Online Transfer Learning for Concept Drifting Data Streams

Helen McKay*
Department of Computer Science
University of Warwick
Coventry, UK

Nathan Griffiths
Department of Computer Science
University of Warwick
Coventry, UK

Phillip Taylor
Department of Computer Science
University of Warwick
Coventry, UK

Theo Damoulas
Department of Computer Science
Department of Statistics
University of Warwick
Coventry, UK

Zhou Xu
Jaguar Land Rover Research
Coventry, UK

ABSTRACT

Transfer learning uses knowledge learnt in a source domain to aid predictions in a target domain. When both source and target domains are online, each are susceptible to concept drift, which may alter the mapping of knowledge between them. Drifts in online domains can make additional information available, necessitating knowledge transfer both from the source to the target and vice versa. To address this we introduce the Bi-directional Online Transfer Learning framework (BOTL), which uses knowledge learnt in each online domain to aid predictions in others. We also introduce two variants of BOTL that incorporate model culling to minimise negative transfer in frameworks with large numbers of domains. We provide a theoretical performance guarantee that indicates BOTL achieves a loss no worse than the underlying local concept drift detection algorithm. Empirical results are presented using two data stream generators: the drifting hyperplane emulator and the smart home heating simulator, and real-world data predicting Time To Collision (TTC) from vehicle telemetry. The evaluation shows BOTL and it's variants outperform the existing state-of-the-art technique.

CCS CONCEPTS

• **Computing methodologies** → **Online learning settings**; *Transfer learning*; *Learning under covariate shift*; Ensemble methods.

KEYWORDS

Transfer Learning, Online Learning, Concept Drift

ACM Reference Format:
Helen McKay, Nathan Griffiths, Phillip Taylor, Theo Damoulas, and Zhou Xu. 2019. Online Transfer Learning for Concept Drifting Data Streams. In *The 8th International Workshop on Big Data, IoT Streams and Heterogeneous Source Mining - A KDD Workshop, August 2019, Anchorage, AK*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456

*Corresponding author (H.McKay@warwick.ac.uk)

KDD BigMine-19, August 2019, Anchorage, AK

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Online Learning (OL) and Transfer Learning (TL) have been extensively studied within the machine learning community [6, 19, 24]. OL enables supervised learning to be conducted upon data streams that are susceptible to concept drift [6]. Concept drift can cause the distribution of data to change over time, modifying the underlying concept, meaning predictive models must be updated or discarded to maintain effective predictions. To build accurate models, many real-world applications require large amounts of training data, which is often limited due to concept drifts [19].

TL enables models to be learnt in domains where training data is readily available, and used where it is limited to build more effective predictors [19]. TL has typically been conducted offline, limiting its use in real-world online environments [30]. It may be desirable to use on-device learning to personalise the functionalities of user facing applications, but a rich history of data may not be available locally due to memory limitations, and drifts may be encountered frequently. Predictive performances could be enhanced using TL in an online setting by using knowledge learnt from others to aid the target predictor.

The Online Transfer Learning framework (OTL), developed by Zhao *et al.* [30], was proposed to enable TL to be used within an online setting. Current versions of OTL, such as [7, 9, 26], assume the source is in an offline environment, ignoring the possibility of concept drift occurring in the source domain.

In this paper, we propose the Bi-directional Online Transfer Learning framework (BOTL)[1], which considers both the source and target to be online. This has three benefits over existing approaches. Firstly, individual concepts are learnt, using concept drift detection strategies, and transferred to improve performance in the target [6]. Secondly, additional knowledge is transferred as new concepts are encountered in the source domain. Thirdly, knowledge can be transferred bi-directionally, enabling more effective predictions to be made in both domains. Specifically, we:

- introduce the BOTL framework, enabling each domain to benefit from online TL in a regression setting,
- provide a theoretical performance guarantee showing predictions made by BOTL are no worse than the underlying local concept drift detection algorithm, and
- show the performance of BOTL exceeds existing state-of-the-art techniques using a variety of datasets.

[1] Available here: https://github.com/hmckay/BOTL

We evaluate BOTL in a regression setting using two synthetic datasets and one real-world dataset containing both sudden and gradual drifts. We compare BOTL with a state-of-the-art online TL framework, the Generalised Online Transfer Learning framework (GOTL), which assumes the source is offline [9].

The remainder of this paper is organised as follows. Section 2 outlines related work. Section 3 formulates the setting in which BOTL is used. Section 4 presents the proposed framework, and its theoretical performance guarantee is presented in Section 5. Section 6 specifies the data used for results presented in Section 7. Finally, Section 8 concludes the paper.

2 RELATED WORK

Online TL combines *OL* and *TL*. The aim of TL is to use knowledge learnt in one task, referred to as the source, to improve the effectiveness of predictions in another, referred to as the target [18]. There are three distinct types of TL: inductive, transductive and unsupervised [1, 3, 19]. 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 [3]. Typically, large amounts of labelled target data is required to create a mapping between domains [19]. Unsupervised TL is applied in a similar way, but to tasks such as clustering [19]. Transductive TL is used when source and target tasks are the same, transferring knowledge to improve predictive performances for a target where no labelled data is available [1]. TL can be further categorised as homogeneous, where the domains of source and target are the same, or heterogeneous, where they differ [30]. In this paper we consider a homogeneous setting, and use inductive TL to improve the predictive performances within both source and target domains.

It is desirable, for many modern applications, such as smart home heating systems, to predict future events from historical data. However, applications are often limited by memory constraints, preventing a complete history of data being retained [8]. Additionally due to the dynamic and non-stationary environment of data streams, the underlying concept may evolve or drift over time [13]. Concept drift is a change in the distribution of the observed data, or a change in the mapping between observations and values to be predicted [11]. If the underlying concept changes, the previously built model may no longer make effective predictions, requiring the model to be modified or re-learnt [6].

To maintain effective predictions, concept drift detection algorithms are frequently used in OL. Concept drift detection algorithms typically use a sliding window to maintain a subset of recent instances, usually used to update or rebuild the model. Strategies to update a model include ensemble learning approaches, where the window of recent instances is used to create a new model and combined with previously learnt models to improve the predictive performance. Model predictions are aggregated, for example, DWM uses a mean weighted by the models' estimated performance [14]. Alternatively, concept drift detection algorithms such as ADWIN [2] use the window of data to create a model that represents the current concept independently of previous ones.

A challenge associated with these concept drift detection strategies is that every time a concept is encountered, a new model must be learnt, and data must be collected to build the model. RePro [27] uses an approach similar to ADWIN, but retains a history of concepts and concept transitions to prevent learning recurring concepts [28]. This prevents the need to collect new data each time a recurring concept is encountered, however, data must still be collected to build models for new concepts. For many real-world applications, particularly those that are user facing, knowledge obtained from other users could enhance predictions when new concepts are encountered through the use of online TL.

