An Anti-Hebbian Learning Rule to Represent Drive Motivations for Reinforcement Learning

Varun Raj Kompella, Sohrob Kazerounian, and Jürgen Schmidhuber

IDSIA, Galleria 2, Manno-Lugano, Switzerland
{varun,sohrob,juergen}@idsia.ch

Abstract. We present a motivational system for an agent undergoing reinforcement learning (RL), which enables it to balance multiple drives, each of which is satiated by different types of stimuli. Inspired by drive reduction theory, it uses Minor Component Analysis (MCA) to model the agent's internal drive state, and modulates incoming stimuli on the basis of how strongly the stimulus satiates the currently active drive. The agent's dynamic policy continually changes through least-squares temporal difference updates. It automatically seeks stimuli that first satiate the most active internal drives, then the next most active drives, etc. We prove that our algorithm is stable under certain conditions. Experimental results illustrate its behavior.

Keywords: Motivational Drives, Reinforcement Learning, MCA, Animats

1 Introduction

Reinforcement Learning (RL) methods [1] have proven quite powerful in endowing agents with the ability to learn to achieve goals across a wide variety of settings. Typically however, within any given setting, the agent lacks a motivational system that would allow it to differentially value various types of simultaneous (possibly conflicting) goals and actions [2, 3]. We introduce a novel method inspired by drive theory [4–6], which enables an agent to learn from multiple types of rewarding stimuli [7, 8], even as its preferences for those stimuli change over time. Importantly, this method achieves behavioral success under these conditions, while learning online.

Although definitions of drive and motivation abound across a number of interdisciplinary fields, including psychology, neuroscience, and artificial intelligence, we follow that of Woodworth [4], who suggested hunger and thirst as prototypical bodily drives. As the levels of hunger and thirst change in time, an agent is motivated to initiate behaviors that satisfy one or the other drive. There are two primary methods by which such drives are typically represented in the artificial intelligence literature: The first makes use of homeostatic drive regulation, wherein actions that push a physiological state variable towards its equilibrium are rewarded, while actions that push the state variable away from that equilibrium are punished [9, 3]. The second, following Hull [10, 2] makes use of drive states that vary from "fully satiated", to "fully unsatiated", with actions that satiate active drives being rewarded.

Hullian and homeostatic drive reduction are highly dependent on physiological parameters however, and are therefore not always ideal in modeling robotic agents. Rather

than explicitly model time-varying drive states and the changes to those drive levels resulting from various types of rewarding behavior, our method instead attempts to *balance* the various types of rewarding stimuli an agent has received. Doing so allows an artificial or robotic agent to successfully modulate its behavior in response to active drives, without making its successful behavior dependent on careful parameter selection. In this system, an agent's drive towards a particular stimulus depends, in part, on how much of that stimulus it has acquired in its recent history, weighed against how much it desires alternative stimuli. When the agent receives one type of rewarding stimulus, its drive for that particular stimulus should decrease, while its drive for other types of stimuli should increase. One elegant method for modeling this input-dependent drive switching, is to note that as the agent experiences a changing distribution in its input stimuli, estimating the covariance of this distribution yields a minor component (MC; [11]) which points in the direction of the least received stimuli. In order to compute the minor component, we use Peng's Minor Component Analysis (MCA; [12]) algorithm, which uses a low-complexity, online, anti-hebbian updating rule, making it suitable for open-ended learning. Moreover, such a representation system, allows us to incorporate intrinsically rewarding behaviors as just another drive of the agent. As discussed by White [13], there is no simple way to reconcile curiosity driven behaviors, with drive reduction theory. As we show in simulations however, it is rather simple to do with an MCA based drive representation.

On its own, this enables an agent to represent drives. It does not however, explain how an agent can learn which actions bring about the desired types of input stimuli. To this end, we propose an online, model-based least-squares policy iteration technique, called MCA-PI, to combine our MCA based drive-representation and action selection for a simulated robotic agent. We prove that MCA-PI is stable under certain conditions and present experimental results to demonstrate its performance.

The rest of the paper is organized as follows: Sec. 2 presents details of representing drives using MCA. Sec. 3 discusses our MCA-PI algorithm and an analysis of its dynamical behavior. Sec. 4 presents experimental results and Sec. 6 concludes.

2 Representing Drives with MCA

We present a method to represent drive motivations using Minor Component Analysis (MCA):

(a) Input Stimuli Vector: The input stimulus is an n-dimensional real-valued vector $\boldsymbol{\xi} = [\xi_1, ..., \xi_n]^T$, where each dimension represents a particular type of stimulus. For example, let food (ξ_1; 🍕) and water (ξ_2; 🧴) be two types of stimuli for an agent. When the agent receives only $\xi_1 = 2$ units of food, the corresponding input stimulus vector is $\boldsymbol{\xi} = [2, 0]$.

