

A Scalable Approach to Covariate and Concept Drift Management via Adaptive Data Segmentation

Vennela Yarabolu*

Computer Science
Indian Institute of Technology, Bombay
Mumbai, India
210050168@iitb.ac.in

Sonia Gupta

Mastercard
Gurugram, India
sonia.gupta@mastercard.com

Govind Waghmare

Mastercard
Gurugram, India
govind.waghmare@mastercard.com

Siddhartha Asthana

Mastercard
Gurugram, India
siddhartha.asthana@mastercard.com

Abstract

In many real-world applications, continuous machine learning (ML) systems are crucial but prone to data drift—a phenomenon where discrepancies between historical training data and future test data lead to significant performance degradation and operational inefficiencies. Traditional drift adaptation methods typically update models using ensemble techniques, often discarding drifted historical data, and focus primarily on either covariate drift or concept drift. These methods face issues such as high resource demands, inability to manage all types of drifts effectively, and neglecting the valuable context that historical data can provide. We contend that explicitly incorporating drifted data into the model training process significantly enhances model accuracy and robustness. This paper introduces an advanced framework that integrates the strengths of data-centric approaches with adaptive management of both covariate and concept drift in a scalable and efficient manner. Our framework employs sophisticated data segmentation techniques to identify optimal data batches that accurately reflect test data patterns. These data batches are then utilized for training on test data, ensuring that the models remain relevant and accurate over time. By leveraging the advantages of both data segmentation and scalable drift management, our solution ensures robust model accuracy and operational efficiency in large-scale ML deployments. It also minimizes resource consumption and computational overhead by selecting and utilizing relevant data subsets, leading to significant cost savings. Experimental results on classification task on real-world and synthetic datasets show our approach improves model accuracy while reducing operational costs and latency. This practical solution overcomes inefficiencies in current methods, providing a robust, adaptable, and scalable approach to maintaining high-performance ML systems across various applications.

*Work done during an internship at Mastercard, India

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

CODS-COMAD Dec '24, December 18–21, 2024, Jodhpur, India

© 2024 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-1124-4/24/12

<https://doi.org/10.1145/3703323.3703337>

CCS Concepts

• **Information systems** → **Data management systems**; • **Computing methodologies** → **Machine learning**.

Keywords

Concept Drift, Covariate Shift, Data Segmentation

ACM Reference Format:

Vennela Yarabolu, Govind Waghmare, Sonia Gupta, and Siddhartha Asthana. 2024. A Scalable Approach to Covariate and Concept Drift Management via Adaptive Data Segmentation. In *8th International Conference on Data Science and Management of Data (12th ACM IKDD CODS and 30th COMAD) (CODS-COMAD Dec '24)*, December 18–21, 2024, Jodhpur, India. ACM, New York, NY, USA, 9 pages. <https://doi.org/10.1145/3703323.3703337>

1 Introduction

In the rapidly evolving landscape of modern technology, ML systems have become indispensable for various applications. However, these systems are frequently challenged by data drift—a phenomenon where discrepancies between historical training data and future test data cause significant performance degradation and operational inefficiencies. Addressing data drift is critical to maintaining the reliability and efficiency of ML systems, yet traditional methods often fall short in several respects. Data drift is classified into two main categories: covariate shift and concept drift [17, 18, 34, 38]. Covariate shift is a type of data drift where the distribution of the input features (covariates) changes between the training and test datasets, but the relationship between the input features X and the target variable y (the conditional distribution $P(y|X)$) remains the same. Concept drift refers to a phenomenon where the relationship between the input data (features) and the target variable changes over time. This change affects the conditional distribution $P(y|X)$, meaning the way the output is generated from the input features evolves. Concept drift can significantly degrade model performance if not properly managed. Common methods like periodic retraining and re-weighting recent data are often ineffective, leading to accuracy drops and performance variations. Traditional approaches usually focus on either covariate or concept drift, often neglecting comprehensive solutions and discarding valuable historical data. This paper argues for incorporating drifted data into the training process to enhance model accuracy and robustness. We propose a scalable framework that combines data-centric approaches with

adaptive management of both covariate and concept drift. Our solution uses sophisticated data segmentation to select optimal data batches for training, ensuring models remain accurate over time.

Our framework integrates data segmentation and drift management to enhance model accuracy and efficiency in large-scale ML deployments. By focusing on relevant data subsets, we reduce resource use, lowering costs and latency. Experimental results show improved accuracy, reduced costs, and adaptability to evolving data. The framework addresses both covariate shift and concept drift, maintaining model performance over time, and easily integrates with existing ML pipelines for smooth transitions and tracking. This approach enables organizations to maintain high-quality predictions and informed decisions in dynamic data environments. The summary of our contributions is as follows:

- **Scalability and Efficiency:** We introduce a robust framework designed to handle data drift, including both concept drifts and covariate shifts. Our approach is scalable and efficient, combining the strengths of data-centric methods with multiple drift management techniques.
- **Adaptive Data Subset Selection:** We develop an efficient data subset selection algorithm that initially identifies core data segments while discarding those affected by concept drift. Subsequently, it selects core data batches from these segments that are similar to the test set, thereby mitigating covariate shift. These steps reduce the amount of data required for training leading to operational efficiencies.
- **Optimal Performance:** Extensive experiments on synthetic and real datasets demonstrate that our method achieves better results while maintaining efficiency. Ablation on the trade-off between the % of data used and prediction accuracy underscore the cost benefits for practical deployments.

2 Related Work

Drift detection is crucial in the environments where data distributions evolve. When drifts occur, the model performance can drop. Various drift detection techniques ([17, 29, 30, 45]) have been developed to identify drifts by pinpointing change points or time intervals [2]. Effective drift detection methods ensure that models remain accurate and relevant by signaling the need for retraining or adjustments, thereby allowing the model to adapt to the new data distribution. These techniques are broadly classified into supervised methods, such as DDM [15], EDDM [1], and ADWIN [3], and unsupervised methods, like HDDDM [8] and DAWIDD [21].

Model-centric and data-centric approaches address data drift differently. Model-centric methods, like retraining and online learning, adapt models to changing patterns, enhancing adaptability but at high cost and complexity [4, 10, 12, 16, 19, 39]. Data-centric strategies, such as subset selection and reweighting, ensure training data remains relevant, improving efficiency. Combining both manages drift effectively, balancing accuracy and resource use to maintain model reliability and performance in dynamic environments. This hybrid approach ensures models stay robust against evolving data trends over time.