Existing online TL frameworks aim to transfer knowledge learnt from an offline source to an online target for classification tasks. OTL [30], combines the offline source model with the online target model using a weighting mechanism that is updated with respect to the performance of the models on a sliding window of data in the target domain. GOTL [9] 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 [8]. 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.

The field of online TL relates to Online Multi-task Learning (OMTL) [17, 20, 21], and Multistream Regression (MSR) [10]. MSR can be seen as a special case, where the source and target data streams are drawn from the same underlying distribution, and concepts encountered in the target domain have previously been encountered in the source [4]. This means 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 the data streams are drawn from the same distribution. The goal of OMTL is to minimise the cumulative global loss across all domains [16], 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.

Although online TL has been actively studied [7, 9, 25, 26, 29, 30], existing approaches assume the source is offline. We propose BOTL, which considers both source and target in online environments, as might be expected in real-world applications such as smart home heating, or vehicle Adaptive Cruise Control (ACC), personalisations.

3 PROBLEM FORMULATION

Let domain $\mathcal{D}$ consist of a feature space χ, where $x_t \in \mathbb{R}^m$ is the instance observed at time t such that $x_t = \{x_{t_1}, \ldots, x_{t_m}\} \in \chi$. Given domain $\mathcal{D}$, a task consists of the target response variable, $y \in Y$, where $y \in \mathbb{R}$, and a regression function, $f : \chi \rightarrow Y$, which is learnt to map observed data to the target concept [19]. The knowledge learnt in a source domain, $\mathcal{D}^{\mathcal{S}}$, can be transferred to the target domain, $\mathcal{D}^{\mathcal{T}}$, and used to enhance predictions [24].

Online TL aims to learn the target predictive function, $f^{\mathcal{T}}$, that effectively predicts the response variable, $y_t^{\mathcal{T}} \in Y^{\mathcal{T}}$, for each instance, $x_t^{\mathcal{T}} \in \chi^{\mathcal{T}}$, observed in the target data stream, such that

Table 1: Notation

	Definition
$\mathcal{D}^{\alpha}$	Domain α: target, $\mathcal{T}$, or source, $\mathcal{S}$
χ^{α}	Data stream α
Y^{α}	Response variables of α
$x_t \in \chi^{\alpha}$	The t^{th} observed instance in χ^{α}
$y_t \in Y^{\alpha}$	The response variable of instance x_t
$\mathcal{M}$	Knowledge base of models
$f^{\alpha}_{\beta}: \chi^{\alpha} \rightarrow Y^{\alpha}$	Model β learnt in domain α
$F^{\mathcal{M}}: \chi^{\mathcal{T}} \rightarrow Y^{\mathcal{T}}$	Meta-model of $\mathcal{M}$
$\hat{y}_t$	Prediction using $F^{\mathcal{M}}(x_t)$
$\hat{y}_t^{'\alpha_\beta}$	Prediction using $f^{\alpha}_{\beta}(x_t)$
W	Sliding window of instances
W_{max}	Maximum window size
err_t	Predictive error of instance x_t
err_W	Predictive error across W
λ_l	Loss threshold
λ_d	Drift threshold
λc_{perf}	Performance culling threshold
λc_{MI}	Mutual information culling threshold

$\hat{y}_t^{'\mathcal{T}_i} = f_i^{\mathcal{T}}(x_t^{\mathcal{T}})$. Model transfer is used to enhance the target predictor 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, we may be able to utilise knowledge learnt from another user, $f_j^{\mathcal{S}}$, to enhance the predictive performance of the target learner. Identifying concept drift in the source allows models to be transferred, $f_j^{\mathcal{S}}$ where $j = 1 \ldots k$, for each of the k concepts encountered.

BOTL aims to minimise the predictive error in the target domain by combining knowledge learnt from the target data stream with models previously learnt in a source domain. Focusing on minimising the loss with respect to the local, or target, domain makes BOTL highly applicable to the task of application personalisation, where predictions are made to benefit a specific individual. To achieve this, if we have a source domain, $\mathcal{D}^{\mathcal{S}}$, that has previously learnt models $f_1^{\mathcal{S}}, \ldots, f_j^{\mathcal{S}}$, and a target domain, $\mathcal{D}^{\mathcal{T}}$, that has previously learnt models $f_1^{\mathcal{T}}, \ldots, f_i^{\mathcal{T}}$, at time t, then models $f_1^{\mathcal{S}}, \ldots, f_j^{\mathcal{S}}$, should be made available to the target domain such that the target learner can benefit from the knowledge learnt in the source domain, $\mathcal{D}^{\mathcal{S}}$. As both domains are online, and knowledge transfer is bi-directional, the models $f_1^{\mathcal{T}}, \ldots, f_i^{\mathcal{T}}$ should also be made available to the source domain, $\mathcal{D}^{\mathcal{S}}$, such that the source learner can benefit from the knowledge learnt in the target domain, $\mathcal{D}^{\mathcal{T}}$.

In this paper, the source and target domains are considered to be homogeneous, such that they share the same underlying feature space, $\chi^{\mathcal{S}} = \chi^{\mathcal{T}}$, and $Y^{\mathcal{S}} = Y^{\mathcal{T}}$. Although the domains are homogeneous, the underlying concepts to be learnt within source and target domains may not be equivalent, therefore models from a source domain may not be relevant to the current target concept.

Algorithm 1 Adapted RePro for regression.

Input: W_{max}, λ_l, λ_d, $\chi^{\mathcal{T}}$, $H^{\mathcal{T}}=\emptyset$ (historical concepts), $M^{\mathcal{T}}=\emptyset$ (transition matrix).
Learn $f_1^{\mathcal{T}}$ using $x_1 \ldots x_{W_{max}}$, add to $H^{\mathcal{T}}$
for $t=W_{max}+1, W_{max}+2, \ldots$ **do**
  Receive x_t and predict $\hat{y}_t'=f_i^{\mathcal{T}}(x_t)$
  Receive y_t, add $\langle x_t, \hat{y}_t', y_t \rangle$ to W
  if $f_i^{\mathcal{T}}$ is new and stable **then**
    Add $f_i^{\mathcal{T}}$ to $H^{\mathcal{T}}$ and $f_{i-1}^{\mathcal{T}} \rightarrow f_i^{\mathcal{T}}$ to $M^{\mathcal{T}}$
  if $R^2(f_i^{\mathcal{T}}, W) < \lambda_d$ **then**
    $f_{i+1}^{\mathcal{T}}$ = SelectModel($H^{\mathcal{T}}, M^{\mathcal{T}}, W$) or learn new model over W
  else if $|W| \geq W_{max}$ **then**
    Remove $x_{(t-|W|)}$ from W
    while $|f_i^{\mathcal{T}}(x_{(t-|W|)}) - y_{(t-|W|)}| \leq \lambda_l$ **do**
      Remove $x_{(t-|W|)}$ from W

BOTL provides a mechanism to combine models and maximise the impact of transferred models on the target. In presenting BOTL we use the notation detailed in Table 1.

