Kalman Optimization for Value Approximation

Shirli Di-Castro Shashua
Technion, Israel
shirlidi@campus.technion.ac.il

Shie Mannor
Technion, Israel
shie@ee.technion.ac.il

Abstract

Policy evaluation is a key process in reinforcement learning. It assesses a given policy by estimating the corresponding value function. When using a parameterized value function, its set of parameters are commonly optimized by minimizing the sum of squared Bellman Temporal Differences errors. However, this approach ignores distributional properties of the errors and value parameters. Considering these distributions in the optimization process can provide useful information on the amount of confidence in value estimation. In this work we propose to optimize the value by minimizing a regularized objective function which forms a trust region over its parameters. We present a novel optimization method, called KOVA, based on the Extended Kalman Filter. KOVA minimizes the regularized objective function by adopting a Bayesian perspective over both the value parameters and noisy observed returns. This distributional property provides information on parameter uncertainty in addition to value estimates. We provide theoretical results of our approach and analyze the performance of KOVA on domains with large state and action spaces.

1 Introduction

Reinforcement learning (RL) solves sequential decision making problems by considering an agent that interacts with the environment and seeks the optimal policy [24]. During the learning process, the agent is required to evaluate its policies using a value function. In many real world RL domains, such as robotics, games and autonomous driving cars, the state and action spaces are large; hence, the value function is approximated, e.g., using a Deep Neural Network (DNN). A common approach is to optimize a set of parameters by minimizing the sum of squared Bellman Temporal Differences (TD) errors [4]. There are two underlying assumptions in this approach: first, the value and its parameters are deterministic; second, the Bellman TD errors are independent Gaussian random variables (RVs) with zero mean and a fixed variance. Although a commonly used objective function, these underlying assumptions may not be suitable for the policy evaluation task in RL. Distributional RL [2] refers to the second assumption and argues in favor of a full distribution perspective over the sum of discounted rewards for a fixed policy. In particular, learning this distribution is meaningful in presence of value approximation. However, in their formulation the value parameters are still considered deterministic and they do not provide an amount of confidence for the value estimates.

Treating the value or its parameters as RVs has been investigated in the RL literature. Engel et al. [7, 8], in their algorithm GPTD, used Gaussian Processes (GPs) for the value and the return to capture uncertainties in policy evaluation. Geist & Pietquin [9] proposed an algorithm called KTD which uses the Unscented Kalman filter (UKF) to learn the uncertainty in value parameters. Their formulation requires many samples of parameters in each training step, which is not feasible in Deep Reinforcement Learning (DRL) with large state and action spaces.

Motivated by the works of Engel et al. [7, 8] and Geist & Pietquin [9], we present in this work a unified framework for addressing uncertainties while approximating the value in DRL domains. Our framework, unlike GPTD and KTD, is feasible when using complex nonlinear approximation functions as DNNs, it adapts the uncertainty concepts to modern RL tasks and can be adjusted to estimate the value in both on-policy and off-policy optimization algorithms. Our framework incorporates the well-known Kalman filter estimation techniques with RL principles to improve value

Optimization Foundations for Reinforcement Learning Workshop at NeurIPS 2019, Vancouver, Canada.

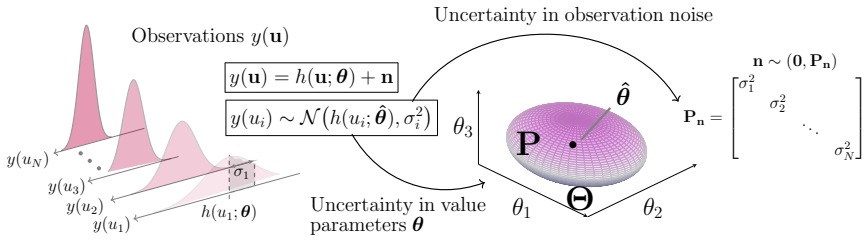


Figure 1: A Bayesian perspective for the policy evaluation problem in RL. The randomness of noisy observation $y(u)$ originates from two sources: (i) the randomness of its mean, the value $h(u;\boldsymbol{\theta})$ through the dependency on the random parameters $\boldsymbol{\theta}$. (ii) the random zero-mean noise n which relates to the stochastic transitions in the trajectory and to the possibly random policy.

approximation. The Kalman filter [13] and its variant for nonlinear approximations, the Extended Kalman filter (EKF) [1, 10], are used for on-line tracking and for estimating states in dynamic environments through indirect noisy observations. These methods have been successfully applied to numerous control dynamic systems such as navigation and tracking targets [20]. The Kalman filter can also be used for parameter estimation in approximation functions, in which parameters replace the states of dynamic systems.

