Proceedings of Machine Learning Research vol 144:1–12, 2021

Fast Stochastic Kalman Gradient Descent for Reinforcement Learning

Simone Totaro SIMONE.TOTARO@GMAIL.COM
Dept. Information and Communication Technologies, Universitat Pompeu Fabra, Barcelona, Spain

Anders Jonsson ANDERS.JONSSON@UPF.EDU
Dept. Information and Communication Technologies, Universitat Pompeu Fabra, Barcelona, Spain

Abstract

As we move towards real world applications, there is an increasing need for scalable, online optimization algorithms capable of dealing with the non-stationarity of the real world. We revisit the problem of online policy evaluation in non-stationary deterministic MDPs through the lense of Kalman filtering. We introduce a randomized regularization technique called Stochastic Kalman Gradient Descent (SKGD) that, combined with a low rank update, generates a sequence of feasible iterates. SKGD is suitable for large scale optimization of non-linear function approximators. We evaluate the performance of SKGD in two controlled experiments, and in one real world application of microgrid control. In our experiments, SKGD is more robust to drift in the transition dynamics than state-of-the-art reinforcement learning algorithms, and the resulting policies are smoother.

Keywords: Non-stationary MDPs, Reinforcement Learning, Tracking

1. Introduction

We consider the problem of continual, non-stationary reinforcement learning, also known as tracking. In our setting, a learning agent interacts with a changing environment, and the agent needs to adapt to changes in the environment without making any explicity assumption about the model dynamics. We are concerned with the task of tracking the value functions arising from the drift in the transition dynamics. We take a statistical approach, where we generalize the state-space model introduced by Geist and Pietquin (2010) to high dimensional settings via stochastic regularization, similar in spirit to the dropout technique. We call this algorithm Stochastic Kalman Gradient Descent. It is known that the time and space complexity of Kalman methods is $O(d^2)$, where d is the number of dimensions of the parameter space. We present a low rank approximation, which directly updates a low-rank approximation of the initial covariance matrix. We show that our method can deal with non-stationarity both in the case of simulated drift and in a real world task that aims to control an electrical microgrid. For reproducibility, the code of the algorithm is publicly available[1].

1.1. Related Work

Our work can be placed at the intersection of reinforcement learning, control, and optimization. In reinforcement learning, the term non-stationarity refers to the time dependence of the transition dynamics and/or reward function (Puterman, 2014). If exploration is not possible in the time dimension, then the set of stationary policies may not include the optimal policy. In such settings,

1. https://github.com/d3sm0/skgd.

further assumptions are usually needed to explicitly characterize the drift in the transition dynamics (Lecarpentier and Rachelson, 2019), or an underlying discrete time process that decomposes the non-stationary task into subtasks, possibly with stationary transition dynamics (Choi et al., 2000). Assumptions are exploited to learn one or several models of the world (Doya et al., 2002), switch between them (Wiering, 2001) or condition a global optimal policy (Raileanu et al., 2020).

Our work differs from those described above because we do not characterize the shift in distribution directly, only indirectly through the sequence of prediction errors observed throughout the learning process. As such, our method is model-free, but crucially it is limited to deterministic changes in the transition dynamics. Our work builds upon Kalman Temporal Difference (Geist and Pietquin, 2010), which in the case of deterministic dynamics and identity covariance noise is equivalent to Recursive Least Squares Temporal Difference, or RLSTD (Bradke and Barto, 1996). However, these algorithms were designed for linear function approximation, while we are interested in nonlinear function approximation with large parameter spaces.

Several existing works perform low-rank updates of a positive semi-definite matrix (Nocedal and Wright, 2006; Seeger, 2004; Spantini et al., 2015) or apply gradient descent for stochastic optimization (Vuckovic, 2018; Mahsereci, 2018; Chen et al., 2020). Our work differs in that we do not filter the gradient dynamics of the loss function, but rather the prediction, for which the Bellman residual is the innovation. There exists a low-rank version of RLSTD that updates the subspace in each iteration (Gehring et al., 2015); in contrast, we maintain a fixed subspace and only update a low-rank matrix, which is possible since actions are deterministic. Most similar to our work is that of Shashua and Mannor (2020) which, unlike our algorithm, has an update rule with time complexity $O(d^2)$, and does not address the known issues of Extended Kalman Filtering (Huang et al., 2008).

