DDG-DA: Data Distribution Generation for Predictable Concept Drift Adaptation

Wendi Li[1,2*], **Xiao Yang**[2*], **Weiqing Liu**[2], **Yingce Xia**[2], **Jiang Bian**[2]

[1] University of Wisconsin–Madison
[2] Microsoft Research
Wendi.Li@wisc.edu, {Xiao.Yang, Weiqing.Liu, Yingce.Xia, Jiang.Bian}@microsoft.com

Abstract

In many real-world scenarios, we often deal with streaming data that is sequentially collected over time. Due to the non-stationary nature of the environment, the streaming data distribution may change in unpredictable ways, which is known as concept drift. To handle concept drift, previous methods first detect when/where the concept drift happens and then adapt models to fit the distribution of the latest data. However, there are still many cases that some underlying factors of environment evolution are predictable, making it possible to model the future concept drift trend of the streaming data, while such cases are not fully explored in previous work. In this paper, we propose a novel method DDG-DA, that can effectively forecast the evolution of data distribution and improve the performance of models. Specifically, we first train a predictor to estimate the future data distribution, then leverage it to generate training samples, and finally train models on the generated data. We conduct experiments on three real-world tasks (forecasting on stock price trend, electricity load and solar irradiance) and obtain significant improvement on multiple widely-used models.

Introduction

Machine learning algorithms have been widely applied to many real-world applications. One of the foundations of its success lies in assuming that the data is independent and identically distributed (briefly, the i.i.d. assumption). However, such an assumption is not always true for those learning tasks under broadly existing scenarios with streaming data. In a typical learning task over streaming data, as shown in Figure 1, the data comes in a sequential mode, meaning that, at any timestamp t, the learning task can only observe the new coming information, i.e., $\text{Data}^{(t)}$, in addition to historical ones, i.e., $\text{Data}^{(t-1)}$, $\text{Data}^{(t-2)}$, $\cdots$, $\text{Data}^{(1)}$. The goal of the task is usually to predict a certain target related to $\text{Data}^{(t+1)}$. Due to the non-stationary nature of the real-world environment, the data distribution could keep changing with continuous data streaming. Such a phenomenon/problem is called *concept drift* (Lu et al. 2018), where the basic assumption is that concept drift happens unexpectedly and it is unpredictable (Gama et al. 2014) for streaming data.

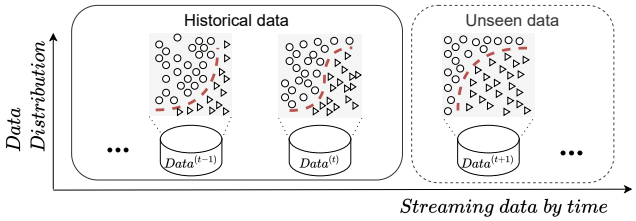


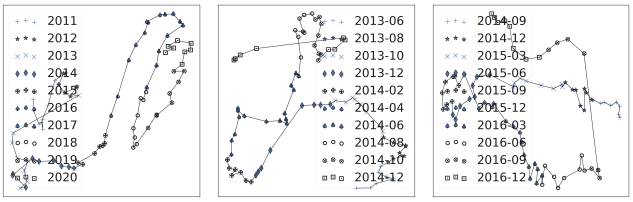
Figure 1: An example of concept drifts on streaming data. Triangle and circle represent two classes of data. Data comes like a stream. The data distribution changes over time.

To handle concept drift, previous studies usually leverage a two-step approach. Particularly, the first step is detecting the occurrence of concept drift in the streaming data, followed by the second step, if concept drift does occur, which adapts the model with new coming data by training a new model purely with latest data or making an ensemble between the new model with the current one, or fine-tuning the current model with latest data (Lu et al. 2018). Most of these studies share the same assumption that the latest data contains more useful information w.r.t. the upcoming streaming data (Zhao, Cai, and Zhou 2020), and thus the detection of concept drift and following model adaptation is mainly applied to the latest data.

Therefore, existing methods for handling concept drift will detect concept drift on the latest arrived data $Data^{(t)}$ at timestamp t and adapt the forecasting model accordingly. The concept drift continues, and the adapted model on $Data^{(t)}$ will be used on unseen streaming data in the future (e.g., $Data^{(t+1)}$). The previous model adaptation has a one-step delay to the concept drift of upcoming streaming data, which means a new concept drift has occurred between timestamp t and $t+1$.

Nevertheless, apart from detecting unexpectedly occurred concept drift, most of the existing studies paid less attention to the scenarios that concept drift evolves in a gradual nonrandom way, which are in fact more common in streaming data. In Figure 2, we visualize the trace of concept drift in streaming data on three real-world tasks: forecasting on

*These authors contributed equally.



(a) Stock price (b) Electricity load (c) Solar irradiance

Figure 2: The visualization of concept drifts over time; though the concept drifts unexpectedly at some time, it drifts in a gradual nonrandom way most of the time. If the concept drifts are all sudden and do not follow any pattern, it may not be handled. However, in real-world scenarios, as we have shown, most concept drifts have a nonrandom trend rather than completely random.

stock price trend, electricity load and solar irradiance. Each point indicates the data distribution (a.k.a. concept) represented by several features in a specific time period and is plotted on the figure with t-SNE (Van der Maaten and Hinton 2008). To better visualize the concept drift, we connect points contiguous in time to a line. From this figure, we can find that unexpected drastic concept drifts only occur occasionally. On the other hand, most of the points that are contiguous in time are spatially adjacent to each other and form a nonrandom trend, demonstrating a strong indication that a large portion of concept drifts do evolve in a gradual nonrandom way. More importantly, such gradual nonrandom concept drifts are in some sense predictable since patterns exist in the concept drift trend.