4 BI-DIRECTIONAL ONLINE TRANSFER LEARNING (BOTL)

To utilise knowledge of distinct concepts, BOTL hinges upon a sliding window based concept drift detection algorithm. In this paper we use an adaptation of RePro [28] for regression as the underlying drift detector, however, other concept drift detection algorithms could be used. Although RePro requires some domain expertise to select appropriate parameter values, such as window size and drift threshold, it has been selected as the foundation of BOTL as it provides two unique benefits that outweigh this limitation. Firstly, RePro aims to select a single model that represents the current concept rather than using an ensemble of models to make predictions [27]. If instead an ensemble-based detection algorithm was used, such as DWM [14], the knowledge learnt to represent a single source concept is encompassed across multiple models in the ensemble, therefore all models would need to be transferred. Limiting the number of models needing to be transferred to the target domain reduces the communication and computational overhead of combining knowledge. Secondly, RePro retains a history of previously encountered concepts, preventing the need to re-learn models for recurring concepts [28]. RePro uses a sliding window of data, W, in the target domain to identify if a previously learnt model represents the current concept. If the concept has not previously been encountered a new model is created.

RePro was initially developed specifically for classification, therefore modifications are required for regression tasks, highlighted in Algorithm 1. The original RePro algorithm detects drifts by measuring the target model's classification accuracy across W. To apply RePro to regression, we instead use the R^2 performance of the target model, $f_i^{\mathcal{T}}$, across W. A concept drift is said to have occurred if the R^2 performance drops below a predefined drift threshold, λ_d. RePro traditionally maintains a sliding window of data with a maximum size of W_{max} by discarding one incorrectly classified instance, and all subsequent correctly classified instances when the window is full. In order to encapsulate this behaviour we introduce

Algorithm 2 BOTL: transferring models.

Input: W_{max}, λ_l, λ_d, $\chi^{\mathcal{T}}$, $\mathcal{M}$.
for t=1,2,... **do**
  if $f^{\mathcal{S}}_{j+1}$ available **then**
    Receive $f^{\mathcal{S}}_{j+1}$ from source, add to $\mathcal{M}$
  Receive x_t
  Select $f^{\mathcal{T}}_i$ and detect drifts using Alg. 1, add to $\mathcal{M}$
  $x_t{}'$=$\langle f^{\mathcal{S}}_1(x_t),...,f^{\mathcal{S}}_k(x_t),f^{\mathcal{T}}_i(x_t)\rangle$
  $\hat{y}_t$=$F^{\mathcal{M}}(x_t{}')$ (see Eq. 1)
  Receive y_t
  if $f^{\mathcal{T}}_i$ is a new stable model **then**
    Send $f^{\mathcal{T}}_i$ to all other domains in framework

ϵ-insensitivity through a loss threshold, λ_l, that allows instance $x_{(t-|W|)}$ to be discarded from the window if the predicted value, $\hat{y}'_{(t-|W|)}$, satisfies

$$|\hat{y}'_{(t-|W|)} - y_{(t-|W|)}| \leq \lambda_l.$$

Concept drift detection in BOTL is conducted solely using the locally learnt model. Drift detection should be conducted independently of any knowledge transfer as transfer may enhance the predictive performance across the current window of target data, as such, hindering drift detection.

RePro uses a transition matrix to determine the likelihood of encountering recurring concepts. To prevent the reuse of unstable models that make poor predictions, only those that have been used to make predictions over a predefined number of instances, without causing a drift to be detected, are considered to be stable and added to the transition matrix.

BOTL adopts this notion of a stable model to reduce negative transfer, which occurs when an ineffective model is transferred to the target domain [19]. Unstable models are not transferred, preventing them from negatively impacting the target predictor. A stable model is defined in BOTL to be one that has been used across $2|W|$ instances, while maintaining a performance above the drift threshold, λ_d. Once a model is deemed to be stable, it can be transferred to other domains to aid their respective predictors, as shown in Algorithm 2. This means that model transfer is limited to those that RePro considers to have successfully learnt a concept in their local domain.

Knowledge transfer is achieved in BOTL by communicating models across domains. When model $f^{\mathcal{S}}_j$ is received, it is added to the set of transferred models, $\mathcal{M}$, and combined with the target predictor, $f^{\mathcal{T}}_i$, to enhance the overall predictive performance. Our instantiation of 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 learners could be used in place of OLS.

Each transferred model, $f^{\mathcal{S}}_j \in \mathcal{M}$, and the current target model, $f^{\mathcal{T}}_i$, are used to generate a new window of data. Each sample $x_t{}'$ in the newly generated window of data is of the form $x_t{}' = \{\hat{y}'^{\mathcal{S}_1}_t, \ldots, \hat{y}'^{\mathcal{S}_k}_t, \hat{y}'^{\mathcal{T}_i}_t\}$, where $\hat{y}'^{\mathcal{S}_j}_t$ for all $j = 1, \ldots, k$ is the predicted values of source model $f^{\mathcal{S}}_j$ on instance x_t from the original window of target data, and $\hat{y}'^{\mathcal{T}_i}_t$ is the predicted value of the locally learnt target model, $f^{\mathcal{T}}_i$, selected by RePro for the current concept, c_i. This window of model predictions is used by the OLS meta learner to obtain the overarching predictive function,

$$\begin{aligned}\hat{y}_t &= F^{\mathcal{M}}(x_t{}') \\ &= w_0 + \left(\sum_{j=1}^{k} w_j f^{\mathcal{S}}_j(x_t)\right) + w_{(k+1)} f^{\mathcal{T}}_i(x_t).\end{aligned} \tag{1}$$

Algorithm 3 BOTL-C.II: model culling.

Input: W, λc_{perf}, λc_{MI}, $\mathcal{M}$, $f^{\mathcal{T}}_i$
for $f^{\mathcal{S}}_j \in \mathcal{M}$ **do**
  if $R^2(f^{\mathcal{S}}_j, W) \leq \lambda c_{perf}$ **then**
    Remove $f^{\mathcal{S}}_j$ from $\mathcal{M}$
  for $f^{\mathcal{S}}_k \in \mathcal{M}$ **do**
    if MutualInformation$(f^{\mathcal{S}}_j, f^{\mathcal{S}}_k, W) \geq \lambda c_{MI}$ **then**
      if $R^2(f^{\mathcal{S}}_j, W) \geq R^2(f^{\mathcal{S}}_k, W)$ **then**
        Remove $f^{\mathcal{S}}_k$ from $\mathcal{M}$
      else
        Remove $f^{\mathcal{S}}_j$ from $\mathcal{M}$

4.1 Bi-directional Transfer

BOTL considers the scenario where all domains are online, therefore distinctions between source and target can be disregarded. In this paper, 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 updates it's model set, $\mathcal{M}$, when a concept drift is encountered.

