

A Probabilistic Framework for Adapting to Changing and Recurring Concepts in Data Streams

Ben Halstead*, Yun Sing Koh*, Patricia Riddle*, Mykola Pechenizkiy[†], Albert Bifet[‡]

*School of Computer Science, University of Auckland, New Zealand

Email: bhal636@aucklanduni.ac.nz, ykoh@cs.auckland.ac.nz, pat@cs.auckland.ac.nz

[†]Eindhoven University of Technology, The Netherlands, Email: m.pechenizkiy@tue.nl

[‡]University of Waikato, Hamilton, New Zealand, and LTCI, Télécom Paris, IP-Paris, Email: abifet@waikato.ac.nz

arXiv:2408.09324v1 [cs.LG] 18 Aug 2024

Abstract—The distribution of streaming data often changes over time as conditions change, a phenomenon known as concept drift. Only a subset of previous experience, collected in similar conditions, is relevant to learning an accurate classifier for current data. Learning from irrelevant experience describing a different concept can degrade performance. A system learning from streaming data must identify which recent experience is irrelevant when conditions change and which past experience is relevant when concepts reoccur, *e.g.*, when weather events or financial patterns repeat. Existing streaming approaches either do not consider experience to change in relevance over time and thus cannot handle concept drift, or only consider the recency of experience and thus cannot handle recurring concepts, or only sparsely evaluate relevance and thus fail when concept drift is missed. To enable learning in changing conditions, we propose SELeCT, a probabilistic method for continuously evaluating the relevance of past experience. SELeCT maintains a distinct internal state for each concept, representing relevant experience with a unique classifier. We propose a Bayesian algorithm for estimating state relevance, combining the likelihood of drawing recent observations from a given state with a transition pattern prior based on the system’s current state. The current state is continuously maintained using a Hoeffding bound based algorithm, which unlike existing methods, guarantees that every observation is classified using the state estimated as the most relevant, while also maintaining temporal stability. We find SELeCT is able to choose experience relevant to ground truth concepts with recall and precision above 0.9, significantly outperforming existing methods and close to a theoretical optimum, leading to significantly higher accuracy and enabling new opportunities for learning in complex changing conditions.

Index Terms—Data Streams, Recurring Concepts

I. INTRODUCTION

A growing number of systems collect data in real-time, such as internet-connected sensors or online activity. Learning from these data streams requires adaptive systems capable of reacting to change. Consider the data stream classification task shown in Figure 1 where the air quality level at y is forecast from air pollution readings captured at W and E . New observations may arrive at a fast pace in volumes that may not fit in memory, requiring the distribution of data, the *concept*, to be learned incrementally by accumulating experience over time [1]. The distribution of data is determined by the *context* [2], unknown features which effect classification. For example, wind direction is an unknown factor determining how pollution affects air quality. While the wind direction is *constant* a classifier may accumulate

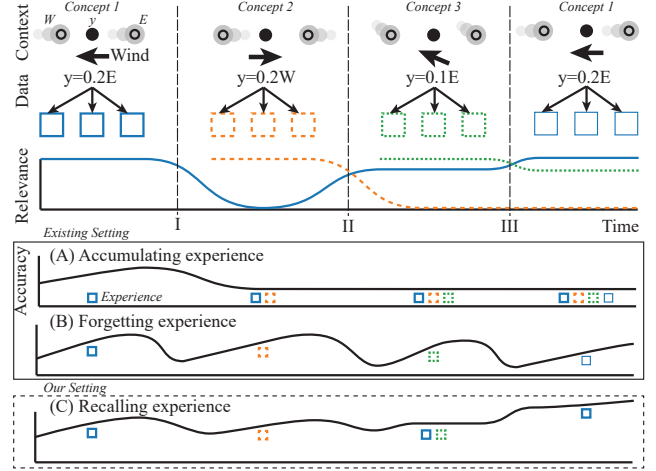


Fig. 1: Relevance of experience over concept drift, and the effect on accuracy over time when accumulating, forgetting and additionally recalling experience.

experience to learn a forecasting function which is accurate for the current concept. However, in a streaming setting context may change over time [2], causing *concept drift* [3] where the distribution of incoming data changes. Experience learned under one concept may not generalize to other concepts, *e.g.*, if wind direction changes then the forecasting function learned by the current classifier may *conflict* with the new distribution of data [4]. The task shown in Figure 1 has three distinct wind directions, with the east and west directions producing conflicting concepts. The accuracy of a system which only accumulates experience will degrade when concept drift occurs at point (I). Experience made irrelevant due to concept drift must be *forgotten* in order to maintain accuracy. Most existing methods of learning from streaming data focus on the two challenges of accumulating experience and forgetting irrelevant experience [5].

In this work we investigate the additional requirement of *recalling* forgotten experience when concepts reoccur, *e.g.*, when similar wind conditions appear in multiple discrete periods across a stream. Recalling relevant experience from previous occurrences of a concept is required for long-term accumulation of experience in the presence of concept drift,

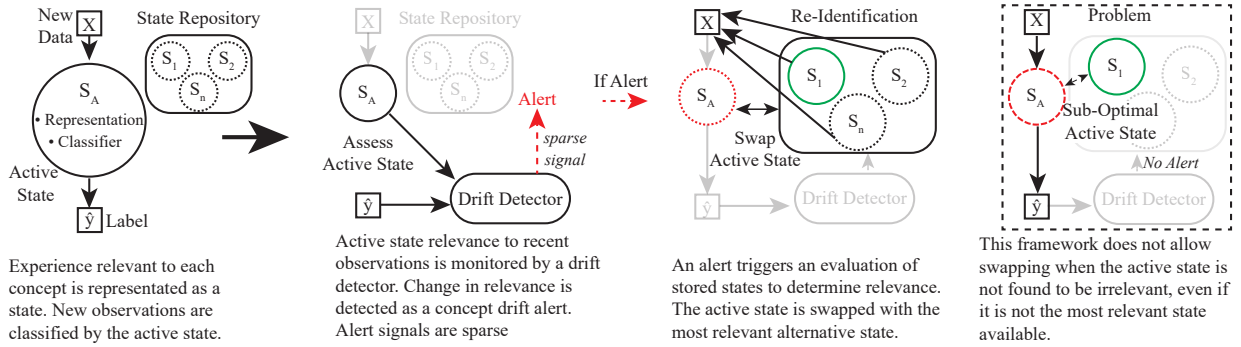


Fig. 2: Standard Adaptive Learning Framework

particularly for handling rare or short-term concepts where we cannot learn an accurate predictive function over one occurrence. In these cases recalling experience is difficult as it requires the conditions where experience is relevant to be accurately represented, however building such a representation across concept drift requires accurately recalling experience in the first place. Solving these dual problems requires a temporally stable system, capable of learning a stable initial representation of each concept. Learning from streaming data with changing and recurring conditions is an open research challenge, requiring *accumulating* experience over time, *forgetting* experience made irrelevant by concept drift, and *recalling* previous experience made relevant when concepts reoccur.

Existing streaming approaches fall into three broad categories, which all face challenges in this setting. Approaches that do not consider the relevance of experience, or do not consider changes in relevance over time, (A) in Figure 1, cannot adapt to concept drift and are not suitable in changing conditions. Approaches that determine relevance based only on forgetting irrelevant experience, (B) in Figure 1, encounter *catastrophic forgetting* where forgotten experience becomes relevant in the future, hindering the learning of long-term or recurring concepts. The final category, *adaptive learning* methods, monitor recent observations to determine changes in relevance over time to recall recurring concepts. However, in this work we identify a flaw in existing approaches in that the relevance of past experience is only evaluated *sparingly* when current experience is deemed irrelevant. Existing methods cannot guarantee that each prediction is made using relevant experience, failing at point (III) in Figure 1 where the most relevant experience cannot be recalled because current experience is not considered irrelevant.

In this work, we address these challenges by proposing a framework for adapting to concept drift able to guarantee that each prediction uses the experience estimated as most relevant, producing (C) in Figure 1. Similarly to adaptive learning, our approach accumulates experience relevant to each concept into a repository of internal *states* each represented by a distinct classifier. By selecting a relevant *active* state from the repository to classify and learn from each incoming observation, experience can be forgotten and recalled dynamically

in response to changing distributions.

In existing methods, the relevance of the active state is monitored by a *concept drift detector* [6, 7], producing an alert when changes in distribution occur, deactivating the active state to forget irrelevant experience and selecting a new active state to handle the new concept. A re-identification [8] procedure queries previously deactivated states to determine if any are relevant to the new concept, or, if no suitable state is found, the concept is deemed novel and a new active state is initialized. In this procedure, the relevance of the active state is continuously monitored, identifying when relevant experience becomes irrelevant, however the relevance of inactive states is only *sparingly* evaluated when drift is detected, which does not identify drift where irrelevant experience becomes relevant, as shown in Figure 2. Additionally, relevance in existing methods is often a *binary* signal due to separating detection and re-identification. We often cannot compare *how* relevant experiences are, causing failure cases when concept drift detection and re-identification conflict.

