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 the concept drift in the literature. 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.

1 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., $\mathrm{Data}^{(t)}$, in addition to historical ones, i.e., $\mathrm{Data}^{(t-1)}$, $\mathrm{Data}^{(t-2)}$, $\cdots$, $\mathrm{Data}^{(1)}$. The goal of the task is usually to predict a certain target related to $\mathrm{Data}^{(t+1)}$. 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.

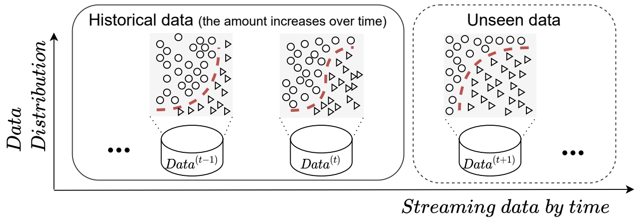


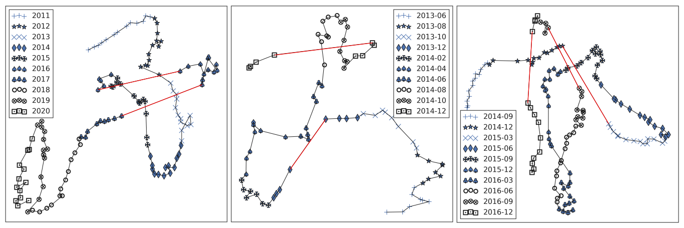
Figure 1: An example of concept drift on streaming data. Triangle and circle represent two classes of data, and the data comes like a stream. The data distribution changes over time.

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 con-

*These authors contributed equally.



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

Figure 2: The visualization of concept drift over time; though the concept drifts unexpectedly at some time (the red lines), 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. Even for some quick concept drifts in the scenarios, the case study in Appendix E illustrates that DDG-DA can handle them to some extent.

cept drift in streaming data on three real-world tasks: forecasting on stock price trend, electricity load, and solar irradiance (detailed settings can be found in Appendix C). 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 using 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 (as highlighted by red lines) 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 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 (by weighted resampling on historical data) and the ground truth future data distribution (in the format of data samples). To address this challenge, we propose to first represent a data distribution by using a model learned over 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 (Appendix F). 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 predicts the future data distribution in the circumstance of predictable concept drift. The predicted data distribution could benefit the learning task by enabling it to proactively adapt to drifting concepts. More detailed case studies will be presented to demonstrate that DDG-DA has learned reasonable and explainable knowledge.

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 adjusted data distribution.

2 Background and Related Work

2.1 Streaming data and concept drift

General Concept Drifts 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 in the future. Due to the non-stationary nature of the environment, the joint distribution p_t is not static and will change 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

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

where $f^{(t)}$ is learned from the training data in the previous stream $D_{train}^{(t)} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=t-k}^{t-1}$ 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 comes 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 necessary 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.

2.2 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 comes 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 can only 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 adapt models to the future joint distribution $p(\boldsymbol{x}, y)$ is the key to catch 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.

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 from the data under limited memory size. Incremental adaptation methods will update the current model incrementally.

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

3 Method Design

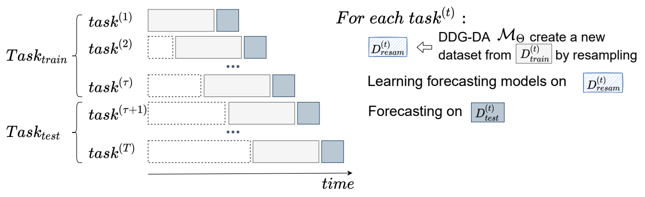

$Task_{train}$
$task^{(1)}$
$task^{(2)}$
$task^{(\tau)}$
$Task_{test}$
$task^{(\tau+1)}$
$task^{(T)}$
$time$
$For\ each\ task^{(t)}$:
$D_{resam}^{(t)}$
DDG-DA $\mathcal{M}_\Theta$ create a new dataset from $D_{train}^{(t)}$ by resampling
Learning forecasting models on $D_{resam}^{(t)}$
Forecasting on $D_{test}^{(t)}$

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.

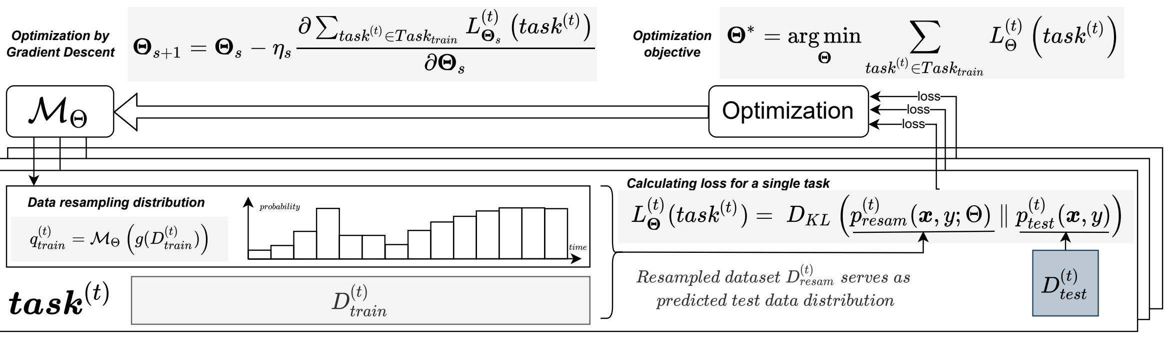

Optimization by Gradient Descent
$\Theta_{s+1} = \Theta_s - \eta_s \frac{\partial \sum_{task^{(t)} \in Task_{train}} L^{(t)}_{\Theta_s}\left(task^{(t)}\right)}{\partial \Theta_s}$
Optimization objective
$\Theta^* = \arg\min_{\Theta} \sum_{task^{(t)} \in Task_{train}} L^{(t)}_{\Theta}\left(task^{(t)}\right)$
$\mathcal{M}_\Theta$
Optimization
loss
loss
loss
Data resampling distribution
$q^{(t)}_{train} = \mathcal{M}_\Theta\left(g(D^{(t)}_{train})\right)$
probability
time
Calculating loss for a single task
$L^{(t)}_{\Theta}(task^{(t)}) = D_{KL}\left(p^{(t)}_{resam}(x, y; \Theta) \parallel p^{(t)}_{test}(x, y)\right)$
Resampled dataset $D^{(t)}_{resam}$ serves as predicted test data distribution
$task^{(t)}$
$D^{(t)}_{train}$
$D^{(t)}_{test}$

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 $D^{(t)}_{resam}$ 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.

