Journal of Artificial General Intelligence 8(1) 1-22, 2017
DOI: 10.1515/jagi-2017-0001

Submitted 2017-03-27
Accepted 2017-05-11

Homeostatic Agent for General Environment

Naoto Yoshida NAOTO.YOSHIDA@GROOVE-X.COM
GROOVE X
1-8-12 Nihonbashi-Horidomecho
Chuo-ku, Tokyo, Japan

Editor: Laurent Orseau

Abstract

One of the essential aspect in biological agents is dynamic stability. This aspect, called homeostasis, is widely discussed in ethology, neuroscience and during the early stages of artificial intelligence. Ashby's homeostats are general-purpose learning machines for stabilizing essential variables of the agent in the face of general environments. However, despite their generality, the original homeostats couldn't be scaled because they searched their parameters randomly. In this paper, first we re-define the objective of homeostats as the maximization of a multi-step survival probability from the view point of sequential decision theory and probabilistic theory. Then we show that this optimization problem can be treated by using reinforcement learning algorithms with special agent architectures and theoretically-derived intrinsic reward functions. Finally we empirically demonstrate that agents with our architecture automatically learn to survive in a given environment, including environments with visual stimuli. Our survival agents can learn to eat food, avoid poison and stabilize essential variables through theoretically-derived single intrinsic reward formulations.

Keywords: Homeostat, Reward, Reinforcement Learning, Survival

1. Preliminaries

Survival strategies are essential for biological agents in the wild. Many researchers have developed various types of survival agents since the early days of artificial intelligence. Ashby (1960) developed Homeostat which dynamically stabilizes the state of the machine, and Walter (1953) developed simple robotic agents that could explore the environment of a room and automatically recharge their batteries at a recharging station. Toda (1962, 1982) discussed the survival problem of artificial agents in a natural environment. He speculated about the functional requirements for an autonomous survival agent based on decision theory. Based on Toda's works, Pfeifer and Scheier (1999) pointed out the importance of 'complete' autonomous agents and research on embodied cognitive science. In this sense, they developed a simple self-sufficient autonomous robot. McFarland and Bösser (1993) discussed the autonomous agent from the perspective of research on animal behavior. They suggested several requirements for intelligent agents through comparison of the market economy with natural selection, and they also developed simple robots that were self-sufficient autonomous agents (McFarland and Spier, 1997). Also, Lin (1992) compared several reinforcement learning (RL) architectures for a complex survival task in a non-Markovian environment through a simulation study. Sibly and McFarland (1976) introduced the state space approach in ethology and suggested that animal behaviors are the consequence of optimal control with respect to the cost function given by the fitness function. Keramati and Gutkin (2011)

suggested a similar perspective, but they also suggested changes in the distance between the current homeostatic state and the optimal desired state as the reward function of RL. Konidaris and Barto (2006)developed RL architecture that automatically balances multiple required nutrients (protein, fat, water, etc.) through tuning of the reward function, which depends on the agent's homeostatic state and tested the architecture in a simulation experiment.

Ogata and Sugano (1997) developed a real robot agent intended for survival. Their robot evaluates the sensor signals (motor temperature, battery level, etc.) in real time and learns to avoid undesired stimuli. Doya and Uchibe (2005) developed robots termed "Cyber rodents" intended for the study of learning agents for survival and evolutionary robotics. Cyber rodents can recharge their batteries by touching special battery packs in the environment. Wireless communication modules of robots enable the software evolution of control algorithms (Elfwing et al., 2005).

Reward stimuli were introduced in most of the preceding research studies, with a reward function being necessary for the RL paradigm. However, almost all of the previous studies on survival adopted hand-crafted reward functions that do not guarantee the intended behaviors, which are the survival strategies in this case. The reward and objective function given by the designer may work well in simple RL tasks. However, for more complex tasks and life-long learning settings in which agents must learn a large amount of data, a badly hand-crafted reward function may have damaging effects on system performance in the end.

From these considerations, we first focus on the mathematical formulation of the classical survival problem as a maximization of the multi-step survival probability. To solve this problem, we introduce an iterative model-based method for the maximization of the objective function through an EM algorithm with variational approximations. After the M-step, the negative free-energy function (variational lower bound) in our algorithm is identical to the form of the classical RL objective function with a specific reward function. This result suggests that it can be maximized by classical model-free RL algorithms. Our contributions are i) a probabilistic formulation of the classical survival problem, ii) a suggested RL approach to solve this problem and iii) a demonstration that maximization of the multi-step survival probability through RL algorithms is identical with maximization of the log of that probability from the variational lower bound and iv) we empirically examined these issues from the toy environment to the 3D environment using raw vision input.

1.1 Objective Functions of Reinforcement Learning Problem

Reinforcement learning (RL) is the field of research that constructs learning agents that obtain an optimal policy through interactions with the environment. We first introduce the general form of the objective function in the POMDP (partially observable Markov decision process) model. For the sake of simplicity, we restrict the discussion to a finite set of actions, states, and observations.

Many realistic environments for the agent can be modeled by the partially observable Markov decision process (POMDP) (Kaelbling, Littman, and Moore, 1996). The POMDP model consists of the state set $\mathcal{S}$, the action set $\mathcal{A}$, the observation set $\mathcal{O}$, the transition probability to state $s' \in \mathcal{S}$ given a state $s \in \mathcal{S}$ and an action $a \in \mathcal{A}$ as $P(s'|s,a)$, the observation probability $P(o|s)$, and the reward function $r(s,a)$.

At each time step t, the agent receives an observation o_t from the state s_t by the observation probability, and replies with an action a_t. Then, the state of the environment changes to s_{t+1}, and the agent receives a reward $r_t = r(s_t, a_t)$. In the POMDP setting, the agent can not access the true state $s \in \mathcal{S}$ of the environment. Therefore, the agent needs to infer the current true state from a sequence

of observations and actions. We call the sequence of observations and actions a **history** of the agent $h_t = \{o_0, a_0, o_1, a_1, \ldots, o_{t-1}, a_{t-1}, o_t\}$. Using the history, how the agent acts in the environment can be expressed by a probability $\pi(a|h)$ called the **policy**. In the POMDP model, the probability of generating the T-step **trajectory** $\tau_T = \{s_0, o_0, a_0, s_1, o_1, a_1, \ldots, s_{T-1}, o_{T-1}, a_{T-1}, s_T, o_T\}$ is

$$\begin{aligned} P(\tau_T|\pi) &= P(o_0|s_0)P(s_0) \\ &\quad \times \prod_{t=0}^{T-1} P(o_{t+1}|s_{t+1})P(s_{t+1}|s_t, a_t)\pi(a_t|h_t). \end{aligned}$$

Therefore, the expectation of the T-step average reward is given by

$$J_T(\pi) = \sum_{\tau_T} P(\tau_T|\pi)\Big[\frac{1}{T}\sum_{t=0}^{T-1} r_t\Big].$$

We denote the limit $T \to \infty$ by $J(\pi) = \lim_{T\to\infty} J_T(\pi)$. This $J_T(\pi)$ or $J(\pi)$ (or the product with some constant) is a typical objective function in the reinforcement learning literature[1]. The objective of the reinforcement learning is to find the optimal policy π^* that achieves the maximum of the objective function defined above through interactions with the environment.