1.2. Contribution

We propose a fast optimizer called Stochastic Kalman Gradient Descent (SKGD) for learning value functions in deterministic, non-stationary, online Markov decision processes (MDPs). SKGD extends Kalman Temporal Difference by improving the computational complexity and guaranteeing the feasibility of the iterates. Both problems are addressed by designing an update rule that produces sparse, symmetric, positive semi-definite and rank-preserving covariance matrices. We introduce sparsity via a dropout regularization over the rows of the covariance matrix, and show that it is equivalent to regularizating the norm of the matrix. Then we use the eigendecomposition to design an update rule that generates a sequence of low rank matrices, which are positive semi-definite and rank-preserving. Our method is suitable for large scale optimization of online non-stationary regression tasks. In the present work we show its performance when used in actor-critic architectures for online reinforcement learning. We present two controlled experiments that illustrate its behaviour, and one real world application in the form of microgrid control (Totaro et al., 2020).

2. Background

In this section we describe the background for our work and introduce notation used throughout.

2.1. Tracking in Dynamical Systems

We consider discrete-time dynamical systems described by a hidden variable X and an observable variable Y, both of which can be multivariate with dimensions d and n, respectively. We assume

that n is small, i.e. $n \ll d$. The values of Y are given by a non-stationary *observation function* $f : \mathbb{R}^d \to \mathbb{R}^n$, which is either linear or nonlinear. The values of X and Y at time step t are given by

$$x_t = x_{t-1} + \xi_t,$$
$$y_t = f_t(x_t) + \psi_t,$$

where ξ_t and ψ_t are additive, white and independent noise components. Tracking is the problem of computing an estimate $\hat{x}_t$ of the true value x_t given a sequence of observations $y_1, \ldots, y_t$.

2.2. Linear Observation Function

We first assume that the observation function is linear, i.e. $f_t(x_t) = H_t x_t$, where $H_t \in \mathbb{R}^{n \times d}$ is the observation matrix at time t. In this case, we can use recursive least squares estimation (Plackett, 1950) or Kalman filtering (Kalman, 1960) to track the value of X. The former assumes a constant value of X, i.e. $\xi_t = 0$ for each t; we present the derivation for the more general case $\xi_t \neq 0$.

A linear recursive estimator is given by

$$\hat{x}_t = (I - K_t H_t)\hat{x}_{t-1} + K_t y_t,$$

where $K_t \in \mathbb{R}^{d \times n}$ is a gain matrix. The estimation error $\epsilon_t = x_t - \hat{x}_t$ can be recursively defined as

$$\epsilon_t = (I - K_t H_t)(\epsilon_{t-1} + \xi_t) - K_t \psi_t.$$

The objective is to minimize the expected square norm of the estimation error, $L_t = \mathbb{E}[\|\epsilon_t\|^2] = \mathbb{E}[\epsilon_t^\top \epsilon_t] = \mathbb{E}[\mathrm{Tr}(\epsilon_t \epsilon_t^\top)] = \mathrm{Tr}\, P_t$. Here, $P_t = \mathbb{E}[\epsilon_t \epsilon_t^\top] \in \mathbb{R}^{d \times d}$ is the estimation error covariance, recursively defined as

$$P_t = (I - K_t H_t)(P_{t-1} + \Xi_t)(I - K_t H_t)^\top + K_t \Psi_t K_t^\top,$$

where Ξ_t and Ψ_t are the covariance matrices of the noise components ξ_t and ψ_t, respectively. We make the common assumption that the noise terms are i.i.d., implying that Ξ_t and Ψ_t are diagonal.

Setting the gradient of L_t to 0 yields the following expression for the gain matrix:

$$\tilde{P}_t = P_{t-1} + \Xi_t, \tag{1}$$
$$K_t = \tilde{P}_t H_t^\top (H_t \tilde{P}_t H_t^\top + \Psi_t)^{-1}. \tag{2}$$

In turn, this simplifies the expression for P_t to yield the following recursive update rules:

$$\hat{x}_t = (I - K_t H_t)\hat{x}_{t-1} + K_t y_t, \tag{3}$$
$$P_t = (I - K_t H_t)\tilde{P}_t. \tag{4}$$

At time t, Kalman filtering computes the gain matrix K_t (2), and then recursively updates the estimate $\hat{x}_t$ (3) and estimation error covariance P_t (4). We remark that the update rules for recursive least squares estimation are obtained by setting Ξ_t to 0, i.e. (1) simply becomes $\tilde{P}_t = P_{t-1}$.

There exist other algorithms that perform similar recursive updates of the estimate and estimation error covariance. One such algorithm is Least Squares Temporal Difference, or LSTD (Bradke and Barto, 1996), in which the objective function is given by

$$L_t(\theta) = \frac{1}{t} \sum_{i=1}^{t} \left(y_i - f(\theta, x_i)\right)^2,$$

where variables X and Y are both observed. Hence the aim is to find a parameter vector θ that minimizes $L_t(\theta)$. A related objective function is given by

$$L_t(\theta) = \mathbb{E}\left[(y_t - f(\theta, x_t))^2 \mid (x_1, y_1, \ldots, x_{t-1}, y_{t-1})\right],$$

where the objective at time t is conditional on the history of all previous observations. Our algorithm is fundamentally concerned with efficiently computing the gain matrix and recursive updates, and hence it readily applies to these other objective functions as well.