In this paper, we focus on predictable concept drift by forecasting the future data distribution. We propose a novel method DDG-DA[1] to predict the data distribution of the next time-step sequentially, such that the model of the downstream learning task can be trained on the data sample from the predicted distribution instead of catching up with the latest concept drift only. In practice, DDG-DA is designed as a dynamic data generator that can create sample data from previously observed data by following predicted future data distribution. In other words, DDG-DA generates the resampling probability of each historical data sample to construct the future data distribution in estimation. However, it is quite challenging in reality to train this data generator to maximize the similarity between the predicted data distribution (represented by weighted resampling on historical data) and the ground truth future data distribution (represented by data in the future). To address this challenge, we propose to first represent a data distribution by learning a model under this data distribution and then create a differentiable distribution distance to train the data generator. To verify the effectiveness of this approach, we also conduct a thorough theoretical analysis to prove the equivalence between traditional distribution distance, e.g. KL-divergence, and the proposed differentiable distribution distance. Furthermore, it is worth noting that our proposed new approach is quite efficient since the distribution distance is differentiable and could be easy for optimization.

Extensive experiments have been conducted on a variety of popular streaming data scenarios with multiple widely-used learning models to evaluate the effectiveness of our proposed method DDG-DA. Experimental results have shown that DDG-DA could enhance the performance of learning tasks in all these scenarios. Further analysis also reveals that DDG-DA can successfully predict the future data distribution in the circumstance of predictable concept drift. The predicted data distribution could benefit the learning task by proactively adapting to drifting concepts. The code of DDG-DA is open-source on Github[2].

To sum up, our contributions include

- To the best of our knowledge, DDG-DA is the first method to model the evolving of data distribution in *predictable concept drift* scenarios.
- We create a differentiable distribution distance and provide theoretical analysis to prove its equivalence to KL-divergence distribution distance.
- We conduct extensive experiments on different real-world concept-drift-predictable scenarios to show that DDG-DA outperforms SOTA concept drift adaptation methods by training models on resampled dataset(predicted data distribution).

Background and Related Work

Streaming Data and Concept Drift

General Concept Drift In a large number of practical scenarios, data are collected over time sequentially. The streaming data could be formalized as $\boldsymbol{X} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(T)}\}$, where each element $\boldsymbol{x}^{(t)} \in \mathbb{R}^m$ pertaining to $\boldsymbol{X}$ is an array of m values such that $\boldsymbol{x}^{(t)} = \{x_1, x_2, \ldots, x_m\}$. Each one of the m scalar values corresponds to the input variable observed at time t. Given a target sequence $\boldsymbol{y} = \{y^{(1)}, y^{(2)}, \ldots, y^{(T)}\}$ corresponding to $\boldsymbol{X}$, algorithms are designed to build the model on historical data $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{t}$ and forecast $\boldsymbol{y}$ on the future data $D_{test}^{(t)} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=t+1}^{t+\tau}$ at each timestamp t. Data are drawn from a joint distribution $(\boldsymbol{x}^{(t)}, y^{(t)}) \sim p_t(\boldsymbol{x}, y)$. The goal of algorithms is to build a model to make accurate predictions on unseen data $D_{test}^{(t)}$ in the future. Due to the non-stationary nature of the environment, the joint distribution p_t is not static and changes over time, which is a well-known problem named *concept drift*. Formally, the concept drift between two timestamps t and $t+1$ can be defined as $\exists \boldsymbol{x} : p_t(\boldsymbol{x}, y) \neq p_{t+1}(\boldsymbol{x}, y)$.

To improve the forecasting accuracy in the future, adapting models to accommodate the evolving data distribution is necessary, which is the focused problem in this paper. It can be formalized as

$$\min_{f^{(t)}, f^{(t+1)}, \ldots, f^{(t+\tau)}} \sum_{i=t}^{t+\tau} l\left(f^{(i)}(\boldsymbol{x}^{(i)}), y^{(i)}\right)$$

[1]The code is available at https://github.com/Microsoft/qlib/tree/main/examples/benchmarks_dynamic/DDG-DA

[2]https://github.com/Microsoft/qlib

where $f^{(t)}$ is learned from the training data in the previous stream $D^{(t)}_{train} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=t-k}^{t-1}$ with window size k and will be adapted continuously to minimize the target metric $loss$ in given time period $[t, t+\tau]$. In many practical scenarios, the streaming data come continually over time and might never end. Accommodating such an amount of data in memory will be infeasible. Therefore only a limited memory size k is allowed for an online setting. Data in the memory could be used for model adaptation.

The Categorization of Concept Drifts and Our Scope The concept drifts can be categorized into different types when based on different criteria (Ren et al. 2018). (Webb et al. 2016) provides quantitative measures to categorize concept drifts. For instance, a concept drift can be either *abrupt* or *gradual* according to its changing speed. It can also be *recurring* or *non-recurring* depending on whether similar concepts have appeared in the data stream.

Our method aims to handle scenarios with *predictable* concept drifts defined in (Minku, White, and Yao 2009), which refers to the concept drifts that follow a pattern rather than completely random due to the seasonality and reoccurring contexts. As (Ren et al. 2018; Žliobaitė, Pechenizkiy, and Gama 2016) state, the concept drifts in real-world evolving data are often caused by changes of the hidden contexts. Taking stock market, electricity load and solar irradiance as examples, because they are affected by some cyclic elements (time of the day, day of the week, month of the year, etc.), all of them have periodic, recurrent contexts. Abrupt drifts are rare due to nonrandom time-varying contexts, which is shown in Figure 2. Such reasons give these concept drifts nonrandom trends rather than completely unpredictable ones. (Khamassi et al. 2018; Webb et al. 2016) demonstrate that some characteristics of concept drift can be measured and predicted.

Related Work

Concept drift is a well-known topic in recent research works (Lu et al. 2018; Gama et al. 2014). The concept drift research works usually focus on streaming data, which come in a streaming form and are usually collected in non-stationary environments as Figure 1 shows. In the setting of most of the previous works, concept drift happens unexpectedly and is unpredictable. Without proactively predicting concept drift, these concept drift adaptation methods usually assume that the newer the data, the smaller the distribution gap between it and the future coming data. For instance, forgetting-based methods (Koychev 2000; Klinkenberg 2004) decay sample weights according to their age. When new labelled data are available, the model adaptation method detects concept drift and optionally updates the current model. However, the adapted new model will be used in the future, and the concept drift continues, which could make the model adaptation fail on unseen data in the future.

