Reinforcement Field

Po-Hsiang Chiu and Manfred Huber
Department of Computer Science and Engineering
University of Texas at Arlington, Arlington, TX 76019, USA
po-hsiang.chiu@mavs.uta.edu, huber@cse.uta.edu

Abstract— Complex control tasks involving varying or evolving system dynamics often pose a great challenge to mainstream reinforcement learning algorithms. Specifically, in most standard methods, actions are often assumed to be a concrete and fixed set that applies to the state space in a predefined manner. Consequently, without resorting to a substantial re-learning procedure, the derived policy lacks the ability to adapt to variations in action outcomes or shifts in the action set. In addition, the standard action representation and its attendant state transition mechanism limit the applicability of the RL framework in complex domains primarily due to the intractability of the resulting large state space and lack of the facility to generalize the learned policy to the unknown parts of the state space. This paper proposes an alternative view of reinforcement learning by establishing the notion of the reinforcement field through a collection of policy-embedded particles gathered during the policy learning process. The reinforcement field serves as a policy generalization mechanism through the use of kernel functions as a state correlation hypothesis in combination with Gaussian process regression as a value function approximator.

Keywords- kernel methods, Gaussian process regression, parametric actions, reinforcement learning.

I. INTRODUCTION

Optimal control in complex domains often requires adaptive policies in response to the continuously varying environment. The standard reinforcement learning framework [4] establishes a unique way of deriving such control policies over the state space representation in which the policy unfolds as a sequence of actions, guiding the agent towards a trajectory with a maximized aggregate reward. A potential function maps input states or state-action pairs to their corresponding values, which serve as a fundamental basis for evaluating a control policy. The complexity of RL algorithms thus depends largely on the dimensionality of the state space which then governs how effectively potential function values (or utilities) can be evaluated. The challenge arises when applying RL methods to real-world control tasks that often require numerous features to be accurately represented, leading to an exponential growth in the state space and the difficulty in estimating the potential function. One of the core issues in the standard RL framework lies in the fact that the control policy depends on a concrete set of actions, modeling allowable behaviors of the agent, while the overall behaviors themselves are often dynamic processes. In high-precision control tasks such as vehicle steering, for instance, more than a few physical attributes are required to express a particular movement (as an action) such as steering the vehicle within a certain range of speeds and angles. Furthermore, once an action is executed, the outcomes may not be as consistent as expected due to the stochastic nature of the environment or of the actuators themselves. In a robotic navigation domain, for instance, the varying environment dynamics could be related to a bumpy surface during travel or simply a poor calibration of the motor. As a result, the agent may deviate away from the course where an optimal policy is feasible. Real-time error recovery or policy adjustments are not possible without reevaluating the policy from the start since the agent is not aware of the “meaning” of the actions beyond the policy being followed. To address the uncertainty in the ever-changing environment and varying action performance (including the variations in the permissible action set) without excessive computational burden, this paper develops a more compact yet fluid RL formulation that integrates the action dynamics and policy evaluation for a better policy generalization capacity.

Recent work in function approximation methods [2] has been used extensively to ameliorate the dimensionality barrier in complex systems by using a selective set of basis functions to represent the potential function. However, challenges arise in choosing the right basis set that, if sufficiently large, generalizes well enough to reflect the topology of the state space. Still, the size of the basis set also grows rapidly with the dimension of the state space. The Proto-reinforcement learning framework [3] identifies a task-independent basis set from the samples of prior learning experiences by analyzing the topology of the state space encoded via the graph Laplacian. In this paper, Gaussian process regression (GPR) is used as an alternative route in representing the potential function, which is also driven by samples of experiences but without the need to learn the basis functions directly. Related work in GP-based RL can be found in [5]. Despite the relation with the methods above, the primary objective for this paper is to first define a more flexible expression for actions such that they combine the local dynamics of state transitions and the policy search as an integrated continuous process. In particular, actions altogether are treated as an operator-like construct that explicitly encodes the dynamic process of state transitions. The action operator is also linked to the potential function through action parameters, enabling the result to be predictable whenever acting on states. Further, with actions being parameterized in a manner that exhibits properties of a continuous function, we further establish the notion of the reinforcement field as a policy generalization mechanism. The property of the field comes from using GP as a kernel machine in which the kernel represents an adaptive correlation hypothesis over samples of experiences while operating on the vector field in Hilbert space. The potential function in the reinforcement field is driven by an underlying control learner using a temporal-difference learning algorithm [4] for value predictions. In
particular, the potential value is taken to represent a fitness value, similar to a utility estimate, in response to the action operator acting on a state. The exact behavior of the action operator depends on the action parameters with their values stochastically determined by the related random processes.

II. RELATED BACKGROUND AND NOTATIONS

The goal of this paper is to establish the notion of the reinforcement field, in which the policy learning and generalizing takes place. The policy derivation in the field encompasses the following integrated processes: sampling, time-varying action operator, kernel tuning and sample-driven (or non-parametric) value function approximation. Prior to the construction of the reinforcement field, it is instructive to first visit two principal algorithmic frameworks upon which this new policy learning method is based: i) temporal-difference learning and ii) Gaussian process regression (GPR). Following, each section also briefly outlines the thought process leading towards the transformation from the standard RL framework to the field-based policy learning approach.