2. Survival Problem

In this section, we formulate the survival problem from the models of an animal proposed by Ashby (1960) and similar idea suggested from the view point of ethology (Sibly and McFarland, 1976; McFarland and Bösser, 1993). In their model, an animal has several variables that are observed by the animal and have some survival importance (for example, the water level and the energy level in the body, as shown in Figure 1). We call these variables the 'physiological state'. On the other hand, an agent also has a 'perceptual state' which is the perception of the environmental stimuli (vision, touch, etc.). The combined physiological state and perceptual state, which may be represented in the animal's brain, is called the 'motivational state' (McFarland and Bösser, 1993)[2]. The animal has one compact manifold, which is defined in the physiological state space. This manifold is called the viability zone (Meyer and Guillot, 1991), and we define the state of the animal as 'Alive' when the current physiological state is in the manifold. The adaptive behavior is expressed as an optimal process that drives the physiological state toward the optimal state using observed data from outside and inside the body.

2.1 Formulation of Survival Problem

The assumptions of the problem are shown in Figure 2. Similar settings are suggested elsewhere (Ashby, 1960; McFarland and Houston, 1981; Cañamero, 1997; Barto, Singh, and Chentanez, 2004; Konidaris and Barto, 2006). A unit of the agent consists of an RL agent and a body, and the RL agent interacts with the world through the body. Because the sensors may not perfectly determine the current situation of in the world and the physiological state of the body, the sensor may exhibit

1. Even though many studies derive the algorithm from another objective function like $\sum_{t=0}^{T-1} \gamma^t r_t$, the performance of the algorithm is usually evaluated by the total or the average reward criteria.
2. The terms 'physiological state', 'perceptual state' and 'motivational state' may be misleading in the context of RL, because these "states" do not necessarily follow Markovian dynamics.

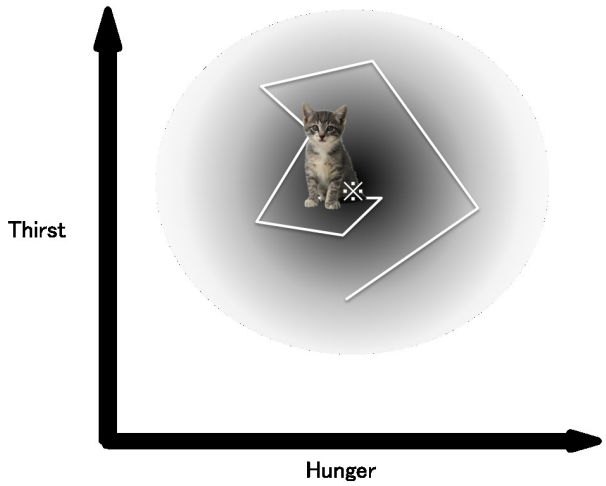


Figure 1: The State Space Model: The figure shows the relationships between the physiological state and the viability zone. In this figure, the viability zone depends on only two continuous variables; the "energy level" and "water level", for the sake of simplicity. The star represents the position of the optimal physiological state. The agent can remain alive in the viability zone (darker area) and the agent should be able to recover from a position separated from this area.

partial observability. Since sensing the physiological state is a process inside of the body, we may be able to assume that there is no information loss with this sensing.

We now formalize the survival problem as an optimization problem. We mostly follow the usual definition of the dynamics in the POMDP model explained in the previous section. Like the POMDP model, the agent interacts with the environment. At the state $s_t \in \mathcal{S}$, the agent receives the current observation $o_t \in \mathcal{O}$ with probability $P(o|s)$. Then, the agent takes an action $a_t \in \mathcal{A}$ following some policy π; the state then changes at the next time step following the probability $P(s_{t+1}|s_t, a_t)$. Because o_t is the observation that consists of the temporal stimulus to the agent at the time of step t, we can understand that the observation in POMDP is a generalization of the motivational state. In this definition, we do not explicitly separate the observation caused by external stimuli (the perceptual state: vision, touch, etc.) and the observation caused by internal stimuli (the observed physiological state: energy level, water level, etc.).

In order to formulate the survival problem, we introduce a binary signal A_t instead of the reward, which represents the "alive flag" of the agent at times step $t = 0, 1, 2, 3 \cdots$. $A_t = 1$ represents that the agent is 'alive' at the time step t, and $A_t = 0$ represents 'dead'. To generalize the problem, we soften the definition of the boundary of the viability zone by introducing the temporal survival probability $P(A_{t+1} = 1|s_t) > 0$. Because the survival probability is ultimately caused by the physiological state of the agent (animal), we assume that the survival probability is only dependent on the current state. However, the following discussion is directly applicable for other definitions of the temporal survival probability $P(A_{t+1} = 1|o_t)$, $P(A_{t+1} = 1|s_t, a_t)$, and so on. Also, we assume $A_0 = 1$, and this probability is known for the agent unit (either in the body or RL agent).

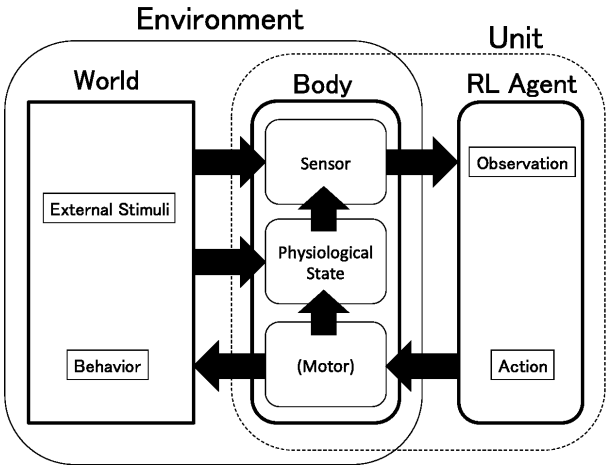


Figure 2: The settings in the survival problem with an RL agent. A unit of the agent consists of a body and an RL agent. The RL agent interacts with the world through the body. The world receives the motor outputs from the body, returns the (external) stimulus, and changes the physiological state of the body. The body receives an external stimulus through the sensors, while monitoring the physiological state through other sensors. Then, the body sends an observation to the RL agent. The RL agent receives the observation and returns the action.

A natural interpretation of 'survival' is to stay alive as long as possible in the environment, so the agent requires policy π that realizes the signal sequence $A_t = 1$ $(t = 0, 1, \dots)$. Using these definitions, the multi-step survival probability given a policy π is expressed by a joint probability

$$P(\bar{A}_T|\pi), \tag{1}$$

where $\bar{A}_T$ denotes the sequence $\{A_0 = 1, A_1 = 1, \dots, A_{T+1} = 1\}$. We then define the objective of the agent for the survival problem as the maximization of the probability defined by (1).

2.2 Maximization of Survival Probability by Variational Method

In the following discussion, we show that the maximization of the objective function (1) through the maximization of the variational bound can be reduced to the maximization of the conventional objective function of RL.

Firstly we discuss the situation of the planning problem, in which we search for the optimal policy given the true transition probability $P(s'|s, a)$, the observation probability $P(o|s)$, and the temporal survival probability $P(A|s)$. The logarithm of the objective function (1) can be transformed by introducing the arbitrary probability distribution $Q(\tau)$ on the T-step trajectory τ

as

$$\begin{aligned}\log P(\bar{A}_T|\pi) &= \sum_\tau Q(\tau)\log P(\bar{A}_T|\pi)\\ &= \sum_\tau Q(\tau)\log\frac{P(\bar{A}_T,\tau|\pi)}{P(\tau|\bar{A}_T,\pi)}\frac{Q(\tau)}{Q(\tau)}\\ &= \sum_\tau Q(\tau)\Big(\log P(\bar{A}_T|\tau)-\log\frac{Q(\tau)}{P(\tau|\pi)}\Big)+\sum_\tau Q(\tau)\log\frac{Q(\tau)}{P(\tau|\bar{A}_T,\pi)}\\ &= -F(Q(\cdot),P(\cdot|\pi))+\mathrm{KL}(Q(\cdot)||P(\cdot|\bar{A}_T,\pi)).\end{aligned}$$