2.3. Nonlinear Observation Function

When the observation function f is nonlinear, the tracking problem becomes significantly harder. In this case, the standard approach of Extended Kalman Filtering (Anderson and Moore, 1979) is to linearize the observation function around the current estimate. At time step t, let $F_t = \nabla_x f_t(\hat{x}_{t-1}) \in \mathbb{R}^{n \times d}$ be the Jacobian matrix containing the first-order partial derivatives of the observation function evaluated in $\hat{x}_{t-1}$. The idea of linearization is to replace the observation matrix H_t with the Jacobian F_t in the update rules for the gain matrix K_t and estimation error covariance P_t:

$$K_t = \tilde{P}_t F_t^\top (F_t \tilde{P}_t F_t^\top + \Psi_t)^{-1}, \tag{5}$$

$$\hat{x}_t = \hat{x}_{t-1} + K_t(y_t - f_t(\hat{x}_{t-1})), \tag{6}$$

$$P_t = (I - K_t F_t)\tilde{P}_t. \tag{7}$$

Note that the estimate $\hat{x}_t$ still depends directly on the actual observation function f.

It is well known that linearization propagates uncertainty only around the mean estimate (Huang et al., 2008). In addition, the update of P_t may not lie in the feasible set, i.e. P_t may not be symmetric and positive semidefinite (PSD). Moreover, the matrix multiplications in the update rules for K_t and P_t have time complexity $O(d^2 n)$, which is prohibitive when the number of dimensions d is large.

2.4. Reinforcement Learning

In reinforcement learning (Sutton and Barto, 2018), the environment is modelled as a Markov decision process (MDP), i.e. a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, p, r \rangle$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ a finite set of actions, $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ a transition kernel and $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ a reward function. Here, $\Delta(\mathcal{X})$ denotes the probability simplex of any set $\mathcal{X}$, i.e. the set of all probability distributions over $\mathcal{X}$.

At each time step t, the learning agent observes a state $s_t \in \mathcal{S}$, selects an action $a_t \in \mathcal{A}$, transitions to a new state $s_{t+1} \sim p(\cdot|s_t, a_t)$ and obtains a reward r_t with expected value $\mathbb{E}[r_t] = r(s_t, a_t)$. The aim is to compute a policy, i.e. a mapping $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ from states to probability distributions over actions. The value of policy π in state s is given by

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r(s_t, a_t) \,\middle|\, s_1 = s\right],$$

i.e. the expected sum of future rewards when starting in state s and using policy π to select actions. Here, $\gamma \in (0, 1]$ is a discount factor that determines the relative importance of distant rewards. The value function V^π satisfies the recursive Bellman equation

$$V^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left[r(s, a) + \gamma \sum_{s'} p(s'|s, a) V^\pi(s')\right].$$

An *optimal policy* $\pi^* = \arg\max_\pi V^\pi$ maximizes the value function in each state.

In most applications of reinforcement learning, the state space is too large to explicitly represent the value of each state. Instead, it is assumed that each state $s \in \mathcal{S}$ corresponds to a feature vector $\phi(s)$, and the value function is approximated as $\hat{V}_\theta(s) = g(\theta, \phi(s))$, where θ is a parameter vector. In this case, the aim of reinforcement learning is to compute the parameter vector θ^* that corresponds to the value function of the optimal policy π^*, or the closest approximation thereof.

Estimating θ^* can be modelled as a tracking problem in a dynamical system where a hidden variable Θ models the parameter vector and an observed variable R models the obtained reward:

$$\theta_t = \theta_{t-1} + \xi_t, \tag{8}$$
$$r_t = f_t(\theta_t) + \psi_t, \tag{9}$$

Note that r_t is a scalar, i.e. $n = 1$. The observation function f is given by

$$f_t(\theta_t) = \hat{V}_{\theta_t}(s_t) - \gamma \hat{V}_{\theta_t}(s_{t+1}) = g(\theta_t, \phi(s_t)) - \gamma g(\theta_t, \phi(s_{t+1})),$$

where s_t and s_{t+1} are the states at times t and $t+1$, respectively, i.e. the observed reward $r_t = f_t(\theta_t) + \psi_t$ is the discounted difference between the values in s_t and s_{t+1}. The noise component ψ_t models variations in reward around the mean $r(s_t, a_t)$. For deterministic MDPs with transition kernel $p : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, the noise ψ_t is white and independent. However, this assumption does *not* hold in the stochastic case. In this paper, as in previous work, we focus on the deterministic case.