In addition to model-centric methods, data-centric approaches have been developed to adapt to concept drift. Data reduction techniques [37] focus on cleaning data by removing noisy samples and

features. Drift understanding techniques [11] filter out obsolete data using the newest data segment as a pattern, based on cumulative distribution function comparisons. Once filtered out, samples are not reselected, even if they could be beneficial later. A notable technique in this category is CVDTE [13], which aims to select samples that do not yield conflicting predictions between previous and current models. However, these techniques share a fundamental limitation: they lack mechanisms to validate if the data preprocessing steps genuinely enhance model accuracy. In comparison, we take a more data-driven approach by explicitly evaluating models on selected data segments, while minimizing computational costs.

The issue of data drift, which refers to the variation in model accuracy over time, has been a significant area of research in the ML community. It arises when the model encounters test data that differ substantially from the training data, leading to reduced prediction accuracy. Numerous studies have explored this challenge, particularly in the context of streaming or continuously arriving data, and have proposed various approaches to address it [5, 34, 42, 43]. In supervised learning tasks, where features X are used to predict labels y , data drift is commonly caused by two factors [34]: (1) Covariate shift, which occurs when the distribution of features X changes, such as when new types of incidents with previously unseen feature values arise; and (2) Concept drift, which happens when the underlying relationship between features X and labels y shifts, for example, due to changes in a system and its dependencies, leading to different causal relationships between symptoms and components. To tackle the problem of data drift, existing methods can be categorized into three main types: (a) Window-based approaches [18], which employ a sliding window of recent data for training updated models; (b) Shift detection methods [36], which utilize statistical tests to identify the occurrence of data drift and trigger model retraining only when such shifts are detected; and (c) Ensemble-based strategies [6], which create ensembles of models trained on previous data, combining their predictions through a weighted average to maintain accuracy.

3 Background

3.1 Covariate Shift

Covariate shift occurs when the distribution of input features changes between training and test datasets, while their relationship with the target variable remains the same [22, 38]. This shift can result from environmental changes, data collection methods, or sampling procedures. Addressing covariate shift involves reweighting training data or using domain adaptation methods to maintain model accuracy on new data. Most approaches focus on training and test datasets without considering continuous time [9], addressing the issue by modifying training objectives or adjusting the importance of training data to improve test accuracy [20, 32, 38, 40, 46]. For input features X , the covariate shift is defined as follows:

$$P_{Train}(X) \neq P_{Test}(X) \quad (1)$$

3.2 Concept Drift

Concept drift occurs when the relationship between input features and the target variable evolves over time [44]. This shift can happen

gradually or suddenly, altering the underlying data patterns. Concept drift challenges ML models by potentially decreasing their accuracy and reliability if not detected and addressed. To handle concept drift effectively, it's essential to continuously monitor model performance and update the model to adapt to new data patterns, ensuring sustained accuracy and relevance. Broadly, there are three solution categories to handle concept drift as window-based [18, 27, 28], detection-methods [5, 15, 23, 35, 36, 43] and ensemble methods [6, 12, 41, 47]. For input features X and target y , the concept drift is defined as follows:

$$P_{Train}(y|X) \neq P_{Test}(y|X) \quad (2)$$

4 Overview

The overall process of our method is shown in Figure 3. Our approach tackles the two primary causes of data drift: covariate shift and concept drift. The idea is to select the most relevant batches from the training data segments based on their relationship to the test samples, and use those batches to train the model for accurate inference. Inline with [31], this method is based on two conjectures: (1) If the decline in accuracy is due to the training and test data residing in different regions of the data space (covariate shift), it is logical to prioritize the training batch t whose features (X_t) are closest to those of the test data (x_*). (2) If the accuracy drop results from changes in the $x \rightarrow y$ relationship over time (concept drift), then it is prudent to exclude data segments that exhibit concept drift relative to the current data segment, as indicated by gradient scores.

To address covariate shift, a random forest R is trained meticulously on all labeled training batches $\{(X_1, y_1), \dots, (X_T, y_T)\}$. This sophisticated technique partitions the training data space, harnessing the random forest's prowess in organizing complex data distributions [7, 31]. During testing, we rank the training batches based on their similarity to the test sample by analyzing the leaf nodes in R where the test sample is mapped. Batches are then ranked according to the concentration of training points that fall within these leaf nodes, ensuring that the most pertinent data is utilized to refine model accuracy. This process is highlighted in the Figure 1.

We build on top of concept drift approach Quilt [26]. The concept drift detection component monitors each sample in the data stream to identify potential changes in data patterns. Various drift detection methods can be employed, based on shifts in data distribution or model performance. When a drift is detected, a new data segment is created from the drift point and becomes the current segment. The data segment selection component then updates the model using selected segments. If no drift is detected, the sample is added to the existing segment. The framework performs two main operations for selecting data segments: (1) discarding segments that no longer align with the current data pattern, and (2) selecting a core subset of stable segments for efficient model training. This approach leverages gradient-based disparity and gain scores as described in the [26], which are computationally efficient and independent of specific data characteristics, unlike traditional statistical distance measures that can struggle with high-dimensional data and scalability issues. This method allows for adaptive handling of data drift without needing ground truth labels for retraining.

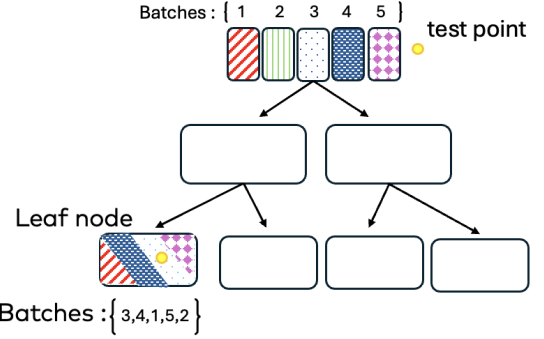


Figure 1: Covariate shift ranking of training batches 1,2,3,4 and 5 with respect to test point shown in yellow. In the leaf node, batches are ranked as 3,4,1,5 and 2. Here, batch 3 has lowest covariate shift.

