Optimizing for the Future in Non-Stationary MDPs

Anonymous Authors[1]

Abstract

Most reinforcement learning methods are based upon the key assumption that the transition dynamics and reward functions are fixed, that is, the underlying Markov decision process (MDP) is stationary. However, in many practical real-world applications, this assumption is often violated. We discuss how current methods can have inherent limitations for non-stationary MDPs, and therefore searching for a policy that is good for the future, unknown MDP, requires rethinking the optimization paradigm. To address this problem, we develop a method that builds upon ideas from both counter-factual reasoning and curve-fitting to proactively search for a good future policy, without ever modeling the underlying non-stationarity. Interestingly, we observe that *minimizing* performance over some of the data from past episodes might be beneficial when searching for a policy that *maximizes* future performance. The effectiveness of the proposed method is demonstrated on problems motivated by real-world applications.

1. Introduction

The flexibility offered by Markov decision processes (MDPs) to model a large class of problems, combined with modern reinforcement learning (RL) algorithms, has provided several success stories in various domains (Mnih et al., 2013; Silver et al., 2017; Sutton & Barto, 2018). However, real-world applications of reinforcement learning are still lacking (Dulac-Arnold et al., 2019).

One reason for the lack of real-world applications is that most existing methods implicitly assume that the environment remains stationary over time. This assumption is often violated in practical problems of interest. For example, consider an assistive driving system. Over time, tires suffer from wear and tear, leading to increased friction and thus, change in the system dynamics. Similarly, in almost all human-computer interaction applications, e.g., automated medical care, dialogue systems, and marketing, human behavior changes over time. In such scenarios, if the automated system is not adapted to take such changes into account, or if it is adapted only after observing such changes, then the system might quickly become sub-optimal, incurring severe loss (Moore et al., 2014). This raises our main question: *how do we build systems that proactively search for a policy that will be good for the future MDP?*

In this paper we present a policy gradient based approach to search for a policy that maximizes the forecasted future performance when the environment is non-stationary. To capture the impact of changes in the environment on a policy's performance, first, the performance of each policy of interest over the past episodes is estimated using counter-factual reasoning. Subsequently, a regression curve is fit over these estimates to model the performance trend of each policy over time, thereby enabling the prediction of future performance. This performance forecast is then used to develop a gradient-based optimization procedure that can proactively search for a policy that will perform well in the future. The proposed method has several advantages:

1. It does not require modeling the transition or reward function in a non-stationary environment, and thus scales gracefully with respect to the number of states and actions in the environment.

2. For ease of prediction, it concisely models the effect of changes in the environment on a policy's performance using a *univariate* time-series.

3. It is data-efficient in that it leverages all available data, and also mitigates performance lag by proactively optimizing performance for episodes in both the immediate and near future.

4. It degenerates to an estimator of the ordinary policy gradient if the system is stationary, meaning that there is little reason not to use our approach if there is a possibility that the system *might* be non-stationary.

As a passing remark, we note that even when a reinforcement learning agent is being trained on a stationary environment, the observed transition tuples come from a 'non-stationary' distribution. This is due to the changing state

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

distribution induced by updates in the policy parameters over the course of the training. While such non-stationarity exists in our setup as well, it is not the focus of this work. Here, 'non-stationarity' refers to the transition dynamics and reward function of an environment changing across episodes as described further in Section 3.

2. Notation

An MDP $\mathcal{M}$ is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, d^0)$, where $\mathcal{S}$ is the set of possible states, $\mathcal{A}$ is the set of actions, $\mathcal{P}$ is the transition function, $\mathcal{R}$ is the reward function, γ is the discount factor, and d^0 is the start state distribution. Let $\mathcal{R}(s, a)$ denote the expected reward of taking action a in state s. For any given set $\mathcal{X}$, we use $\Delta(\mathcal{X})$ to denote the set of distributions over $\mathcal{X}$. A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ is a distribution over the actions conditioned on the state. When π is parameterized using $\theta \in \mathbb{R}^d$, we denote it as π^θ. In a non-stationary setting, as the MDP changes over time, we use $\mathcal{M}_k$ to denote the MDP during episode k. In general, we will use sub-script k to denote the episode number and a super-script t to denote the time-step within an episode. S_k^t, A_k^t, and R_k^t are the random variables corresponding to the state, action, and reward at time step t, in episode k. Let H_k denote a trajectory in episode k: $(s_k^0, a_k^0, r_k^0, s_k^1, a_k^1, ..., s_k^T)$, where T is the finite horizon. The value function evaluated at state s, during episode k, under a policy π is $v_k^\pi(s) = \mathbb{E}[\sum_{j=0}^{T-t} \gamma^j R_k^{t+j} | S_k^t = s, \pi]$, where conditioning on π denotes that the trajectory in episode k was sampled using π. The start state objective for a policy π, in episode k, is defined as $J_k(\pi) := \sum_s d_0(s) v_k^\pi(s)$. Let $J_k^* = \max_\pi J_k(\pi)$ be the performance of an optimal policy for $\mathcal{M}_k$. Often we write θ in place of π^θ when the dependence on θ is important.

3. Problem Statement

To model non-stationarity, we let an exogenous process change the MDP from $\mathcal{M}_k$ to $\mathcal{M}_{k+1}$, i.e., between episodes. Let $\{\mathcal{M}_k\}_{k=1}^\infty$ represent a sequence of MDPs, where each MDP $\mathcal{M}_k$ is denoted by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}_k, \mathcal{R}_k, \gamma, d^0)$.

In many problems, like adapting to friction in robotics, human-machine interaction, etc., the transition dynamics and rewards function change, but every other aspect of the MDP remains the same throughout. Therefore, we make the following assumption.

Assumption 1 (Similar MDPs)**.** *For any two MDPs, $\mathcal{M}_k$ and $\mathcal{M}_{k+1}$, the state set $\mathcal{S}$, the action set $\mathcal{A}$, the starting distribution d^0, and the discount factor γ are the same.*

If the exogenous process changing the MDPs is arbitrary and changes it in unreasonable ways, then there is little hope of finding a good policy for the future MDP as $\mathcal{M}_{k+1}$ can be wildly different from everything the agent has observed by interacting with the past MDPs, $\mathcal{M}_1, ..., \mathcal{M}_k$. However, in many practical problems of interest, such changes are smooth and have an underlying (unknown) structure. To make the problem tractable, we therefore assume that both the transition dynamics $(\mathcal{P}_1, \mathcal{P}_2, ...)$, and the reward functions $(\mathcal{R}_1, \mathcal{R}_2, ...)$ vary smoothly over time.

Problem Statement. Under the assumptions of similar MDPs and smooth changes, we seek to find a sequence of policies that minimizes lifelong regret:

$$\underset{\{\pi_1, \pi_2, ... \pi_k, ...\}}{\arg\min} \sum_{k=1}^{\infty} J_k^* - \sum_{k=1}^{\infty} J_k(\pi_k).$$

4. Related Work

The problem of non-stationarity has a long history and no effort is enough to thoroughly review it. Here, we briefly touch upon the most relevant work and defer a more detailed literature review to the appendix.

Perhaps the work most closely related to ours is that of Al-Shedivat et al. (2017). They consider a setting where an agent is required to solve test tasks that have different transition dynamics than the training tasks. Using meta-learning, they aim to use training tasks to find an initialization vector for the policy parameters that can be quickly fine-tuned when facing tasks in the test set. In many real-world problems, however, access to such independent training tasks may not be available *a priori*. In this work, we are interested in the continually changing setting where there is no boundary between training and testing tasks. As such, we show how their proposed online adaptation technique that fine-tunes parameters, by discarding past data and only using samples observed online, can create performance lag and can therefore be data-inefficient. In settings where training and testing tasks do exist, our method can be leveraged to better adapt during test time, starting from any desired parameter vector.

Recent work by Finn et al. (2019) aims at bridging both the continuously changing setting and the train-test setting for supervised-learning problems. They propose continuously improving an underlying parameter initialization vector and running a Follow-The-Leader (FTL) algorithm (Shalev-Shwartz et al., 2012) every time new data is observed. A naive adaption of this for RL would require access to all the underlying MDPs in the past for continuously updating the initialization vector, which would be impractical. Doing this efficiently remains an open question and our method is complementary to choosing the initialization vector. Additionally, FTL based adaptation always lags in tracking optimal performance as it uniformly maximizes performance over all the past samples that might not be directly related to the future. Further, we show that by explicitly capturing the trend in the non-stationarity, we can

mitigate this performance lag resulting from the use of an FTL algorithm during the adaptation process.

More importantly, in many real-world applications, it can be infeasible to update the system frequently if it involves high computational or monetary expense. In such a case, even optimizing for the immediate future might be greedy and sub-optimal. The system should optimize for a longer term in the future, to compensate for the time until the next update is performed. None of the prior approaches can efficiently tackle this problem.

5. Optimizing for the Future

The problem of minimizing lifelong regret is straightforward if the agent has access to sufficient samples, in advance, from the future environment, $\mathcal{M}_{k+1}$, that it is going to face (where k denotes the current episode number). That is, if we could estimate the start-state objective, $J_{k+1}(\pi)$, for the future MDP $\mathcal{M}_{k+1}$, then we could search for a policy π whose performance is close to J^*_{K+1}. However, obtaining even a single sample from the future is impossible, let alone getting a sufficient number of samples. This necessitates rethinking the optimization paradigm for searching for a policy that performs well when faced with the future unknown MDP. There are two immediate challenges here:

1. *How can we estimate $J_{k+1}(\pi)$ without any samples from $\mathcal{M}_{k+1}$?*
2. *How can gradients, $\partial J_{k+1}(\pi)/\partial\theta$, of this future performance be estimated?*

In this section we address both of these issues using the following idea. When the transition dynamics $(\mathcal{P}_1, \mathcal{P}_2, ...)$, and the reward functions $(\mathcal{R}_1, \mathcal{R}_2, ...)$ are changing smoothly, the performances $(J_1(\pi), J_2(\pi), ...)$ of any policy π will also vary smoothly over time. The impact of smooth changes in the environment thus manifests as smooth changes in the performance of any policy, π. In cases where there is an underlying, unknown, structure in the changes of the environment, one can now ask: *if the performances $J_{1:k}(\pi) := (J_1(\pi), ..., J_k(\pi))$ of π over the course of past episodes were known, can we analyze the trend in its past performances to find a policy that maximizes future performance $J_{k+1}(\pi)$?*

5.1. Forecasting Future Performance

In this section we address the first challenge of estimating future performance $J_{k+1}(\pi)$ and pose it as a time series forecasting problem.

Broadly, this requires two components: (a) A procedure to compute past performances, $J_{1:k}(\pi)$, of π. (b) A procedure to create an estimate, $\hat{J}_{k+1}(\pi)$, of π's future performance, $J_{k+1}(\pi)$, using these estimated values from component (a). An illustration of this idea is provided in Figure 1.

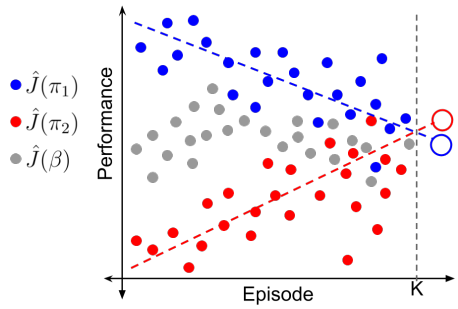

Figure 1. An illustration, where the blue and red filled circles represent counter-factually reasoned performance estimates of policies π_1 and π_2, respectively, using data collected from a given policy β. The open circles represent the forecasted performance of π_1 and π_2 estimated by fitting a curve on the counter-factual estimates represented by filled circles.

Component (a). As we do not have access to the past MDPs for computing the true values of past performances, $J_{1:k}(\pi)$, we propose computing their estimates, $\hat{J}_{1:k}(\pi)$, from the observed data. That is, in a non-stationary MDP, starting with the fixed transition matrix $\mathcal{P}_1$ and the reward function $\mathcal{R}_1$, we want to estimate the performance $J_i(\pi)$ of a given policy in episode $i \leq k$. Leveraging the fact that the changes to the underlying MDP are due to an exogenous processes, we can estimate $J_i(\pi)$ as, $J_i(\pi) = \sum_{t=0}^{T} \gamma^t \mathbb{E}\left[R_i^t | \pi, \mathcal{P}_i, \mathcal{R}_i\right]$, where $\mathcal{P}_i$ and $\mathcal{R}_i$ are also random variables. Next we show how an estimate of $J_i(\pi)$ can be obtained using information only from the ith episode.

