UNIVERSITY OF CALIFORNIA,
IRVINE

Collaborative Concept Drift Detection for Formerly Independent Models and Features

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Electrical and Computer Engineering

by

Beverly Abadines Quon

Dissertation Committee:
Professor Jean-Luc Gaudiot, Chair
Professor Nader Bagherzadeh
Professor Phillip Sheu

2023

DEDICATION

I dedicate this work to my family and friends. My parents and siblings for patiently waiting and working hard alongside me. My friends who supplied morale and guidance. Most importantly, I dedicate this work to the people I love the most, Charlo and Richard.

TABLE OF CONTENTS

Page

LIST OF FIGURES **v**

LIST OF TABLES **vii**

ACKNOWLEDGMENTS **viii**

VITA **ix**

ABSTRACT OF THE DISSERTATION **xi**

1 Introduction **1**

2 Literature Review **3**
2.1 Introduction . . . 3
2.2 Drift Definition . . . 4
2.3 Drift Detectors . . . 5
2.3.1 Performance Based . . . 6
2.3.2 Data Distribution Based . . . 8
2.4 Summary . . . 11

3 Definition of the Problem **12**
3.1 Introduction . . . 12
3.2 Technical Debt in Machine Learning Systems . . . 13
3.3 Data-centric approach to Machine Learning . . . 13
3.4 Collaborative Approach to Concept Drift Detection . . . 15
3.5 Summary . . . 17

4 Our Approach: Global Drift Detection with Collaborative Filtering **18**
4.1 Introduction . . . 18
4.2 Related Work . . . 21
4.2.1 Concept Drift . . . 21
4.2.2 Drift detection methods . . . 23
4.2.3 Feature store . . . 23
4.2.4 SHAP . . . 25
4.2.5 Collaborative Filtering . . . 25
4.2.6 Local Detectors . . . 26

4.3 Proposed Concept Drift Detector with Collaborative Filtering for multiple models . . . 27
4.3.1 Detect Drift for each model . . . 28
4.3.2 Augment the labels based on shared features . . . 28
4.3.3 Make the holistic features of the system more resistant to drift . . . 29
4.4 Evaluation . . . 30
4.4.1 Metrics . . . 30
4.4.2 Analysis . . . 30
4.5 Discussion . . . 31
4.6 Summary . . . 32

5 Our Approach: Collaborative Concept Drift Detection with Fast Correlation Based Filtering 33
5.1 Introduction . . . 33
5.2 Proposed Collaborative Concept Drift Detection with Fast Correlation Based Filtering . . . 35
5.2.1 Reasoning behind detecting invariants . . . 36
5.3 Experimental Setup . . . 37
5.3.1 Performance Gain to Update Cost Ratio . . . 38
5.4 Discussion . . . 39
5.5 Summary . . . 40

6 Our Approach: Adaptive Aggregated Drift Detector 44
6.1 Introduction . . . 44
6.2 Preliminary Work . . . 45
6.3 Methodology . . . 46
6.3.1 Adaptive Phase . . . 46
6.3.2 Aggregative Phase . . . 48
6.4 Evaluation . . . 50
6.4.1 Artifical Dataset Configuration . . . 50
6.4.2 Artificial Datasets . . . 50
6.4.3 Real World Datasets . . . 51
6.5 Summary . . . 51

7 Conclusion 55
7.1 Summary . . . 55
7.2 Future Directions . . . 56

Bibliography 57

LIST OF FIGURES

Page

3.1 Assortment of detectors per models in stream 16

4.1 Examples of concept drift compared to (4.1a) either due to (4.1b) changes in class distribution (i.e. fixed space) or (4.1c) changes in $P(\boldsymbol{X})$, (i.e. non-fixed space). Shapes indicate different classes of data. 22
4.2 Performance based detector such as Drift Detector Method (DDM) measures the error rate which is the sum of p and s. T_m is the error rate at time m and T_n is the error rate at time n. Drift is suspected when the error rate is greater than the warning level and confirmed once it reaches the drift level. 24
4.3 Distribution based detectors such as KS tests measures the cumulative distributive function between the reference and current feature distributions. If the CDF are significantly then it indicates covariate drift and either virtual drift or concept drift. 25
4.4 Overview of QuaD in a single-stream, multiple model system. 29

5.1 C2D2 Architecture 36
5.2 SU scores per feature per batch on one of the Agrawal dataset. Drift occurs from instances 2K to 3K (i.e. batch 2). Feat 2 decreases while Feat 3 increases at a crossover point between batch 2 and 3. 39
5.3 SU scores per feature per batch on one of the LED dataset. Drift occurs from instances 2K to 3K (i.e. batch 2). Feat 0, 1, and 2 decrease while Feat 7, 8 and 9 increase at a crossover point at batch 2. 41
5.4 SU scores per feature per batch on one of the RandTree_55 dataset. Drift occurs from instances 1.5 K to 2.5K (i.e. batches 1 and 2) 41
5.5 SU scores per feature per batch on one of the RandTree_100 dataset. Drift injected from instances 1 K to 2K (i.e. batch 1 42
5.6 SU scores per feature per batch on one of the Sea dataset. Drift injected from instances 2 K to 3K (i.e. batches 2) 42
5.7 Mean PGUCR values with penalty of 0.1. Each dataset contained the following number of features: Agrawal (9), LED (24), RandTree_55 (55), RandTree_105 (105), Sea (4). 43
5.8 Post-hoc Tukey HSD test at 99% confidence interval indicates that C2D2 is significantly better than T, MW, CVM, KS, ED, and WD. C2D2 has a mean of 0.43. 43

6.1 A2D2 with its adaptive (exploit) and aggregative (explore) phases. 53
6.2 Post-hoc Nemenyi test at confidence level of 99% of the PGUCR from RandTree100 dataset. Critical distance is 7.144. With respect to FCBF there is a significant difference between it and T, MW, KS, and CVM. 54

LIST OF TABLES

Page

5.1 Parameters for C2D2 38
5.2 Average number of signals for updates on synthetic datasets. Ablation excluded as the count will always be zero. Counts can range from 0 to 9. . . . 39
5.3 Average F1 score on the datasets with 10 samples. The F1 scores of the tests, ablation (ABL), and number of signals are used to calculate PGUCR. 40

ACKNOWLEDGMENTS

Thank you to Charlo and Richard. Charlo, for attending meetings and making every day memorable. Richard, for supporting us and providing the energy to complete this long chapter in life.

I would like to thank Professor Jean-Luc Gaudiot for his persistent encouragement. Additionally, thanks to Professor Athina Markopoulou for inspiring me to start the Society of GEECS and thank you to the second floor department staff. Thank you to the committee members, Professor Nader Bagherzadeh and Professor Phillip Sheu.

I thank IEEE COMPSAC for permission to include Chapter 4 of my dissertation. I thank ICLR for permission to include Chapter 5 of dissertation and similarly to DMLR at ICML for Chapter 6. This work was supported by Graduate Assistance in Areas of National Need (GAANN) under award P200A10052 and by the National Science Foundation under award CCF-176379.

VITA

Beverly Abadines Quon

EDUCATION

Doctor of Philosophy in Electrical and Computer Engineering **2023**
University of California Irvine *Irvine, CA*

Master of Science in Electrical and Computer Engineering **2018**
University of California Irvine *Irvine, CA*

Bachelor of Science in Computer Engineering **2017**
California State University, San Bernardino *San Bernardino, CA*

RESEARCH EXPERIENCE

Graduate Research Assistant **2020–2023**
University of California, Irvine *Irvine, California*

Computer Engineering Intern **June–Aug. 2017**
Monterey Bay Aquarium Research Institute *Moss Landing, California*

Undergraduate Research Student **June–Sept. 2016**
Center for Advanced Functional Materials, CSUSB *San Bernardino, California*

Undergraduate Research Student **July–Aug. 2015**
PRISM, CSUSB *San Bernardino, California*

TEACHING EXPERIENCE

Teaching Assistant **2018–2020**
University of California, Irvine *Irvine, California*

REFEREED CONFERENCE PUBLICATIONS

Adaptive Aggregative Drift Detector **June 2023**
Data-centric Machine Learning Research Workshop at ICML 2023

Collaborative Concept Drift Detection **May 2023**
First Tiny Papers Track at International Conference on Learning Representations 2023

Concept drift detection for distributed multi-model machine learning systems **June 2022**
IEEE 46th Annual Computer Software and Applications Conference (COMPSAC) 2022

ABSTRACT OF THE DISSERTATION

Collaborative Concept Drift Detection for Formerly Independent Models and Features

By

Beverly Abadines Quon

Doctor of Philosophy in Electrical and Computer Engineering

University of California, Irvine, 2023

Professor Jean-Luc Gaudiot, Chair

Unstationary data can cause a change or drift in the machine learning model's context (i.e. understanding of information) and/or concept (i.e. relationship between context and target). Resultantly, the unaccounted effects of unstationary data is referred to as drift. Drift can lead to model performance degradation, despite the lack of change from an optimally performing model prior to drift's occurrence. Many works have been proposed to detect and recover from these moments of drift. Despite the curation of data from the data pipeline, many of these detectors operate per model and per data stream, overlooking the shared data pipeline between these models and their streams. In other words, although models operate independently, they exist in an ecosystem consisting of models, features, and streams integrated altogether. Arguably, there are resources that once considered holistically can help improve our understanding of factors related to drift.

This work focuses mainly on concept drift. The contribution of this dissertation is the advancement made towards adaptive, global drift detection with respect to retraining costs. This will go over: i) creating model associations based on shared features using a method traditionally used for recommendation systems, ii) relating the cost of recovery after retraining from concept drift with respect to performance and iii) creating an adaptive and aggregated approach to detecting drift.

Chapter 1

Introduction

The era of Big Data, where both the volume of information and demand of data processing has escalated to grand scale has necessitated the construction of data pipelines. These pipelines must be capable of collecting, cleaning, and shepherding information in preparation for stages of prediction and analysis. Machine learning (ML) models are imperative for these stages of prediction and analysis. Naturally, the performance of ML models are only as good as the data on which they are trained, where goodness is 1) the reflection of how similar the distributions of the training data is with respect to streams of data on which they are examining and 2) how well the context of data encompasses the knowledge for prediction or analysis. Having said this there are moments where 1) or 2) are violated. This is due from the nature of dynamic and unstationary data. The distribution of data can change (violating premise 1)) or a confounding variable can be introduced that affects the *concept* (i.e. relationship between the context developed and the purpose of the model), violating premise 2).

Unstationary data can cause a change or *drift* in the model's understanding of information or its developed *context*. Resultantly, the unaccounted effects of unstationary data is referred

to *drift* of which there are several types as indicated in **Chapter 2**. Drift can lead to model performance degradation, despite the lack of change from an optimally performing model prior to drift's occurrence.

Many works have been proposed to detect and recover from these moments of so called drift. Despite the curation of data from the data pipeline, many of these detectors operate per model and per data stream, overlooking the shared data pipeline between these models and their streams. In other words, although models operate independently, they exist in an ecosystem consisting of models, features, and streams integrated altogether. Arguably, there are resources that once considered holistically can help improve our understanding of factors related to drift.

This work focuses namely on *concept drift* which is a result from a violation of premise 2). The contribution of this dissertation is the advancement made towards adaptive, global drift detection with respect to retraining costs. **Chapter 4** proposes how to create model associations based on shared features using a method traditionally used for recommendation systems. **Chapter 5** relates the cost of recovery after retraining from concept drift with respect to performance and **Chapter 6** combines both works to create an adaptive and aggregated approach to detecting drift. **Chapter 7** discusses future directions hence forth from the contributions of this dissertation.

Chapter 2

Literature Review

2.1 Introduction

A major problem in the constantly changing big data environment is the ability to ensure reliable data-driven decisions and predictions. The vast and rapid collection of information under dynamic data streams, however, elicits unstationary behaviors that endanger said reliability. Because many machine learning strategies have a model-centric view which prioritizes fitting models onto a static dataset with performance (e.g. accuracy, F1 score) as the main objective, they mistakenly assume that the distribution of training data (i.e. context) and the relationship learned between features and labels (i.e. concept) remains stationary. In reality, both context and concept can drift. This chapter discusses the types of drift that affect reliability and performance as well as the state of the art methods to address these drifts.

2.2 Drift Definition

Drift can be seen as an unforeseen change from a previously established pattern. In the machine learning setting, the factors that can change are 1) the distribution of features $p(\boldsymbol{X})$, 2) distribution of labels $p(y)$, and 3) relationship between features and labels $p(\boldsymbol{X}, y)$. The distribution of features and distribution of labels describe the **context** or patterns learned from fitting a model on a dataset. The relationship between features and labels describe the **concept**, which is used to predict the outcome of a label, y given a set of features, $\boldsymbol{X}$.

Under the data streaming setting, a data window D_t at time t consists of data instances, $d = (\boldsymbol{X}, y)$. Where $\boldsymbol{X}$ is an ordered list of qualitative (nominal, ordinal, or binary) and/or quantitative (discrete or continuous) features. The label, y, can be one of l classes such that $y \subset \{C_1, C_2, ...C_l\}$. D_t of m number of d is $D_t = \{d_1^t, d_2^t, ...d_m^t\} = \{(\boldsymbol{X}^t_1, y_1^t), (\boldsymbol{X}^t_2, y_2^t), ...(\boldsymbol{X}^t_m, y_m^t)\}$. The joint probability of features and labels at time t is $p(\boldsymbol{X}, y)_t$.