5 Data Selection

5.1 Covariate Shift Ranking

To prioritize training data segments based on covariate shift, we focus on their proximity to test points in the data space. Although one might consider ranking training batches by their average Euclidean distance from the test point, this method has limitations. Euclidean distance computation becomes expensive with larger batches, is prone to outliers [14], and struggles with high-dimensional data. Instead, we recommend using decision trees and random forests for ranking batches, similar to [31]. This approach scales well, is more robust to outliers, and handles high-dimensional data more effectively, making it a practical choice for complex datasets. We rely on methods described in MatchMaker [31] to rank batches using decision trees and random forest as follows:

Covariate Shift Ranking of the Batches. Decision trees classify data by partitioning it at feature thresholds that optimize prediction accuracy, grouping similar samples into the same leaf nodes. When a new sample is tested, it is routed to a leaf node, and its label is predicted based on the majority label within that node. We use this mechanism to evaluate training batches for covariate shift, prioritizing those that are closer to the test sample. Let $\{(X_1, y_1), \dots, (X_T, y_T)\}$ denote training batches. Once decision tree is constructed on these batches, let $N[k][t]$ indicate the number of samples from batch t that fall into leaf node k . Now, for a test point that is assigned to leaf node k_* , one has to calculate covariate shift ranking $\mathbf{Rank}_{cov_shift}$ of the training batches. This ranking is computed by ordering $N[k_*][t]$ starting from lowest covariate shift to highest as shown:

$$\mathbf{Rank}_{cov_shift} = \text{argsort}\{N[k_*][1], \dots, N[k_*][T]\} \quad (3)$$

As random forest are capable of modeling high-dimensional data, this approach is extended to the random forests for better performance. The visual summary of covariate-ranking of the batches is shown in the Figure 1.

5.2 Concept Drift

To tackle the concept drift, our approach employs a robust framework Quilt [26], centered on two key tasks: (1) removing data segments that show concept drift compared to the current segment, and (2) selecting a core set of stable data segments to train the model efficiently while maintaining accuracy. This method utilizes disparity and gain scores, calculated from gradient values on training and validation sets, ensuring minimal computational cost. In the next sections, we discuss the gradient computation along with disparity and gain score formulation as defined in the Quilt [26].

Gradient Computation: The last layer of the neural network calculates the logits for each class. Let $X'_i \in \mathbb{R}^{d'}$ be the embedding feature of the i th input data X_i with a hidden layer dimension of d' , and $z_i \in \mathbb{R}^c$ be the logit outputs computed by $z_i = w \cdot X'_i + b$ using the last layer weights $w \in \mathbb{R}^{d' \times c}$ and bias $b \in \mathbb{R}^c$. To convert a logit z_i into a probability vector \hat{y}_i , softmax function is used as follows: $\hat{y}_i = \text{softmax}(z_i) = \frac{e^{z_{ij}}}{\sum_{j=1}^c e^{z_{ij}}}$. We can also rewrite the model output \hat{y}_i as a function of the model parameters θ and input data X_i as $\hat{y}_i = f_\theta(X_i)$. Given the model output \hat{y}_i and the true label y_i , the cross-entropy loss between them is $L_i = L(y_i, \hat{y}_i) = -\sum_{j=1}^c y_{ij} \log(\hat{y}_{ij})$. The last layer gradient approximation is given as $g = (\nabla_b L, \nabla_w L)$ where gradients of the front layers are not used. Using the chain rule, one can compute the gradient of the i^{th} sample as follows: $g = (\nabla_b L, \nabla_w L) = (\hat{y}_i - y_i, (\hat{y}_i - y_i) \cdot X'_i)$.

Disparity Score: The disparity score (D), is a measurement of dissimilarity between two data distributions. It detects segments exhibiting concept drift. Concept drift is characterized by a change in the posterior distribution $P(y|X)$ while the data distribution $P(X)$ remains constant. Essentially, it reflects variations in the predicted labels y for the same input data. To quantify this change, one can use the measure $\mathbb{E}[\|y_t - y_v\|]$, which represents the expected label difference between a training subset and a validation set, where y_t and y_v denote the true labels from the training and validation sets, respectively. This measure is analogous to the concept drift severity [33]. Direct computation of this measure is computationally expensive as it requires identifying similar samples across the training and validation sets and comparing their label differences. To overcome this, Quilt [26] proposed a gradient-based score as an efficient approximation as follows. The disparity score D of a training subset T with respect to a validation set V is defined as:

$$D(T, V) = \left\| \frac{1}{|T|} \sum_{t=1}^{|T|} g_t - \frac{1}{|V|} \sum_{v=1}^{|V|} g_v \right\| = \|\mathbb{E}[g_t] - \mathbb{E}[g_v]\|, \quad (4)$$

where $|V|$ denotes the size of the validation set. Also, the D score measures the L_2 -norm distance between two gradient vectors.

Gain Score: The gain score is built on top of methods introduced in [24, 25]. Consider historical data for both training and validation. Studies indicate that selecting a subset where the inner product of the average gradients between the subset and the validation set (known as the gain) is positive can lower the model's validation loss during training. Essentially, gradient vectors represent the direction and size of updates in gradient descent, and aligning these gradients between the training and validation sets helps improve model performance. The gain score G for a training subset T with

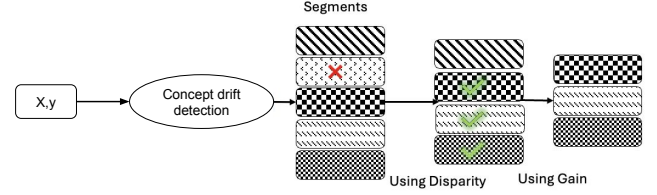


Figure 2: Subsegment selection approach using gradient based disparity and gain scores

respect to a validation set V is:

$$G(T, V) = \frac{1}{|T|} \sum_{t=1}^{|T|} g_t \cdot \frac{1}{|V|} \sum_{v=1}^{|V|} g_v = \mathbb{E}[g_t] \cdot \mathbb{E}[g_v], \quad (5)$$

where \cdot represents the dot product of the gradient vectors. The subsegment selection procedure based on gradient based disparity and gain score is highlighted in the Figure 2.

6 Data Subset Selection Algorithm

Algorithm 1 provides a method for detecting covariate shift by examining how sample distributions vary across different batches through the lens of a trained model's decision trees. Initially, the entire training data segments are utilized to train the model R . For each individual decision tree T_i within the model, the algorithm computes a score $S[k][t]$ for every batch t across each leaf node k . This score quantifies how many other batches within the same leaf node contain fewer samples $N[k][t]$ compared to the batch t under consideration. By evaluating these scores, the algorithm can detect shifts in feature distributions among different batches, which may signal potential covariate shifts.