A. TD Learning and SARSA

Consider the policy learning over a Markov Decision Process (MDP) where the agent perceives the current state \( s \in S \), chooses an action \( a \in A \), followed by a transition into the next state \( s' \). The agent subsequently receives a reward (or cost) \( r \) determined by a reward function \( R: S \times A \to \mathbb{R} \). The state transition is governed by the environment dynamic represented through a map \( T: S \times A \times S \to [0,1] \) specifying the probability upon observing \( s' \) by executing action \( a \) in \( s \). The objective of the agent is to learn a control policy \( \pi: S \times A \to [0,1] \) that leads to a maximized accumulated reward where \( \pi \) effectively maps states to a probability distribution over actions. In particular, given a policy \( \pi \), the corresponding action-value function \( Q^\pi(s,a) \) computes the long-term accumulated reward in the form of the Bellman equation:

\[
Q^\pi(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \sum_{a' \in A} \pi(s',a') Q^\pi(s',a')
\]  

However, very often the environment dynamic (\( T \)) and the reward function (\( R \)) are unknown to the agent, thus giving rise to the RL framework. Our description of policy learning in the reinforcement field will use SARSA [4] as the underlying control learner although other RL algorithms such as Q-Learning, LSPI, etc could be used instead. SARSA uses temporal differences to estimate Q-values with the following update rule, where \( \alpha \) denotes a learning rate:

\[
Q(s,a) \leftarrow Q(s,a) + \alpha [R(s,a) + \gamma Q(s',a') - Q(s,a)]
\]

In order to express the notion that actions are not merely immutable choices with predefined behavioral details, the first step in generalizing the standard RL framework is to introduce parameters to actions. The rationale follows from the fact that in reality, executing an action involves a variational procedure over the configuration of the active state; that is, the state feature values change in response to the continuous process during which an action is applied. The variation could come from the unpredictability of the actions when performed (e.g. errors from the related actuator), affordance or feasibility of the action with respect to a particular state (i.e. a usage scenario), or certain localized effects the action may impose on the environment only through a subset of the state features (e.g. inclusions or removals of objects). An example of local effects due to the action can be found in the related work [6]. Thus, parameterizing the actions introduces extra degrees of freedom necessary to express such action dynamics. For clarity, an action choice from the standard RL (i.e. \( a \in A \)) is referred to as a primitive action. This is in contrast to a parametric action that is essentially a primitive action further modeled by the selected parameters expressing the action dynamics when applied to states. In this manner, each primitive action is represented by a vector \( x_a \) with relevant features that encode specifically what local variations can occur when it is chosen to act on a state. Fig. 1 illustrates a 2D navigation domain in which the actions of the agent are parameterized by an angle and a radius increment per move relative to the current position; i.e. \( x_a = (\Delta r, \Delta \theta) \). The parametric-action model, however, effectively introduces an extended state space where (2) no longer applies without a further extension. Expanding upon the standard RL formulation, three special constructs are defined that help in realizing the desired properties using the parametric-action model: i) augmented state space (Definition 1) ii) potential function over augmented state space (Definition 2) and iii) experience particles (Definition 3).

Definition 1 (Augmented State Space) The augmented state space \( S' \) is a tensor product of the state space \( S \) and the action parameter space \( A' : S' = S \otimes A' \).

Definition 2 (Fitness Function) A fitness function is a potential function \( Q^\pi: S' \to \mathbb{R} \) that maps an augmented state \( s' \in S' \) to a fitness value \( Q^\pi(s') \).

Definition 3 (Experience Particle) An experience particle is an abstraction of a functional datum consisting of the tuple \((s',Q^\pi(s'))\) representing an augmented state \( s' \) and its corresponding fitness value \( Q^\pi(s') \).

Note that \( A' \) in Definition 1 refers to the parameter space consisting of action vectors \( \{x_a\} \) in contrast to the (primitive) action set \( a(x_a) \in A \). However, with the parametric action formulation, each primitive action is now parameterized. In this paper, each coordinate of the action vector is modeled according to a yield function of the following form:

\[
Y(x) = P(x_i + w_i) \in \Gamma \]

such that \( (x_i)_i = x_i \ | \ Y(x_i) \geq \eta \), where \( x_i \) represents a target value, \( w_i \) represents a random variation, \( \Gamma \) is the set of acceptable values and finally \( \eta \in [0,1] \) is a probability threshold. The target value \( x_i \) models the expected local variation by which the related state feature will shift upon a transition to the next state. This corresponds to a “normal
behavior” of the action along this coordinate. The uncertainty in action performance is modeled by \( w_i \) which is chosen to follow a task-dependent probability distribution. For instance, a Gaussian distribution is often a reasonable choice. An action parameter value outside the boundary of \( \Gamma \) is taken to be either infeasible or empirically improbable by assuming \( \eta \) being close to 1. In Fig.1, the left-most circle indicates 12 possible orientations of movements in which each pie-shaped boundary in between two concentric circles represents a feasible range of the motion in terms of the action parameters: \((\Delta r, \Delta \theta)\). For instance, action 3 represents a motion moving along the direction to the east but with a tolerance of any errors within ±15 degrees. Similarly, the radius is allowed to fall within the inner and outer circles. Note that the uncertainty in action parameters can alternatively be interpreted as the uncertainty inherent in the observations of states, depending on the choice of approximating an MDP or a Partially Observable MDP (POMDP). This paper will focus on the MDP approximation.