The following definitions describe the different types of drift in relation to $p(y), p(\boldsymbol{X})$, and $p(\boldsymbol{X}, y)$ with respect to t and t' where $t \neq t'$.

Types of Drift

- **Virtual Drift**: $p(\boldsymbol{X})$ or $p(y)$ changes but $p(\boldsymbol{X}, y)_t = p(\boldsymbol{X}, y)_{t'}$.
- **Label Drift** [37]: Class labels evolve such that a class disappears or contrarily new classes emerge (i.e. $p(y)_t \neq p(y)_{t'}$) which may or may not result in $p(\boldsymbol{X}, y)_t \neq p(\boldsymbol{X}, y)_{t'}$
- **Covariate Shift** [36]: $p(\boldsymbol{X})_t \neq p(\boldsymbol{X})_{t'}$ which may or may not result in $p(\boldsymbol{X}, y)_t \neq p(\boldsymbol{X}, y)_{t'}$
- **Concept Drift** [20]: $p(\boldsymbol{X}, y)_t \neq p(\boldsymbol{X}, y)_{t'}$ and is caused by a change in relationship between $\boldsymbol{X}$ and y that is not necessarily describable solely by label drift or covariate

shift. Concept drift is further describable in terms of **speed, severity, distribution,** and **recurrence.**

Hu et al describe the trade-offs between cost of labeled data and performance in detection [23]. Due to the sheer volume and indefinite size of data streams, labeled data becomes explosively expensive and to a certain degree, impractical in terms of training and predicting expectations. Streaming data must be able to operate with sparsely labeled data to have practical use and detectors must adapt to that constraint [35].

There are two common types of detectors: **performance based** and **data distribution based.** Performance based detectors monitor drops in performance thresholds (e.g. F1 score, accuracy, error rate). Data distribution based detectors monitor divergences between current and historical distributions. Performance based detectors have the advantage of detecting different types of drifts while data distribution based detectors have the advantage of not relying on labeled data. Under this generalization, performance based detectors are more performance effective while data distribution detectors are more cost effective.

The reason that performance based detectors can outperform data distribution based detectors are the fact that labels can elucidate a dimension undetectable through unlabeled data in the event of virtual drift or concept drift.

2.3 Drift Detectors

The following sections describe a generalization or design pattern for state of the art detectors. In this case, we highlight them as performance based detectors and data distribution based detectors. Streams must be processed in the event that data arrives. Most detectors process their data in windows, where the data they are predicting on is the current stream and historical streams before it are seen as reference streams. They are processed

prequentially, meaning that they predict on the current stream, but update their model on the reference stream. As they process across the streams, current streams become part of the reference streams.

2.3.1 Performance Based

These detector operate on the assumption that as long as there is no drift, the error rate should decrease as the number of samples increase. They monitor the error rate which can be considered as its performance rather than the data stream directly.

Drift Detector Method (DDM) [19] is one of the first algorithm to specify the concept drift detection alert level and drift level. DDM determines if there has been a significant increase in the overall online error rate within the time span. If the observed error rate confidence level surpasses the alert level, DDM begins building a new learner while continuing to use the previous learner for predictions. The previous learner will be replaced by the new learner for additional prediction problems if the change has reached the drift level. The likelihood of an observer being false, p_i, is used to calculate the *error rate* for each point i in the series, and the *standard deviation* is given by $s_i = \sqrt{(p_i(1-p_i)/i)}$.

If the probability distribution remains constant under a static environment, then the confidence interval for p with $n > 30$ cases is about $p_i \pm \alpha s_i$, where α is a parameter set by the confidence interval. During the learning algorithm's training phase, two registers, p_{min} and s_{min}, are managed by the drift detection mechanism.

Early Drift Detector Method (EDDM) [7] was developed to extend DDM and its ability to improve detection of gradual concept drift while maintaining its ability to detect abrupt drift. Rather than monitoring the error rate up to a certain confidence level (as was the case for DDM), EDDM monitors the distance between two error classifications. As the learning

model is trained with more samples, predictions are expected to improve while the distance between errors (measured in error rates p'_i and standard deviation s'_i) increase. When at least 30 classification errors have occurred, the approach determines the max distance of p'_{max} and its respective s'_{max}. Similarly, to DDM it determines its detections based on thresholds Eq.(2.1), where α is an adjustable parameter, related to the confidence interval. Because it is comparing the ratio of the current distance with the max distance, α represents 90% - 95% of the sample distribution.

$$(p'_i + 2s'_i)/(p'_{max} + 2s'_{max}) < \alpha \tag{2.1}$$

ADaptive WINdowing (ADWIN) [8, 10] monitors the average values and predicts the expected value, μ_t at time t_i under distribution streams, D_t within windows, W, of varying sizes. W shrinks when the observed average μ_W differs from μ_t for $t \in W$and grows when it has not. Its only parameter is the confidence bound, δ and assumes that the values are always within 0 and 1. Using exponential histograms [16], it utilizes W of length n samples using only O(logn) memory and O(logn) processing time.

Because it does not require setting window sizes directly and recomputes itself online, ADWIN has been accompanied into ensemble algorithms for detection such as ADWIN Bagging, DDM, or even other classifiers as simple as Naive Bayes. ADWIN serves as the change detector and weight estimator for the boosting method, where the worst classifier is dropped upon changes.

2.3.2 Data Distribution Based

Rather than comparing the performance of the a detector, data distribution methods detect whether there are statistical differences in two distributions, the current stream and reference stream. In a way these methods extend the windowing approach, where streams are divided into windows with previously seen values serving as the historical reference window. The following methods detect drift based on changes between windows, but they differ in the type of information they monitor.

KS Test [17, 47] is non-parametric in form and compares the location and shape between probability distributions, $F_{A,n}$ and $F_{B,m}$ across samples A with n observations and samples B with m observations. Their empirical distribution functions are computed as:

$$F_n(t) = \frac{1}{n} \sum_{i=1}^{n} 1\{x_i \leq t\} \tag{2.2}$$

where $(x_1, ...x_n)$ are independent and identically distributed (i.i.d) random variables in the real numbers domain.

Concept drift can be detected when the KS test rejects the null hypothesis at α if:

$$KS_{stat} > c(\alpha)\sqrt{\frac{n+m}{nm}}, \tag{2.3}$$

KS_{stat} is the KS statistic (i.e. obtained p-value), $c(\alpha)$ is the confidence interval at α, and the product on the right side of the inequality is the obtained target p-value. Lastly, KS_{stat}

is defined as:

$$KS_{stat} = \max_{x \subset A \cup B} |F_A(x) - F_B(x)| \tag{2.4}$$

Mann-Whitney U-rank (MW) or **Wilcoxon Signed Rank** was developed by Wilcoxon [53] is a nonparametric test that monitors the difference between two windows of paired values, such that $H_0 : WS+ = WS-$. It computes the difference between paired values and ranks them without regarding the sign and excluding tied values. Followed by summing the ranks in terms of sign to create $WS+$ and $WS-$. Lastly, it identifies if the sums are are equal or significantly different (rejecting H_0). The test statitic is $min(WS+, WS-)$. Overall, it tests via rankings assigned to the data points rather than the actual observed values.

Cramer Von Mises (CMV) [29] is a refinement of KS test where instead of calculating the test statistic as Eq.(2.4), it calculates it as a quadratic statistic as a weight function. Specifically, one has the CMV statistic under Eq. (2.5). It is the measurement of the mean squared difference in cumulative distribution functions.

$$CMV_{stat} = \int [F(x) - F_n(x)]^2 \, dF(x) \tag{2.5}$$

Energy Distance (ED) [41] compares the distance between random vector distributions. Analogous as to how potential energy between objects should be zero *iff* the gravitational centers of two objects coincide and increases as they diverge. Again, it compares the cumulative distribution functions in terms of Euclidean norm.

Rizzo [41] considers two independent random vectors, X and $Y \in R^d$. A random variable X' is an i.i.d copy of X and Y' is an i.i.d of Y. The squared energy distance between random

variables and vectors is the following:

$$E_{stat}(F,G) = 2E||X-Y|| - E||X-X'|| - E||Y-Y'|| \geq 0 \tag{2.6}$$

where F and G are said cumulative distribution functions.

To elaborate on the analagy, distributions are equal if the distance is nearly 0 (indicating that $F = G$) and are unequal otherwise. CMV is closely related to ED except that the CMV is univarite while ED is multivariate. To test for multiple features, CMV, similar to KS, must run multiple tests per feature, while ED can process them as a single test seen as a vector of features.

Wasserstein Distance (WD) or Kantorovich-Rubinstein or Earth mover's distance [45, 51] measures the amount of "work" (also analogous to physics) to transport one distribution to another. WD is a joint probability distribution whose marginals are given by paired measurements (x, y), where the resulting matrix, $\mathbf{T}$ forms the transport plan.

The 1-WD measures the minimum of all the transport plans multiplied by the cost of the transport plan (i.e. the minimum expected cost of work).

$$WD_p(f,g) = inf(E||X-Y||^p)^{1/p}, p \geq 1 \tag{2.7}$$

2.4 Summary

Concept drift is the change in relationship between the independent variables used to train a model and develop its context with respect to the target variable to be predicted. Detectors, which indicate when drift occurs and when a model is subjected to degradation, can be categorized into two groups: performance based or data distribution based. The detectors mentioned focused on comparing difference either via error rates or via cumulative distribution functions. Upon detection, it was implied that an update or retraining on a model is necessary.

Chapter 3

Definition of the Problem

3.1 Introduction

The works mentioned in Chapter 2 highlight how they address detection with an implicit means of updating, but little work has been done to relate the factors of drift as part of a system. Although ensemble techniques such as bagging or boosting have been utilized to extend the knowledge gained from single detectors, they still detect on a single stream. This chapter reviews how systems have developed into integrated subsystems with shared resources and how we should adopt a data-centric approach. This chapter concludes with the argument of why there needs to be a collective *system* approach to concept drift detection.

3.2 Technical Debt in Machine Learning Systems

"Developing and deploying ML systems is relatively fast and cheap, but maintaining them over time is difficult and expensive" [43]. There are long term costs for short term improvements. The idea of paying the hidden technical debt is to "enable future improvements, reduce errors, and improve maintainability" within an environment which requires vast growth in terms of complexity and functionality. Machine learning systems not only have the problems of maintenance with respect to non-machine learning applications, but have specific costs that are not only 1) more difficult to detect as these costs accrue at the *system level* but 2) are also unable to be paid of by traditional means of making code level corrections. By treating models as black boxes made of calibrated layers that are haphazardly reused or chained together, we may unintentionally erode abstraction and assumptions imposed by the model. Other challenges include data consistency scalability, robustness and minority-class and multi-class issues for systems which have been known to have dozens or hundreds of models running simultaneously [15, 44].

The short term improvements can be viewed as the immediate integration of models into a machine learning system. More abstractly, I extend this example to our shortsightedness to view models as stand alone components, disregarding their place as an integrated component that is part of a larger system.

3.3 Data-centric approach to Machine Learning

Much of the shortsightedness and accruence of technical debt is from the convenient, yet expensive model-centric perspective. The model-centric view prioritizes the tuning of model parameters on the assumption that the data it is trained on will indefinitely represent the streams that it predicts upon. The convenience from such a perspective allows one to explore

and optimize not only parameters but also other competitive classifiers or algorithms that can engender best performance. For many works, performance is the main objective, but as indicated in the previous section, there are many more factors to consider. Additionally, fixating on only model performance can create more costs.

One cost is the creation of siloed data. Developers like to pretend that only they have access to their data. Again, models are only as good as their data. They end up creating closed access to their data to be used specifically for a single model. Alternatively, they may also create copies of data if they do not have privileges to limit access. The outcome is siloed data, which are large quantities of isolated information or even redundancies. Both incur the cost of higher memory usage at the expense training one model at a time.

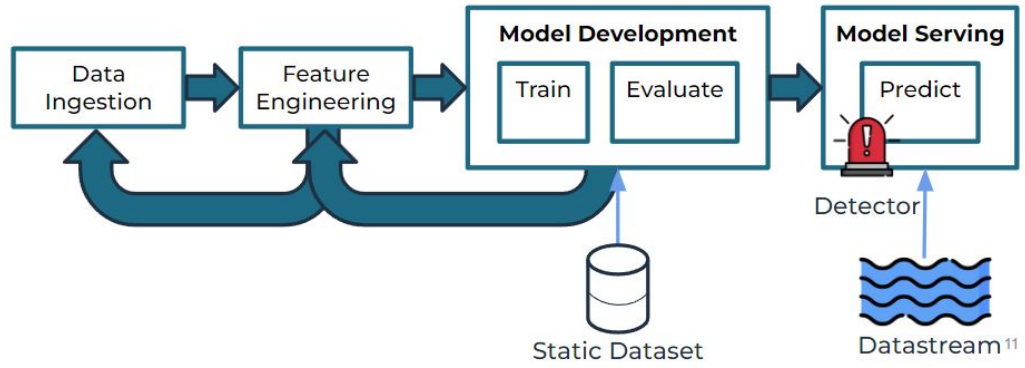
Another weakness is the susceptibility to drift. By assuming that streams are stationary and that training data will always represent live data, models will always face the consequences of drift (Chapter 2). This emphasizes the importance of detectors.