Algorithm 2 outlines the procedure for selecting data segments to optimize model training. The process starts by initializing the model parameters and proceeds through a series of epochs. For each epoch, the algorithm initializes an empty training subset S . It then calculates the average gradient over the validation set d_{V_N} . Next, the algorithm iterates over previous data segments to compute their gradient averages and evaluates their gain and disparity scores. Data segments with a positive gain score and a disparity score below a specified threshold are added to the training subset S . The current training data d_{T_N} is always included in S to ensure recent data is used in training.

Additionally, the algorithm initializes an empty set for the best batches B_{best} . For each sample v in the validation set d_{V_N} , it retrieves the rankings of the batches that we had from algorithm 1 based on the mapped leaf of v in a random forest (rf). The algorithm then iterates through these ranked batches and adds them to B_{best} if they are part of the selected segments S , breaking the loop once a suitable batch is found. This ensures that the best batches from the validation set, which are also part of the selected training segments, are prioritized. The model parameters are updated using the learning rate η and the computed gradients from the best batches B_{best} . This process is repeated for T epochs. Finally, the algorithm returns the updated model parameters θ_T .

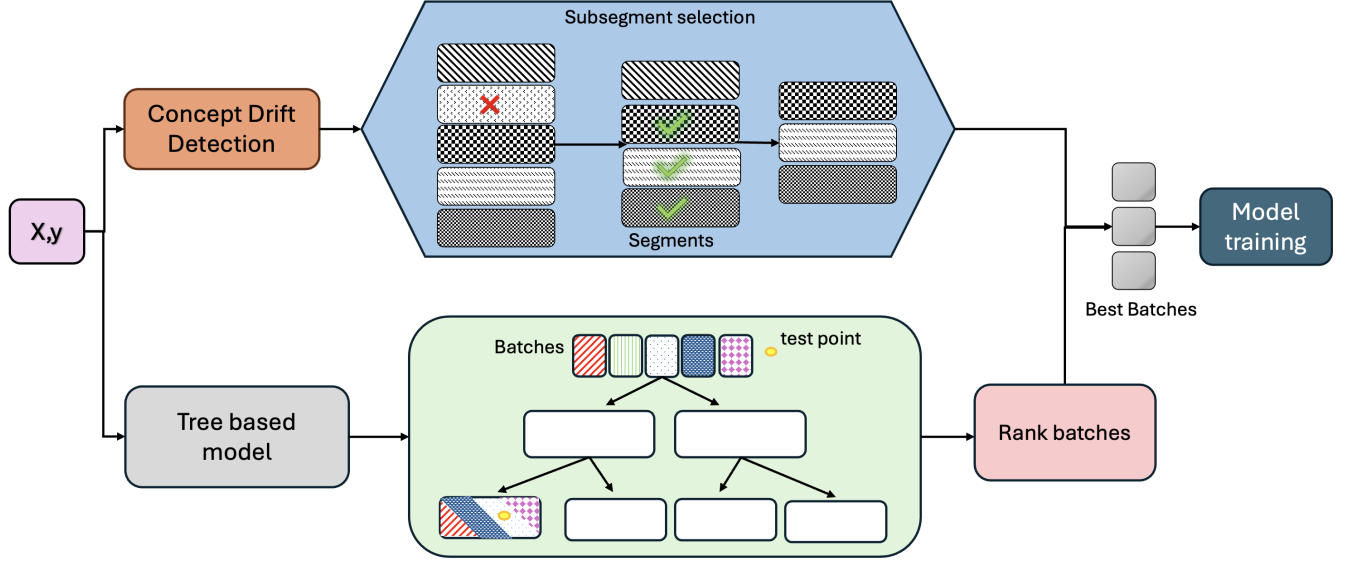


Figure 3: Complete architecture of our method. The top branch focuses on concept drift while bottom branch ranks train batches based on covariate shift.

Algorithm 1: Covariate Shift Scoring

Input: Training data batches $\{(X_1, y_1), \dots, (X_T, y_T)\}$
Output: Stored values $S[k][t]$ for each tree T
 Train model R on entire data $(X_1, y_1), \dots, (X_T, y_T)$;
for each tree $T_i \in R$ do
 | Store $S^i[k^i][t] = \sum_{(t \neq t')} \{1 \text{ if } N[k^i][t] > N[k^i][t']\}$;

7 Experiments

7.1 Datasets

We conducted experiments using a varied selection of datasets, including five synthetic and five real-world datasets. Table 1 provides detailed descriptions and summary statistics for each dataset used in our research. The synthetic datasets were deliberately crafted to represent different forms of concept drifts, as outlined by [30]. All datasets except Covcon are taken and preprocessed inline with [26] while Covcon is taken and preprocessed as done in [31].

- **SEA:** Streaming Ensemble Algorithm (SEA) is a standard dataset for simulating sudden concept drifts. The samples are in a 3D feature space with random numeric values between 0 and 10.
- **RandomRBF:** Random Radial Basis Function is used to make a number of random centroids and new samples are generated by selecting the center of centroids.
- **Sine:** It contains four numerical features with values that range from 0 to 1. Two of the features are relevant to a given binary classification task, while the two other features simulate noise.
- **Hyperplane:** Here, hyperplanes are viewed as concepts and varied orientations are used to simulate drifts. A hyperplane is defined by feature weights, and weights are drifted over time. There are ten relevant features including two drift features.

Algorithm 2: Data Selection Algorithm

Input: Previous data segments $D_{prev} = \{d_1, \dots, d_{N-1}\}$, current segment d_{T_N} , validation set d_{V_N} , loss function L , learning rate η , maximum epochs T , disparity threshold T_d , batch size B , number of estimators $n_{estimators}$, maximum depth d_{max}