To get an unbiased estimate, $\hat{J}_i(\pi)$, of π's performance during episode i, consider the past trajectory H_i of the ith episode that was observed when executing a policy β_i. By using counter-factual reasoning (Rosenbaum & Rubin, 1983) and leveraging the per-decision importance sampling (PDIS) estimator (Precup, 2000), an unbiased estimate of $J_i(\pi)$ is thus given by:[1]

$$\hat{J}_i(\pi) := \sum_{t=0}^{H} \left(\prod_{l=0}^{t} \frac{\pi(A_i^l | S_i^l)}{\beta_i(A_i^l | S_i^l)} \right) \gamma^t R_i^t. \tag{1}$$

It is worth noting that computing (1) does not require storing all the past policies β_i, one needs to only store the actions and the probabilities with which these actions were chosen.

Component (b). To obtain the second component, which captures the structure in $\hat{J}_{1:k}(\pi) := (\hat{J}_1(\pi), ..., \hat{J}_k(\pi))$ and predicts future performances, we make use of a forecasting function Ψ that estimates future performance $\hat{J}_{k+1}(\pi)$

[1] We assume that $\forall i \in \mathbb{N}$ the distribution of H_i has full support over the set of all possible trajectories of the MDP $\mathcal{M}_i$.

conditioned on the past performances:

$$\hat{J}_{k+1}(\theta) := \Psi(\hat{J}_1(\pi), \hat{J}_2(\pi), ..., \hat{J}_k(\pi)). \quad (2)$$

While Ψ can be any forecasting function, we consider Ψ to be an ordinary least squares (OLS) regression model with parameters $w \in \mathbb{R}^{d\times 1}$, and the following input and output variables,

$$X := [1, 2, ..., k]^\top \qquad \in \mathbb{R}^{k\times 1},$$
$$Y := [\hat{J}_1(\pi), \hat{J}_2(\pi), \hat{J}_2(\pi), ..., \hat{J}_k(\pi)]^\top \qquad \in \mathbb{R}^{k\times 1}.$$

For any $x \in X$, let $\phi(x) \in \mathbb{R}^{1\times d}$ denote a d-dimensional basis function for encoding the time index. For example, an identity basis $\phi(x) := \{x, 1\}$, or a Fourier basis, where

$$\phi(x) := \{\sin(2\pi n x | n \in \mathbb{N}_{>0})\} \cup \{\cos(2\pi n x) | n \in \mathbb{N}_{>0}\} \cup \{1\}.$$

Let $\Phi \in \mathbb{R}^{k\times d}$ be the corresponding basis matrix. The solution to above least squares problem is $w = (\Phi^\top \Phi)^{-1}\Phi^\top Y$ (Bishop, 2006) and the forecast of the future performance can be obtained using,

$$\hat{J}_{k+1}(\pi) = \phi(k+1)w = \phi(k+1)(\Phi^\top\Phi)^{-1}\Phi^\top Y. \quad (3)$$

This procedure enjoys an important advantage – by just using a univariate time-series to estimate future performance, it bypasses the need for modeling the environment, which can be prohibitively hard or even impossible. Further, note that $\Phi^\top\Phi \in \mathbb{R}^{d\times d}$, where $d << k$ typically, and thus the cost of computing the matrix inverse is negligible. These advantages allows this procedure to gracefully scale to more challenging problems, while being robust to the size, $|\mathcal{S}|$, of the state set or the action set $|\mathcal{A}|$.

5.2. Differentiating Forecasted Future Performance

In the previous section, we addressed the first challenge and showed how to proactively estimate future performance, $\hat{J}_{k+1}(\theta)$, of a policy π^θ by explicitly modeling the trend in its past performances $\hat{J}_{1:k}(\theta)$. In this section, we address the second challenge to facilitate a complete optimization procedure. A pictorial illustration of the idea is provided in Figure 2.

Gradients for $\hat{J}_{k+1}(\theta)$ with respect to θ can be obtained as follows,

$$\frac{d\hat{J}_{k+1}(\theta)}{d\theta} = \frac{d\Psi(\hat{J}_1(\theta), ..., \hat{J}_k(\theta))}{d\theta} = \sum_{i=1}^{k} \underbrace{\frac{\partial\Psi(\hat{J}_1(\theta), ..., \hat{J}_k(\theta))}{\partial\hat{J}_i(\theta)}}_{(a)} \cdot \underbrace{\frac{d\hat{J}_i(\theta)}{d\theta}}_{(b)}. \quad (4)$$

The decomposition in (4) has an elegant intuitive interpretation. The terms assigned to (a) in (4) correspond to how

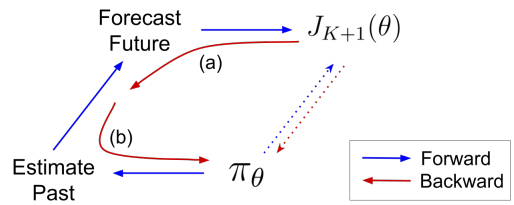

Figure 2. The proposed method from the lens of differentiable programming. At any time k, we aim to optimize policy's parameters, θ, to maximize its performance in the future, $J_{k+1}(\theta)$. However, conventional methods (dotted arrows) can not be used to directly optimize for this. In this work, we achieve this as a composition of two programs: one which connects the policy's parameters to its past performances, and the other which forecasts future performance as a function of these past performances. The optimization procedure then corresponds to taking derivatives through this composition of programs to update policy parameters in a direction that maximizes future performance. Arrows (a) and (b) correspond to the respective terms marked in (4).

the future prediction would change as a function of past outcomes, and the terms in (b) indicate how the past outcomes would change due to changes in the parameters of the policy π^θ. In the next paragraphs, we discuss how to obtain the terms (a) and (b).

To obtain term (a), note that in (3), $\hat{J}_i(\theta)$ corresponds to the ith element of Y, and so using (2) the gradients of the terms (a) in (4) are,

$$\frac{\partial\hat{J}_{k+1}(\theta)}{\partial\hat{J}_i(\theta)} = \frac{\partial\phi(k+1)(\Phi^\top\Phi)^{-1}\Phi^\top Y}{\partial Y_i} = [\phi(k+1)(\Phi^\top\Phi)^{-1}\Phi^\top]_i, \quad (5)$$

where $[Z]_i$ represents the ith element of a vector Z. Therefore, (5) is the *gradient of predicted future performance with respect to an estimated past performance*.

The term (b) in (4) corresponds to the gradient of the PDIS estimate $\hat{J}_i(\theta)$ of the past performance with respect to policy parameters θ. The following Property provides a form for (b) that makes its computation straightforward. See Appendix B for the proof.

Property 1. *Let* $\rho_i(0, l) := \prod_{j=0}^{l} \frac{\pi^\theta(A_i^j|S_i^j)}{\beta_i(A_i^j|S_i^j)}$, *then*

$$\frac{d\hat{J}_i(\theta)}{d\theta} = \sum_{t=0}^{T} \frac{\partial \log \pi^\theta(A_i^t|S_i^t)}{\partial\theta} \left(\sum_{l=t}^{T} \rho_i(0, l)\gamma^l R_i^l\right).$$

5.3. Algorithm

We provide a sketch of our proposed *Prognosticator* procedure for optimizing the future performance of the policy in Algorithm 1. To make the method more practical, we incorporated two additional modifications to reduce computational cost and variance.

Algorithm 1: Prognosticator

Input Learning-rate η, time-duration δ, entropy-regularizer λ
Initialize Forecasting function Ψ, Buffer $\mathbb{B}$
while *True* **do**
  # Record a new batch of trajectories using π^θ
  for $episode = 1, 2, ..., \delta$ **do**
    $h = \{(s_{0:T}, a_{0:T}, \Pr(a_{0:T}|s_{0:T}), r_{0:T})\}$
    $\mathbb{B}$.insert(h)
  # Update for future performance
  for $i = 1, 2, ...$ **do**
    # Evaluate past performances
    for $k = 1, 2, ..., |\mathbb{B}|$ **do**
      $\hat{J}_k(\theta) = \sum_{t=0}^{T} \rho(0,t)\gamma^t R_k^t$ ▷ (1)
    # Future forecast and its gradient
    $\mathcal{L}(\theta) = \frac{1}{\delta}\sum_{\Delta=1}^{\delta} \hat{J}_{k+\Delta}(\theta)$ ▷ (3)
    $\theta \leftarrow \theta + \eta \frac{\partial}{\partial\theta}(\mathcal{L}(\theta) + \lambda\mathcal{H}(\theta))$ ▷ (4)

First, it is often desirable to perform an update only after a certain episode interval δ to reduce computational cost. This raises the question: if a newly found policy will be executed for the next δ episodes, should we choose this new policy to maximize performance on just the single next episode, or to maximize the average performance over the next δ episodes? An advantage of our proposed method is that we can easily tune how far in the future we want to optimize for. Thus, to minimize lifelong regret, we propose optimizing for the mean performance over the next δ episodes. That is, $\arg\max_\theta (1/\delta) \sum_{\Delta=1}^{\delta} \hat{J}_{k+\Delta}(\theta)$.

Second, notice that if the policy becomes too close to deterministic, there would be two undesired consequences. (a) The policy will not cause exploration, precluding the agent from observing any changes to the environment in states that it no longer revisits—changes that might make entering those states worthwhile to the agent. (b) In the future when estimating $\hat{J}_{k+1}(\theta)$ using the *past* performance of θ, importance sampling will have high variance if the policy executed during episode $k+1$ is close to deterministic. To mitigate these issues, we add an entropy regularizer $\mathcal{H}$ during policy optimization. More details are available in Appendix D.

6. Understanding the Behavior of Prognosticator

Notice that as the scalar term (a) is multiplied by the PDIS gradient term (b) in (4), the gradient of future performance can be viewed as a weighted sum of off-policy policy gradients. In Figure 3, we provide visualization of the weights

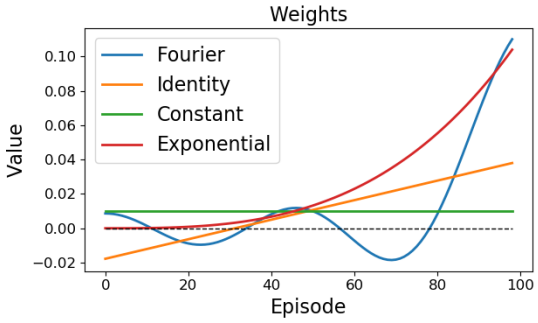

Figure 3. The value of $\frac{\partial \hat{J}_{100}(\theta)}{\partial \hat{J}_i(\theta)}$ for $i \in [1, 99]$ using different basis functions to encode the time index. Notice that many weights are negative when using the identity or Fourier bases.

$\partial\hat{J}_{100}(\theta)/\partial\hat{J}_i(\theta)$ for PDIS gradients of each episode i, when the performance for 100th episode is forecasted using data from the past 99 episodes. For the specific setting when Ψ is an OLS estimator, these weights are independent of Y in (5) and their pattern remains constant for any given sequence of MDPs.

Importantly, note the occurrence of negative weights in Figure 3 when the identity basis or Fourier basis is used, suggesting that the optimization procedure should move towards a policy that had *lower* performance in some of the past episodes. While this negative weighting seems unusual at first glance, it has an intriguing interpretation. We discuss this in the following paragraphs.

To better understand these negative weights, consider a qualitative comparison when weights from different methods in Figure 3 are used along with the performance estimates of policies π_1 and π_2 in Figure 1. Despite having lower estimates of return everywhere, π_2's rising trend suggests that it might have higher performance in the future, that is, $J_{k+1}(\pi_2) > J_{k+1}(\pi_1)$. Existing online learning methods like FTL, maximize performance on all the past data uniformly (green curve in Figure 3). Similarly, the exponential weights (red curve in Figure 3) are representative of approaches that only optimize using data from recent episodes and discard previous data. Either of these methods that use only non-negative weights can never capture the trend to forecast $J_{k+1}(\pi_2) > J_{k+1}(\pi_1)$. However, the weights obtained when using the identity basis would facilitate *minimization* of performances in the distant past and maximization of performance in the recent past. Intuitively, this means that it moves towards a policy whose performance is on a linear rise, as it expects that policy to have better performance in the future.

