Slowness Learning for Curiosity-Driven Agents

Doctoral Dissertation submitted to the

Faculty of Informatics of the Università della Svizzera Italiana

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

presented by

Varun Raj Kompella

under the supervision of

Prof. Jürgen Schmidhuber

December 2014

Dissertation Committee

Prof. Stefan Wolf	Università della Svizzera italiana, Switzerland
Prof. Matthias Hauswirth	Università della Svizzera italiana, Switzerland
Prof. Laurenz Wiskott	Ruhr-Universität Bochum, Germany
Prof. Srini Narayanan	Google, Zurich & Univ. of California-Berkeley, USA
Prof. Benjamin Kuipers	University of Michigan, Michigan, USA

Dissertation accepted on 18 December 2014

Research Advisor	PhD Program Director
Prof. Jürgen Schmidhuber	**Prof. Igor Pivkin**

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Varun Raj Kompella
Lugano, 18 December 2014

To my beloved parents and soon-to-be wife

Abstract

In the absence of external guidance, how can a robot learn to map the many raw pixels of high-dimensional visual inputs to useful action sequences? I study methods that achieve this by making robots self-motivated (curious) to continually build compact representations of sensory inputs that encode different aspects of the changing environment. Previous curiosity-based agents acquired skills by associating intrinsic rewards with world model improvements, and used reinforcement learning (RL) to learn how to get these intrinsic rewards. But unlike in previous implementations, I consider streams of high-dimensional visual inputs, where the world model is a set of compact low-dimensional representations of the high-dimensional inputs. To learn these representations, I use the *slowness learning* principle, which states that the underlying causes of the changing sensory inputs vary on a much slower time scale than the observed sensory inputs. The representations learned through the slowness learning principle are called slow features (SFs). Slow features have been shown to be useful for RL, since they capture the underlying transition process by extracting spatio-temporal regularities in the raw sensory inputs. However, existing techniques that learn slow features are not readily applicable to curiosity-driven online learning agents, as they estimate computationally expensive covariance matrices from the data via batch processing.

The first contribution called the incremental SFA (IncSFA), is a low-complexity, online algorithm that extracts slow features without storing any input data or estimating costly covariance matrices, thereby making it suitable to be used for several online learning applications. However, IncSFA gradually forgets previously learned representations whenever the statistics of the input change. In open-ended online learning, it becomes essential to store learned representations to avoid re-learning previously learned inputs.

The second contribution is an online active modular IncSFA algorithm called the curiosity-driven modular incremental slow feature analysis (Curious Dr. MISFA). Curious Dr. MISFA addresses the forgetting problem faced by IncSFA and learns expert slow feature abstractions in order from least to most costly, with theoretical guarantees.

The third contribution uses the Curious Dr. MISFA algorithm in a continual curiosity-driven skill acquisition framework that enables robots to acquire, store, and re-use both abstractions and skills in an online and continual manner.

I provide (a) a formal analysis of the working of the proposed algorithms; (b) compare them to the existing methods; and (c) use the iCub humanoid robot to demonstrate their application in real-world environments. These contributions together demonstrate that the online implementations of slowness learning make it suitable for an open-ended curiosity-driven RL agent to acquire a repertoire of skills that map the many raw pixels of high-dimensional images to multiple sets of action sequences.

Acknowledgements

To begin with, I would like to express my deep-felt gratitude towards my supervisor, Jürgen Schmidhuber, who advised, encouraged and supported my ideas right from the beginning of my PhD research at IDSIA. I adopted one of his *mantras* "maximize the cumulative wow-effects in life" as a motto of my life.

My research work in the area of slow feature analysis began from an internship with Mathias Franzius at the Honda Research Institute Offenbach. I thank Mathias for introducing me to the topic and inspiring me to apply it on the Honda ASIMO robot. It was an amazing experience working with Mathias and the ASIMO robot. At the near end of the internship, I felt that robots needed more than just slow features; some unguided means of active exploration, however, I was not able to jot the points together to form a clearer picture. That is when I stumbled across Jürgen's web-page on "Active Exploration, Artificial Curiosity and What's Interesting". At that point, it became clear to me on what topic I wanted to do my PhD research: using slow features for a curiosity-driven agent.

It was an immense pleasure working in IDSIA during the four years of my PhD and I thank all my colleagues for providing a friendly and competitive environment. I am grateful to Matthew Luciw for guiding and motivating me through out my PhD work. It was a pleasure collaborating with him on several research papers and I hope it continues in the future. I thank Alan Lockett, Faustino Gomez and Jan Koutnik for providing valuable suggestions for several of my papers and talks. I had great fun spending time discussing and debating several technical and non-technical topics with Jonathan Masci, Marijn Stollenga, Sohrob Kazerounian, Hung Ngo and many other idsiani. I would like to especially thank Cinzia Daldini, our secretary, for making my life easier by helping me with procedures till the very end of my stay in Lugano.

I also wish to show my indebtedness to my dissertation committee, Benjamin Kuipers, Srini Narayanan, Laurenz Wiskott, Stefan Wolf and Matthias Hauswirth, for reviewing my thesis and providing useful suggestions for my future work.

And finally, I must thank my parents, my brothers and soon-to-be wife, for their loving support and inspiring me to approach and achieve my goals that I had kept during the PhD program. I couldn't have achieved anything without their continued love.

Contents

Chapter 1

Introduction

Over the past decade, there has been a growing trend in humanoid robotics research towards robots with a large number of joints, or degrees of freedom, notably the ASIMO [Honda], PETMAN [Boston-Dynamics] and the iCub [Metta et al., 2008]. These robots demonstrate high dexterity and are potentially capable of carrying out complex human-like manipulation. When interacting with the real world, these robots are faced with several challenges, not least of which is the problem of how to solve tasks that require processing an abundance of high-dimensional sensory data.

In the case of well structured environments, these robots can be carefully programmed by experts to solve a particular task. But real-world environments are usually unstructured and dynamic, which makes it a daunting task to program these robots manually. This problem can be simplified by using reinforcement learning [RL; Kaelbling et al., 1996; Sutton and Barto, 1998], where a robot learns to acquire task-specific behaviors by maximizing the accumulation of task-dependent external rewards through trial-and-error interactions with the environment.

Unfortunately, for humanoid robots equipped with vision, the sensory and joint state space is so large that it is extremely difficult to obtain rewards (if any exist) by random exploration. For example, if the robot receives a reward for sorting objects, it could take an extremely long time to obtain the reward for the first time. Therefore, it becomes necessary to (a) build lower-dimensional representations of the state-space to make learning tractable and (b) to explore the environment efficiently. But how can these robots learn to do this in the presence of external rewards that are typically only sparsely available? This thesis explores this topic.

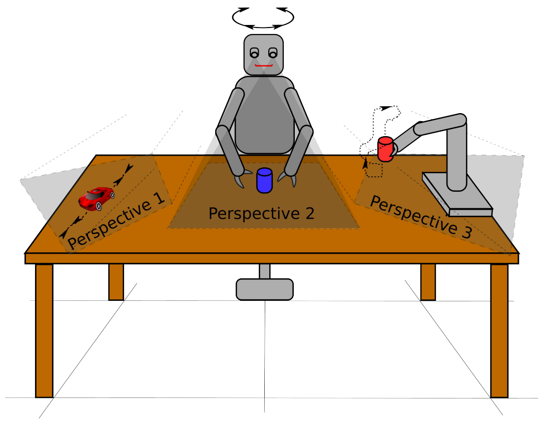


Figure 1.1. A playroom scenario for a baby humanoid-robot in a lab environment, where it is placed next to a table with a few moving objects. The robot has a limited field-of-view and encounters continuous streams of images as it holds or shifts its gaze. The figure shows three such perspectives oriented towards the moving objects. How can the robot learn to solve tasks in the absence of external guidance?

1.1 Approach to Task-Learning

Much of the human capacity to explore and solve problems is driven by self-supervised learning [White, 1959; Norman and Schmidt, 1992], where we seek to acquire behaviors by creating novel situations and learning from them. As an example consider a simple playroom scenario for a baby humanoid as shown in Figure 1.1. Here, the robot is placed next to a table with a few moving objects. The robot has a limited field-of-view and encounters continuous streams of images as it holds or shifts its gaze. If the robot can learn compact representations and predictable behaviors (*e.g.,* grasping) from its interactions with the cup, then by using these learned behaviors, it can speed up the acquisition of external rewards related to some teacher-defined task, such as placing the cup at a particular location. Continually acquiring and reusing a repertoire of behaviors and representations of the world, learned through self-supervision, can therefore make the robot adept in solving many external tasks.

But how can the robot (a) self-supervise its exploration, (b) build representations of high-dimensional sensory inputs, and (c) continually acquire skills that enable it to solve new tasks? These problems have been researched separately in the machine learning and robotics literature [Jolliffe, 1986; Abut, 1990; Schmidhuber, 1992a,c,b; Lindstädt, 1993; Comon, 1994; Ring, 1994; Lee and Seung, 1999; Kohonen, 2001; Klapper-Rybicka et al., 2001; Hinton, 2002; Wiskott and Sejnowski, 2002; Jenkins and Matarić, 2004; Singh et al., 2004; Hart et al., 2008; Lee et al.; Stout and Barto, 2010; Konidaris et al., 2011; Gisslén et al., 2011; Pape et al., 2012]. However, to develop a *single* system that addresses all these important issues together is a challenging open problem in artificial intelligence (AI) research. In this thesis, I propose an online-learning framework that addresses this open problem.

In order to make the robot self-supervised or intrinsically-motivated to explore new environments, I use the theory of Artificial Curiosity [AC; Schmidhuber, 2006b, 2010b]. AC mathematically describes curiosity and creativity. AC-driven agents are interested in the learnable but as-yet-unknown aspects of their environment and are disinterested in the already learned and inherently unlearnable (noisy) aspects. Specifically, the agent receives *intrinsic rewards* for action sequences, and these rewards are proportional to the improvement of the agent's internal model or predictor of the environment. Using RL and the self-generated intrinsic rewards derived using AC [Schmidhuber, 1991b; Storck et al., 1995; Schmidhuber, 1999a, 2006a, 2010a; Pape et al., 2012], the agent is motivated to explore the environment where it makes maximum learning progress.

Previous implementations of the curiosity theory have been applied only to low-dimensional inputs or simple domains. I consider streams of high-dimensional visual inputs where the internal world model of the agent is a set of compact low-dimensional *abstractions* of the environment. An *abstraction* maps the high-dimensional input to a low-dimensional output. But how can these compact abstractions be learned without a teacher? The high-dimensional data sensed by a robot is often temporally correlated and can be greatly compressed into compact abstractions if the temporal coherence in the data is exploited. Slow Feature Analysis [SFA; Wiskott and Sejnowski, 2002; Franzius et al., 2007; Legenstein et al., 2010] is an unsupervised learning algorithm that extracts temporal regularities from rapidly changing raw sensory inputs. SFA is based on the Slowness Principle [Földiák and Young, 1995; Mitchison, 1991; Wallis and Rolls, 1997], which states that the underlying causes of changing signals vary more slowly than the primary sensory stimulus. For example, individual retinal receptor responses or gray-scale pixel values of video may change quickly compared to latent abstract variables, such as the position of a moving object. SFA has achieved success in many problems and scenarios, e.g., extraction of driving forces of a dynamical system [Wiskott, 2003], nonlinear blind source separation [Sprekeler et al., 2014], preprocessing for reinforcement learning [Legenstein et al., 2010; Kompella et al., 2011b], learning of place-cells, head-direction cells, grid-cells, and spatial view cells from high-dimensional visual input [Franzius et al., 2007], dynamic scene classification [Theriault et al., 2013], recognition of postures of a biped humanoid robot [Höfer et al., 2012], and human action sequences [Zhang and Tao, 2012; Sun et al., 2014].

Existing SFA techniques are not readily applicable to curiosity-driven online learning agents, as they estimate covariance matrices from the data via batch processing. The next section explores how online implementations of SFA make it suitable for an open-ended curiosity-driven RL agent to acquire a repertoire of skills that map the high-dimensional inputs to multiple sets of action sequences.

1.2 Contributions

In this section, I will briefly list out the contributions of this thesis that address the open problems discussed earlier.

My first contribution, called the Incremental Slow Feature Analysis [IncSFA; Kompella et al., 2011a, 2012a], is a low complexity, online implementation of batch SFA (BSFA). IncSFA extracts slow features without storing any input data or estimating costly covariance matrices. A few earlier techniques with temporal continuity objectives were incremental as well [Hinton, 1989; Bergstra and Bengio, 2009]. But IncSFA follows the SFA formulation and uses *hebbian* and *anti-hebbian* update rules to extract features that would be uncovered by BSFA, over which it has the following advantages: (a) it is adaptive to changing input statistics; (b) it has linear computational efficiency as opposed to cubic of BSFA; (c) it has reduced sensitivity to outliers; and (d) it adds to the biological plausibility of BSFA. These advantages make IncSFA suitable to use for several online learning applications. However, in the case of open-ended curiosity-driven RL, IncSFA has a shortcoming. IncSFA, like most online learning approaches, gradually forgets previously learned representations whenever the statistics of the input change, for example, when the robot shifts its gaze among the perspectives of Figure 1.1. It becomes essential to store learned representations to avoid re-learning previously learned inputs.

My second contribution is an online active modular IncSFA algorithm, called Curiosity-Driven Modular Incremental Slow Feature Analysis [Curious Dr. MISFA; Kompella* et al., 2013; Kompella et al., 2012b]. Curious Dr. MISFA uses the theory of artificial curiosity to address the forgetting problem faced by IncSFA, by retaining what was previously learned in the form of *expert modules* [Ring, 1994]. From a set of input video streams, Curious Dr. MISFA actively learns multiple expert modules comprising slow feature abstractions in the order of increasing learning difficulty, with theoretical guarantees. These theoretical optimality guarantees were lacking in previous practical implementations of the curiosity theory [Schmidhuber, 2010b]. The algorithm continually estimates the initially unknown learning difficulty through intrinsic rewards generated by exploring the input streams. Using Curious Dr. MISFA, the robot in Figure 1.1 finds its interactions with the plastic cup more interesting (easier to encode) than the complex movements of the other ob-

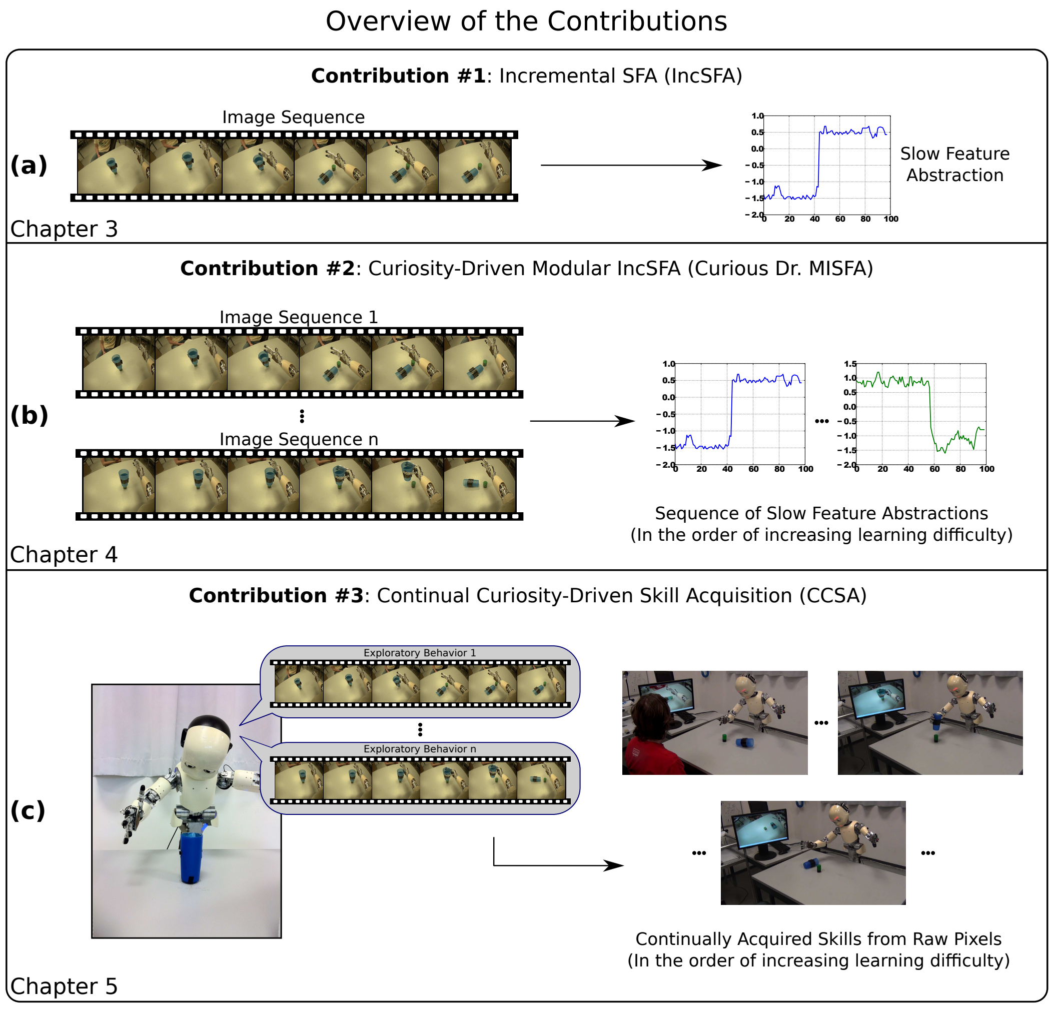

Overview of the Contributions
Contribution #1: Incremental SFA (IncSFA)
(a)
Image Sequence
Slow Feature Abstraction
Chapter 3
Contribution #2: Curiosity-Driven Modular IncSFA (Curious Dr. MISFA)
(b)
Image Sequence 1
Image Sequence n
Sequence of Slow Feature Abstractions
(In the order of increasing learning difficulty)
Chapter 4
Contribution #3: Continual Curiosity-Driven Skill Acquisition (CCSA)
(c)
Exploratory Behavior 1
Exploratory Behavior n
Continually Acquired Skills from Raw Pixels
(In the order of increasing learning difficulty)
Chapter 5

Figure 1.2. Overview of the contributions. (a) Incremental Slow Feature Analysis (IncSFA): Learns a slow feature abstraction from a raw image sequence. (b) Curiosity-Driven Modular IncSFA (Curious Dr. MISFA): Learns multiple slow feature abstractions from multiple image sequences, in the order of increasing learning difficulty. (c) Continual Curiosity-Driven Skill Acquisition (CCSA): Translates slow feature abstraction-learning problem to a continual curiosity-driven skill acquisition problem.

jects. This results in a compact slow feature abstraction that encodes its interactions with the cup. Eventually, the robot finds the cup-interaction boring and its interest shifts towards encoding other perspectives while retaining the learned abstraction. Can the robot simultaneously acquire reusable skills while acquiring abstractions? Each learned abstraction encodes some previously unknown regularity in the input observations that can then be used as a basis for acquiring new skills.

As a final contribution, I propose a framework for Continual Curiosity-Driven Skill Acquisition [CCSA; Kompella et al., 2014b] for acquiring, storing and re-using both abstractions and skills in an online and continual manner. CCSA uses the Curious Dr. MISFA algorithm to learn a slow feature abstraction that encodes the easiest to learn yet unknown regularity in the streams of high-dimensional visual information. This representation augments the robot's state space with new information about the environment. I show how this information can have a higher-level (compared to pixels) and useful interpretation, for example, if the robot has grasped a cup in its field of view or not. After learning a representation, large intrinsic rewards are given to the robot for performing actions that greatly change the feature output, which has the tendency otherwise to change slowly in time. An acquired skill includes both the learned actions and the learned slow feature representation. Skills are stored and reused to generate new observations, enabling continual acquisition of complex skills. In the experiments, using CCSA, an iCub humanoid robot addresses the open problems discussed earlier, acquiring a repertoire of skills (topple, grasp) from raw-pixel vision, driven purely by its intrinsic motivation. Figure 1.2 summarizes the contributions of this thesis.

1.3 Thesis Outline

The outline of the thesis is as follows. Chapter 2 presents some related background to make the thesis stand on its own. Chapters 3, 4, 5 present details of my contributions: Incremental SFA, Curious Dr. MISFA and CCSA algorithms respectively, along with their experimental results. Chapter 6 discusses related research work carried out by other researchers prior to this thesis and concludes by presenting insights for future work. Appendix A presents detailed proofs of the all the theorems presented in the thesis.

Chapter 2

Background

This chapter provides the necessary background to the research topics presented in the following chapters. The chapter begins with a discussion on slow feature analysis (Section 2.1), followed by an overview or a mini-introduction to reinforcement learning (Section 2.2) and the theory of artificial curiosity (Section 2.3). Due to vastness of these background topics, the description presented in each of these sections is not intended to be comprehensive, but merely to enable the subsequent discussion.

2.1 Slow Feature Analysis

At the core of all my contributed methods is the Slow Feature Analysis [SFA; Wiskott and Sejnowski, 2002]. I use SFA to compactly encode regularities in the localized image-stream of a robot. Here, I present a brief overview of SFA. SFA is a form of unsupervised learning (UL) in which like in principal component analysis [PCA; Jolliffe, 1986], the algorithm searches for a set of mappings $g_i, i \in \mathbb{N}$ from data $\mathbf{x} \in \mathscr{R}^I, I \in \mathbb{N}$ to output components $\mathbf{y}_i = g_i(\mathbf{x}), \mathbf{y} \in \mathscr{R}^J, J \in \mathbb{N}$ that are *separate* from each other in some sense and express information that is in some sense *relevant*. In SFA the features are separated via mutual decorrelation of their outputs, while relevance is defined as *minimal but nonzero change over time* (slowness). Ordering the functions $g_1, g_2, ..., g_I$ by slowness, we can discard all but the $J < I$ slowest, getting rid of irrelevant information such as quickly changing noise assumed to be useless. See Fig. 2.1 for a visual example of the meaning of a slow feature.

SFA-based UL learns *instantaneous* features from sequential data [Hinton, 1989; Wiskott and Sejnowski, 2002; Doersch et al., 2010]. Relevance cannot be *uncovered* without taking time into account, but once it is known, each input frame in most cases can be encoded on its own. Due to this, SFA differs from both (1) many well-

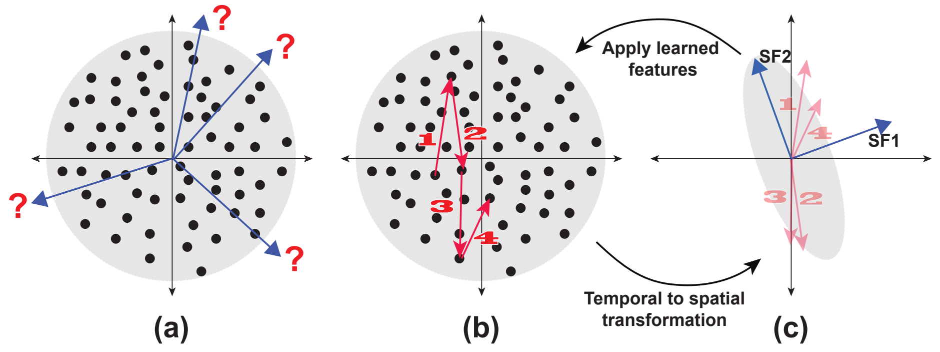


Figure 2.1. A toy example to explain what a slow feature is. (A) Consider a zero-mean input signal that spatially resembles white noise. Input points (the black dots) are drawn from within the gray circle area. Linear spatial feature extractors (such as PCA) will not prefer any direction over any other (since for PCA the variance in all directions is the same). (B) If we recode the data in terms of how it changes between subsequent time instants, certain directions can be more informative than others. Here, the arrows show a short representative sequence of input. All difference vectors (not just the four shown) create the space shown in (C). In this space, the second principal component, or the minor component — gives the direction of the slowest change (the slowest feature SF1). While, the first principal component gives the direction of the quickest change (SF2).

known unsupervised feature extractors [Abut, 1990; Jolliffe, 1986; Comon, 1994; Lee and Seung, 1999; Kohonen, 2001; Hinton, 2002] that ignore dynamics, and (2) other UL systems that both learn and apply features to sequences [Schmidhuber, 1992a,c,b; Lindstädt, 1993; Klapper-Rybicka et al., 2001; Jenkins and Matarić, 2004; Lee et al.; Gisslén et al., 2011], thereby assuming that the state of the system itself can depend on past information.

The compact relevant encodings uncovered by SFA reduce the search space for downstream goal-directed learning procedures [Schmidhuber, 1999b; Barlow, 2001], especially reinforcement learning. As an example, consider a robot sensing with an onboard camera. Reinforcement learning algorithms applied directly to pixels can be quite inefficient due to the size of the search space. Slow features can encode each image into a small set of useful state variables, and the robot can use these few state variables to quickly develop useful control policies. The state variables from SFA are approximations of low-order eigenvectors of the graph Laplacian [Sprekeler, 2011], i.e., proto-value functions [Mahadevan and Maggioni, 2007]. This is why they are typically more useful as features in reinforcement learning in comparison with other types of features, such as principal components.

2.1.1 Formulation

SFA optimizes the following formal problem [Wiskott and Sejnowski, 2002]: *given an I-dimensional sequential input signal* $\mathbf{x}(t) = [x_1(t), ..., x_I(t)]^T$*, find a set of* J *instantaneous real-valued functions* $\mathbf{g}(x) = [g_1(\mathbf{x}), ..., g_J(\mathbf{x})]^T$*, which together generate a* J*-dimensional output signal* $\mathbf{y}(t) = [y_1(t), ..., y_J(t)]^T$ *with* $y_j(t) := g_j(\mathbf{x}(t))$*, such that for each* $j \in \{1, ..., J\}$

$$\Delta_j := \Delta(y_j) := \langle \dot{y}_j^2 \rangle \quad \textit{is minimal} \tag{2.1}$$

under the constraints

$$\langle y_j \rangle = 0 \quad \textit{(zero mean)}, \tag{2.2}$$
$$\langle y_j^2 \rangle = 1 \quad \textit{(unit variance)}, \tag{2.3}$$
$$\forall i < j : \langle y_i y_j \rangle = 0 \quad \textit{(decorrelation and order)}, \tag{2.4}$$

with $\langle \cdot \rangle$ *and* $\dot{y}$ *indicating temporal averaging and the derivative of* y*, respectively.*

The problem is to find instantaneous functions g_j that generate different output signals varying as slowly as possible. The constraints (2.2) and (2.3) together avoid a trivial constant output solution. The decorrelation constraint (2.4) ensures that different functions g_j do not code for the same features.

2.1.2 Solution

Solving this learning problem requires variational calculus and is in general difficult to solve [Franzius et al., 2007]. But a linear-approximate solution to the problem can be found through a simpler eigenvector approach. If the g_j are linear combinations of a finite set of nonlinear functions $\mathbf{h}$, then

$$y_j(t) = g_j(\mathbf{x}(t)) = \mathbf{w}_j^T\ \mathbf{h}(\mathbf{x}(t)) = \mathbf{w}_j^T\ \mathbf{z}(t), \tag{2.5}$$

and the SFA problem now becomes how to find weight vectors $\mathbf{w}_j$ to minimize the rate of change of the output variables,

$$\Delta(y_j) = \langle \dot{y}_j^2 \rangle = \mathbf{w}_j^T\ \langle \dot{\mathbf{z}}\dot{\mathbf{z}}^T \rangle\ \mathbf{w}_j, \tag{2.6}$$

subject to the constraints (2-4). The slow feature learning problem has become linear on the derivative signal $\dot{\mathbf{z}}$.

If the functions of $\mathbf{h}$ are chosen such that $\mathbf{z}$ has identity covariance matrix and zero mean, the three constraints will be fulfilled if and only if the weight vectors $\mathbf{w}_j$ are orthonormal. Eq. 2.6 will be minimized, and the orthonormal constraint satisfied, with the set of J normed eigenvectors of $\langle \dot{\mathbf{z}}\dot{\mathbf{z}}^T \rangle$ with the J smallest eigenvalues (for any $J \leq I$).

The batch SFA (BSFA) technique implements this solution by using batch PCA twice. Referring back to Eq. 2.6, to select $\mathbf{h}$ appropriately, a well-known process called whitening (or sphering) is used to map $\mathbf{x}$ to a $\mathbf{z}$ with zero mean and identity covariance matrix, thus decorrelating signal components and scaling them such that there is unit variance along each principal component (PC) direction. Whitening serves as a bandwidth normalization, so that slowness can truly be measured (slower change will not simply be due to a low variance direction). Whitening requires the PCs of the input signal (PCA #1). The orthonormal basis that minimizes the rate of output change are the minor components – principal components with smallest eigenvalues – in the derivative space. So, another PCA (#2) on $\dot{\mathbf{z}}$ yields the slow features (eigenvectors) and their order (via eigenvalues). In Chapter 3, I will present an alternative incremental solution that avoids the computation of the costly covariance matrices required by the batch PCAs.

2.2 Reinforcement Learning

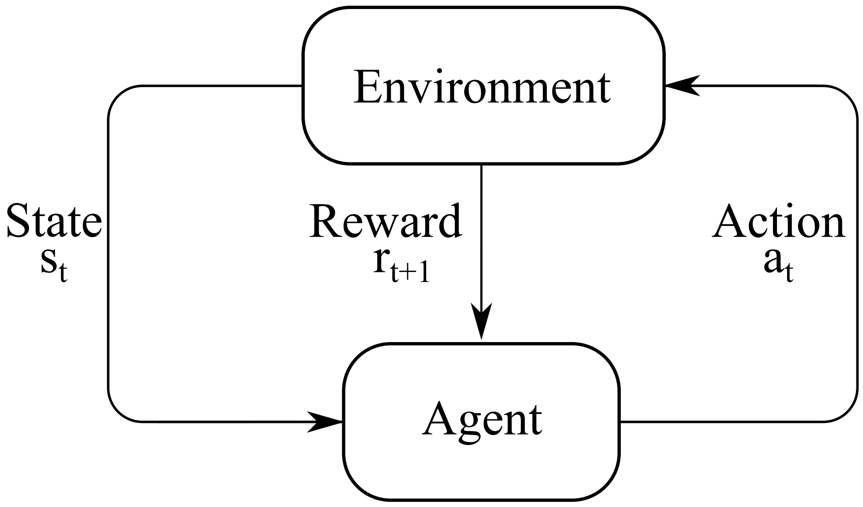


Figure 2.2. Agent-environment interactions in reinforcement learning.

In this section, I will present a brief overview of the relevant topics in reinforcement learning [RL; Kaelbling et al., 1996; Sutton and Barto, 1998]. For a more comprehensive review, refer to the books by Sutton and Barto [1998] and Szepesvári [2010]. The reinforcement learning problem is summarized in Figure 2.2. An agent interacts with its environment at each time t by first observing the current state s_t

followed by taking an action a_t. For each action taken, it collects a real-valued reward r_{t+1} as an outcome. The goal of the agent is to maximize the accumulation of rewards received over time. RL has been formally studied under the framework of Markov Decision Processes (MDPs).

2.2.1 Markov Decision Process

A *Markov Decision Process* for the reinforcement learning problem is defined as a 5-tuple $(\mathscr{S},\mathscr{A},\mathscr{P},\mathscr{R},\gamma)$. Each element of this tuple is described as follows. An agent is in an environment that has a finite state space $\mathscr{S}$. At each state $s \in \mathscr{S}$, the agent carries out an action $a \in \mathscr{A}$ and transitions to a new state $s' \in \mathscr{S}$ according to a transition model defined as a probability mass function $\mathscr{P} : \mathscr{S} \times \mathscr{A} \times \mathscr{S} \to [0,1]$, such that $\sum\limits_{s' \in \mathscr{S}} \mathscr{P}(s'|s,a) = 1$. $\mathscr{R} : \mathscr{S} \times \mathscr{A} \times \mathscr{S} \to \mathbb{R}$ is the reward model that defines the task of the reinforcement learning problem and γ represents the discount factor based on which future reward values are discounted. The defining characteristic of an MDP is that the environment and the task defined by $\mathscr{R}$ has the *Markov* property, which is stated as follows [Sutton and Barto, 1998]: $\forall s', s_t \in \mathscr{S}, a_t \in \mathscr{A}$ the following equality holds

$$\Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t, r_t, ..., r_1, s_0, a_0\} = \Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t\} \quad (2.7)$$

where Pr denotes the probability distribution. The Markov property guarantees that the best policy for selecting actions as a function of a state is the same as a function of complete histories of states. The policy π can be defined as a state dependent probability mass function $\pi : \mathscr{S} \times \mathscr{A} \to [0,1]$, such that $\sum\limits_{a \in \mathscr{A}} \pi(s,a) = 1$.

2.2.2 Bellman Equation

While the reward function determines what is good in the immediate sense, a *value function* represents whether a particular state is *valuable* in the long run, w.r.t achieving the goal of the RL problem. In other words, the value of a state $V(s)$ is equal to the total amount of reward the agent can expect to accumulate over the future, starting from the state s.

Reinforcement learning can be broadly divided into *prediction* and *control* problems. The goal of the *prediction* problem is to find the value-function for a given policy. This is also called *policy evaluation*. While following a policy π, the value function serves as a prediction for rewards in the future:

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,\middle|\, s_t = s \right\}, \ \forall s \in \mathscr{S}, \tag{2.8}$$

where $\mathbb{E}_{\pi}$ denotes the expected value given the agent follows the policy π. Eq. 2.8 can be expressed as a recursive equation, called the *Bellman equation for* V^{π}:

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s' \in \mathscr{S}} \mathscr{P}(s,a,s')[\mathscr{R}(s,a,s') + \gamma V^{\pi}(s')], \ \forall s \in \mathscr{S}. \tag{2.9}$$

If the dynamics of the environment (transition-probability function $\mathscr{P}$ and reward-function $\mathscr{R}$) are completely known, then the value function V^{π} is a unique solution to its Bellman equation and can be found using *iterative policy evaluation*: The right-hand side of the Eq. 2.9 can be interpreted as a mathematical contraction-operator [Bertsekas and Tsitsiklis, 1995] called the Bellman operator T^{π}, which maps a value-function to another value-function. Starting with some initial V, due to Banach's fixed point theorem, Eq. 2.9 converges to V^{π} upon repeatedly applying the Bellman operator.

The goal of the *control* problem is to optimize the value-function by finding an optimal policy. The optimal policy can be found using *policy iteration*: Upon determining the value function V^{π} for an arbitrary policy π, the policy is improved to π' by choosing actions that lead to states with the highest values. The new policy π' is then evaluated and further improved in an iterative manner

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \ldots \xrightarrow{I} \pi^* \xrightarrow{E} V^*, \tag{2.10}$$

where $\xrightarrow{E}$ denotes a policy evaluation and $\xrightarrow{I}$ denotes a policy improvement.

The policy-evaluation step (Eq. 2.9) in the policy iteration requires multiple sweeps through the state-set. The *value iteration* algorithm truncates the policy-evaluation step and combines it with the policy improvement into a recursive equation as follows:

$$V(s) \leftarrow \max_{a \in \mathscr{A}} \left(\sum_{s' \in \mathscr{S}} \mathscr{P}(s,a,s')[\mathscr{R}(s,a,s') + \gamma V(s')] \right), \ \forall s \in \mathscr{S}. \tag{2.11}$$

The optimal value function $V^*(s)$ can be then found by applying the iterative policy evaluation technique and the optimal policy π^* chooses actions that always lead to states with the highest value:

$$V^*(s) = \max_{a \in \mathscr{A}} \left(\sum_{s' \in \mathscr{S}} \mathscr{P}(s,a,s')[\mathscr{R}(s,a,s') + \gamma V^*(s')] \right), \ \forall s \in \mathscr{S}, \tag{2.12}$$

$$\pi^*(s) = \arg\max_{a \in \mathscr{A}} \left(\sum_{s' \in \mathscr{S}} \mathscr{P}(s,a,s')[\mathscr{R}(s,a,s') + \gamma V^*(s')] \right), \ \forall s \in \mathscr{S}. \tag{2.13}$$

For control with finite action spaces, action values are often used. The action-value function is defined as $Q : \mathscr{S} \times \mathscr{A} \rightarrow \mathbb{R}$, which represents whether a particular (s,a) tuple is valuable in the long run. Formally,

$$Q^\pi(s,a) = \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,\middle|\, s_t = s, a_t = a \right\} \tag{2.14}$$

$$= \sum_{s' \in \mathscr{S}} \mathscr{P}(s,a,s')[\mathscr{R}(s,a,s') + \gamma V^\pi(s')], \ \forall s \in \mathscr{S}, a \in \mathscr{A}. \tag{2.15}$$

The optimal action-value function and policy are given by:

$$Q^*(s,a) = \sum_{s' \in \mathscr{S}} \mathscr{P}(s,a,s')[\mathscr{R}(s,a,s') + \gamma \max_{a'} Q^*(s',a')], \ \forall s \in \mathscr{S}, a \in \mathscr{A}, \tag{2.16}$$

$$\pi^*(s) = \arg\max_{a \in \mathscr{A}} Q^*(s,a), \ \forall s \in \mathscr{S}. \tag{2.17}$$

2.2.3 Temporal Difference Learning

Value iteration is a model-based approach where the transition model $\mathscr{P}$ and the reward model $\mathscr{R}$ are known. However, in practice the transition and reward models are generally unknown. Temporal Difference (TD) learning methods enable reinforcement learning in model-free settings, where no explicit knowledge of the environment model is required. By just using the transition samples $(s_t, a_t, r_{t+1}, s_{t+1})$ generated by a policy, TD methods that estimate the value function, in most cases (such as table-based and linear function approximation, see [Sutton and Barto, 1998]) are guaranteed to converge to V^π. The simplest TD method, known as TD(0), is

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]. \tag{2.18}$$

For control, TD methods use policy iteration and fall into two main classes: *on-policy* and *off-policy*. On-policy methods, such as SARSA (see Chapter 6 Section 4 of the book by Sutton and Barto [1998]), attempt to evaluate and improve the policy that is used to make decisions. An example update equation for action-value function using SARSA is as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right], \tag{2.19}$$

where $0 < \alpha < 1$ is a learning rate that decreases over time. Balancing exploration and exploitation becomes very important here and an ϵ-greedy strategy [Sutton and Barto, 1998] is popularly used where the agent mostly chooses actions that maximize expected action values but occasionally chooses a random action with probability ϵ.

On the other hand, off-policy methods, such as Q-Learning (see Chapter 6 Section 5 of the book by Sutton and Barto [1998]), use a policy to generate behavior (often called a *behavior* policy) that may be unrelated to the policy that is evaluated and improved (often called *target* policy). An example update equation for action-value function using Q-Learning is as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]. \tag{2.20}$$

An advantage of the off-policy methods is that the target-policy may be deterministic while following a stochastic behavior policy.

Approximate Temporal Difference Learning

We have assumed thus far that the value functions (V or Q) can be represented in an appropriate way, for example, in the form of a table. However, when dealing with high-dimensional state spaces, table-based representations become infeasible and they do not generalize to unseen transition samples. This problem is alleviated by representing value functions using *function approximators*, such as linear functions or artificial neural networks. Linear function approximators are popularly used, since they are easy to implement and there exist convergence guarantees for TD methods using them [Tsitsiklis and Roy, 1997]. A few examples are: least squares temporal difference [LSTD; Bradtke and Barto, 1996] for prediction problems and least squares policy iteration [LSPI; Lagoudakis and Parr, 2003] for control. Least squares methods make efficient use of data as compared to the conventional TD methods.

I use LSPI in my work, so I will present a few more details on the algorithm. LSPI is an off-policy algorithm that combines the efficient computation of the action-value function of a fixed policy (LSTDQ) and the policy iteration. The action-value function $Q^\pi(s, a)$ is approximated ($\widehat{Q}^\pi(s, a; w)$) by linear parametric combinations of k fixed basis functions (features) and w free parameters:

$$\widehat{Q}^{\pi}(s,a;w) = \sum_{j=1}^{k} \psi_j(s,a) w_j \tag{2.21}$$

The basis functions are generally required to have compact descriptions, be linearly independent and have $k \ll |\mathscr{S}||\mathscr{A}|$. Popularly used basis functions are polynomials, radial basis functions and proto-value functions [Mahadevan and Maggioni, 2007]. Given the environment model, the parameters w for a given policy can be found analytically:

$$w^{\pi} = \left(\Psi^T(\Psi - \gamma P \Pi \Psi)\right)^{-1} \Psi^T R, \tag{2.22}$$

where Ψ denotes the basis function matrix, P is a matrix that contains the transition model of the process ($P((s,a),s') = \mathscr{P}(s,a,s')$), R is the reward vector and Π is a matrix that describes the policy π: $\Pi(s',(s',a')) = \pi(a';s')$.

In the absence of the environment model (P, R), the terms $A = \Psi^T(\Psi - \gamma P \Pi \Psi)$ and $b = \Psi^T R$ can be approximated using the transition samples (s,a,r,s') via the following update rules:

$$\widehat{A} \leftarrow \widehat{A} + \psi(s,a)(\psi(s,a)^T - \gamma \psi(s',\pi(s'))^T) \quad \text{and} \quad \widehat{b} \leftarrow \widehat{b} + \psi(s,a) r.$$

The policy improvement in LSPI is carried out by:

$$\pi_{t+1}(s,w) = \arg\max_a \widehat{Q}(s,a;w) = \arg\max_a \psi(s,a)^T w. \tag{2.23}$$

It has been shown that LSPI is a stable algorithm [Lagoudakis and Parr, 2003]. Chapters 4 and 5 present more details on how LSPI can be used to learn behaviors for a humanoid robot.

2.3 Theory of Artificial Curiosity

Here I present a brief overview of the theory of *Artificial Curiosity* [AC; Schmidhuber, 2006a, 2010a]. The theory of AC mathematically formalizes driving forces and value functions behind all kinds of curious and creative behavior. Consider an agent living in an initially unknown environment. At any given time, it uses one of the many reinforcement learning (RL) methods [Kaelbling et al., 1996] to maximize not only expected future external reward for achieving certain goals, such as avoiding hunger, empty batteries, obstacles etc., but also *intrinsic* reward for action

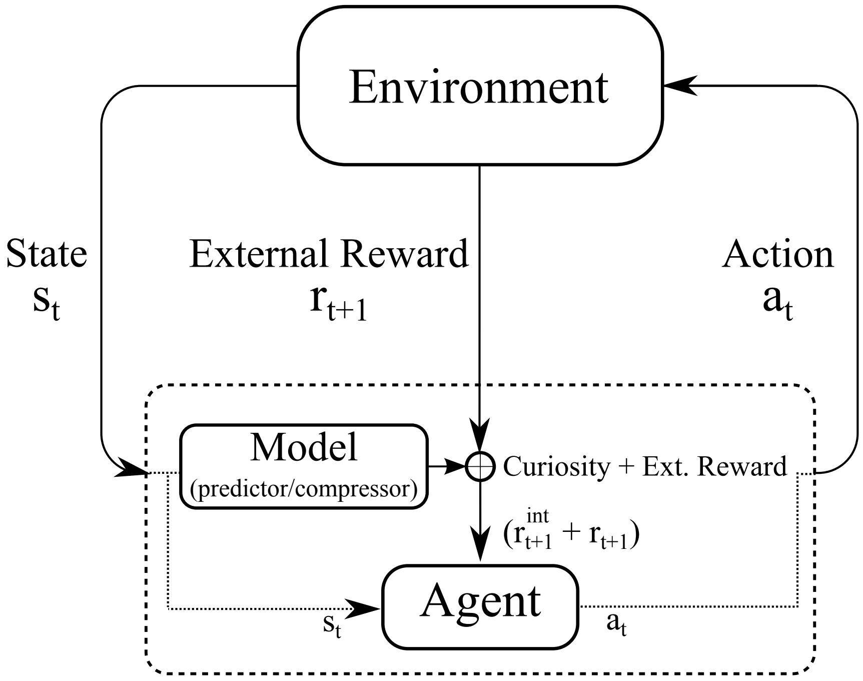


Figure 2.3. Agent-environment interactions in Curiosity-driven reinforcement learning.

sequences that improve an internal model of the environmental responses to its actions (see Figure 2.3). Such an agent continually learns to better predict, explain or compress the growing history of observations influenced by its experiments, actively influencing the input stream such that it contains previously unknown but learnable algorithmic regularities that become known and boring once there is no additional subjective *compression progress* or *learning progress* [Schmidhuber, 1991b; Storck et al., 1995; Schmidhuber, 1999a, 2010a]. Schmidhuber et al. have argued that the particular utility functions based on compression progress as described in this theory explain essential aspects of intelligence including selective attention, curiosity, creativity, science, art, music, humor, e.g., [Schmidhuber, 2006a, 2010a].

Essentially, curiosity-driven agents not only focus on potentially hard-to-solve externally posed tasks but also creatively invent self-generated tasks that have the property of currently being still unsolvable but easily learnable given the agent's present knowledge, so that the agent is continually motivated to improve its understanding of how the world works and what can be done in it. Its growing skill repertoire may at some point help to achieve greater external rewards as well [Schmidhuber, 1991b, 1999a, 2010a].

2.4 Conclusion

With the above briefly discussed topics as a background, the following chapters present the contributions of this thesis in detail. Firstly, in Chapter 3, I will present an incremental low-complex implementation of the batch SFA (discussed in Section 2.1), making it feasible to use SFA for online learning applications. I show in Chapter 4 how a curiosity-driven reinforcement learning agent (discussed in Sections 2.2, 2.3) can be partially approximated to achieve an intrinsic goal of compressing and predicting the observation history. SFA is effective at discovering invariant spatio-temporal properties of the input stream, supporting this goal. Note that any such invariance must reflect an environmental regularity that allows for better compression of the observed data. Hence a curious, playful robot can be implemented (Chapter 5) by simply making it wish to learn in order to create additional, still unknown, SFA-encodable invariances.

Chapter 3

Incremental Slow Feature Analysis

As discussed in the last chapter slow feature analysis (SFA) captures the invariant and slowly varying features from input signals and has been successfully applied in many problems and scenarios. SFA is elegant and sample efficient, but it has only been applied to data in batches. Therefore, it is not readily applicable to online learning agents because it estimates the covariance matrices from all data offline. I present here an incremental version of SFA [IncSFA; Kompella et al., 2011a, 2012a] that does not need to store any input data or computationally expensive covariance matrix estimates. This makes it feasible for handling high-dimensional image data in an online manner.

3.1 Overview

In this section, I will present an overview of the IncSFA algorithm. Like batch SFA (BSFA), IncSFA employs the eigenvector tactic but uses incremental algorithms for the two required principal component analysis steps (PCAs; see Section 2.1.2). Therefore, IncSFA can update existing slow feature estimates on any amount of new data, even on a single data point $\mathbf{x}(t)$.

Figure 3.1 shows the control flow of both BSFA and IncSFA algorithms. IncSFA replaces the batch PCA algorithms with their incremental alternatives. To replace PCA #1, IncSFA needs to incrementally whiten the input $\mathbf{x}$. To this end, I use the state-of-the-art Candid Covariance-Free Incremental PCA [CCIPCA; Weng et al., 2003]. CCIPCA incrementally updates both the eigenvectors (the principal components) and eigenvalues necessary for whitening, and does not keep an estimate of the covariance matrix. It has been shown that the PCs learned by CCIPCA converge to the true PCs [Zhang and Weng, 2001]. CCIPCA is optionally used to reduce dimensionality at this intermediate stage by only computing the K highest-order

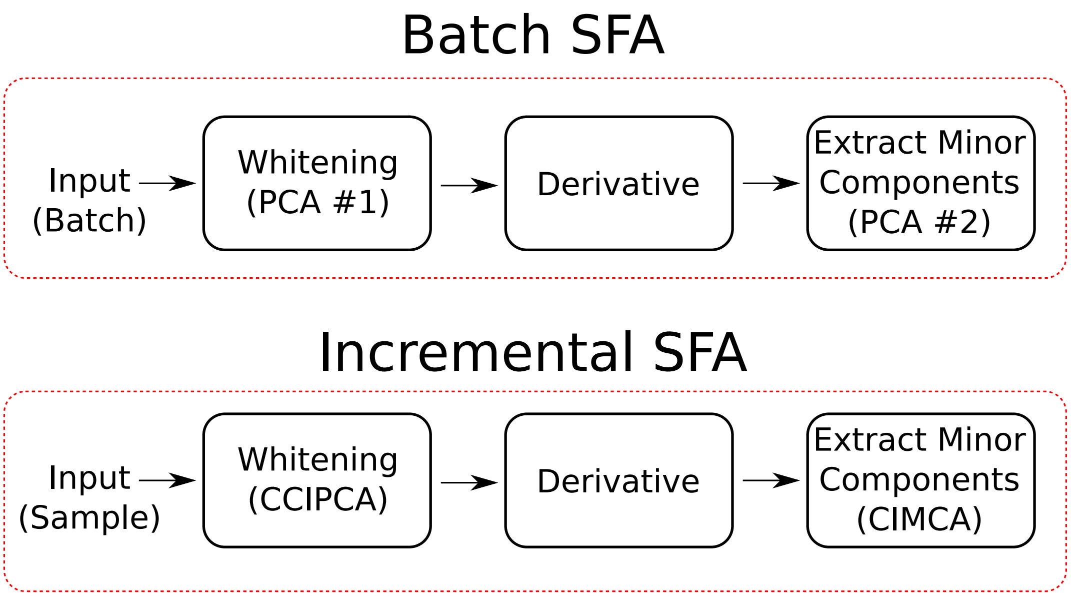


Figure 3.1. Control flow of Batch and Incremental SFA algorithms. IncSFA replaces the batch PCAs with their incremental alternatives.

eigenvectors.

Except for the low-dimensional derivative signals $\dot{\mathbf{z}}$, CCIPCA cannot replace the second PCA step. It takes a long time to converge to the slow features, since they correspond to the least significant components. Minor Components Analysis [MCA; Oja, 1992] incrementally extracts principal components, but with a reversed preference: it extracts the components with the smallest eigenvalues fastest. I use a modified version of Peng's low complexity MCA updating rule [Peng et al., 2007]. Peng proved its convergence even for constant learning rates—good for open-ended learning. MCA with sequential addition [Chen et al., 2001; Peng and Yi, 2006] will extract multiple slow features in parallel. In IncSFA, this method is modified to be covariance-free. A high-level formulation of IncSFA is

$$(\phi(t+1), \mathbf{V}(t+1)) = IncSFA\left(\phi(t), \mathbf{V}(t), \mathbf{x}(t), \theta(t)\right), \qquad (3.1)$$

where $\phi(t) = \left(\phi_1(t), ..., \phi_J(t)\right)$ is the matrix of existing slow feature vector estimates for J slow features and $\mathbf{V} = \left(\mathbf{v}_1, ..., \mathbf{v}_K\right)$ is the matrix of K principal component vector estimates used to construct the whitening matrix and for dimensionality-reduction.[1] Here $\mathbf{x}(t) \in \mathscr{R}^I, I \in \mathbb{N}$ is the input observation and θ contains parameters about setting learning rates (See Section3.5.4).

The rest of the chapter is organized as follows. Section 3.2 discusses different parts of the IncSFA algorithm in detail. Section 3.3 presents the pseudo-code of the

[1]In general $K < I$ and $J < K$.

algorithm along with other implementation details. Section 3.4 presents experimental results. Section 3.5 discusses the convergence aspects, learning-rates scheduling and the biological-link of the algorithm. Section 3.6 concludes.

3.2 Method Description

IncSFA algorithm has two learning update rules: Candid-Covariance Free Incremental Principal Component Analysis (CCIPCA) for normalizing the input and Minor Component Analysis (MCA) for extracting the slow features. The following sections discuss these components in turn.

3.2.1 Principal Components for Whitening

Given zero-mean data $\mathbf{u} = \mathbf{x} - \mathbb{E}[\mathbf{x}]$, a PC is a normed eigenvector $\mathbf{v}_i^*$ of the data covariance matrix $\mathbb{E}[\mathbf{u}\mathbf{u}^T]$. An eigenvalue λ_i^* is the variance of the samples along $\mathbf{v}_i^*$. By definition, an eigenvector and eigenvalue satisfy

$$\mathbb{E}[\mathbf{u}\mathbf{u}^T]\mathbf{v}_i^* = \lambda_i^*\mathbf{v}_i^*. \tag{3.2}$$

The set of eigenvectors are orthonormal and ordered such that $\lambda_1^* \geq \lambda_2^* \geq ... \geq \lambda_K^*$. The whitening matrix is generated by multiplying the matrix of principal component length-one eigenvectors $\mathbf{V}^*$ by the diagonal matrix $\mathbf{D}^*$, where component $\hat{d}_{i,i} = \frac{1}{\sqrt{\lambda_i^*}}$. After whitening via $\mathbf{z}(t) = \mathbf{D}^*\mathbf{V}^{*T}\mathbf{u}(t)$, the data will be normalized in scale and decorrelated so that the covariance matrix of $\mathbf{z}$ will be the identity matrix: $\mathbb{E}[\mathbf{z}\mathbf{z}^T] = I$.