Output: Final model parameters θ_T

for epoch t in $[1, \dots, T]$ do
 | Initialize training subset $S = \emptyset$;
 | $g_V = \frac{1}{|d_{V_N}|} \sum_{j=1}^{|d_{V_N}|} g_j$;
 | **for segment d in D_{prev} do**
 | | $g_d = \frac{1}{|d|} \sum_{k=1}^{|d|} g_k$;
 | | $G_d = g_d \cdot g_V$;
 | | $D_d = \|g_d - g_V\|$;
 | | **if $G_d > 0$ and $D_d < T_d$ then**
 | | | $S = S \cup d$;
 | $S = S \cup d_{T_N}$;
 | Initialize best batches $B_{best} = \emptyset$;
 | **for each sample v in d_{V_N} do**
 | | Get the rankings of the batches based on the mapped leaf of v in r ;
 | | **for each batch in the rankings do**
 | | | **if batch is in S then**
 | | | | $B_{best} = B_{best} \cup \text{batch}$;
 | | | | break ;
 | Update $\theta_t = \theta_{t-1} - \eta \frac{1}{|B_{best}|} \sum_{e \in B_{best}} \nabla_{\theta} L(e)$;

return final model parameters θ_T

- **Covcon and Covcon_5M**: A 2-dimensional dataset to have covariate shift and concept drift. The decision boundary at each point is given by $\alpha * \sin(\pi x_1) > x_2$.
- **Electricity [48]**: This is Australian New South Wales Electricity Market data from 1996 to 1998, measured every 30 minutes.
- **Weather**: Points measures the weather in Bellevue NE during the period of 1949–1999.
- **Spam**: It consists of email messages from the Spam Assassin Collection. There are 9,324 samples of messages and a message is represented by 499 features of boolean bag-of-words. The labels denote whether a message is spam or not.
- **Usenet1 & 2**: Two real datasets are based on the 20 news group collection with three topics: medicine, space, and baseball. Each sample contains messages about different topics, and a user labels them sequentially by personal interests whether the topic of a message is interesting (1) or junk (0).
- **Covertyp [31]**: This dataset contains 581K samples describing 7 forest cover types for 4 region in the Roosevelt National Forest.

7.2 Model Training and Hyperparameters

Our approach involves employing two main strategies for training our models. First, we utilize a Random Forest to partition (Algorithm 1) and rank batches of the data segment based on a specified batch size. It addresses covariate shift. Then, we employ a simple neural network classifier trained using cross-entropy loss to deal with concept drift (Algorithm 2). For a random forest, we use grid search over batch size [grid over 3-5 values], number of estimators $n_{\text{estimators}}$, maximum depth d_{max} . For all experiments, $n_{\text{estimators}} = 50$ and maximum depth $d_{\text{max}} = 20$. Batch size is reported in Table 1. In Algorithm 2, a neural network classifier with a single hidden layer with 256 nodes is employed. The value of disparity threshold for each data segment is calculated using Bayesian optimization with the search interval in (0,2) inline with [26]. The learning rate is set to 1×10^{-3} and early stopping with patience 10 is used for termination with maximum number of epochs limited to 2000. For computation, we have used RTX Quadro with 24 GB of VRAM and 32 GB of RAM on Linux machine. Codebase is developed using PyTorch. The subset selection method discards the segments based on gain and disparity scores. Further, only relevant batches from the remaining segments are selected leading to reduction in data used for training. This value for each dataset is reported in Table 2 and 3 on the last line '% of data used'.

7.3 Baseline Algorithms

- **Naïve Methods**:
 - **Full Data and Current Segment**: Full data uses all data segments for training, ensuring maximum information but ignoring the relevance of older data. Current segment trains only on the latest data, assuming it to be the most relevant.
- **Model-centric Methods**:
 - **HAT [3]**: Trains a Hoeffding Adaptive Tree classifier online using the full dataset, adapting to data distribution changes by incrementally updating its structure.
 - **ARF [19]**: Implements an Adaptive Random Forest classifier on each sample, leveraging the entire dataset. ARF combines multiple decision trees to improve prediction accuracy.

- **Learn++.NSE [12]**: Constructs an ensemble of models trained on previous data segments and adjusts their weights based on their performance on the current data segment.
- **SEGA [39]**: Ensembles models trained from equal-length segments of historical data and selects segments with minimum kNN-based distributional discrepancy with the current data.
- **Data-centric Method**:
 - **CVDTE [13]**: This method trains a Cross-Validation Decision Tree Ensemble classifier on individual samples that do not have conflicting predictions caused by shifted decision boundaries due to concept drifts.
- **Data Subset Selection Methods**:
 - **GLISTER [25]**: Trains a neural network classifier based on data subset selection, ranking data subsets by gains and selecting the top-k subsets within a predefined budget.
 - **GRAD-MATCH [24]**: Trains a neural network classifier via data subset selection, simultaneously selecting data subsets and adjusting their weights to minimize gradient error.
- **Quilt [26]**: A data-centric framework designed to identify and select data segments that maximize model accuracy, utilizing gradient-based scores for efficient data segment selection.

7.4 Experimental Results

For each dataset, we report accuracy, F1 score and runtime results of our method by setting the last (latest) segment as the current segment. This means that we use the most recent segment of data to evaluate how well our method performs in a real-world scenario where data is continuously evolving. We compare our method with other baseline methods across all ten datasets, as shown in Table 2 and 3. Our results indicate that our method consistently outperforms all the baselines in terms of accuracy. This superior performance is attributed to our method's effective utilization of drifted data, which allows it to maintain high accuracy even when the data distribution changes over time. We have used the public codebase of Quilt [26] to get the results of all baselines.

In comparison, the Full Data method, which uses all available data including drifted data, does not perform as well because it is forced to incorporate data that may no longer be relevant to the current segment. On the other hand, the Current Segment method, which only uses the most recent segment of data, fails to leverage valuable historical data, leading to lower accuracy. HAT, another baseline, performs worse than our method because it adaptively learns from recent data without using previous models or historical data, limiting its ability to adapt effectively to data drift. The ensemble methods, including ARF, Learn++.NSE, and SEGA, also underperform compared to our method. ARF, for example, can lose useful previous knowledge when replacing an obsolete tree for drift adaptation, which negatively impacts its performance. Learn++.NSE and SEGA attempt to save all past models or a buffer's worth of them and use the current data segment to create ensembles. However, these models, trained on previous data segments, struggle to fit the current data segment accurately with simple ensemble techniques. CVDTE, another baseline, performs worse than our method because it simply collects samples that do not have conflicting predictions, regardless of whether these samples actually benefit model accuracy. This method overlooks the importance and effectiveness of

Table 1: Dataset statistics.