We propose an alternative approach to determining state relevance, **Streaming Extraction of Likelihoods for Concept Transitions**, (SELeCT), which integrates concept drift detection and re-identification into a single, probabilistic algorithm able to continuously evaluate how relevant each state is to current observations. We propose a Bayesian algorithm to compute the relevance of a state by combining the likelihood of drawing recent observations with a prior probability learned from previous concept transitions. To maintain the temporal stability of states, an important property for learning each distinct underlying concept, we propose a continuous selection algorithm based on the Hoeffding bound, giving a guarantee within some risk level that the selected state at any point in the stream has the highest estimated relevance to the underlying concept displayed in recent observations, even in noisy conditions. Our evaluation shows that the SELeCT can choose states that match ground truth concepts with a recall and precision above 0.9, showing no significant difference compared to a theoretically optimal selection strategy and significantly outperforming existing methods. By more accurately selecting relevant experience, we show that SELeCT is able to achieve a classification κ statistic up to 0.15 higher than existing methods. Our contributions are as follows:

- Our main contribution is proposing the first method of explicitly accumulating, forgetting and recalling experience relevant to changing conditions in a continuous, probabilistic manner, enabling us to learn recurring concepts.
- We develop: 1) a method of estimating relevance based on the likelihood of drawing recent observations and a prior probability encoding learned transition patterns. 2) A continuous selection algorithm based on the Hoeffding bound, able to guarantee, even in noisy real-world conditions, that the experience used for prediction is the best choice given our estimator, and, crucially, is temporally stable, enabling us to learn a representation of each concept. 3) A strategy for merging states with similar relevance dynamics to reduce memory costs.
- We contribute a novel adaptive learning framework, SELeCT, enabling near-optimal adaptations to changing conditions. We evaluate SELeCT in streaming tasks with recurring concepts using decision tree and neural network classifiers, demonstrating accuracy up to 43% higher than existing adaptive learning methods.

II. PROBLEM FORMULATION

We consider the supervised data stream classification problem, where a sequence of $\langle X, y \rangle$ observations is received over time. We consider labels to be immediately available, however our framework generalizes to the unsupervised setting if an unsupervised concept representation [8] is used. Each observation at time t , O^t , occurs in some *context*, H^t , the set of all unobserved features which affect y [9]. Each observation is drawn from the distribution $p(X, y|H^t)$, known as the *concept* at time t . As H^t is unobserved, we denote the concept at t as $C^t = p_t(X, y)$.

It is common for H^t to change over time, leading to observations at different points in a data stream being drawn from different concepts. A *concept drift* is a timestep t_D where $p_{t < t_D}(X, y) \neq p_{t > t_D}(X, y)$ [10]. A concept drift can be gradual, occurring over a period of observations, or abrupt [11]. Assuming that context changes smoothly over time, concepts are *temporally stable*, *i.e.*, a data stream can be viewed as a sequence of segments, each containing observations drawn from a distinct concept, separated by concept drift. Concepts can reoccur over the stream, *i.e.*, distinct segments may share a distribution. Classification in this setting is difficult, as experience describing one distribution may not accurately describe a different distribution. A classifier trained on experience from one concept may inaccurately predict observations drawn from another. We define experience, or a classifier trained on this experience, as *relevant* to a concept if it accurately describes the distribution of data drawn from that concept or *irrelevant* if it does not. To maximize accuracy we aim to learn each prediction from all *relevant* previous experience without *irrelevant* experience.

Adaptive learning approaches accumulate new experience as distinct *states*, $S_i = \langle m_i, \zeta_i \rangle$, defined in Definition 1, each representing a specific concept C_i . At time t , we have access to a *repository* R of previously constructed states, and we can

create a new *background* state B from a window of recent observations ω . The goal of adaptive learning is to select the *optimal* state from $R \cup B$ to classify each observation. Definition 2 defines optimal where ζ_i fully describes C_i and *sim* is a perfect measure of distribution similarity. In the real-world ζ_i is an approximation so our similarity measures are estimates, sim' , thus we find the *optimal estimated state*.

Definition 1: A **state** $S_i = \langle m_i, \zeta_i \rangle$ represents a concept C_i as a relevant classifier m_i , *i.e.*, trained on observations drawn from C_i , and a *concept representation*, ζ_i , summarizing the distribution C_i .

Definition 2: Given current concept C_i , repository R , background state B , a distribution similarity measure *sim*, estimated as sim' we define the **Optimal state** as $S_o = \arg \max_{S_j \in \{R \cup B\}} sim(\zeta_j, C_i)$ and the **Optimal Estimated state** as $S'_o = \arg \max_{S_j \in \{R \cup B\}} sim'(\zeta_j, C_i)$

An adaptive learning approach aims to, at each t , select the state S_A^t which best approximates the optimal state under a given sim' , $S_A^t = S'_o{}^t \approx S_o{}^t$. Identifying the optimal state for each observation achieves our problem formulation, assuming that experience is relevant to a specific concept. While learning, selecting the optimal state allows us to accumulate all experience relevant to each concept into a single state, containing no irrelevant experience, which we can recall to optimally predict future observations. Additionally, selecting optimal states allows the dynamics of S_A^t to be mined to reveal the dynamics of the hidden context H^t to, for example, reveal important features not currently included in X [12].

Existing adaptive learning methods, discussed in Section VI, select S_A^t by following the procedure shown in Figure 2. Initially, an *active* state S_A is constructed to learn from new observations. A concept drift detector monitors the relevance of the active state, producing an alert when significant change is detected in a similarity measure $sim'(\zeta_A, \zeta_\omega)$, where ζ_ω describes a window ω of recent observations. An alert indicates S_A is no longer relevant, *i.e.*, should be deactivated and stored in R , and a new S_A should be selected. A re-identification procedure can query R for previous $S_i \in R$ which have become relevant, *i.e.*, show a high $sim'(\zeta_i, \zeta_\omega)$. If no $S_i \in R$ is relevant, we determine the new concept is novel and initialize a new S_A . This approach only evaluates the relevance of $S_i \in R$ *sparsely*, when S_A is found to be irrelevant and an alert is triggered. While efficient, this approach cannot identify concept drift which presents as an increase in the relevance of a state in R , rather than a significant decrease in the relevance of S_A , *e.g.*, due to noise or partial changes. Sparse evaluation cannot be solved by simply running re-identification every step, as re-identification does not maintain a temporally stable active state, *i.e.*, S_A may change at any t due to noise. This behaviour does not reflect true change in context, and means we cannot accumulate experience of each distinct concept. We require an efficient method of continuously evaluating relevance that maintains a temporally stable active state.

In Section V, we empirically show that the active state in existing methods often fails to capture the ground truth concept, causing accuracy to be degraded. Alternative stream

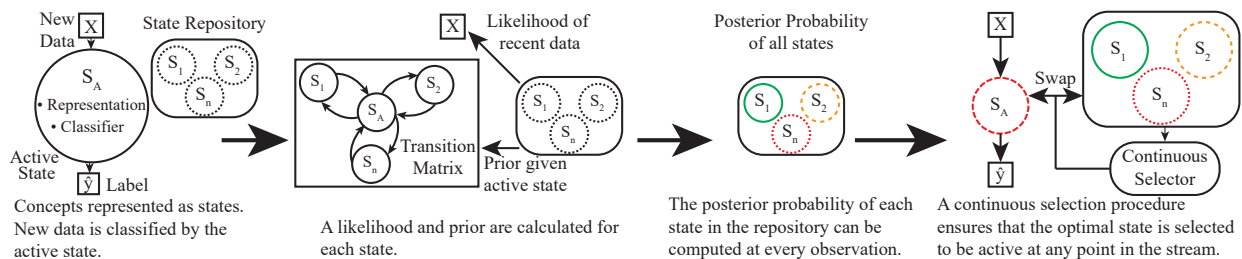


Fig. 3: SELeCT Framework

Algorithm 1 SELeCT

In: New observation O^t (Input X^t , Class y^t), State repository R , Background state B , Active state S_A^t

Out: Label \hat{y}^t , Next state S_A^{t+1}

- 1: $\hat{y}^t \leftarrow \text{Classify}(S_A^t, X^t)$.
- 2: $\text{Train}(S_A^t, y^t)$
- 3: $\text{StateProbabilities} \leftarrow []$
- 4: **for** $S_i \in R \cup B$ **do**
- 5: $K^t \leftarrow \text{CurrentKnowledge}(S_A^t, O^t)$
- 6: $p_i \leftarrow \text{ComputePrior}(K^t, S_i)$
- 7: $l_i \leftarrow \text{UpdateLikelihood}(O^t, S_i)$
- 8: $\text{StateProbabilities}[S_i] \leftarrow p_i \times l_i$
- 9: **end for**
- 10: $S_A^{t+1} \leftarrow \text{SelectionTest}(\text{StateProbabilities}, S_A^t)$
- 11: **if** $S_A^{t+1} = B$ **then**
- 12: $R \leftarrow R \cup B$, $B \leftarrow \text{NewState}()$
- 13: **end if**
- 14: **return** \hat{y}^t, S_A^{t+1}

learning approaches, such as dynamic selection and continual learning discussed in Section VI, cannot explicitly detect irrelevant experience, or store and recall relevant past experience. In the next section we propose SELeCT to accurately identify relevant experience in streaming data with changing and recurring concepts and show that we achieve classification performance and context tracking closer to optimal.