The relationship $P(\bar{A}_T|\pi)P(\tau|\bar{A}_T,\pi)=P(\bar{A}_T,\tau|\pi)$ was used at the second equality. $P(\tau|\bar{A}_T,\pi)$ is the posterior of $P(\tau|\pi)$ given $\bar{A}_T$, that is $P(\tau|\bar{A}_T,\pi)=\frac{P(\bar{A}_T|\tau)P(\tau|\pi)}{\sum_{\tau'}P(\bar{A}_T|\tau')P(\tau'|\pi)}$ from Bayes's theorem. And $P(\bar{A}_T,\tau|\pi)=P(\bar{A}_T|\tau)P(\tau|\pi)$ was used at the third equality. The first term on the RHS in the last row is

$$-F(Q(\cdot),P(\cdot|\pi))=\sum_\tau Q(\tau)\Big(\log P(\bar{A}_T|\tau)-\log\frac{Q(\tau)}{P(\tau|\pi)}\Big),$$

and F is called the free energy by analogy with statistical mechanics. If there is some restriction on the distribution $Q(\tau)$ (for example, $Q(\tau)$ is given by a specific class of the probability distribution), F is called the variational free energy. $\mathrm{KL}(Q(\cdot)||P(\cdot))$ is the Kullback-Leibler (KL) divergence $\mathrm{KL}(Q(\cdot)||P(\cdot))=\sum_\tau Q(\tau)\log\frac{Q(\tau)}{P(\tau)}$. The KL divergence is known to be non-negative and zero only if the two probability distributions P and Q are equivalent. Because of the non-negativity of the KL divergence and the above equality, the log-probability $\log P(\bar{A}_T|\pi)$ is lower bounded by $-F$. Hence, this value is termed the variational (lower) bound.

Also from the equality above,

$$\log P(\bar{A}_T|\pi)+F(Q(\cdot),P(\cdot|\pi))=\mathrm{KL}(Q(\cdot)||P(\cdot|\bar{r}_T,\pi)),$$

and there is a relationship

$$\operatorname*{argmin}_Q[F(Q(\cdot),P(\cdot|\pi))]=\operatorname*{argmin}_Q[\mathrm{KL}(Q(\cdot)||P(\cdot|\bar{A}_T,\pi))]$$

because $\log P(\bar{A}_T|\pi)$ is not a function of Q.

In the maximization problem of the likelihood $\log P(\bar{A}_T|\pi)$, the method that introduces the restricted class of $Q(\tau)$ and maximizes the variational bound $-F$ is known to be the variational method and it is widely used in machine learning (Jordan et al., 1999). The probability distribution of the T-step trajectory $\tau=\{s_0,o_0,a_0,s_1,o_1,a_1,\ldots,s_{T-1},o_{T-1},a_{T-1},s_T,o_T\}$ given a policy π is

$$P(\tau|\pi) = P(o_0|s_0)P(s_0)\prod_{t=0}^{T-1}P(o_{t+1}|s_{t+1})P(s_{t+1}|s_t,a_t)\pi(a_t|h_t).$$

And we restrict the distribution $Q(\tau)$ with arbitrary policy $\pi_Q(a|h)$ to

$$Q(\tau|\pi_Q) = P(o_0|s_0)P(s_0)\prod_{t=0}^{T-1}P(o_{t+1}|s_{t+1})P(s_{t+1}|s_t,a_t)\pi_Q(a_t|h_t).$$

Algorithm 1 Iterative Policy Improvement Algorithm for Survival

1. Set $k = 0$ and an arbitrary policy π^0.
2. (E-step) Obtain π_Q^k by optimization

$$\begin{aligned}\pi_Q^k &= \underset{\pi_Q}{\mathrm{argmin}}[\mathrm{KL}(\mathrm{Q}(\cdot|\pi_\mathrm{Q})||\mathrm{P}(\cdot|\bar{\mathrm{A}}_\mathrm{T};\pi^\mathrm{k}))] \\ &= \underset{\pi_Q}{\mathrm{argmax}}[-F(Q(\cdot|\pi_Q), P(\cdot|\pi^k))].\end{aligned}$$

3. (M-step) Update π by using π_Q^k. That is

$$\begin{aligned}\pi^{k+1} &= \underset{\pi}{\mathrm{argmax}}[-F(Q(\cdot|\pi_Q^k), P(\cdot|\pi))] \\ &= \pi_Q^k.\end{aligned}$$

4. **if** π^{k+1} is converged, **then**
5. return π^{k+1}
6. **else**
7. $k \leftarrow k + 1$ and go to E-step.
8. **end if**

By using these distributions, we introduce the EM algorithm as in Algorithm 1. The maximization in the M-step is simply the replacement of π^k by π_Q^k from the equality

$$\begin{aligned}&\underset{\pi}{\mathrm{argmax}}[-F(Q(\cdot|\pi_Q), P(\cdot|\pi))] \\ =\ &\underset{\pi}{\mathrm{argmax}}\Big[\sum_\tau Q(\tau|\pi_Q)\Big(\log P(\bar{A}_T|\tau) - \log\frac{Q(\tau|\pi_Q)}{P(\tau|\pi)}\Big)\Big] \\ =\ &\underset{\pi}{\mathrm{argmin}}\Big[\sum_\tau Q(\tau|\pi_Q)\log\frac{Q(\tau|\pi_Q)}{P(\tau|\pi)}\Big] \\ =\ &\underset{\pi}{\mathrm{argmin}}[\mathrm{KL}(Q(\cdot|\pi_Q)||P(\cdot|\pi))] \\ =\ &\pi_Q.\end{aligned}$$

Because of the restriction on $Q(\tau|\pi_Q)$, the minimization of the KL divergence in the E-step is variational sense and may not be zero. If the environment is MDP and we assume that π_Q is a Markov policy, Rawlik et al. derived the analytical solution of the E-step and it is given by the softmax policy $\pi_Q(a|s) = \exp\{\Psi(s,a)\}/Z$, where $\Psi(s,a)$ is some energy function and Z is the normalization term (Rawlik, Toussaint, and Vijayakumar, 2013).

In POMDP, on the other hand, no analytical solution to the E-step is known. To tackle this problem, we may parametrize the policies π and π_Q by parameters θ, ϕ as π_θ, π_ϕ. And then, we assume $\theta = \phi \Rightarrow \pi_\theta = \pi_\phi$. Therefore $\theta = \phi \Rightarrow P(\tau|\pi_\theta) = Q(\tau|\pi_\phi)$ by definition. The variational method that introduces the parametrized variational distribution Q_ϕ and maximizes the variational bound $-F(\phi,\theta) := -F(Q(\cdot|\pi_\phi), P(\cdot|\pi_\theta))$ by gradient methods are well known in the neural computing community (Dayan and Hinton, 1996; Ranganath, Gerrish, and Blei, 2013; Kingma and Welling, 2013; Mnih and Gregor, 2014). Following this idea, we replace the E-step

and M-step in Algorithm 1 by

$$\phi^k = \underset{\phi}{\operatorname{argmax}}[-F(\phi, \theta^k)] \tag{2}$$

and

$$\theta^{k+1} = \underset{\theta}{\operatorname{argmax}}[-F(\phi^k, \theta)] = \phi^k. \tag{3}$$