Type	Dataset	Size	Features	Classes	Num. Segments	Segment size	Num. batches per segment	Batch size
Synthetic	SEA	16K	3	2	8	2K	20	100
	Random RBF	16K	10	2	8	2K	20	100
	Sine	16K	4	2	8	2K	20	100
	Hyperplane	16K	10	2	8	2K	20	100
	Covcon	10K	2	2	5	2K	2	1K
	Covcon_5M	5M	2	2	10	500K	10	50K
Real	Electricity	43.2K	6	2	10	4.32K	20	216
	Weather	18K	8	2	10	1.8K	20	90
	Spam	9.3K	499	2	9	1.036K	14	74
	Usenet1	1.5K	99	2	5	300	2	150
	Usenet2	1.5K	99	2	5	300	3	100
	Covertypes	581K	54	7	10	58.1K	10	5.81K

Table 2: Accuracy, F1 score and runtime (sec) on synthetic datasets. % of data used indicates how much data is used compared to Quilt with 100% data to get best performance. Best numbers are in bold. "-" indicates either memory error or time limit error.

Methods	SEA			RandomRBF			Sine			Hyperplane			Covcon			Covcon_5M		
	Acc	F1	Time	Acc	F1	Time	Acc	F1	Time	Acc	F1	Time	Acc	F1	Time	Acc	F1	Time
Full Data	.849	.881	3.36	.821	.820	9.43	.449	.436	2.25	.843	.844	2.41	.561	.576	1.57	.401	.444	800
Current segment	.864	.888	0.20	.679	.673	0.56	.899	.898	0.94	.893	.894	0.46	.941	.540	2.11	.904	.551	30
HAT	.825	.862	1.38	.514	.519	2.35	.293	.305	1.67	.862	.862	2.23	.83	.674	.733	-	-	-
ARF	.825	.863	23.49	.645	.642	44.64	.821	.823	21.40	.793	.793	26.86	.873	.89	9.689	-	-	-
Learn++-NSE	.804	.836	6.65	.611	.612	5.68	.925	.925	5.73	.755	.755	7.05	.690	.671	6.007	-	-	-
SEGA	.797	.842	4.37	.825	.825	4.47	.253	.260	4.35	.851	.851	4.49	.786	.601	0.67	-	-	-
CVDTE	.806	.810	0.02	.614	.621	0.05	.857	.835	0.02	.752	.752	0.04	.940	.540	0.06	-	-	-
GLISTER	.857	.885	25.89	.794	.794	63.73	.879	.876	14.93	.905	.905	21.64	-	-	-	-	-	-
GRAD-MATCH	.853	.884	2.13	.790	.790	6.66	.547	.529	0.80	.845	.845	1.40	-	-	-	-	-	-
Quilt	.893	.909	.993	.833	.833	2.836	.938	.936	3.909	.911	.912	1.810	.988	.918	.95	.923	.415	27
Our Method	.899	.912	1.457	.839	.838	3.382	.955	.949	4.369	.924	.922	2.681	.988	.922	1.44	.968	.603	731
% of data used	(89.14%)			(93.43%)			(94.25%)			(87.94%)			(62.28%)			(68.49%)		

Table 3: Accuracy, F1 score and runtime (sec) on real datasets. % of data used indicates how much data is used compared to Quilt with 100% data to get best performance. Best numbers are in bold. "-" indicates either memory error or time limit error.

Methods	Electricity			Weather			Spam			Usenet1			Usenet2			Covertypes		
	Acc	F1	Time	Acc	F1	Time	Acc	F1	Time	Acc	F1	Time	Acc	F1	Time	Acc	F1	Time
Full Data	.694	.758	7.42	.800	.641	4.33	.970	.973	1.17	.576	.512	0.23	.701	.413	0.18	.642	.650	105
Current segment	.709	.756	0.52	.756	.509	0.26	.955	.963	0.16	.752	.716	0.16	.745	.613	0.18	.598	.584	1.76
HAT	.691	.743	6.43	.729	.452	2.10	.888	.847	25.67	.622	.558	0.87	.730	.472	0.87	.536	.538	250
ARF	.713	.762	57.36	.775	.542	30.51	.921	.931	44.83	.629	.616	4.12	.682	.31	4.04	.483	.398	1843
Learn++-NSE	.698	.734	17.26	.703	.523	7.86	.928	.942	3.81	.433	.412	0.35	.637	.251	0.33	-	-	-
SEGA	.637	.697	10.26	.777	.602	4.11	.858	.851	6.67	.403	.318	0.84	.630	.207	0.84	.513	.488	553
CVDTE	.689	.736	0.04	.731	.497	0.03	.917	.918	0.12	0.718	.624	0.01	.689	.523	0.01	.604	.6	2.013
GLISTER	.698	.741	77.46	.793	.649	40.79	.971	.974	14.52	.771	.736	2.07	.744	.603	1.89	-	-	-
GRAD-MATCH	.686	.748	5.97	.795	.622	3.51	.968	.972	1.13	.630	.591	0.13	.679	.420	0.12	-	-	-
Quilt	.831	.775	3.613	.776	.635	3.284	.988	.976	2.498	.892	.782	.324	.771	.640	1.03	.622	.615	115
Our Method	.833	.825	2.528	.778	.696	1.302	.992	.996	.89	.904	.841	.119	.879	.794	.147	.689	.665	120
% of data used	(92.10%)			(56.28%)			(79.56%)			(81.48%)			(73.33%)			(53.42%)		

the gathered samples in enhancing the model’s accuracy on the present data segment. Among the data subset selection methods, GLISTER’s targeted sample selection demonstrates more consistent results compared to GRAD-MATCH’s random batch selection. However, the substantial computational overhead of GLISTER renders it less feasible for use in real-time scenarios. In contrast, our method’s approach of adaptively selecting suitable data segments with explicit model evaluations demonstrates the advantage of taking a data-centric approach. By explicitly evaluating and selecting the most relevant data segments, our method achieves better accuracy and efficiency, highlighting its effectiveness in managing data drift in ML systems.

7.5 Ablation Study

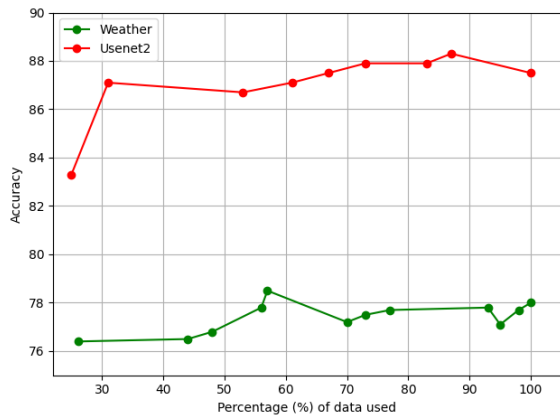


Figure 4: Trade-off between % of data used vs accuracy.