In the new framework, every time the agent executes a parametric action in a given state, an experience particle (Definition 3) is created, encapsulating the corresponding augmented state along with a “measurement” of its potential value, addressed specifically as the fitness value (Definition 2). As an illustration, suppose that \( x_i \) and \( x_j \) represent a feature vector for a state \( s \in S \) and a primitive action \( a \in A \) respectively, then the corresponding augmented state \( S^a \) can often be considered as the joint vector space spanned by merging the coordinates of the tensor product: \((x_i, x_j) \in S \otimes A^r\) to give \( x = (x_i^1, x_j^1, x_i^2, x_j^2, \ldots, x_i^m, x_j^m)\), assuming that \( x_i \) has a dimension \( n \) and \( x_j \) has a dimension \( m \). The fitness function \( Q^a \) evaluated over an augmented state is given by:

\[
Q^a(s^a) = Q^a(x) = Q^a((x_i^1, x_j^1, x_i^2, x_j^2, \ldots, x_i^m, x_j^m))
\]

(3)

Thus, if the agent traverses the state space with \( m \) steps, a sequence of training samples will be generated and retained in the memory: \( \Omega := \{(x_1, q_1), \ldots, (x_m, q_m)\} \subset X \times Q \), where \( X \) and \( Q \) respectively, denote the set of augmented states and their corresponding observed fitness values. It is helpful for the moment to consider these training samples in \( \Omega \) as a functional data set referenced by the experience particles distributed over the state space. The mechanism for estimating their values are deferred to later in the section. Each experience particle effectively defines a control policy that generalizes into the neighboring (augmented) state space through properties of the kernel to be discussed shortly. The similarity of any two particles (or equivalently, two referenced augmented states) takes into account both the state vector \( x_i \) and action vector \( x_j \). This formulation is made possible by allowing the action to take on continuous variables along with the use of a kernel function as a correlation hypothesis associating one augmented state to another through their inner products in the kernel-induced feature space \([1]\). With the above constructs defined, the next step toward establishing the reinforcement field is to represent the fitness function in a manner that integrates with parametric actions and, in the meantime, serves as a “critic” for the policy embodied in experience particles. This paper represents the fitness value function through a progressively-updated Gaussian process.

### B. Gaussian Process Regression

Consider the training set \( \Omega := \{(x_1, q_1), \ldots, (x_n, q_n)\} \) given earlier. The goal is to identify a function \( Q^a(x) \) fitting these samples with a tradeoff between quality of the prediction (i.e. data fit) and smoothness assumptions of the function (i.e. model complexity) \([8]\). Value predictions using GPR first assume a normal prior distribution over functions, and subsequently reject those not consistent with the observations (i.e. experience particles). Suppose that we wish to reconstruct the target function \( Q^a \) from the noisy observations of fitness values, i.e. \( q_i = Q^a(x_i) + \epsilon_i \) where \( \epsilon_i \) is assumed to be i.i.d. Gaussian noise such that any set \{\( q_i \)\} follows a Gaussian distribution with a covariance function \( k(\cdot, \cdot) \). The prior distribution over the observed values \{\( q_i \)\} is thus given by: \( q \sim N(0, K) \), where \( K = k(x, x) \) and the fitness value observations are assumed for simplicity to be centered around a zero mean. In this paper, we will explore two different kernels – an SE kernel in (4) and a product kernel in (5). Equation (4) is a noisy squared exponential kernel (SE) that combines all the features and action parameters in the exponential term; i.e.

\[
k(x, x') = \theta_0 \exp \left(-\frac{1}{2} (x - x')^T \Sigma^{-1} (x - x') \right) + \theta_2 \sigma_{noise}^2
\]

(4)

In (4), the diagonal elements in \( D \), signal amplitude \( \theta_0 \) and the noise magnitude \( \theta_2 \) correspond to the hyperparameters of the kernel. Alternatively, the state features and action parameters can be represented via two separate kernels so as to distinguish them as two different objects; this gives:

\[
k(x, x') = \theta_2 k(x, x')k(x, x') + \theta_2 \sigma_{noise}^2
\]

(5)

where \( k \) and \( k_2 \) represent the state kernel and action kernel, respectively, both assuming to be an SE kernel in this paper. After observing a series of experience particles, the predictive distribution for the value of a novel augmented state \( x^* \), is given by: \( q | x^*, q, x \sim N(\overline{q}, \sigma) \) where:

\[
\overline{q}(x^*) = \overline{q}_0 = k(x^*, x)^T K(x, x)^{-1} q
\]

(6)

and

\[
k(x, x') = k(x, x') - k(x, x)^T K^{-1} k(x, x')
\]

(7)

In particular, the mean prediction following (6) is used for evaluating the fitness value \( Q^a \) for a given augmented state.