For stationary MDPs, the optimal parameter vector θ^* is constant, i.e. $\xi_t = 0$ for each t. By setting $\xi_t \neq 0$ we can estimate the value of Θ for *non-stationary* MDPs. This is precisely the idea behind Kalman Temporal Difference, or KTD (Geist and Pietquin, 2010). In a non-stationary MDP, the transition kernel $p = \{p_t\}_{t \geq 1}$ and reward function $r = \{r_t\}_{t \geq 1}$ can have different definitions at each time step t, and θ_t is defined as the optimal parameter vector for the MDP $\mathcal{M}_t = \langle \mathcal{S}, \mathcal{A}, p_t, r_t \rangle$.

In the special case for which the value function approximation $\hat{V}_\theta$ is linear, i.e. $g(\theta, \phi(s)) = \phi(s)^\top \theta$, the observation function f is also linear:

$$f_t(\theta_t) = \phi(s_t)^\top \theta_t - \gamma \phi(s_{t+1})^\top \theta_t = (\phi(s_t) - \gamma \phi(s_{t+1}))^\top \theta_t.$$

In this case, estimating the value of Θ reduces to Kalman filtering. However, most successful applications of reinforcement learning use nonlinear function approximation in the form of neural networks. Even though we can linearize the observation function, the update rules in (5)-(7) become inefficient when the parameter vector θ is high-dimensional.

Most algorithms for non-stationary MDPs explicitly model the values of latent variables that determine the distribution shift of the transition kernel p_t and/or reward function r_t at each time step t. In contrast, KTD does not need to explicitly model latent variables, and instead performs tracking directly on the unknown parameter vector θ. This results in an algorithm with fewer parameters and faster updates, and no prior knowledge about latent variables is needed. The aim of the present work is to develop a practical algorithm that brings those qualities to the general nonlinear case.

3. Stochastic Kalman Gradient Descent

The strength of KTD is that it is purely online and derived from the purpose of tracking. It does not make any assumptions about the structure of the MDP, which is common in applications of RL to the problem of control. KTD has three key shortcomings: 1) it assumes a linear parameterization

of the value function; 2) the time and space complexity is quadratic in the number of dimensions d; and 3) it does not handle stochastic MDPs.

In this section we present a novel algorithm that we call Stochastic Kalman Gradient Descent, or SKGD, that addresses the first two shortcomings of KTD. The third shortcoming is a pitfall of all projected Bellman error methods (Baird, 1995), which we aim to address in future work. For a fixed policy π, and deterministic MDP, the policy evaluation problem is equivalent to performing nonlinear regression on the observation produced by the dynamical system in (8)-(9). We derive the nonlinear extension of KTD via this more general framework.

3.1. Regularization via Row Dropout

The linearization of the predictor around the mean estimate discards all higher order terms of the true estimation error covariance P, concentrating the uncertainty in a few highly correlated parameters. In low dimensions, one can regularize the eigenvalues directly with a matrix norm like the Frobenius norm $\|\cdot\|_F$, but in high dimensions this is prohibitively expensive. We propose to apply a stochastic mask, constructed by sampling a vector of indices $\mathbf{m} \sim Bern(\sigma)$ with $\mathbf{m} \in \{0,1\}^d$ and forming the corresponding diagonal matrix $M = \text{diag}(\mathbf{m})$. This dropout technique allows us to control the norm of P and (indirectly) the rank via the parameter σ of the Bernoulli distribution. The error covariance matrix P is effectively a per-parameter learning rate, and setting some eigenvalues to 0 has the effect of resetting the learning rate to 0 after some iterations.

In the following lemma we show that $\|MP\|_F / \sigma$ is an unbiased estimator of $\|P\|_F$.

Lemma 1 *For $\mathbf{m} \sim Bern(\sigma)$, $M = \text{diag}(\mathbf{m})$ and any matrix P, it holds that*

$$\frac{1}{\sigma}\mathbb{E}\{\|MP\|_F\} = \|P\|_F .$$

Proof Follows from the definitions of M and $\|\cdot\|_F$ and the linearity of the expectation:

$$\frac{1}{\sigma}\mathbb{E}\{\|MP\|_F\} = \frac{1}{\sigma}\mathbb{E}\{\sum_i \sum_j (M_{ii}P_{ij})^2\} = \frac{1}{\sigma}\sum_i \mathbb{E}\{m_i^2\} \sum_j P_{ij}^2 = \frac{\sigma}{\sigma}\sum_i \sum_j P_{ij}^2 = \|P\|_F .$$

■

The expected time complexity of $\|MP\|_F$ is $O(\sigma d^2)$, which is smaller than $O(d^2)$ when $\sigma \ll 1$.