Real-world applications, such as smart home heating system personalisations, may be comprised of a large number of domains, rapidly increasing the number of models to be transferred as the number of domains grow. 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 [5]. To combat this, we introduce culling to BOTL, referred to as BOTL-C.

4.2 Model Culling

Culling transferred models from the model set, $\mathcal{M}$, helps to prevent the OLS meta learner overfitting when a large number of models have been transferred and only a small window of data is available. We introduce two variants of BOTL-C. Firstly BOTL-C.I reduces the number of models available to the OLS meta learner by temporarily removing transferred models from the model set, $\mathcal{M}$, when their R^2 performance across the current window of data drops below a threshold, λc_{perf}. These models are re-added to $\mathcal{M}$ when a concept drift is encountered to enhance predictions of future concepts in the target domain. Although this method of culling is naïve, it can reduce the impact of negative transfer.

In scenarios with high volumes of model transfer, BOTL-C.I requires a high R^2 threshold to sufficiently reduce the number of models to prevent the OLS meta learner overfitting. This can be detrimental as a high proportion of the transferred models containing useful information are culled and no longer available to enhance

the predictive performance of the target learner. To overcome this, BOTL-C.II, outlined in Algorithm 3, evaluates transferred models based on both performance and diversity, metrics commonly used in ensemble pruning literature [31]. Initially, BOTL-C.II reduces the impact of negative transfer by culling models that achieve an R^2 performance less than λc_{perf}, on W. A low λc_{perf} value is preferred, ensuring transferred models containing some useful information are retained. Using a low threshold may not sufficiently reduce the model set, $\mathcal{M}$, to prevent overfitting, therefore a second round of culling is performed based on model diversity. BOTL-C.II measures the diversity between transferred models using mutual information when making predictions on the local window of data. If two transferred models have a mutual information greater than λc_{MI}, BOTL-C.II culls the model that performs worse. A high λc_{MI} should be selected as the window of locally available data is often small, therefore if a complex concept is to be learnt, the target learner may benefit from utilising knowledge transferred from similar concepts. However, if this threshold is too high the model set, $\mathcal{M}$, will not be reduced sufficiently to prevent overfitting.

4.3 Initialisation

The underlying adaptation of RePro requires an initial window of data, W, to create the first predictive model, $f_1^{\mathcal{T}}$. Prior to obtaining this data no predictions can be made using RePro. BOTL allows models transferred from other domains to be used to make predictions during this period. Models transferred are initially weighted equally to obtain,

$$\hat{y}_t = \frac{1}{|\mathcal{M}|} \sum_{j=1}^{|\mathcal{M}|} f_j^{\mathcal{S}}(x_t). \tag{2}$$

Before the first target model, $f_1^{\mathcal{T}}$, has been learnt, and only a small amount of data has been observed, the OLS regressor can create a model, $F^{\mathcal{M}}$, using only source models, $f_j^{\mathcal{S}}$. 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 2 over the entire initial window of data.

The BOTL-C variants help to reduce overfitting within this initial period, however, as the amount of available data is small, all transferred models may have R^2 performances below the culling threshold. In this scenario, both BOTL-C variants select the best three transferred models, regardless of the culling threshold.

5 PERFORMANCE GUARANTEE

Theorem. *Assuming an i.i.d data generating process, BOTL has a loss less than or equal to the model learnt locally using RePro with no knowledge transfer,*

$$\mathcal{L}(f_i^{\mathcal{T}}) \geq \mathcal{L}(F^{\mathcal{M}}), \tag{3}$$

where $\mathcal{L}(f_i^{\mathcal{T}})$ denotes the loss of the local model, $f_i^{\mathcal{T}}$, created using RePro, and $\mathcal{L}(F^{\mathcal{M}})$ is the loss of the OLS meta learner, $F^{\mathcal{M}}$, created using the set of k models transferred from the source, $\{f_1^{\mathcal{S}}, \ldots, f_k^{\mathcal{S}}\}$ and the current target model, $f_i^{\mathcal{T}}$.

Proof. We measure loss over the local window of data, W, using the mean squared error of predictions,

$$\mathcal{L}(\cdot) = \frac{1}{|W|} \sum_{t=1}^{|W|} (y_t - \hat{y}_t)^2, \tag{4}$$

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, $f_i^{\mathcal{T}}$, is used to predict $\hat{y}_t$ for each instance x_t such that $\hat{y}_t = f^{\mathcal{T}}(x_t)$.

BOTL uses the set of models, $\mathcal{M}$, to obtain predictions $\hat{y}_t'^{\mathcal{T}_i}$ and all $\hat{y}_t'^{\mathcal{S}_j}$ for instance x_t, using the locally learnt model, $f_i^{\mathcal{T}}$, and each of the j transferred model, $f_j^{\mathcal{S}} \in \{f_1^{\mathcal{S}}, \ldots, f_k^{\mathcal{S}}\}$, respectively. Predictions are used to create a meta instance, x_t', which the OLS meta learner, $F^{\mathcal{M}}$, uses to obtain an overarching prediction,

$$\begin{aligned} \hat{y}_t &= F^{\mathcal{M}}(x_t') \\ &= F^{\mathcal{M}}\left(\langle f_1^{\mathcal{S}}(x_t), \ldots, f_k^{\mathcal{S}}(x_t), f_i^{\mathcal{T}}(x_t)\rangle\right) \\ &= F^{\mathcal{M}}\left(\langle \hat{y}_t'^{\mathcal{S}_1}, \ldots, \hat{y}_t'^{\mathcal{S}_k}, \hat{y}_t'^{\mathcal{T}_i}\rangle\right), \end{aligned} \tag{5}$$

where

$$F^{\mathcal{M}}(x_t') = w_0 + \sum_{j=1}^{j=k} w_j \hat{y}_t'^{\mathcal{S}_j} + w_{(k+1)} \hat{y}_t'^{\mathcal{T}_i}. \tag{6}$$

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

$$\min_{w_0, \ldots, w_{(k+1)}} \sum_{t=1}^{|W|} \left(y_t - \left(w_0 + \sum_{j=1}^{j=k} w_j \hat{y}_t'^{\mathcal{S}_j} + w_{(k+1)} \hat{y}_t'^{\mathcal{T}_i} \right) \right)^2. \tag{7}$$

$F^{\mathcal{M}}$ is used to make predictions, $\hat{y}_t$, for instance x_t, using Equation 6. Using Equation 4, we can rewrite the loss of $F^{\mathcal{M}}$ as,

$$\mathcal{L}(F^{\mathcal{M}}) = \frac{1}{|W|} \sum_{t=1}^{|W|} \left(y_t - \left(w_0 + \sum_{j=1}^{j=k} w_j \hat{y}_t'^{\mathcal{S}_j} + w_{(k+1)} \hat{y}_t'^{\mathcal{T}_i} \right) \right)^2. \tag{8}$$

