2021 60th IEEE Conference on Decision and Control (CDC)
December 13-15, 2021. Austin, Texas

Maximum-Entropy Progressive State Aggregation for Reinforcement Learning

Christos N. Mavridis, Nilesh Suriyarachchi, and John S. Baras

2021 60th IEEE Conference on Decision and Control (CDC) | 978-1-6654-3659-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/CDC45484.2021.9682927

***Abstract*— We propose a reinforcement learning algorithm based on an adaptive state aggregation scheme defined by a progressively growing set of codevectors placed in the joint state-action space according to a maximum-entropy vector quantization scheme. The proposed algorithm constitutes a two-timescale stochastic approximation algorithm with: (a) a fast component that executes a temporal-difference learning algorithm, and (b) a slow component, based on an online deterministic annealing algorithm, that adaptively partitions the state-action space according to a dissimilarity measure that belongs to the family of Bregman divergences. The proposed online deterministic annealing algorithm is a competitive-learning neural network that shows robustness with respect to the initial conditions, requires minimal hyper-parameter tuning, and provides online control over the performance-complexity trade-off. We study the convergence properties of the proposed methodology and quantify its performance in simulated experiments. Finally, we show that the generated codevectors can be used as training samples for sparse and progressively more accurate Gaussian process regression.**

I. INTRODUCTION

Reinforcement learning algorithms are being extensively studied, not only due to their effectiveness in numerous applications [1], [2], but also due to their promise to solve difficult optimal control problems in an online and data-driven fashion.

Temporal-difference learning methods dominate current reinforcement learning algorithms due to their data efficiency, and, when applied to a Markov decision process with finite state and action spaces, they have been well understood and quite successful [3]. However, because they make use of look-up tables, they can hardly be used with large or infinite state/action spaces due to the exponential increase of the number of states with respect to the dimensionality of the space. For this reason, parametric models have been widely used in reinforcement learning and value function approximation [2], with linear combinations of fixed basis functions, such as artificial neural networks, being the staple [4]. While reinforcement learning algorithms based on parametric models can deal with the dimensionality issues, convergence properties can be difficult to establish, especially in the nonlinear case or in off-policy scenarios [3], and their performance in practice heavily depends on the choice of the basis functions [5]. This choice is almost solely experimental, and there is little understanding of the size and complexity of the approximation model, which can result in computationally expensive algorithms that can only be trained in simulated environments by huge dedicated computers units.

As a middle point between the two approaches, state aggregation has been proposed as a quantization scheme for large or infinite spaces [6], and can be viewed as a special case of linear models with the basis functions being indicator functions of a partition of the state/action space. Although this simplicity of the feature space is often desirable, crude approximation can decrease the overall performance of the algorithm, while state aggregation schemes are typically fixed and ad-hoc [7], which results to a sub-optimal representation of the state/action space. In [8], an adaptive state aggregation algorithm that updates the state partition with a vector quantization algorithm while implementing a version of Q-learning in the discretized space is proposed. This leads to a better representation of the state space compared to naive discretization with fewer number of discrete states. This approach is able to use the online observations to create a piece-wise constant approximation of the quality function Q, but heavily depends on two major design parameters: (a) the number of prototypes, which, defines the complexity of the model, and (b) the initial conditions, that affect the performance of the algorithm.

In this work, we extend this result by introducing an online deterministic annealing algorithm for progressive maximum-entropy state-action aggregation. The online deterministic annealing algorithm is a prototype-based clustering algorithm that is robust with respect to the initial conditions, and provides a means to progressively adjust the number of clusters used, via an intuitive bifurcation phenomenon that controls the performance-complexity trade-off created by the interplay of minimum-distortion and maximum-entropy [9], [10]. The proposed algorithm constitutes a two-timescale stochastic approximation algorithm with: (a) a fast component that executes a temporal-difference learning algorithm, and (b) a slow component for adaptive aggregation of the state-action space, based on the online deterministic annealing algorithm. We study the convergence properties of the proposed methodology and propose a way to incorporate sparse Gaussian process regression, calculated on the progressively growing set of the generated codevectors, in order to provide a smoother approximation of the quality function, and an estimation of the uncertainty of the model in a region of the state-action space. Finally, we validate the proposed methodology in simulated experiments.

The authors are with the Electrical and Computer Engineering Department and the Institute for Systems Research, University of Maryland, College Park, USA. emails:{mavridis, nileshs, baras}@umd.edu

Research partially supported by the Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR00111990027, by ONR grant N00014- 17-1-2622, and by a grant from Northrop Grumman Corporation.