Previous Methods to Handle Predictable Concept Drift Predictable concept drift is a scenario that the evolving concept drift is predictable to a certain extent **before concept drift happens**. In this scenario, methods are encouraged to predict and adapt to the new concept before drift happens. Proactively adapting models to the future joint distribution $p(\boldsymbol{x}, y)$ is the key to catching up with the concept evolution. Though most previous works assume that concept drift happens unexpectedly and is unpredictable. There are still a few of them noticing that concept drift is predictable (Minku, White, and Yao 2009; Khamassi et al. 2018) under certain additional assumptions. Some case studies in real-world scenarios demonstrate that the prediction of concept drift is possible (Gama et al. 2014). (Yang, Wu, and Zhu 2005) try to deal with recurrent concept drifts. They use a small trigger window to detect the concept drift. Based on the small amount of data in the window, they build a model to predict the new concept **after concept drift happens**. It is designed for the traditional concept drift setting instead of predictable concept drift. DDG-DA tries to predict and adapt to the new concept **before it happens**. As far as we know, we are the first method designed for predictable concept drift.

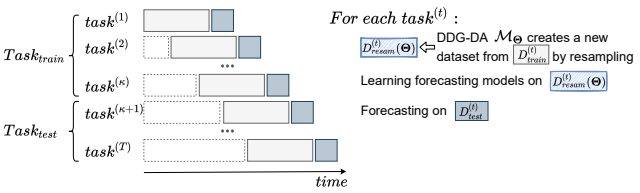


Figure 3: Training data (historical data) and test data (recent unseen data) change over time; the objective of each task is to improve the forecasting performance on test data.

Types of Methods to Handle Concept Drift According to the learning mode, methods can be categorized into *retraining* (Bach and Maloof 2008) and *incremental adaptation* (Alippi and Roveri 2008). Retraining methods will discard the current model and learn a new model on a limited size of data. Incremental adaptation methods will update the current model incrementally.

From another perspective, most adaptive learning methods are designed for specific models. In lots of practical forecasting tasks, customized models will be designed specifically. These model-specific adaptive learning methods can only be carried out on specific forecasting models and are not general solutions, which narrows their practicability in real applications. Only a few adaptation methods are model-agnostic (Klinkenberg 2004; Koychev 2002, 2000; Žliobaitė 2009). DDG-DA focuses on predicting future data distribution, which is a model-agnostic solution and can benefit different customized forecasting models on streaming data.

Method Design

Overall Design

In streaming data, forecasting models are trained and adapted on historical data (training data $D^{(t)}_{train}$) and make predictions on unseen data (test data $D^{(t)}_{test}$) in the future. As shown in Figure 3, training data and test data change over time. For each timestamp t, the target of $task^{(t)} :=$

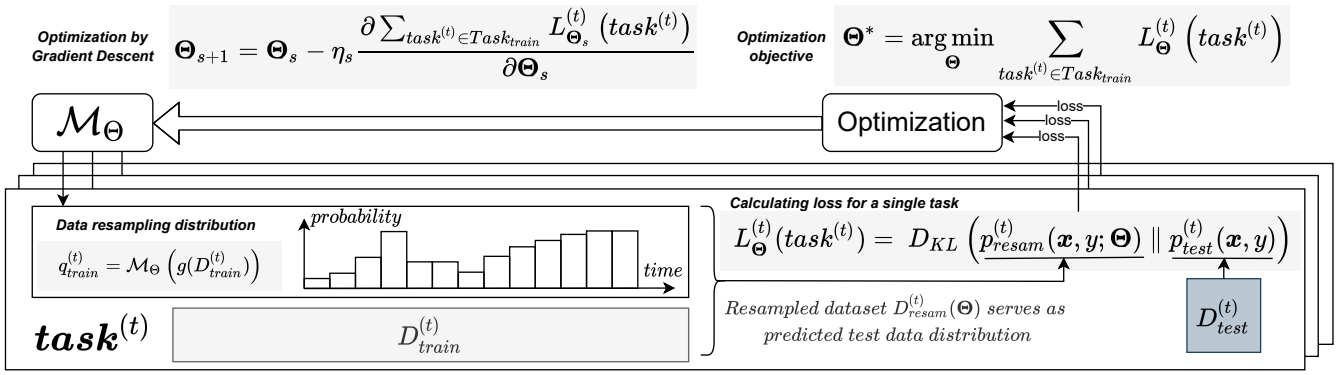


Figure 4: The learning process of DDG-DA; DDG-DA $\mathcal{M}_\Theta$ learns to guide the training process of forecasting model by generating dataset $D^{(t)}_{resam}(\boldsymbol{\Theta})$ resampled from $D^{(t)}_{train}$ with probability $q^{(t)}_{train}$. $q^{(t)}_{train}$ is the resampling probability given by $\mathcal{M}_\Theta$ at timestamp t.

$(D^{(t)}_{train}, D^{(t)}_{test})$ is to learn a new model or adapt an existing model on historical data $D^{(t)}_{train}$ and minimize the loss on $D^{(t)}_{test}$. The data come continuously and might never end. Therefore models can leverage a limited size of $D^{(t)}_{train}$ at timestamp t due to storage limitation. $D^{(t)}_{train}$ is a dataset sampled from training data distribution $p^{(t)}_{train}(\boldsymbol{x}, y)$ and $D^{(t)}_{test}$ from test data distribution $p^{(t)}_{test}(\boldsymbol{x}, y)$. The two distributions may be different (a.k.a. non-i.i.d.). This distribution gap is harmful to the forecasting accuracy on $D^{(t)}_{test}$ when learning forecasting models on $D^{(t)}_{train}$ with a distribution different from $D^{(t)}_{test}$.

DDG-DA Learning To bridge this gap, DDG-DA (annotated as $\mathcal{M}_\Theta$) tries to model the concept drift and predict the test data distribution $p^{(t)}_{test}(\boldsymbol{x}, y)$. The framework of DDG-DA is demonstrated in Figure 3. DDG-DA will act like a weighted data sampler to resample on $D^{(t)}_{train}$ and create a new training dataset $D^{(t)}_{resam}(\boldsymbol{\Theta})$ whose data distribution is $p^{(t)}_{resam}(\boldsymbol{x}, y; \boldsymbol{\Theta})$ (the distribution of the resampled dataset serves as the prediction of test distribution). DDG-DA tries to minimize difference between the $p^{(t)}_{resam}(\boldsymbol{x}, y; \boldsymbol{\Theta})$ (the predicted data distribution) and the test data distribution $p^{(t)}_{test}(\boldsymbol{x}, y)$ (the ground truth data distribution).