III. SELeCT FRAMEWORK OVERVIEW

SELeCT is an adaptive learning framework based around collecting a repository of states, each accumulating experience relevant to a distinct concept, and identifying the optimal state to handle each observation. The main novelty is that, rather than a cycle of continuous concept drift detection and sparse re-identification, SELeCT continuously evaluates the relevance of all states, enabling us to detect both when the active state becomes irrelevant and when an inactive state becomes relevant. A continuous selection method using the Hoeffding bound solves the temporal stability challenge, guaranteeing that every active state S_A^t is the optimal estimated state S_o^t on recent data, even in noisy conditions. By maintaining temporally stable active states we match the dynamics of the underlying concepts and can learn experience relevant to each distinct concept. SELeCT is an abstract framework, so

each component shown in Figure 3 and Algorithm 1 may be implemented in different ways. In this section we discuss the overall algorithm and in the next section we discuss component implementation.

State Representation SELeCT must accumulate experience as states, such that we can identify the conditions where each state is relevant. SELeCT models each state S_i as a pair, $\langle m_i, \zeta_i \rangle$, an incremental classifier [1] m_i and concept representation ζ_i . Experience can be accumulated by training m_i on new observations drawn from a given concept, and can be applied by making predictions using m_i . The *concept representation* ζ_i defines the distribution each state is relevant to using a similarity function sim' , e.g., cosine similarity (Eq. 2). We may calculate sim' between two representations to estimate the distance between the distributions they represent, and thus estimate relevance. Different concept representations may capture different information about a distribution, for example, a supervised representation capturing $p(y|X)$ may be used when labels are available, otherwise an unsupervised representation capturing only $p(X)$ may be required [8].

A concept representation ζ_i for a given state S_i is learned by maximizing similarity to observations seen when S_i is active. A *background* state B is maintained to represent a sliding window ω of the most recent observations. We discuss details of the state representation used in our implementation in the next section. At each time t , SELeCT computes the relevance of each state $S_i \in R \cup B$, to select S_A^t for the next observation using a probabilistic Bayesian approach.

State Probability To identify the optimal active state, at each time t , SELeCT computes the probability of each state $S_i \in R \cup B$ being optimal at the next time step, $p(S_o^{t+1} = S_i)$, given current knowledge of the system, K^t . General or domain specific knowledge describing concept drift characteristics can be captured in K^t , i.e., transition patterns. Given K^t and the current observation $\langle X^t, y^t \rangle$, the goal is to compute the probability $p(S_o^{t+1} = S_i | K^t, \langle X^t, y^t \rangle)$. SELeCT uses a Bayesian approach to break this into two sub-steps, computing the prior probability p_i of S_i being active on the next observation given current knowledge, $p_i = p(S_o^{t+1} = S_i | K^t)$, and computing the likelihood l_i of the current observation being generated from the concept described by S_i , $l_i = p(\langle X^t, y^t \rangle | S^{t+1} = S_i)$. Bayes Theorem can combine these into a posterior probability:

$$p(S_o^{t+1} | K^t, \langle X^t, y^t \rangle) \sim p(\langle X^t, y^t \rangle | S^{t+1}) p(S_o^{t+1} | K^t) \quad (1)$$

Continuous Selection Finally, SELeCT uses a continuous selection test to select the next active state based on the computed posterior probabilities of each state. A simple *maximum a posteriori* approach, selecting the state with maximum posterior probability, suffers from issues with temporal stability, where noise may cause an erroneous transition to, or a failure to transition away from, a sub-optimal active state. Both cases make it difficult to learn a distinct state for each concept as they disrupt our learning of ζ_A , meaning S_A cannot be accurately recalled in the future. We instead use a hypothesis test, with the null hypothesis that the current active state had a higher probability over recent observations than any alternative state, giving a guarantee, up to some risk level, that the active state at any given observation is the optimal achievable state for a given sim' from the current set of possible states.

Theoretical Analysis: Time and Memory Complexity

While the time complexity of SELeCT depends on the implementation of each component, we may assume that components follow standard online learning restrictions of constant time and memory complexity per observation. In this case, SELeCT has the same time complexity as the standard adaptive learning framework. The active state classification and training steps are $O(1)$ per observation. Computing state probability can be assumed to be $O(1)$ per observation per stored state if implemented with a fixed amount of memory. Across a state repository of size, $|R|$, SELeCT is then $O(|R|)$ per observation. Repository size can be fixed to a constant $|R|$ while retaining performance by implementing standard adaptive learning memory management techniques [13, 14], which we omit for simplicity. Under these implementation assumptions, the SELeCT framework has a constant time complexity per observation making it suitable for online use.

For comparison, the standard framework replaces the probability calculation with a drift detection step and runs a re-identification step if an alert is triggered. Given a d chance of triggering a drift per observation, the time complexity of the standard framework is $O(d|R|)$ per observation. This is also constant time per observation under the same implementation constraints. SELeCT has the same constant memory complexity as the standard framework given a fixed size repository, storing $|R|$ states in the repository and one background state, $O(|R|+1)$. We empirically validate that the time and memory performance of our implementation of SELeCT is equivalent to existing methods, discussed in the supplementary materials available at <https://github.com/BenHals/SELeCT>.

IV. COMPONENT IMPLEMENTATION

In this section, we propose methods for each component of the SELeCT framework. SELeCT requires three main components, a method of representing a concept as a system state, a method of computing state priors and likelihoods, and a continuous selection statistical test. All code is available at <https://github.com/BenHals/SELeCT>.

System State We base our state representation on the representation proposed in FiCSUM [8], where each ζ_i describes the distribution of a concept C_i using a vector of meta-features.

Each *meta-feature* in ζ_i is the result of a summary function calculated over a set of observations drawn from C_i , for example, the mean of X or the variance in y , describing one aspect of how a concept behaves over time. We use the default set of meta-feature functions from FiCSUM to represent a general range of concept behaviours. We use a weighted cosine distance, with meta-feature weights W calculated as in FiCSUM [8], to calculate the similarity between ζ_i , approximating the difference between distributions C_i and C_j as

$$sim'(\zeta_i, \zeta_j) = \frac{W\zeta_i \cdot W\zeta_j}{\|W\zeta_i\| \cdot \|W\zeta_j\|}. \quad (2)$$

To train the active representation ζ_A , we monitor a sliding window of recent observations, ω , and a lagged sliding window B . The representation ζ_B built on B is a stable representation of the active concept, as long as we have not detected concept drift since it was captured. We update the active ζ_A as the online mean of ζ_B , captured every $|B|$ steps.

State Priors We calculate the prior probability of a state S_j at time $t + 1$, *i.e.*, the probability $p(S_o^{t+1} = S_j|K^t)$, using a transition matrix based K^t which is informed by a concept drift detector. Given the active state at the current time step, S_A^t , the probability of S_j being the next optimal state S_o^{t+1} is given by the probability $p(S_o^{t+1} = S_j|S_A^t, D^t)$ of observing a transition from S_A^t to S_j in current conditions, *i.e.*, when recent observations are stationary or drifting. We model current conditions as D^t using a drift detector, setting $D^t=1$ in periods where a drift detector detects significant changes in the likelihood of S_A^t , or $D^t=0$ otherwise. We calculate $p(S_o^{t+1} = S_j|S_A^t, D^t)$ using two transition matrices, TM^1 and TM^0 which capture transitions when D^t is 1 or 0 respectively. Each entry TM_{ij}^d represents the number of transitions seen from $S_A^t=S_i$ to S_j when $D^t=d$. We calculate the prior probability for each $S_j \in R$ as

$$p(S_o^{t+1} = S_j|S_A^t = S_i, D^t = d) = \frac{TM_{ij}^d}{\sum_{S_k \in R} TM_{ik}^d}. \quad (3)$$

We also use three adjustment parameters. Firstly, a minimum prior probability allows unobserved transitions to occur. Secondly, a ‘backwards transition’ prior of strength b allows a new state to transition back to the previous state to repair false positive transitions, implemented by setting $TM_{ij}^d = b$ when we first observe a transition to state j . Finally, we track transitions of up to m steps, *e.g.*, if transitions A to B and B to C have been observed, we assign some prior probability to the 2-step transition A to C. Probability for m steps is given by $(TM^d)^m$. We calculate the transition count as $max(TM^d, (TM^d)^2, \dots, (TM^d)^m)$. The SELeCT framework enables further information to be incorporated into K^t , *e.g.*, stream volatility, which has been used to proactively predict concept drift locations [15], could be used to update the prior probability of S_A^t .