Overall because many of these costs are related to data, it would only make sense to go towards a *data*-centric approach. There have been progress towards data-centric mindsets that have made strides in topics not limited to transfer learning, multi-task learning, crowd sourced labeling, semi-supervised learning, weak supervision, active learning, and data assimilation [46], but as expected many applications and users are unwilling to shift the entirety of their system.

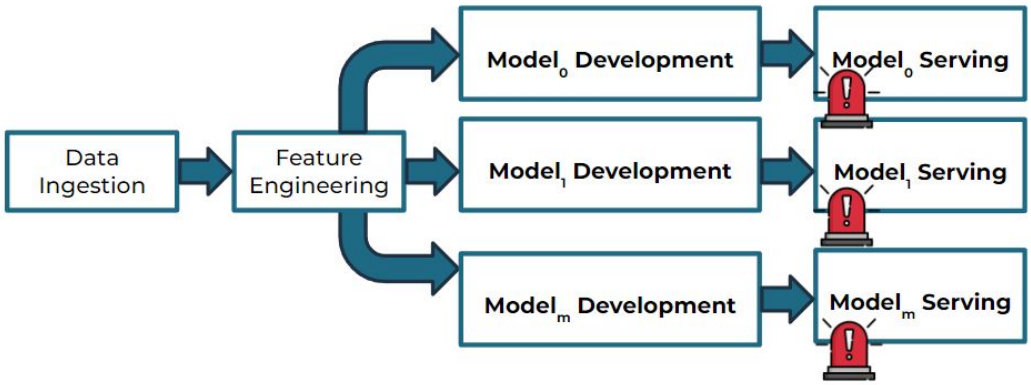
3.4 Collaborative Approach to Concept Drift Detection

As long as systems fail to adopt a data-centric view, we must come up with a way to bring about the advantages of data-centric design to a model-centric system. One method is to no longer view models and the streams they act on as individual entities. By respecting the fact that they operate with shared dependencies with entangled costs, we will not only be able to develop a collective understanding of models within a system, but extend it beyond further and propose a means to have formerly independent models work together collaboratively.

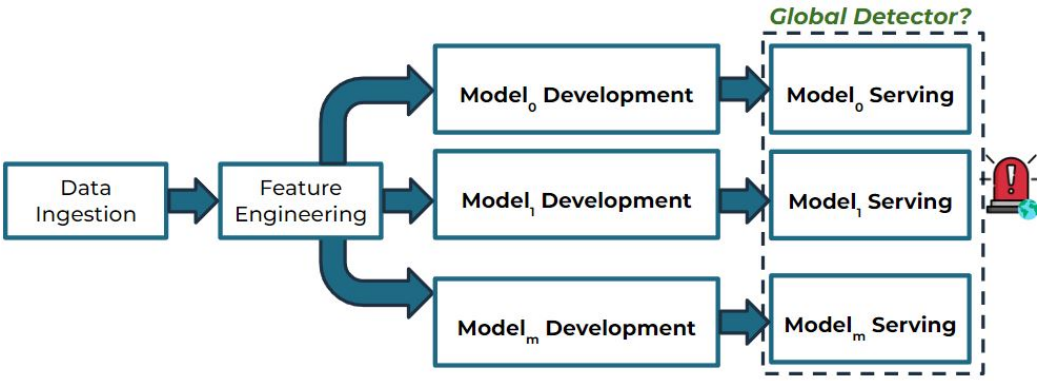
With machine learning models being susceptible to drift and continuing to prioritize the model perspective, which results in the under utilization of resources, this dissertation *advocates for* and *introduces* progress towards collaborative drift detection as shown in the figure below.



(a) Single detector for single model



(b) Multiple single detectors for multiple models



(c) Global detector for multiple models

Figure 3.1: Assortment of detectors per models in stream

3.5 Summary

This chapter reviewed the technical debt of machine learning systems which are often tied to the preference of a model-centric view over a data-centric view. There must be a means to bring about the advantages of data-centric design to systems as they continue to uphold the status quo of model-centric design. This work advocates for the creation of models, features, and streams working together to form collaborative concept drift detection.

Chapter 4

Our Approach: Global Drift Detection with Collaborative Filtering

4.1 Introduction

Many works focus on optimizing machine learning (ML) models during their training phase, but fail to account how these models adapt into their model-serving phase once they are deployed into real world applications (e.g. online sentiment analysis, intrusion detection, fraud detection, etc). In this phase models must process through streams of data that can evolve over time and distort the relationship between incoming data, $\boldsymbol{X}$, and target variables (e.g. class labels for classification, regression, or unsupervised problems), $\boldsymbol{y}$. If left unaccounted, models that performed optimally prior to this change ceases to be optimal, despite the fact that the model itself is unmodified, and results in the phenomena known as *concept drift*. Concept drift is defined as an unexpected change in the *context* or joint distribution of $P(\boldsymbol{X}, \boldsymbol{y})$, such that $P_t(\boldsymbol{X}, \boldsymbol{y}) \neq P_{t+1}(\boldsymbol{X}, \boldsymbol{y})$ for time t.

Many forms of drift detection and recovery models have been proposed to mitigate con-

cept drifts for online data streams and can be categorized as performance-based and data distribution-based approaches. Designing such approaches is non-trivial as there is a trade-off between performance and cost-efficiency [24]. Performance-based approaches monitor a model's performance measurements, such as accuracy, F-measure, precision, and recall. They have the advantage of detecting all types of drift (e.g. gradual, incremental, abrupt, fixed space, or non-fixed space), but can only process labeled data, which is cost inefficient. Expecting most of the data to be labeled is impractical and expensive in terms of the scale of ML applications. Rather than measuring classifier performance metrics, data distribution-based approaches track changes in location, density, and range of the data itself. These approaches have the advantage of being able to process both labeled and unlabeled data, but are limited in the types of drift they can detect. For example, they cannot detect fixed space drift for unlabeled data without combining multiple approaches together. Hence, the trade off is that the ability to detect all types of drift is dependent on whether unlabeled data or labeled data are used.

Interestingly, these drift detectors only demonstrate how to react to drift acting on a single model and/or single pair of source and target stream. They do not take into account the possibility of multiple live models acting on different streams. In the real world setting, multiple models can run simultaneously across streams and share subsets of training data. This idea of managing offline training data and online streams for multiple models and creating logically centralized features based on physically distributed data has been gaining popularity, so much so that it has given rise to several feature stores [2, 1, 3]. The purpose of these feature stores are to create an abstraction layer between the offline and online data and promote data reuse by removing data silos among models, reproducibility in training data, and mitigate training-serving skews. Additionally, if feature stores are for maintaining and deploying ML models in production, then SHAP values are a means to promote explainable ML. SHAP values apply cooperative game theory to distinguish each feature's contribution to a model.

This paper proposes leveraging the advantages of emerging features stores in order to improve drift detection on unlabeled, dynamic data streams across multiple ML models. The purpose of this work is two-fold.

Firstly, we introduce Drift Detection on Distributed Datasets (QuaD), which combines classical drift detectors to make use of labeled and unlabeled data, and create local context (i.e. per live model) and global context (i.e. across multiple models). Secondly, we propose using feature store entities, SHAP values, and Collaborative Filtering (CF) to augment unlabeled data across multiple models.

4.2 Related Work

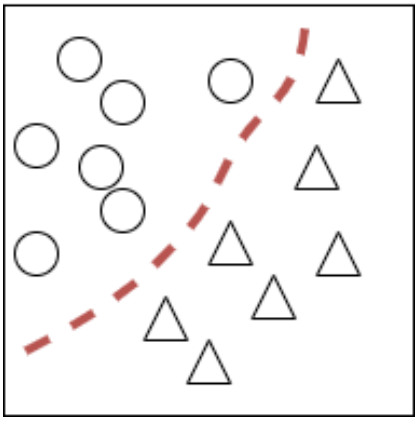
4.2.1 Concept Drift

The goal of ML models is that given a set of input features $\boldsymbol{X} \in \mathcal{R}$, predict a target variable $\boldsymbol{y} \in \mathcal{R}$ for regression tasks (or classes for classification tasks) [21]. The prediction of y is dependent on the prior probability of p($\boldsymbol{y}$) and of $P(\boldsymbol{X}|\boldsymbol{y})$. Using Bayesian Decision Theory, the prediction of $\boldsymbol{y}$ given $\boldsymbol{X}$ can be represented as

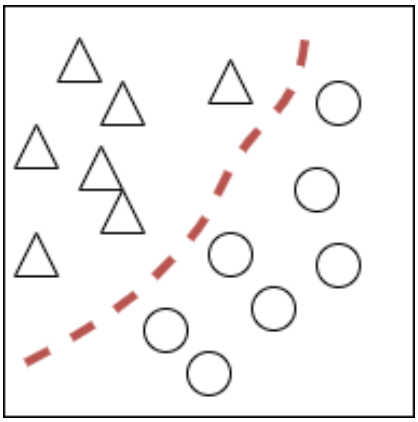
$$p(\boldsymbol{y}|\boldsymbol{X}) = \frac{p(y)p(X|y)}{p(X)}, \tag{4.1}$$

$$p(\boldsymbol{X}) = \sum_{i=1}^{c} p(y)P(\boldsymbol{X}|\boldsymbol{y}), \tag{4.2}$$

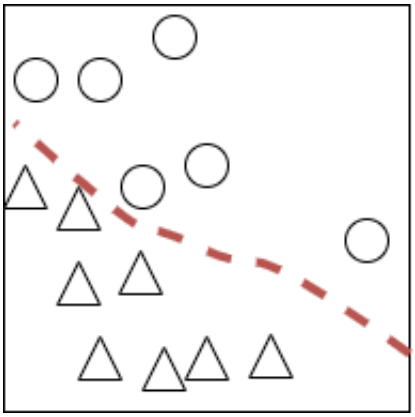
Concept drift occurs when there is a statistically significant difference in Eq.(4.1) as the model consumes streams of online data in its model serving phase, such that $P_t(\boldsymbol{X}, \boldsymbol{y}) \neq P_{t+1}(\boldsymbol{X}, \boldsymbol{y})$ for time t. [31, 32, 40, 24]. Note that for our definition, p($\boldsymbol{X}$) may have changed or the class label has changed as shown in Fig. 4.1a. Other factors can contribute to the type of drift such as the rate of drift (gradual, incremental, sudden) and change in distribution (fixed space or non-fixed space).



(a) Original distribution



(b) Class distribution changed



(c) Input sample distribution changed

Figure 4.1: Examples of concept drift compared to (4.1a) either due to (4.1b) changes in class distribution (i.e. fixed space) or (4.1c) changes in $P(\boldsymbol{X})$, (i.e. non-fixed space). Shapes indicate different classes of data.

4.2.2 Drift detection methods

The most common forms of drift detection for single concept drift can be broken down into performance-based and data distribution-based detectors [24]. Performance-based detectors are supervised approaches that require labeled data to monitor performance metrics, such as accuracy, precision, and recall. Performance-based techniques such as DDM [19] and STEPD [39] use error rate as their performance metric. These methods utilize thresholds to indicate when drift has occurred and which samples should be used to update the model. They act on the premise that the model's error rate will decrease as samples increase so long as the stream is stationary and therefore capturing instances when data is non-stationary as is the case for concept drift.

Data distribution techniques on the other hand can operate in a semi-supervised or unsupervised approach, but their disadvantage is that they are unable to capture all forms of drift and cannot identify for concept drifts due to class distribution changes in unlabeled data as is the case in Fig (4.1b). Rather than monitoring model metrics, these techniques monitor the distribution of the data itself. They rely on clustering and density estimations to detect whether distributions are significantly statistically different [48, 17, 47, 18].