We use dropout in the covariance update, multiplying the previous covariance matrix by M:

$$\tilde{P}_t = MP_{t-1} + \Xi_t. \tag{10}$$

Since Ξ_t is a diagonal matrix by assumption, the time complexity of the matrix multiplications in (5) and (7) becomes $O(\sigma d^2 n)$ with the new definition of $\tilde{P}_t$.

3.2. Feasibility and Complexity

Even if we use the more efficient update rules derived from (10), the sequence of covariance matrices may not be feasible, i.e. P_t may not be symmetric and PSD for $t \geq 1$. We would like to have an update rule that is always feasible and depends on the rank of the true covariance P.

We borrow ideas from low-rank Kalman filtering (Bonnabel and Sepulchre, 2012) to derive a low-rank update rule for P. A low-rank approximation of P is given by $P = Q\Lambda Q^\top$, where

Input: horizon T, trajectory $\{(s_t, r_t, s_{t+1})\}_{t\in[T]}$
$[\Lambda_0, U] \leftarrow \text{eigen}(\alpha I_d, r)$
$Q \leftarrow \text{QR}(U - (UU^\top U))$
for $t = 0 \dots T$ **do**
  $\tilde{\Lambda}_t \leftarrow M\Lambda_{t-1} + \epsilon I_r, \quad M \sim Bern(\sigma)$
  $\tilde{P}_t \leftarrow Q\tilde{\Lambda}_t Q^\top$
  $F_t \leftarrow \nabla_\theta f_t(\hat{\theta}_{t-1})$
  $A^{-1} \leftarrow (F_t\tilde{P}_t F_t^\top + \varepsilon I_n)^{-1}$
  $\hat{\theta}_t \leftarrow \hat{\theta}_{t-1} + \tilde{P}_t F_t^\top A^{-1}(r_t + \gamma V_{\hat{\theta}_{t-1}}(s_{t+1}) - V_{\hat{\theta}_{t-1}}(s_t))$
  $\Lambda_t \leftarrow \tilde{\Lambda}_t - \tilde{\Lambda}_t Q^\top F_t A^{-1} F_t^\top Q\tilde{\Lambda}_t$
end for

Figure 1: Pseudo-code of the SKGD algorithm.

$\Lambda \in \mathbb{R}^{r\times r}$ models the error covariance in a lower dimensional subspace such that $r \leq d$, and $Q \in \mathbb{R}^{d\times r}$ is a unitary matrix (satisfying $Q^\top = Q^{-1}$) defining this subspace. In fact, an exact such decomposition exists for symmetric matrices whose rank is r: in this case, $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_r)$ is the diagonal matrix containing the r distinct eigenvalues of P, and Q has the associated r eigenvectors as columns. However, we define the low-rank approximation for arbitrary r, and Λ does not have to be a diagonal matrix. As long as Λ is symmetric and PSD, then so is P. An advantage of this approach is that it does not need to perform singular value decomposition at each time step.

We first note that the update rule in (7) can be rewritten as

$$P_t = (I - K_t F_t)\tilde{P}_t = \tilde{P}_t - \tilde{P}_t F_t^\top A^{-1} F_t \tilde{P}_t,$$

where $A = F_t\tilde{P}_t F_t^\top + \Psi_t$. We temporarily assume that $\Xi_t = 0$, implying $\tilde{P}_t = P_{t-1}$. If the matrix Q is constant, we can substitute the expression $P_t = Q\Lambda_t Q^\top$ and solve for Λ_t to obtain

$$\begin{aligned}\Lambda_t &= Q^\top P_t Q = Q^\top P_{t-1} Q - Q^\top P_{t-1} F_t^\top A^{-1} F_t P_{t-1} Q \\ &= \Lambda_{t-1} - \Lambda_{t-1} Q^\top F_t^\top A^{-1} F_t Q \Lambda_{t-1}.\end{aligned}$$

It is easy to show that if Λ_{t-1} is symmetric and PSD, then so is Λ_t. Moreover, we can control the estimated rank of P by initializing the values of r, Q and Λ_0 appropriately.

Bonnabel and Sepulchre (2012) show that the noise component ξ_t helps stabilize tracking, but if we reintroduce the covariance Ξ_t, the update rule above is no longer rank-preserving. Instead, we introduce a noise component $\tilde{\Xi}_t \in \mathbb{R}^{r\times r}$ in the subspace defined by Q, and modify the update rule:

$$\tilde{\Lambda}_t = \Lambda_{t-1} + \tilde{\Xi}_t, \tag{11}$$

$$\Lambda_t = \tilde{\Lambda}_t - \tilde{\Lambda}_t Q^\top F_t^\top A^{-1} F_t Q \tilde{\Lambda}_t. \tag{12}$$

As long as $\tilde{\Xi}_t$ is a diagonal matrix with positive elements, the update rule preserves the rank and feasibility, and the time complexity of the matrix multiplications in (5) and (7) becomes $O(rdn)$.