If we constrain the optimisation problem in Equation 7 to obtain the meta learner $F^{\mathcal{M}^*}$ by fixing the weights, w_a, such that the weight associated with the locally learnt model $f_i^{\mathcal{T}}$ is 1, while all others are 0, we obtain a meta model of the form,

$$F^{\mathcal{M}^*}(x_t') = \left(0 + \sum_{j=1}^{j=k} 0 \hat{y}_t'^{\mathcal{S}_j} + 1 \hat{y}_t'^{\mathcal{T}_i} \right), \tag{9}$$

giving the loss function

$$\mathcal{L}(F^{\mathcal{M}^*}) = \frac{1}{|W|} \sum_{t=1}^{|W|} \left(y_t - \hat{y}_t'^{\mathcal{T}} \right)^2, \tag{10}$$

equivalent to only using the locally learnt model, $\mathcal{L}(F^{\mathcal{M}^*}) = \mathcal{L}(f_i^{\mathcal{T}})$. As the optimisation problem in Equation 7 is convex,

$$\mathcal{L}(F^{\mathcal{M}^*}) \geq \mathcal{L}(F^{\mathcal{M}}). \tag{11}$$

Finally, as the constrained optimisation problem in Equation 9 is equivalent to using only the locally learnt model, $f_i^{\mathcal{T}}$, the loss of BOTL is less than or equal to the loss of the locally learnt model. □

6 EXPERIMENTAL SET-UP

Many benchmark datasets have been created to evaluate concept drift detection algorithms [12, 22, 23], however, most are categorically labelled. In order to evaluate BOTL in a regression setting, we present a modification to the benchmark drifting hyperplane dataset [15]. Additionally, a simulation of a smart home heating system was created using data from a UK weather station to derive desired heating temperatures for a user. The use of such data enables BOTL to be evaluated on data streams containing drifts that are more typical within real-world environments. Finally we evaluate the performance of BOTL using a following distance dataset[2], created from vehicular data, and used to predict the time to the vehicle it is following.

6.1 Drifting Hyperplane

For this benchmark data generator, an instance at time t, x_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. For example, an alternative concept, c_b, may be represented by function $y_t = (x_{t_u} + x_{t_v} + x_{t_w})/3$, where $\{p, q, r\} \neq \{u, v, w\}$ such that $c_a \neq c_b$. We introduce uniform noise, ± 0.1, by modifying y_t for each instance x_t with probability 0.1.

A variety of drift types have been synthesised in this generator including sudden drift, gradual drift and recurring drifts. A sudden drift from concept c_a to concept c_b is encountered immediately between time steps t and $(t+1)$ by changing the underlying function used to create y_t and $y_{(t+1)}$. A gradual drift from concept c_a to c_b occurs 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 to determine their 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 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 we achieve conceptual equivalence, $c_c = c_a$.

6.2 Heating Simulation

A simulation of a smart home heating system was created, deriving the desired room temperature of a user. Heating temperatures were derived using weather data collected from a weather station in Birmingham, UK, from 2014 to 2016. This dataset contained rainfall, temperature and sunrise patterns, which were combined with a schedule, obtained from sampling an individual's pattern of life, to determine when the heating system should be engaged. The schedule was synthesised to vary desired temperatures based on time of day, day of week, and external weather conditions, creating complex concepts. To create multiple domains, weather data was sampled from overlapping time periods and used as input to the synthesised schedule to determine the desired heating temperatures. Due to the dependencies on weather data, each stream was subject to large amounts of noise. Concept drifts were introduced manually by changing the schedule, however, drifts also occurred naturally due to changing weather conditions. By sampling weather data from overlapping time periods, and due to the seasonality of weather, data streams follow similar trends, ensuring predictive performance can benefit from knowledge transfer. By using complex concepts, dependent on noisy data, the evaluation of BOTL on this data is more indicative of what is achievable when used in real-world environments.

6.3 Following Distance

This dataset uses a vehicle's following distance and speed to calculate TTC when following a vehicle. Vehicle telemetry data such as speed, gear position, brake pressure, throttle position and indicator status, alongside sensory data that infer external conditions, such as temperature, headlight status, and windscreen wiper status, were recorded at a sample rate of 1Hz. Additionally, some 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. Vehicle telemetry and environmental data can be used to predict TTC and used to personalise vehicle functionalities such as ACC by identifying the preferred following distance, reflecting current driving conditions. Data was collected from 4 drivers for 17 journeys which varied in duration, collection time and route. Each journey is considered to be an independent domain and BOTL enables knowledge to be learnt and transferred across journeys and between drivers. 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.

7 EXPERIMENTAL RESULTS

We compared BOTL against existing algorithms designed to make predictions in online data streams, namely RePro [27] and GOTL [9], using the drifting hyperplane, heating simulation and following distance datasets. BOTL is model agnostic, however, in order to make comparisons between BOTL and existing techniques, all implementations used an ϵ-insensitive Support Vector Regressor (SVR).

RePro is used to determine a baseline performance threshold, obtained when no knowledge is transferred [27]. The RePro implementation defines parameters including the maximum window size, W_{max}, loss threshold, λ_l, and drift threshold, λ_d, dependant upon the data stream and complexity of concepts.

GOTL was designed to learn from an offline source, however, as we are considering the implications of both domains being online, we used RePro to detect individual concepts in the source. This is necessary as many online applications cannot retain an entire history of data, preventing a single model from being learnt. We used RePro to identify the model that had been used in the source for the largest proportion of the data stream so is considered the most stable. GOTL transferred this model from the source domain to the target to enhance the effectiveness of the target predictor. A small step-size, $\delta = 0.025$, was chosen, as suggested by Grubinger *et al.* [8], which slowly modified the weights used to combine source and target models.

[2]Data generators, sample vehicle data, and reproducibility documentation are available at: https://github.com/hmckay/BOTL

Table 2: Drifting Hyperplanes: predictions using RePro, GOTL, BOTL and BOTL-C variants for five sudden and five gradual drifting domains, where * indicates $p < 0.01$ in comparison to RePro and GOTL, and bold indicates the highest R^2 performance.