3.1 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 $task^{(t)} := (D^{(t)}_{train}, D^{(t)}_{test})$ of the algorithm 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 comes continuously and might never end. Therefore model can leverage a reasonable 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 . This distribution gap is harmful to the forecasting accuracy on $D^{(t)}_{test}$ when learning models on $D^{(t)}_{train}$ with a different distribution.

To bridge this gap, DDG-DA tries to model the concept drift and predict the test data distribution $p^{(t)}_{test}(\boldsymbol{x}, y)$. DDG-DA will act like a weighted data sampler to resample on $D^{(t)}_{train}$ and create a new training dataset $D^{(t)}_{resam}$ whose data distribution is $p^{(t)}_{resam}(\boldsymbol{x}, y)$. DDG-DA tries to minimize difference between the $p^{(t)}_{resam}(\boldsymbol{x}, y)$ (the predicted data distribution) and the test data distribution $p^{(t)}_{test}(\boldsymbol{x}, y)$.

The framework of DDG-DA is demonstrated in Figure 3. DDG-DA is annotated as $\mathcal{M}_\Theta$ in the framework. It takes data distribution evolving information extracted from $D^{(t)}_{train}$ as input and outputs weighted resampling probability on $D^{(t)}_{train}$. The distribution of the resampled dataset $D^{(t)}_{resam}$ is $p^{(t)}_{resam}(\boldsymbol{x}, y)$ and serves as the test distribution prediction. During the training process of DDG-DA on $task^{(t)} \in Task_{train}$, $\mathcal{M}_\Theta$ tries to learn patterns to minimize the data distribution distance between $p^{(t)}_{resam}(\boldsymbol{x}, y)$ and $p^{(t)}_{test}(\boldsymbol{x}, y)$. The knowledge learned by $\mathcal{M}_\Theta$ from $Task_{train}$ is expected to transfer to new tasks $Task_{test}$.

For a given task $task^{(t)} \in Task_{test}$, the forecasting model is trained on dataset $D^{(t)}_{resam}$ under distribution $p^{(t)}_{resam}(\boldsymbol{x}, y)$ and used for forecasting on test data $D^{(t)}_{test}$. $p^{(t)}_{resam}(\boldsymbol{x}, y)$ is the distribution of resampled dataset $D^{(t)}_{resam}$ and the decisive factor to the trained forecasting model . The distribution of $D^{(t)}_{resam}$ is more similar to $D^{(t)}_{test}$ than $D^{(t)}_{train}$ with the help of DDG-DA, the performance of the forecasting model on $D^{(t)}_{test}$ is expected to be improved.

3.2 Model Design and Learning Process

The overall model design and learning process of DDG-DA is shown in Figure 4. DDG-DA will build a set of training tasks $Task_{train} = \{task^{(1)}, \ldots, task^{(\tau)}\}$ for training. 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 data distribution. Historical data distribution information is useful to predict the target distribution of $D^{(t)}_{test}$ and is put 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 $D^{(t)}_{train}$. More details about the feature design are attached in Appendix D.

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

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

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

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

According to the Theorem 1 in Appendix F, the estimated expectation of Equation (1) on empirical sampled dataset can be reformulated as

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

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

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

The optimized 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 $D_{resam}^{(t)}$. Learning the forecasting models on $D_{resam}^{(t)}$, it adapts to upcoming streaming data better.

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

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

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) ($y_{resam}^{(t)}(\boldsymbol{x}; \boldsymbol{\Theta}^{(t)})$ is replaced by $y_{proxy}^{(t)}(\boldsymbol{x}; \boldsymbol{\phi}^{(t)})$). The goal of the lower-level optimization Equation (4) can be regarded as a constraint.

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 computed 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 $y_{proxy}^{(t)}(\boldsymbol{x}; \phi^{(t)})$. We have many choices, such as logistic regression (Kleinbaum et al. 2002), kernel-based non-linear model (Liu 2003) and differentiable closed-form solvers (Bertinetto et al. 2018). We choose a linear model $h_{\boldsymbol{\phi}_{(t)}}(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{\phi}^{(t)}$ for $y_{proxy}^{(t)}(\boldsymbol{x}; \phi^{(t)})$ for simplicity. The resampling probability $q_{train}^{(t)}$ 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

$$\begin{aligned} l_{\boldsymbol{\phi}^{(t)}}(D_{resam}^{(t)}) =& \frac{1}{2} \sum_{(\boldsymbol{x},y)\in D_{train}^{(t)}} q_{train}^{(t)}(\boldsymbol{x}\boldsymbol{\phi}^{(t)} - y)^2 \\ =& \frac{1}{2}(\boldsymbol{X}^{(t)}\boldsymbol{\phi}^{(t)} - \boldsymbol{y}^{(t)})^\top \boldsymbol{Q}^{(t)} \boldsymbol{X}^{(t)}\boldsymbol{\phi}^{(t)} - \boldsymbol{y}^{(t)} \end{aligned}$$