II. Mathematical Background and Notation

We consider a discrete-time MDP $(\mathcal{X}, \mathcal{U}, \mathcal{P}, C)$ with $\mathcal{X}$ being the state space, $\mathcal{U}$ being the action (control) space, $\mathcal{P} : (x, u, x') \mapsto \mathbb{P}[x'|x, u]$ being the transition probabilities associated with a stochastic state transition function $f : (x, u) \mapsto x'$, and $C : \mathcal{X} \times \mathcal{U} \to \mathbb{R}_+$, being the immediate cost function, assumed deterministic. Reinforcement Learning (RL) examines the problem of learning a control policy $u := (u_0, u_1, \ldots)$ that solves the discounted infinite-horizon optimal control problem

$$\min_u \mathbb{E}\left[\sum_{l=0}^{\infty} \gamma^l C(x_l, u_l)\right]$$

where $\gamma \in (0, 1]$. We define the value function V^u of a policy u as

$$\begin{aligned} V^u(x_k) :&= \mathbb{E}\left[\sum_{l=k}^{\infty} \gamma^{l-k} C(x_l, u_l)\right] \\ &= C(x_k, u_k) + \gamma \mathbb{E}\left[V^u(x_{k+1}) \mid x_k\right] \\ &= Q^u(x_k, u_k) \end{aligned}$$

where Q^u represents the quality function of a policy u, i.e. the expected return for taking action u_k at time k and state x_k, and thereafter following policy u. As a result of Bellman's principle, we get the (discrete-time) Hamilton-Jacobi-Bellman (HJB) equation

$$\begin{aligned} V^*(x_k) :&= \min_u \ \mathbb{E}\left[\sum_{l=k}^{\infty} \gamma^{l-k} C(x_l, u_l)\right] \\ &\overset{(HJB)}{=} \min_u \{\ C(x_k, u_k) + \gamma \mathbb{E}\left[V^*(x_{k+1}) \mid x_k\right]\ \} \\ &= \min_{u_k} Q^*(x_k, u_k) \end{aligned} \tag{1}$$

where $V^* := V^{u^*}$ and $Q^* := Q^{u^*}$ represent the optimal value and Q functions, respectively. Reinforcement learning algorithms consist mainly of temporal-difference learning algorithms [11] that try to approximate a solution to (1) using iterative optimization methods. The optimization is performed over a finite set of parameters which are used to describe the value (or Q) function. These parameters typically correspond to a parametric model (e.g. a neural network) used for function approximation, or to the different values of the vector $V(\mathcal{X})$ (or $Q(\mathcal{X}, \mathcal{U})$), in which case $\mathcal{X}$ and $\mathcal{U}$ are assumed finite either by definition or as a result of discretization. In this work we will assume that the state and action spaces are finite-dimensional vector spaces that have been appropriately discretized in a finite set. When $\mathcal{X}$ and $\mathcal{U}$ are finite, the Q-learning algorithm, which is a stochastic approximation algorithm [12], can be used:

$$\begin{aligned} Q_{j+1}(x, u') = Q_j(x, u') + \alpha_j [C(x, u') \\ + \gamma \min_u Q_j(x', u) - Q_j(x, u')] \end{aligned}$$

that provably asymptotically minimizes the mean-squared Bellman error:

$$\min_q \ \mathbb{E}\left[\|C(x, u) + \min_u \{\ \gamma Q(x', u)\ \} - q\|^2 \mid x\right].$$

Q-learning depends on a stochastic exploration policy $\pi_L = u'$ which decides the next action given the observed state. Given the policy π_L, the original MDP becomes a Markov Chain, and the Q-learning algorithm becomes an off-policy $TD(0)$ algorithm [3] for value function approximation.

III. State Aggregation with Online Deterministic Annealing

Unsupervised analysis can provide valuable insights into the nature of a data space. In particular, vector quantization [13] can reduce storage needs, reveal structures, such as clusters in the data, or pre-process large datasets for further analysis, through the representation of the data space by a set of prototypes. Given a random variable $X : \Omega \to S$ defined in the probability space $(\Omega, \mathcal{F}, \mathbb{P})$, a quantizer $Q : S \to S$ is defined such that $Q(X) = \sum_{h=1}^{K} \mu_h \mathbb{1}_{[X \in S_h]}$, where $V := \{S_h\}_{h=1}^K$ forms a partition of S and $M := \{\mu_h\}_{h=1}^K$ represents a set of codevectors such that $\mu_h \in ri(S_h)$, $h \in \{1, \ldots, K\}$. Given a dissimilarity measure $d : S \times ri(S) \to [0, \infty)$ one seeks the optimal M, V in terms of minimum average distortion:

$$\min_{M,V} \ D(Q) := \mathbb{E}\left[d\left(X, Q(X)\right)\right]$$

Vector quantization algorithms assume that Q is a deterministic function of X and are proven to converge to locally optimal configurations even when formulated as online learning algorithms [13]. However, their convergence properties and final configuration depend heavily on two design parameters: (a) the number of clusters (neurons), and (b) their initial configuration. To deal with this phenomenon, the Online Deterministic Annealing approach [9] makes use of a probabilistic framework, where input vectors are assigned to clusters in probability, thus dropping the assumption that Q is a deterministic function of X. For the randomized partition, the expected distortion becomes:

$$D = \mathbb{E}\left[d_\phi(X, Q)\right] = \mathbb{E}\left[\mathbb{E}\left[d_\phi(X, Q)|X\right]\right]$$

The central idea of deterministic annealing is to seek the distribution that minimizes D subject to a specified level of randomness, measured by the Shannon entropy

$$H(X, M) = H(X) - \mathbb{E}\left[\mathbb{E}\left[\log p(Q|X)|X\right]\right]$$

with $p(\mu|x)$ representing the association probability relating the input vector x with the codevector μ. This is essentially a realization of the Jaynes's maximum entropy principle [14] which states: of all the probability distributions that satisfy a given set of constraints, choose the one that maximizes the entropy. The resulting multi-objective optimization is conveniently formulated as the minimization of the Lagrangian

$$F = D - TH \tag{2}$$

where T is the temperature parameter that acts as a Lagrange multiplier. Clearly, (2) represents the scalarization method for trade-off analysis between two performance metrics. For large values of T we maximize the entropy, and, as T is lowered, we essentially transition from one Pareto point to

another in a naturally occurring direction that resembles an annealing process. In this regard, the entropy H, which is closely related to the "purity" of the clusters, acts as a regularization term which is given progressively less weight as T decreases. As is the case in vector quantization, one minimizes F via a coordinate block optimization algorithm. Minimizing F with respect to the association probabilities $p(\mu|x)$ is straightforward and yields the Gibbs distribution

$$p(\mu|x) = \frac{e^{-\frac{d(x,\mu)}{T}}}{\sum_{\mu} e^{-\frac{d(x,\mu)}{T}}} \tag{3}$$

while, in order to minimize F with respect to the codevector locations μ we set the gradients to zero

$$\frac{d}{d\mu} D = 0 \implies \frac{d}{d\mu} \mathbb{E}\left[\mathbb{E}\left[d(X,\mu)|X\right]\right] = 0 \tag{4}$$

Adding to the physical analogy, it is significant that, as the temperature is lowered, the system undergoes a sequence of "phase transitions", which consists of natural cluster splits where the cardinality of the codebook (number of prototypes) increases. This is a bifurcation phenomenon that provides a useful tool for controlling the size of the model relating it to the scale of the solution. At very high temperature ($T \to \infty$) the optimization yields uniform association probabilities $p(\mu|x) = 1/K$, and, all the codevectors are located at the same point. As we lower the temperature, the cardinality of the codebook changes. The bifurcation can be traced by generating a perturbed pair of codevectors for each effective cluster, which, after convergence, can either merge together or get separated, depending on whether a phase transition has occurred [9].

A. Bregman Divergences as Dissimilarity Measures

The proximity measure d need not be a metric, and can be generalized to more general dissimilarity measures inspired by information theory and statistical analysis. In particular, the family of Bregman divergences, which includes the widely used Kullback-Leibler divergence, can offer numerous advantages in learning applications compared to the Euclidean distance alone [15]. Notably, in the case of deterministic annealing, Bregman divergences play an even more important role, since we can show that, if d is a Bregman divergence, the solution to the second optimization step (4) can be analytically computed in a convenient centroid form [9]:

$$\mu^* = \mathbb{E}\left[X|\mu\right] \tag{5}$$

B. The online learning rule

In an offline approach, the approximation of the conditional expectation $\mathbb{E}\left[X|\mu\right]$ is computed by the sample mean of the data points weighted by their association probabilities $p(\mu|x)$. To define an online training rule for the deterministic annealing framework, we formulate a stochastic approximation algorithm to recursively estimate $\mathbb{E}\left[X|\mu\right]$ directly. As a direct consequence of Theorem 4 in [9], the following corollary provides an online learning rule that solves the optimization problem of the deterministic annealing algorithm.