3.3. Algorithm

Figure 1 shows the pseudo-code of the SKGD algorithm. The input to the algorithm is a time horizon T and a trajectory $(s_0, r_0, s_1, \dots, s_{T-1}, r_{T-1}, s_T)$. We initialize Λ_0 and U to be the eigenvalues and

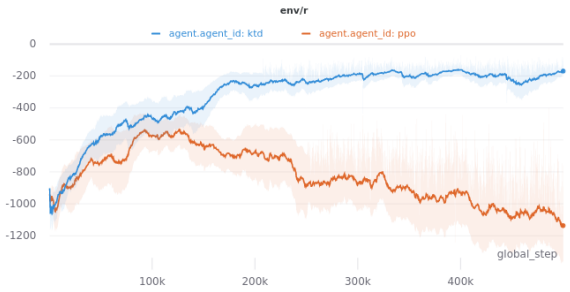

Figure 2: Cumulative return of the Forced Van Der Pol Oscillator, averaged over 10 seeds.

eigenvectors of the diagonal matrix αI_d, projected onto the subspace $\mathbb{R}^r$. To initialize Q, we follow the proposal of Bonnabel and Sepulchre (2012) and perform a QR-decomposition of the matrix $U - UU^\top U$. The parameter vector $\hat{\theta}_0$ is initialized using standard neural network techniques. At each time step t, we first perform dropout using the stochastic mask M. This allows us to control the time complexity also in the case for which the rank of P is large. We then update $\hat{\theta}_t$ and Λ_t using the update rules in (6) and (12).

4. Experiments

We evaluate SKGD in an online control setting under non-stationary dynamics. That is, we assume a sequence of transitions $(s, a, r, s') \sim \mathcal{M}$ under policy $\pi_{\theta_t}(a \mid s)$ and dynamics $p_t(\cdot \mid s, a)$ at every decision time t, and we estimate the value of the policy π_{θ_t} from a single tuple. A common benchmark algorithm is Proximal Policy Optimization, or PPO (Schulman et al., 2017), in which the policy and value functions are parametrized by neural networks. The policy π_{θ_t} is learned via direct policy optimization while the value function V_{ψ_θ} is optimized via the n-step temporal difference error (Sutton and Barto, 2018). The optimizer used is Adam (Kingma and Ba, 2015). Both algorithms are trained from scratch and they only differ in the policy evaluation step, where SKGD uses (5) and (7) to update the value estimator. We expect that the likelihood ratio estimator used in the policy optimization step suffers from high variance for short trajectories and thus performs poorly in this online setting. Hence we pick the smallest number of samples that allows SKGD or PPO to not degenerate over time. Our main focus was to capture different types of noise structures rather than different dynamics. We compare the two algorithms in two benchmarks domains: the Cartpole domain (Sutton and Barto, 2018; Coumans and Bai, 2016–2019) and the Forced Van Der Pol Oscillator (der Pol and Mark, 1927). We also test the algorithms on the real world problem of controlling an electrical microgrid (Totaro et al., 2020).

In the benchmark domains, we inject the non-stationarity as a time dependent force $\beta_{t+1} = \beta_t + \delta$ bounded in value $\beta_t \in (\beta_{min}, \beta_{\max}), \forall t \in 0, \ldots, T$, which resets at every $t = T$. The force is applied at the cart and pole joint in the case of the cartpole, and as a change in the drift coefficient μ in the case of the Forced Van Der Pol Oscillator.

We now discuss the specific experiments. The Forced Van Der Pol Oscillator is described by the following equation of motion and reward function:

$$\ddot{x} = \mu\dot{x}(1 - x^2) - x + a, \qquad r(x, a) = \|a\|^2 + \|x\|^2, \tag{13}$$