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

The overall bi-level formulation of the DDG-DA can be represented as

$$\begin{aligned} \arg\min_{\boldsymbol{\Theta}} & \sum_{task^{(t)}\in Task_{train}} \left(\sum_{(\boldsymbol{x},y)\in D_{test}^{(t)}} \| y_{proxy}(\boldsymbol{x}; \boldsymbol{\phi}^{(t)}) - y \|^2 \right) \\ \text{s.t.} \quad & \boldsymbol{\phi}^{(t)} = \arg\min_{\boldsymbol{\phi}} \sum_{(\tilde{\boldsymbol{x}},\tilde{y})\in D_{resam}^{(t)}(\boldsymbol{\Theta})} \| y_{proxy}(\tilde{\boldsymbol{x}}; \boldsymbol{\phi}) - \tilde{y} \|^2 \end{aligned} \tag{5}$$

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

$\boldsymbol{\phi}^{(t)}$ has a closed-form solution formalized as

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

This closed-form solution of 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).

The pseudo-code of learning and forecasting process of DDG-DA can be found in Appendix B.

4 Experiments

In this section, we conduct experiments aiming to answer the following 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?

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	<u>1.0804</u>	0.1739	1.4590	-0.2690	0.1843	0.9109	<u>9.3503</u>	21.6878	<u>46.9522</u>
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**	<u>0.1733</u>	<u>0.8901</u>	8.6267	<u>21.0962</u>	47.3270
Condor	<u>0.1273</u>	1.0635	<u>0.2157</u>	<u>2.1105</u>	-0.1624	—	—	—	—	—
DDG-DA	**0.1312**	**1.1299**	**0.2565**	**2.4063**	<u>-0.1381</u>	**0.1622**	**0.8498**	**12.1327**	**18.7997**	**45.5110**

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

Further experiments are conducted in Appendix E. (1) case studies of DDG-DA;(2) execution time of different methods; (3) experiments in a concept-drift-**unpredictable** scenario to show our limitation.

4.1 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)). The detailed experiment settings are described in Appendix C.

4.2 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).

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 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 uses DDG-DA to generate 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[1] as our candidate forecasting model. On the validation data, LightGBM performs best on average. Therefore, we select LightGBM (Ke et al. 2017) 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 demonstrated in Table 1. RR is the most naive method and simply periodically retrains models on data in memory. Its performance is the 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 try to adapt their models to the most recent data to handle concept drift. The forecasting performance is improved further. However, the concept drift continues after adapting the model to the most recent historical data. The adapted model could fail again when the concept drifts in future. DDG-DA aims to solve such a problem. It tries to model the trend of concept drift and generate a new dataset whose data distribution is closer to that in the future. Then the forecasting model can be trained on the new dataset to handle the future concept drift. As we can see in the experiment results, DDG-DA performs best in scenarios.

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 adaptation solutions. For each forecasting model in each sce-

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

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**
Cat-Boost	RR	0.1047	0.7088	0.1782	1.3246	-0.2860	0.1899	0.9304	9.9895	20.5114	46.6211
	GF-Exp	0.1058	0.7114	0.1879	1.3968	-0.2414	0.1884	0.9291	10.0384	20.4882	46.5958
	DDG-DA	**0.1173**	**0.8386**	**0.1908**	**1.5787**	**-0.2102**	**0.1811**	**0.8905**	**10.4187**	**20.3605**	**46.3988**
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**
ALSTM	RR	0.1091	0.8257	0.1954	1.6497	-0.1399	0.1334	0.6965	8.6276	20.1666	47.3265
	GF-Exp	0.1100	0.8360	0.1985	1.6684	-0.1857	0.1224	0.6687	9.5213	21.1583	46.8636
	DDG-DA	**0.1106**	**0.8592**	**0.2005**	**1.7032**	**-0.1101**	**0.1217**	**0.6310**	**11.6311**	**18.4822**	**45.7709**

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

nario, 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), **CatBoost** (Dorogush, Ershov, and Gulin 2018), **LSTM** (Hochreiter and Schmidhuber 1997), **GRU** (Chung et al. 2014) and **ALSTM** (Qin et al. 2017).

As we can see in the results, 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 $p^{(t)}_{resam}(\boldsymbol{x}, y; \Theta)$. DDG-DA adapt a proxy model with closed-form solution and convert Equation (6) into a differentiable operator. Therefore, the distribution distance between $D^{(t)}_{resam}$ 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 adopting a proxy model with a closed-form solution, GHO only requires that the proxy model is differentiable. It initializes 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 $\phi_{(t),k}$ serve as $\phi^*_{(t)}$ in Equation (6) and makes Equation (2) differentiable.

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[GHO]	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[GHO] is base on GHO and an MLP proxy model.

To compare the different bi-level optimization algorithms, we create a new solution on DDG-DA framework with GHO optimization and an MLP proxy model in Table 3. Its name is DDG-DA[GHO]. RR, GF-Lin and GF-Exp. This justifies the effectiveness of our DDG-DA framework. However, it slightly underperforms DDG-DA. DDG-DA[GHO] leverage a model with larger capacity to model data distribution. It alleviates the problem of underfitting $p^{(t)}_{resam}(\boldsymbol{x}, y; \Theta)$. 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 cost more computing resources and result in overfitting.

5 Conclusions

In this paper, we proposed a novel concept drift adaptation method DDG-DA that can adapt models to the future data distribution rather than the latest historical data only like previous works. DDG-DA captures the concept drift dynamic and generate a new dataset by resampling historical data to guide the training process. Experiments on diverse scenarios demonstrate its effectiveness in concept-drift-predictable scenarios. Moreover, we provided detailed case study in Appendix E to show that the patterns learned by DDG-DA are reasonable and explainable. Our method is model-agnostic, and the learned knowledge 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 true in some scenarios (e.g. the concept drift may happen more frequently in $Task_{test}$ than $Task_{train}$). 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*.