State Likelihood We calculate relevance as the likelihood of drawing a sliding window of recently observed data, ω , from the concept described by each system state as $p(\omega|S^t = S_j)$. We describe ω by computing the state representation ζ_ω . A

TABLE I: Performance against Baselines as Mean \pm Std. The best non-upper bound system in each dataset (row) is bolded.

		FiCSUM	RCD	CPF	DWM	DYNSE	SELeCT	LB	UB
κ	AQS	0.92 \pm 0.04	0.72 \pm 0.03	0.92 \pm 0.03	0.90 \pm 0.01	0.79 \pm 0.02	0.94 \pm 0.00	0.71 \pm 0.03	0.95 \pm 0.00
	AQT	0.44 \pm 0.06	0.39 \pm 0.04	0.50 \pm 0.03	0.57 \pm 0.01	0.47 \pm 0.04	0.57 \pm 0.05	0.33 \pm 0.02	0.61 \pm 0.01
	AD	0.85 \pm 0.03	0.71 \pm 0.08	0.87 \pm 0.03	0.85 \pm 0.03	0.45 \pm 0.16	0.85 \pm 0.02	0.82 \pm 0.04	0.90 \pm 0.01
	CMC	0.22 \pm 0.05	0.16 \pm 0.02	0.19 \pm 0.03	0.22 \pm 0.02	0.19 \pm 0.03	0.25 \pm 0.03	0.24 \pm 0.02	0.27 \pm 0.02
	STGR	0.98 \pm 0.02	0.91 \pm 0.07	0.87 \pm 0.04	0.92 \pm 0.01	0.77 \pm 0.01	0.98 \pm 0.00	0.93 \pm 0.00	0.98 \pm 0.00
	TREE	0.35 \pm 0.10	0.23 \pm 0.03	0.27 \pm 0.05	0.30 \pm 0.05	0.34 \pm 0.03	0.50 \pm 0.06	0.25 \pm 0.03	0.56 \pm 0.05
	WIND	0.90 \pm 0.01	0.70 \pm 0.07	0.78 \pm 0.02	0.86 \pm 0.00	0.58 \pm 0.00	0.92 \pm 0.01	0.89 \pm 0.03	0.94 \pm 0.00
	C-F1	AQS	0.76 \pm 0.10	0.29 \pm 0.02	0.51 \pm 0.07	0.29 \pm 0.00	0.29 \pm 0.04	0.87 \pm 0.06	0.29 \pm 0.00
	AQT	0.63 \pm 0.10	0.26 \pm 0.04	0.46 \pm 0.05	0.29 \pm 0.00	0.22 \pm 0.03	0.89 \pm 0.04	0.29 \pm 0.00	0.97 \pm 0.00
	AD	0.82 \pm 0.09	0.32 \pm 0.02	0.67 \pm 0.09	0.29 \pm 0.00	0.45 \pm 0.01	0.91 \pm 0.03	0.29 \pm 0.00	0.88 \pm 0.00
	CMC	0.73 \pm 0.12	0.47 \pm 0.07	0.51 \pm 0.06	0.67 \pm 0.00	0.57 \pm 0.05	0.84 \pm 0.07	0.67 \pm 0.00	0.88 \pm 0.00
	STGR	0.95 \pm 0.06	0.46 \pm 0.03	0.60 \pm 0.12	0.50 \pm 0.00	0.33 \pm 0.03	0.98 \pm 0.02	0.50 \pm 0.00	0.98 \pm 0.00
	TREE	0.59 \pm 0.19	0.31 \pm 0.03	0.43 \pm 0.03	0.29 \pm 0.00	0.27 \pm 0.02	0.94 \pm 0.06	0.29 \pm 0.00	0.98 \pm 0.00
	WIND	0.98 \pm 0.00	0.35 \pm 0.04	0.32 \pm 0.05	0.40 \pm 0.00	0.29 \pm 0.02	0.99 \pm 0.01	0.40 \pm 0.00	0.98 \pm 0.00

similarity a_j can be calculated between each state representation ζ_j for $S_j \in R$ and ζ_ω as $sim'(\zeta_j, \zeta_\omega)$, measuring the similarity between the experience relevant to S_j and the distribution generating incoming data. We calculate the probability of observing a similarity as high as a_j under the null hypothesis that S_j is relevant. While each state S_j is active, we model the distribution of a_j as a Gaussian, $A_j \sim N(\mu_j, \sigma_j)$ by capturing the mean and standard deviation of a_j as μ_j and σ_j . A_j models the distribution of similarity values we would expect to see if S_j was relevant. We compute the likelihood of drawing a given a_j from A_j , to get the likelihood of seeing a ω with a similarity to S_j of at least a_j , if S_j is relevant. Using Equation 1, we combine this likelihood with the state prior to get the probability $p(S_o^{t+1}|S^t, \omega)$ that each state is relevant.

Continuous Selection To select the the active state for an incoming observation given $p(S_o^{t+1}|S^t, \omega)$, we compare the probability of S_A^t against B and all states $S_i \in R$. We use a statistical test to guarantee that the selected S_A^{t+1} has the highest probability over a window of recent timesteps, in order to maintain temporal stability even in noisy conditions. We test a window of the w_0 most recent posterior probabilities for the active state, which has mean μ_0 , against a window of the w_1 most recent probabilities for each alternative state, which has mean μ_1 . Similarly to the ADWIN [16] concept drift detector, we select each window as the set of recent probabilities with no significant change in mean. The difference in means between the active and alternative states, $\mu_1 - \mu_0$, is tested against a threshold ϵ to test whether the alternative is significantly more relevant than the current active state. We use the Hoeffding bound to select ϵ based on the size of w_0 and w_1 in order to constrain the false positive and negative error rate for this test to be a parameter δ [16]. We transition to the alternative state with the highest μ_1 at least ϵ above μ_0 . Any conflict between alternative states is resolved in the next evaluation. If we transition to B , it is added to the repository, and a new B is initialized to represent ω .

State Merge Over time, the repository can accumulate duplicate states representing the same concept [17]. We remove

duplicates by calculating correlations between state probabilities and merge states with a correlation above a threshold parameter of 0.95 by combining entries in the transition matrix and removing the state with less training from the repository.

V. EVALUATION

Our hypothesis is that by selecting more relevant active states, SELeCT is able to more accurately accumulate and apply experience matching the underlying concept of a stream, enabling increased accuracy. We evaluate this hypothesis in this section, comparing SELeCT against an optimal reference and alternative streaming methods, and studying the effect of each of SELeCT’s components.

Evaluation measures We evaluate two aspects of learning in changing conditions: classification accuracy using the kappa statistic, κ , and context tracking performance using the C-F1 measure [8]. C-F1 measures the relevance of the experience chosen by a system to the ground truth concept at each point in a stream, encompassing standard measures like drift detection delay, false alarms, as well as re-identification performance. For each state s and concept c , we calculate the recall and precision of timesteps when c is active against the timesteps where s is active, measuring how much of the experience accumulated by s describes a c , and how often did the system identify s as relevant when c was present. C-F1 ranges from 0 to 1, reporting the average F1 score of the best matching for each c as an overall measure of how well the experience captured by a system matched underlying concepts.

Datasets We require datasets with known ground truth concepts to measure the use of relevant experience. We use four real-world datasets [18] with known concepts. The *Aedes* (AQS, AQT) datasets classify insect characteristics, with six different temperature ranges as concepts. *Arabic-Digets* (AD) classifies a speaker based on speech patterns, with six different digits as concepts. CMC classifies a survey result, with two age ranges as concepts. We use three synthetic datasets to investigate specific settings where SELeCT is beneficial, STAGGER (STGR) and RandomTree (TREE) are benchmarks from previous literature, and we provide a new synthetic data

TABLE II: Performance varying drift width, noise and transition noise. Mean \pm Std, best non upper bound method is bolded.