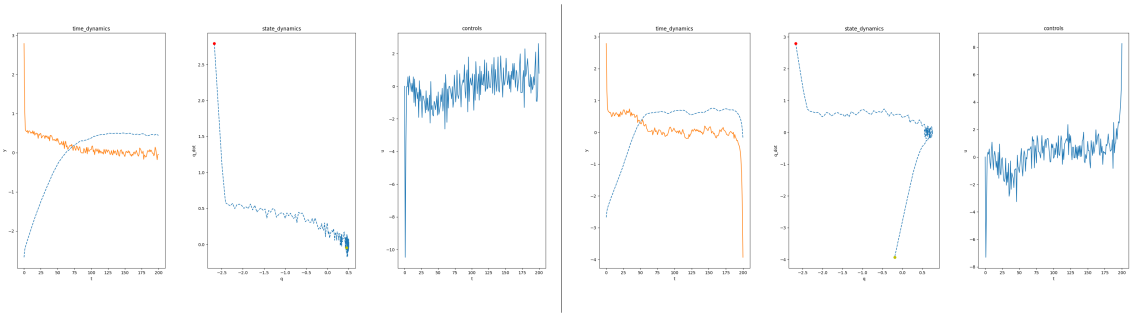

Figure 3: The left group represents the policy learnt by SKGD, and the right group that of PPO. For each group we plot the evolution the state component $\dot{x}_t, x_t$, the full state $\mathbf{x}_t = (x_t, \dot{x}_t)$ with starting position (red dot) and terminal position (grey dot), and the control a_t.

where $x \in R^2$, $a \in R$ and μ indicates the strength of the damping. The increment δ is chosen as $\mathcal{N}^+(\mu_0 + \delta, t\sigma)$ with $(\mu_0 = 0, \sigma = 0.03)$. In Figure 2 we show the return achieved by the two algorithms during training. Our method is able to adapt to the change in damping quickly enough to stop the oscillator in minimum time. In Figure 3, the policy found by our method is smoother than that found by PPO.

The Non-Stationary CartPole is described by the following equations:

$$f = 2\ddot{x} + \ddot{\theta}\cos\theta - \dot{\theta}^2 \sin\theta \tag{14}$$

$$0 = \ddot{x}\cos\theta + \ddot{\theta} + \sin\theta \tag{15}$$

$$r(x, a) = \begin{cases} 1, & \text{if} \quad \theta \in (\theta_{min}, \theta_{\max}), x \in (x_{min}, x_{max}), \\ 0, & \text{otherwise}, \end{cases} \tag{16}$$

obtained by setting all constants to 1 (Tedrake, 2009). We require $\beta_t \in (0, 5), \forall t \in \{0, \ldots, T\}$, with $\delta_t \sim \mathcal{N}(1e-4, 0.1t)$. In this domain the difference between the two algorithms is less evident because the necessary number of samples to obtain a non-degenerate policy is much larger than in the other domain.

4.1. Microgrid

Microgrids are small electrical networks composed of flexible consumption, distributed power generation (renewable and/or conventional) and storage devices. The non-stationarity is inherent in the demand and production of renewable energy. Direct estimation of the non-stationary processes is known to be an open research question. We test our methods on an open source benchmark (Totaro et al., 2020). Different from the other domains, this a finite horizon control problem, where data is seen only once. Due to the small data availability of 18 months, we split the dataset in two parts, training and testing, of 6 and 12 months respectively. The training set is used to pre-train a neural network that predicts the next demand and renewable production. We then transfer the features to augment the state space for the control problem. In Figure 5 we report the average reward.

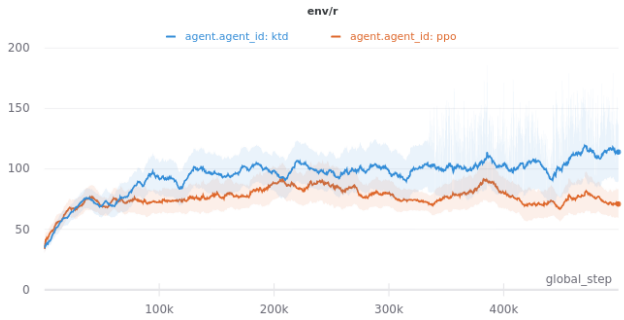

Figure 4: Cumulative return for the Non Stationary Cartplole, averaged over 10 seeds.

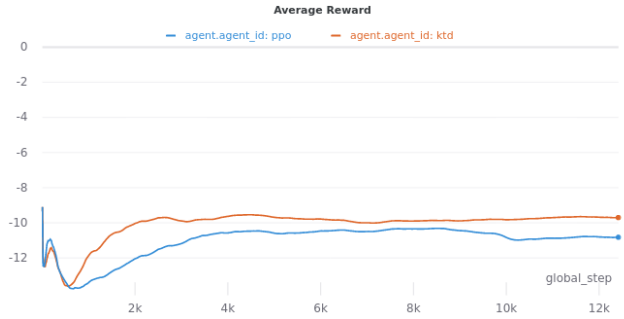

Figure 5: Average reward for the Microgrid benchmark.

