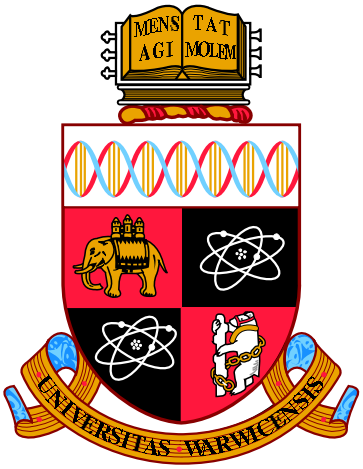


Online Transfer Learning for Concept Drifting Data Streams

by

Helen McKay

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy

Department of Computer Science

January 2022

Contents

List of Tables

List of Figures

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my supervisor, Nathan Griffiths, and primary advisor, Phillip Taylor. Their mentorship has been outstanding and has helped me grow as a researcher. I am privileged to have had their support and guidance, and this thesis would not have been possible without them. I would also like to thank Theo Damoulas, who has provided invaluable guidance, support and advice.

Second, I would like to thank Richard Cunningham and the technical support team in the Department of Computer Science for their assistance in providing the computational resources for gathering results. I would also like to thank the administrative team in the Department of Computer Science, specifically Sharon Howard, whose support and pastoral care has enabled me to complete this thesis. Additionally, I would like to thank many of the academic staff in the Department of Computer Science, Abhir, Claire, Sara, to name a few. Your belief and support has been greatly appreciated and allowed me to grow in confidence throughout my time at the University of Warwick.

Third, I would like to thank the friends that have supported me throughout my PhD. Liam, Caroline, Matt, Jack and Richard, my PhD experience would not have been the same without you all.

Finally, and most importantly, I would like to thank my family. My parents, Ian and Fran, and siblings, Sara and David. I don't think I can put into words how much your support and belief in me means. I will forever be grateful to you.

Declarations

This thesis has not been submitted for a degree at another university.

1 Publications

Parts of this thesis have previously been published by the author in the following:

[54] Helen McKay, Nathan Griffiths, Phillip Taylor, Theo Damoulas, and Zhou Xu. Online transfer learning for concept drifting data streams. In *BigMine@ KDD*, volume 2579, pages 1–15, 2019

[55] Helen McKay, Nathan Griffiths, Phillip Taylor, Theo Damoulas, and Zhou Xu. Bi-directional online transfer learning: a framework. *Annals of Telecommunications*, 75(9):523–547, 2020. doi: 10.1007/s12243-020-00776-1

[56] Helen McKay, Nathan Griffiths, and Phillip Taylor. Conceptually diverse base model selection for meta-learners in concept drifting data streams. *arXiv preprint arXiv:2111.14520*, 2021

2 Sponsorship and Grants

Parts of this research were supported by the UK EPSRC and Jaguar Land Rover under the iCase Scheme.

Abstract

Online Transfer Learning (TL) allows knowledge to be learnt from a data rich source domain to aid predictions in an online target domain. However, when all domains are online, and a data rich source domain does not exist, we must determine what to transfer, how to combine transferred knowledge, and whether to transfer knowledge. To ensure the feasibility of online TL methods in real-world applications, they should not only aid predictions in receiving domains, but should consider the communication and computational overheads of knowledge transfer. To address these challenges, this thesis presents methods for online TL when all domains are online, which are evaluated using synthetic and real-world regression-based datasets.

First, the BOTL framework is introduced, which enables knowledge transfer to be conducted bi-directionally between online data streams, where knowledge is transferred in the form of predictive models, and combined using an OLS meta-learner. Second, two methods of selecting a relevant yet diverse subset of transferred and locally learnt models are presented, namely parameterised thresholding and conceptual clustering. These approaches help to prevent overfitting when the number of models transferred is large in comparison to the window of available data. To reduce the computational overhead of selecting subsets of models, a static diversity metric is introduced, which estimates the conceptual similarity between models using the Principal Angles (PAs) between their underlying subspaces. Third, two methods for determining whether to transfer knowledge are presented, namely IdDT and IdCS, which maintain comparable predictive performances to when all models are transferred, while reducing the number of models received in each domain by 47.1% and 30% respectively across the experiments conducted for this thesis.

Acronyms

ACC Adaptive Cruise Control.

CDD Concept Drift Detector.

ERM Empirical Risk Minimisation.

MI Mutual Information.

MSR Multi-Stream Regression.

OLS Ordinary Least Squares.

OMTL Online Multi-Task Learning.

PAs Principal Angles.

PCs Principal Components.

RR Ridge Regression.

SC Spectral Clustering.

SVD Singular Value Decomposition.

SVR Support Vector Regression.

TL Transfer Learning.

TTC Time-To-Collision.

Notation

α	Domain α, where $\mathcal{A}$ is used in this thesis to denote a single source domain, and $\mathcal{B}$ is used to denote a receiving domain.
X^α	Data stream in domain α, where $X^\alpha = \{x_1, \ldots, x_t, \ldots, x_n\}$.
$x_t \in X^\alpha$	The t^{th} observed instance in X^α.
Y^α	The response variable space in domain α.
$y_t \in Y^\alpha$	The t^{th} response variable in Y^α.
C^α	The set of concepts encountered in domain α.
$c_i^\alpha \in C^\alpha$	The i^{th} concept encountered in domain α.
$X_i^\alpha \in X^\alpha$	The data stream segment corresponding to concept c_i^α in domain α.
$f_i^\alpha : X_i^\alpha \rightarrow Y_i^\alpha$	Model i learnt in domain α.
$\hat{y}_t^{\alpha_i}$	Prediction using $f_i^\alpha(x_t)$.
$\mathcal{M}^\alpha$	Knowledge base of models available in domain α.
x_t^*	Meta instance of base model predictions for instance x_t.
$F : X^\alpha \rightarrow Y^\alpha$	Meta-learner in domain α.
$\hat{y}_t^*$	Prediction using $F^{\mathcal{M}'}(x_t)$ where $\mathcal{M}' \subseteq \mathcal{M}$.
W	Sliding window of $\lvert W \rvert$ instances, $W = \{x_{t-\lvert W \rvert}, \ldots, x_t\}$.
W_{max}	Maximum window size (RePro).
W_{min}	Minimum window size (ADWIN, AWPro).
err_t	Predictive error of instance x_t.
err_W	Predictive error across W.
λ_l, λ_d	Loss and drift thresholds (RePro).
λ_r	Recurrence threshold (AWPro).
δ	Confidence value (ADWIN, AWPro).
λ_{perf}	Performance culling threshold.
λ_{MI}	Mutual information culling threshold.
λ_{CS}	Conceptual similarity culling threshold.
U_i	Principal Components of the window used to learn model f_i.

$\mathcal{M}'$	A subset of models available in domain α, where $\mathcal{M}' \subseteq \mathcal{M}^{\alpha}$.
Υ_{sim}	A subset of similar models in $\mathcal{M}$.
$\mathcal{R}(\vec{w})$	True risk of model with parameters $\vec{w}$.
$z_t = (x_t, y_t)$	Instance x_t at time t and respective response variable y_t.
$Q(z_t, \vec{w})$	Loss function of a model parameterised by $\vec{w}$ for instance z_t.
$\vec{w}_0$	Optimal parameters for a given function.
$Z_{\lvert W \rvert}$	Training sample of size $\lvert W \rvert$.
$\mathcal{R}_{\text{emp}}\left(\vec{w}_{\lvert W \rvert}\right)$	Empirical risk of model with parameters $\vec{w}_{\lvert W \rvert}$ learnt over $Z_{\lvert W \rvert}$.
$\vec{w}^{*}_{\lvert W \rvert}$	Model parameters that minimise the empirical risk over $Z_{\lvert W \rvert}$.

Chapter 1

Introduction

The functionality of many real-world applications must be tailored to their surrounding environments. For example, a smart home heating system should operate to meet a user's heating preferences to suit their pattern of life. Tailoring functionalities such as these often requires large numbers of user defined parameters as input. For example, if we consider Adaptive Cruise Control (ACC) for a vehicle, a driver may prefer to follow leading vehicles at different distances depending on factors such as weather conditions, road conditions, or even the purpose of the journey. Relying on a user to modify parameters to configure the functionality of applications can become a burden to users. For example, the performance of a smart home heating system is not only dependent on a user's pattern of life, but it is also influenced by external weather conditions, and therefore the user's preference may change seasonally [78]. To overcome this, machine learning techniques can be used to predict tailored functionalities or user preferences from sensing the environment and, to achieve this, learning algorithms must operate in online environments. Learning in online environments can be considered as learning from streaming data, where instances are observed sequentially. In this thesis, we consider the scenario where the response variable associated with each instance is observed immediately after a prediction has been made. In order to make predictions, an online learning algorithm must be used, which can be trained either by incrementally updating model parameters as each instance is sequentially observed, or by retraining a predictive model using recent batches of data [26].

Characteristics of online environments introduce three key challenges that must be considered by online learning algorithms. First, data required to train learning algorithms can only be obtained in real-time, meaning that instances of data are observed in the form of a data stream in which future instances are un-

known [53]. Second, in many online learning applications, a rich history of data cannot be retained since data may grow without limit [95]. Third, the underlying distributions in the data may change over time, a phenomenon known as concept drift [26, 43, 78]. Concept drift may be caused by changes in the distribution of observations in the data stream, as a consequence of changes in the environment, changes in the response variable to be predicted, or changes to the mapping between observations and response variable [37]. In order to maintain effective predictions, online learning algorithms must be able to adapt in the presence of concept drift. One way this can be achieved is using concept drift detection strategies to identify when predictive models must be updated or re-learnt due to concept drift [26].

A challenge of learning in concept drifting data streams is that data availability is limited [32]. Even in situations where a rich history of data can be retained, the presence of concept drift means that historical data may not be relevant to the current mapping between observations and response variable, referred to as the current concept. When a new concept is encountered, sufficient observations belonging to the new concept must be observed before a new model can be learnt, or an existing model updated [9]. To maintain effective predictions, online learning strategies need to react quickly to concept drift [95]. However, this means that a predictive model may need to be learnt or updated when few instances belonging to the new concept have been observed. This can cause predictive models to be created that do not generalise well, due to the limited availability of training data, and may not make effective predictions for future instances belonging to the same concept. Therefore, there is a trade-off between reacting quickly to concept drift and creating models that generalise well [9].

Other challenges associated with online learning manifest from the applications in which it is beneficial. Many such applications are situated in dynamic real-world environments, where a rapid response to the environment is paramount. A wide variety of devices may be used for these applications, ranging from low cost sensors to high performance computers. For example, a low cost sensor may be used for environmental monitoring, to make predictions that are influenced by weather conditions [33]; a smart phone may be used to monitor a user's behavioural patterns in order to make predictions with respect to their pattern of life [38]; and a high performance computer may be used to monitor and make predictions of network and communication irregularities [103]. Although the memory and computation capabilities of the devices used vary greatly, two factors are common across such applications. First, the memory and computation available on each device may be limited. Second, applications of this nature are typically not stand-alone and may

learn similar predictive tasks from independent data streams. For example, two low cost sensors may be used for environmental monitoring in two different locations [86]. Since each device may have limited data to learn predictive models and must react to concept drift, the models learnt on each device may not generalise well. Therefore, knowledge can be learnt on each device and shared to enhance their respective predictive performances through online Transfer Learning (TL) [104].

Existing online TL frameworks typically consist of one or more data rich offline source domains, where knowledge can be learnt in the form of predictive models. This knowledge can then be transferred to an online target domain that has limited data availability [27, 32, 104]. The knowledge transferred is combined with knowledge learnt in the target domain to improve predictive performance [65, 85, 107]. The primary challenges addressed by online TL research are determining what to transfer, and how to combine transferred knowledge with locally learnt knowledge to improve the predictive performance in the target domain [27, 52, 89, 97, 104]. However, these challenges have not been considered when all domains are online, and where a data rich offline source domain does not exist.

In this thesis, we consider the scenario where all domains in a TL framework are online. We address the challenges of determining what to transfer and how to use transferred knowledge, focusing specifically on the characteristics of each domain being online and susceptible to concept drift. In real-world applications where knowledge is transferred between domains that correspond to devices situated in different physical locations, online TL frameworks may also be required to consider the communicational overhead of knowledge transfer, since devices may be networked with limited bandwidth. This limitation must be considered when determining what to transfer, and also introduces the question of whether knowledge should be transferred between domains. Without this consideration, knowledge that is not beneficial to another domain may be transferred, incurring unnecessary communication overheads. Therefore, in this thesis, we also address the challenge of determining whether it is appropriate to transfer knowledge to other domains.

1.1 Objectives

The aim of online TL is to improve the predictive performance achieved in the target domain by transferring knowledge learnt from a source domain. However, when all domains are online, each domain can be considered as both a source and a target. Therefore, in this thesis, a domain that has learnt knowledge which can be transferred is referred to as a source domain, and a domain that has received knowledge

from another domain is referred to as a receiving domain. Each domain in the framework can be both a source domain and a receiving domain, with the overarching aim of online TL being to improve the predictive performance in each domain. This thesis addresses the challenges of determining what to transfer, determining how to combine transferred knowledge with locally learnt knowledge, and determining whether knowledge should be transferred to receiving domains. Specifically, the objectives of this thesis are as follows.

1. **To determine what can be learnt from an online data stream that may be beneficial to other independent online data streams when transferred throughout an online TL framework.**

   Since data streams are subject to concept drift, addressing the challenge of what to transfer requires predictive models to be created that represent the underlying concepts encountered in each domain.

2. **To determine how transferred predictive models can be combined in receiving domains to aid predictive performance.**

   Since each domain is online, knowledge transfer is bi-directional, and therefore a novel online TL framework must be developed to transfer and combine predictive models.

3. **To determine whether a predictive model should be transferred to other domains in an online TL framework.** The transfer of a predictive model may not always be beneficial to the receiving domain and, therefore, it should be determined whether a predictive model will be beneficial to a receiving domain prior to transfer to prevent unnecessary communication and computation overheads.

1.2 Contributions

This thesis consists of 6 main contributions, which focus on the challenges of determining how to combine knowledge, and determining whether knowledge should be transferred in online TL. The main contributions of the thesis are as follows.

- A novel online TL framework, the Bi-directional Online Transfer Learning (BOTL) framework, is introduced in Chapter 5, which allows knowledge to be transferred where all domains are online and there is no data rich offline source domain. Knowledge is transferred in the form of predictive models, which are combined with knowledge learnt locally from a receiving domain,

using an Ordinary Least Squares (OLS) meta-learner, addressing the challenge of how to combine transferred and locally learnt knowledge.

- As the number of models available in each domain, referred to as base models, becomes large, the OLS meta-learner in BOTL becomes prone to overfitting. Therefore, in Chapter 5, naïve culling mechanisms are also introduced, allowing the number of models available to the meta-learner to be reduced. Model culling is achieved using common ensemble pruning metrics, namely predictive performance and Mutual Information (MI). BOTL frameworks that employ these naïve culling techniques are referred to as P-Thresh and MI-Thresh respectively.

- Selecting a subset of relevant yet diverse base models is fundamental to preventing meta-learners from overfitting in online environments. However, existing diversity metrics, such as MI, measure diversity by the level of disagreement between base model predictions. Metrics such as MI must be re-calculated as the data stream progresses due to its dependency on the underlying distribution of the data. A novel diversity metric, in the form of conceptual similarity, that remains static even in the presence of concept drift, is introduced in Chapter 6. The conceptual similarity of base models is estimated using the Principal Angles (PAs) between the subspaces in which each base model was learnt. The applicability of this diversity metric is not limited to online TL frameworks, and it can be used in any online learning setting where the diversity of base models must be calculated.

- We show, in Chapter 6, how conceptual similarity can be used as a diversity metric for parameterised thresholding, which allows a subset of base models to be used as input to the OLS meta-learner in the BOTL framework, referred to as CS-Thresh. CS-Thresh achieves comparable predictive performances to MI-Thresh while requiring fewer diversity metric calculations in order select a subset of base models as input to the meta-learner.

- We also introduce parameterless conceptual clustering in Chapter 6, which uses conceptual similarity as a diversity metric and Self-Tuning Spectral Clustering (STSC) [100] to cluster base models available to a meta-learner, in order to obtain a relevant yet diverse subset of base models without requiring a user defined culling parameter. Parameterless conceptual clustering achieves comparable predictive performances to MI-Thresh and CS-Thresh, but is not dependent on domain expertise to select an appropriate culling threshold.

- Since a relevant yet diverse subset of base models should be selected as input to a meta-learner, models transferred throughout the framework that are not diverse to those already available in a receiving domain provide little benefit. Transferring such models incurs unnecessary communication and computation overhead. Therefore, two novel approaches that address the challenge of determining whether a base model should be transferred are introduced in Chapter 7, namely Inter-domain Diversity Thresholding (IdDT) and Inter-domain Conceptual Similarity (IdCS). These approaches maintain comparable predictive performances to frameworks where all models are transferred, while reducing the number of models received by each domain. Averaged across all datasets and Concept Drift Detectors (CDDs) considered in this thesis, IdDT reduces the number of models received by each domain by 47.1%, while IdCS achieves a 30% reduction, reducing communication and computation overheads.

To evaluate the main contributions of this thesis, 3 secondary contributions were required. The secondary contributions consist of novel datasets, containing online data streams specifically for regression tasks, and address the challenge of determining what can be learnt from these data streams that can be transferred in online TL. The secondary contributions of the thesis are as follows.

- Existing online TL research focuses on learning for classification tasks. However, many real-world applications require predictions to be made for regression tasks. Therefore, three types of novel datasets have been created and are presented in Chapter 3. These consist of two types of synthetic datasets, namely the drifting hyperplane datasets and the smart home heating simulator dataset, and one real-world dataset, namely the following distance dataset[1].

- To address the challenge of determining what to transfer, two existing CDDs, namely RePro [95] and ADWIN [8], have been adapted to create predictive models from regression based data streams that can be transferred throughout online TL frameworks. These CDDs are used to create predictive models that represent the concepts encountered in a data stream.

- A novel CDD, called AWPro, is presented in Chapter 4, which combines key characteristics of RePro and ADWIN that are beneficial when learning in online TL frameworks that have computation and communication limitations.

[1]These datasets and associated data generators are publicly available: `https://github.com/hmckay/BOTL`.

The remainder of this thesis is organised as follows. Chapter 2 provides an overview of relevant areas of research, including online learning, ensemble learning, and online TL. Chapter 3 presents the three types of datasets used in the remainder of this thesis to present experimental results, namely the drifting hyperplane datasets, the smart home heating simulator dataset, and following distance dataset. The underlying CDDs used to learn predictive models from online data streams are discussed in Chapter 4. Specifically, modifications to RePro [95] and ADWIN [8], to enable their use in regression settings, are presented, and a novel CDD, namely AWPro, is introduced. These CDDs are used subsequently throughout the thesis to create base models for transfer. The BOTL framework is introduced in Chapter 5, alongside naïve culling thresholding approaches to reduce the number of base models available to the meta-learner, namely P-Thresh and MI-Thresh. A novel diversity metric, conceptual similarity, is defined in Chapter 6, and used alongside parameterised thresholding, CS-Thresh, and parameterless conceptual clustering, CS-Clust, to obtain a subset of base models to be used as input to the OLS meta-learner in BOTL. In Chapter 7, two techniques for determining whether a base model should be transferred to a receiving domain are presented, namely Inter-domain Diversity Thresholding (IdDT) and Inter-domain Conceptual Similarity (IdCS). Finally Chapter 8 concludes the thesis by discussing the contributions made and outlining avenues of future research.

Chapter 2

Background

Learning in online environments can be challenging since data availability may be limited, and future distributions in the data are unknown [53]. Despite this, many real-world applications require learning to be conducted in such environments, and are typically not stand-alone [38]. This means that knowledge can be shared between applications that perform similar tasks to improve predictive performance [65]. To achieve this, two fields of machine learning research can be considered, namely online learning and Transfer Learning (TL).

In this chapter, common approaches to online learning are discussed in Section 2.1, highlighting the challenges associated with learning in online environments, and how knowledge can be learnt from online data streams. Section 2.2 discusses methods of handling concept drift using single model based approaches. Ensemble-based methods for online learning are discussed in Section 2.3, which consider how knowledge learnt within a data stream can be leveraged to improve predictive performance. Section 2.4 introduces TL, where knowledge can be learnt from one or more sources of data and used to improve the predictive performance for a similar task where data availability may be limited. TL has been used in online environments to improve the predictive performance when the receiving domain is online, and Section 2.5 presents some existing online TL frameworks, before Section 2.6 concludes this chapter by identifying common limitations of existing online TL approaches, motivating the contributions of this thesis.

2.1 Online Learning

When learning in online environments, predictions must be made from data streams, where data is received one sample at a time [26]. Often in machine learning, an

assumption is held that future samples of a data stream will be drawn from the same distribution as samples previously seen [43]. However, this may not be the case in real-world environments. In a data stream X, an instance $x_t \in X$ is observed at time t, for which a response variable $y_t \in Y$ must be predicted. To achieve this, a predictive model f can be learnt to map previously observed instances in the data stream to the response variable, such that $f : X \rightarrow Y$. The mapping between observations and the response variable is referred to as a concept, and can be considered as the joint probability distribution $P(X, Y)$ [26]. Since future instances within the data stream are unknown, and learning is conducted in dynamic, non-stationary environments, the joint probability distribution, $P(X, Y)$, may change over time, a phenomenon known as concept drift. Concept drift can be caused by changes to the underlying distribution of observations, $P(X)$, often referred to as virtual drift, or changes to the distribution of the response variable, $P(Y|X)$, often referred to as real drift [1, 26, 67, 78, 84, 108]. Often these types of drift occur together, however, regardless of the type of concept drift the joint probability distribution, $P(X, Y)$, changes, altering the learnt mapping between the observations and response variable, $f : X \rightarrow Y$ [37, 78, 108]. This means that a predictive model that had previously been learnt to represent the underlying concept, f, may no longer effectively represent the current mapping, $X \rightarrow Y$ [26]. Therefore, in order to maintain effective predictions, the predictive model, f, must be updated or relearnt [78].

Three types of concept drift are often considered, namely sudden drift, gradual drift and recurring drift [1]. These three types of concept drift can be described using c_i and c_j to denote two consecutive concepts observed within a data stream. Sudden drifts occur instantaneously between consecutive instances in the data stream such that the instance observed at time t belongs to the concept c_i, while the instance observed at time $t + 1$ belongs to concept c_j. This means that a predictive model learnt to represent concept c_i, denoted by f_i, may not be able to make an effective prediction for instance x_{t+1}, since instance x_{t+1} belongs to concept c_j. Alternatively, gradual drifts occur over a period of multiple instances, for example between times t and $t + m$ in the data stream. This means that instances observed prior to instance x_t belong to concept c_i, and instances observed after x_{t+m} belong to concept c_j. However, instances observed between times t and $t + m$ may belong to either concept c_i or c_j. This means that m instances are observed during the period of concept drift, where the probability of an instance belonging to concept c_i decreases proportionally to the number of instances observed after time t, while the probability of an instance belonging to concept c_j increases proportionally as

we approach x_{t+m} [108]. Recurring drifts occur when a historical concept reoccurs throughout the data stream. This means that a predictive model, for example model f_i that has been learnt to represent the concept c_i, could be used to make effective predictions if concept c_i reoccurs in the future. Reoccurring drifts can occur suddenly or gradually, and may be encountered periodically, or sporadically, throughout the duration of the data stream [53]. Sudden, gradual and recurring drifts must be handled when learning in online data streams to ensure that the model used to make predictions for unseen instances effectively represents the current concept.

2.2 Handling Concept Drift

Approaches to handling concept drift include implicitly adapting to concept drift by continually updating predictive models as the data stream progresses [49], or explicitly identifying concept drift through the use of detection mechanisms [76].

2.2.1 Implicitly Adapting to Concept Drift

Incremental learners can be used to implicitly adapt to concept drift by gradually updating model parameters as new instances are observed in the data stream [78, 108]. Gradually updating model parameters in this way means that incremental learners can be effective when reacting to gradual drifts. However, model parameters can be influenced by historical instances associated with a previously encountered concept [48]. This means that incremental learners may react slowly in the presence of sudden drifts [49]. Instance weighting and forgetting mechanisms can be used to prevent historical instances from negatively impacting predictive performance when concept drifts are encountered [26]. These techniques enable recently observed instances to have a greater influence on model parameters in comparison to historical instances, which may no longer be associated with the current concept [41]. Although incremental learners can be used in online environments, the effect of continually updating model parameters, without the ability to detect occurrences of concept drift, means that knowledge of previously encountered concepts can be lost as the data stream progresses [53]. This can be detrimental in environments that contain recurring concepts since the model parameters for a recurring concept must be re-learnt [95].

2.2.2 Explicitly Identifying Concept Drift

Alternatively, concept drift can be managed explicitly by detecting concept drifts as they are encountered throughout the data stream [1]. Methodologies that explicitly identify concept drifts are comprised to two stages, first a concept drift must be detected, and second, once a drift has been detected, an alternative predictive model must be used in order to adapt to the drift [5]. For example, a new predictive model could be learnt from recent observations, or a previous predictive model could be reused or updated. Concept Drift Detectors (CDDs) typically learn a predictive model over a small window of recent observations in the data stream and capture its predictive performance over a sliding window of instances, where new instances are added to the window as they are observed [40]. Concept drifts can be detected by monitoring the predictive performance [94], or identifying changes to the distribution of predictive error [8]. When concept drifts are detected, alternative predictive models can be learnt or reused (discussed in Chapter 4) [8, 54, 55, 95]. The use of a detection strategy means that sudden drifts can be identified quickly, and an alternative predictive model can be used which is not influenced by instances belonging to the previous concept. However, gradual drifts can be challenging to identify, particularly if the gradual drift occurs slowly and over a long period of time since there may not be an obvious change in a models predictive performance [49].

CDDs typically only retain recent observations, which means that they are not influenced by historical instances that belong to a previously encountered concept when detecting drifts or learning model parameters [8]. Since model parameters are not usually incrementally updated when using CDDs, historical knowledge of previously encountered concepts can be retained, and in some approaches is reused when recurring concepts are encountered. For example, RePro [94] detects drifts by monitoring predictive performance. When the predictive performance of a model drops below a threshold, either a new model is learnt from the recent observations captured in the sliding window, or a historical model is reused in the presence of recurring concepts [67]. Other approaches to detecting concept drifts estimate the precise point of drift by monitoring the distribution of predictive error. ADWIN [8] uses this approach so that instances belonging to the previous concept can be discarded prior to learning a new model. However, ADWIN does not make use of historical models when encountering recurring concepts, and therefore new predictive models are created in the presence of recurring concepts [8].

To learn a model that generalises well when using a CDD, sufficient instances belonging to the new concept must be observed. During this period, ineffective predictions may be made as the previously learnt model continues to be used. This

is known as the cold start problem, and introduces a trade-off between reacting quickly to drifts to prevent prolonged use of an ineffective model, and waiting for sufficient observations in order to learn a model that generalises well [7]. This means that predictive models may have to be learnt from small amounts of data. However, to improve generalisation abilities, ensembles are often used to combine multiple predictive models that have been learnt throughout the data stream [9, 22, 30, 47]. This allows previously learnt models to be used to enhance predictive performance and reduce the impact of the cold start problem, particularly when concepts re-occur throughout the data stream [31, 46].

2.3 Ensemble-based Methods for Online Learning

Ensemble-based approaches can be used alongside both implicit and explicit approaches to handling concept drift, for example, an ensemble-based approach could be used in combination with a drift detection mechanism [76]. However, instead of relying upon a single predictive model, multiple predictive models learnt throughout the history of the data stream, referred to as base models, are combined to improve predictive performance [9].

Dynamic Weighted Majority (DWM) [47], and the Additive Expert Ensemble (AddExp) [46], follow similar approaches to handling concept drift through the use of ensembles in classification settings. DWM and AddExp create base models, referred to as experts, using incremental learners. As new instances are observed, the weights associated with each base model are updated. These experts are combined through the use of a discounted weighting mechanism, where the influence of an expert is decreased by a multiplicative factor if it misclassifies an instance [46, 47]. If the overall prediction obtained by the ensemble is incorrect, then a new expert is created [46, 47]. AddExp has also been adapted to a regression setting, where the same principals are used, however misclassifications associated with the creation of new experts, and updating ensemble weights, are identified using the absolute error of an expert with respect to the current instance. Therefore, AddExp considers a misclassification as occurring when the absolute error of an expert, or the ensemble, is greater than a threshold [46]. The Online Weighted Ensemble (OWE) [31] follows a similar approach to AddExp in regression settings, where new base models are created when the predictive error of the ensemble exceeds a threshold. However, OWE weights base models based on their errors over a recent window of data, rather than incrementally discounting the weights [31, 46]. The weights assigned to base models in the ensemble are determined using a discount

factor that allows the predictive errors that were made in the past to have less impact in comparison to recent mistakes [31]. Using a discount factor to assign weights in this manner is beneficial when encountering recurring concepts since smaller discount factor parameters can be used to assign larger weights to historical models that achieve good predictive performances on recent windows of data [31].

DWM, AddExp and OWE all use incremental learners to update the model parameters for each base model in the ensemble as new instances are observed in the data stream [31, 46, 47]. However, this means that the model parameters of historically created base models continue to be updated, even when newly observed instances belong to a different concept. This means that knowledge of previously learnt concepts may be lost as the data stream progresses, requiring model parameters to be updated, or relearnt, in the presence of recurring concepts (discussed in Chapter 4) [54]. Accuracy Weighted Ensemble (AWE) [82] prevents this loss of information through the use of a chunk-based learning strategy, where the data stream is partitioned into chunks such that a new classifier is created for each chunk [82]. The weights assigned to each base model in the ensemble are proportional to their predictive accuracy on the most recent chunk of data. Therefore, the model parameters of each base model are not influenced by future instances in the data stream that may not belong to the same concept. In the presence of recurring concepts, a historical base model can be assigned a larger weight to increase its influence in the ensemble if it has previously been learnt to represent the same concept [82].

Since data streams are unbounded in size, and may have no limit [1, 95], predictive models could be added to the ensemble indefinitely, therefore approaches to selecting subsets of base models to retain in the ensemble are necessary. This is often achieved by defining a maximum number of base models that can be retained [46]. One approach to identify which base models should be removed from the ensemble is to use the age of the model such that the oldest base models are discarded first [46]. However, in concept drifting data streams, age may not be a good indicator of relevancy, since concepts can reoccur throughout the data stream [31, 46]. Many ensemble-based approaches to handling concept drift use the predictive performance of base models to determine which of those available should be retained, or discarded [23]. The predictive performance of base models could be evaluated over recent observations in the data stream [46], or the accumulated error could be considered [31]. In AddExp [46], AWE [82] and OWE [31] the worst performing base model is removed when the size of the ensemble becomes larger than a constant K, whereas in DWM base models are removed from the ensemble if the weight assigned to it drops below a threshold [47].

Although most ensemble-based approaches for handling concept drift determine which base models should be retained by evaluating their predictive performance, the diversity of base models can also be considered [36]. Maintaining a diverse subset of base models is required to maintain good generalisation in both classification and regression ensembles [11, 12, 23, 60]. Diversity for Dealing with Drifts (DDD) [58] maintains two ensembles to handle concept drift, one where base models have low diversity, which is used between concept drifts, and one where base models have high levels of diversity, which is used immediately after a concept drift. This approach enables DDD to maintain high generalisation abilities in the presence of concept drift. As new instances are observed in the data stream, all base models used by the ensembles, and the ensemble weights, are updated incrementally. Diversity is measured by considering the similarity between base model predictions for classification tasks using Yule's Q statistic [99].

DDD uses a drift detection strategy to determine which of the two ensembles should be used to make predictions. When a drift is detected, the high diversity ensemble is used to make predictions until sufficient instances belonging to the new concept have been observed so that a low diversity ensemble that effectively represents the new concept can be created. Once the new low diversity ensemble has been created, it can be used to make more accurate predictions than what could be achieved by the high diversity ensemble [58]. Since the weights of each base model are incrementally updated, the weights of base models that were previously learnt throughout the data stream are lost. This means that in the presence of recurring concepts, base model weights historically learnt are no longer available.

Concept Drift handling based on Clustering in the Model Space (CDCMS) [15] was introduced to address the loss of historically learnt weights caused by the use of incremental learners in DDD. CDCMS uses a methodology similar to DDD to handle concept drift, where an ensemble with high diversity, and an ensemble with low diversity are used to maintain high generalisation [58]. However, CDCMS creates new base models to be used by the ensemble every b time-steps so that predictive models that have been learnt to represent concepts are retained in case a recurring concept is encountered. CDCMS creates ensembles with high and low diversity using base model clustering such that base models are clustered using the similarity between their predictions over a recent window of observations. Therefore, base models in the same cluster make similar predictions, and as such, are likely to represent similar concepts [15].

CDCMS creates a highly diverse ensemble by selecting a base model from each cluster, whereas a low diversity ensemble is created by selecting base mod-

els from the same cluster. Additionally, CDCMS uses base model clustering as a memory management strategy to prevent all base models that have been learnt to represent the same concept from being discarded, allowing at least one base model to be retained in case the concept reoccurs [15]. Using ensemble-based approaches such as these, allows knowledge of previously encountered concepts to be used to improve predictions for unseen data [60], which can be considered as a form of TL [57].

2.4 Transfer Learning

Offline TL frameworks typically consist of one or more data rich domains, referred to as source domains, and one domain that has limited data availability, known as the target domain [65]. The aim of TL is to learn knowledge in the data rich source domains, and transfer this knowledge to the target domain to improve predictive performance in the target domain [85]. In TL, a domain consists of a feature space, $\mathcal{X}$ and a marginal probability distribution, $P(X)$, where X is the set of instances observed, such that $X = \{x_1, \dots x_n\}$ and $x_i \in \mathcal{X}$. Each domain is associated with a task, consisting of a label space, $\mathcal{Y}$, and a predictive function, f [65, 107]. TL aims to learn the mapping between the observations and response variable in a data rich source domain, such that $f : x_i \rightarrow y_i$, where $y \in \mathcal{Y}$ [65]. TL allows this knowledge to be used in a target domain, where data availability may be limited, to improve the predictive performance in the receiving domain [85].

There are three distinct types of TL: inductive, transductive and unsupervised [3, 17, 65]. Inductive TL is used when source and target predictive tasks are different. Knowledge is transferred from the source to induce a supervised predictive function in the target [17]. Typically, large amounts of labelled target data are required to create a mapping between domains [65]. Unsupervised TL is applied in a similar way, but to unsupervised learning tasks, such as clustering [65]. Transductive TL is used when the source and target predictive tasks are the same, transferring knowledge to improve the predictive performance in a target domain where no labelled data is available [3]. TL can be further categorised as homogeneous, where the source and target domains are the same, or heterogeneous, where they differ [104]. In this thesis, we consider a homogeneous setting, and use inductive TL to improve the predictive performances in the receiving domains.

Pan *et al.* outline four approaches of knowledge transfer typically considered when deciding what to transfer and how each type of knowledge transfer can be used in a target domain to improve predictive capabilities [65].

Instance transfer: This assumes that some, or all, instances of data available in

the source domain will be beneficial in the target domain. Therefore instances are transferred to the target domain so that they can be combined with the instances available locally in the target domain, enabling a predictive model to be learnt from source and target data [107]. This type of knowledge transfer is typically used with instance weighting mechanisms so that the locally available data in the target domain has greater influence on the predictive model created [39].

Feature-representation transfer: In this variant of knowledge transfer, a feature representation for the target domain is learnt in the source. For example, this may involve the identification of a subset of features available in the source domain that will be beneficial to the target domain. The feature representation is then transferred to the target domain and used to aid predictions [65].

Parameter transfer: This assumes that the source and target domains share some parameters of a machine learning model. Therefore, a model can be learnt from the data rich source domain, and model parameters are transferred to the target domain. Model parameters are often tuned in the target domain using the limited available data [85], or combined with model parameters learnt from locally available data using ensembles or meta-learners [96].

Relational-knowledge transfer: In this approach to knowledge transfer the relationships between source and target domain are identified and transferred. This typically involves approaches such as domain adaptation, where both source and target domains are mapped to a shared latent feature space such that knowledge available in the source domain can be used in the target domain [107].

Another research question that must be addressed by TL frameworks is whether knowledge learnt in a source domain should be transferred to the target domain [65]. The main reason for considering whether to transfer is to prevent negative transfer [65, 70, 83, 85, 107], which is the transfer of knowledge that is detrimental to the receiving domain [4, 6, 65]. This typically occurs when the tasks learnt in the source domains are too dissimilar to the tasks in the receiving domain [2, 4]. Some approaches to preventing negative transfer include analysing the relatedness of tasks [4, 6, 65], and clustering domains based on task relatedness [2, 4]. Negative transfer can become more prominent when transferring knowledge from multiple source domains, since it is highly unlikely for all source domains to be relevant to the receiving domain [28]. Approaches to address this when parameter transfer is

used as a method of knowledge transfer typically include the use of weighting mechanisms that assign weights to transferred models based on their predictive capabilities in the receiving domain [28, 77, 85].

2.5 Online Transfer Learning

When learning in online environments, data availability is often limited and a rich history of data cannot be retained [53, 95]. Therefore, TL can be used in online environments to improve the predictive performance obtained in an online data stream by transferring knowledge learnt from other sources of data [104]. Online TL can be broadly categorised into frameworks where knowledge is learnt from one or more offline source domains, and transferred to an online target domain, and frameworks where knowledge is learnt from online source domains and transferred to an online target domain. In both of these settings, online TL approaches must address research questions similar to those presented for TL, including what should be transferred, how transferred knowledge should be used, and whether knowledge should be transferred [65]. However, using TL in online environments is more challenging than in offline environments since the receiving domain is online, and therefore may be subject to concept drift. Since the presence of concept drift in the receiving domains changes the joint probability distribution of observations and response variable, $P(X, Y)$, how the transferred knowledge is used in a receiving domain must be adapted so that the transferred knowledge is best used to aid predictions with respect to the current concept observed in the target domain. Online TL can benefit many real-world applications where data availability is limited, however, in order to ensure that the use of online TL is feasible, the communication and computation overheads of knowledge transfer should also be considered [38].

Although instance, feature-representation, parameter and relational-knowledge transfer are all applicable methods of knowledge transfer in traditional offline TL settings [65, 85, 107], their feasibility may differ when using TL in some online environments. For example, in frameworks where communication and computational resources are limited, as might be expected in real-world environments where knowledge must be transferred between domains situated in different physical locations. Feature representation and relational knowledge transfer may be the least applicable in environments such as these. This is because in order to learn a feature representation for the target domain, or to identify the relationship between source and target domains, knowledge of the underlying distribution of data in the target domain must first be made available to the source domain, which may not be

feasible in frameworks where all domains are online and communication and computational resources are limited. Additionally, since the future distribution of the data is unknown in advance, feature representations and relational knowledge can only be learnt using data observed historically in the online receiving domain. Therefore, the feature representation or relational knowledge initially learnt may not be learnt from a distribution that is representative of the current concept observed in the target domain. The presence of concept drift may also invalidate the feature representations or relational knowledge learnt as the data stream progresses. Feature representations and relational knowledge could be relearnt as new concepts are encountered, however, the computational overhead associated with these processes must be considered in order for them to be feasible for real-world applications of online TL.

Instance transfer could be used in online TL frameworks, however, the communication overhead of transferring all, or a subset of, instances from the source domain to the target domain is usually considered to be too high [63]. Therefore, most online TL frameworks are restricted to using parameter transfer, where predictive models are learnt from data in the source domains, and are transferred to the target domain, allowing knowledge to be transferred with low communication and computation overheads [38]. When using parameter transfer in online TL, the process of combining transferred knowledge with locally learnt knowledge shares fundamental similarities to ensemble-based approaches to handling concept drift. This is because multiple predictive models are available in the receiving domain, which must be combined to achieve improved predictive performances in comparison to using a single model alone, while adapting to concept drift [57].

2.5.1 Offline Source Domains

The most frequently considered setting for online TL is where knowledge is transferred from one or more data rich offline source domains to improve the predictive performance in an online target domain [18]. The Online Transfer Learning (OTL) framework was one of the first frameworks to suggest using this approach [104]. OTL creates a predictive model from source domain data, which is transferred to the receiving domain. This predictive model is combined with a predictive model learnt in the receiving domain via incremental learning. To combine the models, a loss based weighting mechanism is used. As instances are observed in the online target domain, the model parameters associated with the locally learnt model are updated, and the weights associated with source and target models are adjusted with respect to their loss on the most recently observed instance in the target do-

main [104]. Incrementally updating the target model's parameters and adjusting the weights assigned to each model allow OTL to adapt to concept drift.

The OTL framework is used in Concept-Drifting Online Learning (CDOL) [105] as an ensemble-based approach to handling concept drifts in classification data streams. In CDOL, a single data stream is partitioned, such that the previously encountered partition is considered to be the offline source domain, and the current partition is considered to be the online target domain. In each partition an incremental learner is used to create a predictive model of recent observations, and combined with the predictive model transferred from the previous partition. Which ever is the best performing model in that period is transferred to the next partition to be used as a source model [105].

The Generalized Online Transfer Learning framework (GOTL) [32, 33] extends the OTL weighting mechanism such that online TL can be used for both classification and regression. The weighting mechanism used by GOTL incrementally updates in steps to obtain weightings for source and target models. If the step size, δ, used to modify the weights is small enough, the ensemble of source and target models approximates the optimal weight combination [32]. However, if the step size is too small, it may take substantial time for the weights to update to their desired values, making predictions unreliable during this period.

Many other online TL frameworks have been developed for the scenario where knowledge is learnt from multiple data rich offline source domains [27, 88, 91, 97]. Since negative transfer can become more prominent when knowledge is transferred from multiple sources [28], online TL frameworks have been developed to reduce the impact of negative transfer [21, 28, 89]. For example, Online Multiple Source Transfer Learning (OMS-TL) [27] obtains weights by solving a convex optimisation problem to combine source models and the target model when new instances are observed in the target domain [27]. Online Transfer Learning with Multiple Sources (OTLMS) [88] creates an ensemble of classifiers, where base models are learnt from source domains, which are weighted based on their predictive performance on new instances observed in the target domain. The ensemble of source models is then combined with an incremental learner in the target domain using a linear weighting combination strategy. OTLMS uses these two distinct stages to combine transferred and locally learnt knowledge to help prevent the source domains that are irrelevant to the current target concept from influencing the final prediction in the target domain [88].

OMS-TL and OTLMS reduce the impact of negative transfer through the use of weighting mechanisms. The Hybrid Ensemble Concept Drift tolerant Transfer

Learning framework (HE-CDTL), aims to reduce the impact of negative transfer by partitioning the target data stream into chunks, and projecting the data from the offline source domains to the target space for each partition in the target data stream [92]. In doing so, predictive models can be learnt from the projected source domain data so that the knowledge learnt from the source domain is relevant to the target domain, thereby reducing negative transfer [92]. These predictive models are then combined with a locally learnt model using weights that are proportional to the predictive performance of each model on the current partition of data observable in the target domain. Although this approach is effective at reducing negative transfer, in order to project the source domain data into the target space, the feature spaces of the domains must be transferred, requiring large communication overheads, which may not be feasible in some real-world applications of online TL.

2.5.2 Online Source Domains

For many real-world applications, a data rich offline source domain may not exist, and therefore, more recently, new online TL frameworks have been proposed that allow knowledge to be learnt from one or more online source domains, which is transferred to an online target domain [20, 21, 57, 87]. This paradigm of online TL is also referred to as TL in non-stationary environments [57]. When all source domains are online, every domain in the framework can encounter concept drift. However, knowledge can be leveraged from each of the concepts encountered in a source domain, or from historical concepts in the target domain, and used to improve predictive performance with respect to the current concept observed in the target domain [20].

Multi-sourcE onLine trAnsfer learning for Non-statIonary Environments (Melanie) [20] achieves knowledge transfer by creating a pool of ensembles associated with each data stream, where a new ensemble is created for every concept encountered in the source and target domains, identified using a CDD. Each ensemble is comprised of base learners, also referred to as sub-classifiers, that have been learnt to represent sub-concepts. Melanie is used for classification tasks, and therefore a sub-classifier is learnt to predict the probability for each class label, and the ensemble combines base models to obtain the most likely class label for unseen instances [20]. For every new instance of data observed in a source or target domain, the most recent ensemble in that domain, representing the current concept, is incrementally updated. If a new instance is observed in the target domain the weights assigned to every sub-classifier, for every ensemble, in every domain are updated such that they are assigned a weight proportional to the sub-classifier's ac-

curacy on the target data. A performance threshold is used to assign zero weights to sub-classifiers that have poor performance to prevent them negatively impacting the overall prediction. Resultant predictions are made by multiplying the sub-classifier weights with the probabilistic prediction made by each sub-classifier, and a final classification is made using majority voting across all sub-classifiers predictions [20].

Melanie's approach to knowledge transfer is beneficial in online TL when all domains are online since knowledge can be learnt from historically encountered concepts in each domain and transferred to the target domain to make predictions. However, in order to achieve this, large communication overheads are required, since an ensemble of sub-classifiers must be transferred to the target domain for every concept encountered in every source domain. Additionally, since the most recent ensemble in each of the source domains is continually updated until a concept drift is detected, ensemble weights, and base model parameters, must be continually transferred to the target domain.

Melanie combines transferred knowledge through the use of weighted averaging, and therefore the target domain may not fully benefit from knowledge learnt in source domains when the task to be learnt in the source domain is dissimilar to the target [21]. To further prevent negative transfer, Melanie was extended to develop Multi-source mApping with tRansfer LearnIng for Non-stationary Environments (Marline) [21]. Marline extends Melanie so that the target domain can benefit from source concepts that are not similar to the current target concept by projecting the target concept onto the space of each source concept [21]. This allows new instances received in the target domain to also be projected into the space of each source concept so that sub-classifiers for each source concept can make more effective predictions. The mapping between target and source concept is achieved using a pair of reference points for each concept. For example, a pair of reference points are obtained for a classification task by taking the centroids of the distributions for positive and negative class labels [21]. The reference points are used to create a vector for each concept, which allows a transition matrix to be identified, mapping the target concept to the source concept by rotating the vectors between the reference points of source and target concepts. Marline then assigns weights to sub-classifiers and ensembles learnt in source and target domains and combines them in a similar way to Melanie to obtain a classification for instances observed in the target domain [20, 21]. This approach is beneficial since it allows the target domain to make better use of predictive models learnt in source domains, without requiring significant additional communication overheads to achieve this mapping. However, similarly to Melanie, Marline requires significant communication overheads to trans-

fer all ensembles, all sub-classifiers, and all weights for every concept encountered in source domains.

Adaptive Online TrAdaBoost (AOTrAdaBoost) [87] achieves online TL in a slightly different manner to Melanie and Marline. For every concept encountered in an online source or target domain a predictive model is created and stored along with the set of training instances used to create each predictive model. When a new concept is encountered in the target domain, a predictive model is created. This model is then compared to other models that have been learnt from online source domains to identify a single model in each domain that is the closest match to the current target concept [87]. Similar models in the source domains are identified using their predictions for instances observed in the target domain, and the Kappa statistic is used to determine which model in each source domain makes the most similar predictions to the model learnt in the target domain. The training instance associated with each similar model, and newly observed instances in the target domain, are used by a boosting ensemble. If any of the base models in the boosting ensemble achieve an improved predictive performance over the locally learnt target model, then it is replaced by the base model from the boosting ensemble [87]. This approach to online TL uses both parameter transfer and instance transfer, which may incur high communication overheads if the size of training data is large, however, only model parameters for a single base model are transferred for each concept. AOTrAdaBoost may encounter large computational overheads in order to identify source models that match the current target concept. This is because model similarity is determined using the predictive performances of source models over a window of data in the target domain. Therefore, model similarity must be recalculated as the data stream progresses since the similarity between model predictions may differ when evaluated at different intervals in the target data stream due to concept drift.

2.5.3 Related Research Areas

The field of online TL relates to Online Multi-task Learning (OMTL) [62, 71, 73], and Multi-Stream Regression (MSR) [35]. MSR can be seen as a special case, where the source and target data streams are drawn from the same underlying distribution, and all concepts encountered in the target domain have previously been encountered in the source [19]. This means that the models transferred from the source can be used to make predictions in the target without requiring a target learner. This is unrealistic for many real-world applications as although source and target domains may be similar, it is unlikely that the data streams are drawn from the same distribution. The goal of OMTL is to minimise the cumulative global

loss across all domains [50], whereas online TL aims to minimise the predictive losses within each individual domain. Considering loss in this way is beneficial when applied to tasks such as application personalisation, where each domain represents a different user, and prediction errors should be minimised for that specific individual.

2.6 Summary

Research into the use of online TL is limited, particularly in the paradigm where both source and target domains are in online environments. Existing approaches to online TL mostly focus on classification tasks, and depend on methodologies that are not easily extended to regression tasks. Therefore, in this thesis we consider a regression setting where all domains in the framework are online. The communication and computation overheads of the techniques presented in Section 2.5 are also rarely considered, which may limit their applicability to real-world applications. Additionally, although some online TL frameworks consider the scenario where a data rich offline source domain is not available, existing online TL frameworks focus only on improving the predictive performance in a single receiving domain. When all domains are online, and a data rich source domain does not exist, all domains in the framework could benefit from knowledge transfer. However, this may have significant computation and communication implications which must be considered. Therefore, in this thesis we consider the challenge of online TL when all domains are online and knowledge is transferred in a peer-to-peer fashion so that every domain in the framework can benefit from knowledge transfer. We also consider the computation and communication overheads that impact the applicability of the proposed approaches in real-world environments where each data stream is collected from sensing independent environments. An example of this is when each domain is associated with a low cost sensor, and each sensor is situated in a different physical location. Although most online TL frameworks aim to reduce the impact of negative transfer, through the use of weighting mechanisms or transforming the feature space of the target domain onto the feature space of the source domain, determining whether the knowledge selected for transfer will be beneficial to the receiving domain prior to transfer is rarely considered. This consideration is important when using online TL to transfer knowledge between independent data streams, particularly when they are situated in a distributed environment, since the transfer of knowledge that is unlikely to be beneficial to a receiving domain incurs unnecessary computation and communication overheads. Therefore, determining whether to transfer is also considered in this thesis.

Chapter 3

Datasets

Many benchmark datasets have been created to evaluate online learning research, including concept drift detection strategies and online ensembles [40, 74, 75], however, most are categorically labelled. Additionally, most existing online TL research also focuses on classification tasks [46, 52, 104]. However, many real-world applications require predictions to be made in regression settings [16, 59, 66]. Therefore, the thesis presents new regression datasets containing concept drift which have been created in order to evaluate online TL techniques in regressive settings.

In this chapter, three types of novel datasets are introduced. First, a modification of the classification based drifting hyperplane benchmark dataset [46] is introduced, enabling it to be used for regression tasks [54]. Second, a simulation of a smart home heating system is presented, using data from a UK weather station to derive desired heating temperatures. Third, a real-world following distance dataset is introduced, created from vehicle telemetry data, and used to predict the Time-To-Collision (TTC).

The datasets presented in this chapter have been used throughout Chapters 4–7 to evaluate CDDs, the BOTL framework, base model selection strategies, and deciding whether to transfer knowledge. In Section 3.1, dataset characteristics that are required to effectively evaluate online learning and online TL techniques are discussed. The drifting hyperplane, smart home heating simulator, and following distance datasets are then introduced in Sections 3.2, 3.3 and 3.4 respectively.

3.1 Dataset Characteristics for Online TL

In order to be used in online TL frameworks, a dataset must have certain characteristics. Each dataset must be comprised of multiple independent, but related, data

streams. This allows each data stream to be used as source and target domains in online TL frameworks. Each data stream must contain observations, denoted by $x \in X$, a response variable, $y \in Y$, and a mapping between them, $X \rightarrow Y$. Since learning is conducted in an online environment, each data stream may be subject to concept drift, where the mapping between observation and the response variable changes over time.

The research presented in this thesis is limited to homogeneous online TL. This means that the feature space of observations, and the response variable are consistent across the data streams in a dataset. However, since each domain is independent, the mappings between observations and the response variable may differ between data streams, and therefore the concepts observed in each domain may differ. Additionally, since each data stream is independent, concept drifts do not occur synchronously across data streams.

For each of the datasets, different types of concept drifts are considered, namely sudden drifts and gradual drifts. Sudden drifts occur when the mapping between observations and the response variable changes instantaneously between consecutive observations in a data stream, whereas gradual drifts occur slowly over a period of multiple observations. Recurring drifts are also considered in each of these datasets, where historical concepts occur repeatedly throughout the duration of a data stream. Table 3.1 summarises the key characteristics of each of the datasets presented in this chapter.

3.2 Drifting Hyperplane Datasets

The drifting hyperplane datasets are modifications of a commonly used benchmark dataset [46], adapted for regression settings [55]. Each instance x_t at time t, is a vector, $x_t = \{x_{t_1}, x_{t_2}, \ldots, x_{t_n}\}$, containing n randomly generated, uniformly distributed, variables, $x_{t_n} \in [0, 1]$. For each instance, x_t, a response variable, $y_t \in [0, 1]$, is created using the function $y_t = (x_{t_p} + x_{t_q} + x_{t_r})/3$, where p, q, and r reference three of the n variables of instance x_t. This function represents the underlying concept, c_a, to be learnt and predicted. Concept drifts are introduced by modifying which features are used to create y_t. For example, an alternative concept, c_b, may be represented by the function $y_t = (x_{t_u} + x_{t_v} + x_{t_w})/3$, where $\{p, q, r\} \neq \{u, v, w\}$ such that $c_a \neq c_b$.

A variety of drift types have been synthesised using this data generator, including sudden drifts, gradual drifts and recurring drifts. In addition to synthesising different types of drifts, this data generator can also be extended to synthesize data

Dataset	Characteristic	Dataset Type	Drift Type S/G/R	# Streams	Avg. $\#x_t$ per Stream	# Features	Artificial Drifts per Stream
SuddenA	Uniform Noise	Synthetic	✓ / – / ✓	5	10,000	12	20
SuddenB	Sensor Failure	Synthetic	✓ / – / ✓	5	10,000	12	20
SuddenC	Intermittent Sensor Failure	Synthetic	✓ / – / ✓	5	10,000	12	20
SuddenD	Sensor Deterioration	Synthetic	✓ / ✓ / ✓	5	10,000	12	20
GradualA	Uniform Noise	Synthetic	– / ✓ / ✓	5	11,900	12	20
GradualB	Sensor Failure	Synthetic	✓ / ✓ / ✓	5	11,900	12	20
GradualC	Intermittent Sensor Failure	Synthetic	✓ / ✓ / ✓	5	11,900	12	20
GradualD	Sensor Deterioration	Synthetic	– / ✓ / ✓	5	11,900	12	20
Heating	Weather Data	Hybrid	✓ / ✓ / ✓	5	17,664	9	NA
Following	Vehicular Data	Real-World	✓ / ✓ / ?	17	1909	31	NA

Table 3.1: Dataset characteristics. A drift type of S indicates sudden drift, G indicates gradual drift, and R indicates recurring drifts. A "?" indicates that the presence of the type of drift is unknown due to the use of real-world data. All datasets use numerical features.

streams with differing severities of drifts, which could be considered in future work. A sudden drift from concept c_a to concept c_b is created between time steps t and $t+1$ by instantaneously changing the underlying function used to create y_t to an alternative function for y_{t+1}. A gradual drift from concept c_a to c_b is introduced between time steps t and $t+m$, where m instances of data are observed during the drift. Instances of data created between t and $t+m$ use one of the underlying concept functions, c_a or c_b, to determine the response variable. The probability of an instance belonging to concept c_a decreases proportionally to the number of instances seen after time t, while the probability of it belonging to c_b increases proportionally as we approach $t+m$. Recurring drifts are created by introducing a concept c_c that reuses the underlying function defined by a previous concept, c_a, such that conceptual equivalence is achieved, where $c_c = c_a$.

Four variations of the drifting hyperplane datasets are considered in this thesis, introducing concept drifts that represent different problems that may be encountered when learning in real-world environments[1]. The first variation simply introduces uniform noise, where $y_t \pm 0.05$ with probability 0.2. Datasets generated in this way are denoted as SuddenA and GradualA for sudden and gradual drifting data streams respectively. The second variant simulates sensor failure by setting a feature vector, i, to 0 at time t for the remainder of the data stream with probability 0.001, such that $x_{t_i} = 0$. In the case where feature i is used to create the response variable y, two other randomly selected feature vectors, j and k, are modified such that $x_{t_j} = x_{t_i}/4$ and $x_{t_k} = 3x_{t_i}/4$. This ensures that the underlying concept can still be learnt from the data. Datasets generated in this way are denoted as SuddenB and GradualB for sudden and gradual drifting data streams respectively.

The third variation simulates intermittent sensor failure by selecting a feature vector i to fail at time $t-1$ with probability 0.001. Once selected to fail, the feature value at all subsequent time steps t is set to 0 such that $x_{t_i} = 0$ with probability 0.3. Datasets generated using this scenario are denoted as SuddenC and GradualC.

The final variant simulates the deterioration of a sensor by including noise depending on the time step t, such that $x_{t_i} = x_{t_i} \pm (0.2(t/|X|))$, where 0.2 is the maximum amount of noise added to x_{t_i} and $|X|$ is the number of instances in the dataset. This means that as the data stream progresses, more noise is added to an individual feature, simulating the gradual deterioration in accuracy of a sensor over time. Additionally, the probability of a sensor deteriorating increases as the data stream progresses, such that the probability of a feature being selected for

[1]The drifting hyperplane datasets are available at: `https://github.com/hmckay/BOTL/tree/master/HyperplaneDG`

deterioration at time t is $0.001(t/|X|)$. Datasets generated in this way are denoted as SuddenD and GradualD for sudden and gradual drifting data streams respectively.

Each drifting hyperplane dataset consists of 5 data streams, each containing 5 concepts, where each concept occurs for durations of 500 consecutive instances. Each concept occurs 4 times throughout the duration of a data stream, ensuring that every data stream contains recurring concepts. Data streams within each drifting hyperplane dataset share at least 3 concepts with other data streams in the same dataset variant. In gradual drifting hyperplane datasets, gradual drifts occur over a period of 100 instances. These synthetic datasets are useful when evaluating online TL frameworks since some of the knowledge learnt from one data stream will be beneficial to other data streams while other knowledge learnt will not be of use to receiving domains. Since concept drifts are artificially introduced, these datasets are also useful for evaluating CDDs, since it is known how many concepts are present in each data stream, and the points of concept drifts are known. However, due to their synthetic nature, they may not be good indicators of how online TL frameworks operate in real-world environments.

3.3 Smart Home Heating Simulator

The heating simulator dataset was created to generate data streams that are more representative of real-world environments. Desired heating temperatures for a smart home heating system were derived using data collected from a weather station in Birmingham, UK, from 2014 to 2016. This dataset contains rainfall, temperature and sunrise data, which are combined with a synthesised schedule, obtained from sampling an individual's pattern of life, to determine when the heating system should be engaged[2].

To create data streams for the heating simulator dataset, a weather simulator was created to convert daily weather data into a data stream containing instances at half hourly intervals. The weather simulation uses average daily temperatures and monthly rainfall, obtained from the UK Met Office[3]. This is used, with the addition of noise, to estimate temperature and rainfall at half hour intervals. To create a representative model for changing temperatures over daily cycles, an algorithm proposed by Reicosky *et al.* was used [68], as shown in Algorithm 1. This algorithm takes the daily maxima, Temp_{max}, and minima, Temp_{min}, temperatures,

[2]The heating simulator dataset is available at: `https://github.com/hmckay/BOTL/tree/master/HeatingSimDG`

[3]UK Met Office data is available at: `https://www.metoffice.gov.uk/research/climate/maps-and-data/historic-station-data#?tab=climateHistoric`

Algorithm 1: Calculation of temperature distributions using Temp_{max}, Temp_{min} and RISE_{sun} [68].

Input: Temp_{max}, Temp_{min}, RISE_{sun} (time of sunrise)

$\text{Temp}_{\text{avg}} = (\text{Temp}_{\text{max}} + \text{Temp}_{\text{min}})/2$

$\text{AMP} = (\text{Temp}_{\text{max}} - \text{Temp}_{\text{min}})/2$

if $H < RISE_{sun}$ **then**

    $H' = H + 1000h$

else

    $H' = H - 1400h$

for $0000h \leq H < RISE_{sun}$ *and* $1400h < H \leq 2400h$ **do**

    $\text{Temp}(H) = \text{Temp}_{\text{avg}} + \text{AMP}(cos(\pi H'/(10.0 + \text{RISE}_{\text{sun}})))$

for $RISE_{sun} \leq H \leq 1400h$ **do**

    $\text{Temp}(H) = \text{Temp}_{\text{avg}} - \text{AMP}(cos(\pi(H - \text{RISE}_{\text{sun}})/(1400 - \text{RISE}_{\text{sun}})))$

and the estimated time of sunrise, RISE_{sun}[4], to obtain a temperature distribution that effectively represents changing temperature for different hours in the day, H, in real-world environments [68]. Additionally, the average rainfall for each month is used to estimate the probability of rain, which in turn is used to simulate the occurrence of rainfall.