4.2.3 Feature store

There is a distinction between a model's offline training phase and online serving phase. Offline training ensures the availability of a finite training set during the process. The data is often retrieved in batches and possibly used by different models or analyzed by multiple parties of data scientists and engineers. The online serving phase must work along windows of non-finite, real time data that are processed in streams. Many problems can arise between the two phases while the data layers are disconnected. Problems such as silos of redundant data and susceptibility to training-serving skews can occur. Feature stores serve as an

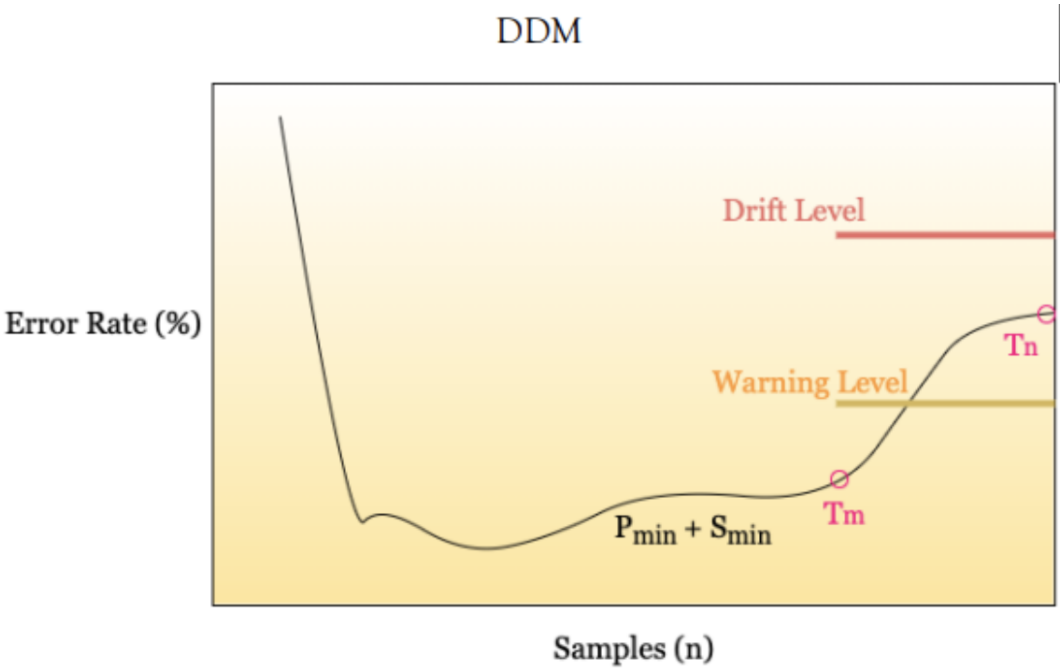


Figure 4.2: Performance based detector such as Drift Detector Method (DDM) measures the error rate which is the sum of p and s. T_m is the error rate at time m and T_n is the error rate at time n. Drift is suspected when the error rate is greater than the warning level and confirmed once it reaches the drift level.

abstraction layer between the model training data from offline store and the model serving data from online store. They can provide the benefits of removing siloed data, promoting feature resuse over rebuild, and decreasing occurrences of training-serving skews using point in time consistency.

Feature stores provide a logically centralized registry physically distributed data by creating a catalog of feature data and their metadata [1]. It is made of a hierarchy of project, feature view and the triplet (feature, entity, and data source). Data source is the raw underlying data that can be located anywhere. The entity is a collection of semantically related features. For example, a ride sharing service can have entities of values customer or driver, while both entities have a shared feature of trips taken. Feature views are made of the triplet and represents a logical grouping or *context*. Finally, updates of features can be done easily using the registry.

4.2.4 SHAP

Shapley values employ a coalition game theory to fairly distribute the contribution of features for a prediction. It takes the average marginal contribution of a feature across all coalitions aka all possible permutations. SHAP uses this notion of Shapley values to get a fair, order agnostic payout of the features. The disadvantage to SHAP is that they are appropriate for linear models, but are costly for models with many features because of the complexity of SHAP is on the order O($2^{feature\_size}$) [50, 4].

4.2.5 Collaborative Filtering

Collaborative Filtering (CF) is often used for recommender systems. Given a set of relationship scores between users to items, it builds an association between any the relationships among user-to-items, user-to-user, or item-to-items. It works on a labeled set and is depen-

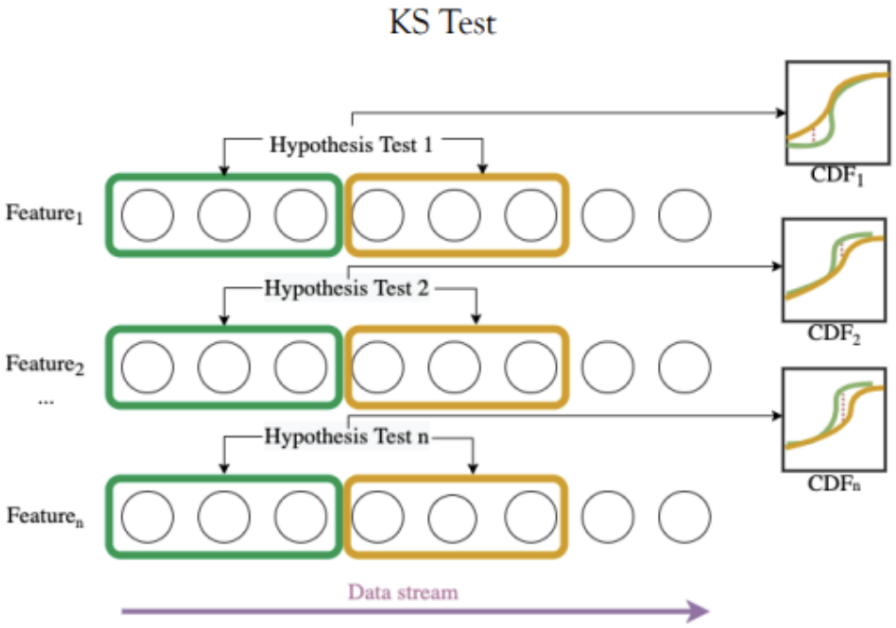


Figure 4.3: Distribution based detectors such as KS tests measures the cumulative distributive function between the reference and current feature distributions. If the CDF are significantly then it indicates covariate drift and either virtual drift or concept drift.

dent on the sparsity of data. CF makes prediction on the empty user to item values based on the similarity scores it generates from performing matrix factorization or support vector machine. Modeling based on the interactions of user-to-user and user-to-items is tricky, since users themselves can change their mind. Hence, these models are susceptible to concept drift and performance-based methods have been used to track whether drift occurs.

Matrix Factorization (MF) which can help us discover latent features underlying the interactions between models and feature views. These latent features give a more compact representation of model drift susceptivity and feature association. MF is useful for sparse data and can enhance the quality of recommendations or in this case associative drift susceptivity. The algorithm works by factorizing the original model-feature matrix into two factor matrices We are reducing the dimensions of our original matrix into "drift susceptivity" dimensions. We cannot interpret what each latent feature k represents (k is the number of latent features per feature views). However, we could imagine that one latent feature may represent models who are tracking time dependent factors, while another latent feature may represent features which are independent.

4.2.6 Local Detectors

Drift Detection Method (DDM)

DDM is a performance-based method for drift detection and tests for the statistical distribution of its model's performance. It is measured by its error rate, p and standard deviation, s (4.3) and detects drift using thresholds, such as the warning level (4.4) and the drift level (4.5). The values, s_{min} and p_{min} are defined in the training phase and are updated if the sample, i at time t achieves (4.6).

$$s_t = \sqrt{p_t(1 - p_t)/i} \tag{4.3}$$

$$p_t + s_t \geq p_{min} + 2s_{min} \tag{4.4}$$

$$p_t + s_t \geq p_{min} + 3s_{min} \tag{4.5}$$

$$p_t + s_t < p_{min} + s_{min} \tag{4.6}$$

For example, if the warning level is triggered at instance t_m and reaches the drift level at t_n, then the model should be retrained on the samples stored between t_m and t_n.

4.3 Proposed Concept Drift Detector with Collaborative Filtering for multiple models

Given a system where streams of data are fed into multiple models that share intersections of features, we aim to develop a method to accomplish the following:

1. Detect drift for each model (local context)

2. Augment the labels based on shared features

3. Make the holistic features of the system more resistant to drift (improve global context)

4.3.1 Detect Drift for each model

Drift Detection on Distributed Datasets (QuaD) is comprised of two classical drift detection methods, DDM and KS test. It combines both methods in order to detect varying types of concept drifts and switch between labeled and unlabeled data streams.

4.3.2 Augment the labels based on shared features

For this scenario, we assume that the models running have shared feature dependencies. Unlike other models, we plan to employ feature views from feature stores to retrieve and update shared features between models with the combination of SHAP and CF. Analagous to the user-item relationship, we translate models as our users and feature views as our items. This will generate connections between models and features within the feature view. To generate the weight of the connection, which is expected from CF, we propose to use SHAP. SHAP will fairly distribute the weight of a feature for a prediction. From here we apply CF onto a recommender system and build predictions of whether certain models should be augmented with other features outside of their original parameters, where the new features may or may not be labeled.

4.3.3 Make the holistic features of the system more resistant to drift

We continue with our CF relationships. From here we not only analyze the model-to-feature relationship, but also the model-to-model relationship and try to discern whether some models are more susceptible to concept drift. If drift is detected on one model, the system can either 1) activate local drift detector(s) on associated models or 2) update models without measuring for drift. There is a trade-off between false alarms and delay in detection between the two actions. Updating will rely on the materialization process of feature stores.

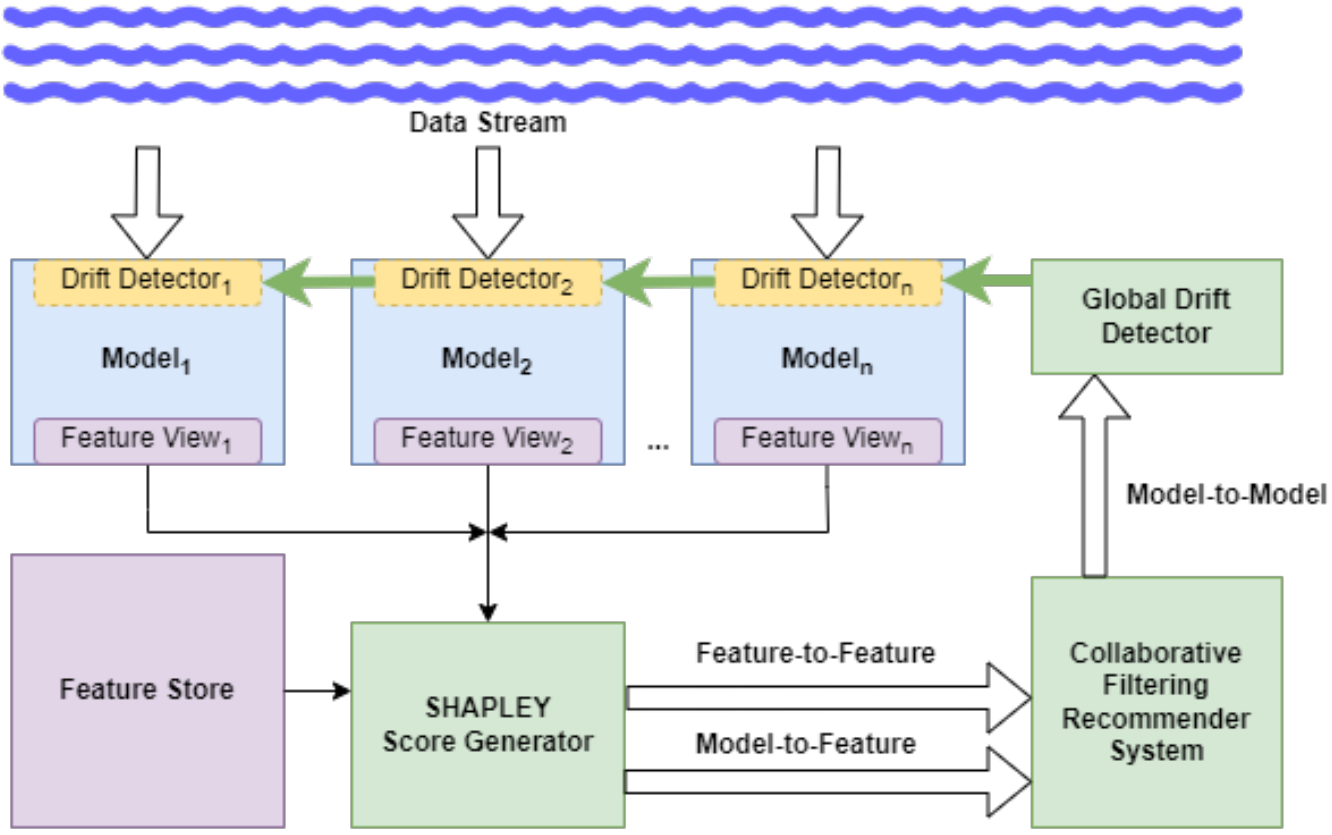


Figure 4.4: Overview of QuaD in a single-stream, multiple model system.

4.4 Evaluation

4.4.1 Metrics

The following are characteristics to evaluate the quality of our concept drift detector.

Probability of true change detection

This requires synthetic data or ground truth and characterizes the capacity to detect drift occurrences.

Probability of false alarms

This characterizes resiliency and is equivalent to the inverse of the time to detection, which is the expected time between false-positive detections. This can be used on real data without drifts and the resulting detections are considered as false alarms.

Delay of detection

This estimates the number of instances required to detect a change after the actual occurence of drift. Average time to detection is used on synthetic data.

4.4.2 Analysis

Both DDM and KS test are reactive methods towards concept drift, meaning that updates are made upon detection of drift. They do not employ forecasting to prevent concept drift from occurring in the first place. QuaD creates an ensemble of these methods and tests

across multiple models that share a significant portion of their training data. The novelty of our method is the consideration of multiple models and features to form a global context and better explain concept drift. Upon detection of drift, QuaD calls for an update on that specific model *and* for models whose intersection with its training sets is significant. Thereby calling for an update before concept drift occurs on these susceptible models.