During the training process of DDG-DA, $\mathcal{M}_\Theta$ tries to learn patterns on $task^{(t)} \in Task_{train}$ by minimizing the data distribution distance between $p^{(t)}_{resam}(\boldsymbol{x}, y; \boldsymbol{\Theta})$ and $p^{(t)}_{test}(\boldsymbol{x}, y)$. The knowledge learned by $\mathcal{M}_\Theta$ from $Task_{train}$ is expected to transfer to new tasks from $Task_{test}$.

DDG-DA Forecast For a given task $task^{(t)} \in Task_{test}$, the forecasting model is trained on dataset $D^{(t)}_{resam}(\boldsymbol{\Theta})$ under distribution $p^{(t)}_{resam}(\boldsymbol{x}, y; \boldsymbol{\Theta})$ and forecasts on test data $D^{(t)}_{test}$. $p^{(t)}_{resam}(\boldsymbol{x}, y; \boldsymbol{\Theta})$ is the distribution of resampled dataset $D^{(t)}_{resam}(\boldsymbol{\Theta})$ and the decisive factor to the trained forecasting model . With the help of DDG-DA, the distribution of $D^{(t)}_{resam}(\boldsymbol{\Theta})$ is more similar to $D^{(t)}_{test}$ than $D^{(t)}_{train}$. The forecasting model's performance on $D^{(t)}_{test}$ is expected to be improved.

A Practical Example To make our method more understandable, we explain it with an example in the stock price trending forecasting scenario. To handle the concept drift in data, we retrain a new model each month (the rolling time interval is 1 month) based on two years of historical data (memory size is limited in an online setting). Each chance to retrain a new model to handle concept drift is called a *task*. For example, the first $task^{(2011/01)}$ contains training data $D^{(2011/01)}_{train}$ from 2009/01 to 2010/12 and a month of test data $D^{(2011/01)}_{test}$ in 2011/01. DDG-DA creates $D^{(2011/01)}_{resam}(\boldsymbol{\Theta})$ based on $D^{(2011/01)}_{train}$ to train a forecasting model to achieve better performance on $D^{(2011/01)}_{test}$. A new task is created in each month. These tasks are arranged in chronological order and separated at the beginning of 2016 into $Task_{train}$ (all $D^{(t)}_{test}$ in $Task_{train}$ range from 2011 to 2015) and $Task_{test}$ (all $D^{(t)}_{test}$ in $Task_{test}$ range from 2016 to 2020). DDG-DA is trained on $Task_{train}$ (learning to generate $D^{(t)}_{resam}(\boldsymbol{\Theta})$ and minimize its distribution distance with $D^{(t)}_{test}$). Then DDG-DA is evaluated on $Task_{test}$.

Model Design and Learning Process

The overall model design and learning process of DDG-DA are shown in Figure 4. DDG-DA will build a set of tasks $Task_{train} = \{task^{(1)}, \ldots, task^{(\tau)}\}$ for training DDG-DA. The goal of the learned DDG-DA is to improve the performance on test tasks $Task_{test} := \{task^{(\tau+1)}, \ldots, task^{(T)}\}$.

Feature Design DDG-DA is expected to guide the model learning process in each $task^{(t)}$ by forecasting test data distribution. Historical data distribution information is useful to predict the target distribution of $D^{(t)}_{test}$ and is input into DDG-DA. DDG-DA will learn concept drift patterns from training tasks and help to adapt models in test tasks.

DDG-DA could be formalized as $q^{(t)}_{train} =$

$\mathcal{M}_\Theta\left(g(D^{(t)}_{train})\right)$. g is a feature extractor. It takes $D^{(t)}_{train}$ as input and outputs historical data distribution information. $\mathcal{M}_\Theta$ leverages the extracted information and output the resampling probabilities for samples in $D^{(t)}_{train}$.

Objective Function $\mathcal{M}_\Theta$ accepts the extracted feature and outputs the resampling probability on $D^{(t)}_{train}$. The resampled dataset's joint distribution $p^{(t)}_{resam}(\boldsymbol{x}, y; \boldsymbol{\Theta})$ serves as the distribution prediction. The learning target of DDG-DA is to minimize the difference between $p^{(t)}_{resam}(\boldsymbol{x}, y; \boldsymbol{\Theta})$ and $p^{(t)}_{test}(\boldsymbol{x}, y)$. We focus on the most important concept drift subject $p(y \mid \boldsymbol{x})$ (Kelly, Hand, and Adams 1999) and assume the difference between $p^{(t)}_{test}(\boldsymbol{x})$ and $p^{(t)}_{resam}(\boldsymbol{x}; \boldsymbol{\Theta})$ are minor. The loss of DDG-DA could be reformulated as

$$\begin{aligned} & L_{\boldsymbol{\Theta}}(task^{(t)}) \\ &= D_{KL}\left(p^{(t)}_{test}(\boldsymbol{x}, y) \parallel p^{(t)}_{resam}(\boldsymbol{x}, y; \boldsymbol{\Theta})\right) \\ &= \mathbb{E}_{\boldsymbol{x}\sim p^{(t)}_{test}(\boldsymbol{x})}\left[D_{KL}\left(p^{(t)}_{test}(y \mid \boldsymbol{x}) \parallel p^{(t)}_{resam}(y \mid \boldsymbol{x}; \boldsymbol{\Theta})\right)\right] \end{aligned} \tag{1}$$

where D_{KL} represents the Kullback–Leibler divergence.

Normal distribution assumption is reasonable for unknown variables and often used in maximum likelihood estimation (Goodfellow et al. 2016). Under this assumption, $p^{(t)}_{test}(y \mid \boldsymbol{x}) = \mathcal{N}(y^{(t)}_{test}(\boldsymbol{x}), \sigma)$ and $p^{(t)}_{resam}(y \mid \boldsymbol{x}) = \mathcal{N}(y^{(t)}_{resam}(\boldsymbol{x}; \boldsymbol{\Theta}), \sigma)$ where σ is a constant.