The weather simulator is used alongside a heating schedule to determine the desired heating temperature for a simulated user. The heating schedule has been created by sampling an individual's pattern of life in order to create a synthetic rule set which changes desired heating temperatures based on features such as outside temperature, day of the week, time, and rainfall. An example of a rule contained within the heating schedule is:

$$\textit{If } \text{day} = \text{weekday} \wedge \text{Temp}_{\text{outside}} < 8°C \wedge \text{rain} = \text{True } \textit{then } \text{Temp}_{\text{desired}} = 24°C + \alpha,$$

where α is noise. Figure 3.1 shows two samples of data generated via the smart home heating simulation. The data samples are selected from February and June respectively, highlighting the dependency of desired heating temperature upon external weather conditions.

To create multiple data streams, weather data is sampled from overlapping periods of time, and used as input to the synthesised heating schedule to determine the desired heating temperatures. Due to the dependencies on weather data, each data stream is subject to large amounts of noise. Concept drifts are introduced manually by changing the schedule, however, drifts also occur naturally due to changing weather conditions. By sampling weather data from overlapping time

[4]Sunrise data is available at: `http://www.sunsettimes.co.uk/`

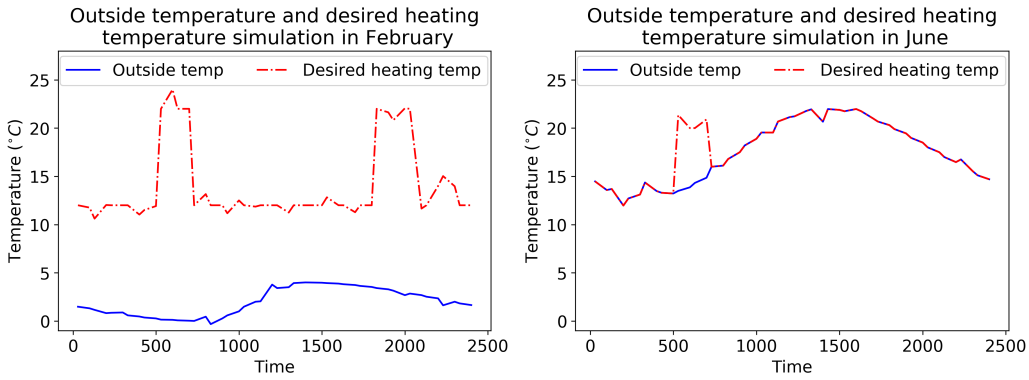


Figure 3.1: Heating schedule and outside temperatures in February and June.

periods, and due to seasonality, data streams follow similar trends, which can be seen in Figure 3.2, ensuring that some of the knowledge learnt in each data stream is beneficial to other data streams. The data streams in the heating simulator dataset contain desired heating temperatures over an average duration of 12.2 months, with a maximum of 18 months, and a minimum of 7. By using concepts that have a more complex underlying distribution, and are dependent on noisy data, the evaluation of online learning and online TL techniques on this dataset is more indicative of what is achievable when used in real-world environments.

3.4 Following Distance Datasets

To determine the effectiveness of online TL approaches, real-world data is needed to ensure that the methodologies developed can be applied to real-world applications. In this section, the following distance dataset is presented, which uses a vehicle's following distance and speed to calculate the Time To Collision (TTC) when following another vehicle [54]. Vehicle telemetry data such as speed, gear position, brake pressure, throttle position and indicator status, alongside sensor data that infer external conditions, such as temperature, headlight status, and windscreen wiper status, were recorded at a sample rate of 1Hz. Additionally, a selection of signals such as vehicle speed, brake pressure and throttle position were averaged over a window of 5 seconds to capture a recent history of vehicle state. The vehicle telemetry and environmental data captured within these data streams can be used to make predictions that allow vehicle functionalities to be personalised and reflect current driving conditions. For example, Adaptive Cruise Control (ACC) can be personalised by predicting TTC to identify a driver's preferred following distance.

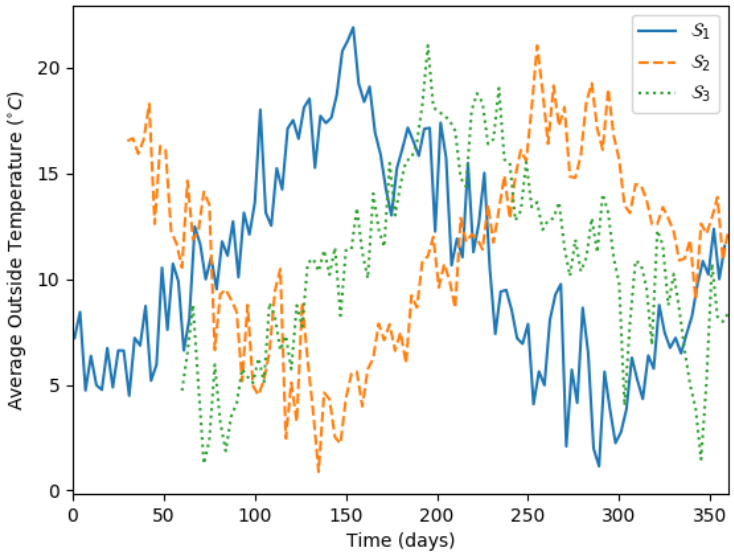


Figure 3.2: Samples of average daily temperatures within three heating simulator data streams.

3.4.1 Characteristics of ACC

An ACC system is an example of a real-world application that may collect data similar to the data streams contained in the following distance dataset. ACC typically lets the driver set what they consider to be a comfortable following distance. Vehicle manufacturers determine following distance using TTC, which calculates the following distance of the vehicle using its current speed, which can then be adjusted by a driver. A driver's preferred following distance may change over time or with changes in environment, such as traffic and weather conditions, introducing concept drift. Online TL could be used to calibrate a vehicle's ACC by combining knowledge learnt from other drivers. Types of drifts that could be encountered include geographical drifts induced by regional norms, sudden drifts induced by changes in road type or driving context, gradual drifts introduced by weather conditions and traffic, and recurring drifts where historical concepts repeatedly occur. Possible causes of concept drift when calibrating a personalised following distance for ACC include:

Driver: When deploying ACC an important factor to consider is the driver. Some drivers may feel comfortable following vehicles at a relatively close proximity, however, others may feel unsafe when following at the same distance. Sudden concept drift could be encountered if more than one person were to drive a

vehicle. Personalisation would be achievable if every driver was known to the vehicle, however, if an unknown driver were to use the vehicle, online TL could be used to enhance personalisations based on knowledge learnt from known drivers.

Location: Acceptable following distances differ from country to country but may also differ based on regional norms. For example, the preferred following distance of vehicles within highly populated areas, such as urban environments, may differ from rural areas. Geographical drifts such as these can occur both suddenly and gradually, however, it may be possible to enhance personalisations by transferring knowledge of the desired following distance learnt from other drivers and other vehicles.

Context and Purpose: The context and purpose of a journey may impact the following distance desired by a driver. Journey context, such as time of day, may greatly impact ACC customisation for an individual driver. For example, the time of day may be indicative of traffic conditions or visibility. The purpose of a journey may also impact personalisation, for example, the desired following distance when commuting may be different in comparison to a leisurely drive. Both context and purpose can cause sudden drifts from journey to journey, however, gradual drifts could also be observed throughout the duration of a single journey, particularly when travelling long distances. Additionally, drifts caused by context and purpose are likely to reoccur, and therefore the ability to transfer knowledge of historical concepts between drivers and vehicles may help to maintain effective customisations.

Road Type: ACC personalisation may largely be affected by speed, however, road type and layout will also be influential. For example, the desired following distances on country roads may be different to those wanted for motorway driving. Drifts induced by road type will typically occur suddenly with a high likelihood of recurring in the future.

External Conditions: Other external factors that may impact comfortable following distances include weather conditions and lighting conditions. Examples of concept drifts of this nature include preferring a larger following distance when travelling in heavy rain, or at night, in comparison to dry day-time driving where good light conditions improve visibility. Drifts caused by external conditions such as these can occur both suddenly and gradually, with a high probability of recurring in future journeys.

3.4.2 Data Collection

To evaluate online TL techniques the data collected for the following distance dataset encapsulated the driving behaviours of multiple participants upon different road types with varying environmental factors. As such, 2 drivers were asked to drive 3 scripted routes, and 2 drivers were asked to record data during daily commutes to and from the University of Warwick, UK.

The scripted routes were selected to include driving on motorways, duel carriageways and A roads where vehicle functionalities such as ACC are most suitable, as shown in Figure 3.3. The diversity of road types travelled induce concept drift as an individual's driving style naturally may change throughout the journey. Each of these routes were recorded at different times of day, since following distance is dependent on external and environmental factors such as traffic and weather. To capture a wider variety of driving conditions, 2 drivers were asked to record their daily commute, and 11 journeys were recorded in this manner, creating a total of 17 scripted and unscripted journeys. Each of the 17 journeys varied in duration, collection time and route, where the maximum, minimum and average durations were 83 minutes, 15 minutes and 43 minutes respectively. Each journey is considered to be an independent data stream, such that 6 data streams were generated by the 2 drivers driving 3 scripted routes[5], while the remaining 11 were generated by 2 drivers commuting[6]. Each data stream is subject to concept drifts that occur naturally due to changes in the surrounding environment, such as road types and traffic conditions.

In Chapters 4–7, a subset of 7 following distance data streams is also used. This subset contains vehicle telemetry data from one driver driving 2 scripted routes, while the remaining 5 data streams were collected from two drivers commuting. This smaller subset of data streams is representative of the dataset containing 17 data streams, and allows insights to be gained into the scalability of proposed online TL techniques by comparing these results to those obtained using all 17 journeys.

Using real-world data to evaluate online TL frameworks is important to ensure that the techniques developed can be used in real-world applications. However, the presence, frequency and types of concept drift cannot be artificially managed within this dataset. This means that there is no guarantee that knowledge learnt from one data stream will be beneficial in another. Since each following distance data stream captures the data for a single driver over a single journey, and a large

[5]This dataset is available at: `https://github.com/hmckay/BOTL/blob/master/FollowingDistanceData.zip`

[6]For privacy reasons the commute data cannot be released.

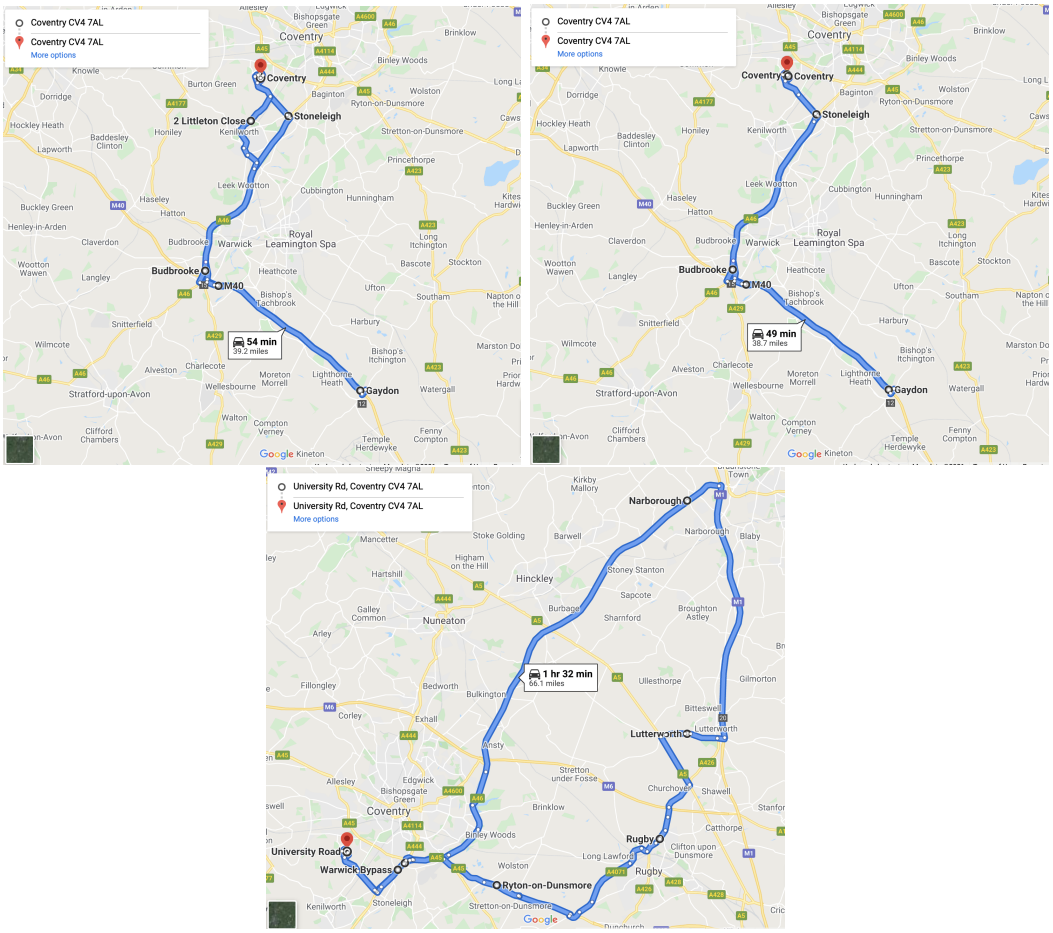


Figure 3.3: The three scripted routes used for following distance data collection.

proportion of journeys are recorded in similar geographical locations, the likelihood of encountering similar concepts in different data streams is high. However, since each data stream captures only a single journey, the likelihood of encountering recurring concepts is less likely than in the drifting hyperplane data streams, where recurring concepts are introduced artificially.

3.5 Summary

The datasets presented in this chapter can be used to evaluate concept drift detection strategies, online ensembles, and online TL frameworks in regression settings. Each of these datasets exhibit useful characteristics to evaluate the effectiveness of proposed approaches. The drifting hyperplane datasets allow online learning and online TL approaches to be evaluated in a controlled environment, where the pre-

cise number of concepts, point of concept drift, and similarity between concepts are known. This means that methods such as drift detection strategies can be evaluated with respect to their reactivity to sudden and gradual drift. Additionally, the underlying concepts are known in advance, and identical concepts are known to be present in different data streams, therefore online TL frameworks can be evaluated to ensure that knowledge transfer is beneficial since it is known that some knowledge available in one data stream will be beneficial to other data streams. However, since the drifting hyperplane datasets are synthetic, they do not provide a good indication of how online learning and online TL approaches will perform in real-world environments.

The smart home heating simulation dataset contains data streams that are synthetically generated based on real-world weather data. This provides a better indication of how online learning and online TL approaches will perform in real-world environments where the types of concept drift and occurrences of concept drift are unknown. However, since data streams are created from overlapping periods of time, there is still some control over the amount of beneficial knowledge that could be shared among data streams when using this dataset to evaluate the effectiveness of online TL approaches.

In the following distance dataset there is no control over the types or occurrences of concept drift. Additionally there is no guarantee that a data stream will contain knowledge that is useful to another data stream when evaluating online TL frameworks. This dataset allows online learning and online TL approaches to be evaluated using data collected from a real-world environment, therefore the performance observed using this dataset is indicative of what can be achieved when deploying online TL in real-world applications.

Overall, the drifting hyperplane datasets are useful for developing new strategies for online learning and online TL. This ensures that proposed approaches perform as expected within a controlled environment. The following distance dataset allows approaches to be evaluated where there is no control over the underlying distribution of data in each data stream. This provides insight into what might be expected when using online learning and online TL in real-world applications. The smart home heating simulation dataset also provides useful insight into how approaches are likely to perform in real-world environments where the existence of large quantities of real-world data is not available when developing online learning and online TL strategies.

Chapter 4

Determining What to Transfer

To address the challenge of determining what to transfer in online TL, where all domains are online, and where a data rich offline source domain does not exist, the knowledge that can be learnt from each online data stream must be considered. In Chapter 2, existing methodologies for learning in online environments have been discussed, including incremental learning, Concept Drift Detectors (CDDs), and ensembles. In this chapter, two existing CDDs are used to create predictive models that could be used to transfer knowledge between domains in an online TL framework, namely RePro [95], originally created for classification tasks, and ADWIN [8]. In this thesis, we consider the case where knowledge is learnt from regression based data streams, and so RePro has been adapted so that it can be used in regression settings. RePro and ADWIN both present characteristics that are beneficial in the context of determining what to transfer in online TL. As such, a novel CDD is presented, which combines key characteristics of RePro and ADWIN to create a CDD that is beneficial in online TL frameworks that may have computation and communication limitations.

The main contributions of this chapter are as follows.

- Adaptations of RePro and ADWIN are presented so that they can be used to create predictive models for regression tasks in online TL frameworks.
- A novel CDD, namely AWPro, is presented, which combines key characteristics of RePro and ADWIN that are beneficial in online TL.
- The impact of parameters, such as drift sensitivity and window size, required by RePro, ADWIN and AWPro to detect concept drifts are discussed, including considering how their values may impact the use of each CDD in online TL.

In this chapter, each CDD is used to create predictive models for the datasets presented in Chapter 3. Each CDD is dependent on user defined parameters, which requires domain expertise to select appropriate values. The impact of the user defined parameters on online learning is discussed, and their impact in the context of online TL is considered.

The remainder of this chapter is organised as follows. Section 4.1 discusses desirable characteristics when considering the challenge of determining what to transfer. Section 4.2 presents RePro, and the adaptations required to make it applicable in regression settings, Section 4.3 presents ADWIN and Section 4.4 presents AWPro. Sections 4.2, 4.3 and 4.4 also discuss the parameters of each CDD that are required to detect concept drifts. In Section 4.5 we consider the impact of CDD parameters and how they may impact the challenge of determining what to transfer in online TL.

4.1 Determining What to Transfer

When considering the challenge of what to transfer in online TL, we must consider how the knowledge available in a source domain can be captured and transferred such that the transfer of knowledge is beneficial to a receiving domain. Since the underlying distribution of data observed in a receiving domain is unknown to a source domain, we cannot infer whether knowledge learnt in the source domain will be beneficial to the receiver prior to transfer. However, in this thesis we consider the case where domains are homogeneous such that they share the same feature space, and each domain is learning independent but related tasks. Therefore, we assume that knowledge of the concepts encountered in a source domain may be beneficial to a receiving domain.

Knowledge can be learnt from an online source domain using online learning techniques. However, the choice of online learning technique is not only dependent on its effectiveness when predicting in its own domain, but also whether it is beneficial to a receiving domain. Additionally, in online TL, the communication and computation overhead of knowledge transfer must be considered to ensure that online TL can be used in real-world applications.

Ensemble learning has been shown to have great success in online learning, particularly on concept drifting data streams [30, 49, 72]. As discussed in Chapter 2, ensemble based approaches combine multiple predictive models that have been learnt historically throughout the data stream [31, 106]. However, the use of an ensemble learner in each online domain is not easily applicable when addressing the challenge

of determining what to transfer in online TL. This is because the predictions made by an ensemble depend upon the combination of multiple predictive models. This means that the knowledge learnt to represent a single concept encountered in a source domain may be encompassed across multiple models in the ensemble. Therefore, in order to use an ensemble as a method of knowledge transfer, all predictive models in the ensembles, and the weights that determine their influence with respect to a concept, must be transferred between domains. This may incur high communication overheads, but also could become computationally expensive since multiple ensembles, learnt to represent different concepts, may be transferred throughout an online TL framework, which must be combined in a receiving domain.

Instead, to reduce communication and computation overheads, it is desirable to transfer a single model between domains. This is the approach of many online TL frameworks, where the existence of a data rich offline source domain is assumed [27, 32, 88, 105]. However, if all domains are online and susceptible to concept drift, new knowledge is made available as each data stream progresses. In order to encapsulate this evolution of data, an incremental learner can be used [108]. As discussed in Chapter 2, incremental learners gradually update model parameters as new instances are observed, allowing models to adapt to concept drift [49]. Although the use of an incremental learner would reduce communication and computation overheads in comparison to an ensemble, knowledge learnt historically in a data stream may be lost due to incrementally updating model parameters [95]. This means that, in the presence of concept drift, knowledge of a previously encountered concept may be forgotten, and subsequently may have to be relearnt in the presence of recurring concepts. This is detrimental to online TL where all domains are online, since concept drifts are not guaranteed to occur synchronously between domains. Therefore, the knowledge learnt historically within each data stream may be of use to another domain.

Alternatively CDDs can be used [5], which are ideal for addressing the challenge of determining what to transfer in online TL. This is because they can be used to detect concept drift in an online data stream, allowing predictive models to be learnt that represent each of the concepts encountered in a domain. Although it is not known whether a receiving domain will encounter the same, or similar, concepts to those identified in a source domain, by transferring a single model that has been learnt to represent each concept encountered in a source domain, a receiving domain can then determine whether those models are useful for predicting its own current concept. Additionally, this approach allows receiving domains to be provided with knowledge of historical concepts encountered in a source domain in case a similar

Table 4.1: CDD parameters: notation and descriptions.

RePro	
W_{max}	Maximum window size: number of instances used to monitor accuracy and train predictive models.
λ_l	Loss threshold: ϵ-insensitivity for small margins of predictive errors.
λ_d	Drift threshold: determines sensitivity to concept drift and determines when historical models can be reused.
ADWIN	
W_{min}	Minimum window size: number of instances used to train predictive models.
δ	Confidence value: determines sensitivity to concept drift.
AWPro	
W_{min}	Minimum window size: number of instances used to train predictive models.
δ	Confidence value: determines sensitivity to concept drift.
λ_r	Recurrence threshold: determines when historical models can be reused.

concept is later encountered. Transferring knowledge in this way is beneficial due to requiring low communication overheads, since the knowledge transferred between domains consists only of the model parameters that have been learnt in a source domain to represent each concept.

Since CDDs allow new predictive models to be created each time a concept drift is encountered, knowledge learnt of historical concepts can be retained, and knowledge can be transferred periodically so that a receiving domain can benefit from the knowledge learnt in a source domain as new concepts are encountered, and without having to wait for model parameters to be incrementally updated [49].

Therefore, in this thesis, RePro [95], ADWIN [8], and AWPro [55], are used as CDDs to address the challenge of determining what to transfer in online TL. These CDDs are used to create predictive models that represent the concepts encountered in each data stream, which are then transferred to other online domains. However, in order to detect concept drifts, each CDD requires parameters to be defined that determine their sensitivity to concept drift, and to determine how much data is retained to create predictive models. Table 4.1 presents the notation used by RePro, ADWIN and AWPro with respect to these parameters. Since drift sensitivity and window size can impact what knowledge can be learnt and transferred throughout an online TL framework, the parameter values chosen must be considered carefully. In the ideal scenario, these parameters will allow a single predictive model to be

learnt for each concept encountered in a data stream, which is representative of the underlying distribution of that concept.

4.2 RePro

In this section, an adaptation of RePro [95] for regression is presented. RePro encapsulates key characteristics that allow few models to be learnt in each domain. First, RePro is a sliding window based detection algorithm that learns a single model to represent the current concept [94]. Second, RePro prioritises the reuse of existing models over learning new models. This is achieved by retaining a history of previously learnt models, H, and concept transitions, TM, to proactively determine which concept is likely to occur next [95]. However, RePro is dependent on three user defined parameters that impact its ability to detect concept drift, which must be specified in advance, namely the window size, W_{max}, a drift threshold, λ_d, and a loss threshold, λ_l.

RePro was initially developed specifically for classification tasks [95], therefore modifications are required for regression settings, as shown in Algorithm 2. The original RePro algorithm for classification tasks detects drifts by measuring the model's classification accuracy across the sliding window, W. The window is populated as new instances are observed in the data stream. When the window is full, if the oldest instance in the window has been correctly classified, then it is discarded from the window. If the window is full and the oldest instance captured within the window has been incorrectly classified, then the sliding window of instances, W, is evaluated. If the classification accuracy over the sliding window drops below a threshold, λ_d, a drift is detected. However, if the window is full, $|W| = W_{max}$, but the classification accuracy does not drop below the drift threshold, then the sliding window is maintained by discarding the oldest incorrectly classified instance and all subsequent correctly classified instances in the window [95]. To apply RePro to regression tasks, the sliding window must be maintained, however, the notion of a correctly classified instance must be altered since small inaccuracies are inevitable in regression settings due to noise. To overcome this, ϵ-insensitivity can be used, allowing for a small margin of error between the prediction and the response variable. To maintain a sliding window (Algorithm 2: lines 10 – 13), a loss threshold, λ_l, is used, which allows instance $x_{(t-|W|)}$ to be discarded from the window if the predicted value, $\hat{y}_{(t-|W|)}$ satisfies

$$|\hat{y}_{(t-|W|)} - y_{(t-|W|)}| \leq \lambda_l.$$

Algorithm 2: Adapted RePro for regression.

Input: W_{max}, λ_l, λ_d, X, $H = \emptyset$ (historical concepts), $TM = \emptyset$ (transition matrix)

1 Learn f_1 using $x_1 \dots x_{W_{max}}$, add to H
2 **for** $t = W_{max} + 1, W_{max} + 2, \dots$ **do**
3 Receive x_t and predict $\hat{y}_t = f_i(x_t)$
4 Receive y_t, add $\langle x_t, \hat{y}_t, y_t \rangle$ to W
5 **if** f_i *is new and stable* **then**
6 Add f_i to H and $f_{i-1} \rightarrow f_i$ to TM
7 **if** $R^2(f_i, W) < \lambda_d$ **then**
8 $f_{i+1} = \text{getNextModel}(H, TM, W)$ using Alg. 3
9 $W = \{\}$
10 **else if** $|W| \geq W_{max}$ **then**
11 Remove $x_{(t-|W|)}$ from W
12 **while** $|f_i(x_{(t-|W|)}) - y_{(t-|W|)}| \leq \lambda_l$ **do**
13 Remove $x_{(t-|W|)}$ from W

Algorithm 3: Model selection and creation for RePro. For additional details see [95].

Input: H, TM, W, λ_d

1 f_{i+1} = next model in TM
2 **if** $R^2(f_{i+1}, W) \geq \lambda_d$ **then**
3 **return** f_{i+1}
4 f_{i+1} = best performing model in H
5 **if** $R^2(f_{i+1}, W) \geq \lambda_d$ **then**
6 **return** f_{i+1}
7 Learn f_{i+1} using W
8 **return** f_{i+1}

The R^2 performance of the predictive model, f_i, across W is used to detect drifts (Algorithm 2: line 7). A drift is said to have occurred when the performance of the model drops below a predefined drift threshold, λ_d, akin to observing the classification accuracy dropping below an error threshold.

The original formulation of RePro used the notion of a stable learning size, specifying how much data is required to learn a stable model. This was necessary for the simulated classification tasks presented by Yang *et al.* [95] since small window sizes were required to allow drifts to be detected quickly. However, this meant that insufficient instances were available in the sliding window that could be used to learn a predictive model that adequately represented the current concept [94]. To overcome this Yang *et al.* initially create predictive models using W_{max} instances,

however, once a further $2W_{max}$ instances have been observed, the predictive model is relearnt using the previous $3W_{max}$ instances of data. This predictive model is then deemed to be stable [95]. Since real-world environments are often considerably more noisy than the simulated or synthetic environments used by Yang *et al.*, using a small window size can cause drifts to be falsely detected, and therefore a larger window size is necessary [8]. Increasing the window size also increases the stable learning size, however, if the stable learning size is increased, the data used to create a model may encapsulate multiple underlying concepts. To overcome this challenge, the adaptation of RePro for regression defines a stable model to be one that is learnt from W_{max} instances and is used to make predictions over a further $2W_{max}$ of previously unseen instances in the data stream without a drift being detected.

To proactively determine future concepts, RePro maintains a transition matrix, TM, to determine the likelihood of encountering a recurring concept. To prevent the reuse of unstable models that make poor predictions, only those that are considered to be stable are added to the transition matrix. If the transition matrix indicates that it is equally likely that two or more concepts may be encountered next, RePro evaluates the performance of each model on the current window of data and selects the model with the highest accuracy. If the transition matrix does not indicate a likely successor concept, each historical model is considered for reuse. A new model is only learnt when all historical models perform worse than the drift threshold, λ_d, as shown in Algorithm 3.

Parameter Selection

The characteristics of RePro as a drift detection strategy are desirable for online TL, however, the selection of parameter values, W_{max}, λ_d, and λ_l, may not be intuitive. Given an online data stream, trade-offs must be considered for each parameter. For example, selecting a large window size, W_{max}, allows more data to be retained to train the predictive models, increasing their accuracy and stability [7]. However, a window size that is too large may cause RePro to react slowly to concept drifts, and may retain data from multiple concepts, preventing a model from being created to represent each concept independently. Alternatively, using a small window size may allow RePro to react quickly to drifts, but selecting a window size that is too small may prevent a representative sample from being retained to build a model that generalises well for unseen instances belonging to the same concept.

The drift and loss thresholds, λ_d and λ_l, determine RePro's sensitivity to concept drift. Small drift thresholds and large loss thresholds decrease RePro's sensitivity to concept drifts, since small λ_d values allow the performance of a model

to greatly decrease before a drift is detected, while large λ_l values allow instances to be removed from the sliding window when a model's predictive error is high.

Large λ_d and small λ_l values increase RePro's sensitivity to drifts. As RePro becomes more sensitive to concept drifts, it also becomes more likely that a drift is detected before a full window of instances belonging to the new concept have been observed. This means that the window of data, W, used to build a model for the newly encountered concept may contain instances belonging to the previous concept. Models built using data belonging to both the previous and new concepts may not make effective predictions as more instances belonging to the new concept are observed. Since RePro monitors the model performance to detect drifts, this can cause RePro to repeatedly detect drifts and create unstable models immediately after a concept drift, and during periods of gradual drift. The repeated creation of unstable models increases computation in each data stream, however, these models are not added to the transition matrix, TM, or the model history, H, and therefore do not greatly impact the overarching performance of RePro across the data stream, and do not impact the communicational overhead of knowledge transfer when only models considered to be stable are transferred. Due to this, selecting values for λ_d that are too large, and values for λ_l that are too small, may prevent RePro from creating stable models that can be added to the model history, since drifts may be falsely detected frequently in the presence of noise.

4.3 ADWIN

ADWIN, presented by Bifet *et al.* [8], detects drifts by monitoring changes in the distribution of a data stream. For use in a regression setting, the distribution of predictive error is monitored across a sliding window. Instead of using a fixed length sliding window, the size of the window is determined according to the rate of change of predictive error observed in the data stream [8].

ADWIN operates on the principal that if two large enough sub-windows have distinct enough mean values, then the expected values within each sub-window will differ [8, 26]. As such, a drift is said to be detected when

$$|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| \geq \epsilon_{\text{cut}}, \tag{4.1}$$

where $\hat{\mu}_{W_0}$ and $\hat{\mu}_{W_1}$ are the mean values of sub-windows W_0 and W_1, and ϵ_{cut} is

Algorithm 4: ADWIN. For full implementation details see [8].

Input: W_{min}, X, δ (confidence value), driftFlag = False.

1 Learn f_1 using $x_1 \dots x_{W_{min}}$
2 adwin = new ADWIN(δ)
3 **for** *each* $t = W_{min} + 1, W_{min} + 2, \dots$ **do**
4 Receive x_t and predict $\hat{y}_t = f_i(x_t)$
5 Receive y_t, and calculate diff = $(\hat{y}_t - y_t)$
6 Add $\langle x_t, \hat{y}_t, y_t, \text{diff} \rangle$ to W
7 **if** ***not*** *driftFlag* **then**
8 driftFlag, W_0, W_1 = adwin.detectDrift(W) (using Equ. 4.1 & 4.2)
9 **if** *driftFlag* **then**
10 $W = W_1$
11 **if** *driftFlag and* $|W| \geq W_{min}$ **then**
12 Learn f_{i+1} using $W = \{x_{t-W_{min}}, \dots, x_t\}$
13 driftFlag = False
14 adwin = new ADWIN(δ)

defined by the Hoeffding bound

$$\epsilon_{\text{cut}} = \sqrt{\frac{1}{2m} \cdot \ln \frac{4|W|}{\delta}}, \tag{4.2}$$

where m is the harmonic mean of the sub-windows, $m = \frac{2}{1/|W_0| + 1/|W_1|}$, and δ is a confidence value, defined by the user, which determines the sensitivity of drift detection [8, 26].

ADWIN can be used in online TL, as presented in Algorithm 4, to detect drifts within the data stream by monitoring the distribution of the predictive error of the model learnt from the data stream, f_i,

$$|f_i(x_t) - y_t|.$$

Once a drift is detected, a new model is learnt, f_{i+1}, using the subsequent sub-window such that $W = W_1$ (Algorithm 4: lines 8 – 14). Monitoring the distribution of predictive error allows drifts to be detected rapidly. However, if two consecutive concepts are dissimilar, then drifts are frequently detected when only a small number of instances from the new concept have been observed, and therefore the data contained in the sliding window, W, after a drift is detected is unlikely to be representative of the newly encountered concept. To address this, the implementation of ADWIN presented in this section only creates a new model, f_{i+1}, once W_{min} instances belonging to the new concept have been observed, such that $|W| = W_{min}$

(Algorithm 4: lines 11 – 14). Until sufficient data has been observed to build a model that adequately represents the new concept, the previously learnt model, f_i, must continue to be used to make predictions.

Parameter Selection

Using ADWIN in this way requires two user defined parameters, the minimum window size, W_{min}, and the confidence value, δ. Similarly to RePro, domain expertise is required to select these parameter values. Since ADWIN uses a dynamic sliding window, the minimum window size parameter, W_{min}, does not directly impact ADWIN's ability to detect concept drifts. Instead, W_{min} is only used to determine how much data should be retained to build a model that adequately represents the current concept [7]. Similar to RePro's W_{max} parameter, large values of W_{min} allow more data to be made available to train the predictive model, f_i, creating an accurate and stable model [7]. However, if W_{min} is too large, then the data contained within the window may encapsulate data from multiple concepts, preventing individual models being learnt for each concept, and if W_{min} is too small, then the data used to build a model may not be representative of the current concept, resulting in predictive models that do not generalise well.

ADWIN detects drifts by monitoring the distribution of predictive error, and therefore W_{min} can indirectly effect ADWIN's ability to correctly detect concept drifts and the overall predictive performance. If W_{min} is too small then it may cause a model to be learnt that overfits the available window of data, and obtains high predictive errors for unseen instances belonging to the same concept. Since ADWIN monitors the change in distribution of predictive error, using a model that initially has a high predictive error may prevent or delay the detection of a concept drift because no significant change in the distribution of predictive error is observed during periods of concept drift. However, large W_{min} values increase the number of instances observed before a new model can be learnt, prolonging the use of the previously learnt model, and decreasing the predictive performance observed throughout the data stream.

The confidence value, δ, is used to determine ADWIN's sensitivity to concept drifts through the ϵ_{cut} threshold (Equations 4.1 and 4.2). High values of δ increase drift sensitivity, however, in noisy data streams, this can cause drifts to be falsely detected due to the increased variability of sub-window mean values, $\hat{\mu}_{W_0}$ and $\hat{\mu}_{W_1}$. To overcome this, lower values of δ can be chosen, however, this may prevent drifts from being detected in domains containing similar consecutive concepts, or slow gradual drifts, where the sub-window mean values do not change greatly.

4.4 AWPro

The use of RePro as a concept drift detection algorithm can be computationally demanding due to the creation of high volumes of unstable models, caused by its inability to detect the precise point of drift within the sliding window. ADWIN allows the point of drift to be estimated, such that the sliding window can be split into two sub-windows, where the first sub-window contains instances belonging to the previous concept, which are discarded, while the second sub-window contains instances belonging to the new concept. However, if the number of remaining instances in the second sub-window is small, then ADWIN must wait until sufficient instances have been observed before a new model can be learnt, negatively impacting its performance. Additionally, ADWIN does not consider the presence of recurring concepts, and therefore predictive models are re-learnt instead of reusing a historical model when a concept reoccurs. This prevents previously learnt models from being used to make predictions when few instances from a recurring concept have been observed, and increases the number of models transferred between domains when ADWIN is used in online TL.

To reduce the computational overhead of creating of large numbers of unstable models, while also preventing duplicate models from being learnt for recurring concepts, a novel CDD, called Adaptive Windowing with Proactive predictions (AWPro) is presented, as shown in Algorithm 5, which combines desirable characteristics of ADWIN and RePro which are beneficial in the context of online TL. AWPro uses ADWIN to monitor the change in distribution of predictive error, allowing the drift detection strategy to partition instances belonging to different concepts within a dynamic sliding window. Concept drifts are identified using Equations 4.1 and 4.2, which use a confidence value, δ, to determine the sensitivity to changes in the distribution of the predictive error (Algorithm 5: line 9). Once a model has been learnt and is considered to be stable, it is added to the model history, H, and the transition between concepts is added to the transition matrix, TM (Algorithm 5: lines 26 – 27). As in RePro, a stable model is one that is used to make predictions over $2W_{min}$ instances without a drift being detected.

When a concept drift is encountered, AWPro drops the first sub-window of instances, W_0, such that all instances that remain in the window belong to the new concept, $W = W_1$. If the remaining window of data is less than $\frac{1}{2}W_{min}$ instances, then the previous model continues to be used, and newly observed instances are added to the window until $\frac{1}{2}W_{min}$ have been observed (Algorithm 5: lines 12 – 13).

When $\frac{1}{2}W_{min}$ or more instances are captured in W, the proactive nature of

Algorithm 5: AWPro.

Input: W_{min}, X, δ (confidence value), driftFlag = False, λ_r, $H = \emptyset$ (historical concepts), $TM = \emptyset$ (transition matrix)

driftFlag, tempModel = False
Learn f_1 using $x_1 \dots x_{W_{min}}$, add to H
adwin = new ADWIN(δ)
for *each* $t = W_{min} + 1, W_{min} + 2, \dots$ **do**
    Receive x_t and predict $\hat{y}_t = f_i(x_t)$
    Receive y_t, and calculate diff = $(\hat{y}_t - y_t)$
    Add $\langle x_t, \hat{y}_t, y_t, \text{diff} \rangle$ to W
    if *not* *driftFlag* **then**
        driftFlag, W_0, W_1 = adwin.detectDrift(W) (using Equ. 4.1 & 4.2)
        if *driftFlag* **then**
            $W = W_1$
    if *driftFlag and* $|W| < \frac{1}{2}W_{min}$ **then**
        Continue using f_i or f_{temp}, add x_t to W
    else if *driftFlag and* $\frac{1}{2}W_{min} \leq |W| < W_{min}$ *and* ***not*** *tempModel* **then**
        if $R^2(getNextModel(H, TM, W, \lambda_r, W_{min}), W) \geq \lambda_r$ **then**
            f_{i+1} = getNextModel($H, TM, W, \lambda_r, W_{min}$) using Alg. 6
            driftFlag,tempModel = False
            adwin = new ADWIN(δ)
        else
            tempModel = True
            Learn f_{temp} using W
    else if *driftFlag and* $|W| \geq W_{min}$ **then**
        f_{i+1} = getNextModel($H, TM, W, \lambda_r, W_{min}$) using Alg. 6
        driftFlag,tempModel = False
        adwin = new ADWIN(δ)
    if f_i *is new and stable* **then**
        Add f_i to H and $f_{i-1} \rightarrow f_i$ to TM

Algorithm 6: Model selection and creation for AWPro.

Input: $H, TM, W, \lambda_r, W_{min}$

1 f_{i+1} = next model in TM
2 **if** $R^2(f_{i+1}, W) \geq \lambda_r$ **then**
3 **return** f_{i+1}
4 f_{i+1} = best performing model in H
5 **if** $R^2(f_{i+1}, W) \geq \lambda_r$ **then**
6 **return** f_{i+1}
7 **if** $|W| \geq W_{min}$ **then**
8 Learn f_{i+1} using $W = \{x_{t-W_{min}}, \ldots, x_t\}$
9 **return** f_{i+1}

RePro is used by AWPro to determine whether an existing model can be reused to represent the current concept (Algorithm 5: lines 14 – 18). This allows AWPro to identify an existing model, using Algorithm 6, that represents the current concept, before a full window of instances have been observed after the concept drift. AWPro uses a recurrence threshold that determines whether an existing model can be reused, λ_r, which acts in the same way as the drift threshold, λ_d, in RePro, when considering the reuse of existing models. If a model's R^2 performance is greater than the recurrence threshold, λ_r, then it is reused, and the process of detecting concept drifts through monitoring changes to the distribution of predictive error is resumed. However, if no model exists, then a temporary model is created using W, where $\frac{1}{2}W_{min} \leq |W| < W_{min}$. Although these temporary models are akin to unstable models learnt using RePro, the window used to build these models only contains instances belonging to the new concept. Having few instances available increases the likelihood of learning a model that is not representative of the entire concept. However, this may be preferable to ADWIN's approach of continuing to use a model that represents the previous concept, or RePro's approach where a model may be learnt from data belonging to both concepts. Once a temporary model has been created, instances continue to be added to the window until W_{min} instances belonging to the new concept have been observed. Once the size of the window grows to W_{min}, AWPro uses the transition matrix and historical models to identify existing models that could be reused now a more representative sample of data is contained within the window. If no existing model exceeds the recurrence threshold, λ_r, then a new model is learnt using the W_{min} most recently observed instances (Algorithm 5: lines 22 – 25, and Algorithm 6).

Parameter Selection

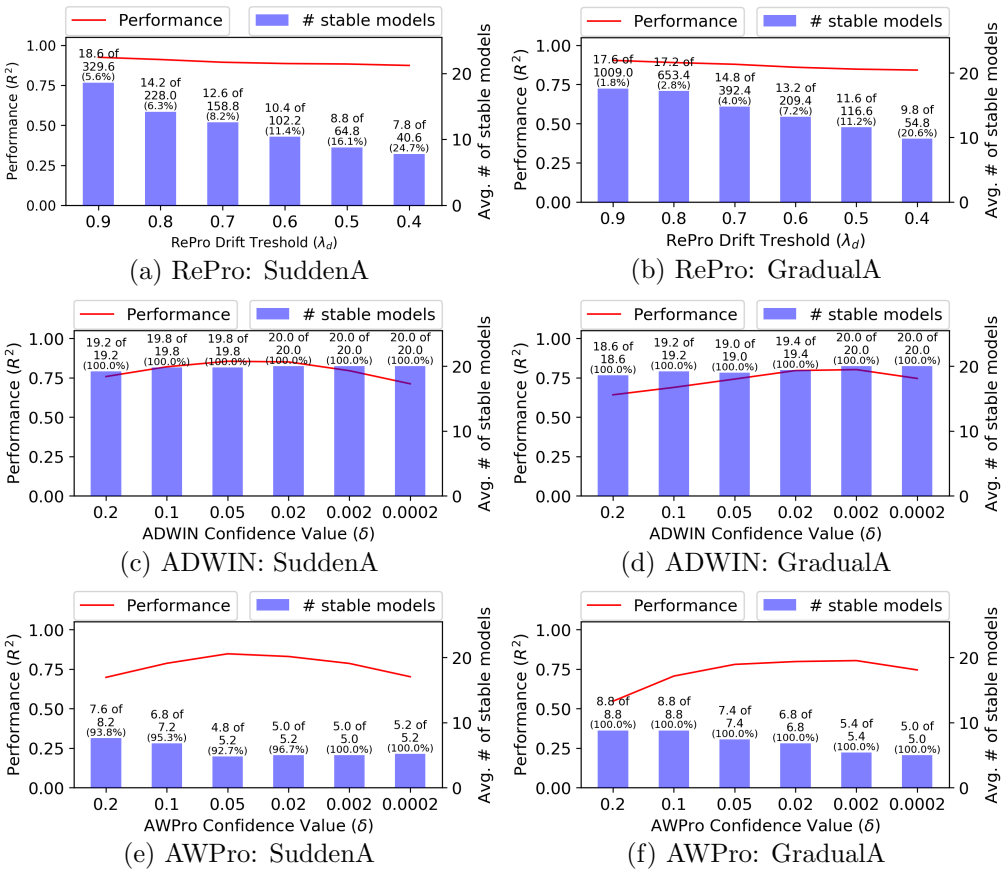
AWPro relies on three user defined parameters: the confidence value, δ, the window size, W_{min}, and the recurrence threshold, λ_r. Since AWPro adopts ADWIN's approach to detecting concept drifts, many of the challenges of parameter selection outlined in Section 4.3 with respect to the confidence value, δ, and the window size, W_{min}, are also applicable to AWPro. However, instead of waiting for W_{min} instances to be observed after a drift is detected, AWPro uses the recurrence threshold, λ_r, to determine whether an existing model can be reused.

If large values are chosen for λ_r, then the likelihood of reusing a historical model decreases, since a historical model must exhibit low predictive error in order to be selected for reuse. Therefore, high λ_r values will increase the number of models learnt by AWPro. To increase the reuse of models in the presence of recurring concepts smaller values should be chosen for λ_r. However, if λ_r is too small, then an existing model may be incorrectly reused for a concept that has not previously been encountered, lowering the predictive performance of AWPro. This may also hinder the detection of concept drifts since the predictive error across the sliding window of data may initially be high if a historical model is reused which does not represent the current concept, and therefore identifying concept drifts through monitoring changes to the distribution of predictive error becomes challenging.

4.5 Impact of Parameter Values

In order to investigate how online TL frameworks may be impacted by the parameters used by RePro, ADWIN and AWPro, each CDD is used to detect concept drifts and create predictive models for data streams in the datasets presented in Chapter 3, namely the sudden and gradual drifting hyperplane datasets, the heating simulator dataset and the following distance dataset. The performance of each CDD and the number of stable and unstable models created by each CDD are evaluated to consider the implications of their use in online TL. The parameter values chosen for each CDD should be selected with the aim of maximising the performance of the predictive models created, while reducing both the number of stable and unstable models. These aims ensure that the transfer of knowledge can be beneficial to a receiving domain, while minimising unnecessary computation, and reducing communication overheads.

The first parameter to be considered is the loss threshold, λ_l, used by RePro, which determines how close a prediction must be to the response variable in order for it to be discarded from the sliding window. Since the hyperplane datasets are



(a) RePro: SuddenA (b) RePro: GradualA

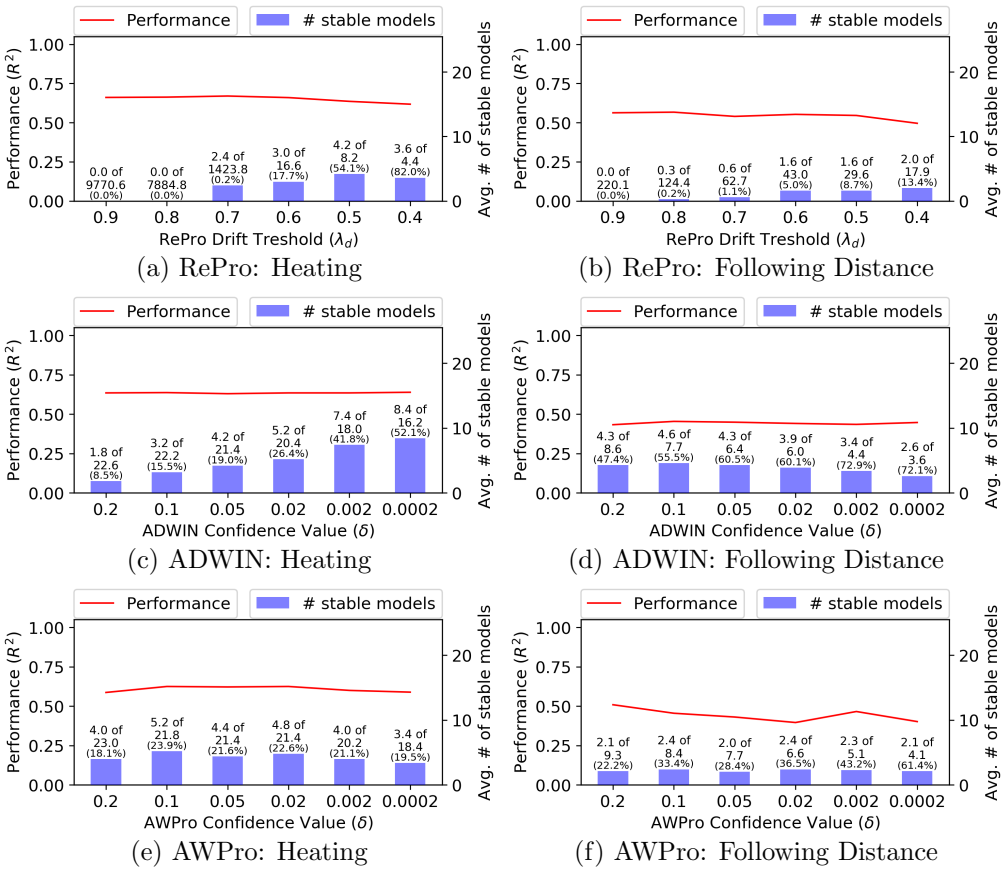
(c) ADWIN: SuddenA (d) ADWIN: GradualA

(e) AWPro: SuddenA (f) AWPro: GradualA

Figure 4.1: Performance and number of stable models created by RePro, ADWIN and AWPro with varying drift sensitivities for the SuddenA and GradualA drifting hyperplane datasets using a window size of 30 instances. Annotated with the total number of models learnt and percent that are considered stable.

synthetic, and the response variables are in the range [0,1], a loss threshold of $\lambda_l = 0.01$ is used, allowing for a 1% error in predictions. The heating simulator and following distance datasets do not have a definitive range for their respective response variables, and therefore a percentage of error cannot be used. Since these datasets require predictions to be made for user facing applications, namely a smart home heating system and ACC, λ_l was selected by considering errors that would not be noticeable to a user. For the heating simulator dataset a loss threshold $\lambda_l = 0.5$ was used, since a prediction error of 0.5°C would not be detectable by an individual. Similarly, a loss threshold of $\lambda_l = 0.1$ was used for the following distance dataset, allowing for a predictive error of 0.1 seconds. These loss thresholds are used by RePro throughout the remainder of this thesis.

Drift sensitivities and window sizes must also be chosen for RePro, ADWIN



(a) RePro: Heating (b) RePro: Following Distance

(c) ADWIN: Heating (d) ADWIN: Following Distance

(e) AWPro: Heating (f) AWPro: Following Distance

Figure 4.2: Performance and number of stable models created by RePro, ADWIN and AWPro with varying drift sensitivities for the heating simulator and following distance datasets using a window sizes of 480 for the heating simulator dataset (capturing instances across a period of 10 days), and 90 for the following distance dataset. Annotated with the total number of models learnt and percent that are considered stable.

and AWPro. To consider how predictive performance and the number of models created by each CDD are affected by parameter values, both the drift sensitivity and the window size have been varied for each CDD.

Figures 4.1 and 4.2 display the results of varying drift sensitivities for each CDD[1]. These results are obtained using a fixed window size for each dataset type, representative of the results presented in Figures 4.3 and 4.4, which were obtained by varying window size. Figure 4.1 uses a window size of 30 instances for the drifting hyperplane datasets, and Figure 4.2 uses window sizes of 480 instances for

[1]For Figures 4.1 and 4.2, all plots show parameter values with decreasing sensitivity to concept drift on the x-axis, with the most sensitive parameter value on the left of each plot, and the least sensitive on the right.

the heating simulator dataset, and 90 instances for the following distance dataset.

Figures 4.1a, 4.1b, 4.2a and 4.2b present the results obtained by varying RePro's sensitivity to concept drift using λ_d in the sudden drifting hyperplane, gradual drifting hyperplane, smart home heating simulation and following distance datasets respectively. These results indicate that higher drift sensitivity values used by RePro typically obtain a higher performance across all dataset types, however, the number of unstable models is also larger, and very few stable models are created. Lowering the drift sensitivity of RePro introduces a slight decrease in performance but significantly reduces the number of unstable models, and increases the proportion of stable models. Therefore, a trade-off between performance and the number of models created is necessary when using RePro as a CDD.

If the sole concern is the performance of the underlying CDD, then RePro obtains the best performance overall. RePro outperforms ADWIN and AWPro in Figures 4.1 and 4.2, due to its drift detection mechanism. Unlike ADWIN (Figures 4.1c, 4.1d, 4.2c and 4.2d) and AWPro (Figures 4.1e, 4.1f, 4.2e and 4.2f), RePro monitors the predictive performance of the current model, and detects drifts when its performance drops below the drift threshold, λ_d. This means that poorly performing models are replaced as new instances of data are observed, until a model that achieves a performance greater than λ_d is learnt. ADWIN and AWPro only monitor the distribution of predictive error, regardless of how poorly the model performs, and therefore a poorly performing model will only be replaced when a change in the distribution of predictive error is observed.

Although RePro uses a sliding window to capture the most recent instances, it does not estimate the precise point a drift occurs within the window. This means that RePro frequently builds models from windows containing instances belonging to both the previous and new concept, causing unstable models to be learnt. Since RePro monitors the performance of each model, unstable models are often created in quick succession during gradual drifts, or immediately after sudden drifts. This is highlighted in the annotations in Figures 4.1a, 4.1b, 4.2a and 4.2b, which show the percentage of models that were considered stable and useful for knowledge transfer. RePro creates significantly more unstable models, wasting computation that may mean it is impractical in real-world environments with limited computational resources. ADWIN and AWPro may be more applicable in these environments, since they estimate the point of drift using a dynamic sliding window, therefore reducing the number of unstable models. This is illustrated by considering the drifting hyperplane datasets presented in Figure 4.1. As discussed in Chapter 3, each drifting hyperplane data stream contains 5 concepts that occur 4 times throughout the

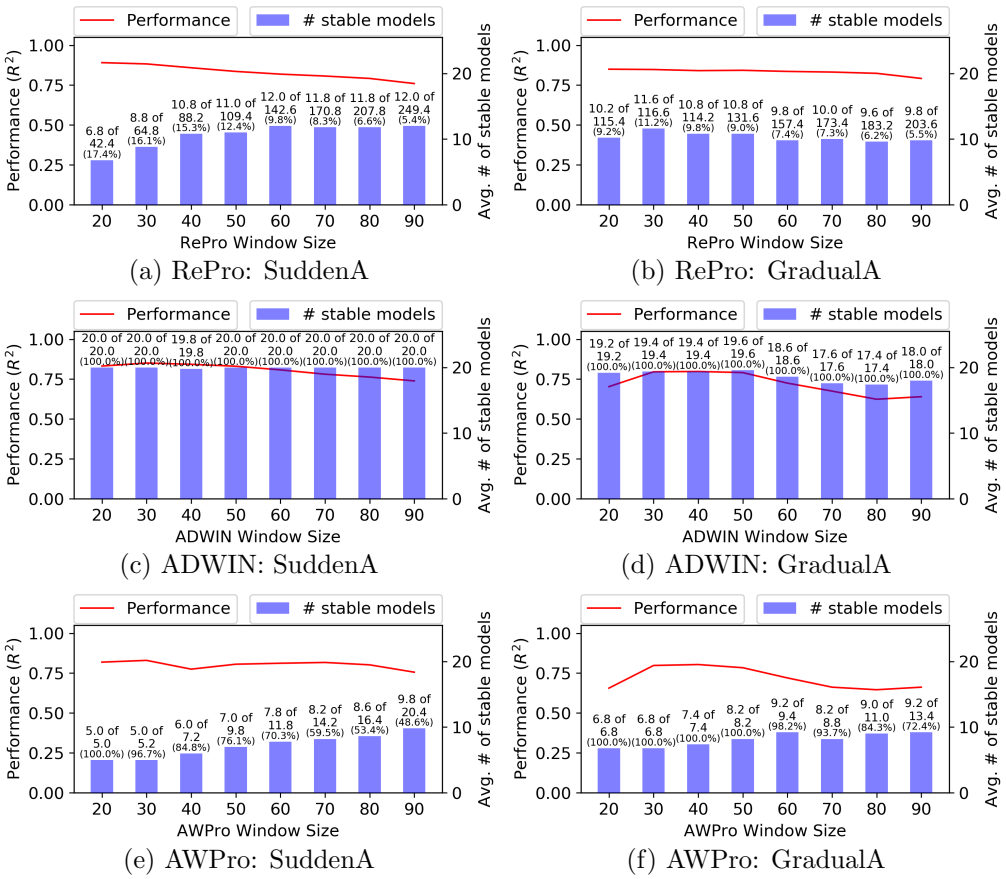
stream. ADWIN identifies drifts effectively, since close to 20 models are created across all drift sensitivity values for the sudden and gradual drifting hyperplane datasets in Figures 4.1c and 4.1d respectively. When evaluating RePro using the drifting hyperplane datasets, Figures 4.1a and 4.1b highlight the high frequency in which unstable models are created by RePro. For example, in Figure 4.1b, nearly 1000 unstable models are created per data stream in the GradualA dataset when a drift threshold of $\lambda_d = 0.9$ is used. Although ADWIN creates fewer unstable models in the drifting hyperplane datasets (Figures 4.1c and 4.1d), it cannot reuse existing models in the presence of recurring concepts, and therefore a larger number of stable models are learnt in comparison to AWPro (Figures 4.1e and 4.1f).

AWPro combines ADWIN's drift detection strategy with RePro's ability to prioritise the reuse of existing models. Therefore, AWPro creates very few unstable models, and creates close to 5 predictive models over a variety of drift sensitivity parameter values for the sudden and gradual drifting hyperplane datasets, as shown in Figures 4.1e and 4.1f respectively. These 5 predictive models correspond to the 5 independent concepts encountered in the drifting hyperplane datasets, indicating that AWPro is able to successfully identify recurring concepts in these datasets. This reduces communication overheads and reduces the risk of overfitting introduced by transferring multiple models that have been learnt to represent the same concept, making AWPro more applicable to online TL where computation and communication resources may be restricted.

By considering performance and the number of stable and unstable models learnt, a drift sensitivity value of $\lambda_d = 0.5$ is selected for RePro since the performance obtained remains high across the drifting hyperplane datasets in Figures 4.1a and 4.1b, and in the heating simulation and following distance datasets in Figures 4.2a and 4.2b. Additionally, the percentage of models learnt that are considered stable increases compared to using drift sensitivity values of $\lambda_d = 0.6$ and $\lambda_d = 0.7$.

The percentage of models learnt by ADWIN that are considered stable varies little across drift sensitivity values within the hyperplane datasets, however, a drop in performance is observed when $\delta > 0.02$ and $\delta > 0.002$ for sudden and gradual drifting datasets respectively, as shown in Figures 4.1c and 4.1d. Additionally, an increase in the percentage of models that are considered stable is observed in Figure 4.2c for the heating simulation dataset for $\delta < 0.02$, while no significant change in performance or number of stable or unstable models is observed in the following distance datasets in Figure 4.2d. Therefore, a drift sensitivity value $\delta = 0.02$ is selected for ADWIN.