5. Discussion

We present a practical algorithm for online control in non-stationary deterministic MDPs. Our method scales as $O(rd)$, where r is a low-rank estimation chosen by the system designer, and attempts to solve a long-standing issue in Extended Kalman Filtering via a low-rank, regularized update rule, that is guaranteed to be rank-preserving and of small norm. Despite having space complexity $O(rd)$, for large networks this is still prohibitive, but a promising direction for future work is to use a distributed implementation via a consensus algorithm, which might help to mitigate the computational burden. We conjecture that it might also mitigate the sampling bias incurred in stochastic transitions. Furthermore, non-stationarity can be addressed at decision time instead of at prediction time (Chandak et al., 2020), the two methods are not exclusive, and their combination is an interesting direction for future work.

Acknowledgments

The authors would like to thanks Matthieu Geist, Roberta Raileanu, and Gergely Neu for the insightful discussions. Anders Jonsson is partially funded by Spanish grants PID2019-108141GB-I00 and PCIN-2017-082.

References

B. D. O. Anderson and J. B. Moore. *Optimal Filtering*. PrenticeHall, 1979.

L. Baird. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pages 30–37. Elsevier, 1995.

S. Bonnabel and R. Sepulchre. The geometry of low-rank kalman filters. In *Matrix Information Geometry*, pages 53–68. Springer, 2012.

S. J. Bradke and A. G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22:33–57, 1996.

Y. Chandak, G. Theocharous, S. Shankar, S. Mahadevan, M. White, and P. S. Thomas. Optimizing for the future in non-stationary mdps. *arXiv preprint arXiv:2005.08158*, 2020.

R. T. Q. Chen, D. Choi, L. Balles, D. Duvenaud, and P. Hennig. Self-tuning stochastic optimization with curvature-aware gradient filtering. In *Proceedings on "I Can't Believe It's Not Better!" at NeurIPS Workshops*, Proceedings of Machine Learning Research 137, pages 60–69, 2020.

S. P. M. Choi, D.-Y. Yeung, and N. L. Zhang. Hidden-mode markov decision processes for nonstationary sequential decision making. In *Sequence Learning*, pages 264–287. Springer, 2000.

E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2019.

B. Van der Pol and J. Van Der Mark. Frequency demultiplication. *Nature*, 120(3019):363–364, 1927.

K. Doya, K. Samejima, K. Katagiri, and M. Kawato. Multiple model-based reinforcement learning. *Neural computation*, 14(6):1347–1369, 2002.

C. Gehring, Y. Pan, and M. White. Incremental truncated lstd. *arXiv preprint arXiv:1511.08495*, 2015.

M. Geist and O. Pietquin. Kalman temporal differences. *Journal of artificial intelligence research*, 39:483–532, 2010.

G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis. Analysis and improvement of the consistency of extended kalman filter based slam. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 473–479, 2008.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82:35–45, 1960.

D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. In *ICLR 2015 : International Conference on Learning Representations 2015*. Ithaca, NYarXiv.org, 2015.

E. Lecarpentier and E. Rachelson. Non-stationary markov decision processes, a worst-case approach using model-based reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 7216–7225, 2019.

M. Mahsereci. *Probabilistic Approaches to Stochastic Optimization*. PhD thesis, Eberhard Karls Universität Tübingen, Germany, 2018.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, second edition, 2006.

R. E. Plackett. Some theorems in least squares. *Biometrika*, 37:149, 1950.

M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

R. Raileanu, M. Goldstein, A. Szlam, and R. Fergus. Fast adaptation via policy-dynamics value functions. *arXiv preprint arXiv:2007.02879*, 2020.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. 2017.

M. Seeger. Low rank updates for the cholesky decomposition. 01 2004.

S. Di-Castro Shashua and S. Mannor. Kalman meets bellman: Improving policy evaluation through value tracking. *arXiv preprint arXiv:2002.07171*, 2020.

A. Spantini, A. Solonen, T. Cui, J. Martin, L. Tenorio, and Y. Marzouk. Optimal Low-rank Approximations of Bayesian Linear Inverse Problems. *SIAM J. Sci. Comput.*, 37(6):2451–2487, 2015.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

R. Tedrake. Underactuated robotics: Learning, planning, and control for efficient and agile machines course notes for mit 6.832. *Working draft edition*, 3, 2009.

S. Totaro, I. Boukas, A. Jonsson, and B. Cornélusse. Lifelong control of off-grid microgrid with model based reinforcement learning. *arXiv preprint arXiv:2005.08006*, 2020.

J. Vuckovic. Kalman gradient descent: Adaptive variance reduction in stochastic optimization, 2018.

M. A. Wiering. Reinforcement learning in dynamic environments using instantiated information. In *Machine Learning: Proceedings of the Eighteenth International Conference (ICML2001)*, pages 585–592, 2001.