We develop a new optimization method for policy evaluation based on the EKF formulation. Figure 1 illustrates our Bayesian perspective over value parameters and noisy observed returns. Our main contributions are: (1) developing a new regularized objective function for approximating values in the policy evaluation task - the regularization term accounts for both parameters and observations uncertainties; (2) presenting a novel optimization algorithm, **K**alman **O**ptimization for **V**alue **A**pproximation (KOVA), and proving that it minimizes at each time step the regularized objective function - this optimizer can be easily plugged into any policy optimization algorithm that relay on value function estimation, and improve it; (3) demonstrating the improvement achieved by our optimizer on several control tasks with large state and action spaces.

2 Background

Reinforcement Learning and MDPs: The standard RL setting considers an interaction of an agent with an environment $\mathcal{E}$ for a discrete number of time steps. The environment is modeled as a Markov Decision Process (MDP) $\{\mathcal{S}, \mathcal{A}, P, R, \gamma\}$ where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state transition probabilities for each state s and action a, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a bounded reward function and γ is a discount factor. At each time step t, the agent observes state $s_t \in \mathcal{S}$ and chooses action $a_t \in \mathcal{A}$ according to a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$. The agent receives an immediate reward $r_t(s_t, a_t)$ and the environment stochastically steps to state $s_{t+1} \in \mathcal{S}$ according to the probability distribution $P(s_{t+1}|s_t, a_t)$. The state value function and the state-action Q-function are used for evaluating the performance of a fixed policy π [24]: $V^\pi(s) = \mathbb{E}^\pi\big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)|s_0 = s\big]$ and $Q^\pi(s, a) = \mathbb{E}^\pi\big[\sum_{t=0}^{\infty} \gamma^t r_t(s_t, a_t)|s_0 = s, a_0 = a\big]$, where $\mathbb{E}^\pi$ denotes the expectation with respect to the state (state-action) distribution induced by transition law P and policy π.

Value Function Estimation: Policy evaluation is a core element in RL algorithms. We will use the term *value function* (VF) to address the following functions: $V^\pi(s)$, $Q^\pi(s, a)$ and the advantage function $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$. When the state or action space is large, the VF is approximated using a parameterized function, $h(\cdot;\boldsymbol{\theta})$. A common approach for optimizing VF parameters is to minimize at each time step t the empirical mean of the squared *Bellman TD error* $\delta(u;\boldsymbol{\theta}_t) \triangleq y(u) - h(u;\boldsymbol{\theta}_t)$, over N samples generated form environment $\mathcal{E}$ under a given policy:

$$L_t^{\text{MLE}}(\boldsymbol{\theta}_t) = \frac{1}{2N} \sum_{i=1}^{N} \delta^2(u_i;\boldsymbol{\theta}_t). \tag{1}$$

We use the general notation u to specify the *input* for the *target label* $y(u)$ and for the approximated value at time t, $h(u;\boldsymbol{\theta}_t)$. For example, for $h(u;\boldsymbol{\theta}_t) = V(s_m;\boldsymbol{\theta}_t)$, $u = s_m$ is the state at a discrete time m; For $h(u;\boldsymbol{\theta}_t) = Q(s_m, a_m;\boldsymbol{\theta}_t)$, $u = (s_m, a_m)$ is the state-action pair. In Table 1 we provide examples of several options for $y(u)$ and $h(u;\boldsymbol{\theta}_t)$ which clarify how this general notation can be utilized in known policy optimization algorithms. Traditionally, the VF is trained by a

Table 1: Examples for policy optimization algorithms and their Bellman TD error $\delta(u;\boldsymbol{\theta}_t)$ type. The decomposition of $\delta(u;\boldsymbol{\theta}_t)$ into the observation function $h(u;\boldsymbol{\theta}_t)$ and the target label $y(u)$ in the EKF model (2) enables the integration of our KOVA optimizer with any policy optimization algorithm that relay on value function estimation. $\boldsymbol{\theta}'$ refers to a fixed network, different than the one being trained $\boldsymbol{\theta}_t$.

Algorithm	$\delta(u;\boldsymbol{\theta}_t)$ *type*	$h(u;\boldsymbol{\theta}_t)$	$y(u)$
A3C [17]	k-step V-evaluation	$V(s_m;\boldsymbol{\theta}_t)$	$\sum_{i=0}^{k-1}\gamma^i r_{m+i}+\gamma^k V(s_{m+k};\boldsymbol{\theta}')$
DDPG [15]	1-step Q-evaluation	$Q(s,a;\boldsymbol{\theta}_t)$	$r+\gamma Q(s',\pi(s');\boldsymbol{\theta}')$
PPO [23] TRPO [21]	GAE [22]	$V(s_m;\boldsymbol{\theta}_t)$	$\sum_{i=0}^{\infty}(\gamma\lambda)^i\big(r_{m+i}+\gamma V(s_{m+i+1};\boldsymbol{\theta}')$ $-V(s_{m+i};\boldsymbol{\theta}')\big)+V(s_m;\boldsymbol{\theta}')$
DQN [16]	Optimality equation	$Q(s,a;\boldsymbol{\theta}_t)$	$r+\gamma\max_{a'}Q(s',a';\boldsymbol{\theta}')$
SAC [11]	1-step V-evaluation 1-step Q-evaluation	$V(s;\boldsymbol{\theta}_t)$ $Q(s,a;\tilde{\boldsymbol{\theta}}_t)$	$\mathbb{E}_a[Q(s,a;\tilde{\boldsymbol{\theta}}_t)-\log\pi(s\|a)]$ $r+\gamma\mathbb{E}_{s'}[V(s';\boldsymbol{\theta}')]$