Additionally, it uses the strength of DDM to detect drifts on non-fixed space and KS to function over unlabeled data.

The constraints, however, is that multiple models must have a significant portion of their training set be shared. Moreover the rate of false positives may be high, since our method calls for an update based on association of similarly trained models. There is an assumption that updates are computationally inexpensive due to the resources (e.g. data parallelism) available in the system. This is especially true for SHAP, which generates the weight for our novel way of using CF. There are methods for model and data parallelism in ML that can be explored.

4.5 Discussion

To the best of our knowledge, QuaD is the first work that examines the collective behavior of concept drift across multiple models and discerns associations between models that may share a susceptibility in a dynamic setting. QuaD uses a combination of performance-based and data distribution-based drift detectors and CF to capture varying types of concept drifts for labeled and unlabeled data streams and is modeled around the data abstraction provided by emerging feature stores.

Developing QuaD will require frameworks, such as River [5] to enable models to run concurrently and to process data as streams. Metrics mentioned in Section 4 should be used to

evaluate QuaD's performance and reliability.

Future work can explore if association of models based on feature store values can benefit from other drift detectors. For example, DDG-DA [30] forecasts drift by generating datasets based on a sampling of historical data instead of the most recent data. The notion of generating synthetic data may relax the constraint of relying on shared datasets between models.

4.6 Summary

Many works focus on optimizing machine learning models during their training phase, but fail to account how these models adapt into their model-serving phase once they are deployed into real world applications. In this phase models must process through streams of data that can evolve over time and distort the relationship between incoming data, causing concept drift. This paper proposes leveraging the advantages of emerging features stores in order to improve concept drift detection on unlabeled, dynamic data streams across multiple models. Firstly, we introduce QuaD, which combines classical drift detectors to make use of labeled and unlabeled data, and create local context (i.e. per live model) and global context (i.e. across multiple models). Secondly, we propose using feature store entities, SHAP values, and Collaborative Filtering (CF) to augment unlabeled data across multiple models. To the best of our knowledge, QuaD is the first work that examines the collective behavior of concept drift across multiple models and discerns associations between models that may share a susceptibility in a dynamic setting. QuaD uses a combination of performance-based and data distribution-based drift detectors and CF to capture varying types of concept drifts for labeled and unlabeled data streams and is modeled around the data abstraction provided by emerging feature stores.

Chapter 5

Our Approach: Collaborative Concept Drift Detection with Fast Correlation Based Filtering

5.1 Introduction

Machine learning models train on data with the expectation that the **concept** or relationship discerned between the predictors and target variable are consistent with post training data (e.g. online data streams). Data streams, however, are unstationary and can result in **concept drifts**, where said relationship no longer holds [52, 25]. Consequently, concept drifts can result in model performance degradation despite the fact that the model itself is unchanged [34]. Additionally, models incur an update cost when they are retrained upon detection of drift. Although there are many types of drift detectors, data distribution or divergence tests can provide more explainability than performance based detectors. [33]. This paper extends a method originally used for feature selection in order to detect concept

drift. We compare our findings to divergence tests and formulate a metric that relates the F1 obtained with the cost of retraining.

5.2 Proposed Collaborative Concept Drift Detection with Fast Correlation Based Filtering

Collaborative Concept Drift Detection (C2D2) applies a window based **Fast Correlated Based Filtering (FCBF)** [54, 38], a multivariate feature selection method that considers both the class relevance and the dependency between each feature pair through the **Symmetrical Uncertainty (SU)** eq.(6.6) computed from entropy. SU calculates the mutual dependencies of random variables X and Y such that values closer to 0 indicate independence while values closer to 1 indicate dependence, where knowledge of one can predict the outcome of its pair. IG is the information gain of X given Y and $H(X), H(Y)$ are entropies of X, Y.

Rather than remove the redundant features directly, the calculated SU are fed as a matrix, A for Singular Value Decomposition (SVD) eq.(5.2) where V^T relates to batches and U corresponds to the features. Taking only the top four V^T and U components noted by S, we calculate sum of the stepwise difference of V^T for every batch. The resulting argmax and values within 1 standard deviation of the max are signalled as batches with drift.

$$SU(X,Y) = \frac{2 \times IG(X|Y)}{H(X) + H(Y)} \tag{5.1}$$

$$A = USV^T \tag{5.2}$$

Each row i dictates a feature space to predict a respective y_j. Solid or dashed lines within

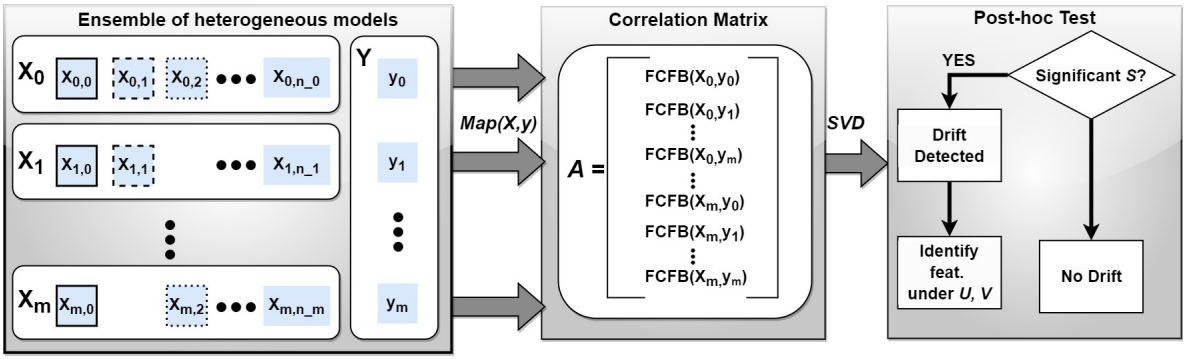


Figure 5.1: C2D2 Architecture

the feature space indicate shared predictors while gaps indicate the absence of features. The feature and target spaces are mapped $m : n$ and their correlations are calculated via FCBF. SVD is used to break down its invariant representations and post-hoc tests are applied to test significance and indicate the occurrence of drift.

5.2.1 Reasoning behind detecting invariants

To fit a model is to take a function f onto a feature vector, such that it provides a systematic estimate of the target variable, y ([28]). ϵ is independent of x and represents random error, a combination of **irreducible error** and **reducible** error (5.3).

$$y = f(x) + \epsilon \tag{5.3}$$

We speculate that ϵ of $f(x)$ can serve as a predictor for another model, $f'(x)$. In other words, ϵ is independent of x, but is related to x' or y. The underlying assumption is that as a concept evolves, predictors that turn into confounding variables (i.e. predictors that affect both the target and other predictor(s)) are related to the source of drift. Additionally, there is the assumption that such confounding variables will be invariant across an ensemble of classifiers with heterogeneous training and prediction subspaces and can be used to indicate

the occurrence of drift.

$$\sum_{y} p(x, y) = \sum_{y'} p(x, y') \tag{5.4}$$

The exploration of our subspaces has basis on the **transfer learning** and **ensembling techniques**. Under transfer learning, models share the same predictor or feature space, but varying targets. This method engenders the context of x on or a set consisting of y. For homogeneous ensembles, models are trained on different subspaces of x to predict the same y. This provides the context of , which consists of set of x on a specific target y.

5.3 Experimental Setup

C2D2 was tested on 4 artificial data sets generated by the MOA framework (Massive Online Analysis) ([9, 12]). Concept drift was modeled on MOA by joining data streams as a weighted combination of distributions whose probability of an instance stemming from the new concept is defined by a sigmoid function. Each dataset contained 10K instances and were injected with concept drifts of widths from 0.5K to 4K instances. The midpoints of drift ranged from 1.5k to 7.5K. 10 tests were generated from each dataset by modifying the instantiation of streams via a random seed.

C2D2 was evaluated against 6 divergence tests: Cramer Von Mises (CMV), Energy Distance test (ED), Kolmogorov-Smirnov test (KS), Mann-Whitney U-test (MW), T test, and Wasserstein Distance (WD). The 6 tests were implemented in a fixed sliding window fashion, which split the data into 10 batches and compared the i^{th} batch with the i^{th-1} batch as its respective current and reference windows. Batches that were deemed significantly different

were signalled for retraining. Hoeffding tree (HFT) was used as the base classifier and was prompted to retrain on the most recent batch according to the signals generated by the detectors. In addition to these test was an ablation test, where the HFT made predictions without any retraining.

5.3.1 Performance Gain to Update Cost Ratio

An overly cautious and ineffective method would arbitrarily signal for an update at every batch. To take in consideration the computational and temporal debt of retraining with respect to the performance gained, we introduce the **Performance Gain to Update Cost Ratio (PGUCR)** 5.5. Ratios of 0 are ineffective detectors and 1 are effective detectors. $F1_{new}$ is the $F1$ score of the base classifier that was retrained according to the signals, while $F1_{ablation}$ is the score without any retraining. N_{update} is the number of batches signalled (i.e. max is 9 as the first batch is used for training). $Cost_{update}$ which can be seen as a penalty for updating is an adjustable parameter whose value is related to the importance of improving the $F1$ score. Our experiments set $Cost_{update}$ to 0.1.

$$PGUCR = \frac{1}{2}\left(1 + \frac{F1_{\text{new}} - F1_{\text{ablation}}}{F1_{\text{ablation}}}\right) / \left(1 + N_{\text{update}} \times \ Cost_{\text{update}}\right) \tag{5.5}$$

Table 5.1: Parameters for C2D2

Datasets: Agrawal, LED, RandTree_55, RandTree_105, SEA
Drift Widths: 0.5k - 4k instances
Midpoint locations: 1.5k - 7.5k instances
Divergence Tests: Cramer Von Mises (CMV), Energy Distance test (ED), Kolmogorov-Smirnov test (KS), Mann-Whitney U-test (MW), T test, Wassertein Distance (WD)
Batch Size: 10
Base Classifier: Hoeffding Tree

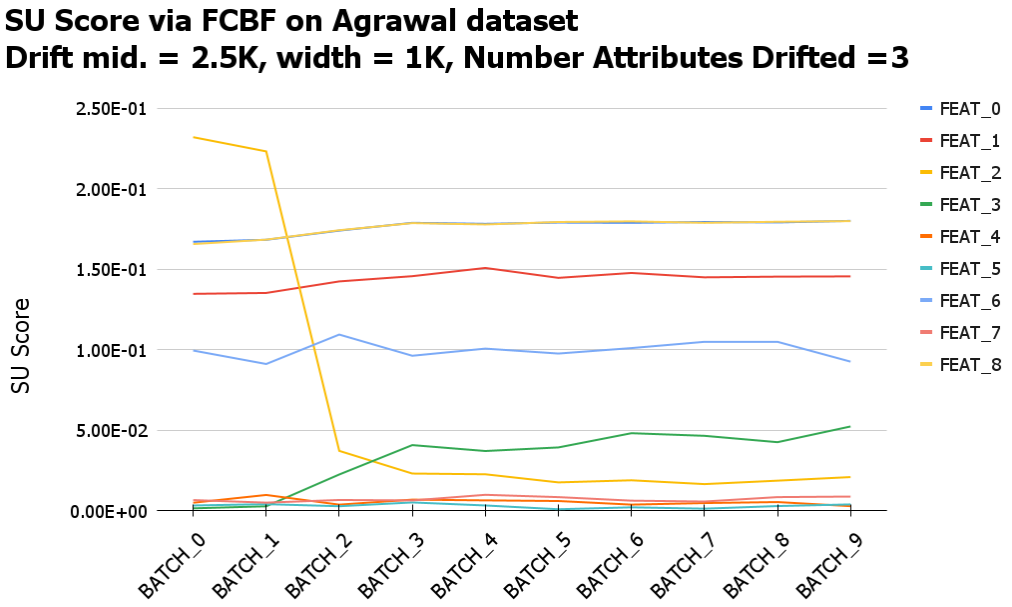


Figure 5.2: SU scores per feature per batch on one of the Agrawal dataset. Drift occurs from instances 2K to 3K (i.e. batch 2). Feat 2 decreases while Feat 3 increases at a crossover point between batch 2 and 3.

5.4 Discussion

ANOVA indicated that there is a difference in mean values of PGUCR amongst the tests. Post-hoc Tukey HSD test confirmed that C2D2 provided significant improvement. Much of C2D2's improvement is based on the fact that it signalled for updates conservatively in

Table 5.2: Average number of signals for updates on synthetic datasets. Ablation excluded as the count will always be zero. Counts can range from 0 to 9.