Corollary 0.1: The online training rule

$$\begin{cases} \rho_i(n+1) & = \rho_i(n) + \beta(n)\left[\hat{p}(\mu_i|x_n) - \rho_i(n)\right] \\ \sigma_i(n+1) & = \sigma_i(n) + \beta(n)\left[x_n \hat{p}(\mu_i|x_n) - \sigma_i(n)\right] \end{cases} \tag{6}$$

where $\sum_n \beta(n) = \infty$, $\sum_n \beta^2(n) < \infty$, and the quantities $\hat{p}(\mu_i|x_n)$ and $\mu_i(n)$ are recursively updated as follows:

$$\mu_i(n) = \frac{\sigma_i(n)}{\rho_i(n)}, \quad \hat{p}(\mu_i|x_n) = \frac{\rho_i(n) e^{-\frac{d(x_n,\mu_i(n))}{T}}}{\sum_i \rho_i(n) e^{-\frac{d(x_n,\mu_i(n))}{T}}} \tag{7}$$

converges almost surely to a solution of the block optimization (3), (5).

The learning rule (6), (7) is a stochastic approximation algorithm [12] which can be used for the adaptive aggregation of the state-action space $\mathcal{X} \times \mathcal{U}$ of a Markov decision process, as shown in Alg. 1.The finite set $\{\mu_i\}_{i=1}^K = \{(m_i, v_i)\}_{i=1}^K$, where $m_i \in \mathcal{X}$ and $v_i \in \mathcal{U}$, constitutes an optimal representation of $\mathcal{X} \times \mathcal{U}$ with respect to (2), and can be used as is for piece-wise constant approximation or as the training set for the Gaussian process regression algorithm. Notably, the size of $\{(m_i, v_i)\}_i$ progressively increases with respect to a trade-off between accuracy and complexity. A detailed discussion on the implementation of Alg. 1 and the effect of its parameters can be found in [9].

Algorithm 1 State-Action Aggregation Algorithm (ODA)

Select parameters and initial configuration $\{\mu_i\}$
while $K < K_{max}$ **and** $T > T_{min}$ **do**
  Perturb $\mu^i \leftarrow \{\mu_i + \delta, \mu_i - \delta\}$, $\forall i$
  Set $n \leftarrow 0$
  repeat
    Observe state x
    for $i = 1, \ldots, K$ **do**
      Update:

$$p(\mu_i|x) \leftarrow \frac{p(\mu_i) e^{-\frac{d_\phi(x,\mu_i)}{T}}}{\sum_i p(\mu_i) e^{-\frac{d_\phi(x,\mu_i)}{T}}}$$

$$p(\mu_i) \leftarrow p(\mu_i) + \beta_n\left[p(\mu_i|x) - p(\mu_i)\right]$$

$$\sigma(\mu_i) \leftarrow \sigma(\mu_i) + \beta_n\left[x p(\mu_i|x) - \sigma(\mu_i)\right]$$

$$\mu_i \leftarrow \frac{\sigma(\mu_i)}{p(\mu_i)}$$

$$n \leftarrow n + 1$$

    end for
  until Convergence
  Keep effective codevectors
  Lower temperature $T \leftarrow \gamma T$
end while

IV. Reinforcement Learning with Online Deterministic Annealing

We consider an MDP $(\mathcal{X}, \mathcal{U}, \mathcal{P}, C)$, where $S \subseteq \mathbb{R}^{d_X}$, $S \subseteq \mathbb{R}^{d_U}$ are compact convex sets. We are interested in the approximation of the quality function $Q : \mathcal{X} \times \mathcal{U} \to \mathbb{R}_+$. To this end, we use the online deterministic annealing (ODA) algorithm (Alg. 1) as an online greedy algorithm that finds an optimal representation of the data space with

respect to a trade-off between minimum average distortion and maximum entropy. We define a quantizer $Q_P(x,u) = \sum_{h=1}^{K} \mu_h \mathbb{1}_{[(x,u)\in P_h]}$, where $\{P_h\}_{h=1}^{K}$ is a partition of $\mathcal{X}\times\mathcal{U}$. The parameters $\mu_h := (m_h, v_h)$ define a state-action aggregation scheme with k clusters (aggregate state-action pairs), each represented by $m_h \in \mathcal{X}$ and $v_h \in \mathcal{U}$, for $h = 1,\ldots,K$. After convergence, if the representation is meaningful, the finite set $\{\mu_h\}_{h=1}^{K}$, where $\mu_h \in \mathcal{X}\times\mathcal{U}$, can be used directly for piece-wise constant approximation of the Q function, or, as will be discussed bellow, as pseudo-inputs for Gaussian process regression. We stress that the cardinality K of the set of representatives of the space $\mathcal{X}\times\mathcal{U}$ is automatically chosen by Alg. 1 and progressively increases as needed, with respect to the complexity-performance trade-off.