While weights from the identity basis are useful for forecasting whether $J_{k+1}(\pi_2) > J_{k+1}(\pi_1)$, it cannot be expected that the trend will always be linear as in Figure 1. To be more flexible and allow for any smooth trend, we opt to

use the Fourier basis in our experiments. Observe the alternating sign of weights in Figure 3 when using the Fourier basis. This suggests that the optimization procedure will take into account the *sequential differences* in performances over the past, thereby favoring the policy that has shown the most performance increments in the past. This also avoids restricting the performance trend of a policy to be linear.

7. Mitigating Variance

While model-free algorithms for finding a good policy are scalable to large problems, they tend to suffer from high-variance (Greensmith et al., 2004). In particular, the use of importance sampling estimators can increase the variance further (Guo et al., 2017). In our setup, high variance in estimates of past performances $\hat{J}_{1:k}(\pi)$ of π can hinder capturing π's performance trend, thereby making the forecasts less reliable.

Notice that a major source of variance is the availability of only a *single* trajectory sample per MDP $\mathcal{M}_i$, for all $i \in \mathbb{N}$. If this trajectory H_i, generated using β_i is likely when using β_i, but has near-zero probability when using π then the estimated $\hat{J}_i(\pi)$ is also nearly zero. While $\hat{J}_i(\pi)$ is an unbiased estimate of $J_i(\pi)$, information provided by this single H_i is of little use to evaluate $J_i(\pi)$. Subsequently, discarding this from time-series analysis, rather than setting it to be 0, can make the time series forecast more robust against outliers. In comparison, if trajectory H_i is unlikely when using β_i but likely when using π, then not only is H_i very useful for estimating $J_i(\pi)$ but it also has a lower chance of occurring in the future, so this trajectory must be emphasized when making a forecast. Such a process of (de-)emphasizing estimates of past returns using the collected data itself can introduce bias, but this bias might be beneficial in this few-sample regime.

To capture this idea formally, we build upon the insights of Hachiya et al. (2012) and Mahmood et al. (2014), who draw an equivalence between weighted least-squares (WLS) estimation and the weighted importance sampling (WIS) (Precup, 2000) estimator. Particularly, let $G_i := \sum_{t=0}^{T} \gamma^t R_i^t$ be the discounted return of the ith trajectory observed from a stationary MDP, and $\rho_i^\dagger := \rho_i(0, T)$ be the importance ratio of the entire trajectory. The WIS estimator, $\hat{J}^\dagger(\pi)$, of π's performance in a stationary MDP can then be obtained as,

$$\hat{J}^\dagger(\pi) := \operatorname*{argmin}_{c \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} \rho_i^\dagger (G_i - c)^2 = \frac{\sum_{i=1}^{n} \rho_i^\dagger G_i}{\sum_{i=1}^{n} \rho_i^\dagger}.$$

To mitigate variance in our setup, we propose extending WIS. In the non-stationary setting, to perform WIS while capturing the trend in performance over time, we use a modified forecasting function $\Psi^\dagger$, which is a weighted least-squares regression model with a $d-$dimensional basis function ϕ, and parameters $w^\dagger \in \mathbb{R}^{d \times 1}$,

$$w^\dagger := \operatorname*{argmin}_{c \in \mathbb{R}^{d \times 1}} \frac{1}{n} \sum_{i=1}^{n} \rho_i^\dagger (G_i - c^\top \phi(i))^2. \tag{6}$$

Let $\Lambda \in \mathbb{R}^{k \times k}$ be a diagonal weight matrix such that $\Lambda_{ii} = \rho_i^\dagger$, let $\Phi \in \mathbb{R}^{k \times d}$ be the basis matrix, and let the following be input and output variables,

$$X := [1, 2, ..., k]^\top \qquad \in \mathbb{R}^{k \times 1},$$
$$Y := [G_1, G_2, ..., G_k]^\top \qquad \in \mathbb{R}^{k \times 1}.$$

The solution to the weighted least squares problem in (6) is then given by $w^\dagger = (\Phi^\top \Lambda \Phi)^{-1} \Phi^\top \Lambda Y$ and the forecast of the future performance can be obtained using,

$$\hat{J}_{k+1}^\dagger(\pi) := \phi(k+1) w^\dagger = \phi(k+1)(\Phi^\top \Lambda \Phi)^{-1} \Phi^\top \Lambda Y.$$

$\hat{J}_{k+1}^\dagger(\pi)$ has several desired properties. It incorporates a notion of how relevant each observed trajectory is towards forecasting, while also capturing the desired trend in performance. The forecasts are less sensitive to the importance sampling variances and the process is still differentiable.

8. Strictly Generalizing the Stationary Setting

As the agent is unaware of how the environment is changing, a natural question to ask is what if the agent wrongly assumed a stationary environment was non-stationary? What is the quality of of the agent's performance forecasts? What is the impact of the negative weights on past evaluations of a policy's performance? Here we answer these questions.

Before stating the formal results, we introduce some necessary notation and two assumptions. Let $J(\pi)$ be the performance of policy π for a stationary MDP. Let $\hat{J}_{k+\delta}(\pi)$ and $\hat{J}_{k+\delta}^\dagger(\pi)$ be the non-stationary importance sampling (NIS) and non-stationary weighted importance sampling (NWIS) estimators of performance δ episodes in future. Further, let the basis function ϕ used for encoding the time index in both Ψ and $\Psi^\dagger$ be such that it satisfies the following conditions: (a) $\phi(\cdot)$ always contains 1 to incorporate a bias/intercept coefficient in least-squares regression (for example, $\phi(\cdot) = [\phi_1(\cdot), ..., \phi_{d-1}(\cdot), 1]$, where $\phi_i(\cdot)$ are arbitrary functions). (b) Φ has full column rank such that $(\Phi^\top \Phi)^{-1}$ exists. Both these properties are trivially satisfied by most basis functions.

With this notation and these assumptions, we then have the following results indicating that NIS is unbiased like ordinary importance sampling and NWIS is biased like weighted importance sampling.

Theorem 1 (Unbiased NIS). *For all $\delta \geq 1$, $\hat{J}_{k+\delta}(\pi)$ is an unbiased estimator of $J(\pi)$, that is $\mathbb{E}[\hat{J}_{k+\delta}(\pi)] = J(\pi)$.*

Theorem 2 (Biased NWIS). *For all $\delta \geq 1$, $\hat{J}^{\dagger}_{k+\delta}(\pi)$ may be a biased estimator of $J(\pi)$, that is, $\mathbb{E}[\hat{J}^{\dagger}_{k+\delta}(\pi)] \neq J(\pi)$ always.*

Theorem 3 (Consistent NIS). *For all $\delta \geq 1$, $\hat{J}_{k+\delta}(\pi)$ is a consistent estimator of $J(\pi)$, that is as $N \to \infty$, $\hat{J}_{N+\delta}(\pi) \xrightarrow{a.s.} J(\pi)$.*

Theorem 4 (Consistent NWIS). *For all $\delta \geq 1$, $\hat{J}^{\dagger}_{k+\delta}(\pi)$ is a consistent estimator of $J(\pi)$, that is as $N \to \infty$, $\hat{J}^{\dagger}_{N+\delta}(\pi) \xrightarrow{a.s.} J(\pi)$.*

See Appendix A for all of these proofs. Since NWIS is biased and consistent like the WIS estimator, we expect it to have similar variance reduction properties that can potentially make the optimization process more efficient in a non-stationary MDP.

9. Empirical Analysis

This section presents empirical evaluations using several environments inspired by real-world applications that exhibit non-stationarity. In the following paragraphs, we briefly discuss each environment; a more detailed description is available in Appendix D.

Non-stationary Diabetes Treatment: This environment is based on an open-source implementation (Xie, 2019) of the FDA approved Type-1 Diabetes Mellitus simulator (T1DMS) (Man et al., 2014) for treatment of Type-1 Diabetes. Each episode consists of a day in an *in-silico* patient's life. Consumption of a meal increases the blood-glucose level in the body and if the blood-glucose level becomes too high, then the patient suffers from hyperglycemia and if the level becomes too low, then the patient suffers from hypoglycemia. The goal is to control the blood-glucose level of a patient by regulating the insulin dosage to minimize the risk associated with both hyper and hypoglycemia.

However, the insulin sensitivity of a patient's internal body organs vary over time, inducing non-stationarity that should be accounted for. In the T1DMS simulator, we induce this non-stationarity by oscillating the body parameters (e.g., insulin sensitivity, rate of glucose absorption, etc.) between two known configurations available in the simulator.

Non-stationary Recommender System: In this environment a recommender engine interacts with a user whose interests in different items fluctuate over time. In particular, the rewards associated with each item vary in seasonal cycles. The goal is to maximize revenue by recommending an item that the user is most interested in at any time.

Non-stationary Goal Reacher: This is a 2D environment with four (left, right, up, and down) actions and a continuous state set representing the Cartesian coordinates. The goal is to make the agent reach a moving goal position.

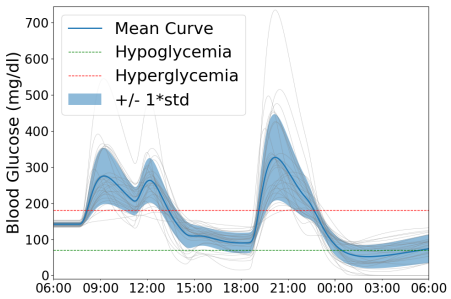

Figure 4. Blood-glucose level of an *in-silico* patient for 24 hours (one episode). Humps in the graph occur at times when a meal is consumed by the patient.

For all of the above environments, we regulate the *speed* of non-stationarity to characterize an algorithms' ability to adapt. Higher speed corresponds to a greater amount of non-stationarity; A speed of zero indicates that the environment is stationary.

We consider the following algorithms for comparison:

Prognosticator: Two variants of our algorithm, **Pro-OLS** and **Pro-WLS**, which use OLS and WLS estimators for Ψ.

ONPG: Similar to the adaptation technique presented by Al-Shedivat et al. (2017), this baseline performs purely online optimization by fine-tuning the existing policy using only the trajectory being observed online.

FTRL-PG: Similar to the adaptation technique presented by Finn et al. (2019), this baseline performs Follow-the-(regularized)-leader optimization by maximizing performance over both the current and all the past trajectories.

9.1. Results

In the non-stationary recommender system, as the exact value of J^*_k is available from the simulator, we can compute the true value of regret. However, for the non-stationary goal reacher and diabetes treatment environment, as J^*_k is not known for any k, we use a surrogate measure for regret. That is, let $\tilde{J}^*_k$ be the maximum return obtained in episode k by any algorithm, then we use $(\sum_{k=1}^{N}(\tilde{J}^*_k - J_k(\pi)))/(\sum_{k=1}^{N}\tilde{J}^*_k)$ as the surrogate regret for a policy π.

In the non-stationary recommender system, all the methods perform nearly the same when the environment is stationary. FTRL-PG has a slight edge over ONPG when the environment is stationary as all the past data is directly indicative of the future MDP. It is interesting to note that while FTRL-PG works the best for the stationary setting in the recommender system and the goal reacher task, it is not the best in the diabetes treatment task as it can suffer from high variance. We discuss the impact of variance in later paragraphs.

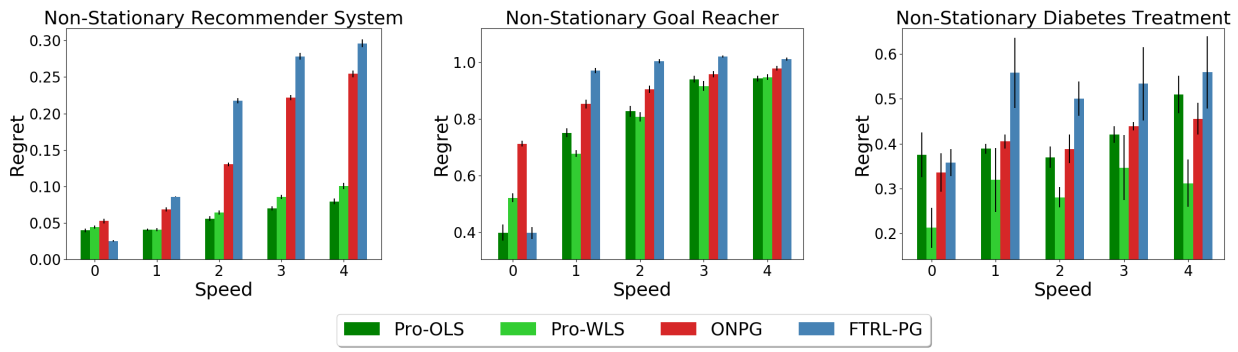

Figure 5. Best performances of all the algorithms obtained by conducting a hyper-parameter sweep over 2000 hyper-parameter combinations per algorithm, per environment. For each hyper-parameter setting, 30 trials were executed for the recommender system and the goal reacher environments, and 10 trials for the diabetes treatment environment. Error bars correspond to the standard error. The x-axis represents how fast the environment is changing and the y-axis represents regret (lower is better). Individual learning curves for each speed, for each domain, is available in Appendix E.