Beyaz, E.; Tekiner, F.; Zeng, X.-j.; and Keane, J. 2018. Comparing Technical and Fundamental indicators in stock price forecasting. In *2018 IEEE 20th International Conference on High Performance Computing and Communications*, 1607–1613. IEEE.

Briere, M.; Oosterlinck, K.; and Szafarz, A. 2015. Virtual currency, tangible return: Portfolio diversification with bitcoin. *Journal of Asset Management*, 16(6): 365–373.

Chekhlov, A.; Uryasev, S.; and Zabarankin, M. 2004. Portfolio optimization with drawdown constraints. In *Supply chain and finance*, 209–228. World Scientific.

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*.

Dorogush, A. V.; Ershov, V.; and Gulin, A. 2018. CatBoost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*.

Fan, X.; and Du, D. 2017. The spillover effect between CSI 500 index futures market and the spot market. *China Finance Review International*.

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.

Jegadeesh, N.; and Titman, S. 1993. Returns to buying winners and selling losers: Implications for stock market efficiency. *The Journal of finance*, 48(1): 65–91.

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.

Qin, Y.; Song, D.; Chen, H.; Cheng, W.; Jiang, G.; and Cottrell, G. 2017. A dual-stage attention-based recurrent neural network for time series prediction. *arXiv preprint arXiv:1704.02971*.

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.

Salinas, D.; Flunkert, V.; Gasthaus, J.; and Januschowski, T. 2020. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3): 1181–1191.

Sharpe, W. F. 1994. The sharpe ratio. *Journal of portfolio management*, 21(1): 49–58.

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

Wang, J.; Zhang, Y.; Tang, K.; Wu, J.; and Xiong, Z. 2019. Alphastock: A buying-winners-and-selling-losers investment strategy using interpretable deep reinforcement attention networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1900–1908.

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, X.; Liu, W.; Zhou, D.; Bian, J.; and Liu, T.-Y. 2020. Qlib: An AI-oriented Quantitative Investment Platform. *arXiv preprint arXiv:2009.11189*.

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.

Yu, H.-F.; Rao, N.; and Dhillon, I. S. 2016. Temporal regularized matrix factorization for high-dimensional time series prediction. *Advances in neural information processing systems*, 29: 847–855.

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.

A Notations

To make it easier to follow the formulas in the paper, Table 4 provides a notation list for Section 3.

B Algorithm Pseudo-code

Algorithm 1: DDG-DA Learning Algorithm

```
1: **Input**: Historical streaming data $\mathcal{D}$
2: Create $Task_{train}$ from $\mathcal{D}$ via Fig. (3)
3: Initialize parameters $\mathbf{\Theta}$ of DDG-DA.
4: while parameters $\mathbf{\Theta}$ are not converged do
5:     for each $task^{(t)}$ in $Task_{train}$ do
6:         $\{D^{(t)}_{train}, D^{(t)}_{test}\} \leftarrow task^{(t)}$
7:         Resampling $D^{(t)}_{resam}$ from $D^{(t)}_{train}$ with prob. $q^{(t)}_{train} = \mathcal{M}_\Theta\left(g(D^{(t)}_{train})\right)$
8:         Learn $y^{(t)}_{proxy}(\boldsymbol{x};\phi^{(t)})$ as a proxy for $y^{(t)}_{resam}(\boldsymbol{x};\mathbf{\Theta}^{(t)})$ to evaluate the distribution distance between $D^{(t)}_{resam}$ and $D^{(t)}_{test}$ via Eq.(2)
9:         Add the distribution distance to the accumulated loss
10:    end for
11:    Update $\mathbf{\Theta}$ via stochastic gradient descent
12: end while
13: Return DDG-DA $\mathcal{M}_{\mathbf{\Theta}}$
```

Algorithm 2: DDG-DA Forecasting Algorithm

```
1: Input: $task^{(t)} \in Task_{test}$, DDG-DA $\mathcal{M}_{\mathbf{\Theta}}$
2: $\{D^{(t)}_{train}, D^{(t)}_{test}\} \leftarrow task^{(t)}$
3: Resample on $D^{(t)}_{train}$ to with prob. $q^{(t)}_{train} = \mathcal{M}_\Theta\left(g(D^{(t)}_{train})\right)$ to create $D^{(t)}_{resam}$
4: Train forecasting model $f_{\boldsymbol{\theta}^*_{(t)}}$ on $D^{(t)}_{resam}$
5: $\hat{\boldsymbol{y}}^{(t)}_{test} \leftarrow f_{\boldsymbol{\theta}^*_{(t)}}(\boldsymbol{X}^{(t)}_{test})$ { $\boldsymbol{X}^{(t)}_{test}$ is from $D^{(t)}_{test}$}
6: Return $\hat{\boldsymbol{y}}^{(t)}_{test}$
```

C Dataset Details

C.1 Stock Trend forecasting

The target of this task is constructing investment portfolios to maximize profit and minimize risk. An investment portfolio contains a basket of securities and cash. Portfolio management is constructing and balancing the investment portfolio periodically. Accurate forecasting the securities' price trend is the key to successful portfolio management. Financial data is typically time-series.

The experiments are carried out on the stock data of the Chinese stock market, which includes time-series data of price from 2009 to 2020 in daily frequency. Besides the open, high, low, close price and trading volume data, about 300 fundamental and technical factors (Beyaz et al. 2018) (a.k.a. features in Machine Learning) are included as well. There are about 4 thousand different stocks. We focus on investment in stocks in the experiments of the paper. Due to the memory limitation in an online setting, only two years of historical data is accessible in our experiment setting in our experiments. The test time range (evaluation time range) is from 2016 to 2020.

Evaluation and Metrics

The metrics to evaluate a strategy are categorized into signal-based metrics and portfolio-based metrics.

The signal-based metrics are designed to evaluate the predictive abilities of a given forecasting. IC (Information Coefficient) and $ICIR$ (IC Information Ratio) are the most typical metrics in this category. At each timestamp t, IC could be measured by $IC^{(t)} = corr(rank(\hat{\boldsymbol{y}}^{(t)}), rank(\boldsymbol{ret}^{(t)}))$, where $\hat{\boldsymbol{y}}^{(t)}$ is the forecasting of stock price trend and $\boldsymbol{ret}^{(t)}$ is the ground truth of stock return. For a given time period, a sequence of IC could be measured $\boldsymbol{IC} = (IC^{(1)}, \ldots, IC^{(T)})$. For that time period, $IC = mean(\boldsymbol{IC})$ and $ICIR = \frac{mean(\boldsymbol{IC})}{std(\boldsymbol{IC})}$. The $rank$ function returns the ordered position of a given value within an array in ascending order. The $corr$ function is the correlation function, which is defined as:

$$corr(\boldsymbol{x}, \boldsymbol{y}) = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}}$$

Besides signal-based metrics, portfolio-based metrics are widely used in practical investment analysis as well. The portfolios are generated based on the forecasting of the stock price trends. Portfolio-based metrics evaluate the performance of portfolios by a specific trading strategy and backtest in the corresponding period. Portfolio-based metrics cover more factors than signal-based metrics, such as trading costs and portfolio weights. In our experiments, we use a buying-winners-and-selling-losers investment strategy (Jegadeesh and Titman 1993; Wang et al. 2019). The strategy and backtest system are based on an open-source quantitative investment platform Qlib (Yang et al. 2020). We will hold 50 stocks and trade daily. The benchmark index is CSI500 (Fan and Du 2017), and the metrics are based on the excess returns to the CSI500. Excess return measures how much a portfolio outperforms the benchmark. The metrics include $Ann.Ret.$ (Annualized Return) (Briere, Oosterlinck, and Szafarz 2015), $Sharpe$ (Sharpe ratio) (Sharpe 1994) and MDD (Max Drawdown) (Chekhlov, Uryasev, and Zabarankin 2004), which are widely used in quantitative investment to evaluate the performance of portfolios. $Ann.Ret.$ indicates the return of given portfolios each year. $Sharpe = \frac{Ann.Ret.}{Ann.Vol.}$ and $Ann.Vol.$ indicates the annualized volatility. To achieve higher $Sharpe$, portfolios are expected to maximize the total excess return and minimize the volatility of the daily excess returns. MDD is the maximum relative asset value loss to the previous peak, and the lower $abs(MDD)$ is preferred. The trading strategies will hold the 50 stocks with the highest forecasting scores on the first trading day in the backtesting process. For each following trading day, stocks with the lowest forecasting score inside the current portfolio will be replaced by the stock with the high-

Notation	Notation Explanation
$\boldsymbol{X}$	The feature of the streaming data.
$\boldsymbol{y}$	The label of the streaming data.
$p_{train}^{(t)}(\boldsymbol{x}, y)$ / $p_{test}^{(t)}(\boldsymbol{x}, y)$	The joint distribution of $\boldsymbol{x}$ and y on the training/test data at the timestamp t.
$p_{train}^{(t)}(y \mid \boldsymbol{x})$ / $p_{test}^{(t)}(y \mid \boldsymbol{x})$	The conditional distribution of $\boldsymbol{x}$ and y on the training/test data at the timestamp t.
$D_{train}^{(t)}$ / $D_{resam}^{(t)}$ / $D_{test}^{(t)}$	The training/resampled/test dataset at timestamp t. $D_{resam}^{(t)}$ is resampled from $D_{train}^{(t)}$ with resampling probability $q_{train}^{(t)}$
$task^{(t)}$	The t-th task with training data $D_{train}^{(t)}$ and test data $D_{test}^{(t)}$.
$\mathcal{N}(\mu, \sigma)$	The normal distribution with expectation μ and standard deviation σ.
$Task_{train}$ / $Task_{test}$	A set of concept drift adaptation tasks. The tasks are generated on a rolling basis.
$\mathcal{M}_{\boldsymbol{\Theta}}$	The DDG-DA model with parameters $\boldsymbol{\Theta}$.
$y_{train}^{(t)}(\boldsymbol{x})$ / $y_{test}^{(t)}(\boldsymbol{x})$	The expectation of y under conditional distribution $p_{train}^{(t)}(y \mid \boldsymbol{x})$ / $p_{test}^{(t)}(y \mid \boldsymbol{x})$ on the training/test data of $task^{(t)}$.
$y_{resam}^{(t)}(\boldsymbol{x}; \boldsymbol{\Theta})$	The expectation of y under the predicted conditional distribution of DDG-DA (i.e. $D_{resam}^{(t)}$).
$q_{train}^{(t)}$	The resampling probability of $task^{(t)}$, provided by DDG-DA. It is used in the resampling process of $D_{train}^{(t)}$.
$L_{\boldsymbol{\Theta}}\left(task^{(t)}\right)$	The loss of DDG-DA of $task^{(t)}$.
$\phi^{(t)}$	The parameters of the proxy model for $task^{(t)}$.

Table 4: Notations in Section 3.

est forecasting score outside. The transaction cost(slippage is included) is 2 ‰.

C.2 Electricity Load Forecasting

Electricity load forecasting plays an essential role in the electricity corporation arrangement. Operation and maintenance costs can be greatly reduced by accurate forecasting, which improves the reliability of the power supply and the power transmission systems.

The electricity load dataset in our experiments is a UCI repository dataset that contains electricity usage from 370 users between 2011 and 2015 with a quarter-hour frequency. The memory limitation is one year of data for electricity load forecasting. In our experiments, the evaluation time range is from 2013 to 2015.

Evaluation and Metrics

The metrics used in electricity load forecasting are $NMAE$ (Normalized Mean Absolute Error) and $NRMSE$ (Normalized Root Mean Square Error), which are same with (Yu, Rao, and Dhillon 2016; Salinas et al. 2020):

$$NMAE = \frac{\sum_{i=1}^{N} |y_i - f(\boldsymbol{x}_i)|}{\sum_{i=1}^{N} |y_i|}$$

$$NRMSE = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N} (y_i - f(\boldsymbol{x}_i))^2}}{\frac{1}{N}\sum_{i=1}^{N} |y_i|}$$

where N is the number of samples, and the i-th sample is represented as $(\boldsymbol{x}_i, y_i)$.

C.3 Solar Irradiance Forecasting

Solar irradiance forecasting reduces solar power generation's uncertainty caused by the intermittence of solar irradiance. Solar irradiance forecasting can be used to schedule power generation and balance energy production and consumption.

The solar irradiance forecasting dataset (Pedro, Larson, and Coimbra 2019) is collected from 2014 to 2016 with a 5-minute frequency. It combines both irradiance features and sky-image features. The memory limitation is one year of data. In our experiments, the time range of test data is from 2015 to 2016.

Evaluation and Metrics

The metrics include $Skill$ (forecast skill), MAE (Mean Absolute Error), and $RMSE$ (Root Mean Square Error), which are specified in (Pedro, Larson, and Coimbra 2019):

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |y_i - f(\boldsymbol{x}_i)|$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (y_i - f(\boldsymbol{x}_i))^2}$$

$$skill \simeq 1 - \frac{RMSE}{RMSE_p}$$

where $RMSE_p$ is the $RMSE$ of the persistence forecasting results, which is already provided in the dataset.

D Model Details

In this section, we will introduce some model details of DDG-DA for better reproducibility .

D.1 Detailed Feature Design

We briefly described the feature design of DDG-DA in Section 3.2. In this section, we will introduce the detailed feature design. Because DDG-DA tries to capture the pattern of concept drift over time, the change of data distribution over time is the critical information for $\mathcal{M}_\Theta$. Feature extractor g is rule-based currently and extracts features based on the similarity of data in different periods. g output a matrix with shape $< sample, feature >$. It will extract features for each sample. The extracted features of each sample $\boldsymbol{x}$ can be represented as $g_{\boldsymbol{x}}(D_{train}^{(t)}) = (sim^{(\tau)}, sim^{(\tau-1)}, \dots, sim^{(\tau-k)})$, where τ represents the timestamp closest to previous test data in $D_{train}^{(t)}$ (now it has become training data at t) and $sim^{(\cdot)}$ represents the similarity of data $\boldsymbol{x}$ to the data $D_{test}^{(\tau)}$. Kullback–Leibler divergence is a widely-used metric for distribution distance, which is adopted as the similarity metric, as well as the optimization target in our paper. The sequence of data similarity reflects the change of distribution of streaming data. For more robustness in statistics, we use all the data in the short period where the sample $\boldsymbol{x}$ belongs to calculate the data similarity.

D.2 An Intuitive Description of Framework

To make our method more understandable, we try to explain it with a specific 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 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 $D_{train}^{(2011/01)}$ from 2009/01 to 2010/12 and a month of test data $D_{test}^{(2011/01)}$ in 2011/01. DDG-DA creates $D_{resam}^{(2011/01)}$ based on $D_{train}^{(2011/01)}$ to train a forecasting model to achieve better performance on $D_{test}^{(2011/01)}$. 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 $D_{test}^{(t)} \in Task_{train}$ ranges from 2011 to 2015) and $Task_{test}$ (all $D_{test}^{(t)} \in Task_{test}$ ranges from 2016 to 2020). DDG-DA is trained on $Task_{train}$ (learning to generate $D_{resam}^{(t)}$ and minimize its distribution distance with $D_{test}^{(t)}$). Then DDG-DA is evaluated on $Task_{test}$.

E Further Experimental Results

The experiments are carried out on the machines with Intel Xeon Platinum 8171M Processors and Tesla K80 GPU.

E.1 Experiment Running Time

Method	Stock Trend	Electricity	Solar
RR	185.37	160.40	146.38
GF-Lin	194.63	188.49	166.90
GF-Exp	196.65	174.63	147.50
ARF	1377.53	760.20	570.00
Condor	34.47	—	—
DDG-DA	216.50	185.53	158.23

Table 5: Running time (minutes) comparison of the concept drift adaptation methods. The is slower than other methods because its toolkit does not support parallelism. Condor is an ensemble method that doesn't retrain models, so its time cost is lower than other retraining methods.

E.2 Performance on the Unpredictable Concept Drift

In addition to the concept-drift-predictable real-world scenarios, we conduct experiments on concept-drift-unpredictable scenarios in this part. We construct 4 regression streaming datasets with random/unpredictable abrupt concept drifts. For each period under a certain concept, features $\boldsymbol{x} \in \mathbb{R}^{N\times d}$ are randomly generated, where $x_{i,j} \sim \mathcal{N}(0, I)$. The corresponding true labels are $\boldsymbol{y} = \boldsymbol{x}\boldsymbol{W} + \epsilon$, where $\boldsymbol{W} \in \mathbb{R}^d$ is randomly generated, $\|\boldsymbol{W}\|_2^2 = 1$ and ϵ is a random noise from standard normal distribution. We changes the regression model randomly to simulate random abrupt concept drift. The experiment results in Figure 5. The horizontal axis represents different generated datasets and the average results. The vertical axis represents the correlation of the adapted model's forecasting and the labels(the higher the better). It shows that DDG-DA can't achieve stable and good performance compared to others. DDG-DA is designed for predictable concept drift and fail to handle unpredictable concept drift which is out of our scope.