(d) Stimulus Priority Vector: A stimulus priority vector $\boldsymbol{\rho} = [\rho_1, ..., \rho_n]$, $\rho_i \in (0, 1]$ determines a priority weighting for each stimulus type. A high-value of ρ_i indicates that the agent takes longer time to satiate stimulus ξ_i.

(b) Drive Vector: The agent's drive at any time t is represented by an n-dimensional unit-vector $D(t) = [d_1(t), ..., d_n(t)]^T$, where each dimension $d_i(t) \in [0, 1]$ represents

an individual drive component for the stimulus ξ_i. A high value of $d_i(t)$ indicates that the agent desires the corresponding stimulus ξ_i.

(c) Drive-Vector Update: The drive vector is updated incrementally using Peng's MCA learning rule:

$$D(t) = (1-\eta)\ D(t-1)\ -\ \eta\ (D(t-1)\cdot \Lambda\boldsymbol{\xi}(t))\,\Lambda\boldsymbol{\xi}(t) \tag{1}$$

$$D(t) \leftarrow D(t)/\|D(t)\| \tag{2}$$

where η is a constant learning rate, Λ is a $n \times n$ diagonal matrix with ρ_i^{-1} as its entries. The normalization step in (2) is required to make it adaptive to non-stationary input data [14].

(e) Scalar Reward: The scalar reward $r(t)$ given to the agent is computed by projecting the input stimulus on to the current drive vector:

$$r(t) = D(t)\cdot \Lambda\boldsymbol{\xi}(t) \tag{3}$$

The agent gets higher scalar rewards if it receives a stimulus-vector $\boldsymbol{\xi}(t)$ whose direction-cosine (DC) w.r.t $D(t)$ is close to one. That is to say, the more closely the stimulus vector matches the drive vector, the more rewarding that stimulus will be. The agent, driven by higher rewards r, will be motivated to visit and then remain in the places where it is getting the currently rewarding stimuli. However, as the agent continues to remain in those places, the recent history of the MCA comes to be dominated by samples of the current stimulus distribution, which drives the minor component away from the current drive direction (see Figure 1). As a consequence, the longer an agent continues to receive the same stimulus, the less and less rewarding it becomes (i.e., the agent becomes *satiated*)

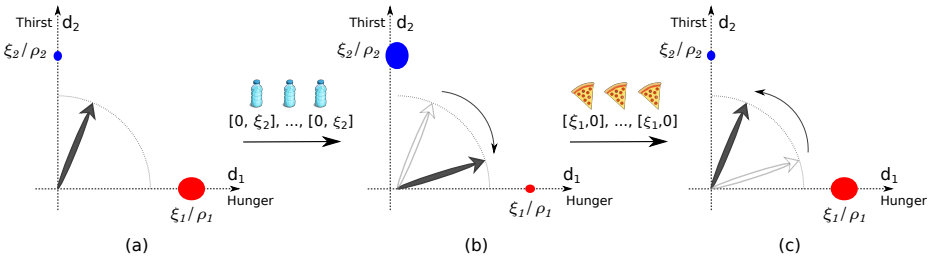

Fig. 1. The MCA drive-vector $(D(t)$, indicated by the bold arrow) points to the direction of stimuli that it received least in its recent history (anti-hebbian like behavior). (a) $D(t)$ at some arbitrary time t. (b) When the agent receives water stimulus (ξ_2), $d_2(t)$ decreases and $d_1(t)$ increases, therefore $D(t)$ slowly turns toward the "hunger" drive-direction. (c) Similarly, when the agent receives food, the vector slowly turns toward the "thirst" drive-direction.

Algorithm 1: MCA-PI $(\mathcal{S}, \mathcal{A}, \mathcal{P})$

// Ξ : Stimulus function $(|\mathcal{S}||\mathcal{A}| \times n)$ matrix
// $\mathbf{R}$: Reward function $(|\mathcal{S}||\mathcal{A}| \times 1)$ vector
// $\phi^{\mathcal{S}\times\mathcal{A}}$: State-Action basis function
// $\mathbf{D}$: MCA drive vector
// Λ : Diagonal matrix with $\{\rho_1^{-1}, ..., \rho_n^{-1}\}$ entries

for $t \leftarrow 0$ *to* ∞ **do**
  $s_t \leftarrow$ current state
  $a_t \leftarrow$ action selected by policy π_t in state s_t
  Take action a_t, observe next state s_{t+1} and stimulus $\boldsymbol{\xi}(t+1)$
  //Update Stimulus Function
  $\Xi(s_t, a_t) \leftarrow \Xi(s_t, a_t) + \eta_{t+1}^{\text{stim}}\big(\boldsymbol{\xi}(t+1) - \Xi(s_t, a_t)\big)$
  //Update MCA Drive Vector
  $D \leftarrow (1-\eta)\ D\ -\ \eta\ (D \cdot \Lambda\boldsymbol{\xi}(t+1))\,\Lambda\boldsymbol{\xi}(t+1)$
  $D \leftarrow D/\|D\|$
  //Update Reward Function
  $R \leftarrow |\Xi\Lambda D|$
  // Update Policy
  $\pi_{t+1} \leftarrow$ LSTDq-Model-Update($\phi^{\mathcal{S}\times\mathcal{A}}, \mathcal{P}, R, \gamma, \pi_t$)
end