	Sudden Drift		Gradual Drift	
	R^2	RMSE	R^2	RMSE
RePro	0.950 (±0.001)	0.058	0.855 (±0.001)	0.070
GOTL	0.928 (±0.001)	0.069	0.825 (±0.001)	0.076
BOTL	***0.958** (±0.001)	0.053	***0.893** (±0.001)	0.060
BOTL-C.I	*0.957 (±0.001)	0.054	*0.886 (±0.003)	0.062
BOTL-C.II	*0.956 (±0.001)	0.054	*0.889 (±0.004)	0.061

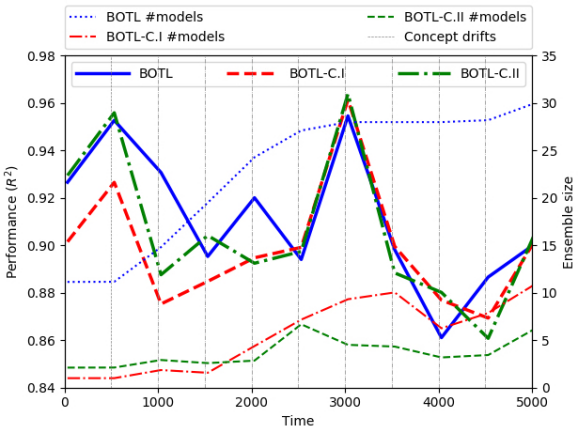


Figure 1: Drifting Hyperplanes: R^2 performance and number of models used by BOTL and BOTL-C variants.

BOTL combines knowledge via the OLS meta learner therefore no additional parameters are required, however the BOTL-C culling parameters must be defined. We set $\lambda c_{perf} = 0$ for BOTL-C.I, thereby discarding models that performed worse than the average predictor ($R^2 < 0$). For BOTL-C.II, we used $\lambda c_{perf} = 0.2$, such that poorly performing models were culled more aggressively, and $\lambda c_{MI} = 0.95$, such that two models with extremely high mutual information were not both retained by the meta learner. This parameter value was high to ensure knowledge of similar concepts were retained when predicting complex concepts.

When evaluating BOTL and BOTL-C variants, 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. For RePro, all data streams were learnt from independently without knowledge transfer. When evaluating GOTL, experiments were conducted such that each data stream was paired with every other data stream as source and target domains respectively. Due to only transferring the most stable model when using GOTL, learning in the target domain only commenced once learning in the source domain had completed such that the most stable source model could be identified and transferred to the target domain.

7.1 Drifting Hyperplane

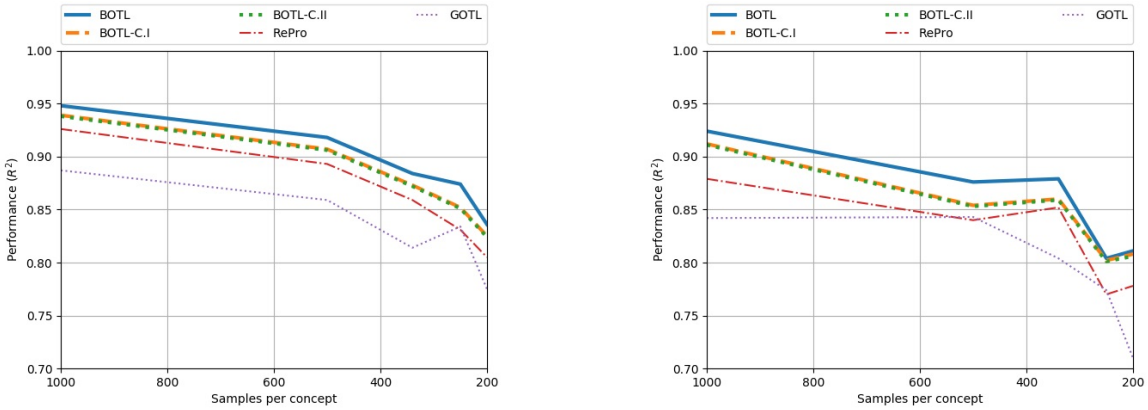
We considered the effectiveness of BOTL on synthetic data created using the drifting hyperplane data generator containing two types of drift: sudden and gradual. Five data streams of 10,000 instances were created for each drift type with drifts encountered every 500 time steps. Sudden drifts occurred immediately, however, gradual drifts occurred over a period of 100 time steps. Each data stream shared at most three concepts with another domain, ensuring some models transferred were useful to the target learner, while others were not. Sudden and gradual drifting data streams were separated such that transfer occurred only between domains of the same drift type. RePro parameters $W_{max} = 30$, $\lambda_l = 0.05$, and $\lambda_d = 0.6$ were used across all frameworks.

The results in Table 2 indicate that, although GOTL transferred the most stable model, a performance increase was not achieved over RePro. Although GOTL did not benefit from knowledge transfer, BOTL was able to outperform both GOTL and RePro with statistical t-tests achieving p-values < 0.01, highlighting the importance of acknowledging the online nature of the source domain when transferring knowledge.

The performance increase observed over GOTL can be attributed to the availability of all source models in the target domain. Additionally, GOTL's step-wise weighting mechanism prevents the influence of a model changing drastically over a small period of time. This means 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 predictor with instantaneous effect.

The performances of BOTL-C variants were also significantly better than RePro and GOTL, obtaining t-test values of $p < 0.01$, however, they performed slightly worse than BOTL. Figure 1 highlights the aggressive nature of the culling techniques used by BOTL-C.I and BOTL-C.II. It also shows BOTL used at least four times more models than BOTL-C variants and highlights correlations between the number of models used and performance. When the number of models used was small, the predictive performance of BOTL-C variants decreased. This performance decrease can be attributed to the aggressive nature of these culling mechanisms. Culling based on model performance alone prohibited the inclusion of a diverse set of models, reducing the overarching predictive performance of the meta learner. When BOTL-C variants retained a larger proportion of the transferred models a performance similar to BOTL was achieved.

We also considered the performance of BOTL on data streams with varying concept durations, showing that BOTL is more effective than existing approaches, irrespective of the duration of each concept. Additional data streams of 10,000 instances were created with different time periods between drifts. Figures 2a and 2b show the performance of BOTL, RePro and GOTL in these data streams. Since BOTL variants are underpinned by RePro, their performance dropped similarly as drifts occur more frequently. This is because RePro must observe a local drop in performance over the sliding window of data to identify drifts, therefore a higher proportion



(a) Sudden concept drift.

(b) Gradual concept drift, with drift durations of 100 time steps.

Figure 2: Drifting Hyperplanes: performance of RePro, GOTL and BOTL variants with sudden and gradual concept drifts encountered at different frequencies.

Table 3: Heating Simulations: predicting desired heating temperatures across five domains using RePro, GOTL, BOTL and BOTL-C variants, where * indicates $p < 0.01$ in comparison to RePro and GOTL, and bold indicates the highest R^2 performance.

	R^2	RMSE
RePro	0.607 (±0.003)	2.601 (±0.015)
GOTL	0.709 (±0.003)	2.231 (±0.009)
BOTL	***0.786** (±0.009)	1.914 (±0.042)
BOTL-C.I	*0.779 (±0.010)	1.946 (±0.044)
BOTL-C.II	*0.744 (±0.008)	2.102 (±0.036)

Table 4: Following Distances: predicting TTC across seven domains using RePro, GOTL, BOTL and BOTL-C variants, where * indicates $p < 0.01$ in comparison to RePro and GOTL, and bold indicates the highest R^2 performance.