Test	Agrawal	LED	RandTree_55	RandTree_105	Sea
CVM	4.3 ± 0.9	8.1 ± 0.9	9.0 ± 0.9	9.0 ± 0.0	2.2 ± 0.8
ED	9.0 ± 0.0	**2.0 ± 0.7**	2.4 ± 0.5	2.0 ± 1.1	4.3 ± 1.1
C2D2	3.1 ± 1.4	3.6 ± 1.3	**2.6 ± 1.1**	**1.8 ± 0.6**	1.7 ± 0.5
KS	3.5 ± 1.8	3.6 ± 1.7	9.0 ± 0.0	9.0 ± 0.0	2.0 ± 1.0
MW	5.4 ± 0.5	7.7 ± 2.1	9.0 ± 0.0	9.0 ± 0.0	2.2 ± 1.0
T	6.0 ± 0.7	8.3 ± 0.7	9.0 ± 0.0	9.0 ± 0.0	2.2 ± 1.0
WD	**3.0 ± 0.7**	4.7 ± 0.9	8.5 ± 0.7	8.0 ± 0.8	**0.2 ± 0.4**

Table 5.3: Average F1 score on the datasets with 10 samples. The F1 scores of the tests, ablation (ABL), and number of signals are used to calculate PGUCR.

Test	Agrawal	LED	RandTree_55	RandTree_105	Sea
CVM	0.76 ± 0.08	0.66 ± 0.04	0.76 ± 0.01	0.67 ± 0.03	0.84 ± 0.02
ED	0.77 ± 0.05	0.64 ± 0.05	0.65 ± 0.03	0.63 ± 0.05	0.84 ± 0.01
C2D2	0.74 ± 0.10	0.65 ± 0.05	0.64 ± 0.04	0.63 ± 0.03	0.84 ± 0.01
KS	0.68 ± 0.11	0.65 ± 0.04	0.76 ± 0.01	0.67 ± 0.03	0.84 ± 0.02
MW	0.78 ± 0.08	0.66 ± 0.04	0.76 ± 0.01	0.67 ± 0.03	0.83 ± 0.02
T	0.78 ± 0.08	0.66 ± 0.04	0.76 ± 0.01	0.67 ± 0.03	0.83 ± 0.02
WD	0.71 ± 0.09	0.66 ± 0.04	0.76 ± 0.01	0.66 ± 0.04	0.83 ± 0.02
ABL	0.66 ± 0.12	0.58 ± 0.10	0.58 ± 0.07	0.61 ± 0.05	0.83 ± 0.02

comparison to the other tests[5.3], thereby decreasing its false positive rate. In contrast to the other tests, which monitored whether a feature's distribution had changed in comparison to itself, C2D2 was able to hint at the relationship of how the features have drifted collectively as shown in Fig. 5.2, Fig.5.3. Future work should identify whether the collective nature of C2D2 can be applied to bringing explainability towards formerly independent models with overlapping feature spaces.

5.5 Summary

C2D2 combines FCBF and SVD to detect concept drifts in 5 synthetic datasets. We compare our results against 6 diveregence tests and introduce PGUCR. Post-hoc Tukey HSD test confirmed that C2D2 outperformed the other tests in terms of PGUCR. Much of C2D2's improvement is based on its conservative signals for updates.

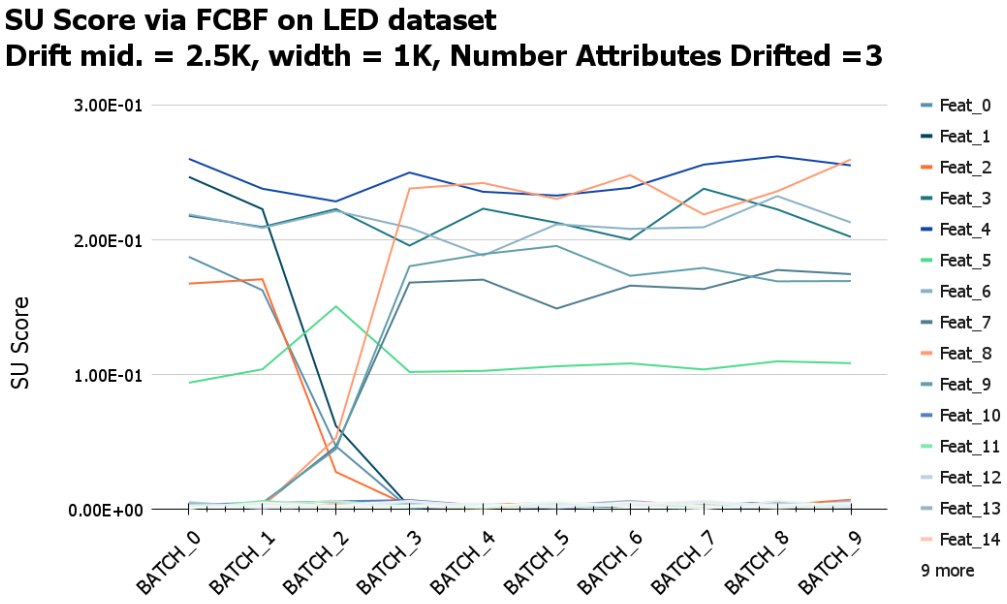


Figure 5.3: SU scores per feature per batch on one of the LED dataset. Drift occurs from instances 2K to 3K (i.e. batch 2). Feat 0, 1, and 2 decrease while Feat 7, 8 and 9 increase at a crossover point at batch 2.

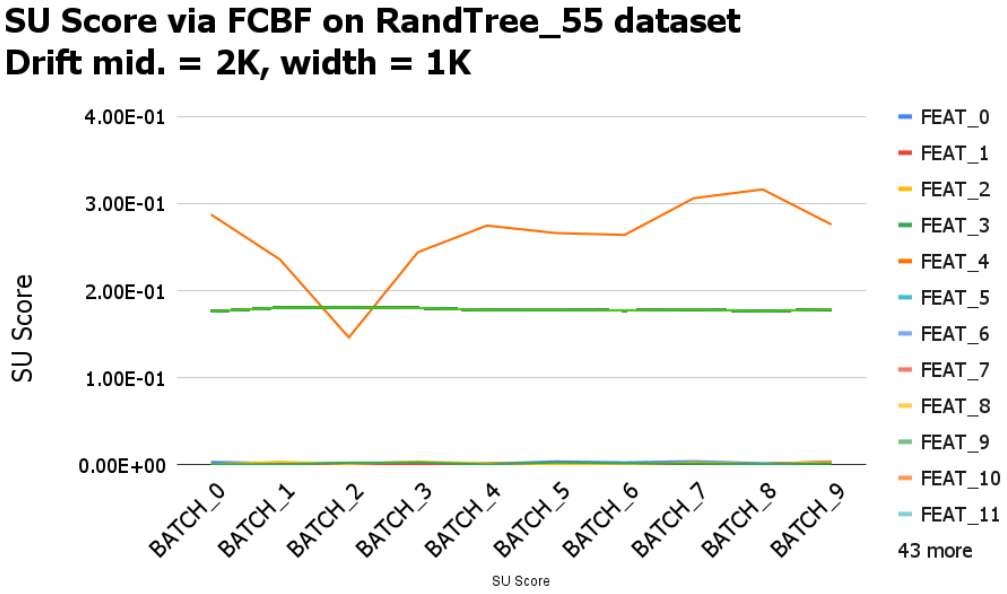


Figure 5.4: SU scores per feature per batch on one of the RandTree_55 dataset. Drift occurs from instances 1.5 K to 2.5K (i.e. batches 1 and 2)

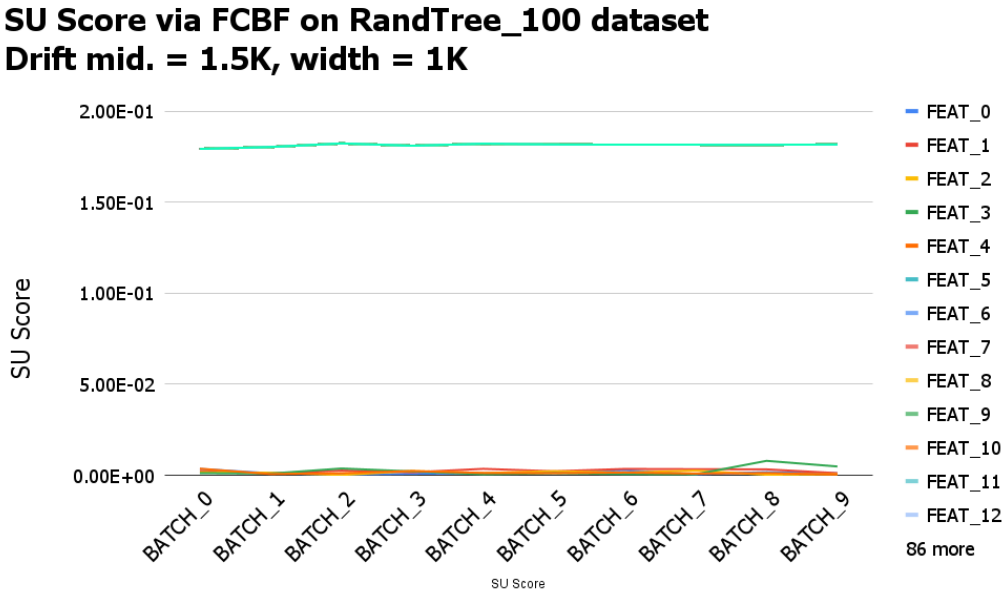


Figure 5.5: SU scores per feature per batch on one of the RandTree_100 dataset. Drift injected from instances 1 K to 2K (i.e. batch 1

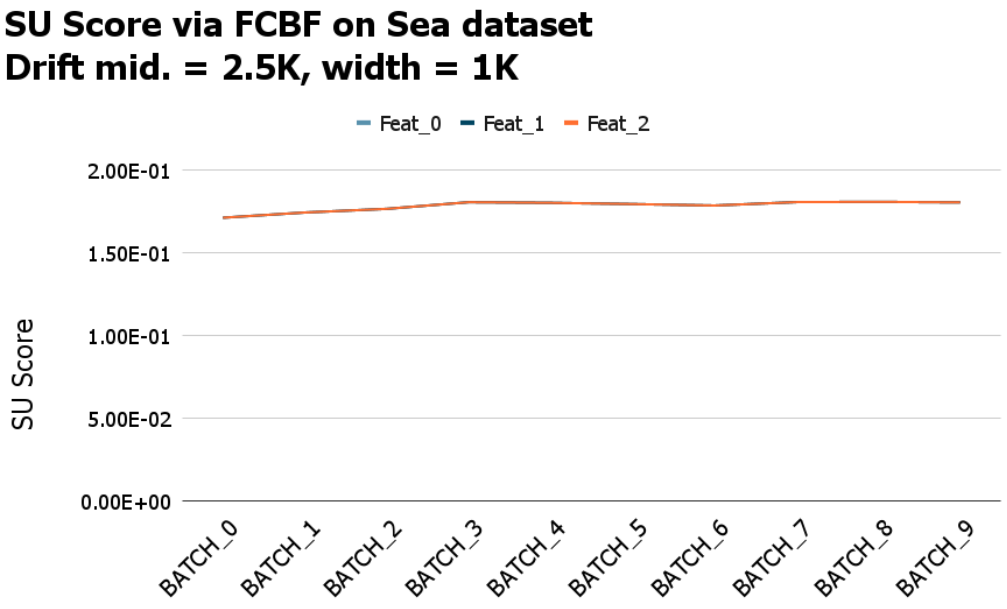


Figure 5.6: SU scores per feature per batch on one of the Sea dataset. Drift injected from instances 2 K to 3K (i.e. batches 2)

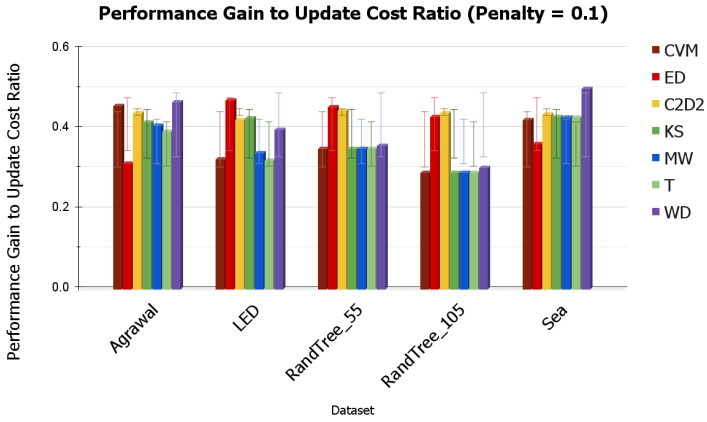


Figure 5.7: Mean PGUCR values with penalty of 0.1. Each dataset contained the following number of features: Agrawal (9), LED (24), RandTree_55 (55), RandTree_105 (105), Sea (4).

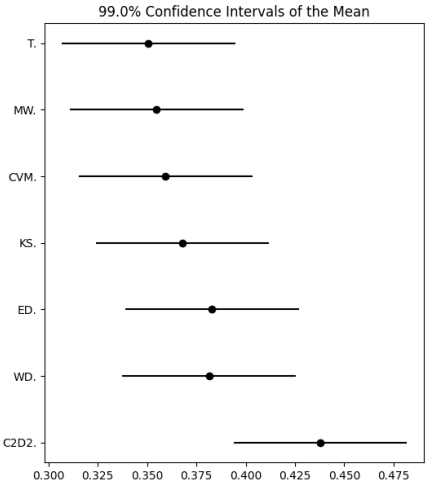


Figure 5.8: Post-hoc Tukey HSD test at 99% confidence interval indicates that C2D2 is significantly better than T, MW, CVM, KS, ED, and WD. C2D2 has a mean of 0.43.