3 Action Selection: MCA-based Policy Iteration (MCA-PI)

Unlike previous implementations that represent drives independently in an RL framework [2, 15], an MCA-based drive representation takes all drives into consideration and computes a resultant unit drive-vector $D(t)$. $D(t)$ does not necessarily indicate the level of satiation, instead it optimally points in the direction of stimuli that the agent received least in its recent history. Based on the drive at time t, the agent needs to shape its behavior to acquire the least received stimuli. However, to learn an optimal behavior (policy), in principle, one needs to take into account the internal drive vector components $(d_1, ..., d_n)$ as a part of the agent's state-space, along with the external world state. This makes the resultant state space large - exponential in the number of drive components. Konidaris and Barto [2] have used a multi-goal RL approach with SARSA(0) [16] instead, to learn a composite value function for action selection. In a similar way, the MCA drive vector can be combined with SARSA(0), where each drive-component $d_i(t)$ corresponds to a particular goal. However, a drawback of this approach is that since the internal drive-function changes quite quickly over time, the resulting decision process is *non-Markovian*. Therefore, single-step on-policy SARSA(0) algorithm that requires a decaying exploration rate for optimal performance [17], may not converge to an optimal policy. This problem can be overcome if a transition model of the external world environment is known.

Algorithm 1 shows the pseudocode of MCA-PI algorithm. Given a transition model for the external world $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$, for each time-step the algorithm incrementally updates its estimate of the stimulus function Ξ (a matrix of size $(|\mathcal{S}||\mathcal{A}| \times n)$), MCA-drive vector D, and the reward function R (a vector of size $|\mathcal{S}||\mathcal{A}| \times 1$). It then evaluates the current policy for the new reward function R using simulated samples (s, a) from $\mathcal{P}$, and generates a policy for the next time-step based on the updated value-function.

3.1 Dynamical Analysis

In this section, we study the dynamical behavior of MCA-PI algorithm. The main goal here is to show that the algorithm makes the agent balance between multiple drives in an uniform manner.

Outline: We first define *policy-sets*, such that for any arbitrary trajectory of policies within each set, the reward function converges to a unique fixed point (Definition 1). We then show that the policy-sets are non-empty in Theorem 1. Since MCA-PI is an approximate policy-iteration technique, we show in Theorem 2 that the error between the approximate value-function and the true-value function is bounded. Finally, in Theorem 3 we show that the sequence of policies generated by the MCA-PI algorithm, shifts between the policy-sets in a cyclical manner.

Conditions: The following conditions are necessary for the rest of the analysis:
(1) The learning rate of MCA satisfies: $\eta\lambda_1 < 0.5,\ 0 < \eta \leq 0.5$, where λ_1 is the largest eigenvalue of the expected covariance matrix C (=$E[\boldsymbol{\xi}\boldsymbol{\xi}^T]$) of the input stimulus data ($\boldsymbol{\xi} \in \mathbb{R}^n$).
(2) C is a symmetric nonnegative definite matrix. This condition is initially met by the agent's exploration using Gaussian *optimistic-initialization* [18], and later by the algorithm switching dynamics.
(3) The columns of stimulus-function matrix Ξ are not all-ones or a constant multiple of all-ones vector[1]. This condition, which is trivial, says that the agent does not receive equal amounts (or zero) of a particular stimulus at all world states. In which case, there is no planning required for that stimulus and the drive-vector dimension can be reduced to $(n - 1)$.

Using Condition (2), C can be factorized into VLV^{-1}, where V is the eigenvector matrix (columns representing unit-eigenvectors v_i) and L is a diagonal matrix with corresponding eigenvalues (λ_i). In addition, the eigenvectors $\{v_i | i = 1, 2, ..., n\}$ (sorted according to $\lambda_1 > \lambda_2 ... > \lambda_n$) form an orthonormal basis spanning $\mathbb{R}^n$, where v_1 is the principal-component and v_n is the minor-component. Therefore, the drive vector $D \in \mathbb{R}^n$ can be represented as a linear combination of the basis-vectors $D(t) = \sum_{i=1}^{n} w_i(t)v_i$, where $w_i(t)$ are some coefficients. Lemma 1 shows that the drive vector $D(t)$ converges to the component with the least-eigen value v_n (minor-component).

[1] All-ones vector is a vector where every element is 1.