With the increase in the speed of non-stationarity, performance of both the baselines deteriorate quickly. Of the two, ONPG is better able to mitigate performance lag as it discards all the past data. In contrast, both the proposed methods, Pro-OLS and Pro-WLS, can leverage all the past data to capture the impact of non-stationarity and thus are consistently more robust to the changes in the environment.

In the non-stationary goal reacher environment, a similar trend as above is observed. While considering all the past data equally is useful for FTRL-PG in the stationary setting, it creates drastic performance lag as the speed of the non-stationarity increases. Between Pro-OLS and Pro-WLS, in the stationary setting, once the agent nearly solves the task all subsequent trajectories come from nearly the same distribution and thus the variance resulting from importance sampling ratio is not severe. In such a case, where the variance is low, Pro-WLS has less advantage over Pro-OLS and additionally suffers from being biased. However, as the non-stationarity increases, the optimal policy keeps changing and there is a higher discrepancy between distributions of past and current trajectories. This makes the lower variance property of Pro-WLS particularly useful. Having the ability to better capture the underlying trend, both Pro-OLS and Pro-WLS consistently perform better than the baselines when there is non-stationarity.

The non-stationary diabetes treatment environment is particularly challenging as it has a continuous action set. This makes importance sampling based estimators subject to much higher variance. Consequently, Pro-OLS is not able to reliably capture the impact of non-stationarity and performs similar to ONPG. In comparison, the combination of both high variance and performance lag makes FTRL-PG perform poorly across all the speeds. The most advantageous algorithm in this environment is Pro-WLS. As it is designed to better tackle variance stemming from importance sampling, Pro-WLS is able to efficiently use the past data to capture the underlying trend and performs well across all the speeds of non-stationarity.

10. Conclusion

We presented a policy gradient-based algorithm that combines counter-factual reasoning with curve-fitting to proactively search for a good policy for future MDPs. Irrespective of the environment being stationary or non-stationary, the proposed method can leverage all the past data, and in non-stationary settings it can pro-actively optimize for future performance as well. Therefore, our method provides a single solution for mitigating performance lag and being data-efficient.

Our framework highlights new research directions for studying bias-variance trade-offs in the non-stationary setting. While tackling the problem from the point of view of a univariate time-series is advantageous as the model-bias of the environment can be reduced, this can result in higher variance in the forecasted performance. Developing lower variance off-policy performance estimators for $J_k(\pi)$ is also an active research direction which directly complements our algorithm. In particular, often a partial model of the environment is available and using it through doubly-robust estimators (Jiang & Li, 2015; Thomas & Brunskill, 2016) is an interesting future direction.

Further, there are other forecasting functions, like kernel regression, Gaussian Processes, ARIMA, etc., and some break-point detection algorithms that can potentially be used to incorporate more domain knowledge in the forecasting function Ψ, or make Ψ robust to jumps in the time series.

References

Abbasi, Y., Bartlett, P. L., Kanade, V., Seldin, Y., and Szepesvári, C. Online learning in Markov decision processes with adversarially chosen transition probability distributions. In *Advances in Neural Information Processing Systems*, pp. 2508–2516, 2013.

Abdallah, S. and Kaisers, M. Addressing environment non-stationarity by repeating q-learning updates. *The Journal of Machine Learning Research*, 2016.

Al-Shedivat, M., Bansal, T., Burda, Y., Sutskever, I., Mordatch, I., and Abbeel, P. Continuous adaptation via meta-learning in nonstationary and competitive environments. *arXiv preprint arXiv:1710.03641*, 2017.

Basso, E. W. and Engel, P. M. Reinforcement learning in non-stationary continuous time and space scenarios. In *Artificial Intelligence National Meeting*, volume 7, pp. 1–8. Citeseer, 2009.

Bastani, M. Model-free intelligent diabetes management using machine learning. *M.S. Thesis, University of Alberta*, 2014.

Besbes, O., Gur, Y., and Zeevi, A. Stochastic multi-armed-bandit problem with non-stationary rewards. In *Advances in Neural Information Processing Systems*, pp. 199–207, 2014.

Bishop, C. M. *Pattern recognition and machine learning*. springer, 2006.

Bowling, M. Convergence and no-regret in multiagent learning. In *Advances in Neural Information Processing Systems*, pp. 209–216, 2005.

Cheevaprawatdomrong, T., Schochetman, I. E., Smith, R. L., and Garcia, A. Solution and forecast horizons for infinite-horizon nonhomogeneous Markov decision processes. *Mathematics of Operations Research*, 32(1):51–72, 2007.

Cheung, W. C., Simchi-Levi, D., and Zhu, R. Reinforcement learning under drift. *arXiv preprint arXiv:1906.02922*, 2019.

Choi, S. P., Yeung, D.-Y., and Zhang, N. L. An environment model for nonstationary reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 987–993, 2000.

Conitzer, V. and Sandholm, T. Awesome: A general multi-agent learning algorithm that converges in self-play and learns a best response against stationary opponents. *Machine Learning*, 67(1-2):23–43, 2007.

Cuzick, J. A strong law for weighted sums of i.i.d. random variables. *Journal of Theoretical Probability*, 8(3):625–641, 1995.

Dulac-Arnold, G., Mankowitz, D., and Hester, T. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

Even-Dar, E., Kakade, S. M., and Mansour, Y. Experts in a Markov decision process. In *Advances in Neural Information Processing Systems*, pp. 401–408, 2005.

Finn, C., Rajeswaran, A., Kakade, S., and Levine, S. Online meta-learning. *arXiv preprint arXiv:1902.08438*, 2019.

Foerster, J., Chen, R. Y., Al-Shedivat, M., Whiteson, S., Abbeel, P., and Mordatch, I. Learning with opponent-learning awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 122–130. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

Foster, D. J., Li, Z., Lykouris, T., Sridharan, K., and Tardos, E. Learning in games: Robustness of fast convergence. In *Advances in Neural Information Processing Systems*, pp. 4734–4742, 2016.

Gajane, P., Ortner, R., and Auer, P. A sliding-window algorithm for Markov decision processes with arbitrarily changing rewards and transitions. *arXiv preprint arXiv:1805.10066*, 2018.

Garcia, A. and Smith, R. L. Solving nonstationary infinite horizon dynamic optimization problems. *Journal of Mathematical Analysis and Applications*, 244(2):304–317, 2000.

Ghate, A. and Smith, R. L. A linear programming approach to nonstationary infinite-horizon Markov decision processes. *Operations Research*, 61(2):413–425, 2013.

Greene, W. H. *Econometric analysis*. Pearson Education India, 2003.

Greensmith, E., Bartlett, P. L., and Baxter, J. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(Nov): 1471–1530, 2004.

Guo, Z., Thomas, P. S., and Brunskill, E. Using options and covariance testing for long horizon off-policy policy evaluation. In *Advances in Neural Information Processing Systems*, pp. 2492–2501, 2017.

Hachiya, H., Sugiyama, M., and Ueda, N. Importance-weighted least-squares probabilistic classifier for covariate shift adaptation with application to human activity recognition. *Neurocomputing*, 80:93–101, 2012.

Hopp, W. J., Bean, J. C., and Smith, R. L. A new optimality criterion for nonhomogeneous Markov decision processes. *Operations Research*, 35(6):875–883, 1987.

Jacobsen, A., Schlegel, M., Linke, C., Degris, T., White, A., and White, M. Meta-descent for online, continual prediction. In *AAAI Conference on Artificial Intelligence*, 2019.

Jagerman, R., Markov, I., and de Rijke, M. When people change their mind: Off-policy evaluation in non-stationary recommendation environments. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, Melbourne, VIC, Australia, February 11-15, 2019*, 2019.

Jiang, N. and Li, L. Doubly robust off-policy value evaluation for reinforcement learning. *arXiv preprint arXiv:1511.03722*, 2015.

Jong, N. K. and Stone, P. Bayesian models of nonstationary Markov decision processes. *Planning and Learning in A Priori Unknown or Dynamic Domains*, pp. 132, 2005.

Kearney, A., Veeriah, V., Travnik, J. B., Sutton, R. S., and Pilarski, P. M. TIDBD: Adapting temporal-difference step-sizes through stochastic meta-descent. *arXiv preprint arXiv:1804.03334*, 2018.

Lecarpentier, E. and Rachelson, E. Non-stationary Markov decision processes a worst-case approach using model-based reinforcement learning. *arXiv preprint arXiv:1904.10090*, 2019.

Levine, N., Crammer, K., and Mannor, S. Rotting bandits. In *Advances in Neural Information Processing Systems*, pp. 3074–3083, 2017.

Li, C. and de Rijke, M. Cascading non-stationary bandits: Online learning to rank in the non-stationary cascade model. *arXiv preprint arXiv:1905.12370*, 2019.

Li, Y., Zhong, A., Qu, G., and Li, N. Online Markov decision processes with time-varying transition probabilities and rewards. In *Real-world Sequential Decision Making workshop at ICML 2019*, 2019.

Lu, K., Mordatch, I., and Abbeel, P. Adaptive online planning for continual lifelong learning. *arXiv preprint arXiv:1912.01188*, 2019.

Mahmood, A. R., van Hasselt, H. P., and Sutton, R. S. Weighted importance sampling for off-policy learning with linear function approximation. In *Advances in Neural Information Processing Systems*, pp. 3014–3022, 2014.

Mahmud, M. and Ramamoorthy, S. Learning in non-stationary mdps as transfer learning. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pp. 1259–1260. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

Man, C. D., Micheletto, F., Lv, D., Breton, M., Kovatchev, B., and Cobelli, C. The UVA/PADOVA type 1 diabetes simulator: New features. *Journal of Diabetes Science and Technology*, 8(1):26–34, 2014.

Mealing, R. and Shapiro, J. L. Opponent modelling by sequence prediction and lookahead in two-player games. In *International Conference on Artificial Intelligence and Soft Computing*, pp. 385–396. Springer, 2013.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Mohri, M. and Yang, S. Accelerating online convex optimization via adaptive prediction. In *Artificial Intelligence and Statistics*, pp. 848–856, 2016.

Moore, B. L., Pyeatt, L. D., Kulkarni, V., Panousis, P., Padrez, K., and Doufas, A. G. Reinforcement learning for closed-loop propofol anesthesia: A study in human volunteers. *The Journal of Machine Learning Research*, 15(1):655–696, 2014.

Moulines, E. On upper-confidence bound policies for non-stationary bandit problems. *arXiv preprint arXiv:0805.3415*, 2008.

Nagabandi, A., Clavera, I., Liu, S., Fearing, R. S., Abbeel, P., Levine, S., and Finn, C. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018a.

Nagabandi, A., Finn, C., and Levine, S. Deep online learning via meta-learning: Continual adaptation for model-based rl. *arXiv preprint arXiv:1812.07671*, 2018b.

Ornik, M. and Topcu, U. Learning and planning for time-varying mdps using maximum likelihood estimation. *arXiv preprint arXiv:1911.12976*, 2019.

Padakandla, S., J., P. K., and Bhatnagar, S. Reinforcement learning in non-stationary environments. *CoRR*, abs/1905.03970, 2019.

Precup, D. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, 2000.

Rakhlin, A. and Sridharan, K. Online learning with predictable sequences. *arXiv preprint arXiv:1208.3728*, 2013.

Ring, M. B. *Continual learning in reinforcement environments*. PhD thesis, University of Texas at Austin, Texas 78712, 1994.

Rosenbaum, P. R. and Rubin, D. B. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983.

Schmidhuber, J. A general method for incremental self-improvement and multi-agent learning. In *Evolutionary Computation: Theory and Applications*, pp. 81–123. World Scientific, 1999.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Seznec, J., Locatelli, A., Carpentier, A., Lazaric, A., and Valko, M. Rotting bandits are no harder than stochastic ones. *arXiv preprint arXiv:1811.11043*, 2018.

