Comenius University in Bratislava

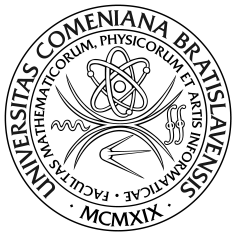
Faculty of Mathematics, Physics and Informatics

Modelling Early Sensorimotor Development with Intrinsic Motivation

2015 Ján Tóth

Comenius University in Bratislava

Faculty of Mathematics, Physics and Informatics



Modelling Early Sensorimotor Development with Intrinsic Motivation

Diploma Thesis

Study programme:	Cognitive Science
Field of study:	2503 Cognitive Science
Training work place:	Department of Applied Informatics
Supervisor:	RNDr. Martin Takáč, PhD.

Bratislava, 2015

Ján Tóth

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Ján Tóth
Študijný program: kognitívna veda (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: 9.2.11. kognitívna veda
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Modelling Early Sensorimotor Development with Intrinsic Motivation
Modelovanie raného senzomotorického vývinu s intrinzickou motiváciou

Cieľ: Vysvetlenie a overenie správania sa modelu developmentálnej robotiky Intelligent Adaptive Curiosity (IAC). Súčasťou práce je vytvoriť novú implementáciu modelu a otestovať ju na rozsahu viacerých učiacich scenárov, od jednoduchých málorozmerných až po realistickejšie. Interpretácia učiaceho procesu by mala poukázať na podobné míľniky vývinu, aké publikovali autori modelu. Práca sa pokúsi popísať daný výpočtový model v kontexte senzomotorickej fázy Piagetovej teórie kognitívneho vývinu.

Literatúra: Oudeyer, P-Y., Frédéric Kaplan, and Verena Vanessa Hafner. "Intrinsic motivation systems for autonomous mental development." Evolutionary Computation, IEEE Transactions on 11, no. 2 (2007): 265-286.

Vedúci: RNDr. Martin Takáč, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, PhD.

Dátum zadania: 31.08.2014

Dátum schválenia: 21.04.2015 prof. RNDr. Pavol Zlatoš, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Bc. Ján Tóth
Study programme:	Cognitive Science (Single degree study, master II. deg., full time form)
Field of Study:	9.2.11. Cognitive Science
Type of Thesis:	Diploma Thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Modelling Early Sensorimotor Development with Intrinsic Motivation

Aim: Explanation and verification of Intelligent Adaptive Curiosity (IAC), a developmental robotics model. Part of the thesis is a new implementation, which will be benchmarked with a variety of learning scenarios, ranging from simple low-dimensional environments and tasks to the more realistic ones. The interpretation of the learning process should point to milestones of development similar to those described by the authors of the model. The thesis will try to explain the computational model in the context of the sensorimotor stage of the Piaget's theory of Cognitive Development.

Literature: Oudeyer, P-Y., Frédéric Kaplan, and Verena Vanessa Hafner. "Intrinsic motivation systems for autonomous mental development." Evolutionary Computation, IEEE Transactions on 11, no. 2 (2007): 265-286.

Supervisor:	RNDr. Martin Takáč, PhD.
Department:	FMFI.KAI - Department of Applied Informatics
Head of department:	prof. Ing. Igor Farkaš, PhD.
Assigned:	31.08.2014
Approved:	21.04.2015

prof. RNDr. Pavol Zlatoš, PhD.
Guarantor of Study Programme

.....
Student

.....
Supervisor

I declare that this thesis is the result of my own work and I have properly cited all used sources.

Signature: .....

Acknowledgements

I wish to convey sincere thanks to my supervisor RNDr. Martin Takáč, PhD. for constant guidance and support, for generating motivating ideas, but also for providing words of encouragement when needed. Furthermore, I thank friends and classmates (you know who you are) for bearing with me during the final days of my work on this thesis.

Abstrakt

Výskumníci umelej inteligencie majú dlho ambície vytvoriť model vykazujúci podobné vývojové črty ako ľudia, podobné motivácie a potreby. Pri komplikovaných systémoch sa s hľadaním odpovedí musíme uchýliť k počiatočnému obdobiu vývoja, kde sú pozorované subjekty alebo modely zatiaľ nepopísané a my máme lepšie možnosti ich analyzovať.

Raný senzomotorický vývoj ľudí je dnes dobre vysvetlený, napríklad konštruktivizmom. Existuje viacero prístupov, ako ho výpočtovo modelovať, od tradičných kognitivistických, až po emergentné systémy. Skúmame učiaci algoritmus inšpirovaný stelesnením, ktorého neoddeliteľnou súčasťou je ľuďom prirodzená zvedavosť slúžiaca ako nutkanie objavovať svet.

Implementovali sme učiaci sa model motivovaný zvedavosťou, ktorého správanie sme následne pozorovali vo viacerých virtuálnych prostrediach. Prinášame možné interpretácie tohoto správania v konkrétnych situáciach a ich prepojenie na teórie zvedavosti a senozomotorického vývoja.

Kľúčové slová: *autonómne systémy, intrinzická motivácia, developmentálna robotika, zvedavosť*

Abstract

AI researchers have long had the ambition to build a model possessing human-like traits of development, displaying the same drives and motivations. It is reasonable to search for answers in the early period of development, where both the observed subjects and the models are still tabula rasa, and thus easier to reason about.

Early human development is believed to be quite well described, for instance by the constructivists. There is a variety of approaches for computational modelling of sensorimotor development, ranging from traditional cognitivist to the data grounded emergent systems. We chose to explore a particular embodiment inspired model, one where curiosity that is so natural for humans is factored in and serves as a drive to actively explore the world.

We implemented a learning system motivated by curiosity and tested it on a suite of virtual worlds. We also provide possible interpretations of the model's behavior in concrete situations from the perspectives of curiosity and sensorimotor development theories.

Keywords: *autonomous systems, intrinsic motivation, developmental robotics, curiosity*

Contents

List of Figures

Introduction

Exploration is important for cognitive development. When exploring, we expose ourselves to novel situations, learning more about the world in the process. From a naive point of view, it might seem that when exploring, organisms expend valuable energy without focusing on a specific goal, or achieving some desired outcome. White [1] proposed, that the basic needs, such as food, sleep and reproduction, may not account for animal exploratory tendencies and that exploration itself may be a source of reward, motivating the animal to change its behaviour.

It seems quite intuitive that active exploration helps self-preservation. It is an investment; organisms actively study their environment to control the levels of surprise they expose themselves to. While surprise is useful in facilitating learning, too much of it usually comes hand in hand with dire situations. Thus, to prevent irreversible change, such as passing through an undesired phase-boundary (death), organisms actively explore [2].

Similar, curiosity-based exploration drive may guide steps taken by a human child, or any organism for that matter, prompting it to sample the environment with its modalities [3]. These first probes create a model of the world. The model is grounded by sensorimotor experience at first, but as the child grows, representations of the world become more abstract, allowing her to interact with the world on a higher level. Similar evolution of intelligence from sensorimotor interactions to abstract thinking is described in Piaget's Theory of Cognitive and Affective Development [4].

We explore Intelligent Adaptive Curiosity (IAC), a model designed by

Oudeyer and Kaplan [3]. IAC combines elements of embodiment hypothesis, sensorimotor development and intrinsic motivation in a mix producing child-like, curiosity based exploration, the benefits of which are discussed throughout thesis.

The chapters are organized as follows:

- First, a brief introduction to the neccessary background in cognitive modelling and theories of sensorimotor development and motivation in Chapter 1.
- In Chapter 2, we explain the IAC model.
- Its implementation details are briefly discussed in Chapter 3.
- We finish by interpreting its behaviour from both the perspectives of curiosity and sensorimotor development based on the experiments in virtual environments in Chapter 4.

Chapter 1

Autonomous Development

Building a believable AI is starting to seem like an insurmountable problem. One possible path to success is letting the intelligence "develop" itself. We can build a learning system inspired by animal (or human) development and look for similarities in behaviour to see if we are on the right track. Naturally, such model will also help us better understand and confirm existing developmental theories.

Before we present IAC, the curiosity motivated developmental model in Chapter 2, there are topics we want to prime. First, we briefly summarize approaches to modelling cognitive development. We then bridge into early human development for inspiration and take a short dive into motivation in agents, organic or artificial. That should provide us with enough background to better understand the design goals of IAC.

1.1 Modelling Development

Cognitive development is, in general, a very good candidate for being modelled. Even though there are critics of computational modelling in psychology, the benefits of clarity, explanation and prediction prove their reservations somewhat moot [5, p. 10].

There are many perspectives how to view a cognitive system. How does

it take in new information? On what policy does it choose strategies and actions? Does it anticipate the future? If so, does it use past experience to do so? What are its drives? All of these aspects shape the way we think about and design intelligent autonomous agents [6].

Both Cognitivist and Emergent approaches have unique merits and niche applications. They are often combined in hybrid cognitive systems to bring benefits of both to the table.

1.1.1 Cognitivist Models

Cognitivist approaches are about manipulating representations on the level of facts and symbols. They originate in the work of early cybernetists attempting to formally describe cognition by a logical apparatus. Their capabilities lie in representing statements about the world, inferring new information from them, and using it to predict and plan. Cognitivist models are strongly intertwined with design decisions of their architect, which is a double-edged sword. Having their representations in symbolic form makes it easier to inspect their internal state and their architecture usually makes them very powerful in the domains they have been created for. However, their ability to adapt to new situations is constrained by the cases accounted for by the designer and they thus have severe development limitations [6].

Good developmental examples would include the work of Drescher [7], attempting to model human infant sensorimotor development in a manner inspired heavily by the constructivist schema mechanism, of which we talk in Section 1.2. In the schema mechanism, the child model manipulates symbolic units of behaviour - the schemata. Schemata are activated in order to affect the child's surroundings. An important part of the schema mechanism is chaining, which allows for combining motor primitives to higher level actions, and so forth. Another similar approach is described by Cohen et. al. [8], where symbolic structures describing sensory input and motor actions are combined in executable contexts.

1.1.2 Emergent Approaches and Embodiment

In emergent models, representations are distributed (subsymbolic), and meaning "emerges" from many data points. Contrasting with Cognitivist systems, which tries to abstract away as much low-level information as possible, they are firmly defined by their interaction with the world. The environment and the form of embodiment affect what should be consedered meaningful behaviour. Some examples of Emergent approaches are the biologically inspired connectionist models able to recognize statistical patterns in data, or enactive systems not depending on any pre-determined rules, only drawing information from interactions [6].

Embodiment is a relevant part of Emergent cognitive systems. It emphasizes inspiration from living agents. When building and training an embodied cognitive system, we should take proper care not to oversimplify the learning system, its physical (or virtual) body, nor its environment. These are some interesting points about embodiment that Smith and Gasser [9] bring up when talking about inspirations coming from human child development:

Multimodality

Multimodal sensory systems, when enabled by architecture, allow for communication and coordination between "brain areas". Such communication has been shown not only to help learning, but also to be an important aspect of functioning of human brain. The connection between modalities is utilized for example in a situation, where we see an object and immediately know, how its texture would feel under our touch, how it would taste, etc. The modalities bootstrap and sometimes also educate one another [9]. Multimodality also bestows the benefit of sensory redundancy - the cognitive system may still be able to function without some of its senses working.

Incremental learning

In contrast to usual machine learning, we cannot present the whole training dataset at once. As will be discussed in Section 1.2, it is either biological maturing, or cognitive processing of experiences that allow the child to reach new information, or be able to make sense of it at all. Both animal experiments [10, 11] and neural network simulations [12] show that order of observations plays a great role not only in achieved learning progress, but also in the developmental trajectory the system takes.

Physical Bodies

Realistic physical bodies help development, as they provide a natural interface between the environment and the learning system [9]. They allow for tricks like offloading working memory to the environment [13], e.g. counting on fingers or writing intermediate results of a computation on paper. Also, having a complex body - two legs instead of four wheels, for instance - some laws of the world are natively understood by experiencing them first-hand.

Exploration

Any manner of learning without a teacher must somehow show a viable teacher substitute to drive the learning process. How else would the learner know what to learn, or what the possibilities and limitations might be? Exploration is a topic we touch upon in multiple sections throughout the thesis.

Social Interaction

Having the learner socially interact brings depth to its world. A mother caring for her child has tremendous effect on its development. By communicating, she guides and educates, leading the child on an optimal developmental trajectory. As time passes, the social interaction becomes more bidirectional - the child partakes more actively in communication.

1.2 Early Cognitive Development

When we talk about human cognitive development, or Constructivism, we mean the set of theories of knowledge and learning formalized by Piaget in his Theory of Cognitive and Affective Development [4, 7]. The theory's building block is the schema. Schemata represent knowledge about the world in a procedural fashion. In a newly born infant, actions produced by schemata would be equivalent to only reflex responses. Therefore, the child does not yet possess goal orientation or problem solving in the common meaning. However, there are processes at work that enable schemata to evolve and better describe the world, such as assimilation and accomodation.

- *Assimilation* is a way of absorbing new experience by reusing existing structures. A schema may utilize objects like sensory inputs, but also mental representations - other schemata - for its own functioning; to act accordingly when activated. New observations are fitted to and interpreted with existing schemata, serving to explain novel situations with already attained knowledge.

- *Accomodation* is the modification of existing schemata or construction of new ones in order to adjust to novel observations. It is applied, when assimilation does not satisfyingly explain the encountered phenomenon. For instance when some attribute of the new event has never been seen before and pre-existent schemata do not form a sufficient orthogonal basis to fit new data.

These developmental processes are triggered by interactions with the world, which happen as a part of *active exploration* [4, p. 30]. This fact is rather important in the context of the grounded, curiosity driven learning presented in Chapter 2, where a similar processes are used to organize the model of the world.

Using assimilation and accomodation, schemata can be composed or modified; become parametrized and more sophisticated. In time, they abstract

away from physical representations - later schemata need not concern themselves with the low-level sensorimotor events. In fact, while the cognitive development is continuous, Piaget's theory specifies four periods of development for explanatory purposes, only the first of which actually focuses intelligence on executing actions and affecting the world. The stages are:

- Sensorimotor Period
- Pre-operational Period
- Concrete Operational Period
- Formal Operational Period

The latter three periods no longer focus on the sensorimotor, but rather on exploring truth assertions. The assertions are about the observed world at first, and later deal with hypothetical and formal problems. However, even throughout the later periods schemata maintain their procedural aspect.

1.2.1 The Sensorimotor Period

Mental development of the child starts as soon as birth with the Sensorimotor Period. While it is not the thinking and acting we are used to in mature people, these first steps are nevertheless crucial for the develpment of more abstract cognitive skills, like language or formal reasoning [4, p. 35]. The development process is rather continuous, but there are several documented milestones. When the differences between them are studied, they show how higher functions could evolve.

The Sensorimotor Period describes the first two years of life, and has six substages. Each consecutive stage defines some novel class of problem that can be solved or goal that can be reached only by applying the new type of schema developed in the current stage. The simpler, already existing, types of schemata remain in use, and are actually still created when necessary - only most of the time, they need not be activated directly, but as a part of a higher order schema.

Motor Reflexes

From birth to approximately first month of life, the child has only the native reflexes to drive its action, for example grasping or sucking. The reflexes do not discriminate objects in any way, for instance the child would grasp any object that stimulates her palm, be it a mother's hand, or a toy [4, p. 37]. This is an example of nondiscriminate assimilation, when any sensory data is interpreted by the reflex schemata. Importantly, she has no idea objects exist when she does not currently interact with them. The perception of objects is dependent on activation of appropriate schemata [4, p. 40].

Towards the end of this stage, results of some accomodation processes can be seen, for example in the behaviour of searching for an object to suck, instead of just idly waiting to be presented with stimuli.

There are two important connections to note between the Motor Reflex stage and the model presented in Chapter 2:

1. It shows a stage development, where action selection is purely reflexive, and new data is merely assimilated.

2. The model too has no notion of permanence of perceived objects at first.

First Differentiations

From the first to fourth month after birth, the child's schemata evolve. As described by Wadsworth [4, p. 41], changes can be seen in the following behaviours:

- Perception is cross-modally coordinated. The child is now able to track objects visually, or to tilt her head in the direction of perceived sounds.

- Reflexes are partially accomodated, even habituated. Earlier, actions were only executed randomly, when their schema prerequisites have been met. Now observable behaviour to trigger the reflexes is present, e.g. thumb sucking.

Reproduction

In the phase from the fourth to the eight month of life, the child grows more aware of the connections between modalities and therofore can coordinate her senses better. Also, for the first time she starts to intentionally reproduce events, a phenomenon called *circulator reactions* or *reproductory assimilation* [4, p. 47]. Similar behaviour to reproduce events will also be seen in our learning model. Another thing to note in the Reproduction stage is that the child believes herself to be the sole cause of change in the world.

Coordination of Schemata

In the eighth to twelth month of life, the child displays clear signs of intentional, goal-oriented action. She is able to execute two seemingly unrelated actions where the result of the first is only the means to an end achieved by the second. An example might be reaching for a toy, or setting aside an object that is in the way to reach for it.

In later months of this stage, this rudimentary planning also allows for searching behaviours; the child can now search for an object that has disappeared [4, p. 50]. Interestingly enough, in the *A not B task*, she searches for displaced objects in the place they are seen to disappear the most, not where they disappeared this time.

In this phase, she is now also aware that other objects might be the causes of change in the environment.

Experimentation

Wadsworth [4, p. 55] describes that from twelth to sixteenth month the child starts to intentionally experiment and accomodate new schemata. Percieved consequences of arbitrary actions can become a base for a new schema which she is willing to activate and study. Generalization of schemata is also observed - some of the object's properties are understood before the object is examined, as she might have encountered a similar object before.

The child is now fully aware of object's existence even when not currently percieved. However, so *A not B task* is still not comprehended correctly - she searches for displaced objects on a statistical basis, that is: in the places most associated with the disappearance, not where the object disappeared this particular time [4, p. 57].

She is also now fully aware of her dependency on adults and that she is not the only cause of change.

Representation

In the last sensorimotor stage, leading up to the second year of the child's life, intelligence moves to a representational level, meaning she can draw conclusions and infer facts without experimentation [4, p. 59]. That serves as a bridge to later developmental periods.

Representational thinking allows her to understand properties of the environment, facilitating exploration of new, previously unobserved aspects of the world. Such exploration then serves as a gateway to a more sophisticated perception of the world, paving the road for further exploration in these novel domains. For instance, she is now able to successfully find displaced objects and account for changes in displacement scenarios, if they happen.

This is the last stage of development relevant for us at the moment, as more complex development definitely falls out of scope of our developmental model.

1.3 Motivation

Ryan and Deci [14] state that behaviour of organisms is driven by motivation. Unless we are compelled, we have literally no reason to act and move towards our goals. Our drives are categorized as either:

- *extrinsic*, where our actions always lead to clearly separable outcomes

- or *intrinsic* - actions that are by themselves enjoyable or otherwise interesting to us.

While the need for both is obvious, extrinsic motivation may seem less interesting than its intrinsic counterpart. However, Ryan and Deci [15] posit that there are types of extrinsic motivation very well capable of driving the organism towards complex behaviour. For instance, studying for exams to receive good grades is not a case of purely intrinsically motivated behaviour. It is complex and exploratory, but the grades themselves are source of reward, suggesting extrinsic motivation.

1.3.1 Extrinsic Motivation

Drawing from the definition, most activities done by organisms must be extrinsically motivated, as they lead to separate outcomes or rewards [14]. Eating for sustenance, hunting/gathering food, going to school or obeying authorities are all examples of extrinsically motivated behaviour. Some of them are the organism's biological needs that it must satiate, some are imposed on it as social constructs. In both cases there is a separable outcome, therefore the motivation is extrinsic.

When we relax the definition a little, we see that behaviour of most organisms cannot be clearly classified as either intrinsically or extrinsically motivated, but these two sources of motivation still play a role. Organisms must weight both extrinsic and intrinsic motivations when making a decision, for example taking an interesting but miserably paid job instead of a boring, well-paid one.

1.3.2 Intrinsic Motivation

Intrinsic motivation governs situations, where there is no obvious benefit, but the activity itself is pleasant [14]. A cat playing with a toy, or someone reading detective stories are both examples of intrinsically motivated behaviour.

The emphasis is on the activity being enjoyable, stimulating, or challenging instead of producing reward, or being done under pressure.

Pure intrinsic motivation can mostly be seen in organisms not yet conditioned by social interaction. A child playing in her crib does not give any thought to how her endeavours may look to an outside observer, and therefore merrily continues her exploratory adventures.

In our simulation, we focus on intrinsic motivation, especially its connection to curiosity and active exploration written about in the next chapter. There are, however, ways to combine both extrinsic and intrinsic motivations from multiple sources to achieve more realistic models.

Chapter 2

Intelligent Adaptive Curiosity

Exploration clearly plays a very important part in cognitive development of children, bringing about new experience to assimilate. If an artificial intelligence, for instance embodied in a robot body, could be motivated to explore, maybe it would also show signs of similar mental development. It would be fascinating to see a robot develop in an autonomous and open-ended fashion, as is characteristic for people.

Satisfying two conditions helps along the goal to reach open-ended development [3]. Development should be:

- *Incremental* – the tasks we want our robot to perform should start out simple, and gradually become more difficult. When learning to stand, children already know how to crawl.

- *Autonomous* – even though scaffolding provided by teachers is useful in learning, by no means is it feasible for it to always be present. Manually designing easier alternatives for all tasks just would not scale, as the potential number of activities is infinite.

2.1 Base Concepts

Intelligent Adaptive Curiosity (IAC) is a learning algorithm emphasizing a few key properties [3].

- It is motivation based, but unlike some (extrinsic) motivation systems, it solely maximizes an inherently intrinsic value called **the learning progress**.
- It is *curious* because it cares about the learning progress and therefore explores new situations.
- It is *adaptive* because interest in situations is evaluated dynamically. Once familiar, a situation is no longer interesting.
- It is *intelligent*, as it avoids both situations that are too predictable and too unknown, thus potentially dangerous.

Let us review the concepts of IAC from the perspectives of psychology and neuroscience to see how plausible the model is.

2.1.1 Psychology

Children are playful, and often engage in activities they find interesting, regardless of their utility in real world. Playing and exploring is clearly a rewarding experience on its own. This internal reward drives them to explore and learn about concepts that might be useful in later stages of their lives, even if such utility is not foreseeable at the moment [1].

Berlyne [16] proposes that exploration is intrinsically rewarded most in situations where surprise and novelty is present, serving as reinforcement for further exploratory activities. He also observed that exploration on a fine line between already known and completely novel situations tends to be most rewarding. This fact seems to support "The Flow Theory" by Csikszentmihalyi [17], which describes the optimal experience in mental activities (studying,

playing games). Situations that are either boring, or too challenging teach us very little, since they have either been experienced already, or there is little to none structure present to assimilate the novelty. In contrast, situations in the middle, or within "the Flow" sit in a sweet spot, and bring about most learning progress.

2.1.2 Neuroscience

Usually, when talking about mid-brain dopamine neurons, we mean predicting expected reward, which connects us to many reinforcement learning theories [18]. More recent studies [19, 20] show that dopamine releases correlate not only to reward prediction error, but to prediction error in general, suggesting that learning itself can be a pleasurable experience, compelling organisms to engage in learning activities.

Overall it seems that learning and satisfying curiosity can be a source of dopamine release in living organisms. Therefore, a reinforcement learning-like model based on this kind of reward is not completely implausible.

2.2 IAC Overview

This section provides the general overview of IAC, the intrinsically motivated learning algorithm by Oudeyer et. al. [3].

One thing to keep an eye out for is that IAC is designed in a rather modular way. Multiple data structures, criteria, or learning algorithms can be swapped for an alternative – within reasonable bounds. This modularity allows for starting small, allowing us to initially understand and explain simulation data, and incrementally upgrading the modules with more complex behaviour.

2.2.1 Summary

IAC is a simulation of a robot interacting with its environment and learning from each interaction. Since we are modelling sensorimotor development, all the interactions can be understood as low-level sensorimotor primitives, like touching, moving, feeling haptic feedback, seeing light, etc.

Over the course of learning, IAC experiences sensorimotor events, and remembers them. These remembered events are called **exemplars** in IAC terminology. Exemplars are stored in form of real-valued vectors, and are available to the algorithm at all times, although their storage is partitioned in a manner that not all parts of IAC have access to all the exemplars (explained later).

The internal structure of IAC consists of **regions** organized in a binary tree (binary search tree, or its k-dimensional variant [21] to be more specific). Each region governs its own exclusive subset of exemplars. Exclusivity is achieved by the k-d tree, each exemplar in a region must have satisfied the tree decision criteria in order to end up in that region in the first place.

There is a learning machine called **the expert** living in each region. The expert of each region is trained solely on the exemplars available to that region, so it literally is an expert in a limited field from the perspective of sensorimotor space.

Upon experiencing a sensorimotor interaction, IAC selects (by using the k-d tree) a region best suited to predict the future. Once selected, the region makes a prediction. When the outcome of an action becomes known, the selected region is given feedback on its performance and records the error of its prediction in a list associated with it. These **error lists** are used to compute the **learning progress** for the region.

The action selection mechanism generates a set of all possible actions for a situation and selects the action that would be executed by the expert of the region with highest current learning progress.

Throughout the experiment, as IAC gathers data, it employs a mechanism that incrementally splits regions, effectively creating new leaf nodes of the

k-d tree. The leaves inherit all exemplars passing the respective splitting criteria from the parent, and also the parent's error list.

Now let us explore the model in depth.

2.2.2 Sensorimotor Context and Memory

The robot has a sensory apparatus, each sensor produces real number values at any given time t. These inputs form a vector $\boldsymbol{S}(t)$. Similarly, at time t the robot acts by outputting real-valued actuator parameters, forming a vector $\boldsymbol{M}(t)$. The robot's interface with the world, its form of embodiment is fully described by the Sensorimotor Context $\boldsymbol{SM}(t)$, which is merely a concatenation of vectors $\boldsymbol{S}(t)$ and $\boldsymbol{M}(t)$.

After acting with $\boldsymbol{M}(t)$, the robot perceives the outcome – $\boldsymbol{S}(t+1)$. The original sensorimotor context $\boldsymbol{SM}(t)$, together with new sensory data $\boldsymbol{S}(t+1)$ form an exemplar to store – $(\boldsymbol{SM}(t), \boldsymbol{S}(t+1))$, the inputs and targets for the expert prediction machine. The expert tries to predict the state of the world as will be perceived in the next unit of time.

2.2.3 Regions and Experts

There is an expert $\boldsymbol{E_n}$ assigned to each region $\mathcal{R}_n$. These experts are trained on the $(\boldsymbol{SM}(t), \boldsymbol{S}(t+1))$ exemplars, and predict $\boldsymbol{S}(t+1)$ for input $\boldsymbol{SM}(t)$. The expert is a learning machine capable of regression. IAC does not care overmuch, what kind of machine this actually is, although there are a few properties that it should satisfy.

- It should support incremental training. As the exemplars are also obtained incrementally, it could quickly become unfeasible (even if possible) to batch retrain the machine every time a new exemplar is recorded.
- Since $\boldsymbol{S}(t+1)$ is a vector, the prediction is a case of multivariate regression, which may impose some limit the learning machine used to implement the expert.

Neural networks, Bayesian machines, or support vector machines (SVMs) are all good candidates for implementing the expert.

2.2.4 Region Splitting

Throughout its functioning, IAC must organize its regions to better reflect the observed world. It starts with one region, $\mathcal{R}_1$, which is incrementally split as deemed necessary, forming a tree (in our case, the splits are two-way, so the tree is binary). The original region becomes inactive (no longer eligible for use), and its children inherit mutually exclusive parts of its hoarded dataset. Two criteria are employed to steer the splitting process:

- C_1 defines the condition to be satisfied in order to initiate splitting of a region.
- C_2 tells us, how the region is going to be split.

After a region split occurs, C_1 is evaluated recursively for the resulting regions, and if eligible, they are again split with C_2.

In our experiments, we decided to choose the same criteria as Oudeyer [3]. C_1 criterion dictates that a region is eligible for splitting after the size of its dataset exceeds a defined threshold T, e.g. $T = 250$. Choosing a sufficiently small T also gives IAC good computational efficiency, as it prevents the size of the exemplar set from becoming too big, which could adversely affect either the computational complexity of the training, or the prediction, depending on the kind of learning machine used as expert.

C_2 splits the dataset in such a way that after the split, it minimizes for each created dataset the sum of variances in $\boldsymbol{S}(t+1)$ weighted by its cardinality. More formally, let us have

$$\Gamma_n = \{(\boldsymbol{SM}(t), \boldsymbol{S}(t+1))_i\}$$

as the dataset owned by region $\mathcal{R}_n$, j as a cutting dimension, and v_j as a cutting value. Splitting of a dataset Γ_n into Γ_{n+1} and Γ_{n+2} has to satisfy the following:

- Exemplars $(\boldsymbol{SM}(t), \boldsymbol{S}(t+1))_i$ from Γ_{n+1} have to have their jth value of their $\boldsymbol{SM}(t)$ smaller than v_j.
- Exemplars $(\boldsymbol{SM}(t), \boldsymbol{S}(t+1))_i$ from Γ_{n+2} have to have their jth value of their $\boldsymbol{SM}(t)$ greater than v_j.
- The following value is minimal:

$$\begin{aligned} &|\Gamma_{n+1}| \cdot \sigma(\{\boldsymbol{S}(t+1) | (\boldsymbol{SM}(t), \boldsymbol{S}(t+1)) \in \Gamma_{n+1}\}) \\ +&|\Gamma_{n+2}| \cdot \sigma(\{\boldsymbol{S}(t+1) | (\boldsymbol{SM}(t), \boldsymbol{S}(t+1)) \in \Gamma_{n+2}\}) \end{aligned}$$

  where

$$\sigma(\mathcal{S}) = \frac{\sum_{v \in \mathcal{S}} ||v - \frac{\sum_{v \in \mathcal{S}} v}{|\mathcal{S}|}||^2}{|\mathcal{S}|}$$

  and $\mathcal{S}$ is a set of vectors and $|\mathcal{S}|$ is its cardinality.

The rule for evaluating variance in a dataset naturally has no simple analytic solution, so a random sample of potential splits has to be generated and evaluated by the rule to determine the quasi-optimal split [22]. The sampling density can be adjusted to either favor computational complexity or precision.

Since the regions are organized in a k-d tree and each region stores the cutting dimension and value, it is always unambiguously determinable, which expert is responsible for handling a concrete sensorimotor experience. One simply has to follow the $\boldsymbol{SM}(t)$ vector down the tree until the corresponding region is encountered.

2.2.5 Error and Learning Progress

Upon acting, a new state of the world becomes available to the robot. This is a time, where the expert responsible for the action must reflect, how precise its prediction was. Given a prediction $\widehat{\boldsymbol{S}}(t+1)$ and a measurement $\boldsymbol{S}(t+1)$ we can compute the prediction error for our expert $\boldsymbol{E_n}$ at time $t+1$.

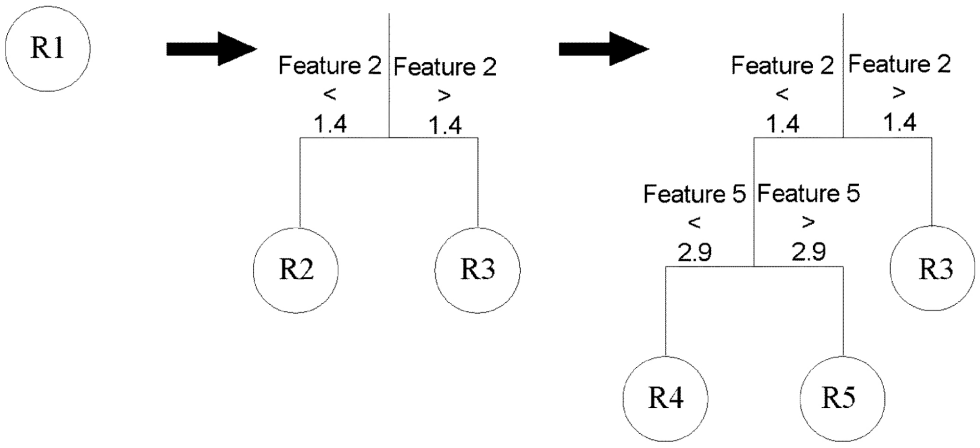


Figure 2.1: The region splitting process. R1 splits into R2 and R3 based on Feature 2 of value 1.4. R3 splits into R4 and R5 based on Feature 5 of value 2.9. Upon splitting, the child regions inherit complete error lists and mutually exclusive parts of the exemplar set from their parent. Only leafs of the tree remain active in predicting, inner nodes are no longer used. Source: Oudeyer et. al. [3].

$$e_n(t+1) = ||\boldsymbol{S}(t+1) - \widehat{\boldsymbol{S}}(t+1)||^2$$

This error is appended in the **error list** for the region $\mathcal{R}_n$:

$$e_n(t), e_n(t-1), ..., e_n(0)$$

Since an error is appended to this list only when the associated expert makes the prediction, the time t here is specific to the region, not global time as perceived by the robot.

The error lists are used to evaluate learning progress after each action $\boldsymbol{M}(t)$. The learning progress itself is computed as a smoothed derivative of errors made by $\boldsymbol{E_n}$ up to the point it observed its last exemplar. The mathematical rules for its computation are

$$< e_n(t+1) > = \frac{\sum_{i=0}^{\theta}(t+1-i)}{\theta+1}$$
$$< e_n(t+1-\tau) > = \frac{\sum_{i=0}^{\theta}(t+1-\tau-i)}{\theta+1}$$

where τ is a time window, and θ is a smoothing parameter.

The learning progress in time $t+1$ can then be defined as the difference between smoothed error before our time window and now.

$$L(t+1) =< e_n(t+1-\tau) > - < e_n(t+1) >$$

2.2.6 Action Selection

IAC is said to be motivated by curiosity and rewarded by its learning progress. It should be ok for us to define reward drawn from exploration, whether positive or negative, as the learning progress.

$$r(t) = L(t)$$

Note that this exploration reward can be simply integrated with other possible sources of reward, e.g. by a weighted sum

$$r(t) = \sum_i a_i \cdot r_i(t)$$

Now, given a way to operationalize reward, IAC could employ any number of reinforcement learning techniques to maximize reward expected in future,

$$E\left\{ \sum_{t \geq t_n} \gamma^{t-t_n} r(t) \right\}$$

where γ is the discount factor and is $(0 \leq \gamma \leq 1)$, which serves as a balance that assigns less reward for actions performed in the future, than in present time.

Although any reinforcement learning could be plugged into the action selection process, for instance Q-learning by Watkins [23], at the moment we decided to use a rather greedy action selection rule to better document the behaviour of IAC itself – one that maximizes the immediate reward. Being greedy, we do not have to concern ourselves (yet) with what effect could delayed gratification have on our learning system. For now, the expected reward at time $t+1$ is roughly equivalent to the actual documented learning progress in the region that is chosen to predict and execute the action

$$E\{r(t+1)\} \approx L(t - \theta_{R_n})$$

where $t - \theta_{R_n}$ is the time of last activation of the region – the learning progress is the last learning progress documented in that region.

Before activating a region, we need to be able to select it. That can be done by searching the k-d tree of regions with sensorimotor context $\boldsymbol{SM}(t)$ parameter. We can construct a sensorimotor context and select our action in the following way:

1. $\boldsymbol{S}(t)$ is the actual sensory input.

2. We generate all possible action candidates $\widetilde{\boldsymbol{M}}(t)$ that we can execute in the current situation. If our environment is continuous, we instead generate a sufficiently dense random sample of actions to choose from.

3. Combined together, they form the candidate $\widetilde{\boldsymbol{SM}}(t)$, which is used to locate a corresponding region and look at its learning progress. We apply the candidate $\widetilde{\boldsymbol{M}}(t)$ that brings us most learning progress.

4. Sometimes we want to add random behaviour. We can do that by introducing exploration factor $\epsilon \quad (0 \leq \epsilon \leq 1)$, and only act greedily in the cases not governed by it.

This concludes the design of IAC. We next delve into some of our technical decisions of our implementation.

Chapter 3

Implementation

We would like to present our implementation of IAC, some of the design choices, and low level decisions. Let us begin by discussing the tool set.

3.1 Technical Choices

Our version of IAC was programmed in the Python programming language [24], mostly for practical reasons. The language is high-level and expressive, which allows to quickly and relatively painlessly formulate complex ideas in a few lines of code. It is also readable – Python code looks very much like pseudo-code and therefore should be more understandable even to people who don't program in Python themselves.

Perhaps it is the combination of these two properties that first attracted the academic community to Python. Indeed, it is famous for its many scientific libraries. There are many tools for numerical computing and linear algebra [25], image processing, and machine learning [26].

Since this work essentially falls into category of developmental robotics, we want to be able to interface with either real robots, or at least a detailed robot simulation environment in the future. Python can help with both; if nothing else, we can use C bindings. One specific environment we had in mind to use is the Webots Robot Simulator [27], which provides a dedicated

Python API.

3.1.1 SciPy Stack

Python is competitive in the scientific computation community. Although unlike MATLAB or GNU Octave, it does not serve everything to the user and some effort must be made to install and configure the tools. The SciPy stack [25] is a capable, free substitute for a MATLAB-like environment. It entails

- NumPy – the numerical library which we will discuss shortly
- SciPy – the library building on NumPy, adding
  - efficient sparse matrices
  - optimization techniques
  - Fourier transformations
  - signal processing methods
- matplotlib – a plotting library
- IPython – an interactive Python console
- pandas – a time series analysis toolkit

and others, overall having rich feature set similar to that of MATLAB.

NumPy

NumPy revolves around an efficient data-structure called ndarray. It is a multi-dimensional array supporting flexible indexing [28]. NumPy is capable of broadcasting functions and operations over ndarrays in a configurable manner, making large scale data manipulation efficient and easy to implement. Apart from that, it provides strong random number generation capabilities and support for linear algebra operations, since the ndarray objects usually represent vectors and matrices.

3.1.2 Scikit Learn

Scikit-learn [26] is a machine learning, data mining and analysis library for Python. It can be used for classification, regression, clustering, dimensionality reduction, and many more tasks associated with machine learning. Its offer of high quality implementations of many commonly used algorithms makes it a very attractive choice. Sadly, at the time of working on this thesis it did not yet have a (stable) implementation of a Multilayer Perceptron, so as an exercise in futility we implemented our own.

The models from Scikit-learn can be interfaced with by using NumPy ndarrays, a de facto standard data format for computing in Python.

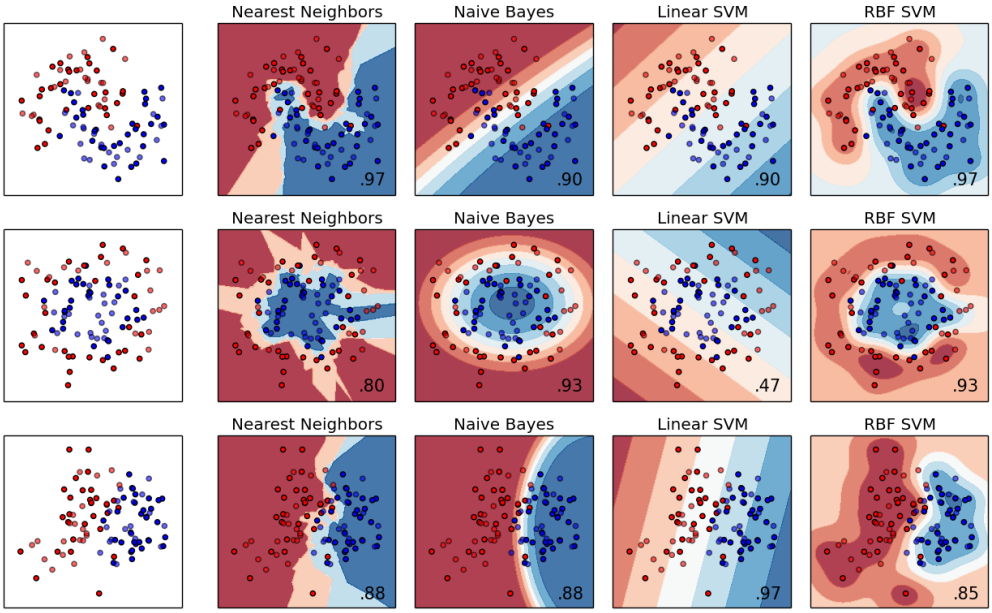


Figure 3.1: Classifier comparison on various datasets. From left to right: Nearest Neighbours, Naive Bayes, Linear SVM, RBF kernel SVM. Source: Scikit-learn [26]

We used the Nearest Neighbours based regression from Scikit, reasons for which we discuss in its own subsection.

3.2 Learning Algorithms

In our IAC implementation we have so far only used the k-NN learning algorithm as an expert, although we initially also implemented a Multilayer Perceptron and thought about using the nonlinear version of SVM from Scikit-learn. The k-NN's simplicity and the fact that it can also learn to classify linearly non-separable classes have advocated its use. Because it produced good results, we did not feel compelled to upgrade yet.

3.2.1 K Nearest Neighbours

K Nearest Neighbours is a very simple algorithm that can be used for either classification or regression. In n-dimensional feature space it simply records the data points along with their targets. k is an integer parameter, often a small number, or even $k = 1$. It denotes, how many nearest (any distance metric can be used, e.g. euclidean) entries to the new data point have a "vote" in the classification/regression result.

- If used as a classifier, the k nearest data points "vote" with their respective classes. The output is the class with most votes.
- When used in regression, the mean of k nearest data points target values is used as the result.

Compared to more complex learning models, k-NN shows several valuable properties:

- The implementation is very simple, which has great debugging value – one less component of a complex system to worry about.
- The training is virtually free. In Nearest Neighbors, the dataset *is* the model, so retraining upon IAC region splitting is computationally inexpensive. This, however, also means that computing predictions can be costly when the dataset is big (comparing many data points against

each other). Scikit implementation does some optimizations to mitigate this cost, such as the k-d tree or Ball tree storage indexes. They should not be very necessary in our case though, as our C_1 splitting criterion guaranties a low number of training exemplars, and the IAC region tree basically also is a k-d tree.

- Given a sufficiently representative sample as a training set, it gives reasonably good predictions.

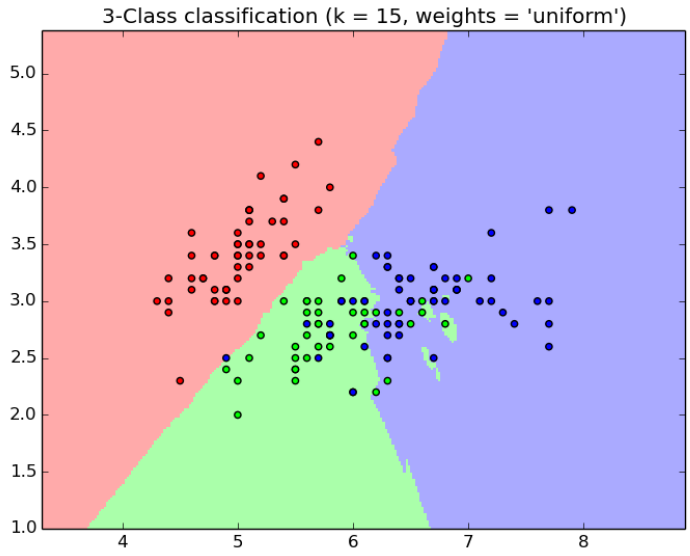


Figure 3.2: K Nearest Neighbours classification example. A 3 class classification, showing the space "owned" by the classes (how would a new sample be classified). Source: Scikit-learn [26]

We used the Scikit-learn's k-NN in our IAC implementation.

3.2.2 Multilayer Perceptron

Our initial plan was to use the Multilayer Perceptron as expert in all regions. Even though we did not plug in our MLP into IAC in the end, we feel our quality time spent developing (debugging) it warrants its brief inclusion to the thesis.

Multilayer Perceptron is a biologically inspired machine learning algorithm that can perform classification and regression tasks. It belongs to a class of algorithms called neural networks, which in general try to present an at least partially biologically plausible alternative to computation. Naturally, there are exceptions and criticisms [5, 29].

A Perceptron consists of base building units, the artificial neurons – a computational model of biological neurons. The concepts are roughly mappable as follows:

biological neuron	artificial neuron
dendrites	input connections
axon	output connections
synaptic connections	weights

where inputs and weights are in general real numbers, and the output can be either binary or real-valued, depending on the type of the perceptron.

The artificial neuron computes a weighted sum of the input values, producing a *net* value to which it applies the activation function. A process that is very similar to the biological neuron receiving electrical impulses over time and firing when the accumulated voltage reaches a certain threshold.

Perceptron is a single layer neural network, potentially consisting of multiple neurons. It usually fits the data in a supervised manner, so both input and target data have to be presented during the training session. It can not by itself classify linearly non-separable data. To be able to do so, hidden neural layers with nonlinear activation functions are necessary [5, p. 28].

After some time, a Multilayer Perceptron model emerged, featuring multiple layers of nonlinear transformation that allowed it to classify linearly

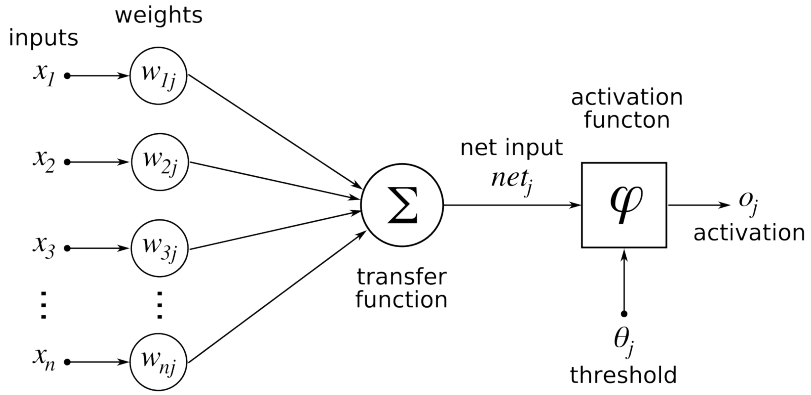


Figure 3.3: An artificial neuron. Activation is computed by applying the activation function on the weighted sum (net) of the input vector **X**. The activation function has a threshold parameter – θ. Source: Wikibooks [30]

non-separable data. The greatest challenge in designing the MLP was developing a way to train it. Single layer networks can be trained by presenting a target value serving as a source of error that is in turn used to adjust the network's weights. However, before the discovery of back-propagation, there was no obvious rule defining how to adjust weights of hidden units.

There are 2 stages in back-propagation training. First, the activations are propagated forward in the network, producing a result on the output neurons. This result is compared with the target value to determine an error on each output unit. In the second stage, these errors are propagated backwards throughout the network, computing weight adjustments [5, p. 42]. A thorough mathematical explanation of the back-propagation algorithm would take up significant space, therefore we encourage the reader to study one of the many excellent descriptions or Internet tutorials available, for example by Shultz [5, p. 37].

The back-propagation algorithm was criticized for its biological implausibility – no backward propagation of "error" was observed in biological networks. However, backward propagation of activation is very well possible. There were several attempts to MLP learning by back-propagating activa-

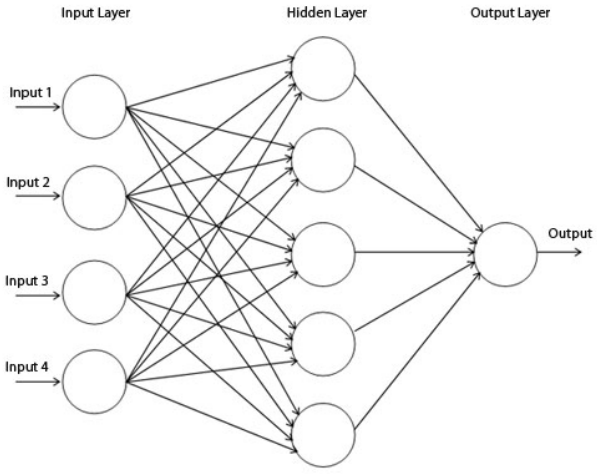


Figure 3.4: Multilayer Perceptron. This example has one hidden layer providing additional nonlinear transformation. Source: Multilayer Perceptron for Activity Recognition [31]

tions, e.g. O'Reilly's GeneRec algorithm [5, p. 48][32]. Another reason of back-propagation's biological implausibility is that it in no way accounts for synaptogenesis or neurogenesis [33]. It is reasonable to presume that sudden leaps in development could also be explained by much more large-scale structural changes in the brain [5, p. 48].

Even given these criticisms, back-propagation remains very useful and is widely used.

Our implementation of a back-propagation trained Multilayer Perceptron lives on GitHub in the IAC repository, together with the IAC implementation [34].