The estimated expectation of Equation (1) on empirical sampled dataset can be reformulated as

$$L_{\boldsymbol{\Theta}}(task^{(t)}) = \frac{1}{2}\sum_{(\boldsymbol{x},y)\in D^{(t)}_{test}} \left\| y^{(t)}_{resam}(\boldsymbol{x}; \boldsymbol{\Theta}) - y \right\|^2 \tag{2}$$

Summarizing losses of all the training tasks, the optimization target of DDG-DA could be formalized as

$$\boldsymbol{\Theta}^* = \arg\min_{\boldsymbol{\Theta}} \sum_{task^{(t)}\in Task_{train}} L_{\boldsymbol{\Theta}}\left(task^{(t)}\right) \tag{3}$$

DDG-DA learns knowledge from $Task_{train}$ and transfers it to unseen test tasks $Task_{test}$. In each task, DDG-DA forecasts the future data distribution and generate dataset $D^{(t)}_{resam}(\boldsymbol{\Theta})$. Learning the forecasting models on $D^{(t)}_{resam}(\boldsymbol{\Theta})$, it adapts to upcoming streaming data better.

Optimization In this section, we will introduce the optimization process of (2). $y^{(t)}_{test}(\boldsymbol{x})$ can be get directly from dataset $D^{(t)}_{test}$. To approximate $y^{(t)}_{resam}(\boldsymbol{x}; \boldsymbol{\Theta})$, DDG-DA builds a regression proxy model $y_{proxy}(\boldsymbol{x}; \phi^{(t)})$ on $D^{(t)}_{resam}(\boldsymbol{\Theta})$. The optimization of the proxy model can be formulated as

$$\boldsymbol{\phi}^{(t)} = \arg\min_{\boldsymbol{\phi}} \sum_{(\boldsymbol{x},y)\in D^{(t)}_{resam}(\boldsymbol{\Theta})} \left\| y_{proxy}(\boldsymbol{x}; \boldsymbol{\phi}) - y \right\|^2 \tag{4}$$

The learning process of DDG-DA becomes a *bi-level optimization problem* (Gould et al. 2016). The goal of the upper-level is Equation (2) and (3) ($y^{(t)}_{resam}(\boldsymbol{x}; \boldsymbol{\Theta})$ is replaced by $y_{proxy}(\boldsymbol{x}; \boldsymbol{\phi}^{(t)})$). The goal of the lower-level optimization Equation (4) can be regarded as a constraint.

The overall bi-level optimization formulation of the DDG-DA is

$$\begin{aligned} \arg\min_{\boldsymbol{\Theta}} \quad & \sum_{task^{(t)}\in Task_{train}} \left(\sum_{(\boldsymbol{x},y)\in D^{(t)}_{test}} \left\| y_{proxy}(\boldsymbol{x}; \boldsymbol{\phi}^{(t)}) - y \right\|^2 \right) \\ \text{s.t.} \quad & \boldsymbol{\phi}^{(t)} = \arg\min_{\boldsymbol{\phi}} \sum_{(\boldsymbol{x}',y')\in D^{(t)}_{resam}(\boldsymbol{\Theta})} \left\| y_{proxy}(\boldsymbol{x}'; \boldsymbol{\phi}) - y' \right\|^2 \end{aligned} \tag{5}$$

where $\boldsymbol{\Theta}$ is the parameters of DDG-DA.

Many methods have been proposed to solve such a bi-level optimization problem (Gould et al. 2016). Some methods based on hyperparameter optimization (Bengio 2000; Baydin and Pearlmutter 2014; Maclaurin, Duvenaud, and Adams 2015) are proposed. But they have limitations on either the network size or optimization accuracy.

One of the key barriers that stop researchers from optimizing Equation (2) directly is that the lower-level optimization Equation (4) usually can not be solved in a closed-form. The $\arg\min$ operator is usually not differentiable, which makes some popular and efficient optimization algorithms (such as gradient-descend-based methods) impossible in the optimization of the upper-level (Equation (3)). (Lee et al. 2019; Bertinetto et al. 2018) argue that (Bengio 2000; Baydin and Pearlmutter 2014) it is too costly and adopt algorithms with closed-form solutions in the lower-level optimization.

DDG-DA adopts a model with a closed-form solution as $y^{(t)}_{proxy}(\boldsymbol{x}; \phi^{(t)})$. We have many choices, such as logistic regression (Kleinbaum et al. 2002), kernel-based nonlinear model (Liu 2003) and differentiable closed-form solvers (Bertinetto et al. 2018). We choose a linear model $h(\boldsymbol{x}; \boldsymbol{\phi}_{(t)}) = \boldsymbol{x}\boldsymbol{\phi}^{(t)}$ for $y_{proxy}(\boldsymbol{x}; \phi^{(t)})$ for simplicity. The resampling probability $q^{(t)}_{train}$ outputted by $\mathcal{M}_\Theta$ could be regarded as sample weights when learning forecasting models. The loss function in Equation (4) could be formulated as

$$\begin{aligned} l_{\boldsymbol{\phi}^{(t)}}\left(D^{(t)}_{resam}(\boldsymbol{\Theta})\right) =& \frac{1}{2}\sum_{(\boldsymbol{x},y)\in D^{(t)}_{train}} q^{(t)}_{train}\left(\boldsymbol{x}\boldsymbol{\phi}^{(t)} - y\right)^2 \\ =& \frac{1}{2}\left(\boldsymbol{X}^{(t)}\boldsymbol{\phi}^{(t)} - \boldsymbol{y}^{(t)}\right)^\top \boldsymbol{Q}^{(t)}\left(\boldsymbol{X}^{(t)}\boldsymbol{\phi}^{(t)} - \boldsymbol{y}^{(t)}\right) \end{aligned}$$

where $\boldsymbol{X}^{(t)}$, $\boldsymbol{y}^{(t)}$ and $\boldsymbol{Q}^{(t)}$ represent the concatenated features, labels and resampling probability in $D^{(t)}_{train}$.

