 9 cases.

When we compare the two pruning methods, CCRP is slightly better performing than CCP. CCRP performs better than original ensemble for 8 cases, where CCP performs better on 7 cases with both ADWIN and DDM. Even though they perform similar with ADWIN, CCRP achives accuracy results better than CCP on 7 cases out of 12 with DDM.

Table 5.5: Predictive performance of CCRP and CCP when ADWIN detect a concept drift

	AWE			GOOWE		
	Original	CCRP	CCP	Original	CCRP	CCP
COVTYPE	0.538	0.542	0.538	0.830	0.814	0.811
Poker Hand	0.542	0.543	0.542	0.688	0.676	0.677
Rialto	0.490	0.496	0.496	0.423	0.428	0.426
Moving Squares	0.782	0.772	0.782	0.639	0.699	0.714
Moving RBF	0.532	0.534	0.534	0.531	0.507	0.513
Rotating Hyperplane	0.893	0.894	0.894	0.884	0.892	0.892
Avg. Rank	2.58	1.58	1.83	1.91	2.25	1.83

5.2.3 Discussion

In Section 5.2.1 we show that pruning significantly reduces memory consumption and execution time of the models without any statistically significant change in predictive performance. Both CCRP and CCP perform relatively similar in case of memory consumption and execution time.

Figures 5.2 and 5.1 shows that pruning increases predictive performance locally; however, when we look at the overall accuracy results there is no statistically significant difference in the overall accuracy. This might be due to the nature of batch-incremental base models; after each data chunk, ensembles are trained with the data from the latest chunk. At first ensembles have younger classifiers which are more susceptible to misclassification errors. Over time, classifiers within the ensemble are trained with more data points and become better in their predictions. Pruning the ensemble when ensemble reaches its maximum size, with its randomness, might remove older and better-trained classifiers prematurely. Pruning should be performed when the knowledge learned by the classifiers become obsolete. In Section 5.2.2 we show that models integrated with pruning achieve better accuracy results with concept drift detection compared to when pruning is done when the ensemble reaches maximum size.

Table 5.6: Predictive performance of CCRP and CCP when DDM detect a concept drift

	AWE			GOOWE		
	Original	CCRP	CCP	Original	CCRP	CCP
COVTYPE	0.538	0.540	0.539	0.830	0.825	0.850
Poker Hand	0.542	0.543	0.546	0.688	0.693	0.691
Rialto	0.490	0.498	0.494	0.423	0.461	0.388
Moving Squares	0.782	0.777	0.776	0.639	0.652	0.657
Moving RBF	0.532	0.534	0.533	0.531	0.501	0.510
Rotating Hyperplane	0.893	0.894	0.893	0.884	0.883	0.883
Avg. Rank	2.33	1.33	2.33	2	1.75	2.25

Chapter 6

Conclusion and Future Work

Data stream classification demands strict time and memory constraints, due to the huge amount of data and robust classifiers, in order to adapt the changes in the distribution of data over time. We propose two ensemble pruning methods for ensemble classifiers on data streams. Our first proposed method, CCRP, is a class imbalance-aware pruning method where pruned ensemble does not lose its ability to classify rare or less-frequent classes. For each target class in the latest data chunk, classifiers are ranked based on their predictive performance for that target class. The final ranking of classifiers is obtained by a rank fusion algorithm that guarantees that the best performing classifiers for each class to be in the selected ensemble. Our second proposed method, CCP, is a misclassified instances aware method which uses the Cover Coefficient concept to find the relations between classifiers based on their misclassification errors. With the Cover Coefficients, it identifies classifiers that make mispredictions on different data instances hence, selected classifiers can cover up each other's misclassification errors. In our experiments, we show that pruned ensembles significantly reduces time and memory consumption of the model without hurting the predictive performance. CCRP can reduce memory consumption up to 84% and execution time up to 47%, where CCP can reduce memory consumption up to 90% and execution time up to 49% without no statistically significant changes in the predictive performance of the models.

CCRP ranks the classifiers for each target classes that appeared in the latest data chunk. For future work, CCRP can be extended to keep the best performing classifier for each target class in each data chunk. with this approach, CCRP can select the best performing classifier for all target classes, even if a target class does not appear in the latest data chunk.

CCP uses decoupling values of classifiers based on their misclassified instances for the latest data chunk; however, two components with identical misclassification errors would have the same decoupling scores. Given the case that, the decoupling scores of these two classifiers are low enough to be chosen for the pruned ensemble, they both be included in the pruned ensemble. Such a case might cause the pruned ensemble to be more inclined to repeat the same misclassification errors in the future. As an additional future work, CCP can be improved to select components for pruned ensemble iteratively, while considering the coupling scores of the candidate components with respect to the already selected components.