Since AWPro is based upon the drift detection mechanism used by ADWIN,



(a) RePro: SuddenA (b) RePro: GradualA

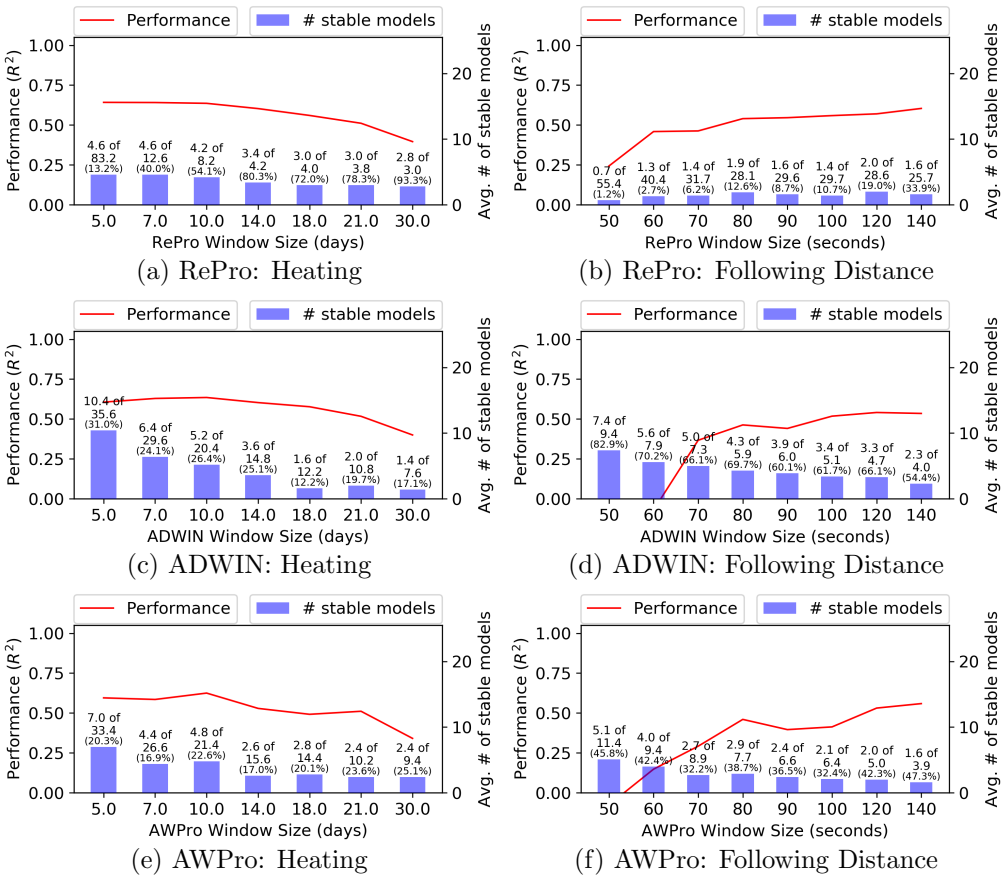
(c) ADWIN: SuddenA (d) ADWIN: GradualA

(e) AWPro: SuddenA (f) AWPro: GradualA

Figure 4.3: Performance and number of stable models created by RePro, ADWIN and AWPro with varying window sizes for the SuddenA and GradualA drifting hyperplane datasets using drift sensitivities of $\lambda_d = 0.5$ for RePro, and $\delta = 0.02$ for ADWIN and AWPro. Annotated with the total number of models learnt and percent that are considered stable.

observations in changes to performance with varying drift sensitivities are similar, as shown in Figures 4.1e, 4.1f, 4.2e and 4.2f. To enable fair comparisons, a drift sensitivity value $\delta = 0.02$ is also chosen for AWPro.

In addition to the impact of drift sensitivity, the window size parameter must be defined for RePro, ADWIN and AWPro. Figures 4.3 and 4.4 present the results obtained when varying window size for each CDD. This parameter determines how much data is made available to learn a new predictive model after a drift has been identified, and how much data is retained in order to detect drifts in the case of RePro. The results presented in Figures 4.3 and 4.4 use fixed drift sensitivity values for each CDD, which are representative of the results presented in Figures 4.1 and 4.2, and displayed in Table 4.2.



(a) RePro: Heating (b) RePro: Following Distance

(c) ADWIN: Heating (d) ADWIN: Following Distance

(e) AWPro: Heating (f) AWPro: Following Distance

Figure 4.4: Performance and number of stable models created by RePro, ADWIN and AWPro with varying window sizes for the heating simulator and following distance datasets using drift sensitivities of $\lambda_d = 0.5$ for RePro, and $\delta = 0.02$ for ADWIN and AWPro. Annotated with the total number of models learnt and percent that are considered stable.

Across synthetic sudden and gradual drifting hyperplane datasets, presented in Figure 4.3, RePro, ADWIN and AWPro maintain similar ratios of stable to unstable models, regardless of window size, however, the performance of each CDD typically decreases as the window size increases. This is observed since the underlying concepts to be learnt in the drifting hyperplane datasets are simplistic, therefore effective predictive models can be learnt from little data. Increasing the window size decreases predictive performance in these datasets since larger window sizes delay drift detection. However, a significant increase in performance is observed as the window size increases for each CDD for the real-world following distance dataset in Figures 4.4b, 4.4d and 4.4f since the concepts to be learnt are more complex, and therefore require more data to be made available to build a predictive model

	RePro			ADWIN		AWPro		
	W_{max}	λ_d	λ_l	W_{min}	δ	W_{min}	δ	λ_r
SuddenA	30	0.5	0.01	30	0.02	30	0.02	0.5
GradualA	30	0.5	0.01	30	0.02	30	0.02	0.5
Heating	10 days	0.5	0.5°C	10 days	0.02	10 days	0.02	0.5
Following	90s	0.5	0.1s	90s	0.02	90s	0.02	0.5

Table 4.2: Window size and drift sensitivity parameters used by RePro, ADWIN, and AWPro to obtain results presented in Chapters 5–7.

that effectively represents the current concept [7]. Figures 4.3 and 4.4 highlight that the performance and ratio of stable to unstable models is affected by the window size. However, all CDDs perform similarly for each dataset, indicating that the window size is dependant on the data stream to be learnt from, rather than the drift detection strategy. From the results presented in Figure 4.3, window sizes of 30 instances are selected for the sudden and gradual drifting hyperplane datasets, whereas window sizes of 480 instances for smart home heating simulator dataset, which encapsulates 10 days of observations, and 90 instances for the following distance dataset, which encapsulates 90 seconds of observations are chosen based on the results presented in Figure 4.4.

AWPro has an additional parameter, λ_r, which determines whether a historical model can be reused. Since it is used to identify recurring concepts in a similar way to how λ_d is used by RePro, the parameter value, λ_r, for AWPro has been selected based on the analysis of RePro in Figures 4.1 –4.4 to allow fair comparisons between RePro and AWPro.

4.6 Summary

In order to determine what to transfer in online TL frameworks we must consider what information can be learnt in each online domain that may be beneficial to other domains. In frameworks where all domains are online and a data rich offline source domain does not exist, CDDs can be used to identify each of the concepts encountered in a domain. Knowledge of these concepts can be transferred between domains by transferring the parameters of predictive models that have been learnt to represent each of the concepts. Transferring knowledge in this way has a reduced communication overhead in comparison to instance based transfer, which may be beneficial to applications of online TL that have limited communication bandwidth to transfer knowledge between domains.

In this chapter, RePro, ADWIN and AWPro have been chosen as CDDs to learn predictive models, however, other CDDs can also be used, including ensemble based approaches. RePro, ADWIN and AWPro have key characteristics that are beneficial when used in online TL frameworks. For example, a key characteristic common to RePro, ADWIN and AWPro, is that a single predictive model is learnt to represent each of the concepts encountered in a data stream. This means that there is a lower communication overhead associated with knowledge transfer in comparison to ensemble based concept drift detection strategies, where the representation of a single concept may be encapsulated across multiple predictive models in the ensemble.

Another key characteristic, exhibited by RePro and AWPro, is the reuse of historical models in the presence of recurring concepts. Reusing historically learnt models reduces the computational overhead of having to unnecessarily relearn predictive models and reduces communication overheads when transferring knowledge of a recurring concept. Additionally, a key characteristic exhibited by ADWIN and AWPro is that the precise point of concept drift is estimated within a dynamic sliding window. This means that instances belonging to a previous concept are discarded from the sliding window before a predictive model is learnt to represent a newly encountered concept. Therefore, the knowledge learnt to represent each of the concepts is not influenced by the preceding concept.

In order to use RePro, ADWIN and AWPro as CDDs, user defined parameters must be used to determine their sensitivity to concept drifts and the amount of data available to learn predictive models. Section 4.5 has shown that these parameter values are dependent on the data streams in which concept drifts are to be detected, and therefore, in order to use these techniques in real-world applications, initial analysis of the impact of parameter values must be conducted on data streams which are representative of those encountered in real-world environments. When such data streams are not available, considerable domain expertise may be required.

To address the challenge of selecting drift sensitivity and window size parameter values, and to ensure fair comparisons throughout the remainder of this thesis, a single window size has been selected per dataset type, which is used by all CDDs, and a single drift sensitivity value has been chosen for each CDD, which is used across all dataset types. This means that the parameter values chosen are not fine tuned to each independent data stream, as might be expected when using CDDs in real-world environments where an in-depth analysis of parameter values for every data stream is infeasible. The parameter values displayed in Table 4.2 are

used throughout the remainder of this thesis, and are used to obtain the results presented in Chapters 5–7. Overall, parameter values have been selected such that the window size is small to allow rapid drift detection, but ensure that sufficient data is retained to build an effective predictive model. This was inferred by considering the percentage of models that were considered stable. Drift sensitivity parameters have been selected that prioritise high performance, but also take into account communication and computational overheads, which was inferred by considering both the number of stable and unstable models.

Chapter 5

The Bi-directional Online Transfer Learning Framework

Learning in online data streams can be challenging since data availability is often limited and a rich history of data cannot be retained [78]. Additionally, as discussed in Chapter 2, learning in dynamic non-stationary environments can introduce the problem of concept drift. Concept drift occurs when the underlying distribution of data, or the mapping between observations and response variable, changes over time [26]. This means that the knowledge that can be learnt from a data stream is often limited [95].

Online TL allows knowledge to be learnt from independent sources of data and transferred to others to improve predictive capabilities in an online data stream [107]. Existing online TL frameworks typically assume that knowledge is learnt from offline source domains, and transferred to online receiving domains [104]. However, this means that a data rich offline source domain must be available to learn the predictive models that will be beneficial to the online receiving domain [27]. This may not be the case for many real-world applications that consist of multiple devices, each learning in online environments [54]. For example, online learning may be used to personalise the functionality of a user facing application. This means that personalisations must be learnt from the online data stream associated with each user, however a data rich offline source domain does not exist.

In this Chapter, a novel online TL framework is introduced, namely the Bi-directional Online Transfer Learning (BOTL) framework. BOTL enables knowledge to be learnt from independent online data streams, and transferred to other online data streams to enhance the predictive performances achieved in each data stream. This means that both source and receiving domains are considered to be online.

Considering each domain to be online has three benefits over the existing online TL frameworks that are dependent on data rich offline source domains. First, concept drift detection strategies, as discussed in Chapter 4, can be used to learn each of the concepts encountered in a source domain and transferred to a receiving domain. This means that the receiving domains can benefit from knowledge learnt about each individual source concept. Second, as the data stream progresses in the online source domain, knowledge of newly encountered concepts can be learnt and transferred to a receiving domain. Third, knowledge can be transferred bi-directionally such that knowledge learnt from the data stream in the online source domain can benefit predictions made in the online receiving domain and vice versa.

The main contributions of this chapter are as follows.

- A novel online TL framework, namely BOTL, is presented, which enables knowledge to be transferred bi-directionally between online domains in a regression setting, and combined using an Ordinary Least Squares (OLS) meta-learner.
- A proof of the empirical loss of BOTL is provided, showing that BOTL is guaranteed to have an empirical loss no worse than the underlying CDD.
- Naïve model culling strategies are introduced, which can be used to prevent overfitting when the number of models transferred throughout the framework becomes large.

In this chapter, BOTL is evaluated in a regression setting using the synthetic and real-world datasets introduced in Chapter 3, namely the drifting hyperplane, heating simulator, and following distance datasets. The three CDDs discussed in Chapter 4 are used to detect concept drift in each independent data stream, allowing predictive models to be learnt to represent each of the concepts encountered, which are then transferred bi-directionally throughout the framework. BOTL is compared to the Generalised Online Transfer Learning (GOTL) framework [32], which assumes the source domain is offline.

The remainder of this Chapter is organised as follows. Section 5.1 introduces existing online TL frameworks, including GOTL. Section 5.2 formulates the setting in which BOTL is used. The proposed BOTL framework is presented in Section 5.3, and the empirical loss of BOTL is presented in Section 5.4. Section 5.5 briefly discusses the experimental setup used to evaluate BOTL, which includes details of how GOTL was adapted for use when all domains are online. Empirical results of BOTL and BOTL with model culling are presented in Section 5.6, and Section 5.7

presents further results and discussion on the effects of increasing the number of models transferred throughout the framework. Finally, Section 5.8 concludes this chapter, highlighting the limitations of BOTL and online TL in frameworks when all domains are online.

5.1 Online TL

Most existing online TL frameworks aim to transfer knowledge learnt from an offline source domain to an online target domain, referred to in this thesis as a receiving domain, typically for classification tasks [52]. The Online Transfer Learning (OTL) framework [104] was the first TL framework designed specifically to aid the predictions of an online target domain. OTL creates a predictive model in the offline source domain, which is transferred to the online target domain. A predictive model is also learnt in the target domain, which is updated as more instances are observed in the online data stream. Source and target models are then combined using a weighting mechanism, which is updated with respect to the performance of the source and target models over a sliding window of data in the target domain [104]. This allows the influence of the source and target models to adapt as the data stream progresses such that once sufficient instances are observed in the target domain, an effective predictive model can be learnt in the target domain, and the influence of the source model can be reduced. Using a weighting mechanism also allows the influence of the source and target models to adapt to concept drifts encountered in the target domain [104].

GOTL [32] extends the OTL weighting mechanism so that online TL can be used for both classification and regression tasks. To achieve this, GOTL incrementally updates the weights associated with the source and target models using step sizes. If the step size, δ, used to modify the weights is small enough, the ensemble of source and target models approximates the optimal weight combination [32]. However, if the step size is too small, it may take a substantial time for the weights to update to their desired values, making predictions unreliable during this period. Other online TL frameworks, such as Online Transfer Learning with Multiple Sources (OTLMS) [88] and Online Multi-Source Transfer Learning (OMS-TL) [27], follow similar approaches to online TL as OTL, but allow knowledge to be transferred from multiple offline source domains. This knowledge is then combined in the online target domain using ensemble methods [27, 88].

Although online TL has been actively studied [27, 33, 88, 91, 98, 104], existing frameworks are typically limited to learning knowledge from one or more data rich

Table 5.1: Notation for online TL and BOTL.

	Definition
α	Domain α, where $\mathcal{A}$ is used to denote a single source domain, and $\mathcal{B}$ denotes a receiving domain
X^α	Data stream in domain α, where $X^\alpha = \{x_1, \ldots, x_t, \ldots, x_n\}$
$x_t \in X^\alpha$	The t^{th} observed instance in X^α
Y^α	The response variable space in domain α
$y_t \in Y^\alpha$	The t^{th} response variable in Y^α
$\mathcal{M}^\alpha$	Knowledge base of models available in domain α
$f_i^\alpha : X_i^\alpha \rightarrow Y_i^\alpha$	Model i learnt in domain α
$F^{\mathcal{M}^\alpha} : X^\alpha \rightarrow Y^\alpha$	Meta-model of $\mathcal{M}$ in domain α
$\hat{y}_t^*$	Prediction using $F^{\mathcal{M}^\alpha}(x_t)$ in domain α
$\hat{y}_t^{\alpha_i}$	Prediction using $f_i^\alpha(x_t)$
W	Sliding window of $\|W\|$ instances, $W = \{x_{t-\|W\|}, \ldots, x_t\}$
W_{max}	Maximum window size (RePro)
W_{min}	Minimum window size (ADWIN, AWPro)
err_t	Predictive error of instance x_t
err_W	Predictive error across W
λ_l	Loss threshold (RePro)
λ_d	Drift threshold (RePro)
λ_r	Recurrence threshold (AWPro)
δ	Confidence value (ADWIN, AWPro)
λ_{perf}	Performance culling threshold
λ_{MI}	Mutual information culling threshold

offline source domains which can be transferred to an online receiving domain [104]. This means that online TL frameworks are dependent on acquiring and storing large amounts of data before knowledge can be transferred to aid predictions in the online receiving domain. This requirement of online TL frameworks limits the applicability of online TL for many real-world applications, where learning is conducted across multiple online devices without access to an offline source domain [20, 21]. The BOTL framework has been developed to mitigate the need for offline source domains, allowing knowledge to be learnt from online data streams and transferred to aid predictions in other online data streams, as might be expected in real-world applications such as smart home heating, or vehicle personalisation such as Adaptive Cruise Control (ACC).

5.2 Problem Formulation

To present the BOTL framework, online TL is first formalised using the notation given in Table 5.1. Let $\mathcal{A}$ and $\mathcal{B}$ denote two domains, each of which consist of a feature space X, where $x_t \in \mathbb{R}^m$ is the instance observed at time t such that $x_t = \{x_{t_1}, \ldots, x_{t_m}\} \in X$. Given a domain, for example domain $\mathcal{B}$, a task consists of the response variable, $y \in Y^{\mathcal{B}}$, where $y \in \mathbb{R}$, and a regression function, $f^{\mathcal{B}} : X^{\mathcal{B}} \rightarrow Y^{\mathcal{B}}$, which is learnt to represent the current concept, mapping observed data to response variables [65]. The knowledge learnt in a source domain, for example domain $\mathcal{A}$, can be transferred to the receiving domain $\mathcal{B}$, and used to enhance predictions [78].

Online TL aims to learn the predictive function in the receiving domain, $f^{\mathcal{B}}$, that effectively predicts the response variable, $y_t^{\mathcal{B}} \in Y^{\mathcal{B}}$, for each instance, $x_t^{\mathcal{B}} \in X^{\mathcal{B}}$, observed in the receiving domain's data stream, such that $\hat{y}_t^{\mathcal{B}_i} = f_i^{\mathcal{B}}(x_t^{\mathcal{B}})$. Model transfer is used to enhance predictions in the receiving domain by combining knowledge learnt in the local domain with knowledge learnt from other domains. For example, if we consider the scenario of application personalisation, where each domain represents an individual user, each instance, x_t, may describe the user's current environmental setting. If we wish to personalise application functionality by predicting some unknown value, y_t, knowledge learnt from another user, $f_j^{\mathcal{A}}$, can be used to enhance the predictive performance in the receiving domain $\mathcal{B}$.

Since most existing online TL frameworks consider the source domain to be offline [32, 88, 104], typically only a single model is transferred from each source domain to the receiving domain. However, if all domains in the framework are in online environments, concept drift detection strategies can be used to learn models that represent each of the concepts encountered in the data stream of a source domain. For example, model $f_j^{\mathcal{A}}$ can be transferred, where $j = 1, \ldots, k$, for each of the k concepts encountered in the source domain.

BOTL aims to minimise the predictive error in the receiving domain by combining knowledge learnt from the receiving domain's data stream with models previously learnt in a source domain. Focusing on minimising the loss with respect to the local, or receiving, domain makes BOTL highly applicable to the task of application personalisation, where predictions are made to benefit a specific individual. To achieve this, if there is source domain, $\mathcal{A}$, that has previously learnt models $f_1^{\mathcal{A}}, \ldots, f_j^{\mathcal{A}}$, and a receiving domain, $\mathcal{B}$, that has previously learnt models $f_1^{\mathcal{B}}, \ldots, f_i^{\mathcal{B}}$, at time t, then models $f_1^{\mathcal{A}}, \ldots, f_j^{\mathcal{A}}$, should be made available to the receiving domain such that the predictions made in the receiving domain can benefit from the knowledge learnt from the source domain $\mathcal{A}$. Since both domains are on-

line, and knowledge transfer is bi-directional, the models $f_1^{\mathcal{B}}, \ldots, f_i^{\mathcal{B}}$ should also be made available to domain $\mathcal{A}$, such that the predictions made in domain $\mathcal{A}$ can benefit from the knowledge learnt in domain $\mathcal{B}$, thus domain $\mathcal{A}$ becomes the receiving domain, and domain $\mathcal{B}$ becomes the source domain.

The source and receiving domains considered in this thesis are homogeneous, such that they share the same underlying feature space, $X^{\mathcal{A}} = X^{\mathcal{B}}$, and $Y^{\mathcal{A}} = Y^{\mathcal{B}}$. Although the domains are homogeneous, the underlying concepts to be learnt within each domain may not be equivalent, and therefore models from a source domain may not be relevant to the current concept observed in a receiving domain. BOTL provides a mechanism to combine models and maximise the impact of transferred models in the receiving domain.

5.3 BOTL

To utilise knowledge of distinct concepts, BOTL relies upon sliding window based CDDs that employ batch learners to create base models, f_i, from a window of data within a domain. With small windows, batch learners are susceptible to overfitting, however, larger window sizes can cause a reduction in sensitivity to gradual drifts [26]. Alternatively, incremental or online learners can be used to create base models. However, during periods of gradual drifts, data belonging to a new concept may be used to incrementally update the base model, preventing a drift from being detected. This is problematic for the BOTL framework since a pair of consecutive concepts present in one domain may not exist in another domain, meaning that transferred models may be less effective than if they were learnt using data from individual concepts. The three CDDs introduced in Chapter 4 have been used to learn predictive models from the data streams in each domain, namely RePro [95], ADWIN [8], and AWPro [55].

Although BOTL uses knowledge learnt from other domains to improve the predictive performance observed in the receiving domain, concept drift detection is conducted solely using the locally learnt model. Performing drift detection independently of any knowledge transfer is necessary since the use of transferred knowledge may enhance the predictive performance across the current window of data in the receiving domain, hindering drift detection.

A common challenge encountered by TL frameworks is that of negative transfer [65], which occurs when an ineffective model is transferred between domains. To address this, BOTL adopts the notion of model stability used by AWPro (in Chapter 4) [55], where a model is deemed to be stable if it has been used to make predic-

Algorithm 7: BOTL: transferring models to the receiving domain $\mathcal{B}$.

Input: W_{max}, $X^{\mathcal{B}}$, $\mathcal{M}^{\mathcal{B}}$
for t=1,2,... **do**
  if f^{α}_{j+1} *available* **then**
    Receive f^{α}_{j+1} from source domain α, add to $\mathcal{M}^{\mathcal{B}}$
  Receive $x_t \in X^{\mathcal{B}}$
  Update $W = \{x_{t-W_{max}}, \ldots, x_t\}$
  Learn $f^{\mathcal{B}}_i$ and detect drifts using RePro, ADWIN or AWPro (see Chapter 4), add to $\mathcal{M}^{\mathcal{B}}$
  $x_t{}' = \langle f^{\alpha}_1(x_t), \ldots, f^{\alpha}_k(x_t), f^{\mathcal{B}}_i(x_t) \rangle$
  $\hat{y}^*_t = F^{\mathcal{M}^{\mathcal{B}}}(x_t{}')$ (see Equ. 5.1)
  Receive y_t
  if $f^{\mathcal{B}}_i$ *is a new stable model* **then**
    Send $f^{\mathcal{B}}_i$ to all other domains in framework

tions across $2W_{max}$ instances without a drift being detected. Unstable models are not transferred, preventing them from negatively impacting predictions in a receiving domain. Defining model stability in this way prevents BOTL from transferring models that have been learnt from short, noisy periods of data, for example, during drifting periods as one concept changes to another. Once a model is considered to be stable, it is transferred to other domains to aid their respective predictors, as shown in Algorithm 7. This means that the models transferred by BOTL are limited to those that have successfully learnt a concept in their local domain.

Knowledge transfer is achieved in BOTL by communicating models across domains. When model $f^{\mathcal{A}}_j$ is received in domain $\mathcal{B}$ from the source domain $\mathcal{A}$, it is added to the set of received models, $\mathcal{M}^{\mathcal{B}}$, and combined with the locally learnt model, $f^{\mathcal{B}}_i$, to enhance the overall predictive performance in domain $\mathcal{B}$. Each predictive model is referred to as a base model, and those models are combined in the receiving domain through the use of a meta-learner. BOTL uses an Ordinary Least Squares (OLS) regressor as a meta-learner to combine the available models such that the squared error of the predicted values, $\hat{y}^*$, across W is minimised. Other regression meta-learners that are less prone to overfitting on small windows of data, such as Ridge Regression [13], could be used in place of OLS. However, OLS has been chosen as the meta-learner since it does not require additional parameters which would have to be determined from domain expertise or parameter tuning prior to learning in each data stream.

Each transferred model, $f^{\alpha}_j \in \mathcal{M}^{\mathcal{B}}$, where α denotes any domain in the framework from which a base model has been received, and the current locally learnt

model, $f_i^{\mathcal{B}}$, are used to generate a new window of data. Each sample x_t^* in the newly generated window of data is of the form $x_t^* = \{\hat{y}_t^{\alpha_1}, \ldots, \hat{y}_t^{\alpha_k}, \hat{y}_t^{\mathcal{B}_i}\}$, where $\hat{y}_t^{\alpha_j}$ for all $j = 1, \ldots, k$ is the predicted value of source model f_j^{α} on instance x_t from the original window of data observed in domain $\mathcal{B}$, and $\hat{y}_t^{\mathcal{B}_i}$ is the predicted value of the model learnt locally in the receiving domain $\mathcal{B}$, $f_i^{\mathcal{B}}$, learnt using the underlying CDD, for the current concept, c_i. This window of model predictions is used by the OLS meta-learner to obtain the overarching predictive function,

$$\begin{aligned} \hat{y}_t^* &= F^{\mathcal{M}^{\mathcal{B}}}(x_t^*) \\ &= w_0 + \left(\sum_{j=1}^{k} w_j f_j^{\alpha}(x_t) \right) + w_{(k+1)} f_i^{\mathcal{B}}(x_t). \end{aligned} \tag{5.1}$$

The OLS meta-learner is prone to overfitting when the window size is small and the number of base models is large, and therefore the BOTL framework only uses the current predictive model, $f_i^{\mathcal{B}}$, learnt in the receiving domain $\mathcal{B}$, as opposed to all historically learnt models from domain $\mathcal{B}$, as input to the meta-learner. Other historical models learnt within the data stream are excluded since the underlying CDD deems the current predictive model, $f_i^{\mathcal{B}}$, to be the most relevant with respect to the current concept.

5.3.1 Bi-directional Transfer

BOTL considers the scenario where all domains are online, and therefore distinctions between the source and receiving domains can be disregarded. BOTL conducts peer-to-peer model transfer, allowing knowledge transfer to enhance the predictive performances of all domains. When a newly learnt model is stable, it is transferred to all other domains in the framework, and each domain α updates its model set, $\mathcal{M}^{\alpha}$, when a concept drift is encountered.

Real-world applications, such as smart home heating system personalisation, may be comprised of a large number of domains, rapidly increasing the number of models to be transferred as the number of domains grows. Such applications can suffer in predictive performance due to the curse of dimensionality, where the number of input features to the OLS meta-learner becomes large in comparison to the window size [25]. To combat this, model culling is introduced to BOTL.

Algorithm 8: MI-Thresh: model culling.

Input: W, λ_{perf}, λ_{MI}, $\mathcal{M}^{\mathcal{B}}$, $f_i^{\mathcal{B}}$

for $f_j^{\alpha} \in \mathcal{M}^{\mathcal{B}}$ **do**
  if $R^2(f_j^{\alpha}, W) \leq \lambda_{perf}$ **then**
    Remove f_j^{α} from $\mathcal{M}^{\mathcal{B}}$
  for $f_k^{\alpha} \in \mathcal{M}^{\mathcal{B}}$ **do**
    if $MI(f_j^{\alpha}, f_k^{\alpha}, W) \geq \lambda_{MI}$ **then**
      if $R^2(f_j^{\alpha}, W) \geq R^2(f_k^{\alpha}, W)$ **then**
        Remove f_k^{α} from $\mathcal{M}^{\mathcal{B}}$
      else
        Remove f_j^{α} from $\mathcal{M}^{\mathcal{B}}$

5.3.2 Model Culling

Culling transferred models from the model set, $\mathcal{M}^{\mathcal{B}}$, available to the meta-learner in domain $\mathcal{B}$, helps to prevent the OLS meta-learner overfitting when a large number of models have been received but only a small window of data is available. This could be achieved by limiting the maximum number of models used by the meta-learner. However, due to the dynamic nature of the online environment, the maximum number of models that will prevent the meta-learner overfitting cannot be known in advance. Therefore, a conservative estimate would have to be made, requiring additional domain expertise. Using a conservative estimate may prevent beneficial transferred knowledge from being used to enhance predictions in the receiving domain.

Alternatively, transferred models can be evaluated on the current window of data observable in the receiving domain, W, allowing the models that are considered to be the least beneficial to the receiving domain to be discarded. This is achieved by introducing two variants of BOTL that use model culling, namely P-Thresh and MI-Thresh. P-Thresh reduces the number of models available to the OLS meta-learner by temporarily removing transferred models from the model set, $\mathcal{M}^{\mathcal{B}}$, in domain $\mathcal{B}$, when their R^2 predictive performance across the current window of data drops below a threshold, λ_{perf}. These models can be considered to be the least beneficial to the meta-learner in the receiving domain since they achieve poor predictive performance on the current window of observable data. Culled models are re-added to $\mathcal{M}^{\mathcal{B}}$ when a concept drift is encountered to enhance the predictions of future concepts in the receiving domain. Although this method of culling is naïve, it can reduce the impact of negative transfer.

In scenarios with high volumes of model transfer, P-Thresh requires a high λ_{perf} to sufficiently reduce the number of models to prevent the OLS meta-learner

overfitting. This can be detrimental since a high proportion of the models received from other domains containing useful information are culled and no longer available to enhance the predictive performance in the receiving domain. To overcome this, MI-Thresh, outlined in Algorithm 8, evaluates transferred models based on both performance and diversity, which are metrics commonly used in ensemble pruning [106]. Initially, MI-Thresh reduces the impact of negative transfer by culling models that achieve an R^2 predictive performance less than λ_{perf} on W. A low λ_{perf} value is preferred, ensuring that transferred models containing some useful information are retained. Using a low threshold may not sufficiently reduce the model set, $\mathcal{M}^{\mathcal{B}}$, to prevent overfitting, therefore a second round of culling is performed based on model diversity. MI-Thresh measures the diversity between transferred models using Mutual Information (MI). MI allows models that obtain similar predictions on the current window of observable data to be identified [23]. If two transferred models have a high MI, using both models as input to the meta-learner will provide little benefit since a high MI indicates that the predictions of the two models on the current window of data are highly correlated. Therefore, no additional knowledge is provided by keeping both in the model set. If two transferred models have a MI greater than λ_{MI}, then MI-Thresh culls the model that performs worse. This enables redundant models to be removed from the model set, helping to prevent overfitting.

Culling thresholds, λ_{perf} and λ_{MI}, could be updated as the data stream progresses using cross-validation, allowing alternative culling parameter values to be compared. However, due to the online nature of the data streams, instances within a window cannot be considered independent, therefore the i.i.d. assumption cannot be made [34], since three consecutive instances in the window, x_{t-1}, x_t, and x_{t+1}, are likely to be dependent. Therefore, any validation set created from the window has some dependence on the training set. This can cause cross-validation to provide an overestimate of the performance of culling parameters. Additionally, using cross-validation for this purpose would require $p * k$ models to be trained and validated every time the meta-learner is updated, where p is the number of culling parameter values compared, and k is the number of folds. Since the BOTL framework is to be used in domains with concept drifts, the meta-learner must be updated regularly in order to adapt to concept drifts. Therefore, the use of cross-validation would significantly increase the computation and storage requirements of the BOTL framework, while overestimating the performance of the culling parameters considered. This would limit the use of BOTL in applications that have limited computational resources.

Alternatively, regularisation techniques could be used instead of model culling to prevent the meta-learner overfitting. However, in order to achieve this, regularisation parameters must be selected to determine how aggressively regularisation is used to reduce the influence of base models. Determining an appropriate regularisation parameter value may be challenging since it is dependent on the underlying distribution of data and the number of models available to the meta-learner. Regularisation parameters may also be considered to be less intuitive than the use of predictive performance or MI thresholds since the selection of a regularisation parameter value provides little indication of how beneficial the knowledge retained after regularisation will be to the underlying concepts encountered in the data stream. Similarly to culling thresholds, the regularisation parameter could be updated as the data stream progresses through the use of cross-validation, however the predictive performance obtained from cross-validation is also likely to be an overestimate, and increases computation and storage when used in the BOTL framework. Therefore, within this chapter, the naïve culling approaches of P-Thresh and MI-Thresh are used, since predictive performance and MI culling parameters may be more intuitive and interpretable than regularisation parameters. The values of culling parameters are chosen in advance by evaluating threshold values across multiple different datasets, mitigating the need for cross-validation. Further analysis on the effects of culling parameters is provided in Chapter 6.

5.3.3 Initialisation

For any underlying CDD, an initial window of data, W, is required to create the first predictive model, $f_1^{\mathcal{B}}$, in domain $\mathcal{B}$. Prior to obtaining this data, no predictions can be made since no local knowledge has been learnt in domain $\mathcal{B}$. BOTL allows models transferred from other domains to be used to make predictions during this period. Models transferred are initially weighted equally to obtain,

$$\hat{y}_t^* = \frac{1}{|\mathcal{M}^{\mathcal{B}}|} \sum_{j=1}^{|\mathcal{M}^{\mathcal{B}}|} f_j^{\alpha}(x_t). \tag{5.2}$$

Before the first predictive model, $f_1^{\mathcal{B}}$, has been learnt and only a small amount of data has been observed, the OLS meta-learner can create a model, $F^{\mathcal{M}^{\mathcal{B}}}$, using only the models received from source domains, f_j^{α}. This approach is prone to overfitting due to the small amount of data available, but may be preferred over making no predictions or using Equation 5.2 over the entire initial window of data.

P-Thresh and MI-Thresh help to reduce overfitting within this initial period,

however, as the amount of available data is small, all transferred models may have R^2 predictive performances below the culling threshold. In this scenario, both P-Thresh and MI-Thresh select the best k transferred models, where $k < |W|$, regardless of the culling threshold. Since the amount of available data is limited during this period, the three best transferred models are selected such that $k = 3$.

5.4 Empirical Risk Guarantee

Using OLS as a meta-learner provides BOTL with a guarantee on the empirical risk. In this section, we consider the empirical risk observed by the receiving domain $\mathcal{B}$, where knowledge is transferred from source domains, denoted by α.

Theorem 1. *BOTL has a squared loss less than or equal to the model learnt locally in domain $\mathcal{B}$, using a CDD with no knowledge transfer,*

$$\mathcal{L}\left(f_i^{\mathcal{B}}\right) \geq \mathcal{L}\left(F^{\mathcal{M}^{\mathcal{B}}}\right), \tag{5.3}$$

where $\mathcal{L}\left(f_i^{\mathcal{B}}\right)$ denotes the squared loss of the local model, $f_i^{\mathcal{B}}$, created using a CDD, and $\mathcal{L}\left(F^{\mathcal{M}^{\mathcal{B}}}\right)$ is the squared loss of the OLS meta-learner, $F^{\mathcal{M}^{\mathcal{B}}}$, created using the set of k models transferred from source domains, denoted by α, $\{f_1^{\alpha}, \ldots, f_k^{\alpha}\}$ and the current locally learnt predictive model, $f_i^{\mathcal{B}}$.

Proof. The loss over the local window of data, W, is measured using the mean squared error of predictions,

$$\mathcal{L}(\cdot) = \frac{1}{|W|} \sum_{t=1}^{|W|} \left(y_t - \hat{y}_t^*\right)^2, \tag{5.4}$$

where y_t is the response variable for instance x_t, and $\hat{y}_t^*$ is the predicted value. If no transfer is used, the local model, $f_i^{\mathcal{B}}$, is used to predict $\hat{y}_t^*$ for each instance x_t such that $\hat{y}_t^* = \hat{y}_t^{\mathcal{B}_i}$, and $\hat{y}_t^{\mathcal{B}_i} = f_i^{\mathcal{B}}(x_t)$.

BOTL uses the set of models, $\mathcal{M}^{\mathcal{B}}$, to obtain predictions $\hat{y}_t^{\mathcal{B}_i}$ and all $\hat{y}_t^{\alpha_j}$ for instance x_t, using the locally learnt model, $f_i^{\mathcal{B}}$, and each of the j transferred model, $f_j^{\alpha} \in \{f_1^{\alpha}, \ldots, f_k^{\alpha}\}$, respectively. Predictions are used to create a meta instance, x_t^*, which the OLS meta-learner, $F^{\mathcal{M}^{\mathcal{B}}}$, uses to obtain an overarching prediction,

$$\begin{aligned} \hat{y}_t^* &= F^{\mathcal{M}^{\mathcal{B}}}\left(x_t^*\right) \\ &= F^{\mathcal{M}^{\mathcal{B}}}\left(\langle f_1^{\alpha}(x_t), \ldots, f_k^{\alpha}(x_t), f_i^{\mathcal{B}}(x_t)\rangle\right) \\ &= F^{\mathcal{M}^{\mathcal{B}}}\left(\langle \hat{y}_t^{\alpha_1}, \ldots, \hat{y}_t^{\alpha_k}, \hat{y}_t^{\mathcal{B}_i}\rangle\right), \end{aligned} \tag{5.5}$$

where

$$F^{\mathcal{M}^{\mathcal{B}}}(x_t^*) = w_0 + \sum_{j=1}^{j=k} w_j \hat{y}_t^{\alpha_j} + w_{(k+1)} \hat{y}_t^{\mathcal{B}_i}. \tag{5.6}$$

Weights $w_0, \ldots, w_{(k+1)}$ are assigned to each prediction, $\hat{y}_t^n$, for each model n in $\mathcal{M}^{\mathcal{B}}$, where $|\mathcal{M}^{\mathcal{B}}| = (k+1)$, to obtain an ensemble prediction, $\hat{y}_t^*$, for instance x_t by solving the optimisation problem that minimises the squared error of $F^{\mathcal{M}^{\mathcal{B}}}$:

$$\min_{w_0,\ldots,w_{(k+1)}} \sum_{t=1}^{|W|} \left(y_t - \left(w_0 + \sum_{j=1}^{j=k} w_j \hat{y}_t^{\alpha_j} + w_{(k+1)} \hat{y}_t^{\mathcal{B}_i} \right) \right)^2. \tag{5.7}$$

$F^{\mathcal{M}^{\mathcal{B}}}$ is used to make predictions, $\hat{y}_t^*$, for instance x_t, using Equation 5.6. Using Equation 5.4, we can rewrite the loss of $F^{\mathcal{M}^{\mathcal{B}}}$ as,

$$\mathcal{L}\left(F^{\mathcal{M}^{\mathcal{B}}}\right) = \frac{1}{|W|} \sum_{t=1}^{|W|} \left(y_t - \left(w_0 + \sum_{j=1}^{j=k} w_j \hat{y}_t^{\alpha_j} + w_{(k+1)} \hat{y}_t^{\mathcal{B}_i} \right) \right)^2. \tag{5.8}$$

If we constrain the optimisation problem in Equation 5.7 to obtain the meta-learner $F^{\mathcal{M}^{\mathcal{B}*}}$ by fixing the weights, w_a, such that the weight associated with the locally learnt model $f_i^{\mathcal{B}}$ is 1, while all others are 0, we obtain a meta-learner of the form,

$$F^{\mathcal{M}^{\mathcal{B}*}}(x_t^*) = \left(0 + \sum_{j=1}^{j=k} 0 \hat{y}_t^{\alpha_j} + 1 \hat{y}_t^{\mathcal{B}_i} \right), \tag{5.9}$$

giving the loss function

$$\mathcal{L}\left(F^{\mathcal{M}^{\mathcal{B}*}}\right) = \frac{1}{|W|} \sum_{t=1}^{|W|} \left(y_t - \hat{y}_t^{\mathcal{B}_i} \right)^2, \tag{5.10}$$

equivalent to only using the locally learnt model, $\mathcal{L}\left(F^{\mathcal{M}^{\mathcal{B}*}}\right) = \mathcal{L}\left(f_i^{\mathcal{B}}\right)$. As the optimisation problem in Equation 5.7 is convex,

$$\mathcal{L}\left(F^{\mathcal{M}^{\mathcal{B}*}}\right) \geq \mathcal{L}\left(F^{\mathcal{M}^{\mathcal{B}}}\right). \tag{5.11}$$

Finally, as the constrained optimisation problem in Equation 5.9 is equivalent to using only the locally learnt model, $f_i^{\mathcal{B}}$, the loss of BOTL is less than or equal to the loss of the locally learnt model. □

5.5 Experimental Set-Up

The datasets introduced in Chapter 3 are used to evaluate BOTL, namely the drifting hyperplane, heating simulator and following distance datasets. Base models are created in each domain using the underlying CDDs introduced in Chapter 4, namely RePro [95], ADWIN [8] and AWPro [55]. The base models created in this chapter are Support Vector Regressors (SVRs), however, since BOTL is model agnostic, other regression based machine learning algorithms can be used. The results obtained using SVRs as base models are presented in Section 5.6, and similar results are presented in Appendix A which are obtained when using Ridge Regressors (RRs) as base models. The underlying performance of each CDD is used as a baseline so that BOTL can be compared against frameworks without knowledge transfer. Additionally, BOTL is compared to the GOTL framework. GOTL assumes that there exists an offline data rich source domain. However, as the BOTL framework is being evaluated to consider the implications when all domains are online, minor adaptations to GOTL were required. In this section, the adaptation of GOTL is presented, and any user parameter values that were required by GOTL and BOTL model culling strategies are provided.

5.5.1 GOTL Adaptation

Since each data stream is subject to concept drift, GOTL uses the CDDs outlined in Chapter 4 to create models that represent each of the concepts encountered in a source domain. Once knowledge of every concept has been learnt from the data stream, the model that is considered to be the most stable is transferred to receiving domains. Model stability was determined by the duration over which each model was used to make predictions. Therefore, the model learnt in the source domain that made predictions for the largest number of instances in the source domain was considered to be the most stable, and transferred to the receiving domains. A small step-size, $\delta = 0.025$, was chosen, as suggested by Grubinger *et al.* [32], which slowly modifies the weights used to combine the transferred model with the model learnt locally in the receiving domain.

When evaluating GOTL, experiments were conducted such that each data stream was paired with every other data stream as source and receiving domains respectively. This allows the predictive performances of GOTL and BOTL to be compared over the same test examples. Due to only transferring the most stable model when using GOTL, learning in the receiving domain only commenced once learning in the source domain had completed, such that the most stable model

learnt in the source domain could be identified and transferred. This meant that GOTL was able to learn the most stable predictive model from the entire data stream available in the source domain prior to knowledge transfer, whereas in BOTL, learning and knowledge transfer is conducted simultaneously across data streams such that knowledge of the stability of future predictive models is not known in advance. Due to this, the performance of GOTL presented in this chapter accounts for both the performance of the source and receiving domains, since GOTL requires models to be learnt in the source domain without knowledge transfer before learning can commence in the receiving domain.

5.5.2 BOTL Frameworks

BOTL combines knowledge via the OLS meta learner and therefore no additional parameters are required, however P-Thresh and MI-Thresh culling parameters must be defined. We set $\lambda_{\text{perf}} = 0.2$ for P-Thresh, thereby discarding models that performed worse than a poor predictor ($R^2 < 0.2$). MI-Thresh also used a performance threshold parameter of $\lambda_{\text{perf}} = 0.2$ and used a MI threshold of $\lambda_{MI} = 0.2$. These threshold parameter values were chosen as a result of investigations of a wide rage of culling parameter values, which is discussed further in Chapter 6.

When evaluating BOTL, P-Thresh, and MI-Thresh, all data streams for a given experiment were used as source domains with bi-directional transfer. Repeat experiments were conducted by randomising the ordering and interval between the commencement of learning in each domain. This experimental approach has also been used to obtain results in Chapters 6 and 7.

5.6 Experimental Results

In this section, BOTL, P-Thresh and MI-Thresh are compared to an existing online TL framework, namely GOTL, and the underlying CDD, using performance metrics such as R^2 and squared Product Moment Correlation Coefficient (PMCC^2) to highlight the benefits of using online TL where all domains are online. T-tests have also been used to identify approaches that outperform the underlying CDD with statistical significance ($p < 0.01$). Since t-tests are used in this thesis to compare the predictions obtained from each data stream in a framework, where the number of data streams in a framework ranges between 5 and 17 depending on the dataset type, and results have been recorded from 30 repeat experiments, the number of observations is large enough for the t-tests to be robust even though they may not be normally distributed [24, 61]. For example, for the sudden drifting hyperplane

dataset, the predictive performance of 5 data streams have been recorded for a single experiment, which has been repeated 30 times, providing 150 observations used by the t-test. In this thesis, every domain both transfers knowledge and receives knowledge, and therefore when knowledge is learnt in a domain, it is referred to as a source domain, and if it has received knowledge then it is referred to as a receiving domain.

5.6.1 Drifting Hyperplane Datasets

BOTL is evaluated on variants of the drifting hyperplane datasets presented in Chapter 3. Tables 5.2 and 5.3 present the results obtained by the CDDs, where no knowledge is transferred, GOTL, and BOTL variants, for the sudden and gradual drifting hyperplane datasets respectively.

These results show that GOTL obtained a poorer predictive performance than the underlying CDD for most variants of the sudden and gradual drifting hyperplane datasets, and was only able to outperform the underlying CDD with statistical significance ($p < 0.01$) on the GradualD variant when AWPro is used as the CDD. For the drifting hyperplane datasets, GOTL does not benefit from knowledge transfer, even though the most stable model learnt from the source domain is transferred to the receiving domain. This is because these datasets contain high frequencies of concept drifts, where concept drifts are encountered periodically every 500 instances in each data stream. Since GOTL uses a step-wise mechanism to incrementally update the weights associated with each model, the influence of a model cannot change drastically over a small period of time. This means that a large amount of data must be observed after each drift to converge on an approximation of the optimal weights. To overcome this, a larger step size could be used, however, this may prevent or hinder convergence. BOTL overcomes this by using the OLS meta-learner to minimise the squared error of the combined models with instantaneous effect.

Although BOTL uses the OLS meta-learner to update the influence of the locally learnt and received models, it also only outperforms the underlying CDD with statistical significance ($p < 0.01$) for a minority of the drifting hyperplane datasets. Additionally, in the SuddenB, SuddenD, GradualB and GradualD datasets, BOTL achieves a negative R^2 predictive performance. This is caused by the OLS meta-learner in BOTL overfitting. The OLS meta-learner drastically overfits in these datasets since a window size of only 30 instances is used to train the OLS meta-learner when a large number of base models are used as input. For example, in the SuddenB dataset where ADWIN is used as the CDD, the BOTL meta-learner has on

(a) SuddenA: sudden drifting hyperplanes with uniform noise.

	RePro				ADWIN				AWPro			
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$
CDD	0.883 (±0.019)	0.884	1	1	0.851 (±0.008)	0.854	1	1	0.830 (±0.027)	0.835	1	1
GOTL	0.814 (±0.036)	0.844	2	2	0.745 (±0.056)	0.752	2	2	0.685 (±0.082)	0.690	2	2
BOTL	0.834 (±0.034)	0.845	24	31	0.828 (±0.026)	0.839	41	57	***0.884** (±0.016)	0.886	17	21
P-Thresh	***0.902** (±0.008)	0.903	6	8	*0.886 (±0.010)	0.888	9	15	*0.881 (±0.016)	0.882	5	6
MI-Thresh	*0.902 (±0.009)	0.902	4	5	***0.887** (±0.011)	0.889	3	4	*0.880 (±0.015)	0.882	3	4

(b) SuddenB: sudden drifting hyperplanes with single sensor failure.

	RePro				ADWIN				AWPro			
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$
CDD	0.883 (±0.010)	0.884	1	1	0.830 (±0.018)	0.836	1	1	0.808 (±0.038)	0.814	1	1
GOTL	0.811 (±0.024)	0.836	2	2	0.718 (±0.061)	0.725	2	2	0.664 (±0.106)	0.676	2	2
BOTL	-2e+21 (±5e+21)	0.507	26	34	-7e+22 (±1e+23)	0.499	41	60	-3e+22 (±5e+22)	0.532	20	27
P-Thresh	***0.905** (±0.007)	0.905	5	8	-3e+18 (±9e+18)	0.767	8	14	***0.874** (±0.019)	0.875	4	6
MI-Thresh	*0.904 (±0.007)	0.904	4	5	***0.879** (±0.012)	0.880	3	5	*0.873 (±0.020)	0.874	3	4

(c) SuddenC: sudden drifting hyperplanes with intermittent single sensor failure.

	RePro				ADWIN				AWPro			
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$
CDD	0.873 (±0.020)	0.875	1	1	0.845 (±0.015)	0.848	1	1	0.847 (±0.010)	0.850	1	1
GOTL	0.817 (±0.031)	0.843	2	2	0.770 (±0.024)	0.780	2	2	0.752 (±0.032)	0.762	2	2
BOTL	0.841 (±0.022)	0.850	25	32	0.813 (±0.040)	0.827	41	57	***0.888** (±0.009)	0.889	18	21
P-Thresh	***0.906** (±0.013)	0.906	6	8	***0.884** (±0.013)	0.886	8	13	*0.883 (±0.010)	0.884	4	6
MI-Thresh	*0.905 (±0.013)	0.905	3	5	*0.882 (±0.014)	0.883	3	4	*0.881 (±0.010)	0.882	3	4

(d) SuddenD: sudden drifting hyperplanes with gradual sensor deterioration.

	RePro				ADWIN				AWPro			
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$
CDD	0.870 (±0.026)	0.871	1	1	0.845 (±0.007)	0.849	1	1	0.810 (±0.059)	0.816	1	1
GOTL	0.809 (±0.023)	0.838	2	2	0.764 (±0.013)	0.773	2	2	0.649 (±0.135)	0.662	2	2
BOTL	-1e+22 (±2e+22)	0.341	27	35	-9e+21 (±2e+22)	0.334	41	59	-2e+21 (±3e+21)	0.356	18	22
P-Thresh	***0.902** (±0.012)	0.903	5	8	-2e+19 (±5e+19)	0.588	7	12	***0.874** (±0.022)	0.875	4	5
MI-Thresh	*0.900 (±0.014)	0.901	3	5	***0.885** (±0.008)	0.886	3	4	*0.873 (±0.021)	0.874	3	4

Table 5.2: Sudden Drifting Hyperplane: R^2 and PMCC^2 predictive performance, the average number of base models used to make predictions ($|\mathcal{M}'|$), and the maximum number of base models used to make predictions ($\lceil\mathcal{M}'\rceil$) for the underlying CDD, GOTL, BOTL, P-Thresh and MI-Thresh for variants of the sudden drifting hyperplane datasets when transferring base models between 5 data streams. Improved predictive performances with statistical t-test values $p < 0.01$ compared to the underlying CDD are indicated with *, and bold type indicates the approach with highest performance.

(a) GradualA: gradual drifting hyperplanes with uniform noise.

	RePro				ADWIN				AWPro			
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$
CDD	0.848 (±0.027)	0.848	1	1	0.796 (±0.049)	0.803	1	1	0.798 (±0.059)	0.804	1	1
GOTL	0.792 (±0.019)	0.820	2	2	0.712 (±0.053)	0.717	2	2	0.710 (±0.055)	0.718	2	2
BOTL	0.776 (±0.046)	0.800	31	37	0.792 (±0.046)	0.811	39	54	*0.883 (±0.011)	0.886	20	25
P-Thresh	***0.892** (±0.020)	0.892	9	13	*0.880 (±0.023)	0.881	12	21	***0.888** (±0.023)	0.889	6	9
MI-Thresh	***0.892** (±0.020)	0.892	4	6	***0.885** (±0.023)	0.885	4	5	*0.887 (±0.024)	0.887	4	5

(b) GradualB: gradual drifting hyperplanes with single sensor failure.

	RePro				ADWIN				AWPro			
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$
CDD	0.859 (±0.010)	0.860	1	1	0.757 (±0.090)	0.771	1	1	0.697 (±0.094)	0.719	1	1
GOTL	0.784 (±0.023)	0.808	2	2	0.706 (±0.081)	0.715	2	2	0.641 (±0.120)	0.649	2	2
BOTL	-6e+20 (±9e+20)	0.323	29	38	-1e+21 (±2e+21)	0.322	40	57	-1e+21 (±2e+21)	0.348	21	28
P-Thresh	***0.890** (±0.014)	0.891	7	10	-3e+17 (±1e+18)	0.750	9	16	***0.848** (±0.033)	0.849	5	8
MI-Thresh	*0.887 (±0.012)	0.888	4	5	***0.862** (±0.026)	0.863	3	5	*0.845 (±0.034)	0.846	3	4

(c) GradualC: gradual drifting hyperplanes with intermittent single sensor failure.

	RePro				ADWIN				AWPro			
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$
CDD	0.842 (±0.008)	0.842	1	1	0.773 (±0.067)	0.779	1	1	0.759 (±0.091)	0.773	1	1
GOTL	0.778 (±0.028)	0.802	2	2	0.695 (±0.075)	0.700	2	2	0.697 (±0.073)	0.704	2	2
BOTL	0.781 (±0.044)	0.803	29	36	0.776 (±0.051)	0.799	39	54	***0.873** (±0.016)	0.875	21	27
P-Thresh	***0.892** (±0.007)	0.893	8	11	***0.881** (±0.015)	0.882	10	15	*0.872 (±0.015)	0.872	6	8
MI-Thresh	*0.890 (±0.007)	0.890	4	6	*0.878 (±0.020)	0.878	4	5	*0.870 (±0.014)	0.871	4	5

(d) GradualD: gradual drifting hyperplanes with gradual sensor deterioration.

	RePro				ADWIN				AWPro			
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$
CDD	0.846 (±0.015)	0.847	1	1	0.484 (±0.468)	0.593	1	1	0.519 (±0.487)	0.629	1	1
GOTL	0.791 (±0.032)	0.813	2	2	*0.685 (±0.044)	0.691	2	2	*0.640 (±0.052)	0.643	2	2
BOTL	-3e+22 (±5e+22)	0.147	31	41	-2e+22 (±3e+22)	0.143	26	44	-2e+22 (±3e+22)	0.129	17	26
P-Thresh	*0.897 (±0.012)	0.898	7	10	***0.830** (±0.165)	0.831	6	12	***0.832** (±0.169)	0.833	4	7
MI-Thresh	***0.898** (±0.010)	0.898	4	6	*0.812 (±0.191)	0.813	3	5	*0.821 (±0.195)	0.821	3	4

Table 5.3: Gradual Drifting Hyperplane: R^2 and PMCC^2 predictive performance, the average number of base models used to make predictions ($|\mathcal{M}'|$), and the maximum number of base models used to make predictions ($\lceil\mathcal{M}'\rceil$) for the underlying CDD, GOTL, BOTL, P-Thresh and MI-Thresh for variants of the gradual drifting hyperplane datasets when transferring base models between 5 data streams. Improved predictive performances with statistical t-test values $p < 0.01$ compared to the underlying CDD are indicated with *, and bold type indicates the approach with highest performance.

average 41 transferred models that are used as input, and therefore 41 weights must be learnt from 30 instances captured within the sliding window of data, to determine the influence of each of the models received from other domains. The largest number of models used as input to the meta-learner, $\lceil \mathcal{M}' \rceil$, is also recorded in Tables 5.2 and 5.3, highlighting the correlation between obtaining a negative R^2 predictive performance and using large numbers of base models as input to the meta-learner. This phenomenon, where the number of input features is large in comparison to the window of available data, is known as the curse of dimensionality [25].

BOTL drastically suffers from the curse of dimensionality in the drifting hyperplane datasets. To prevent overfitting, model culling strategies have been used by P-Thresh and MI-Thresh. P-Thresh simply removes models if they have an R^2 predictive performance below λ_{perf}. This can be considered as removing models that provide no beneficial information to the meta-learner. MI-Thresh also culls transferred models using their predictive performance, but continues to cull transferred models if two transferred models obtain predictions with high MI over a recent window of observations. This means that MI-Thresh culls models that provide no beneficial information to the meta-learner and those that provide no additional information to the meta-learner.

P-Thresh obtains predictive performances that are statistically significantly greater than the underlying CDDs with no knowledge transfer ($p < 0.01$) for all drifting hyperplane datasets except for when ADWIN is used as the CDD in SuddenB, SuddenD and GradualB datasets, as shown in Tables 5.2b, 5.2d and 5.3b. In addition to outperforming the CDDs in these datasets, P-Thresh also outperforms GOTL with statistical significance ($p < 0.01$), and outperforms BOTL, where all base models are used as input to the meta-learner, except for when AWPro is used as the CDD in SuddenA, SuddenC and GradualC datasets. In these three datasets BOTL achieves the best predictive performance of all frameworks, as shown in Tables 5.2a, 5.2c and 5.3c respectively.

Although P-Thresh shows that the transfer of knowledge can be used to benefit predictions in the receiving domain, the OLS meta-learner used in P-Thresh continues to suffer from overfitting when ADWIN is used as the CDD in SuddenB, SuddenD and GradualB datasets, as shown in Tables 5.2b, 5.2d and 5.3b respectively. Overfitting continues to occur, even with model culling in these frameworks because ADWIN is used as the underlying CDD. Unlike RePro and AWPro, ADWIN does not reuse historical models in the presence of recurring concepts. This means that a large number of models, which have been learnt to represent the same concept, are created and transferred throughout the framework. Therefore, even

	RePro				ADWIN				AWPro			
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$
CDD	0.636 (±0.057)	0.652	1	1	0.635 (±0.066)	0.655	1	1	0.625 (±0.066)	0.645	1	1
GOTL	0.666 (±0.051)	0.677	2	2	0.656 (±0.047)	0.670	2	2	0.618 (±0.074)	0.635	2	2
BOTL	***0.728** (±0.055)	0.735	10	15	***0.717** (±0.061)	0.725	9	15	***0.727** (±0.059)	0.734	10	15
P-Thresh	*0.716 (±0.054)	0.723	6	10	*0.712 (±0.060)	0.719	6	10	*0.718 (±0.059)	0.725	7	10
MI-Thresh	*0.706 (±0.051)	0.713	3	5	*0.705 (±0.056)	0.713	3	5	*0.708 (±0.054)	0.716	3	4

Table 5.4: Heating Simulator: R^2 and PMCC^2 predictive performance, the average number of base models used to make predictions ($|\mathcal{M}'|$), and the maximum number of base models used to make predictions ($\lceil\mathcal{M}'\rceil$) for the underlying CDD, GOTL, BOTL, P-Thresh and MI-Thresh for the smart home heating simulator dataset when transferring base models between 5 data streams. Improved predictive performances with statistical t-test values $p < 0.01$ compared to the underlying CDD are indicated with *, and bold type indicates the approach with highest performance.

though models with poor predictive performance are culled, the meta-learner still suffers from overfitting.

MI-Thresh uses a more aggressive culling technique in comparison to P-Thresh, since the similarity of model predictions are also used to determine when transferred models should be culled. This aggressive nature can be seen in Tables 5.2 and 5.3, when comparing the average number of models used as input to the meta-learner, $\mathcal{M}'$, in the P-Thresh and MI-Thresh frameworks. In addition to this, the largest number of models used as input to the meta-learner, $\lceil\mathcal{M}'\rceil$, presented in Tables 5.2 and 5.3, indicates that the largest number of models used by the meta-learner in MI-Thresh is almost half of the largest number of models used by the meta-learner in P-Thresh. Due to this, MI-Thresh prevents the meta-learner from overfitting in all sudden and gradual drifting hyperplane datasets, obtaining predictive performances that are statistically significantly greater than both the underlying CDD and GOTL ($p < 0.01$) for all drifting hyperplane datasets.

5.6.2 Heating Simulator Dataset

Lower predictive performances are observed across the heating simulator dataset in comparison to drifting hyperplane datasets. This is because the data streams typically contain more complex concepts and additional noise due to their dependency on real-world weather data. The addition of knowledge transfer, using BOTL and GOTL, provides an increase in performance in comparison to the CDDs, except in the case of AWPro, where GOTL obtains a slightly poorer predictive performance.

Table 5.4 shows that BOTL achieves the highest predictive performance, outperforming both the CDDs and GOTL with statistical significance ($p < 0.01$). Unlike the drifting hyperplane datasets, BOTL does not suffer from the curse of

dimensionality, even when ADWIN is used as the underlying CDD. This is because the size of the window of data used to train the meta-learner in the heating simulator dataset is much larger than the window size used in the drifting hyperplane datasets. Additionally, fewer predictive models are learnt and transferred throughout the framework. For example, in the heating simulator dataset, an average of 10 models are received by each domain in the BOTL framework, and the largest number of models received by a domain, $\lceil \mathcal{M}' \rceil$, is 15, while a window size of 480 instances, capturing 10 days of weather observations, is used. Since fewer transferred models and a larger window size are used to train the OLS meta-learner, BOTL is able to create a meta-learner that does not overfit, and using culling mechanisms such as P-Thresh and MI-Thresh obtain predictive performances lower than BOTL.

5.6.3 Following Distance Dataset

The results for GOTL, BOTL, P-Thresh and MI-Thresh for the real-world following distance dataset with 7 data streams are shown in Table 5.5. These results show that GOTL is able to out perform RePro and AWPro with statistical significance ($p < 0.01$), highlighting the benefits of sharing knowledge in real-world online data streams. BOTL continues to suffer from overfitting when ADWIN and AWPro are used as CDDs. The use of P-Thresh to cull base models also suffers from overfitting when ADWIN is used. Although the R^2 predictive performance in these frameworks is drastically reduced in comparison to the underlying CDD, where no knowledge is transferred between domains, the PMCC^2 predictive performance of BOTL is greater than that of the CDD when AWPro is used to detect drifts, as is the PMCC^2 performance of P-Thresh when ADWIN is used as the CDD. Although the average number of models used as input to the meta-learner is small for P-Thresh when ADWIN is used as the meta-learner, the largest number of models used as input to the meta-learner, $\lceil \mathcal{M}' \rceil$, remains high. This indicates that the meta-learner may suffer from overfitting in small periods of the data stream, allowing an improved PMCC^2 predictive performance to be observed in comparison to the underlying CDD, while also incurring a negative R^2 predictive performance.