It equals to a *weighted linear regression* problem (Ruppert and Wand 1994). $\boldsymbol{\phi}^{(t)}$ has a closed-form solution formalized as

$$\begin{aligned} \boldsymbol{\phi}_{(t)} =& \arg\min_{\boldsymbol{\phi}} \; l_{\boldsymbol{\phi}}(D^{(t)}_{resam}(\boldsymbol{\Theta})) \\ =& \left(\left(\boldsymbol{X}^{(t)}\right)^\top \boldsymbol{Q}^{(t)}\boldsymbol{X}^{(t)}\right)^{-1}\left(\boldsymbol{X}^{(t)}\right)^\top \boldsymbol{Q}^{(t)}\boldsymbol{y}^{(t)} \end{aligned} \tag{6}$$

This closed-form solution to Equation (4) makes the distribution distance differentiable. It makes the optimization objective of $\mathcal{M}_\Theta$ (Equation (3)) differentiable. Therefore, simple and efficient optimization algorithms can be used to train DDG-DA (e.g. stochastic gradient descent).

Method	Stock Price Trend Forecasting					Electricity Load		Solar Irradiance		
	IC	*ICIR*	*Ann.Ret.*	*Sharpe*	*MDD*	*NMAE*	*NRMSE*	*Skill (%)*	*MAE*	*RMSE*
RR	0.1178	1.0658	0.1749	1.5105	-0.2907	0.1877	0.9265	7.3047	21.7704	48.0117
GF-Lin	0.1227	1.0804	0.1739	1.4590	-0.2690	0.1843	0.9109	9.3503	21.6878	46.9522
GF-Exp	0.1234	1.0613	0.1854	1.5906	-0.2984	0.1839	0.9084	9.2652	21.6841	46.9963
ARF	0.1240	1.0657	0.1994	1.8844	**-0.1176**	0.1733	0.8901	8.6267	21.0962	47.3270
Condor	0.1273	1.0635	0.2157	2.1105	-0.1624	———	———	———	———	———
DDG-DA	**0.1312**	**1.1299**	**0.2565**	**2.4063**	-0.1381	**0.1622**	**0.8498**	**12.1327**	**18.7997**	**45.5110**

Table 1: Performance comparison of the concept drifts adaptation methods.

Experiments

In this section, we conduct experiments aiming to answer the following research questions:

- **Q1**: Can DDG-DA outperform SOTA concept drift adaptation methods in predictable concept drift scenarios?
- **Q2**: Can DDG-DA generalize to different forecasting models in different scenarios?

Experiment Setup

The experiments are conducted on multiple datasets in three real-world popular scenarios (forecasting on stock price trend, electricity load and solar irradiance (Grinold and Kahn 2000; Pedro, Larson, and Coimbra 2019)).

Experiments Results

Concept Drift Adaptation Methods Comparison (Q1) In this part, we compare DDG-DA with concept drift adaptation methods in different streaming data scenarios to answer Q1 that DDG-DA is the SOTA in concept-drift-predictable scenarios. We compared all methods both in classification tasks (stock price trend forecasting) and regression tasks (electricity load forecasting, solar irradiance forecasting).

The following methods are compared:

- **RR**: Periodically **R**olling **R**etrain model on data in memory with equal weights. The memory only stores the most recent data in a limited window size.
- **GF-Lin** (Koychev 2000): Based on RR, **G**radual **F**orgetting by weights decaying **Lin**early by time.
- **GF-Exp** (Klinkenberg 2004): Based on RR, **G**radual **F**orgetting by weights decaying **Exp**onentially by time.
- **ARF** (Gomes et al. 2017, 2018): To deal with concept drift in the streaming data, **A**daptive **R**andom **F**orest does both internal and external concept drift detecting for each newly-created tree. The final prediction will be obtained by its voting strategy.
- **Condor** (Zhao, Cai, and Zhou 2020): **Con**cept **D**rift via m**o**del **R**euse is an ensemble method that handles non-stationary environments by both building new models and assigning weights for previous models.
- **DDG-DA**: Our proposed method. DDG-DA is based on the setting of RR and generates a new dataset by resampling to retrain forecasting models.

The compared methods can be categorized into two sets based on if they are model-agnostic. A model-agnostic solution can conduct concept drift adaptation without knowing the details of the forecasting model. Therefore, a model-agnostic solution can be applied to any model, which could be designed specifically to target scenarios. RR, GF-Lin, GF-Exp, DDG-DA are model-agnostic. They use the same forecasting model and historical data memory size. ARF and Condor are not model-agnostic. Condor is designed for classification, so it is not evaluated on the regression scenarios.

We use the open-source models released in Qlib's model zoo as our candidate forecasting model. On the validation data, LightGBM (Ke et al. 2017) performs best on average. Therefore, we select LightGBM as our final forecasting model. The same forecasting model is also adopted by RR, GF-Lin and GF-Exp. For the training process of DDG-DA, we create tasks according to Figure 3. The tasks are created in a more coarse-grained frequency to reduce the retraining cost. The time interval of two adjacent tasks is 20 trading days, 7 days and 7 days in stock price trend forecasting, electricity load forecasting and solar irradiance forecasting respectively. The test data still arrives in fine granularity.

The results are shown in Table 1. RR is the most naive method and simply periodically retrains models on data in memory. It performs worst among the compared methods. GF-Lin and GF-Exp assume that the recent data distribution is more similar to the upcoming data. The weights of the most recent data are the highest and then decay by time. Such an assumption is valid in most datasets, and they outperform RR in most scenarios. ARF and Condor are the most recent SOTA solutions for concept drift adaptation. They adapt models to the most recent data to handle concept drift. The forecasting performance is improved further. However, the concept drift continues after adapting models to the most recent historical data. The adapted model could fail again when the concept drifts in the future. DDG-DA solves such a problem and performs best. It models the trend of concept drifts and generates a new dataset whose distribution is closer to that in the future. Then the forecasting model is trained on the new dataset to handle the future concept drift.

Generalization to Different Forecasting Models (Q2) To answer Q2, we conduct experiments to demonstrate that DDG-DA can enhance the performance of different forecasting models in different streaming data scenarios.