A. The Algorithm

In essence, we are approximating the Q function with a piece-wise constant parametric model with the parameters that define the partition living in the data space and being chosen by the vector quantization algorithm 1. However the system observes its states and actions online while learning its optimal policy using a temporal-difference reinforcement learning algorithm. Therefore, the two estimation algorithms need to run at the same time. This can become possible by observing that Algorithm 1, as well as most temporal-difference algorithms, are stochastic approximation algorithms. Therefore, we design a reinforcement learning algorithm as a two-timescale stochastic approximation algorithm with (a) a fast component that updates the values $Q := \{Q(h)\}_{h=1}^{K}$ with a temporal-difference learning algorithm, and (b) a slow component that updates the representation $\mu := \{\mu_h\}_{h=1}^{K}$ based on Alg. 1. Such a framework can incorporate different reinforcement learning algorithms, including the proposed algorithm presented in Alg. 2. The exploration policy $\pi_L(h|\mu)$ in Alg. 2 depends on the aggregate state h and balances the ratio between exploration and exploitation [16].

The convergence properties of the algorithm can be studied by directly applying the theory of the O.D.E. method for stochastic approximation in two timescales:

Theorem 1 (Ch. 6 of [12]): Consider the sequences $\{x_n\} \in S \subseteq \mathbb{R}^d$ and $\{y_n\} \in \Sigma \subseteq \mathbb{R}^k$, generated by the iterative stochastic approximation schemes:

$$x_{n+1} = x_n + \alpha(n)\left[f(x_n, y_n) + M_{n+1}^{(x)}\right] \quad (8)$$

$$y_{n+1} = y_n + \beta(n)\left[g(x_n, y_n) + M_{n+1}^{(y)}\right] \quad (9)$$

for $n \geq 0$ and $M_n^{(x)}$, $M_n^{(y)}$ martingale difference sequences, and assume that $\sum_n \alpha(n) = \sum_n \beta(n) = \infty$, $\sum_n(\alpha^2(n) + \beta^2(n)) < \infty$, and $\beta(n)/\alpha(n) \to 0$, with the last condition implying that the iterations for $\{y_n\}$ run on a slower timescale than those for $\{x_n\}$. If the equation

$$\dot{x}(t) = f(x(t), y),\ x(0) = x_0$$

has an asymptotically stable equilibrium $\lambda(y)$ for fixed y and some Lipschitz mapping λ, and the equation

$$\dot{y}(t) = g(\lambda(y(t)), y(t)),\ y(0) = y_0$$

has an asymptotically stable equilibrium y^*, then, almost surely, (x_n, y_n) converges to $(\lambda(y^*), y^*)$.

Algorithm 2 Reinforcement Learning Algorithm with ODA

Initialize μ_h, $Q_0(h)$, $\forall h \in \{1,\ldots,K\}$
repeat
  Observe x and find

$$h = \arg\min_{\tau=1,\ldots,k}\ d_\phi((x,u'), \mu_\tau)$$

  Choose $u' = \pi_L(h|\mu)$
  Observe $x' = f(x,u')$ and find

$$h' = \arg\min_{\tau=1,\ldots,k}\ d_\phi(x', \mu(\tau))$$

  Update $Q(h)$:

$$Q_{i+1}(h) = Q_i(h) + \alpha_i[C(x,u') + \gamma \min_u Q_i(h') - Q_i(h)]$$

  if $i \mod N = 0$ **then**
    Update partition $\mu := \{\mu_h\}_{h=1}^{K}$ using Alg. 1
  end if
until Convergence
Update Policy:

$$u^*(x) = \arg\min_u \{\ Q(h(x))\ \}$$

Proposition 1: Algorithm 2 converges almost surely to (μ^*, Q^*) where $\mu*$ is a solution of the block optimization problem (3), (5), and Q^* minimizes the temporal-difference error:

$$J_h = \|\mathbb{E}\left[C(x,u) + \gamma \min_u Q(h')|(x,u) \in P_h\right] - Q(h)\|^2 \quad (10)$$

where $h = 1,\ldots,K$, and $\{P_h\}_{h=1}^{K}$ is a partition of $\mathcal{X}\times\mathcal{U}$ with every P_h assumed to be visited infinitely often.