	R^2	RMSE
RePro	0.497 (±0.002)	0.636 (±0.002)
GOTL	0.619 (±0.001)	0.554 (±0.003)
BOTL	0.665 (±0.002)	0.524(±0.002)
BOTL-C.I	0.666 (±0.002)	0.523(±0.002)
BOTL-C.II	***0.673** (±0.001)	0.518(±0.002)

of the data stream has poor predictions as the number of concept drifts increases. Despite the decrease in performance, BOTL withstood the impact of frequent drifts more readily than RePro and GOTL as BOTL uses knowledge learnt in other domains to aid the target predictor prior to a new model being learnt by RePro. While GOTL also uses transferred knowledge, its weighting mechanism prevented effective use of this knowledge at drift points.

7.2 Heating Simulation

We chose RePro parameters $\lambda_l = 0.5$, $\lambda_d = 0.6$, and $W_{max} = 700$, creating a sliding window that encapsulated approximately two weeks of heating and weather data, to analyse frameworks on this data. These parameters meant that instances were removed from the start of the sliding window when predictions were made within $\pm 0.5^{\circ}$C of the desired heating temperature, and drifts were detected when the R^2 performance of the target model on the current window of data dropped below 0.6. The concepts to be learnt in these domains were more complex than those in the drifting hyperplane data streams, causing lower R^2 performances overall.

The addition of knowledge transfer, using GOTL and BOTL, provided a significant increase in performance in comparison to RePro with no transfer, as shown in Table 3. GOTL, BOTL and BOTL-C variants performed better than RePro with statistical t-test p-values of $p < 0.01$. The ability to transfer knowledge was more advantageous in comparison to the drifting hyperplane setting because concepts were more complex, preventing RePro from building effective models on the window of available data. This meant the knowledge transferred helped boost the performance of the target predictor, even when only a single model was transferred using GOTL. Transferring multiple models provided a significant benefit as BOTL performed better than GOTL with a t-test p-value < 0.01.

7.3 Following Distance

Finally we evaluated BOTL on real-world data using the following distance dataset, where the task was to predict TTC. Due to the real-world nature of this data, concept drifts occur frequently and data is noisy. The RePro parameters $\lambda_l = 0.1$, $\lambda_d = 0.5$ and $W_{max} = 80$ were used, encapsulating 80 seconds of vehicle data. This meant that instances were removed from the start of the sliding window if predictions were made within ±0.1 seconds of the recorded TTC value, and drifts were detected when the R^2 performance of the target model on the current window of data dropped below 0.6.

Table 4 shows the performance of RePro, GOTL and BOTL variants across seven data streams. These results highlight GOTL was less suitable when the relationship between source and target concepts were unknown. All variants of BOTL and BOTL-C performed better than RePro and GOTL, with BOTL-C variants slightly outperforming BOTL. BOTL-C.II achieved statistical t-test p-values of $p < 0.01$ when compared to both RePro and GOTL. This can be attributed to the number of domains in the framework, indicating

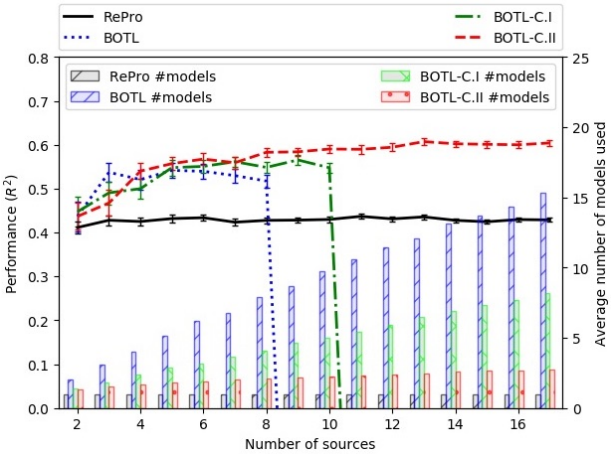


Figure 3: Following Distances: performance (with standard error) and number of models used by RePro, BOTL and BOTL-C variants as the number of domains increase.

large numbers of transferred models caused the OLS meta learner to overfit the local window of data.

To investigate scalability, Figure 3 displays the average R^2 performance per domain and number of models used for predictions using RePro, BOTL and BOTL-C variants as the number of domains increased. For settings with a small number of domains, BOTL and BOTL-C variants performed similarly, both outperforming the baseline RePro algorithm. As the number of domains expanded, and the number of models transferred increased, the performance of BOTL dropped below RePro. This can be attributed to the OLS meta learner overfitting the small window of local data when the number of models in the ensemble was large. Culling using the performance of transferred models alone (BOTL-C.I) enabled a larger number of domains to be used, however, cannot be considered scalable as the performance of BOTL-C.I dropped below that of RePro when more domains were added. BOTL-C.II culled more aggressively, using diversity alongside performance, ensuring enough beneficial knowledge was retained to enhance the target learner's performance, while minimising negative transfer and preventing the OLS meta learner overfitting the small window of locally available data.

The results indicate that the ability to consider both source and target domains to be online is beneficial. In doing so, the number of transferred models greatly increases, requiring culling mechanisms, particularly when used in noisy real-world data streams, to retain the benefit of transferring knowledge between domains.

8 CONCLUSION

Online domains that must learn complex models often have limited data availability, and are hindered by the presence of concept drift. We have presented the BOTL framework, and two BOTL-C variants, that enable knowledge to be transferred across online domains to aid the target learner. By using RePro as the underlying concept drift detection algorithm we ensured effective models were learnt from the available data. We enhanced predictive performance by combining knowledge transferred from other online domains using an OLS meta learner, enabling additional knowledge to be used to minimise the error of the overarching prediction.

In this paper, RePro was chosen as the underlying concept drift detection algorithm. Although RePro requires some domain expertise to identify appropriate parameter values, such as window size and drift threshold, it's ability to retain a history of models to prevent relearning recurring concepts helped to reduce the number of models transferred between domains and therefore allowed more domains to be included in the framework before the OLS meta learner suffered from overfitting. However, in real-world environments with many domains, the number of models transferred may need to be reduced further. BOTL-C variants achieved this using common ensemble pruning strategies. These pruning strategies also required culling parameter values to be specified. To overcome the need to specify these parameters we will investigate the use of task relatedness to identify similar concepts across domains without parameterised thresholds in future work. This will reduce the dependency on domain expertise and will allow BOTL to be used for applications that require scalability to larger numbers of domains. Additionally, BOTL considers only the homogeneous setting, therefore a natural extension is to incorporate domain adaptation to enable knowledge transfer across heterogeneous domains using the BOTL framework.

ACKNOWLEDGMENTS

This work was supported by the UK EPSRC and Jaguar Land Rover under the iCASE scheme.