Chapter 6

Our Approach: Adaptive Aggregated Drift Detector

6.1 Introduction

This chapter introduces **Adaptive Aggregated Drift Detector (A2D2)**. It takes in a suite of detectors consisting of a mixture of performance based detectors and data distribution based (e.g. divergence) detectors. Its goal is to adaptively select the detector that optimizes N_{update} and $Gain_{\text{perf}}$ to achieve minimal costs. It applies exploitation and exploration strategies through the use of its Adaptive and Aggregative Phases.

Section 6.2 describes the utilization of a metric accounting for costs in terms of performance. Section 6.3 describes A2D2 and its metrics for evaluation under Section 6.4. Potential contributions under Section 7.

6.2 Preliminary Work

We proposed **Performance Gained Update Cost Ratio (PGUCR)** Eq.(6.1) in order to relate a model's gain in performance with its cost of retraining updates in response to the detection of drift. PGUCR is normalized with values from 0 (ineffective) to 1 (effective). $F1_{new}$ represents the F1 score of a base classifier equipped with a detector. $F1_{ablation}$ is the score without any detectors and thereby without any updates to the model. N_{update} is the number of times a batch was triggered to retrain within a set number of batches. $Cost_{\text{update}}$ is an adjustable parameter representing the importance of updates in comparison to performance.

$$\text{PGUCR} = \frac{1}{2}\left(1 + \frac{F1_{\text{new}} - F1_{\text{ablation}}}{F1_{\text{ablation}}}\right) \div (1 + N_{\text{update}} \times \ Cost_{\text{update}}) \quad (6.1)$$

We manipulate the PGUCR to be in terms of $Cost_{update}$ and describe its upper and lower bounds. For clarity performance gain is:

$$Gain_{\text{perf}} = \frac{F1_{\text{new}} - F1_{\text{ablation}}}{F1_{\text{ablation}}} \quad (6.2)$$

The lower bound of $Cost_{\text{update}}$ when PGUCR equals 1 simplifies to:

$$Cost_{\text{update}} = \frac{1}{2N_{\text{update}}}(Gain_{\text{perf}} - \frac{1}{2}) \quad (6.3)$$

The upper bound when PGUCR equals 0 signifies that regardless of any updates, the base classifier $Gain_{\text{perf}}$ is -1.

Our work applies the lower bound Eq.(6.3) which attempts to achieve max efficiency by

minimizing costs improvements and PGUCR as metrics.

6.3 Methodology

A2D2 architecture has Adaptive and Aggregative Phases as shown in Fig(6.1). The Adaptive Phase applies exploitation and exploration strategies in order to select the most cost effective detector out of the suite of detectors. Simultaneously it creates an ensemble of detectors composed of the selected detector and its most complimentary detector (i.e. detector with the least similarity in terms of batches triggered). The Aggregative Phase updates its Aggregated Embeddings (AgE) via the SU matrix and the updated rankings from the Cost Based Ordinality (CBO). To generate complimentary ensembles, batches triggered per detector are inputted to the Collaborative Filtering Recommender System (CFRS),which scores detectors of their similarities. If the newly created ensemble detector is more cost effective than the selected detector, then the ensemble grouping is added to the suite of detectors.

6.3.1 Adaptive Phase

The Adaptive Phase dynamically selects the detector that has the best fit of achieving the highest PGUCR on top of generating new ensembles of detectors to add to the suite. It consists of the Terminal FCBF, Prequential Training and Testing, and Select & Explore units.

Terminal Fast Correlated Based Filtering (FCBF)

Fast Correlated Based Filtering (FCBF) [54, 38] is a multivariate feature selection method that takes into account the dependencies between features and the class relevance. It uses

information gain via entropy Eq.(6.4) to generate values under **Symmetrical Uncertainty (SU)** [26]. SU Eq.(6.6) calculates the dependencies between random variables X and Y such that it measures the effect of having knowledge on one has on the information learned from the other. SU is normalized between 0 (i.e. complete independence) to 1 (i.e. mutual dependence). $IG(X|Y)$ Eq.(6.5) denotes the information gain of X given Y.

$$H(X) = -\sum_{x} P(x_i) \log_2(P(x_i)) \tag{6.4}$$

$$\mathrm{IG}(X|Y) = H(X) - H(X|Y) \tag{6.5}$$

$$SU(X,Y) = \frac{2 \times IG(X|Y)}{H(X) + H(Y)} \tag{6.6}$$

FCBF sorts features from highest to lowest SU ordering them from most to least relevance to the class. Yu & Lie [54] iteratively removed redundant features by using the most predominant feature to heuristically compare and filter against lower valued features. Our method does not remove redundancies (hence the name Terminal FCBF) and keeps the SU values per features as a matrix. The generated SU matrix is then used as a blueprint for the data set in hand.

Prequential Training and Testing

The role of the base classifier is to predict the class, y based on the incoming features, X. Data is processed prequentially as windows, W with instances first used for testing followed by training. Every W is split into 10 batches, b. A selected detector processes through W

and indicates if it suspects drift under any of b, serving as a list of triggers that the base classifier must retrain under. Each W is processed at least twice under the Adaptive Phase. Once for predicting under the triggered retrainings and another under ablation without any triggers. The F1 scores and triggers are sent to the Aggregative Phase.

Select & Explore units

Select takes the SU matrix and the Aggregated Embeddings (AgE) as inputs in order to predict which existing detector is the best for current W. The Explore unit takes in the selected detector, DD and references the Collaborative Filtering Recommender System (CFRS) in order to identify which detector has the least similarity with (if any). If there exists a pair, the union of triggers between DD and the complementary detector, DD' are used. If an ensemble is created, then the classifier must run for a third time and subsequently compares if $\text{PGUCR}_{DD} \leq \text{PGUCR}_{DD'}$ noting that DD' is worth adding to the suite.

6.3.2 Aggregative Phase

The Aggregative Phase develops a collective knowledge of each detector with respect to the W processed. If the Adaptive Phase is considered as online testing, then the Aggregative Phase would be considered the offline training. Hence W, which was the current data from the Adaptive Phase is viewed as the reference data under the Aggregative Phase. Under this phase, all the detectors take turns detecting drift and measuring their performance under the Detector Test Suite. The Detector Test Suite works similarly as the Adaptive Phase, where classifier with detector is compared with the ablation test. The Detector Test Suite outputs the F1 score and triggers to the Cost Based Ordinality (CBO) and only the triggers to CFRS.

Collaborative Filtering Recommender System

CFRS takes the key value pairs of detectors and triggers and represents them into a matrix, where rows indicate detectors and columns indicate whether a drift was detected at b.It applies the Jaccard similarity coefficient [27] to calculate the pairwise similarities amongst detectors and outputs their similarity scores. Unorthodoxically, CFRS is used for finding the least similar detector for the DD under the Adaptive Phase.

Cost Based Ordinality

The CBO ranks the detectors according to Eq.(6.3). Heuristically, the detectors are ranked from largest to smallest cost, but an alternative method not implemented is to have the detectors fall under rankings based on where their $Cost_{\text{update}}$ falls under the intervals of $1/N_{\text{Detectors}}$, where $N_{\text{Detectors}}$ is the current number of detectors in the suite. The key value pairs of detector to ranking are processed to AgE.

Aggregated Embeddings

AgE saves the SU matrix from the Adaptive Phase and connects it with the rankings after the Detector Test Suite and CBO are complete. AgE appends the table of SU with the detector-rankings, where SU can be interpreted as the features that predict the classification defined by the detector-rankings. In other words, AgE is used to predict which detectors would have the smallest $Cost_{\text{update}}$ under a specific SU.

6.4 Evaluation

This work will use the MOA framework [9, 12] to evaluate A2D2 on 3 artificial datasets and 2 real-world datasets that were injected with concept drift. Our method will include 6 divergence tests capable of detecting changes in distribution specifically, Cramer Von Mises test (CMV), Energy Distance test (EDT), Kolmogorov-Smirnov test (KS), Mann-Whitney U-rank test (MWT), T test, and Wasserstein Distance test (WD). It will also include 6 performance based detectors DDM, EDDM, KSWIN, PH, ADWIN, HDDM_A, and HDDM_W as part of the Detector Test Suite.

6.4.1 Artifical Dataset Configuration

MOA can inject concept drift by connecting data streams as a weighted combination of distributions whose likelihood of an instance originating from the new concept is defined by a sigmoid function. Each dataset will contain 10K instances with widths ranging from 0.5K to 4K instances. The drift's midpoints will fall in between 1.5 and 7.5 kilometers. By altering the instantiation of streams using a random seed, 10 tests will be created from each dataset.

6.4.2 Artificial Datasets

Agrawal [6] describes 6 numerical and 3 categorical features mapped to 10 different predefined loan functions

LED [11, 42, 13] comprises of 24 binary attributes, but only 7 are relevant for predicting the next digit on a seven-segment LED display (i.e. 10 classes).

Sea [11, 42, 49] uses 3 attributes, but 1 is irrelevant. All attributes have values between

0 and 10, and comparing the sum of the relevant attributes with a threshold parameter determines which of the 4 classes it is mapped to.

6.4.3 Real World Datasets

The following datasets are nomalized by MOA, so that the numerical values are between 0 and 1.

Electricity [22] contains 8 attributes to predict whether the Australian New South Wales Electricity Market from May 1996 to December 1998 rises or falls (i.e. 2 classes). The dataset contains 45,312 instances.

Poker is a modified version of [14] without duplicates and is sorted by rank and suit. Each instance is a hand of 5 cards drawn from a 52 card deck. It is made up of 10 attributes (rank and suit of cards in hand) to predict 10 poker hands or classes. The dataset has 1,000,000 instances.

The potential advantages of A2D2 will be its ability to 1) embed W as training data to predict rankings of detectors and adaptively select the one with least cost and 2) develop a collective understanding of detectors that continues to grow. The notable contribution may be 2) as it enables an ecosytem to not only adaptively combat drift, but to also expand the information learned across a suite of detectors.

6.5 Summary

There needs to be an adaptive approach that combines both performance and distribution based concept drift detectors in order to harness the benefits of unlabeled data and the ability to detect varying types of drifts. This paper proposes Adaptive Aggregated Drift Detector

(A2D2), which consists of a suite of performance and data distribution based detectors that can adaptively select detectors based on rankings of least cost. The notable contribution is that it enables an ecosytem to not only adaptively combat drift, but to also expand the information learned across a suite of detectors.

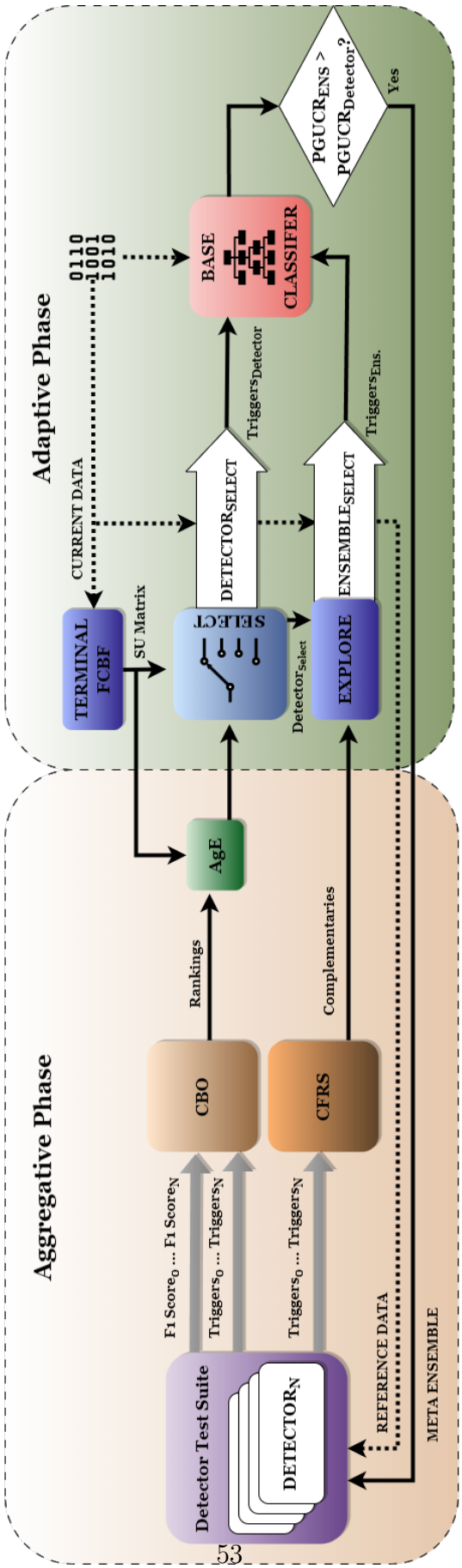


Figure 6.1: A2D2 with its adaptive (exploit) and aggregative (explore) phases.