Proof: From Theorem 0.1, it follows that Algorithm (1) is a stochastic approximation algorithm of the form (9) that converges to a solution of (3), (5). Also (13) is a stochastic approximation algorithm of the form (8), for $f(Q(h)) = -\nabla_{Q(h)} J_h$. The result follows from Theorem 1. ■

We note that the condition $\beta_i/\alpha_i \to 0$ is of great importance. Intuitively, Algorithm 2 consists of two components running in different timescales. The slow component updates μ and is viewed as quasi-static when analyzing the behavior of the fast transient Q which updates the approximation of the quality function. As an example, the condition $\beta_n/\alpha_n \to 0$ is satisfied by stepsizes of the form $(\alpha_n, \beta_n) = (1/n, 1/1+n\log n)$, or $(\alpha_n, \beta_n) = (1/n^{2/3}, 1/n)$. Another way of achieving the two-timescale effect is to run the iterations for the slow component $\{\mu_n\}$ with stepsizes $\{\alpha_{n(k)}\}$, where $n(k)$ is a subsequence of n that becomes increasingly rare (i.e. $n(k+1) - n(k) \to \infty$), while keeping its values constant between these instants. In practice, it has been observed that a good policy is to run the slow component with slower stepsize schedule β_n and update it along a subsequence keeping its values constant in between

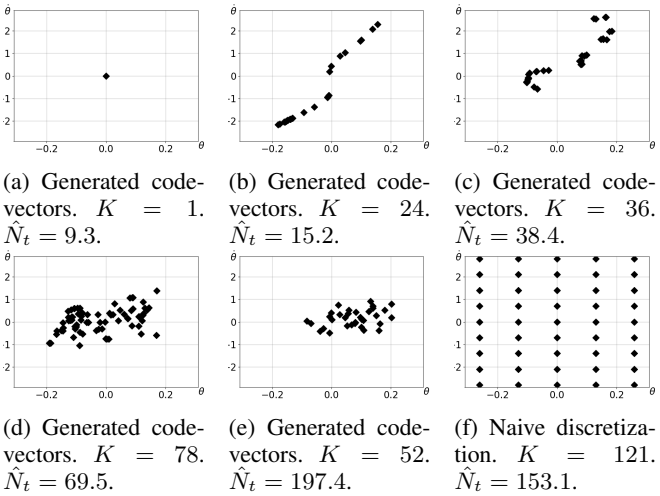
([12], Ch. 6). This explains the parameter N in Alg. 2 whose value should increase with time.

B. Experimental Results

We illustrate the properties and quantify the performance of the proposed methodology on the Cart-pole (inverted pendulum) problem [17]. The state variable of the cart-pole system has four components $(x, \theta, \dot{x}, \dot{\theta})$, where x and $\dot{x}$ are the position and velocity of the cart on the track, and θ and $\dot{\theta}$ are the angle and angular velocity of the pole with the vertical. The cart is free to move within the bounds of a one-dimensional track. The pole is free to move only in the vertical plane of the cart and track. The action space consists of an impulsive "left" or "right" force $F \in \{-10, +10\}$N of fixed magnitude to the cart at discrete time intervals. The transition function for the state x is $x_{n+1} = x_n + \tau\dot{x}$, where $\tau = 0.02$s. The initial state is set to $X_0 = (u_x, u_\theta, u_{\dot{x}}, u_{\dot{\theta}})$ where u_x, u_θ, $u_{\dot{x}}$, and $u_{\dot{\theta}}$ follow a uniform distribution $U(-0.05, 0.05)$. Failure occurs when $|\theta| > 12°$ or when $|x| > 2.4$m. An episode terminates successfully after N_t timesteps, and the average number of timesteps $\hat{N}_t \leq N_t$ across different attempts, is used to quantify the performance of the learning algorithm. We use the squared Euclidean distance as the Bregman divergence d_ϕ.

In Fig. 1, we illustrate the main property of the proposed adaptive state aggregation, which is the ability to automatically adjust the number of the aggregate states with respect to the observed state-action pairs. We focus on the two-dimensional subspace $(\theta, \dot{\theta})$ and visualize the progression of the number and placement of the centroids of the aggregate states generated by Alg. 2. In Fig. 1(f) we show the codevectors corresponding to a naive discretization for $n = 11$ bins per dimension. We note that the codevector placement for Fig. 1 (f) is crucial for the performance of the algorithm and was selected after experimentation. In contrast, Alg. 2 shows increased performance and reduced time and memory complexity, compared to the naive approach, while automatically adapting the codevector placement to the state-action observations.

In Fig. 2 we compare the average number of timesteps (here $N_t = 1000$) with respect to the number of aggregate states used, for three different state aggregation algorithms. The first one is naive discretization without state aggregation, the second is the SOM-based algorithm proposed in [8], and the last is the proposed algorithm Alg. 2. We initialize the codevectors μ by uniformly discretizing over $\hat{S} \times \{-10, 10\}$, for $\hat{S} = [-1, 1] \times [-4, 4] \times [-1, 1] \times [-4, 4]$. We use $K \in \{16, 81, 256, 625\}$ clusters, corresponding to a standard discretization scheme with only $n \in \{2, 3, 4, 5\}$ bins for each dimension. As expected, state aggregation outperforms standard discretization of the state-action space. The ability to progressively adapt the number and placement of the centroids of the aggregate states is an important property of the proposed algorithm, 5 instances of which are presented in Fig. 2 for different parameters T_{min}, which result to $K \in \{56, 118, 136, 202, 252\}$ aggregate states. As shown, the behavior of Alg. 2 depends on the temperature schedule T, as well as on hyperparameters such as the the profile of the stepsizes α_i and β_i.