Shalev-Shwartz, S. et al. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2012.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Singh, S., Kearns, M., and Mansour, Y. Nash convergence of gradient dynamics in general-sum games. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pp. 541–548. Morgan Kaufmann Publishers Inc., 2000.

Sinha, S. and Ghate, A. Policy iteration for robust nonstationary Markov decision processes. *Optimization Letters*, 10(8):1613–1628, 2016.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Thomas, P. and Brunskill, E. Data-efficient off-policy policy evaluation for reinforcement learning. In *International Conference on Machine Learning*, pp. 2139–2148, 2016.

Thomas, P. S. *Safe reinforcement learning*. PhD thesis, University of Massachusetts Libraries, 2015.

Thomas, P. S., Theocharous, G., Ghavamzadeh, M., Durugkar, I., and Brunskill, E. Predictive off-policy policy evaluation for nonstationary decision problems, with applications to digital marketing. In *Twenty-Ninth Innovative Applications of Artificial Intelligence Conference*, 2017.

Thrun, S. Lifelong learning algorithms. In *Learning to learn*, pp. 181–209. Springer, 1998.

Wagener, N., Cheng, C.-A., Sacks, J., and Boots, B. An online learning approach to model predictive control. *arXiv preprint arXiv:1902.08967*, 2019.

Wang, J.-K., Li, X., and Li, P. Optimistic adaptive acceleration for optimization. *arXiv preprint arXiv:1903.01435*, 2019a.

Wang, L., Zhou, H., Li, B., Varshney, L. R., and Zhao, Z. Be aware of non-stationarity: Nearly optimal algorithms for piecewise-stationary cascading bandits. *arXiv preprint arXiv:1909.05886*, 2019b.

Xie, J. *Simglucose v0.2.1 (2018)*, 2019. URL https://github.com/jxx123/simglucose.

Yang, S. and Mohri, M. Optimistic bandit convex optimization. In *Advances in Neural Information Processing Systems*, pp. 2297–2305, 2016.

Yu, J. Y. and Mannor, S. Online learning in Markov decision processes with arbitrarily changing rewards and transitions. In *2009 International Conference on Game Theory for Networks*, pp. 314–322. IEEE, 2009.

Zhang, C. and Lesser, V. Multi-agent learning with policy prediction. In *Twenty-fourth AAAI conference on artificial intelligence*, 2010.

Appendix

A. Properties of NIS and NWIS Estimators

Here we provide proofs for the properties of NIS and NWIS estimators. While NIS and NWIS are developed for non-stationary setting, these properties ensure that these estimators *strictly generalize* for the stationary setting as well. That is, when used in stationary setting, NIS estimator is both unbiased and consistent like the PDIS estimator, and NWIS estimator is biased and consistent like the WIS estimator; when NIS and NWIS are used in non-stationary setting they can provide more accurate estimates of a future performance as they explicitly model the trend of policy's performance over time.

Our proof technique draws inspiration from the results presented by Mahmood et al. (2014). The primary difference is that their results are for estimators that are developed only for the stationary setting and can not be readily used for the non-stationary setting, unlike ours. The key modification that we make to leverage their proof technique is that instead of using the features of the state as the input and the observed return from that corresponding state as the output to the regression function, we use the features of the *time index of an episode* as the input and the observed return for that corresponding episode as the output. In their setup, as states are stochastically drawn from a distribution their analysis is not directly applicable for our setting, where inputs (time indices) form a deterministic sequence. For analysis of our estimators, we leverage techniques discussed by Greene (2003) for analyzing properties of the ordinary least squares estimator.

Before proceeding, we impose the following constraints on the set of policies, and the basis function ϕ used for encoding the time index in both Ψ and $\Psi^{\dagger}$.

(a) $\phi(\cdot)$ always contains 1 to incorporate a bias coefficient in least-squares regression (for example, $\phi(\cdot) = [\phi_1(\cdot), ..., \phi_{d-1}(\cdot), 1]$, where $\phi_i(\cdot)$ are arbitrary functions).

(b) There exists a finite constant C_1, such that $\forall i,\ |\phi_i(\cdot)| < C_1$.

(c) Φ has full column rank such that $(\Phi^\top \Phi)^{-1}$ exists.

(d) We only consider set of policies Π that have non-zero probability of taking any action in any state. That is $\exists C_2 > 0$, such that $\forall \pi \in \Pi, \forall s \in \mathcal{S}, \forall a \in \mathcal{A}, \pi(a|s) > C_2$.

Satisfying condition (a) is straightforward as it is already done by all basis functions. Intuitively, this constraint ensure that the regression problem is not ill-defined and there exists a model in the model class that can capture a fixed constant, which corresponds to absence of any trend. This is useful for our purpose as in the stationary setting, there exists no trend in the expected performance between the episodes for any given policy. The forecaster should be capable to infer this fact from the data and represent it. That is, the optimal model parameter is [2]

$$w^* = [0, 0, ..., 0, J(\pi)]^\top,$$

such that for any k,

$$\phi(k)w^* = [\phi_1(k), ..., \phi_{d-1}(k), 1][0, ..., 0, J(\pi)]^\top = J(\pi). \tag{7}$$

Conditions (b) and (c) are also readily satisfied by popular basis functions. For example, features obtained using Fourier basis is bounded by $[-1, 1]$, and features from polynomial/identity basis are also bounded when inputs are adequately normalized. Further, when the basis function does not repeat any feature, and number of samples are more than number of features, condition (c) is satisfied. This ensures the the least-squares problem is well-defined and has a unique-solution.

Condition (d) ensures that the denominator in any importance ratio is always bounded below, such that the importance ratios are bounded above. This condition implies that the importance sampling estimator for any evaluation policy has finite variance. Use of entropy regularization with common policy parameterizations (softmax/Gaussian) can prevent violation of this condition.

[2] If the domain knowledge is available to select an appropriate basis function that can be used to represent the performance trend of all the policies for the required non-stationary environment, then all the following finite-sample and large-sample properties can be extended for that environment as well.

A.1. Finite Sample Properties

In this subsection, finite sample properties of NIS and NWIS are presented. Specifically, it is established that NIS is an unbiased estimator, whereas NWIS is a biased estimator of $J(\pi)$, the performance of a policy π in a stationary MDP.

Theorem 1 (Unbiased NIS)**.** *For all $\delta \geq 1$, $\hat{J}_{k+\delta}(\pi)$ is an unbiased estimator of $J(\pi)$, that is $\mathbb{E}[\hat{J}_{k+\delta}(\pi)] = J(\pi)$.*

Proof. Recall from (3) that

$$\hat{J}_{k+\delta}(\pi) = \phi(k+\delta)w = \phi(k+\delta)(\Phi^\top\Phi)^{-1}\Phi^\top Y.$$

Therefore, the expected value of $\hat{J}_{k+\delta}(\pi)$ is

$$\begin{aligned}\mathbb{E}[\hat{J}_{k+\delta}(\pi)] &= \mathbb{E}\left[\phi(k+\delta)(\Phi^\top\Phi)^{-1}\Phi^\top Y\right] \\ &= \phi(k+\delta)\left(\Phi^\top\Phi\right)^{-1}\left(\Phi^\top\mathbb{E}\left[Y\right]\right)\end{aligned}$$

As $Y = [\hat{J}_0(\pi), ..., \hat{J}_k(\pi)]^\top$ and the MDP is stationary, expected value of each element of Y is $J(\pi)$. Let $[J(\pi)]$ denote the vector of size similar to Y, where all elements are $J(\pi)$, then

$$\mathbb{E}[\hat{J}_{k+\delta}(\pi)] = \phi(k+\delta)\left(\Phi^\top\Phi\right)^{-1}\left(\Phi^\top[J(\pi)]\right) \tag{8}$$

Now using (7) in (8),

$$\begin{aligned}\mathbb{E}[\hat{J}_{k+\delta}(\pi)] &= \phi(k+\delta)\left(\Phi^\top\Phi\right)^{-1}\left(\Phi^\top\Phi w^*\right) \\ &= \phi(k+\delta)\left(\Phi^\top\Phi\right)^{-1}\left(\Phi^\top\Phi\right)w^* \\ &= \phi(k+\delta)w^* \\ &= J(\pi).\end{aligned}$$

□

Proof. (Alternate) Here we present an alternate proof for Theorem 1 which does not require invoking w^*.

$$\begin{aligned}\mathbb{E}\left[\hat{J}_{k+\delta}(\pi)\right] &= \mathbb{E}\left[\phi(k+\delta)(\Phi^\top\Phi)^{-1}\Phi^\top Y\right] \\ &\overset{(a)}{=} \mathbb{E}\left[\sum_{i=0}^{k}\left[\phi(k+\delta)(\Phi^\top\Phi)^{-1}\Phi^\top\right]_i Y_i\right] \\ &\overset{(b)}{=} \sum_{i=0}^{k}\left[\phi(k+\delta)(\Phi^\top\Phi)^{-1}\Phi^\top\right]_i \mathbb{E}\left[Y_i\right],\end{aligned}$$

where (a) is the dot product written as summation, and (b) holds because the multiplicative constants are fixed values, as given in (5). Since the environment is stationary, $\forall i\ \mathbb{E}\left[Y_i\right] = J(\pi)$, therefore,

$$\mathbb{E}\left[\hat{J}_{k+1}(\pi)\right] = J(\pi)\sum_{i=0}^{k}\left[\phi(k+\delta)(\Phi^\top\Phi)^{-1}\Phi^\top\right]_i. \tag{9}$$

In the following we focus on the terms inside the summation in (9). Without loss of generality, assume that the for a given matrix of features Φ, the feature corresponding to value 1 is in the last column of Φ. Let $\mathbf{A} := (\Phi^\top\Phi)^{-1}\Phi^\top \in \mathbb{R}^{d\times k}$, and let $B := \Phi[1, ..., k; 1, ..., k-1] \in \mathbb{R}^{k\times(d-1)}$ be the submatrix of Φ such that $\mathbf{B}$ has all features of Φ except the ones column, $\mathbb{1} \in \mathbb{R}^{k\times 1}$. Let $\mathbf{I}$ be the identity matrix $\in \mathbb{R}^{k\times k}$, then it can seen that $(\Phi^\top\Phi)^{-1}(\Phi^\top\Phi)$ can be expressed as,

$$\left[\ \mathbf{A}\ \right]\left[\ \mathbf{B}\ \middle|\ \mathbb{1}\ \right] = \mathbf{I}. \tag{10}$$

In (10), as the jth row in last column of $\mathbf{I}$ corresponds to the dot product of the jth row of $\mathbf{A}$, $\mathbf{A_j}$, with $\mathbb{1}$,

$$\mathbf{A}_j\,\mathbb{1} = \begin{cases} 0 & j \neq d, \\ 1 & j = d. \end{cases} \tag{11}$$

Equation (11) ensures that the summation of all rows of $\mathbf{A}$, except the last, sum to 0, and the last one sums to 1. Now, let $\phi(k+\delta) := [\phi_1(k+\delta), \phi_2(k+\delta), ..., \phi_{d-1}(k+\delta), 1] \in \mathbb{R}^{1\times d}$. Therefore,

$$\begin{aligned}
\sum_{i=1}^{k} \left[\phi(k+\delta)(\Phi^\top\Phi)^{-1}\Phi^\top\right]_i &= \sum_{i=1}^{k} \left[\phi(k+\delta)\mathbf{A}\right]_i \\
&= \sum_{i=1}^{k}\sum_{j=1}^{d} \left[\phi(k+\delta)\right]_j \mathbf{A}_{j,i} \\
&= \sum_{j=1}^{d} \left[\phi(k+\delta)\right]_j \sum_{i=1}^{k} \mathbf{A}_{j,i} \\
&= \left(\sum_{j=1}^{d-1} \left[\phi(k+\delta)\right]_j \sum_{i=1}^{k} \mathbf{A}_{j,i}\right) + \left(\left[\phi(k+\delta)\right]_d \sum_{i=1}^{k} \mathbf{A}_{d,i}\right) \\
&= \left(\sum_{j=1}^{d-1} \left[\phi(k+\delta)\right]_j \left(\mathbf{A}_j \mathbb{1}\right)\right) + \left(\left[\phi(k+\delta)\right]_d \left(\mathbf{A}_d \mathbb{1}\right)\right) \\
&= \left(\sum_{j=1}^{d-1} \left[\phi(k+\delta)\right]_j \cdot 0\right) + \left(\left[\phi(k+\delta)\right]_d \cdot 1\right) \\
&= \left[\phi(k+\delta)\right]_d \\
&= 1. \qquad (12)
\end{aligned}$$