As a final remark, integrating the main ideas of the two proposed would result in a pruner which is both class imbalance and misclassified instances aware. The best performing component for each class would be added to the pruned ensemble first, then the remaining slots can be filled according to CCP algorithm.

Bibliography

[1] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits and Systems Magazine*, vol. 6, no. 3, pp. 21–45, 2006.

[2] 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.

[3] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*. CRC press, 2012.

[4] I. Žliobaitė, "Learning under concept drift: an overview," *arXiv preprint arXiv:1010.4784*, 2010.

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

[6] J. Gama, *Knowledge Discovery from Data Streams*. Chapman and Hall/CRC, 2010.

[7] A. Tsymbal, "The problem of concept drift: Definitions and related work," *Computer Science Department, Trinity College Dublin*, vol. 106, no. 2, p. 58, 2004.

[8] Y. Kadwe and V. Suryawanshi, "A review on concept drift," *IOSR J. Comput. Eng*, vol. 17, no. 1, pp. 20–26, 2015.

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

[10] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet, "A survey on ensemble learning for data stream classification," *ACM Computing Surveys*, vol. 50, no. 2, p. 23, 2017.

[11] P. B. Dongre and L. G. Malik, "A review on real time data stream classification and adapting to various concept drift scenarios," in *2014 IEEE International Advance Computing Conference*, pp. 533–537, IEEE, 2014.

[12] C. C. Aggarwal, *Data Streams: Models and Algorithms*, vol. 31. Springer Science & Business Media, 2007.

[13] 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.

[14] 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.

[15] 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.

[16] B. Krawczyk, "One-class classifier ensemble pruning and weighting with firefly algorithm," *Neurocomputing*, vol. 150, pp. 490–500, 2015.

[17] R. M. Cruz, D. V. Oliveira, G. D. Cavalcanti, and R. Sabourin, "Fire-des++: Enhanced online pruning of base classifiers for dynamic ensemble selection," *Pattern Recognition*, vol. 85, pp. 149–160, 2019.

[18] 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, 2013.

[19] 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.

[20] 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.

[21] G. Martínez-Muñoz, D. Hernández-Lobato, and A. Suárez, "An analysis of ensemble pruning techniques based on ordered aggregation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 245–259, 2009.

[22] Z.-H. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: many could be better than all," *Artificial Intelligence*, vol. 137, no. 1-2, pp. 239–263, 2002.

[23] Z.-H. Zhou and W. Tang, "Selective ensemble of decision trees," in *International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing*, pp. 476–483, Springer, 2003.

[24] G. Martınez-Munoz and A. Suárez, "Aggregation ordering in bagging," in *Proc. of the IASTED International Conference on Artificial Intelligence and Applications*, pp. 258–263, Citeseer, 2004.

[25] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, "Ensemble diversity measures and their application to thinning," *Information Fusion*, vol. 6, no. 1, pp. 49–62, 2005.

[26] D. D. Margineantu and T. G. Dietterich, "Pruning adaptive boosting," in *ICML*, vol. 97, pp. 211–218, Citeseer, 1997.

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

[28] X. Yao and Y. Liu, "Making use of population information in evolutionary artificial neural networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 28, no. 3, pp. 417–425, 1998.

[29] 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*, vol. 30, no. 9, pp. 2735–2745, 2019.

[30] 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 twenty-fifth ACM CIKM International Conference on Information and Knowledge Management*, pp. 2053–2056, ACM, 2016.

[31] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley & Sons, 2012.

[32] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[33] J. R. Quinlan, "Simplifying decision trees," *International Journal of Man-machine Studies*, vol. 27, no. 3, pp. 221–234, 1987.

[34] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes, "Ensemble selection from libraries of models," in *Proceedings of the Twenty-first International Conference on Machine learning*, p. 18, ACM, 2004.

[35] G. Martínez-Muñoz and A. Suárez, "Pruning in ordered bagging ensembles," in *Proceedings of the Twenty-third International Conference on Machine learning*, pp. 609–616, ACM, 2006.

[36] Q. Dai, T. Zhang, and N. Liu, "A new reverse reduce-error ensemble pruning algorithm," *Applied Soft Computing*, vol. 28, pp. 237–249, 2015.

[37] G. Giacinto, F. Roli, and G. Fumera, "Design of effective multiple classifier systems by clustering of classifiers," in *Proceedings of the Fifteenth International Conference on Pattern Recognition. ICPR-2000*, vol. 2, pp. 160–163, IEEE, 2000.