If each stage of the algorithm is performed exactly, a monotonic increase of the variational bound

$$-F(\phi^0, \theta^1) \leq -F(\phi^1, \theta^2) \leq -F(\phi^2, \theta^3) \leq \ldots$$

after the M-step is guaranteed. Moreover, there is a relationship after the M-step

$$\begin{aligned} -F(\phi^k, \theta^{k+1}) &= -F(\theta^{k+1}, \theta^{k+1}) \\ &= \sum_{\tau} P(\tau|\pi_{\theta^{k+1}}) \log P(\bar{A}_T|\tau) \\ &= \sum_{\tau} P(\tau|\pi_{\theta^{k+1}}) \Big[\sum_{t=0}^{T-1} \log P(A_{t+1} = 1|s_t)\Big] \\ &= T J_T(\pi_{\theta^{k+1}}), \end{aligned}$$

where $J_T(\pi)$ denotes the objective function of the reinforcement learning with respect to the reward function $r_t = \log P(A_{t+1} = 1|s_t)$. Here, the equality $\log P(\bar{A}_T|\tau) = \sum_{t=0}^{T-1} \log P(A_{t+1} = 1|s_t)$ is used at the third equality. Because T is a constant, the increase of the variational bound is equivalent to the increase of $J_T(\pi_{\theta^{k+1}})$. Therefore, from the discussion above, the maximization of the log-form of the objective function $\log P(\bar{A}_T|\pi_\theta)$ through the variational bound $-F(\phi, \theta)$ is reduced to the maximization of $J_T(\pi_\theta)$ with respect to the parameter of the agent θ.

2.3 Solving the Survival Problem by Reinforcement Learning Algorithms

Now we consider the survival problem in the reinforcement learning setting: that is, the maximization of the log of the objective function $\log P(\bar{A}_T|\pi)$ while the agent cannot access the true environment model $P(o|s)$, $P(s'|s, a)$. In this setting, we can not perform the iterative algorithms above. However, from the discussion of the second algorithm (equation 2, 3), the variational bound is proportional to $J_T(\pi_\theta)$ after the each M-step. Then, in order to maximize the variational bound, we can take a direct maximization of $J_T(\pi_\theta)$ with respect to θ, instead of the exact execution of the iterative algorithm. Because $J_T(\pi_\theta)$ with reward function

$$r_t = \log P(A_{t+1} = 1|s_t)$$

is the conventional objective function of the reinforcement learning paradigm, we can apply the RL algorithms to the maximization of the survival probability.

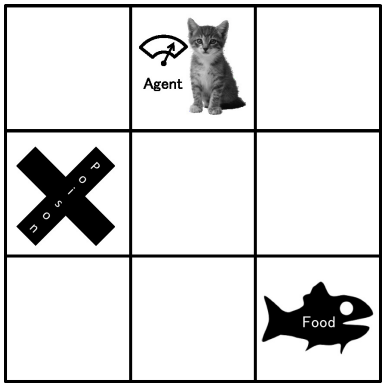


Figure 3: The grid world environment. There are two objects **A** (Food) and **B** (Poison), but the agent initially does not know which corresponds to the "food" object. Further details of the environment are explained in the main text.

2.4 Necessity of the Research of General Survival Agent

Our result suggests that the survival problem, in terms of the maximization of the multi-step survival probability, can be treated as a formal RL problem. Although there have been discussions about a potential danger of sufficiently advanced artificial agents that have a strong drive toward self-preservation (Omohundro, 2008), researches of general survival agents that have limited abilities for problem solving are still important. The reason is two-fold. First, the reward setting problem is a headache in the RL paradigm. In our survival agent, the setting of the fundamental reward function is given and it is driven by intrinsic signals (physiological state) through the temporal survival probability. The intuition of the survival probability is much clear than that of hand-crafting the reward function, and we can obtain the "right" reward function for free (however the applicable problem is limited into the survival problem). Second, survival is a fundamental problem not only for our agent, but for all of biological agents (Young, 1966; Dawkins, 1976). Although our survival agent is quite limited in terms of problem solving (a specific application of the general reinforcement learning), our direction consistent with that of biological agents and hence our approach would be "the weakest strong-AI".

3. Experiment

In the experiment, first we verified the reward setting by evaluating survival time steps in several environments. And then, we tested our reward setting in survival problem with low-dimensional inputs. Finally, we tested in more a challenging environment, that is, a 3D environment with high-dimensional visual inputs.

3.1 Discretized Environment with Continuous Energy Resource

Environment The environment consists of a 3x3 grid world (Figure 3). The agent selects an action at each time step from UP, DOWN, RIGHT, LEFT and EAT. When the agent takes the action UP, DOWN, RIGHT or LEFT and if the wall is not in that direction, the agent moves one step in

the selected direction. Otherwise, the agent stays at the current position. In the environment, there are two types of objects, **A** and **B**, at uniformly random positions, such that the two objects never overlap. The position of the objects changes if the agent selects the EAT action at the corresponding position. Also, the position of **B** may randomly change at every time step with a probability of 0.01.

The agent has a continuous battery level $E \in [0, 100]$, which decreases by 1% at each time step. The battery level is recharged $+5$ if the agent selects EAT at the position of object **A**. Therefore, the object **A** corresponds to the food object.

The temporal survival probability of the agent when $A_t = 1$ is defined as $P(A_{t+1} = 1|s_t) = P(A_{t+1} = 1|E_t, C_t) = f(E_t)g(C_t)$ where E_t is the battery level at time $t = 1, 2, 3, \ldots$, $C_t \in \{0, 1\}$ is the flag bit whether the agent ate the object **B** ($C = 1$) or not ($C = 0$). f and g are defined as

$$f(E_t) = \exp\Big\{-\frac{(E_t - 60)^2}{1000}\Big\}$$

and

$$g(C_t) = \begin{cases} 0.5 & (C_t = 1) \\ 1 & (C_t = 0). \end{cases}$$

The observation of the agent is defined by the set $o = \{x, p^A, p^B, c, \hat{E}\}$, where x is the position of the agent, p^A is the position of the object **A**, p^B is the position of the object **B**, c_t is the type of object that the agent EATs ($c = 1$ for nothing, $c = 2$ for **A**, and $c = 3$ for **B**), and $\hat{E}$ is the discrete state of the current battery level. In this experiment, we discretized the continuous battery level $[0, 100]$ into 20 separate discrete regions, and $\hat{E}$ receives the class of the corresponding region of the current battery level E. Therefore, this environment is a simple POMDP setting because of the discretization of the battery level. In order to survive in this environment, the agent must take the food (**A**), avoid the poisonous object (**B**) and regulate its energy level (E). Even though this might be an over simplified model of a biological agent, this kind of situation exists everywhere in life. Importantly, in this setting, agents initially do not know which object information (p^A, p^B) corresponds to the "food" or "poison". Therefore the agent has to associate these objects with changes in the physiological values (E and c). Also, agents never receive positive rewards when they take food and the reward values for food-capture depend on the agent's battery level. Therefore, this experiment is fundamentally different from task-oriented problems like the "food capturing".

Agent Settings A sarsa(λ) agent (Rummery and Niranjan, 1994) was used in this task and the action-value function was expressed by the tabular function. In this experiment, the learning rate α was 0.1, the discount rate γ was 0.95 and the decay rate of the eligibility trace λ was 0.1. The agent follows the ϵ-greedy policy in which the action is almost entirely selected by greedy action selection $a = \mathrm{argmax}_b Q(s, b)$ but with a small probability of $\epsilon = 0.01$, the action is selected from the uniformly random distribution over the action set. The training procedure of the agent was as follows. An episode starts at the optimal battery level (that is, $B_t = 60$) with random allocation of objects in the environment. At each time step, the alive flag is updated according to the temporal survival probability $P(A_{t+1}|s_t)$. If the agent receives $A_{t+1} = 0$ after the t-th update, the episode ends and the next episode starts.