In the following distance datasets, MI-Thresh obtains the greatest predictive performances across all frameworks for every CDD. This indicates that the transfer of knowledge learnt from each data stream provides beneficial information to the receiving domain, however, overfitting can occur when all transferred models are used as input to the meta-learner. To prevent this, MI-Thresh uses MI and performance culling, and is able to outperform BOTL with statistical significance ($p < 0.09$). This highlights the importance of removing base models from the meta-learner that

	RePro				ADWIN				AWPro			
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$
CDD	0.546 (±0.081)	0.565	1	1	0.441 (±0.153)	0.502	1	1	0.396 (±0.194)	0.515	1	1
GOTL	*0.602 (±0.024)	0.657	2	2	0.419 (±0.416)	0.560	2	2	*0.487 (±0.358)	0.603	2	2
BOTL	*0.646 (±0.117)	0.678	6	12	-2e+15 (±4e+15)	0.384	12	25	-9e+9 (±5e+10)	0.665	8	18
P-Thresh	*0.648 (±0.117)	0.677	4	8	-2e+8 (±1e+9)	0.652	6	12	*0.688 (±0.103)	0.700	4	8
MI-Thresh	***0.665** (±0.116)	0.522	2	3	***0.663** (±0.126)	0.527	3	4	***0.693** (±0.107)	0.503	2	4

Table 5.5: Following Distance: R^2 and PMCC^2 predictive performance, the average number of base models used to make predictions ($|\mathcal{M}'|$), and the maximum number of base models used to make predictions ($\lceil\mathcal{M}'\rceil$) for the underlying CDD, GOTL, BOTL, P-Thresh and MI-Thresh for the following distance dataset when transferring base models between 7 data streams. Improved predictive performances with statistical t-test values $p < 0.01$ compared to the underlying CDD are indicated with *, and bold type indicates the approach with highest performance.

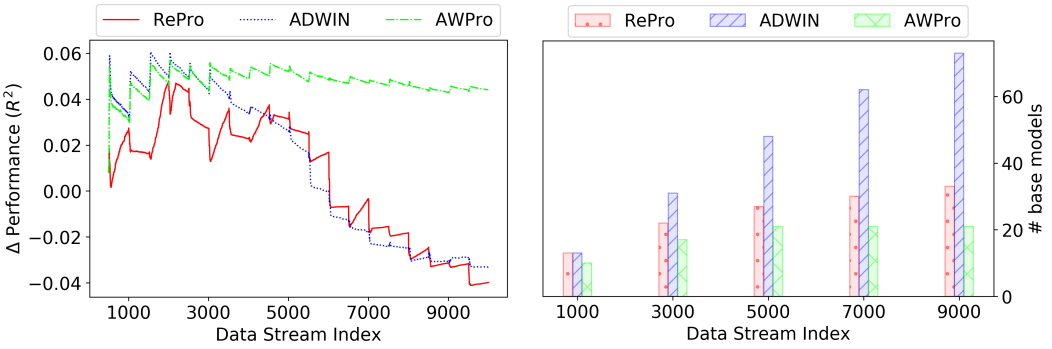
provide no beneficial or additional information to the meta-learner, when the number of base models is large in comparison to the window of available data.

Table 5.5 shows that AWPro obtains the worst predictive performance of all CDDs without knowledge transfer. However, using MI-Thresh with AWPro as the underlying CDD obtains the highest predictive performance across all CDDs, GOTL, BOTL, P-Thresh and MI-Thresh variants for the real-world following distance dataset. This indicates that although AWPro may not be the most effective prediction method when learning from a single data stream, its ability to estimate the precise point of concept drift within the sliding window, and its ability to reuse historical models in the presence of recurring concepts, are beneficial to online TL in real-world environments.

5.7 Increasing Model Availability

The BOTL framework enables knowledge to be transferred bi-directionally between frameworks that contain multiple online domains. However, using an OLS meta-learner to combine transferred base models with the locally learnt model often suffers from overfitting. In this section, the scalability of BOTL is considered by presenting results for BOTL, P-Thresh and MI-Thresh as the number of base models available to meta-learners increases.

Figure 5.1 shows the difference in R^2 predictive performance of the OLS meta-learner in comparison to using the model learnt via the underlying CDD alone, as a sudden drifting hyperplane data stream progresses. Combining base model predictions initially improves predictive performance. However, as each data stream progresses, the number of available base models increases as new models are learnt



(a) Difference in R^2 predictive performance.

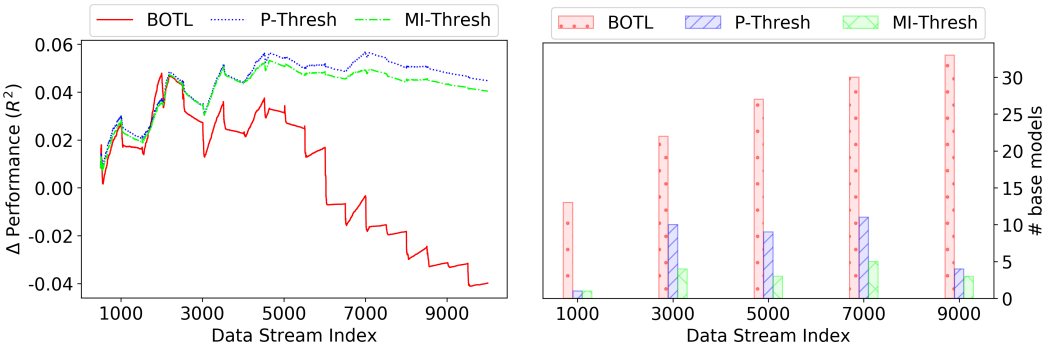
(b) Number of base models.

Figure 5.1: BOTL vs. CDDs: The difference in R^2 performance (a) between the OLS meta-learner in BOTL vs. the underlying CDDs of RePro, ADWIN, and AWPro, and the number of models used as base models (b) for a sudden drifting hyperplane data stream (SuddenA). Base models are learnt locally and transferred from 4 other sudden drifting hyperplane data streams.

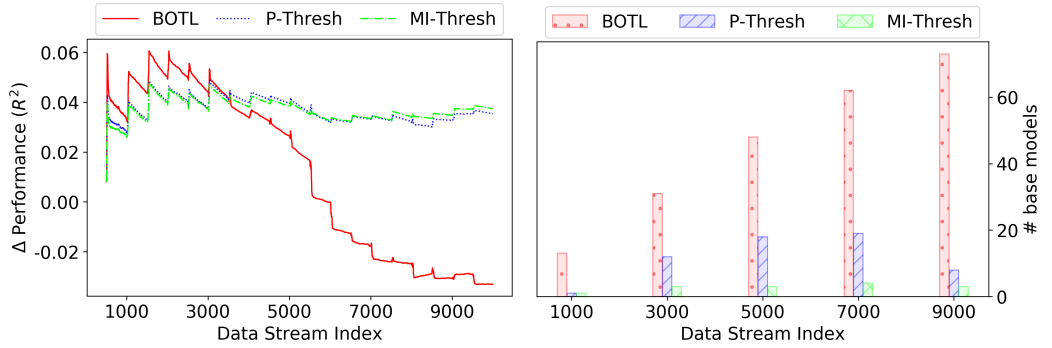
and transferred throughout the framework. Since the window of available data remains fixed in size, the likelihood of overfitting increases as the number of base models becomes large in comparison to the window of available data [25]. This means that as the number of base models grows, the meta-learner becomes more susceptible to overfitting, causing a reduction in predictive performance, eventually resulting in worse performance than using the current locally learnt model alone.

When using AWPro as the underlying CDD, a decrease in performance as the data stream progresses is not observed. This is because AWPro prioritises the reuse of historical models in the presence of recurring concepts, reducing the number of base models available to the meta-learner in comparison to ADWIN [55]. Additionally, unlike RePro, the models created by AWPro are not influenced by instances belonging to the previous concept [55] due to estimating the precise point of drift, and therefore unstable models are less likely to be created, and the detection of recurring concepts is not hindered. This means that when AWPro is used as the underlying CDD, there are fewer base models available to the meta-learner in comparison to when ADWIN and RePro are used as CDDs. These factors indicate why BOTL is able to outperform AWPro in the SuddenA, SuddenC, GradualA and GradualC datasets, as shown in Tables 5.2a, 5.2c, 5.3a and 5.3c, with statistical significance ($p < 0.01$).

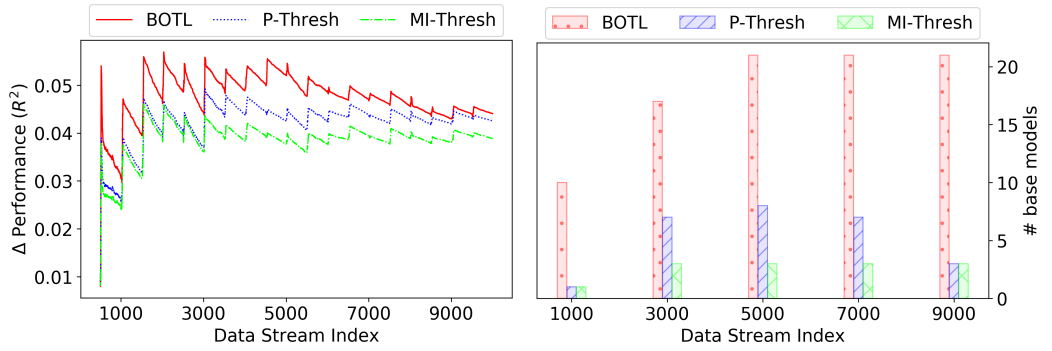
Figure 5.2 shows the difference in R^2 predictive performance between the OLS meta-learner in BOTL, P-Thresh and MI-Thresh in comparison to the underlying CDDs for a sudden drifting hyperplane data stream. P-Thresh and MI-Thresh



(a) RePro



(b) ADWIN



(c) AWPro

Figure 5.2: BOTL, P-Thresh and MI-Thresh vs. CDDs: The difference in R^2 performance (left) between the OLS meta-learner in BOTL, P-Thresh and MI-Thresh vs. the underlying CDDs of (a) RePro, (b) ADWIN, and (c) AWPro, and the number of models used as base models (right) for a sudden drifting hyperplane data stream (SuddenA). Base models are learnt locally and transferred from 4 other sudden drifting hyperplane data streams.

allow the number of base models used as input to the meta-learner to remain small, even when the number of base models received from other domains in the framework becomes large as the data stream progresses. Using P-Thresh and MI-Thresh to cull base models reduces the impact of overfitting when RePro and ADWIN are used as the underlying CDDs, presented in Figures 5.2a and 5.2b respectively. However, in Figure 5.2c, where AWPro is used as the underlying CDD for BOTL, a reduction in predictive performance is not observed as the data stream progresses, and therefore, when P-Thresh and MI-Thresh are used to reduce the number of base models used as input to the meta-learner, the increase in R^2 predictive performance obtained using P-Thresh and MI-Thresh compared to the underlying CDD is less than the increase observed by BOTL. This is because the meta-learner learnt in BOTL is able to generalise well, and therefore reducing the number of models used as input to the meta-learner via P-Thresh and MI-Thresh, removes base models that provide useful information to the meta-learner. However, the difference between the R^2 predictive performances of BOTL, P-Thresh and MI-Thresh decreases as the data stream progresses due to the increasing number of base models available to the meta-learner. Therefore, it is likely that BOTL will suffer from overfitting if data streams in the framework are long-lived and continue to receive base models from other domains, even when AWPro is used as the underlying CDD.

The BOTL framework can also become susceptible to overfitting when the number of domains in the framework is large. This can be seen in Figure 5.3 which shows the R^2 and PMCC^2 predictive performance for BOTL, P-Thresh and MI-Thresh for the following distance dataset in frameworks with an increasing number of domains. The average number of models used as input to the meta-learner is also shown in Figure 5.3. For frameworks with a small number of domains, BOTL, P-Thresh and MI-Thresh obtain similar predictive performances, outperforming their respective underlying CDD. As the number of domains increases, and the number of transferred models increases, the R^2 predictive performance drops drastically, however, the PMCC^2 performance gradually decreases to below the performance of each CDD. These characteristics are observed due to the nature of the R^2 and PMCC^2 performance metrics. PMCC^2 values range between $[0, 1]$, and therefore, when one domain in the framework performs poorly, it does not greatly impact the average PMCC^2 across all domains. R^2, on the other hand, ranges between $(-\infty, 1]$, and therefore, when one domain performs poorly, the average R^2 performance can be greatly impacted. This means that R^2 as a performance metric can be highly skewed by small periods of the meta-learner overfitting.

The difference between performance metrics, shown in Figure 5.3, indicates

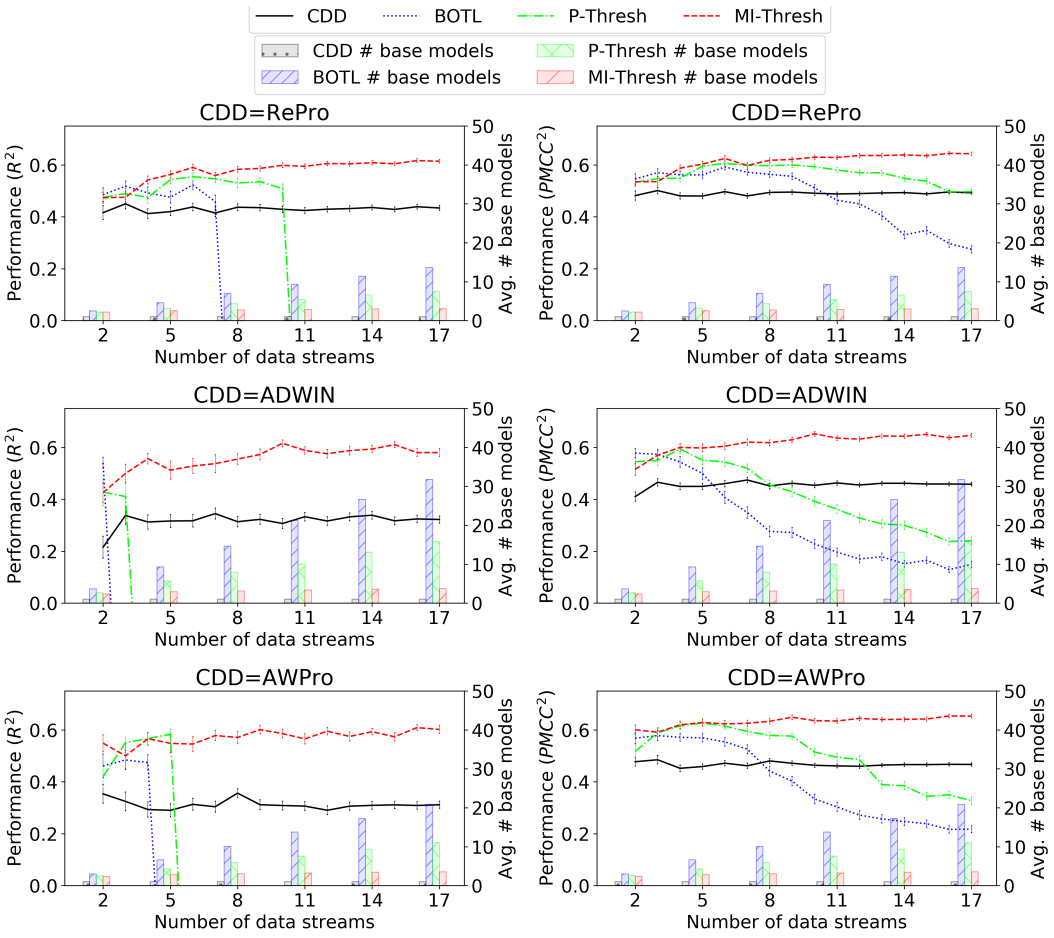


Figure 5.3: R^2 and PMCC^2 predictive performance, and number of base models used by BOTL meta-learners for increasing numbers of following distance data streams.

that BOTL and P-Thresh suffer from their respective OLS meta-learners overfitting the small window of available data when the number of models transferred is large. Culling base models using the performance of transferred models alone (P-Thresh) enables a larger number of domains to be used in the framework, however, this cannot be considered scalable since the performance of P-Thresh decreases below that of the CDD when more domains are added. MI-Thresh uses a more aggressive culling mechanism, using diversity alongside performance, ensuring that enough beneficial knowledge is retained to enhance the predictive performance in each domain, while minimising negative transfer and preventing the OLS meta-learner overfitting the small window of locally available data.

Additionally, Figure 5.3 shows the PMCC^2 and R^2 performance when ADWIN is used as the underlying CDD. Compared to when RePro and AWPro are used

as CDDs, the PMCC^2 and R^2 performances obtained by ADWIN reduce quickly, even when a small number of domains are included. This again, highlights the importance of selecting a CDD that reuses existing models in the presence of recurring concepts, instead of relearning and transferring duplicate models.

5.8 Summary

In this chapter, the BOTL framework has been presented, which allows knowledge to be learnt and transferred bi-directionally between online data streams. Using BOTL for online TL is beneficial since CDDs are used in each online data stream, allowing predictive models to be learnt to represent each of the individual concepts encountered in a domain. These predictive models are transferred between domains so that every receiving domain can benefit from the knowledge learnt about each individual source concept. Since both source and receiving domains are online, additional knowledge can be transferred when new concepts are encountered as each data stream progresses. This means that if a new concept is encountered in one domain, all other domains can benefit from the knowledge learnt about that concept, prior to encountering it themselves. Additionally, since knowledge transfer is bi-directional, every domain in the framework can benefit from the knowledge learnt in other domains. This is beneficial for real-world applications where a data rich source domain does not exist, and it is unlikely that the sole purpose of the application is to improve the predictive performance in a single domain. For example, if considering the use of online TL to tailor the functionality of an ACC, it is unlikely that the aim of the online TL technique would be to improve the predictive performance for a single driver. Instead, online TL should be used to improve the predictive performance for every driver in the framework, so that predictions can be tailored to each driver, but are aided by the knowledge learnt from other drivers.

The results presented in this chapter have shown that knowledge transfer can be used to obtain improved predictive performances in receiving domains through the use of an OLS meta-learner. However, when the number of models becomes large in comparison to the window of available data, the meta-learner can suffer from overfitting, caused by the curse of dimensionality. To prevent this, naïve culling mechanisms can be used, as shown by P-Thresh and MI-Thresh.

Although P-Thresh and MI-Thresh help to reduce the likelihood of overfitting, their success is dependent on culling parameter values being chosen that sufficiently reduce the number of models available to the meta-learner while retaining sufficient transferred knowledge to be of benefit to the receiving domain.

Selecting an appropriate subset of base models is one of the key challenges faced by the BOTL framework. This issue is not limited to online TL research, and is a common challenge considered in meta-learning and ensemble research [11, 12, 58]. Therefore, in Chapter 6, the process of selecting base models to be used as input to meta-learners or ensembles is investigated.

Chapter 6

Base Model Selection for Meta-Learners in Concept Drifting Data Streams

The problem of the meta-learner in BOTL overfitting is not unique to online TL frameworks. In fact, overfitting caused by the curse of dimensionality is frequently encountered by meta-learners and ensembles in online environments [11, 12, 36]. Therefore, in this chapter the more general problem of how to prevent meta-learners and ensembles from overfitting in online environments is addressed, regardless of whether online TL is used.

Learning in online data streams can be challenging since data availability may be limited if a rich history of observations cannot be retained [78], and as a result of concept drift [26, 108]. Meta-learners and ensembles are often used in online learning environments to improve predictive capabilities by combining models learnt historically throughout the data stream [9]. Historical models, referred to as base models, can be learnt to represent each concept encountered as the data stream progresses. However, as the data stream progresses, and more base models are learnt, meta-learners and ensembles can become prone to overfitting, particularly when data availability is limited and the number of base models is large (as seen in Chapter 5) [55].

To improve generalisation, a relevant yet diverse subset of base models can be selected to be used as input to the meta-learner or ensemble [11]. Base model selection is often achieved within offline environments using metrics such as predictive performance to indicate relevancy, and Mutual Information (MI) between base model predictions to indicate pairwise diversity [23]. However, determining rele-

vancy and diversity can be more challenging in online environments due to concept drift [12]. Encountering concept drift can impact the relevancy of a base model, and therefore requires the relevancy of base models to be continually re-evaluated as the data stream progresses. Many diversity metrics, such as MI, must also be recalculated due to their dependency on the current underlying distribution of data when identifying the covariance between model predictions. Repeatedly recalculating relevancy and diversity metrics may be computationally undesirable when using meta-learners or ensembles in data streams, particularly when drifts are encountered frequently or a large number of potential base models are available.

The contributions of this chapter are as follows.

- A novel diversity metric is proposed, which estimates the conceptual similarity of base models in concept drifting data streams.
- We show that conceptual similarity can be used to identify a diverse subset of base models using parameterised thresholding.
- Parameterless conceptual clustering is introduced, which uses conceptual similarity to cluster base models such that a relevant yet diverse subset of base models can be selected without requiring a user defined culling parameter.

The proposed approach for determining conceptual similarity uses the Principal Angles (PAs) between the subspaces in which each base model was created. Thus, the similarity between pairs of base models remains static in the presence of concept drift and does not require recalculation. Empirical results are presented, using the proposed base model selection techniques to identify a subset of base models to be used by the meta-learner in BOTL for the datasets introduced in Chapter 3. Base models are obtained using the three CDDs introduced in Chapter 4, specifically RePro [94], ADWIN [8], and AWPro [55]. BOTL [55], as introduced in Chapter 5, is used to transfer models bi-directionally between online data streams. The thresholding and clustering approaches proposed in this chapter are used to select a subset of base models from those learnt locally within a data stream, and received from other data streams for the OLS meta-learner in BOTL.

Conceptual similarity thresholding obtains predictive performances comparable to the existing approaches of performance and MI thresholding, denoted as P-Thresh and MI-Thresh in Chapter 5, while reducing the computational overhead associated with base models comparisons. Additionally, conceptual clustering achieves similar predictive performances without the need for a user defined culling parameter. Although conceptual clustering increases the computational overhead

in comparison to conceptual similarity thresholding due to the clustering of base models, it can be less computationally expensive than thresholding using diversity metrics that are not static, such as MI, when the number of base models is large.

The remainder of this chapter is organised as follows. Section 6.1 discusses the use of meta-learners and ensembles in online learning frameworks, highlighting existing approaches to selecting subsets of base models to prevent overfitting. A novel approach for estimating the conceptual similarity of base models in concept drifting data streams is defined in Section 6.2. Section 6.3 introduces parameterised thresholding and parameterless conceptual clustering as methods for selecting subsets of base models. In Section 6.4, further insight is provided into why the subset of base models used by a meta-learner should be relevant yet diverse. Section 6.5 provides details of the experimental set-up used to obtain the results presented in Section 6.6. Finally, Section 6.7 concludes this chapter and highlights how diversity can be used in online TL to reduce the number of models transferred throughout an online TL framework.

6.1 Ensembles, Meta-Learners and Model Selection

Three important issues must be addressed when using meta-learners or ensembles in online environments. First, base models must initially be created in online data streams. Second, base models must be combined to obtain an overarching prediction. Third, if the number of base models becomes large in comparison to the available data, a subset of base models must be selected to be used by the meta-learner or ensemble to prevent overfitting.

Chapter 2 discussed methods through which predictive models can be learnt from online data streams using incremental learners and CDDs, and combined using meta-learners and ensembles to improve predictive performance [9]. Meta-learners and ensembles often use weighting mechanisms to learn how much influence each base model should have on the overarching prediction [106]. In this chapter, we assume that base models are created using CDDs, representing the concepts historically encountered throughout a data stream, and that a meta-learner is used to combine base model predictions by learning weights using the predictions of base models over a recent window of instances as input to the meta-learner.

Increasing the number of base models used by the meta-learner increases its representational capacity. This allows a better approximation of the underlying distribution of observable data to be made [81], which decreases the training error of the meta-learner, known as the empirical risk. However, when the number of

base models becomes large, the meta-learner may overfit the window of observable data [25], which can increase the predictive error for unseen instances, known as the true risk [106]. This often occurs when the representational capacity of the meta-learner is too large to be able to effectively learn the meta-learner model parameters from the fixed sized window of observable data [14, 64, 81]. In such cases, the empirical risk can be a poor approximation of the true risk [14], which may lead to poor predictive performance for unseen instances. Therefore, to prevent overfitting, the representational capacity of the meta-learner can be reduced by selecting a subset of base models to be used as input. To ensure that the meta-learner makes effective predictions and generalises well for unseen instances, a relevant yet diverse subset of base models should be selected [11]. Relevancy and diversity metrics are considered in this chapter in order to select a subset of base models to be used as input to the OLS meta-learner in the BOTL framework. Further discussion of how the empirical risk of a meta-learner can be minimised, while remaining a good approximation of the true risk, through the use of relevancy and diversity, is presented in Section 6.4.

Identifying the best subset of base models is challenging in online environments since future concepts are unknown, a rich history of data often cannot be retained, and due to the presence of concept drift. Existing online ensemble techniques, such as Accuracy Weighted Ensemble (AWE) [82], Online Weighted Ensemble (OWE) [31] and Additive Expert ensemble (AddExp) [46], select base models using the recency of a model, the predictive performance, or a combination of the two, as indicators of relevancy. Using only the recency of a model as an indicator of relevancy may be undesirable in online data streams since a historical model may be more relevant than a recently learnt model due to recurring concepts [31, 46]. The diversity among base models in online environments is considered less frequently. Five diversity metrics for regression ensembles were presented by Dutta, namely the correlation coefficient between base model predictions, the covariance between model predictions, the pairwise Chi-square of model predictions, the standard deviation of predictions as a disagreement bound, and the MI between model predictions [23]. MI is a frequently used regression ensemble diversity metric, where the pairwise diversity of base models is measured using the MI between predictions of base models over recent observations [23, 30]. However, each of the measures of diversity proposed by Dutta [23] are dependent on the recent window of observations used to evaluate base models. This is problematic in concept drifting data streams and online TL environments since the diversity of base models that obtain similar predictive performances for some windows of observations, but have been learnt from different distributions, or learnt to represent different concepts, is not accounted

Table 6.1: Notation for conceptual similarity.

	Definition
α	Domain α, where $\mathcal{A}$ is used to denote a single source domain, and $\mathcal{B}$ denotes a receiving domain
X^α	Data stream in domain α, where $X^\alpha = \{x_1, \ldots, x_t, \ldots, x_n\}$
$x_t \in X^\alpha$	The t^{th} observed instance in X^α
Y^α	The response variable space in domain α
$y_t \in Y^\alpha$	The t^{th} response variable in Y^α
C^α	The set of concepts encountered in domain α
$c_i^\alpha \in C^\alpha$	The i^{th} concept encountered in domain α
$X_i^\alpha \in X^\alpha$	The data stream segment corresponding to concept c_i^α in domain α
W	Sliding window of $\|W\|$ instances, $W = \{x_{t-\|W\|}, \ldots, x_t\}$
$f_i^\alpha : X_i^\alpha \rightarrow Y_i^\alpha$	Model i learnt in domain α
U_i	Principal Components of the window used to learn model f_i
$\mathcal{M}$	Set of stable, locally learnt and transferred models
$\mathcal{M}'$	A subset of stable, locally learnt and transferred models, $\mathcal{M}' \subseteq \mathcal{M}$
$F^{\mathcal{M}} : X^\alpha \rightarrow Y^\alpha$	Meta-learner in domain α using models in $\mathcal{M}$
x_t^*	Meta instance of base model predictions for instance x_t
$\hat{y}_t^*$	Prediction using $F^{\mathcal{M}'}(x_t)$ where $\mathcal{M}' \subseteq \mathcal{M}$

for [30]. Additionally, due to the dependency on a window of recent observations, measuring diversity by the level of disagreement between base model predictions can be influenced by concept drift, and therefore must be recalculated as the data stream progresses.

To overcome this, a novel method of measuring the diversity of base models for meta-learners in online environments is introduced. This method allows the pairwise conceptual similarity between base models to be estimated independently of the current distribution of observable data. This is achieved using the similarity between the underlying subspaces in which each base model was learnt to obtain a measure of diversity that remains static as the data stream progresses, even in the presence of concept drift.

6.2 Conceptual Similarity

In this section, a novel method for estimating the conceptual similarity between pairs of base models is presented. The conceptual similarity determines the diversity among base models to be used by meta-learners in online environments. For ease of reference, Table 6.1 presents the notation used in this chapter.

Given a set, $\mathcal{M}$, of k base models and a data stream, X, the meta-learner, F, must learn a predictive function, mapping the predictions of base models, $f_i(x_t)$, for instance $x_t \in X$ at time t to the response variable y_t, such that $F\left(x_t^*\right) \rightarrow y_t$, where $x_t^* = \langle f_1(x_t), \ldots, f_k(x_t) \rangle$. Due to the presence of concept drift, the meta-learner must re-weight base model predictions as the data stream progresses. Weights are learnt using the predictions of base models over a sliding window of recent observations, W, which are then used to combine base model predictions for unseen instances. As the data stream progresses, new models are learnt, and are transferred in the case of online TL, so that the meta-learner can use previously learnt knowledge to aid predictive performance.

To prevent the meta-learner from overfitting, a relevant yet diverse subset of base models must be selected. Although the relevancy of a base model is dependent on the current distribution of data observable in W, the pairwise diversity of two base models can be considered independently of W. To assess diversity, it may be desirable to measure model similarity by comparing the underlying distributions of all data belonging to each of the concepts that the base models were learnt to represent [30]. However, for many real-world applications, it may not be feasible to retain a rich history of data belonging to each concept, and its transfer may be impractical due to communication overheads when knowledge is transferred via a network in an online TL framework. Instead, conceptual similarity is approximated by considering the subspace similarity using the PAs between the data in which each base model was trained. The PAs between the subspaces in which a pair of base models were learnt is defined as follows [10, 29, 45].

Definition 1 (Principal Angles (PAs) between subspaces)**.** *For two base models i and j, let $X_i \in \mathbb{R}^{N \times m}$ and $X_j \in \mathbb{R}^{N \times m}$ denote the subspaces in which they were learnt, containing N instances, $m-1$ features and the associated response variable, and let $U_i \in \mathbb{R}^{N \times p}$ and $U_j \in \mathbb{R}^{N \times q}$ represent orthonormal bases of X_i and X_j respectively. Using Singular Value Decomposition (SVD) we obtain $SVD(U_i^\intercal U_j) = A\Sigma B$, where A and B are the unitary matrices and $\Sigma \in \mathbb{R}^{p \times q}$ is the diagonal matrix of singular values, $\Sigma = diag([s_1, \ldots, s_r])$, where $r = min(p, q)$ [45]. The PAs between subspaces are given by*

$$\overrightarrow{\Theta}(X_i, X_j) = [\arccos(s_1), \ldots, \arccos(s_r)]. \tag{6.1}$$

6.2.1 Estimating Conceptual Distance

The conceptual similarity between the underlying concepts a pair of base models were learnt to represent can be estimated using PAs. In order to obtain the PAs

in Definition 1, an orthonormal representation of the subspaces in which each base model was learnt is required. The Principal Components (PCs) of a subspace is an orthonormal representation, and can be obtained using Singular Value Decomposition (SVD) on the covariance matrix,

$$\text{SVD}(X_i^\intercal X_i) = U_i \Sigma V, \tag{6.2}$$

such that $U_i \in \mathbb{R}^{N \times N}$ are the PCs, and Σ their singular values.

Therefore, to estimate the conceptual similarity of base models, both the model learnt to represent a concept, and an orthonormal representation of the training data must be stored in online learning frameworks, and transferred between data streams when using online TL. In these online learning frameworks, the memory and communication overhead required to store and transfer orthonormal representations of the subspace associated with each base model, i, can be reduced by retaining only the first p PCs that capture 99.9% of the variance of the original training data, X_i. The diagonal matrix of singular values, Σ, obtained through SVD in Equation 6.2, can be used to determine the number of PCs to retain.

Let $\Sigma = \text{diag}([S_{11}, \ldots, S_{NN}])$ represent the diagonal matrix of singular values in Equation 6.2. The number of PCs, p, that capture 99.9% of the variance can be identified using

$$1 - \frac{\sum_{i=1}^{p} S_{ii}}{\sum_{j=1}^{N} S_{jj}} \leq 0.001. \tag{6.3}$$

The percentage of variance captured by the p PCs can be decreased to reduce the impact of noise in the data stream. This allows the orthonormal representations, and the PAs between them, to be more robust to noise. However, alternative methods of obtaining orthonormal representations of each subspace, such as Laplacian PCA [102] and Robust PCA [90], can be used to improve robustness to noise and outliers in noisy data streams [42]. Once p has been identified, the matrix of N PCs, $U_i \in \mathbb{R}^{N \times N}$, can be reduced such that only the first p PCs are retained, $\tilde{U}_i \in \mathbb{R}^{N \times p}$. Since the reduced orthonormal representation, $\tilde{U}_i$, captures 99.9% of variance, for ease of notation throughout the remainder of this chapter, U_i is used to denote the reduced matrix of PCs. Therefore, from this point forward, $U_i \in \mathbb{R}^{N \times p}$.

Using Definition 1 the PAs between the subspaces in which two base models i and j were learnt, X_i and X_j, can be calculated using the reduced PCs, U_i and U_j, as orthonormal bases. The conceptual distance between base models i and j is defined as follows.

Definition 2 (Conceptual distance between base models)**.** *Let* $\overrightarrow{\Theta}(X_i, X_j) \in \mathbb{R}^r$ *be*

the vector of PAs, obtained using Definition 1, between the PCs U_i and U_j for X_i and X_j respectively. The conceptual distance between base models i and j is defined as

$$d(i,j) = \frac{1}{r}\left(\sum_{h=1}^{r}(1-\cos(\theta_h))\right) = 1 - \frac{1}{r}\left(\sum_{h=1}^{r}\cos(\theta_h)\right), \tag{6.4}$$

where $\theta_h \in \overrightarrow{\Theta}(X_i, X_j)$ and $r = min(p,q)$. Therefore, $d(i,j) \to 0$ when base models i and j have been learnt from similar subspaces, and $d(i,j) \to 1$ when subspaces are dissimilar.

6.2.2 Estimating Conceptual Similarity

Finally, to estimate the conceptual similarity among base models, an affinity matrix is created.

Definition 3 (Conceptual similarity)**.** *The affinity matrix, $\Delta \in \mathbb{R}^{|\mathcal{M}|\times|\mathcal{M}|}$, where $|\mathcal{M}|$ is the number of available base models, is given by*

$$\Delta_{ij} = \begin{cases} exp(\frac{-d(i,j)^2}{\tilde{d}_i\tilde{d}_j}) & \text{if } i \neq j \\ 0 & \text{otherwise,} \end{cases} \tag{6.5}$$

where $\tilde{d}_i$ and $\tilde{d}_j$ are local scaling parameters, which allows the conceptual difference between base models to be scaled by the surrounding neighbourhoods of i and j [100] such that $\tilde{d}_i = d(i,k)$, where k is the k^{th} nearest neighbour of base model i.

Zelnik-Manor and Perona [100] suggest that a value of $k = 7$ yields good results for local scaling, even for high-dimensional image segmentation and document classification tasks. However, since the dimensionality of the affinity matrix for base models is likely to be low in comparison to the affinity matrices used by Zelnik-Manor and Perona [100], the impact of using local scaling parameter values between $k = 2$ and $k = 7$ are considered. The results of using different local scaling parameters are briefly discussed in Section 6.6, and show that these scaling parameter values obtain similar results, and therefore $k = 7$ is used to remain consistent with [100].

Local scaling is used to allow better affinities to be obtained when the density of conceptually similar base models varies [100]. Although this requires a parameter, k, to be defined, it has been shown that parameter tuning is not typically needed for local scaling to perform well [42]. Alternatively, other scaling techniques could be used, such as density-aware kernels [42, 101], to amplify intra-cluster similarities in order to account for locally dense areas of conceptually similar base models [42].

Algorithm 9: Concept similarity thresholding.

Input: $\mathcal{M}$, f_i, U_i, Dist, λ_{CS}
for $j \in \mathcal{M}$ **do**
    $\text{Dist}_{i,j} = \text{getDistances}(U_i, \mathcal{M}.\text{getPCs}(j))$ using Def 1 and Def 2
    $\text{Dist}_{j,i} = \text{Dist}_{i,j}$
iAffinities = getAffinities(Dist, i, $\mathcal{M}$) using Def 3
if $\forall j \in iAffinities < \lambda_{CS}$ **then**
    Add $\{f_i, U_i\}$ to $\mathcal{M}$
else
    $\forall j \in \mathcal{M}$: Remove $\text{Dist}_{i,j}$ and $\text{Dist}_{j,i}$ from Dist
return $\mathcal{M}$

Using Definition 3, the affinity matrix, Δ, is created, where element $\Delta_{ij} \to 1$ when base models i and j are conceptually similar, and $\Delta_{ij} \to 0$ when they are dissimilar.

6.3 Base Model Selection

Two methods for selecting a relevant yet diverse subset of base models are presented in this section, namely (i) parameterised thresholding and (ii) parameterless conceptual clustering, using the estimated conceptual similarity of base models as a diversity metric.

6.3.1 Parameterised Thresholding

Parameterised thresholding is the first method of base model selection introduced in this section. This approach follows a similar methodology to MI-Thresh, presented in Chapter 5, but uses the conceptual similarity of base models, instead of MI, as a diversity metric.

As a data stream progresses, new models are made available when encountering concept drifts, or are received from other domains via online TL. The pairwise conceptual similarities between a new model, f_i, and existing base models, $f_j \in \mathcal{M}$, are calculated using the average PA between the two subspaces in which each model was learnt (Definitions 1 and 2), to obtain a locally scaled affinity metric (Definition 3). Given a user defined culling threshold, λ_{CS}, models are added to $\mathcal{M}$ if no existing base model in $\mathcal{M}$ is considered conceptually similar. This means that in order for a new model, f_i, to be used as input to the meta-learner, its affinity to all other available base models, $\forall j \in \mathcal{M} : \Delta_{ij}$, is less than the conceptual similarity threshold, λ_{CS}. Since conceptual similarity is calculated independently of the un-

derlying distribution of data, a new model, f_i, with an affinity to an existing base model greater than λ_{CS}, can be discarded and does not need to be reconsidered as input to the meta-learner. This process, shown in Algorithm 9, obtains a diverse subset of base models.

Using a static diversity metric such as conceptual similarity prevents the need to re-evaluate base models since they do not need to be reconsidered once culled. However, the relevancy of remaining base models is dependent on the current concept. To ensure that a relevant subset of the remaining base models are used by the meta-learner, the predictive performance of each base model is evaluated to create the subset of base models, $\mathcal{M}'$, that are used by the meta-learner to make predictions for unseen instances. The predictive performance of the models that remain after executing Algorithm 9, $\mathcal{M}$, are evaluated over the current sliding window of data, W. Those that achieve an R^2 performance greater than a performance threshold, λ_{perf}, are included in $\mathcal{M}'$. Base models that achieve an R^2 performance less than λ_{perf} are temporarily excluded from the meta-learner until a concept drift is encountered. This ensures that the subset of base models used by the meta-learner remains relevant to the current concept.

Selecting culling parameters can be challenging since they are dependent on the presence of noise in the underlying data stream, the number of possible base models, and their separability for a given similarity metric [55]. Values that promote aggressive culling may result in discarding base models that are beneficial to the meta-learner, while values that are less aggressive may not prevent overfitting [55].

To overcome this, culling parameters could be updated as the data stream progresses using cross-validation. However, this would require the meta-learner to be validated for various threshold values over a small window of data. This increases computation, and the predictive performance obtained through cross-validation would likely be an overestimate since validation splits are unlikely to be independent due to the dependencies between consecutive instances within online data streams [34]. This means that considerable domain expertise is required to select the culling parameters for meta-learners in online environments.

6.3.2 Parameterless Conceptual Clustering

To prevent the need for domain expertise, parameterless conceptual clustering is introduced. As with conceptual similarity thresholding, when new models are learnt in a data stream or received via online TL, the affinity between existing base models, $f_j \in \mathcal{M}$, and the new model, f_i, must be calculated using Definitions 1, 2 and 3. Once the affinity matrix, Δ, has been obtained it can be considered as a fully

Algorithm 10: Conceptual clustering base model selection.

Input: W, $\mathcal{M}$, f_i, U_i, Dist, Δ
for $j \in \mathcal{M}$ **do**
    $\text{Dist}_{i,j}$ = getDistances($U_i, \mathcal{M}$.getPCs(j)) using Def 1 and Def 2
    $\text{Dist}_{j,i} = \text{Dist}_{i,j}$
Δ=getAffinityMatrix(Dist) using Def 3
clusterGroups = STSC(Δ) [100]
for $c \in$ *clusterGroups* **do**
    Add bestPerformingModel(c, W) to $\mathcal{M}'$
    if *bestPerformingModel*(c, W) *is* f_i **then**
        Also add secondBestPerformingModel(c, W) to $\mathcal{M}'$
return $\mathcal{M}'$

connected graph, where nodes represent available base models, and edges represent their pairwise conceptual similarity. This allows graph clustering algorithms, such as Spectral Clustering (SC), to be used to identify groups of conceptually similar base models. SC algorithms typically have complexity $O(n^2)$ to create the similarity matrix, and $O(n^3)$ for spectral analysis [93], where n is the number of available base models, $|\mathcal{M}|$. Therefore, it is important to use a static similarity metric, such as conceptual similarity, so that updating the similarity matrix and clustering available base models is only required when a new base model is learnt or received from another data stream. Using metrics such as MI would require both the similarity matrix and spectral analysis to be repeatedly updated due to the dependency on the current distribution of observable data. The computational complexity of this makes the use of metrics such as MI for clustering similar base models infeasible for most real-world applications.

Self-Tuning Spectral Clustering (STSC) is used to create clusters of similar models. STSC is a well known SC algorithm [100], which allows the number of clusters to be determined automatically. STSC uses the local scaling in Equation 6.5, and incrementally rotates the eigenvectors traditionally obtained from SC to estimate the number of clusters [100]. Automatically determining the number of clusters is advantageous when considering the diversity among base models in online environments, particularly in online TL, where it is not known how many concepts will be encountered in each data stream, or how similar the concepts learnt from different data streams will be.

Once clusters of conceptually similar base models have been identified using Algorithm 10, a subset of relevant yet diverse base models, $\mathcal{M}'$, is created by selecting one base model from each cluster. To achieve this, the predictive performance

of base models in a cluster over the current window of observable data, W, is used as an indicator of relevancy. Therefore, the base model with the highest R^2 performance in each cluster is selected (Algorithm 10: lines 6 – 7). In the case where the best performing model in a cluster is the model, f_i, which has been learnt to represent the current concept, the second best performing model in that cluster is also added to $\mathcal{M}'$ (Algorithm 10: lines 8 – 9). This ensures that the meta-learner can benefit from the additional support of a model learnt historically, or from another data stream, that is conceptually similar to the current concept.

6.4 Minimising Online Meta-Learner Risk Using Relevancy and Diversity

Before presenting experimental results for the base model selection strategies introduced in Section 6.3, the impact of the number of available base models on the the empirical risk and true risk of a meta-learner is considered. This highlights the importance of selecting a subset of relevant yet diverse base models as input to a meta-learner in an online environment.

6.4.1 Increasing the Number of Base Models

To consider the effect of increasing the number of base models available to a meta-learner, the OLS meta-learner used in BOTL is used as an example. As discussed in Section 5.4, increasing the number of base models used by the meta-learner reduces the empirical risk. This was demonstrated in Section 5.4 by considering the loss of a constrained meta-learner where the current locally learnt base model was given a weight 1, while all other base models were given a weight 0. Since the optimisation problem used to learn the weights of the OLS meta-learner is convex, any constraints added increases the empirical risk. This occurs because adding constraints reduces the representational capacity of the meta-learner, preventing complex underlying distributions in the data stream from being captured.

However, as discussed in Section 5.7, since the window of available data remains fixed in size, the likelihood of overfitting increases as the number of base models becomes large in comparison to the window of available data [25], previously illustrated in Figure 5.1 (Chapter 5). Using the principal of Empirical Risk Minimisation (ERM) [14], and the notation in Table 6.2, factors that impact a meta-learner's ability to generalise well for unseen instances in a data stream between concept drifts can be considered.

Table 6.2: Notation for Empirical Risk Minimisation.

	Definition
$\mathcal{R}(\vec{w})$	True risk of model with parameters $\vec{w}$
$z_t = (x_t, y_t)$	Instance x_t at time t and respective response variable y_t
$Q(z_t, \vec{w})$	Loss function of a model parameterised by $\vec{w}$ for instance z_t
$\vec{w}_0$	Optimal parameters for a given function
$Z_{\|W\|}$	Training sample of size $\|W\|$
$\mathcal{R}_{\text{emp}}\left(\vec{w}_{\|W\|}\right)$	Empirical risk of model with parameters $\vec{w}_{\|W\|}$ learnt over $Z_{\|W\|}$
$\vec{w}^*_{\|W\|}$	Model parameters that minimise the empirical risk over $Z_{\|W\|}$

6.4.2 ERM for Meta-Learners in Online Environments

In ERM, the overarching aim is to learn a function that maps the input space, X, to the response variable space, Y, drawn from some unknown distribution. If the distribution was known, then the function parameters, $\vec{w}$, that minimise the risk, $\mathcal{R}(\vec{w})$, could be found such that

$$\begin{aligned} \mathcal{R}(\vec{w}) &= \int Q(z, \vec{w}) dD \\ &= \sum Q(z, \vec{w}) \\ \mathcal{R}(\vec{w}_0) &= \min_{\vec{w} \in \mathcal{W}} \mathcal{R}(\vec{w}) \,, \end{aligned} \tag{6.6}$$

where z is an instance and response variable pair, $z = (x, y)$, and $Q(z, \vec{w})$ is the loss function, parameterised by $\vec{w}$, on instance z. For example, the squared loss of instance x_t is $Q(z, \vec{w}) = (y_t - f(x_t))^2$ where model f has parameters $\vec{w}$. Therefore, $\mathcal{R}(\vec{w})$ is the risk, or loss, of a model parameterised by $\vec{w}$. $\mathcal{R}(\vec{w}_0)$ is used to denote the risk of a model with optimal parameters, $\vec{w}_0$, which minimise the risk over the known distribution, D. However, since the distribution is unknown, ERM approximates the optimal risk by evaluating the loss over a window, W, of training samples, $Z_{|W|}$, to obtain the empirical risk, $\mathcal{R}_{\text{emp}}\left(\vec{w}^*_{|W|}\right)$. The empirical risk is defined as,

$$\mathcal{R}_{\text{emp}}\left(\vec{w}^*_{|W|}\right) = \sum_{i=1}^{|W|} Q(z_i, \vec{w}^*_{|W|}). \tag{6.7}$$

The empirical risk is inherently biased towards the training sample, $Z_{|W|}$, and therefore the function learnt often underestimates the risk of the same function when used on unseen instances of data belonging to the same concept. The risk over

unseen instances is known as the true risk, and is denoted by $\mathcal{R}\left(\vec{w}^*_{|W|}\right)$ [14]. This means that for a given training sample

$$\mathcal{R}_{\text{emp}}\left(\vec{w}^*_{|W|}\right) < \mathcal{R}\left(\vec{w}^*_{|W|}\right).$$

Due to the randomness of $Z_{|W|}$, the empirical risk and true risk, $\mathcal{R}_{\text{emp}}\left(\vec{w}^*_{|W|}\right)$ and $\mathcal{R}\left(\vec{w}^*_{|W|}\right)$, are also random sequences, and the law of large numbers states that the average converges to its expected value as the number of samples grows large [14, 64, 81]. Therefore, as $|W| \to \infty$,

$$\begin{aligned} \mathcal{R}\left(\vec{w}^*_{|W|}\right) &\to \mathcal{R}(\vec{w}_0)\,, \\ \mathcal{R}_{\text{emp}}\left(\vec{w}^*_{|W|}\right) &\to \mathcal{R}(\vec{w}_0)\,. \end{aligned} \tag{6.8}$$

As the number of training samples grows, a better estimate of the parameter values $\vec{w}^*_{|W|}$ can be obtained for a fixed number of parameters k, where $|\vec{w}^*_{|W|}| = k$, such that the empirical risk tends to the true risk, $\mathcal{R}_{\text{emp}}\left(\vec{w}^*_{|W|}\right) \to \mathcal{R}\left(\vec{w}^*_{|W|}\right)$. To obtain a set of parameters that are a good approximation of the optimal parameters, such that $\vec{w}^*_{|W|} \to \vec{w}_0$, the representational capacity of the model must be increased to encapsulate more complex underlying distributions in the data stream [14]. In turn, increasing the representational capacity of the model increases the number of parameters in $\vec{w}^*_{|W|}$ which must be learnt. However, by increasing the complexity of the model, more training samples are required in order to ensure that the empirical risk, $\mathcal{R}_{\text{emp}}\left(\vec{w}^*_{|W|}\right)$, is a good approximation of the true risk, $\mathcal{R}\left(\vec{w}^*_{|W|}\right)$ [64].

This means that to ensure a learnt model generalises well and approximates the model with optimal parameters, $\vec{w}_0$, the number of training samples and the model complexity must grow as a function of one another to guarantee convergence [64]. However, in online settings, it may not be feasible to retain a large number of historical instances. Additionally, due to the dynamic nature of learning in online environments, even when a large history of instances can be retained, a concept drift may be encountered prior to observing sufficient instances to estimate the optimal parameters, $\vec{w}_0$. In the BOTL framework, the meta-learner is trained on a window of instances $W = \{x_{t-|W|}, \ldots, x_t\}$, where $|W|$ is small (Chapter 5) [54]. As the data stream progresses, the number of available base models, learnt locally or transferred from other data streams, increases, which increases the complexity of the meta-learner, thereby reducing the empirical risk. However, since the window of historical data remains small, the likelihood of the meta-learner having poor generalisation increases as more base models are made available. Therefore, the conditions

under which the probability that the empirical risk, $\mathcal{R}_{\text{emp}}\left(\vec{w}^*_{|W|}\right)$, is greater than the true risk, $\mathcal{R}\left(\vec{w}^*_{|W|}\right)$, plus some ϵ, is bounded, must be considered, such that

$$\mathbb{P}\left[|\mathcal{R}_{\text{emp}}\left(\vec{w}^*_{|W|}\right) - \mathcal{R}\left(\vec{w}^*_{|W|}\right)| > \epsilon\right]. \tag{6.9}$$

Vapnik [81] showed that the bound on the empirical risk in Equation 6.9 can be written as

$$\mathcal{R}\left(\vec{w}^*_{|W|}\right) \leq \mathcal{R}_{\text{emp}}\left(\vec{w}^*_{|W|}\right) + \varphi\left(\frac{n}{h_k}\right), \tag{6.10}$$

where $\varphi(\frac{n}{h_k})$ is a confidence interval, dependent on the ratio between the number of training samples, n, and the Vapnik-Chervonenkis (VC) dimension, h_k, which is a measure of the complexity of a model. Therefore, although a complex model may have a low empirical risk, the confidence interval may be large when the ratio of training samples to model complexity is large [81]. In such cases, a model is said to overfit the training data. In order to reduce the right-hand side of Equation 6.10, a small confidence interval is required, which would indicate that a model generalises well on unseen instances. To minimise the confidence interval, $\varphi(\frac{n}{h_k})$, a model with a small VC dimension, h_k, must be learnt. However, models with a small VC dimension have poorer representational capacity, thereby increasing the empirical risk, $\mathcal{R}_{\text{emp}}\left(\vec{w}^*_{|W|}\right)$ [81]. This introduces a trade off between the confidence interval and representational capacity of the model. In order to obtain a model that generalises well, we must simultaneously find the VC dimension that minimises the confidence interval, and the parameter values that minimise the empirical risk [14, 64, 81].

To identify the number of base models that should be used, the meta-learner must repeatedly solve this optimisation problem every time a concept drift is encountered, or a model is received via online TL. For many online learning applications, solving this optimisation problem is not possible, or practical, due to its computational complexity. Therefore, alternative approaches to decrease the complexity of the meta-learner must be considered in order to reduce the confidence interval, such as selecting a subset of the available base models to be used as input to the meta-learner.

6.4.3 Improving Generalisation for Meta-Learners in Online Environments

Overfitting caused by increasing the representational capacity of a meta-learner is known as the curse of dimensionality [25]. If future concepts were known, the curse of dimensionality could be avoided by discarding base models that are known not

to be beneficial to the meta-learner for current and future concepts. However, in online environments future concepts are not known prior to learning. This means that subsets of base models must be selected by repeatedly evaluating the set of models learnt locally, and transferred when using online TL, as the data stream progresses.

In online ensembles, one approach to solving this challenge is to only use the k most recently learnt models as base models in the ensemble [30, 46]. However, in many real-world environments that encounter recurring concepts, recency may not be a good indicator of usefulness [30]. Instead, other feature selection and ensemble pruning approaches must be used to determine which models may be most beneficial. Ensemble pruning is based on the principle that combining the predictions of an appropriate subset of base models will provide improved predictive capabilities over combining all base models, as exemplified in bagging and boosting offline ensembles [106]. Therefore, ensemble pruning techniques can be used to select a subset of base models for meta-learners in online environments. Feature selection techniques can be applied by considering the predictions of each base model as input features to the meta-learner. These meta-features can be evaluated to consider how useful they are for predicting the current concept [44].

In an ensemble of regressors there is a bias-variance-covariance trade-off [11, 79], where the generalisation error, or the expected error, denoted as, $\mathbb{E}[\cdot]$, of an ensemble of equally weighted regressors is:

$$\mathbb{E}\left[(F^{\mathcal{M}} - y)^2\right] = bias^2 + \frac{1}{M} var + \left(1 - \frac{1}{|\mathcal{M}|}\right) covar, \tag{6.11}$$

where

$$\begin{aligned} bias &= \frac{1}{|\mathcal{M}|} \sum_i \left(\mathbb{E}\left[f_i\right] - y\right), \\ var &= \frac{1}{|\mathcal{M}|} \sum_i \mathbb{E}\left[f_i - \mathbb{E}\left[f_i\right]\right]^2, \text{ and} \\ covar &= \frac{1}{|\mathcal{M}|(|\mathcal{M}| - 1)} \sum_i \sum_{i \neq j} \mathbb{E}\left[f_i - \mathbb{E}\left[f_i\right]\right] \left(f_j - \mathbb{E}\left[f_j\right]\right). \end{aligned}$$

This bias-variance-covariance decomposition also holds for non-uniformly weighted ensembles [11], and therefore the bias and variance of base models, and the covariance between them, can impact the generalisation ability of a meta-learner when base models are not equally weighted. Factors such as these can be considered by evaluating base models using metrics to determine which models should be used.

Equation 6.11 indicates that base models should be relevant yet diverse in order to prevent the meta-learner overfitting.

6.4.4 Evaluating Base Models

When undertaking ensemble pruning in offline settings, it is desirable to obtain a relevant yet diverse subset of base models [11]. However, within non-stationary environments base model diversity is rarely accounted for. The performance and diversity of model predictions can be used to cull base models [55], as used in Chapter 5. Using the R^2 predictive performance of each base model on the current window of data available to the meta-learner allows models that make poor predictions to be removed. However, using performance alone as a metric to cull base models may not sufficiently reduce the number of models, allowing the meta-learner to overfit when the number of potential base models is high [55]. The covariance term, *covar*, in Equation 6.11, accounts for the pairwise difference of base models [69], which relates to their diversity. As the number of base models in the ensemble, $|\mathcal{M}|$, increases, the generalisation error decreases due to the variance term, *var*. However, increasing the number of base models can cause the covariance term, *covar*, to also increase. When simply using the performance of base models as a culling metric, the reduction of the covariance term in Equation 6.11 is not considered. To prevent the covariance term from significantly increasing, base models must be selected that have small, or negative covariance [69].

To account for the covariance term in Equation 6.11, the diversity of base models can be used to remove those that exhibit high covariance. To achieve this, the pairwise MI can be measured between the predictions of each of the base models on the current window of data available to the meta-learner. When a pair of base models have high MI, the base model with the lower predictive performance can be culled from the model set [55]. This reduces the number of redundant models used by the meta-learner, since including models with similar predictions provides no additional information and increases the covariance term in Equation 6.11.

Measuring diversity through the level of disagreement between base model predictions is a common approach in existing online ensemble pruning research [30]. However, this means that the diversity of base models must be recalculated as new instances are observed in the data stream. Although diversity can be estimated using these metrics [23], it does not guarantee that the underlying distributions of data from which each base model was created are diverse [23]. Additionally, disagreement in a regression setting can be highly skewed by a small number of differing predictions made by base models that have been learnt from conceptually

similar distributions of data. Instead, diversity among the concepts learnt by each base model can be considered by estimating conceptual similarity.

6.5 Experimental Set-Up

To evaluate the effectiveness of using conceptual similarity as a metric for base model selection techniques, parameterised thresholding and conceptual clustering are used to obtain a subset of base models for the OLS meta-learner in the BOTL framework. Using the datasets detailed in Chapter 3, RePro [94], ADWIN [8] and AWPro [55] are used as the underlying CDDs to obtain Support Vector Regressors (SVRs) as base models from the concept drifting data streams. SVRs have been chosen to create base models since they have the representational capacity to model the underlying concepts encountered in each of the different data stream types. However, other regression models can be used since both the BOTL framework and base model selection techniques are model agnostic. Section 6.6 presents the results obtained when all base models are created using SVRs. Similar results are obtained in frameworks that use Ridge Regressors (RRs), and a combination of RRs and SVRs, as base models. These results are presented in Appendices A and B respectively. In order to estimate the conceptual similarity between base models, both the model, f_i, which has been learnt to represent the current concept, and the reduced PCs, U_i, obtained through Definition 1 and Equation 6.3 in Chapter 5, are transferred between domains.

6.5.1 Baseline Approaches

To empirically evaluate the effectiveness of estimating conceptual similarity as a diversity metric for selecting a subset of base models, existing ensemble pruning and meta-learner model selection techniques were considered for baseline approaches. However, most online ensembles that can be used in regression settings, such as AWE [82], OWE [31] and AddExp [46], combine base models using weighted averaging, where weights are bound between 0 and 1 [106]. The use of bounded weights introduces an assumption that the response variable each base model was learnt to predict have a consistent range of values. This assumption may not be valid when learning in real-world environments. For example, when predicting the desired heating temperature for a smart home heating system, base models learnt over summer months may have different ranges of response variables in comparison to the base models learnt over winter months. Since the future distribution of the response variable is unknown in an online data stream, the response variable cannot be

normalised to ensure that all base models are learnt over consistent ranges. Therefore, bounding base model weights to be between 0 and 1 may lead to inaccurate predictions. Instead, an OLS meta-learner is used to combine base models, since weights are not bound between 0 and 1, and the underlying techniques employed by existing online ensembles to prune base models are used as baseline approaches. For example AWE [82] and AddExp [46] prune base models using their predictive performance on the current window of observable data, and therefore a similar technique, which evaluates the predictive performance of base models, can be used to obtain a subset of base models to be input to the OLS meta-learner. To achieve this, the BOTL framework and the two variants of BOTL that implement model culling strategies, P-Thresh and MI-Thresh, introduced in Chapter 5, are used as baseline approaches [54, 55].

In addition to the BOTL variants, the underlying CDD is used as a default baseline for what can be achieved without the use of a meta-learner to combine base model predictions. This allows the benefits and drawbacks of using meta-learners with differing base model selection techniques to be considered, and highlights the importance of base model selection strategies when the number of base models becomes large, which can be impacted by the choice of the underlying CDD.

6.6 Experimental Results

To evaluate the use of conceptual similarity as a diversity metric for selecting base models, the base model selection techniques proposed in Section 6.3 are used by the OLS meta-learner in BOTL. Parameterised thresholding is considered with the use of three metrics, namely predictive performance (P-Thresh), MI and predictive performance (MI-Thresh), and conceptual similarity and predictive performance (CS-Thresh). Table 6.3 presents the notion used by P-Thresh, MI-Thresh and CS-Thresh to denote their respective threshold parameters. These parameterised thresholding approaches are then compared to parameterless conceptual clustering (CS-Clust). The predictive performances of each approach are compared to determine their effectiveness, and the number of relevancy and diversity metric calculations required to identify subsets of base models is also considered.

For CS-Thresh and MI-Thresh, a performance threshold, $\lambda_{\text{perf}} = 0.2$, was used to ensure that diverse base models are also relevant to the current concept. This performance threshold value has been chosen for MI-Thresh and CS-Thresh based on the results obtained by P-Thresh, presented in Figures 6.1–6.4, and is consistent with performance thresholds used in [55].

Table 6.3: Parameterised thresholding approaches: notation and descriptions.