gradient method, estimating the loss on each experience as it is encountered, yielding the update: $\boldsymbol{\theta}_{t+1}\leftarrow\boldsymbol{\theta}_t+\alpha\mathbb{E}_{u\sim p(\cdot)}\big[\big(y(u)-h(u;\boldsymbol{\theta}_t)\big)\nabla_{\boldsymbol{\theta}_t}h(u;\boldsymbol{\theta}_t)\big]$, where α is the learning rate and $p(\cdot)$ is the experience distribution. We will show (Section 3) that the underlying assumptions in L_t^{MLE} (1) are that the parameters $\boldsymbol{\theta}_t$ are deterministic and that the target labels $y(u)$ are independent Gaussian RVs with mean $h(u;\boldsymbol{\theta}_t)$ and a fixed variance.

Extended Kalman Filter (EKF): In this section we briefly outline the Extended Kalman filter [1, 10], which serves as a base for our novel regulerized objective function L_t^{EXF} (4). The EKF is a standard technique for estimating the state of a nonlinear dynamic system or for learning the parameters of a nonlinear approximation function. In this paper we will focus on its latter role, meaning estimating $\boldsymbol{\theta}$. The EKF considers the following model:

$$\begin{cases}\boldsymbol{\theta}_t=\boldsymbol{\theta}_{t-1}+\mathbf{v}_t\\ y(\mathbf{u}_t)=h(\mathbf{u}_t;\boldsymbol{\theta}_t)+\mathbf{n}_t\end{cases}, \tag{2}$$

where $\boldsymbol{\theta}_t\in\mathbb{R}^{d\times 1}$ are the parameters evaluated at time t, $\mathbf{v}_t$ is the evolution noise and $\mathbf{n}_t$ is the observation noise, both modeled as additive and white noises with covariances $\mathbf{P}_{\mathbf{v}_t}$ and $\mathbf{P}_{\mathbf{n}_t}$, respectively. $y(\mathbf{u}_t)$ is the N-dimensional observations vector at time t: $y(\mathbf{u}_t)=[y(u_t^1),y(u_t^2),\ldots,y(u_t^N)]^\top\in\mathbb{R}^{N\times 1}$, and $h(\mathbf{u}_t;\boldsymbol{\theta}_t)\in\mathbb{R}^{N\times 1}$ is an N-dimensional vector, where $h(u;\boldsymbol{\theta})$ is a nonlinear observation function with input u and parameters $\boldsymbol{\theta}$: $h(\mathbf{u}_t;\boldsymbol{\theta}_t)=[h(u_t^1;\boldsymbol{\theta}_t),h(u_t^2;\boldsymbol{\theta}_t),\ldots,h(u_t^N;\boldsymbol{\theta}_t)]^\top$. As seen in (2), EKF treats the parameters $\boldsymbol{\theta}_t$ as RVs, similarly to Bayesian approaches that assume that the parameters belong to an uncertainty set Θ governed by the mean and covariance of the parameters distribution. The estimation at time t, denoted as $\hat{\boldsymbol{\theta}}_{t|\cdot}$ is the conditional expectation of the parameters with respect to the observed data: $\hat{\boldsymbol{\theta}}_{t|t}\triangleq\mathbb{E}[\boldsymbol{\theta}_t|y_{1:t}]$ and $\hat{\boldsymbol{\theta}}_{t|t-1}\triangleq\mathbb{E}[\boldsymbol{\theta}_t|y_{1:t-1}]=\hat{\boldsymbol{\theta}}_{t-1|t-1}$. With some abuse of notation, $y_{1:t'}$ are the observations gathered up to time t': $y(\mathbf{u}_1),\ldots,y(\mathbf{u}_{t'})$. The *parameters error* is $\tilde{\boldsymbol{\theta}}_{t|\cdot}\triangleq\boldsymbol{\theta}_t-\hat{\boldsymbol{\theta}}_{t|\cdot}$ and the conditional *error covariance* is $\mathbf{P}_{t|\cdot}\triangleq\mathbb{E}\big[\tilde{\boldsymbol{\theta}}_{t|\cdot}\tilde{\boldsymbol{\theta}}_{t|\cdot}^\top|y_{1:\cdot}\big]$. EKF then uses the following updates:

$$\begin{cases}\hat{\boldsymbol{\theta}}_{t|t}^{\text{EKF}}=\hat{\boldsymbol{\theta}}_{t|t-1}+\mathbf{K}_t\big(y(\mathbf{u}_t)-h(\mathbf{u}_t;\hat{\boldsymbol{\theta}}_{t|t-1})\big),\\ \mathbf{P}_{t|t}=\mathbf{P}_{t|t-1}-\mathbf{K}_t\mathbf{P}_{\tilde{\mathbf{y}}_t}\mathbf{K}_t^\top.\end{cases} \tag{3}$$

where $\tilde{\mathbf{y}}_{t|t-1}$ is the *Observation innovation*, $\mathbf{P}_{\tilde{\mathbf{y}}_t}$ is the *Covariance of the innovation* and $\mathbf{K}_t$ is the *Kalman gain*. In the next section we present how to use the EKF formulation in order to approximate VFs which consider uncertainty both in the parameters and in the noisy observations.