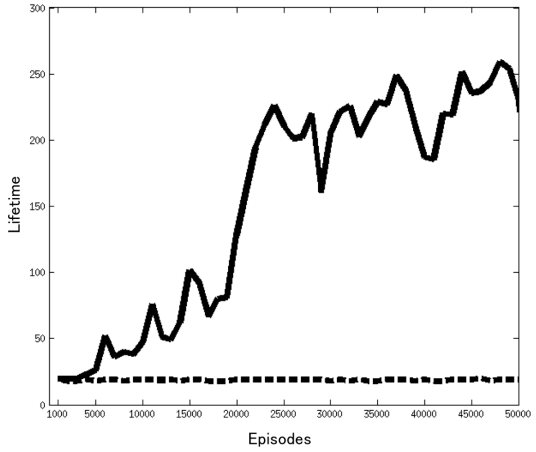


Figure 4: The median of the survival time step along the episodes. The details are explained in the main text.

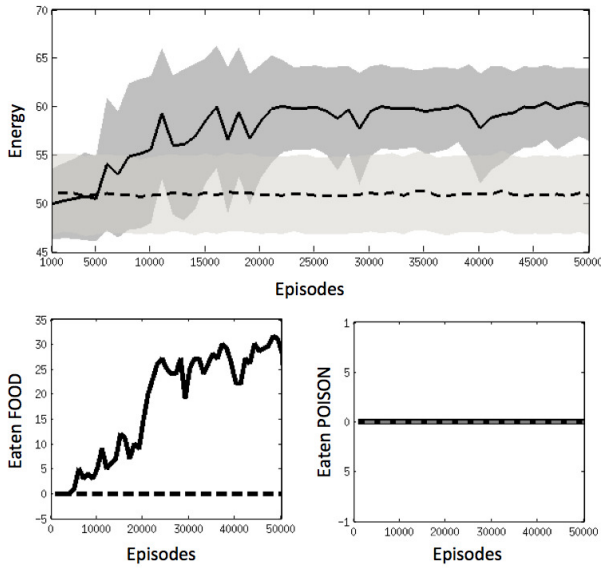


Figure 5: Top: The evolution of the mean battery level and the standard deviation at death. Bottom Left: The median of the number of **A** (food) eaten by the agent during the evaluation step. Bottom Right: the median of the number of **B** (poison) eaten by the agent during the evaluation step. Solid line: RL agent with survival reward settings. Dashed line: The agent with random action selection.

Results Figure 4 shows the evolution of the median of the survival time along the number of episodes. Evaluation was done by freezing parameters of the agent every 1000 episodes, and the

agent is tested for 1000 episodes without learning. The solid line represents the median of the survival time of the sarsa(λ) agent with the survival reward settings introduced above. The dashed line represents an agent which randomly and uniformly selects an action among the 5 actions. The growth of the lifetime clearly shows that the sarsa(λ) agent successfully learns the survival strategy during the process. The result of the random agent shows that the environment has sufficient complexity such that the random agent cannot stay alive longer than 25 time steps. Figure 5 shows the battery level of the agent at its death and the median amount of **A** (left panel) and **B** (right panel) consumed in the evaluation process after the corresponding episodes. From these figures, we can know that the battery level is successfully controlled around the desired level ($E = 60$) and that the amount of eaten food (**A**) increased. On the other hand, the amount of poison (**B**) eaten is always zero. This result also supports the successful learning of the survival strategy by sarsa(λ) with only the survival reward.

3.2 Two-Resource Problem with Low Dimensional Inputs

In the second experiment, we treated the two-resource problem inspired by the work of Spier (1997). In this experiment, agents need to maintain two nutrient resources (Red and Blue) in their body, like the energy resource in the previous experiment. In this experiment, the observation input is a vector with continuous values. Therefore the function approximation is required for RL.

Environment The field of the environment is shown in Figure 6. The size of the field is 500×500 pixels, but the agent can only move in an area of 480×480 pixel. In the field, there are two kinds of food objects (red and blue) and the numbers of the food objects are both 15. Agents have three range finders for red food, blue food and walls (the edge of square field) to detect distance from objects. Because range finders have detection limits, agents can observe only partial information about the field. If the distance of the agent from the food object (red and blue) is less than 30 pixels (that is, if the agent "touches" the food), the nutrient resource corresponding to the food color is supplied by 1.0 and that food disappears (food is eaten). When the food is eaten, a new food object is generated at a random position to maintain the number of food objects in the field. Nutrient resource levels are decreased every step by 0.01.

A similar experiment was treated by Konidaris and Barto (2006) as an RL experiment, but they defined high-level actions as those "approaching" and "consuming" each visible target. In our experiment, the agent has only three primitive actions "Rotate Clockwise", "Rotate anti-Clockwise" and "Forward". Rotation actions rotate the agent by 15 degrees from the current position, agents proceed 30 pixels when the "Forward" action is selected.

The reward is given as a function of the two nutrient resource levels (red and blue)

$$r(s) = \log e^{-(s_{\mathrm{red}}^2 + s_{\mathrm{blue}}^2)} \propto -k(s_{\mathrm{red}}^2 + s_{\mathrm{blue}}^2). \tag{4}$$

This setting corresponds to the exponential bell-shaped probability centered at zero $P(A = 1|s) = \exp(-s^\top s)$ and is settled as the temporal survival probability. In our experiment $k = 0.05$ was chosen for RL. If the red or blue nutrient resource level of the agent exceeded the range $[-10.0, 10.0]$, that episode is terminated. Then the next episode starts with $s_{\mathrm{red}} = s_{\mathrm{blue}} = 0$ and the agent, red food and blue food are randomly and uniformly settled in the field. Because this RL task is a continual task, the agent can survive indefinitely once a successful survival policy is obtained. Therefore, we are forced to terminate episodes if the agent survive 10,000 time steps.

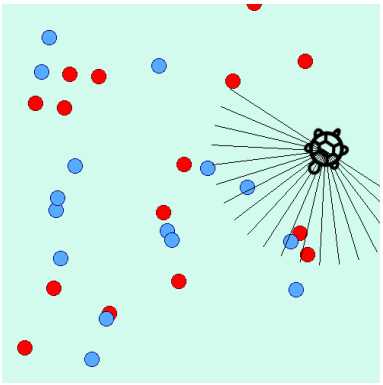


Figure 6: The two-resource environment with low-dimensional observations. There is the agent and there are two nutrient resources (red and blue). The agent has three range finders for walls (the edge of the environment) and nutrient resources. The agent has two internal states, which are red and blue nutrient resources levels. In order to survive, the agent needs to achieve a balance of both nutrient resource levels to around zero.

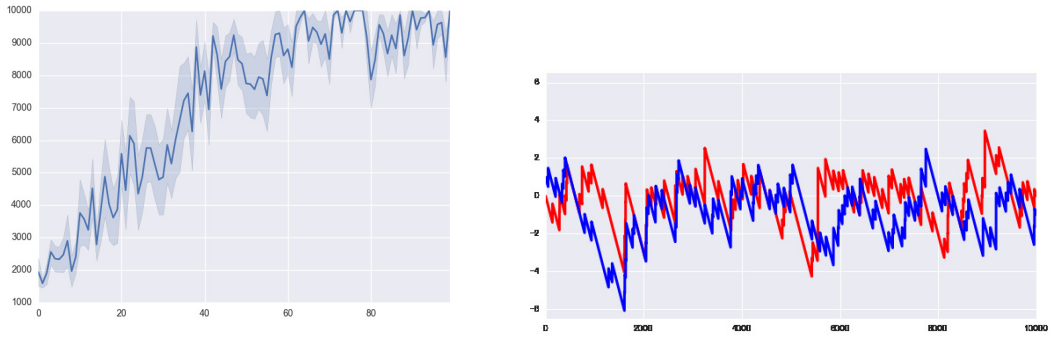
Agent Settings In this experiment, we construct an RL agent by Q-learning with function approximation. We used a three-layer perceptron as the function approximator. The neural network has input units, two thousand hidden units with the rectified linear (ReLU) activation function, and three linear output units. The number of output units corresponds to the number of actions. The input of the network is a vector of concatenated sensor inputs (i.e. $o_1 \sim o_5$). Formally, the input is