Lemma 1 *If Conditions (1)&(2) are satisfied, the following limits of the coefficients w_i hold true:*

$$\lim_{t\to\infty} w_i(t) = 0,\ \forall i \in 1, ..., n-1 \quad \textit{and} \quad \lim_{t\to\infty} w_n(t) = 1$$

Proof. The proof follows from Lemma 3 in [14]. □

Definition 1 *Let $\Pi^i, \forall i \in \{1, ..., n\}$ denote sets of stationary policies ($\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$) defined over the underlying irreducible Markov chain, s.t., for any trajectory $\zeta = \{\pi_t \in \Pi^i, \forall t > 0\}$,*

$$\lim_{t\to\infty} R^{\zeta}(t) = \Xi \Lambda v_i \tag{4}$$

Theorem 1 *If condition (3) is satisfied, $\Pi^i, \forall i \in \{1, ..., n\}$ are non-empty disjoint sets.*

Proof. Since, Ξ does not contain scalar multiples of all-ones column vectors (Condition (3) and Lemma 1), there exists at least one trajectory ζ of policies where $\mathbb{E}[\boldsymbol{\xi}\boldsymbol{\xi}^T|\pi; \forall \pi \in \zeta]$ has the minor-component v_i. Since, each $v_i, \forall i \in \{1, ..., n\}$ are orthonormal, the policy-sets are disjoint. □

Theorem 2 *At any time t, let $\pi_t, \pi_{t+1}, ..., \pi_{t+m}$ be any arbitrary sequence of policies, such that, $\pi_m \in \Pi^i, \forall m = t, t+1, ...,$. Let $\widetilde{Q}^{\pi_t}, \widetilde{Q}^{\pi_{t+1}}, ..., \widetilde{Q}^{\pi_{t+m}}$ be the corresponding sequence of approximate value functions as computed by LSTDQ. Then, there exists a positive scalar δ that bounds the errors between the approximate and the true value functions over all iterations:*

$$\|\widetilde{Q}^{\pi_m} - Q^{\pi_m}\|_\infty \leq \delta, \quad \pi_m \in \Pi^i, \forall m = t, t+1, ...,$$

Proof. The Least-Squares fixed-point approximation [19] of the value function for a stationary transition model P and reward function R can be written as $\widehat{Q}^{\pi} = A^{\pi}\phi^T R$, where $A^{\pi} = \phi\left(\phi^T(\phi - \gamma P^{\pi}\phi)\right)^{-1}$. For a given stationary reward function R, there exists a positive scalar ϵ that bounds the error between the least-squares fixed-point approximation and the true value function :

$$\|\widehat{Q}^{\pi_m} - Q^{\pi_m}\|_\infty \leq \epsilon, \quad \forall m = t, t+1, ..., \tag{5}$$

From Theorem 1, we have since $\pi_m \in \Pi^i, \forall m = t, t+1, ...,$ there exists a fixed point R^* to the sequence of reward functions. Therefore, $\|R_m - R^*\|_\infty \leq \nu \implies \|A^{\pi_m}\phi^T R_m - A^{\pi_m}\phi^T R^*\|_\infty \leq \nu \implies \|\widetilde{Q}^{\pi_m} - \widehat{Q}^{\pi_m}\|_\infty \leq \nu$. Using (5) and *triangle-inequality* we get, $\|\widetilde{Q}^{\pi_m} - Q^{\pi_m}\|_\infty \leq \epsilon + \nu\ (\equiv \delta), \quad \pi_m \in \Pi^i, \forall m = t, t+1, ...,$ □

Theorem 3 *Let $\pi_t, \pi_{t+1}, \pi_{t+2}, ...$ be a sequence of policies generated by the algorithm at any arbitrary time t. Then, $\exists N \in \mathbb{N}^+$, such that:*

$$\{\pi_t, \pi_{t+1}, ..., \pi_{t+N}\} \subseteq \Pi^i, \quad \{\pi_{t+N+1}, \pi_{t+N+2}, ...\} \subseteq \Pi^j, \quad j \neq i, i \in \{1, ..., n\}$$