REFERENCES

[1] Andrew Arnold, Ramesh Nallapati, and William W. Cohen. 2007. A Comparative Study of Methods for Transductive Transfer Learning. In *Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007)*. 77–82.
[2] Albert Bifet and Ricard Gavalda. 2007. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining*. SIAM, 443–448. https://doi.org/10.1137/1.9781611972771.42
[3] Hal Daume III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research* 26 (2006), 101–126. https://doi.org/10.1613/jair.1872
[4] Bo Dong, Yifan Li, Yang Gao, Ahsanul Haque, Latifur Khan, and Mohammad M. Masud. 2017. Multistream regression with asynchronous concept drift detection. In *2017 IEEE International Conference on Big Data*. 596–605. https://doi.org/10.1109/BigData.2017.8257975
[5] Jerome H. Friedman. 1997. On Bias, Variance, 0/1—Loss, and the Curse-of-Dimensionality. *Data Mining and Knowledge Discovery* 1, 1 (Mar 1997), 55–77. https://doi.org/10.1023/A:1009778005914
[6] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A Survey on Concept Drift Adaptation. *ACM Comput. Surv.* 46, 4, Article 44 (2014), 37 pages. https://doi.org/10.1145/2523813
[7] Liang Ge, Jing Gao, and Aidong Zhang. 2013. OMS-TL: A Framework of Online Multiple Source Transfer Learning. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management (CIKM '13)*. ACM, 2423–2428. https://doi.org/10.1145/2505515.2505603
[8] Thomas Grubinger, Georgios Chasparis, and Thomas Natschläger. 2016. Online transfer learning for climate control in residential buildings. In *Proceedings of the 5th Annual European Control Conference (ECC 2016)*. 1183–1188. https://doi.org/10.1109/ECC.2016.7810450
[9] Thomas Grubinger, Georgios Chasparis, and Thomas Natschläger. 2017. Generalized online transfer learning for climate control in residential buildings. *Energy and Buildings* 139 (2017), 63 – 71. https://doi.org/10.1016/j.enbuild.2016.12.074
[10] Ahsanul Haque, Hemeng Tao, Swarup Chandra, Jie Liu, and Latifur Khan. 2018. A framework for multistream regression with direct density ratio estimation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
[11] T. Ryan Hoens, Robi. Polikar, and Nitesh. V. Chawla. 2012. Learning from streaming data with concept drift and imbalance: an overview. *Progress in Artificial Intelligence* 1, 1 (2012), 89–101. https://doi.org/10.1007/s13748-011-0008-0
[12] Geoff Hulten, Laurie Spencer, and Pedro Domingos. 2001. Mining Time-changing Data Streams. In *Proceedings of the Seventh ACM SIGKDD International Conference*

on Knowledge Discovery and Data Mining (KDD '01). ACM, 97–106. https://doi.org/10.1145/502512.502529

[13] Mark G. Kelly, David J. Hand, and Niall M. Adams. 1999. The Impact of Changing Populations on Classifier Performance. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '99)*. ACM, 367–371. https://doi.org/10.1145/312129.312285

[14] Jeremy Z. Kolter and Marcus A. Maloof. 2003. Dynamic weighted majority: a new ensemble method for tracking concept drift. In *Third IEEE International Conference on Data Mining*. 123–130. https://doi.org/10.1109/ICDM.2003.1250911

[15] Jeremy Z. Kolter and Marcus A. Maloof. 2005. Using Additive Expert Ensembles to Cope with Concept Drift. In *Proceedings of the 22nd International Conference on Machine Learning (ICML '05)*. ACM, 449–456. https://doi.org/10.1145/1102351.1102408

[16] Guangxia. Li, Steven CH Hoi, Kuiyu Chang, Wenting Liu, and Ramesh Jain. 2014. Collaborative online multitask learning. *IEEE Transactions on Knowledge and Data Engineering* 26, 8 (2014), 1866–1876. https://doi.org/10.1109/TKDE.2013.139

[17] Keerthiram Murugesan and Jamie Carbonell. 2017. Multi-task multiple kernel relationship learning. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 687–695.

[18] Jianhan Pan, Xuegang Hu, Peipei Li, Huizong Li, Wei He, Yuhong Zhang, and Yaojin Lin. 2016. Domain adaptation via Multi-Layer Transfer Learning. *Neurocomputing* 190 (2016), 10–24. https://doi.org/10.1016/j.neucom.2015.12.097

[19] Sinno J. Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering* 22, 10 (2010), 1345–1359. https://doi.org/10.1109/TKDE.2009.191

[20] Paul Ruvolo and Eric Eaton. 2013. Active Task Selection for Lifelong Machine Learning. In *AAAI*.

[21] Avishek Saha, Piyush Rai, Hal Daumã, Suresh Venkatasubramanian, et al. 2011. Online learning of multiple tasks and their relationships. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*. 643–651.

[22] Jeffery C. Schlimmer and Richard H. Granger. 1986. Incremental Learning from Noisy Data. *Machine Learning* 1, 3 (1986), 317–354. https://doi.org/10.1007/BF00116895

[23] W. Nick Street and YongSeog Kim. 2001. 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)*. ACM, 377–382. https://doi.org/10.1145/502512.502568

[24] Alexey Tsymbal. 2004. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin* 106 (2004).

[25] Qingyao Wu, Hanrui Wu, Xiaoming Zhou, Mingkui Tan, Yonghui Xu, Yuguang Yan, and Tianyong Hao. 2017. Online Transfer Learning with Multiple Homogeneous or Heterogeneous Sources. *IEEE Transactions on Knowledge and Data Engineering* 29, 7 (2017), 1494–1507. https://doi.org/10.1109/TKDE.2017.2685597

[26] Yuguang Yan, Qingyao Wu, Mingkui Tan, Michael K. Ng, Huaqing Min, and Ivor W. Tsang. 2017. Online Heterogeneous Transfer by Hedge Ensemble of Offline and Online Decisions. *IEEE Transactions on Neural Networks and Learning Systems* PP, 99 (2017), 1–12. https://doi.org/10.1109/TNNLS.2017.2751102

[27] Ying Yang, Xindong Wu, and Xingquan Zhu. 2005. 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)*. ACM, 710–715. https://doi.org/10.1145/1081870.1081961

[28] Ying Yang, Xindong Wu, and Xingquan Zhu. 2006. Mining in Anticipation for Concept Change: Proactive-Reactive Prediction in Data Streams. *Data Mining and Knowledge Discovery* 13, 3 (2006), 261–289. https://doi.org/10.1145/1081870.1081961

[29] Haibo Yin and Yun-an Yang. 2017. Online transfer learning with extreme learning machine. In *AIP Conference Proceedings*, Vol. 1839. AIP Publishing, 020199. https://doi.org/10.1063/1.4982564

[30] Peilin Zhao and Steven C. Hoi. 2010. OTL: A framework of online transfer learning. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. 1231–1238.

[31] Zhi-Hua Zhou. 2012. *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC.