DATA DRIFT CORRECTION VIA TIME-VARYING IMPORTANCE WEIGHT ESTIMATOR

Anonymous authors
Paper under double-blind review

ABSTRACT

Real-world deployment of machine learning models is challenging when data evolves over time. And data does evolve over time. While no model can work when data evolves in an arbitrary fashion, if there is some pattern to these changes, we might be able to design methods to address it. This paper addresses situations when data evolves gradually. We introduce a novel time-varying importance weight estimator that can detect gradual shifts in the distribution of data. Such an importance weight estimator allows the training method to selectively sample past data—not just similar data from the past like a standard importance weight estimator would but also data that evolved in a similar fashion in the past. Our time-varying importance weight is quite general. We demonstrate different ways of implementing it that exploit some known structure in the evolution of data. We demonstrate and evaluate this approach on a variety of problems ranging from supervised learning tasks (multiple image classification datasets) where the data undergoes a sequence of gradual shifts of our design to reinforcement learning tasks (robotic manipulation and continuous control) where data undergoes a shift organically as the policy or the task changes.

1 INTRODUCTION

Real-world machine learning performance often drops during deployment when test data no longer stem from the same distribution from which previous training data were sampled. Thus many tools have been developed to address data distribution shift[1] (Heckman, 1979; Shimodaira, 2000; Huang et al., 2006; Bickel et al., 2007; Sugiyama et al., 2007b; 2008; Gretton et al., 2008), often using an estimate of the Radon-Nikodym (Rosenbaum & Rubin, 1983) derivative between the two distributions (also known as a *propensity score* or *importance weight*) to re-weight the training data so that its weighted distribution better matches the test data (Agarwal et al., 2011; Wen et al., 2014; Reddi et al., 2015b; Chen et al., 2016; Fakoor et al., 2020b;a; Tibshirani et al., 2019). These methods mostly consider offline settings with one training and one test dataset.

In many applications, data for supervised learning is continuously collected from a constantly evolving distribution (e.g. due to a pandemic (Callaway, 2020)) such that the single train/test paradigm no longer applies. Settings in which the data distribution drifts gradually over time (rather than experiencing abrupt changes) are particularly ubiquitous (Shalev-Shwartz, 2012). Assuming observations are not statistically dependent over time (as in time-series), how to best train models in such streaming/online learning settings remains a key question (Lu et al., 2018). Some basic options include: fitting/updating the model to only the recent data (which is statistically suboptimal if the distribution has not significantly shifted) or fitting the model to all previous observations (which leads to biased estimates if shift has occurred). Here we consider a less crude approach in which past data are weighted during training with continuous-valued weights that vary over time. Our proposed estimator of these weights generalizes standard two-sample propensity scores, allowing the training process to selectively emphasize past data collected at time t based on the distributional similarity between the present and time t.

[1]Covariate, label, and concept shifts are all referred to as distribution shift in the paper to simplify things wherever it is clear from the context.

We evaluate our proposed method in supervised and reinforcement learning settings involving a sequence of gradually changing tasks with slow repeating patterns. In such settings, the model not only must continuously adapt to changes in the environment/task but also learn how to select past data which may have become more relevant for the current task. Comprehensive experiments demonstrate that our method can effectively detect shift in data and statistically account for such changes during learning and inference time.

2 Related Work

Given its importance in real-world applications, the problem of how to learn from shifting distributions has been widely studied. Much past work has focused on a single shift between training/test data (Lu et al., 2021; Wang & Deng, 2018; Fakoor et al., 2020b) as well as restricted forms of shift involving changes in only the features (Sugiyama et al., 2007a; Reddi et al., 2015a), labels (Lipton et al., 2018; Garg et al., 2020; Alexandari et al., 2020), or in the underlying relationship between the two (Zhang et al., 2013; Lu et al., 2018). Past approaches to handle distributions evolving over time have been considered in the literature on: concept drift Gomes et al. (2019); Souza et al. (2020), reinforcement learning (shift between the target policy and behavior policy) Schulman et al. (2015); Wang et al. (2016); Fakoor et al. (2020a), (meta) online learning Shalev-Shwartz (2012); Finn et al. (2019); Harrison et al. (2020); Wu et al. (2021), and task-free continual/incremental learning Aljundi et al. (2019); He et al. (2019), but to our knowledge, existing methods for these settings do not employ time-varying data weights like we propose here. Time-varying data weights have been considered in survival analysis which models longitudinal observations subject to censoring Lu (2005); Cox (1972), but our weights here are implemented differently (via deep learned estimates) and utilized for general supervised learning and reinforcement learning with drift.

3 Approach

Consider a standard learning problem where training data are drawn from a probability distribution $p(x)$ and test data have a different probability distribution $q(x)$. Our goal is to build a model that can predict equally well on the training and test data distributions. We can do so by observing that

$$\mathbb{E}_{x\sim q(x)}[\ell(x)] = \int \frac{\mathrm{d}p(x)}{\mathrm{d}p(x)}\,\mathrm{d}q\,(x)\ell(x) = \mathbb{E}_{x\sim p(x)}\left[\frac{\mathrm{d}q(x)}{\mathrm{d}p(x)}\ell(x)\right] = \mathbb{E}_{x\sim p(x)}\Big[\beta(x)\,\ell(x)\Big] \tag{1}$$

where $\ell(x)$ is any function, say the loss of our model. The propensity score $\beta(x) = \frac{\mathrm{d}q(x)}{\mathrm{d}p(x)}$ is the importance ratio; this can also be seen as the Radon-Nikodym derivative (Resnick, 2013) of the two distributions. The propensity score measures the likelihood that a sample x came from distribution p against it coming from a distribution q. We will call $\beta(x)$ the "standard propensity score".

Since densities $q(x)$ and $p(x)$ are often unknown, a binary classifier (Agarwal et al., 2011) needs to be used to estimate $\beta(x)$ by utilizing samples drawn from p and q. In particular, we want to create a data set, $D = \{(x_i, z_i)\}_{i=1}^N$ for the binary classifier such that $z_i = 1$ if $x_i \sim p$ and $z_i = -1$ if $x_i \sim q$. We want our training data to have $z = 1$ in half of the training examples (with $x \sim p$) and $z = -1$ (with $x \sim q$) in the other half. We fit the binary classifier by solving the following:

$$\underset{\theta}{\text{minimize}}\ \frac{1}{N}\sum_{(x_i,z_i)}^{N} \log\Big(1 + \exp(-z_i g_\theta(x_i))\Big) \tag{2}$$

where g_θ can be either a linear or non-linear model parameterized by θ.

3.1 Time-varying importance weight estimator

Let us now consider time-varying probability distribution $p_t(x)$ with $t = 1, \ldots, T$. We will assume that we have recorded tuples $D = \{(x_i, t_i)\}_{i=1}^m$ with each $x_i \sim p_{t_i}(x)$. Our goal is to build a model that can predict well on data from $p_T(x)$ using the historical data D, say we seek to minimize

$$\mathbb{E}_{x\sim p_{T+\mathrm{d}t}(x)}\left[\ell(x)\right]. \tag{3}$$

Observe that we seek to build a model not for data from p_T but instead from $p_{T+\mathrm{d}t}$ where we will think of $\mathrm{d}t$ as small step in the future. **This problem is challenging because we do not have data from** $p_{T+\mathrm{d}t}$.

Let us define $p(t) = N^{-1} \sum_{i=1}^{N} \delta_{\{t=t_i\}}$ which is simply the empirical distribution of the time instants at which we received our samples; here δ is the Dirac-delta distribution. We can now define the marginal on the data as

$$p(x) = \int_0^T \mathrm{d}t \, p(t) \, p_t(x).$$

Inspired by the above expression, we will liken $p_t(x) \equiv p(x \mid t)$.

Assumption 1 (Data evolves gradually). If data evolves arbitrarily, then it is difficult to expect a model learned from past data to be able to predict accurately on future data. But observe that

$$\begin{aligned} \mathbb{E}_{x\sim p_{T+\mathrm{d}t}(x)} \left[\ell(x)\right] &= \mathbb{E}_{x\sim p_T(x)} \left[\ell(x)\right] + \int \mathrm{d}x \left(p_{T+\mathrm{d}t}(x) - p_T(x)\right) \ell(x) \\ &\leq \mathbb{E}_{x\sim p_T(x)} \left[\ell(x)\right] + 2\mathrm{TV}(p_{T+\mathrm{d}t}(x), p_T(x)), \end{aligned} \tag{4}$$

where $\mathrm{TV}(p_{T+\mathrm{d}t}(x), p_T(x)) = \frac{1}{2} \int \mathrm{d}x \, |p_{T+\mathrm{d}t}(x) - p_T(x)|$ is the total variational divergence between the probability distributions at time T and $T + \mathrm{d}t$, and the integrand $\ell(x)$ is upper bounded by 1 in magnitude (without loss of generality, say after normalizing it). We therefore assume that the changes in our time-varying probability distribution $p_T(x)$ are upper bounded by a constant (uniformly over time t); this can allow us to build a model for $p_{T+\mathrm{d}t}$ using data from $\{p_t\}_{t\leq T}$.

Validating and formalizing Assumption 1 This assumption is a natural one in many settings and we can check it in practice using a two-sample test to estimate the total variation $\mathrm{TV}(p_{T+\mathrm{d}t}, p_T)$ (Gretton et al., 2012). We do so for some of our tasks in §4 (see Fig. 2).We can also formalize this assumption mathematically as follows. The transition function P_t of a Markov process X_t satisfies

$$\mathbb{E}\left[\varphi(X_t)\right] = \int_0^t \varphi(y) P_t(X_s, \mathrm{d}y).$$

for any measurable function φ. The distribution of the random variable X_t corresponding to this Markov process is exactly our p_t. We can define the semi-group of this transition function as the conditional expectation $K_t\varphi = \int \mathrm{d}y \, \varphi(y) P_t(x, \mathrm{d}y)$ for any test function φ. Such a semi-group satisfies properties that are familiar from Markov chains in finite state-spaces, e.g., $K_{t+s} = K_t K_s$. The infinitesimal generator of the semi-group K_t is defined as $A\varphi = \lim_{t\to 0}(\varphi - K_t\varphi)/t$. This generator completely defines the local evolution of a Markov process (Feller, 2008)

$$p_t = e^{-tA} \, p_0 \tag{5}$$

where e^{tA} is the matrix exponential of the generator A. If all eigenvalues of A are positive, then the Markov process has a steady state distribution, i.e., the distribution of our data p_t stops evolving (this could happen at very large times and the time-scale depends upon the smallest non-zero eigenvalue of A). On the other hand, if the magnitude of the eigenvalues of A is upper bounded by a small constant, then the stochastic process evolves gradually. Note it is not necessary for the stochastic process to have a steady state distribution for our time-varying importance weight to be meaningful. Our method is expected to be effective so long as the process evolves gradually, which most processes do.

Remark 2 (Evaluating the model learned from data $\{p_t(x)\}_{t\leq T}$ upon test data from $p_{T+\mathrm{d}t}(x)$). We are interested in making predictions on future data, i.e., data from $p_{T+\mathrm{d}t}$. For all our experiments, we will therefore evaluate on test data from "one time-step in the future". The learner does not have access to any training data from $p_{T+\mathrm{d}t}$ in our experiments. This is reasonable if the data evolves gradually. Note our setting is different from typical implementations of online meta-learning (Finn et al., 2019; Harrison et al., 2020) and continual learning (Hadsell et al., 2020; Caccia et al., 2020).

In the sequel, motivated by Assumption 1, we will build a time-dependent propensity score for p_T instead of $p_{T+\mathrm{d}t}$. Using a similar calculation as that of (1) (see Appendix A for a derivation), we can

write our objective as

$$\mathbb{E}_{x\sim p_T(x)}\left[\ell(x)\right] = \mathbb{E}_{t\sim p(t)}\mathbb{E}_{x\sim p_t(x)}\left[\omega(x,T,t)\,\ell(x)\right], \tag{6}$$

where

$$\omega(x,T,t) = \frac{\mathrm{d}p_T}{\mathrm{d}p_t}(x) \tag{7}$$

is a time-varying importance weight estimator.

Remark 3 (Is the time-varying importance weight equivalent to standard propensity score on (x,t)?). One may be inclined to consider whether the development above remains meaningful if we define a new random variable $z \equiv (x,t)$ with a corresponding distribution $p(z) \equiv p_t(x)p(t)$. However, doing so is not useful for estimating $\mathbb{E}_{x\sim p_T(x)}\left[\ell(x)\right]$ because our objective (6) involves conditioning on a particular time T.

We again need to estimate this score using samples from our dataset D. To do so, we will consider two models for $p_t(x)$.

Method 1: Modeling $p_t(x)$ using an exponential family. We model the distribution of x and t as

$$p_t(x) = p_0(x)\exp\left(g_\theta(x,t)\right). \tag{8}$$

for some continuous function $g_\theta(x,t)$ parameterized by weights θ. This gives

$$\omega(x,T,t) = \exp\left(g_\theta(x,T) - g_\theta(x,t)\right) \tag{9}$$

Now observe that the left-hand side can be calculated using the same objective that we had in (2) to estimate $g_\theta(x,t)$. To train the binary classifier, we need to create a triplet (x_i,t',t) for each of samples in the dataset with label 1 if t' equals the time corresponding to x_i, i.e., if $t_i = t'$ and label -1 if t equals the time corresponding to x_i, i.e. $t_i = t$. These steps are detailed in Algorithm 1.

Method 2: Modeling deviations from the marginal Observe that in the above method $p_0(x)$ is essentially undefined. We can exploit this "gauge" to choose $p_0(x)$ differently. We can model $p_t(x)$ as deviations from the marginal on x:

$$p_t(x) = p(x)\exp\left(g_\theta(x,t)\right). \tag{10}$$

This again gives $\omega(x,T,t) = \exp\left(g_\theta(x,T) - g_\theta(x,t)\right)$. For each of the N samples in the dataset, we create a datum (x_i,t_i) with label 1 and another datum (x_i,t_j) for $j \neq i$ with label -1. The logistic regression in (2) is now fitted for the classifier $g_\theta(x,t)$ on this dataset.

The difference between the two methods is subtle. In Method 1, the classifier $g_\theta(x,t)$ is fitted on tuples (x,t',t) while in Method 2, the dataset only consists of tuples (x,t). In the former case, we think of the fitted model $g_\theta(x,t)$ as an estimate of $\log(p(x,t)/p(x,0))$ while in the latter case we think of the fitted model as the estimate of $\log(p(x,t)/p(x))$. In both cases, we can use this function $g_\theta(x,t)$ to calculate our time-varying importance weight. Because of this, we expect the two methods to perform similarly, as our experiments will also show.

3.2 Learning models using data that evolves across time

We next discuss a learning theoretic perspective on how we should learn models when the underlying data evolves with time. For the purposes of this section, suppose we have m input-output pairs $D_t = \{(x_i,y_i)\}_{i=1}^m$ that are independent and identically distributed from $p_t(x,y)$ for $t = 1,\ldots,T$ (note that these are data from each time instant, there can be correlations across time). We seek to learn hypotheses $\overline{h} = (h_t)_{t=1,\ldots,T} \in H^T$ using these samples that can generalize on new samples from distributions $(p_t)_{t=1,\ldots,T}$ respectively. We have denoted the joint hypothesis space as H^T. A typical uniform convergence bound on the population risk of one h_t suggests that if

$$m = \mathcal{O}\left(\epsilon^{-2}(\mathrm{VC}_H - \log\delta)\right),$$

then $e_t(h_t) \leq \hat{e}_t(h_t) + \epsilon$ where $e_t(h_t)$ the population risk of h_t on p_t and $\hat{e}_t(h_t)$ is its average empirical risk on the m samples from D_t (Vapnik, 1998). Here VC is the VC-dimension of $H \ni h_t$.

Consider the setting when we want hypotheses that have a small population risk $e^t(\overline{h}) = T^{-1} \sum_{t=1}^{T} e_t(h_t)$. To achieve this, we may minimize the empirical risk $\hat{e}^t(\overline{h}) = T^{-1} \sum_{t=1}^{T} \hat{e}_t(h_t)$. Baxter (2000) shows that if

$$m = \mathcal{O}\left(\epsilon^{-2}\left(\text{VC}_H(T) - T^{-1}\log\delta\right)\right) \tag{11}$$

then we have $e^t(\overline{h}) \leq \hat{e}^t(\overline{h}) + \epsilon$ for any $\overline{h}$. The quantity $\text{VC}_H(T)$ is a generalized VC-dimension that characterizes the number of distinct predictions made by the T different hypotheses; it depends upon the stochastic process underlying our individual distributions p_t. Larger the time T, smaller the $\text{VC}_H(T)$. Whether we should use the m samples from p_T to train one hypothesis, or use all the mT samples to achieve a small *average* population risk across all the tasks, depends upon how related the tasks are. If our goal is only the latter, then we can train on data from all the tasks because

$$\text{VC}_H(T) \leq \text{VC}_H, \text{ for all } T \geq 1.$$

This is fundamentally because all the tasks are together used by the learner to learn an inductive bias, and effectively a smaller hypothesis space to fit the hypotheses $\overline{h}$ from. Such a procedure is effective if tasks are related to each other and there are a number of theoretical and empirical procedures to estimate such relatedness; see Baxter (2000); Gao & Chaudhari (2021); Achille et al. (2019) among others. This result is often used as the motivation for training on multiple tasks; it motivates one of our baselines (**Everything**). The procedure that only builds the hypothesis using data from p_T is our second baseline (**Recent**).

Our goal is however not to obtain a small population risk on all tasks, it is instead to obtain a small risk on $p_{T+\text{dt}}$—our proxy for it being the latest task p_T. Even if the average risk on all tasks of the hypotheses trained above is small, the risk on a particular task p_T can be large. Ben-David & Schuller (2003) studies this phenomena. They show that if we can find a hypothesis $h \circ f_t$ such that $h \circ f_t$ can predict accurately on p_t for all t, then this effectively reduces the size of the hypothesis space $\overline{h} \in H^T$ and thereby entails a smaller sample complexity in (11) at the added cost of searching for the best hypothesis $f_t \in F$ for each task p_t (which requires additional samples that scale linearly with VC_F). The sample complexity of such a two-stage search can be better than training on all tasks, or training on p_T in isolation, under certain cases, e.g., if $\text{VC}_H \geq \log|F|$ for a finite hypothesis space F (Ben-David & Schuller, 2003).

While the procedure that combines such hypotheses to get $h \circ f_t$ in the above theory is difficult to implement in practice, this argument motivates our third baseline (**Finetune**) which first fits a hypothesis on all past tasks $(p_t)_{t \leq T}$ and then adapts it further using data from p_T.

4 Experiments

We provide a broad empirical comparison of our proposed method in both continuous supervised and reinforcement learning settings. We first evaluate our method on synthetic data sets with time-varying shifts. Next, we create continuous supervised image classification tasks utilizing CIFAR-10 (Krizhevsky et al., 2009) and CLEAR (Lin et al., 2021) datasets. Finally, we evaluate our method on ROBEL D'Claw (Yang et al., 2021) simulated robotic environments and MuJoCo (Todorov et al., 2012) simulator. See also Appendices C and D for a description of our settings and more results.

4.1 Continuous Supervised Learning

The goal in these experiments is to build model that can predict well on data from $t+1$ using *only* historical data from $\{p_0, \ldots, p_t\}$ (see also Remark 2). Note we train all different models, including all baselines, completely from *scratch* per time step; we do *not* update the model continually because we focus on understanding the performance of the time-varying importance weight only. Moreover, our method automatically detects shifts in the distribution of data without knowing them as a priori. In the following, we first explain the baseline models, then the data sets and simulated shift setups. Finally, we discuss the results of these experiments.

4.1.1 BASELINE METHODS

We compare our approach to the following baseline methods:

1. Everything: To obtain model used at time t, we train on pooled data from all past times $\{p_s(x) : s \leq t\}$, including the most recent data from $p_t(x)$. More precisely, this involves minimizing the objective

$$\ell^t(\phi) = \frac{1}{N} \sum_{s=1}^{t} \sum_{\{x_i : t_i = s\}} \ell(\phi; x_i). \tag{12}$$

where the loss of the predictions of a model parameterized by weights ϕ is $\ell(\phi; x_i)$ and we have defined $\ell^t(\phi)$ as the cumulative loss of all tasks up to time t.

2. Recent trains only on the most recent data, i.e., data from $p_t(x)$. This involves optimizing

$$\ell_t(\phi) = \frac{1}{N'} \sum_{\{x_i : t_i = t\}} \ell(\phi; x_i), \text{ where } N' = \sum_{\{x_i : t_i = t\}} 1. \tag{13}$$

3. Fine-tune: first trains on all data from the past using the objective

$$\phi^{t-} = \arg\min_{\phi} \frac{1}{N} \sum_{s=1}^{t} \sum_{\{x_i : t_i = s, s < t\}} \ell(\phi; x_i)$$

and then finetunes this model using data from p_t. We can write this problem as

$$\ell^{t+}(\phi) = \frac{1}{N'} \sum_{\{x_i : t_i = t\}} \ell(\phi; x_i) + \Omega(\phi - \phi^{t-}), \tag{14}$$

where $\Omega(\phi - \phi^{t-})$ is a penalty that keeps ϕ close to ϕ^{t-}, e.g., $\Omega(\phi - \phi^{t-}) = \|\phi - \phi^{t-}\|_2^2$. Sometimes this approach is called "standard online learning" (Finn et al., 2019) and it resembles (but is not the same as) Follow-The-Leader (FTL) (Shalev-Shwartz, 2012). We call it "Finetune" make the objective explicit to avoid any confusion due its subtle differences with FTL. In practice, we implement finetuning without the penalty Ω by initializing the weights of the model to ϕ^{t-} in (14).

Remark 4 (Properties of the different baselines). Recall that our goal is to build a model that predicts on data from $p_{t+1}(x)$. It is important to observe that **Everything** minimizes the objective over all the past data; for data that evolves over time, doing so may be detrimental to performance at time t. If we think using a bias-variance tradeoff, the variance of the predictions of **Everything** is expected to be smaller than the variance of predictions made by **Recent**, although the former is likely to have a larger bias. For example, if the data shifts rapidly, then we should expect **Everything** to have a large bias and potentially a worse accuracy than that of **Recent**. But if the amplitude of the variations in the data is small (and drift patterns repeat) then **Everything** is likely to benefit from the reduced variance and perform better than **Recent**. **Fine-tune** strikes a balance between the two methods and although the finetuning procedure is not always easy to implement optimally, e.g., the finetuning hyper-parameters can be different for different times. As our results illustrate neither of these approaches show consistent trend in the experiments and highly depend on a given scenario which can be problematic in practice, e.g. sometimes **Everything** performs better than both but other times **Fine-tune** is better (compare their performance in Fig. 1).

Remark 5 (How does the objective change when we have supervised learning or reinforcement learning?). We have written Eqs. (12) to (14) using the loss $\ell(\phi; x_i)$ only for the sake of clarity. In general, the loss depends upon both the inputs x_i and their corresponding ground-truth labels y_i for classification problems. For problems in reinforcement learning, we can think of our data x_i as entire trajectories from the system and thereby the loss $\ell(\phi; x_i)$ as the standard 1-step temporal difference (TD) error which depends on the weights that parameterize the value function.

We incorporate our time-varying importance weight into (12) as follows:

$$\ell^t(\phi) = \frac{1}{N} \sum_{s=1}^{t} \sum_{\{x_i : t_i = s\}} \boldsymbol{\omega}(x_i, t, t_i)\, \ell(\phi; x_i). \tag{15}$$

Note that the *only* difference between our method and **Everything** is introduction of $\boldsymbol{\omega}$ into (12) and all other details (e.g. network architecture, etc.) are exactly the same. $\boldsymbol{\omega}$ can be trained based on our algorithm explained in §3.1.

4.1.2 Drifting Gaussian Data

In this experiment, we design continuous classification tasks using a periodic shift where samples are generated from a Gaussian distribution with time-varying means (μ_t) and standard deviation of 1. In particular, we create a data set, $D^t = \{(x_i^t, y_i^t)\}_{i=1}^N$, at each time step t where $x_i^t \sim N(\mu_t, 1)$, $\mu_t = \mu_{t-1} + d/10$, $\mu_0 = 0.5$, and label 1 ($y_i^t = 1$) is assigned if $x_i^t > \mu_t$ otherwise $y_i^t = 0$. We change direction of shift every 50 time steps (i.e., set $d \leftarrow -d$) and use $N = 2000$ in our experiments. We run this experiments for 160 time steps. Fig. 1a displays that our method performs much better than **Everything** and similarly to others. We note that both **Recent** and **Fine-tune** are like oracle for this experiment as test time data from $p_{t+1}(x)$ are more similar to the recent data than other historical data. This experiment makes a clear case for our method as it shows that our method allows the training to selectively utilize data similar to the current time and importantly data that evolved in a similar fashion in the past.

4.1.3 Image Classification with Evolving Labels

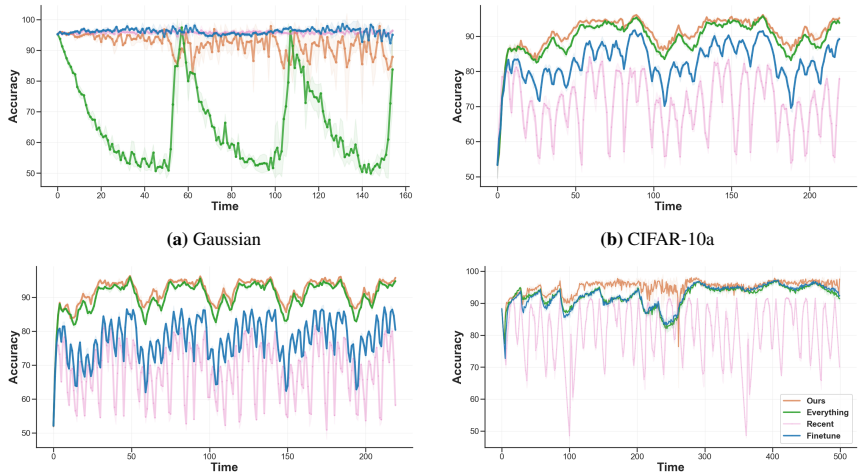
We adopt the setting of Wu et al. (2021) to design experiments for classification under continuous label shift with images from CIFAR-10 (Krizhevsky et al., 2009) and CLEAR (Lin et al., 2021).

CIFAR-10. We split the original CIFAR-10 training data into 90% (train) and 10% (validation), and use original test split as is. Following Wu et al. (2021), we pre-train Resent-18 (He et al., 2016) using training data for 100 epochs. In subsequent continuous label shift experiments, we utilize the pre-trained model and only fine-tune last convolution block and classification layer. We average all results over 3 random splits.

CLEAR. This data set consists of 10 classes where each class has 3000 samples. CLEAR is curated over a large public data set only including data spanning from years 2004 to 2014. We utilize unsupervised pre-trained features provided by Lin et al. (2021) and only use a linear layer classifier As Lin et al. (2021) stated, the linear model using pre-trained features is far more effective than training from scratch for this data set (For more, see Lin et al. (2021)). We average all results over 3 random train-validation-test splits, of relative size 60% (train), 10% (validation), and 30% (test).

Continuous label shifts. We create a sequence of classification tasks where the label distribution changes at each time according to a predefined label distribution in slow repeating patterns (see Appendix C.1 for details). In particular, we train a model at time t from scratch using all training data seen so far and evaluate the model on test data drawn from label distribution of time $t + 1$. In this setting, training data keeps growing while test data stays the same size. This is a challenging problem as label shift occurs at each time step hence it is critical for a method to learn how to only utilize data which are relevant for the current time step and task. Note we only use test data in the evaluation time and never use it to train a model.

Results. We report results comparing our method against the baselines across different data sets and shifts in Fig. 1. These results clearly illustrate that our method performs better than others as our method obtains higher classification accuracy across all benchmarks. Importantly, our method shows consistent performances across various shifts and benchmarks whereas other methods do not. For instance, **Everything** fares poorly in Gaussian experiment than others but it works better than **Fine-tune** and **Recent** on continuous CIFAR-10 benchmark. We also shows in Fig. 2 that our method produces high quality estimates under shifted distribution.



(a) Gaussian **(b)** CIFAR-10a **(c)** CIFAR-10b **(d)** CLEAR

Figure 1: Average test accuracy (higher is better) across 3 runs for continuous supervised learning benchmarks comparing our method with others. The x-axes indicate different test time steps t and the y-axes display the classification accuracy on test data from time t. CIFAR-10a and CIFAR-10b refer two different shifts for this benchmark (see Appendix C.1 for details). All models in these experiments are trained completely from scratch per time step (i.e. every point in these plots is a different model) using exact same architecture and hyper-parameters. A sudden drop in a plot is an indication that the shift direction has changed. We see that our method is the only approach that consistently works across different settings and shifts.

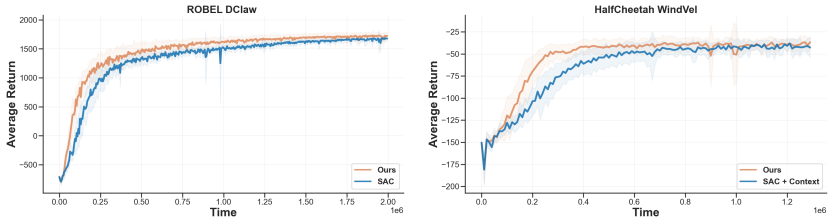


Figure 3: Comparison of the average undiscounted return (higher is better) of our method (orange) against baseline algorithms on ROBEL D'Claw and Half-Cheetah WindVel environments. In all environments, our method performs better than other methods in terms of sample complexity.

4.2 Reinforcement Learning with Changing Tasks

We use two different environments for experiments in this section. For all experiments in this section, we report the undiscounted return averaged across 10 different random seeds.

ROBEL D'Claw. This is a robotic simulated manipulation environment containing 10 different valve rotation tasks and the goal is to turn the valve below 180 degrees in each task. All tasks have the same state and action spaces, and same episode length but each rotates a valve with a different shape. We build a random sequence of these tasks switching from one to another (after an episode finishes) in repeating patterns. We utilize SAC (Haarnoja et al., 2018) as our baseline. In order to incorporate our approach into

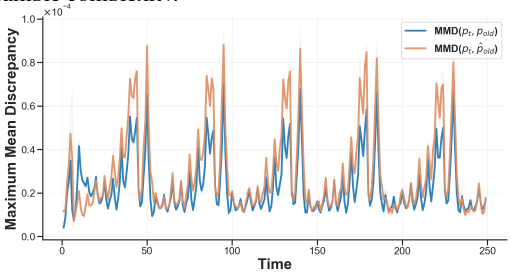
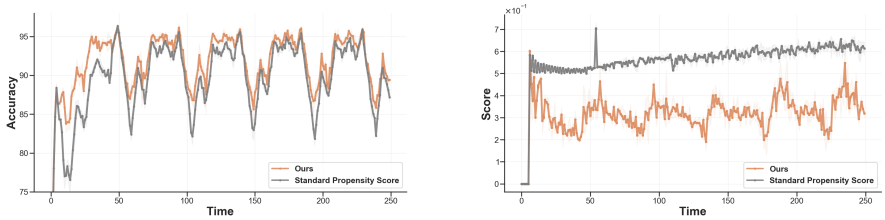


Figure 2: Maximum Mean Discrepancy (MMD) of true distributions (p_t, p_{old}) and $(p_t, \hat{p}_{\text{old}})$ where $\hat{p}_{\text{old}}$ is constructed from past data weighted by our method. Each (blue) point in this plot shows distance between data from $p_t(x)$ and all data from $p_{\text{old}} := \{p_{t_i(x)}\}_{t_i<t}$. Estimates for $\hat{p}_{\text{old}}$ are based on 200 images sampled at each time step from a drifting version of CIFAR-10. As the underlying distributions evolve, their MMD distance grows. To illustrate that our method produces high quality estimates, we also show distance between data from p_t and all data from $\{p_{t_i}(x)\}_{t_i<t}$ which are *weighted* by our ω. The small gap between blue and orange plots shows that our method produces accurate estimates.



(a) (b)

Figure 4: Comparing our method against standard propensity on continuous CIFAR-10 benchmark. Fig 4a shows average test accuracy and Fig 4b shows propensity scores for both methods across time during test time. These results distinctly show our time-varying importance weight estimator weighs data in the same fashion as they evolve and this can explain why our method performs better than standard propensity score where it can not identify when data evolves.

SAC, we only change Q-value update with $\boldsymbol{\omega}$ as follows:

$$\frac{1}{|B|}\sum_{(s,a,t,s')}\left[\boldsymbol{\omega}(s,a,T,t)\Big(Q_\psi(s,a)-r-\gamma Q_{\hat{\psi}}(s',a')+\alpha\log\pi_\phi(a'|s')\Big)^2\right], a'\sim\pi_\phi(\cdot|s') \quad (16)$$

where α is entropy coefficient, T is the current episode, and (s, a) denote state and action, respectively.