The choice of using GPR as a representation for the fitness function boils down to the consideration of a few desired properties. First, the GP-based fitness function assimilates both the state features and action parameters through the kernel as a covariance function such that the similarity in the prediction also reflects the degree of similarity of the policy. Secondly, since the value prediction is driven by the samples (i.e. experience particles) which have certain correlation to one another (as measured by the kernel), the policy can theoretically be deduced everywhere in the state space including those regions not yet being explored. That is, as long as there exist sufficiently correlated particles in the proximity, the kernel as a correlation hypothesis for particles will continue to apply without requiring the agent to explicitly explore the unknown region. We shall see that this property along with the action operator (to be defined in Section III) have benefits that build upon each other. Further, the kernel in GPR adapts to the
ongoing variations in the environment dynamics via the ARD procedure [8] that tunes the kernel hyperparameters using conjugate gradient optimization. The ARD procedure essentially performs an automatic model selection by maximizing the marginal likelihood of the data:

$$\log p(q|X) = -\frac{1}{2} q^T K^{-1} q - \frac{1}{2} \log |K| - n \frac{1}{2} \log 2\pi,$$

(8)

where entries in $K$ are evaluated through a given kernel (e.g. (4), (5), etc). This helps in determining the relevance degree with which each feature contributes to the final prediction.

C. Representing the Policy

Since the objective of the fitness function $Q^+$ is to act as a critic for experience particles, the policy is expected to take on the form of a functional depending on $Q^+$. Using a softmax function, the policy over a pair of state and parametric action (referenced by a particle) can be represented by:

$$\pi(s,a^{(i)}) = \frac{\exp(Q^+(s,a^{(i)})/\tau)}{\sum_j \exp(Q^+(s,a^{(j)})/\tau)}$$

$$= \frac{\exp(Q^+(x^i,y^i)/\tau)}{\sum_j \exp(Q^+(x^j,y^j)/\tau)}$$

(9)

Although $(x^i,y^i)$ in (9) is in general an element of the tensor product $S \otimes A^*$, however, the GP-based fitness value predictor in many cases transforms this tensor element into a joint vector representing an augmented state. This can be seen, for instance, by comparing (3) with the functional form in (4). Consequently, $Q^+(x^i,y^i)$ can be simplified to $Q^+(x)$ for notational convenience where $x$ combines the coordinates of the state and the parametric action. To obtain values for $Q^+$ for each augmented state, GPR works in parallel with an underlying control learner such as SARSA that “perceives” parametric actions (i.e. $a(x^i) \in A$) as individual choices (i.e. primitive actions $a \in A$) without knowing the underlying parameters that govern the action dynamics. That is, from the perspective of the control learner, executing a parametric action leads to an update of a utility estimate as usual. In the case of SARSA, this effectively corresponds to the update rule in (2). The utility estimate for a given state-action pair $(s,a)$ subsequently propagates to GPR to serve as a training signal for the associated augmented state. In this manner, the complexity for the fitness value prediction need not be constrained by the size of the state space. For instance, the SARSA agent can be chosen to operate on a set of discretized state partitions $S^{(i)} \subseteq S$ so that each fitness value for an augmented state is effectively also an approximate utility value for the action applied in the underlying state partition. Each state partition $S^{(i)}$ maintains a reference to several particles, each of which has an assigned utility estimate, reflecting the subtle differences in the exact values of the augmented states. Specifically, this amounts to the partition $S^{(i)}$ referencing the set $\{(x_j,y_j)\}$ with $(x_j,y_j)$ falling into the boundary of $S^{(i)}$, where the subscript index distinguishes different particles. Doing so implies that there is a trade-off between the accuracy in the utility estimate and the granularity of the underlying state partitions (i.e. the particle density).

III. REINFORCEMENT FIELD

A. Action Operator

The reinforcement field combines both the insights from various field theories in physics as well as properties of the kernel function. In the most abstract sense, a field acts on an object (e.g. a particle) in a manner governed by the law of the field. For instance, a charged particle moving in a magnetic field is subject to a Lorentz force that induces a circular motion on the particle. The magnetic field implicitly "prescribes" a particular state sequence for the particle to follow during the course of its travel. The circular motion can be justified by the principle of least action [7] in which an action, for the case of one particle, is defined in terms of the Lagrangian $L$ of the form: $\int_0^t L(q(t),q'(t))dt$, where $q$ refers to a generalized coordinate of a physical quantity of interest such as the position of the particle. The action in this formulation essentially represents a global constraint for the particle trajectory. Incidentally, the term action overlaps with that in the RL terminology. Minimizing the action with respect to $q(t)$ (and $q'(t)$) using calculus of variation leads to the Euler-Lagrange equation that effectively represents the local dynamics of the particle, i.e., the motion equations for the particle. Thus, we see that in a physical system, there exists a dynamic process that drives state transitions systematically towards a certain objective (e.g. shortest path of travel) in a manner consistent with the law of the field. By connecting this analogy to the RL framework, the objective of maximizing the accumulated reward corresponds to a global constraint while the local dynamics can be expressed by the parametric actions. An action in this sense assumes the property of an operator, often in a functional form (e.g. the integral over the Lagrangian $L(\cdot)$) over a set of related parameters (e.g. position and velocity). For the reasons stated above, this paper seeks to generalize the parametric action further into an operator-like construct with tunable parameters characterizing the local dynamics of state transitions and simultaneously links it to the fitness function.