$$
\begin{aligned}
o_t &= [o_1, o_2, o_3, o_4, o_5] && (5)\\
o_1 &= \kappa_1 s_{t-1} && (6)\\
o_2 &= \kappa_2 \Big(s_t - s_{t-1} \Big) && (7)\\
o_3 &= \kappa_3 \Big(d_t^{\text{wall}} - d_{t-1}^{\text{wall}} \Big) && (8)\\
o_4 &= \kappa_3 \Big(d_t^{\text{red}} - d_{t-1}^{\text{red}} \Big) && (9)\\
o_5 &= \kappa_3 \Big(d_t^{\text{blue}} - d_{t-1}^{\text{blue}} \Big) && (10)
\end{aligned}
$$

where s_t represents the nutrient resource level vector (2 dimensions for red and blue) at time step t, d^{obj} represents the range finder input vector with respect to the object obj. κ_1, κ_2 and κ_3 are scaling constants. We used $\kappa_1 = \frac{1}{10}$, $\kappa_2 = 1$ and $\kappa_3 = \frac{1}{150}$ in our experiment. We note that we can reconstruct s_t by $f_1/\kappa_1 + f_2/\kappa_2$. Even though providing o_1 and o_2 is equivalent to providing s_{t-1} and s_t, the different representation is known to be useful for treating the raw sensor input in robotic task (Hester and Stone, 2012). Also, this feature empirically worked well in this experiment.

The training of the network is the same as that with the common Q-learning with function approximation. At every step the agent takes the gradient of the loss function with respect to the network parameter vector θ

$$
L_t = \frac{1}{2} \Big(\mathcal{T}_{t+1} - Q_\theta(o_t, a_t) \Big)^2 + \lambda \theta^\top \theta. \quad (11)
$$



(a) Environment (b) Invisible walls in the environment

Figure 7: (a) The growth of survival time along the episode. The plot shows the mean and the standard deviation over ten individual run of the experiment. (b) The trajectory of two nutrient resource levels during the 100-th trial.

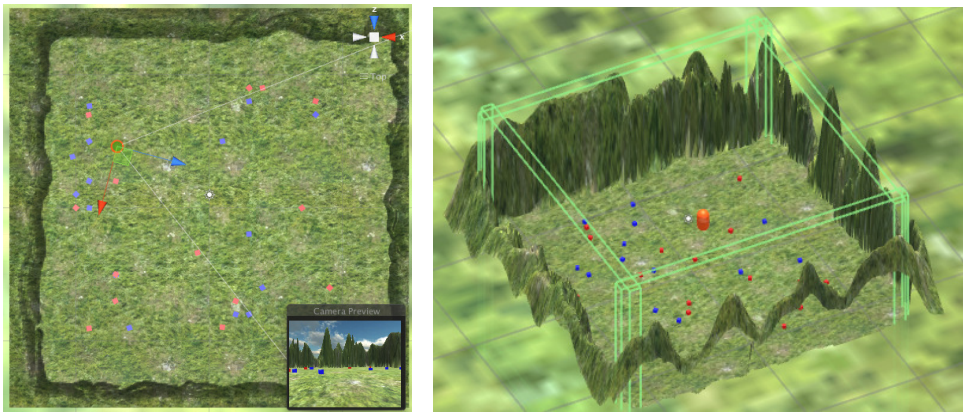
$\mathcal{T}_{t+1} = r_t + \mathrm{argmax}_a Q_\theta(o_{t+1}, a)$ is the temporal target of the network and not be differentiated, and Q_θ is the approximated action-value function represented by multi-layer perceptrons. We used the regularization term $\lambda = 0.0001$ and Adam optimizer[3] (Kingma and Ba, 2014) to update the parameters. Also, We used ϵ-greedy action selection as the exploration strategy. This action selection samples an action from the flat probability function over the action space with probability ϵ, and otherwise takes the greedy action $a = \mathrm{argmax}_a Q(o, a)$. We used $\epsilon = 0.1$ in this experiment.

Results Survival time steps along the episode are shown in Figure 7(a). The plot is the average with a standard deviation of 10 individual runs of experiments. The plot clearly shows that the agent is initially unable to survive longer than 2,000 time steps, but later the survival time successfully increases along with the number of episodes. Figure 7(b) shows the trajectories of two nutrient resources (the colors of the trajectories correspond to red and blue resources) during the 100-th episodes. We notice that the agent roughly switched to obtain two resources in turns. This result suggests that the high-level switching of the behavior between the red-resource seeking behavior and the blue-resource seeking behavior emerged from scratch. The figures in Appendix A represent the same plot corresponding to the first 10 trials (Figure 12) and the last 10 trials (Figure 13) in each experiment.

3.3 Two-Resource Problem with Visual Inputs in 3D Environment

Finally, we scaled the two-resource problem to address the vision input. The 3D environment was constructed by the LIS (Life-in-Silico) framework developed by Nakamura and Yamakawa (2016). LIS is a general-purpose reinforcement learning framework on Unity game engine. Using this framework, we can develop an arbitrary 3D (and 2D) RL environment with a physics engine using Unity editor tools.

3. we used the learning rate 0.01 and the epsilon 1.0 as the parameter of Adam.



(a) Top-view of the environment (b) Invisible walls in the environment

Figure 8: Overview of the 3D Two-Resource Problem Environment

Environment The overall view of our environment is shown in Figure 8(a). The setting is similar to that of the previous experiment but now everything is in 3D. The approximately 12x12[m^2] field is surrounded by mountains, and the bottom of the field is covered with a grass texture. The square field is surrounded by four invisible walls to prevent agents from being embedded in the mountains and seeing the outside of the field (Figure 8(b)). In the field, there are two kinds of food that have different nutritional properties (red and blue food) and there are 15 units for each. These food objects are represented by small cubes and scattered uniformly and randomly over the entire field. The wire frame of the agent is represented in Figure 9. There is an invisible capsule (colored in green for visualization) attached to the front of the agent's orange body (the wire frame is blue). When the invisible capsule touches the food, the agent's nutrient resource corresponding to the food color is increased by 0.3 and the touched food is erased. Then a new food object is generated at a random position to maintain the number of food objects in the experiment. The color (RGB) vision information is provided by the single front camera on the agent and then the vision information is scaled into 32x32x3 pixel images.

Nutrient resource levels are decreased every step by 0.01. Similar to the previous experiment, the agent has three primitive actions "Rotate Clockwise", "Rotate anti-Clockwise" and "Forward". The reward is given as the function of the two nutrient resource levels (red and blue); as in the previous experiment, $k = 0.01$ was chosen in this experiment. When the nutrient resource level of the agent exceeded the range $[-10.0, 10.0]$, that episode is terminated and the next episode starts with $s_{\text{red}} = s_{\text{blue}} = 0$. The agent, red food, and blue food are randomly and uniformly settled over the entire field. Again, this is continual task and the agent can survive indefinitely after a successful survival policy is obtained. Therefore, we are forced to terminate the episodes when the agent survives 20,000 time steps in an episode.