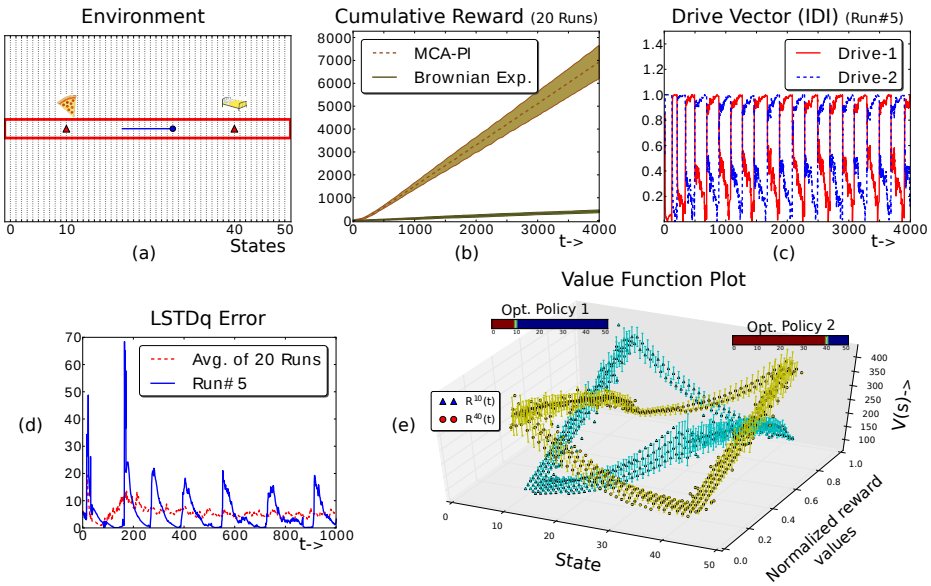

Fig. 2. Figure best viewed in color. See text for accompanying details.

Proof. Since, MCA-PI is an approximate policy iteration algorithm, it follows from Theorem 2 that the generic bound on policy iteration applies (See Theorem 7.1 [19]). Therefore, the value-function converges toward the true value function Q^* corresponding to the current reward function $(\Xi \Lambda v_n)$. However, a policy $\pi^*_{v_n}$ based on Q^* maximises expected cumulative rewards, i.e. $\mathbb{E}[\gamma^t \Xi \Lambda v_n]$, which is maximal only when the principal component of $\mathbb{E}[\boldsymbol{\xi}\boldsymbol{\xi}^T|\pi^*_{v_n}]$ is equal to v_n. Since MCA by definition computes the minor component, it follows that $\pi^*_{v_n} \in \Pi^j, j \neq i$. This implies that there exists a positive integer N, where the policies $\{\pi_{t+N+1}, \pi_{t+N+2}, ...\} \in \Pi^j$. □

Theorems 1, 2 and 3 explain the switching dynamics of the algorithm between multiple drives.

4 Experimental Results

4.1 Experiment 1: Hallway

We evaluate our algorithm here on a classic example of a 50 state closed Markov Chain (Fig. 2(a); [19]). The agent at each state can take three deterministic actions: *left*, *right* and *stay*, except for the boundary states 0 (*right* and *stay*) and 49 (*left* and *stay*). At state 10, the agent receives a stimulus (food) $\xi_1 = [6\ 0]^T + \mathcal{N}(\mu = 0, \sigma = 2)$ and at state 40, it receives a stimulus (rest) $\xi_2 = [0\ 6]^T + \mathcal{N}(\mu = 0, \sigma = 2)$. $\mathcal{N}(\mu, \sigma)$ represents an additive Gaussian noise with mean μ and standard deviation σ. The agent has equal priority towards both stimuli ($\rho = [1, 1]$).

We use a constant learning rate (η = 0.01) for the MCA update, a discount factor $\gamma = 0.95$, and indicator basis functions ($\phi^{\mathcal{S}\times\mathcal{A}}$) to represent each state. An initial policy (π_0) and a drive-vector (D) are set arbitrarily. The approximate stimulus function Ξ is *optimistically-initialized* [18] to $\mathcal{N}(\mu = 0, \sigma = 0.3)$ for all states and dimensions.

As a baseline comparison, we evaluate the performance of an agent carrying out brownian exploration vs. an agent carrying out MCA-PI. The agent using brownian exploration would also continually visit both the stimulus sources over time. Figure 2(b) shows the cumulative reward with respect to time, averaged over 20 trials for the two models (shaded region represents the standard deviation). It is clear from the figure that the MCA-PI approach results in a deliberative behavior in comparison with the standard brownian motion (with a mean velocity equal to 4 states/timestep). Figure 2(c) shows the changing MCA drive-vector over time for a single run. The drive-vector periodically switches between $v_2 \rightarrow v_1 \rightarrow v_2$, as derived in Theorem 3. The LSTDq error $\|\widehat{Q}^{\pi_t} - \widehat{Q}^{\pi_{t+1}}\|$ peaks momentarily (see Fig. 2(d)) whenever the agent switches between the policy sets Π^i of one stimulus, to the other. However, the error drops down quickly thereby resulting in the next switch. The red dashed line indicates the average error for 20 different runs. Figure 2(e) shows the variation of the state-values (mean and the standard deviation over 20 runs) with the changing reward values at states 10 (blue triangles and cyan error bars) and 40 (red circles and yellow error bars). It is clear from the figure that the value of state 10 is higher whenever the projection $\|r\| = \|\Xi \cdot \Lambda v_2\|$ is higher, which makes the agent shift to state 10. It stays there until the drive reduces and the value of state 40 becomes higher. The constant switching between the sources balance both drives in an uniform manner. The video for the experiment can be found at URL: `http://www.youtube.com/watch?v=Mk_wyJ8mQcU`

4.2 Experiment 2: Three Room Maze

Here, we evaluate MCA-PI on a larger discrete-state three room maze environment. There are in total 200 reachable states and two door-ways as shown in Figure 3(a). The agent can take 5 deterministic actions: *left*, *right*, *north*, *south* and *stay*, except at the states next to the room boundary. Each room has a distinct stimulus source placed arbitrarily. The agent has equal priority towards all the stimuli ($\rho = [1, 1, 1]$).