For the sake of discussion, consider three snapshots in a 2D navigation domain in Fig. 2 where the motion of the agent is again parameterized by the radius and angle increment, i.e. $(\Delta r, \Delta \theta)$. Both $\Delta r$ and $\Delta \theta$ are continuous random variables modeled by yield functions discussed earlier. Given this, one can alternatively represent a particular state transition in terms of the matrix multiplication:

$$\begin{bmatrix}
1 + \frac{\Delta x}{x} & 0 \\
0 & 1 + \frac{\Delta y}{y}
\end{bmatrix} \begin{bmatrix}
x \\
y
\end{bmatrix} = \begin{bmatrix}
x + \Delta x \\
y + \Delta y
\end{bmatrix}$$

(10)

where $\Delta x = \Delta r \cos(\Delta \theta)$ and $\Delta y = \Delta r \sin(\Delta \theta)$. That is, action is represented as an operator (e.g. matrix) that takes an input state $(x,y)$ and produces the next state $(x + \Delta x, y + \Delta y)$. Note that the action operator effectively generalizes different primitive actions for the control learner by assuming different parameters.
within certain bounds. With the action operator, the objective of the learning in standard RL is altered slightly to the following statement: determine the action operator that acts on each state in a manner that leads to a trajectory for a maximum accumulated reward. Effectively, the action operator serves as a local constraint each state transition must follow while the fitness function assesses the result of the action operator acting on a state while assuming particular parameters.

B. Policy Learning in the Field

Recall from (6) that the fitness value function is represented through a linear combination of the kernels centered at different augmented states weighted by the observed fitness values. Without explicitly specifying the test augmented state, (6) can be rewritten in the following form:

$$\tilde{Q}^*(\cdot) = \sum_{i=1}^{n} \sigma_i k(x_i, \cdot)$$

(11)

where $a = k^{-1}q$. Since each $k(x_i, \cdot)$ in (11) effectively maps an input pattern $x_i$ (an augmented state) into a function, taking a linear combination of these kernels forms a vector space. Consequently, $\tilde{Q}^*(\cdot)$ in (11) can be treated as a vector in a vector field (of functions) with $k$ as the representer of evaluation. Specifically, $\tilde{Q}^*(\cdot)$ is a vector in Reproducing Kernel Hilbert Space (RKHS) [1] with the reproducing property: $Q^*(x) = \langle Q^*(\cdot), k(\cdot, x) \rangle$, where $\langle \cdot, \cdot \rangle$ denotes an inner product evaluated in the unique RKHS associated with the kernel $k(\cdot, \cdot)$. By virtue of this property, the known experience particles effectively establish a field through which a novel particle obtains its fitness value by comparing against all the other particles. Each comparison evaluates a particular correlation degree to a known particle. The closer the test point (i.e. new augmented state) is to a known state, the closer their fitness values are. As a consequence, the final prediction is governed by those particles in the proximity of the test point. In addition, by Mercer’s Theorem [1], any positive-definite kernel assumes an expansion of its eigenfunctions $\phi_i$ that is:

$$k(x, x_i) = \sum_{i=1}^{\infty} \lambda_i \phi_i(x) \phi_i(x_i)$$

(12)

within the scope of relatively similar decision contexts where action 2 and 7 are optimal. C is being influenced more strongly by 5, the state on the right.

Entails that using GPR with a non-degenerate kernel is equivalent to a Bayesian linear regression model using infinitely many basis functions. This effectively implies that the fitness value prediction for a new particle will be bounded by a relatively small variance (see (7)) so long as certain experience particles exist in the neighborhood such that their associated augmented states are sufficiently correlated. We are now ready for the definition of a reinforcement field.

Definition 4 (Reinforcement Field) A reinforcement field is a vector field in Hilbert space established by one or more kernels through their linear combination as a representation for the fitness function, where each of the kernels centers around a particular augmented state vector.

As suggested by (10) an action operator is a function of both the state features and action parameters. As a result, the exact form of the action operator is not determined until all the parameters in the operator are fixed. Once the state of interest is determined along with the choice of the (primitive) action $a \in A$, the action operator then acts on the state such that it resolves the stochastic effects modeled by the action parameters into a fixed action vector. This in turn leads to the next state that falls within the range stochastically governed by the action parameters. A fixed action operator is always associated with a fitness value given by (6). By this token, an action operator can be applied anywhere in the state space with an associated fitness value evaluated from the knowledge of the environment gathered so far in terms of the experience particles held in memory (i.e. the training set $\Omega$). Further, if the control policy is defined through a fitness value prediction, such as the softmax policy in (9), the action operator also represents an abstraction of a policy. This is due to the fact that the action operator encodes both a particular state transition and its fitness value once the state and action vector are known. Thus, by fixing a state, all (parameterized) action choices can be compared in a similar way through their corresponding fitness value estimates. The result of the fitness value prediction is governed by the property of the reinforcement field, which is established by all the known experience particles distributed over the state space. In this way the reinforcement field connects with the action operator, which, in turn, determines the continuous process of the policy inference. A high-level policy inference in a reinforcement field is illustrated in Fig. 2. The center circle represents the goal area. Also marked in the center circle are 12 possible motions, each with an allowable range for action variations. The outer three pairs of the shaded circles enclose the experience particles (in dots) holding control policies suggested by the associated augmented states. The new query points are marked by crosses. As long as the query point (e.g. an unexplored state) is sufficiently close to the states of the neighboring particles (i.e. within the scope of the shaded circles), the resolved action is expected to be similar (in terms of action parameters) to those in the particles. The exact inference is given in Section IV.

To sum up, a reinforcement field shares the following properties analogous to other physical fields: i) a source of origin ii) a spatial intensity and iii) a governing law that acts on particles in the field. The field source originates from experience particles, each of which embeds a control policy that generalizes into the neighboring states through the kernel.
The intensity of an experience particle is represented by the fitness value evaluated from the combined effect of all the other particles in proportion to their degree of correlation and their intensities (i.e. their fitness values). Upon any state transition, a new particle is created, encapsulating the value of the state vector, action vector, and the mapping to a fitness value. The new particle joins the rest of the population to form an updated reinforcement field that subsequently determines the fitness value of yet another new particle.

IV. POLICY INFERENCE: AN ALGORITHM

A. Particle Evolution

Because the policy learning in the reinforcement field depends on experience particles, a sample update rule is required so that the memory will not exceed a given bound. Several aspects play into the role of constructing a proper mechanism for the particle evolution. First, the new particle retained in the memory needs to reflect the latest dynamics of the system so that the policy generalized from these particles will be as accurate as possible. This can also be explained by noting that the policy is driven by the following two continuous processes in parallel: i) the kernel hyperparameter tuning by ARD and ii) the fitness value prediction by GPR. Recall from (8) that the ARD procedure identifies the hyperparameters in a manner that trades off between data fit and model complexity, leading to the best explanation of the data (from the perspective of maximizing the marginal likelihood). Thus, a periodic update using ARD is a key mechanism for the reinforcement field to keep up with the latest utility signals from the underlying control learner (to be discussed further). The process ii) above is just the flip side of the same coin in that the current fitness values referenced by the experience particles in memory effectively assimilate the update history of the utility estimates from the underlying control learner. Thus, the prediction for a new augmented state will only be as good as the hyperparameters.

On the other hand, since the goal of policy learning is to steer the agent towards a state sequence leading to a maximum utility (as a global constraint), it is imperative for the agent to retain the particles aligned with that objective. This is to say that in each step of the decision process, the action operator shall more often than not resolve to a localized policy that triggers the state transition (as a local constraint) to follow the path in the direction of increasing fitness value. Note that the policy inferred from the action operator at a given augmented state, \((s, a)\), is locally applicable to all the neighboring states as long as they are within the scope of a given correlation degree measured by the kernel. This is a property of policy inference in the reinforcement field in which any given two particles are considered similar as long as the combined effect of the state and action ends up being similar (Fig. 2). Yet from another perspective, this policy generalization property is indeed desirable because the state transition given a fixed state and action will vary over time, depending on the action parameters. The random variations in the action parameters encode the uncertainty in the action dynamics, which can either be interpreted as the uncertainty inherent in the action performance or the uncertainty from the environment. Moreover, depending on the exploration strategy of the policy search, the agent can be chosen to always follow a stochastic policy or a greedy policy after sufficient exploration of the state space. In the empirical study, we will consider a softmax exploration strategy in (9) that gradually reduces to an \(\epsilon\)-greedy policy.

B. Experience Association

In order for the agent to have a means to “interpolate” the best policy given the experience particles, we shall now introduce a central operation for the particle update – experience association. Additionally, two attendant definitions are given: i) hypothetical state and ii) particle polarity.

Definition 5 (Hypothetical State) A hypothetical state \(s'_h\) is an augmented state formulated by the agent at a given state \(s \in S \ (s=x_s)\) replicating the primitive action \(a \in A \ (a=x_a)\) from an experience particle \(a \in \Omega \) to give \(s'_h=(x_s, x_a)\).

Definition 6 (Particle Polarity) Positively-polarized particles (or positive particles) are those associated with a non-negative temporal difference \((TD \geq 0)\) whereas negative particles are associated with a negative temporal difference \((TD < 0)\).