Agent Settings We applied a simple Deep Q-network (DQN) developed by Mnih et al. (2015) in this task. DQN is composed of a convolutional neural network to efficiently treat visual information. The loss function is the same as in the last experiment (equation 11), but DQN has a replay buffer that is used for the experience replay (Lin, 1992) to utilize experienced trials from the past. In the visual environment, we used the observation

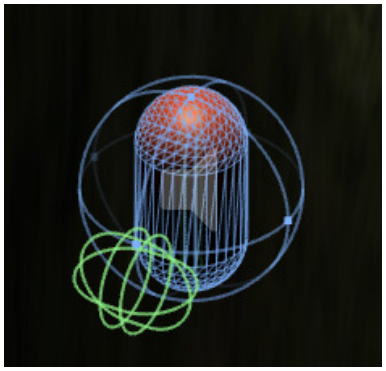


Figure 9: Touch area of the agent: Green capsule is attached in front of the agent body (blue meshed capsule). If the redo or blue nutrient resource cube is attached to the green capsule, that cube is "eaten" and the nutrient resource level of the agent is charged.

$$o_t = [o_1, o_2, o_3] \tag{12}$$
$$o_1 = \kappa_1' s_t \tag{13}$$
$$o_2 = \kappa_2' v_t \tag{14}$$
$$o_3 = \kappa_2' v_{t-1} \tag{15}$$

where v_t is the visual input (32x32 RGB Image) at the t-th time step, $\kappa_1' = 0.1, \kappa_2' = 1/255$ in this experiment. Figure 10 shows the network architecture in our experiment. Inputs include low-dimensional resource information inputs (top flow) and high-dimensional visions (bottom flow), and outputs are approximated action values for the three actions (forward, rotate right, rotate left). Visual inputs are embedded by two convolutional layers (8 filter banks with 3x3 convolution kernels, 16 filter banks with 3x3 convolution kernels) and then connected to the fully connected layers (400 hidden units). The first nutrient resource information inputs are embedded by a fully connected layer (50 hidden units), and then mixed with visual inputs. We used rectifier linear units in all of the hidden layers. We adopted the epsilon-greedy exploration strategy, and decreased the exploration rate linearly along the time steps. Detailed parameters of the training rules of DQN are presented in the table in Appendix B.

Results Figure 11 shows the results of our experiment. The plot shows the average survival steps (thick line) of the 10 individual runs and the standard deviation. The result clearly shows that the agent initially cannot survive longer than 2,500 steps, but the survival time rapidly grows along with the number of episodes experienced. In our experiment, no positive reward was provided to the agent even when in obtained a nutrient resource. However, the agent automatically associated the internal nutrient resource level with the visual information, which enabled it to survive for a sufficiently long time in a given environment.

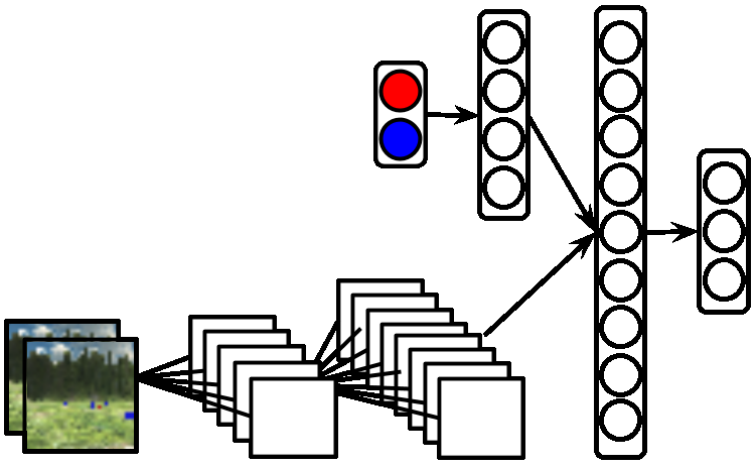


Figure 10: The network architecture in our experiment. Inputs are low-dimensional resource information inputs (top flow) and high-dimensional visions (bottom flow), and outputs are approximated action values. Visual inputs are embedded by two convolutional layers and then connected to fully connected layers. The nutrient level information is embedded by a fully connected layer and then mixed with visual inputs.

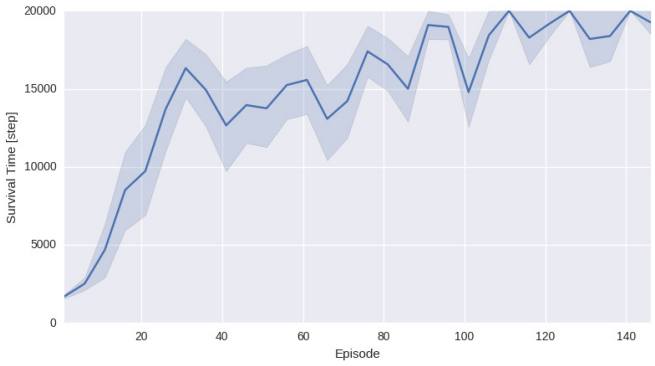


Figure 11: 10 Average result

4. Discussion and Conclusion

The relationships between the planning problem and the inference problem has recently become a hot topic in machine learning communities (Toussaint, Harmeling, and Storkey, 2006; Todorov, 2008; Kappen, Gómez, and Opper, 2012). Vlassis and Toussaint (2009) introduced the perspective that model-free reinforcement learning can be treated as the application of a stochastic EM algorithm to the maximization of the mixture likelihood $p(R = 1; \pi)$. Our objective function (1) and its lower bound were briefly introduced by Toussaint Toussaint (2009) in the context of the stochastic control problem. In this context, the goal of our study is maximization of the joint probability (1) from the

beginning. And we showed the relationships between its lower bound and the EM-based approach, including the POMDP case.

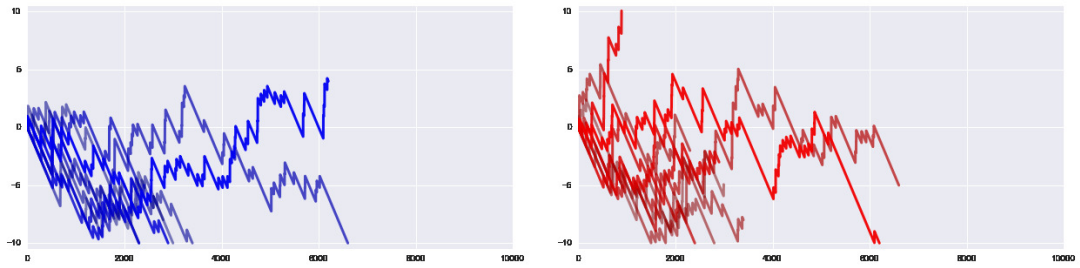
The reward function is crucial for utility-based agents. If we adopt RL as a model of the adaptive behavior of animals, the reward function and agent's ecological niche have to be mutually dependent from the view point of the stability or equilibrium of the agent in the wild environment. Otherwise their "innate" behavior cannot be that commonly observed in the wild, which would be a contradiction. In our approach, we introduced the first "fundamental" reward function of RL for the general survival problem, which had been heuristically defined in previous studies. The key is to soften the definition of the viability zone with the temporal survival probability, so the reward function is simply the log of the temporal survival probability. Using this setting, the agents can learn the survival policy with respect to maximization of the survival probability in the future. The source of the reward function, the temporal survival probability, has an explicit meaning and may be obtainable through the evolutionary process.

However, even though our reward setting is fundamental for survival, it may not be the "best reward" in terms of learning efficiency for an optimal survival policy. It is known that there are reward settings that have the same optimal policy but the RL agent has different learning speeds (Ng, Harada, and Russell, 1999). And recently, pre-training and simulator-based training approaches for RL have been successfully applied to the real-robot domain with direct visual image inputs (Lange, Riedmiller, and Voigtlander, 2012; Rusu et al., 2016). Therefore, to speed up learning and achieve a robotic agent that fits an unknown-dynamic environment in its (sometimes physical) lifetime, the survival problem should be examined in terms of how to equip an agent with moderate prior knowledge of the environment, including a reward function and state transition dynamics.