Data	Classifier	Drift Width			Noise			Transition Noise			
		0	500	2500	0.00	0.10	0.25	0.00	0.10	0.25	
κ	TREE	Upper	0.63 \pm 0.04	0.62 \pm 0.04	0.56 \pm 0.04	0.63 \pm 0.04	0.53 \pm 0.04	0.40 \pm 0.03	0.63 \pm 0.04	0.63 \pm 0.05	0.63 \pm 0.05
		FiCSUM	0.37 \pm 0.11	0.40 \pm 0.11	0.26 \pm 0.05	0.37 \pm 0.11	0.32 \pm 0.10	0.25 \pm 0.07	0.37 \pm 0.11	0.41 \pm 0.11	0.41 \pm 0.11
		SELeCT	0.55\pm0.06	0.53\pm0.07	0.45\pm0.05	0.55\pm0.07	0.45\pm0.08	0.35\pm0.04	0.55\pm0.07	0.55\pm0.06	0.54\pm0.07
	AQS	Upper	0.95 \pm 0.00	0.95 \pm 0.00	0.90 \pm 0.01	0.95 \pm 0.00	0.84 \pm 0.00	0.70 \pm 0.00	0.95 \pm 0.00	0.95 \pm 0.00	0.95 \pm 0.00
		FiCSUM	0.93 \pm 0.05	0.90 \pm 0.05	0.76 \pm 0.06	0.93 \pm 0.05	0.76 \pm 0.08	0.59 \pm 0.07	0.93 \pm 0.05	0.94\pm0.02	0.93 \pm 0.06
		SELeCT	0.94\pm0.01	0.93\pm0.01	0.87\pm0.02	0.94\pm0.00	0.82\pm0.05	0.68\pm0.02	0.94\pm0.00	0.94\pm0.01	0.94\pm0.01
C-F1	TREE	Upper	0.98 \pm 0.00	0.98 \pm 0.00	0.98 \pm 0.00	0.98 \pm 0.00	0.98 \pm 0.00	0.98 \pm 0.00	0.98 \pm 0.00	0.98 \pm 0.00	0.98 \pm 0.00
		FiCSUM	0.59 \pm 0.19	0.64 \pm 0.18	0.40 \pm 0.11	0.59 \pm 0.19	0.60 \pm 0.20	0.62 \pm 0.19	0.59 \pm 0.19	0.61 \pm 0.19	0.59 \pm 0.17
		SELeCT	0.95\pm0.04	0.93\pm0.05	0.87\pm0.04	0.94\pm0.05	0.93\pm0.12	0.94\pm0.04	0.94\pm0.05	0.94\pm0.06	0.93\pm0.08
	AQS	Upper	0.97 \pm 0.00	0.97 \pm 0.00	0.97 \pm 0.00	0.97 \pm 0.00	0.97 \pm 0.00	0.97 \pm 0.00	0.97 \pm 0.00	0.97 \pm 0.00	0.97 \pm 0.00
		FiCSUM	0.78 \pm 0.09	0.72 \pm 0.09	0.56 \pm 0.09	0.78 \pm 0.09	0.73 \pm 0.10	0.72 \pm 0.10	0.78 \pm 0.09	0.79 \pm 0.06	0.78 \pm 0.08
		SELeCT	0.88\pm0.05	0.85\pm0.05	0.74\pm0.05	0.88\pm0.07	0.86\pm0.08	0.86\pm0.05	0.88\pm0.07	0.87\pm0.06	0.88\pm0.05

generator, WIND, based on a simulated air quality prediction task. For all datasets we construct streaming datasets by repeating each concept in the dataset three times in an order generated using an underlying transition pattern, which is varied as described in the following experiments, to explore different scenarios. We evaluate abrupt and gradual concept drift, and report the mean \pm standard deviation over 45 seeds.

Experiment Setup We implement SELeCT using Hoeffding Tree [1] and ADWIN [16] as the base classifier and drift detector, using default parameters in Scikit-Multiflow [19]. We also study a neural network base classifier, implemented in PyTorch. We use the default state representation proposed in FiCSUM, making FiCSUM a comparable standard framework baseline. Parameters for SELeCT were chosen using a linear sensitivity analysis on the *AD* and *CMC* datasets. Datasets and parameters are described in supplementary material.

A. Performance evaluation

We first evaluate SELeCT against existing methods. Table I shows the κ and C-F1 of SELeCT against FiCSUM, CPF, DWM and DYNSE described in Section VI. We discuss the selection of baselines in the supplementary. *UB* reports the performance of a Hoeffding Tree classifier with perfect re-identification and drift detection with a fixed delay of 100 observations to represent the performance upper bound, *i.e.*, selecting the optimal state for each observation. *LB* reports the performance of a non-adaptive Hoeffding Tree classifier to represent the lower bound of adaptive learning performance.

SELeCT achieves a C-F1 at least 80% of the upper bound. In a Nemenyi significance test, shown in the supplementary material, SELeCT C-F1 performance is not significantly lower than the upper bound, and outperforms all competitors. SELeCT achieves a C-F1 of 0.94 in TREE, 63% higher than FiCSUM’s 0.59. This result validates our hypothesis that SELeCT can select the optimal achievable state. SELeCT achieves a higher κ value than all baselines, except CPF in AD. SELeCT achieves a κ of 0.50 in TREE, 43% higher than FiCSUM’s 0.35, indicating that selecting the optimal estimated state does provide increased classification performance. We observe that

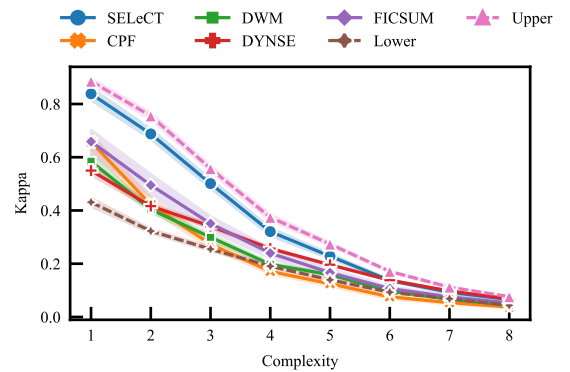


Fig. 4: Performance at increasing complexity.

higher C-F1 does not always translate into significant accuracy improvements, which we explore in the next experiment.

Concept Complexity Experiment In some datasets, recalling relevant experience does not substantially increase accuracy compared to collecting new experience, *e.g.*, if concepts are simple to learn. We hypothesize that the better active state selection shown by SELeCT may not translate into significantly higher classification performance in these cases. Evidence for this comes from the small difference between *LB* and *UB* in some datasets, indicating that even perfect adaptive learning does not improve classification performance. This experiment investigates this effect, reporting the performance of SELeCT against baselines under increasing concept *complexity*. Using the TREE dataset, where $p(y|X)$ for each concept is given by a randomly generated decision tree, we can increase the depth of these trees to create more complex distributions which take longer to learn. Figure 4 shows that as complexity increases, the classification performance of all methods falls. However, SELeCT retains a performance closer to *UB*, especially when adaptive learning is more beneficial, *i.e.*, there is a larger gap between *UB* and *LB*.

Gradual Drift and Noise Experiments In this experiment, we investigate gradual drift and noise, two cases where we

hypothesize the standard framework may fail. We test against FiCSUM as a baseline. Under gradual drift, concept drift occurs over a long period which may be missed by drift detection in the standard framework. Under noise, drift detection may also be less sensitive, as changes due to drift are obscured by changes due to noise. Table II reports performance at increasing concept drift width and with increasing amounts of uniform class noise, on two representative datasets. Both scenarios show a failure case for the standard framework, with the performance of FiCSUM dropping significantly, while SELeCT succeeds in maintaining performance close to the upper bound.

B. Component Evaluation

Table I shows an ablation study comparing the reference SELeCT implementation against variants.

Prior Probability Our prior probability component assumes that concept transitions follow an underlying transition pattern. We evaluate robustness to this assumption in Table II, which shows performance with increasing noise levels in underlying concept transitions. SELeCT shows no substantial changes in performance relative to the upper bound. These results indicate that our prior probability component is robust to the true underlying distribution of concepts. Table III shows that a variant of SELeCT, S_p , using an uninformative, uniform prior, is outperformed by our implementation, indicating our prior component is beneficial.

Continuous Selection To test the effect of continuous selection, Table III shows the performance of a variant of SELeCT, S_{MAP} , using a naïve MAP selection procedure which selects the most probable state at each step rather than using our Hoeffding bound test. SELeCT outperforms this variant, indicating our continuous selection component is beneficial and validating our hypothesis that using the Hoeffding bound to select active states can improve performance. Crucially, we observe a significantly lower C-F1, indicating temporal instability in the selection of active states.

State Merging Table III shows that a variant of SELeCT, S_m , which does not merge states achieves lower performance, especially in C-F1, indicating that by removing redundant representations we can better model underlying context.