We use a constant learning rate (η = 0.01) for the MCA update, a discount factor $\gamma = 0.85$, and 30 Laplacian eigen-map features (Proto Value Functions; [20]) as basis functions ($\phi^{\mathcal{S}\times\mathcal{A}}$) to represent each state. An initial policy (π_0) and a drive-vector (D) are set arbitrarily. The approximate stimulus function Ξ is optimistically initialized to $\mathcal{N}(\mu = 0, \sigma = 0.3)$ for all states and dimensions.

Figure 3(b) shows the cumulative stimulus over time for 20 runs of the experiment. We see that the agent accumulates each of the sources nearly equally. The drive vector D switches periodically between v_1, v_2 and v_3 (Fig. 3(c)), and is stable with a low LSTDq error over time (Fig. 3(d)). It can be seen in Figure 3(c) that the switching starts after a delay $t \gtrapprox 1000$ time-steps. This is due to the updating estimate of the stimulus function (Ξ) upon exploring via optimistic initialization. Figure 3(e)-(g) shows three sets of value functions and corresponding policy plots at different time instants, directing the agent to each of the stimulus sources. The video for the experiment can be found at URL: `http://www.youtube.com/watch?v=ZbvSSmZrOzc`

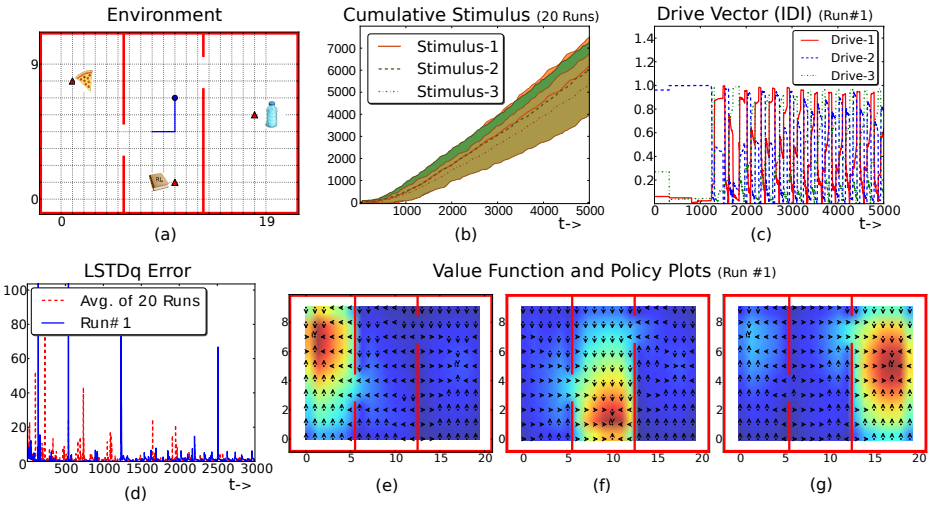

Fig. 3. Figure best viewed in color. See text for accompanying details.

4.3 Experiment 3: Extrinsically and Intrinsically Motivated Agent

In this experiment, we consider an agent that is both extrinsically and intrinsically motivated. Intrinsically motivated (curious) agents not only focus on potentially externally posed tasks, but also creatively invent self-generated tasks that have the property of currently being still unsolvable but easily learnable. The theory of Artificial Curiosity (AC; [21]) introduces a mathematical formalism for describing curiosity and creativity in artificial agents. A creative agent needs two learning components: an adaptive encoder of the growing history of observations and a reinforcement learner. The *learning progress* of the encoder becomes a *curiosity* reward for the reinforcement learner. For the sake of consistency as well as simplicity, we use of another instance of the MCA algorithm coupled with Robust Online Clustering (ROC; [22]) an as an encoder[2]. We refer to it as intMCA for the rest of the paper.