Half-Cheetah WindVel. We closely follow the setup of Xie et al. (2020) to create *Half-Cheetah WindVel* environment where direction and magnitude of wind forces on an agent keep changing but slowly and gradually. This is a difficult problem as an agent needs to learn a non-stationary sequence of tasks (i.e. whenever direction and magnitude of wind changes, it constitutes as a new task). Since task information is not directly available to the agent, we utilize the recent method of Caccia et al. (2022) and Fakoor et al. (2020b) to learn and infer task related information (called SAC-context here). Similar to (16), we only change Q-update to add our method.

Results. We can see in Fig. 3 that our method offers consistent improvements over the baseline method and can observe its effectiveness in correcting shifts across time. Note that the only difference between our method and baseline methods is the introduction of $\boldsymbol{\omega}$ term in the Q-update and all other details are exactly the same.

5 Comparing Standard Propensity with Our Time-Varying Method

Now we compare our method with standard propensity scoring discussed in §3. Fig. 4 shows results of this experiment on CIFAR-10 benchmark. This result provides an empirical verification that our method works better than standard propensity method and gives us some insights why it performs better. Particularly, we can see in Fig. 4 that our method fairly detects shifts across different time steps, whereas standard propensity scoring largely ignores shifts across different time steps.

6 Conclusion

Here we propose a simple yet effective time-varying importance weight estimator to account for gradual shifts in data distributions. One practical property of our drift estimator is that it is not specific to a particular class of problems and can be utilized in all kinds of settings from supervised learning to reinforcement learning with a combination of existing approaches. Importantly, our method automatically detects shift in data without knowing them as a priori. Comprehensive experiments on continuous supervised learning and reinforcement learning show the effectiveness of our method. The broader impact of this work will potentially be to make machine learning methods more robust in the presence of continuous distribution shift, which is inevitable in the real world. Note that while we referred to t as time throughout this paper, our methodology is also applicable to settings where data-generating shifts occur gradually over space or any other continuous-valued index.

7 ETHICS STATEMENT

All datasets and benchmarks used in this work are publicly available. All code and documentation developed as a part of this project will made public. We do not declare any conflicts of interest with respect to the findings discussed in this work.

REFERENCES

Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C Fowlkes, Stefano Soatto, and Pietro Perona. Task2vec: Task embedding for meta-learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6430–6439, 2019.

Deepak Agarwal, Lihong Li, and Alexander Smola. Linear-time estimators for propensity scores. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 93–100, 2011.

Amr Alexandari, Anshul Kundaje, and Avanti Shrikumar. Maximum likelihood with bias-corrected calibration is hard-to-beat at label shift adaptation. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 222–232. PMLR, 13–18 Jul 2020.

Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11254–11263, 2019.

J. Baxter. A Model of Inductive Bias Learning. *Journal of Artificial Intelligence Research*, 12: 149–198, March 2000.

Shai Ben-David and Reba Schuller. Exploiting task relatedness for learning multiple tasks. In *Proceedings of the 16th Annual Conference on Learning Theory*, 2003.

Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative learning for differing training and test distributions. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pp. 81–88. Association for Computing Machinery, 2007. ISBN 9781595937933.

Massimo Caccia, Pau Rodriguez, Oleksiy Ostapenko, Fabrice Normandin, Min Lin, Lucas Page-Caccia, Issam Hadj Laradji, Irina Rish, Alexandre Lacoste, David Vázquez, and Laurent Charlin. Online fast adaptation and knowledge accumulation (osaka): a new approach to continual learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 16532–16545. Curran Associates, Inc., 2020.

Massimo Caccia, Jonas Mueller, Taesup Kim, Laurent Charlin, and Rasool Fakoor. Task-agnostic continual reinforcement learning: In praise of a simple baseline, 2022.

Ewen Callaway. The coronavirus is mutating — does it matter?, September 2020. URL https://www.nature.com/articles/d41586-020-02544-6.

Xiangli Chen, Mathew Monfort, Anqi Liu, and Brian D. Ziebart. Robust covariate shift regression. In *AISTATS*, 2016.

D. R. Cox. Regression models and life-tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, 34(2):187–220, 1972. ISSN 00359246.

Rasool Fakoor, Pratik Chaudhari, and Alexander J. Smola. P3o: Policy-on policy-off policy optimization. In *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115, pp. 1017–1027, 2020a.

Rasool Fakoor, Pratik Chaudhari, Stefano Soatto, and Alexander J. Smola. Meta-q-learning. In *International Conference on Learning Representations*, 2020b.

Willliam Feller. *An introduction to probability theory and its applications, vol 2*. John Wiley & Sons, 2008.

Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 1920–1930, 2019.

Yansong Gao and Pratik Chaudhari. An information-geometric distance on the space of tasks. In *ICML*, 2021.

Saurabh Garg, Yifan Wu, Sivaraman Balakrishnan, and Zachary C. Lipton. A unified view of label shift estimation. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, 2020. ISBN 9781713829546.

Heitor Murilo Gomes, Jesse Read, Albert Bifet, Jean Paul Barddal, and João Gama. Machine learning for streaming data: state of the art, challenges, and opportunities. *ACM SIGKDD Explorations Newsletter*, 21(2):6–22, 2019.

Arthur Gretton, Alex Smola, Jiayuan Huang, Marcel Schmittfull, Karsten Borgwardt, and Bernhard Schölkopf. Covariate Shift by Kernel Mean Matching. The MIT Press, 12 2008. ISBN 9780262170055.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pp. 1861–1870, 2018.

Raia Hadsell, Dushyant Rao, Andrei Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 24:1028–1040, 12 2020. doi: 10.1016/j.tics.2020.09.004.

James Harrison, Apoorva Sharma, Chelsea Finn, and Marco Pavone. Continuous meta-learning without tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 17571–17581. Curran Associates, Inc., 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Xu He, Jakub Sygnowski, Alexandre Galashov, Andrei A Rusu, Yee Whye Teh, and Razvan Pascanu. Task agnostic continual learning via meta learning. *arXiv preprint arXiv:1906.05201*, 2019.

James J. Heckman. Sample selection bias as a specification error. *Econometrica*, 47(1):153–161, 1979. ISSN 00129682, 14680262.

Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex Smola. Correcting sample selection bias by unlabeled data. In B. Schölkopf, J. Platt, and T. Hoffman (eds.), *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006.

Alex Krizhevsky, , and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Zhiqiu Lin, Jia Shi, Deepak Pathak, and Deva Ramanan. The clear benchmark: Continual learning on real-world imagery. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

Zachary C. Lipton, Yu-Xiang Wang, and Alex Smola. Detecting and correcting for label shift with black box predictors, 2018.

Bo Lu. Propensity score matching with time-dependent covariates. *Biometrics*, 61(3):721–728, 2005.

Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363, 2018.

Nan Lu, Tianyi Zhang, Tongtong Fang, Takeshi Teshima, and Masashi Sugiyama. Rethinking importance weighting for transfer learning. *arXiv preprint arXiv:2112.10157*, 2021.

Sashank Reddi, Barnabas Poczos, and Alex Smola. Doubly robust covariate shift correction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015a.