C. Continual Learning Comparison

We evaluate SELeCT against CNDPM [20], an expansion based neural network continual learning method able to allocate new parameters to accumulate experience from new tasks. CNDPM can identify which task is relevant to each observation, but does not consider task relevance to change over time, *i.e.*, cannot forget irrelevant tasks or recall different tasks for the same input. CNDPM includes a prior probability based on the number of observations drawn from each task. In a streaming setting, we find this prior undervalues new concepts relative to existing concepts, making them temporally unstable and difficult to learn. We study two variants of CNDPM, with and without the prior, as $CNDPM_p$ and $CNDPM_{np}$. To control for the effect of using a neural network

rather than a Hoeffding tree, we study variants of SELeCT and the upper bound classifier, denoted S_{cndpm} and UB_{cndpm} respectively, which use the CNDPM base neural network to learn each state. We use the same default hyper-parameters for all networks. Due to the reduced efficiency of the neural network compared to the base Hoeffding tree, we performed 20 runs of each experiment rather than 45. Table III reports the mean and standard deviation.

We observe that in both SELeCT and UB the neural network base classifier reduces performance, however, in both cases, an adaptive learning approach is still able to learn, highlighting the generality of our framework. We observe that removing the prior increases the performance of CNDPM in most cases, however in all cases performs significantly worse than SELeCT, even using the same base classifier. This result verifies our hypothesis that existing methods cannot adequately handle changes and recurrences in the relevance of experience over time, leading to degraded performance in a streaming environment. In contrast, SELeCT is, in many cases, within one standard deviation of an upper-bound system.

VI. RELATED WORK

Our problem formulation shares similarities to dynamic classifier selection [21, 22] and continual learning [5]. Dynamic selection considers the joint distribution of data to be partitioned on X into *regions of competence* specialized in handling distinct inputs [23, 24], *i.e.*, given a region, X , relevant experience describes $p(y|X)$. Here context H^t determines the region data is drawn from, $p(X|H^t)$, but the relevance of experience within a region, $p(y|X)$, is constant, *i.e.*, forgetting is not required as real concept drift [10] is not considered. DYNSE [22] considers concept drift, estimating recent experience as more relevant to handle gradual change in $p(y|X)$ over time, but not recurrences.

The aim of Continual learning is to learn new tasks without forgetting information relevant to past tasks [5]. A major research focus is avoiding catastrophic forgetting. Many continual learning approaches are not practical for streaming tasks as they assume that a task ID encoding relevant experience is known for each observation. Recent *task-free* continual learning approaches [20] can identify experience relevant to a task. However, similarly to dynamic selection, continual learning determines the relevance of past experience to X , rather than to the current concept. While the distributions of observations, $p(X)$, is considered to change over time as observations arrive from different tasks, each specific observation is associated with the same task over the entire stream, *i.e.*, $p(y|X)$ is constant. In contrast, under real concept drift $p(y|X)$ changes over time, *i.e.*, a particular observation is associated with different concepts at different points in time and must be predicted using different experience. Some recent methods update experience over time [20, 25], implicitly forgetting experience to adapt to gradual concept drift. However, no existing continual learning methods can explicitly identify irrelevant experience or store and recall relevant experience to learn recurring concepts. In Section V, we find that current

TABLE III: Ablation Study, reporting mean \pm std. Best non upper-bound method bolded for each set of runs.

		45 Runs				20 Runs					
		SELeCT	S_p	S_{MAP}	S_m	SELeCT	S_{cndpm}^*	CNDPM $_{np}^*$	CNDPM $_p^*$	UB	UB $_{cndpm}^*$
κ	AQS	0.94 \pm 0.00	0.94 \pm 0.00	0.91 \pm 0.06	0.94 \pm 0.00	0.94 \pm 0.00	0.66 \pm 0.15	0.20 \pm 0.20	0.11 \pm 0.07	0.95 \pm 0.00	0.91 \pm 0.01
	AQT	0.57 \pm 0.05	0.55 \pm 0.06	0.44 \pm 0.11	0.56 \pm 0.04	0.53 \pm 0.07	0.47 \pm 0.08	0.30 \pm 0.23	0.03 \pm 0.01	0.61 \pm 0.01	0.60 \pm 0.00
	AD	0.85 \pm 0.02	0.85 \pm 0.02	0.81 \pm 0.05	0.85 \pm 0.02	0.86 \pm 0.01	0.68 \pm 0.11	0.57 \pm 0.12	0.56 \pm 0.18	0.90 \pm 0.01	0.89 \pm 0.02
	CMC	0.25 \pm 0.03	0.23 \pm 0.04	0.23 \pm 0.05	0.25 \pm 0.03	0.24 \pm 0.03	0.17 \pm 0.05	0.04 \pm 0.03	0.10 \pm 0.04	0.27 \pm 0.02	0.26 \pm 0.02
	STGR	0.98 \pm 0.00	0.98 \pm 0.01	0.92 \pm 0.08	0.98 \pm 0.00	0.98 \pm 0.00	0.87 \pm 0.09	0.13 \pm 0.07	0.47 \pm 0.12	0.98 \pm 0.00	0.95 \pm 0.00
	TREE	0.50 \pm 0.06	0.49 \pm 0.06	0.41 \pm 0.11	0.50 \pm 0.06	0.51 \pm 0.06	0.31 \pm 0.05	0.23 \pm 0.07	0.10 \pm 0.02	0.58 \pm 0.03	0.38 \pm 0.02
	WIND	0.92 \pm 0.01	0.92 \pm 0.01	0.89 \pm 0.03	0.92 \pm 0.01	0.92 \pm 0.01	0.54 \pm 0.05	-0.00 \pm 0.00	0.00 \pm 0.00	0.94 \pm 0.00	0.62 \pm 0.00
C-FI	AQS	0.87 \pm 0.06	0.87 \pm 0.04	0.79 \pm 0.10	0.86 \pm 0.06	0.85 \pm 0.06	0.77 \pm 0.11	0.28 \pm 0.00	0.29 \pm 0.00	0.97 \pm 0.00	0.97 \pm 0.00
	AQT	0.89 \pm 0.04	0.87 \pm 0.07	0.71 \pm 0.12	0.88 \pm 0.04	0.88 \pm 0.03	0.89 \pm 0.07	0.28 \pm 0.01	0.29 \pm 0.00	0.97 \pm 0.00	0.97 \pm 0.00
	AD	0.91 \pm 0.03	0.89 \pm 0.04	0.85 \pm 0.07	0.88 \pm 0.03	0.91 \pm 0.01	0.88 \pm 0.03	0.28 \pm 0.01	0.29 \pm 0.00	0.88 \pm 0.00	0.88 \pm 0.00
	CMC	0.84 \pm 0.07	0.76 \pm 0.11	0.82 \pm 0.09	0.80 \pm 0.07	0.83 \pm 0.08	0.84 \pm 0.08	0.62 \pm 0.01	0.67 \pm 0.00	0.88 \pm 0.00	0.88 \pm 0.00
	STGR	0.98 \pm 0.02	0.98 \pm 0.02	0.87 \pm 0.13	0.97 \pm 0.02	0.98 \pm 0.01	0.92 \pm 0.09	0.37 \pm 0.03	0.51 \pm 0.03	0.98 \pm 0.00	0.98 \pm 0.00
	TREE	0.94 \pm 0.06	0.94 \pm 0.06	0.78 \pm 0.17	0.93 \pm 0.05	0.94 \pm 0.07	0.96 \pm 0.02	0.26 \pm 0.01	0.29 \pm 0.00	0.98 \pm 0.00	0.98 \pm 0.00
	WIND	0.99 \pm 0.01	0.98 \pm 0.02	0.95 \pm 0.07	0.97 \pm 0.01	0.99 \pm 0.01	0.99 \pm 0.01	0.40 \pm 0.00	0.40 \pm 0.00	0.98 \pm 0.00	0.98 \pm 0.00

continual learning methods struggle to learn in a streaming setting with recurring concepts. We note that our setting is different from modeling recurrent change points [26].

The adaptive learning framework described in Section II has been used in many approaches, usually with new methods of representing the active state and monitoring changes in its relevance. RCD [27], uses an accuracy based similarity measure to monitor the relevance of the active state continuously, while a distribution similarity test is used during re-identification to evaluate the relevance of stored states. JIT [28] and CPF [29] monitor the active state using error rate and feature distribution, while relevance for re-identification is based on hypothesis testing and classifier equivalence [2]. GraphPool [17] additionally uses a concept transition matrix when evaluating the relevance of stored states. [30] and [31] use a similar transition matrix approach to identify recurring concepts. FiCSUM [8] uses a vector similarity measure to monitor the active state continuously and sparsely evaluate stored states. [2] propose using a secondary meta-learning classifier trained to predict state relevance. DWM [32] and ARF [33] use an ensemble active state, using error rate to monitor the relevance of each member [34, 35], but have no mechanism to recall forgotten experience once a given classifier is dropped from the ensemble.