The procedure to generate a whitening matrix is outlined in Algorithm 1. Via CCIPCA, the magnitudes of the eigenvector estimates are the eigenvalue estimates. The method used to generate estimates of $(\lambda_i^*, \mathbf{v}_i^*)$ is outlined next.

3.2.2 CCIPCA Updating

CCIPCA updates estimates of eigenvalues and eigenvectors from each sample in order to implement incremental whitening. For inputs $\mathbf{u}_i$, the first PC is the expectation of the normalized response-weighted inputs. Eq 3.2 can be rewritten as

$$\lambda_i^*\ \mathbf{v}_i^* = \mathbb{E}\left[(\mathbf{u}_i \cdot \mathbf{v}_i^*)\ \mathbf{u}_i\right]. \tag{3.3}$$

Algorithm 1: ConstructWhiteningMatrix($\mathbf{V}$)

```
1 $\hat{\mathbf{V}} \leftarrow \left( \frac{\mathbf{v}_1}{||\mathbf{v}_1||}, ..., \frac{\mathbf{v}_K}{||\mathbf{v}_K||} \right)$ //$I \times K$ matrix
2 $\mathbf{D} \leftarrow \mathbf{0}$ //$K \times K$ matrix
3 for $i \leftarrow 1$ to $K$ do
4     $D_{i,i} = 1/\sqrt{||\mathbf{v}_i||}$
5 end
6 $\mathbf{S} \leftarrow \hat{\mathbf{V}}\mathbf{D}$ //$I \times K$ matrix
7 return $\mathbf{S}$
```

Algorithm 2: CCIPCA-Update($\mathbf{V}, K, \mathbf{u}, \eta$)

```
  //Candid Covariance-Free Incremental PCA
1 $\mathbf{u}_1 \leftarrow \mathbf{u}$
2 for $i \leftarrow 1$ to $K$ do
      //Principal component update
3     $\mathbf{v}_i \leftarrow (1-\eta)\,\mathbf{v}_i + \eta\,\left[\frac{\mathbf{u}_i \cdot \mathbf{v}_i}{||\mathbf{v}_i||}\,\mathbf{u}_i\right]$
      //Residual
4     $\mathbf{u}_{i+1} = \mathbf{u}_i - \left(\mathbf{u}_i^T \frac{\mathbf{v}_i}{||\mathbf{v}_i||}\right) \frac{\mathbf{v}_i}{||\mathbf{v}_i||}$
5 end
6 return $\mathbf{V}$
```

The corresponding incremental updating equation, where $\lambda_i^* \, \mathbf{v}_i^*$ is estimated by $\mathbf{v}_i(t)$, is

$$\mathbf{v}_i(t) = (1-\eta^{PCA})\,\mathbf{v}_i(t-1) + \eta^{PCA}\,\left[\frac{\mathbf{u}_i(t)\cdot\mathbf{v}_i(t-1)}{||\mathbf{v}_i(t-1)||}\,\mathbf{u}_i(t)\right]. \quad (3.4)$$

where $0 < \eta^{PCA} < 1$ is the learning rate. In other words, both the eigenvector and eigenvalue of the first PC of $\mathbf{u}_i$ can be found through the sample mean-type updating in Eq. 3.3. The estimate of the eigenvalue is given by $\lambda_i = ||\mathbf{v}_i(t)||$. Using both a learning rate η^{PCA} and retention rate $(1 - \eta^{PCA})$ automatically makes this algorithm invariant to the magnitude of the input vectors. Computation of lower-order principal components, is described next.

3.2.3 Lower-Order Principal Components

Any component $i > 1$ not only must satisfy Eq. 3.2 but must also be orthogonal to the higher-order components. The *residual method* [Kreyszig, 1988; Sanger, 1989] generates observations in a complementary space so that lower-order eigenvectors can be found by the update rule of Eq. 3.4.

Denote $\mathbf{u}_i(t)$ as the observation for component i. When $i = 1$, $\mathbf{u}_1(t) = \mathbf{u}(t)$. When $i > 1$, $\mathbf{u}_i$ is a residual vector, which has the "energy" of $\mathbf{u}(t)$ from the higher-order components removed. Solving for the first PC in this residual space solves for the i-th component overall. To create a residual vector, $\mathbf{u}_i$ is projected onto $\mathbf{v}_i$ to get the energy of $\mathbf{u}_i$ that $\mathbf{v}_i$ is responsible for. Then, the energy-weighted $\mathbf{v}_i$ is subtracted from $\mathbf{u}_i$ to obtain $\mathbf{u}_{i+1}$:

$$\mathbf{u}_{i+1}(t) = \mathbf{u}_i(t) - \left(\mathbf{u}_i^T(t)\frac{\mathbf{v}_i(t)}{\|\mathbf{v}_i(t)\|}\right)\frac{\mathbf{v}_i(t)}{\|\mathbf{v}_i(t)\|}. \tag{3.5}$$

Together, Eqs. 3.4 and 3.5 constitute the CCIPCA technique described in Algorithm 2. We now examine how to extract the slow features from the whitened data.

3.2.4 MCA Updating

After using CCIPCA components to generate an approximately whitened signal $\mathbf{z}$, the derivative is approximated by $\dot{\mathbf{z}}(t) = \mathbf{z}(t) - \mathbf{z}(t-1)$. In this derivative space, the minor components on $\dot{\mathbf{z}}$ are the slow features.

To find the minor components, Peng's MCA [Peng et al., 2007] is used. The updates for MCA are given by

$$\begin{aligned}\mathbf{w}_i(t) = 1.5\mathbf{w}_i(t-1) - \eta^{MCA}\,\mathbf{C}_i\,\mathbf{w}_i(t-1)\\ -\eta^{MCA}\left[\mathbf{w}_i^T(t-1)\mathbf{w}_i(t-1)\right]\mathbf{w}_i(t-1),\end{aligned} \tag{3.6}$$

where for the first minor component, $\mathbf{C}_1 = \dot{\mathbf{z}}(t)\dot{\mathbf{z}}^T(t)$.

For other minor components, the *sequential addition* technique [Chen et al., 2001] shifts each observation into a space where the minor component of the current space will be the first PC, and all other PCs are reduced in order by one. Sequential addition allows IncSFA to extract more than one slow feature in parallel. Sequential addition updates the matrix $\mathbf{C}_i,\ \forall i > 1$ as follows:

$$\mathbf{C}_i(t) = \mathbf{C}_{i-1}(t) + \gamma(t)\,\left(\mathbf{w}_{i-1}(t)\mathbf{w}_{i-1}^T(t)\right) / \left(\mathbf{w}_{i-1}^T(t)\mathbf{w}_{i-1}(t)\right) \tag{3.7}$$

Algorithm 3: CIMCA-Update($W, J, \dot{\mathbf{z}}, \gamma, \eta$)

```
  //Covariance-Free Incremental MCA
1 $\mathbf{l}_1 \leftarrow 0$
2 for $i \leftarrow 1$ to $J$ do
      //Minor component update
3     $\mathbf{w}_i \leftarrow (1-\eta)\mathbf{w}_i - \eta\left[(\dot{\mathbf{z}} \cdot \mathbf{w}_i)\,\dot{\mathbf{z}} + \mathbf{l}_i\right]$.
      //Normalize
4     $\mathbf{w}_i \leftarrow \mathbf{w}_i / \|\mathbf{w}_i\|$.
      //Lateral competition from "lower" components
5     $\mathbf{l}_{i+1} \leftarrow \gamma\ \sum_j^i (\mathbf{w}_j \cdot \mathbf{w}_i)\mathbf{w}_j$
6 end
7 return W
```

Note Eq. 3.7 introduces parameter γ, which must be larger than the largest eigenvalue of $\mathbb{E}[\dot{\mathbf{z}}(t)\dot{\mathbf{z}}^T(t)]$. To automatically set γ, the greatest eigenvalue of the derivative signal is computed through another CCIPCA rule to update only the first PC. Then, $\gamma = \lambda_1(t) + \epsilon$ for small ϵ. Peng's MCA technique computes costly single data-point covariance-matrix of the whitened input: $\mathbf{C}_1 = \dot{\mathbf{z}}(t)\dot{\mathbf{z}}^T(t)$. The next section presents a modification to the MCA algorithm to make it covariance free.

3.2.5 Covariance-Free MCA

We can avoid the potentially costly outer products via the same trick that made CCIPCA covariance-free: $(\dot{\mathbf{z}}\dot{\mathbf{z}}^T)\ \mathbf{w}_i = (\dot{\mathbf{z}} \cdot \mathbf{w}_i)\dot{\mathbf{z}}$. Considering only the first slow feature for now, Eq. 3.6 can be re-written as:

$$\begin{aligned} \mathbf{w}_1 &\leftarrow 1.5\mathbf{w}_1 - \eta^{MCA}\ \dot{\mathbf{z}}\ [\dot{\mathbf{z}}^T\ \mathbf{w}_i] - \eta^{MCA}\ [\mathbf{w}_i^T \mathbf{w}_i]\ \mathbf{w}_i, \\ &= \left(1.5 - \eta^{MCA}\ \|\mathbf{w}_1\|^2\right)\ \mathbf{w}_1 - \eta^{MCA}\ (\dot{\mathbf{z}} \cdot \mathbf{w}_1)\ \dot{\mathbf{z}}, \end{aligned} \tag{3.8}$$

as shown in Section 3.5.6.

When dealing with non-stationary input, due to the simultaneously learning CCIPCA components, it is acceptable[2] to normalize the magnitude of the slow feature vectors: $\mathbf{w}_i \leftarrow \mathbf{w}_i / \|\mathbf{w}_i\|$. Normalization at least ensures non-divergence (see Section 3.5.1). If we normalize, Eq. 3.8 can be rewritten in an even simpler form

$$\mathbf{w}_1 \leftarrow (1-\eta^{MCA})\mathbf{w}_1 - \eta^{MCA}(\dot{\mathbf{z}} \cdot \mathbf{w}_1)\ \dot{\mathbf{z}}, \tag{3.9}$$

$$\mathbf{w}_1 \leftarrow \mathbf{w}_1 / \|\mathbf{w}_1\|. \tag{3.10}$$

[2]Peng: personal communication.

Now, for all other slow features $i > 1$, the update can be written so that the sequential addition becomes a Gram-Schmidt procedure.

$$\mathbf{w}_i \leftarrow (1-\eta^{MCA})\mathbf{w}_i - \eta^{MCA}\left((\dot{\mathbf{z}}\cdot\mathbf{w}_i)\,\dot{\mathbf{z}} + \gamma \sum_{j}^{i-1} (\mathbf{w}_j\cdot\mathbf{w}_i)\mathbf{w}_j \right). \tag{3.11}$$

The covariance-free MCA is outlined in algorithm 3. The above discussed learning components together constitute the IncSFA algorithm. The resultant slow feature vectors at any time t is equal to the product of the whitening vectors $\mathbf{S}(t)$ and the minor components $\mathbf{W}(t)$: $\phi(t) = \mathbf{S}(t)\mathbf{W}(t)$.

3.3 Pseudocode

Algorithm 4 summarizes Incremental SFA. A *Python*-based implementation of the algorithm can be found at the URL: www.idsia.ch/~kompella/codes/incsfa.html. *Matlab* code is available at www.idsia.ch/~luciw/incsfa.html.

3.4 Experimental Results

In this section, I present results of several experiments conducted on synthetic input signals and real world images to illuminate the performance of the IncSFA algorithm. These results show that the features extracted by IncSFA closely match the slow features extracted by the batch SFA (BSFA).

3.4.1 Proof of Concept

As a basic proof of concept, IncSFA is applied to the introductory problem from the original SFA paper [Wiskott and Sejnowski, 2002] to show that IncSFA can derive the same set of features as BSFA. The input signal is

$$x_1(t) = \sin(t) + \cos(11\,t)^2, \tag{3.12}$$
$$x_2(t) = \cos(11\,t),\ \ t \in [0, 2\pi]. \tag{3.13}$$

Both input components vary quickly over time (see Figure 3.2(a)). The slowest feature hidden in the signal is $y_1(t) = x_1(t) - x_2(t)^2 = \sin(t)$. The second slowest feature is $y_2(t) = x_2(t)^2$.

Algorithm 4: IncSFA(J, K, θ)

```
   //Incremental update of $J$ slow features from samples
     $\mathbf{x} \in \mathscr{R}^I$
   //$\mathbf{V}$ :  $K$ columns:  CCIPCA weight vectors
   //$\mathbf{W}$ :  $J$ columns:  CIMCA weight vectors
   //$\phi$ :  $J$ columns:  SFs
   //$\mathbf{v}^\gamma$ :  First PC in $\dot{\mathbf{z}}$-space
   //$\bar{\mathbf{x}}$ :  Mean of $\mathbf{x}$
 1 $\{\mathbf{V}, \mathbf{W}, \phi, \mathbf{v}^\gamma, \bar{\mathbf{x}}\} \leftarrow$ Initialize ()
 2 for $t \leftarrow 1$ to $\infty$ do
 3     $\mathbf{x} \leftarrow$ Sense($worldstate$)
 4     $\{\eta_t^{PCA}, \eta_t^{MCA}\} \leftarrow$ LrnRateSchedule ($\theta, t$)
 5     $\bar{\mathbf{x}} \leftarrow (1 - \eta_t^{PCA})\, \bar{\mathbf{x}} + \eta_t^{PCA}\, \mathbf{x}$ //Update mean
 6     $\mathbf{u} \leftarrow (\mathbf{x} - \bar{\mathbf{x}})$ //Centering
       //Candid Covariance-Free Incremental PCA
 7     $\mathbf{V} \leftarrow$ CCIPCA-Update ($\mathbf{V}, K, \mathbf{u}, \eta_t^{PCA}$)
 8     $\mathbf{S} \leftarrow$ ConstructWhiteningMatrix ($\mathbf{V}$)
 9     If $t > 1$ then $(\mathbf{z}_{prev} \leftarrow \mathbf{z}_{curr})$ //Store prev.
       //Whitening and dim.  reduction
10     $\mathbf{z}_{curr} \leftarrow \mathbf{S}^T \mathbf{u}$
11     if $t > 1$ then
12         $\dot{\mathbf{z}} \leftarrow \left(\mathbf{z}_{curr} - \mathbf{z}_{prev}\right)$ //Approx.  derivative
           //For seq.  addition ($\gamma$)
13         $\mathbf{v}^\gamma \leftarrow$ CCIPCA-Update ($\mathbf{v}^\gamma, 1, \dot{\mathbf{z}}, \eta_t^{PCA}$)
14         $\gamma \leftarrow \mathbf{v}^\gamma / \|\mathbf{v}^\gamma\|$
           //Covariance-free Incremental MCA
15         $\mathbf{W} \leftarrow$ CIMCA-Update ($\mathbf{W}, J, \dot{\mathbf{z}}, \gamma, \eta_t^{MCA}$)
16     end
17     $\mathbf{y} \leftarrow \mathbf{z}_{curr}^T \mathbf{W}$ //Slow feature output
18     $\phi \leftarrow \mathbf{SW}$ //Slow features
19 end
```

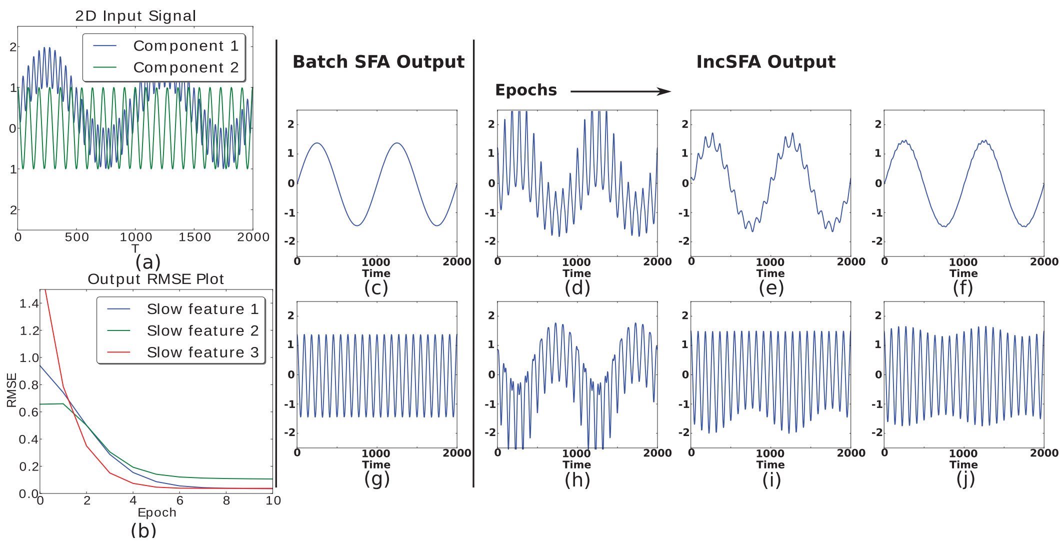


Figure 3.2. Extracting slow features incrementally from a simple non-linear input signal. (a) Input Signal (b) Output root mean square error (RMSE) plot showing convergence of the first three IncSFA features to the corresponding BSFA features. (c) BSFA output of the first slow feature (d)-(f) IncSFA output of feature 1 at t = 2, 5, 10 epochs. (g) BSFA output of the second slow feature (h)-(j) IncSFA output of feature 2 at t = 2, 5, 10 epochs. IncSFA performs like BSFA, as expected.

Each epoch contains a total of $2,000$ discrete datapoints, over the entire range of t, which are used for learning. A quadratic input expansion is done. A learning rate of $\eta^{MCA} = 0.08$ is used.

Both BSFA and IncSFA extract the slow features. Figure 3.2(b) shows the Root Mean Square Error (**RMSE**) of three IncSFA feature outputs compared to the corresponding BSFA outputs over multiple epochs of training, showing that the IncSFA features converge to the correct ones. Figures 3.2(c) and (g) show feature outputs of BSFA, and (to the right) IncSFA outputs at 2, 5, and 10 epochs. Figures 3.2(g)-(j) show this comparison for the second feature. This basic result shows that it is indeed possible to extract multiple slow features in an online way without storing covariance matrices.

3.4.2 Feature Adaptation to a Changing Environment

The purpose of this experiment is to illustrate how IncSFA's features *adapt* to an unpredicted sudden shift in the input process. The input used is the same signal as in Experiment #1, but broken into two partitions. At epoch 60, the two input lines x_1 and x_2 are switched such that the x_1 signal suddenly carries what x_2 used to, and vice versa. IncSFA can first learn the slow features in the first partition, then is able

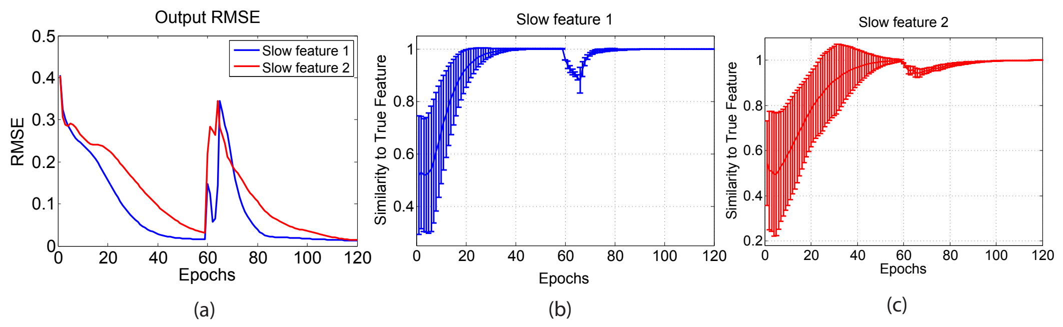


Figure 3.3. (a) RMSE of IncSFA's first two output functions with respect to the true functions for original signal (epochs 1-59), and switched signal (epochs 60-120). (b) Normalized similarity (direction cosine) of the first slow feature to the true first slow feature of the current process, over 25 independent runs. (c) Normalized similarity of the second incremental slow feature.

to adapt to learn the slow features in the second partition.

Here, the signal is sampled 500 times per epoch. The CCIPCA learning rate parameters, also used to set the learning rate of the input average $\bar{\mathbf{x}}$, were set to $t_1 = 20, t_2 = 200, c = 4, r = 5000$ (See Section3.5.4). The MCA learning rate is a constant $\eta_{mca} = 0.01$.

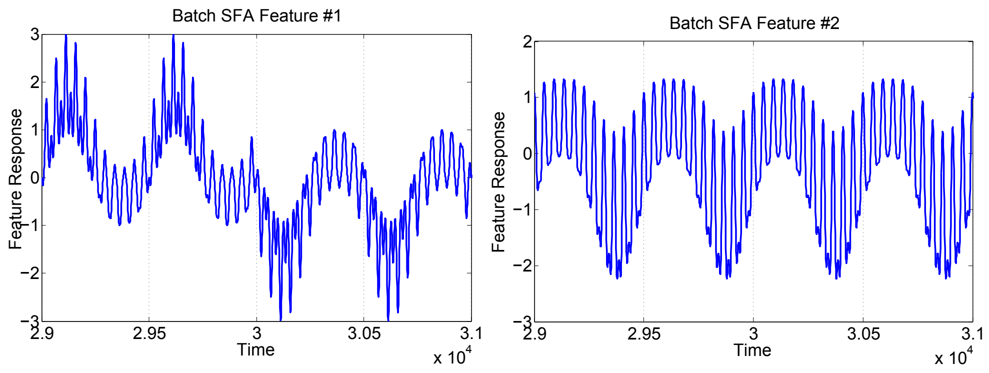


Figure 3.4. Outputs of first two slow features, from epoch 59 through 61, extracted by BSFA over the input sequence.

Results of IncSFA are shown in Figure 3.3, demonstrating successful adaptation. To measure convergence accuracy, the direction cosine [Chatterjee et al., 2000] between the estimated feature $\mathbf{w}(t)$ and true (unit length) feature $\mathbf{w}^*$ is used,

$$DirectionCosine(t) = \frac{|\mathbf{w}^T(t) \cdot \mathbf{w}^*|}{\|\mathbf{w}^T(t)\| \cdot \|\mathbf{w}^*\|}. \tag{3.14}$$

The direction cosine equals one when the directions align (the feature is correct) and zero when they are orthogonal.

BSFA results are shown in Figure 3.4. The first batch slow feature somewhat catches the meta-dynamics and could actually be used to roughly sense the signal switch. However, the dynamics within each partition are not extracted. The BSFA result might be improved by generating embedding-vector time series [Wiskott, 2003] and increasing the non-linear expansion. But due to long duration of the signals and the unpredicted nature of the signal switch, time-embedding with a fixed delay might not be able to recover the dynamics appreciably. This experiment demonstrated that IncSFA, unlike BSFA, is adaptive to the changing input statistics.

3.4.3 Recovery from Outliers

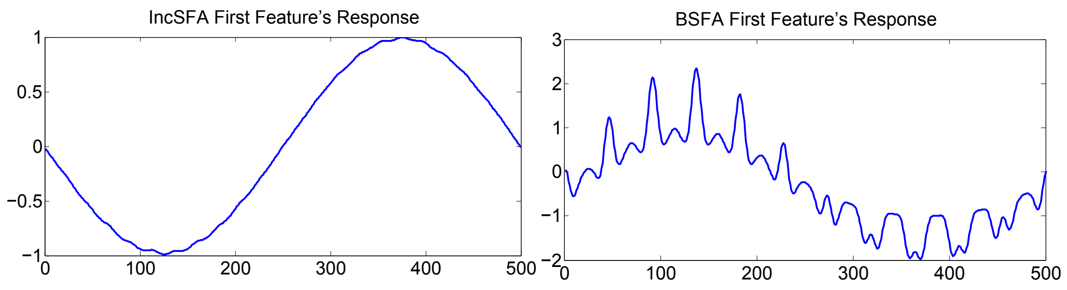


Figure 3.5. First output signals of IncSFA and BSFA on the simple signal with a single outlier.

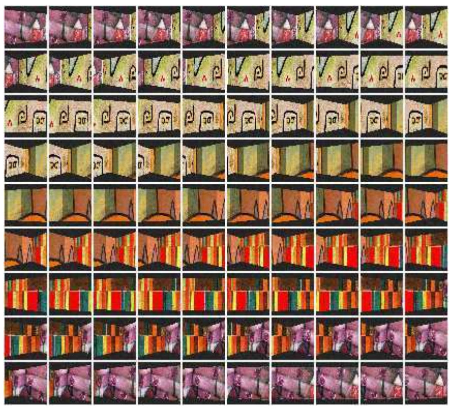
Next, the effect of a single extreme outlier on both BSFA and IncSFA is shown. Again, the learning rate setup and basic signal from the previous experiments are used, with 500 samples per epoch, over 150 epochs. A single outlier point is inserted at time 100, only in the first epoch: $x_1(100) = x_2(100) = 2000$.

Figure 3.5 shows the first output signal of BSFA and IncSFA. The one outlier point at time 100 (out of 75,000) is enough to corrupt the first feature of BSFA, whereas IncSFA recovers. It is possible to include clipping [Franzius et al., 2007] in BSFA, so that the effect of the outliers that have different variance statistics compared to the signal can be overcome.

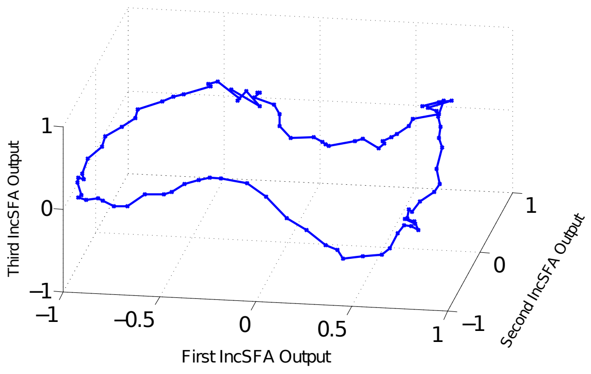
Outliers that are generated from another signal source lying within the variance of the main signal can affect the BSFA output in a different way. I refer to a real-world experiment [Kompella et al., 2011b], using AutoIncSFA (where the input to

IncSFA is the output at a bottleneck layer of an autoencoder neural net — for image compression) on an image sequence, in which a person moves back and forth in front of a stable camera. At only one point in the training sequence, a door in the background is opened. The BSFA hierarchical network's first slow feature became sensitive to this event. Yet, the AutoIncSFA network's first slow feature encodes the relative distance of the moving interactor.

3.4.4 High-Dimensional Video with Linear IncSFA



(a)



(b)

Figure 3.6. (a) Stream of 90 $41 \times 41 \times 3$ images as the agent completes one rotation (360 degrees). There are 10 subsequent images per row, starting from the top-left. The image after that at the far right of a row starts at the far left of the lower row. (b) All 90 images (noise-free) projected onto the first three features learned by IncSFA. We can easily see the 1D and circular nature of the agent's movement within the environment. This embedding can be used as a compact encoding of the agent's state.

IncSFA makes it possible to use SFA in high-dimensional video processing applications without using deep receptive-field based networks. CCIPCA provides an intermediate dimensionality reduction, which, when low enough compared to the input dimension, can greatly reduce the computational and space complexities as well as the search space for the slow features via MCA.

As a first experiment to show this, SFs are extracted from a rotating vision-based agent in a square room. The room has four complex-textured walls. See Figure 3.6(a). Each image is dimension $41 \times 41 \times 3$.

In each episode[3], starting from a different orientation, the agent rotates slowly (4 degree shifts from one image to the next) by 360 degrees, each episode. At any time, a slight amount of Gaussian noise is added to the image ($\sigma = 8$).

Each $5,043$ dimensional image is fed into a linear IncSFA directly. Only the 40 most significant principal components are computed by CCIPCA, using learning rate parameters $t_1 = 20$, $t_2 = 200$, $c = 4$, $r = 5000$ (See Section 3.5.4). Computation of the covariance matrix and its full eigendecomposition (over 5000 eigenvectors and eigenvalues) is therefore avoided. On the 40 dimensional whitened difference signal, only the first 5 slow features are computed via CIMCA.

Computation of 500 epochs through the data took approximately 15 minutes using Matlab on a machine with an Intel i3 CPU and 4 GB RAM. This corresponds to a framerate of about 50fps.

The result of projecting the (noise-free) data onto the first three slow features is shown in Figure 3.6(b). A single linear IncSFA has incrementally compressed this high-dimensional noisy sequence to a nearly unambiguous compact form, learning to ignore the details at the pixel level and attend to the true cyclical nature underlying the image sequence. A few subsequences have somewhat ambiguous encodings, probably because certain images associated with slightly different angles are similar. This experiment demonstrated that the IncSFA algorithm scales to high-dimensional image inputs.

3.4.5 iCub Experiment

Here, I conduct an experiment with real high-dimensional vision sequences generated from the camera-eyes of an exploring iCub [Metta et al., 2008] humanoid robot. Two plastic cups are placed in the iCub robot's field of view. The robot performs motor babbling in one joint of its right arm, using a movement paradigm used by Franzius et al. [2007]. During the course of babbling, it happens to topple both cups in one of two possible orders. The episode ends a few frames after it has knocked both down. A new episode begins with the cups upright again and the arm in the beginning position. A total of 50 separate episodes were recorded and the images used as training data.

IncSFA updates from each 80×60 (grayscale) image. Only the 20 most significant principal components are computed by CCIPCA, using learning rate param-

[3]IncSFA can be readily extended to episodic tasks, with a minor modification: The derivative signal, which is computed as a difference over a single time step, is simply not computed for the starting sample of each episode. The first data point in each episode is used for updating the PCs, but not the slow feature vectors.

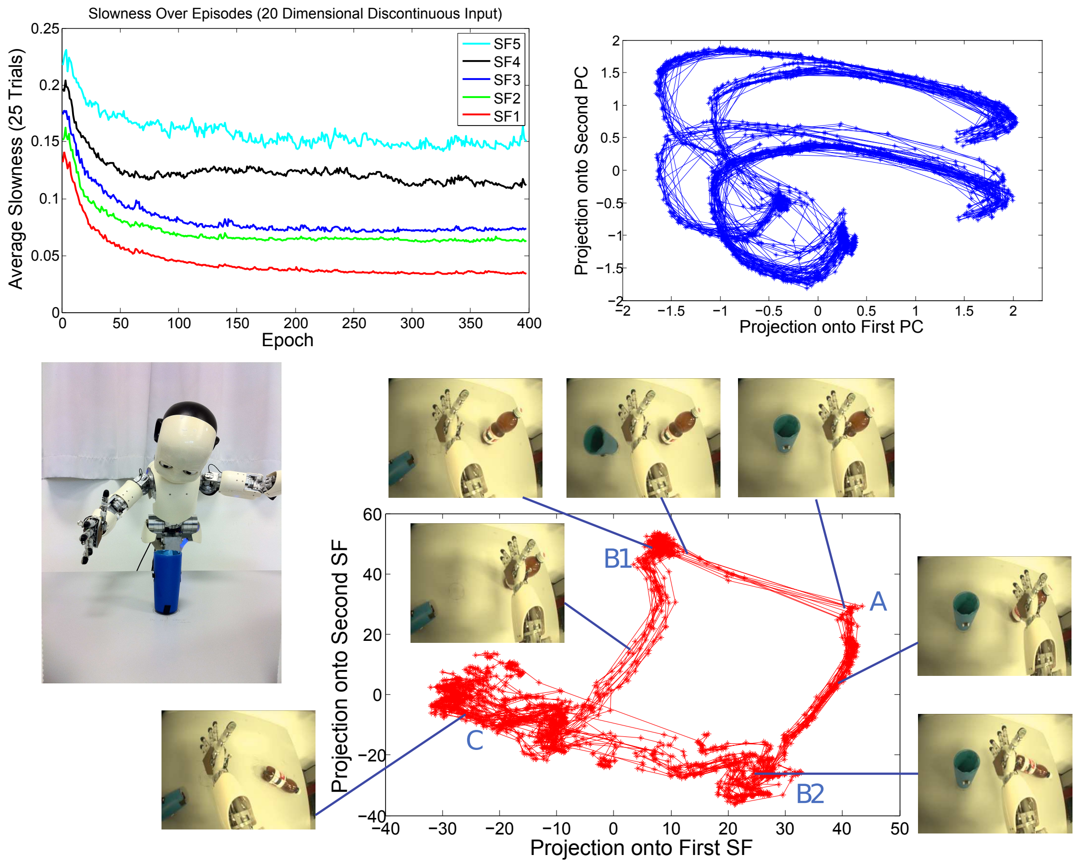


Figure 3.7. Experimental result of IncSFA on episodes where the iCub knocks down two cups via motor babbling on one joint. Upper left: The average slowness of the five features at each episode. Upper right: after training, several episodes (each episode is an image sequence where the cups are eventually both knocked down) are embedded in the space spanned by the first two PCs. Lower right: the same episodes are embedded in the space spanned by the first two slow features. I show some example images and where they lie in the embedding. The cluster in the upper right (A) represents when both cups are upright. When the robot knocks down the blue cup first, it moves to the cluster in the upper left (B1). If it instead knocks down the brown cup, it moves to the lower right cluster (B2). Once it knocks down both cups, it moves to the lower left area (C).

eters $t_1 = 20$, $t_2 = 200$, $c = 2$, $r = 10000$ (See Section3.5.4). Only the first 5 slow features are computed via CIMCA with learning rate 0.001. The MCA vectors are normalized after each update during the first 10 episodes, but not thereafter (for faster convergence). The algorithm runs for 400 randomly-selected (of the 50 possible) episodes. The experiment is replicated 25 times.

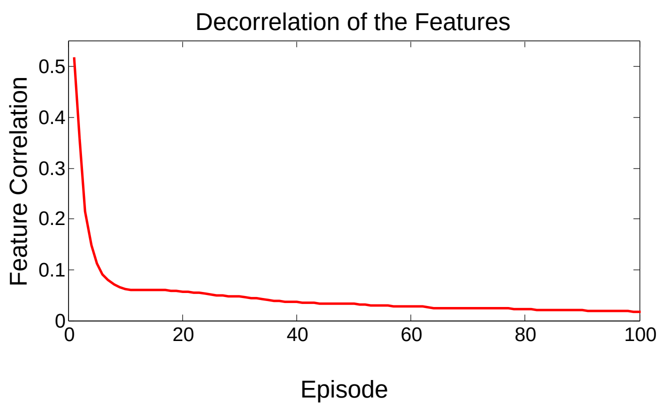


Figure 3.8. Average slow feature similarity over episodes in the iCub experiment.

Results are shown in Figure 3.7. The slowness of the feature outputs is measured on three "testing" episodes after each episode of training. The upper left plot shows that all five features get slower as they are trained over the 400 episodes. Figure 3.8 shows the average mutual direction cosine between non-identical pairs of slow features, and we can see the features quickly become nearly decorrelated.

After training completes, the images are embedded in a lower dimension using the learned features. The embedding of trajectories of 20 different episodes are shown with respect to the first two PCs as well as the first two slow features. Since the cups being toppled or upright are the slow events in the scene, IncSFA's encoding is keyed on the object's state (toppled or upright). PCA does not find such an encoding, being much more sensitive to the arm. Such clear object-specific low-dimensional encoding, invariant to the robot's arm position, is useful, greatly facilitating training of a subsequent regressor or reinforcement learner.[4]

3.4.6 Hierarchical IncSFA

Deep networks composed of multiple stacked SFA nodes, each sensitive to only a small part of the input (i.e., receptive fields), are typically used for SFA processing of high-dimensional images. The slow features are linear combinations of the input space components. Since there is no guarantee the useful information is linear in the original sensory space, an expanded space is often used. For example, a quadratic expansion adds all combinations of input components, or a cubic expansion adds all triples. But the degree of expansion required to construct a space where the "interesting information" will be some linear combination may increase the dimension

[4]A video of the experimental result can be found at `http://www.idsia.ch/~luciw/IncSFAArm/IncSFAArm.html`.

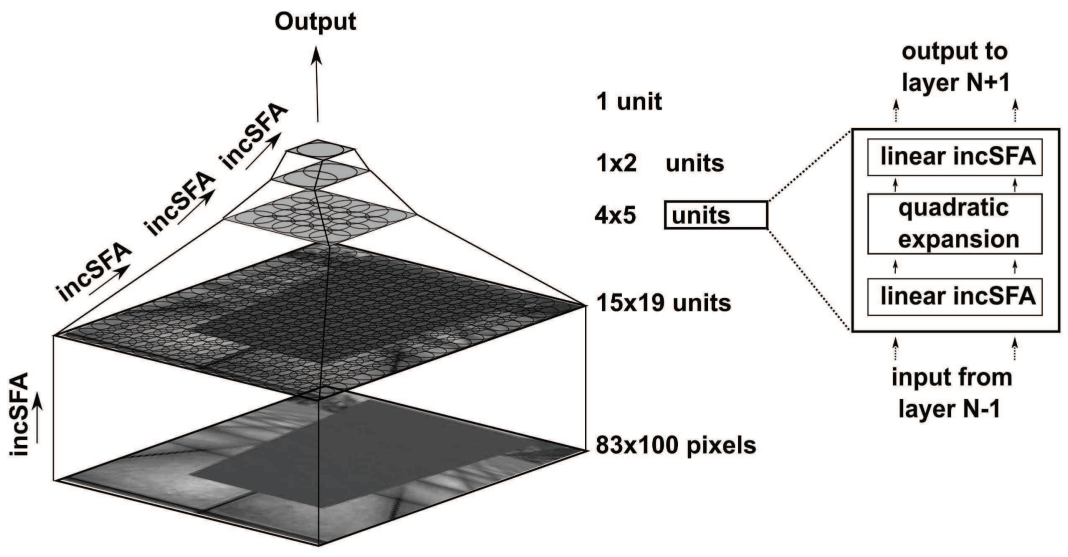


Figure 3.9. Example Hierarchical-IncSFA Architecture. This also shows the structure of an IncSFA node, which contains a linear IncSFA unit followed by nonlinear expansion followed by another linear IncSFA unit.

intractably. To deal with these cases, one can use multilayer, receptive-field based networks [Wiskott and Sejnowski, 2002; Franzius et al., 2007], which reduce the complexity for any SFA module by partitioning spatially on each layer into receptive fields while having a low-order (e.g., quadratic) expansion within each receptive field. A succession of low-order expansions over multiple layers lead to an overall expansion which is high-order.

The utility of IncSFA is tested in this network context. Hierarchical networks introduce new parameters (receptive field size, number of layers, etc.) that can be difficult to tune. There is another applicable tactic, that is, to apply IncSFA monolithically to the (possibly even expanded) high-dimensional input, extracting $K << I$ principle components with CCIPCA and J slow features. But IncSFA can also be used within a deep network architecture.

Figure 3.9 shows an example deep network, motivated by the human visual system and based on the one specified by Franzius et al. [2007]. The network is made up of a converging hierarchy of layers of IncSFA nodes, with overlapping rectangular receptive fields. Each IncSFA node finds the slowest output features from its input within the subspace of quadratically expanded inputs.

Input images come from a high-dimensional video stream generated by the iCub simulator [V. Tikhanoff and Nori, 2008], an OpenGL-based software simulator specif-

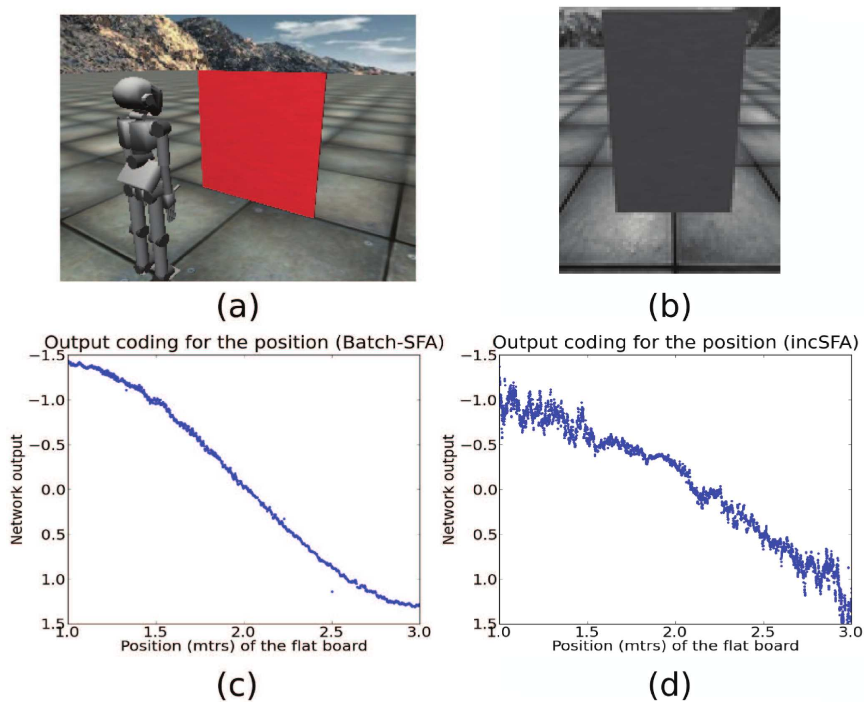


Figure 3.10. (a) Experimental Setup: iCub Simulator (b) Sample image from the input dataset (c) BSFA output (d) IncSFA output ($\eta_{mca} = 0.005$)

ically built for the iCub robot. This experiment mimics the robot observing a moving interactor agent, which in the simulation takes the form of a rectangular flat board moving back and forth in depth over the range $\{1, 3\}$ (meters) in front of the robot. This movement paradigm was developed by Franzius et al. [2007]. Figure 3.10(a) shows the experimental setup in the iCub simulator. Figure 3.10(b) shows a sample image from the dataset. $20,000$ monocular images are captured from the robot's left eye and downsampled to 83×100 pixels (input dimension of $8,300$).

A three-layer IncSFA network is used to encode the images. Each SFA node operates on a spatial receptive field of the layer below. The first layer uses 15×19 nodes, each with 10×10 image patch receptive field and a 5 pixel overlap. Each node on this layer develops 10 slow features. The second layer uses 4×5 nodes, each having a 5×5 receptive field, and developing 5 slow features. The third layer uses two nodes, one sensitive to the top half, the other sensitive to the bottom half (5 slow features). The forth layer uses a single node and a single slow feature. The network is trained layer-wise from bottom to top, with the lower layers frozen once a new layer begins its training. The CCIPCA output of all nodes is clipped to $[-5, 5]$, to avoid any outliers that may arise due to close-to-zero eigenvalues in some of the receptive fields that contain unchanging stimuli. Each IncSFA node is trained individually, that is, there is no weight sharing among nodes.

For comparison, a BSFA hierarchical network was also trained on this data. Figures 3.10 show BSFA and IncSFA outputs. The expected output takes the form of a sinusoid extending over the range of board positions. IncSFA gives a slightly noisy output, probably due to the constant dimensionality reduction value for all units in each layer of the network, selected to maintain a consistent input structure for the subsequent layer; hence some units with eigenvectors corresponding to very small eigenvalues emerge in the first stage, with receptive fields observing comparatively few input changes, thus slightly corrupting the whitening result, and adding small fluctuations to the overall result.

Finally, how well the IncSFA feature codes for distance is evaluated. A supervised quadratic regressor is trained with ground truth labels on 20% of the dataset, and tested on the other 80%, to measure the quality of features for some classifier or reinforcement learner using them. The **RMSE** was found to be equal to 0.043 meters. This experiment demonstrated the application of IncSFA algorithm in a deep hierarchical network to extract high-degrees of non-linearities in the inputs.

3.5 Supplementary Topics

In this section, I will discuss the convergence aspects of the IncSFA algorithm along with a guide to setting the learning rates and its link to the biological systems.

3.5.1 On Non-Divergence and Convergence

For CCIPCA; if the standard conditions on learning rate hold [Papoulis et al., 1965] (including convergence at zero), the first stage components will converge to the true PCs, leading to a "nearly-correct" whitening matrix in reasonable time. Thus if the input $\mathbf{x}$ is stationary, the slow feature estimates are likely to grow close to the true slow features in a reasonable amount of updates.

In open-ended learning, convergence is usually not desired. Yet by using a learning rate that is always nonzero, the stability of the algorithm is reduced. This corresponds to the well-known stability-plasticity dilemma [Grossberg, 1980].

For stability and convergence of incremental MCA, the following constraints must be satisfied [Peng et al., 2007],

$$\eta^{MCA}\lambda_1^* < 0.5, \quad ||\mathbf{w}(0)||^2 \leq \frac{1}{2\eta^{MCA}}, \quad \mathbf{w}^T(0)\mathbf{w}^* \neq 0 \tag{3.15}$$

where $\mathbf{w}(0)$ is the initial feature estimate, $\mathbf{w}^*$ the true eigenvector associated with the smallest eigenvalue, and λ_1^* the largest eigenvalue. In other words, the learning rate must not be too large, and the initial estimate must not be orthogonal to the true component.

It is clear that if whitened signal $\mathbf{z}$ is drawn from a stationary distribution, the MCA convergence proof [Peng et al., 2007] applies. But typically the whitening matrix is being learned simultaneously. In this early stage, while the CCIPCA vectors are learning, care must be taken to ensure that the slow feature estimates will not diverge.

Peng showed that for any initial vector $\mathbf{w}(0)$ within the set $\mathcal{S}$ given by

$$\mathcal{S} = \left\{ \mathbf{w}(t) \middle| \mathbf{w}(t) \in \mathcal{R}^K \text{ and } ||\mathbf{w}(t)||^2 \leq \frac{1}{2\eta^{MCA}} \right\}, \tag{3.16}$$

$\mathbf{w}(t)$ $(\forall t \geq 0)$ will remain in $\mathcal{S}$ throughout the dynamics of the MCA updating. Thus, $||\mathbf{w}||$ must be prevented from getting too large until the whitening matrix is close to accurate. With respect to lower-order slow features, there is additional dependence on the sequential addition technique, parameterized by $\gamma(t) = \lambda_1(t) + \epsilon$. This $\gamma(t)$ also needs time to estimate a close value to the first eigenvalue λ_1. Before these estimates become reasonably accurate, the input can knock the vector out of $\mathcal{S}$.

In IncSFA, $\mathbf{w}$ is normalized after each update. If $||\mathbf{w}(0)|| = 1$ then any learning rate $\eta_{mca} \leq 0.5$ ensures non-divergence.

Even if $\mathbf{w}$ remains in $\mathcal{S}$, the additional constraint $\mathbf{w}^T(0)\mathbf{w}^* \neq 0$ is needed for convergence. But this is an easy condition to meet, as it is unlikely that any $\mathbf{w}(t)$ will be exactly orthogonal to the true feature. In practice, it may be advisable to add a small amount of noise to the MCA update. But I did not find this to be necessary in experiments.

3.5.2 Intermediate dimensionality reduction

Since often only a relatively small number of principal components of $\mathbf{x}$ are needed to explain most of the variance in the data, the other components do not even have to be estimated. With IncSFA, dimensionality reduction can be done during PC estimation, and no time needs to be wasted on computing insignificant lower-order PCs. The whitening output dimension K must be set by hand[5]. However, some prior problem knowledge seems necessary: the insignificant lower-order PCs may contain

[5]For example, one might set K such that 95% of the estimated total data variance is kept.

Algorithm 5: LearningRateSchedule(θ, t)

```
  //Example Learning Rate Schedule
  //$\theta = (t_1, t_2, c, r, \eta_l, \eta_h, T)$
  //example:  $t_1 = 20, t_2 = 200, c = 3, r = 2000$
1 $\mu_t = \begin{cases} 0 & \text{if } t \le t_1, \\ c(t-t_1)/(t_2-t_1) & \text{if } t_1 < t \le t_2, \\ c + (t-t_2)/r & \text{if } t_2 < t. \end{cases}$
2 $\eta_t^{PCA} \leftarrow (1+\mu_t)/t$
3 $\eta_t^{MCA} = \eta^{MCA}$.
4 return $\{\eta_t^{PCA}, \eta_t^{MCA}\}$
```

data corresponding to the slowest varying signal in the input. It would be unwise to remove them in this case, since discarding these might eliminate an important slow feature.

3.5.3 On complexity

From the algorithm, it can be seen that IncSFA complexity with respect to input dimension is $O(I)$. With respect to the K eigenvectors after the CCIPCA step, and J slow feature eigenvectors, every IncSFA update is of complexity $O(K+J^2)$. The quadratic complexity on J is due to the Gram-Schmidt procedure in the CIMCA algorithm. CCIPCA uses the residual method, which has linear complexity. Typically, $J < K$ and $K < I$, so the quadratic complexity on J should not make the computation inefficient. One must choose K and J and set a learning rate schedule. A discussion on setting learning rates is next.

3.5.4 Learning Rate Scheduling

The methods used to schedule the learning rates η^{PCA} and η^{MCA} are presented in Algorithm 5. There are certainly many other ways to set the learning rates.

Pseudo-optimality

For CCIPCA, the learning rate schedule is based around the optimal $\eta_t^{PCA} = \frac{1}{t}$. If we use $1/t$, Eq. 3.4 will be the most efficient estimator of the principal component. The most efficient estimator on average requires the least samples for learning (among all unbiased estimators). For several common distribution types, e.g., Gaussian, the sample mean is the maximum likelihood estimator of the population mean. And observe that Eq. 3.4 reformulates the eigenvector estimation problem as a mean estimation problem. Therefore, Eq. 3.4 and learning rate $1/t$ has a *spatiotemporal optimality*: at *any* t the estimate is expected to be the best as compared to any other unbiased estimator.

Learning rate $1/t$ is only spatiotemporally optimal if every sample from $t = 1, 2, ..., \infty$ is drawn from the same distribution, which will not be the case for the lower-order components, and in general for autonomous agents. I use an amnesic averaging technique, where the influence of old samples on the current estimates diminish over time. The three-sectioned amnesic averaging function μ is shown in the algorithm. It uses three stages, defined by points t_1 and t_2. In the first stage, the learning rate is $\frac{1}{t}$. In the second, the learning rate is scaled by c to speed up learning of lower-order components. In the third, it changes with t, eventually converging to $1/r$ where r is an amnesic average constant.

This amnesic average remains an unbiased estimator of the true PCs, and it allows components to adapt to changing input statistics. But this plasticity introduces an expected error into the IncSFA whitening process that will not vanish with more samples [Weng and Zhang, 2006]. Results show that this is not problematic for many applications, but it can lead to a slight oscillatory behavior around the true features.

3.5.5 Other Methods of Neural Updating in PC and MC Extraction

Neural layers that compute incremental PCA (IPCA) and MCA build on the work of Amari (1977) and Oja (1982). They showed that a linear neural unit using Hebbian updating could incrementally compute the first principal component of a data set [Amari, 1977; Oja, 1982][6]. Many IPCA algorithms emerged after that. Some well-known ones are Oja and Karhunen's Stochastic Gradient Ascent [SGA; Oja, 1985], Oja's Subspace algorithm [Oja, 1989], Sanger's Generalized Hebbian Algorithm [GHA; Sanger, 1989], the Weighted Subspace algorithm [Oja, 1992], and CCIPCA. For more information on comparisons, see [Oja, 1992; Hyvärinen et al., 2001; Weng et al., 2003]. CCIPCA [Weng et al., 2003] modified GHA to be "candid" — meaning it is invariant to input vector magnitude, thus learning rate tuning became more intuitive, which increased the practicality of the algorithm for high-dimensional inputs such as in appearance-based computer vision. There is another recent IPCA algorithm that adds GSO to CCIPCA [Park and Choi, 2008], so that

[6]Earlier work of a non-neural network flavor had shown how the first PC, including the eigenvalue could be learned incrementally [Krasulina, 1970].

the lower-order components should converge quicker, but with higher complexity.

With handling high-dimensional data in mind, CCIPCA is a chosen for IncSFA for the following reasons:

1. **Covariance-Free**. Mentioned earlier. Due to high-dimensionality, it is important to keep space complexity down.

2. **Avoid Gram-Schmidt orthonormalization (GSO)** for enforcing orthogonality. CCIPCA uses the residual [Kreyszig, 1988] method. GSO will give a more accurate result for lower-order components, but at quadratic complexity (in the number of components). The residual method is local (linear complexity), but can be less accurate. Again, due to high-dimensionality, it is important to avoid quadratic complexity. Further, the experimental results presented earlier showed that effective slow features could emerge even when the whitening matrix was not perfect.

3. **Both Eigenvectors and Eigenvalues Needed**. The method needs to converge to both eigenvectors and eigenvalues, since whitening requires both.

4. **Intuitive to Tune the Learning Rate**. It is not practical to spend a lot of time tuning learning rates for every different type or set of data.

As for MCA: Xu et al. [1992] were the first to show that a linear neural unit equipped with anti-Hebbian learning could extract minor components. Oja modified SGA's updating method to an anti-Hebbian variant [Oja, 1992], and showed how it could converge to the MC subspace. Studying the nature of the duality between PC and MC subspaces [Wang and Karhunen, 1996; Chen et al., 1998], Chen et al. [2001] introduced the sequential addition technique. This enabled linear networks to efficiently extract multiple MCs simultaneously. Building upon previous MCA algorithms, Peng et al. [2007] derived the conditions and a learning rule for extracting MCs for a constant learning rate. Sequential addition was added to this rule so that multiple MCs could be extracted [Peng and Yi, 2006].

I use a modified version of Peng's MCA updating method, slightly altered to be covariance-free and using GSO (CIMCA). Unlike simple MCA algorithms, Peng's MCA is a deterministic discrete time (DDT) method, which requires setting a constant learning rate to achieve convergence. The method has low computational complexity and is shown to work even for singular or near-singular correlation matrix of the input. This makes it practical and especially feasible for non-stationary data where the correlation coefficient behaves like a random variable [Kompella et al., 2014a].

CIMCA extracts the actual minor components (slow features), not just the subspace they span. It allows for a constant learning rate, which can be quite high, leading to a quick reasonable estimate of the true components, and making learning rate tuning more intuitive. Unlike at the IPCA stage, here GSO is useful and plausible since IncSFA does not use many features (minimizing the effect of the quadratic complexity) and their magnitude is not important.

It should be noted that there may be many different ways of combining an incremental PCA and an incremental MCA. My reasons for selecting the methods in IncSFA were presented above, with a motivation to apply IncSFA on real-world image sequences and on vision-based robotic platforms, aiming towards autonomous learning, which is open-ended and continuous.

3.5.6 Links to Biological Systems

BSFA has been shown to derive slow features that operate like biological grid cells from quasi-natural image streams, which are recorded from the camera of a moving agent exploring an enclosure [Franzius et al., 2007]. In rats, grid cells are found in entorhinal cortex [EC; Hafting et al., 2005], which feeds into the hippocampus. Place cells and head-direction cells are found in rat hippocampus [O'Keefe and Dostrovsky, 1971; Taube et al., 1990], while spatial view cells are found in primate hippocampus [Rolls, 1999]. Augmenting the BSFA network with an additional competitive learning (CL) layer derives units similar to place, head-direction, and spatial view cells.

Although BSFA results exhibit the above biological link, it is not clear how the full SFA technique might be realized in the brain. IncSFA with its Hebbian and anti-Hebbian updating provides a more biologically plausible implementation of the full SFA algorithm.

Hebbian Updating in CCIPCA

Hebbian updates of synaptic strengths of some neuron make it more sensitive to expected input activations [Dayan and Abbott, 2001]:

$$\mathbf{v} \leftarrow \mathbf{v} + \eta \; g(\mathbf{v}, \mathbf{u}) \, \mathbf{u}, \tag{3.17}$$

where $\mathbf{u}$ represents pre-synaptic (input) activity and g post-synaptic activity (a function of similarity between synaptic weights $\mathbf{v}$ and input potentials $\mathbf{u}$). Eq. 3.17 requires additional care (e.g., normalization of $\mathbf{v}$) to ensure stability during updating. To handle this in one step, learning rate η and retention rate $1 - \eta$ can be used,

$$\mathbf{v} \leftarrow (1-\eta)\mathbf{v} + \eta\, g(\mathbf{v},\mathbf{u})\,\mathbf{u}. \tag{3.18}$$

where $0 \leq \eta \leq 1$. With this formulation, Eq. 3.4 is Hebbian, where the post-synaptic activity is the normalized response $g(\mathbf{v},\mathbf{u}) = \dfrac{\mathbf{u}(t)\cdot\mathbf{v}(t-1)}{\|\mathbf{v}(t-1)\|}$ and the presynaptic activity is the input $\mathbf{u}_i$.

Anti-Hebbian Updating in CIMCA

The general form of anti-Hebbian updating simply results from flipping the sign in Eq. 3.17. In IncSFA notation:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta\, g(\mathbf{w},\dot{\mathbf{z}})\,\dot{\mathbf{z}}. \tag{3.19}$$

To see the link between Peng's MCA updating and the anti-Hebbian form, in the case of the first MC, note that Eq. 3.6 can be rewritten as

$$\begin{aligned}
\mathbf{w}_1 &\leftarrow 1.5\mathbf{w}_1 - \eta\, \left[\mathbf{C}_1\,\mathbf{w}_1 + [\mathbf{w}_1^T\mathbf{w}_1]\,\mathbf{w}_1\right], && (3.20)\\
&= 1.5\mathbf{w}_1 - \eta\, \left[(\dot{\mathbf{z}}\cdot\mathbf{w}_1)\,\dot{\mathbf{z}} + (\mathbf{w}_1\cdot\mathbf{w}_1)\,\mathbf{w}_1\right], && (3.21)\\
&= 1.5\mathbf{w}_1 - \eta\, \|\mathbf{w}_1\|^2\,\mathbf{w}_1 - \eta\, \left((\dot{\mathbf{z}}\cdot\mathbf{w}_1)\,\dot{\mathbf{z}}\right), && (3.22)\\
&= \left(1.5 - \eta\, \|\mathbf{w}_1\|^2\right)\,\mathbf{w}_1 - \eta\,(\dot{\mathbf{z}}\cdot\mathbf{w}_1)\,\dot{\mathbf{z}}, && (3.23)
\end{aligned}$$

where $(\dot{\mathbf{z}}\cdot\mathbf{w}_1)$ indicates post-synaptic strength, and $\dot{\mathbf{z}}$ pre-synaptic strength.

Hebbian Learning on Filtered Output

There is an alternative to using anti-Hebbian learning. Sprekeler et al. [2007] reformulated the slowness objective: instead of minimizing the variance of the time derivative of the output signal, they try to *maximize* the variance of the *low-pass filtered* output signal. They show analytically that the extraction of the single most slowly varying direction from pre-whitened input can be implemented in a linear continuous model with spiking model neurons by means of a modified hebbian learning rule with a specific learning window.

Hebbian learning between a temporally filtered output and input is the basis of several other temporal-stability based learning rules [Földiák, 1991; O'Reilly and Johnson, 1994; Wallis and Rolls, 1997]. Links between these and slowness learning are provided by Sprekeler et al. [2007]. However, even though Sprekeler's method is only for the first feature, it might lead to alternate approach to reach a fully incremental SFA.

3.5.7 Velocity Estimates of the Input Signal

The velocity estimates (the derivative signal) in the original SFA technique are approximated via a backward difference method $\dot{\mathbf{z}}(t) = \mathbf{z}(t) - \mathbf{z}(t-1)$. This method behaves badly in the presence of input noise compared to other methods (that are more computationally expensive) such as higher order difference estimation, Cauchy's differentiation formula, or Lanczos derivative computation [Groetsch, 1998]. However, noise is usually not a severe problem, since it changes at a faster time-scale compared to the slowest components and therefore does not show up in the higher-order slow features. Therefore, I opted to use the same backward difference method for the IncSFA to keep it computationally simple.

3.6 Conclusion

Incremental Slow Feature Analysis is an unsupervised learning technique that updates slow features incrementally without computing covariance matrices. For many instances, there is no need to use IncSFA instead of BSFA. But at higher dimensionality, IncSFA becomes more and more appealing. For some problems with very high dimensionality and limited memory, IncSFA could be the only option, e.g., an autonomous robot with limited onboard hardware, which could still learn slow features from its visual stream via IncSFA. Experiments showed how IncSFA enables an adaptive SFA, and how it enables SFA to be applied to high-dimensional image streams without using multilayer receptive-field based BSFA architectures. The following summarizes the advantages IncSFA has over BSFA:

- **Adaptation to Changing Input Statistics.** In the BSFA paradigm, new data cannot be used to modify already learned slow features. If input statistics change, IncSFA can adapt existing features without outside intervention, while BSFA has to discard previous features to process the new data.

- **Computational Efficiency.** BSFA techniques rely upon batch Principal Component Analysis (PCA). For input observations in an I-dimensional space, the computational complexity of PCA using the Jacobi method [Forsythe and Henrici, 1958] is of the order $O(I^3)$. IncSFA's updating complexity scales linearly with dimensionality ($O(I)$). Thus it has an advantage when the input dimension is large.

- **Space Complexity.** First, IncSFA can discard each observation immediately after an update. Second, note that IncSFA uses *covariance-free* techniques, where the data covariance matrices never need to be computed, even in passing. In a covariance-free technique, the features are updated directly from the new data. The $I(I+1)/2$ parameters in the covariance matrix do not have to be estimated.

- **Simplicity.** For extracting features from high-dimensional image sequences, IncSFA presents a simpler solution method than the alternate technique of deep receptive-field based BSFA networks. By simpler, I mean that IncSFA has just a handful of parameters, instead of the multitude of parameters associated with the deep nets.

- **Reduced Sensitivity to Outliers.** Outlier observations in a dataset can cause problems for BSFA, as these outliers can corrupt the slow features. In some cases, a feature may even become sensitive to an outlier. In a typical batch implementation, each observation has the same amount of influence on the features. In IncSFA, the influence of a single observation fades as newer observations are experienced. The learning rate implicitly controls this forgetting factor. Different learning rate settings can lead to different features — features that emerge from a high learning rate setting are biased to detect slowly-changing phenomena that occur with more regularity than if a lower learning rate were to be used.

- **Biological Plausibility.** IncSFA adds further biological plausibility to SFA. SFA itself has been linked to biological systems due to the results in deriving place cell, grid cells, etc., but it is difficult to see how BSFA could be realized in the brain. IncSFA's updates can be described in incremental Hebbian and anti-Hebbian forms.

These advantages make IncSFA suitable to use for several online learning applications. IncSFA does not store previously learned representations. This poses a problem in the case of open-ended learning, where it becomes essential to not relearn previously encoded inputs. Therefore learning progress must be cached for future use. The next chapter presents a modular slow feature learning algorithm that addresses this issue.

Chapter 4

Curiosity-Driven Modular Incremental Slow Feature Analysis

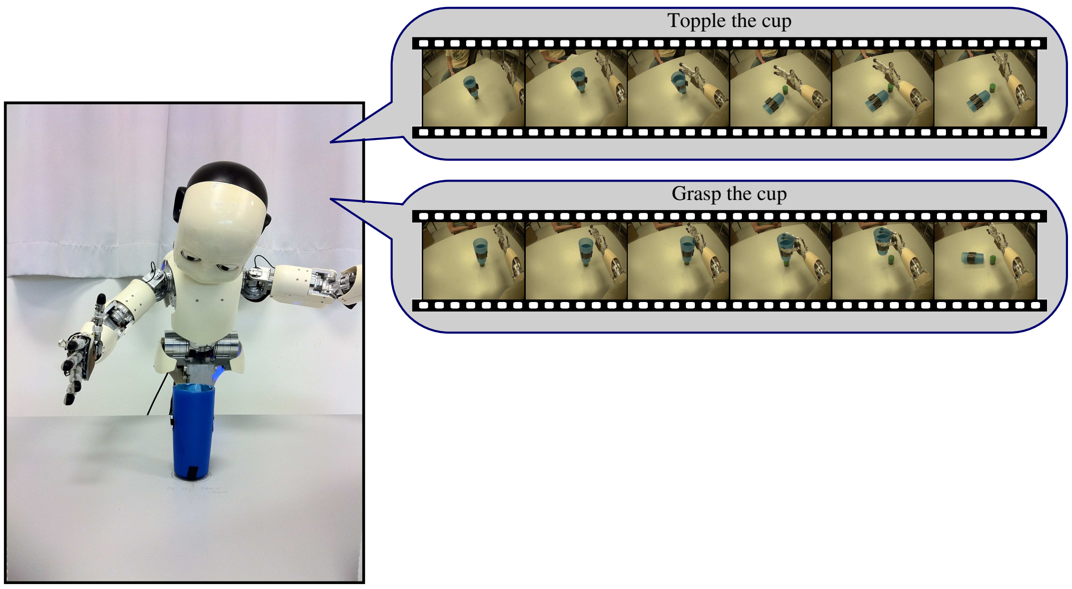


Figure 4.1. How can a robot quickly acquire multiple abstractions from a large set of unknown (including random) behaviors? Curiosity-Driven Modular Incremental Slow Feature Analysis (Curious Dr. MISFA) addresses this by combining IncSFA with a curiosity drive to autonomously learn multiple slow feature abstractions in the order from least to most costly.

In Chapter 3, the Incremental SFA (IncSFA) algorithm was presented, which extracts slow features without storing or estimating computationally expensive covariance matrices of the input data. This makes it suitable to use IncSFA for applications with high-dimensional images as inputs. However, if the statistics of the inputs change over time, like most online-learning approaches, IncSFA gradually forgets previously learned representations, for instance, if the robot (see Figure 4.1) changes executing actions from toppling the cup to grasping the cup. How can the robot quickly acquire multiple abstractions from a large set of such unknown (including random) action-sequences? I present here an online-learning algorithm called Curiosity-Driven Modular Incremental Slow Feature Analysis [Curious Dr. MISFA; Kompella* et al., 2013; Kompella et al., 2012b]. Curious Dr. MISFA combines IncSFA with a curiosity drive to autonomously learn multiple slow feature abstractions in order from least to most costly, with theoretical guarantees. A gating feature is used to store good abstractions and preserve them for later use with learned tasks. The overall system is shown to have interesting and useful properties through experiments.

4.1 Overview

In this section, I will present an overview of the Curious Dr. MISFA algorithm. Figure 4.2 shows the architecture of Curious Dr. MISFA. The input to the algorithm is a set of pre-defined high-dimensional observation streams $X = \{\mathbf{x_1}, ..., \mathbf{x_n} : \mathbf{x_i}(t) \in \mathbb{R}^I, I \in \mathbb{N}\}$, which may or may not be unique. At any time t, the agent observes an input sample from only one of the observations streams. This is analogous to watching different channels on a television or making observations while executing different tasks[1]. The desired outcome of the learning process is a sequence of abstractions $\Phi_t = \{\phi_1, ..., \phi_m; m \leq n\}$ that are learned in order of increasing learning difficulty. Each abstraction $\phi_i : \mathbf{x} \to \mathbf{y}$ is unique and maps one or more observation streams $\mathbf{x} \in X$ to a low-dimensional output $\mathbf{y}(t) \in \mathbb{R}^J, J \in \mathbb{N}$. Since the learning difficulty of the observation streams is not known *a priori*, the learning process involves estimating not just the abstractions, but also the order in which the observation streams need to be encoded. To this end, Curious Dr. MISFA uses reinforcement learning to learn an optimal observation stream selection policy, based on the intrinsic rewards proportional to the progress made while learning the abstractions. The architecture of Curious Dr. MISFA includes (a) a reinforcement learning (RL) agent that generates an observation stream selection policy, (b) an adaptive Incremental Slow Feature Analysis coupled with a Robust Online Clustering (IncSFA-ROC; see Section 4.3.2 for details) module that updates an abstraction based on the incoming

[1] It is straightforward to solve the problem of learning multiple abstractions if the agent can observe samples from all the streams at time t. Abstractions corresponding to each stream can simply be learned in parallel.

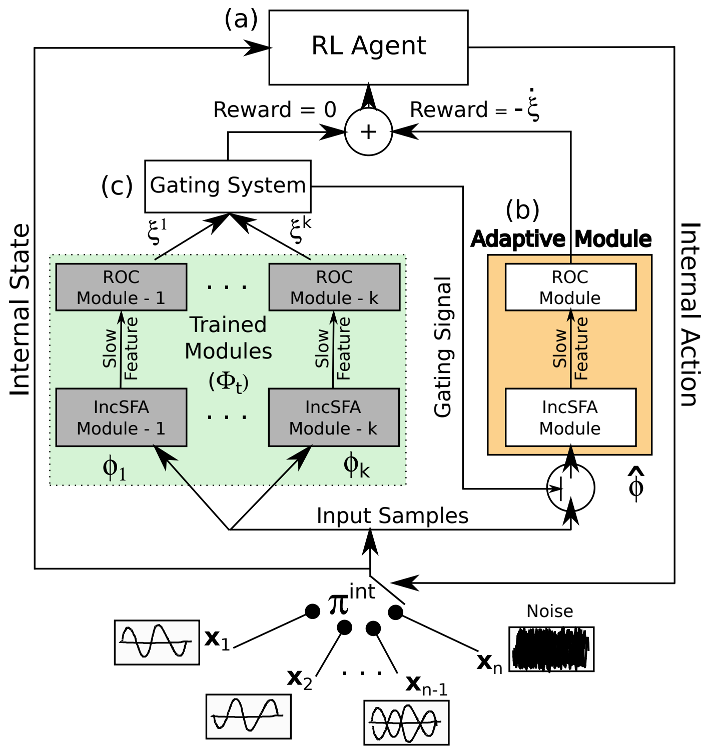


Figure 4.2. Architecture of Curious Dr. MISFA includes (a) a reinforcement learning agent that generates an observation stream selection policy based on the intrinsic rewards, (b) an adaptive Incremental SFA coupled with a Robust Online Clustering module that updates an abstraction based on the incoming observations, and (c) a gating system that prevents encoding observations that have been previously encoded.

observations, and (c) a gating system that prevents encoding observations that have been previously encoded.

The design of the RL agent and the intrinsic rewards are **crucial** to ensure stability of the method (see Section 4.3.7 for details). An overview of these and the control flow of the algorithm are discussed next.

Design of the RL agent. The RL agent is within an internal environment that has a set of discrete internal states $\mathcal{S}^{\text{int}} = \{s_1^{\text{int}}, ..., s_n^{\text{int}}\}$, equal to the number of observation streams. In each state s_i^{int}, the agent is allowed to take only one of the two actions ($\mathcal{A}^{\text{int}}$): *stay* or *switch*. The action *stay* makes the agent's internal state to be the same as the previous state, while *switch* randomly shifts the agent's state to one of the other internal states (see Section 4.3.1 for details on why this is crucial). The agent at each state s_i^{int}, receives a fixed τ time step sequence of observations ($\mathbf{x}$) of the corresponding stream $\mathbf{x}_i$.

The agent maintains a single adaptive abstraction $\widehat{\phi} \in \mathbb{R}^{I \times J}, \widehat{\phi} \notin \Phi_t$ that updates via an *IncSFA-ROC* abstraction-estimator (denoted by Θ) based on the observations $\mathbf{x}$. To prevent it from encoding inputs that have been previously encoded, a *gating function* $\mathscr{G} : X \to \Phi_t \cup \widehat{\phi}$ assigns the appropriate abstraction $\phi_i \in \Phi_t$ to the observation stream if the observed estimation error is low, otherwise $\widehat{\phi}$ is assigned. Let $\xi = \|\Theta(\mathbf{x}, \mathscr{G}(\mathbf{x})) - \mathscr{G}(\mathbf{x})\|$ denote the error made by the abstraction-estimator for the input-samples $\mathbf{x}$.

Design of the intrinsic reward. The goal of the RL agent is to learn an observation stream selection policy $\pi^{\text{int}} : \mathscr{S}^{\text{int}} \to \mathscr{A}^{\text{int}}$ that optimizes the following cost-function:

$$\mathscr{J} = \min_{\pi^{\text{int}}} \left(\langle \dot{\xi} \rangle_t^\tau, \langle \xi \rangle_t^\tau \right). \tag{4.1}$$

Eq. (4.1) is a multi-objective reinforcement learning problem [MORL; Gábor et al., 1998; Vamplew et al., 2011], where $\langle \dot{\xi} \rangle_t^\tau$ is time-derivative of the τ-window averaged error $\langle \xi \rangle_t^\tau$. Minimization of the first objective would result in a policy that will shift the agent to states where the error decreases sharply ($\langle \dot{\xi} \rangle_t^\tau < 0$), indicating faster learning progress of the abstraction-estimator. Minimization of the second objective results in a policy that improves the developing abstraction to better encode the observations.

The two objectives are correlated but partially conflicting: optimizing the first objective aids in optimizing the second, however, optimizing the second objective would result in an increasing error-gradient $\dot{\xi}$ (from a negative value to 0), which conflicts with the first. Therefore, there does not exist a single solution that simultaneously optimizes each objective. I instead use an approach to find a dynamically changing *pareto-optimal policy* [Vamplew et al., 2011], by prioritizing each objective based on the error ξ. To this end, the cost is scalarized in terms of a scalar *reward* r^{int} that evaluates the input $\mathbf{x}$ received for the tuple $(s^{\text{int}}, a^{\text{int}}, s_{-}^{\text{int}})$ as follows:

$$r^{\text{int}} = \left(\alpha \langle \dot{\xi} \rangle_t^\tau + \beta Z(|\delta - \langle \xi \rangle_t^\tau|) \right), \tag{4.2}$$

where Z is a normally distributed random variable with mean δ and standard deviation σ. Here, α, β, δ and σ are constant scalars (see Sections 4.3.3 and 4.3.4 for details on how this reward is realized for the IncSFA-ROC abstraction-estimator).

Control flow of the algorithm. Figure 4.3 shows the control flow diagram of Curious Dr. MISFA. A *reward function* $R_t^{\text{int}} : \mathscr{S}^{\text{int}} \times \mathscr{A}^{\text{int}} \to \mathbb{R}$ is estimated based on the scalar-rewards r^{int} received for $(s^{\text{int}}, a^{\text{int}}, s_{-}^{\text{int}})$ tuples (see Section 4.3.4 for details). The observation stream selection policy π^{int} at time t, is learned using Model-based Least-Squares Policy Iteration Technique [LSPI; Lagoudakis and Parr, 2003] based on the current estimated reward function R_t^{int} and the known transition-model. I use

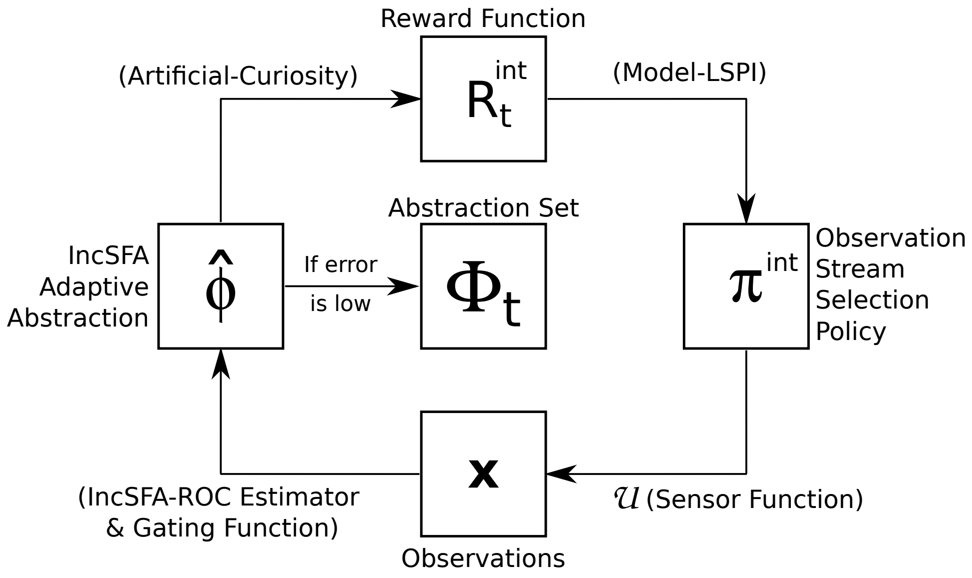


Figure 4.3. The control flow of Curious Dr. MISFA involves a simultaneous estimation of (a) reward function (R_t^{int}) using the principle of artificial curiosity, (b) an abstraction ($\widehat{\phi}$) using IncSFA-ROC algorithm and (c) an observation stream selection policy (π^{int}) using model Least Squares Policy Iteration (LSPI) algorithm. The gating-function prevents encoding previously encoded observation streams. $\widehat{\phi}$ is added to the learned abstraction set (Φ_t) if the estimation-error is below a low threshold (δ).

epsilon-greedy [Sutton and Barto, 1998] strategy over π^{int} to balance between exploration and exploitation (see Section 4.3.4 for details). The observation stream selection policy is then used to generate $(s^{\text{int}}, a^{\text{int}}, s_-^{\text{int}})$ samples and the corresponding new observations **x** via the sensor-function $\mathcal{U}$. These new observations, if not encodable by previously learned abstractions, are used to update the adaptive abstraction $\widehat{\phi}$ and re-estimate the reward function R_t^{int}. The updated abstraction $\widehat{\phi}$ is added to Φ_t, (*i.e.*, $\Phi_t \leftarrow \Phi_t \cup \widehat{\phi}$) if the error ξ falls below a low threshold δ. If and when added, a new adaptive abstraction $\widehat{\phi}$ is instantiated and the process continues.

The rest of the chapter is organized as follows. Section 4.2 presents a theoretical formulation of the learning problem associated with Curious Dr. MISFA. Section 4.3 discusses different parts of the Curious Dr. MISFA algorithm in detail. Section 4.4 presents a pseudocode of the algorithm along with other implementation details. Section 4.5 explores the application of Curious Dr. MISFA to environments such as a room-maze where each room has a time-varying audio or a video source. Section 4.6 presents experimental results. Section 4.7 discusses neurophysiological correlates of the algorithm and Section 4.8 concludes.

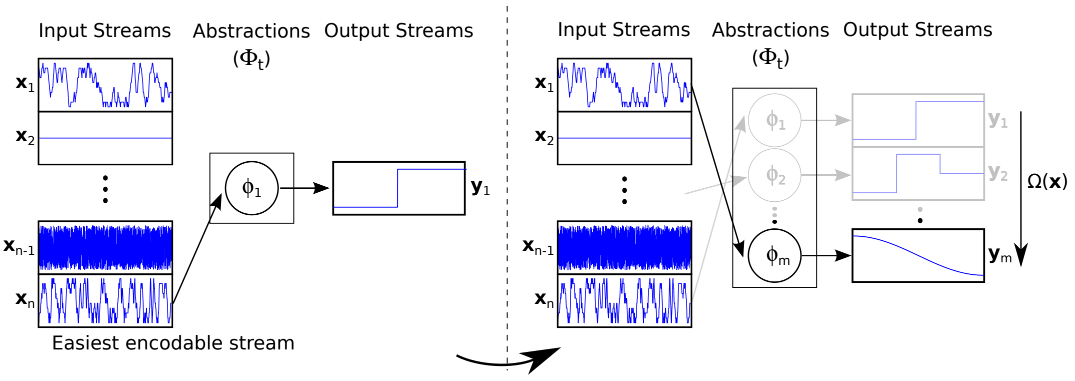


Figure 4.4. Given a set of time-varying observation streams, an abstraction corresponding to the easiest-encodable yet unknown observation stream is learned first. (Left-Figure) An example result after the first abstraction was learned. (Right-Figure) $\Omega(\mathbf{x})$ denotes the curiosity-function that maps an observation stream $\mathbf{x}$ to a scalar value in $[0,1]$. Difficult-to-encode observation streams have higher Ω values. Figure shows the desired result, which is a sequence of abstractions $\{\phi_i\}_{i\in\mathbb{N}}$ learned in the order of increasing Ω-values of the observation streams that they encode. However, Ω is not known *a priori*, therefore, the learning process involves estimating both the abstractions and the Ω-values. The curved arrow indicates the temporal evolution of the learning process.

4.2 Learning Problem Formalized

The underlying learning problem associated with Curious Dr. MISFA can be formulated as an *optimization* problem. Simply put, the problem states that for a given set of time-varying observation streams, an abstraction corresponding to the most easily encodable yet unknown observation stream is learned first. The result is an ordered sequence of learned abstractions $\{\phi_1, ..., \phi_m\}$, where each abstraction maps one or more inputs to a lower-dimensional output. The optimization problem is not specific to a particular type of abstraction-estimator and therefore addresses a class of related problems. Later in the chapter, I will show that the Curious Dr. MISFA algorithm converges to the *optimal* solution of the proposed optimization problem. Figure 5.2 illustrates the learning process. The formalized problem is as follows:

Notation:

Input: Let $\mathscr{X} = \{\mathbf{x} : \mathbf{x}(t) \in \mathbb{R}^I, I \in \mathbb{N}\}$ denote a set of of I-dimensional observation streams. Let $X \subset \mathscr{X}$ be a finite subset with $n \in \mathbb{N}$ elements that may or may not be unique. Each $\mathbf{x_i} \in X$ is a stream generated by a sensor-function[2] $\mathscr{U}$ of an arbitrary

[2]A sensor-function could represent a single or a collection of different sensor modalities.

discrete-time process[3] $\mathscr{F}^i$ indexed by $t \in \mathbb{N}$, *i.e.* $\mathbf{x_i}(t) = \mathscr{U}(\mathscr{F}^i(t)), \forall i \in \{1, ..., n\}$. At each time t however, an input sample is available from only ***one*** of the n observation streams.

Abstraction: Let Θ denote some online abstraction-estimator that updates a feature-abstraction ϕ, where $\Theta(\mathbf{x}, \phi)$ returns an updated abstraction for an input $\mathbf{x}$. The abstraction $\phi : \mathbf{x} \mapsto \mathbf{y}$ maps a high-dimensional observation stream $\mathbf{x} \in X$ to a lower-dimensional output $\mathbf{y}(t) \in \mathbb{R}^J, J \ll I, J \in \mathbb{N}$, such that $\mathbf{y}(t) = \phi(\mathbf{x}(t))$. Let Φ_t denote the set of learned-abstractions at time t, such that, $\Phi_{t_1} \subseteq \Phi_{t_2}, \forall t_1 < t_2$. Let X^{Φ_t} denote the set of *pre-images* of the corresponding feature outputs $\mathbf{y}_i$, $X^{\Phi_t} = \{\phi_i^{\leftarrow} \mathbf{y}_i, \ \forall \phi_i \in \Phi_t\}$. X^{Φ_t} represents the encoded observation streams at time t.

Curiosity Function: Let $\Omega : \mathscr{X} \rightarrow [0, 1)$ denote a function indicating the speed of learning an abstraction by the abstraction-estimator Θ. Easily encodable inputs have lower values of Ω. Ω induces a total ordering among the input streams making them comparable in terms of the learning difficulty (see Section 4.3.7 for a proof on the existence of such a function). Let $\widehat{X}$ and $\widehat{X}^{\Phi_t}$ denote the ordered-sets (induced by Ω) of X (excluding constant streams) and X^{Φ_t} respectively. Let $\Omega[X] = [\Omega(\mathbf{x}_1), ..., \Omega(\mathbf{x}_n)]^T$ denote a column-vector of the Ω-values corresponding to the components of the set X.

Other Notation: Let Λ_t denote a $|\widehat{X}^{\Phi_t}| \times |\widehat{X}|$ *diagonal-matrix* at time t, with the main-diagonal entries equal to 1 and rest 0. $|.|$ indicates cardinality of a set, $||.||$ indicates Euclidean norm, $\langle . \rangle_t$ indicates averaging over time, $\langle . \rangle_t^\tau$ indicates windowed-average with a fixed window size τ over time, δ is a small scalar constant (≈ 0) and $\forall$ indicates forall.

With the above notation, the optimization problem is formulated as follows: Given the input X, find a finite **sequence** of m abstractions $\{\phi_i\}_{i=0}^m$, $m \leq n$, such that **at any time** t, the following objective is minimized:

$$\min_{\Phi_t} \ \left\| \Omega[\widehat{X}^{\Phi_t}] - \Lambda_t \Omega[\widehat{X}] \right\|, \quad \forall t = 1, 2, ...$$

under the constraints,

$$\langle y_j^i \rangle_t = 0, \quad \langle \left(y_j^i \right)^2 \rangle_t = 1, \ \forall i \in \{1, ..., J\}, \forall j \in \{1, ..., \left|\Phi_t\right|\} \ \text{(std. normal stats)} \quad (4.3)$$

$$\begin{pmatrix} \forall \phi_i \in \Phi_t, \exists j \in \{1, ..., n\}, \\ \text{and} \quad \forall \phi_{k \neq i} \in \Phi_t \end{pmatrix} : \begin{array}{ll} \langle ||\Theta(\mathbf{x_j}, \phi_\mathrm{i}) - \phi_\mathrm{i}|| \rangle_\mathrm{t}^\tau \leq \delta & \text{(at least one stream)} \\ \langle ||\Theta(\mathbf{x_j}, \phi_\mathrm{k}) - \phi_\mathrm{k}|| \rangle_\mathrm{t}^\tau > \delta & \text{(unique abstraction)} \end{array} \quad (4.4)$$

[3] $\mathscr{F}^i$ represents *driving-forces* [Wiskott, 2003] generating the observation stream.

The first term $\Omega[\widehat{X}^{\Phi_t}]$ in the objective denotes an ordered array of Ω values of the observation streams that have been encoded until time t, while the second term $\Lambda_t \Omega[\widehat{X}]$ indicates an ordered array of Ω values of the top $|\widehat{X}^{\Phi_t}|$ easily-encodable observation streams. The objective is to minimize the difference between the two terms while satisfying two constraints: Constraint (4.3) requires that the abstraction-output components have zero mean and unit variance. This constraint enables the abstractions to be non-zero and avoids learning features for constant observation streams. Constraint (4.4) requires that a *unique* abstraction be learned that encodes *at least* one of the observation streams, avoiding redundancy. Each learned abstraction is therefore a lower-dimensional *invariant* representation of some underlying regularity among the discrete-time processes $\{\mathscr{F}^i\}_{i \in \{1,\dots,n\}}$.

Optimal Solution: For the objective to be minimized under the Constraints (4.3)-(4.4), at any time t, the optimal solution is to learn an abstraction corresponding to the current *easiest but not-yet-learned regularity* among the observation streams as measured by the curiosity-function Ω. The result is a set of learned abstractions $\{\phi_1, \dots, \phi_m\}$, ordered according to the increasing Ω-value of the corresponding observation streams that they encode.

However, since Ω is not known *a priori*, it needs to be estimated online by actively exploring the observation streams. One possible approach is to find (a) an analytical expression of Ω for the particular abstraction-estimator Θ and (b) an input sampling technique that can estimate the Ω values for each observation stream. This approach is dependent on Θ and may not be applicable to other abstraction-estimators. A more general approach is to use the *learning progress* of Θ, while exploring using reinforcement learning (RL) to estimate the Ω values in the form of *curiosity rewards* for each observation stream. This approach is independent of the abstraction-estimator used. However, it requires learning an observation stream selection policy π^{int}, and at the same time the abstraction from the incoming input-samples based on the (imperfect) policy π^{int}. Curious Dr. MISFA is an iterative algorithm that solves the optimization problem (see Section 4.3.7). A detailed description of the method is discussed next.

4.3 Method Description

Curious Dr. MISFA is a curiosity-based abstraction learning algorithm that uses incremental slow feature analysis to learn abstractions by exploring among a set of input observation streams using a *stay-switch* action selection mechanism. The following sections discuss the components of Curious Dr. MISFA in detail.

4.3.1 RL Agent's Internal Environment

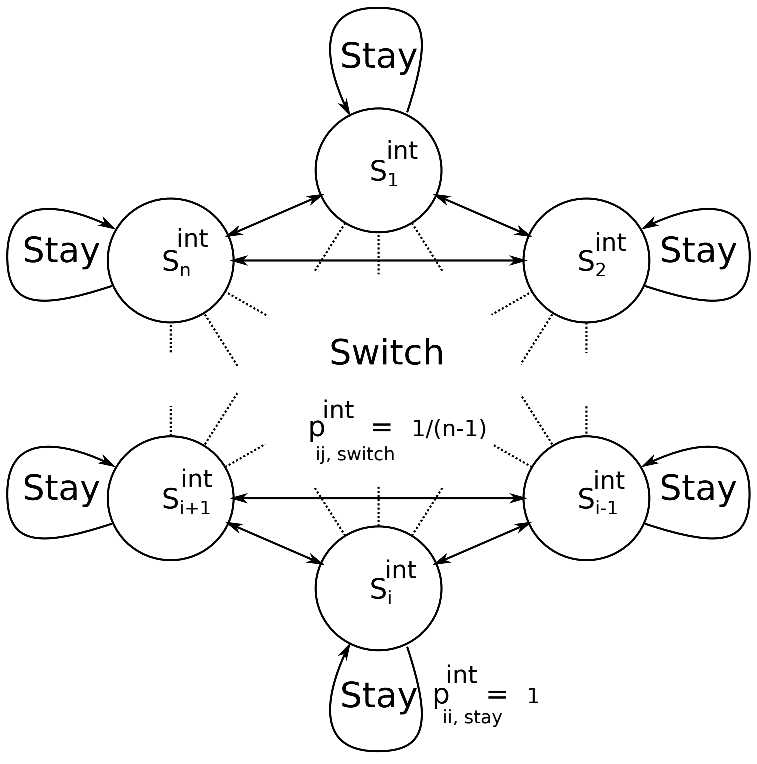


Figure 4.5. The state-action transition model of the agent's internal environment resembles that of a complete-graph markov chain.

Section 4.1 presented an overview on the design of the RL agent. Here, I will discuss more details. For the sake of completeness, some of the notation used in Section 4.1 is re-defined here. The RL agent learns to select the current easiest but not yet encoded stream among the observation streams. The environment comprises a set of discrete internal states $\mathscr{S}^{\text{int}} = \{s_1^{\text{int}}, ..., s_n^{\text{int}}\}$ each corresponding to an input stream $\mathbf{x}_i, \forall i \in \{1, ..., n\}$. The agent at any state $s^{\text{int}} \in \mathscr{S}^{\text{int}}$ can take only one of the two actions: *stay* or *switch*. The action *stay* makes the agent's internal state to be the same as the previous state, while *switch* **randomly** (uniformly) shifts the agent's internal state to one of the other neighboring states. The random nature of the switching forces the IncSFA-ROC abstraction-estimator not to encode any regularity while switching between the states. This is crucial to avoid combinatorial possibilities of generating a coherent stream of data (both in space and time) by deterministically switching between a few states at different times.

The transition model $\mathscr{P}^{\text{int}} : \mathscr{S}^{\text{int}} \times \mathscr{A}^{\text{int}} \times \mathscr{S}^{\text{int}} \rightarrow [0, 1]$ of the environment dynamics resembles that of a *Complete-Graph*, where each state $s_i^{\text{int}} \in \mathcal{S}^{\text{int}}$ is represented by a node in a fully-connected undirected-graph.

$$P_{ij,\text{stay}}^{\text{int}} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}, \; P_{ij,\text{switch}}^{\text{int}} = \begin{cases} 0, & \text{if } i = j \\ \frac{1}{n-1}, & \text{if } i \neq j \end{cases}, \quad \forall i, j \in [1, ..., n]. \tag{4.5}$$

The agent has the capability to shift between any of observation streams, similar to switching between channels of a television or switching between several tasks resulting in different observation streams of data. Figure 4.5 illustrates the model.

Abstracted State Space: Let the discrete-time processes (driving forces) $\mathcal{F}^i$ be defined over a finite state space $\mathcal{S}$. The agent perceives these states partially through previously learned abstractions $\phi_i \in \Phi_t$. The agent's abstracted state space $\mathcal{S}^\Phi = \{s_1^\Phi, ..., s_p^\Phi\}$ is defined such that it contains the space spanned by the outputs $\mathbf{y}$ of all the abstractions that were previously acquired using Θ. The agent learns a new abstraction for an observation stream $\mathbf{x}_i$ if the encoded abstraction outputs are predictable with respect to $\mathcal{S}^\Phi$. In the experiments, I assume that the agent has a default abstracted state space. For example, in the robot experiment, the agent's default abstracted state space contains low-level kinematic body joint poses of the robot learned offline using Task Relevant Roadmaps [Stollenga et al., 2013]. The next section presents an intuitive example of the abstracted state space.

4.3.2 Abstraction-Estimator (Θ): IncSFA-ROC

Curious Dr. MISFA's abstraction-estimator method is the Incremental Slow Feature Analysis (IncSFA) coupled with a Robust Online Clustering [ROC; Guedalia et al., 1999; Zhang et al., 2005] algorithm. IncSFA is used to learn a real-valued slow feature abstraction (ϕ) of the input whereas ROC is used to learn a discrete model mapping the slow feature outputs with respect to the abstracted state space $\mathcal{S}^\Phi$.

As an example, consider a robot viewing its moving arm that topples an object in the scene (Figures 4.6(a)-(c)). The state space $\mathcal{S}^\Phi$ here consists of discretized joint angles of the shoulder into p=*20* bins ($\mathcal{S}^\Phi = \{s_1^\Phi, ..., s_p^\Phi\}$). For each s_i^Φ, there is an associated instance of the clustering algorithm, leading to p=*20* instances of the clustering algorithm. A developing slow feature output here is a step function (Figure 4.6(d)), e.g., when the object is not toppled the feature output equals ≈ -1.5, and when the object is toppled the feature output equals ≈ 0.5. Upon convergence of IncSFA first and the clustering algorithm second, each joint angle will be mapped to two cluster centres (Figure 4.6(e)), (except for the joint angles 15-20, where the iCub's hand is to the left of the object's position and the object cannot be in not-toppled position) providing information about invariants captured with IncSFA.

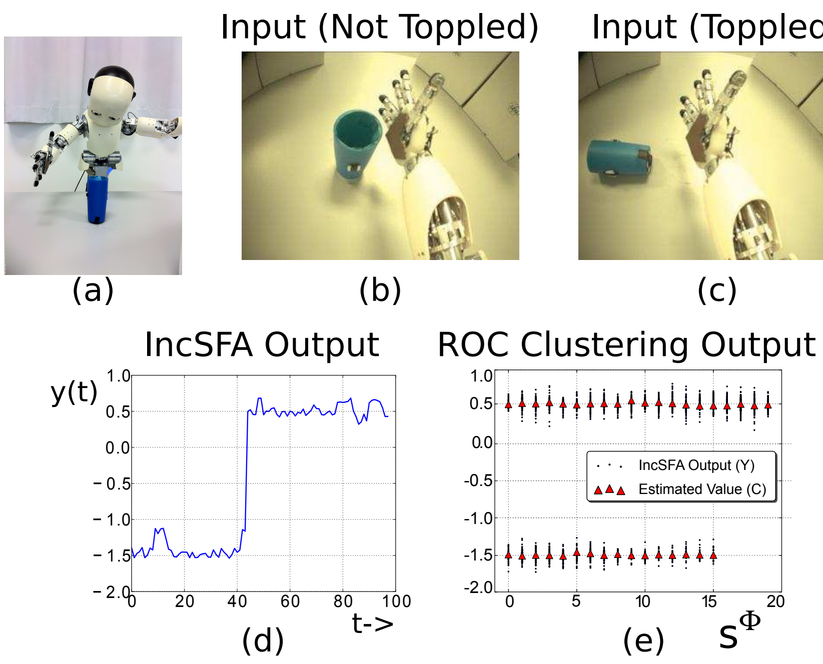


Figure 4.6. IncSFA-ROC example. (a) The iCub is placed next to a table with a plastic cup in its reach. It can interact with the cup by moving its right hand along its shoulder joint. (b) A sample input image when the cup is not toppled. (c) A sample input image when the cup is toppled. (d) IncSFA output plotted against time after a few time steps of exploration. The output is a step function indicating whether the cup is toppled or not, invariant to the position of its hand visible in the image. (e) Learned ROC cluster centers map the feature output with respect to the angle of the shoulder joint.

The slow feature outputs can change rapidly during the training phase. Therefore, learning these clusters is not as straightforward as the above example makes it seem. The estimator has to be able to change its estimates to this non-stationary input, while converging to a good estimate when the input becomes stable. ROC algorithm is especially suitable for handling non-stationary data. It is similar to an incremental K-means algorithm [Forgy, 1965] — a set of cluster centers is maintained, and with each new input, the most similar cluster center (the winner) is adapted to become more like the input. Unlike K-means, with each input, the adaptation step is followed by *merging* the two most similar cluster centers and *creating a new cluster center* at the latest input. In this way, ROC can quickly adjust to non-stationary input distributions by directly adding a new cluster for the newest input sample, which may mark the beginning of a new input process.

But is this plasticity at the cost of stability? No. In order to enforce stability, clusters maintain a weight, which increases faster for inputs more similar to the

Algorithm 6: ROC-Amnesic($\mathbf{y}, s^{\Phi}, N^{\max}, \nu^{\text{amn}}$)

```
    //Cluster SFA-encoded samples $\mathbf{y} \in \mathscr{R}^J$
    //$\mathbf{y}$ : IncSFA feature output
    //$s^{\Phi}$ : Current abstracted state
    //$N^{\max} > 1$ : Maximum number of clusters
    //$0 \leq \nu^{\text{amn}} \leq 1$ : Amnesic parameter
    //Determine which set of clusters to use
    //$\mathbf{C}$ : Set of cluster centers
    //$\mathbf{a}$ : Set of cluster weights
 1  $\{\mathbf{C}, \mathbf{a}\} \leftarrow$ GetClusteringInstance ($s^{\Phi}$)
 2  if $|\mathbf{C}| < N^{max}$ then
        //Cluster center is $\mathbf{y}$, weight is 0
 3      $\{\mathbf{C}, \mathbf{a}\} \leftarrow$ AddNewCluster ($\mathbf{y}, \mathbf{C}, \mathbf{a}$)
 4  else
 5      $winner \leftarrow \arg\max_i \text{Response}(\mathbf{y}, \mathbf{c}_i)$
 6      $\mathbf{c}_{winner} \leftarrow \mathbf{c}_{winner} + \frac{\mathbf{y} - \mathbf{c}_{winner}}{a_{winner} + 1}$
 7      $a_{winner} \leftarrow a_{winner} + \text{Response}(\mathbf{y}, \mathbf{c}_{winner})$
 8      $\{\gamma, \rho\} \leftarrow \arg\max_{\gamma,\delta,\gamma \neq \delta} \text{Response}(\mathbf{c}_{\gamma}, \mathbf{c}_{\delta})$   //Merge the two closest
 9      $\mathbf{c}_{\gamma} \leftarrow \frac{\mathbf{c}_{\gamma} a_{\gamma} + \mathbf{c}_{\delta} a_{\delta}}{a_{\gamma} + a_{\rho}}$
10      $a_{\gamma} \leftarrow a_{\gamma} + a_{\rho}$
11      $\mathbf{c}_{\rho} \leftarrow \mathbf{y}$        //Latest input becomes new cluster
12      $a_{\rho} \leftarrow 0$
13      for $i \leftarrow 1$ to $N^{max}$ do
14          $a_i \leftarrow a_i(1 - \nu^{\text{amn}})$        //Forgetting (leak)
15      end
16  end
```

cluster center. A large weight prevents a cluster center from changing that much. When two clusters are merged, their weights are combined.

A sketch of the ROC per-sample update is in Alg. 6. The ROC algorithm repeatedly iterates through the following steps. For every input sample the algorithm finds the closest cluster $winner$ and updates the center $\mathbf{c}_{winner}$ towards it, also increasing the weighting parameter a_{winner}. Next, the closest two clusters are merged into one cluster. Then, a new cluster is created around sample $\mathbf{y}$. Finally, all clusters weights decrease slightly. Parameters required are $N^{\max}$, the maximum number of clusters, an amnesic parameter ν^{amn} to prevent convergence, and the response function for similarity measurement. The next section discusses how the error while learning the abstractions is computed.

4.3.3 Estimation error (ξ)

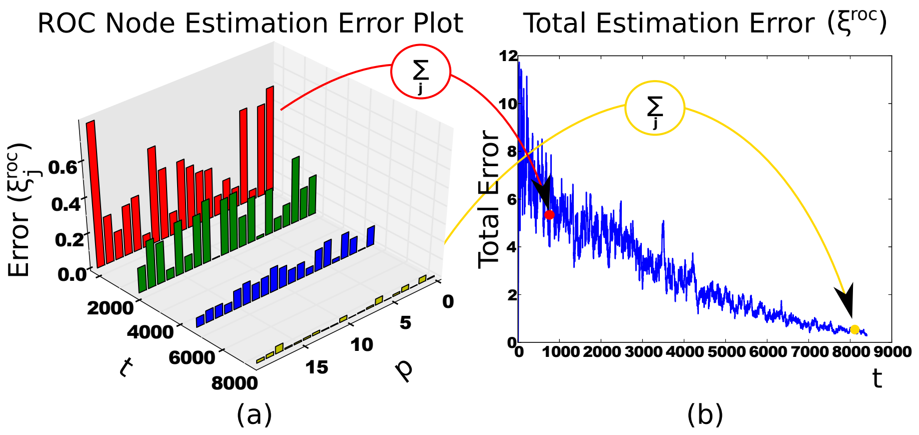


Figure 4.7. An example estimation error over time of the ROC-algorithm. (a) The estimation error of 20 instances of ROC nodes to estimate the IncSFA output to 20 values. (b) The total estimation-error ξ^{roc} is the sum of stored errors of all the nodes.

The estimation error of the IncSFA algorithm (ξ^{sfa}) at any time t is computed as:

$$\xi^{\text{sfa}}(t) = ||\widehat{\phi}_t - \widehat{\phi}_{t-1}||, \tag{4.6}$$

where $||.||$ indicates the Euclidean norm and $\widehat{\phi}_t$ denotes the current adaptive IncSFA abstraction.

The estimation error of the ROC algorithm is computed differently than for IncSFA. There are a total of variable $p \in \mathbb{N}$ instantiations of the ROC algorithm, estimating the IncSFA feature output $\mathbf{y}(t)$ to p values. Each ROC algorithm node j has an associated error ξ_j^{roc} (Figure 4.7(a)). These errors are initialized to 0 and then updated whenever the node is activated by

$$\xi_j^{\text{roc}}(t) = \min_w ||\mathbf{y}(t) - \mathbf{v}_w||, \tag{4.7}$$

where $\mathbf{y}(t)$ is the slow feature output vector and $\mathbf{v}_w$ is the estimate of the wth cluster of the activated node. The total ROC estimation error (Figure 4.7(b)) is the sum of stored errors of the nodes:

$$\xi^{\mathrm{roc}}(t) = \sum_{j=1}^{p} \xi_j^{\mathrm{roc}}(t). \tag{4.8}$$

The estimation-error of the IncSFA-ROC abstraction-estimator is the tuple $\xi = (\xi^{\mathrm{sfa}}, \xi^{\mathrm{roc}})$. How this estimation-error tuple is translated to intrinsic rewards is discussed next.

4.3.4 Intrinsic Reward Function Estimate (R^{int})

Section 4.1 presented an overview on the design of intrinsic rewards to learn an observation stream selection policy that minimizes a multi-objective cost function. Here, details on how the reward function is estimated for the IncSFA-ROC algorithm is discussed. The reward function $R^{\mathrm{int}} : \mathscr{S}^{\mathrm{int}} \times \mathscr{A}^{\mathrm{int}} \times \mathscr{S}^{\mathrm{int}} \to \mathbb{R}$ is updated as:

$$\begin{aligned}\widetilde{R}_t(s^{\mathrm{int}}, a^{\mathrm{int}}, s_-^{\mathrm{int}}) &= \alpha \left(Clip\left(\sum_\tau -\dot{\xi}^{\mathrm{sfa}}, -\mathscr{L}, \mathscr{L} \right) + \beta \sum_\tau (Z(|\delta - \xi^{\mathrm{roc}}|) \right) \\ &\quad + (1-\alpha)\widetilde{R}_{t-1}(s^{\mathrm{int}}, a^{\mathrm{int}}, s_-^{\mathrm{int}}) \\ R_t^{\mathrm{int}} &\leftarrow \widetilde{R}_t / \|\widetilde{R}_t\|, \end{aligned} \tag{4.9}$$

where $Clip(x,a,b) = \min(\max(x,a),b)$, $\mathscr{L}$ is a clipping constant, $\dot{\xi}^{sfa}$ is the time-derivative of the estimation-error of IncSFA for the τ input samples $\mathbf{x}$, ξ^{roc} is the estimation-error of ROC estimator based on the IncSFA feature output, Z is a normally distributed random variable with mean equal to the threshold δ and a constant standard-deviation σ, β is a positive constant, α is a constant smoothing coefficient and $\|.\|$ represents the Euclidean norm. $\widetilde{R}_t$ represents a *reinforcement function* that updates at every time t. The derivative of the IncSFA progress is computed via *backward-difference* approximation and is clipped to make it bounded. It represents the curiosity reward term, while the estimation-error term of the ROC-clustering algorithm contributes to the reward based on how well the agent is able to estimate the IncSFA outputs. The intrinsic reward function R_t^{int} is the normalized reinforcement-function. In the next section, I will show how the intrinsic rewards are used in shaping the observation stream selection policy.

4.3.5 Observation Stream Selection Policy (π^{int})

Least squares temporal difference Q-learning (LSTDq) is used to efficiently evaluate the current observation stream selection policy $\pi_t^{\mathrm{int}} : \mathscr{S}^{\mathrm{int}} \to \mathscr{A}^{\mathrm{int}}$ for the next iteration based on simulated $(s^{\text{int}}, a^{\text{int}}, r^{\text{int}})$ tuples sampled from the current environment model $(\mathscr{P}^{\text{int}}, R_t^{\text{int}})$. The Least-Squares fixed-point approximation of the value function for a transition model $\mathscr{P}^{\text{int}}$ and reward function R^{int} is given by:

$$\widehat{Q}^{\pi_t^{\text{int}}} = \Psi\left(\Psi^T(\Psi - \gamma P^{\text{int}}\Pi_t^{\text{int}}\Psi)\right)^{-1}\Psi^T R_t^{\text{int}}, \tag{4.10}$$

where Ψ denotes the basis functions used to represent the $(s^{\text{int}}, a^{\text{int}}, r^{\text{int}})$ tuple, P^{int} is a matrix of size $|\mathscr{S}^{\text{int}}||\mathscr{A}^{\text{int}}| \times |\mathscr{S}^{\text{int}}|$ that contains the transition model of the process $\left(P^{\text{int}}((s^{\text{int}}, a^{\text{int}}), s_{-}^{\text{int}}) = \mathscr{P}(s^{\text{int}}, a^{\text{int}}, s_{-}^{\text{int}})\right)$ and Π_t^{int} is a matrix of size $|\mathscr{S}^{\text{int}}| \times |\mathscr{S}^{\text{int}}||\mathscr{A}^{\text{int}}|$ that describes the current policy π_t^{int}:

$$\Pi_t^{\text{int}}(s', (s', a')) = \pi_t^{\text{int}}(a'; s').$$

The policy π_{t+1}^{int} for the next iteration is computed via *policy-improvment* from the value function as:

$$\pi_{t+1}^{\text{int}}(s) = \underset{a \in \{\text{stay,switch}\}}{\arg\max}\ \widehat{Q}^{\pi_t^{\text{int}}}(s, a), \quad \forall s \in \mathscr{S}^{\text{int}}. \tag{4.11}$$

To balance between exploration and exploitation, a *decaying ϵ-greedy* strategy [Sutton and Barto, 1998] over the observation stream selection policy is used to carry out an action (*stay* or *switch*) for the next iteration. When the ϵ value decays to zero, the agent exploits the policy and an abstraction is learned. The next section discusses the gating system that prevents the agent from learning the same abstraction.

4.3.6 Gating Function ($\mathscr{G}$)

The *gating function* $\mathscr{G} : X \to \Phi_t \cup \widehat{\phi}$ assigns the appropriate abstraction $\phi_i \in \Phi_t$ to the incoming observation stream if Constraint (4.4) is satisfied with respect to the ROC clustering algorithm, and the adaptive abstraction module $\widehat{\phi}$ will be prevented from learning via a "gating signal" (see Figure 4.8). Inputs badly encoded by all $\phi \in \Phi_t$ serve to train the adaptive module $\widehat{\phi}$. Hence the adaptive module will encode only data from observation streams that were not encoded earlier. The gating function of Curious Dr. MISFA is:

$$\mathscr{G}(\mathbf{x}) = \begin{cases} \phi_l, & \text{if } \min_i\{|\xi_i^{roc}|\} < \delta,\ l = \arg\min_i\{|\xi_i^{roc}|\}, \forall i \in \{1, ..., \#\Phi_t\} \\ \widehat{\phi}, & \text{otherwise} \end{cases} \tag{4.12}$$

where $\#$ denotes the cardinality of a set, $|.|$ represents the absolute value and ξ_i^{roc} is the ROC estimation error for the input $\mathbf{x}$, corresponding to the i^{th} abstraction-module $\phi_i \in \Phi_t$, $\forall i \in \{1, ..., \#\Phi_t\}$.

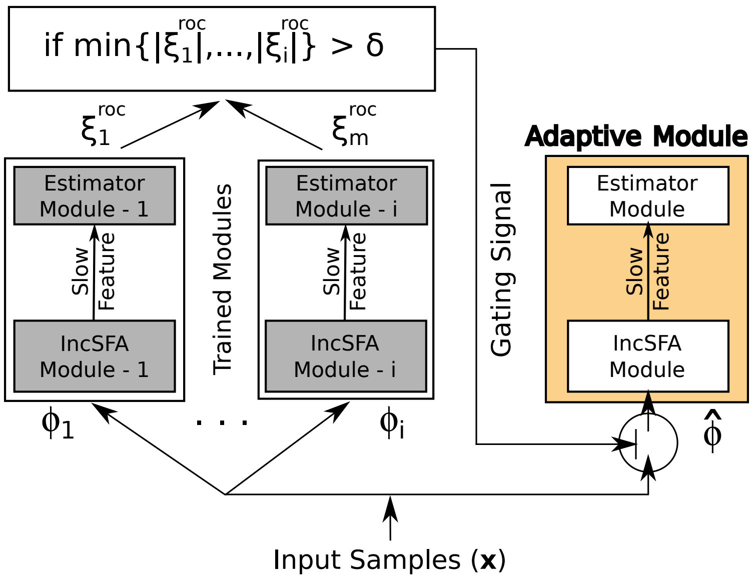


Figure 4.8. Block diagram of the Gating System. See text for details.

4.3.7 Dynamical Analysis

In this section, I will present a formal analysis of the dynamical behavior of Curious Dr. MISFA algorithm. Proofs of all theorems introduced here are included in Appendix A.

Outline: The existence of a curiosity-function Ω that indicates the speed-of-learning of an abstraction-estimator is first shown in Theorem 1. The curiosity-function of IncSFA is then defined (Definition 1). Based on Ω, *optimal fixed-points* for the adaptive abstraction $\widehat{\phi}$ and the observation stream selection policy π^{int} are shown in Theorems 2 & 3. Finally, for a set of mild conditions, the convergence of the algorithm's policy π^{int} and the adaptive abstraction $\widehat{\phi}$ to their respective optimal fixed-points is shown in Theorems 4 & 5. The following analysis is for $n > 2$, i.e., there are more than two observation streams. The specific case of $n \leq 2$ is discussed later in the section.

Convergence Conditions: I assume that the following conditions are satisfied for the rest of the analysis: Let $t_0 \in \mathbb{N}$ denote the time whenever a new adaptive abstraction $\widehat{\phi}$ is instantiated,

$$\eta^{mca} \max(\lambda_1^1, ..., \lambda_1^n) < 0.5, \quad 0 < \eta^{mca} \leq 0.5, \tag{4.13}$$

$$||\widehat{\phi}_i(t_0)||^2 = 1, \quad \widehat{\phi}_i(t_0)^T \overline{\phi}_i \neq 0, \ \forall i \in \{1, ..., J\}, \tag{4.14}$$

$$\langle ||\Theta(\mathbf{x_i}, \widehat{\phi}) - \widehat{\phi}|| \rangle_{\mathbf{t}}^{\tau} > \delta, \ \forall \mathbf{i} \in \{\mathbf{1}, ..., \mathbf{n}\}, \quad \tau < \min(\overline{\tau}_{\mathbf{1}}^{\mathbf{1/2}}, ..., \overline{\tau}_{\mathbf{n}}^{\mathbf{1/2}}), \tag{4.15}$$

$$\frac{ln(\Omega(\mathbf{x}_i))}{ln(\Omega(\mathbf{x}_j))} < \frac{\gamma}{(n-1-\gamma(n-2))}, \ \forall \mathbf{x}_i, \mathbf{x}_j \in X \ \ s.t. \ \Omega(\mathbf{x}_i) > \Omega(\mathbf{x}_j), \tag{4.16}$$

where λ_1^i is the largest eigenvalue of expected correlation-matrix $E[\dot{\mathbf{z}}_i \dot{\mathbf{z}}_i^T]$, $\dot{\mathbf{z}}_i$ is the derivative of the whitened output of $\mathbf{x}_i$. $\widehat{\phi}_i(t_0)$ is the i^{th} column of the abstraction $\widehat{\phi}(t_0)$ at time $t = t_0$, $\overline{\phi}_i$ is the i^{th} minor-component of $E[\dot{\mathbf{z}}\dot{\mathbf{z}}^T]$, $\overline{\tau}_i^{1/2} = \frac{ln(1/2)}{ln(\Omega(\mathbf{x}_i))}$ and γ is a constant discount-factor ($0 < \gamma < 1$) for RL.

Conditions (4.13) & (4.14) are required for the convergence of the IncSFA algorithm for a stationary input-distribution. Condition (4.15) determines the range of values for time-constant τ and δ. δ is usually selected to be close to 0. Finally, Condition (4.16) is required for optimal ordering of the abstractions learned. If the condition does not hold for a few observation streams, the result is a *sub-optimal* solution (Theorem 6). There are a few other minor conditions, which will be covered in the proofs for the theorems.

Let $\mathscr{X} = \{\mathbf{x} : \mathbf{x}(t) \in \mathbb{R}^I, I \in \mathbb{N}\}$ denote a set of of I-dimensional observation streams. Let $X \subset \mathscr{X}$ be a finite subset with $n \in \mathbb{N}$ elements that may or may not be unique. Let Θ denote an abstraction-estimator that updates a real-valued abstraction and ensures an *almost sure* convergence for an I-dimensional stationary input signal. Let Φ^* denote the space of all learnable abstractions by Θ for the input X satisfying Constraints (4.3)-(4.4) (see Section 4.2). Let $\widehat{\phi} \in \mathbb{R}^{I \times J}, J \in \mathbb{N}, \widehat{\phi} \notin \Phi^*$ denote the current adaptive abstraction.

Theorem 1. *There exists a curiosity-function* $\Omega : \mathscr{X} \rightarrow [0, 1)$ *corresponding to* Θ *that induces a total ordering on* $\mathscr{X}$*.*

Definition 1. *The curiosity-function of the IncSFA algorithm for an observation stream* $\mathbf{x}$ *is defined as*

$$\Omega(\mathbf{x}) = \left[1 - \frac{\eta^{mca}(\lambda_{K-1} - \lambda_K)}{1 - \eta^{mca} - \eta^{mca}\lambda_K} \right], \tag{4.17}$$

where $\lambda_K \neq \lambda_{K-1}$ *denote the smallest two eigenvalues of* $E[\dot{\mathbf{z}}(t)\dot{\mathbf{z}}(t)^T]$*, and* $\mathbf{z}(t) \in \mathbb{R}^K$ *is the* whitened *output of* $\mathbf{x}(t)$*.*

Note that the curiosity function is used only to evaluate whether the abstractions learned by the algorithm correspond to the observation streams with an increasing order of Ω value. The Curious Dr. MISFA algorithm however, does not use the curiosity function for its computation.

Definition 2. *At time t, let $\mathbf{x}_l$ denote the current easiest but not yet learned observation stream and s_l denote the corresponding internal state. Then, the index l is given by*

$$l = \underset{\forall i:\ \mathbf{x}_i \in X'}{\arg\min}\, \Omega(\mathbf{x}_i), \quad X' = \{\mathbf{x}_i : \mathscr{G}(\mathbf{x}_i) = \widehat{\phi}, \mathbf{x}_i \in X\}. \tag{4.18}$$

Theorem 2. *At time t, the optimal fixed-point $\phi^* \in \Phi^*$ of the adaptive abstraction $\widehat{\phi}$ is equal to the J slow features of the observation stream $\mathbf{x}_l$.*

Theorem 3. *The optimal observation stream selection policy ($\pi^* : \Phi^* \times \mathscr{S}^{int} \to \mathscr{A}^{int}, \mathscr{A}^{int} = \{0\ (stay), 1\ (switch)\}$) to learn an abstraction $\phi_i \in \Phi^*$ is given by:*

$$\pi^*(\phi_i, s) = 1 - \mathbb{1}_{\{s_l\}}(s),\ \forall s \in \mathscr{S}^{int},$$

Theorem 4. *Let $\{\pi_t^{int}\}_{t\in\mathbb{N}}$ denote the sequence of observation stream selection policies generated by the algorithm for $\epsilon = 1$. If Conditions (4.13),(4.14),(4.15) and (4.16) hold, then for $t > t_0$,*

$$\lim_{t\to\infty} \pi_t^{int}(s) = \pi^*(\phi^*, s),\ \forall s \in \mathscr{S}^{int}$$

Theorem 4 shows that during exploration ($\epsilon = 1$), the π_t^{int} converges to a policy with an action *stay* ($= 0$) for the state (s_l) corresponding to the current *easiest but not yet encoded* observation stream ($\mathbf{x}_l$), and the action *switch* ($= 1$) for rest of the states. Also, since the policies π_t^{int} and $\pi^*(\phi^*)$ are binary-vectors, it follows that $\exists t_c \in \mathbb{N}\ (t_0 < t_c < \infty)$, s.t. for $t = t_c$, $\pi_t^{\text{int}} = \pi^*(\phi^*)$.

Theorem 5. *Let $\{\widehat{\phi}_t\}_{t\in\mathbb{N}}$ denote the sequence of adaptive abstractions generated by the algorithm for $\epsilon = 0$. If $t_c(> t_0) \in \mathbb{N}$ is the time when $\pi_t^{int} = \pi^*(\phi^*)$ and if Conditions (4.13),(4.14),(4.15) and (4.16) hold, then for $t > t_c$,*

$$\lim_{t\to\infty} \widehat{\phi}_t = \phi^*$$

When the agent *exploits* the converged policy π^{int} (Theorem 4), its internal state shifts to s_l where it continually receives input-observations $\mathbf{x}_l(t)$. As a result, the adaptive abstraction converges to the optimal abstraction ϕ^* (Theorem 3).

When, the window-averaged ROC estimation error drops below the threshold δ ($\langle \|\Theta(\mathbf{x}_l, \widehat{\phi}) - \widehat{\phi}\| \rangle_t^\tau < \delta$), the adaptive abstraction $\widehat{\phi}$ is frozen and saved to the abstraction set Φ_t ($\Phi_t \leftarrow \Phi_t \cup \widehat{\phi}$). Theorems 2-5 show that the saved abstraction satisfies Constraints (4.3)-(4.4) and the cardinality of the abstraction set increments by a value 1. The process repeats until all the abstractions have been learned.

Theorem 5 requires t_c, the time at which the policy π^{int} has converged. However, in practice t_c is not known *a priori* and needs to be estimated online. In large state spaces estimating t_c can be difficult. As an alternative, I use a heuristic ϵ-greedy strategy (see Section 4.3.5) in the algorithm. By selecting an appropriate decay constant we can get the desired result.

Case $n \leq 2$: For $n = 2$, *i.e.* two states $\mathcal{S}^{\text{int}} = \{s_1^{\text{int}}, s_2^{\text{int}}\}$, the stochastic *switch* action is equivalent to a deterministic *switch* to the other state. Therefore, Curious Dr. MISFA can learn an abstraction by switching between the states (*i.e.*, only if both the stimulus streams are individually encodable). If $\Omega(\mathbf{x}_{\text{mix}}) < \min(\Omega(\mathbf{x}_1), \Omega(\mathbf{x}_2))$, where $\mathbf{x}_{\text{mix}}$ denotes the mixture signal, then the algorithm will learn a policy that switches the agent's internal state to the other ($\pi^{\text{int}} = [1, 1]$). This results in an abstraction corresponding to the mixture signal. In this special case, the algorithm will learn a total of 3 abstractions. For $n = 1$, the solution is trivial, the algorithm learns an abstraction corresponding to the observation stream (if it is encodable), irrespective of the observation stream selection policy.

Sub-Optimality: Convergence Conditions (4.13), (4.14) and (4.15) can easily be met by setting the algorithm-parameters appropriately. However, Condition (4.16) does not involve setting any algorithm parameter and is a direct condition on the observation stream complexity. I will discuss here a scenario when a few of the observation streams violate the condition. Figure 4.9(a) shows a plot of the ratio

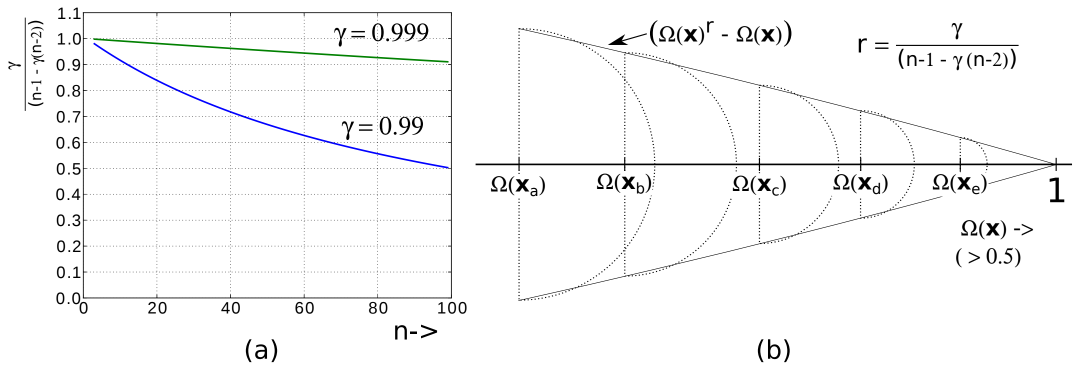


Figure 4.9. (a) A plot of the ratio $r = \dfrac{\gamma}{(n - 1 - \gamma(n-2))}$ over the number of observation streams n for two values of $\gamma = 0.99$ & 0.999. (b) An example illustration, where observation stream $\mathbf{x}_b$ violates the Condition (4.16), resulting in a sub-optimal performance of the algorithm. Figure also shows that for higher values of Ω, the required separation distance between the Ω values of observation streams decreases.

$r = \dfrac{\gamma}{(n-1-\gamma(n-2))}$ over the number of observation streams n for two values of $\gamma = 0.99$ & 0.999. It is clear from the figure that r decreases as n increases. Substituting for r in the Condition (4.16) we get $\Omega(\mathbf{x}_i) > \Omega(\mathbf{x}_j)^r$, $\forall \Omega(\mathbf{x}_i) > \Omega(\mathbf{x}_j)$. For observation streams with a higher Ω values, the condition $\left(\Omega(\mathbf{x}_i) > \Omega(\mathbf{x}_j)^r\right) \rightarrow \left(\Omega(\mathbf{x}_i) > \Omega(\mathbf{x}_j)\right)$, therefore the condition is most-likely to be met. Figure 4.9(b) illustrates this with an example.[4] The separation distance required between $\Omega(\mathbf{x}_a)$ and $\Omega(\mathbf{x}_b)$ is larger than $\Omega(\mathbf{x}_c)$ and $\Omega(\mathbf{x}_d)$. In this case, the stream $\mathbf{x}_b$ violates the condition. But, how does this affect the working of the algorithm? Theorem 6 shows that this results in a sub-optimal performance by the algorithm.

Definition 3. *Let* $r = \dfrac{\gamma}{(n-1-\gamma(n-2))}$. *A stream* $\mathbf{x}$ *is r-dominated by another stream* $\mathbf{x}'$ *if* $ln\left(\Omega(\mathbf{x})\right) < r\ ln\left(\Omega(\mathbf{x}')\right)$.

Theorem 6. *Let* $\{\pi_t^{int}\}_{t\in\mathbb{N}}$ *denote the sequence of observation stream selection policies generated by the algorithm for* $\epsilon = 1$. *Let* $\mathcal{S}_r$ *be the set of internal states whose observation streams are not r-dominated by* $\mathbf{x}_l$. *If Conditions (4.13),(4.14) and (4.15) hold, then, for* $t > t_0$, $\pi_t^{int}(s)$ *has two limits points equal to* $\left(1 - \mathbb{1}_{\{s_l\}}(s)\right)$ *or* $\left(1 - \mathbb{1}_{\mathcal{S}_r}(s)\right)$, $\forall s \in \mathcal{S}^{int}$.

Theorem 6 shows that if a few observation streams violate the Condition (4.16), then π^{int} is not guaranteed to converge to the optimal policy (Theorem 3). It may instead converge to a sub-optimal policy, which returns an action *stay* in all the states whose observation streams violate the Condition (4.16) and *switch* in all the remaining states. This differs from the optimal policy, where the action *stay* is returned for the internal state s_l corresponding to the easiest but not yet learned observation stream and *switch* in all the other states. The suboptimal policy during exploitation makes the algorithm converge to an abstraction encoding any of the observation streams corresponding to the states in $\mathcal{S}_r$, with an uniform probability. For the example illustrated in Figure 4.9(b), the algorithm generates the following sequence of abstractions with equal probability

1. $\Phi = \{\phi_1^{\mathbf{x}_a}, \phi_2^{\mathbf{x}_b}, \phi_3^{\mathbf{x}_c}, \phi_4^{\mathbf{x}_d}, \phi_5^{\mathbf{x}_e}\}$ (optimal),
2. $\Phi = \{\phi_1^{\mathbf{x}_b}, \phi_2^{\mathbf{x}_a}, \phi_3^{\mathbf{x}_c}, \phi_4^{\mathbf{x}_d}, \phi_5^{\mathbf{x}_e}\}$,

where $\phi_1^{\mathbf{x}_a}$ denotes the abstraction learned corresponding to the stream $\mathbf{x}_a$. These two sequences differ only in the first two terms. Therefore, for half the number of trials the algorithm converges to the optimal sequence (first sequence) and the rest to the sub-optimal sequence (second sequence) resulting in a sub-optimal behavior.

[4]The values of Ω are generally > 0.5

Algorithm 7: Curious Dr. MISFA (X)

```
   //$\Theta$ :  IncSFA-ROC Estimator
   //$\Phi$ :  IncSFA Abstraction set
   //$\widehat{\phi}$ :  Current adaptive IncSFA abstraction
   //$\mathscr{G}$ :  Gating Function
   //$\pi^{\mathrm{int}}$ :  Input-stream selection policy
   //$R^{\mathrm{int}}$ :  Reward function
   //$Z$ :  Normally distributed random variable with
     mean $\delta$ and std. deviation $\sigma$
   //$\tau, \alpha, \beta, \delta, \sigma, \mathscr{L}$ :  Scalar constants
1  $\Phi_0 \leftarrow \{\}$, $\pi_0^{\mathrm{int}} \leftarrow$ Random (), $\widehat{\phi} \leftarrow 0, \widetilde{R} \leftarrow 0$
2  for $t \leftarrow 0$ to $\infty$ do
       //Sense
3      $s_t^{\mathrm{int}} \leftarrow$ current internal state
4      $a_t^{\mathrm{int}} \leftarrow \epsilon$-greedy($\pi_t^{\mathrm{int}}(s_t^{\mathrm{int}})$)
5      Take action $a_t^{\mathrm{int}}$, observe next internal state $s_{t+1}^{\mathrm{int}} (= \mathscr{P}^{\mathrm{int}}(s_t^{\mathrm{int}}, a_t^{\mathrm{int}}))$ and $\tau$
       input-samples $\mathbf{x}$
       //Compute IncSFA-ROC error vector
6      $\{\xi_{t+1}^{\mathrm{sfa}}, \xi_{t+1}^{\mathrm{roc}}\} = ||\Theta(\mathbf{x}, \mathscr{G}(\mathbf{x})) - \mathscr{G}(\mathbf{x})||$
       //Compute the derivative of IncSFA-error
7      $\dot{\xi}_{t+1}^{\mathrm{sfa}} \leftarrow$ Backward-Difference($\xi_{t+1}^{\mathrm{sfa}}$)
       //Update the adaptive-abstraction
8      if $\langle \xi_{t+1}^{roc} \rangle_\tau > \delta$ then
9          $\widehat{\phi} \leftarrow \Theta(\mathbf{x}, \widehat{\phi})$
10     end
       //Update the reward function
11     $\widetilde{R}(s_t^{\mathrm{int}}, a_t^{\mathrm{int}}, s_{t+1}^{\mathrm{int}}) =$
       $\alpha \left( \sum_\tau -\dot{\xi}^{\mathrm{sfa}}|_{-\mathscr{L}}^{\mathscr{L}} + \beta \sum_\tau (Z(|\delta - \xi^{\mathrm{roc}}|) \right) \; + \; (1-\alpha)\widetilde{R}(s_t^{\mathrm{int}}, a_t^{\mathrm{int}}, s_{t+1}^{\mathrm{int}})$
12     $R_{t+1}^{\mathrm{int}} \leftarrow \widetilde{R}/||\widetilde{R}||$
       //Update observation stream selection policy
13     $\pi_{t+1}^{\mathrm{int}} \leftarrow$ Model-LSPI ($\mathscr{S}^{\mathrm{int}}, \mathscr{A}^{\mathrm{int}}, \mathscr{P}^{\mathrm{int}}, R_{t+1}^{\mathrm{int}}$)
       //Update abstraction set
14     if $\langle ||\Theta(\mathbf{x}, \widehat{\phi}) - \widehat{\phi}|| \rangle_\tau < \delta$ then
15         $\Phi_{t+1} \leftarrow \Phi_t \cup \widehat{\phi}$
16         $\pi_{t+1}^{\mathrm{int}} \leftarrow$ Random (), $\widehat{\phi} \leftarrow 0, \widetilde{R} \leftarrow 0$.
17     end
18 end
```

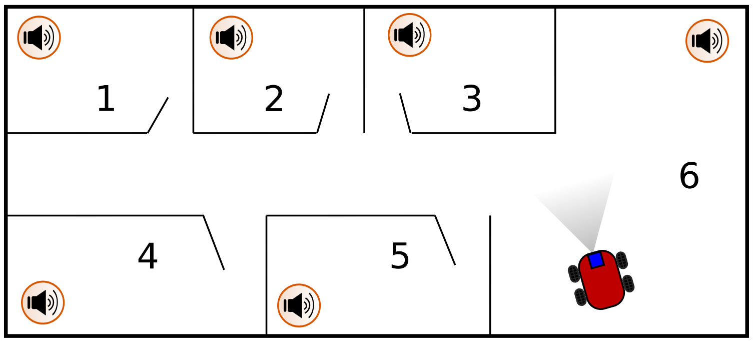


Figure 4.10. An example room maze scenario.

4.4 Pseudocode

I showed how Curious Dr. MISFA solves the curiosity-based abstraction learning problem presented in Section 4.2 under certain conditions. Algorithm 7 summarizes Curious Dr. MISFA. A *Python*-based implementation of the algorithm can be found at the URL: www.idsia.ch/~kompella/codes/cdmisfa.html. The method as presented above can be used in several online learning applications and is especially suited for acquisition of abstractions and skills on humanoid platforms. The next section discusses a few environments where the method requires some modifications.

4.5 Design Considerations: Maze Environments

This section explores the application of Curious Dr. MISFA to environments such as a room-maze where each room has a time-varying audio or a video source (Figure 4.10). Therefore, each room represents an internal state of Curious Dr. MISFA. In such cases, the internal environment's transition dynamics (see Section 4.3.1) are not similar to that of a complete-graph model, i.e., the agent cannot *switch* between all the rooms without passing through the other rooms. I will present here design modifications to the algorithm that enable the agent to take deterministic actions (unlike the stochastic *switch* action) to move to the internal state with the easiest encodable observation stream, in order to learn an abstraction. It is hard to provide theoretical guarantees to these modifications because they depend on the unknown transition model of the maze. Instead, experimental results are presented in Section 4.6 to demonstrate the algorithm's performance in such domains. The following are the design modifications of the algorithm for maze environments:

Internal Environment's Transition Model: Curious Dr. MISFA uses a model-

Algorithm 8: Observation Stream Selection

```
//$\pi_t^{\text{int}}$: Current stay-switch policy
//$\pi_t^{\text{dint}}$: Current deterministic policy
//$\mathscr{P}^{\text{int}}$: Transition model for action-space $\mathscr{A}^{\text{int}}$
//$\mathscr{P}^{\text{dint}}$: Transition model for action-space $\mathscr{A}^{\text{dint}}$
//$U(a,b)$: Uniform random variable in (a,b)
//$U^i(a,b)$: Uniform random integer in [a,b]
//$s_t^{\text{int}}$: Current internal environment state
//$\epsilon, \nu$: Scalar variables
1 if $U(0,1) < \epsilon$ then
2     $s_{t+1}^{\text{int}} = \mathscr{P}^{\text{int}}(s_t^{\text{int}}, U^i(0,1))$
3 else
4     if $U(0,1) < \nu$ then
5         $s_{t+1}^{\text{int}} = \mathscr{P}^{\text{int}}(s_t^{\text{int}}, \pi_t^{\text{int}}(s_t^{\text{int}}))$
6     else
7         $s_{t+1}^{\text{int}} = \mathscr{P}^{\text{dint}}(s_t^{\text{int}}, \pi_t^{\text{dint}}(s_t^{\text{int}}))$
8     end
9 end
```

based least-squares temporal difference learning to learn the observation stream selection policy (see Section 4.3.5). Therefore the transition model for general maze environments needs to be learned *a priori*. It can be learned either using lookup tables in deterministic environments or using Bayesian inference in stochastic environments.

Observation Stream Selection Policy: A drawback of using an ϵ-greedy strategy over the *stay-switch* policy is that for a large number of observation streams in maze environments, it takes a considerable amount of time to get to a desired internal state. This can be improved by simultaneously learning another policy π^{dint} defined over the same internal state space $\mathscr{S}^{\text{int}}$ but with a deterministic action-space ($\mathscr{A}^{\text{dint}} = \{a_1^{\text{dint}}, ..., a_n^{\text{dint}}\}$). Let $\mathscr{P}^{\text{dint}}$ denote the transition model of the internal environment for the action-space $\mathscr{A}^{\text{dint}}$. When the agent shifts to a state s_i^{int}, $\forall i \in \{1, ..., n\}$, this implies that the agent took an action a_i^{dint}, and vice-versa. The *switch* action stochastically selects an observation stream, while the action $a_i^{\text{dint}} \in \mathscr{A}^{\text{dint}}$ deterministically selects the observation stream $\mathbf{x}_i$. The agent therefore maintains a pair of value functions, one for the *stay-switch* action space (Eq.

4.10) and the other for the deterministic action-space ($\mathscr{A}^{\text{dint}}$):

$$\widehat{Q}^{\pi_t^{\text{dint}}} = \Psi_d \left(\Psi_d^T(\Psi_d - \gamma_d P^{\text{dint}}\Pi_t^{\text{dint}}\Psi_d)\right)^{-1} \Psi_d^T R_t^{\text{dint}} \tag{4.19}$$

$$\pi_{t+1}^{\text{dint}}(s) = \arg\max_{a^{\text{dint}}} \widehat{Q}^{\pi_t^{\text{dint}}}(s,a),\ s \in \mathscr{S}^{\text{int}},\ a \in \mathscr{A}^{\text{dint}} \tag{4.20}$$

where Ψ_d denote the basis functions to represent the $(s^{\text{int}}, a^{\text{dint}}, r^{\text{dint}})$ tuple, P^{dint} denotes the transition-matrix for the action-space $\mathscr{A}^{\text{dint}}$ and Π_t^{dint} describes the current deterministic observation stream selection policy π_t^{dint}.

The agent then chooses between the the policies π^{int} and π^{dint} probabilistically to get to the desired states quickly. The observation stream selection by the agent at any time t is summarized in Algorithm 8.

Reward Function: For maze environments, the desired target internal state may not be reachable in few time-steps. In such cases the discount-factor γ may lead to suboptimal behavior [Mahadevan, 1996]. To minimise this, average-reward reinforcement learning approaches [Mahadevan, 1996] can be used. I use a simple trick by reducing the reinforcement function to a binary state-reward function as:

$$R_t^{\text{int}}(s) = \mathbb{1}_{\{s_l\}}(s),\ s_l = \arg\max_{s_i} \left(\widetilde{R}_t(s_i, stay, .)\right),\ s \in \mathscr{S}^{\text{int}} \tag{4.21}$$

Therefore, at any time t, the observation stream selection policy learned using this reward function is a shortest path to get to the internal state corresponding to the maximum reward.

Basis Functions: I use *linear function approximation* methods to find approximate value functions for large discrete environments. The value function is represented as a linear combination of basis functions. The selection of basis functions plays an important role in solving the problem. *Krylov Basis Functions* [KBFs; Petrik, 2007] are *reward-sensitive* basis-vectors ($\mathscr{K}$), which are constructed by taking the product of reward function (R) with geometric powers of transition matrix (P) of a policy:

$$\mathscr{K} = \{R, PR, P^2R, ...\}. \tag{4.22}$$

Proto-Value Basis Functions [PVFs; Mahadevan and Maggioni, 2007] are however *reward-insensitive* basis-vectors, which are constructed by finding the eigenvectors of the symmetric graph Laplacian matrix based on the neighborhood relationsips among the states. PVFs capture invariant subspaces (*bottlenecks*) of the model transition matrix. However, they lead to poor approximations when the reward function is spiky, because the basis vectors are smooth. While KBFs tend to work well for spiky reward functions, they require costly recomputations of the basis functions whenever the reward function changes. *Augmented Krylov Basis Functions* [AKBFs; Petrik, 2007] combines the methods to take advantage of both their approximation properties. This basis is constructed by augmenting a finite number of Krylov-basis and proto-value basis vectors, followed by an iterative orthogonalization using *Arnoldi iteration technique* [Arnoldi, 1951]. I use AKBFs (Ψ) for evaluating the *stay-switch* observation stream selection policy π_t^{int} and the deterministic observation stream selection policy π_t^{dint}.

These design modifications together enable Curious Dr. MISFA to be applied to maze environments with time-varying observation streams. The next section presents results of experiments conducted in such environments.

4.6 Experimental Results

I present here experimental results to illuminate the algorithm's performance in different environments. These results show that the algorithm learns slow feature abstractions in the order of increasing learning difficulty, as predicted by the theoretical analysis presented in Section 4.3.7.

4.6.1 Proof of Concept: Synthetic Signals

In this experiment, the convergence of the algorithm is illustrated for an input that consists of three 2D nonlinear oscillatory audio streams $X = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$, each encodable by IncSFA:

$$\mathbf{x}_1 \; : \; \begin{cases} x_1(t) = \sin(4\,t - \pi/4.) - \cos(44\,t)^2 \\ x_2(t) = \cos(44\,t) \end{cases}, \tag{4.23}$$

$$\mathbf{x}_2 \; : \; \begin{cases} x_1(t) = \sin(3\,t) + \cos(27\,t)^2 \\ x_2(t) = \cos(27\,t) \end{cases}, \text{ and} \tag{4.24}$$

$$\mathbf{x}_3 \; : \; \begin{cases} x_1(t) = \cos(12\,t) \\ x_2(t) = \cos(2\,t) + \cos(12\,t)^2 \end{cases}, \tag{4.25}$$

where t takes 500 discrete values in $[0, 2\pi]$. The environment has three internal states $\mathscr{S}^{\mathrm{int}} = \{s_1^{\mathrm{int}}, s_2^{\mathrm{int}}, s_3^{\mathrm{int}}\}$ associated with the observation streams. Each observation stream has 500 abstracted states $\mathscr{S}^{\Phi} = \{s_1^{\Phi}, ..., s_{500}^{\Phi}\}$, where s_i^{Φ} represents the angle $2\pi * 500/i$. Figure 4.11(a) illustrates the environment. The slowest feature in the stream $\mathbf{x}_1$ is $\mathbf{y}_1(t) = x_1(t) + x_2(t)^2 = \sin(4\,t - \pi/4.)$. For the streams $\mathbf{x}_2$ and $\mathbf{x}_3$, the features are $\mathbf{y}_2(t) = x_1(t) - x_2(t)^2 = \sin(3\,t)$ and $\mathbf{y}_3(t) = -x_1(t)^2 + x_2(t)^2 = \cos(2\,t)$ respectively. To extract these slow features, each observation stream is expanded via a polynomial expansion of degree 2 to a **5 dimensional stream** [Wiskott and Sejnowski, 2002]. The expanded streams have the following curiosity function values: $\Omega_1 = 0.98979$, $\Omega_2 = 0.99611$, and $\Omega_3 = 0.99924$. Therefore, observation stream $\mathbf{x}_1$ is the easiest signal to encode followed by $\mathbf{x}_2$ and then $\mathbf{x}_3$.

Experiment parameters: I use a fixed parameter setting for the entire experiment. CCIPCA learning rate is equal to $1/t$ with amnesic parameter set to 0.4, while the MCA learning rate is 0.05. The output dimension is set to 1, so only the first IncSFA feature is used as an abstraction. However, more number of features can be used if desired. There are a total of 500 ROC clustering nodes estimating the slow feature output for each of the 500 abstracted states (see Section 4.3.2). Each clustering implementation has its maximum number of clusters set to $N^{max} = 2$, such that it can encode multiple slow feature values for each abstracted state. Higher values can be used, however, very high values may lead to spurious clusters. The estimation error threshold, below which the current module is saved and a new module is created, is set to a low value $\delta = 0.3$. The amnesic parameter is set to $\nu^{\text{amn}} = 0.2$. The initial ϵ-greedy value is set to $\epsilon = 1.1$, with a 0.998 decay multiplier. However, when $\epsilon < 0.9$, the decay multiplier is set to 0.992 to speed up the experiment. I use indicator basis functions (Ψ) for LSPI. The discount factor γ is set equal to 0.99. The window-averaging time constant is set to $\tau = 100$, that is, 100 sample observations are used to compute the window-averaged progress error ξ and the corresponding curiosity reward. The values for σ, β, $\mathscr{L}$, α are set to 50, 1, 200, 0.0198, respectively to satisfy the convergence Conditions (4.13)-(4.16). Refer to the proof of Theorem 5 for more details on setting these parameters.

The dynamics of the algorithm can be observed by studying the time varying reward function R^{int} and the ROC estimation error ξ^{roc}. Figure 4.11(b) shows the reward function for a single run of the experiment. Solid lines represent the reward for the action *stay* in each state s_i^{int} ($R^{\text{int}}(s_i^{\text{int}}, \mathit{stay}, s_i^{\text{int}})$), while the dotted lines represent the *marginalized* reward for the action *switch* at each state s_i^{int}, ($\frac{1}{2}\sum_j R^{\text{int}}(s_i^{\text{int}}$, $\mathit{switch}, s_j^{\text{int}}$)). For the sake of explanation, the learning process can be thought of as passing through three phases, where each phase corresponds to learning a single abstraction module.

Phase 1: At the beginning of Phase 1, the agent starts exploring by executing either *stay* or *switch* at each state. After a few hundred algorithm iterations, the reward function begins to stabilize and is such that $R^{\text{int}}(s_1^{\text{int}}, \mathit{stay}) > R^{\text{int}}(s_2^{\text{int}}, \mathit{stay}) > R^{\text{int}}(s_3^{\text{int}}$, $\mathit{stay}) > 0$, ordered according to the learning difficulty of the observation streams. However, the reward components for the *switch* action are either close to zero or negative. Therefore, the policy π^{int} converges to the optimal policy (*i.e.* to *stay* at

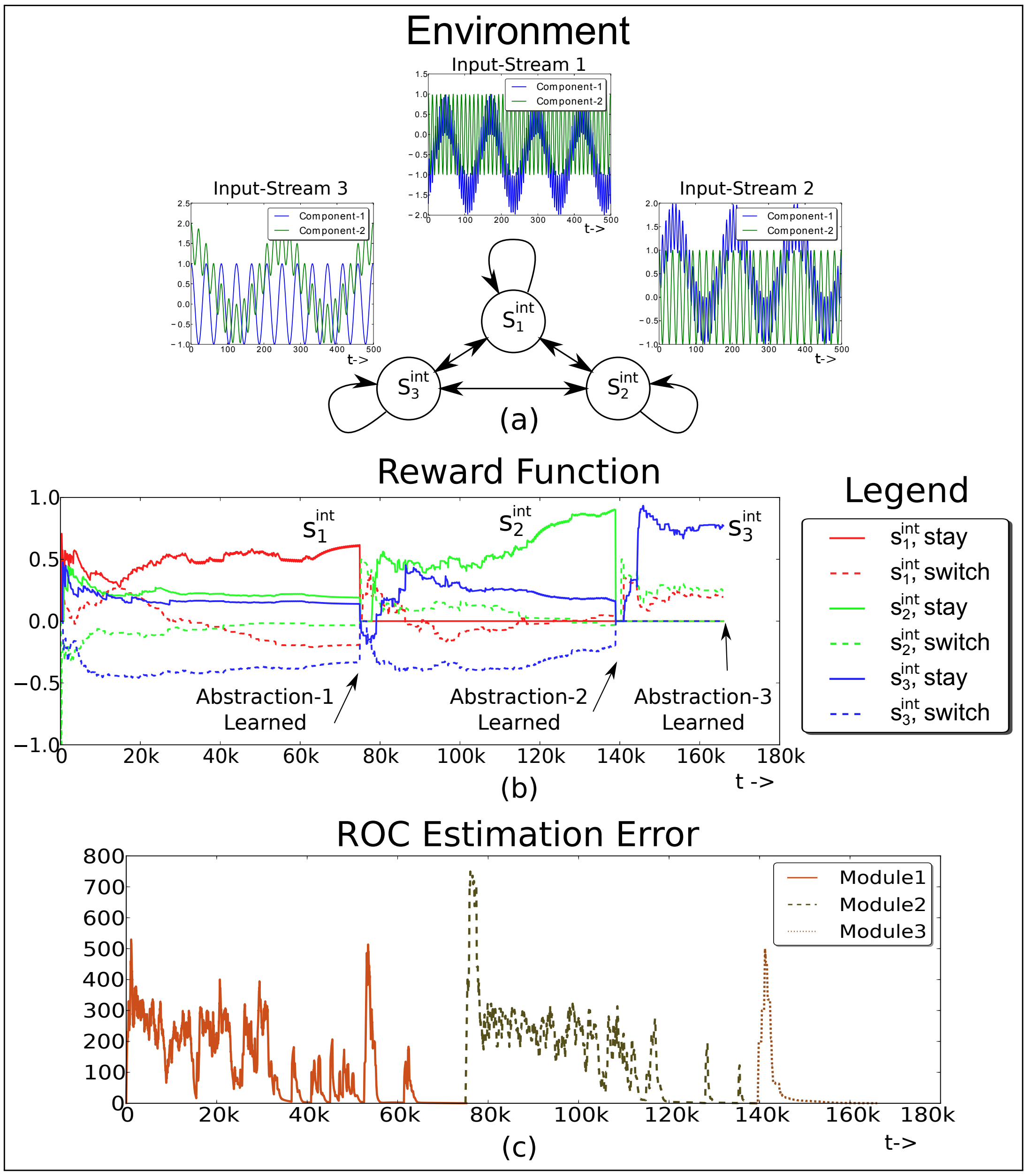


Figure 4.11. **Synthetic Signals**: See text for details. (Figures are best viewed in color)

the state corresponding to the easiest observation stream $\mathbf{x}_1$ and *switch* at every other state). As ϵ decays, the agent begins to exploit the learned policy, and the adaptive IncSFA-ROC abstraction $\widehat{\phi}$ converges to ϕ^* (slow feature corresponding to the observation stream $\mathbf{x}_1$). The ROC estimation error (Figure 4.11(c)) decreases and falls below the threshold δ, at which point, the abstraction is added to the abstraction

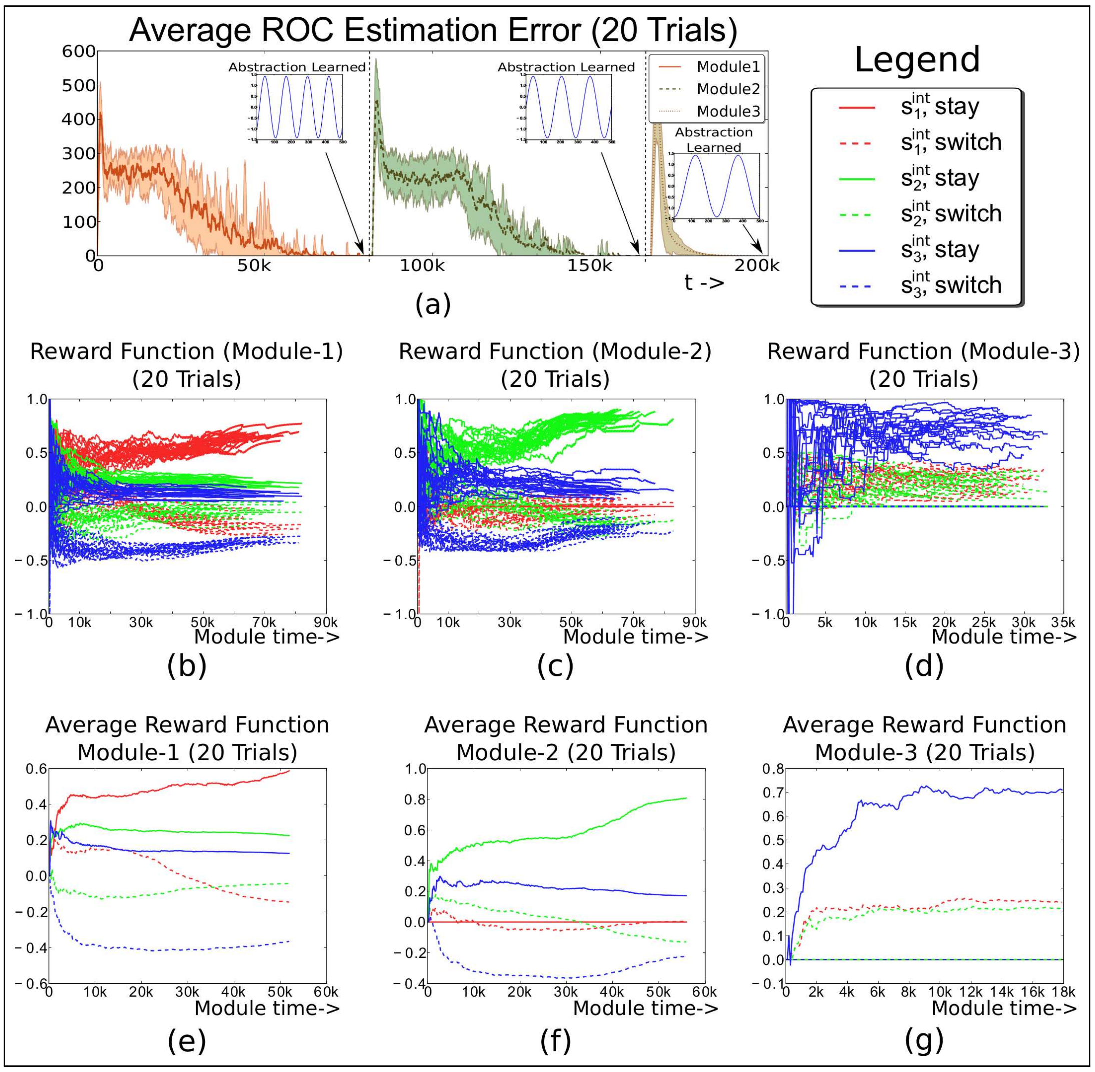


Figure 4.12. Results of the experiment conducted over 20 Trials. See text for details. (Figures are best viewed in color)

set Φ. The increase in the reward value of $R^{\text{int}}(s_1^{\text{int}}, stay)$ near the end of the phase is caused by the second term $(\sum_\tau (Z(|\delta - \xi^{roc}|))$ in Eq. (4.9). Both ϵ and R^{int} are reset and the algorithm enters Phase 2 at $(t \approx 75k)$.

Phase 2: The agent begins to explore again, however, it does not receive any reward for the $(s_1^{\text{int}}, stay)$ tuple because of the gating system. After a few hundred algorithm iterations, $R^{\text{int}}(s_2^{\text{int}}, stay) > R^{\text{int}}(s_3^{\text{int}}, stay) > R^{\text{int}}(s_1^{\text{int}}, stay) = 0$, the adaptive abstraction converges, but to the slow feature corresponding to the observation stream $\mathbf{x}_2$.

Phase 3: The process continues again until the third abstraction is learned.

Figure 4.12 shows results of the experiment conducted for 20 trials with different

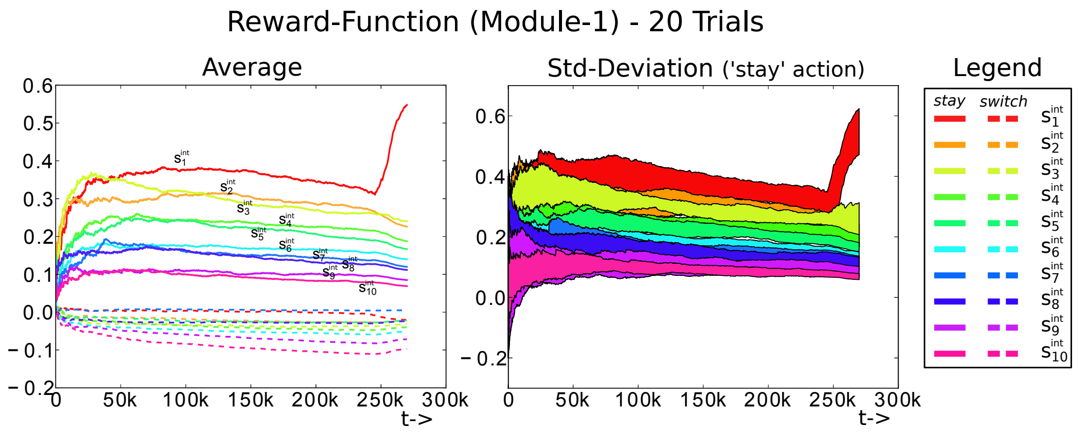


Figure 4.13. **10 Different Input Signals**: See text for details. (Figures are best viewed in color)

random initializations. Figure 4.12(a) shows the average ROC estimation error plot, where the shaded region represents the standard deviation. Figure 4.12(b) shows the reward function for the first abstraction module until a new module is created. It is clear that for all the 20 trials the algorithm learns the abstraction corresponding to the easiest signal $\mathbf{x}_1$ as its first abstraction module. Figures 4.12(c)-(d) show plots of the reward function for the second and third abstraction module from their time of creation until they are frozen. Figures 4.12(e)-(g) show the reward function (clipped until the shortest trial time) averaged over the 20 trials for each module. This experiment result shows that the algorithm learns abstractions for the observation streams in the order of their learning difficulty, which supports the theoretical analysis of the problem.

4.6.2 10 Different Input Signals

The next experiment demonstrates that the algorithm scales well to a larger number (10) of different input signal streams (similar to the ones in Experiment 1). These streams have the following increasing curiosity-function values ($\Omega_1 - \Omega_{10}$): (0.981140, 0.984279, 0.987169, 0.989791, 0.991922, 0.993411, 0.995511, 0.996260, 0.997685, 0.998256). Observation stream $\mathbf{x}_1$ is the easiest to learn compared to the other streams. I use parameters similar to the previous experiment, except for the decay multiplier which is set to 0.99986 and is reset to a value 0.99 when $\epsilon < 0.9$. This was done to speed up the experiment.

The experiment is conducted for 20 trails with different random seed initialization. In 19 out of the 20 trials, the algorithm successfully converged to the optimal solution. Figure 4.13 shows the average and standard-deviation of the reward

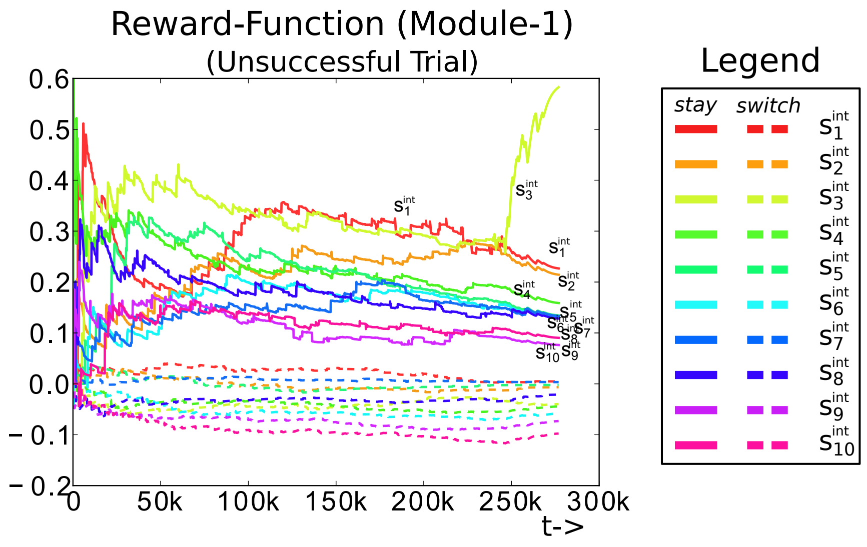


Figure 4.14. Reward function of the unsuccessful trial. (Figure is best viewed in color)

function of module-1 for 20 trials. Clearly, the expected reward function stabilizes to a state such that: $R^{\text{int}}(s_1^{\text{int}}, \textit{stay}) > R^{\text{int}}(s_2^{\text{int}}, \textit{stay}) > R^{\text{int}}(s_3^{\text{int}}, \textit{stay}) > R^{\text{int}}(s_4^{\text{int}}, \textit{stay}) > R^{\text{int}}(s_5^{\text{int}}, \textit{stay}) > R^{\text{int}}(s_6^{\text{int}}, \textit{stay}) > R^{\text{int}}(s_7^{\text{int}}, \textit{stay}) > R^{\text{int}}(s_8^{\text{int}}, \textit{stay}) > R^{\text{int}}(s_9^{\text{int}}, \textit{stay}) > R^{\text{int}}(s_{10}^{\text{int}}, \textit{stay}) > 0$.

However, for the one unsuccessful trial an abstraction corresponding to the observation stream $\mathbf{x}_3$ was learned as the first abstraction. Figure 4.14 shows the reward function for the unsuccessful trial. During exploration, the reward function did not yet stabilize in the order of Ω values of the observation streams. Therefore, as the ϵ of the ϵ-greedy strategy quickly decreased to zero, the result converged to a suboptimal solution. The result can be improved by using a larger decay multiplier. In this experiment, I showed the result for only the first module here since other modules follow a similar trend (if not better).

4.6.3 Maze Environment with Noisy Streams

Here, I test the design modifications presented in Section 4.5 on a bounded 1D-chain maze environment and in the presence of uncompressible noisy streams as shown in Figure 4.15(a). Each internal state corresponds to a room with an audio source (see Figure 4.10). Each internal state has only two neighbors, except for the boundary states. Action switch shifts the agent's internal state to one of its 2 neighboring states. States s_1^{int} and s_7^{int} are associated with a white noise stream, and s_3^{int} and s_5^{int} are associated with two signal streams, shown in Eq. (4.23) and Eq. (4.24), respectively. The rest of the states have no observation streams (zero value). The 2D observation

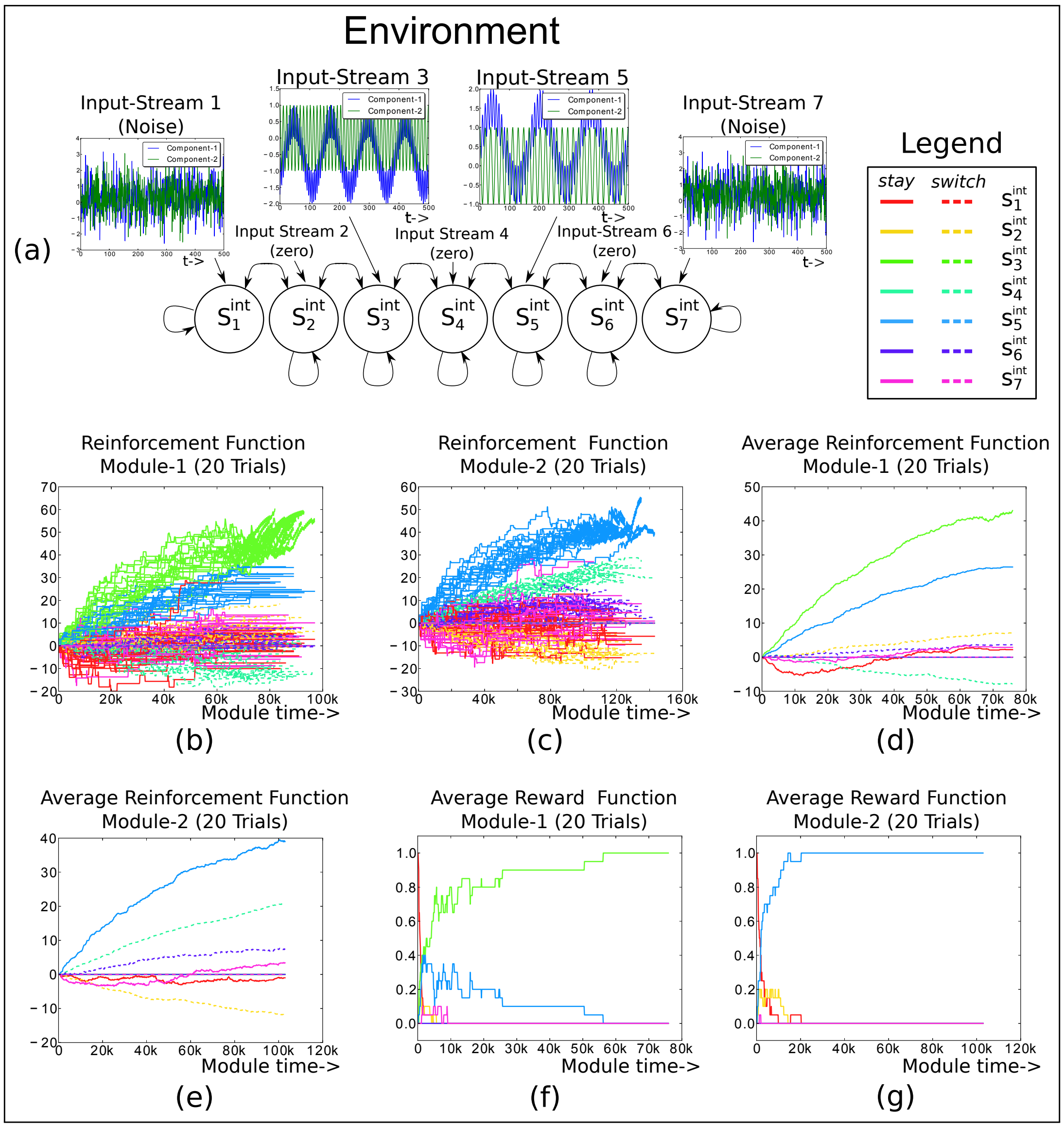


Figure 4.15. **Maze Environment with Noisy Streams**: See text for details. (Figures are best viewed in color)

streams are expanded to 5 dimensions to handle non-linearity in the input. Based on the Ω values, observation stream $\mathbf{x}_1$ is the easiest signal to encode followed by $\mathbf{x}_2$ and then $\mathbf{x}_3$.

Experiment parameters: Since, the environment is a maze, I use the design changes to the algorithm as discussed in Section 4.5. Action-space $\mathscr{A}^{\text{dint}}$ is equal to $\{\text{Left}, \text{Home}, \text{Right}\}$. *Left* and *Right* actions shifts the agent's state from s_i^{int} to s_{i-1}^{int} and s_{i+1}^{int} respectively, while *Home* action makes the agent to remain in the same state. IncSFA-ROC parameters are set to the same values as in Experiment 4.6.1. The initial ϵ-greedy value is set to $\epsilon = 1.1$, with a 0.999 decay multiplier. However, when $\epsilon < 0.9$, the decay multiplier is set to 0.992 to speed up the experiment. I use indicator basis functions (for both Ψ and Ψ_d) for LSPI. γ is set to 0.99 and γ_d is set to 0.9. The values for τ, σ, β, $\mathscr{L}$, α are set to the same values as in Experiment 4.6.1.

The experiment is conducted for 20 trails with different random seed initializations. Figures 4.15(b)-(c) show the *reinforcement function* (see Section 4.3.4) over time for each trial and Figures 4.15(d)-(e) show the reinforcement function averaged over the 20 trials. The average reinforcement values corresponding to the noise are close to zero. Figures 4.15(f)-(g) show the average thresholded reward function over time (see Section 4.5). The algorithm successfully converges to the optimal solution in all the 20 trials avoiding the noisy streams. The two abstractions corresponding to the observation streams $\mathbf{x}_3$ and $\mathbf{x}_5$ are learned sequentially.

4.6.4 Large Maze Environment with Duplicated Streams

Here, I evaluate the algorithm on a larger maze environment as shown in the Figure 4.16(a). The environment has 100 grid points. Each grid point topologically represents a room (see Figure 4.10) with an arbitrarily associated audio stream such that, there are in total 10 grid points each of $\mathbf{x}_1$ (Eq. (4.23)), $\mathbf{x}_2$ (Eq. (4.24)) and a random stream. The remaining grid points are associated with an empty (zero) stream. The agent is unaware of the audio stream distribution and can traverse along the grid points to observe samples from the associated time-varying audio streams. The objective here is to learn an abstraction corresponding to $\mathbf{x}_1$ first by moving into any of the grid points containing $\mathbf{x}_1$, followed by an abstraction for $\mathbf{x}_2$.

Since there are in total 100 observation streams, Curious Dr. MISFA's environment has 100 internal states ($\mathscr{S}^{\text{int}} = \{s_1^{\text{int}}, ..., s_{100}^{\text{int}}\}$). Each internal state has only four neighbors (except for the boundary states). Action $switch$ shifts the agent's state to one of its 4 neighboring states.

Experiment parameters: Similar to Experiment 3, I use the design changes to the algorithm as discussed in Section 4.5. Action-space $\mathscr{A}^{\text{dint}}$ is {North, South, Home, East, West}. IncSFA-ROC parameters are set to the same values as in Experiment 4.6.1. The initial ϵ-greedy value is set to $\epsilon = 1.1$, with a 0.999 decay multiplier. However, when $\epsilon < 0.9$, the decay multiplier is set to 0.99 to speed up the experiment. I use *Augmented Krylov* basis functions (see Section 4.5), with 10 Krylov bases, 30 LEM bases for Ψ and 0 Krylov bases, 40 LEM bases for Ψ_d. γ is set to 0.99 and γ_d is set to 0.85. The values for τ, σ, β, $\mathscr{L}$ and α are set to the same values as in Experiment 4.6.1.

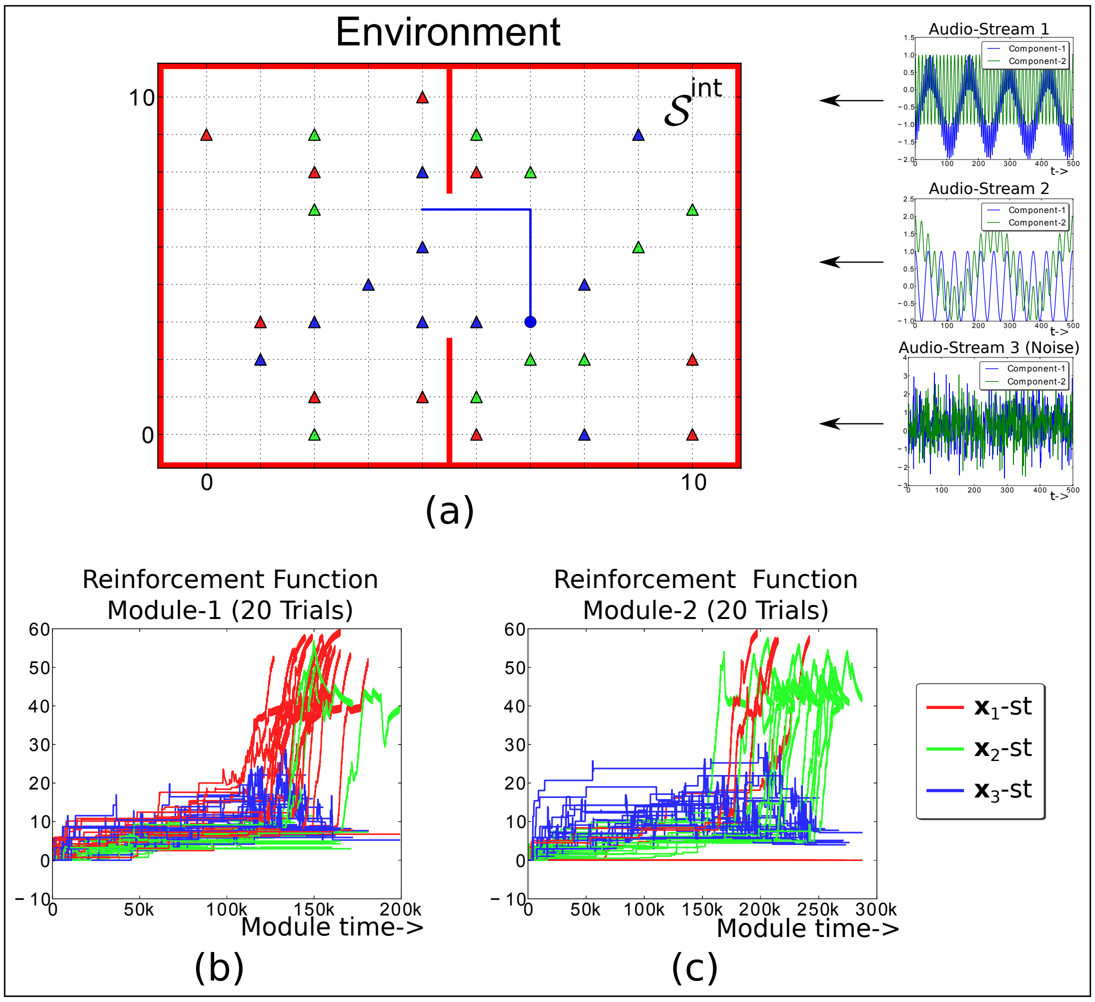


Figure 4.16. **Large Maze with Duplicated Streams**: See text for details. (Figures are best viewed in color)

The secondary policy π^{dint}, learned during the exploration phase ($\epsilon \gtrsim 1$), enables the agent to get to the desired states quickly using the deterministic action-space $\mathscr{A}^{\text{int}}$ when $\epsilon < 1$. The experiment is conducted for 20 trails with different random initializations. Figures 4.15(b)-(c) show the plot of *reinforcement function* over time for each trial. Each red curve represents the maximum reward of all the 10 states associated with the audio stream $\mathbf{x}_1$ over time for each trial. While the green and the blue curves represent the same but for streams $\mathbf{x}_2$ and random stream respectively. In 16 out of the 20 trials, an abstraction corresponding to the audio stream $\mathbf{x}_1$ (Eq. (4.23)) is learned first followed by audio stream $\mathbf{x}_2$ (Eq. (4.24)).

This experiment demonstrates that Curious Dr. MISFA can successfully be applied to maze environments. The algorithm learns an abstraction while simultane-

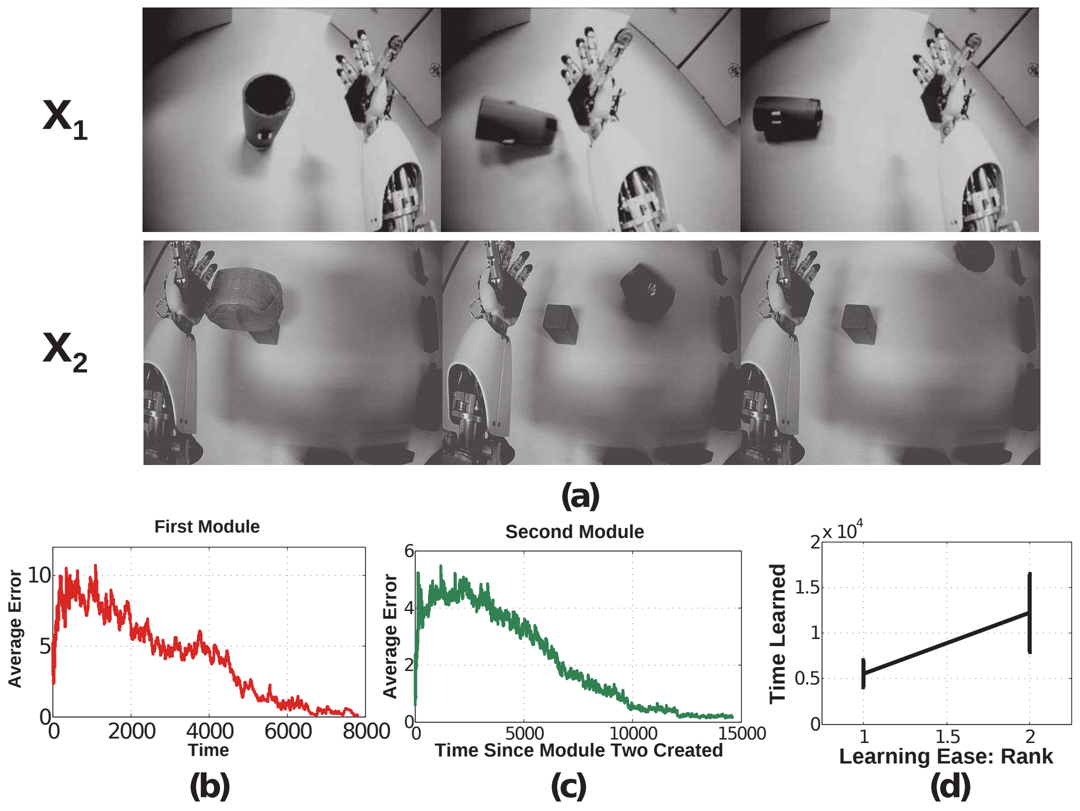


Figure 4.17. This experiment uses image sequences from the iCub's cameras, while it moves its arm and interacts with objects. See text for details.

ously developing a policy to get to the grid point with the easiest learnable observation stream.

4.6.5 An iCub Experiment: High-Dimensional Image Streams

This experiment uses an embodied agent (iCub) with real high-dimensional images (grayscale 75×100) from the robot's eyes. There are two video streams $\{\mathbf{x}_1, \mathbf{x}_2\}$, where each is generated by the iCub's exploration via random movement of its shoulder joint, causing the outstretched hand to eventually displace the single object in its field of view. The iCub is not given any prior knowledge about the objects, itself, or any concepts at all. It merely observes the pixel values, and uses Curious Dr. MISFA for learning and decision making. Since there are two observation streams, Curious Dr. MISFA's environment has two internal states: $\mathscr{S}^{\text{int}} = \{s_1^{\text{int}}, s_2^{\text{int}}\}$. In s_1^{int}, the object is a cup, which topples over upon contact with very predictable outcome. In the other state s_2^{int}, the object will roll in different directions.

Experiment parameters: CCIPCA learning rate is equal to $1/t$ with amnesic

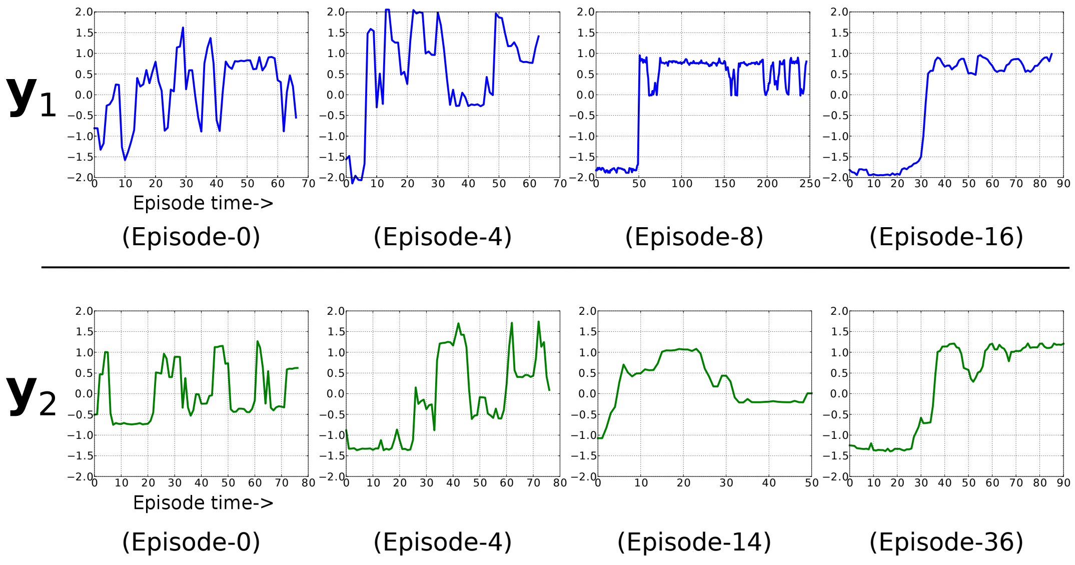


Figure 4.18. Example emergence of object-centric slow features for both observation streams. The final result encodes two states of each object — upright or displaced.

parameter set to 0.4, while the MCA learning rate is 0.01. CCIPCA did variable size dimension reduction by calculating how many eigenvalues would be needed to keep 98% of the input variance (typically this was between 10 and 15) so the 7500 pixels could be effectively reduced to only about 10 dimensions. The output dimension of IncSFA is set to 1. Each clustering implementation has a maximum number of clusters set to $N^{\max} = 3$. The estimation error threshold, below which the current module is saved and a new module is created, is $\delta = 2.3$. The amnesic parameter is set to $\nu^{\text{amn}} = 0.2$. The initial ϵ-greedy value is set to $\epsilon = 0.6$, with a 0.93 decay multiplier. The experiment is carried out in an episodic manner wherein each episode involves random exploration and an object-robot interaction event and has between 50-250 images. At the beginning of each episode, the object is always replaced in the same position.

Three example images from each of the two observation streams are shown in Figure 4.17(a). The stream $\mathbf{x}_1$ is easier to learn than $\mathbf{x}_2$, since the Ω value for $\mathbf{x}_1$ is 0.9982, and the Ω value for the $\mathbf{x}_2$ is 0.9988. This experiment is an example of the case $n = 2$ (see Section 4.3.7), where the stochastic *switch* action is equivalent to a deterministic *switch* to the other state. The slowest feature here would encode the mixture-signal and extract the identity of the two video streams, which is to be expected when the input signals from widely different clusters [Zhang and Tao, 2012]; in a sense this is similar to a multiple rooms case [Mahadevan and Maggioni, 2007], where the features code for room ID. *A simple hack (not required for $n > 2$)*: in order to prevent learning progress from continual switching, the following rule was implemented. When the agent decides to remain in its current internal state, it experiences two subsequent episodes, but when it decides to switch to the other, it only experiences one. In other words, the agent is given more time to learn by staying rather than by switching.

15 experimental runs were performed. Figure 4.17(b)-(d) show results. Part (b) shows the average estimation error during the first module's learning, while part (c) shows average estimation error for the second. In part (d), one can see that an abstraction for the stream $\mathbf{x}_1$ was indeed mostly learned first (in 14 of the 15 runs, this was the case). Examples of the learned slow feature outputs over time are shown in Figure 4.18. Both representations eventually encode whether the object is displaced or not. Most of the information in the image sequences can be broken down into three components: a baseline (the background), the object, and the arm. The object changes slower than the arm, so it is preferentially extracted by SFA. Moreover, the object-based features are invariant to the arm's position. Generalization is also possible in a limited sense. If the arm were replaced by some other object (e.g., a stick), the feature output would not be perturbed. For more robust generalization, better pre-processing is probably needed, as is typical with appearance-based vision techniques [Cui and Weng, 2000].

These experimental results show that the Curious Dr. MISFA algorithm learns slow feature abstractions for the observation streams, ranging from simple oscillatory signals to high-dimensional visual inputs, in the order of increasing learning difficulty. These empirical results support the theoretical justifications presented in Section 4.3.7.

4.7 Neural Correlates to Curious Dr. MISFA

In this section, I will discuss some of the neurophysiological systems realized in the brain, whose functional roles mirror those of Curious Dr. MISFA; namely, the interactions between the neuromodulatory systems involved in intrinsic motivation, task engagement, task switching, and value approximation. The ideas presented in this section were produced in collaboration with Matthew Luciw and Sohrob Kazerounian.

4.7.1 SFA and Competitive Learning — Entorhinal Cortex and Hippocampus

Slow Feature Analysis variants have been used to simulate representation learning in a number of biological scenarios. Based on the general principle that underlying driving forces manifest through slow changes in sensory observations, the features that emerge from SFA often encode important *invariants*. Hierarchical SFA has been shown to develop *grid cells* from high-dimensional visual observation streams [Franzius et al., 2007]. Grid cells, found in the entorhinal cortex [EC; Hafting et al., 2005], have a pattern of firing that effectively represent hexagonal codes of any two-dimensional environment. As such, grid cells are effective *general* representations for spatial navigation in typical environments.

A competitive learning layer over the top-layer of slow features leads to features acting as *place cells* or *head-direction cells*, depending on what changes more slowly from the observation sequences. A place cell will fire when the animal is in a specific location in the environment, typically invariant to its heading direction. Head-direction cells fire when the animal faces a certain direction, no matter what coordinate position it is in. Place cells and head-direction cells are found in the hippocampus [O'Keefe and Dostrovsky, 1971; Taube et al., 1990], which has input from EC. It's been hypothesized that the hippocampus acts as a relatively fast encoder of specific, episodic information on top of the cortex, which learns general structure from data over a long period [Cohen and O' Reilly, 1996] — "It has been proposed that this universal spatial representation might be recoded onto a context-specific code in hippocampal networks, and that this interplay might be crucial for successful storage of episodic memories [Fyhn et al., 2007]."

SFA's biological plausibility was furthered by IncSFA, which avoids batch processing and has Hebbian and anti-Hebbian updating equations. Hierarchical SFA [Franzius et al., 2007] and Hierarchical IncSFA [Luciw et al., 2012], with competitive learning on top, was shown to develop place and head-direction cell representations. Inspired by these results, the structure used for learning representations in Curious Dr. MISFA is as follows: A slow feature learner (possibly hierarchical) for *global* features (via IncSFA), inputs into a competitive learner for development of *local* features (via Robust Online Clustering).

4.7.2 Neuromodulatory Subsystems for Intrinsic Reward and Task Switching

Intrinsic Rewards: Dopamine and Learning Progress

Dopaminergic projections originate from the ventral tegmental area (VTA). Dopamine has been implicated in reward prediction [Schultz et al., 1997], leading to plausible relation to the theory of reinforcement learning — specifically, dopamine may be acting as a temporal-difference (TD) error signal. However, this account remains controversial [Redgrave et al., 1999; Kakade and Dayan, 2002]. A major deviation from the dopamine as TD-error theory comes from data implicating dopamine in responding to novel salient stimuli [Schultz, 1998; Redgrave and Gurney, 2006], even for stimuli that are not predictive of reward. Dopaminergic responses to such stimuli fade over subsequent trials. It has been proposed that this characteristic serves the purpose of a "novelty bonus" — e.g., a reward addendum serving as an "optimistic initialization".

These data present intriguing correlations to the curiosity theory. Dopamine release in response to novel stimuli could potentially signal a predicted intrinsic reward — an expectation of *learning progress*. Could DA in some situations signal the intrinsic reward? Dopamine's potential role in intrinsic motivation has been discussed before [Redgrave and Gurney, 2006; Kaplan and Oudeyer, 2007], but not with respect to the formal theory of curiosity [Schmidhuber, 2010b], which predicts that intrinsic reward should be proportional to compression progress. Computational models in neuroscience often treat intrinsic reward as resulting from the novelty of a stimulus. If intrinsic reward really does result from novelty, we would expect persistent high levels of dopamine in response to unpredictable noisy stimuli (as it remains novel from moment to moment). On the other hand, if intrinsic rewards encode compression progress, we would expect decreases in the level of dopamine as the predictive model becomes unable to learn anything more about the structure of the noise.[5]

Engagement and Disengagement (and Switching): Norepinephrine

Neurons of the locus coeruleus (LC), in the brainstem, are the sole source of norepinephrine (NE). NE is linked to arousal, uncertainty, vigilance, attention, motivation, and task-engagement. The LC-NE system is more traditionally thought to affect levels of arousal, but more recently has been implicated in optimization of behavioral performance [Usher et al., 1999; Aston-Jones and Cohen, 2005; Sara, 2009].

In that context, the activity of the LC-NE system can be understood as modulation of exploration-exploitation. The tonic differences in LC-NE response are associated with levels of arousal. Tonic NE response is correlated with task performance levels [Usher et al., 1999]. Low tonic activity coincides with low attentiveness and alertness [Aston-Jones et al., 1991], while high tonic activity coincides with agitation and distractibility [Aston-Jones and Cohen, 2005]. Good task performance coincides with an intermediate tonic level during which phasic bursts of activity are observed, while poor task performance due to distraction is associated with high tonic activity. In phasic mode during periods of intermediate tonic NE activity, NE

[5] To my knowledge, this has not been tested yet.

is released in response to task-relevant events [Dayan and Yu, 2006]. As suggested by Usher et al. and others [Usher et al., 1999; Aston-Jones and Cohen, 2005], the phasic modes might correspond to exploitation, whereas high tonic states of NE activity might correspond to exploration.

When it is beneficial for the agent to remain engaged in the current task, the tonic NE level remains moderate, and only relevant task stimuli will be salient. However, when it is not beneficial to remain engaged in the current task, the NE level raises and task-irrelevant stimuli become more salient. This drives the agent to distractibility, and task performance suffers. Attending to some distractor stimuli could have the effect of causing the agent to switch to another task in which this distractor becomes relevant, ostensibly with the purpose of exploring among available tasks (i.e., it "throws the ball in the air so another team can take it" [Aston-Jones and Cohen, 2005]).

In Curious Dr. MISFA, the agent's two internal-actions (stay or switch) and the reasons they are taken link to the NE-driven task engagement/disengagement model. Boredom (low NE) indicates that a good representation has already been learned, leading to low estimation error and low potential intrinsic reward. Distractibility (high NE) indicates that the errors are too high, not decreasing quickly enough, or cannot be reduced. In this case, it becomes valuable to switch and find some other source of information, where learning may progress faster. When the agent has found a good source, the estimation errors decrease regularly, providing intrinsic reward that leads to a high value estimate and a desire to stay in that state.

4.7.3 Frontal Cortex: Value Function and Representation Selection

The NE and DA neuromodulatory systems each have reciprocal connectivity with the prefrontal cortex — executive areas, which deal with cognitive aspects such as decision making, and top-down control of other functions, such as selective attention [Miller, 2000]. If the LC-NE system is handling task-engagement and disengagement based on some value judgement, then this system needs to be controlled by another system that is estimating these values. The prefrontal cortex (PFC) plausibly plays a role in value estimation, and might use the utility information to provide top-down regulation of the activities of the LC neurons [Ishii et al., 2002].

PFC and nearby structures, specifically, the anterior cingulate cortex (ACC) and the orbital frontal cortex (OFC), are implicated in value-based judgements. The ACC is involved in error detection (i.e., recognizing a prediction error) and estimating the costs of these errors [Bush et al., 2002]. OFC is thought to be of import in the motivational control of goal-directed behaviors [Rolls et al., 1996] — OFC damage leads to responses to objects which are no longer rewarding [Meunier et al., 1997; Rolls et al., 1994]. The dorsolateral pre-frontal cortex (DLPF) is implicated in value-based working memory [Rao et al., 1997]. Thus, these structures could possibly work together to estimate a value function, in the RL sense [Ishii et al., 2002].

Another important property of PFC is to maintain an appropriate task representation, i.e., imposing internal representations that guide subsequent performance, and switching these for another when it is no longer appropriate [Miller, 2000; Cohen et al., 2004]. This property requires mechanisms to keep goal-relevant information (i.e., what should be considered salient and what should be considered a distractor) enabled in resonance with lower structures. Further, it requires a mechanism to maintain a context despite bottom-up disturbances, and a mechanism to switch the context. The PFC has connections from and to higher-order associative cortices, so it is in a good position to impose task-relevant representations from the top-down. Such "executive attention" enables memory representations to be "maintained in a highly accessible state in the presence of interference, and these representation may reflect action plans, goal states or task-relevant stimuli in the environment [Kane and Engle, 2002]".

4.8 Conclusion

In this chapter, I presented an autonomous curiosity-driven modular incremental slow feature learning algorithm that learns invariant slow feature abstractions from multiple time-varying input observation streams sequentially, in the order of increasing learning difficulty. The method continually estimates the initially unknown learning difficulty through intrinsic rewards generated by exploring the observation streams using a *stay-switch* action selection mechanism. The architecture of the method includes (a) a reinforcement learner that generates policies to select an observation stream based on the intrinsic rewards, (b) an adaptive IncSFA module that updates an abstraction based on the incoming observations and (c) a gating system that prevents encoding inputs that have been previously encoded. I formalized the learning problem as an optimization problem and also presented a formal analysis to prove that the Curious Dr. MISFA algorithm converges to the optimal solution under a few mild conditions. Experimental results show that the method successfully learns abstractions in the order of increasing learning difficulty, over different experimental settings. Each learned abstraction encodes some previously unknown regularity in the input observations, which can then be used as a basis for acquiring new skills. The next chapter presents a framework to translate the learned abstractions into skills.

Chapter 5

Continual Curiosity-Driven Skill Acquisition

In Chapter 4, the Curious Dr. MISFA algorithm was presented, which actively learns multiple slow feature abstractions in the order from least to most costly, as predicted by the theory of artificial curiosity. Each abstraction learned encodes some previously unknown regularity in the input observations, which forms a basis for acquiring new skills. In this chapter, I will present the Continual Curiosity-Driven Skill Acquisition [CCSA; Kompella et al., 2014b] framework, which translates the abstraction learning problem of Curious Dr. MISFA to a continual skill acquisition problem. Using CCSA, a humanoid robot driven purely by its intrinsic motivation can continually acquire a repertoire of skills that map the many raw-pixels of image streams to action-sequences.

5.1 Overview

Figure 5.1 illustrates the overall CCSA framework. The learning problem associated with CCSA can be described as follows: From a set of pre-defined or previously acquired input exploratory behaviors that generate potentially high-dimensional time-varying observation streams, the objective of the agent is to (a) acquire an easily learnable yet unknown target behavior and (b) re-use the target behavior to acquire more complex target behaviors. The target behaviors represent the skills acquired by the agent. A sample run of the CCSA framework to acquire a skill is as follows (see Figure 5.1):

(a) The agent starts with a set of pre-defined or previously acquired exploratory behaviors. I make use of the *options* framework [Sutton et al., 1999] to for-

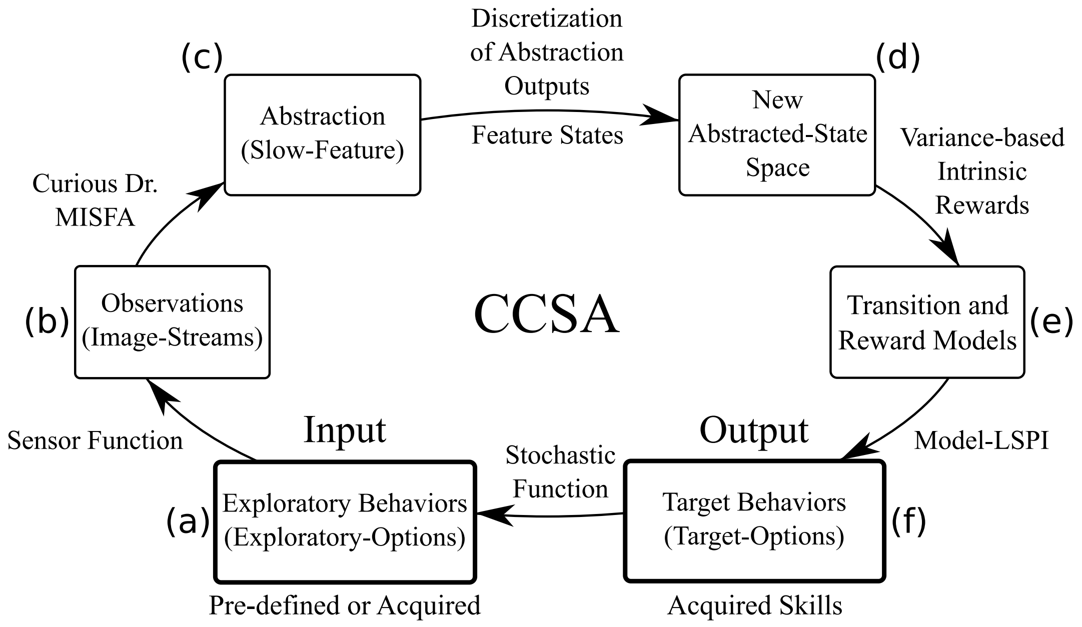

(c)
Abstraction (Slow-Feature)
Discretization of Abstraction Outputs
Feature States
(d)
New Abstracted-State Space
Variance-based Intrinsic Rewards
Curious Dr. MISFA
(b)
Observations (Image-Streams)
CCSA
Transition and Reward Models
(e)
Sensor Function
Input
Output
Model-LSPI
Stochastic Function
(a)
Exploratory Behaviors (Exploratory-Options)
Target Behaviors (Target-Options)
(f)
Pre-defined or Acquired
Acquired Skills

Figure 5.1. High-level control flow of the Continual Curiosity-Driven Skill Acquisition (CCSA) framework. (a) The agent starts with a set of pre-defined or previously acquired exploratory behaviors (represented as exploratory options). (b) It makes high-dimensional observations upon actively executing the exploratory options. (c) Using the Curious Dr. MISFA algorithm, the agent learns a slow feature abstraction that encodes the easiest to learn yet unknown regularity in the input observation streams. (d) The slow feature abstraction outputs are clustered to create feature states that are augmented to the agent's abstracted state space. (e) A Markovian transition model of the new abstracted state space and an intrinsic reward function are learned through exploration. (f) A deterministic policy is then learned via model-based Least Squares Policy Iteration (Model-LSPI) and a target option is constructed. The deterministic target-option's policy is modified to a stochastic policy in the agent's new abstracted states and is added to the set of exploratory options.

mally represent the exploratory behaviors as exploratory options (see Section 5.2 for a formal definition of the terminology used here).

(b) The agent makes high-dimensional observations through a sensor-function, such as a camera, upon actively executing the exploratory options.

(c) Using the curiosity-driven modular incremental slow feature analysis (Curious Dr. MISFA) algorithm, the agent learns a slow feature abstraction that encodes the easiest to learn yet unknown regularity in the input observation streams (see Section 4).

(d) The slow feature abstraction outputs are clustered to create feature states that are augmented to the agent's abstracted state space, which contains previously encoded feature states (see Section 5.3.3).

(e) A Markovian transition model is learned by exploring the new abstracted state space. The reward function is also learned through exploration, with the agent being intrinsically rewarded for making state transitions that produce a large variation (high statistical variance) in slow feature outputs. This specialized reward function is used to learn action-sequences (policy) that drives the agent to states where such transitions will occur.

(f) Once the transition and reward functions are learned, a *deterministic* policy is learned via model-based Least-Squares Policy Iteration [LSPI; Lagoudakis and Parr, 2003]. The learned policy and the learned slow feature abstraction together constitute a target option, which represents the acquired skill (see Section 5.3.3).

(f)-(a) The deterministic target-option's policy is modified to a stochastic policy in the agent's new abstracted states and is added to the set of exploratory options (see Section 5.3.4). This enables the agent to reuse the skills to acquire more complex skills in a continual open-ended manner [Ring, 1994, 1997].

CCSA is a task-independent algorithm, i.e., it does not require any design modifications when the environment is changed. However, CCSA makes the following assumptions: (a) The agent's default abstracted state space contains low-level kinematic joint poses of the robot learned offline using Task Relevant Roadmaps [Stollenga et al., 2013]. This is done to limit the iCub's exploration of its arm to a plane parallel to the table. This assumption can be relaxed resulting in a larger space of arm exploration of the iCub, and the skills thus developed may be different. (b) CCSA requires at least one input exploratory option. To minimize human inputs into the system, in the experiments at $t = 0$, the agent starts with only a single input exploratory option, which is a random-walk in the default abstracted state space. However, environment or domain specific information can be used to design several input exploratory options in order to shape the resulting skills. For example, random-walk policies mapped to different sub-regions in the robot's joint space can be used.

The rest of the chapter is organized as follows. Section 5.2 presents a theoretical formulation of the learning problem associated with the CCSA algorithm. Section 5.3 discusses different parts of the CCSA algorithm in detail. Section 5.4

presents the pseudo-code of the algorithm along with other implementation details. Section 5.5 presents experimental results conducted using an iCub humanoid and Section 5.6 concludes.

5.2 Learning Problem Formalized

In this section, I will present a theoretical formulation of the learning problem associated with the Continual Curiosity-Driven Skill Acquisition (CCSA) framework. I will first formalize the curiosity-driven skill acquisition problem and then later in the section I will present a continual extension of it.

5.2.1 Curiosity-Driven Skill Acquisition Problem

Given a fixed set of input exploratory options, which generate potentially high-dimensional observation streams that may or may-not be unique, the objective is to acquire a previously unknown target option corresponding to the easily-encodable observation stream. Figure 5.2 illustrates the learning process. The learning process iterates over the following steps:

(a) Estimate the easily-encodable yet unknown observation stream, while simultaneously learning a compact encoding (abstraction) for it.

(b) Learn an option that maximizes the statistical variance of the encoded abstraction output. The problem is formalized as follows:

Notation:

Environment: An agent is in an environment that has a state space $\mathscr{S}$. It can take an action $a \in \mathscr{A}$ and transition to a new state according to the transition-model (environment dynamics) $\mathscr{P} : \mathscr{S} \times \mathscr{A} \to \mathscr{S}$. The agent observes the environment state s as a high-dimensional vector, $\mathbf{x} \in \mathbb{R}^I, I \in \mathbb{N}$.

Abstraction: Let Θ denote some online abstraction-estimator that updates a feature-abstraction ϕ, where $\Theta(\mathbf{x}, \phi)$ returns an updated abstraction for an input $\mathbf{x}$. The abstraction $\phi : x \mapsto y$ maps a high-dimensional input observation stream $\mathbf{x}(t) \in \mathbb{R}^I$ to a lower-dimensional output $\mathbf{y}(t) \in \mathbb{R}^J, J \ll I, J \in \mathbb{N}$, such that $\mathbf{y}(t) = \phi(\mathbf{x}(t))$.

Abstracted State Space: As defined earlier in Section 4.3.1, the agent's abstracted state space $\mathscr{S}^\Phi$ contains the space spanned by the outputs $\mathbf{y}$ of all the abstractions that were previously learned using Θ.

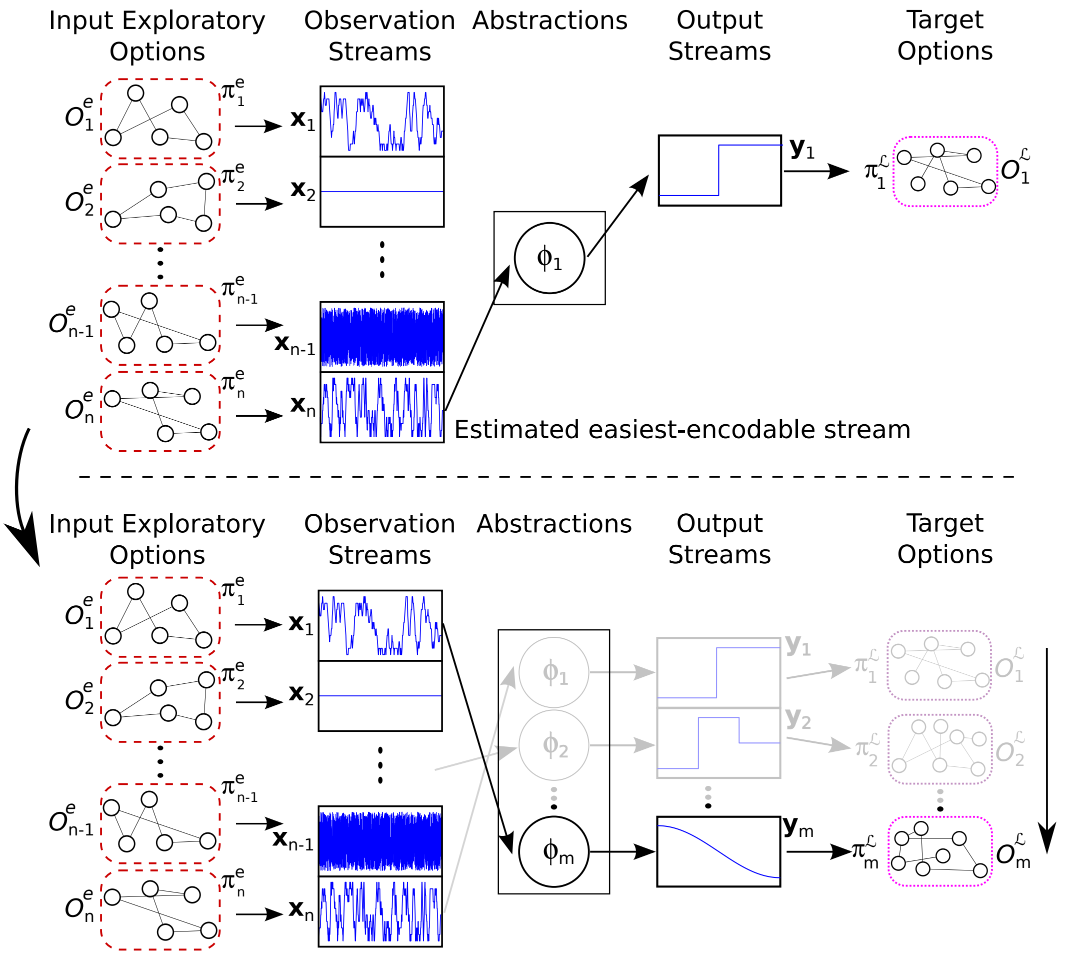


Figure 5.2. Curiosity-Driven Skill Acquisition: Given a fixed set of input exploratory options (represented by red dashed boxes) generating n observation streams, abstractions (represented by circles) and corresponding target options (represented by pink dotted boxes) are learned sequentially in order of increasing learning difficulty. The learning process involves not just acquiring the target options, but also the sequence in which they are acquired. The top figure shows an example of the desired result after the first target option was learned. The bottom figure shows the the desired end result after all possible target options have been learned. The curved arrow indicates the temporal evolution of the learning process.

Input Exploratory Options: The agent can execute an input set of pre-defined temporally extended action sequences, called the *exploratory option set* $\mathcal{O}^e = \{O_1^e, ..., O_n^e; n \geq 1\}$. Each exploratory option is defined as a tuple $\langle \mathcal{I}_i^e, \beta_i^e, \pi_i^e \rangle$, where $\mathcal{I}_i^e \subseteq \mathcal{S}^\Phi$ is the initiation set comprising abstracted states where the option is available, $\beta_i^e : \mathcal{S}^\Phi \to [0,1]$ is the option termination condition, which will determine where the option terminates (e.g., some probability in each state), and $\pi_i^e : \mathscr{I}_i^e \times \mathscr{A} \rightarrow [0,1]$ is a pre-defined *stochastic* policy, such as a random walk within the applicable state space. Each exploratory-option's policy generates an observation stream via a sensor-function $\mathscr{U}$, such as an image sensor like a camera:

$$\mathbf{x_i}(t) = \mathscr{U}(\mathscr{P}(s, \pi_i^e(s^\Phi)))$$

where $\mathscr{P}$ is the unknown transition model of the environment, $s^\Phi \in \mathscr{I}_i^e$ is the agent's current abstracted state while executing the i^{th} exploratory option O_i^e at time t, $s \in \mathscr{S}$ is the corresponding environment state, and $\pi_i^e(s^\Phi)$ returns an action. Let $X = \{\mathbf{x_1}, ..., \mathbf{x_n}\}$ denote the set of n I-dimensional observation streams generated by the n exploratory-option's policies. At each time t however, the learning algorithm's input sample is from only *one* of the n observation-streams.

Curiosity Function: As defined earlier in Section 4.2, Ω denotes an unknown function indicating the speed of learning an abstraction by the abstraction-estimator Θ for a given input observation stream $\mathbf{x} \in X$. Ω induces a total ordering among the observation streams making them comparable in terms of learning difficulty.

Target Options: Unlike the pre-defined input exploratory-option set, a target-option set $\mathscr{O}^\mathscr{L}$ is the outcome of the learning process. A target option $O^\mathscr{L} \in \mathscr{O}^\mathscr{L}$ contains a learned abstraction ϕ_i and a learned *deterministic* policy $\pi_i^\mathscr{L}$. It is defined as a tuple $\langle \mathscr{I}_i^\mathscr{L}, \beta_i^\mathscr{L}, \phi_i, \pi_i^\mathscr{L} \rangle$. $\mathscr{I}_i^\mathscr{L} \subseteq (\mathscr{S}^\Phi \times \mathscr{S}_{\phi_i}^\Phi)$ is the target-option's initiation set defined over the *augmented* state space $(\mathscr{S}^\Phi \times \mathscr{S}_{\phi_i}^\Phi)$, where $\mathscr{S}_{\phi_i}^\Phi$ denotes the space spanned by the abstraction ϕ_i's output $\mathbf{y}(t) = \phi\left(\mathbf{x_j}(t)\right)$, $\mathbf{x_j} \in X$. β_i is the option's termination condition, and $\pi_i^\mathscr{L} : (\mathscr{S}^\Phi \times \mathscr{S}_{\phi_i}^\Phi) \rightarrow \mathscr{A}$ is the learned deterministic policy.

Encoded Observation Streams: Let $X^{\mathscr{O}^\mathscr{L}(t)}$ denote an ordered set (induced by time t) of *pre-images* of the learned abstractions outputs, $X^{\mathscr{O}^\mathscr{L}(t)} = \{\overleftarrow{\phi_i}\mathbf{y}_i, \forall O_i^\mathscr{L} \in \mathscr{O}^\mathscr{L}(t)\}$. $X^{\mathscr{O}^\mathscr{L}(t)}$ represents the set of encoded observation streams at time t.

Other Notation: $|.|$ indicates cardinality of a set, $||.||$ indicates Euclidean norm, $\langle . \rangle_t$ indicates averaging over time, $\langle . \rangle_t^\tau$ indicates windowed-average with a fixed window size τ over time. δ is a small scalar constant (≈ 0). $\mathrm{Var}[\cdot]$ represents statistical variance and $\forall$ indicates for all.

Problem Statement:

With the above notation, the curiosity-driven skill acquisition problem can be formalized as an optimization problem with the following objective. Given a fixed set of input exploratory options $\mathscr{O}^e$, find a target-option set $\mathscr{O}^\mathscr{L}$, such that the number of target options learned **at any time t** is maximized:

$$\max_{\mathscr{O}^{\mathscr{L}}} \quad \left|\mathscr{O}^{\mathscr{L}}(t)\right|, \quad \forall t = 1, 2, \ldots$$

under the constraints,

$$\langle y_i^j \rangle_t = 0, \quad \langle (y_i^j)^2 \rangle_t = 1, \qquad \forall j \in \{1, \ldots, J\}, \forall O_i^{\mathscr{L}} \in \mathscr{O}^{\mathscr{L}}(t), \tag{5.1}$$

$$\begin{pmatrix} \forall O_i^{\mathscr{L}} \in \mathscr{O}^{\mathscr{L}}(t), \exists j \in \{1, \ldots, n\}, \\ \text{and} \quad \forall O_{k \neq i}^{\mathscr{L}} \in \mathscr{O}^{\mathscr{L}}(t) \end{pmatrix} : \begin{matrix} \langle \|\Theta(\mathbf{x_j}, \boldsymbol{\phi}_\mathbf{i}) - \boldsymbol{\phi}_\mathbf{i}\| \rangle_\mathbf{t}^\tau \leq \delta \\ \langle \|\Theta(\mathbf{x_j}, \boldsymbol{\phi}_\mathbf{k}) - \boldsymbol{\phi}_\mathbf{k}\| \rangle_\mathbf{t}^\tau > \delta \end{matrix}, \tag{5.2}$$

$$\Omega(\mathbf{x}_i) \leq \Omega(\mathbf{x}_j), \quad \forall i < j \text{ and } \mathbf{x}_i, \mathbf{x}_j \in X^{\mathscr{O}^{\mathscr{L}}(t)}, \text{ and} \tag{5.3}$$

$$\pi_i^{\mathscr{L}} = \arg\max_{\pi_i} \mathrm{Var}\left[\phi_i\left(\mathscr{U}(\mathscr{P}(s, \pi_i(s^\Phi)))\right)\right], s^\Phi \in \mathscr{I}^{\mathscr{L}}, \forall O_i^{\mathscr{L}} \in \mathscr{O}^{\mathscr{L}}(t). \tag{5.4}$$

Constraint (5.1) requires that the abstraction-output components have zero mean and unit variance. This constraint enables the abstractions to be non-zero and avoids learning features for constant observation streams. Constraint (5.2) requires a *unique* abstraction be learned that encodes *at least* one of the input observation streams, avoiding redundancy. Constraint (5.3) imposes a total-ordering induced by Ω on the abstractions learned. Observation streams that are easier to learn are encoded first. And finally, Constraint (5.4) requires that each target-option's policy maximizes *sensitivity*, determined by the variance of the observed abstraction outputs [Saltelli et al., 2000]. In the rest of the text, I interchangeably use the word **skill** to denote a learned target option $O_i^{\mathscr{L}}$ and a **skillset** to denote the target-option set $\mathscr{O}^{\mathscr{L}}$.

Optimal Solution: For the objective to be minimized, at any time t, the optimal solution is to learn a target option corresponding to the current easiest but not yet learned abstraction among the observation streams (to satisfy Constraints (5.1-5.3)) and a policy that maximizes the variance in the encoded abstraction output (to satisfy Constraint (5.4)).

However, since Ω (see Constraint 5.3) is not known *a priori*, it needs to be estimated online by actively exploring the input exploratory options over time. One possible approach is to find (a) an analytical expression of Ω for the particular abstraction-estimator Θ and (b) an observation stream selection technique that can estimate the Ω values for each observation stream. This approach would be dependent on the abstraction-estimator used. However, like Curious Dr. MISFA, CCSA framework employs an approach independent of the abstraction-estimator used. CCSA makes use of reinforcement learning to estimate the Ω values, in the form of *curiosity rewards* generated through the *learning progress* made by Θ.

5.2.2 Continual Curiosity-Driven Skill Acquisition

In the above formulation, the agent has a fixed set of $n(\geq 1)$ input exploratory options. Therefore, the number of learnable target options is equal to the total number of learnable abstractions, which is at most equal to the number of input exploratory options:

$$\lim_{t\to\infty} \left|\mathcal{O}^{\mathcal{L}}(t)\right| \leq n. \tag{5.5}$$

To enable *continual learning* [Ring, 1994], the number of skills acquired by the agent should not necessarily be bounded and the agent needs to reuse the previously acquired skills to learn *more complex* skills. Therefore, continual curiosity-driven skill acquisition learning problem is a slightly modified version of the above formulation, such that the target options learned form a basis for *new input exploratory options*:

$$\mathcal{O}^{e} \leftarrow \mathcal{O}^{e} \cup \mathscr{F}(O^{\mathcal{L}}), \tag{5.6}$$

where $\mathscr{F}(.)$ denotes some functional variation of a deterministic target option to make it stochastic (exploratory). Therefore, the number of input exploratory options (n) increases whenever a new skill is acquired by the agent.
Sub-Target Options: Constraint (5.4) requires that each target-option's policy maximizes variance of the observed J-dimensional abstraction outputs. In principle, the constraint can be re-written such that only a subset of J dimensions of the abstraction can be used to learn a policy. This results in a maximum number of $2^J - 1$ learnable policies. The set of target options that all share the same abstraction $\{\langle \mathscr{I}_i^{\mathcal{L}}, \beta_i^{\mathcal{L}}, \phi_i, \pi_{ij}^{\mathcal{L}} \rangle; j \leq (2^J - 1)\}$ are denoted as sub-target options. To keep it simple, in the rest of the paper I use J dimensions as presented in Constraint (5.4) to learn the target-option's policy and therefore limiting 1 target option for each learned abstraction.