The experiments involve different forecasting models. Therefore, we only compare model-agnostic concept drift

Model	Method	Stock Price Trend Forecasting					Electricity Load		Solar Irradiance		
		IC	*ICIR*	*Ann.Ret.*	*Sharpe*	*MDD*	*NMAE*	*NRMSE*	*Skill (%)*	*MAE*	*RMSE*
Linear	RR	0.0859	0.7946	0.1578	1.4211	-0.1721	0.2080	1.0207	1.4133	24.0910	51.0632
	GF-Exp	0.0863	0.8018	0.1632	1.4373	**-0.1462**	0.2009	0.9787	1.3660	24.1391	51.0877
	DDG-DA	**0.0971**	**0.9193**	**0.1763**	**1.5733**	-0.2130	**0.1973**	**0.9702**	**3.7378**	**23.6901**	**49.8592**
MLP	RR	0.1092	0.8647	0.1803	1.5200	-0.1797	0.1928	0.9682	8.2145	22.1285	47.5405
	GF-Exp	0.1091	0.8654	0.1889	1.6121	**-0.1738**	0.1898	0.9588	8.4668	21.6236	47.4098
	DDG-DA	**0.1211**	**0.9921**	**0.2181**	**1.9409**	-0.1864	**0.1882**	**0.9537**	**10.0341**	**20.3422**	**46.5980**
Light-GBM	RR	0.1178	1.0658	0.1749	1.5105	-0.2907	0.1877	0.9265	7.3047	21.7704	48.0117
	GF-Exp	0.1234	1.0613	0.1854	1.5906	-0.2984	0.1839	0.9084	9.2652	21.6841	46.9963
	DDG-DA	**0.1312**	**1.1299**	**0.2565**	**2.4063**	**-0.1381**	**0.1622**	**0.8498**	**12.1327**	**18.7997**	**45.5110**
LSTM	RR	0.1003	0.7066	0.1899	1.8919	**-0.1264**	0.1494	0.8386	7.5990	21.7948	49.8593
	GF-Exp	0.1008	0.7110	0.1928	1.9238	-0.1450	0.1370	0.7369	11.4428	18.5193	45.8684
	DDG-DA	**0.1049**	**0.7456**	**0.2122**	**2.0334**	-0.1745	**0.1298**	**0.7290**	**12.4905**	**18.3295**	**45.3257**
GRU	RR	0.1122	0.9638	0.1841	1.6645	-0.1740	0.1352	0.7090	8.5684	21.0297	47.3572
	GF-Exp	0.1182	1.0007	0.1872	1.7588	-0.1207	0.1281	0.6688	8.4578	21.0399	47.4145
	DDG-DA	**0.1183**	**1.0091**	**0.1928**	**1.7906**	**-0.1182**	**0.1250**	**0.6588**	**10.5918**	**20.1891**	**46.3092**

Table 2: Performance comparison of the model-agnostic solutions on various scenarios and forecasting models.

adaptation solutions. For each forecasting model in each scenario, we compare DDG-DA with the RR and GF-Exp (GF-Exp is an advanced version of GF-Lin. So GF-Lin is not included). We conduct experiments on multiple popular forecasting models, including **Linear** (Graybill 1976), **MLP** (Gardner and Dorling 1998), **LightGBM** (Ke et al. 2017), **LSTM** (Hochreiter and Schmidhuber 1997) and **GRU** (Chung et al. 2014).

As the results shown in Table 2, DDG-DA outperforms others in most cases. The results demonstrate that DDG-DA has captured the pattern of concept drift over time, which is the inherent nature of data and model-agnostic.

Comparison of Optimization Methods DDG-DA converts the distribution distance optimization into a bi-level optimization by using a proxy model to model data distribution $p^{(t)}_{resam}(\boldsymbol{x}, y; \boldsymbol{\Theta})$. DDG-DA adapts a proxy model with a closed-form solution and convert Equation (4) into a differentiable operator. Therefore, the distribution distance between $D^{(t)}_{resam}(\boldsymbol{\Theta})$ and $D^{(t)}_{test}$ becomes differentiable. Besides this approach, hyperparameter-optimization-based methods are eligible for the bi-level optimization. Gradient-based Hyperparameter Optimization (GHO) is one of them and adopted by a lot of research works (Maclaurin, Duvenaud, and Adams 2015; Fan et al. 2020; Baydin et al. 2017). Instead of requiring a proxy model with a closed-form solution, GHO only requires a differentiable proxy model. It initializes the proxy model's parameters with $\phi_{(t),0}$ and then update it to $\phi_{(t),k}$ by k-steps with a gradient-based optimization method. At last, the $\phi_{(t),k}$ serves as $\phi_{(t)}$ in Equation (4) and makes Equation (2) differentiable.

To compare different bi-level optimization algorithms, we create a new solution named DDG-DA† based on DDG-DA with GHO for optimization and an MLP proxy model. As the results show in Table 3, DDG-DA† outperforms RR, GF-Lin and GF-Exp. It demonstrates the effectiveness of our framework. However, it slightly underperforms DDG-DA. DDG-DA† leverages a model with a larger capacity to model data distribution. It alleviates the problem of underfitting $p^{(t)}_{resam}(\boldsymbol{x}, y; \boldsymbol{\Theta})$. But the gap between the proxy model and the ground truth distribution still exists. The hyperparameter k is sensitive. A smaller k may result in underfitting. A greater k may waste computing resources and tend to overfit.

Method	*IC*	*ICIR*	*Ann.Ret.*	*Sharpe*	*MDD*
RR	0.1092	0.8647	0.1803	1.5200	-0.1797
GF-Lin	0.1090	0.8641	0.1880	1.5682	**-0.1656**
GF-Exp	0.1091	0.8654	0.1889	1.6121	-0.1738
DDG-DA†	0.1204	0.9732	0.2065	1.9243	-0.1705
DDG-DA	**0.1211**	**0.9921**	**0.2181**	**1.9409**	-0.1864

Table 3: Comparison of DDG-DA with different bi-level optimization algorithms and proxy model; DDG-DA† is based on GHO and an MLP proxy model.

Conclusions and Future Works

In this paper, we propose a novel concept drift adaptation method DDG-DA to adapt models to future data distribution rather than latest historical data only like previous works. DDG-DA models the concept drift and generates a new dataset by resampling historical data to guide the training process. Experiments on diverse scenarios demonstrate its effectiveness in concept-drift-predictable scenarios. DDG-DA is model-agnostic,so it can be transferred to any forecasting model used for downstream tasks.