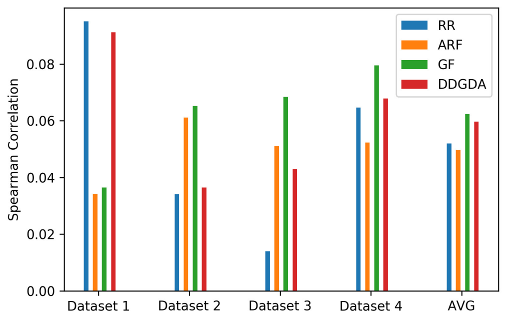


Figure 5: Comparison of different methods in 4 concept-drift-unpredictable scenarios (random abrupt concept drift). DDG-DA can't achieve stable and good performance compared to others.

Effects of Varying Time Intervals The best rolling time interval depends on the frequency of the concept drift, which is an inherent nature of the dataset. In Figure 6, we conduct experiments to show the relationship between different time intervals and final performance in the stock trend forecasting scenario. It shows that smaller interval is helpful to the performance. But the marginal improvement is much smaller when the interval is small. Users can create a validation set to get this curve and pick the elbow.

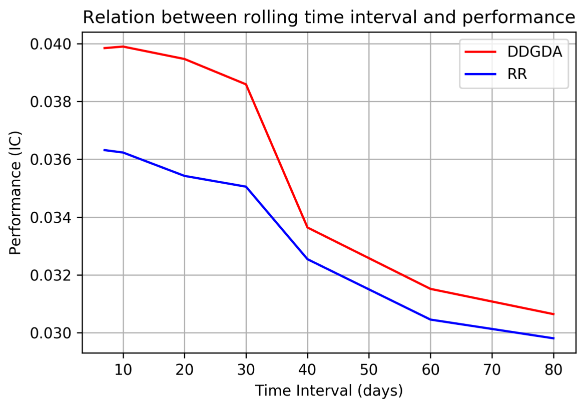


Figure 6: The relation between different time intervals and performance on stock price trend forecasting.

E.3 Case Study

In this section, we will conduct some case studies and explain the learned pattern of DDG-DA. The financial data are highly dynamic and volatile, making it an excellent scenario to present our case studies.

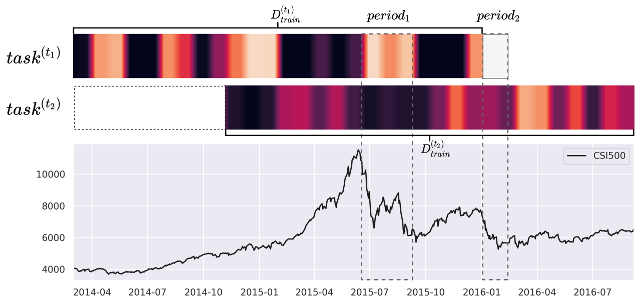


Figure 7: The sample probability is given by DDG-DA on the stock forecasting task in different tasks. DDG-DA gives different sample probability to the time-series data, the sample probability distribution changes dynamically according to stock market states at different time. DDG-DA gives more attention to the data with a similar descending price trend.

Figure 7 demonstrates several case studies of DDG-DA. The two heatmaps on the top indicate two tasks like Figure 3. The brightness of the color indicates the distribution of the sample probability given by DDG-DA. The brighter the color, the greater the probability. The sample probability changes over time. The figure below is the price trend of the CSI500[2], which is a stock index to represents the trend of the stock market. The x-axis of all the figures is aligned, which can help us analyze the behaviors of DDG-DA at different times.

As we can see on $task^{(t_2)}$, DDG-DA gives higher probability to the recent data. Such behavior is similar to GF-Lin and GF-Exp. The probability changes non-monotonically because DDG-DA leverages more information in data and captures more intricate patterns. The color during $period_1$ is dark, which indicates low resample probability on such data. The index trend below suggests that the stock market is suffering a great depression at $period_1$. The data distribution in such a particular market state is quite different, and DDG-DA gives quite reasonable resample probability.

As a concept drift adaptation solution, DDG-DA can adapt the forecasting model to the dynamic data distribution. The state of the market changes dynamically. Therefore, the optimal resample probability reweighting distribution changes over time. $task^{(t_1)}$ demonstrates another case of resample probability given by DDG-DA. The latest data in $D_{train}^{(t_1)}$ is nearly at the very beginning of the stock market crash in early 2016. Thanks to the forward-looking design, DDG-DA is aware of concept drift and gives more attention to the data with a similar pattern (i.e., the descending price trend). For data in $period_1$, DDG-DA gives a total different resample probability in $task^{(t_1)}$ than that in $task^{(t_2)}$.

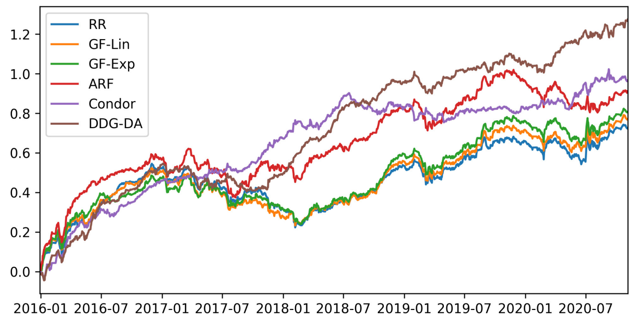


Figure 8: The accumulated excess return curves (the benchmark is CSI500) in stock price trend forecasting of different concept drift adaptation solutions from Table 1. It shows that DDG-DA performs better than other methods.