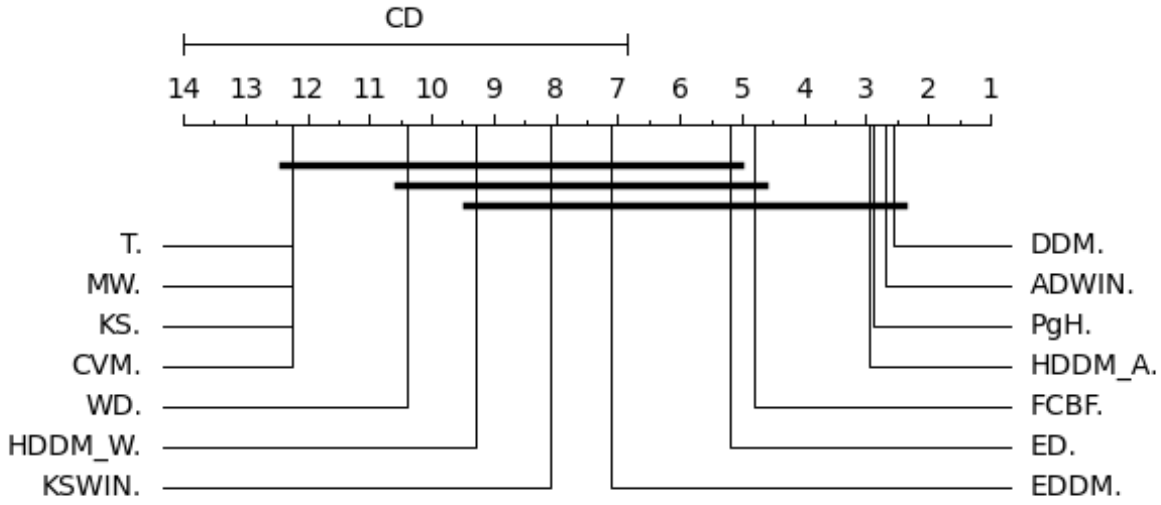


Figure 6.2: Post-hoc Nemenyi test at confidence level of 99% of the PGUCR from RandTree100 dataset. Critical distance is 7.144. With respect to FCBF there is a significant difference between it and T, MW, KS, and CVM.

Chapter 7

Conclusion

7.1 Summary

Concept drift can affect any model in its serving phase, where it is subjected to the unstationary behaviors of data streams. The outcome is an unpredicted degradation in performance despite performing optimally before the occurrence of drift. Existing detectors detect per model per stream but fail to consider shared resources between models in practice, which are utilized under a shared data pipeline. This dissertation emphasizes viewing models as integrated systems. We attempt to improve our understanding of factors related to concept drift by detecting drift across multiple models and/or streams, thereby transforming the problem of drift detection for formerly independent models as a collaborative effort in detecting drift as a whole.

Chapter 4 formed model to model associations based on feature to model associations in order to save costs of detecting on every model. The premise is that like models will share susceptibilities. This also introduced a means to combine many types of detectors and incorporate flexbility in the types of data used. **Chapter 5** introduced the Performance

Gain Update Cost ratio which related the gain in performance with the cost of retraining. This acted as the orthogonal of QuaD. Rather than focus on similarities it operated under the assumption that drift can be pervasive and be the result of latent features. **Chapter 6** designed a way to expand a suite of detectors while predicting which detector to select to garner the most cost-effective approach.

7.2 Future Directions

The strength of incorporating collaborative filtering for concept drift detection is that the weaknesses of collaborative filtering are known. In other words, to improve our implementation of QuaD, one can include ways to improve known problems of collaborative filtering such as cold start and generating implicit feedback. The cold start problem is where many of the scores are unknown beforehand. Another way to extend QuaD is to incorporate implicit feedback on top of the explicit feedback. The explicit feedback are the scores given from the features. Implicit feedback may be calling features from same feature view from feature store (feature feedback) or detecting drift at the same moment as another model - tallied by global drift detector.

The idea of viewing independent models as intertwined components of a systems can benefit beyond concept drift detection. Without a doubt systems will continue to grow and expand. Not only will we want to execute multiple models at once, but memory management and feature engineering involved will play a bigger factor. As long we pull against a data-centric view and push towards the current model-centric mentality, the greater importance it will be to design ways to turn formerly independent models into passively collaborative entities.

Bibliography

[1] Feast. `https://github.com/gojek/feast`. Accessed: 2021-09-30.

[2] Hopsworks: data-intensive ai with a feature store. `https://github.com/logicalclocks/hopsworks`. Accessed: 2021-09-30.

[3] Iguazio. `https://www.iguazio.com/feature-store/`. Accessed: 2021-09-30.

[4] Interpretable machine learning. `https://christophm.github.io/interpretable-ml-book/shap.html`. Accessed: 2021-09-30.

[5] Online-ml/river: nline machine learning in python,". `https://github.com/online-ml/river`. Accessed: 2022-01-24.

[6] R. Agrawal, T. Imielinski, and A. N. Swami. Database mining: A performance perspective. *IEEE Trans. Knowl. Data Eng.*, 5(6):914–925, 1993.

[7] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, and R. Morales-Bueno. Early drift detection method. In *Fourth international workshop on knowledge discovery from data streams*, volume 6, pages 77–86. Citeseer, 2006.

[8] A. Bifet and R. Gavalda. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 443–448. SIAM, 2007.

[9] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. MOA: massive online analysis. *J. Mach. Learn. Res.*, 11:1601–1604, 2010.

[10] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavalda. New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 139–148, 2009.

[11] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà. New ensemble methods for evolving data streams. *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009.

[12] A. Bifet, J. Read, B. Pfahringer, G. Holmes, and I. Žliobaitė. Cd-moa: Change detection framework for massive online analysis. In *Advances in Intelligent Data Analysis XII: 12th International Symposium, IDA 2013, London, UK, October 17-19, 2013. Proceedings 12*, pages 92–103. Springer, 2013.

[13] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[14] R. Cattral, F. Oppacher, and D. Deugo. Evolutionary data mining with automatic rule generalization. 2001.

[15] B. Dalessandro, D. Chen, T. Raeder, C. Perlich, M. Han Williams, and F. Provost. Scalable hands-free transfer learning for online advertising. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1573–1582, 2014.

[16] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. *SIAM journal on computing*, 31(6):1794–1813, 2002.

[17] D. M. dos Reis, P. A. Flach, S. Matwin, and G. E. A. P. A. Batista. Fast unsupervised online drift detection using incremental kolmogorov-smirnov test. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1545–1554. ACM, 2016.

[18] A. Dries and U. Rückert. Adaptive concept drift detection. *Stat. Anal. Data Min.*, 2(5-6):311–327, 2009.

[19] J. Gama, P. Medas, G. Castillo, and P. P. Rodrigues. Learning with drift detection. In *Advances in Artificial Intelligence - SBIA 2004, 17th Brazilian Symposium on Artificial Intelligence, São Luis, Maranhão, Brazil, September 29 - October 1, 2004, Proceedings*, volume 3171 of *Lecture Notes in Computer Science*, pages 286–295. Springer, 2004.

[20] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37, 2014.

[21] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37, 2014.

[22] M. B. Harries. Splice-2 comparative evaluation: Electricity pricing. 1999.

[23] H. Hu, M. M. Kantardzic, and T. S. Sethi. No free lunch theorem for concept drift detection in streaming data classification: A review. *WIREs Data Mining Knowl. Discov.*, 10(2), 2020.

[24] H. Hu, M. M. Kantardzic, and T. S. Sethi. No free lunch theorem for concept drift detection in streaming data classification: A review. *WIREs Data Mining Knowl. Discov.*, 10(2), 2020.

[25] C. Huyen. Data distribution shifts and monitoring. In *Designing machine learning systems*. O'Reilly, 2022.

[26] A. Iserles. Numerical recipes in c—the art of scientific computing, by w. h. press, b. p. flannery, s. a. teukolsky and w. t. vetterling. pp 735. £27·50. 1988. isbn 0-521-35465-x (cambridge university press). *The Mathematical Gazette*, 73(464):167–170, 1989.

[27] G. Ivchenko and S. Honov. On the jaccard similarity test. *Journal of Mathematical Sciences*, 88:789–794, 1998.

[28] G. James, D. Witten, T. Hastie, and R. Tibshirani. An introduction to statistical learning: With applications in r. Springer, 2021.

[29] F. Laio. Cramer–von mises and anderson-darling goodness of fit tests for extreme value distributions with unknown parameters. *Water Resources Research*, 40(9), 2004.

[30] W. Li, X. Yang, W. Liu, Y. Xia, and J. Bian. DDG-DA: data distribution generation for predictable concept drift adaptation. *CoRR*, abs/2201.04038, 2022.

[31] A. Liu, J. Lu, F. Liu, and G. Zhang. Accumulating regional density dissimilarity for concept drift detection in data streams. *Pattern Recognit.*, 76:256–272, 2018.

[32] S. Liu, M. Yamada, N. Collier, and M. Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 43:72–83, 2013.

[33] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang. Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering*, 31(12):2346–2363, 2018.

[34] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang. Learning under concept drift: A review. *IEEE Trans. Knowl. Data Eng.*, 31(12):2346–2363, 2019.

[35] E. Lughofer, E. Weigl, W. Heidl, C. Eitzinger, and T. Radauer. Recognizing input space and target concept drifts in data streams with scarcely labeled and unlabelled instances. *Inf. Sci.*, 355-356:127–151, 2016.

[36] M. M. Masud, Q. Chen, L. Khan, C. C. Aggarwal, J. Gao, J. Han, A. Srivastava, and N. C. Oza. Classification and adaptive novel class detection of feature-evolving data streams. *IEEE Trans. Knowl. Data Eng.*, 25(7):1484–1497, 2013.

[37] M. M. Masud, Q. Chen, L. Khan, C. C. Aggarwal, J. Gao, J. Han, and B. Thuraisingham. Addressing concept-evolution in concept-drifting data streams. In G. I. Webb, B. Liu, C. Zhang, D. Gunopulos, and X. Wu, editors, *ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010*, pages 929–934. IEEE Computer Society, 2010.

[38] H. Nguyen, Y. Woon, W. K. Ng, and L. Wan. Heterogeneous ensemble for feature drifts in data streams. In *Advances in Knowledge Discovery and Data Mining - 16th Pacific-Asia Conference, PAKDD 2012, Kuala Lumpur, Malaysia, May 29 - June 1, 2012, Proceedings, Part II*, volume 7302 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2012.

[39] K. Nishida and K. Yamauchi. Detecting concept drift using statistical testing. In V. Corruble, M. Takeda, and E. Suzuki, editors, *Discovery Science, 10th International Conference, DS 2007, Sendai, Japan, October 1-4, 2007, Proceedings*, volume 4755 of *Lecture Notes in Computer Science*, pages 264–269. Springer, 2007.

[40] F. A. Pinage, E. M. dos Santos, and J. M. P. da Gama. Classification systems in dynamic environments: an overview. *WIREs Data Mining Knowl. Discov.*, 6(5):156–166, 2016.

[41] M. L. Rizzo and G. J. Székely. Energy distance. *wiley interdisciplinary reviews: Computational statistics*, 8(1):27–38, 2016.

[42] J. C. Schlimmer and R. H. Granger. Incremental learning from noisy data. *Mach. Learn.*, 1(3):317–354, 1986.

[43] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison. Hidden technical debt in machine learning systems. *Advances in neural information processing systems*, 28, 2015.

[44] D. Sculley, M. E. Otey, M. Pohl, B. Spitznagel, J. Hainsworth, and Y. Zhou. Detecting adversarial advertisements in the wild. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 274–282, 2011.

[45] J. Shen, Y. Qu, W. Zhang, and Y. Yu. Wasserstein distance guided representation learning for domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[46] P. Singh. Systematic review of data-centric approaches in artificial intelligence and machine learning. *Data Science and Management*, 2023.

[47] P. Sobolewski and M. Wozniak. Concept drift detection and model selection with simulated recurrence and ensembles of statistical detectors. *J. Univers. Comput. Sci.*, 19(4):462–483, 2013.

[48] X. Song, M. Wu, C. M. Jermaine, and S. Ranka. Statistical change detection for multi-dimensional data. In P. Berkhin, R. Caruana, and X. Wu, editors, *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, August 12-15, 2007*, pages 667–676. ACM, 2007.

[49] W. N. Street and Y. Kim. A streaming ensemble algorithm (SEA) for large-scale classification. In D. Lee, M. Schkolnick, F. J. Provost, and R. Srikant, editors, *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, San Francisco, CA, USA, August 26-29, 2001*, pages 377–382. ACM, 2001.

[50] M. Sundararajan and A. Najmi. The many shapley values for model explanation. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9269–9278. PMLR, 13–18 Jul 2020.

[51] C. Villani. *Topics in optimal transportation*, volume 58. American Mathematical Soc., 2021.

[52] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Mach. Learn.*, 23(1):69–101, 1996.

[53] R. F. Woolson. Wilcoxon signed-rank test. *Wiley encyclopedia of clinical trials*, pages 1–3, 2007.

[54] L. Yu and H. Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 856–863, 2003.