VII. CONCLUSION

Adaptive learning provides a framework for data stream classification in the presence of concept drift by identifying and reusing previous experience relevant to current conditions. However, existing methods encounter common failure cases where sub-optimal experience is selected due to sparse and binary evaluation, hindering our ability to learn from streaming data in changing conditions. We propose SELeCT, a probabilistic framework that is able to avoid these failure cases by continuously evaluating the relevance of all states to select the optimal state for each observation. Our evaluation shows the relevance of the experience chosen by SELeCT to ground truth concepts is comparable to a perfect knowledge baseline, and up to 60% higher than five baseline methods, enabling us to learn in new scenarios featuring complex changing and

recurring conditions. Experiments varying data complexity, noise, and drift width show more accurate identification of relevant experience allows SELeCT to achieve a κ statistic up to 43% higher than alternative systems.

ACKNOWLEDGMENT

The work was supported by the Marsden Fund Council from New Zealand Government funding (Project ID 18-UOA-005), managed by Royal Society Te Apārangi.

REFERENCES

- [1] J. Gama, R. Rocha, and P. Medas, “Accurate decision trees for mining high-speed data streams,” in *Proceedings of the SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003, pp. 523–528.
- [2] J. Gama and P. Kosina, “Recurrent concepts in data streams classification,” *Knowledge and Information Systems*, vol. 40, no. 3, pp. 489–507, 2014.
- [3] A. Tsymbal, “The problem of concept drift: definitions and related work,” *Computer Science Department, Trinity College Dublin*, vol. 106, no. 2, p. 58, 2004.
- [4] S. Kessler, J. Parker-Holder, P. Ball, S. Zohren, and S. J. Roberts, “Same state, different task: Continual reinforcement learning without interference,” *arXiv preprint arXiv:2106.02940*, 2021.
- [5] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, “Continual lifelong learning with neural networks: A review,” *Neural Networks*, vol. 113, pp. 54–71, 2019.
- [6] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, “Learning with drift detection,” in *Brazilian Symposium on Artificial Intelligence*. Springer, 2004, pp. 286–295.
- [7] R. S. M. Barros and S. G. T. C. Santos, “A large-scale comparison of concept drift detectors,” *Information Sciences*, vol. 451, pp. 348–370, 2018.
- [8] B. Halstead, Y. S. Koh, P. Riddle, M. Pechenizkiy, A. Bifet, and R. Pears, “Fingerprinting concepts in data streams with supervised and unsupervised meta-information,” in *Proceedings of the International Conference on Data Engineering*, 2021, pp. 1056–1067.

- [9] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Machine learning*, vol. 23, no. 1, pp. 69–101, 1996.
- [10] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys*, vol. 46, no. 4, pp. 1–37, 2014.
- [11] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean, "Characterizing concept drift," *Data Mining and Knowledge Discovery*, vol. 30, no. 4, pp. 964–994, 2016.
- [12] H. Borchani, A. M. Martínez, A. R. Masegosa, H. Langseth, T. D. Nielsen, A. Salmerón, A. Fernández, A. L. Madsen, and R. Sáez, "Modeling concept drift: A probabilistic graphical model based approach," in *International Symposium on Intelligent Data Analysis*. Springer, 2015, pp. 72–83.
- [13] C. W. Chiu and L. L. Minku, "Diversity-based pool of models for dealing with recurring concepts," in *Proceedings of the International Joint Conference on Neural Networks*, 2018, pp. 1–8.
- [14] B. Halstead, Y. S. Koh, P. Riddle, R. Pears, M. Pechenizkiy, and A. Bifet, "Recurring concept memory management in data streams: exploiting data stream concept evolution to improve performance and transparency," *Data Mining and Knowledge Discovery*, vol. 35, no. 3, pp. 796–836, 2021.
- [15] K. Chen, Y. S. Koh, and P. Riddle, "Proactive drift detection: Predicting concept drifts in data streams using probabilistic networks," in *Proceedings of the International Joint Conference on Neural Networks*, 2016, pp. 780–787.
- [16] A. Bifet and R. Gavaldá, "Learning from time-changing data with adaptive windowing," in *Proceedings of the 2007 SIAM International Conference on Data Mining*, 2007, pp. 443–448.
- [17] Z. Ahmadi and S. Kramer, "Modeling recurring concepts in data streams: a graph-based framework," *Knowledge and Information Systems*, vol. 55, no. 1, pp. 15–44, 2018.
- [18] D. Moreira dos Reis, A. Maletzke, D. F. Silva, and G. E. Batista, "Classifying and counting with recurrent contexts," in *Proceedings of the SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1983–1992.
- [19] J. Montiel, J. Read, A. Bifet, and T. Abdesslem, "Scikit-multiflow: A multi-output streaming framework," *Machine Learning Research*, vol. 19, no. 72, pp. 1–5, 2018.
- [20] S. Lee, J. Ha, D. Zhang, and G. Kim, "A neural dirichlet process mixture model for task-free continual learning," *arXiv preprint arXiv:2001.00689*, 2020.
- [21] R. M. Cruz, R. Sabourin, and G. D. Cavalcanti, "Dynamic classifier selection: Recent advances and perspectives," *Information Fusion*, vol. 41, pp. 195–216, 2018.
- [22] P. R. L. De Almeida, L. S. Oliveira, A. D. S. Britto, and R. Sabourin, "Handling concept drifts using dynamic selection of classifiers," in *Proceedings of the International Conference on Tools with Artificial Intelligence*, 2016, pp. 989–995.
- [23] P. R. Almeida, L. S. Oliveira, A. S. Britto Jr, and R. Sabourin, "Adapting dynamic classifier selection for concept drift," *Expert Systems with Applications*, vol. 104, pp. 67–85, 2018.
- [24] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puronen, "Dynamic integration of classifiers for handling concept drift," *Information fusion*, vol. 9, no. 1, pp. 56–68, 2008.
- [25] M. De Lange and T. Tuytelaars, "Continual prototype evolution: Learning online from non-stationary data streams," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 8250–8259.
- [26] A. Masloy, M. Pechenizkiy, Y. Pei, I. Žliobaitė, A. Shklyayev, T. Karkkainen, and J. Hollmén, "BLPA: Bayesian learn-predict-adjust method for online detection of recurrent change points," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 1916–1923.
- [27] P. M. Gonçalves Jr and R. S. M. De Barros, "RCD: A recurring concept drift framework," *Pattern Recognition Letters*, vol. 34, no. 9, pp. 1018–1025, 2013.
- [28] C. Alippi, G. Boracchi, and M. Roveri, "Just-in-time classifiers for recurrent concepts," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 4, pp. 620–634, 2013.
- [29] R. Anderson, Y. S. Koh, and G. Dobbie, "CPF: Concept profiling framework for recurring drifts in data streams," in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2016, pp. 203–214.
- [30] Y. Wang, Z. Li, Y. Zhang, L. Zhang, and Y. Jiang, "Improving the performance of data stream classifiers by mining recurring contexts," in *International Conference on Advanced Data Mining and Applications*. Springer, 2006, pp. 1094–1106.
- [31] A. M. Ángel, G. J. Bartolo, and M. Ernestina, "Predicting recurring concepts on data-streams by means of a meta-model and a fuzzy similarity function," *Expert Systems with Applications*, vol. 46, pp. 87–105, 2016.
- [32] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *Machine Learning Research*, vol. 8, pp. 2755–2790, 2007.
- [33] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Embreck, B. Pfharinger, G. Holmes, and T. Abdesslem, "Adaptive random forests for evolving data stream classification," *Machine Learning*, vol. 106, no. 9, pp. 1469–1495, 2017.
- [34] R. S. M. de Barros and S. G. T. de Carvalho Santos, "An overview and comprehensive comparison of ensembles for concept drift," *Information Fusion*, vol. 52, pp. 213–244, 2019.
- [35] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: A survey," *Information Fusion*, vol. 37, pp. 132–156, 2017.

Supplementary: A Probabilistic Framework for Adapting to Changing and Recurring Concepts in Data Streams

Anon

TABLE I: Dataset Characteristics

Dataset	Context size	# features	# contexts	Dataset Length
AQT	4000	25	6	72000
AQS	4000	25	6	72000
AD	880	10	10	15840
CMC	736	8	2	4416
TREE	5000	10	6	90000
STGR	5000	3	3	90000
WIND	5000	16	6	90000

I. DATASETS

We use four real-world datasets with identified contexts from [?].

- *Aedes-Culex* - AQT: Contains 25 features measuring the flight patterns of male and female mosquitoes, with a class label describing two different species. Context is given by six temperature ranges, modified in an experimental setting to form six concepts to change flight patterns [?].
- *Aedes-Culex* - AQS: Same as AQT using the sex of mosquito as a class label
- *Arabic Digits* (AD): Contains 10 features measuring speech patterns of speakers saying Arabic digits aloud, with speaker gender as class labels. Contexts are the digit being spoken.
- *CMC*: Contains 8 features measuring responses to a survey on contraceptive use in Indonesia, with a binary yes or no answer. Two age ranges designate contexts.