(a) Generated codevectors. $K = 1$. $\hat{N}_t = 9.3$.
(b) Generated codevectors. $K = 24$. $\hat{N}_t = 15.2$.
(c) Generated codevectors. $K = 36$. $\hat{N}_t = 38.4$.
(d) Generated codevectors. $K = 78$. $\hat{N}_t = 69.5$.
(e) Generated codevectors. $K = 52$. $\hat{N}_t = 197.4$.
(f) Naive discretization. $K = 121$. $\hat{N}_t = 153.1$.

Fig. 1: (a)-(e):Illustration of the codevectors generated by Alg. 2 until convergence in the two-dimensional state space $(\theta, \dot{\theta})$. (f) The codevectors corresponding to a naive discretization for $n = 11$ bins per dimension. Here $N_t = 200$.

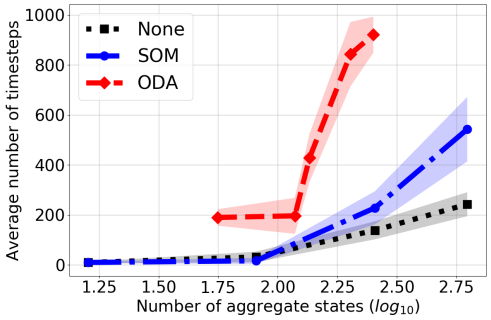


Fig. 2: Average number of timesteps ($N_t = 1000$) over number of aggregate states used. (red) the proposed algorithm. (black) Q-learning without state aggregation. (blue) the SOM-based algorithm of [8].

V. Sparse Gaussian Process Regression with Online Deterministic Annealing

When used for state aggregation as described above, online deterministic annealing can greatly improve the efficiency of approximate Q-learning via a piece-wise constant approximation of the Q function, according to an automatically generated partition. For a smooth approximation, we can use the codevectors generated by the online deterministic annealing algorithm as a training set for Gaussian process regression. In this way, we overcome the well known computational bottleneck of Gaussian processes (time complexity of $O(n^3)$, where n is the number of known data points, see also [21]), while making use of their non-parametric regression properties that also allow for the quantification of the uncertainty of the model in each region of the space [18].

A. Gaussian Processes based on State Aggregation

A Gaussian Process is a distribution over the space of functions where any subset of which has a Gaussian distribution [18]. It can be used to estimate the quality function of a pair $(x, u) \in \mathcal{X} \times \mathcal{U}$ given a representation μ and its corresponding Q values, according to:

$$q(x, u|\mu, Q) = k(x, u, \mu)^{\mathrm{T}}(K(\mu) + \sigma^2\mathbb{I})^{-1}Q \tag{11}$$

where $k(x, u, \mu) = [k((x, u), \mu_1), \ldots, k((x, u), \mu_K)]$. The uncertainty of the prediction is also available for every point (x, u), and is given in terms of the covariance:

$$\begin{aligned}\Sigma(x, u|\mu, Q) &= k((x, u), (x, u)) \\ &\quad - k(x, u, \mu)^{\mathrm{T}}(K(\mu) + \sigma^2\mathbb{I})^{-1}k(x, u, \mu) + \sigma^2\end{aligned} \tag{12}$$

The Q-learning update of Alg. 2:

$$Q_{i+1}(h) = Q_i(h) + \alpha_i[C(x, u') + \gamma \min_u Q_i(h') - Q_i(h)]$$

becomes the temporal-difference algorithm:

$$\begin{aligned}Q_{i+1}(h) = Q_i(h) + \alpha_i[&C(x, u') \\ &+ \gamma \min_u q(x', u|\mu, Q_i) - Q_i(h)]\end{aligned} \tag{13}$$

where $h = \underset{\tau=1,\ldots,k}{\arg\min}\ d_\phi((x, u'), \mu_\tau)$, and N increases over time, as explained above. The convergence of (13) can be shown in a similar way as above, and minimizes the error:

$$J_h = \|\mathbb{E}\left[C(x, u) + \gamma \min_u q(x, u|\mu, Q)|(x, u) \in P_h\right] - Q(h)\|^2 \tag{14}$$