3 EKF for Value Function Approximation

We now derive a novel regularized objective function, L_t^{EXF} (4), and argue in its favor for optimizing value functions in RL. We use general notations in order to enable integration of our proposed VF optimization method with any policy optimization algorithm that relay on value function estimation. The main idea is to decompose the Bellman TD error vector $\delta(\mathbf{u}_t;\boldsymbol{\theta}_t)$ into two parts: $\delta(\mathbf{u}_t;\boldsymbol{\theta}_t)=y(\mathbf{u}_t)-h(\mathbf{u}_t;\boldsymbol{\theta}_t)=[\delta(u_t^1;\boldsymbol{\theta}_t),..,\delta(u_t^N;\boldsymbol{\theta}_t)]^\top$. (i) The observation at time t, $y(\mathbf{u}_t)$ is a vector that contains N target labels $y(u_t^1),\ldots,y(u_t^N)$. (ii) The observation function $h(u;\boldsymbol{\theta}_t)$ may be one

of the following: $V(s;\boldsymbol{\theta}_t), Q(s,a;\boldsymbol{\theta}_t)$ or $A(s,a;\boldsymbol{\theta}_t)$. The observation functions for N inputs are concatenated into the N-dimensional vector $h(\mathbf{u}_t;\boldsymbol{\theta}_t)$. In Table 1 we provide several examples for the Bellman TD error decomposition according to the chosen policy optimization algorithm.

Our goal is to estimate the parameters $\boldsymbol{\theta}_t$. One way is to learn them by maximum likelihood estimation (MLE) using stochastic gradient descent methods: $\boldsymbol{\theta}^{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \log p(y_{1:t}|\boldsymbol{\theta})$. This forms the objective function in Equation (1). Another way is learning them by a Bayesian approach which uses Bayes rule and adds prior knowledge over the parameters $p(\boldsymbol{\theta})$ to calculate the maximum a-posteriori (MAP) estimator: $\boldsymbol{\theta}^{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|y_{1:t}) = \arg\max_{\boldsymbol{\theta}} \log p(y_{1:t}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})$. Given the observations gathered up to time t, we can write the MAP estimator in a different form: $\boldsymbol{\theta}_t^{\text{MAP}} = \arg\max_{\boldsymbol{\theta}_t} \log p(y_t|\boldsymbol{\theta}_t) + \log p(\boldsymbol{\theta}_t|y_{1:t-1})$. Here, instead of using the parameters prior, we use an equivalent derivation for the parameters posterior conditioned on $y_{1:t}$, based on the likelihood of a *single* observation $y_t \triangleq y(\mathbf{u}_t)$ and the posterior conditioned on $y_{1:t-1}$ [27]. This unique derivation is a key step for the incremental Kalman updates and for defining L_t^{EKF} (4). In order do define the likelihood $p(y_t|\boldsymbol{\theta}_t)$ and the posterior $p(\boldsymbol{\theta}_t|y_{1:t-1})$, we adopt the EKF model (2), and make the following assumptions:

Assumption 1. *The likelihood $p(y(\mathbf{u}_t)|\boldsymbol{\theta}_t)$ is Gaussian: $y(\mathbf{u}_t)|\boldsymbol{\theta}_t \sim \mathcal{N}(h(\mathbf{u}_t;\boldsymbol{\theta}_t), \mathbf{P}_{\mathbf{n}_t})$.*

Assumption 2. *The posterior distribution $p(\boldsymbol{\theta}_t|y_{1:t-1})$ is Gaussian: $\boldsymbol{\theta}_t|y_{1:t-1} \sim \mathcal{N}(\hat{\boldsymbol{\theta}}_{t|t-1}, \mathbf{P}_{t|t-1})$.*

These assumptions are common when using the EKF. In the context of RL, these assumptions add the flexibility we want: the value is treated as a RV and information is gathered on the uncertainty of its estimate. In addition, the noisy observations (the target labels), can have different variances and can even be correlated. Based on these Gaussian assumptions, we can derive the following Theorem:

Theorem 1. *Under Assumptions 1 and 2, $\hat{\boldsymbol{\theta}}_{t|t}^{EKF}$ (3) minimizes at each time step t the following regularized objective function:*

$$L_t^{EKF}(\boldsymbol{\theta}_t) = \frac{1}{2}\delta(\mathbf{u}_t;\boldsymbol{\theta}_t)^\top \mathbf{P}_{\mathbf{n}_t}^{-1}\delta(\mathbf{u}_t;\boldsymbol{\theta}_t) + \frac{1}{2}(\boldsymbol{\theta}_t - \hat{\boldsymbol{\theta}}_{t|t-1})^\top \mathbf{P}_{t|t-1}^{-1}(\boldsymbol{\theta}_t - \hat{\boldsymbol{\theta}}_{t|t-1}). \tag{4}$$

We now explicitly write the expressions that appear in Equation (3) (see the supplementary material for Theorem 1 proof and for more detailed derivations). The derivations are based on the first order Taylor series linearization for the observation function $h(\boldsymbol{\theta}_t)$: $h(\mathbf{u}_t;\boldsymbol{\theta}_t) = h(\mathbf{u}_t;\hat{\boldsymbol{\theta}}) + \nabla_{\boldsymbol{\theta}_t} h(\mathbf{u}_t;\hat{\boldsymbol{\theta}})^\top \big(\boldsymbol{\theta}_t - \hat{\boldsymbol{\theta}}\big)$, where $\nabla_{\boldsymbol{\theta}_t} h(\mathbf{u}_t;\hat{\boldsymbol{\theta}}) = \big[\nabla_{\boldsymbol{\theta}_t} h(u_t^1;\hat{\boldsymbol{\theta}}), \nabla_{\boldsymbol{\theta}_t} h(u_t^2;\hat{\boldsymbol{\theta}}), \ldots, \nabla_{\boldsymbol{\theta}_t} h(u_t^N;\hat{\boldsymbol{\theta}})\big] \in \mathbb{R}^{d\times N}$ and $\hat{\boldsymbol{\theta}}$ is typically chosen to be the previous estimation of the parameters at time $t-1$, $\hat{\boldsymbol{\theta}} = \hat{\boldsymbol{\theta}}_{t|t-1}$. Then, *the covariance of the innovation* and the Kalman gain become:

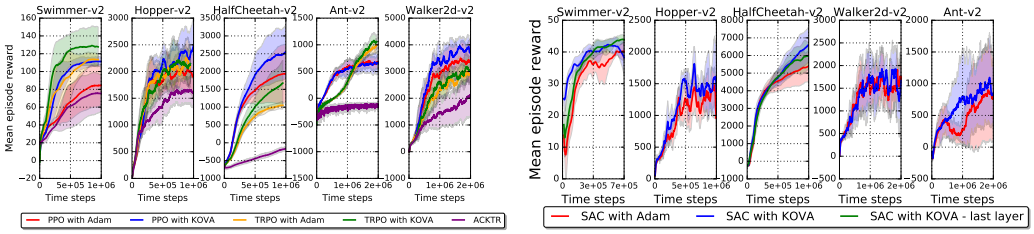
$$\mathbf{P}_{\tilde{\mathbf{y}}_\mathbf{t}} = \nabla_{\boldsymbol{\theta}_t} h(\mathbf{u}_t;\hat{\boldsymbol{\theta}})^\top \mathbf{P}_{t|t-1} \nabla_{\boldsymbol{\theta}_t} h(\mathbf{u}_t;\hat{\boldsymbol{\theta}}) + \mathbf{P}_{\mathbf{n}_t}. \tag{5}$$

$$\mathbf{K}_t = \mathbf{P}_{t|t-1} \nabla_{\boldsymbol{\theta}_t} h(\mathbf{u}_t,\hat{\boldsymbol{\theta}}) \big(\nabla_{\boldsymbol{\theta}_t} h(\mathbf{u}_t;\hat{\boldsymbol{\theta}})^\top \mathbf{P}_{t|t-1} \nabla_{\boldsymbol{\theta}_t} h(\mathbf{u}_t;\hat{\boldsymbol{\theta}}) + \mathbf{P}_{\mathbf{n}_t}\big)^{-1}. \tag{6}$$

Comparing between L_t^{EKF} and L_t^{MLE} for Optimizing Value Functions: We argue in favor of using $L_t^{\text{EKF}}(\boldsymbol{\theta}_t)$ (4) for optimizing VFs instead of the commonly used $L_t^{\text{MLE}}(\boldsymbol{\theta}_t)$ (1). Corollary 1 will assist us to discuss and compare between the two objective functions:

Corollary 1. *Under Assumptions 1 and 2, consider a diagonal covariance $\mathbf{P}_{\mathbf{n}_t}$ with diagonal elements $\sigma_i = N$ and assume $\mathbf{P}_{0|0} = \mathbf{P}_{\mathbf{v}_t} = \mathbf{0}$, then: $L_t^{EKF}(\boldsymbol{\theta}_t) = L_t^{MLE}(\boldsymbol{\theta}_t)$.*