Approach	Parameter	Description
P-Thresh	λ_{perf}	Threshold on R^2 predictive performance
MI-Thresh	λ_{perf}	Threshold on R^2 predictive performance
	λ_{MI}	Threshold on the MI between base model predictions
CS-Thresh	λ_{perf}	Threshold on R^2 predictive performance
	λ_{CS}	Threshold on the conceptual similarity between base models

6.6.1 Parameterised Culling Thresholds

Figures 6.1–6.4 show the increase in performance of the BOTL meta-learner compared to using the underlying CDD alone, with increasingly aggressive culling parameters[1] for the sudden drifting hyperplane variants, gradual drifting hyperplane variants, smart home heating simulator and following data datasets respectively. Analysing the performance of the meta-learner with varying culling parameter values highlights that the selection of such parameter values is challenging, and can be dependent on many underlying factors, including the number of base models available, noise in the data stream, separability of base models for a given diversity metric, and the underlying CDD.

For sudden and gradual drifting hyperplane data streams with uniform noise (SuddenA, GradualA), in Figures 6.1 and 6.2, the meta-learner is unlikely to overfit, regardless of the number of base models, since the concepts to be learnt are simple and there is little noise in these variants of the synthetic data streams. This means that using aggressive culling parameters reduces the overall predictive performance of the meta-learner, since the meta-learner benefits from retaining more base models without suffering from the curse of dimensionality. However, in drifting hyperplane data streams with sensor failure (SuddenB and GradualB), more aggressive culling parameters are required to prevent the meta-learner from overfitting when using ADWIN as the underlying CDD. This is necessary since the increase in noise increases the likelihood of the meta-learner overfitting, and ADWIN does not make use of historical models in the presence of recurring concepts, leading to large covariances between base models that have been learnt to represent the same concept. This indicates that aggressive culling techniques may be required in noisy data streams when using CDDs that do not reuse previously learnt models when

[1]For Figures 6.1–6.4, all plots show increasingly aggressive culling parameter values on the x-axis with the least aggressive parameter value on the left of each plot, and most aggressive on the right.

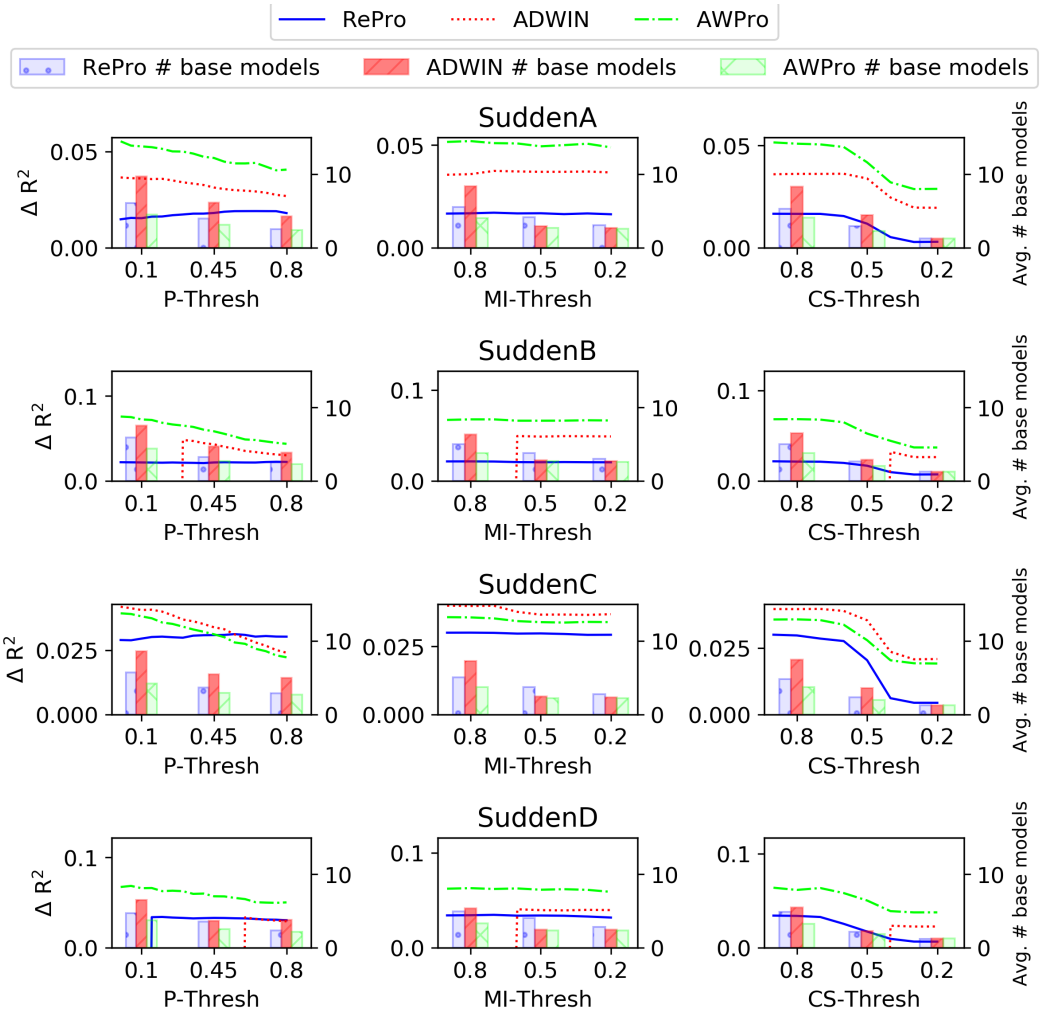


Figure 6.1: Sudden Drifting Hyperplane: increase in R^2 performance compared to using the underlying CDD alone, and number of base models used by the BOTL meta-learner using increasingly aggressive culling threshold parameter values for performance, MI, and conceptual similarity thresholding, for variants of the sudden drifting hyperplane datasets when transferring base models between 5 data streams.

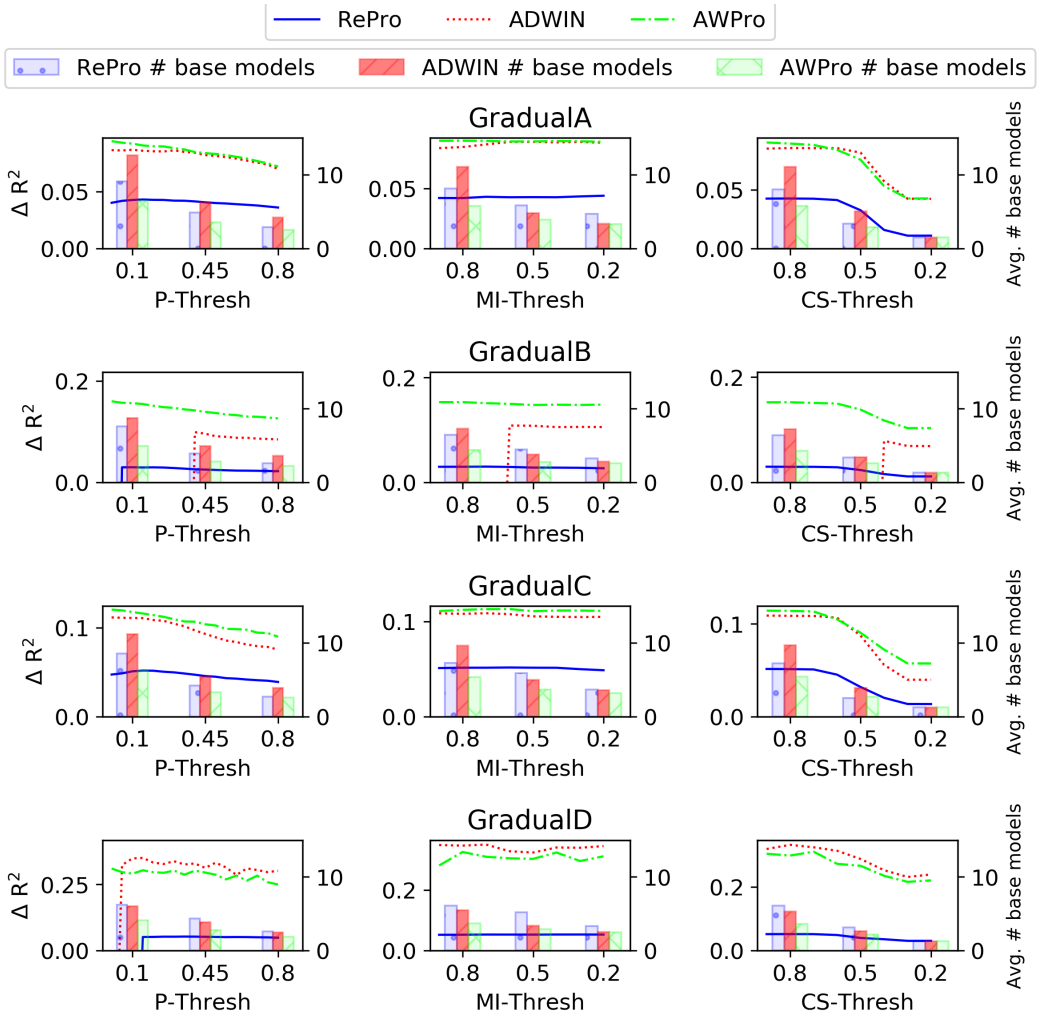


Figure 6.2: Gradual Drifting Hyperplane: increase in R^2 performance compared to using the underlying CDD alone, and number of base models used by the BOTL meta-learner using increasingly aggressive culling threshold parameter values for performance, MI, and conceptual similarity thresholding, for variants of the gradual drifting hyperplane datasets when transferring base models between 5 data streams.

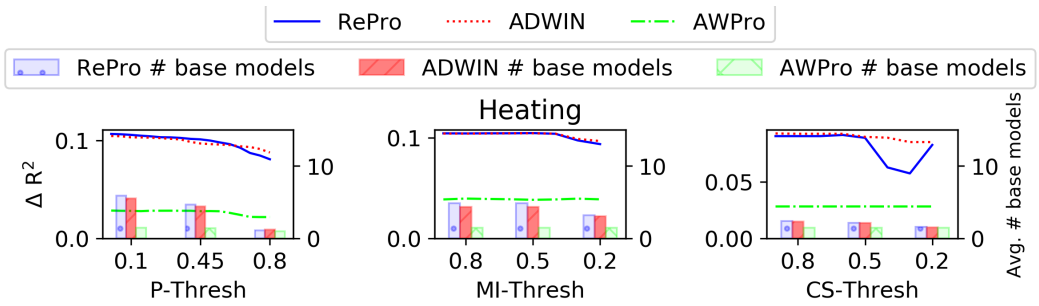


Figure 6.3: Heating Simulator: increase in R^2 performance compared to using the underlying CDD alone, and number of base models used by the BOTL meta-learner using increasingly aggressive culling threshold parameter values for performance, MI, and conceptual similarity thresholding, for the smart home heating simulator dataset when transferring base models between 5 data streams.

concepts re-occur. Conversely, CDDs that reuse base models, such as RePro and AWPro, benefit from less aggressive culling parameters in these synthetic drifting hyperplane data streams.

When learning concepts with more complex underlying distributions, the meta-learner is more likely to overfit when the number of base models is large in comparison to the window of available data. Data streams created using the smart home heating simulator are generated using real-world weather data, meaning that the concepts to be learnt are more complex, and contain more noise, in comparison to the drifting hyperplane data streams. These factors indicate that the meta-learner is likely to overfit when the number of base models becomes large, therefore requiring aggressive culling parameters. However, Figure 6.3 shows that less aggressive culling parameters can be chosen. This is observed since a large window size has been used to detect concept drifts, create base models, and train the meta-learner. This highlights the relationship between the meta-learners generalisation ability, representational capacity, and the amount of available training data.

In addition to this, a wider variety of culling parameters obtain similar predictive performances in the smart home heating simulator data streams. For example, there is little change in the predictive performance and number of base models selected for P-Thresh values ranging between 0.1–0.5, MI-Thresh values 0.8–0.4, and CS-Thresh values 0.8–0.5. A wider range of culling threshold values can be used for base model selection in the smart home heating simulator data streams because the number of base models available to the meta-learner does not change significantly over these ranges of culling parameter values. This indicates that the meta-learner's sensitivity to culling parameter values is also dependent on how separable base mod-

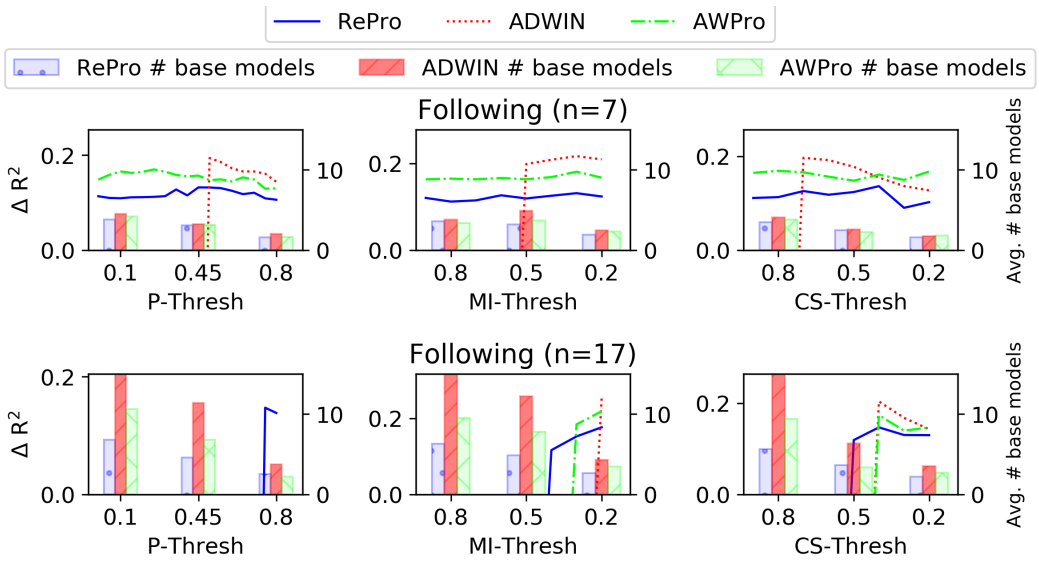


Figure 6.4: Following Distance: increase in R^2 performance compared to using the underlying CDD alone, and number of base models used by the BOTL meta-learner using increasingly aggressive culling threshold parameter values for performance, MI, and conceptual similarity thresholding, for the following distance dataset when transferring base models between 7 and 17 data streams.

els are for a given metric. For example, if large numbers of base models are equally diverse, aggressive culling parameters may be required to sufficiently reduce the number of base models to prevent overfitting.

Figure 6.4 shows the increase in predictive performance of the meta-learner compared to using the underlying CDD alone when transferring base models between 7 and 17 following distance data streams. Results for frameworks with differing numbers of data streams are presented to highlight the difficulty in selecting culling parameter values when the number of data streams in the framework is large. The meta-learners that use ADWIN as the underlying CDD are most sensitive to the culling parameter values due to the increased covariance between base models in the presence of recurring concepts. However, when base models are transferred between 17 data streams, all meta-learners become sensitive to culling parameter values, regardless of the underlying CDD used. To prevent overfitting, aggressive culling parameters are required due to two factors. First, using bi-directional knowledge transfer between 17 data streams increases the number of base models available to the meta-learner, and therefore aggressive culling parameters are required to sufficiently reduce the number of base models used as input to the meta-learner for the given window size of available data. Second, aggressive culling parameters are

required since there may be high levels of covariance between predictions of base models learnt from different data streams that encounter similar concepts. These challenges are highlighted when using P-Thresh as a culling mechanism. P-Thresh selects base models using performance alone, and when selecting base models from 17 following distance data streams the meta-learner overfits, even with aggressive culling parameters. This occurs because, after culling, the remaining base models obtain high R^2 performances, indicating similar predictions. This introduces high levels of covariance between inputs to the meta-learner. MI-Thresh and CS-Thresh also suffer from the increased number of base models and the covariance among the predictions of base models from different data streams. However, a wider range of culling parameters can be used to obtain a meta-learner with improved predictive performance over the underlying CDD. This reiterates the importance of using diversity when selecting base models.

The results presented in this section demonstrate that selecting appropriate culling threshold parameter values is challenging and may require domain expertise in order to prevent the meta-learner from overfitting. The aggressiveness of the culling threshold chosen can be dependent on the amount of training data available to the meta-learner, the number of base models available to select from, the complexity of the underlying distribution of the data stream, and the diversity between base models. Since each of these factors must be considered, selecting a culling parameter value may be difficult to determine in advance, and selecting a single threshold parameter for all online environments is not possible. However, the results presented in Figures 6.1–6.4 show that aggressive culling parameters are typically more beneficial since less aggressive culling parameters can be ineffective at reducing the number of base models sufficiently to prevent the meta-learner overfitting, as can be seen in the following distance dataset. Additionally, only a small reduction in predictive performance is observed when using an aggressive culling parameter in comparison to less aggressive culling parameters that are also able to prevent the meta-learner from overfitting, as seen in the drifting hyperplane and heating simulator datasets. Since the selection of effective culling threshold parameters is challenging, parameterless base model selection techniques are required. This can be achieved using conceptual similarity clustering, as introduced in Section 6.3.2.

The results presented in Figures 6.1–6.4 have been used to select parameter values for each of the parameterised thresholding approaches. Parameter values were selected that showed an increase in predictive performance in comparison to using the underlying CDD alone across all dataset types. Therefore, MI-Thresh uses the MI threshold $\lambda_{MI} = 0.2$, and CS-Thresh uses a conceptual similarity thresh-

old $\lambda_{CS} = 0.4$, which are used to compare parameterised thresholding approaches with parameterless clustering. P-Thresh is not used as a baseline technique in the remainder of this chapter since it is not able to effectively obtain a subset of base models to prevent overfitting in the BOTL framework with 17 following distance data streams.

6.6.2 Parameterless Clustering

Tables 6.4–6.7 show the R^2 and PMCC^2 predictive performances, the average number of base models used by the meta-learner, the maximum number of base models used by the meta-learner, and the average number of relevancy and diversity metric calculations required to compare and evaluate base models for the various datasets and CDDs considered. These results show that, without base model selection, BOTL is more likely to overfit in noisy data streams, as seen in Tables 6.4b, 6.4d, 6.5b, 6.5d, and 6.7. However, using base model selection techniques that obtain a relevant yet diverse subset of base models reduces the likelihood of overfitting.

CS-Thresh obtains improved predictive performances in comparison to using the underlying CDD alone (with statistical significance $p < 0.01$). CS-Thresh also frequently outperforms BOTL without base model selection, and obtains comparable predictive performances to using MI-Thresh. For most datasets, CS-Clust also achieves this. However, CS-Clust obtains a poor R^2 predictive performance on SuddenB data streams when using RePro and ADWIN as underlying CDDs, as shown in Table 6.4b, and for all CDDs on GradualB data streams, as shown in Table 6.5b. This indicates that CS-Clust is still susceptible to overfitting in data streams with large amounts of noise. Noise in these data streams prevents the clustering approach from effectively distinguishing between models learnt to represent different concepts. Conceptual clustering is more challenging in noisy environments because there is more variance in the underlying distribution of data used to create base models. This can affect the PCs created from the underlying distribution of data belonging to each concept, and can also increase the number of PCs required to capture 99.9% of the variance of the window of data. Capturing the increased noise within the PCs can reduce the PAs between these windows of synthetic data, making conceptual clustering more challenging due to a reduction in the separability of base models. To overcome this, fewer PCs could be used to calculate the PAs between the subspaces, thereby capturing less variance in the window of data used to create base models. Alternatively, other methods of obtaining orthonormal representations of data that are more robust to noise, such as Laplacian PCA [102] or Robust PCA [90], can be used [42].

(a) SuddenA: sudden drifting hyperplanes with uniform noise.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.883	0.884	1	1	0	0.851	0.854	1	1	0	0.830	0.835	1	1	0
BOTL	0.834	0.845	24	31	0	0.828	0.839	41	57	0	***0.884**	0.886	17	21	0
MI-Thresh	0.902	0.902	4	5	28466	0.887	0.889	3	4	51163	0.880	0.882	3	4	17031
CS-Thresh	*0.892	0.892	2	4	2494	*0.875	0.877	2	4	2945	*0.863	0.865	2	3	1996
CS-Clust	***0.904**	0.904	5	8	5700	***0.885**	0.887	6	8	5441	*0.878	0.879	5	8	1292

(b) SuddenB: sudden drifting hyperplanes with single sensor failure.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.883	0.884	1	1	0	0.830	0.836	1	1	0	0.808	0.814	1	1	0
BOTL	-2e+21	0.507	26	34	0	-7e+22	0.499	41	60	0	-3e+22	0.532	20	27	0
MI-Thresh	0.904	0.904	4	5	27301	0.879	0.880	3	5	47080	0.873	0.874	3	4	17510
CS-Thresh	***0.893**	0.894	2	3	2868	***0.860**	0.861	2	3	1905	*0.850	0.852	2	3	1984
CS-Clust	-2e+19	0.870	5	8	5322	-2e+18	0.856	6	8	5627	***0.868**	0.870	5	8	1666

(c) SuddenC: sudden drifting hyperplanes with intermittent single sensor failure.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.873	0.875	1	1	0	0.845	0.848	1	1	0	0.847	0.850	1	1	0
BOTL	0.841	0.850	25	32	0	0.813	0.827	41	57	0	***0.888**	0.889	18	21	0
MI-Thresh	0.905	0.905	3	5	27026	0.882	0.883	3	4	45049	0.881	0.882	3	4	16104
CS-Thresh	*0.881	0.881	2	3	2385	*0.867	0.868	2	3	2159	*0.867	0.869	2	3	1765
CS-Clust	***0.909**	0.909	5	7	5276	***0.885**	0.886	6	8	5457	*0.887	0.888	5	9	1369

(d) SuddenD: sudden drifting hyperplanes with gradual sensor deterioration.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.870	0.871	1	1	0	0.845	0.849	1	1	0	0.810	0.816	1	1	0
BOTL	-1e+22	0.341	27	35	0	-9e+21	0.334	41	59	0	-2e+21	0.356	18	22	0
MI-Thresh	0.900	0.901	3	5	28144	0.885	0.886	3	4	42669	0.873	0.874	3	4	14776
CS-Thresh	*0.880	0.880	2	3	2694	*0.870	0.872	2	3	2838	*0.850	0.851	2	3	2278
CS-Clust	***0.901**	0.902	5	8	5850	***0.884**	0.886	6	8	5416	***0.869**	0.870	5	9	1392

Table 6.4: Sudden Drifting Hyperplane: R^2 and PMCC^2 predictive performance, the average number of base models used by the meta-learner ($|\mathcal{M}'|$), the maximum number of base models used by the meta-learner ($\lceil\mathcal{M}'\rceil$), and the average number of relevancy and diversity metric calculations to compare and evaluate base models (M.Calcs.) for variants of the sudden drifting hyperplane datasets when transferring models between 5 data streams in BOTL. Improved predictive performances with statistical t-test values $p < 0.01$ compared to the underlying CDD, while requiring fewer relevancy and diversity metric calculations than MI-Thresh are indicated with *. Of these, bold type indicates the approach with highest performance.

(a) GradualA: gradual drifting hyperplanes with uniform noise.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.848	0.848	1	1	0	0.796	0.803	1	1	0	0.798	0.804	1	1	0
BOTL	0.776	0.800	31	37	0	0.792	0.811	39	54	0	***0.883**	0.886	20	25	0
MI-Thresh	0.892	0.892	4	6	55814	0.885	0.885	4	5	73104	0.887	0.887	4	5	30561
CS-Thresh	*0.864	0.864	2	4	3235	*0.850	0.850	3	5	3380	*0.850	0.850	3	4	2793
CS-Clust	***0.898**	0.899	5	7	12090	***0.880**	0.881	6	8	5167	*0.880	0.880	5	8	1584

(b) GradualB: gradual drifting hyperplanes with single sensor failure.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.859	0.860	1	1	0	0.757	0.771	1	1	0	0.697	0.719	1	1	0
BOTL	-6e+20	0.323	29	38	0	-1e+21	0.322	40	57	0	-1e+21	0.348	21	28	0
MI-Thresh	0.887	0.888	4	5	45279	0.862	0.863	3	5	59037	0.845	0.846	3	4	25333
CS-Thresh	***0.873**	0.873	2	4	3418	***0.835**	0.836	2	4	3652	***0.809**	0.810	2	4	2146
CS-Clust	-1e+17	0.870	6	8	12917	-5e+16	0.829	6	8	5366	-6e+11	0.832	5	7	1712

(c) GradualC: gradual drifting hyperplanes with intermittent single sensor failure.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.842	0.842	1	1	0	0.773	0.779	1	1	0	0.759	0.773	1	1	0
BOTL	0.781	0.803	29	36	0	0.776	0.799	39	54	0	***0.873**	0.875	21	27	0
MI-Thresh	0.890	0.890	4	6	52092	0.878	0.878	4	5	69921	0.870	0.871	4	5	29440
CS-Thresh	*0.862	0.862	2	4	3640	*0.832	0.832	2	4	3375	*0.830	0.830	2	4	2596
CS-Clust	***0.892**	0.893	5	8	12375	***0.865**	0.865	6	8	5059	*0.865	0.866	5	8	1666

(d) GradualD: gradual drifting hyperplanes with gradual sensor deterioration.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.846	0.847	1	1	0	0.484	0.593	1	1	0	0.519	0.629	1	1	0
BOTL	-3e+22	0.147	31	41	0	-2e+22	0.143	26	44	0	-2e+22	0.129	17	26	0
MI-Thresh	0.898	0.898	4	6	45921	0.812	0.813	3	5	33128	0.821	0.821	3	4	19825
CS-Thresh	*0.878	0.879	2	4	3592	*0.759	0.761	2	3	3021	*0.758	0.759	2	3	2525
CS-Clust	***0.898**	0.898	6	7	15333	***0.789**	0.790	5	10	2698	***0.787**	0.789	5	8	1310

Table 6.5: Gradual Drifting Hyperplane: R^2 and PMCC^2 predictive performance, the average number of base models used by the meta-learner ($|\mathcal{M}'|$), the maximum number of base models used by the meta-learner ($\lceil\mathcal{M}'\rceil$), and the average number of relevancy and diversity metric calculations to compare and evaluate base models (M.Calcs.) for variants of the gradual drifting hyperplane datasets when transferring models between 5 data streams in BOTL. Improved predictive performances with statistical t-test values $p < 0.01$ compared to the underlying CDD, while requiring fewer relevancy and diversity metric calculations than MI-Thresh are indicated with *. Of these, bold type indicates the approach with highest performance.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.636	0.652	1	1	0	0.635	0.655	1	1	0	0.625	0.645	1	1	0
BOTL	***0.728**	0.735	10	15	0	***0.717**	0.725	9	15	0	***0.727**	0.734	10	15	0
MI-Thresh	0.706	0.713	3	5	3441	0.705	0.713	3	5	4113	0.708	0.716	3	4	4034
CS-Thresh	*0.705	0.712	3	4	508	*0.700	0.708	3	4	763	*0.696	0.703	3	4	676
CS-Clust	*0.708	0.715	3	5	968	*0.707	0.715	4	5	1072	*0.707	0.715	3	5	1375

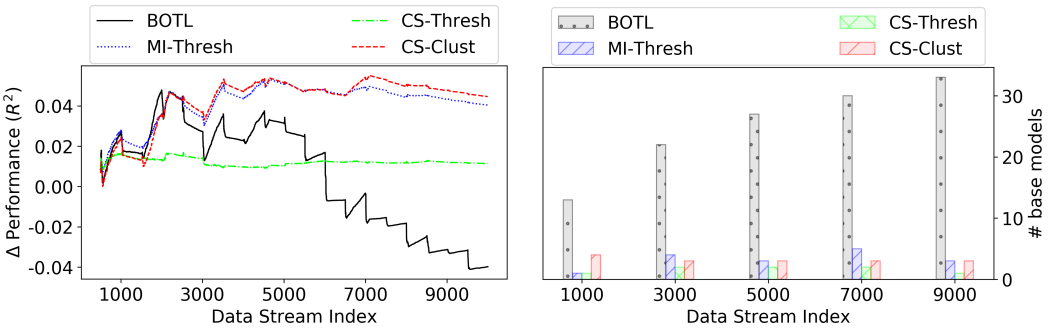
Table 6.6: Heating Simulator: R^2 and PMCC^2 predictive performance, the average ($|\mathcal{M}'|$) and maximum ($\lceil\mathcal{M}'\rceil$) number of base models used by the meta-learner, and the average number of relevancy and diversity metric calculations to compare and evaluate base models (M.Calcs.) for the smart home heating simulator dataset when transferring models between 5 data streams. Improved predictive performances with statistical t-test values $p < 0.01$ compared to the underlying CDD, while requiring fewer relevancy and diversity metric calculations than MI-Thresh are indicated with *. Of these, bold type indicates the approach with highest performance.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.546	0.565	1	1	0	0.441	0.502	1	1	0	0.396	0.515	1	1	0
BOTL	*0.646	0.678	6	12	0	-2e+15	0.384	12	25	0	-9e+9	0.665	8	18	0
MI-Thresh	0.665	0.688	2	3	1116	0.663	0.682	3	4	2096	0.693	0.705	2	4	1287
CS-Thresh	*0.652	0.678	3	5	263	*0.637	0.661	3	5	312	*0.660	0.677	2	4	279
CS-Clust	***0.661**	0.686	3	5	779	***0.673**	0.695	4	8	1143	***0.705**	0.716	3	7	730

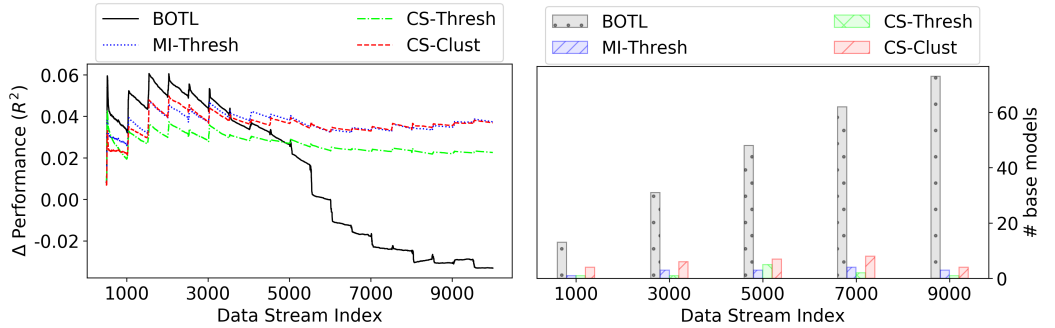
Table 6.7: Following Distance: R^2 and PMCC^2 predictive performance, the average ($|\mathcal{M}'|$) and maximum ($\lceil\mathcal{M}'\rceil$) number of base models used by the meta-learner, and the average number of relevancy and diversity metric calculations to compare and evaluate base models (M.Calcs.) for the following distance dataset when transferring models between 7 data streams in BOTL. The R^2 and PMCC^2 predictive performances and number of base model for other numbers of data streams is shown in Figure 6.6. Improved predictive performances with statistical t-test values $p < 0.01$ compared to the underlying CDD, while requiring fewer relevancy and diversity metric calculations than MI-Thresh are indicated with *. Of these, bold type indicates the approach with highest performance.

Although CS-Clust overfits in these synthetic data streams, the $PMCC^2$ performances are statistically significantly ($p < 0.01$) greater than BOTL with no base model selection, and outperform the $PMCC^2$ performance obtained when using the underlying CDDs alone, except for when RePro is used as the underlying CDD in the GradualD dataset. As discussed in Chapter 5, this is observed since $PMCC^2$ is bound between 0 and 1, whereas R^2 is unbounded, $(-\infty, 1]$. This means that the R^2 predictive performance can be highly skewed if the meta-learner overfits for a small number of instances in the data stream. The observed improvement in $PMCC^2$ performance indicates that although CS-Clust can be susceptible to overfitting in these environments, the meta-learner overfits less frequently in comparison to the BOTL meta-learner that uses all base models. With the exception of these noisy synthetic data streams, CS-Clust obtains R^2 performances statistically significantly ($p < 0.01$) greater than the underlying CDDs, and outperforms the BOTL meta-learner with no base model selection techniques for the majority of CDDs, across all datasets except the heating simulator dataset, as shown in Table 6.6. Table 6.6 shows that the BOTL framework with no base model selection strategy continues to achieve the highest predictive performance in the smart home heating simulator data streams due to the use of a large window size, showing that base model selection strategies may not always be necessary.

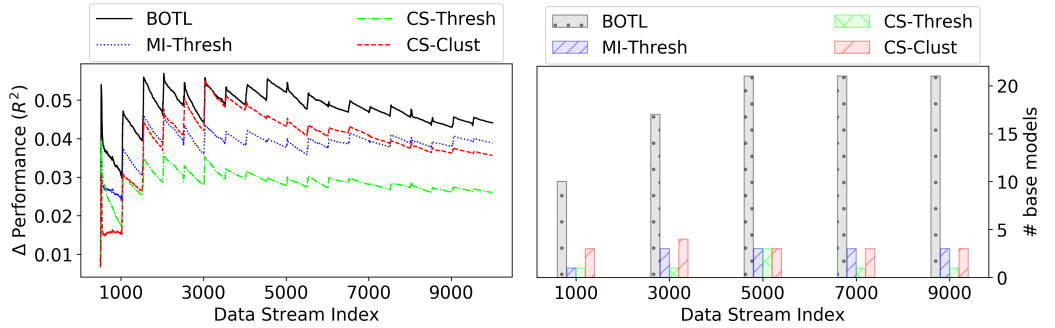
Figure 6.5 shows the difference in predictive performance between the OLS meta-learner in BOTL, MI-Thresh, CS-Thresh and CS-Clust in comparison to the underlying CDDs as a sudden drifting hyperplane dataset (SuddenA) progresses. Figure 6.5 shows that CS-Thresh provides the smallest increase in performance over the underlying CDD for this sudden drifting hyperplane data stream, indicating that using a parameterised conceptual similarity threshold is less effective than using a parameterised MI threshold or using conceptual clustering. However, the improvement in predictive performance obtained when using CS-Thresh does not reduce as the data stream progresses and a larger number of base models are made available to the meta-learner. This means that although CS-Thresh may not select the best subset of base models in order to obtain a meta-learner with the highest predictive performance, CS-Thresh is effective at reducing the likelihood of the OLS meta-learner overfitting, particularly when RePro and ADWIN are used as the underlying CDD, as shown in Figures 6.5a and 6.5b. Figure 6.5 also shows that CS-Clust is able to maintain comparable predictive performances to MI-Thresh as the sudden drifting hyperplane progresses without requiring a user defined parameterised threshold. These results indicate that CS-Clust is and MI-Thresh are similarly effective at selecting a subset of base models which improve the predictive



(a) RePro



(b) ADWIN



(c) AWPro

Figure 6.5: BOTL, MI-Thresh, CS-Thresh and CS-Clust vs. CDDs: The difference in R^2 performance (left) between the OLS meta-learner in BOTL, MI-Thresh, CS-Thresh and CS-Clust vs. the underlying CDDs of (a) RePro, (b) ADWIN, and (c) AWPro, and the number of models used as base models (right) for a sudden drifting hyperplane data stream (SuddenA). Base models are learnt locally and transferred from 4 other sudden drifting hyperplane data streams.

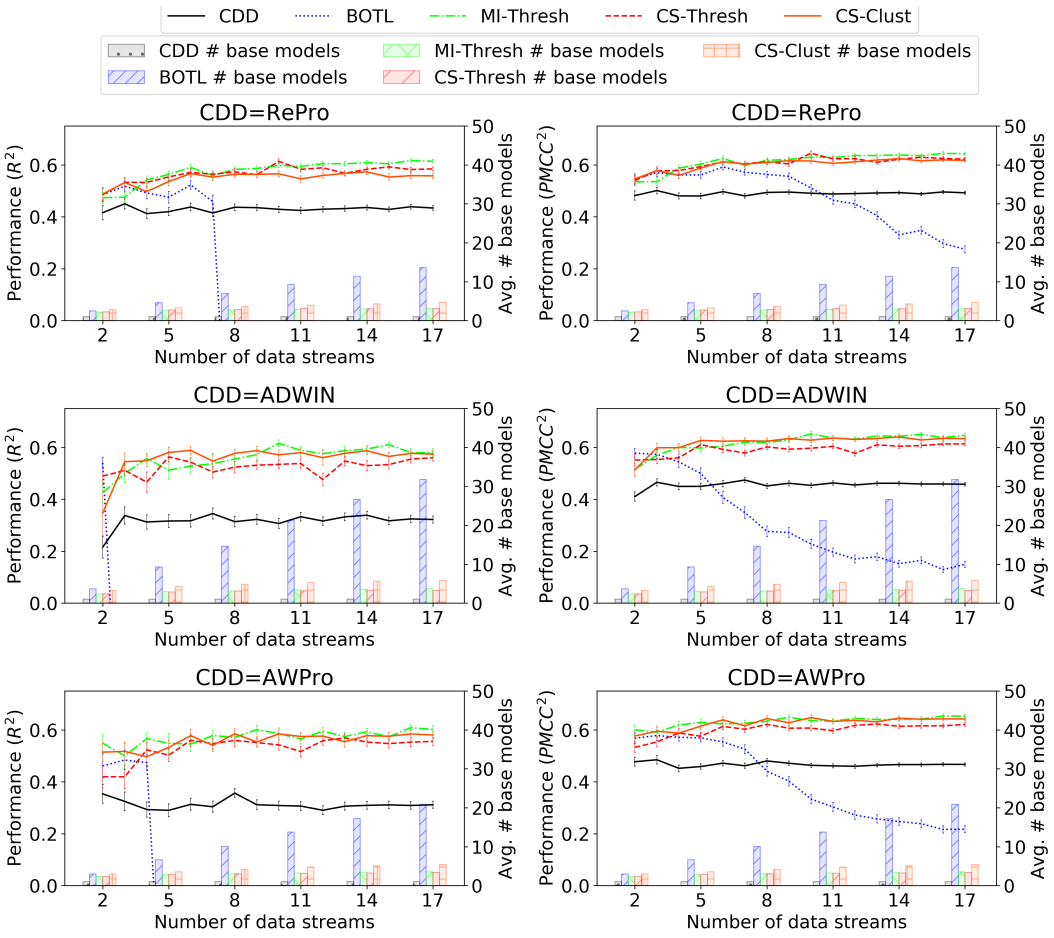


Figure 6.6: R^2 and PMCC^2 predictive performance, and number of base models used by BOTL meta-learners for increasing numbers of following distance data streams.

performance of the OLS meta-learner, while reducing the likelihood of overfitting when the number of base models available to the meta-learner continues to grow.

Figure 6.6 shows the predictive performance and number of base models used by the meta-learner with increasing numbers of following distance data streams. CS-Thresh prevents the meta-learner overfitting in these real-world data streams, obtaining similar predictive performances to MI-Thresh, while CS-Clust achieves this without requiring a user defined threshold parameter. Although CS-Clust requires additional computation for clustering, the use of conceptual similarity as a measure of diversity significantly reduces the number of pairwise comparisons between base models. Unlike MI-Thresh, the diversity metric remains static, independent of concept drifts, and therefore does not need to be recalculated as the data stream progresses. This is shown in Figure 6.7, which highlights the number of relevancy

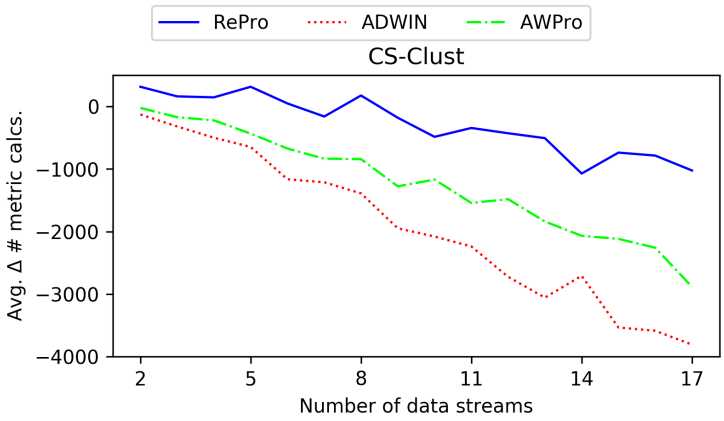


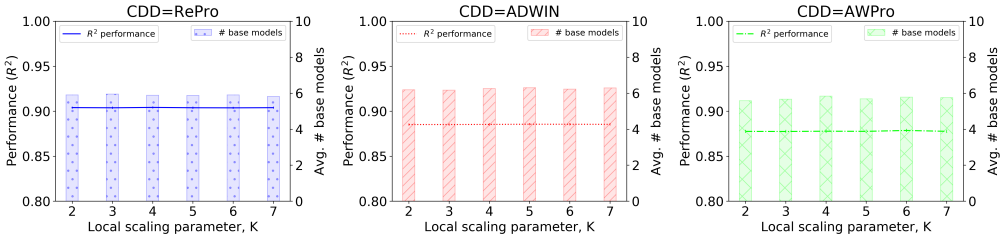
Figure 6.7: Change in number of relevancy and diversity metric calculations required to compare and evaluate base models for CS-Clust vs. MI-Thresh for increasing numbers of following distance data streams.

and diversity metric calculations required to compare and evaluate base models in CS-Clust compared to MI-Thresh, as the number of following distance data streams increases. A significant reduction in relevancy and diversity metric calculations is observed across all datasets, which can be seen in Tables 6.4–6.7.

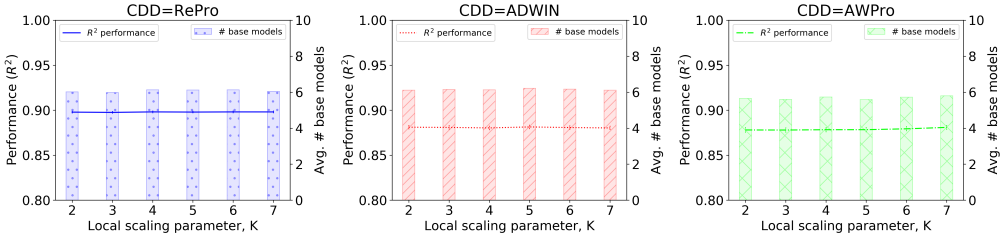
6.6.3 Local Scaling in STSC

Parameterless conceptual clustering uses STSC [100] to create clusters of conceptually similar base models. Although the number of clusters of base models is determined by STSC, in order to perform SC the affinity matrix used by STSC undergoes local scaling, as shown in Section 6.2, using Definition 3. This allows better affinities to be obtained when the density of conceptually similar base models varies [100]. A local scaling parameter of $k = 7$ has been used to generate the results presented in this chapter, as suggested by Zelnik-Manor and Perona [100]. Although parameter tuning is not typically needed for local scaling to perform well [42], the predictive performance of CS-Clust, and the number of base models used by the OLS meta-learner, have been considered for local scaling parameters varying between $k = 2$ and $k = 7$.

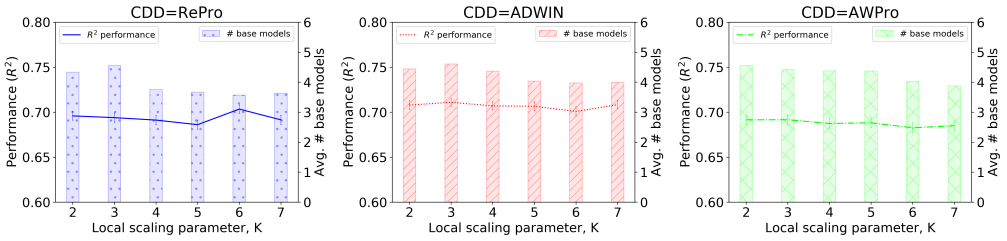
Figure 6.8 presents the predictive performance of CS-Clust and the number of base models selected as input to the OLS meta-learner using the CDDs and datasets presented in this thesis. Figure 6.8 shows that varying the local scaling parameter, k, has minimal effect on the predictive performance of CS-Clust. Additionally, the number of base models selected as input to the OLS meta-learner does not change significantly. Since CS-Clust selects a single model from each cluster, the number of



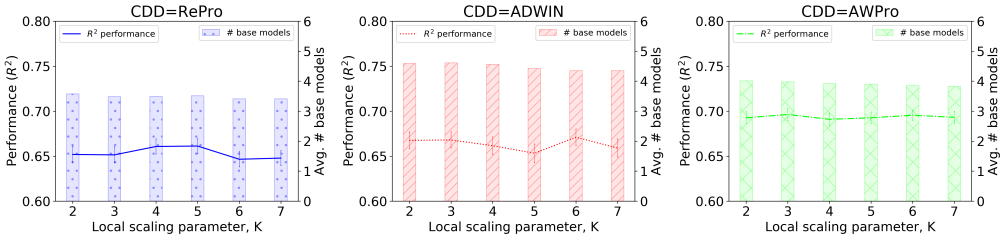
(a) Sudden drifting hyperplane



(b) Gradual drifting hyperplane



(c) Heating simulator



(d) Following distance

Figure 6.8: R^2 and number of base models used by CS-Clust meta-learners for varying local scaling parameters in the SuddenA drifting hyperplane, the GradualA drifting hyperplane, the heating simulator, and the following distance data streams, using RePro, ADWIN and AWPro as underlying CDDs.

base models used by the meta-learner is a good indicator of the number of clusters of conceptually similar base models identified by STSC.

These results support the suggestions in [100] and [42] that $k = 7$ is a reasonable parameter value for local scaling in STSC. However, accounting for locally dense areas in the affinity matrix can be challenging when clustering base models that have been learnt from online environments, particularly if CDDs such as ADWIN are used, where base models are not reused in the presence of recurring concepts. Therefore, future work may consider alternative methods, such as density-aware kernels [42, 101] to overcome the challenges of clustering using affinity matrices with locally dense areas.

6.7 Summary

CDDs can be used to make predictions in data streams that are subject to concept drift, where data availability may be limited. Meta-learners can improve predictive capabilities through the use of historically learnt knowledge, or transferred knowledge when online TL is used. However, as the number of available base models becomes large in comparison to the available data, a relevant yet diverse subset of base models must be selected to prevent the meta-learner overfitting and to improve generalisation.

In this chapter, a novel approach of estimating the similarity between base models has been presented, namely conceptual similarity. Unlike existing diversity metrics, conceptual similarity provides a measure of diversity that is independent of the current distribution of data observed in an online data stream. This is beneficial in online learning environments because it remains static, despite the presence of concept drift, and therefore does not need to be repeatedly recalculated as the data stream progresses.

Since conceptual similarity is model agnostic, it can be used to determine the diversity between any regression-based predictive models. Identifying diversity between models is not only beneficial to online TL, but is also a challenge that is encountered by ensemble pruning strategies in both offline and online environments. In offline environments, a diverse subset of base models should be identified to improve generalisation abilities [106], whereas in online environments a subset of highly diverse base models is beneficial during, and immediately after, periods of concept drift, while a low diversity subset can be used between drifts to improve generalisation [58, 60]. The conceptual similarity metric presented in this chapter can be used in each of these scenarios to calculate the diversity between models.

Two methods of selecting subsets of base models for meta-learners and ensembles have also been presented in this chapter, namely parameterised thresholding and parameterless clustering. Each of these approaches use conceptual similarity to identify the diversity among base models, and use predictive performance as an indicator of relevancy. Comparable predictive performances to performance and MI thresholding have been obtained, while requiring fewer base model comparisons. Although parameterless conceptual clustering requires additional computation due to reforming clusters when new base models are made available, the reduction in base model comparisons, and the avoidance of user defined parameters, may make it more applicable to real-world applications in which the number of base models and frequency of drifts are unknown and could become large.

6.7.1 Other Uses of Diversity in Online TL Frameworks

Online TL frameworks are situated in distributed online environments. In many real-world applications, it may not always be possible, or feasible, to transfer all knowledge between all domains, particularly if the performance benefit of sharing knowledge is minimal and the communication or computational cost of knowledge transfer is high. This chapter has shown that a subset of relevant yet diverse base models should be used as input to meta-learners in online TL frameworks. Therefore, if a model is transferred that is not relevant to the receiving domain, or is not diverse with respect to the models already available to that domain, then the transfer of the model provides no beneficial or additional knowledge to the receiving domain. The transfer of such models incurs unnecessary communication and computational overheads, which is detrimental in real-world applications of online TL where knowledge may be transferred between a network of distributed domains. Therefore, if the diversity of a model with respect to other models available in the framework could be determined prior to transfer, then the number of base models transferred throughout the framework could be reduced. The question of whether a base model should be transferred in an online TL framework is considered in Chapter 7, using diversity metrics to determine inter-domain model similarities.

Chapter 7

Deciding Whether to Transfer

Existing TL research considers three important research questions, namely, what to transfer, how to transfer, and whether to transfer [65]. The questions of what to transfer and how transferred knowledge can be used have been considered throughout Chapters 4–6, and are considered extensively in existing online TL research [27, 32, 55, 104]. However, the question of whether to transfer knowledge has been considered less frequently [65, 85, 107]. In order to address this question, online TL frameworks can be considered as a network of distributed domains, where knowledge transfer incurs a communication overhead in order to share knowledge learnt by each online domain. Therefore, the consideration of whether knowledge transfer will be beneficial and, by extension, whether it may hinder the predictive performance in a receiving domain is necessary [55].

In this thesis, online TL is conducted between multiple online domains which are distributed, as might be expected in many real-world applications. This means that in order to transfer knowledge between domains, the predictive model and PCs used to represent a concept encountered in one domain must be sent to a receiving domain via a network using a communication protocol. Therefore, knowledge transfer incurs a communication overhead in order to make a predictive model available to the receiving domain. However, for many real-world applications, it may not always be possible, or feasible, to transfer all knowledge between all domains, particularly if the performance benefit of sharing knowledge is minimal and the communication or computational cost of knowledge transfer is high. Despite this, the question of whether to transfer is often overlooked in online domains [104], even though communication and computational limitations must be considered to ensure the feasibility of knowledge transfer in real-world applications [56, 63].

In this chapter, diversity metrics are used to determine whether knowledge

should be transferred in an online TL framework, with focus on BOTL, presented in Chapter 5 [54]. The use of diversity metrics enables the similarities between models to be considered prior to transfer. Specifically, the contributions of this chapter are as follows.

- A naïve approach to limiting model transfer is proposed, using Inter-domain Diversity Thresholding (IdDT) to determine whether to transfer.
- A novel technique for determining whether to transfer is presented, using Inter-domain Conceptual Similarity (IdCS).
- These approaches are evaluated in the BOTL framework, showing empirical results for both the synthetic and real-world datasets presented in Chapter 3 [54].

The proposed naïve approach to limiting model transfer, IdDT, uses diversity thresholding to determine whether to transfer a model, and can be used in combination with any regression based diversity metric [23]. In this chapter, the Mutual Information (MI) between model predictions is used to make this decision [23, 55]. Since there is limited research into whether knowledge should be transferred in online TL frameworks, IdDT is used as a baseline approach to evaluate the effectiveness of IdCS. IdCS determines whether to transfer models using conceptual similarity, where the PAs between the underlying subspaces in which each model was learnt are calculated and used as a diversity metric [56]. This chapter shows that the predictive performances obtained by these approaches are comparable to those achieved when all models are transferred, while achieving a reduction in communication and computation overheads. Averaged across the datasets presented in Chapter 3, and CDDs presented in Chapter 4, IdDT reduces the number of models received by each domain by 47.1%, while IdCS achieves a 30.0% reduction.

The remainder of this chapter is organised as follows. Section 7.1 briefly reviews how predictive models are created from online data streams, and how they can be combined in online TL frameworks, and then considers how to determine whether they should be transferred. Two approaches of determining whether to transfer, namely IdDT and IdCS, are introduced in Section 7.2, and Section 7.3 outlines the experimental set-up for evaluating these approaches. This includes a brief discussion on the baseline techniques used in this chapter, and identifies some influential characteristics of the datasets and CDDs presented in Chapters 3 and 4 which may impact the decision of whether to transfer models. Finally, Section 7.4 presents experimental results, and Section 7.5 concludes this chapter.

7.1 Creating, Combining and Transferring Models

Although the decision of whether to transfer is rarely considered in online TL, it is heavily dependent on what knowledge can be learnt in a source domain, what knowledge is available in a receiving domain, and how knowledge is combined in the receiving domain. Following from previous chapters, this chapter considers the online TL setting where both source and receiving domains are in online environments [54, 55, 56], knowledge is transferred in the form of predictive models, and combined using a meta-learner.

7.1.1 Creating and Combining Models

Chapter 2 outlined existing approaches to creating predictive models from online data streams, and how online TL can be used to improve the predictive performance in a receiving domain. Chapter 4 highlights how predictive models can be learnt such that they are created to represent the underlying concepts encountered in a data stream using CDDs such as RePro [95], ADWIN [8] and AWPro [55]. Chapter 5 considers how knowledge, in the form of predictive models, can be used to improve the predictive performance achieved in online data streams using BOTL [54]. However, in Chapter 6 it has been shown that meta-learners can become prone to overfitting when the number of models available becomes large in comparison to the window of available data [14, 25, 55, 64, 81]. To prevent this, a relevant yet diverse subset of base models are selected to be used as input to the meta-learner [11, 22, 79].

When meta-learners are used to combine transferred and locally learnt models in online TL, the approaches used to select subsets of base models to prevent overfitting indicate which of the available models are most influential. Therefore, metrics used for base model selection, such as relevancy and diversity [11, 31, 56], can also be used to determine whether a model selected for transfer is likely to be beneficial in a receiving domain prior to transfer.

7.1.2 Whether to Transfer Models

The main purpose of considering whether to transfer in existing TL research is to prevent negative transfer [65, 70, 83, 85, 89, 107]. Reducing the impact of negative transfer is often embedded in online TL frameworks since weighting mechanisms are used to allow online TL frameworks to adapt to concept drift in the receiving domain. This means that the weighting mechanism inherently reduces the impact of negative transfer in the receiving domain [27, 32, 88, 97, 104]. However, although the impact of negative transfer is reduced, the transfer of a model that is of no, or

Table 7.1: Notation for determining whether to transfer.

	Definition
α	Domain α, where $\mathcal{A}$ is used to denote a single source domain, and $\mathcal{B}$ denotes a receiving domain
X^α	Data stream in domain α, where $X^\alpha = \{x_1, \ldots, x_t, \ldots, x_n\}$
$x_t \in X^\alpha$	The t^{th} observed instance in X^α
Y^α	Response variable space in domain α
$y_t \in Y^\alpha$	The t^{th} response variable in Y^α
C^α	The set of concepts encountered in domain α
$c_i^\alpha \in C^\alpha$	The i^{th} concept encountered in domain α
$X_i^\alpha \in X^\alpha$	The data stream segment corresponding to concept c_i^α in domain α
W	Sliding window of $\lvert W \rvert$ instances, $W = \{x_{t-\lvert W \rvert}, \ldots, x_t\}$
$f_i^\alpha : X_i^\alpha \rightarrow Y_i^\alpha$	Model i learnt in domain α
$\mathcal{M}$	Set of stable, locally learnt and transferred models
$\mathcal{M}'$	A subset of stable, locally learnt and transferred models, $\mathcal{M}' \subseteq \mathcal{M}$
Υ_{sim}	A subset of similar models in $\mathcal{M}$
$F^{\mathcal{M}} : X^\alpha \rightarrow Y^\alpha$	Meta-learner in domain α using models in $\mathcal{M}$
x_t^*	Meta instance of base model predictions for instance x_t
$\hat{y}_t^*$	Prediction using $F^{\mathcal{M}'}(x_t)$ where $\mathcal{M}' \subseteq \mathcal{M}$

little, benefit incurs unnecessary communication and computation overheads. This can limit the applicability of online TL frameworks in real-world online environments where communication and computational resources are constrained [51, 63]. Therefore, the question of whether to transfer should not only consider reducing the impact of negative transfer after a model has been received by a domain, but also how to prevent models from being transferred if they are unlikely to provide additional beneficial knowledge to a receiving domain.

7.2 Determining Whether to Transfer Models

Since a relevant yet diverse subset of base models should be used as input to a meta-learner [30], the same properties can be considered to determine whether to transfer. However, the relevancy of a model with respect to the receiving domain cannot be determined prior to transfer, since domains are distributed and online. Therefore, the current concept observed in each domain is unknown to other domains, and future concepts within each domain are also unknown [95]. Due to this, the relevancy of a model with respect to a receiving domain, typically calculated using metrics such as predictive performance, can only be determined after the model has been

Algorithm 11: Whether to Transfer Models.

Input: $\mathcal{M}^{\mathcal{A}}$, $f_i^{\mathcal{A}}$, {transferredModels}, diversity metric

Using a diversity metric, find $\Upsilon_{\text{sim}} \in \mathcal{M}^a$
sent = {}
delayed = {}
for $\mathcal{B} \in$ *receivers* **do**
    Transfer list of model IDs in Υ_{sim}
    if $\forall j \in \Upsilon_{sim} : j \notin \mathcal{M}^{\mathcal{B}}$ **then**
        Transfer $f_i^{\mathcal{A}}$ to domain $\mathcal{B}$
        Add $\mathcal{B}$ to sent list
    else
        Delay transfer of $f_i^{\mathcal{A}}$ to domain $\mathcal{B}$
        Add $\mathcal{B}$ to delayed list
Continue with BOTL
while *no drift detected* **do**
    Store predictions of all models in Υ_{sim}
    if *drift detected* **then**
        Use Algorithm 12 for model replacement

transferred. Instead, the diversity of base models can be used to determine whether to transfer using the inter-domain similarity of base models learnt locally and received from other domains. Using the notation in Table 7.1, Inter-domain Diversity Thresholding (IdDT), and Inter-domain Conceptual Similarity (IdCS) are proposed as methods of determining whether to transfer models in online TL frameworks.

Both of these techniques use a similar methodology to determine whether to transfer, consisting of two stages. First, a source domain must determine whether the model selected for transfer will be beneficial to receiving domains, as shown in Algorithm 11. Second, once the decision of whether to transfer has been made, the source domain can later reconsider whether the model previously selected for transfer is the most relevant with respect to the concept it was learnt to represent. If an alternative model is considered to be more relevant, then the locally learnt model can be replaced, as shown in Algorithm 12.

7.2.1 Inter-domain Diversity Thresholding (IdDT)

IdDT uses simple thresholding techniques to determine whether to transfer. Given a model $f_i^{\mathcal{A}}$ learnt in a source domain $\mathcal{A}$ to represent concept $c_i^{\mathcal{A}}$, which has been selected for transfer, regression based diversity metrics can be used to determine the pairwise similarity of model $f_i^{\mathcal{A}}$ and other models available in the source domain.

Common regression diversity metrics measure the level of disagreement be-

Algorithm 12: Whether to Replace Previously Transferred Models.

Input: Model set $\mathcal{M}^{\mathcal{A}}$, current local model ID $\langle \mathcal{A}; i \rangle$, model $f_i^{\mathcal{A}}$, receivers, sent, delayed, Υ_{sim}

1 $\langle \alpha; j \rangle$ = getIDbestPerformingModel(Υ_{sim});
2 **if** $\langle \alpha; j \rangle == \langle \mathcal{A}; i \rangle$ **then**
3 **for** $\mathcal{B} \in$ *delayed* **do**
4 Transfer $f_i^{\mathcal{A}}$ to domain $\mathcal{B}$;
5 **else**
6 **for** $\mathcal{B} \in$ *sent* **do**
7 Discard $f_i^{\mathcal{A}}$ from $\mathcal{M}^{\mathcal{B}}$ in domain $\mathcal{B}$;
8 **for** $\mathcal{B} \in$ *receivers* **do**
9 **if** $\langle \alpha; j \rangle \notin \mathcal{M}^{\mathcal{B}}$ **then**
10 Transfer f_j^{α} to domain $\mathcal{B}$;
11 Replace model $\langle \mathcal{A}; i \rangle$ with $\langle \alpha; j \rangle$ in $\mathcal{M}^{\mathcal{A}}$

tween base model predictions [23], meaning that the predictions of model $f_i^{\mathcal{A}}$ are compared to other model predictions over a window of recent observations in the source domain. IdDT compares model $f_i^{\mathcal{A}}$ to other models available in domain $\mathcal{A}$, $f_j^{\alpha} \in \mathcal{M}^{\mathcal{A}}$, where $\mathcal{M}^{\mathcal{A}}$ is the set of models learnt locally, and received from other domains, and α denotes the domain in which model f_j^{α} was originally learnt. In this chapter, IdDT uses MI as a diversity metric, where the pairwise MI between $f_i^{\mathcal{A}}$ and f_j^{α} is denoted as $\text{MI}(f_i^{\mathcal{A}}, f_j^{\alpha})$.