We use three synthetic datasets, code to recreate each is provided.

- *Stagger* (STGR) [?]: Implemented in Scikit-Multiflow [?], simulates objects with three categorical features, each with an accept or reject label. Concepts are three labeling functions which define acceptance. Concepts may conflict.
- *RandomTree* (TREE) [?]: Implemented in FiCSUM [?], generates samples from a decision tree. Concepts use distinct trees, leading to different join probabilities. Sampling is done using a $p(X)$ with a random mean, variance, skew and kurtosis for each concept. Labels are balanced. We define a *complexity* measure based on tree height. A *complexity* = d has a min height of d , and max height of $d + 2$. Increasing d generates deeper trees

with more complex $p(X, y)$, increasing the number of observations required to learn. We use $d = 3$, except for the experiment where complexity was varied. Code for our implementation is available in supplementary material.

- *WIND*: We propose a new air quality classification data generator able to generate synthetic data with concept drift in both $p(X)$ and $p(y|X)$. WIND simulates a circular set of sensors with a central target sensor. Labels are the quantized air quality of the target, features are current and previous readings from surrounding sensors. Spatio-temporal dynamics of pollution generation and transmission are simulated differently for each concept by placing n pollution sources and setting wind speed and direction which dictates pollution movement and dispersal. Each concept specifies n , their placement (upwind of the target), the strength of generation, variance in generation, rate of generation, and a wind speed and direction. Changes in concept can be abrupt, *i.e.*, a new factory has turned on, or gradual, a slow change in wind direction. Code for this synthetic data generator is available open source.

Streams are generated by combing observations drawn from each concept in each data source in varying configurations. We repeat each of the $|C|$ concepts, capped at 6, $r = 3$ times for a maximum of 18 concept drifts and 90,000 observations. The value 3 was chosen as it is large enough to capture recurring concepts in each stream, but small enough that the runtime of each test is not too large, allowing many tests to be run efficiently. We test SELeCTs performance against r , finding that SELeCTs performance improves as r increases. The order of concepts was based on a random transition matrix based on a circular order to ensure that all concepts are reachable from all other concepts. For each concept c_i , a transition probability of p^f was assigned to each of the $f \in [1, 2, \dots, F]$ shuffled concepts following c_i , with p a probability decay parameter and F the number of forward connections. A transition noise parameter tn introduces random transitions to any concept. We set $p = 0.7$, $f = 3$ and $tn = 0$ for most experiments, to create reasonably complex transition patterns. We evaluate sensitivity to transition noise to measure robustness to the transition pattern.

II. BASELINES

We test against FiCSUM, a recent state-of-the-art method using the previous standard adaptive learning framework, and two alternatives: RCD, CPF. We test DYNSE [?] as a dynamic classification method, DWM [?] as an ensemble method, and CNDPM as a recent state-of-the-art task-free continual learning method. Finally, we compare to LB and UB as theoretically minimal and optimal adaptive learning baselines.

- RCD [?]: Default implementation in MOA [?] (Java). Default parameters: KNN statistical test for relevance, Hoeffding Tree base classifier (for comparison to SELeCT), default EDDM drift detector (required).
- CPF [?]: Default implementation in MOA (Java). Default parameters: Hoeffding Tree base classifier.
- FiCSUM [?]: Default implementation in Scikit-Multiflow [?] (Python). We use the same values for shared parameters for the FiCSUM baseline and SELeCT. Default parameters: Hoeffding Tree base classifier, ADWIN drift detector, default meta-information measures (without IMF, MI and PACF as suggested by authors), default weighted cosine distance similarity measure, default Fisher score feature weighting.
- DYNSE [?]: Default implementation in MOA (Java). Default parameters from the paper: default KNORA Eliminate strategy. For C-F1 we consider the active state to be the state selected for each batch.
- DWM [?]: Default implementation in scikit-multiflow (Python). Default parameters. For C-F1 we consider the active state to be static to represent the single ensemble.
- CNDPM [?]: Default implementation in Pytorch (Python). Default parameters: MLP architecture with 1/4 sized layers to better learn the smaller number of features in our datasets (shared with SELeCT and UB variants). Active state is considered the identified task. We compare to a variant with prior set to 1 for all tasks.
- LB: A single scikit-multiflow Hoeffding Tree is run with no adaptation, representing the lower bound of adaptive learning performance.
- UB: System with a scikit-multiflow Hoeffding Tree base classifier and perfect information on ground truth concepts with a static 100 observation delay (no delay is not possible in any real system, so this was chosen as a realistic setting). Adapts to recurring concepts by reusing the last classifier and new concepts by initializing a new Hoeffding tree. Achieves the upper bound for adaptive learning given the Hoeffding Tree base classifier and a 100 delay.

III. EXPERIMENT SETUP

All experiments were repeated 45 times, using the numeric seeds 1 to 45 to seed the Numpy random number generator, using Numpy version 1.20. All systems are evaluated on the same datasets.

Environment All experiments were conducted on an AMD Ryzen Threadripper PRO 3995WX CPU with 64 cores running

at 2.70 GHz, with 256 Gb of RAM. Each process was assigned to a single virtual core to ensure consistent running time. Experiments were run using Python version 3.7 or Java version 16. The Scikit-Multiflow framework version used was 0.5.3, and the MOA framework version was 2020.07.1.

Source Code All source code is available at <https://bit.ly/3gqSKL5>. This includes source code for SELeCT, as well as evaluation and dataset creation scripts, including the WIND data generator. All instructions to run SELeCT as well as demo scripts recreating results and visualizing the proposed WIND dataset are given.

Time & Memory We evaluate the time and memory of each system, finding SELeCT has equivalent runtime and half the memory use of FiCSUM, taking an average over all datasets of 1406s and 96MB compared to FiCSUMs 1452s and 176MB. Full results are shown in the source code repository.

IV. PARAMETER SELECTION

Our SELeCT implementation inherits parameters from the FiCSUM concept representation ζ and ADWIN drift detection, and also requires new parameters, as described in the main paper. We do not see substantial differences in optimal parameters across the datasets tested (CMC and AD), thus propose a set of general default parameters. We use these parameters for all experiments in the paper, and do not tune them to specific datasets.

We selected default parameters using linear searches on the CMC dataset, validated on the AD dataset. For each parameter a valid range was selected and 20 evenly spaced values were tested for accuracy and C-F1 on CMC. The value with the highest harmonic mean of accuracy and C-F1 was selected. Table II lists all parameters in our implementation, values tested and final value. The risk parameters describe the risk level for the Hoeffding bound test and window selection in the continuous selection method, and the state grace period is a number of observations to pause selection when a new state is initialized to allow its representation to stabilize. Min state likelihood is a small value to avoid 0 value likelihoods collapsing probabilities. Prior parameters were described in the implementation. The merge correlation sets the threshold before states are merged, we set this to a high value to keep merging conservative, to avoid merging representations of different concepts.

V. SIGNIFICANCE TESTING

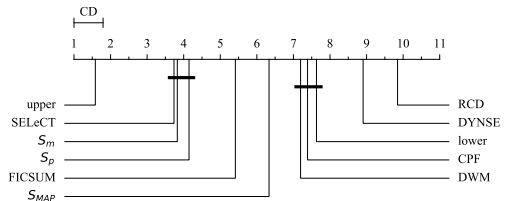


Fig. 1: Critical difference: κ -statistic across systems

TABLE II: SELeCT implementation parameter values

Parameter	Component	Range	Value
Outside Components			
Sensitivity	ADWIN	Default	0.05
Window size	ζ	Default	100
Buffer Ratio	ζ	Default	0.2
Our Components			
Hoeffding bound risk	Selection	0.05 - 0.9	0.75
Min state likelihood	Likelihood	0.0001 - 0.01	0.005
B prior multiplier	Prior	0.05 - 1.0	0.4
Min prior	Prior	0.05 - 1.0	0.7
Multihop prior multiplier	Prior	0.05 - 1.0	0.7
Prev state prior	Prior	0 -500	50
Merge correlation	Merge	0.05 - 1.0	0.95

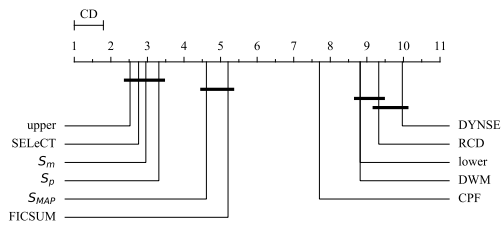


Fig. 2: Critical difference: C-F1 across systems