According to Corollary 1, the two objective functions are the same if we consider the parameters as deterministic and if we assume that the noisy target labels have a fixed variance. So what are the differences between the two objective functions? First, L_t^{EKF} is a regularized version of L_t^{MLE}: the regularization is causing the parameters $\boldsymbol{\theta}_t$ to *track* the recent parameters estimate, $\hat{\boldsymbol{\theta}}_{t|t-1}$, stabilizing the estimate process. The error between the successive estimates is weighted with the inverse of the uncertainty information $\mathbf{P}_{t|t-1}$. L_t^{MLE} does not include a regularization term, meaning it does not account for parametrization uncertainties. Note that when adding a standard L_2 regularization to L_t^{MLE}, often common in DNNs, it reflects staying close to the $\mathbf{0}$ vector which is not always desired. Second, L_t^{EKF} weights the squared Bellman TD error vector $\delta(\mathbf{u}_t;\boldsymbol{\theta}_t)$ with $\mathbf{P}_{\mathbf{n}_t}^{-1}$ which can be interpreted as an additional regularization technique. $\mathbf{P}_{\mathbf{n}_t}$ can be viewed as the amount of confidence we have in the observations, as defined in the EKF model (2): if the observations are noisy, we should consider larger values for the diagonal elements in the covariance $\mathbf{P}_{\mathbf{n}_t}$. In addition, L_t^{EKF} allows us to model correlations between observations errors, unlike the i.i.d assumption in L_t^{MLE}. In Section 4 we discuss possible options for $\mathbf{P}_{\mathbf{n}_t}$.



(a) (b)

Figure 2: Mean episode reward during training for Mujoco environments. **(a)** on-policy algorithms: PPO, TRPO, ACKTR; **(b)** off-policy algorithm SAC. We present the average (solid lines) and standard deviation (shaded area) of the mean episode reward over 8 runnings, generated from random seeds.

Looking at the parameters update in Equation (3) and the definition of the Kalman gain $\mathbf{K}_t$ in Equation (6), we can see that the Kalman gain propagates the new information from the noisy target labels, back down into the parameters uncertainty set Θ, before combining it with the estimated parameter value. Actually, $\mathbf{K}_t$ can be interpreted as an adaptive learning rate for each individual parameter that implicitly incorporates the uncertainty of each parameter. This approach resembles familiar stochastic gradient optimization methods such as Adagrad [6], AdaDelta [29], RMSprop [25] and Adam [14], for different choices of $\mathbf{P}_{t|t-1}$ and $\mathbf{P}_{\mathbf{n}_t}$. We refer the reader to Ruder [19].

The following Theorem formalizes the **connection between L_t^{EKF} and two separate KL-divergences**. It illustrates how EKF is aimed at minimizing two separate KL-divergences. The first is the KL divergence between two conditional distributions and it is equivalent to the loss in L_t^{MLE} (1). The second is the KL divergence between two different parameterizations of the joint learned distribution $P_{u,y}$. This term imposes trust region on the VF parameters in L_t^{EKF} (4).

Theorem 2. *Assume the inputs u are drawn independently from a training distribution $\hat{Q}_u$, and the observations y are drawn from a conditional training distribution $\hat{Q}_{y|u}$. Let $P_{u,y}(\boldsymbol{\theta})$ and $P_{y|u}(\boldsymbol{\theta})$ be the learned joint and conditional distributions, respectively. Define $C = \log\left(\frac{1}{(2\pi)^{N/2}|\mathbf{P}_{\mathbf{n}_t}|^{1/2}}\right)$. Under Assumptions 1 and 2, consider $\mathbf{P}_{\mathbf{n}_t}$ with diagonal elements $\sigma_i = N$, then:*
$L_t^{EKF}(\boldsymbol{\theta}_t) = C + N\mathbb{E}_{\hat{Q}_u}[D_{KL}(\hat{Q}_{y|u}||P_{y|u}(\boldsymbol{\theta}))] + t \cdot D_{KL}\big(P_{u,y}(\boldsymbol{\theta}+\Delta\boldsymbol{\theta})||P_{u,y}(\boldsymbol{\theta})\big) + \mathcal{O}(\|\Delta\boldsymbol{\theta}\|^3)$.

Practical algorithm: KOVA optimizer. We now derive a practical algorithm for approximating VFs, by minimizing L_t^{EKF} (4). In practice we use the update Equations (3) and the Kalman gain Equations in (5)-(6) in order to avoid inversing $\mathbf{P}_{t|t-1}$. In addition, we add a fixed learning rate α to smooth the update. The KOVA optimizer is presented in Algorithm 1. Notice that $\mathcal{R}$ is a sample generator whose structure depends on the policy algorithm for which KOVA is used as a VF optimizer. $\mathcal{R}$ can contain trajectories from a fixed policy or it can be an experience replay which contains transitions from several different policies.