Therefore, combining (12) with (9), $\mathbb{E}\left[\hat{J}_{k+\delta}(\pi)\right] = J(\pi)$. □

Theorem 2 (Biased NWIS)**.** *For all $\delta \geq 1$, $\hat{J}^{\dagger}_{k+\delta}(\pi)$ may be a biased estimator of $J(\pi)$, that is, $\mathbb{E}[\hat{J}^{\dagger}_{k+\delta}(\pi)] \neq J(\pi)$ always.*

Proof. We prove this result using a simple counter-example. Consider the following basis function, $\phi(\cdot) = [1]$, then,

$$\begin{aligned}
J^{\dagger}_{k+\delta}(\pi) &= \phi(k+\delta)w^{\dagger} \\
&= \phi(k+\delta) \operatorname*{argmin}_{c\in\mathbb{R}^{1\times 1}} \frac{1}{n}\sum_{i=1}^{n} \rho_i(0,T)(G_i - c^\top\phi(i))^2 \\
&= \operatorname*{argmin}_{c\in\mathbb{R}^{1\times 1}} \frac{1}{n}\sum_{i=1}^{n} \rho_i(0,T)(G_i - c)^2 \\
&= \frac{\sum_{i=1}^{n} \rho_i(0,T)G_i}{\sum_{i=1}^{n} \rho_i(0,T)},
\end{aligned}$$

which is the WIS estimator. Therefore, as WIS is a biased estimator, NWIS is also a biased estimator of $J(\pi)$. □

A.2. Large Sample Properties

In this subsection, large sample properties of NIS and NWIS are presented. Specifically, it is established that both NIS and NWIS are consistent estimators of, $J(\pi)$, the performance of a policy π in a stationary MDP.

Theorem 3 (Consistent NIS)**.** *For all $\delta \geq 1$, $\hat{J}_{k+\delta}(\pi)$ is a consistent estimator of $J(\pi)$, that is as $N \to \infty$, $\hat{J}_{N+\delta}(\pi) \xrightarrow{a.s.} J(\pi)$.*

Proof. Using (3),

$$\begin{aligned}
\lim_{N\to\infty} \quad \hat{J}_{N+\delta}(\pi) &= \lim_{N\to\infty} \quad \phi(N+\delta)w \\
&= \lim_{N\to\infty} \quad \phi(N+\delta)(\Phi^\top\Phi)^{-1}\Phi^\top Y.
\end{aligned}$$

As $Y = [\hat{J}_0(\pi), ..., \hat{J}_N(\pi)]^\top$ and the MDP is stationary, each element of Y is an unbiased estimate of $J(\pi)$. In other words, $\forall i \in [0, k], \hat{J}_i(\pi) = J(\pi) + \epsilon_i$, where ϵ_i is a mean zero error. Let $\epsilon \in \mathbb{R}^{N\times 1}$ be the vector containing all the error terms ϵ_i. Now using (7),

$$\begin{aligned}
\lim_{N\to\infty} \quad \hat{J}_{N+\delta}(\pi) &= \lim_{N\to\infty} \quad \phi(N+\delta)\left(\Phi^\top\Phi\right)^{-1}\left(\Phi^\top(\Phi w^* + \epsilon)\right) \\
&= \lim_{N\to\infty} \quad \phi(N+\delta)\left(\Phi^\top\Phi\right)^{-1}\left(\left(\Phi^\top\Phi\right)w^* + \left(\Phi^\top\epsilon\right)\right) \\
&= \lim_{N\to\infty} \quad \phi(N+\delta)w^* + \phi(N+\delta)\left(\Phi^\top\Phi\right)^{-1}\left(\Phi^\top\epsilon\right) \\
&= \lim_{N\to\infty} \quad J(\pi) + \phi(N+\delta)\left(\Phi^\top\Phi\right)^{-1}\left(\Phi^\top\epsilon\right) \\
&= \lim_{N\to\infty} \quad J(\pi) + \phi(N+\delta)\left(\frac{1}{N}\Phi^\top\Phi\right)^{-1}\left(\frac{1}{N}\Phi^\top\epsilon\right).
\end{aligned} \tag{13}$$

Using Slutsky's Theorem,

$$\begin{aligned}
\lim_{N\to\infty} \quad \hat{J}_{N+\delta}(\pi) &= J(\pi) + \phi(N+\delta)\left(\lim_{N\to\infty} \quad \frac{1}{N}\Phi^\top\Phi\right)^{-1}\left(\lim_{N\to\infty} \quad \frac{1}{N}\Phi^\top\epsilon\right) \\
&= J(\pi) + \phi(N+\delta)Q^{-1}\left(\lim_{N\to\infty} \quad \frac{1}{N}\Phi^\top\epsilon\right),
\end{aligned} \tag{14}$$

where $Q = \left(\lim_{N\to\infty} \frac{1}{N}\Phi^\top\Phi\right)$, and it holds from Grenander's conditions that Q^{-1} exists. Informally, Grenander's conditions require that no feature degenerates to a sequence of zeros, no feature of a single observation dominates the sum of squares of its series, and the $\Phi^\top\Phi$ matrix always has full rank. These conditions are easily satisfied for most popular basis functions used to create input features. For formal definitions of these conditions, we refer the reader to Chpt. 5, Greene (2003).

In the following, we restrict our focus to the term inside the brackets in the second term of (14). The mean of that term is,

$$\mathbb{E}\left[\frac{1}{N}\Phi^\top\epsilon\right] = \frac{1}{N}\Phi^\top\mathbb{E}\left[\epsilon\right] = 0.$$

Since the mean is 0, variance of that term is given by,

$$\mathbb{V}\left[\frac{1}{N}\Phi^\top\epsilon\right] = \frac{1}{N^2}\mathbb{V}\left[\Phi^\top\epsilon\right] = \frac{1}{N^2}\left(\Phi^\top\epsilon\right)\left(\Phi^\top\epsilon\right)^\top = \frac{1}{N^2}\left(\Phi^\top\epsilon\epsilon^\top\Phi\right).$$

As each policy has a non-zero probability of taking any action in any state, the variance of PDIS (or the standard IS) estimator is bounded and thus each element of $\epsilon\epsilon^\top$ is bounded. Further, as $\phi_i(\cdot)$ is bounded, each element of Φ is also bounded. Therefore, as

$$\lim_{N\to\infty} \quad \mathbb{V}\left[\frac{1}{N}\Phi^\top\epsilon\right] \xrightarrow{\text{a.s.}} 0.$$

As mean is 0 and variance asymptotes to 0, then as $N \to \infty$, $\frac{1}{N}\Phi^\top\epsilon \xrightarrow{\text{a.s.}} 0$. Combining this with (14),

$$\lim_{N\to\infty} \hat{J}_{N+\delta}(\pi) \overset{a.s.}{\to} J(\pi) + \phi(N+\delta)Q^{-1}0 = J(\pi).$$

□

Theorem 4 (Consistent NWIS)**.** *For all $\delta \geq 1$, $\hat{J}^\dagger_{k+\delta}(\pi)$ is a consistent estimator of $J(\pi)$, that is as $N \to \infty$, $\hat{J}^\dagger_{N+\delta}(\pi) \xrightarrow{a.s.} J(\pi)$.*

Proof. Recall from (6),

$$\hat{J}^\dagger_{N+\delta}(\pi) = \phi(N+\delta)w^\dagger = \phi(N+\delta)(\Phi^\top\Lambda\Phi)^{-1}\Phi^\top\Lambda Y.$$

Consistency of $\hat{J}^{\dagger}_{N+\delta}(\pi)$ can be proved similarly to the proof of Theorem 3. Note that over here $Y = [G_0, ..., G_k]^\top$ contains the returns for each episode, ΛY denotes the unbiased estimates for $J(\pi)$. Therefore, similar to (13),

$$\begin{aligned}\lim_{N\to\infty} \quad \hat{J}^{\dagger}_{N+\delta}(\pi) &= \lim_{N\to\infty} \quad \phi(N+\delta)(\Phi^\top\Lambda\Phi)^{-1}(\Phi^\top(\Phi w^* + \epsilon)) \\ &= \lim_{N\to\infty} \quad \phi(N+\delta)(\Phi^\top\Lambda\Phi)^{-1}((\Phi^\top\Phi)w^* + \Phi^\top\epsilon) \\ &= \lim_{N\to\infty} \quad \phi(N+\delta)\left(\frac{1}{N}\Phi^\top\Lambda\Phi\right)^{-1}\left(\left(\frac{1}{N}\Phi^\top\Phi\right)w^* + \frac{1}{N}\Phi^\top\epsilon\right)\end{aligned}$$

Using Slutsky's Theorem,

$$\lim_{N\to\infty} \quad \hat{J}^{\dagger}_{N+\delta}(\pi) = \phi(N+\delta)\left(\lim_{N\to\infty} \quad \frac{1}{N}\Phi^\top\Lambda\Phi\right)^{-1}\left(\left(\lim_{N\to\infty} \quad \frac{1}{N}\Phi^\top\Phi\right)w^* + \lim_{N\to\infty} \quad \frac{1}{N}\Phi^\top\epsilon\right). \tag{15}$$

Now restricting our focus to the terms in the first bracket in (15),

$$\begin{aligned}\left(\lim_{N\to\infty} \quad \frac{1}{N}\Phi^\top\Lambda\Phi\right)^{-1} &= \left(\lim_{N\to\infty} \quad \frac{1}{N}\sum_{k=1}^{N}\rho^{\dagger}_k\phi(k)^\top\phi(k)\right)^{-1} \\ &\overset{(a)}{=} \left(\mathbb{E}\left[\rho^{\dagger}_k\right]\left(\lim_{N\to\infty} \quad \frac{1}{N}\sum_{k=1}^{N}\phi(k)^\top\phi(k)\right)\right)^{-1} \\ &\overset{(b)}{=} \left(\lim_{N\to\infty} \quad \frac{1}{N}\Phi^\top\Phi\right)^{-1},\end{aligned} \tag{16}$$

where to obtain (a), let $\tilde{\rho}_k := \rho^{\dagger}_k - \mathbb{E}[\rho^{\dagger}_k]$ be a mean 0 random variable. Then $(1/N)\sum\rho^{\dagger}_k\phi(k)^\top\phi(k) = (1/N)(\sum_{k=1}^{N}\tilde{\rho}_k\phi(k)^\top\phi(k) + \mathbb{E}[\rho^{\dagger}_k]\sum_{k=1}^{N}\phi(k)^\top\phi(k))$. Now using the strong law for weighted sums of independent random variables (Theorem 1.1, Cuzick (1995)) and under the condition that both $\rho^{\dagger}$ and $\phi(\cdot)$ are bounded, $(1/N)\sum_{k=1}^{N}\tilde{\rho}_k\phi(k)^\top\phi(k) \overset{a.s.}{\to} 0$ in limit. Therefore, $(1/N)\sum\rho^{\dagger}_k\phi(k)^\top\phi(k) = \mathbb{E}[\rho^{\dagger}_k]((1/N)\sum_{k=1}^{N}\phi(k)^\top\phi(k))$ in limit as $N \to \infty$. Consequently, (b) is obtained using the fact that the expected value of importance ratios is 1 (Lemma 3, (Thomas, 2015)). Substituting (16) in (15),

$$\begin{aligned}\lim_{N\to\infty} \quad \hat{J}^{\dagger}_{N+\delta}(\pi) &= \phi(N+\delta)\left(\lim_{N\to\infty} \quad \frac{1}{N}\Phi^\top\Phi\right)^{-1}\left(\left(\lim_{N\to\infty} \quad \frac{1}{N}\Phi^\top\Phi\right)w^* + \lim_{N\to\infty} \quad \frac{1}{N}\Phi^\top\epsilon\right) \\ &= \phi(N+\delta)w^* + \phi(N+\delta)\left(\lim_{N\to\infty} \quad \frac{1}{N}\Phi^\top\Phi\right)^{-1}\left(\lim_{N\to\infty} \quad \frac{1}{N}\Phi^\top\epsilon\right) \\ &= J(\pi) + \phi(N+\delta)\left(\lim_{N\to\infty} \quad \frac{1}{N}\Phi^\top\Phi\right)^{-1}\left(\lim_{N\to\infty} \quad \frac{1}{N}\Phi^\top\epsilon\right) \\ &\xrightarrow{\text{a.s.}} J(\pi),\end{aligned} \tag{17}$$

where (17) follows from simplification used for (14) in the proof of Theorem 3. □

B. Gradient of PDIS Estimator

Recall that NIS and NWIS estimators build upon PDIS and WIS estimators by using them along with OLS and WLS regression, respectively. Consequently, gradients of NIS and NWIS estimators with respect to the policy parameters can be decomposed into terms which consists of gradients of PDIS and WIS estimators with respect to the policy parameters. Here we provide complete derivations for obtaining a straightforward equation for computing the gradients of the PDIS and the WIS estimators with respect to the policy parameters. These might also be of independent interest when dealing with off-policy policy optimization in purely stationary MDPs.