In our experiments with the Usenet2 and Weather dataset, we evaluated the effect of varying the proportion of data used for training on the model’s accuracy. The goal was to determine the minimal amount of data required to achieve optimal performance. The first graph for Usenet2 (red line in Figure 4) presents this relationship, where we see a significant increase in accuracy when utilizing 32% of the data, reaching approximately 87%. This initial boost suggests that even a smaller subset of the data can capture the essential patterns necessary for effective model training. As we continue to use more data, the accuracy sees a gradual increase and then stabilizes, indicating that the additional data provides diminishing returns. The highest accuracy is observed around 87% data utilization, after which the performance slightly decreases, reinforcing the notion that more data does not always equate to better accuracy and might even introduce noise or redundancy.

In the second graph for Weather (green line in Figure 4), depicting the weather dataset, the highlighted points mark a significant insight into data efficiency. At 58% data utilization, the model reaches its peak accuracy of 78.5%, which is higher than the accuracy obtained using the entire dataset. This indicates an optimal subset of data that maximizes the model’s performance while minimizing the computational resources required. Notably, the accuracy drops when nearing 100% data utilization, which underscores the importance of strategic data selection over sheer volume. The pattern

Table 4: For each dataset, we show the runtime results for our method, including random forest (RF) training time (Algorithm 1), Model training time (Algorithm 2), and total time in seconds. We also provide an ablation when only Algorithm 1 is used against both algorithms are used.

Dataset	RF train time	Model train time	Total train time	Accuracy	
				Only Alg. 1	Alg. 1 and 2
SEA	1.213	0.244	1.457	.784	.899
Random RBF	1.360	2.022	3.382	.704	.839
Sine	1.238	3.131	4.369	.274	.955
Hyperplane	1.252	1.429	2.681	.733	.924
Covcon	0.710	0.303	1.441	.421	.988
Covcon_5M	707	24	731	.709	.968
Electricity	1.437	1.476	2.528	.718	.833
Weather	0.590	0.712	1.302	.775	.778
Spam	0.276	0.614	0.890	.883	.992
Usenet1	0.035	0.084	0.119	.808	.904
Usenet2	0.037	0.056	0.147	.771	.879
Coverttype	46	74	120	.647	.689

observed here suggests that careful curation of training data segments, focusing on the most relevant subsets, can lead to superior model performance and operational efficiency. Our ablation study on both the Usenet2 and weather datasets reveals that optimal performance can be achieved with significantly less data than the full dataset. By focusing on the most relevant data, we can maintain or even improve model accuracy, making the training process more resource-efficient and effective. These findings highlight the importance of strategic data selection in developing robust and scalable ML models. The ablation of using just Algorithm 1 and training time across the modules is shown in the Table 4.

8 Limitations & Conclusion

In this paper, we addressed the critical issue of data drift in ML systems by introducing a novel, scalable, and flexible framework. It integrates data-centric approaches with adaptive management of both covariate and concept drift. Our framework employs advanced data segmentation techniques to identify optimal data batches that reflect test data patterns, ensuring models remain relevant and accurate over time. This approach enhances model robustness by including drifted data into the training process, minimizes resource consumption, and reduces computational overhead, leading to significant cost savings. Our results on synthetic and real datasets demonstrate significant improvements in accuracy, operational cost reduction, and faster ML inference compared to state-of-the-art solutions. However, limitations such as challenges in identifying and segmenting data batches in dynamic environments and computational complexity in real-time data segmentation remain. Future work will refine segmentation techniques for better drift detection and model adaptation, conduct extensive tests on diverse datasets, and develop systems to adjust data segment and model importance based on temporal performance. Enhancements will include advanced similarity metrics using deep feature representations and temporal patterns for improved model selection. We aim to inspire further research into effective data drift solutions, enhancing the practicality and reliability of ML systems across applications.