5.3 Method Description

Section 5.1 presented an overview of the CCSA framework. Here, I will present each part of the framework in detail and also show how it addresses the learning problem formalized in Section 5.2.

5.3.1 Input Exploratory Options

As discussed in Section 5.2, I defined a set of input exploratory options that the agent can execute to interact with the environment. Here, I will present details on

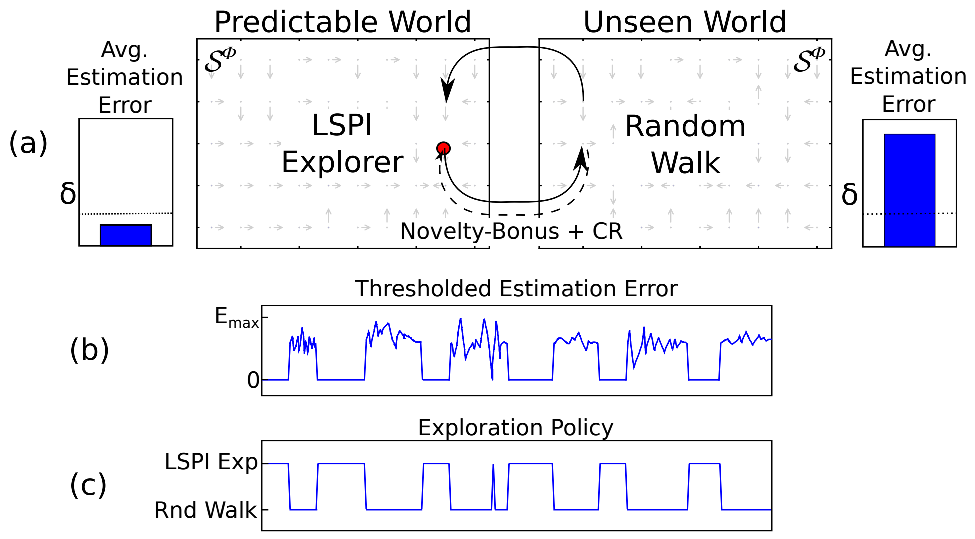


Figure 5.3. (a) Exploratory-option policy has two phases: If the estimation error of any already learned abstraction modules for the incoming observations is lower than threshold δ, the exploratory-option's policy is learned using Least Squares Policy Iteration (LSPI). If the estimation error is higher than the threshold then the policy is a random walk. (b) An example thresholded estimation error and the (c) corresponding exploration policy.

how to construct these options.

The simplest exploratory option policy is a random walk. However, A more sophisticated variant uses a form of *initial artificial curiosity*, derived from *error-based rewards* [Singh et al., 2004]. This exploratory-option's policy π^e is determined by the predictability of the observations $\mathbf{x}(t)$, but can also switch to a random walk when the environment is too unpredictable.

This policy π^e has two phases. If the estimation error of *any* already learned abstraction modules for the incoming observations is lower than threshold δ, the exploratory-option's policy is learned using Least-Squares Policy Iteration Technique [LSPI; Lagoudakis and Parr, 2003] with an estimation of the transition model actively updated over the option's state space $\mathcal{I}_i^e \subseteq \mathcal{S}^\Phi$, and an estimated reward function that rewards *high estimation errors*. Such a policy encourages the agent to explore its "unseen world" (Figure 5.3(a)). But if the estimation error of already learned abstraction modules is higher than the threshold δ, then the exploratory-option's policy is a random-walk over the option's state space. Figure 5.3 illustrates this error seeking exploratory-option's policy. I denote this policy as LSPI-Exploration policy. When the agent selects an exploratory option O_i^e to execute, it follows the option's policy, generating an observation stream $\mathbf{x_i} = \mathcal{U}(\mathcal{P}(s, \pi_i^e(s^\Phi)))$, until the termination condition is met. To keep it general and non-specific to the environment, in all my experiments, each exploratory-option's termination condition is such that the option terminates after a fixed τ time-steps since its execution.

Setting a different input exploratory-option set would influence the skills developed by CCSA. In my experiments at $t = 0$, the agent starts with only a single exploratory option as defined above. The LSPI-Exploration policy only speeds up the agent's exploration by acting deterministically in the predictable world and randomly in unseen world. Since at $t = 0$ the world is unexplored, LSPI-Exploration policy is just a random walk in the agent's abstracted states. Environment or domain-specific information can be used to design the input exploratory-option set in order to shape the resulting skills. For example, exploratory options with random-walk policies mapped to different sub-regions in the robot's joint space can be used.

5.3.2 Curiosity-Driven Abstraction Learning: Curious Dr. MISFA

At the core of the CCSA framework is the Curious Dr. MISFA algorithm. The order in which skills are acquired in the CCSA framework is a direct consequence of the order in which the abstractions are learned by the Curious Dr. MISFA algorithm. The input to the Curious Dr. MISFA algorithm is a set of high-dimensional observation streams $X = \{\mathbf{x_1}, ..., \mathbf{x_n} : \mathbf{x_i}(t) \in \mathbb{R}^I, I \in \mathbb{N}\}$, generated by the input exploratory-option's policies. The result is a slow feature abstraction corresponding to the easiest yet still unknown observation stream.

5.3.3 Learning a Target Option

Let ϕ_i denote the slow feature abstraction learned by the Curious Dr. MISFA algorithm corresponding to the current easiest-yet-unlearned exploratory option stream (say $\mathbf{x_j}$). The abstraction's output stream $\mathbf{y_i} = \phi_i(\mathbf{x_j})$ has zero-mean and unit-variance over time (see Chapter 3) and is a lower-dimensional representation of the input $\mathbf{x}_j$ (satisfies Constraint (5.1); see Section 5.2). The output values $\mathbf{y_i}(t)$ are discretized to a set of abstraction states $\mathscr{S}^{\phi_i}$, which represent the newly discovered abstracted states of the agent. A deterministic target option is then constructed as follows:

Initiation Set ($\mathscr{I}^{\mathscr{L}}$): The initiation set is simply the product state space: $\mathscr{I}_i^{\mathscr{L}} = (\mathscr{I}_j^e \times \mathscr{S}_{\phi_i}^{\Phi})$. Therefore, the option is now defined over a larger abstracted state space that includes the newly discovered abstraction states.

Target Option Policy ($\pi^{\mathscr{L}}$): The target option policy $\pi_i^{\mathscr{L}} : \mathscr{I}_i^{\mathscr{L}} \rightarrow \mathscr{A}$ must be done in such a way as to satisfy Constraint (5.4). To this end, I use Model-based Least-Squares Policy Iteration Technique [LSPI; Lagoudakis and Parr, 2003] over estimated transition and reward models. The target-option's transition model $\mathscr{P}^{O_i^{\mathscr{L}}}$ has been continually estimated from the $(s^{\Phi}, a, s^{\Phi}_{-})$ samples generated via the exploratory-option's policy π^e_j. To estimate the reward function, the agent uses rewards proportional to the difference of subsequent abstraction activations:

$$r^{O_i^{\mathscr{L}}}(t) = \|\mathbf{y}_i(t) - \mathbf{y}_i(t-1)\| \tag{5.7}$$

$$R^{O_i^{\mathscr{L}}}(s^{\Phi}, a) = (1-\alpha) R^{O_i^{\mathscr{L}}}(s^{\Phi}, a) + \alpha r^{O_i^{\mathscr{L}}}(t), \tag{5.8}$$

where $\mathbf{y}_i(t) = \phi_i\left(\mathscr{U}(\mathscr{P}(s_{-}, \pi^e_j(s^{\Phi}_{-})))\right)$ and $\mathbf{y}_i(t-1) = \phi_i\left(\mathscr{U}(\mathscr{P}(s, \pi^e_j(s^{\Phi})))\right)$. Here, s_{-} and s are the corresponding environment states, $\mathscr{P}$ is the unknown transition-model of the environment, and $0 < \alpha < 1$ is a constant smoothing factor. Once the estimated transition and reward models stabilize, LSPI follows the RL objective and learns a policy $\pi^{\mathscr{L}}_i$ that maximizes the expected cumulative reward over time:

$$\pi^{\mathscr{L}}_i = \arg\max_{\pi} \; \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r^{O_i^{\mathscr{L}}}(t) \middle| \pi, R^{O_i^{\mathscr{L}}} \right], \tag{5.9}$$

where γ is a discount factor close to 1. Therefore, $\pi^{\mathscr{L}}_i$ maximizes the average activation differences, which is equivalent to maximizing variance of the activations [Zhang et al., 2012] approximately[1] satisfying Constraint (5.4).

Termination Condition ($\beta^{\mathscr{L}}$): The option terminates whenever the agent reaches the abstracted state where it observes the maximum reward $\max_{(s,a)} R^{O_i^{\mathscr{L}}}$.

Each target option learned is added to the target-option set $\mathscr{O}^{\mathscr{L}}$ and the learning process iterates until all the learnable exploratory option streams are encoded. Since the expected behavior of Curious Dr. MISFA ensures that the Constraints (5.1-5.3) are satisfied and the learned target-option's policy satisfies Constraint (5.4), the target-option set $\mathscr{O}^{\mathscr{L}}$, at any time t, therefore satisfies the required constraints.

In Section 5.2, I discussed an alternative to Constraint (5.4), where different dimensions of the learned abstraction may be used to learn multiple policies, resulting in a set of *sub-target options*. To keep it simple, I used all dimensions of an abstraction to learn a target-option's policy. However, a sub-target option set can be constructed by following the approach discussed above. Multiple reward functions can be simultaneously estimated from the $(s^{\Phi}, a, s^{\Phi}_{-})$ samples generated via exploratory-option's policy, and the set of sub-target options can be constructed via least-squares policy iteration in parallel.

[1]The error between the true and the estimated target-option policy depends on how well the transition and reward models are estimated based on the samples $(s^{\Phi}, a, s^{\Phi}_{-})$ generated by the exploratory-option's policy.

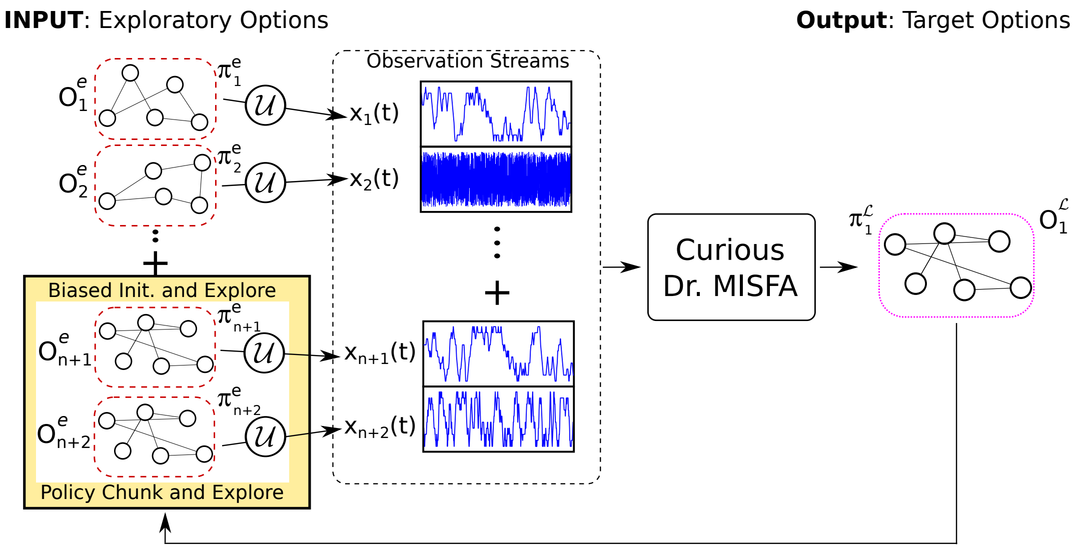


Figure 5.4. Reuse of the learned target options. For each target option learned (represented by pink dotted box), two new exploratory options (*Biased Initialization and Explore* and *Policy Chunk and Explore*) are added to the input exploratory-option (represented by red dashed boxes) set. Biased Initialization and Explore option biases the agent to explore first the state-action tuples where it had previously received maximum intrinsic rewards, while the Policy Chunk and Explore option executes the deterministic target-option's policy before exploration.

5.3.4 Reusing Target Options

To make the skill acquisition open-ended and to acquire more complex skills (see Section 5.2.2), the learned target option $O^{\mathcal{L}}$ can be used to explore the newly discovered abstracted state space. However, a target option may not be reused straightaway, since by definition, it differs from an exploratory option, wherein the target-option's policy is deterministic, while the exploratory-option's policy is stochastic (see Section 5.3.1). *Two* new exploratory options are constructed instead, which are based on the target option $O_i^{\mathcal{L}}$ that was learned last.

In the first option, called **policy chunk and explore**, the initiation-set is the same as that of learned target option $\mathcal{I}_{n+1}^e = \mathcal{I}_i^{\mathcal{L}}$. The policy combines the target-option's policy $\pi_i^{\mathcal{L}}$, which terminates at the state where the variance of subsequent encoded observations is highest, with the LSPI-Exploration policy described in Section 5.3.1. Every time this policy is initiated, the *policy-chunk* (A policy chunk is a non-adaptive frozen policy) $\pi_i^{\mathcal{L}}$ is executed, followed by the LSPI-Exploration policy. This can be beneficial if the target option terminates at a bottleneck state, after which the agent enters a "new world" of experience, within which the LSPI-Exploration policy is useful to explore.

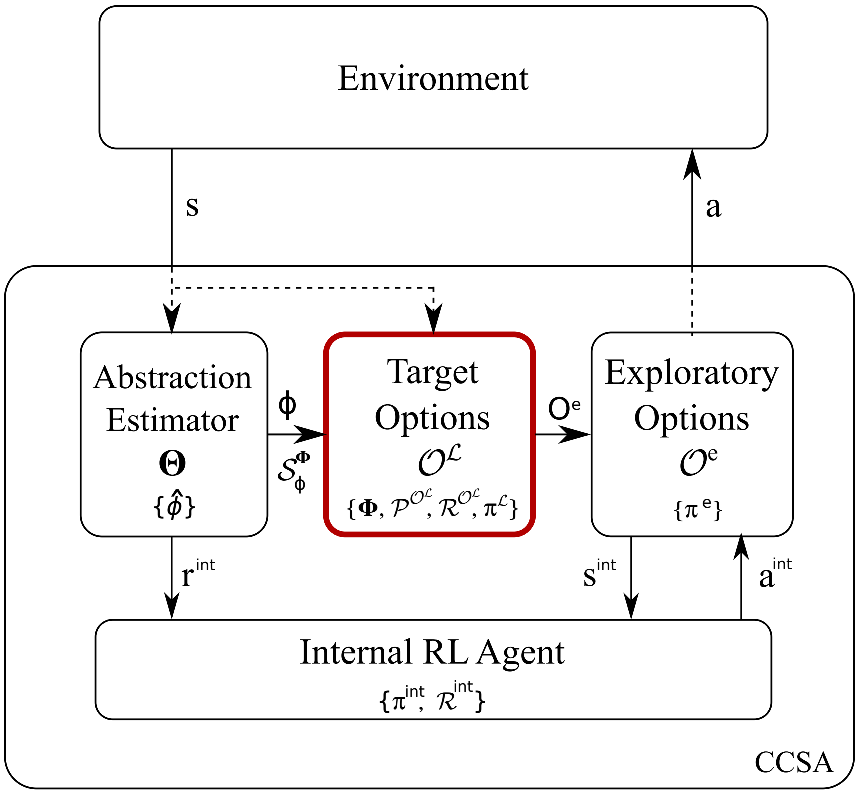

Environment
s
a
Abstraction Estimator
$\boldsymbol{\Theta}$
$\{\hat{\phi}\}$
ϕ
$\mathcal{S}^{\Phi}_{\phi}$
Target Options
$\mathcal{O}^{\mathcal{L}}$
$\{\boldsymbol{\Phi}, \mathcal{P}^{\mathcal{O}^{\mathcal{L}}}, \mathcal{R}^{\mathcal{O}^{\mathcal{L}}}, \pi^{\mathcal{L}}\}$
O^e
Exploratory Options
$\mathcal{O}^e$
$\{\pi^e\}$
r^{int}
s^{int}
a^{int}
Internal RL Agent
$\{\pi^{int}, \mathcal{R}^{int}\}$
CCSA

Figure 5.5. Block diagram of the agent-environment interactions in CCSA. CCSA agent interacts with the environment by taking actions a upon following the current exploratory-option's policy π^e. It makes high-dimensional observations of the environment states s. If previously not encoded, these observations update the adaptive abstraction $\widehat{\phi}$. The updating $\widehat{\phi}$ generates intrinsic rewards r^{int}, which are used to update the internal reward model $\mathscr{R}^{\text{int}}$. Based on $\mathscr{R}^{\text{int}}$, the agent updates its internal policy π^{int}. Using π^{int}, the agent takes an internal action $a^{\text{int}} = \{\text{stay}, \text{switch}\}$ and observes the next internal state s^{int}. The exploratory option corresponding to s^{int} is executed for the next iteration. The process continues until $\widehat{\phi}$'s estimation error drops below a threshold, at which point the abstraction is saved. The agent's abstracted state space is augmented with the new feature states $\mathscr{S}^{\Phi}_{\phi}$. The high-dimensional state observations are now used to update the transition $\mathscr{P}^{O^{\mathscr{L}}}$ and reward $\mathscr{R}^{O^{\mathscr{L}}}$ models of the new abstracted state space. When the models stabilize, a policy $\pi^{\mathscr{L}}$ is learned. With the learned abstraction ϕ and the learned policy $\pi^{\mathscr{L}}$, a target option is constructed and saved. Based on the saved target option, two new exploratory options O^e are constructed and added to the exploratory option set. The entire process iterates with the new input exploratory option set.

Algorithm 9: Int-Policy-Update ($\mathbf{x}$)

```
   // Curious Dr. MISFA Internal Policy Update
1  Abstraction-Learned ← False    // Abstraction learned or not
2  $\phi \leftarrow$ Gating-System($\mathbf{x}$)    //Get the assigned abstraction
3  $\xi_{t+1} = \|\Theta(\mathbf{x}, \phi) - \phi\|$    //Estimation Error
4  if $\langle \xi_{t+1} \rangle_\tau > \delta$ then
5      $\hat{\phi} \leftarrow \Theta(\mathbf{x}, \hat{\phi})$    //Update the adaptive-abstraction
6      if $\langle \|\Theta(\mathbf{x}, \hat{\phi}) - \hat{\phi}\| \rangle_\tau < \delta$ then
7          $\Phi_{t+1} \leftarrow \Phi_t \cup \hat{\phi}$    // Update abstraction set
8          Abstraction-Learned ← True
9      end
10 end
11 $R^{\text{int}}_{t+1} \leftarrow$ UpdateReward ($\dot{\xi}_{t+1}$) //Update the intr. reward func.

12 $\pi^{\text{int}}_{t+1} \leftarrow$ Model-LSPI ($\mathscr{P}^{\text{int}}$,$R^{\text{int}}_{t+1}$)    //Update intr. policy
13 $\pi^{\text{int}}_{t+1} \leftarrow \epsilon$-greedy ($\pi^{\text{int}}_{t+1}$) //Exploration-exploitation tradeoff
14 return ($\pi^{\text{int}}_{t+1}$, Abstraction-Learned)
```

In the second option, called **biased initialization and explore**, the exploratory-option's policy uses the normalized *value function* of the target option as an *initial reward function estimate*. This initialization biases the agent to explore the state-action tuples first where it had previously received maximum intrinsic rewards. Otherwise it is the same as the standard initial error-seeking LSPI-Exploration policy.

For each target option learned, these two exploratory options are added to the input exploratory-option set. In this way, the agent continues the process of curiosity-based skill acquisition by exploring among the new exploratory option set to discover unknown regularities. A complex skill $O^{\mathscr{L}}_k = \langle \mathscr{I}^{\mathscr{L}}_k, \beta^{\mathscr{L}}_k, \phi_k, \pi^{\mathscr{L}}_k \rangle$ can be learned as a consequence of chaining multiple skills that were learned earlier.

5.4 Pseudocode

The entire learning process involves determining three policies (see Figure 5.5):

1. π^e: Exploratory-option's stochastic policy that is determined (see Section 5.3.1) to generate high-dimensional observations.

2. π^{int}: An internal policy that is learned (see Section 5.3.2) to determine for which exploratory option O^e to encode a slow feature abstraction.

Algorithm 10: Continual Curiosity-Driven Skill Acquisition (CCSA)

```
1  $\Phi_0 \leftarrow \{\}$, $\pi_0 \leftarrow$ Random (), $\hat{\phi} \leftarrow 0$, Abstraction-Learned $\leftarrow$ False
2  for $t \leftarrow 0$ to $\infty$ do
3      $s^{\text{int}} \leftarrow$ current internal state, $a^{\text{int}} \leftarrow$ action selected by $\pi^{\text{int}}_t$ in state $s^{\text{int}}$
4      Take action $a^{\text{int}}$, observe next internal state $s^{\text{int}}_{-}(= i)$
       // Execute the exploratory option $O^e_i$
5      while not $\beta^e_i(t)$ do
6          $s^{\Phi} \leftarrow$ current subjective state, $a \leftarrow$ action selected by $\pi^e_i$ in state $s^{\Phi}$
7          Take action $a$, observe next subjective state $s^{\Phi}_{-}$ and the sample $\mathbf{x}$
8          if not Abstraction-Learned then
               // Internal Policy Update
9              $(\pi^{\text{int}}_{t+1}$, Abstraction-Learned) = Int-Policy-Update ($\mathbf{x}$)
10         else
               // Learn target option
11             $\pi^{\text{int}}_{t+1} \leftarrow \pi^{\text{int}}_t$, $R_{\text{prev}} \leftarrow R^{O^{\mathcal{L}}}$, $P_{\text{prev}} \leftarrow P^{O^{\mathcal{L}}}$
12             $R^{O^{\mathcal{L}}}(s^{\Phi}, a) = (1-\alpha)R^{O^{\mathcal{L}}}(s^{\Phi}, a) + \alpha(\|\mathbf{y}_{\text{i}}(t) - \mathbf{y}_{\text{i}}(t-1)\|)$
13             $P^{O^{\mathcal{L}}}(s^{\Phi}, a, s^{\Phi}_{-}) = (1-\alpha)P^{O^{\mathcal{L}}}(s^{\Phi}, a, s^{\Phi}_{-}) + \alpha$
14             if $(\|R^{O^{\mathcal{L}}} - R_{prev}\| < \delta$ and $\|P^{O^{\mathcal{L}}} - P_{prev}\| < \delta)$ then
                   // Get the target-option policy
15                 $\pi^{\mathcal{L}} \leftarrow$ LSPI-Model($P^{O^{\mathcal{L}}}, R^{O^{\mathcal{L}}}$)
                   // Construct target option
16                 $O^{\mathcal{L}} = \langle \mathcal{I}^{\mathcal{L}}, \beta^{\mathcal{L}}, \hat{\phi}, \pi^{\mathcal{L}} \rangle$
                   // Add to target-option set
17                 $\mathcal{O}^{\mathcal{L}} \leftarrow \mathcal{O}^{\mathcal{L}} \cup O^{\mathcal{L}}$
                   // Construct two new exploratory options
18                 $\mathcal{O}^e \leftarrow \mathcal{O}^e \cup$ Biased-Init-Explore($O^{\mathcal{L}}$)
19                 $\mathcal{O}^e \leftarrow \mathcal{O}^e \cup$ Policy-Chunk-Explore($O^{\mathcal{L}}$)
                   // Reset
20                 $\hat{\phi} \leftarrow 0$, Abstraction-Learned $\leftarrow$ False
21             end
22         end
23     end
24 end
```

3. $\pi^{\mathscr{L}}$: Target-option's deterministic policy that is learned (see Section 5.3.3) to maximize variation in the slow feature abstraction output.

The resultant target options (skills) are stored and reused as discussed above to facilitate open-ended continual learning. Algorithms 9 and 10 summarize the entire learning process.[2]

5.5 Experimental Results

I present here experimental results that focus on continual-learning of skills using an iCub humanoid platform. The results here are the **first** in which a humanoid robot such as an iCub, learns a repertoire of skills from *raw-pixel data* in an *online* manner, driven by its own *curiosity*, starting with low-level joint kinematic maps.[3]

Learning a skillset largely depends on the environment that the robot is in. For the sake of developing specific types of skills such as toppling an object, grasping, etc., a safe environment is pre-selected for the iCub to explore, yet the iCub is mostly unaware of the environment properties.

Environment: An iCub robot is placed next to a table, with an object (a plastic cup) in reach of its right arm and within its field-of-view (Figure 5.6(a)). The cup topples over upon contact, and the resulting images after toppling are predictable. There is a human experimenter present, who monitors the robot's safety and replaces the cup in its original position after it is toppled. The iCub does not "know" that the plastic-cup and the experimenter exist. It continually observes the gray-scale pixel values from the high-dimensional images (75×100) captured by the left and right camera eyes (Figure 5.6(b)). In addition to the experimenter and the cup, it also cannot recognize its own moving hand in the incoming image stream, as shown in the Figure 5.6(b).

Task-Relevant Roadmap Exploration is not performed at the level of joint angles due to the complexity of the robot's joint space. Instead, the robot is given a map of poses *a priori*. This compressed actuator joint space representation is called a Task-Relevant Roadmap [TRM; Stollenga et al., 2013]. This map contains a family of iCub postures that adhere to relevant constraints. The TRM is grown offline by repeatedly optimizing cost functions that represent the constraints using a Natural Evolution Strategies [NES; Wierstra et al., 2008] algorithm, such that the task space is covered. This allows us to deal with complex cost functions and the full 41

[2] A *Python*-based implementation of related code excerpts can be found at the URL: www.idsia.ch/~kompella/codes/.

[3] A video for these experiment can be found at URL: http://www.youtube.com/watch?v=OTqdXbTEZpE

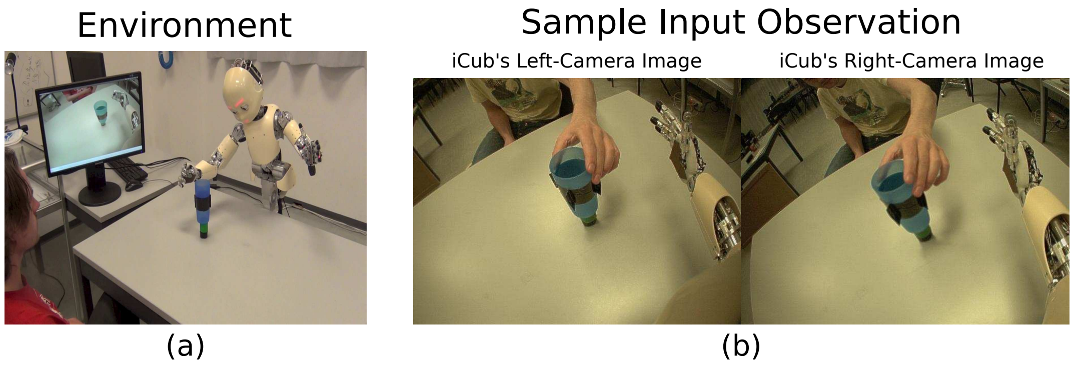


Figure 5.6. (a) An iCub robot is placed next to a table, with an object (a plastic cup) in reach of its right arm and within its field-of-view. (b) Sample input images captured from both left and right iCub camera-eyes are an input to the algorithm.

degrees-of-freedom of the iCub's upper body. The constraints used: (a) the iCub's hand is positioned on a 2D plane parallel to the table while keeping its palm oriented horizontally; (b) the left hand is kept within a certain region to keep it out of the way; and (c) the head is pointed towards the table. The task-space of the TRM comprises the x and y position of the hand, which forms the initial discretized 10×5 abstracted state space $\mathscr{S}^{\Phi} = \mathscr{S}_x^{\Phi} \times \mathscr{S}_y^{\Phi}$. The action space contains 6 actions: *North*, *East*, *South*, *West*, *Hand-close* and *Hand-open*.

Because the full body is used, the movements look more dynamic, but as a consequence, the head moves around and looks at the table from different directions, making the task a bit more difficult. Even so, IncSFA still finds the resulting regularities in the raw camera input stream, and the skill learner continues to learn upon these regularities without any external rewards.

Experiment parameters: I used a fixed parameter setting for the entire experiment.

IncSFA Algorithm: For CCIPCA I use the learning rate $1/t$ with amnesic parameter 0.4 while the MCA learning rate is set to 0.01. CCIPCA does variable size dimension reduction by calculating how many eigenvalues would be needed to keep 99% of the input variance — typically this was between $5-10$ — so the 7500 pixels could effectively be reduced to only about 10 dimensions. The output dimension is set to 1, so only the first IncSFA feature is used as an abstraction. However, more number of features can be used if desired.

Robust Online Clustering (ROC) Algorithm: Each clustering implementation has its maximum number of clusters set to $N^{max} = 3$ so that it can encode multiple slow feature values for each abstracted state (see Section 4.3.2). Higher values can be used, though very high values may lead to spurious clusters. The estimation error threshold, below which the current module is saved and a new module is created, is set to a $\delta = 0.3$. The amnesic parameter is set to $\beta^{amn} = 0.01$. Higher values will make ROC adapt faster to the new data, but at the cost of being less stable.

Curious Dr. MISFA's Internal Reinforcement Learner: To balance between exploration and exploitation, ϵ-greedy strategy is used (see Section 4.3.5). The initial ϵ-greedy value is set to 1.0 (1.0 for pure exploration, 0.0 for pure exploitation), with a 0.995 decay multiplier. The window-averaging time constant is set to $\tau = 20$, that is, 20 sample images are used to compute the window-averaged progress error ξ and the corresponding curiosity-reward (see Section 4.3.4).

Target-option's Reinforcement Learner: Slow features abstractions have unit-variance and are typically in the range of $(-1.5, 1.5)$. Each abstraction's output value is discretized to $-1, 1$, i.e., into two abstracted states.

Experiment Initialization: The iCub's abstracted state space ($\mathscr{S}^{\Phi}$) at $t = 0$ is a 10×5 grid found using TRM. To minimize human input into the system, the input exploratory-option set ($\mathscr{O}^{e}$) has only one exploratory option to begin with (as defined in Section 5.3.1): $\mathscr{O}^{e} = \{O^{e}_{1}\}$, which is a random walk in the iCub's abstracted state space. However, one may pre-define multiple input exploratory options, which could lead to a different result. The exploratory option terminates after $\tau = 20$ time steps since its execution. The internal state space at $t = 0$ is $\mathscr{S}^{\text{int}} = \{s^{\text{int}}_{1}\}$, where s^{int}_{1} corresponds to the exploratory option O^{e}_{1}. The plastic cup is roughly placed around $(2, 2)$ grid-point on the table.

5.5.1 iCub Learns to Topple the Cup

The iCub starts the experiment without any learned modules, so the exploratory-option's policy π^{e}_{1} is a random walk over the abstracted state space $\mathscr{S}^{\Phi}$. It explores by taking one of the six actions: *North*, *East*, *South*, *West*, *Hand-close* and *Hand-open* and grabs high-dimensional images from its camera eyes. The exploration causes the outstretched hand to eventually displace or topple the plastic-cup placed on the table. It continues to explore and after an arbitrary amount of time-steps the experimenter replaces the cup to its original position. After every τ time steps the currently executing option terminates. Since there is only one exploratory option, the iCub re-executes the same option. Figure 5.7(a) shows a sample input image stream of only the left camera.[4]

Figure 5.7(b) shows the developing IncSFA output over the algorithm execution time, since the IncSFA abstraction was created. The outcome of IncSFA abstraction learning is a step-like function, which when discretized, indicates the pose of the cup (toppled vs non-toppled). Figure 5.7(c) shows the ROC estimation error (blue solid

[4]We, however, used both the left and right camera images as an input observation by concatenating them.

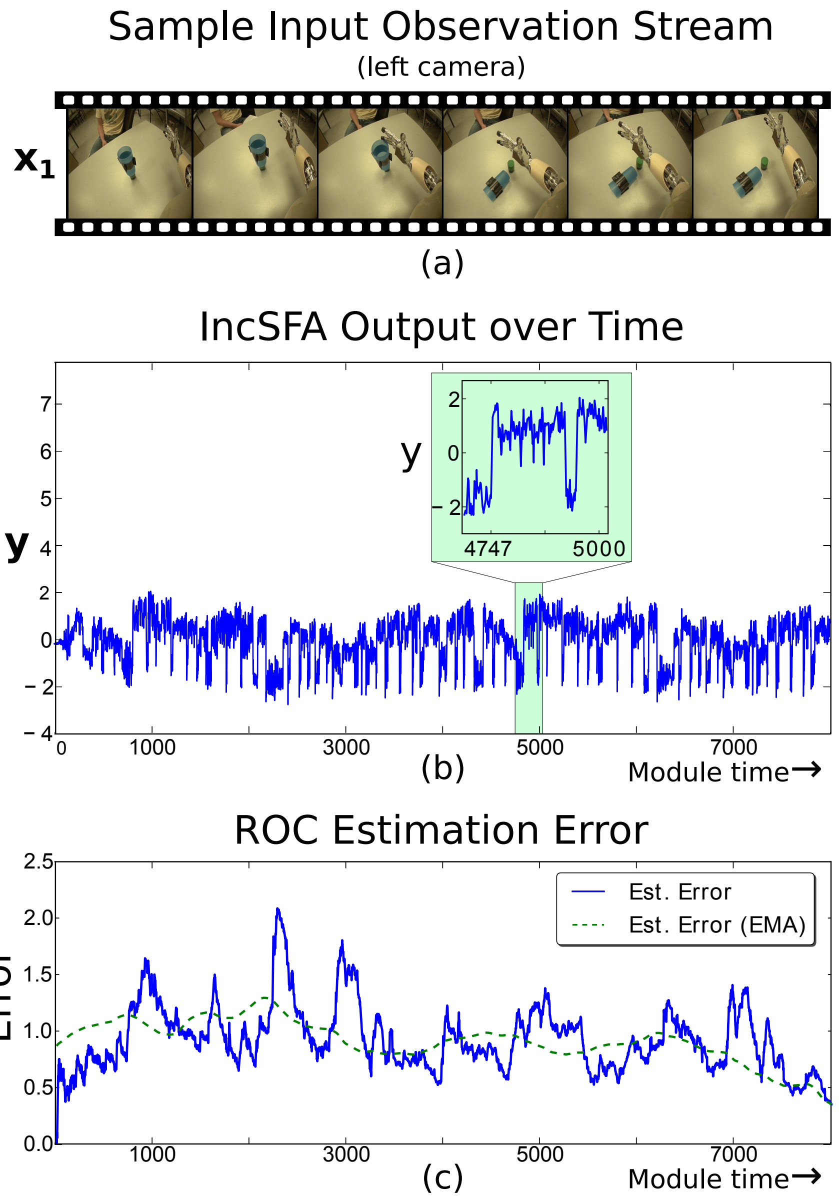


Figure 5.7. (a) A sample image stream of the iCub's left-eye camera showing the topple event. (b) Developing IncSFA abstraction output over algorithm execution time, since it was created. The result is a step-like function encoding the topple event. (c) ROC estimation error over algorithm execution time. The estimation error eventually drops below the threshold ($\delta = 0.3$), after which the abstraction is saved.

line) and an Expected Moving Average (EMA) of the error (green dashed line) over the algorithm execution time. As the process continues, the error eventually drops below the threshold $\delta = 0.3$ and the abstraction module ϕ_1 is saved. Figure 5.8(a)

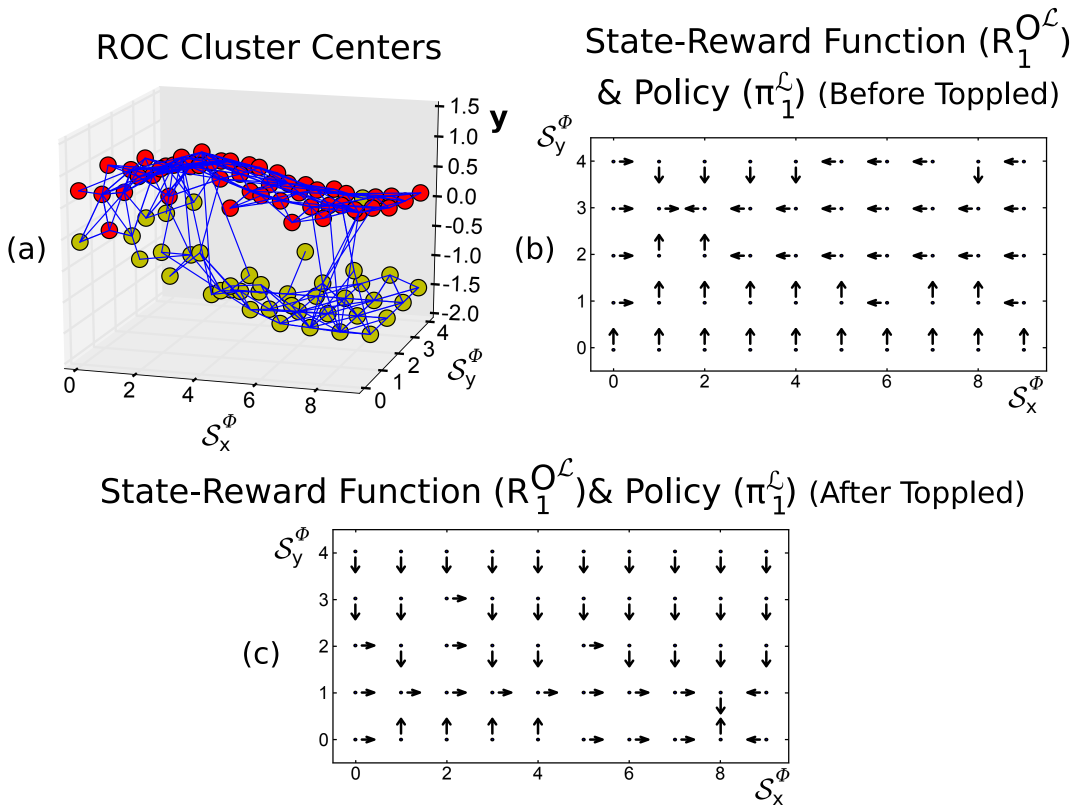


Figure 5.8. (a) The resultant ROC cluster centers, which map the abstraction outputs to the abstracted state space (in this case the X and Y grid locations of the iCub's hand). Red and yellow colors indicate the discretized feature states $\mathscr{S}^{\Phi}_{\phi_1}$. Blue lines connecting the cluster centers illustrate the learned transition model of the new abstracted state space. (b) Part of the learned target-option's policy before the cup is toppled. The arrows indicate the optimal action to be taken at each grid-location (s^{Φ}_x, s^{Φ}_y) of the iCub's hand. They direct the iCub's hand to the grid point $(1,3)$, which will make the iCub topple the cup placed at $(2,2)$. (c) Part of the learned target-option's policy after the cup is toppled. They direct the iCub's hand to move to the right. This is a result of the experimenter replacing the cup only when the iCub has moved its hand away from the $(2,2)$ grid location.

shows the ROC cluster centers that map the feature outputs $(\mathbf{y})$ to each of the 10×5 abstracted states. There are two well separated clusters each representing the state of the plastic-cup.

Immediately after the abstraction is saved, the cluster centers are discretized (Red and yellow colors indicate the discretized feature states $\mathscr{S}^{\Phi}_{\phi_1}$ in Figure 5.8(a)), the transition model (represented by the blue lines in Figure 5.8(a)) and reward model of $O_1^{\mathscr{L}}$ are learned followed by a corresponding target-option's policy $\pi_1^{\mathscr{L}}$

as discussed in Section 5.3.3. Figure 5.8(b) shows a part of the learned policy $\pi_1^{\mathscr{L}}$ before the cup is toppled. The arrows indicate the optimal action to be taken at each grid-location of the iCub's hand. They direct the iCub's hand to the grid point $(1,3)$, which will make the iCub topple the cup placed at $(2,2)$. Figure 5.8(c) shows the part of the policy after the cup has been toppled. The policy directs the iCub's hand to move towards *east*. This is because, during the experiment the experimenter happened to replace the cup only when the iCub's hand is around far east. I label the learned target option $O_1^{\mathscr{L}}$, for the given environment, as a "Topple" skill.

5.5.2 iCub Learns to Grasp the Cup

The iCub continues its learning process by reusing the learned topple skill to construct two additional exploratory options as discussed in Section 5.3.4. One in which the topple policy (Figure 5.8(b)) is executed prior to the LSPI-Exploration policy and the other, where the normalized value function (Figure 5.9(b)) is used to initialize the reward function of the LSPI-Explorer. Let O_2^e and O_3^e denote these two exploratory options respectively. Therefore, including the original exploratory option O_1^e, a total of 3 exploratory options are an input to CCSA.

The system initially explores by executing each of the options until termination, i.e., after τ time steps. When it selects either O_1^e or O_2^e, the cup gets toppled in the process (Figure 5.9(a)-Top) and since there already exists a learned abstraction ϕ_1 that encodes the toppling outcome, it receives no internal reward for executing these options because of the gating system (see Section 4.3.6). This is also the case in the beginning while executing O_2^e because the LSPI-Exploration policy initially causes the iCub to topple the cup yielding no rewards. The initialized values corresponding to the visited state-action tuples soon vanish and the iCub then explores the neighboring state action pairs. Eventually, as a result of the biased exploration, in a few algorithm iterations the iCub ends up grasping the cup (Figure 5.9(a)-Bottom). This gives rise to a high estimation error because of the novelty of the event (Figure 5.9(c)). Figures 5.9(d)-(i) show the state-action LSPI-Exploration reward function after a few time steps. The hand-close action at $(2,2)$ generates the most novel event. This results in a LSPI-Exploration policy that increases the number of successful grasp trials (77 out of 91 total attempts, with most of the unsuccessful trials in the beginning) when the exploratory option O_3^e is executed.

Upon executing option O_3^e, the adaptive abstraction $\hat{\phi}$ begins to make progress by encoding samples corresponding to the observation stream $\mathbf{x_3}$. After a few algorithm iterations, the agent finds that the action *stay* at the internal state s_3^{int} corresponding to the O_3^e is rewarding due to the progress made by IncSFA and the ROC estimator (Figure 5.10(a)). Figure 5.10(b) shows the normalised internal reward function of

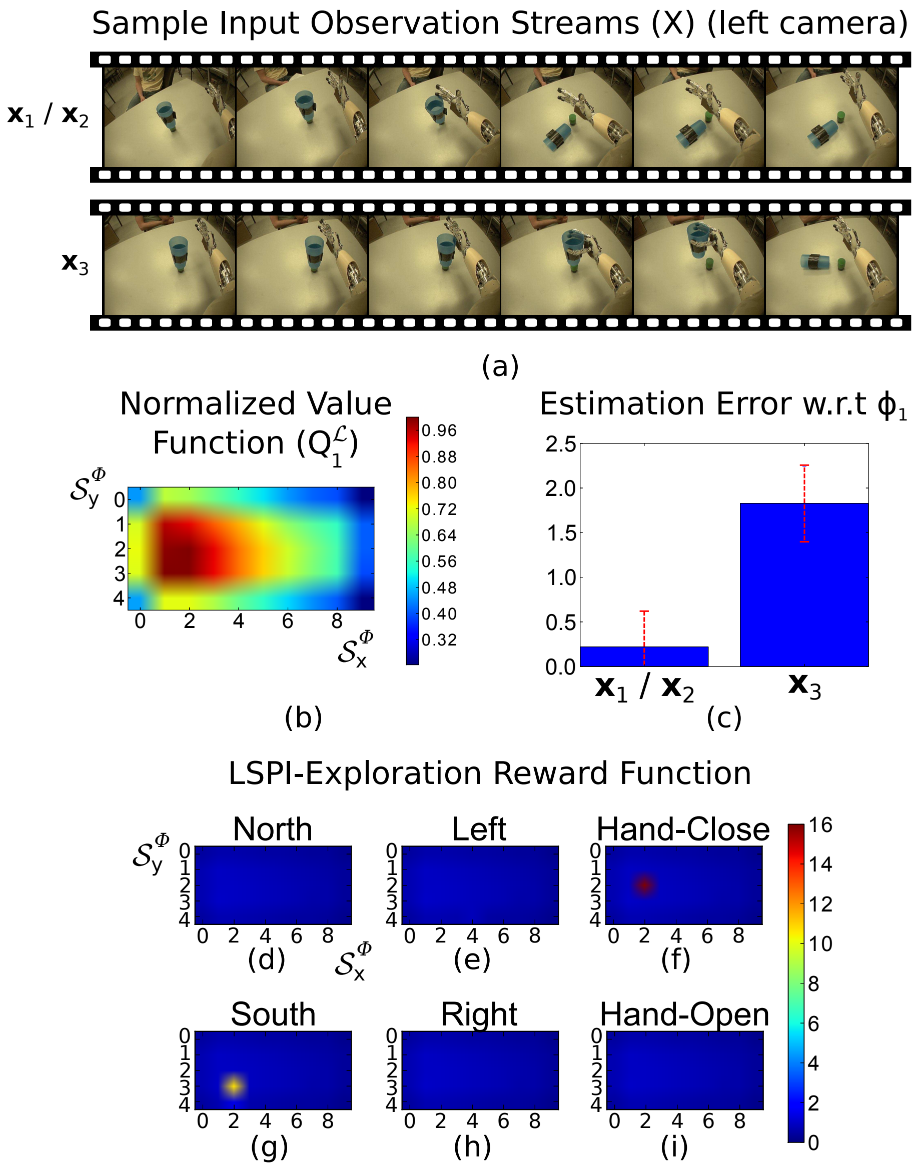


Figure 5.9. (a) Sample iCub's left-eye camera images corresponding to the three input exploratory options. $\mathbf{x}_1$ and $\mathbf{x}_2$ correspond to the original and the *policy chunk & explore* exploratory option respectively, while $\mathbf{x}_3$ corresponds to the *biased init. & explore* exploratory option. (b) Normalized value function of the previously learned target option (topple). It is used for reward-initialization in the *biased init. & explore* exploratory option. (c) Estimation error of the learned topple abstraction module (ϕ_1) for each of the three observation-streams. (d)-(i) LSPI-Exploration reward function estimated using the novelty (& curiosity) signal. The Hand-Close action at $(2,2)$ has the maximum reward value due to the novel grasp event.

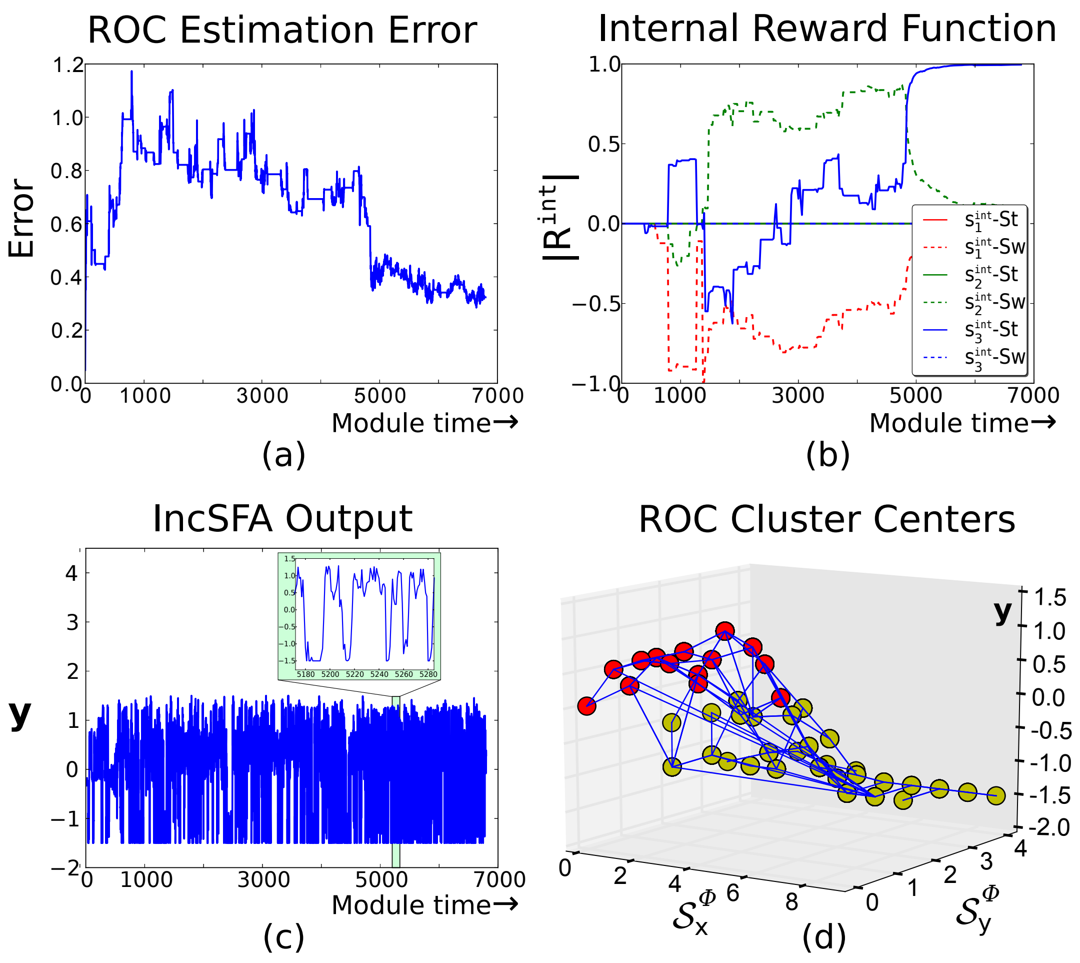


Figure 5.10. (a) ROC estimation error of the current adaptive-module that is encoding the new regularities. (b) Normalized internal reward function of Curious Dr. MISFA. The action *stay* in the state corresponding to the exploratory option 3 (shown as s_3^{int}-St) is most rewarding due to the learning progress made by the IncSFA-ROC module for the grasp-event. (c) IncSFA output over execution time, since it was created. (d) Resultant ROC cluster centers mapping the IncSFA output w.r.t. the abstracted state space. Note that the abstracted states corresponding to the learned topple abstraction $\mathscr{S}_{\phi_1}^{\Phi}$ are not shown here, since the grasp abstraction outputs are uncorrelated to the topple abstraction and it is difficult to illustrate a 4-D plot. Red and yellow colors indicate the discretized states $\mathscr{S}_{\phi_2}^{\Phi}$ and the blue lines illustrate the learned transition model.

Curious Dr. MISFA over algorithm iterations, since the new adaptive module was created. The internal policy π^{int} quickly converges to select and execute the option O_3^e to receive more observations. When the estimation error drops below the thresh-

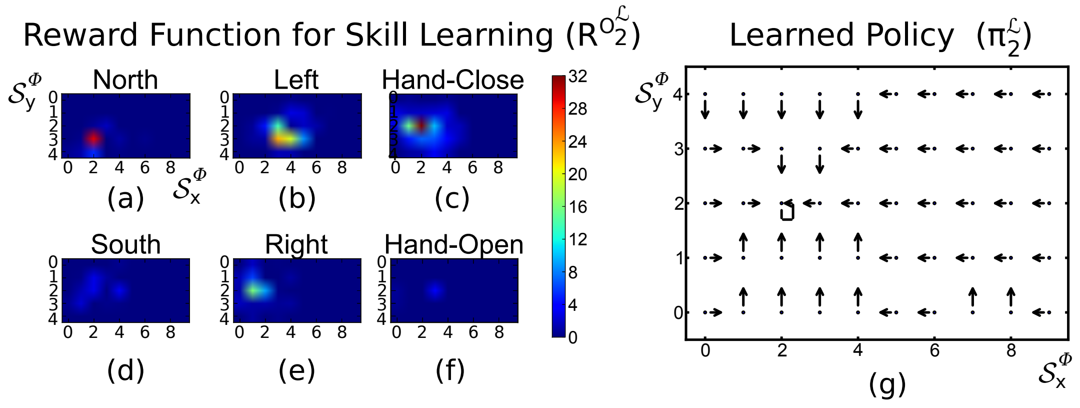


Figure 5.11. (a)-(f) Estimated reward function of the new abstracted state space that is used to learn the target-option's policy. The hand-close action at $(2,2)$ receives the maximum reward as it produces a maximum variation in the slow feature output (from ≈ -1.5 to 1.5). (g) Learned target-option's policy representing the grasp skill. The arrows indicate the optimal actions to be taken at each grid-location (s_x^{Φ}, s_y^{Φ}). The circular arrow represents the hand-close action. The policy directs the iCub's hand to move to $(2,2)$ and then to close its hand, which should result in a successful grasp.

old ($\delta = 0.3$), it saves the module $\phi_2 = \hat{\phi}$. Figure 5.10(c) shows the IncSFA output over the time since the new module was created. Figure 5.10(d) shows the learned cluster centers mapping the slow feature output to the abstracted state space. Note that the abstracted states corresponding to the learned topple abstraction $\mathcal{S}_{\phi_1}^{\Phi}$ and not are shown in Figure 5.10(d), because the grasp abstraction outputs are uncorrelated to the topple abstraction and it is difficult to illustrate a 4-D plot. The iCub then begins to learn the target policy $\pi_2^{\mathcal{L}}$ by learning the target-option's transition and reward model. Figure 5.11(a)-(f) show the target-option's state-action reward model developed after 8000 observation samples (module time=8000). And finally, Figure 5.11(g) shows the corresponding skill learned, *i.e.*, to perform a Hand-Close at $(2,2)$ (the anti clockwise circular arrow represents the Hand Close action).

This experiment demonstrated how the iCub reused the knowledge gained by the topple skill to learn a subsequent skill labeled as "Grasp". The grasp skill includes an abstraction to represent whether the cup has been successfully grasped-or-not and a policy that directs the iCub's hand to move to $(2,2)$ and then to close its hand.

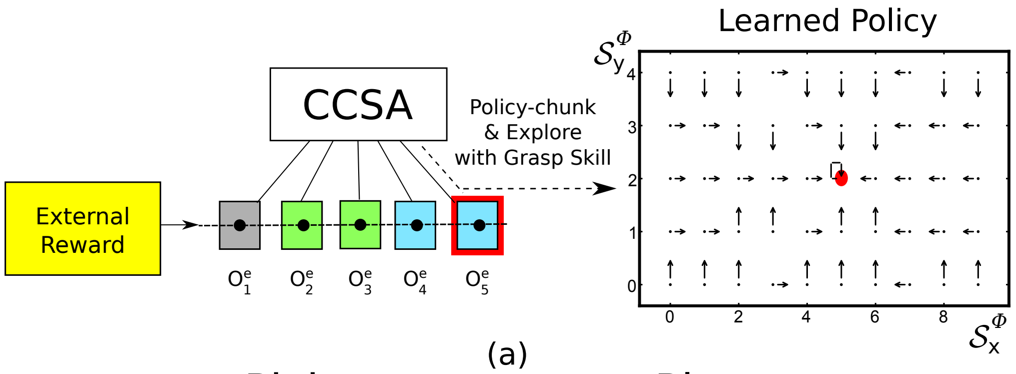
5.5.3 iCub Learns to Pick and Place the Cup at the Desired Location

Here, an experiment is conducted to demonstrate the utility of intrinsic motivation in solving a subsequent external objective. The iCub is in a similar environment as discussed above. However, it is given an external reward if it picks the plastic cup and places (drops) it at a desired location (at any of the following grid locations (s_x^Φ, s_y^Φ): $(6,2), (6,3), (6,1), (5,2), (7,2)$). The agent with no intrinsic motivation finds the reward almost inaccessible via random exploration over its abstracted state space $\mathscr{S}^\Phi$, because the probability of a successful trial is low.[5] ($\approx 10^{-5}$) However, a curiosity-driven iCub greatly improves this by learning to pick/grasp the cup by itself and then reusing the skill to access the reward.

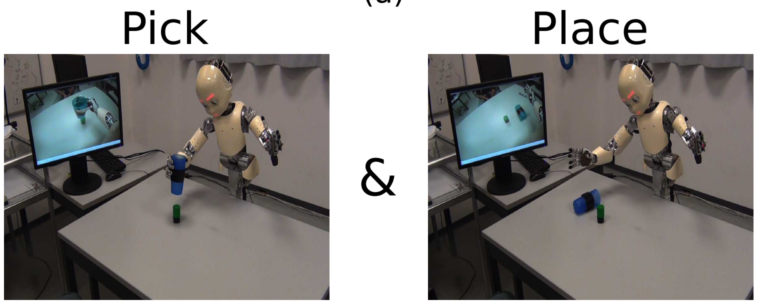
Starting from the 10×5 abstracted state space found via TRM, the iCub learns to topple and then grasp as discussed in the previous sections. The process continues and it adds two more exploratory options $(\mathcal{O}_4^e, \mathcal{O}_5^e)$ corresponding to the grasp skill as discussed in Section 5.3.4. The biased initialization and explore option $\mathcal{O}_4^e$ results in the iCub dropping the cup close to where it has picked it up. Since it doesn't get any reward in this case, the initialized values to the visited state-actions tuples vanish and it explores the neighboring state-action tuples. This option will take a long time before it can execute the desired state-action tuple to drop the cup. The policy chunk and explore option $\mathcal{O}_5^e$, however, first executes the grasp policy and then randomly explores until it receives some novelty or curiosity reward. When it drops the cup in one of the desired states while exploring, it gets an external reward. This results in a LSPI-Exploration policy that executes the rewarding behavior. Curious Dr. MISFA eventually finds the internal action *stay* at the internal-state s_5^{int} corresponding to the option $\mathcal{O}_5^e$ most rewarding. As soon as the experimenter replaces the cup, the iCub repeats the pick and place behavior until the external reward is removed.

This experiment demonstrated how CCSA enabled the iCub to reuse the previously learned grasp skill to learn to pick and place the cup at a desired location. Note that in the experiments, a human experimenter unknown to the robot acted as a part of the environment to speed up the learning process. Without the experimenter the robot might not have acquired the same set of skills, it might have learned to push the object instead [Kompella et al., 2012b].

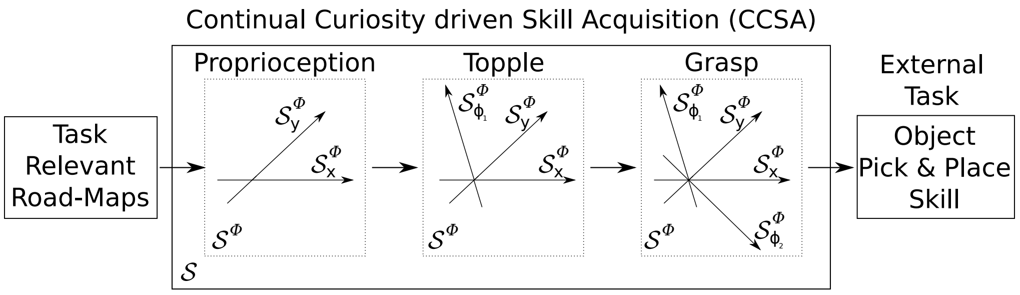
[5]The probability of a successful pick = 1/300, probability of a drop given a successful pick = 1/300 * 1/60.


CCSA
External Reward
Policy-chunk & Explore with Grasp Skill
O_1^e O_2^e O_3^e O_4^e O_5^e
Learned Policy
$\mathcal{S}_y^{\Phi}$
$\mathcal{S}_x^{\Phi}$

(a)



(b)


Continual Curiosity driven Skill Acquisition (CCSA)
Task Relevant Road-Maps
Proprioception
Topple
Grasp
External Task
Object Pick & Place Skill
$\mathcal{S}_y^{\Phi}$ $\mathcal{S}_x^{\Phi}$ $\mathcal{S}^{\Phi}$ $\mathcal{S}_{\phi_1}^{\Phi}$ $\mathcal{S}_{\phi_2}^{\Phi}$ $\mathcal{S}$

(c)

Figure 5.12. (a) CCSA now has 5 exploratory options as an input. Among the 5 options, only the *policy chunk & explore* corresponding to the grasp skill makes it easier for the iCub to access the external-reward present for placing the cup at the desired grid locations. This results in a policy – to place the cup in the desired location (the clockwise circular arrow represents the Hand-Open action). (b) Bird's eye view of the iCub demonstrating the pick & place skill. (b) Figure shows the increasing dimensions in the agent's abstracted state space with every new abstraction learned. This experiment demonstrates how CCSA enables the iCub to reuse the grasp skill, which was previously learned via intrinsic motivation, on learning to pick & place the cup to a desired location.

5.6 Conclusion

I proposed an online learning algorithm that enables a humanoid robotic agent such as an iCub to incrementally acquire skills in order of increasing learning difficulty, from its onboard high-dimensional camera inputs and low-level kinematic joint maps, driven purely by its intrinsic motivation. The method combines the active modular Curious Dr. MISFA algorithm and the *options* framework. I formally defined the underlying learning problem and provided experimental results conducted using an iCub humanoid robot to topple, grasp and pick-place a cup. To my knowledge, this is the **first** method that demonstrates continual curiosity-driven skill acquisition from high-dimensional video inputs in humanoid robots.

Chapter 6

Discussion and Conclusion

This chapter discusses the results from developing the CCSA framework in the context of related research carried out by other researchers prior to this thesis. The current limitations of the CCSA framework are presented and insights for future work are provided along with a final summary of the contributions made in the thesis.

6.1 Related Work on Intrinsically Motivated Autonomous Skill Acquisition

Existing intrinsically-motivated skill acquisition techniques in RL have often been been applied to simple domains. For example, Bakker and Schmidhuber [2004] proposed a hierarchical RL framework called HASSLE in a grid world environment, where high-level policies discover subgoals from clustering distance-sensor outputs and low-level policies specialize on reaching the subgoals. Stout and Barto [2010] explore the use of a competence-based IM as a developmental model for skill acquisition in simple artificial grid-world domains. Pape et al. [2012] proposed a method for autonomous acquisition of tactile skills on a biomimetic robot finger, through curiosity-driven reinforcement learning.

There have been attempts to find skills using feature-abstractions in domains such as those of humanoid robotics. Hart [2009] proposed an intrinsically motivated hierarchical skill acquisition approach for a humanoid robot. The system combines a *discrete event dynamical system* [Huber and Grupen, 1996] as a control basis and an intrinsic reward function [Hart et al., 2008] to learn a set of controllers. However, the intrinsic reward function used is task specific and the system requires a teacher to design a developmental schedule for the robot.

Konidaris et al. [2009; 2010] show how each option might be assigned with an abstraction from a library of many sensorimotor abstractions to acquire skills. The abstractions have typically been hand-designed and learning was assisted by human-demonstration. In their recent work [Konidaris et al., 2011], an intrinsic motivation system makes a robot acquire skills from one task to improve the performance on a second task. However, the robot used augmented reality tags to identify target objects and had access to a pre-existing abstraction library. CCSA autonomously learns a library of abstractions and control policies simultaneously from raw-pixel streams generated via exploration, without any prior-knowledge of the environment.

Mugan and Kuipers [2012] Qualitative Learner of Action and Perception system discretizes low-level sensorimotor experience through defining landmarks in the variables and observing contingencies between landmarks. It builds predictive models on this low-level experience, which it later uses to generate plans of action. It either selects its actions randomly (early) or such that it expects to make fast progress in the performance of the predictive models (artificial curiosity). The sensory channels are preprocessed so that the input variables, for example, track the positions of the objects in the scene. A major difference between this system and CCSA is that CCSA operates upon the raw pixels directly, instead of assuming the existence of a low-level sensory model that can track the positions of the objects in the scene.

Baranes and Oudeyer [2013] proposed an intrinsic motivation architecture called SAGG-RIAC, for adaptive goal-exploration. The system comprises two learning parts, one for self-generation of subgoals within the task-space and the other for exploration of low-level actions to reach the subgoals selected. The subgoals are generated using heuristics methods based on a local measure of *competence progress*. The authors show results using a simulated quadruped robot on reaching tasks. The system assumes, however, that a low-dimensional task space is provided. CCSA on the other hand is a task-independent approach, where subgoals are generated automatically by the slow-feature abstractions that encode spatio-temporal regularities in the raw high-dimensional video inputs.

Ngo et al. [2012; 2013] investigated an autonomous learning system that utilizes a progress-based curiosity drive to ground a given abstract action, e.g., placing an object. The general framework is formulated as a selective sampling problem in which an agent samples any action in its current situation as soon as it sees that the effects of this action are *statistically* unknown. If no available actions have a statistically unknown outcome, the agent generates a plan of actions to reach a new setting where it expects to find such an action. Experiments were conducted using a Katana robot arm with a fixed overhead camera, on a block-manipulation task. The authors show that the proposed method generates sample-efficient curious exploratory behavior and continual skill acquisition. However, unlike CCSA, the sensorimotor abstractions are hand-designed and not learned by the agent.

Schmidhuber [2013] formulated an algorithm called PowerPlay, which can be viewed as a greedy variant of the Formal Theory of Creativity. In PowerPlay, an increasingly general problem solver is improved by searching for the easiest to solve, still not yet known task, while ensuring all previously solved tasks remain solved. PowerPlay, unlike most online-learning algorithms has no problems with forgetting. However, using PowerPlay for high-dimensional video data is a hard non-trivial problem. Similar to PowerPlay, in CCSA when a new representation is learned well enough to be internally predictable (low feature output estimator error), it is frozen and added to a long-term memory storage, and therefore already learned representations are not lost. Additionally, CCSA can learn representations from high-dimensional raw video data.

CCSA uses IncSFA to find low-dimensional manifolds within the raw pixel inputs, providing a basis for coupled perceptual *and* skill learning. I emphasize the special utility of SFA for this task over similar methods such as principal component analysis [Jolliffe, 2005] or predictive-projections [Sprague, 2009], which are based on variance or nearest neighbor learning, whereas slow features through IncSFA extract temporal invariance from input streams that represent "doorway" or "bottleneck" aspects (choke-points between two more fully connected subareas), similar to Laplacian-Eigen Maps [Sprekeler, 2011; Mahadevan and Maggioni, 2007; Luciw and Schmidhuber, 2012]. The hierarchical reinforcement learning literature [Schmidhuber, 1991a; Schmidhuber and Wahnsiedler, 1992; Wiering and Schmidhuber, 1998; Sutton et al., 1999; Menache et al., 2002; Bakker and Schmidhuber, 2004; Mahadevan and Maggioni, 2007; Şimşek and Barto, 2008] illustrates that such bottlenecks can be useful subgoals. Finding such bottlenecks in visual input spaces is a relatively new concept, and one I exploit in the iCub experiments.

At the core of CCSA is the Curious Dr. MISFA algorithm, which learns multiple slow feature abstractions by exploring image streams. Another image-based abstraction learning algorithm worth noting is the Object Semantic Hierarchy (OSH) developed by Xu and Kuipers [2010; 2011]. Motivated by the work of the spatial semantic hierarchy [Kuipers, 2000; Kuipers et al., 2004], OSH builds a collection of multi-level object representations from camera images. It uses "model-learning through tracking" [Modayil and Kuipers, 2004, 2008] strategy to model the static background and the individual foreground objects. Such a semantic knowledge can be useful for an agent to learn effective skills [Stober and Kuipers, 2008]. However, OSH assumes that the image background is static, which is not the case with a moving humanoid robot that causes the intensities of all the image pixels to change. Curious Dr. MISFA does not make such assumptions about the structure in the sensory stream. It captures features that encode the slowest variations in the video, which could correspond to a single or multiple objects, either as a part of the background or foreground. An intuitive example to demonstrate this is a scenario with an object moving in front of a noisy background. Slow features encode the object's pose rejecting the fast-varying background noise.

6.2 Limitations and Future Work

While much of the research in humanoid robot learning has been based upon human demonstrations, human-given task-descriptions, or pre-processed inputs, CCSA makes an important step towards combining several aspects needed to develop an online, continual curiosity-driven humanoid robotic agent. In the following, I will briefly list these aspects along with the current limitations of the CCSA framework and insights for future work:

- **Raw High-Dimensional Information Processing.** CCSA uses an IncSFA algorithm, updated online directly from raw-pixels, to encode slow-feature abstractions that lead to skills. Slow features learned through IncSFA are linear and not positional invariant, that is, once trained the features may not generalize to a spatial shift in the object's position in the image. To learn more complex skills however, CCSA might benefit from extracting non-linearities and translational invariance in the video inputs. Hierarchical extensions of IncSFA (H-IncSFA) over an *expanded input* in quadratic space [Luciw et al., 2012; Wiskott and Sejnowski, 2002] or the recently proposed Deeply-Learned SFA [DL-SFA; Sun et al., 2014] may remedy this. DL-SFA adopts the notion of 3D convolution and max-pooling to capture abstract, structural and translational invariant features. As future work, I plan to combine such non-linear hierarchical structures to improve the quality of the slow-feature abstractions learned.

- **Invariant Skills.** The skill labeled "grasp" in the experiments actually represents "grasp the cylindrical cup from the particular location in the given environment, invariant to the experimenter's actions and the iCub's head/body movements". The invariance picked up by the skills acquired in the system largely depend on the invariance learned by IncSFA from the observations sensed by the exploring iCub. In the experiments however, if the human experimenter had replaced the cup at different locations whenever the cup was toppled or dropped, IncSFA would probably have learned an abstraction that encodes whether the cup has been grasped-or-not invariant to the cup's posi-

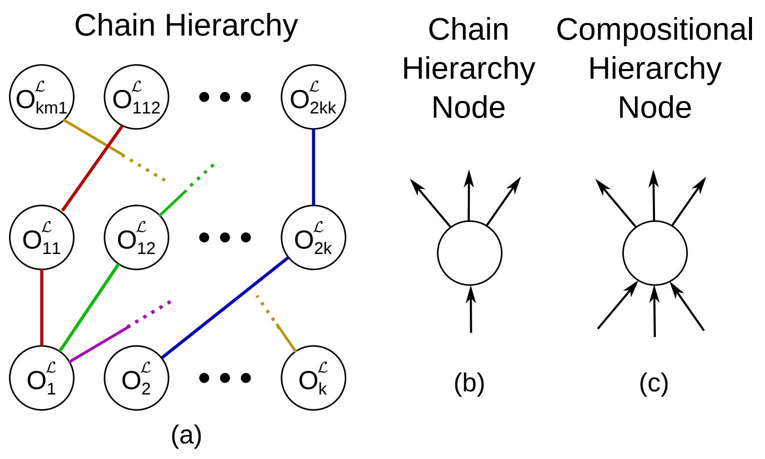


Figure 6.1. (a) The higher-order complex skills acquired using CCSA, are in the form of a chain-like hierarchy. There exists only a single chain-link (shown as a unique color) connecting higher-order to lower-order skills. This is because, a target option in CCSA is learned using observations only from one of the exploratory options. (b) An illustration of a node in a chain hierarchy. Each node has only a single input but can act as an input to many nodes. (c) Whereas, a node in a compositional hierarchy can have multiple inputs.

tion (because the events are uncorrelated). This would result in a “grasp skill” that is invariant to the cup’s position.

- **Continual Learning.** CCSA uses previously acquired knowledge in the form of biased explorations or policy-chunks, to learn more complex skills. This facilitates continual learning of skills. A previously acquired skill may be refined or adapted to suit to changing environments. For example, in the experiments, if the cup’s position has changed after acquiring the grasp skill, the *biased init. and explore* exploratory-option corresponding to the grasp skill can speed up learning a new skill to grasp the cup from the new position. However, the old skills are still retained and reused if the cup’s position is changed back to its original position. The complex skills acquired using CCSA are in the form of a chain-like hierarchy (Figure 6.1(a)), i.e., there exists only a single chain-link connecting higher-order to lower-order skills. This is because a target option in CCSA is learned using observations only from one of the exploratory options (see Section 5.2). Each node in the chain-like hierarchy has only a single input but can act as an input to many nodes (Figure 6.1(b)). In contrast, a node in a general compositional hierarchy (Figure 6.1(c)) can have multiple inputs. One way to achieve compositional hierarchy in CCSA is to add the learned target options to the primitive action set

$\mathscr{A} = \{$North, East, South, West, Hand-close and Hand-open$\}$.

- **Environmental Openness.** CCSA could benefit from a larger set of pre-defined input exploratory options. However, to minimize human inputs into the system, in the experiments the iCub starts with only a single exploratory option (random walk) and autonomously adds more exploratory options derived from the learned target options. Since CCSA acts directly on raw-pixels, no prior calibration of the robot cameras are required. Algorithm parameters are intuitive to tune [Kompella* et al., 2013; Kompella et al., 2012b,a; Luciw and Schmidhuber, 2012]. Therefore, CCSA can be used in different environments (and different humanoid robots) without making any design changes to the learning algorithm. On the motor end, I used a kinematic map that transforms the 41 degrees-of-freedom of the iCub joint configurations to 2D positions of its hand parallel to the table. For more complex manipulations, which are required for handling complicated objects, higher dimensional kinematic-maps could be used [Stollenga et al., 2013]. As a future work, I plan to use different approaches to tackle easier and safer manipulation with the iCub.

- **Quality of Skills Acquired.** I formally presented the underlying learning problem as a constrained optimization problem. The objective function can be used as a metric to tune different parameters of the method. However, the metric does not sufficiently evaluate the quality of skills acquired. One major factor is the type of the abstraction-estimator used. For example, a method that uses a simpler abstraction learning algorithm may acquire a large number of skills, which could be functionally equivalent to acquiring a single skill of a more discriminative abstraction estimator. Therefore, evaluating different task-unrelated intrinsically-motivated (IM) approaches without providing an external goal is an ill-posed problem. As a future work, I plan to build realistic, task-independent, skill-acquisition benchmarks with hidden external tasks to evaluate multiple IM approaches.

- **Scalability.** For each target option acquired by CCSA, the number of input exploratory options increases by a value of two (See Section 5.3.4). Observations from previously encoded exploratory options are automatically filtered out due to the gating system of Curious Dr. MISFA. Therefore, for each target option acquired, the number of *unknown* exploratory options increases by only one. Hence, the space of input exploratory options scales linearly with respect to the number of skills acquired.

- **Sensor Fusion.** And finally, CCSA uses only visual inputs from the onboard cameras and joint angles of the iCub. A humanoid-robot's actions can be improved however, by using different sensory modalities such as tactile and audio in addition to the visual inputs. This should be straightforward addition to CCSA, since IncSFA is agnostic to the modality of sensory information. The raw inputs of different modalities can be concatenated as a single input and fed to the IncSFA algorithm, without causing too much computational overhead (since IncSFA has a linear update complexity [Kompella et al., 2012a]). Related work on combining sensory modalities using SFA methods have shown to achieve good results [Höfer et al., 2012].

The above insights should help to improve CCSA in the future.

6.3 Conclusions

In this thesis, I have presented the following three original contributions addressing some of the open problems in artificial intelligence research:

- In Chapter 3, I proposed Incremental Slow Feature Analysis (IncSFA) as a low-complex, online alternative to the batch SFA (BSFA). I showed through experimental results how IncSFA extracts slow features without storing any input data or estimating costly covariance matrices. This makes IncSFA suitable to use for several online learning applications.

- In Chapter 4, I proposed an online active modular IncSFA algorithm called the Curiosity-Driven Modular Incremental Slow Feature Analysis (Curious Dr. MISFA). Curious Dr. MISFA uses the theory of artificial curiosity to address the forgetting problem faced by IncSFA, by retaining what was previously learned in the form of expert slow feature abstractions. I mathematically proved that under certain technical conditions, Curious Dr. MISFA learns abstractions in the order of increasing learning difficulty. These theoretical optimality guarantees were lacking in previous practical implementations of the curiosity theory [Schmidhuber, 2010b]. Curious Dr. MISFA addresses the open problem of curiosity-driven abstraction learning.

- In Chapter 5, I proposed a framework for Continual Curiosity-Driven Skill Acquisition (CCSA) for acquiring, storing and reusing both abstractions and skills in an online and continual manner. I showed through experiments how CCSA guides an iCub humanoid robot to acquire a repertoire of skills (topple, grasp) from raw-pixel vision driven purely by its intrinsic motivation.

These contributions together demonstrate that the online implementations of slowness learning make it suitable for an open-ended curiosity-driven RL agent to acquire a repertoire of skills that map the many raw-pixels of high-dimensional images to multiple sets of action sequences. In the future, systems like CCSA will be used to develop flexible and autonomous learning machines, with substantial benefits for society as a whole.

Appendix A

Proofs

This section presents the proofs of all the theorems discussed in this chapter. For the sake of readability, I will redefine some of the notation that has been previously defined in Section 4.2. Let $\mathscr{X} = \{\mathbf{x} : \mathbf{x}(t) \in \mathbb{R}^I, I \in \mathbb{N}\}$ denote a set of of I-dimensional observation streams. Let $X \subset \mathscr{X}$ be a finite subset with $n \in \mathbb{N}$ elements that may or may not be unique. Let Θ denote an abstraction-estimator that updates a real-valued abstraction and ensures an *almost sure* convergence for an I-dimensional stationary input signal. Let Φ^* denote the space of all learnable abstractions by Θ for the input X satisfying Constraints (4.3)-(4.4) (see Section 4.2). Let $\widehat{\phi} \in \mathbb{R}^{I \times J}, J \in \mathbb{N}, \widehat{\phi} \notin \Phi^*$ denote the adaptive abstraction. The proof of Theorem 1 is as follows.

Proof for Theorem 1

Theorem 1. *There exists a curiosity-function $\Omega : \mathscr{X} \to [0, 1)$ corresponding to Θ that induces a total ordering on $\mathscr{X}$.*

Proof. Since Θ ensures an almost sure convergence on a stationary input signal $\mathbf{x} \in \mathscr{X}$, there exists a time $T_\mathbf{x} \in \mathbb{R}^+$ s.t. for

$$t > T_\mathbf{x}, \quad |\widehat{\phi}_t - \phi^*| < \delta, \tag{A.1}$$

where ϕ^* represents a fixed-point ($\in \mathbb{R}^{I \times J}$) and δ is a small non-negative scalar constant. $T_\mathbf{x}$ is called the convergence-time for the signal $\mathbf{x}$. For non-stationary signals in $\mathscr{X}$, it can be assumed that the above condition holds at infinity. Let T denote the set of convergence-times of all the signals $\mathbf{x} \in \mathscr{X}$. Therefore, there exists a function $\mathscr{T} : \mathscr{X} \to T$, s.t. $\mathscr{T}(\mathbf{x})$ denotes the convergence-time of the input $\mathbf{x}$. It is straightforward to show that since T is a totally-ordered set, $\mathscr{T}$ induces a total ordering in $\mathscr{X}$. One can easily find an order-preserving transfer function

$f : \mathscr{T} \to [0,1)$, for example $1 - e^{-T}$, such that the composite function $\Omega = f \circ T$ induces a total ordering in $\mathscr{X}$. □

The next section presents a full derivation of the curiosity function for the IncSFA algorithm.

Derivation of Definition 1

I will derive here the curiosity function of the IncSFA algorithm for a stationary observation stream $\mathbf{x}(t) \in \mathbb{R}^I, I \in \mathbb{N}$. To keep it simple, it is assumed that the learning progress made by the CCIPCA algorithm is the same for different observation streams. This assumption approximately holds for signals that have similar ratios between the eigenvalues of the top principal components [Weng et al., 2003], which is the case in our experiments. Therefore, the learning progress of the IncSFA for different observation streams is proportional to the learning progress of the CIMCA algorithm.

Input to the CIMCA algorithm is the derivative of the normalized (*whitened*) observation stream $\mathbf{z}(t) \in \mathbb{R}^K$ (see Section 3). The update rule of CIMCA for the first minor component ($\widehat{\phi}_1$) is given by

$$\widehat{\phi}_1(t) = \left(1 - \eta^{mca}\right)\, \widehat{\phi}_1(t-1) - \eta^{mca}\, (\dot{\mathbf{z}}(t) \cdot \widehat{\phi}_1(t-1))\, \dot{\mathbf{z}}(t) \tag{A.2}$$

$$\widehat{\phi}_1(t) = \widehat{\phi}_1(t) / \|\widehat{\phi}_1(t)\|. \tag{A.3}$$

To analyze the "average" dynamics, Eq. (A.2) is reformulated by taking the conditional expected value $\mathbb{E}[\widehat{\phi}_1(t+1) | \widehat{\phi}_1(0), \dot{\mathbf{z}}(i), i < t]$ at each iteration

$$\widehat{\phi}_1(t) = \left(1 - \eta^{mca}\right)\, \widehat{\phi}_1(t-1) - \eta^{mca}\, \mathbb{E}[\dot{\mathbf{z}}(t)\dot{\mathbf{z}}(t)^T] \widehat{\phi}_1(t-1). \tag{A.4}$$

Here, $\mathbb{E}[\dot{\mathbf{z}}(t)\dot{\mathbf{z}}(t)^T]$ is the correlation matrix of $\dot{\mathbf{z}}(t)$. Since the correlation matrix is a symmetric nonnegative definite matrix, it can be factorized into QDQ^{-1}, where Q is the eigenvector matrix (columns representing unit eigenvectors v_i) and D is a diagonal matrix with corresponding eigenvalues (λ_i). In addition, the eigenvectors $\{v_i | i = 1, 2, ..., K\}$ form an orthonormal basis spanning $\mathbb{R}^K$. The weight vector $\widehat{\phi}_1$ can then be represented as

$$\widehat{\phi}_1(t) = \sum_{i=1}^{K} a_i(t) v_i, \tag{A.5}$$

where $a_i(t)$ are some constant coefficients. The following lemmas are required.

Lemma 1. *Let V_i be denoted as* $V_i = \left[1 - \eta^{mca} - \eta^{mca}\lambda_i\right]$

$$\textit{then,} \qquad a_i(t) = \frac{V_i^t a_i(0)}{\sqrt{\sum_j^K V_j^{2t} a_j^2(0)}}, \quad \forall i \in \{1, 2, ..., K\}. \tag{A.6}$$

Proof. The proof is by the principle of mathematical induction.
t = 1: Substituting Eq. (A.5) in Eq. (A.4) for t=1, we get: $a_i(1) = V_i a_i(0), \quad \forall i \in \{1, 2, ..., K\}$.
At each update, the weight vector $\widehat{\phi}_1(t)$ is normalized according to Eq. (A.3).

$$a_i(1) = \frac{V_i a_i(0)}{\sqrt{\sum_j^K V_j^2 a_j^2(0)}}, \quad \forall i \in \{1, 2, ..., K\}. \tag{A.7}$$

t = m: Assuming the result to be true for some $t = m > 1$ and let $P = \sqrt{\sum_j^K V_j^{2m} a_j^2(0)}$.
t = m+1: Substituting Eq. (A.5) in Eq. (A.4) for t=m, we get

$$a_i(m+1) = V_i a_i(m) = \frac{V_i^{m+1} a_i(0)}{P}.$$

Upon normalizing, $$a_i(m+1) = \frac{\frac{V_i^{m+1} a_i(0)}{P}}{\sqrt{\sum_j^K \frac{V_j^{2m+2} a_j^2(0)}{P^2}}} = \frac{V_i^{m+1} a_i(0)}{\sqrt{\sum_j^K V_j^{2m+2} a_j^2(0)}}, \quad \forall i \in \{1, 2, ..., K\},$$

which is the same as substituting t=m+1 in Eq. (A.6). Therefore, by the principle of mathematical induction, Eq. (A.6) holds true for any $t > 1$. □

Lemma 2. *Let σ_i be denoted as* $\sigma_i = \left[1 - \dfrac{\eta^{mca}(\lambda_i - \lambda_K)}{1 - \eta^{mca} - \eta^{mca}\lambda_K}\right]$

$$\textit{then,} \qquad 0 < \sigma_1 < ... < \sigma_{K-1} < 1. \tag{A.8}$$

Proof. The condition (A.8) is straightforward if $0 < \dfrac{\eta^{mca}(\lambda_i - \lambda_K)}{1 - \eta^{mca} - \eta^{mca}\lambda_K} < 1$ is true. The left inequality is proved first. Clearly, since $\lambda_1 > ... > \lambda_K \geq 0$ and $0 < \eta^{mca} \leq 0.5$, the numerator

$$\eta^{mca}(\lambda_i - \lambda_K) > 0, \ \forall i \in \{1, ..., K-1\} \tag{A.9}$$

and the denominator
$$\begin{aligned} 1 - \eta^{mca} - \eta^{mca}\lambda_K &> 1 - \eta^{mca} - \eta^{mca}\lambda_1 \\ &> 0.5 - \eta^{mca}\lambda_1, \quad \because \eta^{mca} < 0.5 \\ &> 0, \qquad\qquad\quad \because \eta^{mca}\lambda_1 < 0.5 \end{aligned} \tag{A.10}$$

To prove the right inequality, it holds

$$\begin{aligned}
\text{iff,} \quad \eta^{mca}(\lambda_i - \lambda_K) &< 1 - \eta^{mca} - \eta^{mca}\lambda_K \\
\text{iff,} \quad \eta^{mca}\lambda_1 &< 1 - \eta^{mca} \\
\text{iff,} \quad 0.5 &< 1 - \eta^{mca}, \quad \text{which is true.}
\end{aligned}$$

□

Lemma 3. *Let* $C_i = \left[\frac{a_i(0)}{a_K(0)}\right]$ *then,*

$$a_i(t) = C_i \sigma_i^t a_K(t), \quad \forall i \in \{1,...,K-1\} \tag{A.11}$$

$$a_K(t) = \frac{1}{\sqrt{\sum_j^{K-1} \sigma_j^{2t} C_j^2 + 1}}. \tag{A.12}$$

Proof. Using Eq. (A.6) and the condition (A.10), we get

$$\begin{aligned}
\frac{a_i(t+1)}{a_K(t+1)} &= \left[\frac{1-\eta^{mca}-\eta^{mca}\lambda_i}{1-\eta^{mca}-\eta^{mca}\lambda_K}\right] \cdot \left[\frac{a_i(t)}{a_K(t)}\right], \quad \forall i \in \{1,...,K-1\} \\
&= \left[1 - \frac{\eta^{mca}(\lambda_i - \lambda_K)}{1-\eta^{mca}-\eta^{mca}\lambda_K}\right] \cdot \left[\frac{a_i(t)}{a_K(t)}\right] = \sigma_i \cdot \left[\frac{a_i(t)}{a_K(t)}\right] = \sigma_i^{t+1} \cdot \left[\frac{a_i(0)}{a_K(0)}\right]
\end{aligned}$$

This implies that $a_i(t) = C_i \sigma_i^t a_K(t), \ \forall i \in \{1,...,K-1\}$. Using the result from Lemma 1 and substituting for i=n, we get

$$a_K(t) = \frac{V_K^t a_K(0)}{\sqrt{\sum_j^K V_j^{2t} a_j^2(0)}} = \frac{1}{\sqrt{\sum_j^{K-1} \left(\frac{V_j}{V_K}\right)^{2t} \left(\frac{a_j(0)}{a_K(0)}\right)^2 + 1}} = \frac{1}{\sqrt{\sum_j^{K-1} \sigma_j^{2t} C_j^2 + 1}}$$

□

Lemma 4. *Let* $\tau_i^{1/2}$ *denote the half-life period of* $a_i(t)$*, then the following inequality holds:*

$$\tau_1^{1/2} < ... < \tau_{K-1}^{1/2} \tag{A.13}$$

Proof. Since $a_K(t)$ is bounded ($0 < a_K(t) < 1$), coefficients $a_i(t)$ ($\forall i \in \{1,...,K-1\}$) belong to a family of exponential-decay functions: $C_i a_K(t) e^{-t ln(1/\sigma_i)}$. Half-life period $\tau_i^{1/2}$ is the time when the value $a_i(t)$ becomes equal to half its initial value. Therefore,

$$C_i a_K(t) \sigma_i^t = C_i a_K(0)/2$$

Using Lemma 3 and simplifying we get,

$$t = -\frac{ln(2)}{ln\sigma_i} + \frac{0.5}{ln\sigma_i} * ln\left(\frac{\sum_j^{K-1}\sigma_j^{2t}C_j^2 + 1}{\sum_j^{K-1}C_j^2 + 1}\right) \tag{A.14}$$

Let $\frac{\sum_j^{K-1}\sigma_j^{2t}C_j^2 + 1}{\sum_j^{K-1}C_j^2 + 1}$ be denoted by ξ. From Lemma 2 and $t > 0$ we get, $0 < \xi < 1$ and ξ decreases monotonically with respect to t. However, for higher values of t and consecutive σ_i's, ξ does not change significantly with t and can be assumed to be a constant. Substituting the term ξ in Eq. (A.14), we get

$$\tau_i^{1/2} = -\frac{ln(2) - 0.5 * ln(\xi)}{ln\sigma_i} = \frac{ln(2) - 0.5 * ln(\xi)}{ln(1/\sigma_i)} \tag{A.15}$$

Therefore, from Eq. (A.15) and Lemma 2 we have, $\tau_{j-1}^{1/2} < \tau_j^{1/2}$, $\forall j \in \{2, ..., K-1\}$.
□

Definition 1. *The curiosity function of the IncSFA algorithm for an observation stream* $\mathbf{x}(t) \in \mathbb{R}^I$ *is defined as*

$$\Omega(\mathbf{x}) = \sigma_{K-1} = \left[1 - \frac{\eta^{mca}(\lambda_{K-1} - \lambda_K)}{1 - \eta^{mca} - \eta^{mca}\lambda_K}\right], \tag{A.16}$$

where $\lambda_K \neq \lambda_{K-1}$ *denote the smallest two eigenvalues of* $\mathbb{E}[\dot{\mathbf{z}}(t)\dot{\mathbf{z}}(t)^T]$, $\mathbf{z}(t) \in \mathbb{R}^K$ *is the* whitened *output of* $\mathbf{x}(t)$.

The next section provides proofs for the rest of the theorems.

Proofs for Theorems 2-6

The reward function (see Section 4.3.4) R^{int} generated by the algorithm has a *Euclidean norm* equal to one. For the sake of convenience, R^{int} is denoted by R in the rest of this section. It is assumed that R takes only non-negative values. This assumption is trivial since, a scalar positive constant can be added to R^{int} without having any effect on the policies learned by the Least Squares Policy Iteration (LSPI) reinforcement learning algorithm. The following definition is useful for the rest of the theorems.

Definition 2. *At time t, let $\mathbf{x}_l$ denote the current easiest but not yet learned observation stream and s_l denote the corresponding state. Then, the index l is given by*

$$l = \underset{\forall i:\ \mathbf{x}_i \in X'}{\arg\min}\, \Omega(\mathbf{x}_i), \quad X' = \{\mathbf{x}_i : \mathscr{G}(\mathbf{x}_i) = \widehat{\phi}, \mathbf{x}_i \in X\}. \tag{A.17}$$

Based on Ω, *optimal fixed-points* for the adaptive abstraction $\widehat{\phi}$ and the observation stream selection policy π^{int} are defined next.

Theorem 2. *At time t, the optimal fixed-point $\phi^* \in \Phi^*$ of the adaptive abstraction $\widehat{\phi}$ is equal to the J slow features of the observation stream $\mathbf{x}_l$.*

Proof. The proof is straightforward. $\square$

Theorem 3. *The optimal observation stream selection policy ($\pi^* : \Phi^* \times \mathscr{S}^{int} \to \mathscr{A}^{int}, \mathscr{A}^{int} = \{0\ (stay), 1\ (switch)\}$) to learn an abstraction $\phi_i \in \Phi^*$ is given by:*

$$\pi^*(\phi_i, s) = 1 - \mathbb{1}_{\{s_l\}}(s), \ \forall s \in \mathscr{S}^{int}.$$

Proof. The proof is straightforward and follows from Theorem 2. The optimal policy is such that the agent takes the action *stay* $(= 0)$ in the state s_l, which corresponds to the current easiest but not yet encoded observation stream $\mathbf{x}_l$, and takes the action *switch* $(= 1)$ in the rest of the states. $\square$

The following lemma is useful for the rest of the analysis.

Lemma 5. *Let R denote the estimated internal reward function by the algorithm at any time t. Let π^{int} be any arbitrary observation stream selection policy and let k_0 and k_1 denote sets of internal states where the policy returns a zero (*stay*) and one (*switch*) respectively:*

$$k_0 = \{s \mid \pi^{int}(s) = 0, \forall s \in \mathscr{S}^{int}\}$$
$$k_1 = \{s \mid \pi^{int}(s) = 1, \forall s \in \mathscr{S}^{int}\}.$$

Then, the action values corresponding to each (s, a) tuple for the policy π^{int} are *given by:*

$$(a)\ Q^{stay}_{s\in k_0} = \frac{R^{stay}_{ss}}{1-\gamma} \tag{A.18}$$

$$(b)\ Q^{switch}_{s\in k_1} = \frac{1}{(n-1+\gamma)}\left[\sum_{s'\in\mathcal{S}^{int}\backslash s} R^{switch}_{ss'} + \widehat{R}^{switch}\right] + \widehat{R}^{stay} \tag{A.19}$$

$$(c)\ Q^{switch}_{s\in k_0} = \frac{1}{(n-1)}\left[\sum_{s'\in\mathcal{S}^{int}\backslash s} R^{switch}_{ss'} + \widehat{R}^{switch} + (n-1+\gamma)\widehat{R}^{stay} - \frac{\gamma}{1-\gamma}R^{stay}_{ss}\right] \tag{A.20}$$

$$(d)\ Q^{stay}_{s\in k_1} = R^{stay}_{ss} + \frac{\gamma}{(n-1+\gamma)}\left[\sum_{s'\in\mathcal{S}^{int}\backslash s} R^{switch}_{ss'} + \widehat{R}^{switch}\right] + \widehat{R}^{stay} \tag{A.21}$$

where,

$$\widehat{R}^{switch} = \frac{\gamma}{\left(n-1-\gamma(\left|k_1\right|-1)\right)}\sum_{s''\in k_1}\sum_{s'\in\mathcal{S}^{int}\backslash s''} R^{switch}_{s's''} \tag{A.22}$$

$$\widehat{R}^{stay} = \frac{\gamma}{(1-\gamma)\left(n-1-\gamma(\left|k_1\right|-1)\right)}\sum_{s''\in k_0} R^{stay}_{s''s''} \tag{A.23}$$

Proof. The value of a (s,a) tuple is the expected cumulative future reward that the agent can accumulate starting by executing the action a in the state s.

(a) $Q^{\text{stay}}_{s\in k_0} = \sum_{t=0}^{\infty}\gamma^t R^{\text{stay}}_{ss}\mathcal{P}^{\text{stay}}_{ss} = \dfrac{R^{\text{stay}}_{ss}}{1-\gamma}$

(b) $Q^{\text{switch}}_{s\in k_1} = \sum_{s'\in k_0}\left[R^{\text{switch}}_{ss'} + \gamma Q^{\text{stay}}_{s'}\right]\mathcal{P}^{\text{switch}}_{ss'} + \sum_{s'\in k_1\backslash s}\left[R^{\text{switch}}_{ss'} + \gamma Q^{\text{switch}}_{s'}\right]\mathcal{P}^{\text{switch}}_{ss'}$

Substituting $\mathcal{P}^{\text{switch}}_{ss'} = 1/(n-1)$ (see Section 4.3.1), $k_0 \cup k_1 = \mathcal{S}^{\text{int}}$ and the result from (a), we get

$$= \frac{1}{n-1}\left[\sum_{s'\in\mathcal{S}^{\text{int}}\backslash s} R^{\text{switch}}_{ss'} + \frac{\gamma}{1-\gamma}\sum_{s'\in k_0} R^{\text{stay}}_{s's'} + \gamma\sum_{s'\in k_1\backslash s} Q^{\text{switch}}_{s'}\right] \tag{A.24}$$

Taking a summation of Q^{switch}_{s} over all $s\in k_1$ and solving, we get,

$$\sum_{s''\in k_1} Q^{\text{switch}}_{s''} = \frac{1}{n-1}\left[\sum_{s''\in k_1}\sum_{s'\in\mathcal{S}^{\text{int}}\backslash s''} R^{\text{switch}}_{s's''} + \frac{\left|k_1\right|\gamma}{1-\gamma}\sum_{s'\in k_0} R^{\text{stay}}_{s's'} + \gamma(\left|k_1\right|-1)\sum_{s'\in k_1} Q^{\text{switch}}_{s'}\right]$$

$$= \frac{\sum_{s''\in k_1}\sum_{s'\in\mathscr{S}^{\text{int}}\backslash s''} R^{\text{switch}}_{s's''} + \frac{|k_1|\gamma}{1-\gamma}\sum_{s'\in k_0} R^{\text{stay}}_{s's'}}{(n-1-\gamma(|k_1|-1))} \tag{A.25}$$

Substituting Eq. (A.25) in Eq. (A.24) and solving we get,

$$Q^{\text{switch}}_{s\in k_1} = \frac{1}{(n-1+\gamma)}\left[\sum_{s'\in\mathscr{S}^{\text{int}}\backslash s} R^{\text{switch}}_{ss'} + \widehat{R}^{\text{switch}}\right] + \widehat{R}^{\text{stay}}$$

(c) $Q^{\text{switch}}_{s\in k_0} = \sum_{s'\in k_0\backslash s}\left[R^{\text{switch}}_{ss'} + \gamma Q^{\text{stay}}_{s'}\right]\mathscr{P}^{\text{switch}}_{ss'} + \sum_{s'\in k_1}\left[R^{\text{switch}}_{ss'} + \gamma Q^{\text{switch}}_{s'}\right]\mathscr{P}^{\text{switch}}_{ss'}$

$$= \frac{1}{n-1}\left[\sum_{s'\in\mathscr{S}^{\text{int}}\backslash s} R^{\text{switch}}_{ss'} + \frac{\gamma}{1-\gamma}\sum_{s'\in k_0\backslash s} R^{\text{stay}}_{s's'} + \gamma\sum_{s'\in k_1} Q^{\text{switch}}_{s'}\right] \tag{A.26}$$

Substituting Eq. (A.25) in Eq. (A.26) and solving we get,

$$Q^{\text{switch}}_{s\in k_0} = \frac{1}{(n-1)}\left[\sum_{s'\in\mathscr{S}^{\text{int}}\backslash s} R^{\text{switch}}_{ss'} + \widehat{R}^{\text{switch}} + (n-1+\gamma)\widehat{R}^{\text{stay}} - \frac{\gamma}{1-\gamma}R^{\text{stay}}_{ss}\right]$$

(d) $Q^{\text{stay}}_{s\in k_1} = R^{\text{stay}}_{ss} + \gamma Q^{\text{switch}}_{s\in k_1}$

$$= R^{\text{stay}}_{ss} + \frac{\gamma}{(n-1+\gamma)}\left[\sum_{s'\in\mathscr{S}^{\text{int}}\backslash s} R^{\text{switch}}_{ss'} + \widehat{R}^{\text{switch}}\right] + \widehat{R}^{\text{stay}}$$

□

Lemma 6. *Let R denote the estimated reward function by the algorithm at any time t and let* $s_l = \arg\max_s R^{stay}_{ss}$. *If*

$$R^{stay}_{s_l s_l} = \max(R) \quad and \quad R^{stay}_{ss} < \frac{\gamma R^{stay}_{s_l s_l}}{(n-1)-\gamma(n-2)},\ \forall s\in\mathscr{S}^{int}\setminus s_l$$

then,

$$\arg\max_{\pi^{int}} Q^{\pi^{int}} = 1 - \mathbb{1}_{\{s_l\}}(s),\ \forall s\in\mathscr{S}^{int}$$

Proof. Let $\pi^{\text{opt}} = 1 - \mathbb{1}_{\{s_l\}}(s),\ \forall s\in\mathscr{S}^{\text{int}}$. The proof is straightforward if the following hold true:

1. (a) $Q^{\text{switch},\pi^{\text{int}}\neq\pi^{\text{opt}}}_{s\in k_1} < Q^{\text{switch},\pi^{\text{opt}}}_{s\in k_1}$ & (b) $Q^{\text{switch},\pi^{\text{int}}\neq\pi^{\text{opt}}}_{s\in k_0} < Q^{\text{switch},\pi^{\text{opt}}}_{s\in k_1}$
2. (a) $Q^{\text{stay},\pi^{\text{int}}\neq\pi^{\text{opt}}}_{s\in k_1} < Q^{\text{stay},\pi^{\text{opt}}}_{s\in k_1}$ & (b) $Q^{\text{stay},\pi^{\text{int}}\neq\pi^{\text{opt}}}_{s\in k_0} < Q^{\text{stay},\pi^{\text{opt}}}_{s\in k_1}$
3. (a) $Q^{\text{switch},\pi^{\text{int}}\neq\pi^{\text{opt}}}_{s\in k_1} < Q^{\text{switch},\pi^{\text{opt}}}_{s\in k_0}$ & (b) $Q^{\text{switch},\pi^{\text{int}}\neq\pi^{\text{opt}}}_{s\in k_0} < Q^{\text{switch},\pi^{\text{opt}}}_{s\in k_0}$
4. (a) $Q^{\text{stay},\pi^{\text{int}}\neq\pi^{\text{opt}}}_{s\in k_1} < Q^{\text{stay},\pi^{\text{opt}}}_{s\in k_0}$ & (b) $Q^{\text{stay},\pi^{\text{int}}\neq\pi^{\text{opt}}}_{s\in k_0} < Q^{\text{stay},\pi^{\text{opt}}}_{s\in k_0}$

Each of the above inequalities are proved in turn. Let k_0 and k_1 denote sets of states where a policy π^{int} returns a zero (*stay*) and one (*switch*) respectively:

$$\begin{aligned} k_0 &= \left\{s \mid \pi^{\text{int}}(s) = 0, \forall s \in \mathscr{S}^{\text{int}}\right\} \\ k_1 &= \left\{s \mid \pi^{\text{int}}(s) = 1, \forall s \in \mathscr{S}^{\text{int}}\right\}. \end{aligned}$$

For the policy π^{opt}, $k_0 = \{s_l\}$ and $k_1 = \mathscr{S}^{\text{int}} \setminus s_l$ ($(n-1)$ elements). Therefore, for any other policy $\pi^{\text{int}} \neq \pi^{\text{opt}}$, either $\left|k_1\right| = n$ or $\left|k_1\right| < (n-1)$. The result for $\left|k_1\right| = n$ (*switch* at all states) is straightforward to show. Here, the case $\left|k_1\right| < (n-1)$ is considered.

Proof for 1-(a): Using the condition $\left|k_1\right| < n-1$ in Eq. (A.19), we get,

$$\begin{aligned} Q^{\text{switch},\pi^{\text{int}}\neq\pi^{\text{opt}}}_{s\in k_1} &= \frac{1}{(n-1+\gamma)}\left[\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s} R^{\text{switch}}_{ss'} + \frac{\gamma}{\left(n-1-\gamma(\left|k_1\right|-1)\right)}\sum_{s''\in k_1}\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s''} R^{\text{switch}}_{s's''}\right] \\ &\quad + \frac{\gamma}{(1-\gamma)\left(n-1-\gamma(\left|k_1\right|-1)\right)}\sum_{s''\in k_0} R^{\text{stay}}_{s''s''} \\ &< \frac{1}{(n-1+\gamma)}\left[\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s} R^{\text{switch}}_{ss'} + \frac{\gamma}{\left(n-1-\gamma(n-2)\right)}\sum_{s''\in\mathscr{S}^{\text{int}}}\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s''} R^{\text{switch}}_{s's''}\right] \\ &\quad + \frac{\gamma}{(1-\gamma)\left(n-1-\gamma(\left|k_1\right|-1)\right)}\sum_{s''\in k_0} R^{\text{stay}}_{s''s''} \end{aligned}$$

Using the condition $R^{\text{stay}}_{ss} < \frac{\gamma R^{\text{stay}}_{s_l s_l}}{(n-1)-\gamma(n-2)}$, $\forall s \in \mathscr{S}^{\text{int}} \setminus s_l$, we get

$$\begin{aligned} &< \frac{1}{(n-1+\gamma)}\left[\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s} R^{\text{switch}}_{ss'} + \frac{\gamma}{\left(n-1-\gamma(n-2)\right)}\sum_{s''\in\mathscr{S}^{\text{int}}}\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s''} R^{\text{switch}}_{s's''}\right] \\ &\quad + \frac{\gamma}{(1-\gamma)\left(n-1-\gamma(\left|k_1\right|-1)\right)}\left[\frac{\gamma(\left|k_0\right|-1)}{(n-1)-\gamma(n-2)}+1\right] R^{\text{stay}}_{s_l s_l} \end{aligned}$$

Substituting $\left|k_0\right| = n - \left|k_1\right|$ and solving, we get

$$
\begin{aligned}
&= \frac{1}{(n-1+\gamma)}\left[\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s} R^{\text{switch}}_{ss'} + \frac{\gamma}{\big(n-1-\gamma(n-2)\big)}\sum_{s''\in\mathscr{S}^{\text{int}}}\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s''} R^{\text{switch}}_{s's''}\right] \\
&\quad + \frac{\gamma}{(1-\gamma)\big((n-1)-\gamma(n-2)\big)} R^{\text{stay}}_{s_l s_l} \\
&= Q^{\text{switch},\pi^{\text{opt}}}_{s\in k_1}
\end{aligned}
$$

Hence, $Q^{\text{switch},\pi^{\text{int}}\neq\pi^{\text{opt}}}_{s\in k_1} < Q^{\text{switch},\pi^{\text{opt}}}_{s\in k_1}$.