Currently, DDG-DA is a static model that assumes the pattern of concept drift is static. This may not be valid in some scenarios (e.g. the concept drift frequency of $Task_{test}$ and $Task_{train}$ may be different). For future work, we will improve DDG-DA to a dynamic one to solve this limitation.

References

Alippi, C.; and Roveri, M. 2008. Just-in-time adaptive classifiers—Part II: Designing the classifier. *IEEE Transactions on Neural Networks*, 19(12): 2053–2064.

Bach, S. H.; and Maloof, M. A. 2008. Paired learners for concept drift. In *2008 Eighth IEEE International Conference on Data Mining*, 23–32. IEEE.

Baydin, A. G.; Cornish, R.; Rubio, D. M.; Schmidt, M.; and Wood, F. 2017. Online learning rate adaptation with hypergradient descent. *arXiv preprint arXiv:1703.04782*.

Baydin, A. G.; and Pearlmutter, B. A. 2014. Automatic differentiation of algorithms for machine learning. *arXiv preprint arXiv:1404.7456*.

Bengio, Y. 2000. Gradient-based optimization of hyperparameters. *Neural computation*, 12(8): 1889–1900.

Bertinetto, L.; Henriques, J. F.; Torr, P. H.; and Vedaldi, A. 2018. Meta-learning with differentiable closed-form solvers. *arXiv preprint arXiv:1805.08136*.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Fan, Y.; Xia, Y.; Wu, L.; Xie, S.; Liu, W.; Bian, J.; Qin, T.; Li, X.-Y.; and Liu, T.-Y. 2020. Learning to teach with deep interactions. *arXiv preprint arXiv:2007.04649*.

Gama, J.; Žliobaitė, I.; Bifet, A.; Pechenizkiy, M.; and Bouchachia, A. 2014. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4): 1–37.

Gardner, M. W.; and Dorling, S. 1998. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15): 2627–2636.

Gomes, H. M.; Barddal, J. P.; Ferreira, L. E. B.; and Bifet, A. 2018. Adaptive random forests for data stream regression. In *ESANN*.

Gomes, H. M.; Bifet, A.; Read, J.; Barddal, J. P.; Enembreck, F.; Pfharinger, B.; Holmes, G.; and Abdessalem, T. 2017. Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9-10): 1469–1495.

Goodfellow, I.; Bengio, Y.; Courville, A.; and Bengio, Y. 2016. *Deep learning*. 2. MIT press Cambridge.

Gould, S.; Fernando, B.; Cherian, A.; Anderson, P.; Cruz, R. S.; and Guo, E. 2016. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv preprint arXiv:1607.05447*.

Graybill, F. A. 1976. *Theory and application of the linear model*, volume 183. Duxbury press North Scituate, MA.

Grinold, R. C.; and Kahn, R. N. 2000. *Active portfolio management*. McGraw Hill New York.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9(8): 1735–1780.

Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; and Liu, T.-Y. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems*, 3146–3154.

Kelly, M. G.; Hand, D. J.; and Adams, N. M. 1999. The impact of changing populations on classifier performance. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, 367–371.

Khamassi, I.; Sayed-Mouchaweh, M.; Hammami, M.; and Ghédira, K. 2018. Discussion and review on evolving data streams and concept drift adapting. *Evolving systems*, 9(1): 1–23.

Kleinbaum, D. G.; Dietz, K.; Gail, M.; Klein, M.; and Klein, M. 2002. *Logistic regression*. Springer.

Klinkenberg, R. 2004. Learning drifting concepts: Example selection vs. example weighting. *Intelligent data analysis*, 8(3): 281–300.

Koychev, I. 2000. Gradual forgetting for adaptation to concept drift. In *Proceedings of ECAI 2000 Workshop on Current Issues in Spatio-Temporal Reasoning*.

Koychev, I. 2002. Tracking changing user interests through prior-learning of context. In *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, 223–232. Springer.

Lee, K.; Maji, S.; Ravichandran, A.; and Soatto, S. 2019. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10657–10665.

Liu, B. 2003. Kernel-based nonlinear discriminator with closed-form solution. In *International Conference on Neural Networks and Signal Processing, 2003. Proceedings of the 2003*, volume 1, 41–44. IEEE.

Lu, J.; Liu, A.; Dong, F.; Gu, F.; Gama, J.; and Zhang, G. 2018. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12): 2346–2363.

Maclaurin, D.; Duvenaud, D.; and Adams, R. 2015. Gradient-based hyperparameter optimization through reversible learning. In *International conference on machine learning*, 2113–2122. PMLR.

Minku, L. L.; White, A. P.; and Yao, X. 2009. 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.

Pedro, H. T.; Larson, D. P.; and Coimbra, C. F. 2019. A comprehensive dataset for the accelerated development and benchmarking of solar forecasting methods. *Journal of Renewable and Sustainable Energy*, 11(3): 036102.

Ren, S.; Liao, B.; Zhu, W.; and Li, K. 2018. Knowledge-maximized ensemble algorithm for different types of concept drift. *Information Sciences*, 430: 261–281.

Ruppert, D.; and Wand, M. P. 1994. Multivariate locally weighted least squares regression. *The annals of statistics*, 1346–1370.

Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11).

Webb, G. I.; Hyde, R.; Cao, H.; Nguyen, H. L.; and Petitjean, F. 2016. Characterizing concept drift. *Data Mining and Knowledge Discovery*, 30(4): 964–994.

Yang, Y.; Wu, X.; and Zhu, X. 2005. Combining proactive and reactive predictions for data streams. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, 710–715.

Zhao, P.; Cai, L.-W.; and Zhou, Z.-H. 2020. Handling concept drift via model reuse. *Machine Learning*, 109(3): 533–568.

Žliobaitė, I. 2009. Combining time and space similarity for small size learning under concept drift. In *International Symposium on Methodologies for Intelligent Systems*, 412–421. Springer.

Žliobaitė, I.; Pechenizkiy, M.; and Gama, J. 2016. An overview of concept drift applications. *Big data analysis: new algorithms for a new society*, 91–114.