Acknowledgments

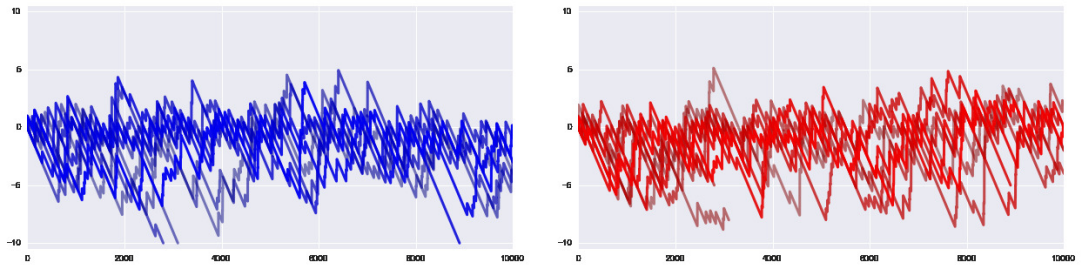
I would like to thank Makoto Otsuka for helpful comments to improve the quality of this paper.

Appendix A. Trajectories of Red and Blue Nutrient Resources in the Initial and Final 10 Trials.



(a) Blue Nutrient Resource (b) Red Nutrient Resource

Figure 12: Initial 10 Trials



(a) Blue Nutrient Resource (b) Rlue Nutrient Resource

Figure 13: Final 10 Trials

Appendix B. Parameters of DQN used in the visual two-resource environment

Table 1: Parameters of DQN

Discount Factor (γ)	0.95
Initial Exploration Step	100,000
Initial Exploration Probability	0.3
Final Exploration Probability	0.1
Initial Exploration Size	1,000
Target Update Interval	5,000
Replay Memory Size	100,000
Batch Size	12
Optimizer	Adam (Kingma and Ba, 2014)
Learning Rate of Adam	0.0001
Epsilon Parameter in Adam	0.001

References

Ashby, W. R. 1960. *Design for a Brain*. Springer Science & Business Media.

Barto, A. G.; Singh, S.; and Chentanez, N. 2004. Intrinsically motivated learning of hierarchical collections of skills. In *Proc. 3rd Int. Conf. Development Learn*, 112–119.

Cañamero, D. 1997. Modeling motivations and emotions as a basis for intelligent behavior. In *Proceedings of the first international conference on Autonomous agents*, 148–155. ACM.

Dawkins, R. 1976. *The Selfish Gene*. Oxford University Press, Oxford, UK.

Dayan, P., and Hinton, G. E. 1996. Varieties of Helmholtz machine. *Neural Networks* 9(8):1385–1403.

Doya, K., and Uchibe, E. 2005. The cyber rodent project: Exploration of adaptive mechanisms for self-preservation and self-reproduction. *Adaptive Behavior* 13(2):149–160.

Elfwing, S.; Uchibe, E.; Doya, K.; and Christensen, H. I. 2005. Biologically inspired embodied evolution of survival. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 3, 2210–2216. IEEE.

Hester, T., and Stone, P. 2012. Learning and using models. In *Reinforcement Learning*. Springer. 111–141.

Jordan, M. I.; Ghahramani, Z.; Jaakkola, T. S.; and Saul, L. K. 1999. An introduction to variational methods for graphical models. *Machine learning* 37(2):183–233.

Kaelbling, L. P.; Littman, M. L.; and Moore, A. W. 1996. Reinforcement learning: A survey. *arXiv preprint cs/9605103*.

Kappen, H. J.; Gómez, V.; and Opper, M. 2012. Optimal control as a graphical model inference problem. *Machine learning* 87(2):159–182.

Keramati, M., and Gutkin, B. S. 2011. A reinforcement learning theory for homeostatic regulation. In *Advances in Neural Information Processing Systems*, 82–90.

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

Kingma, D. P., and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Konidaris, G., and Barto, A. 2006. An adaptive robot motivational system. In *From Animals to Animats 9*. Springer. 346–356.

Lange, S.; Riedmiller, M.; and Voigtlander, A. 2012. Autonomous reinforcement learning on raw visual input data in a real world application. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, 1–8. IEEE.

Lin, L.-J. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning* 8(3-4):293–321.

McFarland, D., and Bösser, T. 1993. *Intelligent behavior in animals and robots*. MIT Press.

McFarland, D., and Houston, A. 1981. *Quantitative ethology*. Pitman Advanced Pub. Program.

McFarland, D., and Spier, E. 1997. Basic cycles, utility and opportunism in self-sufficient robots. *Robotics and Autonomous Systems* 20(2):179–190.

Meyer, J.-A., and Guillot, A. 1991. Simulation of adaptive behavior in animats: Review and prospect. In *In J.-A. Meyer and S.W. Wilson (Eds.) From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*, 2–14.

Mnih, A., and Gregor, K. 2014. Neural variational inference and learning in belief networks. *arXiv preprint arXiv:1402.0030*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Nakamura, M., and Yamakawa, H. 2016. A Game-Engine-Based Learning Environment Framework for Artificial General Intelligence. In *International Conference on Neural Information Processing*, 351–356. Springer.

Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, 278–287.

Ogata, T., and Sugano, S. 1997. Emergence of Robot Behavior Based on Self-Preservation. Research Methodology and Embodiment of Mechanical System. *Journal of the Robotics Society of Japan* 15(5):710–721.

Omohundro, Stephen M, S. M. 2008. The Basic AI Drives. In *Artificial General Intelligence, 2008: Proceedings of the First AGI Conference*, volume 171, 483. IOS Press.

Pfeifer, R., and Scheier, C. 1999. *Understanding intelligence*. MIT press.

Ranganath, R.; Gerrish, S.; and Blei, D. M. 2013. Black box variational inference. *arXiv preprint arXiv:1401.0118*.

Rawlik, K.; Toussaint, M.; and Vijayakumar, S. 2013. On stochastic optimal control and reinforcement learning by approximate inference. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, 3052–3056. AAAI Press.

Rummery, G. A., and Niranjan, M. 1994. *On-line Q-learning using connectionist systems*. University of Cambridge, Department of Engineering.

Rusu, A. A.; Vecerik, M.; Rothörl, T.; Heess, N.; Pascanu, R.; and Hadsell, R. 2016. Sim-to-real robot learning from pixels with progressive nets. *arXiv preprint arXiv:1610.04286*.

Sibly, R., and McFarland, D. 1976. On the fitness of behavior sequences. *American Naturalist* 601–617.

Spier, E. 1997. *From reactive behaviour to adaptive behaviour: motivational models for behaviour in animals and robots.* Ph.D. Dissertation, University of Oxford.

Toda, M. 1962. The design of a fungus-eater: A model of human behavior in an unsophisticated environment. *Behavioral Science* 7(2):164–183.

Toda, M. 1982. *Man, robot, and society: Models and speculations*. M. Nijhoff Pub.

Todorov, E. 2008. General duality between optimal control and estimation. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, 4286–4292. IEEE.

Toussaint, M.; Harmeling, S.; and Storkey, A. 2006. Probabilistic inference for solving (PO) MDPs. Informatics research report 0934, University of Edinburgh.

Toussaint, M. 2009. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 1049–1056. ACM.

Vlassis, N., and Toussaint, M. 2009. Model-free reinforcement learning as mixture learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 1081–1088. ACM.

Walter, W. 1953. *The living brain.* Norton.

Young, J. Z. 1966. *The Memory System of the Brain*. Oxford University Press.