Property 1 (PDIS Gradient)**.** *Let* $\rho_i(0,l) := \prod_{j=0}^{l}\frac{\pi^\theta(A^j_i|S^j_i)}{\beta_i(A^j_i|S^j_i)}$, *then*

$$\nabla\hat{J}_i(\theta) = \sum_{t=0}^{T}\frac{\partial\log\pi^\theta(A^t_i|S^t_i)}{\partial\theta}\left(\sum_{l=t}^{T}\rho_i(0,l)\cdot\gamma^l\cdot R^l_i\right).$$

$_t \backslash^l$	0	1	2	...	T
0	$\gamma^0 \rho_i(0,0)\Psi_i^0 R_i^0$				
1	$\gamma^1 \rho_i(0,1)\Psi_i^0 R_i^1$	$\gamma^1 \rho_i(0,1)\Psi_i^1 R_i^1$			
2	$\gamma^2 \rho_i(0,2)\Psi_i^0 R_i^2$	$\gamma^2 \rho_i(0,2)\Psi_i^1 R_i^2$	$\gamma^2 \rho_i(0,2)\Psi_i^2 R_i^2$		
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
T	$\gamma^T \rho_i(0,T)\Psi_i^0 R_i^T$	$\gamma^T \rho_i(0,T)\Psi_i^1 R_i^T$	$\gamma^T \rho_i(0,T)\Psi_i^2 R_i^T$	...	$\gamma^T \rho_i(0,T)\Psi_i^T R_i^T$

Table 1. let $\Psi_i^t = \partial \log \pi^\theta(A_i^t|S_i^t)/\partial\theta$. This table represents all the terms in (18) required for computing $\nabla \hat{J}_i(\theta)$. Gray color denotes empty cells.

Proof. Recall from (1),

$$\hat{J}_i(\theta) = \sum_{t=0}^{T} \left(\prod_{l=0}^{t} \frac{\pi^\theta(A_i^l|S_i^l)}{\beta(A_i^l|S_i^l)} \right) \gamma^t R_i^t.$$

Computing the gradient of $\hat{J}_i(\theta)$,

$$\begin{aligned}
\nabla \hat{J}_i(\theta) &= \sum_{t=0}^{T} \frac{\partial}{\partial\theta} \left(\prod_{l=0}^{t} \frac{\pi^\theta(A_i^l|S_i^l)}{\beta(A_i^l|S_i^l)} \right) \gamma^t R_i^t \\
&= \sum_{t=0}^{T} \left(\prod_{l=0}^{t} \frac{\pi^\theta(A_i^l|S_i^l)}{\beta(A_i^l|S_i^l)} \right) \frac{\partial \log \left(\prod_{l=0}^{t} \pi^\theta(A_i^l|S_i^l) \right)}{\partial\theta} \gamma^t R_i^t \\
&= \sum_{t=0}^{T} \left(\prod_{l=0}^{t} \frac{\pi^\theta(A_i^l|S_i^l)}{\beta(A_i^l|S_i^l)} \right) \left(\sum_{l=0}^{t} \frac{\partial \log \pi^\theta(A_i^l|S_i^l)}{\partial\theta} \right) \gamma^t R_i^t \\
&= \sum_{t=0}^{T} \rho_i(0,t) \left(\sum_{l=0}^{t} \frac{\partial \log \pi^\theta(A_i^l|S_i^l)}{\partial\theta} \right) \gamma^t R_i^t. \\
&= \sum_{t=0}^{T} \frac{\partial \log \pi^\theta(A_i^t|S_i^t)}{\partial\theta} \left(\sum_{l=t}^{T} \rho_i(0,l) \cdot \gamma^l \cdot R_i^l \right), \qquad (18)
\end{aligned}$$

where, in the last step, instead of the summation over partial derivatives of $\log \pi^\theta$ for each weight $\rho(\cdot,\cdot)$, we consider the alternate form where summation is over the importance weights $\rho(\cdot,\cdot)$ for each partial derivative of $\log \pi^\theta$. To see this step clearly, let $\Psi_i^t = \partial \log \pi^\theta(A_i^t|S_i^t)/\partial\theta$, then Table 1 shows all the terms in (18). The last step in simplification corresponds to taking column-wise sum instead of row wise sum in Table 1.

□

C. Detailed Literature Review

The problem of non-stationarity has a long history. In the operations research community, many dynamic sequential decision-making problems are modeled using infinite horizon *non-homogeneous* MDPs (Hopp et al., 1987). While estimating an optimal policy is infeasible under an infinite horizon setting when the dynamics are changing and a stationary distribution cannot be reached, researchers have been interested in identifying sufficient forecast horizons for performing near-optimal planning (Garcia & Smith, 2000; Cheevaprawatdomrong et al., 2007; Ghate & Smith, 2013) or robust policy iteration (Sinha & Ghate, 2016).

In contrast, non-stationary multi-armed bandits (NMAB) capture the setting where the horizon length is one, but the reward distribution changes over time (Moulines, 2008; Besbes et al., 2014). Many variants of NMAB, like *cascading non-stationary bandits* (Wang et al., 2019b; Li & de Rijke, 2019) and *rotten bandits* (Levine et al., 2017; Seznec et al., 2018) have also been considered. In optimistic online convex optimization, researchers have shown that better performance can be

achieved by updating the parameters using predictions of the gradient of the future loss, based on past gradients (Rakhlin & Sridharan, 2013; Yang & Mohri, 2016; Mohri & Yang, 2016; Wang et al., 2019a).

Non-stationarity also occurs in multi-player games, like rock-paper-scissors, where each episode is a single one-step interaction (Singh et al., 2000; Bowling, 2005; Conitzer & Sandholm, 2007). Opponent modeling in games has been shown to be useful and regret bounds for multi-player games where players can be replaced with some probability p, i.e., the game changes slowly over time, has also been established (Zhang & Lesser, 2010; Mealing & Shapiro, 2013; Foster et al., 2016; Foerster et al., 2018). However, learning sequential strategies in a non-stationary setting is still an open research problem.

For episodic non-stationary MDPs, researchers have also looked at providing regret bounds for algorithms that exploit oracle access to the current reward and transition functions (Even-Dar et al., 2005; Yu & Mannor, 2009; Abbasi et al., 2013; Lecarpentier & Rachelson, 2019; Li et al., 2019). Alleviating oracle access by performing a count-based estimate of the reward and transition function based on the recent history of interactions has also been proposed (Gajane et al., 2018; Cheung et al., 2019). For tabular MDPs, past data from a non-stationary MDP can be used to construct a maximum-likelihood estimate model (Ornik & Topcu, 2019) or a full Bayesian model (Jong & Stone, 2005) of the transition dynamics. Our focus is on the setting which is not restricted to tabular representations.

A Hidden-Mode MDP is an alternate setting that assumes that the environment changes are confined to a small number of hidden modes, where each mode represents a unique MDP. This provides a more tractable way to model a limited number of MDPs (Choi et al., 2000; Basso & Engel, 2009; Mahmud & Ramamoorthy, 2013), or perform model-free updates using mode-change detection (Padakandla et al., 2019). In this work, we are interested in the continuously changing setting, where the number of possible MDPs is unbounded.

The problem of adapting to non-stationarity is also related to continual learning (Ring, 1994), lifelong-learning (Thrun, 1998), and meta-learning (Schmidhuber, 1999). Several meta-learning based approaches for fine-tuning a (mixture of) trained model(s) using samples observed during a similar task at test time have been proposed (Nagabandi et al., 2018a;b). Other works have also shown how models of the environment can be used for continual learning (Lu et al., 2019) or using it along with a model predictive control (Wagener et al., 2019). We focus on the model-free paradigm and our approach is complementary to these model-based methods.

Tracking has also been shown to play an important role in non-stationary domains. Thomas et al. (2017) and Jagerman et al. (2019) have proposed policy evaluation techniques in a non-stationary setting by tracking a policy's past performances. However, they do not provide any procedure for searching for a good future policy. To adapt quickly in non-stationary tasks, TIDBD (Kearney et al., 2018) and AdaGain (Jacobsen et al., 2019) perform TD-learning while also automatically (de-)emphasizing updates to (ir)relevant features by modulating the learning rate of the parameters associated with the respective features. Similarly, Abdallah & Kaisers (2016) propose repeating a Q-value update inversely proportional to the probability with which an action was chosen to obtain a transition tuple. In this work, we aim at going beyond tracking and pro-actively optimizing for the future.

D. Implementation details

D.1. Environments

We provide empirical results on three non-stationary environments: diabetes treatment, recommender system, and a goal-reacher task. Details for each of these environments are provided in this section.

Non-stationary Diabetes Treatment: This MDP models the problem of Type-1 diabetes management. At a high-level, the body of a person with Type-1 diabetes does not produce enough *insulin*, a hormone that promotes absorption of glucose from the blood. Consumption of a meal increases the blood-glucose level in the body and if the blood-glucose level becomes too high, then the patient can suffer from *Hyperglycemia*. Insulin injections can be used externally to reduce the level but if the level becomes too low, then the patient suffers from *Hypoglycemia*. While either of the extremes is undesirable, Hypoglycemia is more dangerous and can triple the five-year mortality rate for a person with diabetes (Man et al., 2014).

Autonomous medical support system have been proposed to decide how much insulin should be injected to keep a person's blood glucose levels near ideal levels (Bastani, 2014). The parameters of such a medical support system are set by a doctor specifically for each patient. However, due to non-stationarities induced over time as a consequence of changes in the body mass index, the insulin sensitivity of the pancreas, diet, etc., the parameters of the controller need to be re-adjusted regularly. Currently, this requires re-visiting the doctor. A viable reinforcement learning solution to this non-stationary problem could

enable the automatic tuning of these parameters for patients without regular access to a physician.

To model this MDP we use an open-source implementation (Xie, 2019) of the U.S. Food and Drug Administration (FDA) approved Type-1 Diabetes Mellitus simulator (T1DMS) (Man et al., 2014) for treatment of Type-1 diabetes, where we induce non-stationarity by oscillating the body parameters between two known configurations. Each episode consists of a day (1440 minutes) in an *in-silico* patient's life and the transition dynamics of a patient's body for each second is governed by a continuous time ordinary differential equation (ODE) (Man et al., 2014). After each second the insulin controller is used to inject the desired amount of insulin for controlling the blood glucose.

For controlling the insulin injection, we use a parameterized policy based on the amount of insulin that a person with diabetes is instructed to inject prior to eating a meal (Bastani, 2014):

$$\text{injection} = \frac{\text{current blood glucose} - \text{target blood glucose}}{CF} + \frac{\text{meal size}}{CR},$$

where 'current blood glucose' is the estimate of the person's current blood glucose level, 'target blood glucose' is the desired blood glucose, 'meal size' is the estimate of the size of the meal the patient is about to eat, and CR and CF are two real-valued parameters, that must be tuned based on the body parameters to make the treatment effective.

Non-stationary Recommender System: During online recommendation of movies, tutorials, advertisements and other products, the system needs to interact and personalize for each user. However, the interests of an user for different items among the products that can be recommended fluctuate over time. For example, interests during online shopping can vary based on seasonality or other unknown factors.

This environment models the desired recommender system setting where reward (interest of the user) associated with each item changes over time. Figure 6 (left) plots how the reward associated with each item changes over time, for each of the considered 'speeds' of non-stationarity. The goal for the reinforcement learning agent is to maximize revenue by recommending the item which the user is most interested at any time.

Non-stationary Goal Reacher: For an autonomous robot dealing with tasks in the open-world it is natural for the problem specification to change over time. An ideal system should quickly adapt to the changes and still complete the task.