The operation of experience association is inspired by a basic learning principle: if a decision had led to a desired result in the past (in terms of the fitness value), then the experience can be reused for similar situations. An example can be observed in Fig. 2 where the ideal action at query points A, B and C are aligned with the control policy embedded in their neighboring particles. Whether two particles are (sufficiently) correlated is again determined by the kernel. Formally speaking, the experience association operation is performed by the agent at a state \(s=x_s\) associating an instance of past memory encapsulated in a particular experience particle \(a \in \Omega\) to the particle to form a hypothetical state \(s'_h=(x_s, x_a)\) (Definition 5). The resulting new state \(s'_h\) is then compared with the particle referencing an augmented state \(s'=a=x_s\) using the given kernel \(k\); that is, by evaluating \(k(s'_h, s')\) where \(s'_h=(x_s, x_a)\) and \(s'=a=x_s\). If \(k(s'_h, s')\) goes above a prescribed threshold \(\tau\), i.e. \(k(s'_h, s') \geq \tau\), then \(s'_h\) is said to be (strongly-)correlated to the augmented state \(s'\). Thus, the threshold \(\tau\) effectively determines the size of the shaded circles in Fig. 2. By applying experience association, the agent is given a higher preference for the policy embedded in a neighboring particle.

Next, in order for the agent to distinguish between a preferable action and undesirable ones, this paper adopts a dichotomy by labeling particles in terms of the temporal difference (TD) obtained from the underlying control learner (Definition 6). Recall from (2) that a non-negative TD (i.e. \(R(s,a)+\gamma Q(s',a) \geq Q(s,a)\)) implies an improvement in the utility estimate and thus that the corresponding action is in general preferable to those with negative TDs. Thus, the particle polarity can be used as a heuristic for the particle evolution. As mentioned earlier, in order for the action operator to resolve a state transition along the direction that increases the fitness value, the older particles are to be replaced
by the new particles with relatively higher fitness values. However, care must be taken in such replacements. The new experience particle subsumes an older one only when they are sufficiently correlated. For a particle update to occur, first the polarity must match: particles with similar states (e.g. nearby positions in a navigation domain) but with an opposite polarity are indicative of a high dissimilarity in actions. This gives rise to the particle update rule in Fig. 3. Notice that the update rule for positive particles is logically opposite to the rule for negative particles. This is due to the fact that the TD values for positive particles are expected to increase over time as the policy learning proceeds, whereas the negative particles have decreasing TD values. The end result of this polarity-oriented update rule is to drive the positive particles toward a better policy while the negative particles remain as counter-examples throughout the whole learning cycle. The iteration for the particle replacement (Step 2 in Fig. 3) terminates as soon as a match is found; otherwise, no update will occur if no match is found among the current population in memory. Note that the update rule can be flexible in the sense that an exploratory strategy can be introduced to the particle update criteria (Step 2.1-2.2). Nonetheless, when the size of the training set $\Omega$ is large, a global search for a matched particle can be expensive. The computational cost can be alleviated by partitioning the state space with each partition referencing a fixed number of positive and negative particles. In this manner, only a local search is required given that the particles in the same state partition are guaranteed to be more correlated than others to the new particle referencing a state within the partition boundary.

Fig. 4 illustrates a navigation domain in which the agent aims to travel from the start position to the goal area following a shortest path. The start position is marked by a cross in the lower left square and the goal area is enclosed by the upper right square with a flag. The agent has the freedom to move in all directions within the boundary of the continuous state space (i.e. borders marked by thick lines). The allowable actions include the 12 movements parameterized by a radius and an angle increment $(\Delta r, \Delta \theta)$ adopted earlier in Fig. 1 and 2. Further, the state space is partitioned into a 4-by-4 grid for the purpose of estimating utility values associated with each state area as an approximation to a continuous-state MDP. As mentioned earlier, SARSA is used as the underlying control learner in order to obtain utilities as guiding signals for estimating a fitness value for each particle (using GPR). In this example, each state partition is configured to reference only two particles with one being positive and the other negative. Positive particles are depicted by small blue circles, each of which points to a relatively preferable direction to move. By contrast, negative particles are represented by the small red squares. To keep the figure uncluttered, only selected negative particles are marked with arrows indicating their navigational policies. Each particle represents a policy induced by the

RF-SARSA($k, M, T, \Omega_0$):