where $h = 1, \ldots, K$, $q(x, u|\mu, Q)$ is given by (11) and $\{P_h\}_{h=1}^K$ is a partition of $\mathcal{X} \times \mathcal{U}$ with every P_h assumed to be visited infinitely often. The exploration policy becomes $\pi_L(x|\mu, Q)$ which now depends on the actual state x, and requires a Gaussian process prediction. In the case of the ϵ-greedy policy [16], it is given by:

$$\pi_L(x|\mu, Q) = \begin{cases} r, \text{ if } \epsilon < \epsilon_r \\ \min_u q(x, u|\mu, Q), \text{ o.w.} \end{cases} \tag{15}$$

where r is uniformly distributed in the action space $\mathcal{U}$, ϵ is uniformly distributed in $[0, 1]$ and $\epsilon_r \in [0, 1]$ is a threshold parameter that monotonically decreases over time. We note that the inversion of matrix K during Gaussian process prediction, need only be done when the partition parameters μ are updated, which happens every N iterations of the temporal-difference algorithm.

VI. Conclusion and Discussion

We introduced a reinforcement learning algorithm based on an online state aggregation scheme that progressively adjusts the aggregate states, as well as their number, with respect to a performance-complexity trade-off measured in terms of maximum entropy. The proposed algorithm constitutes a two-timescale stochastic approximation algorithm with: (a) a fast component that executes a Q-learning algorithm, and (b) a slow component, based on an online deterministic annealing algorithm, a prototype-based unsupervised learning algorithm that shows robustness with respect to the initial conditions, requires minimal hyper-parameter tuning, and offers online control over the performance-complexity trade-off, by generating a progressively increasing number of codevectors. The resulting piece-wise Q-function approximation can be extended with the use of sparse Gaussian processes defined on the progressively growing set of the generated codevectors. This approach can provide a smooth and progressively more accurate Q-function approximation and a means to estimate the uncertainty of the model in a given region of the state-action space.

References

[1] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

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

[3] J. N. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximation," *IEEE Transactions on Automatic Control*, vol. 42, no. 5, pp. 674–690, 1997.

[4] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," in *International Conference on Machine Learning*. PMLR, 2016, pp. 2829–2838.

[5] C. Dann, G. Neumann, J. Peters, *et al.*, "Policy evaluation with temporal differences: A survey and comparison," *Journal of Machine Learning Research*, vol. 15, pp. 809–883, 2014.

[6] J. Baras and V. Borkar, "A learning algorithm for markov decision processes with adaptive state aggregation," in *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No. 00CH37187)*, vol. 4. IEEE, 2000, pp. 3351–3356.

[7] A. George, W. B. Powell, and S. R. Kulkarni, "Value function approximation using multiple aggregation for multiattribute resource management," *Journal of Machine Learning Research*, 2008.

[8] C. N. Mavridis and J. S. Baras, "Vector quantization for adaptive state aggregation in reinforcement learning," in *2021 American Control Conference (ACC)*. IEEE, 2021.

[9] C. Mavridis and J. Baras, "Online deterministic annealing for classification and clustering," 2021.

[10] C. N. Mavridis and J. S. Baras, "Progressive graph partitioning based on information diffusion," in *Conference on Decision and Control (CDC)*. IEEE, 2021.

[11] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine learning*, vol. 3, no. 1, pp. 9–44, 1988.

[12] V. S. Borkar, *Stochastic approximation: a dynamical systems viewpoint*. Springer, 2009, vol. 48.

[13] C. N. Mavridis and J. S. Baras, "Convergence of stochastic vector quantization and learning vector quantization with bregman divergences," in *21rst IFAC World Congress*. IFAC, 2020.

[14] E. T. Jaynes, "Information theory and statistical mechanics," *Physical review*, vol. 106, no. 4, p. 620, 1957.

[15] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh, "Clustering with bregman divergences," *Journal of machine learning research*, 2005.

[16] C. J. C. H. Watkins, "Learning from delayed rewards," 1989.

[17] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE transactions on systems, man, and cybernetics*, pp. 834–846, 1983.

[18] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer school on machine learning*. Springer, 2003, pp. 63–71.

[19] C. E. Rasmussen, M. Kuss, *et al.*, "Gaussian processes in reinforcement learning." in *NIPS*, vol. 4. Citeseer, 2003, p. 1.

[20] A. Ranganathan, M.-H. Yang, and J. Ho, "Online sparse gaussian process regression and its applications," *IEEE Transactions on Image Processing*, vol. 20, no. 2, pp. 391–404, 2010.

[21] E. Snelson and Z. Ghahramani, "Sparse gaussian processes using pseudo-inputs," *Advances in neural information processing systems*, vol. 18, pp. 1257–1264, 2005.