The agent is in an environment with two extrinsic stimulus sources $\{\xi_1 =$*food*, $\xi_2 =$*bed*$\}$ and two different learnable signal sources (represented by a *book* and *music* as shown in Figure 4(a)) that constitute two sources for curiosity-stimulus (ξ_3). When the agent is at the states corresponding to *book* and *music*, it receives a 2-dimensional input signal $\mathbf{x}^{\text{book}}$ (Figure 4(b)) and $\mathbf{x}^{\text{music}}$ (Figure 4(c)) given by:

$$\mathbf{x}^{\text{book}} : \begin{cases} x_1(t) = \sin(t) + \cos(11t)^2 \\ x_2(t) = \cos(11t) \end{cases}, \mathbf{x}^{\text{music}} : \begin{cases} x_1(t) = \sin(2t + \frac{\pi}{3}) - \cos(11t)^2 \\ x_2(t) = \cos(11t) \end{cases} \quad (6)$$

These signals are expanded into a five-dimensional polynomial space ($[x_1, x_2, x_1^2, x_2^2, x_1x_2]$) and normalized (*whitened*) to have unit-variance. MCA when applied to the

[2] Note that this implementation is not strictly limited to the MCA and can easily be replaced with any other adaptive learning machine.

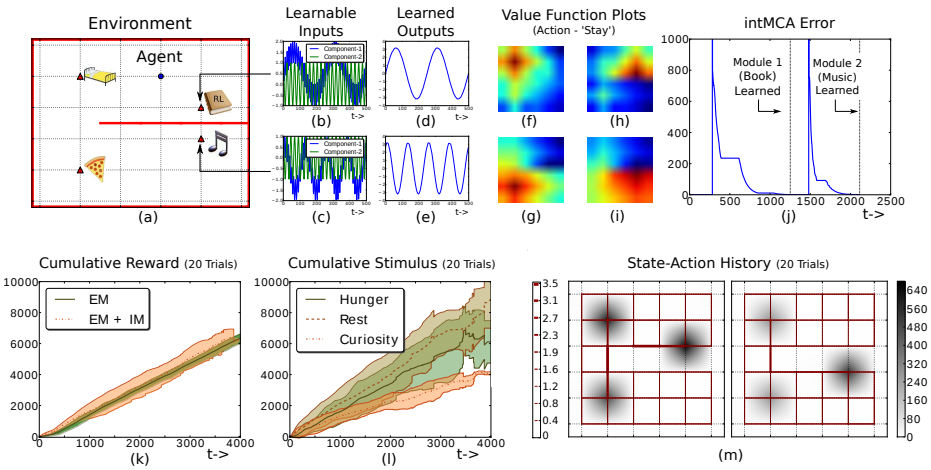

Fig. 4. Figure best viewed in color. See text for accompanying details.

derivative of the normalized signals (approximated by backward-difference), learns the underlying slowly changing driving forces [23], which are $\sin(t)$ (Figure 4(d)) and $\sin(2\ t + \pi/3)$ (Figure 4(e)). When the intMCA feature outputs become stable, the intMCA-error (ε) decreases. This decrease results in a proportional curiosity stimulus:

$$\xi_3 = \text{Clip}(-\dot{\varepsilon}, 0, 12) \tag{7}$$

ξ_3 is clipped to lie in the range $[0, 12]$ to keep it bounded and comparable with the other stimuli (ξ_1, ξ_2). The agent's drive-vector D is a 3-dimensional vector representing *hunger*, *rest* and *curiosity* drives. The agent has equal priority towards all the stimuli ($\rho = [1, 1, 1]$). The approximate stimulus function Ξ is optimistically initialized to $\mathcal{N}(\mu = 0, \sigma = 0.3)$ for all states and dimensions. The agent can take 5 deterministic actions: *left*, *right*, *north*, *south* and *stay*, except at the states next to the room boundary.

Similar to the earlier experiments, the agent quickly learns the model for the stimulus function Ξ. However, in this case it is non-stationary since ξ_3 vanishes when the learning of the intMCA completes (Eq. (7)). The initial behavior of the agent is similar to the extrinsically motivated agent, which sequentially switches between the states corresponding to the stimuli. However, since ξ_3 decreases, the agent continues to seek the stimulus ξ_3 further. This allows it to completely learn the signal. Once the error ε drops down close to zero, intMCA module is saved for future-use and a new intMCA is created. The agent is again initialized with optimistic values to allow it to explore. Now since, the agent no longer receives any curiosity stimulus ξ_3 at the state where the earlier intMCA module was learned, it goes to the other signal source to get ξ_3. Figures 4(f)-(i) show localized state-value functions learned for *stay* action. Figure 4(j) shows the estimation error plot over execution time showing two decaying peaks for each of the signal sources. Figure 4(k) shows cumulative reward averaged over 20 trials (shaded region represents the standard deviation) for the EM+IM agent and the EM-only agent from the earlier experiment. From the plot it is clear that the method works similarly to

the EM-only experiment. Figure 4(l) shows individual cumulative stimulus components averaged over 20 trials. The curiosity stimulus is lower compared to the other stimuli. This is because of its vanishing nature. Figure 4(m) shows the state-action history for each module learned averaged over 20 Trials. The video for the experiment can be found at URL: `http://www.youtube.com/watch?v=cqvw-MxZkOA`