Figure 8 demonstrates the accumulated return curves of different concept drift adaptation solutions from Table 1. The benchmark is CSI500. To avoid the curve skewed exponentially by current assets value, the accumulated return is the summation of daily returns instead of a cumulative product. RR, GF-Lin and GF-Exp adopt a straightforward concept drift adaptation solution. Much detailed information is lost, and the value of their portfolios suffered a significant drawdown in 2017. Other more sophisticated concept drift adaptation methods avoid such a drawdown and achieve better performance. DDG-DA gains the highest returns among them. Besides achieving higher profit, the

[2]http://www.csindex.com.cn/en/indices/index-detail/000905

MDD and $Sharpe$ are better. It indicates that the volatility of the portfolio's returns is low, which results in lower risk. Such volatility is the result of the model's failure to adopt the new data distribution. These results demonstrate that DDG-DA has superior and stable performance on predictable concept drift adaptation.

F Algorithm Analysis

Theorem 1. *Given two normal distributions $q(z) \sim \mathcal{N}(\mu_1, \sigma_1^2)$, $p(z) \sim \mathcal{N}(\mu_2, \sigma_2^2)$, $D_{KL}(q(z) \parallel p(z))$ is proportional to the mean square error of $q(z)$ and $p(z)$ when σ_1 and σ_2 are constants.*

$$\begin{aligned}
&D_{KL}(q(z) \| p(z)) \\
=&\int q(z) \log \frac{q(z)}{p(z)} \mathrm{d}z \\
=&\int q(z)[\log q(z) - \log p(z)] \mathrm{d}z \\
=&\underbrace{\int q(z) \log q(z) \mathrm{d}z}_{\text{Term 1}} - \underbrace{\int q(z) \log p(z) \mathrm{d}z}_{\text{Term 2}} \\
=&\underbrace{\left(-\frac{1}{2}\log(2\pi) - \frac{1}{2}\left(\log\sigma_1^2 + 1\right)\right)}_{\text{Equation (8)}} \\
&-\underbrace{\left(-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log\sigma_2^2 - \frac{1}{2}\left[\frac{\sigma_1^2}{\sigma_2^2} + \frac{(\mu_1-\mu_2)^2}{\sigma_2^2}\right]\right)}_{\text{Equation (9)}} \\
\propto&(\mu_1-\mu_2)^2
\end{aligned} \tag{7}$$

$$\begin{aligned}
&\underbrace{\int q(z) \log q(z) \mathrm{d}z}_{\text{Term 1}} \\
=&\int \mathcal{N}\left(z; \mu_1, \sigma_1^2\right) \log \mathcal{N}\left(z; \mu_1, \sigma_1^2\right) \mathrm{d}z \\
=&E_{q(z)}\left[\log \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{(z-\mu_1)^2}{2\sigma_1^2}\right)\right] \\
=&E_{q(z)}\left[-\frac{1}{2}\log\left(2\pi\sigma_1^2\right) - \frac{(z-\mu_1)^2}{2\sigma_1^2}\right] \\
=&-\frac{1}{2}\log\left(2\pi\sigma_1^2\right) - E_{q(z)}\left[\frac{(z-\mu_1)^2}{2\sigma_1^2}\right] \\
=&-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log\sigma_1^2 - \frac{1}{2}E_{q(z)}\left[\frac{(z-\mu_1)^2}{\sigma_1^2}\right] \\
=&-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log\sigma_1^2 - \frac{1}{2\sigma_1^2}E_{q(z)}\left[(z-\mu_1)^2\right] \\
=&-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log\sigma_1^2 - \frac{1}{2\sigma_1^2}\sigma_1^2 \\
=&-\frac{1}{2}\log(2\pi) - \frac{1}{2}\left(\log\sigma_1^2 + 1\right)
\end{aligned} \tag{8}$$

$$\begin{aligned}
&\underbrace{\int q(z) \log p(z) \mathrm{d}z}_{\text{Term 2}} \\
=&\int \mathcal{N}\left(z; \mu_1, \sigma_1^2\right) \log \mathcal{N}\left(z; \mu_2, \sigma_2^2\right) \mathrm{d}z \\
=&E_{q(z)}\left[\log \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left(-\frac{(z-\mu_2)^2}{2\sigma_2^2}\right)\right] \\
=&E_{q(z)}\left[-\frac{1}{2}\log\left(2\pi\sigma_2^2\right) - \frac{(z-\mu_2)^2}{2\sigma_2^2}\right] \\
=&-\frac{1}{2}\log\left(2\pi\sigma_2^2\right) - E_{q(z)}\left[\frac{(z-\mu_2)^2}{2\sigma_2^2}\right] \\
=&-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log\sigma_2^2 - \frac{1}{2}E_{q(z)}\left[\frac{(z-\mu_2)^2}{\sigma_2^2}\right] \\
=&-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log\sigma_2^2 - \frac{1}{2\sigma_2^2}E_{q(z)}\left[(z-\mu_2)^2\right] \\
=&-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log\sigma_2^2 - \frac{1}{2\sigma_2^2}E_{q(z)}\left[z^2 - 2z\mu_2 + \mu_2^2\right] \\
=&-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log\sigma_2^2 - \frac{1}{2\sigma_2^2}\left[E_{q(z)}z_j^2 - 2E_{q(z)}z\mu_2 + E_{q(z)}\mu_2^2\right] \\
=&-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log\sigma_2^2 - \frac{1}{2\sigma_2^2}\left[\sigma_1^2 + \mu_1^2 - 2\mu_1\mu_2 + \mu_2^2\right] \\
=&-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log\sigma_2^2 - \frac{1}{2\sigma_2^2}\left[\sigma_1^2 + (\mu_1-\mu_2)^2\right] \\
=&-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log\sigma_2^2 - \frac{1}{2}\left[\frac{\sigma_1^2}{\sigma_2^2} + \frac{(\mu_1-\mu_2)^2}{\sigma_2^2}\right]
\end{aligned} \tag{9}$$