To model the above setting, this environment considers a task of reaching a non-stationary goal position. That is, the location of the goal position keeps slowly moving around with time. The goal of the reinforcement learning agent is to control the four (left, right, up, and down) actions to move the agent towards the goal as quickly as possible given the real valued Cartesian coordinates of the agent's current location. Maximum time given to the agent to reach the goal is 15 steps.

D.2. Hyper-parameters

For both the variants of the proposed *Prognosticator* algorithms, we use Fourier basis to encode the time index while performing (ordinary/weighted) least squares estimation. Since Fourier basis requires inputs to be normalized with $|x| \leq 1$, we normalize each time index by dividing it by $K + \delta$, where K is the current time and δ is the maximum time in future we need the forecast for. Further, as we are regressing only on time (which are all positive values), is does not matter whether the function of policy performance over time is odd ($\Psi(x) = -\Psi(-x)$) or not. Therefore, we drop all the terms in the basis corresponding to $\sin(\cdot)$ which are useful for modeling odd functions. This reduces the number of parameters to be estimated by half. Finally, instead of letting $n \in \mathbb{N}$, we restrict it to a finite set $\{1, ..., d-1\}$, where d is a fixed constant that determines the size of feature vector for each input. In all our experiments, d was a hyper-parameter chosen from $\{3, 5, 7\}$.

Other hyper-parameter ranges were common for all the algorithms. The discounting factor γ was kept fixed to 0.99 and learning rate η was chosen from range $[5e-5, 5e-2]$. Entropy regularizer λ was chosen from range $[0, 1e-2]$ Batch size δ was chosen from the set $\{1, 3, 5\}$. Inner optimization over past data for the proposed methods and FTRL-PG was run for $\{10, 20, 30\} \times \delta$ iterations. Inner optimization for ONPG corresponds to one iteration over all the trajectories collected in the current batch. Past algorithms have shown that clipping the importance weights can improve stability of reinforcement learning algorithms (Schulman et al., 2017). Similarly, we clip the maximum value of the importance ratio to a value chosen from $\{5, 10, 15\}$. Additionally, value functions learned online is popularly used as a baseline/control-variate in policy gradient based algorithms. Whether to use of such a value function or not was decided from {True, False}. As the non-stationary diabetes treatment problem has a continuous action space, the policy was parameterized with a Gaussian distribution having a variance chosen from $[0.5, 2.5]$. For the non-stationary goal-reacher environment, the policy was parameterized using a two-layer neural network with number of hidden nodes chosen from $\{16, 32, 64\}$.

In total, 2000 settings for each algorithm, for each domain, were uniformly sampled (loguniformly for learning rates and λ) from the mentioned hyper-parameter ranges/sets. Results from the best performing setting is reported in all the plots. Each hyper-parameter setting was ran using 10 seeds for the non-stationary diabetes treatment (as it was time intensive to run a continuous time ODE for each step) and 30 seeds for the other two environments to get the standard error of the mean performances. The authors had shared access to a computing cluster, consisting of 50 compute nodes with 28 cores each, which was used to run all the experiments.

E. Detailed Empirical Results

Complexity analysis (space, time, and sample size) Space requirement for our algorithms and FTRL is linear in the number of episodes seen in the past, whereas it is constant for ONPG as it discards all past data. Computational cost of our algorithm is also similar to FTRL as the only additional cost is that of differentiating through least-squares estimators which involves computing $(\Phi^\top \Phi)^{-1}$ or $(\Phi^\top \Lambda \Phi)^{-1}$. This additional overhead is negligible as these matrices are of size $d \times d$, where d is the size of feature vector for time index and $d << N$, where N is the number of past episodes. Figures 5, 6, and 7 present an empirical estimate of sample efficiency.

Performance over time: In Figure 5, summary statistics of the results were presented. In this section we present all the results in details. Figure 6 plots the performances of all the algorithms for individual episodes as the user interests changes over time in the recommender system environment. In this environment as the true rewards for each of the item is directly available we provide a visual plot for it as well in Figure 6 (left). Notice the shape of the performance curve for the proposed methods, which closely captures the trend of maximum reward attainable over time.

Figure 7 plots the performances of all the algorithms for non-stationary goal-reacher and diabetes treatment environments. In these environments, the maximum achievable performance for each episode is not readily available.

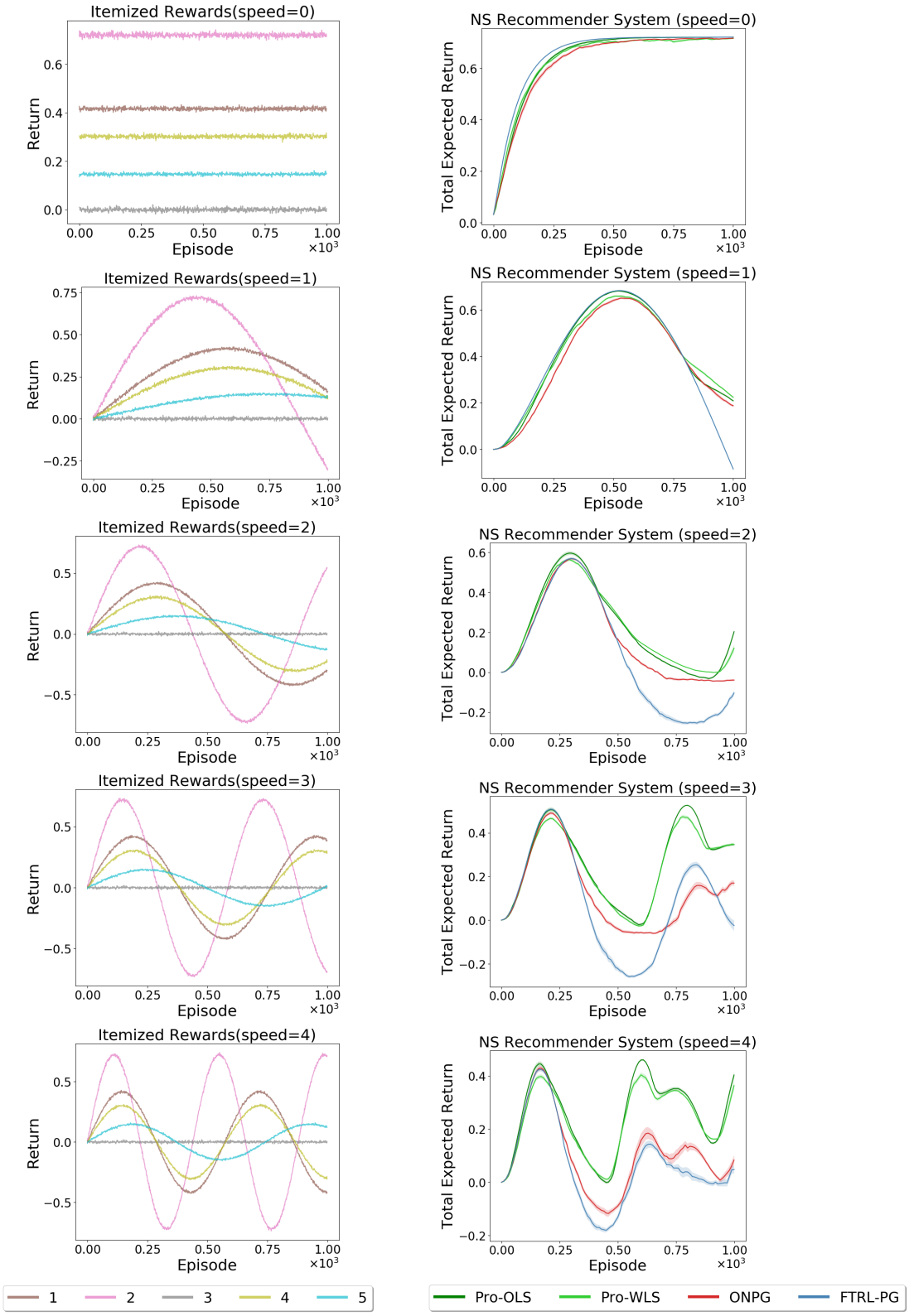

Figure 6. (Left) Fluctuations in the reward associated with each of the 5 items that can be recommended, for different speeds. (Right) Running mean of the best performance of all the algorithms for different speeds; higher total expected return is better. Shaded regions correspond to the standard error of the mean obtained using 30 trials. Notice the shape of the performance curve for the proposed methods, which closely captures the trend of maximum reward attainable over time.

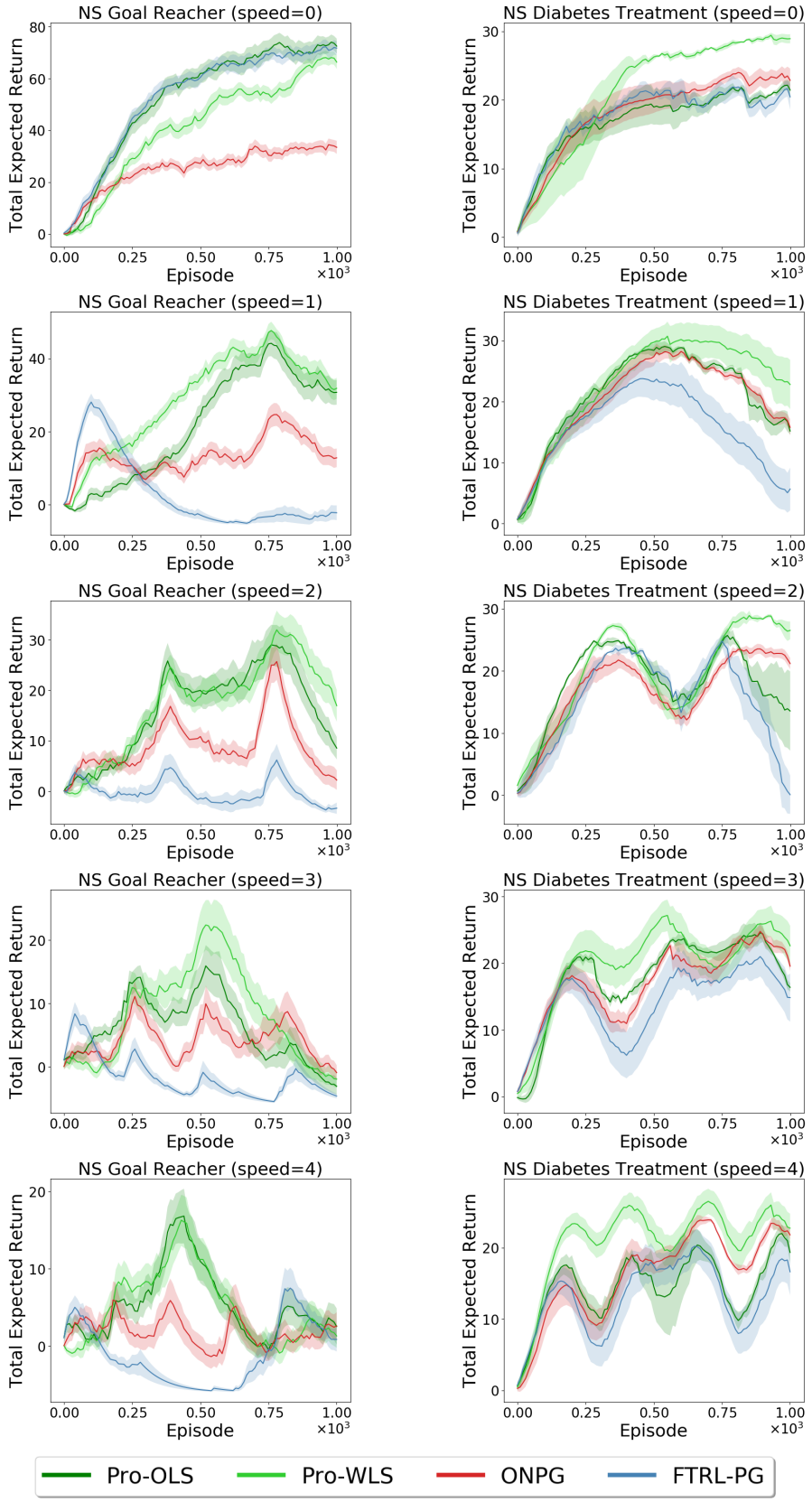

Figure 7. Running mean of the best performance of all the algorithms for different speeds; higher total expected return is better. Shaded regions correspond to the standard error of the mean obtained using 30 trials for NS Goal Reacher and 10 trials for NS Diabetes Treatment.