Sashank J. Reddi, Barnabás Póczos, and Alexander J. Smola. Doubly robust covariate shift correction. In *AAAI*, 2015b.

Sidney I Resnick. *A probability path*. Springer Science & Business Media, 2013.

Paul R. Rosenbaum and Donald B. Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 04 1983. ISSN 0006-3444.

John Schulman, Sergey Levine, Pieter Abbeel, Michael I Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, volume 37, pp. 1889–1897, 2015.

Shai Shalev-Shwartz. *Online Learning and Online Convex Optimization*. 2012.

Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000. ISSN 0378-3758.

Vinicius Souza, Denis M dos Reis, Andre G Maletzke, and Gustavo EAPA Batista. Challenges in benchmarking stream learning algorithms with real-world data. *Data Mining and Knowledge Discovery*, 34(6):1805–1858, 2020.

Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(5), 2007a.

Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul Buenau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in Neural Information Processing Systems*, volume 20, 2007b.

Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul Von Bunau, and Motoaki Kawanabe. Direct importance estimation for covariate shift adaptation. 2008.

Ryan J Tibshirani, Rina Foygel Barber, Emmanuel Candes, and Aaditya Ramdas. Conformal prediction under covariate shift. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312: 135–153, 2018.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. *arXiv:1611.01224*, 2016.

Junfeng Wen, Chun-Nam Yu, and Russell Greiner. Robust learning under uncertain test distributions: Relating covariate shift to model misspecification. In *International Conference on Machine Learning*, pp. 631–639, 2014.

Ruihan Wu, Chuan Guo, Yi Su, and Kilian Q Weinberger. Online adaptation to label distribution shift. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 11340–11351, 2021.

Annie Xie, James Harrison, and Chelsea Finn. Deep reinforcement learning amidst lifelong non-stationarity, 2020.

Fan Yang, Chao Yang, Huaping Liu, and Fuchun Sun. Evaluations of the gap between supervised and reinforcement lifelong learning on robotic manipulation tasks. In *5th Annual Conference on Robot Learning*, 2021.

Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang. Domain adaptation under target and conditional shift. In *International conference on machine learning*, pp. 819–827, 2013.

A DERIVATIONS

Using a similar calculation as that of (1), we can write our time-varying importance estimator objective as follows

$$\mathbb{E}_{x\sim p_T(x)}\Big[\ell(x)\Big] = \mathbb{E}_{x\sim p(x|T)}\Big[\ell(x)\Big] = \int_t \mathrm{d}p(x|T)\,\ell(x) = \int_t\int_x \mathrm{d}p(t)\,\mathrm{d}p(x|T)\,\frac{\mathrm{d}p(x|t)}{\mathrm{d}p(x|t)}\ell(x) \tag{17}$$

$$\begin{aligned}
&= \int_t\int_x \mathrm{d}p(t)\,\mathrm{d}p(x|t)\,\frac{\mathrm{d}p(x|T)}{\mathrm{d}p(x|t)}\ell(x)\\
&= \int_t \mathrm{d}p(t)\int_x \mathrm{d}p(x|t)\,\frac{\mathrm{d}p(x|T)}{\mathrm{d}p(x|t)}\ell(x)\\
&= \mathbb{E}_{t\sim p(t)}\mathbb{E}_{x\sim p(x|t)}\Big[\frac{\mathrm{d}p(x|T)}{\mathrm{d}p(x|t)}\ell(x)\Big]\\
&= \mathbb{E}_{t\sim p(t)}\mathbb{E}_{x\sim p_t(x)}\Big[\frac{\mathrm{d}p_T(x)}{\mathrm{d}p_t(x)}\ell(x)\Big]\\
&= \mathbb{E}_{t\sim p(t)}\mathbb{E}_{x\sim p_t(x)}\Big[\omega(x,T,t)\ell(x)\Big] \qquad (18)
\end{aligned}$$

B ALGORITHM DETAILS

Algorithm 1 describes the detailed implementation steps of our time-varying importance weight estimator. We use the exact same steps for both reinforcement learning and continuous supervised learning experiments. The main difference between these cases is using of (x, y, t) as inputs for supervised learning tasks whereas (s, a, t) are used as inputs in the reinforcement learning experiments. Note although g_θ can be parameterized differently considering its given task (e.g. image classification tasks require g_θ to be a convolutional deep network; however, continuous control tasks in reinforcement learning experiments need fully connected networks), its implementation details remain the same.

In order to implement Method 2 described in §3, we only need to change line 8 of Algorithm 1 as follow (all other details remain the same):

$$\nabla_\theta J \leftarrow \nabla_\theta \sum_{(x,t_2,t_1,z)\in\mathcal{B}} \log\Big(1 + e^{-z(g_\theta(x,t_2))}\Big) \tag{19}$$

For all the experiments in this paper, we utilize 19 in Algorithm 1 implementation, although both perform similarly.

Algorithm 1 Train Time-varying Importance Weight Estimator

1: **Input**: $\mathcal{D} = \{x_j, t_j\}_{j=1}^N$
2: **Input** M: Number of epochs
3: Initialize $\mathcal{D}_o = \varnothing$
4: **for** j=1...M **do**
5: $\quad\mathcal{D}_o \leftarrow GenerateData(D)$
6: $\quad$**repeat**
7: $\quad\quad$Sample mini-batch $\mathcal{B} = \{(x, t_2, t_1, z)\} \sim \mathcal{D}_o$
8: $\quad\quad\nabla_\theta J \leftarrow \nabla_\theta \sum_{(x,t_2,t_1,z)\in\mathcal{B}} \log\Big(1 + e^{-z(g_\theta(x,t_2)-g_\theta(x,t_1))}\Big)$
9: $\quad\quad\theta \leftarrow \theta - \alpha\nabla_\theta J$
10: $\quad$**until** convergence
11: **end for**

Algorithm 2 $GenerateData$

Input: $\mathcal{D} = \{x_j, t_j\}_{j=1}^{N}$
$\mathcal{T} = [1, .., T]$
Initialize $\mathcal{D}_o = \varnothing$
for j=1...N **do**
  t_r is chosen uniformly at random from $\mathcal{T}$ such that $t_j \neq t_r$
  if random() $>= 0.5$ **then**
    $z = 1$
  else
    $z = -1$
  end if
  $\mathcal{D}_o \leftarrow \mathcal{D}_o \cup \{(x_j, t_j, t_r, z)\}$
end for
Return $\mathcal{D}_o$

Computing Infrastructure	
Machine Type	AWS EC2 - g3.16xlarge
CUDA Version	11.0
NVIDIA-Driver	450.142.00
GPU Family	Tesla M60
CPU Family	Intel Xeon 2.30GHz
Library Version	
Python	3.8.5
Numpy	1.22.0
Pytorch	1.10.0

Table 1: Software libraries and machines used in this paper.