Once the pairwise diversity between models has been calculated, IdDT uses a diversity threshold, λ_{MI}, to identify similar models. Thus, the pairwise MI between the current locally learnt model, $f_i^{\mathcal{A}}$, and all other models available in the source domain, $f_j^{\alpha} \in \mathcal{M}^{\mathcal{A}}$, is calculated, and similar models are identified when $\text{MI}(f_i^{\mathcal{A}}, f_j^{\alpha}) > \lambda_{\text{MI}}$. Using this approach a subset of similar models, Υ_{sim}, is created. Therefore, in Algorithm 11 (line 1), Υ_{sim} is created using,

$$\Upsilon_{\text{sim}} = \exists f_j^{\alpha} \in \mathcal{M}^{\mathcal{A}} : \text{MI}(f_i^{\mathcal{A}}, f_j^{\alpha}) > \lambda_{MI}. \tag{7.1}$$

Since the models in Υ_{sim} are similar to model $f_i^{\mathcal{A}}$, if a model in Υ_{sim} is available in a receiving domain $\mathcal{B}$, then it can be assumed that $f_i^{\mathcal{A}}$ will provide little beneficial knowledge to the meta-learner in $\mathcal{B}$, and therefore does not need to be transferred.

When domains in online TL frameworks are distributed and independent there is no global overview of which models are available in each domain. Therefore, a list of model identifiers corresponding to Υ_{sim} must initially be transferred to the receiving domain $\mathcal{B}$, where the identifier $\langle \alpha; j \rangle$ is used to identify model f_j^{α} (Algorithm 11: line 5). Although the transfer of model identifiers incurs a commu-

nication overhead, this overhead is likely to be small in comparison to transferring all predictive models via parameter transfer. This reduction in communication overhead becomes more significant when complex models are used as base models, such as Neural Networks, where large numbers of model parameters must be transferred between domains. Once the list of model identifiers in Υ_{sim} has been transferred, the receiving domain can check whether a model listed in Υ_{sim} is available in its model set, $\mathcal{M}^{\mathcal{B}}$. If none of the models listed in Υ_{sim} are available in the receiving domain, the receiver can notify the source domain that model $f_i^{\mathcal{A}}$ is to be transferred (Algorithm 11: lines 6–8). This allows the source domain to keep track of which domains in the framework have received model $f_i^{\mathcal{A}}$.

Once the decision of whether to transfer a model has been made in the source domain, IdDT can then determine whether a similar model in Υ_{sim} is more relevant than $f_i^{\mathcal{A}}$ with respect to the concept $c_i^{\mathcal{A}}$, which model $f_i^{\mathcal{A}}$ was learnt to represent. To achieve this, the predictive performances of each model in Υ_{sim} are monitored until a concept drift is detected (Algorithm 11: lines 12–15). Once a drift is detected, Algorithm 12 can be used, where the predictive performance of each model in Υ_{sim} is compared to the predictive performance of $f_i^{\mathcal{A}}$ for all instances observed during concept $c_i^{\mathcal{A}}$. If $f_i^{\mathcal{A}}$ achieves the highest predictive performance throughout the duration of the concept, it is considered to be the most relevant model available in the source domain for concept $c_i^{\mathcal{A}}$. Since model $f_i^{\mathcal{A}}$ is considered to be the most relevant, it is also considered to be the most beneficial to receiving domains. As such, model $f_i^{\mathcal{A}}$ is transferred to the domains in the framework that did not initially receive it (Algorithm 12: lines 2–4).

Alternatively, if another model in Υ_{sim}, for example f_j^{α}, where α represents any domain in the framework, including domain $\mathcal{A}$, is found to have obtained a better predictive performance over the concept $c_i^{\mathcal{A}}$, then model $f_i^{\mathcal{A}}$ can be discarded in the source domain, and replaced by model f_j^{α} (Algorithm 12: lines 5–11). In this scenario, model f_j^{α} can be considered to be more relevant to the concept $c_i^{\mathcal{A}}$ than the locally learnt model $f_i^{\mathcal{A}}$, and therefore retaining model $f_i^{\mathcal{A}}$ unnecessarily increases the computation required to select base models as input to the meta-learner. In addition to this, model $f_i^{\mathcal{A}}$ can be discarded by any domain that previously received it (Algorithm 12: lines 6–7), and model f_j^{α} can be transferred if it is not already available in the receiving domains (Algorithm 12: lines 8–10).

Replacing models in this way allows the number of base models available throughout the framework to be reduced, thereby reducing computational overheads within each independent domain. Replacing models is particularly beneficial in frameworks where domains may have differing computational or data availability.

For example, some domains in the framework may have greater data availability or computational resources, allowing base models to be learnt that may have greater representational capacities or generalisation abilities. Model replacement enables base models that have been learnt in domains with more restricted resources to be replaced when a similar model learnt from a less restricted domain is transferred throughout the framework.

Although model replacement is beneficial, if the locally learnt model, $f_i^{\mathcal{A}}$, is the best performing model, then the other models in Υ_{sim} cannot be replaced by $f_i^{\mathcal{A}}$. This is because it cannot be determined whether model $f_i^{\mathcal{A}}$ will achieve an improved predictive performance over the concepts for which models in Υ_{sim} were originally learnt to represent.

IdDT naïvely determines whether to transfer using MI as a diversity metric and requires a user defined diversity threshold. Using this approach can be considered naïve since it makes the assumption that the MI between models is consistent throughout the duration of each concept and across all domains in the framework. However, MI is dependent on the underlying distribution of data used to obtain model predictions. Therefore, there is no guarantee that models considered to be similar in the source domain will continue to be considered similar in the source domain as the data stream progresses due to concept drift, or considered similar in receiving domains due to being independent of one another. To overcome this, a static diversity metric is preferable, which remains consistent across domains and is not affected by concept drift.

7.2.2 Inter-domain Conceptual Similarity (IdCS)

In common with IdDT, IdCS also uses Algorithms 11 and 12 to determine whether to transfer a model, and whether to replace locally learnt models. However, IdCS uses the conceptual similarity between base models as a diversity metric. Conceptual similarity estimates the similarity between the concepts that each base model was learnt to represent, and is calculated by considering the similarity between the underlying subspaces in which base models were learnt. As discussed in Chapter 6, this diversity metric is independent of the current distribution of observable data, and therefore remains static between domains even in the presence of concept drift [56].

The conceptual similarity between two base models has been defined in Section 6.2 [56], and is estimated using the PAs between orthonormal representations of the subspaces used to train each model [10, 29, 45]. This estimates the conceptual distance between two base models, which then undergoes local scaling using the distance to the k-th nearest neighbour of each base model to adjust for locally dense

areas of similar models. The value $k = 7$ is adopted for the local scaling parameter as used in Chapter 6 and existing research [42, 56, 100].

The conceptual similarity of base models can be used as a diversity metric in IdDT to obtain the subset of similar models available in the source domain, Υ_{sim}. However, IdCS exploits the static nature of conceptual similarity to remove the requirement of a user defined threshold to determine whether to transfer. Instead, parameterless conceptual clustering, presented in Section 6.3 is used, where STSC [100] allows clusters of conceptually similar base models to be identified without requiring the number of clusters to be predefined [56, 100]. STSC has previously been used in Chapter 6 for base model selection, identifying a subset of relevant yet diverse base models which are used as input to the meta-learner in BOTL [56]. Therefore, using STSC with conceptual similarity as a diversity metric is well suited to informing the decision of whether to transfer in online TL frameworks. When a model, $f_i^{\mathcal{A}}$, is selected for transfer in the source domain, $\mathcal{A}$, a subset of similar base models, Υ_{sim}, is created using STSC and the affinity matrix, Δ, containing pairwise conceptual similarities between base models, creating clusters of similar base models,

$$\Upsilon = \{\Upsilon_1, \ldots, \Upsilon_{\text{sim}}, \ldots, \Upsilon_l\} \qquad \text{using STSC}(\Delta)\ [100], \text{where} f_i^{\mathcal{A}} \in \Upsilon_{\text{sim}}, \tag{7.2}$$

where $\Upsilon = \{\Upsilon_1, \ldots, \Upsilon_l\}$ are the l clusters of similar models created by applying STSC to the affinity matrix, Δ, of conceptual similarities between models, and Υ_{sim} is the cluster that contains the model selected for transfer, $f_i^{\mathcal{A}}$. Once Υ_{sim} has been identified, IdCS uses the same methodology as IdDT to determine whether to transfer model $f_i^{\mathcal{A}}$, and whether it should be replaced by a model in Υ_{sim} using Algorithms 11 and 12 respectively.

One benefit of using IdCS is that a user defined diversity threshold is not required to identify similar base models due to the use of STSC. While conceptual similarity can be used as a diversity metric in IdDT in combination with a diversity threshold, the use of metrics such as MI, which are dependent on the current distribution of observable data, cannot feasibly be used alongside STSC [56]. This is because SC algorithms typically have complexity $O(n^2)$ to create the similarity matrix, and $O(n^3)$ for spectral analysis [93], where n is the number of base models. Therefore, using metrics such as MI, which must be updated as the data stream progresses, would incur a high computational overhead for updating the similarity matrix, whereas static diversity metrics such as conceptual similarity only require the similarity matrix to be updated when new base models are learnt, or received

from other data streams [56].

7.3 Experimental Set-Up

Variants of the BOTL framework, presented in Chapters 5 and 6, are used to demonstrate the use of IdDT and IdCS for determining whether to transfer models in online TL. This section briefly discusses the BOTL variants used in this chapter, and identifies characteristics of the datasets and CDDs presented in Chapters 3 and 4 which impact the decision of whether to transfer models.

7.3.1 BOTL Frameworks

Three existing variants of BOTL [54, 55, 56] are used to illustrate the use of IdDT and IdCS. First, the original BOTL method presented in Chapter 5 is used, where all stable base models are transferred and combined with the local model that has been learnt to represent the current concept using an OLS meta-learner [54]. BOTL, as presented in Chapter 5, is used as a baseline approach in this chapter, since it shows the predictive performance achievable when no base model selection strategies are used, and the decision of whether to transfer is not considered.

Second, MI-Thresh [54, 55], with the culling parameter values shown to be effective in [55], [56], and in Chapter 6, for the datasets used in this thesis is used, such that $\lambda_{\text{perf}} = 0.2$ and $\lambda_{\text{MI}} = 0.2$. MI-Thresh is used as a baseline to show the predictive performance achievable when a relevant yet diverse subset of base models is obtained via parameterised thresholding. Since MI is used as the diversity metric to select base models in the receiving domain, MI-Thresh is also used as the foundational framework for IdDT, where MI is used to determine whether to transfer. This allows for a direct comparison between MI-Thresh and IdDT, highlighting the efficacy of using MI to determine whether to transfer in online TL frameworks.

Third, CS-Clust is used, as presented in Chapter 6, which uses parameterless conceptual clustering to select a subset of base models [56]. Similarly to MI-Thresh, CS-Clust, is used as a baseline to show the performance that can be achieved without requiring a user defined thresholding parameter. CS-Clust is also used as the foundation for IdCS, where the conceptual similarity of base models is used to determine whether to transfer. Since CS-Clust and IdCS use STSC and conceptual similarity, the clustering of base models can be used to both determine whether to transfer and to select subsets of base models. Therefore, no additional computation is required to use IdCS when CS-Clust is also used. In addition to these three variants of BOTL, the underlying CDD is also used as a baseline to show the predictive

performance achievable when each domain learns independently without knowledge transfer.

7.3.2 CDDs

RePro [95], ADWIN [8] and AWPro [55], presented in Chapter 4, are used as underlying CDDs to detect concept drifts and obtain base models in each data stream. These three CDDs exhibit characteristics that are influential for determining whether to transfer. ADWIN does not reuse historical models in the presence of recurring concepts, and therefore multiple predictive models may be learnt from a single data stream that represent the same concept. Transferring such models provides little to no benefit to the receiving domain, and therefore, identifying similar models that have been learnt to represent the same concept will help prevent unnecessary knowledge transfer.

Another important characteristic of the underlying CDD used in online TL is the window in which a base model was learnt, and the extent to which it is representative of the underlying distribution of the current concept. ADWIN and AWPro estimate the precise point of drift within the window of recent observations, allowing instances belonging to the previous concept to be discarded prior to learning each base model [8, 55]. This is beneficial for determining whether to transfer because the similarity between base models is not influenced by the order in which concepts are encountered, which may not be the same in other domains, or at different periods in the data stream if concepts reoccur. However, this approach is not used by RePro, which may hider the identification of conceptually similar base models. For example, suppose concept c_i is followed by concept c_j in domain $\mathcal{A}$. Model $f_j^{\mathcal{A}}$ is learnt to represent concept c_j, but the window of data used to learn $f_j^{\mathcal{A}}$ is likely to contain instances belonging to concept c_i since it may not take an entire window of instances to be observed before a concept drift is detected. However, suppose that in domain $\mathcal{B}$, concept c_j is preceded by concept c_h, and that model $f_j^{\mathcal{B}}$ is learnt to represent concept c_j, and so the window of data used to learn $f_j^{\mathcal{B}}$ is likely to contain instances belonging to c_h. This means that although models $f_j^{\mathcal{A}}$ and $f_j^{\mathcal{B}}$ have been learnt to represent the same concept c_j, the retention of data belonging to preceding concepts may influence measures of diversity, and prevent base model similarities from being identified.

7.3.3 Datasets

The drifting hyperplane, smart home heating simulator, and real-world following distance datasets presented in Chapter 3 have been used to evaluate IdDT and IdCS. Each of these datasets also exhibit characteristics that may influence the decision of whether to transfer.

The drifting hyperplane datasets contain large numbers of recurring concepts within each data stream which have been artificially introduced. Additionally each data stream is guaranteed to share at least 3 concepts with another data stream. Therefore, to determine whether to transfer, a domain must consider whether a similar locally learnt base model has previously been transferred, and whether a similar base model has been received from another domain.

The heating simulator dataset contains recurring concepts due to the cyclic nature of the weather data used to generate each data stream. However, recurring concepts occur less frequently than in the drifting hyperplane datasets, and concept drifts induced by changing weather conditions occur gradually over long periods throughout the data stream. Although the decision of whether to transfer a model will be impacted by the presence of recurring concepts in these data streams, the presence of similar concepts within independent data streams will be a more significant factor that impacts the decision of whether to transfer since data streams have been generated from sampling overlapping time periods of weather data.

Finally, in the following distance dataset, each data stream captures data for a journey, where a large proportion of the journeys were recorded in similar geographical locations. This means that the likelihood of encountering conceptually similar concepts within different data streams is high. However, since each data stream captures only a single journey, the likelihood of encountering conceptually similar concepts within a single data stream is less likely than in the drifting hyperplane data streams where recurring concepts are repeatedly introduced artificially. Therefore, the identification of similar models learnt in different domains is important for deciding whether to transfer in the following distance data streams.

7.4 Experimental Results

The impact of negative transfer is inherently reduced in online TL frameworks due to the use of meta-learners or ensembles to combine transferred base models. Therefore, the decision of whether to transfer may not significantly impact the predictions obtained in online TL frameworks. Instead, techniques such as IdDT and IdCS should be evaluated with respect to the reduction in communication and com-

putation achieved by preventing the transfer of base models that provide little or no benefit to the receiving domain. To ensure that the proposed approaches are effective, each technique aims to maintain comparable predictive performances to frameworks that transfer all base models, while reducing communication and computation overheads.

7.4.1 Maintaining Predictive Performance

MI-Thresh and CS-Clust are the foundations of IdDT and IdCS respectively, and therefore direct comparisons can be made to consider the effect of determining whether to transfer. First, the predictive performance achieved by each of these approaches is considered to ensure that IdDT and IdCS obtain comparable performances to when all base models are transferred.

Tables 7.2–7.5 show the predictive performances, average and largest number of models used by the meta-learner, and the number of metric calculations required to obtain subsets of base models to be used as input to the meta-learner, for each of the methods used with sudden drifting hyperplane, gradual drifting hyperplane, heating simulator, and following distance datasets respectively. Across each of these datasets IdDT and IdCS achieve similar predictive performances to MI-Thresh and CS-Clust respectively, indicating that the decision of whether to transfer does not negatively impact the predictive performances. Additionally, the average number of base models used by the OLS meta-learner, $\mathcal{M}'$, and the largest number of base models used by the meta-learner, $\lceil \mathcal{M}' \rceil$, remains consistent with the MI-Thresh and CS-Clust counterparts. This indicates that the decision of whether to transfer has little impact on the ability to select a relevant yet diverse subset of base models.

Tables 7.2b, 7.2d, 7.3b, 7.3d and 7.5 present results for SuddenB, SuddenD, GradualB, GradualD and following distance datasets. In these settings BOTL achieves poor R^2 predictive performance in comparison to the underlying CDD, which is caused by the meta-learner overfitting [55, 56]. As discussed in Chapter 6, MI-Thresh prevents the meta-learner from overfitting, however, requires large numbers of relevancy and diversity metric calculations. CS-Clust requires fewer relevancy and diversity metric calculations, but it not able to prevent the meta-learner from overfitting in all datasets.

IdCS performs similarly to CS-Clust in the datasets where BOTL suffers from overfitting. However, when IdCS is used in the SuddenB datasets with RePro as an underlying CDD, the addition of the decision of whether to transfer prevents the meta-learner from overfitting, obtaining R^2 predictive performances that are statistically significantly greater than the underlying CDD ($p < 0.01$), and obtain-

(a) SuddenA: sudden drifting hyperplanes with uniform noise.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.883	0.884	1	1	0	0.851	0.854	1	1	0	0.830	0.835	1	1	0
BOTL	0.834	0.845	24	31	0	*0.828	0.839	41	57	0	***0.884**	0.886	17	21	0
MI-Thresh	0.902	0.902	4	5	28466	0.887	0.889	3	4	51163	0.880	0.882	3	4	17031
IdDT	*0.899	0.899	3	4	11258	***0.887**	0.888	3	4	24640	*0.876	0.878	3	4	7462
CS-Clust	***0.904**	0.904	5	8	5700	*0.885	0.887	6	8	5441	*0.878	0.879	5	8	1292
IdCS	*0.902	0.903	5	7	3523	*0.884	0.886	5	8	2866	*0.876	0.877	4	6	897

(b) SuddenB: sudden drifting hyperplanes with single sensor failure.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.883	0.884	1	1	0	0.830	0.836	1	1	0	0.808	0.814	1	1	0
BOTL	-2e+21	0.507	26	34	0	-7e+22	0.499	41	60	0	-3e+22	0.532	20	27	0
MI-Thresh	0.904	0.904	4	5	27301	0.879	0.880	3	5	47080	0.873	0.874	3	4	17510
IdDT	*0.902	0.902	3	4	13320	***0.879**	0.881	3	5	21097	***0.870**	0.871	3	4	8653
CS-Clust	-2e+19	0.870	5	8	5322	-2e+18	0.856	6	8	5627	*0.868	0.870	5	8	1666
IdCS	***0.905**	0.906	5	7	3715	-4e+18	0.848	5	8	3024	*0.865	0.867	5	7	1194

(c) SuddenC: sudden drifting hyperplanes with intermittent single sensor failure.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.873	0.875	1	1	0	0.845	0.848	1	1	0	0.847	0.850	1	1	0
BOTL	0.841	0.850	25	32	0	0.813	0.827	41	57	0	***0.888**	0.889	18	21	0
MI-Thresh	0.905	0.905	3	5	27026	0.882	0.883	3	4	45049	0.881	0.882	3	4	16104
IdDT	*0.904	0.904	3	4	10224	***0.885**	0.886	3	4	16777	*0.878	0.879	2	3	6901
CS-Clust	***0.909**	0.909	5	7	5276	***0.885**	0.886	6	8	5457	*0.887	0.888	5	9	1369
IdCS	*0.908	0.908	5	7	3108	*0.884	0.885	5	8	2549	*0.884	0.885	4	7	960

(d) SuddenD: sudden drifting hyperplanes with gradual sensor deterioration.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.870	0.871	1	1	0	0.845	0.849	1	1	0	0.810	0.816	1	1	0
BOTL	-1e+22	0.341	27	35	0	-9e+21	0.334	41	59	0	-2e+21	0.356	18	22	0
MI-Thresh	0.900	0.901	3	5	28144	0.885	0.886	3	4	42669	0.873	0.874	3	4	14776
IdDT	*0.897	0.898	3	4	13852	***0.884**	0.885	3	4	20328	*0.867	0.869	2	4	7912
CS-Clust	***0.901**	0.902	5	8	5850	***0.884**	0.886	6	8	5416	***0.869**	0.870	5	9	1392
IdCS	***0.901**	0.901	5	7	4004	***0.884**	0.885	5	7	2898	*0.867	0.868	4	8	1024

Table 7.2: Sudden Drifting Hyperplane: R^2 and PMCC^2 predictive performance, the average number of base models used by the meta-learner ($|\mathcal{M}'|$), the maximum number of base models used by the meta-learner ($\lceil\mathcal{M}'\rceil$), and the average number of relevancy and diversity metric calculations required to compare and evaluate base models (M.Calcs.) for meta-learners in BOTL for variants of the sudden drifting hyperplane datasets when transferring base models between 5 data streams. Improved predictive performances with statistical t-test values $p < 0.01$ compared to the underlying CDD, while requiring fewer relevancy and diversity metric calculations than MI-Thresh are indicated with *. Of these, bold type indicates the approach with highest performance.

(a) GradualA: gradual drifting hyperplanes with uniform noise.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.848	0.848	1	1	0	0.796	0.803	1	1	0	0.798	0.804	1	1	0
BOTL	0.776	0.800	31	37	0	0.792	0.811	39	54	0	***0.883**	0.886	20	25	0
MI-Thresh	0.892	0.892	4	6	55814	0.885	0.885	4	5	73104	0.887	0.887	4	5	30561
IdDT	*0.890	0.890	3	5	17594	***0.884**	0.884	4	5	27556	*0.878	0.878	3	4	11203
CS-Clust	***0.898**	0.899	5	7	12090	*0.880	0.881	6	8	5167	*0.880	0.880	5	8	1584
IdCS	***0.898**	0.899	5	7	7369	*0.880	0.881	5	8	2354	*0.878	0.879	4	6	1023

(b) GradualB: gradual drifting hyperplanes with single sensor failure.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.859	0.860	1	1	0	0.757	0.771	1	1	0	0.697	0.719	1	1	0
BOTL	-6e+20	0.323	29	38	0	-1e+21	0.322	40	57	0	-1e+21	0.348	21	28	0
MI-Thresh	0.887	0.888	4	5	45279	0.862	0.863	3	5	59037	0.845	0.846	3	4	25333
IdDT	***0.887**	0.887	3	5	19477	***0.864**	0.864	3	5	22717	***0.835**	0.835	3	4	11071
CS-Clust	-1e+17	0.870	6	8	12917	-5e+16	0.829	6	8	5366	-6e+10	0.832	5	7	1712
IdCS	-1e+17	0.863	5	7	7728	-3e+17	0.836	5	7	2591	*0.827	0.828	4	7	1020

(c) GradualC: gradual drifting hyperplanes with intermittent single sensor failure.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.842	0.842	1	1	0	0.773	0.779	1	1	0	0.759	0.773	1	1	0
BOTL	0.781	0.803	29	36	0	0.776	0.799	39	54	0	***0.873**	0.875	21	27	0
MI-Thresh	0.890	0.890	4	6	52092	0.878	0.878	4	5	69921	0.870	0.871	4	5	29440
IdDT	*0.887	0.888	4	5	24079	***0.882**	0.883	4	5	32570	*0.854	0.854	3	4	12251
CS-Clust	***0.892**	0.893	5	8	12375	*0.865	0.865	6	8	5059	*0.865	0.866	5	8	1666
IdCS	***0.892**	0.892	5	7	8298	*0.864	0.865	5	8	2637	*0.861	0.862	5	7	1033

(d) GradualD: gradual drifting hyperplanes with gradual sensor deterioration.

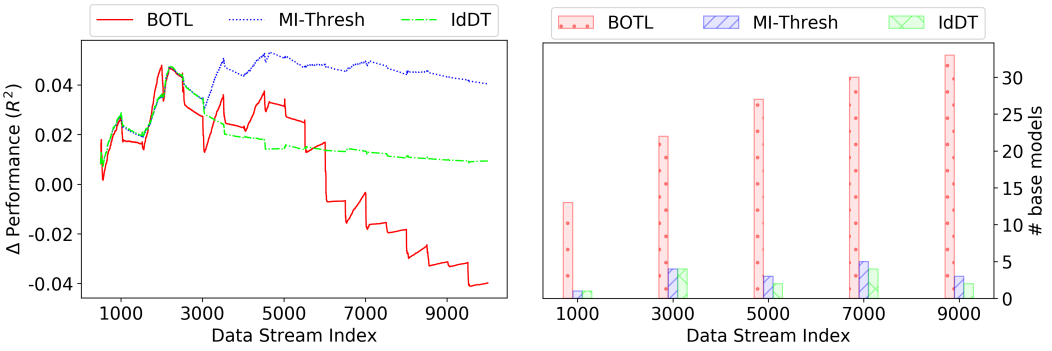
	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.846	0.847	1	1	0	0.484	0.593	1	1	0	0.519	0.629	1	1	0
BOTL	-3e+22	0.147	31	41	0	-2e+22	0.143	26	44	0	-2e+22	0.129	17	26	0
MI-Thresh	0.898	0.898	4	6	45921	0.812	0.813	3	5	33128	0.821	0.821	3	4	19825
IdDT	*0.894	0.895	3	5	25772	***0.819**	0.820	3	4	19072	***0.799**	0.800	3	4	10814
CS-Clust	***0.898**	0.898	6	7	15333	*0.789	0.790	5	10	2698	*0.787	0.789	5	8	1310
IdCS	*0.897	0.898	5	7	10455	*0.778	0.780	5	8	1488	*0.779	0.780	4	7	828

Table 7.3: Gradual Drifting Hyperplane: R^2 and PMCC^2 predictive performance, the average number of base models used by the meta-learner ($|\mathcal{M}'|$), the maximum number of base models used by the meta-learner ($\lceil\mathcal{M}'\rceil$), and the average number of relevancy and diversity metric calculations required to compare and evaluate base models (M.Calcs.) for meta-learners in BOTL for variants of the gradual drifting hyperplane datasets when transferring base models between 5 data streams. Improved predictive performances with statistical t-test values $p < 0.01$ compared to the underlying CDD, while requiring fewer relevancy and diversity metric calculations than MI-Thresh are indicated with *. Of these, bold type indicates the approach with highest performance.

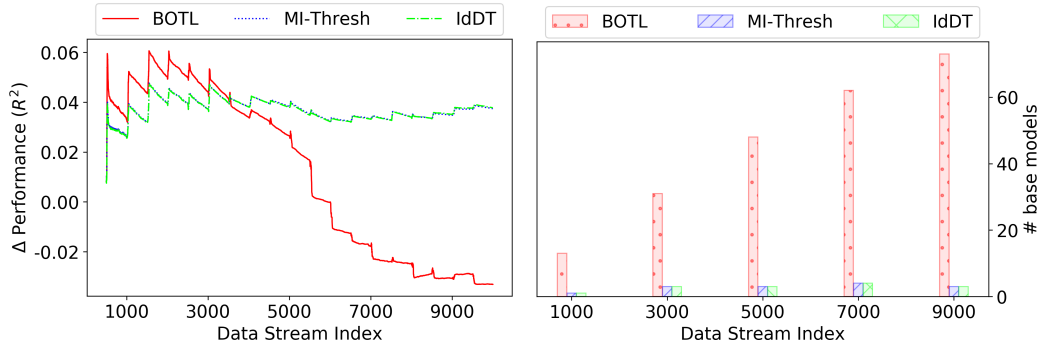
ing the best R^2 and PMCC^2 performances of all framework variants. IdCS is also able to prevent the meta-learner from overfitting for the GradualB dataset when AWPro is used as the CDD, achieving a R^2 predictive performance statistically significantly greater ($p < 0.01$) than the underlying CDD alone. IdDT achieves comparable predictive performances to MI-Thresh for all datasets, achieving R^2 predictive performances that are statistically significantly greater than the underlying CDD ($p < 0.01$). With the exception of SuddenB and GradualB datasets in Tables 7.2b and 7.3b, IdCS also achieves this.

Figures 7.1 and 7.2 show the difference in R^2 predictive performance obtained by BOTL, MI-Thresh and IdDT, and BOTL, CS-Clust and IdCS, in comparison to the underlying CDD respectively, for a sudden drifting hyperplane dataset. The improvement in predicted performance in comparison to the underlying CDD observed by IdDT and IdCS when using ADWIN and AWPro as the underlying CDD are similar to those observed by MI-Thresh and CS-Clust. This highlights that when ADWIN and AWPro are used as CDDs, the decision of whether to transfer has minimal effect on the predictive performances achieved by these approaches. However, when RePro is used as the underlying CDD, IdDT and IdCS obtain a smaller increase in predictive performance than MI-Thresh and CS-Clust, where all base models are transferred between domains. This indicates that the decision of whether to transfer base models has a larger impact on predictive performance when RePro is used as the underlying CDD. This highlights the importance of ADWIN and AWPro's ability to estimate the precise point of concept drift to ensure that predictive models are not learnt from data containing instances belonging to the previous concept. Since RePro does not exhibit this characteristic, the identification of similar base models using metrics such as MI or conceptual similarity can be influenced by the order in which concepts are encountered in each domain. Therefore, IdDT and IdCS achieve smaller increases in predictive performance over the underlying CDD in comparison to transferring all base models between domains. Although a smaller increase in performance is observed, IdDT and IdCS achieve an improved predictive performance in comparison to BOTL due to the use of base model selection techniques to determine the subset of base models to be used as input to the meta-learner.

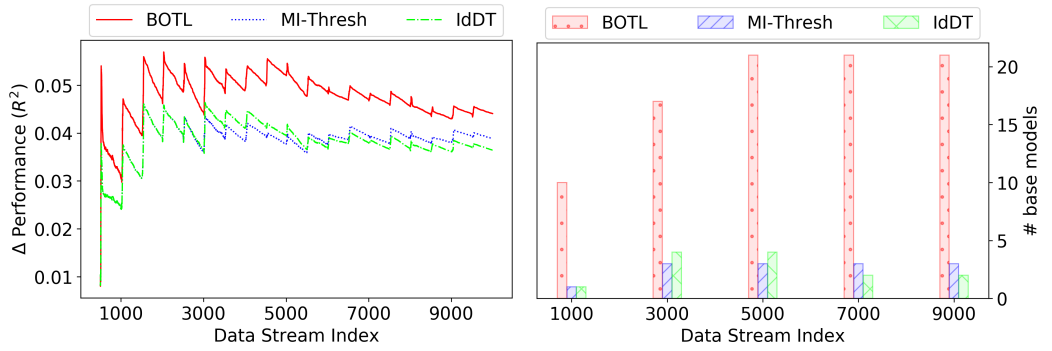
Table 7.4 shows that BOTL achieves the greatest predictive performance in the heating simulator dataset, whereas Table 7.5 shows that MI-Thresh and CS-Clust obtain higher predictive performances than BOTL for the following distance dataset. This occurs due to the larger window size of instances used in the heating simulator data streams, reducing the likelihood of the meta-learner overfitting due



(a) RePro

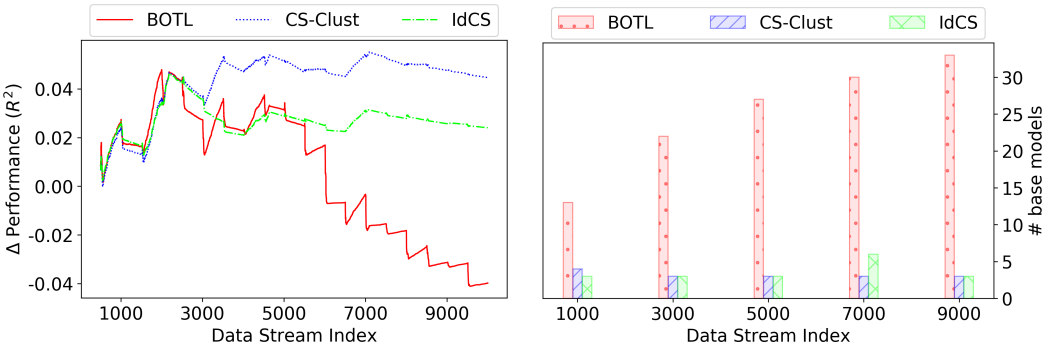


(b) ADWIN

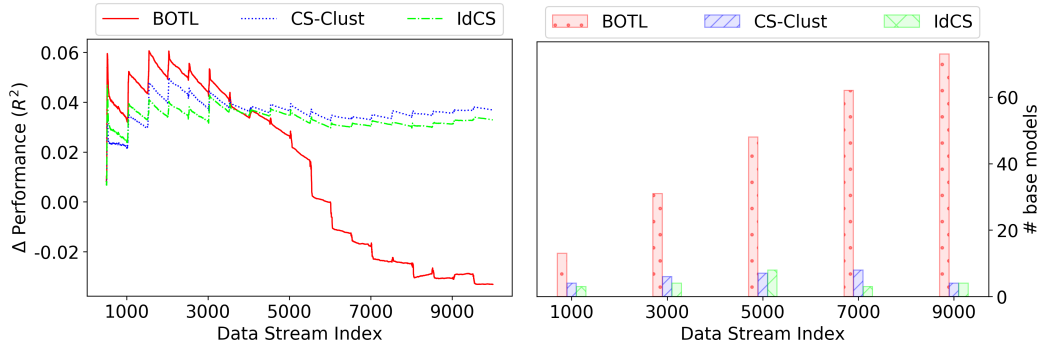


(c) AWPro

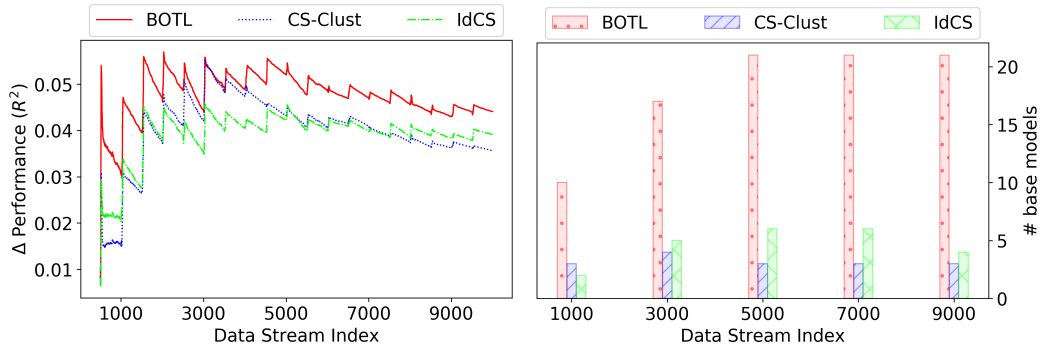
Figure 7.1: BOTL, MI-Thresh and IdDT vs. CDDs: The difference in R^2 performance (left) between the OLS meta-learner in BOTL, MI-Thresh and IdDT vs. the underlying CDDs of (a) RePro, (b) ADWIN, and (c) AWPro, and the number of models used as base models (right) for a sudden drifting hyperplane data stream (SuddenA). Base models are learnt locally and transferred from 4 other sudden drifting hyperplane data streams.



(a) RePro



(b) ADWIN



(c) AWPro

Figure 7.2: BOTL, CS-Clust and IdCS vs. CDDs: The difference in R^2 performance (left) between the OLS meta-learner in BOTL, CS-Clust and IdCS vs. the underlying CDDs of (a) RePro, (b) ADWIN, and (c) AWPro, and the number of models used as base models (right) for a sudden drifting hyperplane data stream (SuddenA). Base models are learnt locally and transferred from 4 other sudden drifting hyperplane data streams.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.636	0.652	1	1	0	0.635	0.655	1	1	0	0.625	0.645	1	1	0
BOTL	***0.728**	0.735	10	15	0	***0.717**	0.725	9	15	0	***0.727**	0.734	10	15	0
MI-Thresh	0.706	0.713	3	5	3441	0.705	0.713	3	5	4113	0.708	0.716	3	4	4034
IdDT	*0.705	0.712	3	4	1646	*0.707	0.715	3	5	1483	*0.694	0.702	3	4	2802
CS-Clust	*0.708	0.715	3	5	968	*0.707	0.715	4	5	1072	*0.707	0.715	3	5	1375
IdCS	*0.708	0.715	3	4	814	*0.704	0.712	3	4	812	*0.705	0.712	3	4	943

Table 7.4: Heating Simulator: R^2 and PMCC^2 predictive performance, the average number of base models used by the meta-learner ($|\mathcal{M}'|$), the maximum number of base models used by the meta-learner ($\lceil\mathcal{M}'\rceil$), and the average number of relevancy and diversity metric calculations required to compare and evaluate base models (M.Calcs.) for meta-learners in BOTL for the smart home heating simulator dataset when transferring base models between 5 data streams. Improved predictive performances with statistical t-test values $p < 0.01$ compared to the underlying CDD, while requiring fewer relevancy and diversity metric calculations than MI-Thresh are indicated with *. Of these, bold type indicates the approach with highest performance.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.546	0.565	1	1	0	0.441	0.502	1	1	0	0.396	0.515	1	1	0
BOTL	*0.646	0.678	6	12	0	-2e+15	0.384	12	25	0	-9e+9	0.665	8	18	0
MI-Thresh	0.665	0.688	2	3	1116	0.663	0.682	3	4	2096	0.693	0.705	2	4	1287
IdDT	***0.673**	0.695	2	3	874	*0.656	0.676	2	4	1431	*0.680	0.692	2	4	821
CS-Clust	*0.661	0.686	3	5	779	***0.673**	0.695	4	8	1143	***0.705**	0.716	3	7	730
IdCS	*0.658	0.684	3	4	647	*0.663	0.687	4	8	788	*0.686	0.701	3	5	594

Table 7.5: Following Distance: R^2 and PMCC^2 predictive performance, the average number of base models used by the meta-learner ($|\mathcal{M}'|$), the maximum number of base models used by the meta-learner ($\lceil\mathcal{M}'\rceil$), and the average number of relevancy and diversity metric calculations required to compare and evaluate base models (M.Calcs.) for meta-learners in BOTL for the following distance dataset when transferring base models between 7 data streams. Improved predictive performances with statistical t-test values $p < 0.01$ compared to the underlying CDD, while requiring fewer relevancy and diversity metric calculations than MI-Thresh are indicated with *. Of these, bold type indicates the approach with highest performance.

to an increased training sample size [25, 56, 81]. Similarly to the drifting hyperplane datasets, both IdDT and IdCS achieve comparable predictive performances to BOTL, MI-Thresh and CS-Clust, while improving predictive performance with statistical significance ($p < 0.01$) when compared to using the underlying CDD alone.

The results presented in Tables 7.2–7.5 also show that the number of base models used as input to the meta-learner, $\mathcal{M}'$, remains consistent between MI-Thresh and IdDT, and CS-Clust and IdCS. This indicates that the addition of the decision of whether to transfer has little impact on the base model selection strategies used by MI-Thresh and CS-Clust. Since the predictive performances obtained by IdDT and IdCS are comparable to MI-Thresh and CS-Clust respectively, and $\mathcal{M}'$ and $\lceil \mathcal{M}' \rceil$ are consistent, this indicates that the models that were not transferred as a result of using IdDT and IdCS provided little to no beneficial knowledge to the receiving domains.

7.4.2 Communication and Computation Overhead

The impact of IdDT and IdCS on the communication and computation overheads in online TL are important factors to consider. Figures 7.3–7.6 show the average number of base models received by each domain for MI-Thresh, IdDT, CS-Clust and IdCS frameworks for sudden drifting hyperplane, gradual drifting hyperplane, heating simulator and following distance datasets respectively. These figures show that deciding whether to transfer using IdDT and IdCS reduces the number of base models received by each domain, indicating that IdDT and IdCS reduce communication overheads across all dataset types, and for every underlying CDD used in this evaluation.

The greatest reduction in number of models received by domains is observed when ADWIN is used as the underlying CDD since ADWIN does not reuse historical models in the presence of recurring concepts. Therefore, without the decision of whether to transfer, multiple models may be learnt and transferred which have been learnt to represent the same concept from a single data stream containing recurring concepts.

Averaged across all results presented in Figures 7.3–7.6, IdDT reduces the average number of models received by each domain by 47.1%, while IdCS reduces this by 30.0%, in comparison to when all models are transferred. Across all dataset types, IdDT achieves a greater reduction in the number of base models received by domains compared to IdCS. However, IdDT makes the assumption that the base models which are considered to be similar to the model selected for transfer in the source

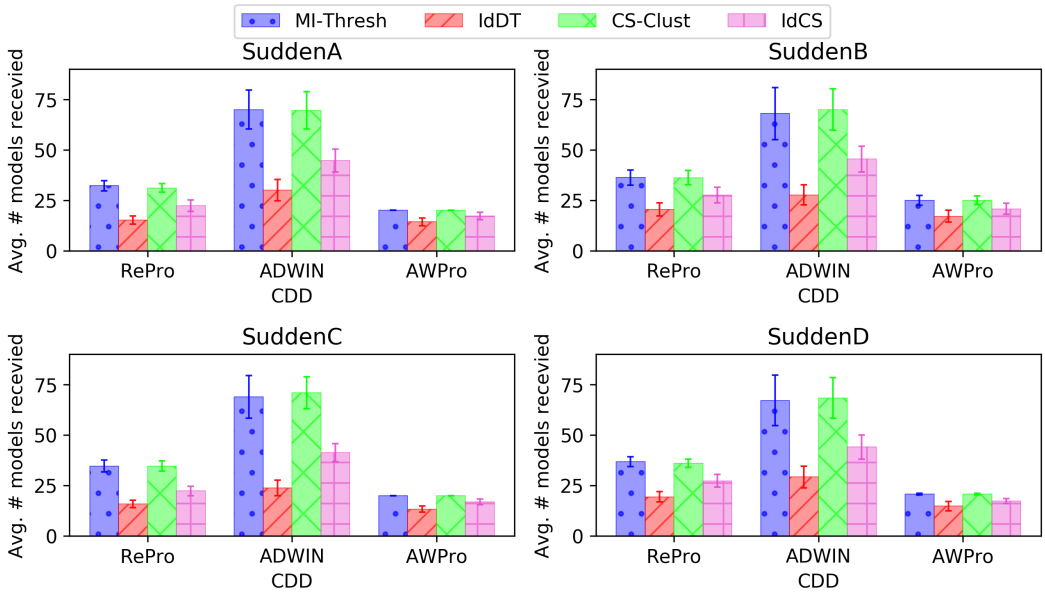


Figure 7.3: Sudden Drifting Hyperplane: The average number of models received per domain for MI-Thresh, IdDT, CS-Clust and IdCS when using RePro, ADWIN and AWPro as the underlying CDDs for 5 drifting hyperplane data streams.

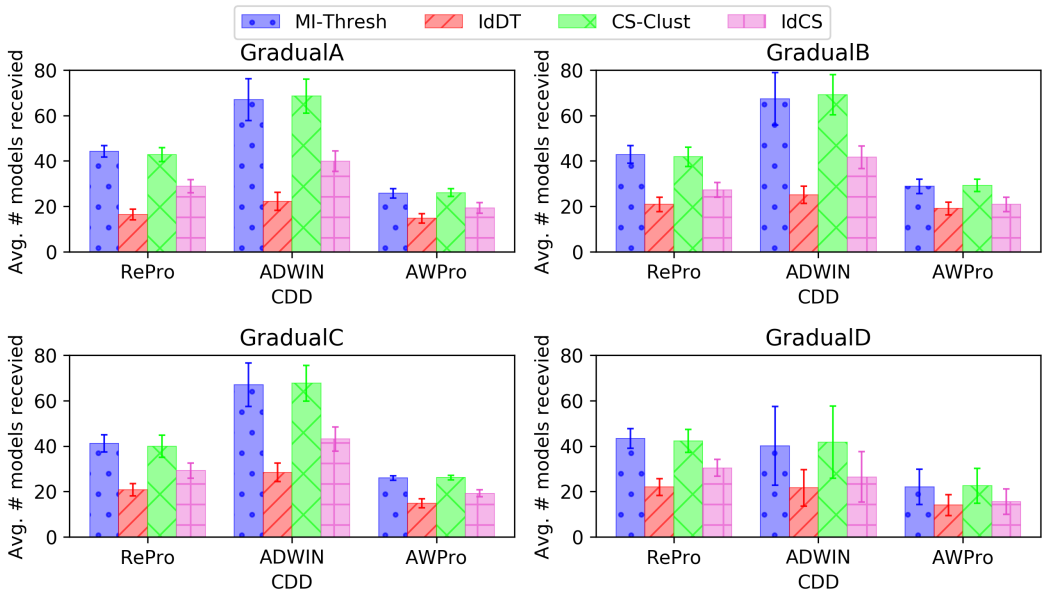


Figure 7.4: Gradual Drifting Hyperplane: The average number of models received per domain for MI-Thresh, IdDT, CS-Clust and IdCS when using RePro, ADWIN and AWPro as the underlying CDDs for 5 drifting hyperplane data streams.

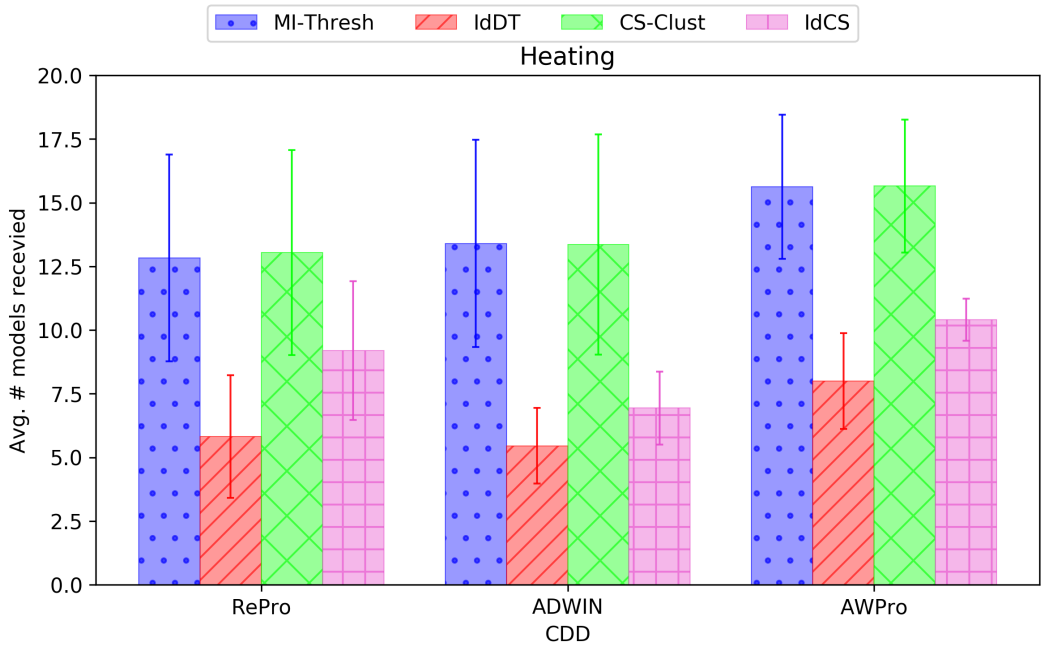


Figure 7.5: Heating Simulator: The average number of models received per domain for MI-Thresh, IdDT, CS-Clust and IdCS when using RePro, ADWIN and AWPro as the underlying CDDs for 5 heating simulator data streams.

domain will also be considered similar in the receiving domains. This may not be the case in real-world environments due to the dependency on the distribution of data used to obtain predictions to calculate the MI between models, however, Figures 7.3–7.6 show that IdDT achieves the greatest reduction in communication overhead. Tables 7.2–7.5 present the number of relevancy and diversity metric calculations required to both select a subset of base models as input to the meta-learner, and to determine whether to transfer, denoted by M.Calcs. These results indicate that IdCS has the smallest computational overhead, despite transferring more models than IdDT.

Although the decision of whether to transfer requires diversity metrics to be calculated to determine whether a model should be transferred to a receiving domain, the reduction in the number of models received by each domain reduces the number of pairwise relevancy and diversity metric calculations that are required to identify a subset of base models to be used as input to the meta-learner. Therefore, across all datasets, IdDT and IdCS reduce the computational overhead compared to MI-Thresh and CS-Clust respectively.

Despite IdCS transferring more models than IdDT, the number of M.Calcs is significantly fewer than for IdDT due to the use of conceptual similarity as a diversity

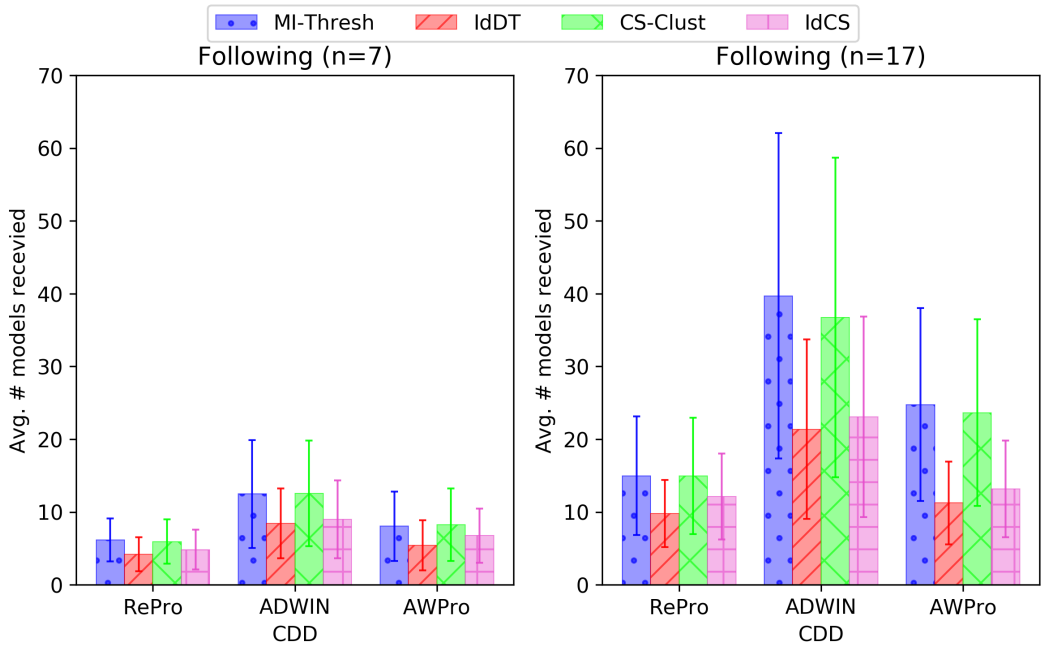


Figure 7.6: Following Distance: The average number of models received per domain for MI-Thresh, IdDT, CS-Clust and IdCS when using RePro, ADWIN and AWPro as the underlying CDDs for 7 and 17 following distance data streams.

metric, which remains static. IdCS is also beneficial since the decision of whether to transfer does not require any user defined threshold parameters. This makes IdCS more applicable to real-world environments since the threshold parameter used by IdDT may be challenging to determine prior to learning.

Figure 7.6 shows the average number of models received for the real-world following distance dataset. Results are presented for frameworks with 7 and 17 domains, which show that IdDT achieves reductions in the number of models received of 32.3%, 32.4 and 32.6% when RePro, ADWIN and AWPro are used as CDDs respectively in frameworks with 7 domains, and achieves reductions of 34.8%, 46.2% and 54.6% in frameworks with 17 domains. Similarly, IdCS achieves reductions of 18.5%, 28.1% and 18.4% in frameworks with 7 domains, and 18.9%, 37.2% and 44.2% with 17 domains for RePro, ADWIN and AWPro respectively. This shows that a higher proportion of base models are prevented from being transferred in IdDT and IdCS with 17 domains compared to 7. Therefore, to consider the scalability of IdDT and IdCS, they have been used in frameworks where the number of domains range between 2 and 17 using the following distance dataset.

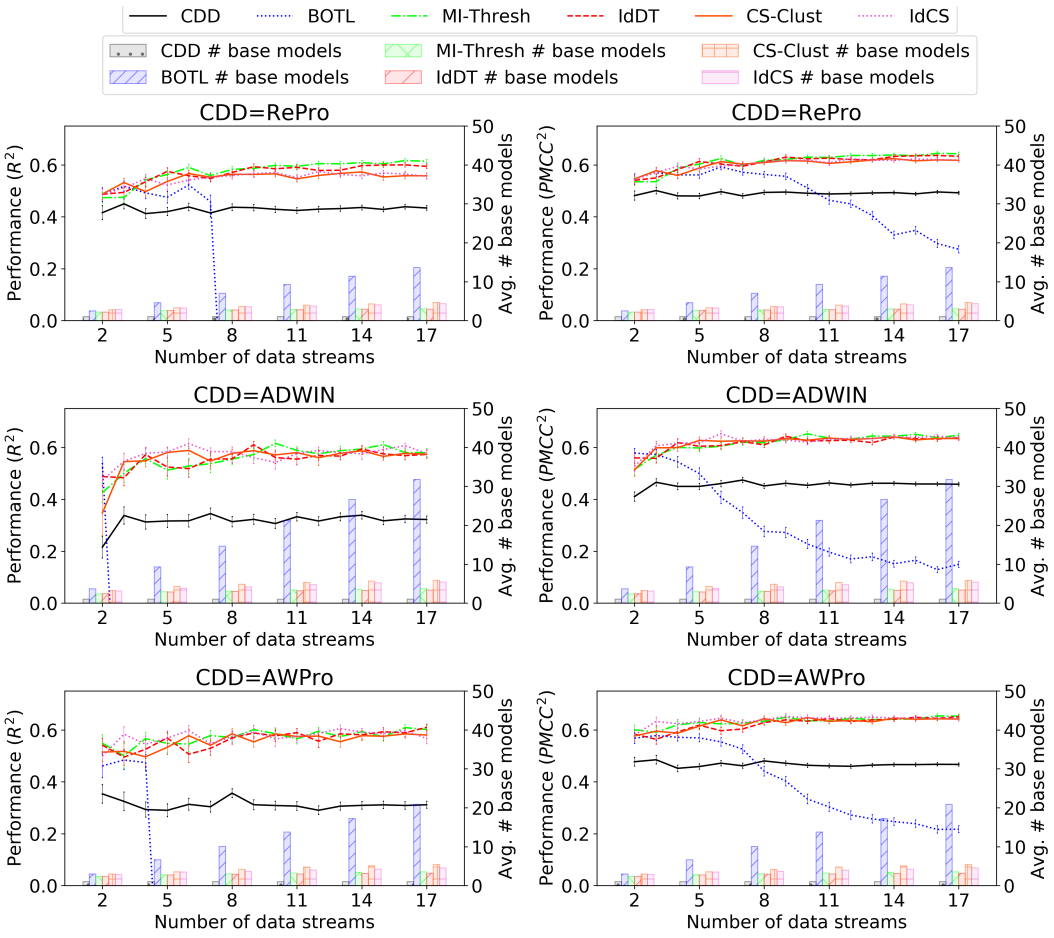


Figure 7.7: Following Distance: R^2 and PMCC^2 predictive performance, and number of base models used by BOTL meta-learners, for the underlying CDD, BOTL, MI-Thresh, IdDT, CS-Clust and IdCS with increasing numbers of following distance data streams.

7.4.3 Increasing Numbers of Domains

In this section, the performances of frameworks with increasing numbers of domains for the following distance dataset are presented. The number of models received by domains, and the average number of relevancy and diversity metric calculations are also considered, and used as indicators of communication and computation overheads as the framework increases in size.

The R^2 and PMCC^2 predictive performances achieved by IdDT and IdCS are shown alongside the underlying CDD, BOTL, MI-Thresh, and CS-Clust in Figure 7.7, for frameworks with increasing numbers of following distance data streams. This highlights that MI-Thresh and CS-Clust achieve the highest predictive perfor-

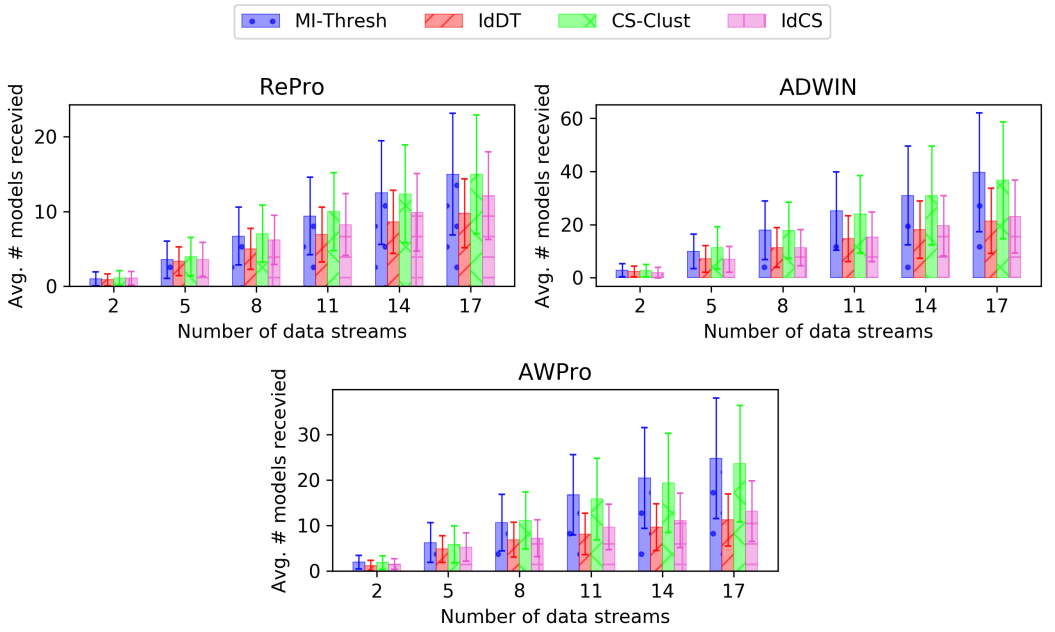


Figure 7.8: Following Distance: The average number of models received per domain for MI-Thresh, IdDT, CS-Clust and IdCS for increasing numbers of following distance data streams, using RePro, ADWIN and AWPro as the underlying CDDs.

mances, while IdDT and IdCS obtain similar predictive performances to MI-Thresh and CS-Clust, where all base models are transferred between domains, for all sized frameworks with differing underlying CDDs. Figure 7.7 also shows the average number of base models used by the OLS meta-learner for each of these approaches. These results highlight that each of the variants that implement base model selection techniques, MI-Thresh and CS-Clust, utilise a similar number of base models in order to achieve their respective predictive performances, which are considerably fewer than the number of base models used by BOTL with no base model selection. Figure 7.7 also shows that the absence of a base model selection technique causes the OLS meta-learner to overfit, even when the number of data streams in the framework is small.

Figure 7.8 shows the average number of base models received by following distance data streams with increasing numbers of domains in each framework. Figures 7.7 and 7.8 highlight that although knowledge transfer is reduced when using IdDT and IdCS, a predictive performance similar to that achieved when all base models are transferred can be achieved. This means that IdDT and IdCS allow communication overheads to be reduced, while maintaining predictive performance. As the number of data streams in the framework increases, IdDT and IdCS en-

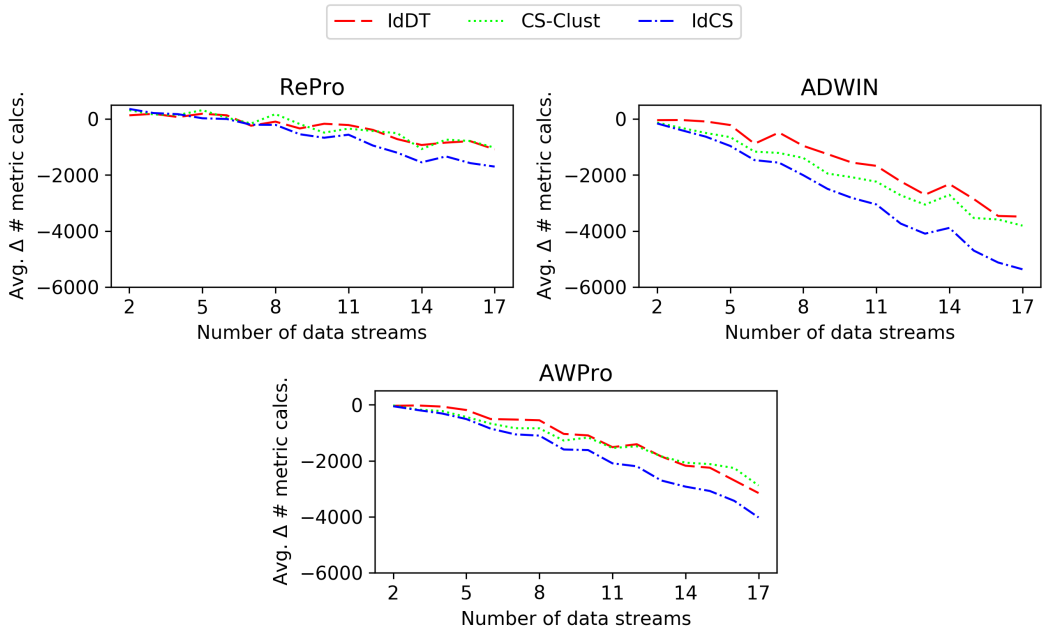


Figure 7.9: Following Distance: Change in number of relevancy and diversity metric calculations required to compare and evaluate base models for IdDT, CS-Clust and IdCS in comparison to MI-Thresh for increasing numbers of following distance data streams.

able greater reductions in the proportion of base models transferred, indicating that the similarity of base models learnt in different domains can be exploited to reduce communication overheads in real-world online TL, where the number of independent data streams can become large. The greatest reduction in average number of base models received in each domain for IdDT and IdCS is observed when ADWIN is used as the underlying CDD. This is because without the aid of IdDT or IdCS to determine whether to transfer, frameworks that implement ADWIN as the underlying CDD must transfer new base models when recurring concepts are encountered.