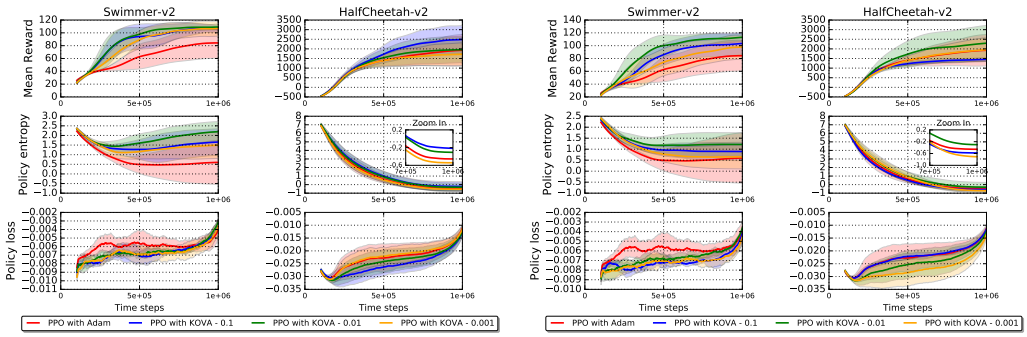
Algorithm 1 KOVA Optimizer

Input: $\mathbf{P}_{0|0}, \mathbf{P}_{\mathbf{v}_t}, \mathbf{P}_{\mathbf{n}_t}, \alpha, \mathcal{R}$. **Initialize:** $\hat{\boldsymbol{\theta}}_{0|0}$.
1: **for** $t = 1, \ldots, T$ **do**
2: Set predictions: $\begin{cases}\hat{\boldsymbol{\theta}} = \hat{\boldsymbol{\theta}}_{t|t-1} = \hat{\boldsymbol{\theta}}_{t-1|t-1} \\ \mathbf{P}_{t|t-1} = \mathbf{P}_{t-1|t-1} + \mathbf{P}_{\mathbf{v}_t}\end{cases}$
3: Sample N tuples $\{y(u^i), h(u^i;\hat{\boldsymbol{\theta}})\}_{i=1}^N$ from $\mathcal{R}$.
4: Construct N-dim vectors $y(\mathbf{u}_t)$ and $h(\mathbf{u}_t, \hat{\boldsymbol{\theta}})$.
5: Compute $(d \times N)$-dim matrix $\nabla_{\boldsymbol{\theta}} h(\mathbf{u}_t; \hat{\boldsymbol{\theta}})$.
6: Compute $\mathbf{P}_{\tilde{\mathbf{y}}_t}$ (5) and $\mathbf{K}_t$ (6).
7: Set updates:
$\begin{cases}\hat{\boldsymbol{\theta}}_{t|t} = \hat{\boldsymbol{\theta}}_{t|t-1} + \alpha\mathbf{K}_t\big(y(\mathbf{u}_t) - h(\mathbf{u}_t; \hat{\boldsymbol{\theta}}_{t|t-1})\big) \\ \mathbf{P}_{t|t} = \mathbf{P}_{t|t-1} - \alpha\mathbf{K}_t\mathbf{P}_{\tilde{\mathbf{y}}_t}\mathbf{K}_t^\top\end{cases}$
8: **end for**
Output: $\hat{\boldsymbol{\theta}}_{t|t}$ and $\mathbf{P}_{t|t}$

4 Experiments

We now present experiments that illustrate the performance attained by our KOVA optimizer. Technical details on policy and VF networks, on hyper-parameters grid search, on the hyper-parameters we used and on the running time of the algorithms are described in the supplementary material.

KOVA optimizer for policy evaluation in on-policy setting: We tested the performance of KOVA in domains with continuous state and action spaces: the robotic tasks benchmarks implemented in OpenAI Gym [3], which use the MuJoCo physics engine [26]. In this experiment we used on-policy algorithms for policy training: PPO [23] and TRPO [21] with their baselines implementations [5]. For VF training we replaced the originally used Adam optimizer [14] with our KOVA optimizer



(a) (b)

Figure 3: Mean episode reward, policy entropy and the policy loss for a PPO agent in Swimmer-v2 and HalfCheetah-v2. We compare between optimizing the VF with Adam vs. our KOVA optimizer. We present three different values for $\eta = 0.1, 0.01, 0.001$ and two different values for the diagonal elements in $\mathbf{P}_{\mathbf{n}_t}$: **(a)** max-ratio and **(b)** batch-size. We present the average (solid lines) and standard deviation (shaded area) of the mean episode reward over 8 runnings, generated from random seeds.

(Algoritm 1) and compared their effect on the mean episode reward in each environment. In addition, we tested the performance of ACKTR [28] which uses second order optimization for both policy and value functions. During training, each episode reward was recorded and in every optimization update we calculated the mean over the last 100 episodes. An optimization update is executed after several timesteps, defined by the horizon hyper-parameter of the algorithm. The results are presented in Figure 2(a), where we chose to use timesteps in order to compare between different algorithms on the same graph. We can see that KOVA improved the agent's performance in four out of five environments, both for PPO and TRPO. In Ant-v2 it kept approximately the same performance. In all environments KOVA outperformed ACKTR.

KOVA optimizer for policy evaluation in off-policy setting:: We investigated how the KOVA optimizer affects the performance of off-policy algorithms. For the policy training we used SAC [11] with their stable baselines implementations [12]. For VF training we replaced the originally used Adam optimizer with our KOVA optimizer (Algoritm 1) and compared their effect on the mean episode reward in each environment. In addition, we tested how we can benefit from the KOVA performance and keep the running time low. For this purpose we performed an additional experiment, where we applied the KOVA optimizer only on the last layer of the VF. The experiment description for the off-policy setting is similar to the on-policy setting. The results are presented in Figure 2(b). We can see that KOVA improved the agent's performance in four out of five environments. In Walker2d-v2 it kept approximately the same performance.

We believe that these improvements, in the on-policy and off-policy settings, come from both using a full covariance instead of a diagonal one, and from properly using $\mathbf{P}_{\mathbf{n}_t}$ to include observations variance. These improvements demonstrate the importance of incorporating uncertainty estimation in value function approximation for improving the agent's performance and suggest that EKF should not be neglected and should be considered as a better optimizer for VFs.

Investigating the evolution and observation noises: The most interesting hyper-parameters in KOVA are related to the covariances $\mathbf{P}_{\mathbf{v}_t}$ and $\mathbf{P}_{\mathbf{n}_t}$. As seen in Corollary 1, for deterministic interpretation of the parameters we simply set $\mathbf{P}_{\mathbf{v}_t} = \mathbf{0}$. However, the more interesting setting would be $\mathbf{P}_{\mathbf{v}_t} = \frac{\eta}{1-\eta}\mathbf{P}_{t-1|t-1}$ with η being a small number that controls the amount of fading memory [18]. $\mathbf{P}_{\mathbf{n}_t}$ can be used for incorporating prior domain knowledge. For example, a diagonal matrix implies independent observations , while if observations are known to be correlated, additional non-diagonal elements can be added. We investigated the effect of different values of η and $\mathbf{P}_{\mathbf{n}_t}$ in the Swimmer and HalfCheetah environments, where KOVA gained the most success. The results are depicted in Figure 3. We tested two different $\mathbf{P}_{\mathbf{n}_t}$ settings: the *batch-size* setting where $\sigma_i = \sigma = N$ and the *max-ratio* setting where $\sigma_i = N \max(1, \frac{1}{\frac{\pi_{\text{old}}(a_i|s_i)}{\pi_{\text{new}}(a_i|s_i)} + \epsilon})$. Interestingly, although using KOVA results in lower policy loss (which we try to maximize), it actually increases the policy entropy and encourages exploration, which we believe helps in gaining higher rewards during training. We can clearly see how the mean rewards increases as the policy entropy increases, for different values of η. This insight was observed in both tested Mujoco environments and in both settings of $\mathbf{P}_{\mathbf{n}_t}$.

Acknowledgement

This work was partially funded by the Israel Science Foundation under ISF grant number 1380/16.

References

[1] Anderson, B. D. and Moore, J. B. Optimal filtering. *Englewood Cliffs*, 21:22–95, 1979.

[2] Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. *arXiv preprint arXiv:1707.06887*, 2017.

[3] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[4] Dann, C., Neumann, G., and Peters, J. Policy evaluation with temporal differences: A survey and comparison. *The Journal of Machine Learning Research*, 15(1):809–883, 2014.

[5] Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y., and Zhokhov, P. Openai baselines. `https://github.com/openai/baselines`, 2017.

[6] Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[7] Engel, Y., Mannor, S., and Meir, R. Bayes meets bellman: The gaussian process approach to temporal difference learning. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 154–161, 2003.

[8] Engel, Y., Mannor, S., and Meir, R. Reinforcement learning with gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*, pp. 201–208. ACM, 2005.

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

[10] Gelb, A. *Applied optimal estimation*. MIT press, 1974.

[11] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[12] Hill, A., Raffin, A., Ernestus, M., Gleave, A., Traore, R., Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., and Wu, Y. Stable baselines. `https://github.com/hill-a/stable-baselines`, 2018.

[13] Kalman, R. E. et al. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[14] Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[16] 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.

[17] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.

[18] Ollivier, Y. et al. Online natural gradient as a kalman filter. *Electronic Journal of Statistics*, 12 (2):2930–2961, 2018.

[19] Ruder, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[20] Särkkä, S. *Bayesian filtering and smoothing*, volume 3. Cambridge University Press, 2013.

[21] Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015.

[22] Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

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

[24] Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[25] Tieleman, T. and Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2), 2012.

[26] Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033. IEEE, 2012.

[27] Van Der Merwe, R. *Sigma-point Kalman filters for probabilistic inference in dynamic state-space models*. PhD thesis, Oregon Health & Science University, 2004.

[28] Wu, Y., Mansimov, E., Grosse, R. B., Liao, S., and Ba, J. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in neural information processing systems*, pp. 5279–5288, 2017.

[29] Zeiler, M. D. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.