C Experiment Details

Implementation Details. Table 3, 2, and Table 1 show hyper-parameters, computing infrastructure, and libraries used for the experiments in this paper. We did minimal random hyper-parameters search for the experiments in this paper and we mostly follow standard and available settings for the experiments whenever it is applicable. We will open source the code of this paper upon publication.

C.1 Continuous Label Shift settings

To simulate the continuous label shift process for image classification experiments in this paper, we follow the setting of Wu et al. (2021) but adopt their settings to more challenging scenarios. In particular, we simulate label shifts for all classes (i.e. 10 classes here) as opposed to two classes in their setting:

$$\forall i \in [1, C] \;\; \forall t \in [1, \mathcal{T}], v_t = (1 - \frac{t}{\mathcal{T}})q_t^i + (\frac{t}{H})q_t^{i+1}, \tag{20}$$

where $\mathcal{T}$ is the number of shift steps per a label pair i and $i+1$, $t \in [1, \mathcal{T}]$, C denotes the number of classes ($C = 10$ in our experiments), $q_t^i \in R^C$ is vector of class probabilities, and $v_t \in R^{\mathcal{T} \times C}$. For CIFAR-10 experiments, we use $\mathcal{T} = 9$ (called CIFAR-10a), $\mathcal{T} = 6$ (CIFAR-10b), and $\mathcal{T} = 30$ (CIFAR-10c). We also use $\mathcal{T} = 30$ for CLEAR experiment.

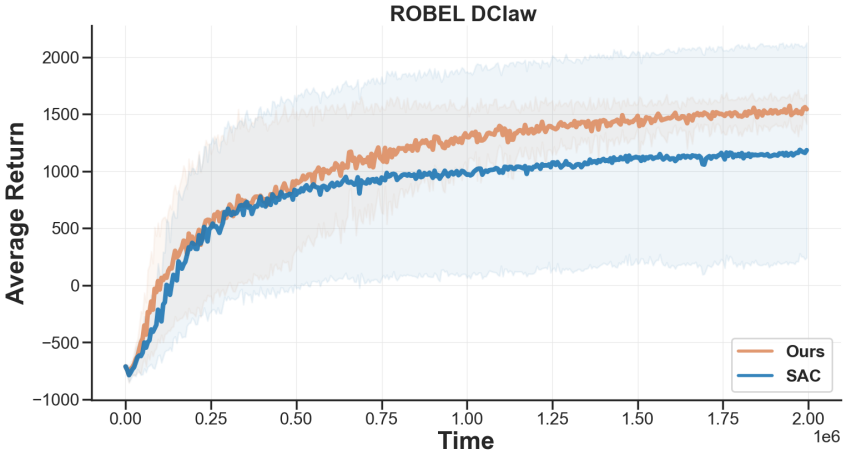
D More results

Hyper-parameters for Algorithm 1	value
Learning rate	1e-4
Number of Updates (M)	4
Batch Size	512
g_θ	4-FC layers with *ReLU* and *BatchNorm*
Hyper-parameters of SAC	**value**
Random Seeds	$\{0, 1, .., 9\}$
Batch Size	256
Number of Q Functions	2
Number of hidden layers (Q)	2 layers
Number of hidden layers (π)	2 layers
Number of hidden units per layer	256
Nonlinearity	*ReLU*
Discount factor (γ)	0.99
Target network (ψ') update rate (τ)	0.005
Actor learning rate	3e-4
Critic learning rate	3e-4
Optimizer	Adam
Replay Buffer size	1e6
Burn-in period	1e4
ω clip	1
HalfCheetah-Wind's number of episodes to evaluate	50
Robel's number of episodes to evaluate	5

Table 2: Hyper-parameters used for all reinforcement learning experiments. All results reported in our paper are averages over repeated runs initialized with *each* of the random seeds listed above and run for the listed number of episodes.

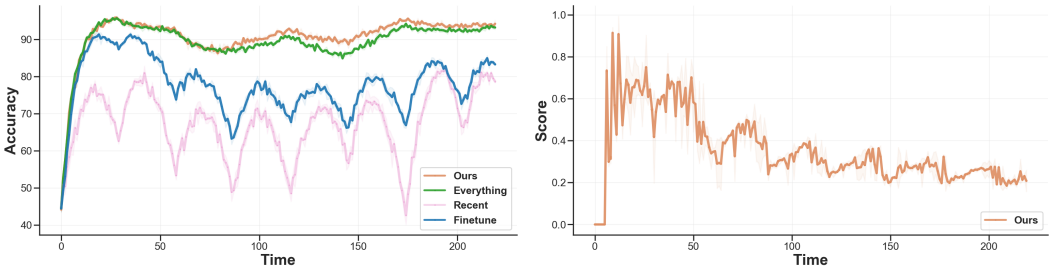
Hyper-parameters for Algorithm 1	value
Learning rate	1e-4
Number of Updates (M)	200
Batch Size	512
g_θ for CIFAR-10	5-Conv layers and 1 linear layer
g_θ for CLEAR	3-FC layers with *ReLU* and *BatchNorm*
Hyper-parameters of CIFAR-10/CLEAR	**value**
Random Seeds	$\{0, 1, 2\}$
CIFAR-10 Network	Resnet-18
CLEAR Network	Linear layer
Samples per time step	200
Learning rate	9e-4
Batch Size	128
Optimizer	Adam
ω clip	1
Number of training epochs per time step	CIFAR-10 (20), CLEAR (25)

Table 3: Hyper-parameters used for all continuous supervised learning experiments for both CIFAR-10 and CLEAR tasks. All continuous supervised learning results reported in our paper are averages over three random seeds.



(a)

Figure 5: Comparison of the average undiscounted return (higher is better) of our method (orange) against baseline algorithm on ROBEL D'Claw. In this environment, our method performs better than standard SAC in terms of both sample complexity and stability (compare shaded area of our method with SAC, larger means less stable). We can see that our method has much more stable performance than SAC. Note the only difference between this experiment and the one in Fig. 3 is that here we only move to next task after repeating each task for 15 time steps whereas each task repeats only for 2 time in the experiment of Fig. 3. Our method works better in both settings and this result again verifies the effectiveness of our method in practice.



(a) (b)

Figure 6: Average test accuracy (higher is better) across 3 runs for continuous supervised learning on CIFAR-10c (see Appendix C.1 for details) benchmark comparing our method with others. The x-axes indicate different test time steps. The y-axis on the left displays the classification accuracy on test data and the one on the right shows our time-varying importance weight scores. Note all models in these experiments are trained completely from scratch per time step (i.e. every point in this plot is a different model) using exact same architectures and hyper-parameters. A sudden drop in a plot is an indication that the shift direction has changed. We see that our method is the only method showing consistent behavior across different benchmarks and shifts, and importantly performs the best overall.