Figure 7.9 shows the average decrease in the number of relevancy and diversity metric calculations, M.Calcs, required to obtain a relevant yet diverse subset of base models to be used as input to the OLS meta-learner, and to determine whether to transfer, for IdDT, CS-Clust and IdCS in comparison to MI-Thresh, for frameworks with increasing numbers of following distance data streams. This shows that the use of conceptual similarity as a diversity metric reduces the computational overhead of selecting a relevant yet diverse subset of base models. However, the reduction in the number of models transferred when using IdDT and IdCS not only reduces communication overheads, but also reduces the computational overhead as-

sociated with base model selection for the OLS meta-learner, since there are fewer base models received in each domain. As the number of domains in the frameworks increases, a greater reduction in M.Calcs is observed because IdDT and IdCS are able to identify a larger number of similar models learnt from different data streams, preventing the transfer of increasing numbers of base models, as shown in Figure 7.8.

Overall, Tables 7.2–7.5, and Figures 7.3–7.9, show that IdDT and IdCS can be used to reduce the number of base models transferred in online TL, allowing communication and computation overheads to be reduced, without significantly impacting the overall predictive performances of the OLS meta-learner in the BOTL framework.

7.5 Summary

In this chapter, the question of whether to transfer models in online TL frameworks has been considered. Unlike existing offline TL research, the question of whether to transfer not only attempts to reduce the impact of negative transfer, but it also prevents the transfer of knowledge that provides little or no beneficial information to the receiving domain to reduce unnecessary communication and computation overheads. Determining whether to transfer is paramount to the feasibility of using online TL for real-world applications since communication and computational resources may be limited, and therefore, it may not be possible to transfer all knowledge between all domains.

IdDT has been proposed, where the decision of whether to transfer is determined by parameterised diversity thresholding, which is demonstrated in this chapter using MI. IdCS has also been proposed, where the decision of whether to transfer is determined by parameterless conceptual clustering. Both approaches have been shown to reduce communication and computation overheads in comparison to frameworks that transfer all available base models, while maintaining comparable predictive performance. This means that IdDT and IdCS successfully prevent the transfer of knowledge that provides little or no beneficial information to the receiving domains, and therefore could be beneficial for online TL in real-world applications where communication and computational resources are limited.

Table 7.6 summarises the reductions in communication and computation, and changes in R^2 and PMCC^2 predictive performances, achieved by IdDT in comparison to MI-Thresh and IdCS in comparison to CS-Clust, for all datasets used in this thesis. IdDT achieves the greatest reduction in communication overheads, and also achieves the greatest reduction in computation proportional to its corre-

Dataset		RePro ΔComm	RePro ΔComp	RePro ΔR^2	RePro Δ_{PMCC^2}	ADWIN ΔComm	ADWIN ΔComp	ADWIN ΔR^2	ADWIN Δ_{PMCC^2}	AWPro ΔComm	AWPro ΔComp	AWPro ΔR^2	AWPro Δ_{PMCC^2}
SuddenA	IdDT	-52.7%	-60.4%	-0.003	-0.003	-57.2%	-51.8%	-0.000	-0.001	-28.3%	-56.2%	-0.004	-0.004
	IdCS	-28.3%	-38.2%	-0.002	-0.002	-35.8%	-47.3%	-0.001	-0.001	-14.0%	-30.5%	-0.002	-0.002
SuddenB	IdDT	-43.5%	-51.2%	-0.002	-0.002	-59.2%	-55.2%	0.000	0.000	-31.3%	-50.6%	-0.003	-0.003
	IdCS	-23.8%	-30.2%	0.905	0.036	-35.1%	-46.3%	0.000	-0.007	-17.0%	-28.3%	-0.003	-0.003
SuddenC	IdDT	-54.1%	-62.2%	-0.001	-0.001	-65.4%	-62.8%	0.003	0.003	-33.5%	-57.1%	-0.003	-0.003
	IdCS	-35.7%	-41.1%	-0.001	-0.001	-41.8%	-53.3%	-0.001	-0.001	-15.5%	-29.8%	-0.003	-0.003
SuddenD	IdDT	-47.4%	-50.8%	-0.003	-0.003	-56.5%	-52.4%	-0.001	-0.001	-28.8%	-46.4%	-0.005	-0.005
	IdCS	-24.0%	-31.6%	-0.001	-0.001	-35.6%	-46.5%	-0.001	-0.001	-16.9%	-26.4%	-0.001	-0.001
GradualA	IdDT	-62.8%	-68.5%	-0.002	-0.002	-66.7%	-62.3%	-0.001	-0.001	-43.0%	-63.3%	-0.009	-0.009
	IdCS	-32.5%	-39.0%	0.000	0.000	-41.8%	-54.4%	0.000	0.000	-25.7%	-35.4%	-0.002	-0.002
GradualB	IdDT	-51.4%	-57.0%	0.000	0.000	-62.8%	-61.5%	0.002	0.002	-33.9%	-56.3%	-0.010	-0.010
	IdCS	-34.7%	-40.2%	0.000	-0.007	-39.8%	-51.7%	0.000	0.008	-28.4%	-40.4%	0.827	-0.004
GradualC	IdDT	-49.5%	-53.8%	-0.003	-0.003	-57.5%	-53.4%	0.004	0.004	-42.8%	-58.4%	-0.017	-0.016
	IdCS	-27.0%	-32.9%	0.000	0.000	-36.3%	-47.9%	-0.001	-0.001	-26.4%	-38.0%	-0.004	-0.004
GradualD	IdDT	-49.2%	-43.9%	-0.003	-0.003	-45.9%	-42.4%	0.007	0.007	-36.2%	-45.5%	-0.021	-0.021
	IdCS	-28.1%	-31.8%	-0.001	-0.001	-36.5%	-44.8%	-0.010	-0.010	-31.2%	-36.8%	-0.008	-0.008
Heating	IdDT	-54.7%	-52.2%	-0.001	-0.001	-59.3%	-63.9%	0.002	0.002	-48.8%	-30.5%	-0.014	-0.014
	IdCS	-29.5%	-15.9%	0.000	0.000	-48.1%	-24.3%	-0.003	-0.003	-33.6%	-31.4%	-0.002	-0.003
Following (n=7)	IdDT	-32.3%	-21.7%	0.008	0.007	-32.4%	-31.7%	-0.007	-0.006	-32.6%	-36.2%	-0.013	-0.013
	IdCS	-18.5%	-16.9%	-0.003	-0.002	-28.1%	-31.1%	-0.010	-0.008	-18.4%	-18.6%	-0.019	-0.015
Following (n=17)	IdDT	-34.8%	-26.5%	-0.020	-0.010	-46.2%	-44.0%	-0.007	-0.010	-54.6%	-58.4%	0.008	-0.002
	IdCS	-18.9%	-22.1%	-0.002	0.001	-37.2%	-37.9%	0.003	0.003	-44.2%	-45.3%	-0.014	0.002

Table 7.6: Summary of comparisons between MI-Thresh and IdDT, and CS-Clust and IdCS, showing the average reduction in number of models received (%), denoted as Δ Comm., the average reduction in performance and diversity metric calculations (%), denoted as Δ Comp., and the average difference between R^2 and PMCC^2 predictive performances.

sponding foundation framework which transfers all models, MI-Thresh. Although the reduction in computational overhead, as a percentage, is greater for IdDT, IdCS has a lower computational overhead since it uses a static diversity metric, conceptual similarity, that does not need to be recalculated, which could previously be seen in Tables 7.2–7.5 and Figure 7.9. Additionally, IdCS does not require user defined diversity threshold parameters to determine whether to transfer, which may make it more applicable for use in real-world online TL.

Chapter 8

Conclusion

This thesis has considered the online TL research paradigm where online TL is used in frameworks where a data rich offline source domain does not exist. This means that knowledge must be learnt from online domains to aid the predictions in other online domains. The research proposed in this thesis has three objectives, namely (i) to determine what can be learnt from online data streams that may be beneficial to other independent online data streams, (ii) to determine how transferred knowledge can be combined with locally learnt knowledge to aid predictions, and (iii) to determine whether knowledge should be transferred to other domains in an online TL framework. These three research objectives were used to guide the development of the contributions presented in this thesis.

For all proposed approaches, a common factor underpinned their evaluation, which was their feasibility with respect to real-world applications. This meant that an increase in predictive performance in receiving domains was not the only measure of success, and factors such as communication and computation overhead were important to consider. Throughout this thesis, all proposed approaches were evaluated using three types of regression based datasets, namely the drifting hyperplane datasets, the smart home heating simulator dataset, and the real-world following distance dataset. These datasets were used to gain insight into how the proposed approaches handled different types of concept drift, and how they performed in real-world environments.

In this chapter, the contributions of this thesis are summarised and avenues of future work are discussed.

8.1 Determining What to Transfer

To address the challenge of determining what to transfer in online TL, a novel CDD, called AWPro [55], was introduced in Chapter 4. AWPro combines key characteristics of existing CDDs, namely RePro [95] and ADWIN [8], that are beneficial when learning in online TL frameworks that have computation and communication limitations. First, it uses the sliding window based concept drift detection mechanism proposed by ADWIN to estimate the precise point of drift. This is beneficial when determining what to transfer in online TL since predictive models can be created from data belonging to a single concept by discarding instances observed prior to the estimated point of drift. This means that predictive models can be learnt without being influenced by data belonging to a previous concept, allowing models to be created that are more representative of the newly encountered concept, which could then be transferred to other online domains. Second, AWPro uses the proactive nature of RePro to prioritise the reuse of historical models in the presence of recurring concepts. This allows AWPro to adopt the use of a historical model when only half a window of instances belonging to the new concept have been observed, allowing AWPro to react quickly to concept drift when the new concept has previously been encountered. More importantly to online TL, this prevents multiple predictive models from being created that have been learnt to represent the same concept, reducing the number of models transferred throughout the framework, thereby reducing communication overheads.

Although AWPro achieves poorer predictive performances in comparison to RePro and ADWIN when used as a stand-alone method of learning, with no knowledge transfer, as shown in Chapter 4, it exhibits communication and computational qualities that are highly desirable in online TL. In Chapter 5, these characteristics enable AWPro to be used in online TL to determine what to transfer, achieving lower computational overhead due to a reduction in the number of unstable models created in comparison to RePro, and a lower communication overhead in comparison to ADWIN due to the reuse of historical models in the presence of recurring concepts. Additionally, these characteristics enabled online TL frameworks that employed AWPro as the underlying CDD to frequently outperform frameworks that used RePro or ADWIN as CDDs, despite achieving poorer predictive performances when used alone.

8.2 Determining How to Combine Transferred Knowledge

A large proportion of this thesis addresses the challenge of how transferred knowledge can be combined with locally learnt knowledge to best aid the predictions in each online domain. In Chapter 5, the BOTL framework [54, 55] was introduced, which allows knowledge to be transferred, in the form of predictive models, bi-directionally between online domains, which are then combined with locally learnt models through the use of an OLS meta-learner. Using BOTL provides three benefits over existing approaches to online TL. First, the use of CDDs enables predictive models to be learnt that represent each of the concepts encountered in a domain. Therefore, each domain in the framework can benefit from knowledge learnt about each individual source concept. Second, knowledge of newly encountered concepts are transferred between domains as each data stream progresses, and therefore, if a domain encounters a new concept, other domains can benefit from its knowledge prior to encountering it themselves. Third, knowledge transfer is bi-directional, and therefore all domains in the framework can benefit from knowledge transfer, rather than only aiding the predictive performance in a single target domain. This approach to transferring and combining models to improve the predictive performance in a receiving domain initially showed promise when the number of models available to the OLS meta-learner was small in comparison to the window of available data. However, as the number of base models transferred becomes large, BOTL is prone to overfitting, resulting in predictive performances that are worse than those achieved without knowledge transfer.

To address this, naïve model culling strategies were introduced in Chapter 5, namely P-Thresh and MI-Thresh [55]. P-Thresh and MI-Thresh reduced the number of base models available to the meta-learner by evaluating each base model's predictive performance, and the similarity of base model predictions over a recent window of observations, using R^2 and MI as relevancy and diversity metrics. These approaches were dependent on a user defined threshold parameter to determine whether base models should be retained as input to the meta-learner. In Chapter 5, P-Thresh and MI-Thresh were shown to reduce the likelihood of the meta-learner overfitting, with MI-Thresh obtaining better predictive performance than P-Thresh due to its more aggressive culling strategy.

The results presented in Chapter 5 for BOTL, P-Thresh and MI-Thresh indicate that the selection of a relevant yet diverse subset of base models is paramount to preventing the meta-learner from overfitting in online TL. However, existing di-

versity metrics, such as MI, must be continually recalculated in concept drifting data streams due to their dependency on the underlying distribution of data used to obtain base model predictions. Therefore, in Chapter 6, a novel diversity metric was introduced, which estimates the conceptual similarity of base models using the PAs between the subspaces in which each base model was learnt [56]. Since the conceptual similarity between base models is estimated using the data used to train each base model, this diversity metric remains static, despite concept drift, and therefore does not need to be recalculated as the data stream progresses. Identifying the diversity between regression models using conceptual similarity is not only beneficial to online TL, and the use of conceptual similarity as a diversity metric could also be extended to ensemble learning in both offline and online environments. In offline environments, ensemble generalisation can be improved by selecting a diverse subset of base models, whereas in online environments it has been shown that highly diverse base models are desirable during, and immediately after, concept drifts, while ensembles with low diversity are preferable between drifts [58, 60].

Conceptual similarity was used as a diversity metric for parameterised thresholding in Chapter 6, in the form of CS-Thresh [56]. CS-Thresh requires a user defined threshold parameter to determine whether base models should be discarded by the meta-learner. If a base model is found to have a conceptual similarity above the threshold with respect to existing models available to a meta-learner, then that model can be excluded from the meta-learner such that a relevant yet diverse subset of base models could be selected as input to the meta-learner.

Using parameterised conceptual similarity thresholding, as in CS-Thresh, for base model selection obtains a comparable predictive performance to P-Thresh and MI-Thresh, but requires significantly fewer diversity metric calculations due to the static nature of the diversity metric. This makes CS-Thresh more appealing for use in online TL with limited computational resources. However, in Chapter 6, it was found that selecting appropriate threshold parameter values for P-Thresh, MI-Thresh and CS-Thresh is challenging. Considerable domain expertise is required in order to sufficiently reduce the number of models available to the meta-learner to prevent overfitting, while retaining beneficial models to improve predictive performance. The results presented in Chapter 6 showed that the aggressiveness of the culling parameter was dependent on the amount of training data available to the meta-learner, the number of base models available to select from, the complexity of the underlying distribution in the data stream, and the separability of base models for a given diversity metric.

To remove the dependency on domain expertise, parameterless conceptual

clustering, namely CS-Clust [56], was also introduced in Chapter 6. CS-Clust uses the novel conceptual similarity diversity metric and STSC [100] to cluster base models into groups of conceptually similar base models. Since STSC does not require the number of clusters to be defined in advance, CS-Clust is used to identify a relevant yet diverse subset of base models to be used as input to the meta-learner. CS-Clust obtained comparable predictive performances to MI-Thresh and CS-Thresh without requiring domain expertise to define a culling parameter. The ability to select a subset of base models to be used as input to the meta-learner without requiring a user defined parameter may make CS-Clust more applicable to uses of online TL in real-world environments, where the aggressiveness of culling parameters is unlikely to be known in advance. Although CS-Clust increases the computational overhead in comparison to CS-Thresh, due to the use of spectral clustering, it can become less computationally expensive than MI-Thresh when the number of base models is large due to the use of a static diversity metric.

MI-Thresh, CS-Thresh and CS-Clust proved to effectively address the challenge of determining how to combine knowledge transferred in online TL, while reducing the likelihood of the meta-learner overfitting. However, each of these approaches succeeds due to selecting a relevant yet diverse subset of base models to be used as input to the meta-learner. This means that the models that are transferred throughout the framework that are not diverse with respect to the models already available in a receiving domain provide little or no benefit to the meta-learner in the receiving domain. Transferring such models incurs unnecessary communication and computation overheads. Therefore, the challenge of deciding whether to transfer was considered.

8.3 Deciding Whether to Transfer

In Chapter 7, the challenge of determining whether to transfer has been considered. This challenge is paramount to the feasibility of using online TL for real-world applications where communication and computational resources may be limited. Two approaches of determining whether to transfer were considered, namely IdDT and IdCS. These approaches allow the diversity of base models to be considered prior to transfer to determine whether a predictive model should be transferred to a receiving domain. In both approaches, if a similar predictive model is already available in the receiving domain, then the predictive model selected for transfer in the source domain is not transferred. Additionally, if a similar model is found in the source domain that achieves a better predictive performance than the locally learnt model,

over the concept for which it was learnt to represent, then the similar model replaces the locally learnt model. Using these approaches enabled the number of predictive models transferred throughout the framework to be reduced, thereby reducing communication overheads. By reducing the number of models received by each domain, computation overheads were also reduced, due to requiring fewer relevancy and diversity metric calculations for base model selection. From the results presented in Chapter 7, IdDT and IdCS maintain comparable predictive performances in comparison to when all base models are transferred, while on average IdDT reduces the number of models received by each domain by 47.1%, while IdCS achieves a 30% reduction. This indicates that these approaches prevent the transfer of base models that provide little or no beneficial information to receiving domains, which is highly desirable in real-world application of online TL where communication and computational resources are limited.

8.4 Future Work

The research presented in this thesis opens up several avenues of future work. In this section, potential improvements to the approaches proposed in this thesis are discussed, and key areas of future work are identified.

8.4.1 Possible Extensions

Extensions to BOTL and the BOTL variants presented in this thesis revolve around scalability, robustness to noise, and adaptations to the proposed methodologies.

- The approaches presented in this thesis all centre around the use of an OLS meta-learner to combine base models. OLS was chosen as the meta-learner since it is a simple approach to combining base models, and does not require parameterisation. However, OLS is highly susceptible to overfitting, as shown in Chapter 5. A simple extension is to consider the use of alternative machine learning models, which are less prone to overfitting, as meta-learners. For example, models that use regularisation to help reduce the likelihood of overfitting could be considered, such as a Ridge Regressors, or a Lasso Regressors [80]. The use of regularisation may help to prevent the meta-learner from overfitting, however, this was not considered within the scope of this thesis since regularisation parameters must be defined. The selection of regularisation parameters may be challenging for meta-learners in online TL since they may be dependent on the complexity of the underlying distribution observed

in each data stream, and the number of models available to each meta-learner, both of which change over time.

- CS-Thresh, CS-Clust and IdCS depend on the conceptual similarity metric introduced in Chapter 6. The conceptual similarity metric uses local scaling to allow better affinities to be obtained when the locally dense areas of conceptually similar base models are present in the affinity matrix. Although the local scaling parameter chosen in this thesis is supported by experimental results, presented in Section 6.6, and existing research in [42, 100], other scaling techniques could be used that amplify intra-cluster similarities when the volume of conceptually similar base models increases, such as density-aware kernels [42, 101].

- The conceptual similarity metric could also be improved by increasing its robustness to noise. In this thesis, the conceptual similarity of base models is calculated using the PAs between orthonormal representations of the training data used to create each base model. Orthonormal representations are created using SVD to obtain the PCs for each of the subsets of training data for each model. Obtaining orthonormal representations of each subspace in this way can be susceptible to noise, and therefore in this thesis only the first p PCs that capture 99.9% of the variance are retained. However, this approach is still susceptible to outliers in the training data, and therefore other methodologies to obtain estimates of the orthonormal representations could be used that are more robust to noise, such as Laplacian PCA [102] and Robust PCA [90].

8.4.2 Future Research Areas

Scalability is the most challenging issue relating to the use of online TL frameworks where no data rich offline source domain exists. In order to use these approaches in real-world applications, online TL frameworks must be able to manage the fact that each online data stream may grow without limit [95]. This means that the number of concepts transferred throughout the framework may also grow without limit. This is a challenge encountered by most online learning frameworks, which is often overcome through the use of forgetting mechanisms. However, using forgetting mechanisms in online TL may not be appropriate since concepts can reoccur, and concepts historically encountered in one domain may be greatly beneficial to other domains. Therefore, future investigations into how knowledge transfer can be managed in these settings are necessary.

Another important factor to consider with respect to the scalability of online

TL is the management of knowledge transfer when the number of domains in the framework becomes very large. In this setting, the use of peer-to-peer knowledge transfer becomes infeasible. An interesting avenue of future research would be to consider how an online TL framework could be partitioned such that a majority of knowledge transfer is conducted between independent subsets of peers in the framework, while knowledge transfer between each of the subsets of peers in the framework occurs less frequently. Subsets of peers in real-world environments could be identified using factors such a geographical locations, for example, when online TL is used in distributed sensor networks. However, another interesting avenue of research would be to consider how domains in an online TL framework could be partitioned so that frequent knowledge transfer is conducted among similar domains, where domains that are likely to learn knowledge that will be the most beneficial to other domains are contained within the same partition. This would allow knowledge transfer to be conducted in a peer-to-peer fashion among domains that are contained within the same partition. The knowledge found to be the most beneficial among peers within a partition could then be transferred to other partitions via inter-partition knowledge transfer.

In each of these scenarios, the ability to identify diversity in online TL is paramount, and therefore further research into the creation of diversity metrics that remain static within non-stationary environments could provide many benefits to both online TL and more general online learning techniques.

8.5 Summary

To ensure the feasibility of online TL in real-world environments, where all domains are online and an offline data rich source domain does not exist, online TL frameworks must determine what to transfer, how to combine transferred knowledge and whether to transfer knowledge. In this thesis, these challenges have been considered and are addressed by the proposed approaches. Bi-directional knowledge transfer allows the predictions in each domain to be aided by knowledge learnt in other domains, while base model selection strategies have been proposed to prevent overfitting when the number of base models transferred throughout the framework becomes large in comparison to the window of available data. Diversity metrics have also been used to limit knowledge transfer to base models that are likely to be beneficial to a receiving domain, significantly reducing the communication overhead of knowledge transfer, while maintaining comparable predictive performances to when all models are transferred. However, further research is required into methods of

knowledge transfer that are scalable, so that online TL can be used in real-world applications where the number of domains in a framework may be much larger, and where the data streams associated with each domain may grow without limit.

Appendix A

Using Ridge Regressor Base Models

This Appendix present the results for each of the approaches proposed in this thesis where Ridge Regressors (RRs) are used to create base models. Tables A.1–A.4 present the results for BOTL, P-Thresh, MI-Thresh, CS-Thresh, CS-Clust, IdDT and IdCS for the sudden drifting hyperplane, gradual drifting hyperplane, heating simulator and following distance datasets respectively. These results show that the proposed approaches can be used to transfer knowledge in the form of RRs, indicating that the techniques used are model agnostic, and therefore any regression based machine learning model can be used to create base models.

Figures A.1–A.4 also show the average number of models received by domains for MI-Thresh, IdDT, CS-Clust and IdCS for each of these datasets. These results indicate that the decision of whether to transfer base models reduces the communication overheads when RRs are used as base models. Figure A.5 shows the R^2 and PMCC^2 predictive performances, and the average number of models used by the meta-learners in BOTL, MI-Thresh, IdDT, CS-Clust and IdCS, in frameworks with increasing numbers of following distance data streams. These results show that BOTL is susceptible to overfitting when the number of base models available to be meta-learner becomes large in comparison to the window of available data. Additionally, Figure A.6 shows the average number of models received by domains for MI-Thresh, IdDT, CS-Clust and IdCS for frameworks with increasing numbers of following distance data streams. Figure A.7 shows the decrease in the average number of relevancy and diversity metric calculations required by IdDT, CS-Clust and IdCS, in comparison to MI-Thresh, as the number of following distance data streams in the framework increases. This shows that the decision of whether to

(a) SuddenA: sudden drifting hyperplanes with uniform noise.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.755	0.831	1	1	0	0.640	0.653	1	1	0	0.637	0.652	1	1	0
BOTL	***0.906**	0.907	27	32	0	***0.875**	0.876	34	51	0	***0.872**	0.873	15	20	0
P-Thresh	*0.889	0.889	7	9	18767	*0.847	0.848	9	15	22626	*0.818	0.820	5	7	9373
MI-Thresh	0.883	0.884	5	6	35958	0.828	0.829	3	5	47425	0.811	0.812	3	4	16015
CS-Thresh	*0.862	0.862	2	3	2502	*0.760	0.761	2	4	2601	*0.754	0.756	2	4	2111
CS-Clust	*0.891	0.892	5	8	5758	*0.802	0.804	5	8	3962	*0.795	0.797	5	8	1021
IdDT	*0.877	0.877	3	5	14889	*0.826	0.827	3	5	18497	*0.794	0.795	3	4	7533
IdCS	*0.891	0.891	5	8	4038	*0.794	0.796	5	8	1847	*0.785	0.787	4	6	732

(b) SuddenB: sudden drifting hyperplanes with single sensor failure.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.763	0.833	1	1	0	0.533	0.543	1	1	0	0.557	0.565	1	1	0
BOTL	-6e+19	0.545	25	33	0	-3e+19	0.553	35	54	0	-2e+19	0.526	18	24	0
P-Thresh	-6e+20	0.780	7	10	17665	-2e+20	0.777	9	16	23042	***0.818**	0.819	5	7	11304
MI-Thresh	0.883	0.884	5	7	34766	0.810	0.811	4	6	44907	0.807	0.809	3	5	18547
CS-Thresh	*0.861	0.861	2	4	2546	*0.694	0.696	2	4	2363	*0.699	0.702	2	3	2100
CS-Clust	-2e+11	0.883	5	8	5258	-5e+19	0.748	6	10	3862	*0.768	0.770	5	9	1367
IdDT	*0.873	0.874	4	6	17732	***0.823**	0.825	4	6	21925	*0.801	0.802	3	5	9743
IdCS	***0.885**	0.886	5	8	3638	*0.752	0.755	5	8	2056	*0.773	0.775	4	6	988

(c) SuddenC: sudden drifting hyperplanes with intermittent single sensor failure.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.756	0.827	1	1	0	0.618	0.637	1	1	0	0.628	0.648	1	1	0
BOTL	***0.905**	0.906	24	30	0	***0.880**	0.881	37	55	0	*0.879	0.880	17	21	0
P-Thresh	*0.883	0.883	6	9	16165	*0.848	0.849	9	14	24718	*0.835	0.836	5	6	10860
MI-Thresh	0.877	0.877	4	5	29082	0.832	0.833	3	5	48504	0.825	0.826	3	4	17350
CS-Thresh	*0.844	0.844	2	3	2168	*0.761	0.763	2	3	2812	*0.766	0.767	2	3	2353
CS-Clust	*0.885	0.886	5	8	4502	*0.803	0.805	5	7	4855	*0.800	0.802	5	9	1274
IdDT	*0.868	0.868	3	4	10525	*0.830	0.831	3	4	19120	***0.808**	0.809	3	4	7301
IdCS	*0.881	0.881	5	7	2764	*0.796	0.798	5	7	2136	*0.793	0.795	4	6	849

(d) SuddenD: sudden drifting hyperplanes with gradual sensor deterioration.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.754	0.836	1	1	0	0.596	0.611	1	1	0	0.585	0.602	1	1	0
BOTL	-2e+20	0.362	24	30	0	-9e+20	0.352	38	56	0	-1e+21	0.352	18	24	0
P-Thresh	***0.890**	0.891	6	9	16367	-2e+19	0.456	9	16	25261	-6e+17	0.801	5	7	11043
MI-Thresh	0.884	0.884	4	6	29905	0.821	0.823	4	6	50212	0.816	0.817	4	5	18508
CS-Thresh	*0.858	0.858	2	3	2379	*0.726	0.727	2	4	2380	*0.710	0.712	2	3	1769
CS-Clust	***0.890**	0.891	5	8	4586	*0.796	0.798	6	8	4800	*0.775	0.777	5	7	1340
IdDT	*0.878	0.878	3	5	13608	***0.827**	0.828	4	5	24978	***0.794**	0.795	3	5	9333
IdCS	*0.888	0.889	5	8	3093	*0.786	0.788	5	7	2343	*0.760	0.762	4	6	927

Table A.1: Sudden Drifting Hyperplane with RR base models: R^2 and PMCC^2 predictive performance, the average number of base models used by the meta-learner ($\lvert\mathcal{M}'\rvert$), the maximum number of base models used by the meta-learner ($\lceil\mathcal{M}'\rceil$), and the average number of relevancy and diversity metric calculations required to compare and evaluate base models (M.Calcs.) for meta-learners in BOTL for variants of the sudden drifting hyperplane datasets when transferring RR base models between 5 data streams. Improved predictive performances with statistical t-test values $p < 0.01$ compared to the underlying CDD, while requiring fewer relevancy and diversity metric calculations than MI-Thresh are indicated with *. Of these, bold type indicates the approach with highest performance.

(a) GradualA: gradual drifting hyperplanes with uniform noise.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.728	0.798	1	1	0	0.520	0.529	1	1	0	0.552	0.564	1	1	0
BOTL	***0.902**	0.903	29	35	0	***0.893**	0.894	35	50	0	***0.889**	0.890	19	24	0
P-Thresh	*0.885	0.886	10	15	25299	*0.854	0.855	12	20	26892	*0.842	0.843	7	11	14717
MI-Thresh	*0.873	0.874	5	8	70628	0.834	0.835	4	6	75363	0.825	0.825	4	6	34538
CS-Thresh	*0.837	0.838	3	5	3179	*0.725	0.726	3	5	2993	*0.731	0.732	2	4	2485
CS-Clust	*0.879	0.879	6	8	10024	*0.773	0.774	6	8	3688	*0.781	0.782	5	7	1427
IdDT	*0.860	0.861	4	6	24906	*0.832	0.833	4	6	32772	*0.788	0.789	3	5	13266
IdCS	*0.875	0.876	5	8	5768	*0.762	0.763	5	8	1791	*0.760	0.761	4	6	829

(b) GradualB: gradual drifting hyperplanes with single sensor failure.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.715	0.790	1	1	0	0.559	0.571	1	1	0	0.626	0.650	1	1	0
BOTL	-6e+17	0.361	30	38	0	-2e+17	0.358	37	53	0	-2e+18	0.356	19	25	0
P-Thresh	-7e+18	0.703	9	14	26561	-2e+18	0.680	10	19	28873	***0.843**	0.843	6	9	14123
MI-Thresh	0.858	0.858	5	7	60837	0.818	0.818	4	6	62251	0.835	0.836	4	5	26405
CS-Thresh	*0.829	0.830	2	4	3278	*0.749	0.750	2	4	2785	*0.780	0.781	2	3	2448
CS-Clust	***0.870**	0.871	6	7	11963	-4e+17	0.777	6	8	4651	-7e+16	0.797	5	8	1475
IdDT	*0.852	0.853	4	6	28281	***0.815**	0.815	4	6	26430	*0.810	0.811	3	5	11597
IdCS	-3e+18	0.862	5	7	7234	-6e+18	0.773	5	7	2275	*0.806	0.807	4	6	972

(c) GradualC: gradual drifting hyperplanes with intermittent single sensor failure.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.706	0.777	1	1	0	0.504	0.518	1	1	0	0.508	0.531	1	1	0
BOTL	***0.899**	0.900	30	39	0	***0.876**	0.877	34	49	0	***0.870**	0.871	17	22	0
P-Thresh	*0.882	0.882	10	13	27767	*0.846	0.847	10	16	26010	*0.820	0.821	6	9	12599
MI-Thresh	0.869	0.869	6	8	74580	0.824	0.825	4	6	61700	0.809	0.809	4	5	26037
CS-Thresh	*0.827	0.827	3	5	3758	*0.703	0.704	2	3	2707	*0.702	0.703	2	3	2238
CS-Clust	*0.874	0.875	6	8	13051	*0.747	0.749	5	7	3956	*0.744	0.745	5	7	1250
IdDT	*0.858	0.858	5	6	33161	*0.817	0.817	4	6	26051	*0.788	0.789	4	5	12275
IdCS	*0.870	0.870	5	7	8235	*0.743	0.745	5	7	1918	*0.715	0.716	4	5	784

(d) GradualD: gradual drifting hyperplanes with gradual sensor deterioration.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.723	0.793	1	1	0	0.620	0.637	1	1	0	0.631	0.648	1	1	0
BOTL	-1e+20	0.177	32	42	0	-3e+20	0.174	35	51	0	-7e+20	0.175	18	23	0
P-Thresh	***0.892**	0.892	9	12	30119	-4e+18	0.702	9	15	27133	***0.858**	0.859	5	8	13185
MI-Thresh	0.882	0.883	5	7	63323	0.849	0.849	4	5	53836	0.848	0.849	4	5	23299
CS-Thresh	*0.852	0.853	2	4	3954	*0.784	0.785	2	3	2655	*0.785	0.786	2	3	2366
CS-Clust	*0.884	0.884	6	8	15127	*0.832	0.833	6	8	3959	*0.828	0.829	5	7	1259
IdDT	*0.876	0.876	5	7	36542	***0.843**	0.844	4	5	23891	*0.832	0.833	3	5	11171
IdCS	*0.879	0.879	5	8	9873	*0.825	0.826	5	7	1828	*0.823	0.824	4	6	862

Table A.2: Gradual Drifting Hyperplane with RR base models: R^2 and PMCC^2 predictive performance, the average number of base models used by the meta-learner ($|\mathcal{M}'|$), the maximum number of base models used by the meta-learner ($\lceil\mathcal{M}'\rceil$), and the average number of relevancy and diversity metric calculations required to compare and evaluate base models (M.Calcs.) for meta-learners in BOTL for variants of the gradual drifting hyperplane datasets when transferring RR base models between 5 data streams. Improved predictive performances with statistical t-test values $p < 0.01$ compared to the underlying CDD, while requiring fewer relevancy and diversity metric calculations than MI-Thresh are indicated with *. Of these, bold type indicates the approach with highest performance.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.625	0.637	1	1	0	0.594	0.607	1	1	0	0.585	0.602	1	1	0
BOTL	***0.742**	0.748	9	16	0	***0.718**	0.725	10	18	0	***0.723**	0.730	9	15	0
P-Thresh	*0.733	0.739	6	11	1103	*0.702	0.710	7	12	1310	*0.717	0.724	6	10	1369
MI-Thresh	0.728	0.734	6	9	3761	0.695	0.703	6	11	5654	0.713	0.720	5	8	4340
CS-Thresh	*0.704	0.710	3	4	607	*0.670	0.678	3	4	764	*0.689	0.697	3	4	580
CS-Clust	*0.718	0.724	3	5	806	*0.699	0.706	3	5	1069	*0.707	0.714	3	6	1236
IdDT	*0.723	0.730	5	8	2871	*0.693	0.701	5	10	4484	*0.707	0.714	4	7	3428
IdCS	*0.714	0.720	3	5	665	*0.694	0.701	3	5	877	*0.699	0.706	3	4	960

Table A.3: Heating Simulator with RR base models: R^2 and PMCC^2 predictive performance, the average number of base models used by the meta-learner ($|\mathcal{M}'|$), the maximum number of base models used by the meta-learner ($\lceil\mathcal{M}'\rceil$), and the average number of relevancy and diversity metric calculations required to compare and evaluate base models (M.Calcs.) for meta-learners in BOTL for the smart home heating simulator dataset when transferring RR base models between 5 data streams. Improved predictive performances with statistical t-test values $p < 0.01$ compared to the underlying CDD, while requiring fewer relevancy and diversity metric calculations than MI-Thresh are indicated with *. Of these, bold type indicates the approach with highest performance.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.437	0.469	1	1	0	0.115	0.330	1	1	0	0.195	0.380	1	1	0
BOTL	*0.521	0.550	2	5	0	-2e+17	0.268	14	28	0	*0.421	0.572	8	16	0
P-Thresh	*0.515	0.544	2	4	100	*0.506	0.558	4	10	1415	***0.510**	0.557	3	7	767
MI-Thresh	0.524	0.551	2	3	118	0.503	0.555	3	4	2043	0.488	0.545	2	4	983
CS-Thresh	*0.498	0.529	2	3	78	*0.351	0.445	2	4	337	*0.445	0.511	2	4	284
CS-Clust	*0.541	0.565	2	4	201	***0.578**	0.621	4	9	1478	*0.499	0.575	3	6	753
IdDT	*0.520	0.548	2	3	156	*0.501	0.546	2	4	1550	*0.462	0.521	2	4	832
IdCS	***0.523**	0.549	2	4	189	*0.550	0.598	4	7	951	*0.507	0.567	3	5	591

Table A.4: Following Distance with RR base models: R^2 and PMCC^2 predictive performance, the average number of base models used by the meta-learner ($|\mathcal{M}'|$), the maximum number of base models used by the meta-learner ($\lceil\mathcal{M}'\rceil$), and the average number of relevancy and diversity metric calculations required to compare and evaluate base models (M.Calcs.) for meta-learners in BOTL for the following distance dataset when transferring RR base models between 7 data streams. Improved predictive performances with statistical t-test values $p < 0.01$ compared to the underlying CDD, while requiring fewer relevancy and diversity metric calculations than MI-Thresh are indicated with *. Of these, bold type indicates the approach with highest performance.

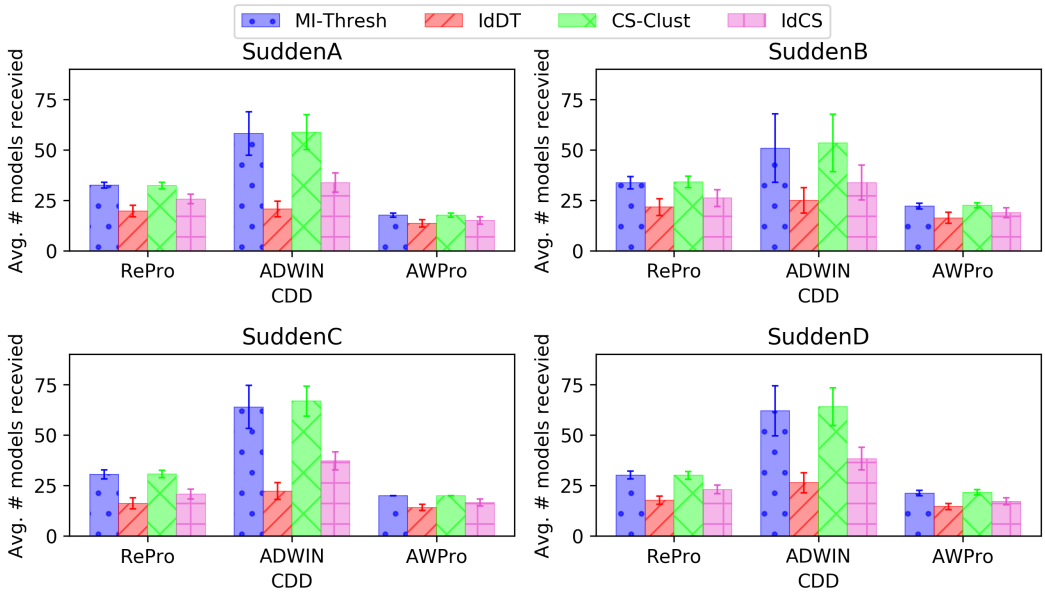


Figure A.1: Sudden Drifting Hyperplane with RR base models: The average number of RR models received per domain for MI-Thresh, IdDT, CS-Clust and IdCS when using RePro, ADWIN and AWPro as the underlying CDDs for 5 drifting hyperplane data streams.

transfer also helps to reduce computational overheads in online TL frameworks, and highlights the benefits of using a static diversity metric. Finally Table A.5 displays a summary of the results obtained by IdDT in comparison to MI-Thresh, and IdCS in comparison to CS-Clust, for the sudden drifting hyperplane, gradual drifting hyperplane, heating simulator and following distance datasets. These results show that both IdDT and IdCS reduce communication and computation overheads while maintaining comparable predictive performances to when all base models are transferred.

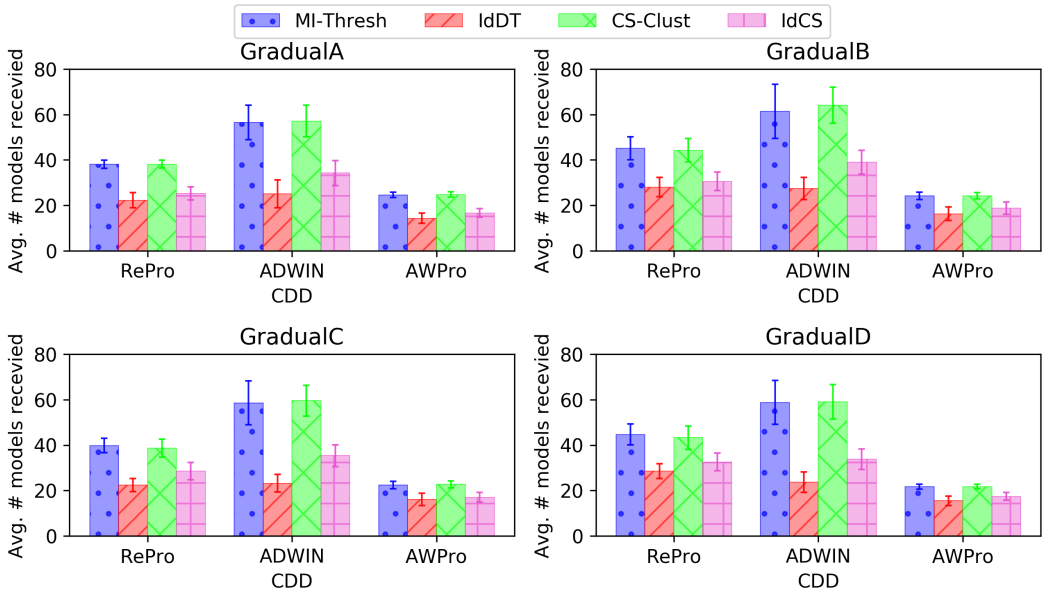


Figure A.2: Gradual Drifting Hyperplane with RR base models: The average number of RR models received per domain for MI-Thresh, IdDT, CS-Clust and IdCS when using RePro, ADWIN and AWPro as the underlying CDDs for 5 drifting hyperplane data streams.

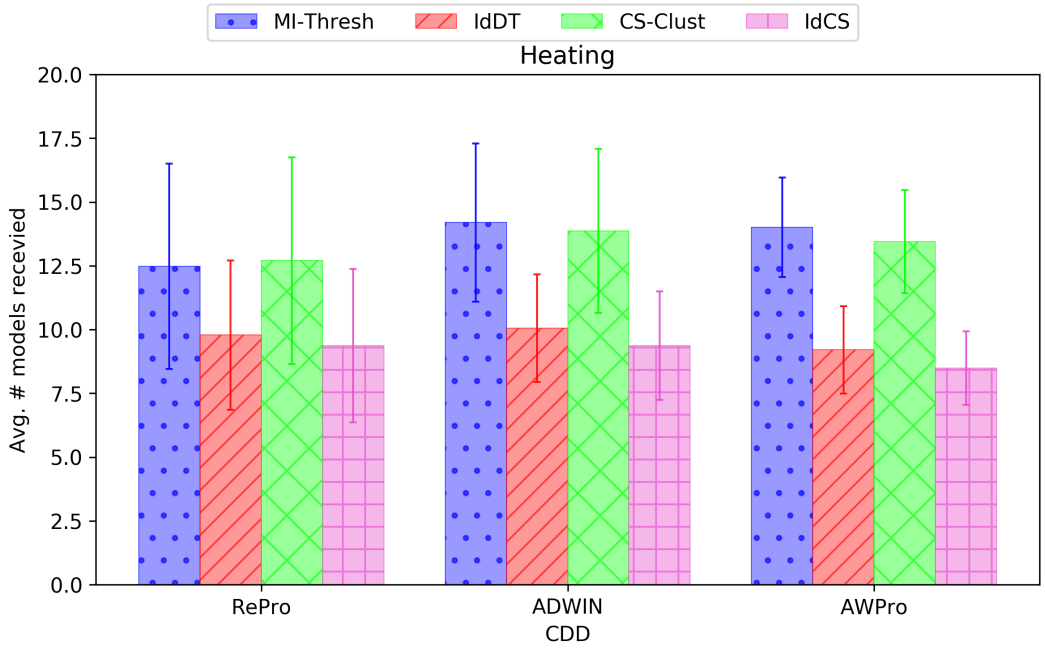


Figure A.3: Heating Simulator with RR base models: The average number of RR models received per domain for MI-Thresh, IdDT, CS-Clust and IdCS when using RePro, ADWIN and AWPro as the underlying CDDs for 5 heating simulator data streams.

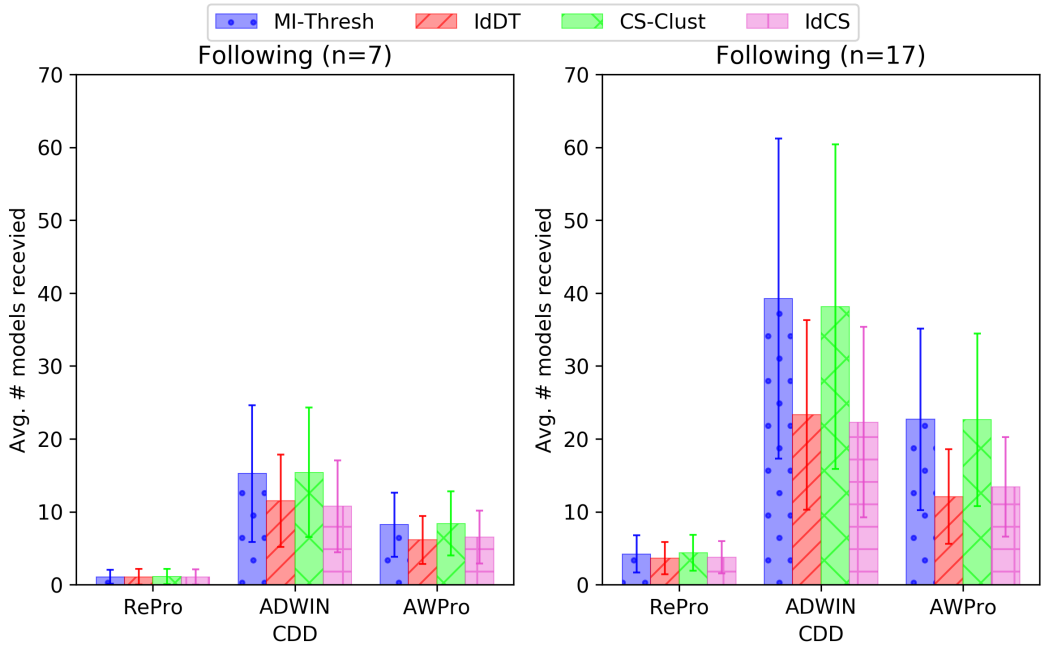


Figure A.4: Following Distance with RR base models: The average number of RR models received per domain for MI-Thresh, IdDT, CS-Clust and IdCS when using RePro, ADWIN and AWPro as the underlying CDDs for 7 and 17 following distance data streams.

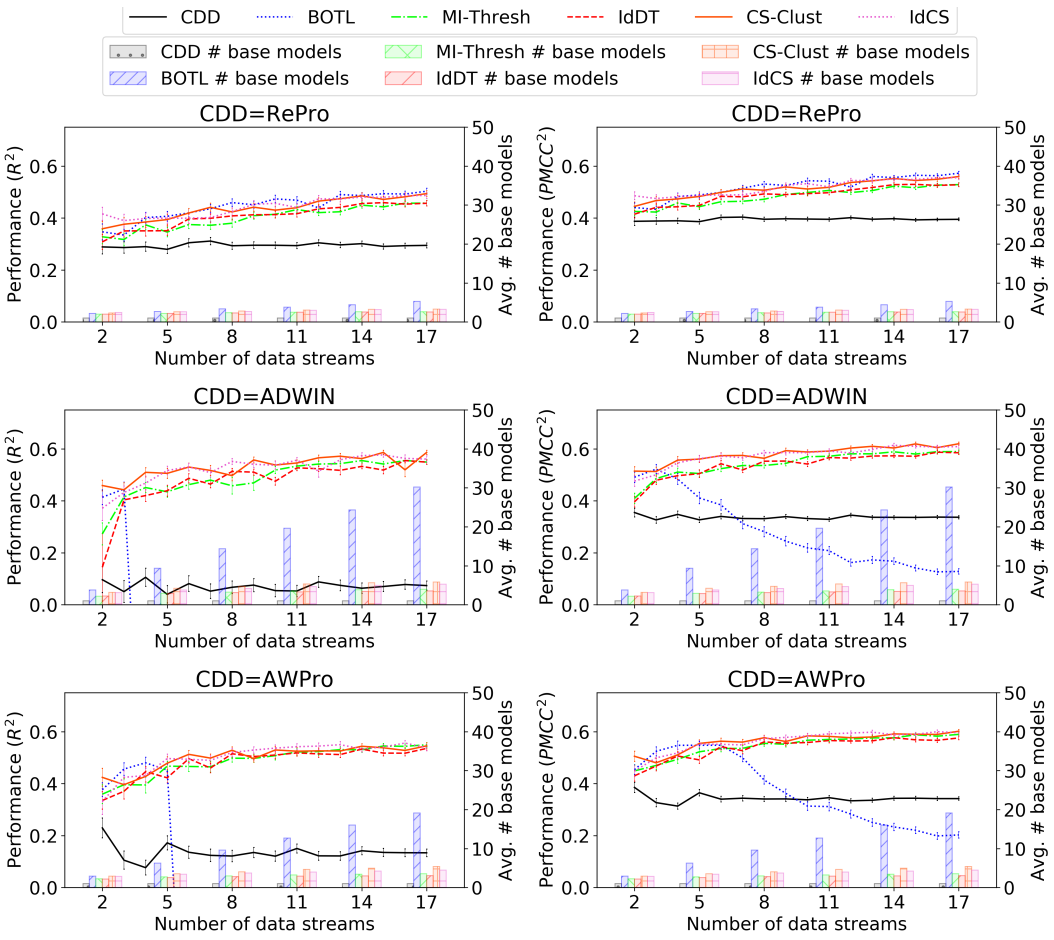


Figure A.5: Following Distance with RR base models: R^2 and PMCC^2 predictive performance, and number of RR base models used by BOTL meta-learners, for the underlying CDD, BOTL, MI-Thresh, IdDT, CS-Clust and IdCS with increasing numbers of following distance data streams.

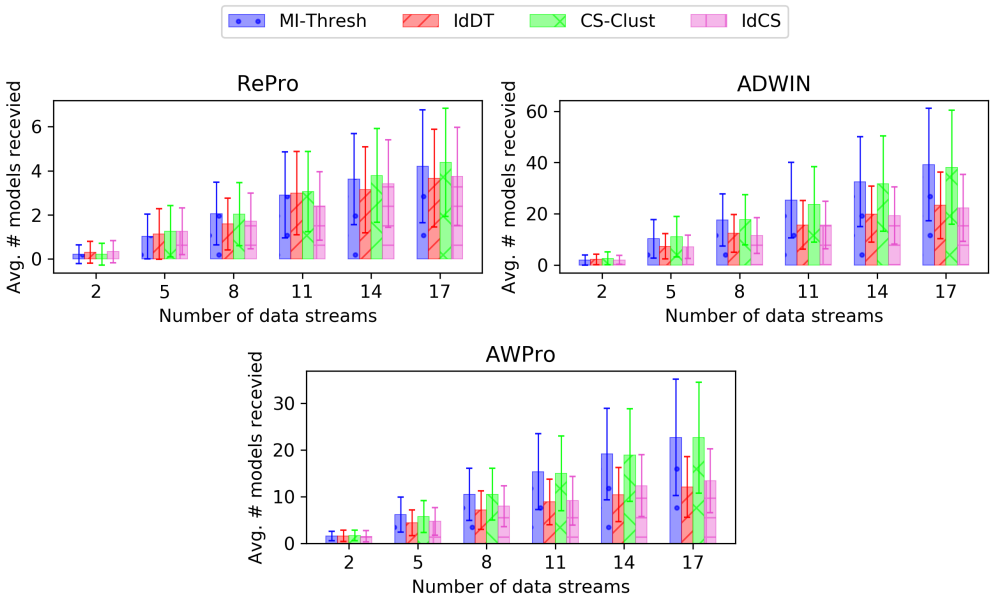


Figure A.6: Following Distance with RR base models: The average number of RR models received per domain for MI-Thresh, IdDT, CS-Clust and IdCS for increasing numbers of following distance data streams, using RePro, ADWIN and AWPro as the underlying CDDs.

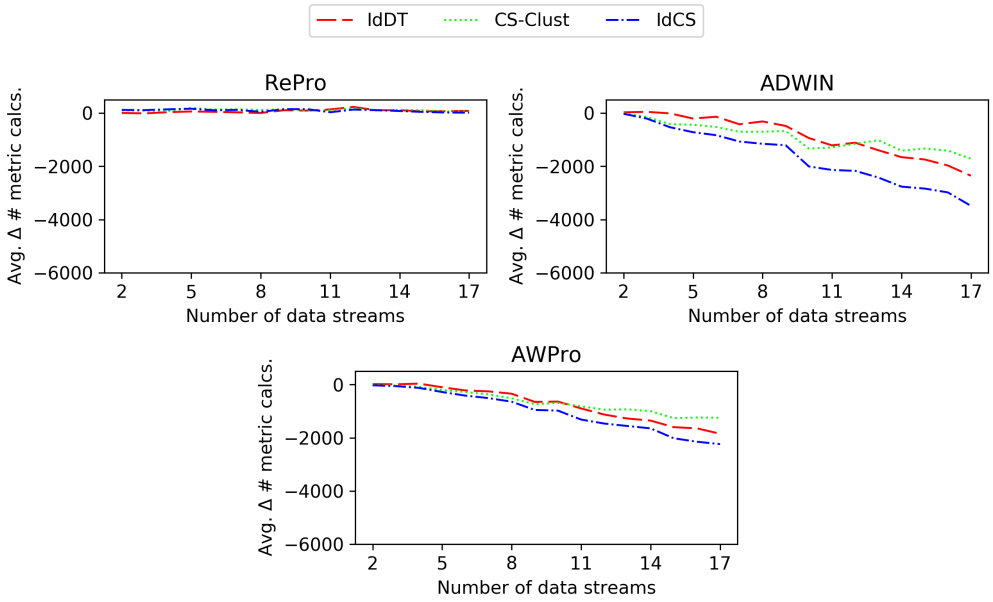


Figure A.7: Following Distance with RR base models: Change in number of relevancy and diversity metric calculations required to compare and evaluate RR base models for IdDT, CS-Clust and IdCS in comparison to MI-Thresh for increasing numbers of following distance data streams.

Dataset		RePro ΔComm	RePro ΔComp	RePro ΔR^2	RePro Δ_{PMCC^2}	ADWIN ΔComm	ADWIN ΔComp	ADWIN ΔR^2	ADWIN Δ_{PMCC^2}	AWPro ΔComm	AWPro ΔComp	AWPro ΔR^2	AWPro Δ_{PMCC^2}
SuddenA	IdDT	-39.5%	-58.6%	-0.007	-0.007	-64.3%	-61.0%	-0.003	-0.003	-22.7%	-53.0%	-0.017	-0.017
	IdCS	-20.6%	-29.9%	-0.001	-0.001	-42.5%	-53.4%	-0.008	-0.008	-14.8%	-28.3%	-0.010	-0.010
SuddenB	IdDT	-35.9%	-49.0%	-0.010	-0.010	-50.8%	-51.2%	+0.013	+0.013	-26.7%	-47.5%	-0.006	-0.006
	IdCS	-23.2%	-30.8%	+0.885	+0.003	-36.8%	-46.7%	+0.752	+0.007	-16.5%	-27.7%	+0.005	+0.005
SuddenC	IdDT	-47.1%	-63.8%	-0.008	-0.008	-65.1%	-60.6%	-0.001	-0.001	-29.5%	-57.9%	-0.017	-0.017
	IdCS	-32.3%	-38.6%	-0.005	-0.005	-44.2%	-56.0%	-0.007	-0.007	-17.0%	-33.3%	-0.007	-0.007
SuddenD	IdDT	-41.3%	-54.5%	-0.006	-0.006	-57.4%	-50.3%	+0.005	+0.005	-31.4%	-49.6%	-0.022	-0.022
	IdCS	-23.2%	-32.6%	-0.002	-0.002	-40.1%	-51.2%	-0.009	-0.009	-20.7%	-30.8%	-0.015	-0.015
GradualA	IdDT	-41.6%	-64.7%	-0.013	-0.013	-55.7%	-56.5%	-0.002	-0.002	-41.8%	-61.6%	-0.037	-0.036
	IdCS	-33.6%	-42.5%	-0.004	-0.004	-40.1%	-51.4%	-0.011	-0.011	-32.9%	-41.9%	-0.021	-0.021
GradualB	IdDT	-37.8%	-53.5%	-0.006	-0.006	-55.3%	-57.5%	-0.003	-0.003	-32.3%	-56.1%	-0.025	-0.025
	IdCS	-31.0%	-39.5%	-0.870	-0.009	-39.2%	-51.1%	+0.000	-0.004	-22.5%	-34.1%	+0.806	+0.010
GradualC	IdDT	-43.6%	-55.5%	-0.011	-0.011	-60.4%	-57.8%	-0.007	-0.007	-28.4%	-52.9%	-0.020	-0.020
	IdCS	-26.4%	-36.9%	-0.005	-0.005	-40.6%	-51.5%	-0.004	-0.004	-25.1%	-37.3%	-0.029	-0.029
GradualD	IdDT	-36.1%	-42.3%	-0.007	-0.007	-59.7%	-55.6%	-0.005	-0.005	-28.2%	-52.1%	-0.016	-0.016
	IdCS	-24.8%	-34.7%	-0.005	-0.005	-42.7%	-53.8%	-0.007	-0.007	-19.9%	-31.5%	-0.006	-0.006
Heating	IdDT	-21.6%	-23.7%	-0.005	-0.005	-29.2%	-20.7%	-0.002	-0.002	-34.3%	-21.0%	-0.006	-0.005
	IdCS	-26.2%	-17.5%	-0.003	-0.003	-32.4%	-18.0%	-0.005	-0.005	-36.8%	-22.4%	-0.008	-0.008
Following (n=7)	IdDT	0.0%	31.9%	-0.003	-0.003	-24.5%	-24.1%	-0.002	-0.009	-25.5%	-15.3%	-0.026	-0.024
	IdCS	-6.8%	-5.9%	-0.018	-0.016	-30.3%	-35.7%	-0.028	-0.022	-22.4%	-21.6%	+0.008	-0.008
Following (n=17)	IdDT	-13.0%	12.1%	-0.001	-0.001	-40.6%	-41.2%	-0.000	-0.004	-46.7%	-51.6%	-0.014	-0.014
	IdCS	-14.4%	-6.9%	-0.007	-0.005	-41.6%	-44.2%	-0.026	-0.011	-40.7%	-42.4%	-0.007	-0.006

Table A.5: Summary of comparisons between MI-Thresh and IdDT, and CS-Clust and IdCS, when RRs are used as base models, showing the average reduction in number of models received (%), denoted as Δ Comm., the average reduction in performance and diversity metric calculations (%), denoted as Δ Comp., and the average difference between R^2 and PMCC^2 predictive performances.

Appendix B

Using Support Vector Regressor and Ridge Regressor Base Models

This Appendix present the results for each of the approaches proposed in this thesis when approximately 50% of the domains in the framework create base models using SVRs, while the remaining domains create base models using RRs. Tables B.1–B.4 present the results for BOTL, P-Thresh, MI-Thresh, CS-Thresh, CS-Clust, IdDT and IdCS for the sudden drifting hyperplane, gradual drifting hyperplane, heating simulator and following distance datasets respectively. These results show that the proposed approaches can be used to transfer knowledge when domains are not able to use the same machine learning algorithm to create base models, which may occur when domains have differing computational availabilities.