References

- [1] Manuel Baena-Garcia, José del Campo-Ávila, Raul Fidalgo, Albert Bifet, Ricard Gavaldà, and Rafael Morales-Bueno. 2006. Early drift detection method. In *Fourth international workshop on knowledge discovery from data streams*, Vol. 6. Citeseer, 77–86.
- [2] Michele Basseville, Igor V Nikiforov, et al. 1993. *Detection of abrupt changes: theory and application*. Vol. 104. prentice Hall Englewood Cliffs.
- [3] Albert Bifet and Ricard Gavaldà. 2007. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*. SIAM, 443–448.
- [4] Albert Bifet and Ricard Gavaldà. 2009. Adaptive Learning from Evolving Data Streams. In *Advances in Intelligent Data Analysis VIII*, Niall M. Adams, Céline Robardet, Arno Siebes, and Jean-François Boulicaut (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 249–260.
- [5] Albert Bifet and Ricard Gavaldà. 2007. Learning from Time-Changing Data with Adaptive Windowing. *Proceedings of the 7th SIAM International Conference on Data Mining*.
- [6] Dariusz Brzezinski and Jerzy Stefanowski. 2014. Reacting to Different Types of Concept Drift: The Accuracy Updated Ensemble Algorithm. *IEEE Transactions on Neural Networks and Learning Systems* (2014).
- [7] Alex Davies and Zoubin Ghahramani. 2014. The random forest kernel and other kernels for big data from random partitions. *arXiv preprint arXiv:1402.4293* (2014).
- [8] Gregory Ditzler and Robi Polikar. 2011. Hellinger distance based drift detection for nonstationary environments. In *2011 IEEE symposium on computational intelligence in dynamic and uncertain environments (CIDUE)*. IEEE, 41–48.
- [9] Gregory Ditzler, Manuel Roveri, Cesare Alippi, and Robi Polikar. 2015. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine* 10, 4 (2015), 12–25.
- [10] Pedro Domingos and Geoff Hulten. 2000. Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
- [11] Fan Dong, Jie Lu, Yiliao Song, Feng Liu, and Guangquan Zhang. 2021. A Drift Region-Based Data Sample Filtering Method. *IEEE Transactions on Cybernetics* (2021).
- [12] Ryan Elwell and Robi Polikar. 2011. Incremental Learning of Concept Drift in Nonstationary Environments. *IEEE Transactions on Neural Networks* (2011).
- [13] Wei Fan. 2004. Systematic data selection to mine concept-drifting data streams. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery.
- [14] Martin A. Fischler and Robert C. Bolles. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* (1981).
- [15] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. 2004. Learning with drift detection. In *Advances in Artificial Intelligence—SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence, Sao Luis, Maranhao, Brazil, September 29–October 1, 2004. Proceedings 17*. Springer, 286–295.
- [16] João Gama, Ricardo Rocha, and Pedro Medas. 2003. Accurate decision trees for mining high-speed data streams. Association for Computing Machinery.
- [17] João Gama, Indrundefined Zliobaitundefined, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A survey on concept drift adaptation. *ACM Comput. Surv.* 46, 4, Article 44 (mar 2014), 37 pages. <https://doi.org/10.1145/2523813>
- [18] Widmer Gerhard and Kubat Miroslav. 1996. Learning in the presence of concept drift and hidden contexts. *Machine Learning* (1996).
- [19] Heitor Murilo Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabricio Enembreck, Bernhard Pfahringer, Geoff Holmes, and Talel Abdesslem. 2017. Adaptive random forests for evolving data stream classification. *Machine Learning* (2017).
- [20] Arthur Gretton, Alex Smola, Jiayuan Huang, Marcel Schmittfull, Karsten Borgwardt, and Bernhard Schölkopf. 2008. Covariate shift by kernel mean matching. (2008).
- [21] Fabian Hinder, André Artelt, and Barbara Hammer. 2020. Towards non-parametric drift detection via dynamic adapting window independence drift detection (daw-idd). In *International Conference on Machine Learning*. PMLR, 4249–4259.
- [22] Mark G Kelly, David J Hand, and Niall M Adams. 1999. The impact of changing populations on classifier performance. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. 367–371.
- [23] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. 2004. Detecting Change in Data Streams.
- [24] Krishnateja Killamsetty, Sivasubramanian Durga, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. 2021. Grad-match: Gradient matching based data subset selection for efficient deep model training. In *International Conference on Machine Learning*. PMLR, 5464–5474.
- [25] Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. 2021. GLISTER: Generalization based Data Subset Selection for Efficient and Robust Learning. [arXiv:2012.10630 \[cs.LG\]](https://arxiv.org/abs/2012.10630) <https://arxiv.org/abs/2012.10630>
- [26] Minsu Kim, Seong-Hyeon Hwang, and Steven Euijong Whang. 2024. Quilt: Robust Data Segment Selection against Concept Drifts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 21249–21257.
- [27] Ralf Klinkenberg. 2004. Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis* 8 (2004).
- [28] Ivan Koychev and Robert Lothian. 2006. Tracking Drifting Concepts by Time Window Optimisation. In *Research and Development in Intelligent Systems XXII*.
- [29] B. Krawczyk, Leandro L. Minku, João Gama, Jerzy Stefanowski, and Michal Wozniak. 2017. Ensemble learning for data stream analysis: A survey. *Inf. Fusion* 37 (2017), 132–156. <https://api.semanticscholar.org/CorpusID:1372281>
- [30] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. 2018. Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering* 31, 12 (2018), 2346–2363.
- [31] Ankur Mallick, Kevin Hsieh, Behnaz Arzani, and Gauri Joshi. 2022. Matchmaker: Data Drift Mitigation in Machine Learning for Large-Scale Systems. In *Proceedings of Machine Learning and Systems*.
- [32] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. 2008. Domain adaptation with multiple sources. *Advances in neural information processing systems* 21 (2008).
- [33] Leandro L Minku, Allan P White, and Xin Yao. 2009. The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering* 22, 5 (2009), 730–742.
- [34] Jose G. Moreno-Torres, Troy Raeder, Rocio Alaiz-Rodriguez, Nitesh V. Chawla, and Francisco Herrera. 2012. A unifying view on dataset shift in classification. *Pattern Recognition* (2012).
- [35] Ali Pesaranghader and Herna L. Viktor. 2016. Fast Hoeffding Drift Detection Method for Evolving Data Streams. In *Machine Learning and Knowledge Discovery in Databases*.
- [36] Ali Pesaranghader, Herna L Viktor, and Eric Paquet. 2018. McDiarmid drift detection methods for evolving data streams. In *2018 International joint conference on neural networks (IJCNN)*. IEEE, 1–9.
- [37] Sergio Ramirez-Gallego, Bartosz Krawczyk, Salvador Garcia, Michal Wozniak, and Francisco Herrera. 2017. A survey on Data Preprocessing for Data Stream Mining: Current status and future directions. (2017).
- [38] Hidetoshi Shimodaira. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference* 90, 2 (2000), 227–244.
- [39] Yiliao Song, Jie Lu, Anjin Liu, Haiyan Lu, and Guangquan Zhang. 2021. A Segment-Based Drift Adaptation Method for Data Streams. *IEEE Transactions on Neural Networks and Learning Systems* (2021).
- [40] Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller. 2007. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research* 8, 5 (2007).
- [41] Yu Sun, Ke Tang, Zexuan Zhu, and Xin Yao. 2018. Concept drift adaptation by exploiting historical knowledge. *IEEE transactions on neural networks and learning systems* 29, 10 (2018), 4822–4832.
- [42] Abhijit Suprem, Joy Arulraj, Calton Pu, and Joao Ferreira. 2020. Odin: Automated drift detection and recovery in video analytics. *arXiv preprint arXiv:2009.05440* (2020).
- [43] Ashraf Tahmasbi, Ellango Jothimurugesan, Srikanta Tirthapura, and Phillip B. Gibbons. 2020. DriftSurf: A Risk-competitive Learning Algorithm under Concept Drift. [arXiv:2003.06508 \[cs.LG\]](https://arxiv.org/abs/2003.06508) <https://arxiv.org/abs/2003.06508>
- [44] Alexey Tsymbal. 2004. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin* 106, 2 (2004), 58.
- [45] Geoffrey I Webb, Roy Hyde, Hong Cao, Hai Long Nguyen, and Francois Petitjean. 2016. Characterizing concept drift. *Data Mining and Knowledge Discovery* 30, 4 (2016), 964–994.
- [46] Keisuke Yamazaki, Motoaki Kawanabe, Sumio Watanabe, Masashi Sugiyama, and Klaus-Robert Müller. 2007. Asymptotic bayesian generalization error when training and test distributions are different. In *Proceedings of the 24th international conference on Machine learning*. 1079–1086.
- [47] Peng Zhao, Le-Wen Cai, and Zhi-Hua Zhou. 2020. Handling concept drift via model reuse. *Machine Learning* (2020).
- [48] Indre Zliobaite. 2013. How good is the electricity benchmark for evaluating concept drift adaptation. *arXiv preprint arXiv:1301.3524* (2013).