Proof for 1-(b): From Eq. (A.20) we have,

$$
\begin{aligned}
Q^{\text{switch},\pi^{\text{int}}\neq\pi^{\text{opt}}}_{s\in k_0} &= \frac{1}{(n-1)}\left[\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s} R^{\text{switch}}_{ss'} + \frac{\gamma}{\big(n-1-\gamma(\left|k_1\right|-1)\big)}\sum_{s''\in k_1}\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s''} R^{\text{switch}}_{s's''}\right] \\
&\quad + \frac{\gamma}{(1-\gamma)(n-1)}\left[\frac{(n-1+\gamma)}{\big(n-1-\gamma(\left|k_1\right|-1)\big)}\sum_{s''\in k_0} R^{\text{stay}}_{s''s''}\right] - \frac{\gamma}{(n-1)(1-\gamma)} R^{\text{stay}}_{ss} \\
&= \frac{1}{(n-1+\gamma)}\left[\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s} R^{\text{switch}}_{ss'} + \frac{\gamma}{n-1}\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s} R^{\text{switch}}_{ss'} + \frac{\gamma(n-1+\gamma)\sum_{s''\in k_1}\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s''} R^{\text{switch}}_{s's''}}{(n-1)\big(n-1-\gamma(\left|k_1\right|-1)\big)}\right] \\
&\quad + \frac{\gamma\left[(n-1+\gamma)\sum_{s''\in k_0} R^{\text{stay}}_{s''s''} - \big(n-1-\gamma(\left|k_1\right|-1)\big) R^{\text{stay}}_{ss}\right]}{(1-\gamma)(n-1)\big(n-1-\gamma(\left|k_1\right|-1)\big)}
\end{aligned}
$$

Substituting the following in the first term of R.H.S.:

- $(n-1)-\gamma(n-2) < (n-1)$,
- since R has all non-negative entries $\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s} R^{\text{switch}}_{ss'} < \sum_{s''\in k_0}\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s''} R^{\text{switch}}_{s's''}$, and
- it can easily be shown that $\dfrac{n-1+\gamma}{n-1} < \dfrac{n-1-\gamma(\left|k_1\right|-1)}{n-1-\gamma(n-2)}$, we get,

$$Q_{s\in k_0}^{\text{switch},\pi^{\text{int}}\neq\pi^{\text{opt}}} < \frac{1}{(n-1+\gamma)}\left[\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s} R_{ss'}^{\text{switch}} + \frac{\gamma\sum\limits_{s''\in k_0}\sum\limits_{s'\in\mathscr{S}^{\text{int}}\setminus s''} R_{s's''}^{\text{switch}}}{(n-1)-\gamma(n-2)} + \frac{\gamma\sum\limits_{s''\in k_1}\sum\limits_{s'\in\mathscr{S}^{\text{int}}\setminus s''} R_{s's''}^{\text{switch}}}{(n-1-\gamma(n-2))}\right]$$
$$+\frac{\gamma\left[(n-1+\gamma)\sum\limits_{s''\in k_0} R_{s''s''}^{\text{stay}} - \left(n-1-\gamma(\left|k_1\right|-1)\right)R_{ss}^{\text{stay}}\right]}{(1-\gamma)(n-1)\left(n-1-\gamma(\left|k_1\right|-1)\right)}$$
$$=\frac{1}{(n-1+\gamma)}\left[\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s'} R_{ss'}^{\text{switch}} + \frac{\gamma\sum\limits_{s''\in \mathscr{S}^{\text{int}}}\sum\limits_{s'\in\mathscr{S}^{\text{int}}\setminus s''} R_{s's''}^{\text{switch}}}{(n-1)-\gamma(n-2)}\right]$$
$$+\frac{\gamma\left[(n-1+\gamma)\sum\limits_{s''\in k_0} R_{s''s''}^{\text{stay}} - \left(n-1-\gamma(\left|k_1\right|-1)\right)R_{ss}^{\text{stay}}\right]}{(1-\gamma)(n-1)\left(n-1-\gamma(\left|k_1\right|-1)\right)}$$

Substituting $\sum\limits_{s''\in k_0} R_{s''s''}^{\text{stay}} = \sum\limits_{s''\in k_0\setminus s_l} R_{s''s''}^{\text{stay}} + R_{s_ls_l}^{\text{stay}}$ and the condition $R_{ss}^{\text{stay}} < \frac{\gamma R_{s_ls_l}^{\text{stay}}}{(n-1)-\gamma(n-2)}$, $\forall s \in \mathscr{S}^{\text{int}}\setminus s_l$ in the second term of R.H.S., we get,

$$<\frac{1}{(n-1+\gamma)}\left[\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s'} R_{ss'}^{\text{switch}} + \frac{\gamma\sum\limits_{s''\in \mathscr{S}^{\text{int}}}\sum\limits_{s'\in\mathscr{S}^{\text{int}}\setminus s''} R_{s's''}^{\text{switch}}}{(n-1)-\gamma(n-2)}\right]$$
$$+\frac{\gamma R_{s_ls_l}^{\text{stay}}\left[\frac{(n-1+\gamma)(\left|k_0\right|-1)\gamma}{(n-1)-\gamma(n-2)} + (n-1+\gamma) - \frac{\left(n-1-\gamma(\left|k_1\right|-1)\right)\gamma}{(n-1)-\gamma(n-2)}\right]}{(1-\gamma)(n-1)\left(n-1-\gamma(\left|k_1\right|-1)\right)}$$
$$=\frac{1}{(n-1+\gamma)}\left[\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s'} R_{ss'}^{\text{switch}} + \frac{\gamma\sum\limits_{s''\in \mathscr{S}^{\text{int}}}\sum\limits_{s'\in\mathscr{S}^{\text{int}}\setminus s''} R_{s's''}^{\text{switch}}}{(n-1)-\gamma(n-2)}\right]$$
$$+\frac{\gamma R_{s_ls_l}^{\text{stay}}\left[(n-1+\gamma)(n-\left|k_1\right|-1)\gamma + (n-1+\gamma)(n-1+2\gamma-n\gamma) - (n-1+\gamma-\gamma\left|k_1\right|)\gamma\right]}{(1-\gamma)(n-1)(n-1-\gamma(n-2))(n-1-\gamma(\left|k_1\right|-1))}$$

Upon factoring we get,

$$= \frac{1}{(n-1+\gamma)} \left[\sum_{s' \in \mathscr{S}^{\text{int}} \setminus s'} R_{ss'}^{\text{switch}} + \frac{\gamma \sum\limits_{s'' \in \mathscr{S}^{\text{int}}} \sum\limits_{s' \in \mathscr{S}^{\text{int}} \setminus s''} R_{s's''}^{\text{switch}}}{(n-1) - \gamma(n-2)} \right]$$
$$+ \frac{\gamma R_{s_l s_l}^{\text{stay}} \left[(n-1)(n-1-\gamma(|k_1|-1)) \right]}{(1-\gamma)(n-1)(n-1-\gamma(n-2))(n-1-\gamma(|k_1|-1))}$$
$$= Q_{s \in k_1}^{\text{switch},\pi^{\text{opt}}}$$

Hence, $Q_{s \in k_0}^{\text{switch},\pi^{\text{int}} \neq \pi^{\text{opt}}} < Q_{s \in k_1}^{\text{switch},\pi^{\text{opt}}}$.

Proof for 2-(a): $Q_{s \in k_1}^{\text{stay},\pi^{\text{int}} \neq \pi^{\text{opt}}} = R_{ss}^{\text{stay}} + \gamma Q_s^{\text{switch},\pi^{\text{int}} \neq \pi^{\text{opt}}} < R_{ss}^{\text{stay}} + \gamma Q_s^{\text{switch},\pi^{\text{opt}}} = Q_{s \in k_1}^{\text{stay},\pi^{\text{opt}}}$. Hence, $Q_{s \in k_1}^{\text{stay},\pi^{\text{int}} \neq \pi^{\text{opt}}} < Q_{s \in k_1}^{\text{stay},\pi^{\text{opt}}}$.

Proof for 2-(b): From Eq. (A.18) we have, $Q_{s \in k_0}^{\text{stay},\pi^{\text{int}} \neq \pi^{\text{opt}}} = \frac{R_{ss}^{\text{stay}}}{1-\gamma} = R_{ss}^{\text{stay}} + \frac{\gamma R_{ss}^{\text{stay}}}{1-\gamma}$. Using the condition $R_{ss}^{\text{stay}} < \frac{\gamma R_{s_l s_l}^{\text{stay}}}{(n-1)-\gamma(n-2)}$, $\forall s \in \mathscr{S}^{\text{int}} \setminus s_l$, we get,

$$Q_{s \in k_0}^{\text{stay},\pi^{\text{int}} \neq \pi^{\text{opt}}} < R_{ss}^{\text{stay}} + \gamma \left(\frac{\gamma R_{s_l s_l}^{\text{stay}}}{(1-\gamma)(n-1-\gamma(n-2))} \right) < R_{ss}^{\text{stay}} + \gamma Q_s^{\text{switch},\pi^{\text{opt}}} = Q_{s \in k_1}^{\text{stay},\pi^{\text{opt}}}.$$

Hence, $Q_{s \in k_0}^{\text{stay},\pi^{\text{int}} \neq \pi^{\text{opt}}} < Q_{s \in k_1}^{\text{stay},\pi^{\text{opt}}}$.

Proof for 3-(a): For the optimal policy π^{opt}, the set $k_0 = \{s_l\}$. Therefore, if $s \in k_0$, then $s = s_l$. Substituting this in the inequality 3-(a) that needs to proved, we get,

$$Q_{s_l \in k_1}^{\text{switch},\pi^{\text{int}} \neq \pi^{\text{opt}}} < Q_{s_l}^{\text{switch},\pi^{\text{opt}}}. \tag{A.27}$$

For the policy π^{int}, since $s_l \in k_1$, this implies $s_l \notin k_0$. From Eq. (A.19) we have,

$$Q_{s_l \in k_1}^{\text{switch},\pi^{\text{int}} \neq \pi^{\text{opt}}} = \frac{1}{(n-1+\gamma)} \left[\sum_{s' \in \mathscr{S}^{\text{int}} \setminus s} R_{ss'}^{\text{switch}} + \frac{\gamma}{\left(n-1-\gamma(|k_1|-1)\right)} \sum_{s'' \in k_1} \sum_{s' \in \mathscr{S}^{\text{int}} \setminus s''} R_{s's''}^{\text{switch}} \right]$$
$$+ \frac{\gamma}{(1-\gamma)\left(n-1-\gamma(|k_1|-1)\right)} \sum_{s'' \in k_0} R_{s''s''}^{\text{stay}}$$

Substituting the following:

- $\left|k_1\right| < n-1$,
- since R has all non-negative entries $\sum_{s''\in k_1}\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s''} R^{\text{switch}}_{s's''} < \sum_{s''\in\mathscr{S}^{\text{int}}}\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s''} R^{\text{switch}}_{s's''}$,
- using the condition $R^{\text{stay}}_{ss} < \frac{\gamma R^{\text{stay}}_{s_l s_l}}{(n-1)-\gamma(n-2)}$, $\forall s \in \mathscr{S}^{\text{int}}\setminus s_l$, and
- since $s_l \notin k_0$, $\sum_{s''\in k_0} R^{\text{stay}}_{s''s''} < \frac{\gamma\left|k_0\right| R^{\text{stay}}_{s_l s_l}}{(n-1)-\gamma(n-2)}$, we get,

$$< \frac{1}{(n-1)}\left[\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s} R^{\text{switch}}_{ss'} + \frac{\gamma}{(n-1-\gamma(n-2))}\sum_{s''\in\mathscr{S}^{\text{int}}}\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s''} R^{\text{switch}}_{s's''}\right] + \frac{\gamma(n-\left|k_1\right|)}{(1-\gamma)\left(n-1-\gamma(\left|k_1\right|-1)\right)}\frac{\gamma R^{\text{stay}}_{s_l s_l}}{(n-1-\gamma(n-2))}$$

Since $\left|k_1\right| \geq 1$ and $\gamma \leq 1$, it can be easily shown that $(n-\left|k_1\right|) \leq (n-1-\gamma(\left|k_1\right|-1))$. Using this result, we get,

$$\leq \frac{1}{(n-1)}\left[\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s} R^{\text{switch}}_{ss'} + \frac{\gamma}{(n-1-\gamma(n-2))}\sum_{s''\in\mathscr{S}^{\text{int}}}\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s''} R^{\text{switch}}_{s's''}\right] + \frac{\gamma}{(1-\gamma)}\frac{\gamma R^{\text{stay}}_{s_l s_l}}{(n-1-\gamma(n-2))}$$

$$= \frac{1}{(n-1)}\left[\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s} R^{\text{switch}}_{ss'} + \frac{\gamma}{(n-1-\gamma(n-2))}\sum_{s''\in\mathscr{S}^{\text{int}}}\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s''} R^{\text{switch}}_{s's''}\right] + \frac{\gamma R^{\text{stay}}_{s_l s_l}}{(1-\gamma)(n-1)}\frac{(n-1)+\gamma-(n-1-\gamma(n-2))}{(n-1-\gamma(n-2))}$$

$$= \frac{1}{(n-1)}\left[\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s} R^{\text{switch}}_{ss'} + \frac{\gamma}{(n-1-\gamma(n-2))}\sum_{s''\in\mathscr{S}^{\text{int}}}\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s''} R^{\text{switch}}_{s's''}\right] + \frac{\gamma R^{\text{stay}}_{s_l s_l}}{(1-\gamma)(n-1)}\frac{n-1+\gamma}{(n-1-\gamma(n-2))} - \frac{\gamma R^{\text{stay}}_{s_l s_l}}{(1-\gamma)(n-1)} = Q^{\text{switch},\pi^{\text{opt}}}_{s\in k_0}$$

Hence, $Q^{\text{switch},\pi^{\text{int}}\neq\pi^{\text{opt}}}_{s\in k_1} < Q^{\text{switch},\pi^{\text{opt}}}_{s\in k_0}$.

Proof for 3-(b): As discussed in the Proof for 3-(a), $s = s_l$. Substituting the condition $\left|k_1\right| < n-1$ in Eq. (A.20), we get,

$$Q^{\text{switch},\pi^{\text{int}}\neq\pi^{\text{opt}}}_{s_l\in k_0} = \frac{1}{(n-1)}\left[\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s} R^{\text{switch}}_{ss'} + \frac{\gamma}{\left(n-1-\gamma\left(\left|k_1\right|-1\right)\right)}\sum_{s''\in k_1}\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s''} R^{\text{switch}}_{s's''}\right]$$
$$+\frac{\gamma}{(1-\gamma)(n-1)}\left[\frac{(n-1+\gamma)}{\left(n-1-\gamma\left(\left|k_1\right|-1\right)\right)}\sum_{s''\in k_0} R^{\text{stay}}_{s''s''}\right] - \frac{\gamma}{(n-1)(1-\gamma)}R^{\text{stay}}_{s_l s_l}$$
$$< \frac{1}{(n-1)}\left[\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s} R^{\text{switch}}_{ss'} + \frac{\gamma}{(n-1-\gamma(n-2))}\sum_{s''\in k_1}\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s''} R^{\text{switch}}_{s's''}\right]$$
$$+\frac{\gamma}{(1-\gamma)(n-1)}\left[\frac{(n-1+\gamma)}{\left(n-1-\gamma\left(\left|k_1\right|-1\right)\right)}\sum_{s''\in k_0} R^{\text{stay}}_{s''s''}\right] - \frac{\gamma}{(n-1)(1-\gamma)}R^{\text{stay}}_{s_l s_l}$$

Substituting $\sum_{s''\in k_0} R^{\text{stay}}_{s''s''} = \sum_{s''\in k_0\setminus s_l} R^{\text{stay}}_{s''s''} + R^{\text{stay}}_{s_l s_l}$ and the condition $R^{\text{stay}}_{ss} < \frac{\gamma R^{\text{stay}}_{s_l s_l}}{(n-1)-\gamma(n-2)}$, $\forall s \in \mathscr{S}^{\text{int}}\setminus s_l$ in the second term of R.H.S. and solving, we get,

$$< \frac{1}{(n-1)}\left[\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s} R^{\text{switch}}_{ss'} + \frac{\gamma}{(n-1-\gamma(n-2))}\sum_{s''\in k_1}\sum_{s'\in\mathscr{S}^{\text{int}}\setminus s''} R^{\text{switch}}_{s's''}\right]$$
$$+\frac{\gamma R^{\text{stay}}_{s_l s_l}}{(1-\gamma)(n-1)}\frac{n-1+\gamma}{(n-1-\gamma(n-2))} - \frac{\gamma R^{\text{stay}}_{s_l s_l}}{(1-\gamma)(n-1)} = Q^{\text{switch},\pi^{\text{opt}}}_{s_l\in k_0}$$

Hence, $Q^{\text{switch},\pi^{\text{int}}\neq\pi^{\text{opt}}}_{s\in k_0} < Q^{\text{switch},\pi^{\text{opt}}}_{s\in k_0}$.

Proof for 4-(a)&(b): This proof is straightforward since,

$$Q^{\text{stay},\pi^{\text{int}}\neq\pi^{\text{opt}}}_{s\in k_1 \text{or} k_0} < \max(Q^{\pi^{\text{opt}}}) = Q^{\text{stay},\pi^{\text{opt}}}_{s\in k_0}.$$

Hence, $Q^{\text{stay},\pi^{\text{int}}\neq\pi^{\text{opt}}}_{s\in k_1} < Q^{\text{stay},\pi^{\text{opt}}}_{s\in k_0}$ & $Q^{\text{stay},\pi^{\text{int}}\neq\pi^{\text{opt}}}_{s\in k_0} < Q^{\text{stay},\pi^{\text{opt}}}_{s\in k_0}$. □

For a set of mild conditions, the convergence of the algorithm's policy π^{int} and the adaptive abstraction $\widehat{\phi}$ to their respective optimal fixed-points is proved next.

Theorem 4. *Let $\{\pi^{int}_t\}_{t\in\mathbb{N}}$ denote the sequence of observation stream selection policies generated by the algorithm for $\epsilon = 1$. If Conditions (4.13),(4.14),(4.15) and (4.16) hold,*

then, for $t > t_0$, $$\lim_{t\to\infty} \pi^{int}_t(s) = \pi^*(\phi^*, s),\ \forall s \in \mathscr{S}^{int}$$

Proof. From Eq. (4.6), the estimation error of the IncSFA algorithm at time t is $\xi^{\text{sfa}}(t) = \|\widehat{\phi}_t - \widehat{\phi}_{t-1}\|$. Let $\dot{\xi}^{\text{sfa}}(t) = \big(\xi^{\text{sfa}}(t) - \xi^{\text{sfa}}(t-1)\big)$ be the *backward-difference* approximation of the derivative of the estimation error. The condition $\epsilon = 1$ implies that the agent executes actions (*stay* or *switch*) randomly (uniformly) at each state. To execute the *stay* action in the current state, the agent was either in the same state or has shifted to the current state from another state at the previous time step. Therefore, the expected learning progress made by the IncSFA is higher for the *stay* action as compared to the *switch* action.

$$\begin{aligned}\mathbb{E}&\left[-\sum_{\tau} \dot{\xi}^{\text{sfa}}(t) \,\middle|\, s(t) = s_i, s(t-1) = s_i \right] \\ &\geq \mathbb{E}\left[-\sum_{\tau} \dot{\xi}^{\text{sfa}}(t) \,\middle|\, s(t) = s_i, s(t-1) = s_j \right], \ \forall s_i, s_j \in \mathscr{S}^{\text{int}},\ i \neq j.\end{aligned} \tag{A.28}$$

The equality might hold for random, constant, or similar observation streams. Using the convergence Condition (4.15) we get,

$$\begin{aligned}\mathbb{E}&\left[-\sum_{\tau} \dot{\xi}^{\text{sfa}}(t) \,\middle|\, s(t) = s_l, s(t-1) = s_l \right] \\ &> \mathbb{E}\left[-\sum_{\tau} \dot{\xi}^{\text{sfa}}(t) \,\middle|\, s(t) = s_i, s(t-1) = s_j \right], \ \forall s_i, s_j \in \mathscr{S}^{\text{int}}.\end{aligned} \tag{A.29}$$

From Eq. (A.29) and Eq. (4.9), if σ is small $\left(\sum_{\tau} \big(\beta Z(|\delta - \xi^{roc}|\big) \approx 0\right)$, we get,

$$\lim_{t\to\infty} R_t(s_l,\ \text{stay}, s_l) = \max(\lim_{t\to\infty} R_t). \tag{A.30}$$

In Lemma 4, I show that the error plot of a converging IncSFA abstraction resembles that of an exponential decay function. Using this result and Condition (4.15) we get,

$$\lim_{t\to\infty} R_t(s_i,\ \text{stay}, s_i) = \mathbb{E}\left[-\sum_{\tau} \dot{\xi}^{\text{sfa}}(t) \,\middle|\, s(t) = s_i, s(t-1) = s_i \right] \underset{\sim}{\propto} \frac{1}{\bar{\tau}_i^{1/2}} \underset{\sim}{\propto} ln(\Omega(\mathbf{x}_i)). \tag{A.31}$$

Using Eqs. (A.30), (A.31), Condition (4.16) in Lemma 6 and Theorem 3, we get for $t > t_0$,

$$\lim_{t\to\infty} \pi_t^{\text{int}}(s) = \pi^*(\phi^*, s),\ \forall s \in \mathscr{S}^{\text{int}}$$

Since the policies π_t^{int} and $\pi^*(\phi^*)$ are binary-vectors, it follows that $\exists t_c \in \mathbb{N}\ (t_0 < t_c < \infty)$, s.t. for $t = t_c$, $\pi_t^{\text{int}} = \pi^*(\phi^*)$. □

Theorem 5. *Let $\{\widehat{\phi}_t\}_{t\in\mathbb{N}}$ denote the sequence of adaptive abstractions generated by the algorithm for $\epsilon = 0$. If $t_c(> t_0) \in \mathbb{N}$ is the time when $\pi_t^{int} = \pi^*(\phi^*)$ and if Conditions (4.13),(4.14),(4.15) and (4.16) hold,*

$$\textit{then, for } t > t_c, \qquad \lim_{t\to\infty} \widehat{\phi}_t = \phi^*$$

Proof. When $\epsilon = 0$, the agent exploits the observation stream selection policy (π^{int}). Therefore, the agent observes samples from $\mathbf{x}_l$ and IncSFA-ROC makes learning progress. As the ROC estimation error approaches the threshold $\xi^{\text{roc}} \to \delta$, the term $\beta \sum_{\tau} (Z(|\delta - \xi^{roc}|) \to \tau\beta$. By selecting a β close to $\frac{\sqrt{2\pi}\sigma\mathscr{L}}{\tau}$, for $t > t_c$, $R_t(s_l,\ \text{stay}, s_l) = \max(R_t) \implies \pi_t^{\text{int}} = \pi^*(\phi^*)$. Therefore, from Theorem 2, it follows that $\lim_{t\to\infty} \widehat{\phi}_t = \pm\phi^*$. □

Convergence Condition (4.16) does not involve setting any algorithm parameter and is a direct condition on the observation stream complexity. The rest of the analysis discusses the scenario when a few of the observation streams violate the condition.

Definition 3. *Let $r = \frac{\gamma}{(n-1-\gamma(n-2))}$. A stream $\mathbf{x}$ is r-dominated by another stream $\mathbf{x}'$ if $ln\left(\Omega(\mathbf{x})\right) < r\ ln\left(\Omega(\mathbf{x}')\right)$.*

Theorem 6. *Let $\{\pi_t^{int}\}_{t\in\mathbb{N}}$ denote the sequence of observation stream selection policies generated by the algorithm for $\epsilon = 1$. Let $\mathscr{S}_r^{int}$ be the set of states whose observation streams are not r-dominated by $\mathbf{x}_l$. If Conditions (4.13),(4.14) and (4.15) hold, then, for $t > t_0$, $\pi_t^{int}(s)$ has two limits points equal to $\left(1 - \mathbb{1}_{\{s_l\}}(s)\right)$ or $\left(1 - \mathbb{1}_{\mathscr{S}_r^{int}}(s)\right)$, $\forall s \in \mathscr{S}^{int}$.*

Proof. From Theorem 4, we get $\frac{R_{ss}^{\text{stay}}}{R_{s_ls_l}^{\text{stay}}} \geq \frac{\gamma}{(n-1-\gamma(n-2))}$, $\forall s \in \mathscr{S}_r^{\text{int}} \setminus s_l$.

Let $R_{ss}^{\text{stay}} = \frac{\gamma R_{s_ls_l}^{\text{stay}}}{(n-1-\gamma(n-2))} + \epsilon_s$, where ϵ_s, $\forall s \in \mathscr{S}_r^{\text{int}} \setminus s_l$ are non-negative constants. Let $\pi^* = 1 - \mathbb{1}_{\{s_l\}}(s)$ and $\widehat{\pi} = 1 - \mathbb{1}_{\mathscr{S}_r^{\text{int}}}(s)$, $\forall s \in \mathscr{S}^{\text{int}}$. From Eq. (A.19) and substituting for R_{ss}^{stay} we have,

$$
\begin{aligned}
Q_{s\in k_1}^{\text{switch},\widehat{\pi}} = \frac{1}{(n-1+\gamma)} &\left[\sum_{s'\in\mathscr{S}^{\text{int}}\backslash s} R_{ss'}^{\text{switch}} + \frac{\gamma}{\left(n-1-\gamma(\left|k_1\right|-1)\right)}\sum_{s''\in k_1}\sum_{s'\in\mathscr{S}^{\text{int}}\backslash s''} R_{s's''}^{\text{switch}}\right] \\
&+\frac{\gamma}{(1-\gamma)\left(n-1-\gamma(\left|k_1\right|-1)\right)}\sum_{s''\in k_0} R_{s''s''}^{\text{stay}}
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{1}{(n-1+\gamma)} \left[\sum_{s'\in\mathscr{S}^{\text{int}}\backslash s} R_{ss'}^{\text{switch}} + \frac{\gamma}{\left(n-1-\gamma(\left|k_1\right|-1)\right)}\sum_{s''\in k_1}\sum_{s'\in\mathscr{S}^{\text{int}}\backslash s''} R_{s's''}^{\text{switch}}\right] \\
&\quad +\frac{\gamma}{(1-\gamma)\left(n-1-\gamma(\left|k_1\right|-1)\right)}\left(R_{s_ls_l}^{\text{stay}} + \sum_{s''\in k_0\backslash s_l} R_{s''s''}^{\text{stay}}\right) \\
&= \frac{1}{(n-1+\gamma)} \left[\sum_{s'\in\mathscr{S}^{\text{int}}\backslash s} R_{ss'}^{\text{switch}} + \frac{\gamma}{\left(n-1-\gamma(\left|k_1\right|-1)\right)}\sum_{s''\in k_1}\sum_{s'\in\mathscr{S}^{\text{int}}\backslash s''} R_{s's''}^{\text{switch}}\right] \\
&\quad +\frac{\gamma}{(1-\gamma)\left(n-1-\gamma(\left|k_1\right|-1)\right)}\left[\left(\frac{\gamma(\left|k_0\right|-1)}{(n-1)-\gamma(n-2)}+1\right)R_{s_ls_l}^{\text{stay}} + \sum_{s''\in k_0\backslash s_l}\epsilon_{s''}\right]
\end{aligned}
$$

$$
\begin{aligned}
&= Q_{s\in k_1}^{\text{switch},\pi^*} - \frac{\gamma}{(n-1+\gamma)}\left[\frac{\sum_{s''\in\mathscr{S}^{\text{int}}}\sum_{s'\in\mathscr{S}^{\text{int}}\backslash s''} R_{s's''}^{\text{switch}}}{\left(n-1-\gamma(n-2)\right)} - \frac{\sum_{s''\in k_1}\sum_{s'\in\mathscr{S}^{\text{int}}\backslash s''} R_{s's''}^{\text{switch}}}{\left(n-1-\gamma(\left|k_1\right|-1)\right)}\right] \\
&\quad + \frac{\gamma}{(1-\gamma)\left(n-1-\gamma(\left|k_1\right|-1)\right)}\sum_{s''\in k_0\backslash s_l}\epsilon_{s''} \\
&= Q_{s\in k_1}^{\text{switch},\pi^*} - A + B
\end{aligned}
$$

Both A and B are non-zero. So, clearly when $B > A$, $Q_{s\in k_1}^{\text{switch},\widehat{\pi}} > Q_{s\in k_1}^{\text{switch},\pi^*}$. Therefore, $\arg\max_{\pi^{\text{int}}} Q^{\pi^{\text{int}}} \neq \pi^*$. Evaluating similarly for $Q_{s\in k_1 \text{ or } k_0}^{\text{stay or switch},\widehat{\pi}}$ we get, for the condition $B > A$, $\arg\max_{\pi^{\text{int}}} Q^{\pi^{\text{int}}} = \widehat{\pi}$. □

Bibliography

H. Abut, editor. *Vector Quantization*. IEEE Press, Piscataway, NJ, 1990.

S. I. Amari. Neural theory of association and concept-formation. *Biological Cybernetics*, 26(3):175–185, 1977.

W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Q. Appl. Math*, 9(17):17–29, 1951.

G. Aston-Jones and J. D. Cohen. An integrative theory of locus coeruleus-norepinephrine function: adaptive gain and optimal performance. *Annu. Rev. Neurosci.*, 28:403–450, 2005.

G. Aston-Jones, C. Chiang, and T. Alexinsky. Discharge of noradrenergic locus coeruleus neurons in behaving rats and monkeys suggests a role in vigilance. *Progress in Brain Research*, 88:501–520, 1991.

B. Bakker and J. Schmidhuber. Hierarchical reinforcement learning based on subgoal discovery and subpolicy specialization. In F. Groen et al., editor, *Proc. 8th Conference on Intelligent Autonomous Systems IAS-8*, pages 438–445, Amsterdam, NL, 2004. IOS Press.

A. Baranes and P. Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1): 49–73, 2013.

H. Barlow. Redundancy reduction revisited. *Network: Computation in Neural Systems*, 12(3):241–253, 2001.

J. S. Bergstra and Y. Bengio. Slow, decorrelated features for pretraining complex cell-like networks. *Advances in Neural Information Processing Systems 22*, pages 99–107, 2009.

D. P. Bertsekas and J. N. Tsitsiklis. Neuro-dynamic programming: an overview. In *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, volume 1, pages 560–564. IEEE, 1995.

Boston-Dynamics. Petman (protection ensemble test mannequin) humanoid military robot: http://www.bostondynamics.com/robot_petman.html.

S. J. Bradtke and A. G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(1):33–57, 1996. ISSN 0885-6125.

G. Bush, B. A. Vogt, J. Holmes, A. M. Dale, D. Greve, M. A. Jenike, and B. R. Rosen. Dorsal anterior cingulate cortex: a role in reward-based decision making. *Proceedings of the National Academy of Sciences*, 99(1):523–528, 2002.

C. Chatterjee, Z. Kang, and V. P. Roychowdhury. Algorithms for accelerated convergence of adaptive pca. *IEEE Transactions on Neural Networks*, 11(2):338–355, 2000.

T. Chen, S. I. Amari, and Q. Lin. A unified algorithm for principal and minor components extraction. *Neural Networks*, 11(3):385–390, 1998.

T. Chen, S. I. Amari, and N. Murata. Sequential extraction of minor components. *Neural Processing Letters*, 13(3):195–201, 2001. ISSN 1370-4621.

J. D. Cohen and R. C. O' Reilly. A preliminary theory of the interactions between prefrontal cortex and hippocampus that contribute to planning and prospective memory. *Prospective memory: Theory and applications*, pages 267–295, 1996.

J. D. Cohen, G. Aston-Jones, and M. S. Gilzenrat. A systems-level perspective on attention and cognitive control: Guided activation, adaptive gating, conflict monitoring, and exploitation vs. exploration, chapter 6. *In M. I. Posner (Ed.), Cognitive*, pages 71–90, 2004.

P. Comon. Independent component analysis, A new concept? *Signal Processing*, 36:287–314, 1994.

Y. Cui and J. Weng. Appearance-based hand sign recognition from intensity image sequences. *Computer Vision and Image Understanding*, 78(2):157–176, 2000.

P. Dayan and L. F. Abbott. *Theoretical neuroscience: Computational and mathematical modeling of neural systems*. MIT press, 2001.

P. Dayan and A. J. Yu. Phasic norepinephrine: a neural interrupt signal for unexpected events. *Network: Computation in Neural Systems*, 17(4):335–350, 2006.

C. Doersch, T. S. Lee, G. Huang, and E. L. Miller. Temporal continuity learning for convolutional deep belief networks. 2010. URL http://www.cs.cmu.edu/afs/cs.cmu.edu/user/mjs/ftp/thesis-program/2010/theses/doersch.pdf.

P. Földiák. Learning invariance from transformation sequences. *Neural Computation*, 3(2):194–200, 1991.

P. Földiák and M. P. Young. Sparse coding in the primate cortex. *The handbook of brain theory and neural networks*, 1:895–898, 1995.

E. W. Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, 21:768–769, 1965.

G. E. Forsythe and P. Henrici. *The cyclic Jacobi method for computing the principal values of a complex matrix.* Applied Mathematics and Statistics Laboratories, Stanford University, 1958.

M. Franzius, H. Sprekeler, and L. Wiskott. Slowness and sparseness lead to place, head-direction, and spatial-view cells. *PLoS Computational Biology*, 3(8):e166, 2007.

M. Fyhn, T. Hafting, A. Treves, M. B. Moser, and E. I. Moser. Hippocampal remapping and grid realignment in entorhinal cortex. *Nature*, 446(7132):190–194, 2007.

Z. Gábor, Z. Kalmár, and C. Szepesvári. Multi-criteria reinforcement learning. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 197–205. Morgan Kaufmann Publishers Inc., 1998.

L. Gisslén, M. Luciw, V. Graziano, and J. Schmidhuber. Sequential constant size compressors for reinforcement learning. In *Artificial General Intelligence*, pages 31–40. Springer, 2011.

C. W. Groetsch. Lanczos' generalized derivative. *American Mathematical Monthly*, 4(105), 1998. ISSN 320-326.

S. Grossberg. How does a brain build a cognitive code?. *Psychological Review*, 87 (1):1, 1980.

I. D. Guedalia, M. London, and M. Werman. An on-line agglomerative clustering method for nonstationary data. *Neural Computation*, 11(2):521–540, 1999.

T. Hafting, M. Fyhn, S. Molden, M. Moser, and E.I. Moser. Microstructure of a spatial map in the entorhinal cortex. *Nature*, 7052:801, 2005.

S. Hart, S. Sen, and R. A. Grupen. Intrinsically motivated hierarchical manipulation. In *Proceedings of the 2008 IEEE Conference on Robots and Automation (ICRA)*, pages 3814–3819, 2008.

S. W. Hart. *The development of hierarchical knowledge in robot systems*. PhD thesis, University of Massachusetts Amherst, 2009.

G. E. Hinton. Connectionist learning procedures. *Artificial Intelligence*, 40(1-3): 185–234, 1989.

G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, August 2002.

S. Höfer, M. Spranger, and M. Hild. Posture recognition based on slow feature analysis. In *Language Grounding in Robots*, pages 111–130. Springer, 2012.

Honda. Asimo robot: `http://world.honda.com/ASIMO`.

M. Huber and R. A. Grupen. A hybrid discrete event dynamic systems approach to robot control. *Univ. Mass., Dept. Comput. Sci., Amherst, MA, Tech. Rep*, pages 96–43, 1996.

A. Hyvärinen, J. Karhunen, and E. Oja. *Independent component analysis*, volume 26. Wiley-interscience, 2001.

S. Ishii, W. Yoshida, and J. Yoshimoto. Control of exploitation–exploration meta-parameter in reinforcement learning. *Neural Networks*, 15(4):665–687, 2002.

O. C. Jenkins and M. J. Matarić. A spatio-temporal extension to isomap nonlinear dimension reduction. In *Proceedings of the twenty-first international conference on Machine learning*, page 56. ACM, 2004.

I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2005.

I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: a survey. *Journal of AI research*, 4:237–285, 1996.

S. Kakade and P. Dayan. Dopamine: generalization and bonuses. *Neural Networks*, 15(4):549–559, 2002.

M. J. Kane and R. W. Engle. The role of prefrontal cortex in working-memory capacity, executive attention, and general fluid intelligence: An individual-differences perspective. *Psychonomic Bulletin & Review*, 9(4):637–671, 2002.

F. Kaplan and P. Y. Oudeyer. In search of the neural circuits of intrinsic motivation. *Frontiers in Neuroscience*, 1(1):225, 2007.

M. Klapper-Rybicka, N. N. Schraudolph, and J. Schmidhuber. Unsupervised learning in lstm recurrent neural networks. In *Lecture Notes on Comp. Sci. 2130, Proc. Intl. Conf. on Artificial Neural Networks (ICANN-2001)*, pages 684–691. Springer: Berlin, Heidelberg, 2001.

T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berlin, 3rd edition, 2001.

V. R. Kompella, M. Luciw, and J. Schmidhuber. Incremental slow feature analysis. In *Proc. 20th International Joint Conference of Artificial Intelligence (IJCAI)*, pages 1354–1359, 2011a.

V. R. Kompella, L. Pape, J. Masci, M. Frank, and J. Schmidhuber. Autoincsfa and vision-based developmental learning for humanoid robots. In *IEEE-RAS International Conference on Humanoid Robots*, pages 622–629, Bled, Slovenia, 2011b.

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, 2012a.

V. R. Kompella, M. Luciw, M. Stollenga, L. Pape, and J. Schmidhuber. Autonomous learning of abstractions using curiosity-driven modular incremental slow feature analysis. In *Proc. of the Joint Conference on Development and Learning and Epigenetic Robotics (ICDL-EPIROB)*, pages 1–8, San Diego, 2012b. IEEE.

V. R. Kompella*, M. Luciw*, S. Kazerounian, and J. Schmidhuber. An intrinsic value system for developing multiple invariant representations with incremental slowness learning. *Frontiers in Neurorobotics, *Joint First Authors*, 7, 2013.

V. R. Kompella, S. Kazerounian, and J. Schmidhuber. An anti-hebbian learning rule to represent drive motivations for reinforcement learning. In *From Animals to Animats 13*, pages 176–187. Springer, 2014a.

V. R. Kompella, M. F. Stollenga, M. Luciw, and J. Schmidhuber. Explore to see, learn to perceive, get the actions for free: Skillability. In *International Joint Conference on Neural Networks (IJCNN)*, pages 2705–2712. IEEE, 2014b.

G. Konidaris and A. G. Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. In *Advances in Neural Information Processing Systems*, pages 1015–1023, 2009.

G. Konidaris, S. Kuindersma, A. G. Barto, and R. Grupen. Constructing skill trees for reinforcement learning agents from demonstration trajectories. In *Advances in Neural Information Processing Systems*, pages 1162–1170, 2010.

G. Konidaris, S. Kuindersma, R. Grupen, and A. G. Barto. Autonomous skill acquisition on a mobile manipulator. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 1468–1473, 2011.

T. P. Krasulina. Method of stochastic approximation in the determination of the largest eigenvalue of the mathematical expectation of random matrices. *Automat. Remote Contr*, 2:215–221, 1970.

E. Kreyszig. *Advanced engineering mathematics*. Wiley, New York, 1988.

B. Kuipers. The spatial semantic hierarchy. *Artificial intelligence*, 119(1):191–233, 2000.

B. Kuipers, J. Modayil, P. Beeson, M. MacMahon, and F. Savelli. Local metrical and global topological maps in the hybrid spatial semantic hierarchy. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 5, pages 4845–4851. IEEE, 2004.

M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *The Journal of Machine Learning Research*, 4:1107–1149, 2003.

D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

H. Lee, Y. Largman, P. Pham, and A.Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems*, pages 1096–1104.

R. Legenstein, N. Wilbert, and L. Wiskott. Reinforcement learning on slow features of high-dimensional input streams. *PLoS Computational Biology*, 6(8), 2010. ISSN 1553-734X.

S. Lindstädt. Comparison of two unsupervised neural network models for redundancy reduction. In M. C. Mozer, P. Smolensky, D. S. Touretzky, J. L. Elman, and A. S. Weigend, editors, *Proc. of the 1993 Connectionist Models Summer School*, pages 308–315. Hillsdale, NJ: Erlbaum Associates, 1993.

M. Luciw and J. Schmidhuber. Low complexity proto-value function learning from sensory observations with incremental slow feature analysis. In *Proc. 22nd International Conference on Artificial Neural Networks (ICANN)*, pages 279–287, Lausanne, 2012. Springer.

M. Luciw, V. R. Kompella, and J. Schmidhuber. Hierarchical incremental slow feature analysis. In *Workshop on Deep Hierarchies in Vision*, Vienna, 2012.

S. Mahadevan. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning*, 22(1-3):159–195, 1996.

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.

I. Menache, S. Mannor, and N. Shimkin. Q-cut–dynamic discovery of sub-goals in reinforcement learning. In *Machine Learning: ECML 2002*, pages 295–306. Springer, 2002.

G. Metta, G. Sandini, D. Vernon, L. Natale, and F. Nori. The icub humanoid robot: An open platform for research in embodied cognition. In *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, PerMIS '08, pages 50–56, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-293-1. doi: http://doi.acm.org/10.1145/1774674.1774683. URL http://doi.acm.org/10.1145/1774674.1774683.

M. Meunier, J. Bachevalier, and M. Mishkin. Effects of orbital frontal and anterior cingulate lesions on object and spatial memory in rhesus monkeys. *Neuropsychologia*, 1997.

E. K. Miller. The prefrontal cortex and cognitive control. *Nature Reviews Neuroscience*, 1(1):59–66, 2000.

G. Mitchison. Removing time variation with the anti-hebbian differential synapse. *Neural Computation*, 3(3):312–320, 1991.

J. Modayil and B. Kuipers. Bootstrap learning for object discovery. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 1, pages 742–747. IEEE, 2004.

J. Modayil and B. Kuipers. The initial development of object knowledge by a learning robot. *Robotics and autonomous systems*, 56(11):879–890, 2008.

J. Mugan and B. Kuipers. Autonomous learning of high-level states and actions in continuous environments. *IEEE Transactions on Autonomous Mental Development*, 4(1):70–86, 2012.

H. Ngo, M. Luciw, A. Förster, and J. Schmidhuber. Learning skills from play: Artificial curiosity on a Katana robot arm. In *Proceedings of the International Joint Conference of Neural Networks (IJCNN)*, pages 1–8, June 2012. doi: 10.1109/IJCNN.2012.6252824.

H. Ngo, M. Luciw, A. Förster, and J. Schmidhuber. Confidence-based progress-driven self-generated goals for skill acquisition in developmental robots. *Frontiers in Psychology*, 4, 2013. ISSN 1664-1078.

G. R. Norman and H. G Schmidt. The psychological basis of problem-based learning: a review of the evidence. *Academic medicine*, 67(9):557–65, 1992.

E. Oja. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3):267–273, 1982.

E. Oja. On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of Mathematical Analysis and Applications*, 106:69–84, 1985.

E. Oja. Neural networks, principal components, and subspaces. *International journal of neural systems*, 1(1):61–68, 1989.

E. Oja. Principal components, minor components, and linear neural networks. *Neural Networks*, 5(6):927–935, 1992. ISSN 0893-6080.

J. O'Keefe and J. Dostrovsky. The hippocampus as a spatial map: Preliminary evidence from unit activity in the freely-moving rat. *Brain Research*, 1971.

R. C. O'Reilly and M. H. Johnson. Object recognition and sensitive periods: A computational analysis of visual imprinting. *Neural Computation*, 6(3):357–389, 1994.

L. Pape, C. M. Oddo, M. Controzzi, C. Cipriani, A. Förster, M. C. Carrozza, and J. Schmidhuber. Learning tactile skills through curious exploration. *Frontiers in neurorobotics*, 6, 2012.

A. Papoulis, S. U. Pillai, and S. Unnikrishna. *Probability, random variables, and stochastic processes*, volume 196. McGraw-hill New York, 1965.

M. Park and J. Choi. Novel incremental principal component analysis with improved performance. *Structural, Syntactic, and Statistical Pattern Recognition*, pages 592–601, 2008.

D. Peng and Z. Yi. A new algorithm for sequential minor component analysis. *International Journal of Computational Intelligence Research*, 2(2):207–215, 2006.

D. Peng, Z. Yi, and W. Luo. Convergence analysis of a simple minor component analysis algorithm. *Neural Networks*, 20(7):842–850, 2007. ISSN 0893-6080.

M. Petrik. An analysis of laplacian methods for value function approximation in mdps. In *IJCAI*, pages 2574–2579, 2007.

S. C. Rao, G. Rainer, and E. K. Miller. Integration of what and where in the primate prefrontal cortex. *Science*, 276(5313):821–824, 1997.

P. Redgrave and K. Gurney. The short-latency dopamine signal: a role in discovering novel actions? *Nature Reviews Neuroscience*, 7(12):967–975, 2006.

P. Redgrave, T. J. Prescott, and K. Gurney. Is the short-latency dopamine response too short to signal reward error? *Trends in Neurosciences*, 22(4):146–151, 1999.

M. B. Ring. *Continual Learning in Reinforcement Environments*. PhD thesis, University of Texas at Austin, 1994.

M. B. Ring. Child: A first step towards continual learning. *Machine Learning*, 28 (1):77–104, 1997.

E. T. Rolls. Spatial view cells and the representation of place in the primate hippocampus. *Hippocampus*, 9(4):467–480, 1999.

E. T. Rolls, J. Hornak, D. Wade, and J. McGrath. Emotion-related learning in patients with social and emotional changes associated with frontal lobe damage. *Journal of Neurology, Neurosurgery & Psychiatry*, 57(12):1518–1524, 1994.

E. T. Rolls, B. J. Everitt, and A. Roberts. The orbitofrontal cortex [and discussion]. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 351(1346):1433–1444, 1996.

A. Saltelli, K. Chan, E. M. Scott, et al. *Sensitivity analysis*, volume 134. Wiley New York, 2000.

T. D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural networks*, 2(6):459–473, 1989.

S. J. Sara. The locus coeruleus and noradrenergic modulation of cognition. *Nature Reviews Neuroscience*, 10(3):211–223, 2009.

J. Schmidhuber. Learning to generate sub-goals for action sequences. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, pages 967–972. Elsevier Science Publishers B.V., North-Holland, 1991a.

J. Schmidhuber. Curious model-building control systems. In *Proceedings of the International Joint Conference on Neural Networks, Singapore*, volume 2, pages 1458–1463. IEEE press, 1991b.

J. Schmidhuber. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242, 1992a.

J. Schmidhuber. Learning factorial codes by predictability minimization. *Neural Computation*, 4(6):863–879, 1992b.

J. Schmidhuber. Learning unambiguous reduced sequence descriptions. In J. E. Moody, S. J. Hanson, and R. P. Lippman, editors, *Advances in Neural Information Processing Systems 4 (NIPS 4)*, pages 291–298. Morgan Kaufmann, 1992c.

J. Schmidhuber. Artificial curiosity based on discovering novel algorithmic predictability through coevolution. In *Congress on Evolutionary Computation (CEC)*, pages 1612–1618. IEEE Press, 1999a.

J. Schmidhuber. Neural predictors for detecting and removing redundant information. In H. Cruse, J. Dean, and H. Ritter, editors, *Adaptive Behavior and Learning*. Kluwer, 1999b.

J. Schmidhuber. Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts. *Connection Science*, 18(2):173–187, 2006a.

J. Schmidhuber. Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts. *Connection Science*, 18(2):173–187, 2006b.

J. Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990-2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230 –247, 2010a. ISSN 1943-0604. doi: 10.1109/TAMD.2010.2056368.

J. Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010b.

J. Schmidhuber. Powerplay: Training an increasingly general problem solver by continually searching for the simplest still unsolvable problem. *Frontiers in psychology*, 4, 2013.

J. Schmidhuber and R. Wahnsiedler. Planning simple trajectories using neural subgoal generators. In J. A. Meyer, H. L. Roitblat, and S. W. Wilson, editors, *Proc. of the 2nd International Conference on Simulation of Adaptive Behavior*, pages 196–202. MIT Press, 1992.

W. Schultz. Predictive reward signal of dopamine neurons. *Journal of Neurophysiology*, 80(1):1–27, 1998.

W. Schultz, P. Dayan, and P. R. Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, 1997.

O. Şimşek and A. G. Barto. Skill characterization based on betweenness. In *NIPS'08*, pages 1497–1504, 2008.

S. Singh, A. G. Barto, and N. Chentanez. Intrinsically motivated reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1281–1288, 2004.

N. Sprague. Predictive projections. In *Proceedings of the International Joint Conference of Artificial Intelligence (IJCAI)*, pages 1223–1229, 2009.

H. Sprekeler. On the relation of slow feature analysis and laplacian eigenmaps. *Neural Computation*, 23(12):3287–3302, 2011.

H. Sprekeler, C. Michaelis, and L. Wiskott. Slowness: An objective for spike-timing–dependent plasticity? *PLoS Computational Biology*, 3(6):e112, 2007.

H. Sprekeler, T. Zito, and L. Wiskott. An extension of slow feature analysis for nonlinear blind source separation. *Journal of Machine Learning Research*, 15: 921–947, 2014.

J. Stober and B. Kuipers. From pixels to policies: A bootstrapping agent. In *Development and Learning, 2008. ICDL 2008. 7th IEEE International Conference on*, pages 103–108. IEEE, 2008.

M. Stollenga, L. Pape, M. Frank, J. Leitner, A. Förster, and J. Schmidhuber. Task-relevant roadmaps: a framework for humanoid motion planning. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 5772–5778. IEEE, 2013.

J. Storck, S. Hochreiter, and J. Schmidhuber. Reinforcement driven information acquisition in non-deterministic environments. In *Proceedings of the International Conference on Artificial Neural Networks, Paris*, volume 2, pages 159–164. EC2 & Cie, 1995.

A. Stout and A. G Barto. Competence progress intrinsic motivation. In *Development and Learning (ICDL), 2010 IEEE 9th International Conference on*, pages 257–262. IEEE, 2010.

L. Sun, K. Jia, T. H. Chan, Y. Fang, G. Wang, and S. Yan. Dl-sfa: Deeply-learned slow feature analysis for action recognition. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 2625–2632. IEEE, 2014.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. Cambridge, MA, MIT Press, 1998.

R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1): 181–211, 1999.

C. Szepesvári. Algorithms for reinforcement learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 4(1):1–103, 2010.

J. S. Taube, R. U. Muller, and J. B. Ranck. Head-direction cells recorded from the postsubiculum in freely moving rats. i. description and quantitative analysis. *The Journal of Neuroscience*, 10(2):420, 1990.

C. Theriault, N. Thome, and M. Cord. Dynamic scene classification: Learning motion descriptors with slow features analysis. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2603–2610. IEEE, 2013.

J. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Trans. on Automatic Control*, 42(5):674–690, 1997.

M. Usher, J. D. Cohen, D. Servan-Schreiber, J. Rajkowski, and G. Aston-Jones. The role of locus coeruleus in the regulation of cognitive performance. *Science*, 283 (5401):549–554, 1999.

P. Fitzpatrick G. Metta L. Natale V. Tikhanoff, A. Cangelosi and F. Nori. An open-source simulator for cognitive robotics research: The prototype of the icub humanoid robot simulator, 2008.

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.

G. Wallis and E.T. Rolls. Invariant face and object recognition in the visual system. *Progress in Neurobiology*, 51(2):167–194, 1997.

L. Wang and J. Karhunen. A unified neural bigradient algorithm for robust pca and mca. *International journal of neural systems*, 7(1):53, 1996.

J. Weng and N. Zhang. Optimal in-place learning and the lobe component analysis. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pages 3887–3894. IEEE, 2006.

J. Weng, Y. Zhang, and W. Hwang. Candid covariance-free incremental principal component analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):1034–1040, 2003.

R. W. White. Motivation reconsidered: the concept of competence. *Psychological review*, 66(5):297, 1959.

M. Wiering and J. Schmidhuber. HQ-learning. *Adaptive Behavior*, 6(2):219–246, 1998.

D. Wierstra, T. Schaul, J. Peters, and J. Schmidhuber. Natural evolution strategies. In *IEEE World Congress on Computational Intelligence*, pages 3381–3387. IEEE, 2008.

L. Wiskott. Estimating driving forces of nonstationary time series with slow feature analysis. *arXiv preprint cond-mat/0312317*, 2003.

L. Wiskott and T. Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, 14(4):715–770, 2002.

C. Xu. *Steps Towards the Object Semantic Hierarchy*. PhD thesis, Computer Science Department, University of Texas at Austin, 2011.

C. Xu and B. Kuipers. Towards the object semantic hierarchy. In *Development and Learning (ICDL), 2010 IEEE 9th International Conference on*, pages 39–45. IEEE, 2010.

L. Xu, E. Oja, and C.Y. Suen. Modified hebbian learning for curve and surface fitting. *Neural Networks*, 5(3):441–457, 1992.

D. Zhang, D. Zhang, S. Chen, K. Tan, and K. Tan. Improving the robustness of online agglomerative clustering method based on kernel-induce distance measures. *Neural processing letters*, 21(1):45–51, 2005.

Y. Zhang and J. Weng. Convergence analysis of complementary candid incremental principal component analysis. *Michigan State University*, 2001.

Y. Zhang, H. Wu, and L. Cheng. Some new deformation formulas about variance and covariance. In *Modelling, Identification & Control (ICMIC), 2012 Proceedings of International Conference on*, pages 987–992. IEEE, 2012.

Z. Zhang and D. Tao. Slow feature analysis for human action recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(3):436–450, 2012.