2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Sample-Based Information-Theoretic Stochastic Optimal Control

Rudolf Lioutikov[1] and Alexandros Paraschos[1] and Jan Peters[1,2] and Gerhard Neumann[1]

***Abstract*— Many Stochastic Optimal Control (SOC) approaches rely on samples to either obtain an estimate of the value function or a linearisation of the underlying system model. However, these approaches typically neglect the fact that the accuracy of the policy update depends on the closeness of the resulting trajectory distribution to these samples. The greedy operator does not consider such closeness constraint to the samples. Hence, the greedy operator can lead to oscillations or even instabilities in the policy updates. Such undesired behaviour is likely to result in an inferior performance of the estimated policy. We reuse inspiration from the reinforcement learning community and relax the greedy operator used in SOC with an information theoretic bound that limits the 'distance' of two subsequent trajectory distributions in a policy update. The introduced bound ensures a smooth and stable policy update. Our method is also well suited for model-based reinforcement learning, where we estimate the system dynamics model from data. As this model is likely to be inaccurate, it might be dangerous to exploit the model greedily. Instead, our bound ensures that we generate new data in the vicinity of the current data, such that we can improve our estimate of the system dynamics model. We show that our approach outperforms several state of the art approaches on challenging simulated robot control tasks.**

I. Introduction

In stochastic optimal control (SOC), see [1], [2], [3], [4], we want to compute the optimal control policy for a finite horizon of time steps. The SOC problem can be solved efficiently by dynamic programming, i.e., by iteratively computing the value function backwards in time. The value function estimates the expected future reward for a given state and time step. However, analytical solutions only exists in few cases, such as for discrete systems, and for linear systems with quadratic cost functions and Gaussian noise (LQG). For more complex systems, we typically need to rely on approximations of the value function and/or linearisations of the underlying system. All these approaches rely on data in terms of state transitions, needed either for approximating the value function or linearizing the system. However, the policy updates used in these approaches neglect the fact that the accuracy of the update depends on this state transition data — if after the update, the policy ends up in regions of the state-action space where no data has been generated, the quality of the resulting policy might be arbitarily bad. To solve this problem, we limit the distance between two subsequent trajectory distributions [5], [6]. Using this bound for SOC results in a new sample-based information theoretic SOC (ITSOC) algorithm that enjoys smooth and stable policy updates. We will further discuss the application of ITSOC for model-based reinforcement learning [7], [8] where we simultaneously estimate the system model from data. In this case, our bound provides us with the additional benefit that we avoid exploiting the possibly inaccurate model greedily. Instead, the transition data generated by the new policy will stay close to the current data, and, hence, can be used to improve the estimate of the model in this area of the state space.

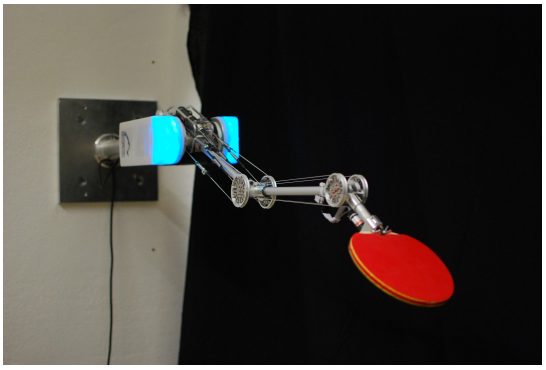


Fig. 1. For the real robot experiments we used a tendon driven arm called BioRob. His biologically inspired architecture makes controling it a very challenging task.

Approaches based on linearisation, such as differential dynamic programming [9], the iLQG [10] or the AICO [3] algorithm, linearise the system at the current estimate of the trajectory. Subsequently, the optimal controller is computed analytically on the linearised system and the resulting trajectory is used as a new linearisation point. However, as the linear dynamics model is only accurate at the point of linearisation, such an update of the controller might result in jumps in the resulting trajectory. These problems can be alleviated by using heuristics, such as regularizing the LQG solution [10] or using a learning rate on the estimated trajectory [3]. A disadvantage of linearisation approaches that for small deviations from the nominal trajectory, the linearisation of the model becomes less accurate, which degrades the robustness of the resulting policy.

Other approaches are based on approximate dynamic programming (ADP) [11], [12], [13]. These approaches iteratively collect data with the currently estimated policy. The data set is subsequently used to improve the estimate of the value function. A challenging problem when using ADP for SOC to compute the optimal action in a continuous action space, which is a very expensive operation. One way to circumvent this problem is to replace the $\max$-operation with a soft-max operator [14] that can be more efficiently implemented. Alternative approaches to continuous actions

[1]TU Darmstadt
[2]Max Plank Institute for Intelligent Systems

are using the approximate inference [4] or the path integral formulations [15] of SOC. A severe drawback is that the approximation of the value function may 'damage' the resulting trajectory distribution of the policy, causing jumps and oscillations between subsequent iterations. This effect is caused by the greedy exploitation of the value function and it is not clear how the approximation affects the resulting trajectory distribution.

In this paper, we are interested in learning locally optimal feedback controllers for certain types of movements, e.g., a stroke in a squash game. We are only interested in locally optimal policies, so we focus on the traditional setup of stochastic optimal control, i.e., we assume a squared value function and that the system dynamics can be linearised in the vicinity of the optimal solution. Our approach is model-based. The required models can be obtained from data as we only require models that are accurate in the local vicinity of the data. Hence, our algorithm can be extended to model-based reinforcement learning (RL). In this paper, we will focus on learning simple models, i.e., time-varying linear models. Other model-based RL approaches, such as the PILCO framework [8] exploit the learned model greedily, and therefore require the usage of more complex models with more accuracy, such as gaussian process (GP) models [16].

One disadvantage of ITSOC is its computational complexity. ITSOC defines a constraint optimization problem with several thousands of parameters. While using standard constrained optimizers failed due to the large scale of the problem, we present an iterative optimization procedure, that can, under soft relaxations of the constraints, be computed more efficiently, outperforming state of the art model-based RL algorithms such as PILCO [8] in terms of computation time. We compare our method to several state of the art SOC and model-based RL methods and show its superior performance. We evaluate our approach on a simulated four-link non-linear pendulum and a tendon driven robot arm.

II. Information Theoretic Stochastic Optimal Control

Many problems in stochastic optimal control (SOC) or reinforcement learning (RL) can be defined as finite horizon problems. We consider systems with continuous state and action spaces such as robotic systems. We denote the state of the robot at time step t as $\boldsymbol{x}_t$ and the control action as $\boldsymbol{u}_t$. The state transitions of the system dynamics $p_t(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t)$ that define the probability of reaching $\boldsymbol{x}_{t+1}$ after being in state $\boldsymbol{x}_t$ and taking the action $\boldsymbol{u}_t$. Due to the finite horizon, the optimal control policy of this problem is time dependent. The goal is to find such an optimal time-dependent control policy $\pi_k^*(\boldsymbol{u}|\boldsymbol{x})$ which maximizes the accumulative reward for a given time horizon H

$$J^\pi = \mathbb{E}_{p^\pi(\boldsymbol{\tau})}\left[\sum_{t=1}^{H-1} r_t(\boldsymbol{x}_t, \boldsymbol{u}_t) + r_H(\boldsymbol{x}_H)\right], \quad (1)$$

where $\boldsymbol{\tau} = (\boldsymbol{x}_{1:H}, \boldsymbol{u}_{1:H-1})$ denotes a trajectory and $p^\pi(\boldsymbol{\tau})$ the trajectory distribution that is generated by policy π. The function $r_t(\boldsymbol{x}, \boldsymbol{u})$ denotes the immediate reward for time step t and $r_H(\boldsymbol{x}_H)$ the final reward for reaching $\boldsymbol{x}_H$ at the horizon H. The expectation of the reward is performed with respect to the trajectory distribution

$$p^\pi(\boldsymbol{\tau}) = p_1(\boldsymbol{x}_1)\prod_{t=1}^{H-1} p_t(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t)\pi_t(\boldsymbol{u}_t|\boldsymbol{x}_t),$$

of the policy π, where $p_1(\boldsymbol{x}_1)$ is the initial state distribution.

A. Formulation as Constrained Optimization Problem

We will now formulate SOC as a constraint optimization problem. Such a formulation allows us to add additional constraints such as information theoretic bounds on the trajectory distribution. By introducing the state-visit distribution $\mu_t^\pi(\boldsymbol{x}_t)$ for time step t, we can restate the *objective function* as

$$J^{\pi,\mu} = \sum_{t=1}^{H-1}\iint \mu_t^\pi(\boldsymbol{x})\pi_t(\boldsymbol{u}|\boldsymbol{x})\, r_t(\boldsymbol{x}, \boldsymbol{u})\, d\boldsymbol{x}d\boldsymbol{u} + \int \mu_H^\pi(\boldsymbol{x}) r_H(\boldsymbol{x})\, d\boldsymbol{x}, \quad (2)$$

where the maximization is performed over $\pi_t(\boldsymbol{u}|\boldsymbol{x})$ and $\mu_t^\pi(\boldsymbol{x})$. In order to keep the notation uncluttered we will drop the index t of $\boldsymbol{x}$ and $\boldsymbol{u}$ wherever the context is clear. By adding several constraints, Equation (2) is equivalent to the original SOC problem.

First, we have the *normalization constraints* which ensure that the policy $\pi_t(\boldsymbol{u}|\boldsymbol{x})$ and the state distribution $\mu_t^\pi(\boldsymbol{x})$ jointly define a distribution for each time step,

$$\iint \mu_t^\pi(\boldsymbol{x})\pi_t(\boldsymbol{u}|\boldsymbol{x})d\boldsymbol{x}d\boldsymbol{u} = 1, \quad \int \mu_H^\pi(\boldsymbol{x})d\boldsymbol{x} = 1. \quad (3)$$

Moreover, the state distributions $\mu_t^\pi(\boldsymbol{x})$ must respect the policy and the system dynamics, i.e., we can not choose $\mu_t^\pi(\boldsymbol{x})$ arbitrarily but the state distributions need to comply with the constraints

$$\mu_{t+1}^\pi(\boldsymbol{x}') = \iint \mu_t^\pi(\boldsymbol{x})\pi_t(\boldsymbol{u}|\boldsymbol{x})\, p_t(\boldsymbol{x}'|\boldsymbol{x}, \boldsymbol{u})d\boldsymbol{x}d\boldsymbol{u}$$

for $1 \leq t < H$ and the initial state distribution constraint $\mu_1^\pi(\boldsymbol{x}) = p_1(\boldsymbol{x})$. These constraints have to be satisfied for all states $\boldsymbol{x}$. Thus, in continuous state and action spaces, we would end up with an infinite amount of constraints. Therefore, we match the distributions by matching only the feature averages of two subsequent state distributions. This approach leads us to the following *feature constraints*

$$\int \mu_{t+1}^\pi(\boldsymbol{x}')\boldsymbol{\phi}(\boldsymbol{x}')\, d\boldsymbol{x}' = \iiint \mu_t^\pi(\boldsymbol{x})\pi_t(\boldsymbol{u}|\boldsymbol{x})\, p_t(\boldsymbol{x}'|\boldsymbol{x}, \boldsymbol{u})\, \boldsymbol{\phi}(\boldsymbol{x}')\, d\boldsymbol{x}d\boldsymbol{u}d\boldsymbol{x}', \quad (4)$$

for all $1 \leq t < H$ and the initial state constraint

$$\int \mu_1^\pi(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x})d\boldsymbol{x} = \hat{\boldsymbol{\phi}}_1, \quad (5)$$

where $\hat{\boldsymbol{\phi}}_1$ are the observed averaged features of the initial state. The original SOC problem is now represented by the

objective function from Eq. (2), the *normalization constraints* from Eq. (3) and the *feature constraints* from Eq. (4). The formulation as a constraint optimization problem also offers a new type of approximation for SOC. Instead of approximating the optimal value function as done by ADP approaches, in this formulation, we approximate the constraints on the state distributions $\mu_t^\pi(\boldsymbol{x})$.

B. Relaxing the Greedy Operator by Information-Theoretic Bounds

The advantage of the constrained optimization formulation of SOC, is that we are now able to introduce new constraints to the problem. Inspired by the policy search community [6], [5], we add information theoretic constraints to bound the 'distance' between the state action distribution $p_t(\boldsymbol{x}, \boldsymbol{u}) = \mu_t^\pi(\boldsymbol{x})\pi_t(\boldsymbol{u}|\boldsymbol{x})$ of the new policy and the old policy $q_t(\boldsymbol{x}, \boldsymbol{u})$ for each time step t. As in [6], [5], we will use the Kullback-Leibler divergence (KL)

$$\mathrm{KL}(p_t(\boldsymbol{x}, \boldsymbol{u})||q_t(\boldsymbol{x}, \boldsymbol{u})) = \iint \mu_t^\pi(\boldsymbol{x})\pi_t(\boldsymbol{u}|\boldsymbol{x}) \log \frac{\mu_t^\pi(\boldsymbol{x})\pi_t(\boldsymbol{u}|\boldsymbol{x})}{q_t(\boldsymbol{x}, \boldsymbol{u})} d\boldsymbol{x}\, d\boldsymbol{u} \quad (6)$$

as distance measure between the two distributions. Due to this constraint, the agent only explores locally and we avoid jumps in two subsequent state distributions $\mu_t^\pi(\boldsymbol{x})$ and $q_t(\boldsymbol{x})$. The aforementioned *information-theoretic constraints* can be formalized as

$$\begin{aligned} \mathrm{KL}(p_t(\boldsymbol{x}, \boldsymbol{u})||q_t(\boldsymbol{x}, \boldsymbol{u})) \leq \epsilon \text{ for } 1 \leq t < H, \\ \mathrm{KL}(p_H(\boldsymbol{x})||q_H(\boldsymbol{x})) \leq \epsilon. \end{aligned} \quad (7)$$

Combined with the reformulated SOC from the previous subsection, these constraints result in the *Information-Theoretic Stochastic Optimal Control* (ITSOC) algorithm. Note that all of those constraints are always satisfiable, since $q_t(\boldsymbol{x}, \boldsymbol{u})$ has been generated by following the system dynamics, and therefore, a trivial sub-optimal solution is given by $\pi_t(\boldsymbol{u}|\boldsymbol{x})\,\mu_t^\pi(\boldsymbol{x}) = q_t(\boldsymbol{x}, \boldsymbol{u})$.

C. Estimating the New Policy

In practice, we will use samples $\boldsymbol{x}_{t,i}, \boldsymbol{u}_{t,i},\ i = 1 \ldots N$, to approximate the integrals in the objectives as well as in the constraints of ITSOC. However, solving directly the primal form of ITSOC is undesireable as the problem contains non-linear constraints and the number of parameters is given by the amount of samples, which is typically a high number. However, we can solve the dual formulation of this optimization problem [6]. The dual is obtained by apply the method of Lagrangian multipliers. From the Lagrangian of the optimization problem, we can obtain a closed form solution for the new state action distribution $\pi_t(\boldsymbol{u}|\boldsymbol{x})\,\mu_t^\pi(\boldsymbol{x})$ that is given by

$$\begin{aligned} \pi_t(\boldsymbol{u}|\boldsymbol{x})\,\mu_t^\pi(\boldsymbol{x}) &\propto q_t(\boldsymbol{x}, \boldsymbol{u}) \exp\left(\frac{A_t(\boldsymbol{x}, \boldsymbol{u})}{\eta_t}\right), \\ \mu_H^\pi(\boldsymbol{x}) &\propto q_H(\boldsymbol{x}) \exp\left(\frac{A_H(\boldsymbol{x})}{\eta_H}\right). \end{aligned} \quad (8)$$

The functions $A_t(\boldsymbol{x}, \boldsymbol{u})$ and $A_H(\boldsymbol{x})$ are denoted as advantage function and they are given by

$$\begin{aligned} A_t(\boldsymbol{x}, \boldsymbol{u}) &= r_t(\boldsymbol{x}, \boldsymbol{u}) + \mathbb{E}_{p_t(\boldsymbol{x}'|\boldsymbol{x}, \boldsymbol{u})}[v_{t+1}(\boldsymbol{x}')] - v_t(\boldsymbol{x}) \\ A_H(\boldsymbol{x}) &= r_H(\boldsymbol{x}) - v_H(\boldsymbol{x}), \end{aligned} \quad (9)$$

where $v_t(\boldsymbol{x}) = \boldsymbol{\theta}_k^T \boldsymbol{\phi}(\boldsymbol{x})$ is denoted as a value function. The parameters $\boldsymbol{\theta}_{1:H}$ and $\eta_{1:H}$ are the Lagrangian multipliers of the *feature constraints* and the KL-constraint respectively. Setting Equation (8) back into the Lagrangian results in the dual function $g(\boldsymbol{\theta}_{1:H}, \eta_{1:H})$ [5] which is omitted here for space reasons.

As the original optimization problem was maximized, we need to minimize the dual function. Moreover, each inequality constraint in the original problem results in an inequality constraint in the dual [17] that is given by $\eta_k > 0$. Hence, solving the primal is equivalent to solving

$$\begin{aligned} [\boldsymbol{\theta}_{1:H}^*, \eta_{1:H}^*] = &\mathrm{argmin}_{\boldsymbol{\theta}_{1:H}, \eta_{1:H}} g(\boldsymbol{\theta}_{1:H}, \eta_{1:H}), \\ &\text{s.t.: } \eta_k > 0, \forall k \end{aligned} \quad (10)$$

By using the dual form of ITSOC the amount of optimization parameters is now reduced to $N_{\text{dual}} = H(\#\text{features} + 1)$ and we eliminated all non-linear constraints.
The dual function is convex in $\boldsymbol{\theta}_{1:H}$ and $\eta_{1:H}^*$ and can be easily optimized for small scale problems. It can also be efficiently approximated by samples that have been generated by following the old policy. By optimizing the dual function, we obtain the values for the Lagrangian multipliers $[\boldsymbol{\theta}_{1:H}, \eta_{1:H}]$ that again determine the distribution $\mu_t^\pi(\boldsymbol{x})\pi_t(\boldsymbol{u}|\boldsymbol{x})$. Given our samples, we can evaluate the probability

$$p_{t,i} \propto \exp\left(\frac{A_t(\boldsymbol{x}_{t,i}, \boldsymbol{u}_{t,i})}{\eta_t}\right) \quad (11)$$

of each sample. However, in order to create new roll-outs with the current policy $\pi_t(\boldsymbol{u}|\boldsymbol{x})$ we need a parametric model $\tilde{\pi}_t(\boldsymbol{u}|\boldsymbol{x}, \boldsymbol{\omega}_t)$, where $\boldsymbol{\omega}_t$ denote the parameters of the policy. We obtain $\boldsymbol{\omega}_t$ by a weighted maximum likelihood estimate where the probabilities $p_{t,i}$ serve as weight of the data points.

III. Learning Feedback Controllers with ITSOC

A typical approach in SOC is to approximate the optimal policy by a time-dependent linear feedback controller as well as to approximate the value function by a quadratic function [3]. Using such approximation, locally optimal controllers can be obtained that work well in practice. The main contribution of this paper is to adapt the general ITSOC algorithm to this specific SOC formulation and to present efficient ways of solving the optimization problem.

A. Representation of the Policy

As in many traditionally SOC formulations, we use a linear representation of the policy $\pi_t(\boldsymbol{u}|\boldsymbol{x})$ at each time step t

$$\pi_t(\boldsymbol{u}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{u}|\boldsymbol{s}_t + \boldsymbol{S}_t\boldsymbol{x}, \boldsymbol{\Sigma}_t)\ , \quad (12)$$

where the parameters $\boldsymbol{s}_t, \boldsymbol{S}_t$ and $\boldsymbol{\Sigma}_t$ are obtained by a weighted maximum likelihood (ML) estimate on the current

set of samples. For the used linear models, the weighted ML estimate is given by

$$\begin{bmatrix} \boldsymbol{s}_t^T \\ \boldsymbol{S}_t^T \end{bmatrix} = (\boldsymbol{X}_t^T \boldsymbol{D}_t \boldsymbol{X}_t)^{-1} \boldsymbol{X}_t^T \boldsymbol{D}_t \boldsymbol{U}_t \tag{13}$$

and

$$\boldsymbol{\Sigma}_t = \frac{\sum_i p_{t,i} (\boldsymbol{\mu}_{t,i} - \boldsymbol{u}_{t,i})(\boldsymbol{\mu}_{t,i} - \boldsymbol{u}_{t,i})^T}{\sum_i p_{t,i}}, \tag{14}$$

where $\boldsymbol{X}_t$ is the input matrix containing the states $\boldsymbol{x}_{t,i}$ for time step t[1], $\boldsymbol{D}_t = \text{diag}([p_{t,i}]_{i=1\ldots N})$ is the weighting matrix and $\boldsymbol{U}_t$ contains the actions $\boldsymbol{u}_{t,i}$ in its rows. The predicted mean for the ith sample is given by $\boldsymbol{\mu}_{t,i} = \boldsymbol{s}_t + \boldsymbol{S}_t \boldsymbol{x}_{t,i}$.

B. Estimating the Expected Feature Vector

The information-theoretic formulation requires estimating $\mathbb{E}_{p_t(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{u})}[v_{t+1}(\boldsymbol{x}')]$ for each sample, i.e., we require a model. In this paper, we assume that the system dynamics are given in form of a time varying linear system

$$p_t(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{u}) = \mathcal{N}(\boldsymbol{x}'|\boldsymbol{a}_t + \boldsymbol{A}_t\boldsymbol{x} + \boldsymbol{B}_t\boldsymbol{u}, \boldsymbol{C}_t). \tag{15}$$

Moreover, we choose a quadratic representation of the state feature vector, such that v_t can be written as

$$v_t(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{V}_t \boldsymbol{x} + \boldsymbol{v}_t^T \boldsymbol{x},$$

with $\boldsymbol{\theta}_t = \{\boldsymbol{V}_t, \boldsymbol{v}_t\}$. Under these assumptions, the expectation of the value of the next state can be performed analytically by

$$\begin{aligned} &\mathbb{E}_{p_t(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{u})}[v_{t+1}(\boldsymbol{x}')] \\ &\quad = \int p_t(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{u}) \left(\boldsymbol{x}'^T \boldsymbol{V}_t \boldsymbol{x}' + \boldsymbol{v}_t^T \boldsymbol{x}'\right) d\boldsymbol{x}' \\ &\quad = \boldsymbol{\mu}_{t,\boldsymbol{x}\boldsymbol{u}}^T \boldsymbol{V}_t \boldsymbol{\mu}_{t,\boldsymbol{x}\boldsymbol{u}} + \text{trace}(\boldsymbol{C}_t \boldsymbol{V}_t) + \boldsymbol{v}_t^T \boldsymbol{\mu}_{t,\boldsymbol{x}\boldsymbol{u}} \end{aligned} \tag{16}$$

with $\boldsymbol{\mu}_{t,\boldsymbol{x}\boldsymbol{u}} = \boldsymbol{a}_t + \boldsymbol{A}_t\boldsymbol{x} + \boldsymbol{B}_t\boldsymbol{u}$. Equation (16) can be rewritten in the feature vector representation $\mathbb{E}_{p_t(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{u})}[\boldsymbol{\phi}(\boldsymbol{x}')^T]\boldsymbol{\theta}_{t+1}$ which we omit due to space constraints.

C. Learning Local Models for Reinforcement Learning

Our approach is easily extensible to model-based reinforcement learning by learning the system dynamics model $p_t(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{u})$ from our generated samples. We will use again simple linear Gaussian models for each time step, i.e., and obtain the parameters $\boldsymbol{a}_t, \boldsymbol{A}_t, \boldsymbol{B}_t$ and $\boldsymbol{C}_t$ of the models from our sampled data points $(\boldsymbol{x}_{t,i}, \boldsymbol{u}_{t,i})$ by a maximum likelihood estimate. In the RL formulation of our algorithm, we assume that the reward function r_t is known as prior knowledge and only the system dynamics need to be learned. In difference to other state of the art model-based policy search methods [8], we focus on learning simple models as they are easy to learn. The information-theoretic bound ensures that we generate more data in the neighbourhood of the already existing data, and, hence, we are likely to improve the estimate of our model. Competing methods such as PILCO [8] would greedily exploit the model, and hence, risk to get unstable policy updates due to inaccurate model estimates.

[1]A constant of 1 was prepended to the states $\boldsymbol{x}$ to account for the offset term of the linear policy.

D. Optimizing the Dual

A major challenge in applying ITSOC is the minimization of the dual function. While in [5] a strongly related formulation has been used to optimize parameters of high-level decisions where an episode consisted only of a small number of decisions, we want to cope with large time horizons of $H = 50$ to $H = 100$. With the quadratic feature representation, we easily reach 40 to 60 parameters per time step, resulting in several thousands of parameters for the optimization. Therefore an efficient and reliable optimization is crucial. In this subsection we will elaborate on different techniques for the optimization. In all cases we provided the algorithms with the analytic gradients and Hessians.

a) Constrained optimization: An intuitive approach is to use a constrained optimizer, such as a trust-region-reflective algorithm [17], [18]. Unfortunately the optimizer did run in numerical instabilities and did not find reliable solutions in a suitable time.

b) Unconstrained optimization: Another approach is to use the exp-trick [19] to transform the constrained problem into an unconstrained one. We now optimize for a non-negative substitute function $\zeta = \log(\eta)$. As optimization method we choose a large-scale algorithm [18]. Again we did not achieve satisfactory results. The cause might be the introduction of additional non-linearities in combination with numerical instability.

c) Iterative optimization: An inspection of the dual reveals, that the η and $\boldsymbol{\theta}$ can be optimized separately using a coordinate descent like method. First, we fix the $\boldsymbol{\theta}$ parameters and optimize for each η individually. Subsequently, we optimize for all $\boldsymbol{\theta}$ vectors, which is an unconstrained optimization. We iterate over these two steps until a solution is found. The $\boldsymbol{\theta}$-optimization was performed by a large-scale method [18] and each of the η-optimizations by a trust-region-reflective method [18]. This approach reliably found good solutions in a suitable amount of time. It can take a considerable amount of time until the dual function is fully optimized. But most of the constraints do not need to be fulfilled perfectly in order to achieve a good performance. Therefore, we relax the KL and feature constraints and stop the optimization if both are approximately satisfied. For the KL-constraints, we compute at each iteration of our optimization the maximum derivation of the KL's for the single time steps to the desired KL ϵ and consider the KL constraints as satisfied if this deviation is smaller than $\epsilon \cdot d_\epsilon$. Simillarly we compute the $f_{\max}$ maximum normalized deviation of the average feature vectors and conider the constrained fullfilled if a threshold is met.

E. Algorithm

We create K real roll-outs using the currently estimated policy $\tilde{\pi}_t(\boldsymbol{u}|\boldsymbol{x},\boldsymbol{\omega}_t)$ and use these samples to replace the distribution $q_t(\boldsymbol{x},\boldsymbol{u})$ in the dual function of the optimization problem. To increase our data-efficiency, we will define $q_t(\boldsymbol{x},\boldsymbol{u})$ not just as the samples from the previous iteration

Input: KL-bound ϵ, number of iterations K, samples N and virtual samples M, L last iterations to reuse

Initialize $\tilde{\pi}_t^0$ using Gaussians with zero mean and high variance.

for $h = 1$ to L ... # iterations

Collect data on the real system following $\tilde{\pi}_t^{h-1}$:

$$\mathcal{D}_k = \{\boldsymbol{x}_{t,i}, \boldsymbol{u}_{t,i}\}_{i=1\ldots N, t=1\ldots H}$$

Re-use last L **iterations**: $\mathcal{D} = \{\mathcal{D}_l\}_{l=\max(1,h-L)\ldots k}$

Estimate time-varying linear models using $\mathcal{D}$

Collect data on the learned system following $\tilde{\pi}_t^{h-1}$:

$$\tilde{\mathcal{D}}_h = \{\tilde{\boldsymbol{x}}_{t,i}, \tilde{\boldsymbol{u}}_{t,i}\}_{i=1\ldots M, t=1\ldots H}$$

Minimize dual function on $\tilde{\mathcal{D}}_k$:

$$[\eta_{1:H}, \boldsymbol{\theta}_{1:H}] = \text{argmin}_{\eta'_{1:H}, \boldsymbol{\theta}'_{1:H}} g\left(\eta'_{1:H}, \boldsymbol{\theta}'_{1:H}; \tilde{\mathcal{D}}_h\right)$$

Estimate new policy $\tilde{\pi}_t^h$ **for each** t:

Compute Weighting:

$$p_{t,i} = \exp\left(\frac{A_t(\tilde{\boldsymbol{x}}_{t,i}, \tilde{\boldsymbol{u}}_{t,i})}{\eta_t}\right), \text{ for all } i \text{ and } k.$$

Compute policy parameters (weighted ML estimate), Equation (13) *and* (14).

Output: Policies $\tilde{\pi}_t^H(\boldsymbol{u}|\boldsymbol{x})$ for all $t = 1, \ldots, T$

TABLE I

THE INFORMATION-THEORETIC SOC ALGORITHM.

but from the last L iterations. Hence, q_t is defined as a mixture of the last L trajectory distributions. We use the samples from q_t solely to learn the local models. These models are later used to generate a large number M of virtual roll-outs. These virtual samples are then used in the dual function to determine the new policy. The resulting algorithm is summarized in Algorithm I. At each iteration, the policy is moving a small, controlled step towards the (locally) optimal policy. We start the algorithm with a rather large variance for the initial policy π_t^0. Subsequently, the exploration is automatically decreased in each iteration by the information theoretic SOC algorithm and will, finally, collapse to a deterministic policy.

IV. EXPERIMENTS

We evaluate ITSOC on a simulated 4-link non-linear planar arm in two different scenarios and on the BioRob[20], a five link tendon driven robot arm where we only control 2 links. We compare ITSOC against: a variant of advantage weighted regression (AWR) [14], where we also use linear policies and a quadratic value function, the AICO algorithm [3], that is a state of the art linearisation-based algorithm and a variant of the model-based policy search algorithm PILCO [8], where we also use learned time dependent linear models instead of Gaussian Processes. Instead of a gradient-based optimizer, we use AICO as optimizer for PILCO. We will denote this algorithm *linear PILCO*. While AICO uses the real system dynamics, linear PILCO uses AICO with the learned system dynamics. We evaluate the robustness of these algorithms to system noise and the accuracy of the learned models.

A. Illustration on a Two-Link Reaching Task

As a first illustration, we evaluated our approach on a simulated two link robot arm which had to reach two via-points in task space. We use 50 time steps and a time interval of dt $= 0.066$s. The reward function is given as squared punishment term for the torques at each time step and a via-point reward $r_{\boldsymbol{v}}(\boldsymbol{x})$ at time steps $t_1 = 25$ and $t_2 = 50$. The via-point reward $r_v(\boldsymbol{x})$ is proportional to the negative squared distance in task space $r_{\boldsymbol{v}}(\boldsymbol{x}) = -(\boldsymbol{y}-\boldsymbol{v})^T \boldsymbol{H}(\boldsymbol{y}-\boldsymbol{v})$, where $\boldsymbol{H}$ is set to 10^4 for all state dimensions which denote a position. The state vector $\boldsymbol{x}$ is four dimensional containing all joint positions and velocities of the robot. To illustrate the behavior of different algorithms, we show subsequent trajectory distributions during the iterations of the algorithms in Figure 2. We show the distributions of ITSOC, AWR and linear PILCO. We can see that the distributions change smoothly for ITSOC allowing the algorithm to efficiently find an optimal solution. For AWR the distributions jump and it quickly converges to a deterministic, but sub-optimal policy. Linear PILCO jumps early on to good solutions, however, the greedy exploitation of the models can already cause jumps in the trajectory distribution for small inaccuracies.

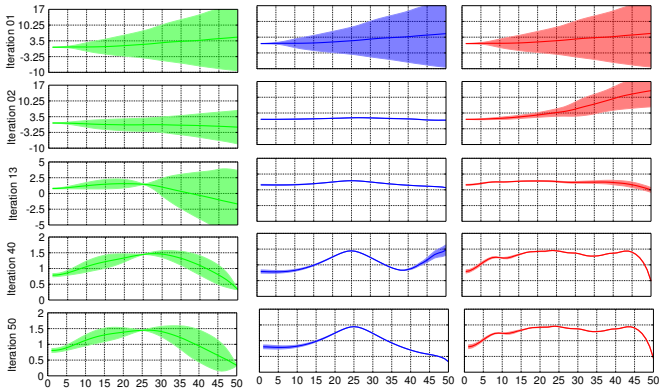


Fig. 2. The plot shows the trajectory distributions for the last joint q_2 of a two link planar arm with ITSOC (left), linear PILCO (middle) and AWR (right). The x-axis depicts the time steps while in the rows we can see the distribution for subsequent iterations. While the AWR approach shows a jumping behavior and a resulting limited learning progress, the information-theoretic approach smoothly transforms the initial exploratory distribution into a goal-directed movement. PILCO quickly jumps to a good solution, but fails to further improve the policy. The average reward of ITSOC is -11.2, while for AWR it is -52.4 and PILCO oscillates between -3 and -2000.

B. Four-Link Reaching Task

In this task, a four link robot has to reach two via-points in task space. The same setup as described for the previous experiment has been used. In addition to the control noise used in some experiments, we always use a Gaussian distribution for the intial state distribution $p_1(\boldsymbol{x})$. Hence, we do not want to estimate a single nominal trajectory but a controller which works well in a broader area of the initial state. The state vector $\boldsymbol{x}$ of the robot is 8 dimensional containing all joint positions and velocities. An $d = 8$ dimensional state vector results in a 44 dimensional feature vector $\boldsymbol{\phi}(\boldsymbol{x})$, consisting of 8 linear and $d(d+1)/2 = 36$ squared terms. Hence, we had to optimize the dual function for $(44+1)50 = 2250$ parameters.

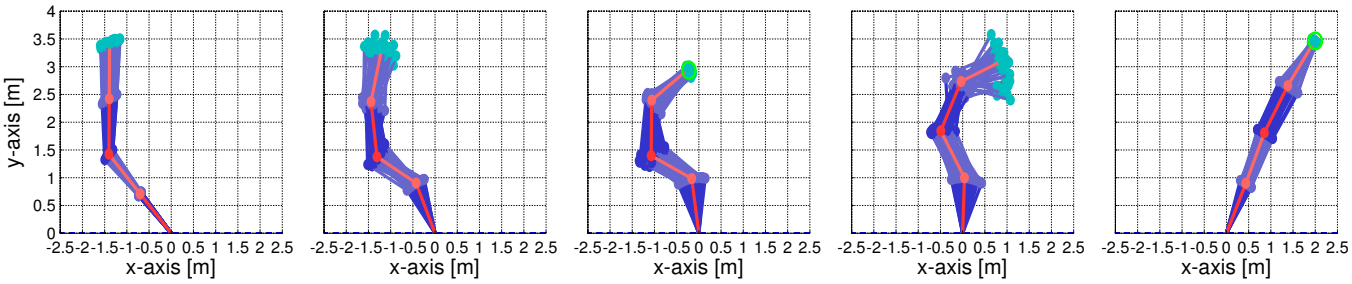
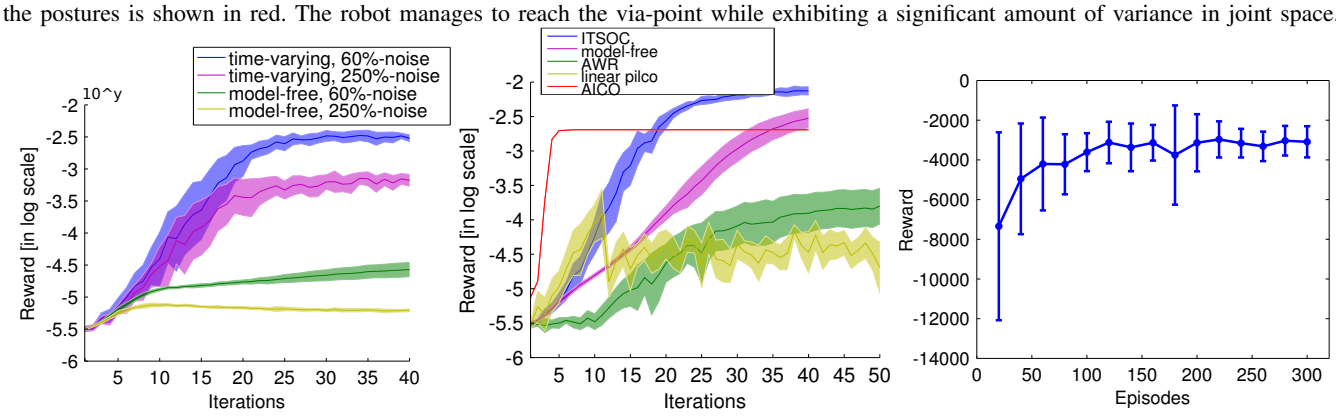


Fig. 3. Illustration of the resulting postures for the via-point task in task space. The non-linear planar arm has to reach the via-points at $t = 25$ and $t = 50$ illustrated in green. The plot shows for each time step sample from the resulting distribution of postures. The mean of the postures is shown in red. The robot manages to reach the via-point while exhibiting a significant amount of variance in joint space.



(a) Noise (b) Comparison (c) BioRob

Fig. 4. (a) Comparison of information-theoretic (IT) SOC with learned models and without learned models where we use a single sample estimate for the expectation of the next features, which results in a bias in our optimization. Without models ITSOC is not able to learn satisfatory policies, while the resulting policies with models can be considered good solutions, as can be seen in Figure 3. The evaluations are done for different noise levels of additive control noise. (b) In addition we compare to advantage weighted regression (AWR) and AICO. Model-based ITSOC clearly shows the best performance while the biased version of ITSOC quickly degrades with an increasing noise level. Note that all plots are in log-scale for the y-axis. (c) Learning curve for the Biorob.

a) Comparison of Algorithms: We first compare our approach to the AWR approach, the linear PILCO and the AICO approach on this scenario. The results are shown in Figure 4(b). The ITSOC approach significantly outperforms all other methods in terms of learning speed and/or quality of the final policy. Linear PILCO quickly jumps to a good solution, but fails to find a solution of the same quality due to the lack of exploration. A similar behavior could be observed for the AICO algorithm that uses the known system dynamics.

b) Robustness to Noise: We also evaluate our algorithm with learned time-varying models against the version without the usage of models, that have been used in [5] for high-level policy search, with different noise levels. For the model-free approach, we do not estimate the expected next features but use just a single-sample estimate. Hence, the model free optimization is biased. We use additive control noise with a standard deviation of 60% and 250% of the maximum torques which can be applied by the robot. We use $N = 500$ samples per iteration and do not keep samples. The results are shown in Figure 4(a). The model-based ITSOC method could estimate a high quality controller even with the considerably increased noise level. In contrast, the performance of the model-free method quickly degrades due to its bias while model-based. We also illustrate the resulting postures for different time points in Figure 3 for the setting with 60% noise. The robot manages to reach the via-points (illustrated in green) in task space while it still exhibits a large variance in joint space. In between the via-points, the variance in task space also grows.

C. Robot Squash

In this task we extend the 4-link arm scenario to the task of playing robot squash. We add the position and velocity of a ball into our state space. The ball starts with random initial x-position and velocity, however, the initial y-position and velocity are fixed. The initial y-velocity always points away from the robot. The ball moves with constant velocity, however, at time step 25 it returns to the robot. At the bounce, the x-velocity is perturbed by a significant amount of noise, changing the incoming position of the ball. The agent has to hit the ball at time step 50. To do so, it has to be within a vicinity of 20cm from the ball position at $t = 50$. If it manages to do so, it gets a positive reward proportional to the velocity of the end-effector in the y-direction. Otherwise, the reward is proportional to the negative squared distance to the ball location. As there is no variance in the y-positions, we add only the x-position and the x-velocities in our state space, resulting in a 10 dimensional state vector and 65 dimensional feature vector which resulted in 3300 parameters for the

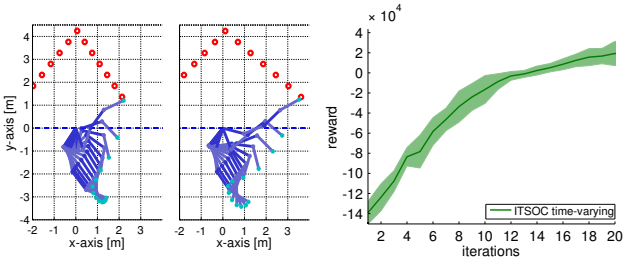


Fig. 5. (Left) Illustration of the robot playing squash for two different configurations of the incoming ball. The ball has been simulated with constant velocity but bounces off the wall in a stochastic way. The robot can react to this perturbation and reliably hits the ball. (Right) Learning curve in the robot squash task for the information theoretic SOC method.

dual function. We used 10 samples per iteration and use the last 100 samples to learn time-varying models. The robot learned to reliably hit the incoming ball at different positions with a high velocity in the y-direction. An illustration of the resulting postures for two different hitting movement is shown in Figure (5).

D. Controller Learning on a Tendon-Driven Robot Arm

In this experiment, we want to learn a controller for the BioRob robot, depicted in figure 1. The BioRob is a biologically motivated, lightweight, and highly compliant robot arm that is driven by tendons and springs. While its mechanical design with springs, enables the execution of fast movements, controlling it's position accurately during the movement is difficult, as the robot does not follow the rigid body dynamics. We want to learn a fast upwards movement with the shoulder and the elbow joints. Both joints should reach a desired via-point at $t = 30$ where we reward high velocities at the via point. At $t = 60$, the robot should again be moving still. The state of the spring acts as unobserved state, however, it can be inferred by using the difference between the joint and the motor encoders. We use 4 state dimensions per joint, the joint position, joint velocity and the motor position and velocity. As we only control two motors, the control action is 2 dimensional. We initialized the policy by using 10 sub-optimal demonstrations and we collected 20 episodes per iteration.

V. Conclusion and Future Work

In this paper, we presented a novel information theoretic stochastic optimal control algorithm. The key idea of our approach is that we want to stay close to the generated data such that we can ensure a stable learning progress. To our knowledge, this notion of closeness to the data is missing in all other stochastic optimal control algorithms. We believe it is a key ingredient for safe approximation of the value function.

The information theoretic formulation provides several advantages over traditional approaches. On a finite set of samples, we can get a closed-form solution for the estimated policy. We also can control the exploration of the policy in a principled manner without heuristics or fine-tuning. Moreover, we can use the roll-outs to learn simple local models which allow us to also use our algorithm for reinforcement learning. For future work, we will investigate dimensionality reduction techniques to reduce the dimensionality of the feature space. We will also investigate the use of different types of parametrized policies.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7-ICT-2013-10) under grant agreement 610878 (3rdHand), and (FP7-ICT-2009-6) under grant agreement 270327 (ComPLACS).

References

[1] R. Stengel, *Stochastic Optimal Control: Theory and Application*. John Wiley & Sons, Inc., 1986.
[2] H. Kappen, "An Introduction to Stochastic Control Theory, Path Integrals and Reinforcement Learning," in *Cooperative Behavior in Neural Systems*, vol. 887, 2007.
[3] M. Toussaint, "Robot Trajectory Optimization using Approximate Inference," in *26th International Conference on Machine Learning*, ser. (ICML), 2009.
[4] K. Rawlik, M. Toussaint, and S. Vijayakumar, "On Stochastic Optimal Control and Reinforcement Learning by Approximate Inference," in *Proceedings of Robotics: Science and Systems*, 2012.
[5] C. Daniel, G. Neumann, and J. Peters, "Learning Sequential Motor Tasks," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, submitted.
[6] J. Peters, K. Mülling, and Y. Altun, "Relative Entropy Policy Search," in *Proceedings of the 24th National Conference on Artificial Intelligence (AAAI)*, 2010.
[7] J. G. Schneider, "Exploiting Model Uncertainty Estimates for Safe Dynamic Control Learning," in *NIPS*, 1997, pp. 1047–1053.
[8] M. Deisenroth and C. Rasmussen, "PILCO: A Model-Based and Data-Efficient Approach to Policy Search," in *28th International Conference on Machine Learning (ICML)*, 2011.
[9] J. Morimoto and C.Atkeson, "Minimax differential dynamic programming: An application to robust bipedwalking," *Neural Information Processing Systems 2002*, 2002.
[10] E. Todorov and W. L., "A Generalized Iterative LQG Method for Locally-Optimal Feedback Control of Constrained Nonlinear Stochastic Systems," in *24th American Control Conference (ACC)*, 2005.
[11] G. J. Gordon, "Stable Function Approximation in Dynamic Programming," in *12th International Conference on Machine Learning (ICML)*, 1995.
[12] M. Azar, V. Gómez, and H. J. Kappen, "Dynamic Policy Programming," *Journal of Machine Learning Research*, vol. 13, no. Nov, pp. 3207–3245, 2012.
[13] M. G. Lagoudakis and R. Parr, "Least-Squares Policy Iteration," *Journal of Machine Learning Research*, vol. 4, pp. 1107–1149, December 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=945365.964290
[14] G. Neumann and J. Peters, "Fitted Q-Iteration by Advantage Weighted Regression," in *Neural Information Processing Systems (NIPS)*, 2009.
[15] E. Theodorou, J. Buchli, and S. Schaal, "Reinforcement Learning of Motor Skills in High Dimensions: a Path Integral Approach," in *2010 IEEE International Conference on Robotics and Automation (ICRA)*, 2010.
[16] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.
[17] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.
[18] T. Mathworks. Matlab Optimization Toolbox User's Guide.
[19] F. Sisser, "Elimination of bounds in Optimization Problems by Transforming Variables," *Mathematical Programming*, vol. 20, no. 1, pp. 110–121, 1981.
[20] T. Lens, "Physical human-robot interaction with a lightweight, elastic tendon driven robotic arm: Modeling, control, and safety analysis," Ph.D. dissertation, TU Darmstadt, Department of Computer Science, July 4 2012.