Figures B.1–B.4 also show the average number of models received by domains for MI-Thresh, IdDT, CS-Clust and IdCS for each of these datasets. These results indicate that the decision of whether to transfer base models reduces the communication overheads when a mixture of model types are used to create base models. Figure B.5 shows the R^2 and PMCC^2 predictive performances, and the average number of models used by the meta-learners in BOTL, MI-Thresh, IdDT, CS-Clust and IdCS, in frameworks with increasing numbers of following distance data streams. These results show that BOTL is susceptible to overfitting when the number of base models available to be meta-learner becomes large in comparison to the window of available data. Additionally, Figure B.6 shows the average number of models received by domains for MI-Thresh, IdDT, CS-Clust and IdCS for frameworks with increasing numbers of following distance data streams. Figure B.7 shows

(a) SuddenA: sudden drifting hyperplanes with uniform noise.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.831	0.862	1	1	0	0.765	0.772	1	1	0	0.755	0.764	1	1	0
BOTL	*0.855	0.862	25	33	0	*0.839	0.847	38	57	0	***0.876**	0.878	17	22	0
P-Thresh	*0.897	0.897	6	9	17418	*0.867	0.869	9	15	25299	*0.859	0.860	5	7	10338
MI-Thresh	0.896	0.896	4	6	32006	0.869	0.870	3	5	49479	0.855	0.856	3	5	16237
CS-Thresh	*0.880	0.880	2	4	2415	*0.834	0.836	2	4	2751	*0.822	0.824	2	4	2019
CS-Clust	***0.900**	0.901	5	8	5761	*0.855	0.857	6	8	4882	*0.849	0.850	5	9	1196
IdDT	*0.893	0.893	3	5	12750	***0.869**	0.870	3	5	21363	*0.851	0.853	3	4	7653
IdCS	*0.899	0.899	5	7	3679	*0.854	0.856	5	7	2485	*0.847	0.849	4	6	850

(b) SuddenB: sudden drifting hyperplanes with single sensor failure.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.835	0.863	1	1	0	0.710	0.718	1	1	0	0.707	0.715	1	1	0
BOTL	-9e+21	0.516	26	36	0	-6e+22	0.520	39	59	0	-4e+22	0.529	19	27	0
P-Thresh	-2e+20	0.864	6	10	18292	-1e+21	0.746	8	15	25421	***0.850**	0.852	5	7	11964
MI-Thresh	0.898	0.899	4	6	29612	0.858	0.860	4	5	47037	0.849	0.851	3	5	17681
CS-Thresh	*0.881	0.881	2	4	2702	*0.795	0.797	2	4	2509	*0.787	0.789	2	3	2043
CS-Clust	-2e+19	0.889	5	8	5362	-2e+17	0.815	6	9	4812	-6e+18	0.821	5	8	1529
IdDT	*0.895	0.895	3	5	15366	***0.858**	0.859	3	5	21450	*0.846	0.847	3	4	9132
IdCS	***0.900**	0.900	5	7	3758	-7e+14	0.823	5	8	2689	*0.838	0.840	4	6	1113

(c) SuddenC: sudden drifting hyperplanes with intermittent single sensor failure.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.826	0.856	1	1	0	0.755	0.765	1	1	0	0.757	0.766	1	1	0
BOTL	*0.852	0.860	24	32	0	*0.832	0.841	40	57	0	***0.879**	0.881	18	21	0
P-Thresh	*0.896	0.897	6	9	16794	***0.868**	0.869	8	14	26045	*0.866	0.867	4	6	11093
MI-Thresh	0.894	0.894	4	5	27819	0.866	0.867	3	4	45940	0.863	0.864	3	4	16500
CS-Thresh	*0.869	0.869	2	3	2348	*0.819	0.821	2	3	2265	*0.825	0.827	2	3	2075
CS-Clust	***0.901**	0.901	5	7	4950	*0.854	0.855	6	8	5188	*0.856	0.858	5	9	1337
IdDT	*0.893	0.893	3	4	10451	***0.868**	0.869	3	4	15917	*0.861	0.863	2	3	6925
IdCS	*0.899	0.899	5	6	2912	*0.850	0.852	5	7	2358	*0.852	0.853	4	6	931

(d) SuddenD: sudden drifting hyperplanes with gradual sensor deterioration.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.822	0.856	1	1	0	0.745	0.752	1	1	0	0.721	0.730	1	1	0
BOTL	-2e+22	0.342	26	34	0	-2e+22	0.340	39	58	0	-1e+22	0.353	18	23	0
P-Thresh	***0.899**	0.900	6	9	17874	-4e+20	0.512	8	15	26339	***0.854**	0.856	4	7	10988
MI-Thresh	0.896	0.897	4	6	28329	0.860	0.862	4	6	46369	0.851	0.852	3	5	16280
CS-Thresh	*0.875	0.875	2	3	2428	*0.815	0.817	2	3	2589	*0.800	0.801	2	3	2140
CS-Clust	*0.897	0.898	5	8	5241	*0.849	0.851	6	8	5134	*0.836	0.837	5	9	1385
IdDT	*0.891	0.892	3	5	14055	***0.862**	0.864	3	5	21502	*0.848	0.849	3	4	8243
IdCS	*0.897	0.897	5	7	3626	*0.849	0.851	5	7	2607	*0.837	0.839	4	6	970

Table B.1: Sudden Drifting Hyperplane with SVR and RR base models: R^2 and PMCC^2 predictive performance, the average number of base models used by the meta-learner ($|\mathcal{M}'|$), the maximum number of base models used by the meta-learner ($\lceil\mathcal{M}'\rceil$), and the average number of relevancy and diversity metric calculations required to compare and evaluate base models (M.Calcs.) for meta-learners in BOTL for variants of the sudden drifting hyperplane datasets when transferring SVR and RR base models between 5 data streams. Improved predictive performances with statistical t-test values $p < 0.01$ compared to the underlying CDD, while requiring fewer relevancy and diversity metric calculations than MI-Thresh are indicated with *. Of these, bold type indicates the approach with highest performance.

(a) GradualA: gradual drifting hyperplanes with uniform noise.

	RePro					ADWIN					AWPro				
	R^2	$PMCC^2$	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	$PMCC^2$	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	$PMCC^2$	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.801	0.830	1	1	0	0.675	0.683	1	1	0	0.697	0.705	1	1	0
BOTL	0.791	0.810	30	37	0	*0.824	0.835	37	53	0	***0.883**	0.885	20	26	0
P-Thresh	*0.888	0.888	9	14	26412	***0.867**	0.868	12	20	28977	*0.869	0.869	7	10	15015
MI-Thresh	0.886	0.886	5	7	61134	0.864	0.864	4	6	73059	0.863	0.864	4	6	31782
CS-Thresh	*0.857	0.857	3	4	3206	*0.796	0.797	3	5	3264	*0.808	0.809	3	4	2631
CS-Clust	***0.893**	0.894	5	8	11089	*0.834	0.835	5	8	4626	*0.843	0.844	5	8	1534
IdDT	*0.881	0.882	4	6	21966	***0.867**	0.867	4	6	26264	*0.851	0.851	3	5	11643
IdCS	*0.892	0.892	5	7	6714	*0.833	0.834	5	7	2134	*0.843	0.844	4	6	951

(b) GradualB: gradual drifting hyperplanes with single sensor failure.

	RePro					ADWIN					AWPro				
	R^2	$PMCC^2$	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	$PMCC^2$	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	$PMCC^2$	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.801	0.831	1	1	0	0.677	0.692	1	1	0	0.677	0.699	1	1	0
BOTL	-9e+20	0.327	30	40	0	-2e+21	0.331	39	56	0	-1e+21	0.350	20	28	0
P-Thresh	-1e+18	0.822	8	14	27037	-6e+18	0.707	9	17	30440	***0.847**	0.847	5	9	15183
MI-Thresh	0.877	0.877	4	6	50643	0.841	0.842	4	6	59678	0.841	0.842	3	5	25919
CS-Thresh	*0.859	0.859	2	4	3518	*0.797	0.798	2	4	3193	*0.796	0.797	2	4	2342
CS-Clust	-3e+16	0.873	6	8	12544	-1e+17	0.827	6	8	5093	-4e+17	0.808	5	7	1630
IdDT	***0.875**	0.875	3	6	22210	***0.846**	0.847	3	6	24643	*0.831	0.832	3	4	10910
IdCS	-1e+14	0.868	5	8	7885	-6e+17	0.806	5	7	2438	*0.827	0.828	4	6	1003

(c) GradualC: gradual drifting hyperplanes with intermittent single sensor failure.

	RePro					ADWIN					AWPro				
	R^2	$PMCC^2$	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	$PMCC^2$	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	$PMCC^2$	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.787	0.816	1	1	0	0.660	0.669	1	1	0	0.654	0.674	1	1	0
BOTL	0.782	0.804	30	39	0	*0.807	0.821	37	55	0	***0.869**	0.871	19	26	0
P-Thresh	***0.887**	0.887	9	13	26902	***0.862**	0.863	10	17	28525	*0.852	0.853	6	8	14505
MI-Thresh	0.883	0.884	5	7	59677	0.861	0.862	4	6	68206	0.849	0.849	4	5	27610
CS-Thresh	*0.850	0.850	2	4	3563	*0.780	0.781	2	4	3056	*0.772	0.773	2	4	2431
CS-Clust	*0.886	0.887	5	7	12769	*0.820	0.821	6	8	4666	*0.814	0.815	5	8	1493
IdDT	*0.879	0.879	4	6	27589	*0.857	0.858	4	5	28660	*0.830	0.830	3	5	12241
IdCS	*0.885	0.885	5	8	8480	*0.820	0.821	5	8	2364	*0.812	0.813	4	7	994

(d) GradualD: gradual drifting hyperplanes with gradual sensor deterioration.

	RePro					ADWIN					AWPro				
	R^2	$PMCC^2$	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	$PMCC^2$	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	$PMCC^2$	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.796	0.825	1	1	0	0.541	0.614	1	1	0	0.537	0.619	1	1	0
BOTL	-4e+22	0.152	32	42	0	-3e+22	0.150	30	53	0	-2e+22	0.146	17	28	0
P-Thresh	-2e+17	0.888	8	11	29709	-1e+18	0.780	7	15	22753	***0.836**	0.836	5	9	13378
MI-Thresh	0.892	0.893	5	7	53032	0.858	0.858	4	5	45607	0.847	0.848	3	5	22295
CS-Thresh	*0.870	0.870	2	4	3714	*0.773	0.774	2	4	2950	*0.773	0.774	2	3	2342
CS-Clust	***0.894**	0.895	6	8	15360	*0.791	0.793	5	9	2927	*0.802	0.803	5	7	1307
IdDT	*0.888	0.889	4	6	27570	***0.829**	0.830	3	5	20614	*0.807	0.808	3	5	11711
IdCS	*0.892	0.893	5	8	10263	*0.799	0.801	5	7	1682	*0.812	0.813	4	7	910

Table B.2: Gradual Drifting Hyperplane with SVR and RR base models: R^2 and $PMCC^2$ predictive performance, the average number of base models used by the meta-learner ($|\mathcal{M}'|$), the maximum number of base models used by the meta-learner ($\lceil\mathcal{M}'\rceil$), and the average number of relevancy and diversity metric calculations required to compare and evaluate base models (M.Calcs.) for meta-learners in BOTL for variants of the gradual drifting hyperplane datasets when transferring SVR and RR base models between 5 data streams. Improved predictive performances with statistical t-test values $p < 0.01$ compared to the underlying CDD, while requiring fewer relevancy and diversity metric calculations than MI-Thresh are indicated with *. Of these, bold type indicates the approach with highest performance.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.629	0.643	1	1	0	0.617	0.634	1	1	0	0.609	0.628	1	1	0
BOTL	***0.741**	0.748	9	15	0	***0.719**	0.727	9	18	0	***0.729**	0.737	10	15	0
P-Thresh	*0.730	0.737	6	11	1205	*0.709	0.716	6	12	1262	*0.721	0.728	6	10	1399
MI-Thresh	0.721	0.728	4	8	3542	0.706	0.714	5	10	4681	0.712	0.720	4	6	4129
CS-Thresh	*0.702	0.709	3	4	664	*0.690	0.698	3	4	764	*0.691	0.698	3	4	714
CS-Clust	*0.721	0.727	3	5	891	*0.705	0.713	3	5	1087	*0.712	0.719	3	6	1164
IdDT	*0.720	0.726	4	7	2323	*0.702	0.710	4	8	2485	*0.707	0.714	4	6	3230
IdCS	*0.715	0.722	3	4	744	*0.700	0.708	3	5	852	*0.707	0.714	3	4	874

Table B.3: Heating Simulator with SVR and RR base models: R^2 and PMCC^2 predictive performance, the average number of base models used by the meta-learner ($|\mathcal{M}'|$), the maximum number of base models used by the meta-learner ($\lceil\mathcal{M}'\rceil$), and the average number of relevancy and diversity metric calculations required to compare and evaluate base models (M.Calcs.) for meta-learners in BOTL for the smart home heating simulator dataset when transferring SVR and RR base models between 5 data streams. Improved predictive performances with statistical t-test values $p < 0.01$ compared to the underlying CDD, while requiring fewer relevancy and diversity metric calculations than MI-Thresh are indicated with *. Of these, bold type indicates the approach with highest performance.

	RePro					ADWIN					AWPro				
	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.	R^2	PMCC^2	$\lvert\mathcal{M}'\rvert$	$\lceil\mathcal{M}'\rceil$	M.Calcs.
CDD	0.497	0.521	1	1	0	0.294	0.429	1	1	0	0.290	0.452	1	1	0
BOTL	*0.628	0.656	4	10	0	-2e+16	0.341	12	26	0	-4e+13	0.584	8	19	0
P-Thresh	***0.640**	0.663	3	7	380	0.467	0.644	5	11	1206	*0.627	0.653	4	8	761
MI-Thresh	0.622	0.645	2	3	570	0.597	0.627	3	4	2019	0.625	0.649	2	4	1107
CS-Thresh	*0.638	0.660	2	5	236	*0.545	0.588	2	5	325	*0.560	0.599	2	4	293
CS-Clust	*0.631	0.655	3	4	562	***0.644**	0.672	4	9	1308	***0.633**	0.663	3	7	796
IdDT	*0.626	0.649	2	3	635	*0.588	0.620	2	4	1457	*0.598	0.624	2	4	823
IdCS	*0.628	0.651	3	4	488	*0.628	0.661	4	8	858	*0.626	0.656	3	5	547

Table B.4: Following Distance with SVR and RR base models: R^2 and PMCC^2 predictive performance, the average number of base models used by the meta-learner ($|\mathcal{M}'|$), the maximum number of base models used by the meta-learner ($\lceil\mathcal{M}'\rceil$), and the average number of relevancy and diversity metric calculations required to compare and evaluate base models (M.Calcs.) for meta-learners in BOTL for the following distance dataset when transferring SVR and RR base models between 7 data streams. Improved predictive performances with statistical t-test values $p < 0.01$ compared to the underlying CDD, while requiring fewer relevancy and diversity metric calculations than MI-Thresh are indicated with *. Of these, bold type indicates the approach with highest performance.

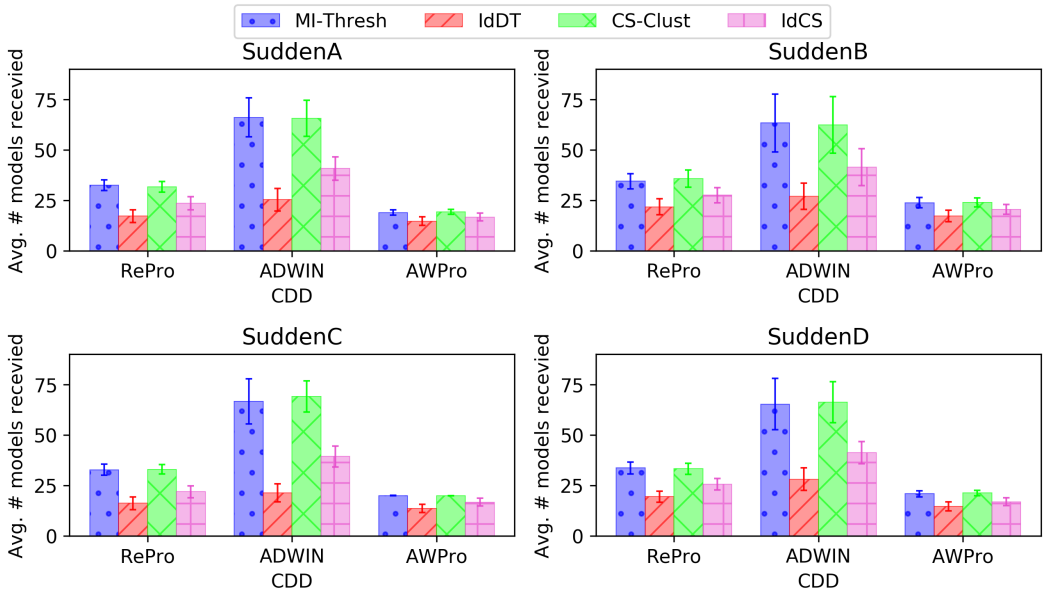


Figure B.1: Sudden Drifting Hyperplane with SVR and RR base models: The average number of SVR and RR models received per domain for MI-Thresh, IdDT, CS-Clust and IdCS when using RePro, ADWIN and AWPro as the underlying CDDs for 5 drifting hyperplane data streams.

the decrease in the average number of relevancy and diversity metric calculations required by IdDT, CS-Clust and IdCS, in comparison to MI-Thresh, as the number of following distance data streams in the framework increases. This shows that the decision of whether to transfer also helps to reduce computational overheads in online TL frameworks, and highlights the benefits of using a static diversity metric. Finally Table B.5 displays a summary of the results obtained by IdDT in comparison to MI-Thresh, and IdCS in comparison to CS-Clust, for the sudden drifting hyperplane, gradual drifting hyperplane, heating simulator and following distance datasets. These results show that both IdDT and IdCS reduce communication and computation overheads while maintaining comparable predictive performances to when all base models are transferred.

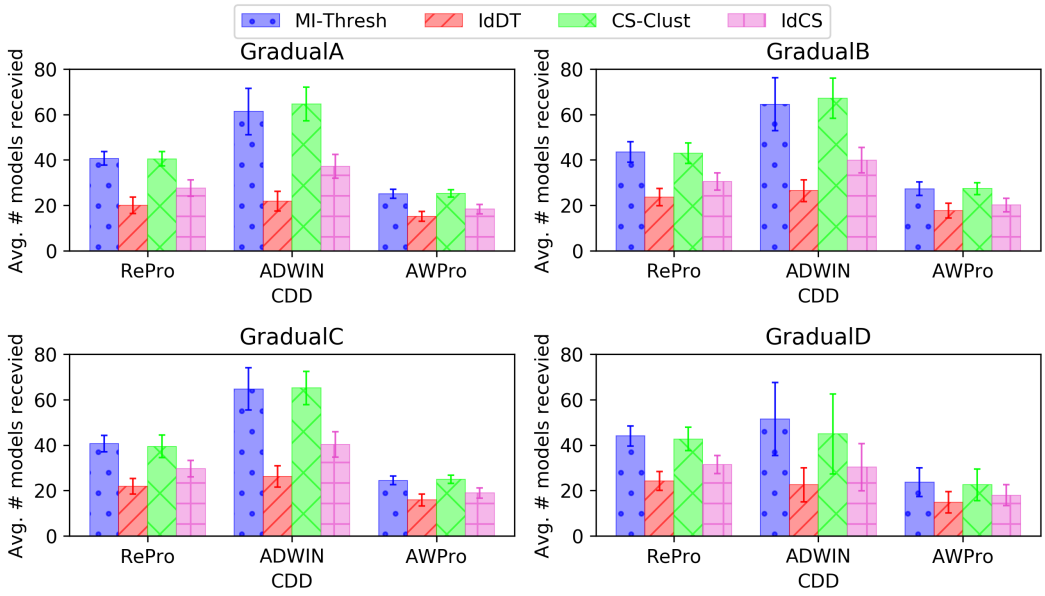


Figure B.2: Gradual Drifting Hyperplane with SVR and RR base models: The average number of SVR and RR models received per domain for MI-Thresh, IdDT, CS-Clust and IdCS when using RePro, ADWIN and AWPro as the underlying CDDs for 5 drifting hyperplane data streams.

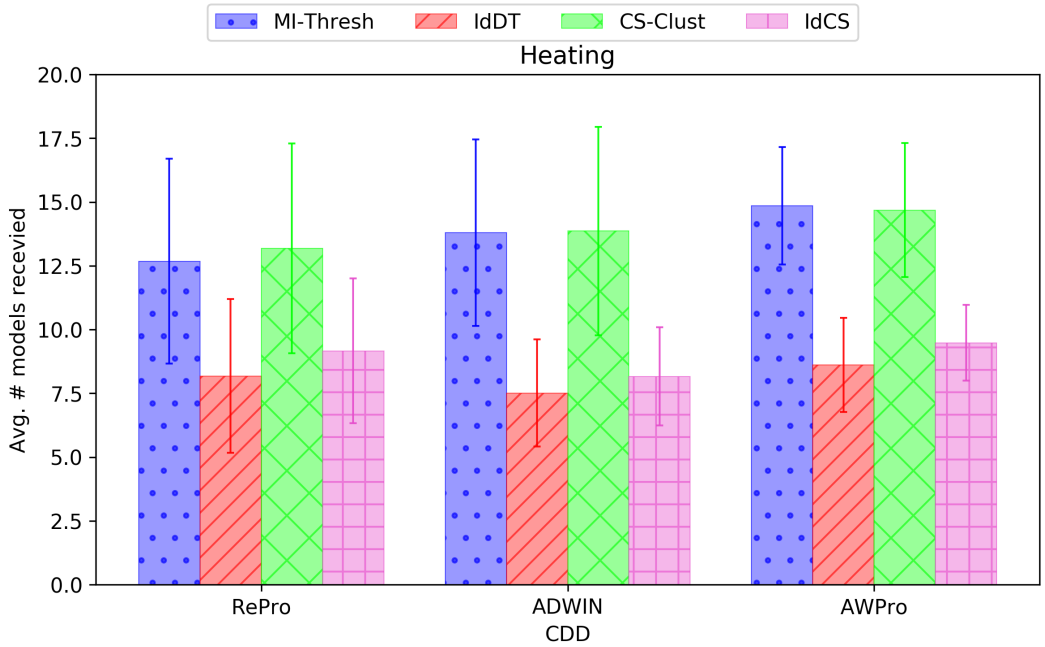


Figure B.3: Heating Simulator with SVR and RR base models: The average number of SVR and RR models received per domain for MI-Thresh, IdDT, CS-Clust and IdCS when using RePro, ADWIN and AWPro as the underlying CDDs for 5 heating simulator data streams.

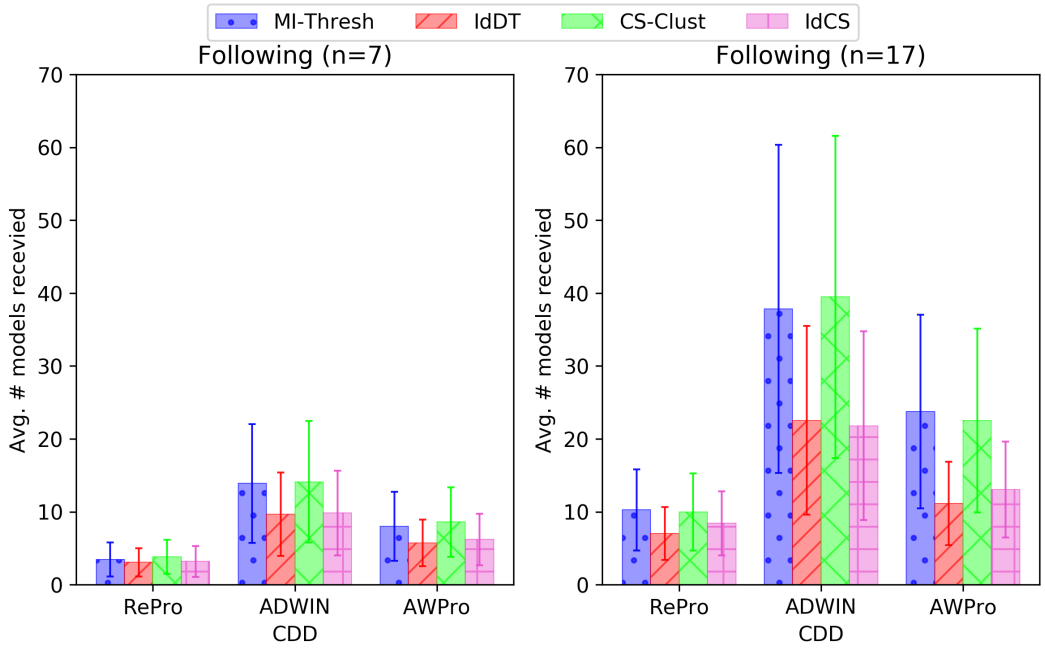


Figure B.4: Following Distance with SVR and RR base models: The average number of SVR and RR models received per domain for MI-Thresh, IdDT, CS-Clust and IdCS when using RePro, ADWIN and AWPro as the underlying CDDs for 7 and 17 following distance data streams.

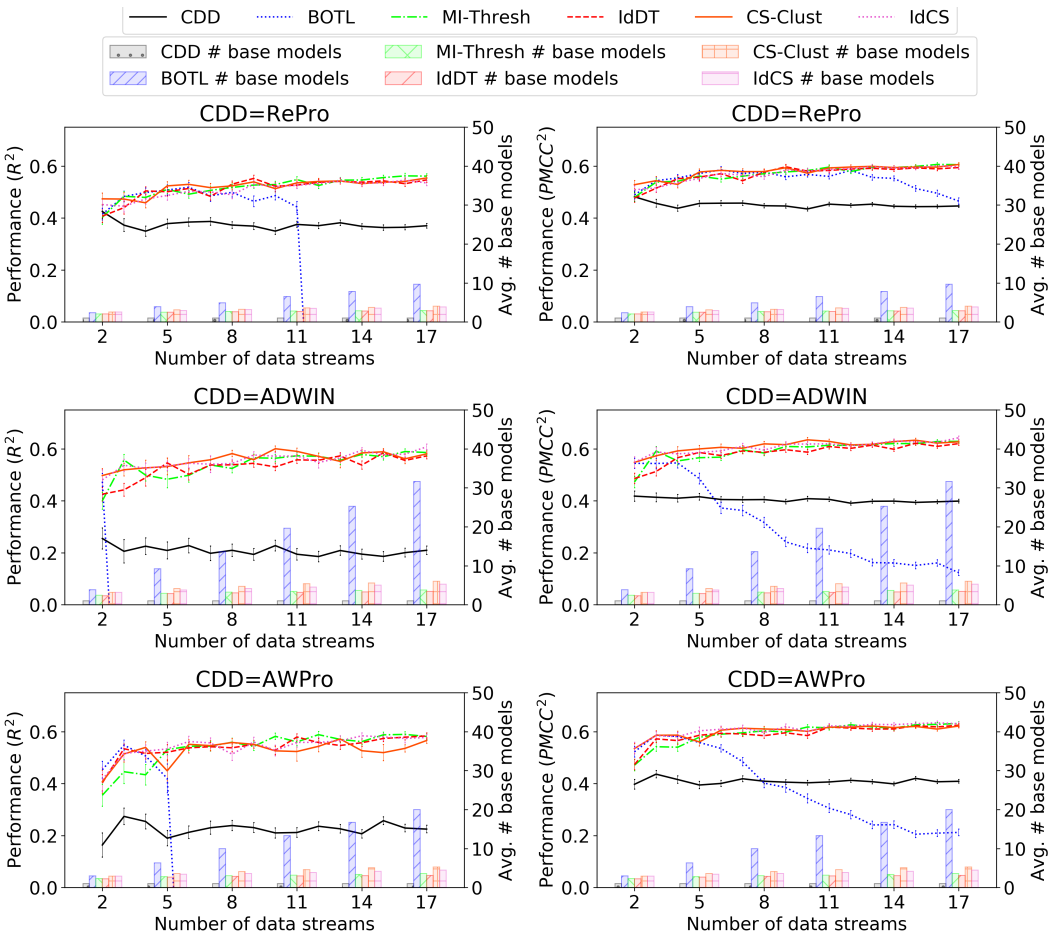


Figure B.5: Following Distance with SVR and RR base models: R^2 and PMCC^2 predictive performance, and number of SVR and RR base models used by BOTL meta-learners, for the underlying CDD, BOTL, MI-Thresh, IdDT, CS-Clust and IdCS with increasing numbers of following distance data streams.

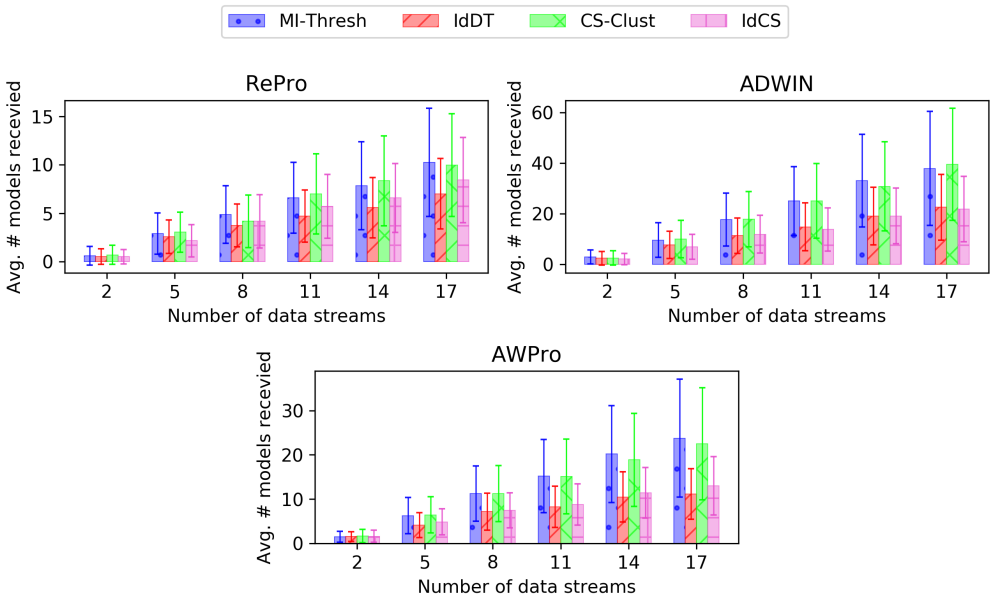


Figure B.6: Following Distance with SVR and RR base models: The average number of SVR and RR models received per domain for MI-Thresh, IdDT, CS-Clust and IdCS for increasing numbers of following distance data streams, using RePro, ADWIN and AWPro as the underlying CDDs.

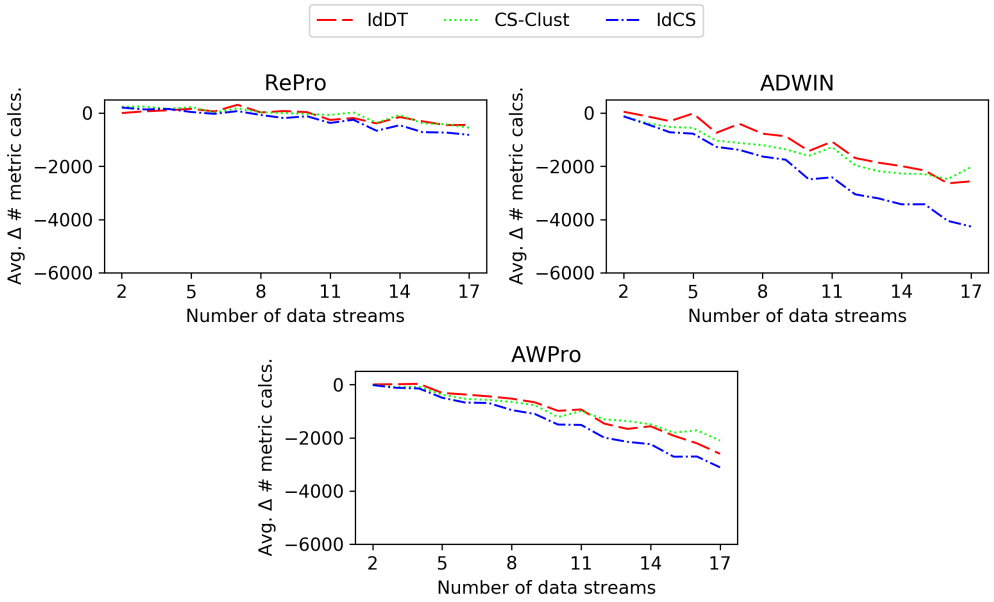


Figure B.7: Following Distance with SVR and RR base models: Change in number of relevancy and diversity metric calculations required to compare and evaluate SVR and RR base models for IdDT, CS-Clust and IdCS in comparison to MI-Thresh for increasing numbers of following distance data streams.

Dataset		RePro ΔComm	RePro ΔComp	RePro ΔR^2	RePro Δ_{PMCC^2}	ADWIN ΔComm	ADWIN ΔComp	ADWIN ΔR^2	ADWIN Δ_{PMCC^2}	AWPro ΔComm	AWPro ΔComp	AWPro ΔR^2	AWPro Δ_{PMCC^2}
SuddenA	IdDT	-47.0%	-60.2%	-0.003	-0.003	-61.7%	-56.8%	0.000	0.000	-22.6%	-52.9%	-0.003	-0.003
	IdCS	-25.4%	-36.1%	-0.001	-0.001	-37.9%	-49.1%	-0.001	-0.001	-13.6%	-28.9%	-0.002	-0.002
SuddenB	IdDT	-36.9%	-48.1%	-0.003	-0.003	-57.4%	-54.4%	0.000	0.000	-27.4%	-48.4%	-0.004	-0.004
	IdCS	-22.9%	-29.9%	+0.900	+0.011	-33.7%	-44.1%	0.000	+0.007	-14.8%	-27.2%	+0.838	+0.019
SuddenC	IdDT	-50.7%	-62.4%	-0.001	-0.001	-68.0%	-65.4%	+0.002	+0.002	-31.9%	-58.0%	-0.001	-0.001
	IdCS	-33.6%	-41.2%	-0.002	-0.002	-43.1%	-54.5%	-0.004	-0.004	-16.1%	-30.4%	-0.005	-0.005
SuddenD	IdDT	-42.3%	-50.4%	-0.005	-0.005	-57.0%	-53.6%	+0.002	+0.002	-29.5%	-49.4%	-0.003	-0.003
	IdCS	-22.9%	-30.8%	0.000	0.000	-37.8%	-49.2%	0.000	0.000	-20.7%	-29.9%	+0.002	+0.001
GradualA	IdDT	-50.9%	-64.1%	-0.005	-0.005	-64.4%	-64.1%	+0.003	+0.003	-39.4%	-63.4%	-0.012	-0.012
	IdCS	-31.7%	-39.5%	-0.001	-0.001	-42.5%	-53.9%	-0.001	-0.001	-27.3%	-38.0%	0.000	0.000
GradualB	IdDT	-45.6%	-56.1%	-0.002	-0.002	-58.9%	-58.7%	+0.005	+0.005	-35.0%	-57.9%	-0.010	-0.010
	IdCS	-29.1%	-37.1%	0.000	-0.005	-40.6%	-52.1%	0.000	-0.021	-26.4%	-38.4%	+0.827	+0.020
GradualC	IdDT	-46.3%	-53.8%	-0.004	-0.004	-59.5%	-58.0%	-0.004	-0.004	-35.0%	-55.7%	-0.019	-0.019
	IdCS	-24.9%	-33.6%	-0.001	-0.001	-38.2%	-49.3%	0.000	0.000	-24.2%	-33.5%	-0.002	-0.002
GradualD	IdDT	-45.0%	-48.0%	-0.004	-0.004	-56.3%	-54.8%	-0.029	-0.029	-37.3%	-47.5%	-0.040	-0.040
	IdCS	-26.3%	-33.2%	-0.002	-0.002	-32.5%	-42.5%	+0.008	+0.008	-20.3%	-30.4%	+0.010	+0.010
Heating	IdDT	-35.5%	-34.4%	-0.001	-0.001	-45.5%	-46.9%	-0.004	-0.003	-42.0%	-21.8%	-0.005	-0.005
	IdCS	-30.5%	-16.5%	-0.005	-0.005	-41.1%	-21.6%	-0.005	-0.005	-35.4%	-25.0%	-0.005	-0.005
Following (n=7)	IdDT	-11.6%	11.2%	+0.005	+0.004	-30.5%	-27.9%	-0.009	-0.007	-28.5%	-25.6%	-0.027	-0.025
	IdCS	-16.9%	-13.1%	-0.003	-0.004	-30.3%	-34.4%	-0.016	-0.011	-27.8%	-31.2%	-0.007	-0.007
Following (n=17)	IdDT	-31.7%	-18.5%	-0.015	-0.011	-40.4%	-39.5%	-0.013	-0.009	-53.1%	-58.6%	+0.002	-0.007
	IdCS	-15.4%	-14.6%	-0.019	-0.009	-44.8%	-50.0%	+0.027	+0.016	-42.1%	-43.1%	+0.012	+0.009

Table B.5: Summary of comparisons between MI-Thresh and IdDT, and CS-Clust and IdCS, when SVRs and RRs are used as base models, showing the average reduction in number of models received (%), denoted as Δ Comm., the average reduction in performance and diversity metric calculations (%), denoted as Δ Comp., and the average difference between R^2 and PMCC^2 predictive performances.

Bibliography

[1] Supriya Agrahari and Anil Kumar Singh. Concept drift detection in data stream mining : A literature review. *Journal of King Saud University - Computer and Information Sciences*, 2021. ISSN 1319–1578. doi: 10.1016/j.jksuci.2021.11.006.

[2] Andreas Argyriou, Andreas Maurer, and Massimiliano Pontil. An algorithm for transfer learning in a heterogeneous environment. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 71–85. Springer, 2008. doi: 10.1007/978-3-540-87479-9_23.

[3] Andrew Arnold, Ramesh Nallapati, and William W. Cohen. A comparative study of methods for transductive transfer learning. In *Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007)*, pages 77–82, 2007. doi: 10.1109/ICDMW.2007.109.

[4] Bart Bakker and Tom Heskes. Task clustering and gating for bayesian multitask learning. *J. Mach. Learn. Res.*, 4(null):83–99, dec 2003. ISSN 1532–4435. doi: 10.1162/153244304322765658.

[5] Roberto Souto Maior Barros and Silas Garrido T. Carvalho Santos. A large-scale comparison of concept drift detectors. *Information Sciences*, 451-452: 348–370, 2018. ISSN 0020-0255. doi: 10.1016/j.ins.2018.04.014.

[6] Shai Ben-David and Reba Schuller. Exploiting task relatedness for multiple task learning. In *Learning theory and kernel machines*, pages 567–580. Springer, 2003. doi: 10.1007/978-3-540-45167-9_41.

[7] Albert Bifet. Adaptive learning and mining for data streams and frequent patterns. *SIGKDD Explor. Newsl.*, 11(1):55–56, November 2009. ISSN 1931-0145. doi: 10.1145/1656274.1656287.

[8] Albert Bifet and Ricard Gavalda. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 443–448. SIAM, 2007. doi: 10.1137/1.9781611972771.42.

[9] Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Richard Kirkby, and Ricard Gavaldà. New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 139–148, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605584959. doi: 10.1145/1557019.1557041.

[10] Åke Björck and Gene H Golub. Numerical methods for computing angles between linear subspaces. *Mathematics of computation*, 27(123):579–594, 1973. doi: 10.2307/2005662.

[11] Gavin Brown, Jeremy L Wyatt, and Peter Tiňo. Managing diversity in regression ensembles. *Journal of machine learning research*, 6(Sep):1621–1650, 2005.

[12] Dariusz Brzezinski and Jerzy Stefanowski. Ensemble diversity in evolving data streams. In Toon Calders, Michelangelo Ceci, and Donato Malerba, editors, *Discovery Science*, pages 229–244, Cham, 2016. Springer International Publishing. doi: 10.1007/978-3-319-46307-0_15.

[13] Marcin Budka and Bogdan Gabrys. Ridge regression ensemble for toxicity prediction. *Procedia Computer Science*, 1(1):193–201, 2010. ISSN 1877-0509. doi: 10.1016/j.procs.2010.04.022. ICCS 2010.

[14] Vladimir Cherkassky and Filip M Mulier. *Learning from data: concepts, theory, and methods*. John Wiley & Sons, 2007. doi: 10.1198/tech.2001.s558.

[15] Chun Wai Chiu and Leandro L. Minku. A diversity framework for dealing with multiple types of concept drift based on clustering in the model space. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–11, 2020. doi: 10.1109/TNNLS.2020.3041684.

[16] Ajay Choudhary, Preeti Jha, Aruna Tiwari, and Neha Bharill. A brief survey on concept drifted data stream regression. In Aruna Tiwari, Kapil Ahuja, Anupam Yadav, Jagdish Chand Bansal, Kusum Deep, and Atulya K. Nagar, editors, *Soft Computing for Problem Solving*, pages 733–

744, Singapore, 2021. Springer Singapore. ISBN 978-981-16-2712-5. doi: 10.1007/978-981-16-2712-5_57.

[17] Hal Daume III and Daniel Marcu. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126, 2006. doi: 10.1613/JAIR.1872.

[18] Oscar Day and Taghi M Khoshgoftaar. A survey on heterogeneous transfer learning. *Journal of Big Data*, 4(1):1–42, 2017. doi: 10.1186/s40537-017-0089-0.

[19] Bo Dong, Yifan Li, Yang Gao, Ahsanul Haque, Latifur Khan, and Mohammad M. Masud. Multistream regression with asynchronous concept drift detection. In *2017 IEEE International Conference on Big Data*, pages 596–605, Dec 2017. doi: 10.1109/BIGDATA.2017.8257975.

[20] Honghui Du, Leandro L. Minku, and Huiyu Zhou. Multi-source transfer learning for non-stationary environments. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2019. doi: 10.1109/IJCNN.2019.8852024.

[21] Honghui Du, Leandro L. Minku, and Huiyu Zhou. Marline: Multi-source mapping transfer learning for non-stationary environments. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 122–131, 2020. doi: 10.1109/ICDM50108.2020.00021.

[22] Ke-Lin Du and M. N. S. Swamy. Combining multiple learners: Data fusion and ensemble learning. In *Neural Networks and Statistical Learning*, pages 737–767. Springer London, London, 2019. ISBN 978-1-4471-7452-3. doi: 10.1007/978-1-4471-7452-3_25.

[23] Haimonti Dutta. Measuring diversity in regression ensembles. In *Proceedings of the 4th Indian International Conference on Artificial Intelligence (IICAI)*, volume 9, pages 2220–2236, 2009.

[24] Michael P Fay and Michael A Proschan. Wilcoxon-mann-whitney or t-test? on assumptions for hypothesis tests and multiple interpretations of decision rules. *Statistics surveys*, 4:1, 2010.

[25] Jerome H. Friedman. On bias, variance, 0/1—loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1(1):55–77, Mar 1997. ISSN 1573-756X. doi: 10.1023/A:1009778005914.

[26] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37, 2014. doi: 10.1145/2523813.

[27] Liang Ge, Jing Gao, and Aidong Zhang. OMS-TL: A framework of online multiple source transfer learning. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 2423–2428, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450322638. doi: 10.1145/2505515.2505603.

[28] Liang Ge, Jing Gao, Hung Ngo, Kang Li, and Aidong Zhang. On handling negative transfer and imbalanced distributions in multiple source transfer learning. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 7 (4):254–271, 2014. doi: 10.1002/sam.11217.

[29] Gene H. Golub and Hongyuan Zha. The canonical correlations of matrix pairs and their numerical computation. In Adam Bojanczyk and George Cybenko, editors, *Linear Algebra for Signal Processing*, pages 27–49, New York, NY, 1995. Springer New York. ISBN 978-1-4612-4228-4. doi: 10.1007/978-1-4612-4228-4_3.

[30] Heitor Murilo Gomes, Jean Paul Barddal, Fabrício Enembreck, and Albert Bifet. A survey on ensemble learning for data stream classification. *ACM Comput. Surv.*, 50(2), March 2017. ISSN 0360-0300. doi: 10.1145/3054925.

[31] Symone Gomes Soares and Rui Araújo. An online weighted ensemble of regressor models to handle concept drifts. *Engineering Applications of Artificial Intelligence*, 37:392–406, 2015. ISSN 0952-1976. doi: 10.1016/j.engappai.2014.10.003.

[32] Thomas Grubinger, Georgios Chasparis, and Thomas Natschläger. Online transfer learning for climate control in residential buildings. In *Proceedings of the 5th Annual European Control Conference (ECC 2016)*, pages 1183–1188, 2016. doi: 10.1109/ECC.2016.7810450.

[33] Thomas Grubinger, Georgios Chasparis, and Thomas Natschläger. Generalized online transfer learning for climate control in residential buildings. *Energy and Buildings*, 139:63–71, 2017. doi: 10.1016/J.ENBUILD.2016.12.074.

[34] Nils Y. Hammerla and Thomas Plötz. Let's (not) stick together: Pairwise similarity biases cross-validation in activity recognition. In *Proceedings of the*

2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp '15, pages 1041–1051. Association for Computing Machinery, 2015. ISBN 9781450335744. doi: 10.1145/2750858.2807551.

[35] Ahsanul Haque, Hemeng Tao, Swarup Chandra, Jie Liu, and Latifur Khan. A framework for multistream regression with direct density ratio estimation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[36] Juan I. G. Hidalgo, Silas G. T. C. Santos, and Roberto S. M. Barros. Dynamically adjusting diversity in ensembles for the classification of data streams with concept drift. *ACM Trans. Knowl. Discov. Data*, 16(2), July 2021. ISSN 1556-4681. doi: 10.1145/3466616.

[37] T. Ryan Hoens, Robi Polikar, and Nitesh V. Chawla. Learning from streaming data with concept drift and imbalance: an overview. *Progress in Artificial Intelligence*, 1(1):89–101, 2012. doi: 10.1007/s13748-011-0008-0.

[38] Tingting Hou, Gang Feng, Shuang Qin, and Wei Jiang. Proactive content caching by exploiting transfer learning for mobile edge computing. *International Journal of Communication Systems*, 31(11):3706, 2018. doi: 10.1002/dac.3706.

[39] Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex Smola. Correcting sample selection bias by unlabeled data. *Advances in neural information processing systems*, 19:601–608, 2006.

[40] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 97–106. ACM, 2001. doi: 10.1145/502512.502529.

[41] Ghazal Jaber, Antoine Cornuéjols, and Philippe Tarroux. Online learning: Searching for the best forgetting strategy under concept drift. In Minho Lee, Akira Hirose, Zeng-Guang Hou, and Rhee Man Kil, editors, *Neural Information Processing*, pages 400–408, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. doi: 10.1007/978-3-642-42042-9_50.

[42] Christopher R John, David Watson, Michael R Barnes, Costantino Pitzalis, and Myles J Lewis. Spectrum: fast density-aware spectral clustering for single and multi-omic data. *Bioinformatics*, 36(4):1159–1166, 09 2019. ISSN 1367-4803. doi: 10.1093/bioinformatics/btz704.

[43] Mark G. Kelly, David J. Hand, and Niall M. Adams. The impact of changing populations on classifier performance. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pages 367–371. ACM, 1999. doi: 10.1145/312129.312285.

[44] Kenji Kira and Larry A. Rendell. A practical approach to feature selection. In Derek Sleeman and Peter Edwards, editors, *Machine Learning Proceedings 1992*, pages 249–256. Morgan Kaufmann, San Francisco (CA), 1992. ISBN 978-1-55860-247-2. doi: 10.1016/B978-1-55860-247-2.50037-1.

[45] Andrew V Knyazev and Peizhen Zhu. Principal angles between subspaces and their tangents. *arXiv preprint arXiv:1209.0523*, 2012.

[46] Jeremy Z. Kolter and Marcus A. Maloof. Using additive expert ensembles to cope with concept drift. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pages 449–456. ACM, 2005. doi: 10.1145/1102351.1102408.

[47] Jeremy Z. Kolter and Marcus A. Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning Research*, 8(Dec):2755–2790, 2007. doi: 10.1109/ICDM.2003.1250911.

[48] Ivan Koychev. Gradual forgetting for adaptation to concept drift. Proceedings of ECAI 2000 Workshop on Current Issues in Spatio-Temporal Reasoning, 2000.

[49] Bartosz Krawczyk, Leandro L. Minku, João Gama, Jerzy Stefanowski, and Michał Woźniak. Ensemble learning for data stream analysis: A survey. *Information Fusion*, 37:132–156, 2017. ISSN 1566-2535. doi: 10.1016/j.inffus.2017.02.004.

[50] Guangxia. Li, Steven CH Hoi, Kuiyu Chang, Wenting Liu, and Ramesh Jain. Collaborative online multitask learning. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1866–1876, 2014. doi: 10.1109/TKDE.2013.139.

[51] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys Tutorials*, 22(3):2031–2063, 2020. doi: 10.1109/COMST.2020.2986024.

[52] Yan-Hui Lin and Liang Chang. An online transfer learning framework for time-varying distribution data prediction. *IEEE Transactions on Industrial Electronics*, pages 1–10, 2021. doi: 10.1109/TIE.2021.3090701.

[53] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, João Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363, 2019. doi: 10.1109/TKDE.2018.2876857.

[54] Helen McKay, Nathan Griffiths, Phillip Taylor, Theo Damoulas, and Zhou Xu. Online transfer learning for concept drifting data streams. In *BigMine@ KDD*, volume 2579, pages 1–15, 2019.

[55] Helen McKay, Nathan Griffiths, Phillip Taylor, Theo Damoulas, and Zhou Xu. Bi-directional online transfer learning: a framework. *Annals of Telecommunications*, 75(9):523–547, 2020. doi: 10.1007/s12243-020-00776-1.

[56] Helen McKay, Nathan Griffiths, and Phillip Taylor. Conceptually diverse base model selection for meta-learners in concept drifting data streams. *arXiv preprint arXiv:2111.14520*, 2021.

[57] Leandro L. Minku. Transfer learning in non-stationary environments. In Moamar Sayed-Mouchaweh, editor, *Learning from Data Streams in Evolving Environments: Methods and Applications*, pages 13–37. Springer International Publishing, Cham, 2019. ISBN 978-3-319-89803-2. doi: 10.1007/978-3-319-89803-2_2.

[58] Leandro L. Minku and Xin Yao. DDD: A new ensemble approach for dealing with concept drift. *IEEE Transactions on Knowledge and Data Engineering*, 24(4):619–633, 2012. doi: 10.1109/TKDE.2011.58.

[59] Leandro L. Minku and Xin Yao. How to make best use of cross-company data in software effort estimation? ICSE 2014, pages 446–456, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450327565. doi: 10.1145/2568225.2568228.

[60] Leandro L. Minku, Allan P. White, and Xin Yao. The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering*, 22(5):730–742, 2010. doi: 10.1109/TKDE.2009.156.

[61] Prabhaker Mishra, Chandra Mani Pandey, Uttam Singh, Amit Keshri, and Mayilvaganan Sabaretnam. Selection of appropriate statistical methods for data analysis. *Annals of cardiac anaesthesia*, 22(3):297, 2019.

[62] Keerthiram Murugesan and Jamie Carbonell. Multi-task multiple kernel relationship learning. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 687–695. SIAM, 2017. doi: 10.1137/1.9781611974973.77.

[63] Cong T Nguyen, Nguyen Van Huynh, Nam H Chu, Yuris Mulya Saputra, Dinh Thai Hoang, Diep N Nguyen, Quoc-Viet Pham, Dusit Niyato, Eryk Dutkiewicz, and Won-Joo Hwang. Transfer learning for future wireless networks: A comprehensive survey. *arXiv preprint arXiv:2102.07572*, 2021.

[64] P. Niyogi and F. Girosi. On the relationship between generalization error, hypothesis complexity, and sample complexity for radial basis functions. *Neural Computation*, 8(4):819–842, 1996. doi: 10.1162/neco.1996.8.4.819.

[65] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. doi: 10.1109/TKDE.2009.191.

[66] Zhengjun Qiu, Shutao Zhao, Xuping Feng, and Yong He. Transfer learning method for plastic pollution evaluation in soil using nir sensor. *Science of The Total Environment*, 740:140118, 2020. ISSN 0048-9697. doi: 10.1016/j.scitotenv.2020.140118.

[67] Sasthakumar Ramamurthy and Raj Bhatnagar. Tracking recurrent concept drift in streaming data using ensemble classifiers. In *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*, pages 404–409, 2007. doi: 10.1109/ICMLA.2007.80.

[68] D.C. Reicosky, L.J. Winkelman, J.M. Baker, and D.G. Baker. Accuracy of hourly air temperatures calculated from daily minima and maxima. *Agricultural and Forest Meteorology*, 46(3):193–209, 1989. ISSN 0168-1923. doi: 10.1016/0168-1923(89)90064-6.

[69] Ye Ren, Le Zhang, and P.N. Suganthan. Ensemble classification and regression-recent developments, applications and future directions [review article]. *IEEE Computational Intelligence Magazine*, 11(1):41–53, 2016. doi: 10.1109/MCI.2015.2471235.

[70] Michael T Rosenstein, Zvika Marx, Leslie Pack Kaelbling, and Thomas G Dietterich. To transfer or not to transfer. In *NIPS 2005 workshop on transfer learning*, volume 898, pages 1–4, 2005.

[71] Paul Ruvolo and Eric Eaton. Active task selection for lifelong machine learning. In *Twenty-seventh AAAI conference on artificial intelligence*, 2013.

[72] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249, 2018. doi: 10.1002/widm.1249.

[73] Avishek Saha, Piyush Rai, Hal Daumã, and Suresh Venkatasubramanian. Online learning of multiple tasks and their relationships. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pages 643–651, 2011.

[74] Jeffery C. Schlimmer and Richard H. Granger. Incremental learning from noisy data. *Machine Learning*, 1(3):317–354, 1986. doi: 10.1007/BF00116895.

[75] W. Nick Street and YongSeog Kim. A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 377–382. ACM, 2001. doi: 10.1145/502512.502568.

[76] Yu Sun, Ke Tang, Zexuan Zhu, and Xin Yao. Concept drift adaptation by exploiting historical knowledge. *IEEE Transactions on Neural Networks and Learning Systems*, 29(10):4822–4832, 2018. doi: 10.1109/TNNLS.2017.2775225.

[77] Tatiana Tommasi and Barbara Caputo. The more you know, the less you learn: from knowledge transfer to one-shot learning of object categories. In *BMVC*, 2009.

[78] Alexey Tsymbal. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, 106(2):58, 2004.

[79] N. Ueda and R. Nakano. Generalization error of ensemble estimators. In *Proceedings of International Conference on Neural Networks (ICNN'96)*, volume 1, pages 90–95, June 1996. doi: 10.1109/ICNN.1996.548872.

[80] Wouter van Loon, Marjolein Fokkema, Botond Szabo, and Mark de Rooij. View selection in multi-view stacking: Choosing the meta-learner. *arXiv preprint arXiv:2010.16271*, 2020.

[81] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1999.

[82] Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 226–235, New York, NY, USA, 2003. Association for Computing Machinery. ISBN 1581137370. doi: 10.1145/956750.956778.

[83] Zirui Wang, Zihang Dai, Barnabás Póczos, and Jaime Carbonell. Characterizing and avoiding negative transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11293–11302, 2019.

[84] Geoffrey I. Webb, Roy Hyde, Hong Cao, Hai Long Nguyen, and Francois Petitjean. Characterizing concept drift. *Data Mining and Knowledge Discovery*, 30(4):964–994, 2016. ISSN 1573-756X. doi: 10.1007/s10618-015-0448-4.

[85] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016. doi: 10.1186/s40537-016-0043-6.

[86] Haiyu Wu, Huayu Li, Alireza Shamsoshoara, Abolfazl Razi, and Fatemeh Afghah. Transfer learning for wildfire identification in UAV imagery. In *2020 54th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6, 2020. doi: 10.1109/CISS48834.2020.1570617429.

[87] Ocean Wu, Yun Sing Koh, Gillian Dobbie, and Thomas Lacombe. Transfer learning with adaptive online tradaboost for data streams. In Vineeth N. Balasubramanian and Ivor Tsang, editors, *Proceedings of The 13th Asian Conference on Machine Learning*, volume 157 of *Proceedings of Machine Learning Research*, pages 1017–1032. PMLR, 17–19 Nov 2021.

[88] Qingyao Wu, Xiaoming Zhou, Yuguang Yan, Hanrui Wu, and Huaqing Min. Online transfer learning by leveraging multiple source domains. *Knowledge and Information Systems*, 52(3):687–707, 2017. doi: 10.1007/s10115-016-1021-1.

[89] Xuetong Wu, Jonathan H Manton, Uwe Aickelin, and Jingge Zhu. Online transfer learning: Negative transfer and effect of prior knowledge. *arXiv preprint arXiv:2105.01445*, 2021.

[90] Huan Xu, Constantine Caramanis, and Sujay Sanghavi. Robust PCA via outlier pursuit. *arXiv preprint arXiv:1010.4237*, 2010.

[91] Yuguang Yan, Qingyao Wu, Mingkui Tan, Michael K. Ng, Huaqing Min, and Ivor W. Tsang. Online heterogeneous transfer by hedge ensemble of offline and online decisions. *IEEE Transactions on Neural Networks and Learning Systems*, PP(99):1–12, 2017. doi: 10.1109/TNNLS.2017.2751102.

[92] Cuie Yang, Yiu-ming Cheung, Jinliang Ding, and Kay Chen Tan. Concept drift-tolerant transfer learning in dynamic environments. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2021. doi: 10.1109/TNNLS.2021.3054665.

[93] Libo Yang, Xuemei Liu, Feiping Nie, and Mingtang Liu. Large-scale spectral clustering based on representative points. *Mathematical Problems in Engineering*, 2019:1–7, 2019. doi: 10.1155/2019/5864020.

[94] Ying Yang, Xindong Wu, and Xingquan Zhu. Combining proactive and reactive predictions for data streams. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 710–715. ACM, 2005. doi: 10.1145/1081870.1081961.

[95] Ying Yang, Xindong Wu, and Xingquan Zhu. Mining in anticipation for concept change: Proactive-reactive prediction in data streams. *Data Mining and Knowledge Discovery*, 13(3):261–289, 2006. doi: 10.1145/1081870.1081961.

[96] Yi Yao and Gianfranco Doretto. Boosting for transfer learning with multiple sources. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1855–1862, 2010. doi: 10.1109/CVPR.2010.5539857.

[97] Rui Ye and Qun Dai. A novel transfer learning framework for time series forecasting. *Knowledge-Based Systems*, 156:74–99, 2018. ISSN 0950-7051. doi: 10.1016/j.knosys.2018.05.021.

[98] Haibo Yin and Yun-an Yang. Online transfer learning with extreme learning machine. In *AIP Conference Proceedings*, volume 1839, page 020199. AIP Publishing, 2017. doi: 10.1063/1.4982564.

[99] G. Udny Yule. On the association of attributes in statistics: With illustrations from the material of the childhood society. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 194:257–319, 1900. ISSN 02643952.

[100] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *Proceedings of the 17th International Conference on Neural Information Processing Systems*, NIPS'04, pages 1601–1608, Cambridge, MA, USA, 2004. MIT Press. doi: 10.5555/2976040.2976241.

[101] Xianchao Zhang, Jingwei Li, and Hong Yu. Local density adaptive similarity measurement for spectral clustering. *Pattern Recognition Letters*, 32(2):352–358, 2011. ISSN 0167-8655. doi: 10.1016/j.patrec.2010.09.014.

[102] Deli Zhao, Zhouchen Lin, and Xiaoou Tang. Laplacian pca and its applications. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, 2007. doi: 10.1109/ICCV.2007.4409096.

[103] Juan Zhao, Sachin Shetty, Jan Wei Pan, Charles Kamhoua, and Kevin Kwiat. Transfer learning for detecting unknown network attacks. *EURASIP Journal on Information Security*, 2019(1):1–13, 2019. doi: 10.1186/s13635-019-0084-4.

[104] Peilin Zhao and Steven C. Hoi. OTL: A framework of online transfer learning. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 1231–1238, 2010.

[105] Peilin Zhao, Steven Hoi, Jialei Wang, and Bin Li. Online transfer learning. *Artificial Intelligence*, 216:76–102, 2014. doi: 10.1016/j.artint.2014.06.003.

[106] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2012. doi: 10.1201/b12207.

[107] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2021. doi: 10.1109/JPROC.2020.3004555.

[108] Indrė Žliobaitė. Learning under concept drift: an overview. *arXiv preprint arXiv:1010.4784*, 2010.