[38] R. C. Dubes and A. K. Jain, *Algorithms for Clustering Data*. Prentice Hall Englewood Cliffs, 1988.

[39] A. Lazarevic and Z. Obradovic, "Effective pruning of neural network classifier ensembles," in *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222)*, vol. 2, pp. 796–801, IEEE, 2001.

[40] Z.-H. Z. J.-X. Wu and Y. J. S.-F. Chen, "Genetic algorithm based selective neural network ensemble," in *IJCAI-01: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, Seattle, Washington*, 2001.

[41] M. Coyle and B. Smyth, "On the use of selective ensembles for relevance classification in case-based web search," in *in Proceedings of the European Conference on Case-Based Reasoning*, pp. 370–384, Springer, 2006.

[42] Y. Zhang, S. Burer, and W. N. Street, "Ensemble pruning via semi-definite programming," *Journal of Machine Learning Research*, vol. 7, no. Jul, pp. 1315–1338, 2006.

[43] N. Li and Z.-H. Zhou, "Selective ensemble under regularization framework," in *International Workshop on Multiple Classifier Systems*, pp. 293–303, Springer, 2009.

[44] A. H. Ko, R. Sabourin, and A. S. Britto Jr, "From dynamic classifier selection to dynamic ensemble selection," *Pattern Recognition*, vol. 41, no. 5, pp. 1718–1731, 2008.

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

[46] R. Nuray and F. Can, "Automatic ranking of information retrieval systems using data fusion," *Information Processing and Management*, vol. 42, no. 3, pp. 595–614, 2006.

[47] F. Can and E. A. Ozkarahan, "Concepts and effectiveness of the cover-coefficient-based clustering methodology for text databases," *ACM Transactions on Database Systems (TODS)*, vol. 15, no. 4, pp. 483–517, 1990.

[48] 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.

[49] J. Gama, R. Sebastião, and P. P. Rodrigues, "Proceedings of the issues in evaluation of stream learning algorithms," in *the fifteenth ACM SIGKDD*, pp. 329–338, ACM, 2009.

[50] 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.

[51] A. Bifet, B. Pfahringer, J. Read, and G. Holmes, "Efficient data stream classification via probabilistic adaptive windows," in *Proceedings of the Twenty-eighth Annual ACM Symposium on Applied Computing*, pp. 801–806, ACM, 2013.

[52] V. Losing, B. Hammer, and H. Wersing, "Knn classifier with self adjusting memory for heterogeneous concept drift," in *IEEE ICDM*, pp. 291–300, 2016.

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

[54] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the American Statistical Association*, vol. 32, no. 200, pp. 675–701, 1937.

[55] F. Chu, Y. Wang, and C. Zaniolo, "An adaptive learning approach for noisy data streams," in *Proceedings of the Fourth IEEE International Conference on Data Mining*, pp. 351–354, IEEE, 2004.

[56] L. L. Minku and X. Yao, "Ddd: A new ensemble approach for dealing with concept drift," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 4, pp. 619–633, 2011.

Appendix A

Detailed Experimental Results for Memory Consumption and Execution Time

Table A.1: Memory Consumption (kB)

	AWE			GOOWE		
	Original	CCRP	CCP	Original	CCRP	CCP
COVTYPE	3167.820	2205.370	2084.638	31449.462	5162.183	3130.679
Poker Hand	661.357	461.147	528.817	2210.549	1012.498	713.115
Rialto	3986.451	3095.580	3142.942	5330.678	2871.927	3118.233
Moving Squares	217.957	538.432	610.326	1104.829	520.076	489.205
Moving RBF	919.682	576.801	630.201	3125.348	1104.554	641.015
Rotating Hyperplane	582.347	356.535	395.304	10652.474	3720.235	2565.072
Avg. Rank	2.58	1.58	1.83	1.91	2.25	1.83

Table A.2: Execution time (s)

	AWE			GOOWE		
	Original	CCRP	CCP	Original	CCRP	CCP
COVTYPE	25183.583	17741.878	15377.431	34192.013	24049.263	23277.075
Poker Hand	9492.621	5424.541	5135.0391	12929.893	11257.176	7942.315
Rialto	5308.762	3871.741	4085.281	4834.361	3175.908	3108.973
Moving Squares	863.595	519.228	529.864	680.195	377.754	384.502
Moving RBF	3028.056	1929.770	1999.609	3054.585	2509.122	1855.346
Rotating Hyperplane	1467.627	909.371	883.543	1554.398	829.035	792.314
Avg. Rank	2	1.834	2.167	1.667	2	2.334