5 Discussion

We showed MCA-PI's performance on a simple and a relatively large discrete state-space maze environment. MCA-PI has a computational complexity of $O(k^2)$ where k represents the number of basis-functions used however, it is sample efficient. The algorithm can be applied to much larger discrete state and continuous domains by either using factored MDP approaches or continuous extensions of Laplacian methods [20]. At a qualitative level, these simulations show that the behavior of an agent undergoing MCA-PI mirrors the behavior one expects from appropriately constructed Hullian, or homeostatic drive reduction based agent. In particular, we desire an agent to be able to seek out stimuli that satiate its currently active drives, and to switch to behaviors that seek out new stimuli when satisfied with prior ones. Unlike Hullian and homeostatic systems however, our model achieves this without explicit need for modeling and parameterizing individual time-varying drive states, which makes it a more elegant solution in situations wherein the agent only needs to balance its behaviors, rather than maintain pre-defined physiological state variables.

6 Conclusions

The canonical RL literature tends to ignore that robotic agents and animats operating in real-time, complex, and changing environments typically have to monitor several continuous, time-varying reward types in an online fashion. While some methods have attempted to address this by developing motivational frameworks which make use of Hullian drives, or homeostatic drive theory, these methods tend to focus on physiological state variables as found in biological agents. Instead, we present a method that is motivated by drive theory, but which represents an agent's drive by means of Minor Component Analysis. Doing so enables an agent to balance between competing drives in a manner which doesn't depend on physiological parameters, but rather, the relative levels of the various rewarding stimuli it seeks.

Acknowledgments. This work was funded through SNF grant #138219 (Theory and Practice of Reinforcement Learning II) and #270247 (NeuralDynamics project).

References

1. R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. Cambridge Univ Press, 1998.

2. G. D. Konidaris and A. G. Barto. An adaptive robot motivational system. In *From Animals to Animats 9*, pages 346–356. Springer, 2006.
3. I. Cos, L. Cañamero, G. M. Hayes, and A. Gillies. Hedonic value: enhancing adaptation for motivated agents. *Adaptive Behavior*, 21(6):465–483, 2013.
4. R. S. Woodworth. *Dynamic psychology, by Robert Sessions Woodworth*. Columbia University Press, 1918.
5. C. L. Hull. *Principles of behavior: an introduction to behavior theory*. Century psychology series. D. Appleton-Century Company, incorporated, 1943.
6. J. Wolpe. Need-reduction, drive-reduction, and reinforcement: a neurophysiological view. *Psychological review*, 57(1):19, 1950.
7. L. Barrett and S. Narayanan. Learning all optimal policies with multiple criteria. In *Proceedings of the 25th international conference on Machine learning*, pages 41–47. ACM, 2008.
8. P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, and E. Dekker. Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine learning*, 84(1):51–80, 2011.
9. M. Keramati and B. S. Gutkin. A reinforcement learning theory for homeostatic regulation. In J. Shawe-Taylor, R.S. Zemel, P. Bartlett, F.C.N. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 82–90. 2011.
10. G. D. Konidaris and G. M. Hayes. An architecture for behavior-based reinforcement learning. *Adaptive Behavior*, 13(1):5–32, 2005.
11. E. Oja. Principal components, minor components, and linear neural networks. *Neural Networks*, 5(6):927–935, 1992.
12. D. Peng, Z. Yi, and W. Luo. Convergence analysis of a simple minor component analysis algorithm. *Neural Networks*, 20(7):842–850, 2007.
13. R. W. White. Motivation reconsidered: the concept of competence. *Psychological review*, 66(5):297, 1959.
14. M. Luciw, V. R. Kompella, S. Kazerounian, and J. Schmidhuber. An intrinsic value system for developing multiple invariant representations with incremental slowness learning. *Frontiers in Neurorobotics*, 7, 2013.
15. E. Shirinov and M. V. Butz. Distinction between types of motivations: Emergent behavior with a neural, model-based reinforcement learning system. In *Artificial Life, 2009. ALife'09. IEEE Symposium on*, pages 69–76. IEEE, 2009.
16. N. Sprague and D. Ballard. Multiple-goal reinforcement learning with modular sarsa (0). In *IJCAI*, pages 1445–1447, 2003.
17. S. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 38(3):287–308, 2000.
18. R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. Cambridge, MA, MIT Press, 1998.
19. M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *The Journal of Machine Learning Research*, 4:1107–1149, 2003.
20. S. Mahadevan and M. Maggioni. Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *Journal of Machine Learning Research*, 8(2169-2231):16, 2007.
21. J. Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010.
22. I. D. Guedalia, M. London, and M. Werman. An on-line agglomerative clustering method for nonstationary data. *Neural Computation*, 11(2):521–540, 1999.
23. V. R. Kompella, M. Luciw, and J. Schmidhuber. Incremental slow feature analysis: Adaptive low-complexity slow feature updating from high-dimensional input streams. *Neural Computation*, 24(11):2994–3024, 2012.