// $k$: The kernel function with hyperparameters $\theta$ (e.g. (4))
// $M$: parametric action model with $n$ choices:
\[
[a_i^{(1)} \in A \mid i = 1 \sim n]
\]
// $T$: period for ARD procedure
// $\Omega_0$: initial source of particles (or simply an empty set)
1. Evaluate the initial hyperparameters $\theta$ via ARD using $\Omega_0$ or simply start with an initial prior $\theta_0$
2. Initialize $Q(s,a^{(0)})$ arbitrarily
3. Initialize particle set (i.e. training set for GPR): $\Omega \leftarrow \Omega_0$
Repeat (for each episode)
4. Initialize $s$
5. Given a state $s$, evaluate $Q^*$ for each $a^{(i)} \in A$; i.e.
\[
\text{foreach } a^{(i)} \in A: \\
\quad 5.1 \quad a^{(i)} \text{ resolves into the value } x_i \text{ with } s \leftarrow x_i \\
\quad \text{according to the yield function model}
\]
5.2 Evaluate $Q'(x_i, x_i')$ using GPR (see (3),(6))
6. Choose $a^{(i)}$ from $s$ following the current policy:
\[
\pi(s,a^{(i)}) = \pi(Q^*) \text{ (e.g. softmax in (6)) using } Q'
\]
values from step 5
Repeat (for each step of episode):
7. Run ARD only every $T$ steps
\quad 7.1 (Optional) $T \leftarrow T + 1$ // gradually increase $T$
8. From $s$ take action $a^{(i)}$, and observe $r$ and $s'$
9. Generate a new particle $p$ referencing the augmented state $(x_i, x_i')$ obtained from step 8
10. Run ParticleUpdate($p$) // add $p$ to $\Omega$ or discard $p$
11. Choose $a^{(i)}$ from $s'$ using $\pi(Q^*)$ (see step 5 and 6)
12. Update $Q(s,a^{(i)})$ using Equation (2).
13. $s \leftarrow s'$; $a^{(i)} \leftarrow a^{(i)}$
Until $s$ is terminal
associated augmented state and its fitness value. Through the property of the kernel with (11) and the GPR prediction in (6), all the red and blue particles combined establish a particular reinforcement field, representing a policy generalization mechanism surrounding the neighboring state of the particles. Note that two conditions limit the range of the motion given an action choice: i) allowable range in action parameters ii) the state space boundary. Recall that an action operator effectively incorporates both the state and action information and always corresponds to a fitness value given all the parameters are fixed. Thus, during the course of navigation, if the action operator resolves into a position and a motion that leads outside of the state boundary, then the action will only be executed up to the boundary; that is, the agent will move as far as possible before hitting the border and remain there until the next decision step. This is illustrated by the dotted circle enclosing two example negative particles (in red squares), each of which embeds a motion that eventually hits the border. Illustrated in Fig. 5 is an example algorithm in the reinforcement field using SARSA as the baseline learner. The policy learning method using the property of the reinforcement field is prefixed by RF-, giving rise to the name RF-SASAR.

V. EMPIRICAL STUDY

This section demonstrates RF-SARSA by reusing the navigation domain in Fig. 4 with a slightly different configuration. In the experiment, the state space is partitioned into 25 evenly-spaced grids and each partition is configured to reference 3 positive and 3 negative particles, leading to a maximum of 150 particles to maintain in the memory. Since the goal is for the agent to learn the policy following a shortest path, a cost (-1) is imposed per step of travel whereas a reward (+100) is given upon reaching the goal area. Assuming that each side of the environment is of 5 unit length, the action parameter $\Delta r$ is set to have a target value of 1.0 unit length but varies between -0.8 to +1.2 unit length modulated by a Gaussian noise, applicable to all action choices. The angular parameter $\Delta \theta$ takes on 12 equal partitions around the circle (Fig. 1, 2) resulting in 12 different target angles for the actions, each of which has ±15 degrees of tolerable error. Following a shortest path to the goal area will obtain a reward of approximately 94. Fig. 6 shows the performance of RF-SARSA with the SE kernel in (4). Using the product kernel in (5) leads to a similar learning curve. The comparison for the policy generalization in different kernels, however, requires a careful analysis in their “field plots” (e.g. Fig. 4). This is to be illustrated in the future work. Note that standard SARSA can be considered a special case of RF-SARSA by removing the parametric action model and the extra layer of fitness value estimation, thereby reducing the action operator to an ordinary action set with predefined behaviors. Nevertheless, doing so effectively drops the policy generalization capacity and more often than not, more features need to be engineered into the state space to cope with potential complications from action dynamics, leading to an exponential growth in the state space.

VI. CONCLUSION AND FUTURE WORK

The reinforcement field enables policy generalization through properties of the kernel as a correlation hypothesis that associates the policy-embedded particles distributed across the state space. Each particle effectively holds a representative policy that generalizes into its neighboring state and thus, all particles collectively form a field (see (11)). The reinforcement field encodes the knowledge of the environment gathered so far, reflecting the progressive advancements in policy search. Through the kernel working with GPR along with the ARD procedure, the agent’s worldview continues on updating, which in turn loops back to adjust the control policy reflecting the shift in the environment dynamics. In addition, an action operator is defined that integrates the local dynamics of state transitions with the (fitness) value prediction that drives the policy learning. In this manner, the agent will always favor a state transition that moves towards a higher utility, eventually leading to a state sequence leading to a maximum accumulated reward – the global objective of the policy learning. In related work in parallel to this formulation we are investigating a further generalization in terms of action abstractions. The goal is to identify similar decision experiences in terms of the similarity in their augmented state representation. Similar augmented states are grouped together by taking into account their corresponding fitness values. Thus, similar control polices are clustered to formulate an action-oriented conceptual model such that the agent can derive policies at the level of abstractions to reduce the decision points.

REFERENCES