3.3 Implementation of IAC

Before delving deeper into design, let us mention that the whole implementation is freely and publicly accessible [34]. Since anyone can view the code

there, we will avoid talking about it in much detail here – it is pretty much the operationalized version of IAC describe in Chapter 2. We will talk more about the "why" of things instead. The code presented in this section is for the most part pseudo-code or simplified version of the actual code to make talking points clearer. However, we are completely willing to talk about implementation details either in person or online, e.g. on GitHub.

The IAC simulation is split in two distinct concepts: the agent and the environment. Their boundary is set, perhaps a little controversially, in the following way: the agent concept handles only the IAC algorithm (the robot's mind), leaving everything else including the robot's body in care of the environment. Both are independently parameterizable. In the code examples that follow, assume the following:

- $\boldsymbol{S}(t)$ is denoted as `s`
- $\boldsymbol{M}(t)$ is denoted as `m`
- $\widehat{\boldsymbol{S}}(t+1)$ is denoted as `s_predicted`
- $\boldsymbol{S}(t+1)$ is denoted as `s_observed`

3.3.1 The Agent

In summary, the agent is responsible for handling the state of the learning algorithm. It governs the IAC region tree, experts, datasets and error lists. From the consumer perspective, there are only two ways to interact with it – acting and training:

- `agent.act(s) -> m, s_predicted`
- `agent.observe(s, m, s_predicted, s_observed) -> void`

Note that when training (`observe` operation) we also set the `s_predicted` vector. IAC is required to keep track of last predictions $\widehat{\boldsymbol{S}}(t+1)$ to be able to

compute the error later. To achieve that we can either delegate the responsibility of providing the correct vector to the consumer (the chosen approach), or have IAC remember the value internally, making it more stateful and requiring the consumer to always call the `observe` operation after acting to ensure correct behaviour. The choice was made in favour of the more explicit option, hopefully making the API clearer. In the chosen case, the IAC model is not actually modified unless `observe` is called.

Region Storage

As mentioned before, the IAC region tree is actually a k-d tree [21]. In the tree, we search for a region corresponding to a particular sensorimotor context. Search trees such as this one usually satisfy some form of the dictionary (or map) API

- `dict.put(key, value) -> void`
- `dict.get(key) -> value`

in our case, the values being IAC regions and keys the sensorimotor context vectors $\boldsymbol{SM}(t)$. We simply need to "follow the key down the tree", as one would in a binary search tree with the exception that we are always looking for a leaf node.

```
def find_region(sm):
    region = root_region
    while not region.is_leaf:
        if sm[region.split_feature] > region.split_value:
            region = region.right
        else:
            region = region.left
    return region
```

Figure 3.5: Region lookup

Region splitting check is triggered after the region updates its model, and is applied recursively. Note that with the simple C_1 splitting criterion we used, recursive split never actually occurs, as the newly created regions always have fewer examples.

```
def try_split(region):
    if not region.should_split():
        return
    left, right = region.split()
    try_split(left)
    try_split(right)

def observe(s, m, s_predicted, s_observed)
    sm = numpy.concatenate((s, m))
    region = find_region(sm)
    region.update(s, m, s_predicted, s_observed)
    try_split(region)
```

Figure 3.6: Region splitting

3.3.2 The Environment

The environment defines the form of embodiment and physical properties of the world. Also, while running the simulation, it holds the physical state of the world (as opposed to the robot's mental state, which is held by the agent).

Its world definition API is rather simple – it helps initialize the agent by providing the shape of sensor and motor arrays, and the callback generating possible actions for the robot in the current situation.

- `env.s_len : int`
- `env.m_len : int`

- `env.all_actions : callable -> array of m`

Apart from defining the world, there are ways for the robot to retrieve state from the environment and to affect it.

- `env.sense() -> s`
- `env.act(m) -> void`

3.3.3 Simulation Loop

The agent and environment concepts are more or less independent and their instances can be parametrized separately. They interact via the simulation loop, which is agnostic to any specifics of either.

```
def loop(n_iters, agent, env):
    s = env.sense()
    for _ in range(n_iters):
        m, s_predicted = agent.act(s)
        env.act(m)
        s_observed = env.sense()
        agent.observe(s, m, s_predicted, s_observed)
        s = s_observed
```

Figure 3.7: Simulation loop

Before the agent and environment can be injected into the loop, they are initialized.

Our programmed environment could be sized as necessary, limit the agents movement capabilities, define toy behaviour when the agent interacted with it, disable some of the agent's senses, etc. The concrete options will be elaborated on further in the experiment design chapter.

```
env = ToyEnvironment(
    size=2,
    max_step_size=0.3,
    boundary_strategy='walled',
    toy_behaviour='static',
    agent_knows_position=True,
    distance_measurement='linear'
)
```

Figure 3.8: Environment initialization

On the other side we have the agent, representing the robot's mind. When created, it is given access to the shapes of the sensory and motor vectors, so it can initialize properly, and the callback for generating all possible actions (remember, both properties of the world, and the robot body are modelled in the environment, therefore the environment is responsible for generating the actions). While the IAC model has quite the number of hyper-parameters, currently only two are exposed.

```
agent = IAC(
    env.s_len,
    env.m_len,
    env.all_actions,
    exploration_factor=0.1,
    dataset_size_threshold=250
)
```

Figure 3.9: Agent initialization

Chapter 4

Results

Finally, let us review how IAC fared in the experiments. We provide some reasoning behind the employed evaluation methods, present the virtual environments, and assess the model's behaviour.

4.1 Evaluation Methodology

In [3], Oudeyer discusses evaluating development of autonomous agents. We should be careful not to introduce task or goal oriented biases to evaluation of intrinsically motivated model – our goal is to assess the system's complexity in time, not task performance.

There are a few methods for assessing the behavioral complexity, we used two.

- Inspect the robot's internal state and quantitatively review evolution of variables in time.
- Observe the robot's behaviour from an external point of view and explain patterns qualitatively. This is essentially the approach taken by Piaget when documenting and explaining children development [4].

4.2 Environments

We programmed two environments for the robot to develop in. One very simple, to test if the expert and region machinery is working correctly. The other contained a toy the robot could interact with. By programming how the toy would react to the robot, we could influence the robot's developmental trajectory. All environments were continuous.

4.2.1 Static Environment

This simple scenario had the robot explore a finite, walled room. Apart from the robot, the room contained an inanimate toy. The robot was equipped with location providing sensors, but it had no idea about the room's dimensions, or that the room is walled. It also had an "infrared sensor" enabling it to measure its distance to the toy. More formally

$$\boldsymbol{S}(t) = (x, y, d)$$
$$\boldsymbol{M}(t) = (\delta x, \delta y)$$

where $(\delta x, \delta y)$ vector denotes the robot's movement.

Assessment

In this setting the robot explored the room, observing. Once enough observations were made, the robot split its internal representation into two regions, and so on. Apart from the obvious region splitting by (x, y) coordinates, we can also see two interesting types of splits in Figure 4.1.

- Splits made on the "distance to toy" feature, creating concentric regions around the the toy in the sensorimotor space.
- Splits decided on the motor part of the sensorimotor context, forcing two regions to share the same sensory space.

We can also see that the k-NN expert correctly learns to predict the distance to toy based on $\boldsymbol{SM}(t)$.

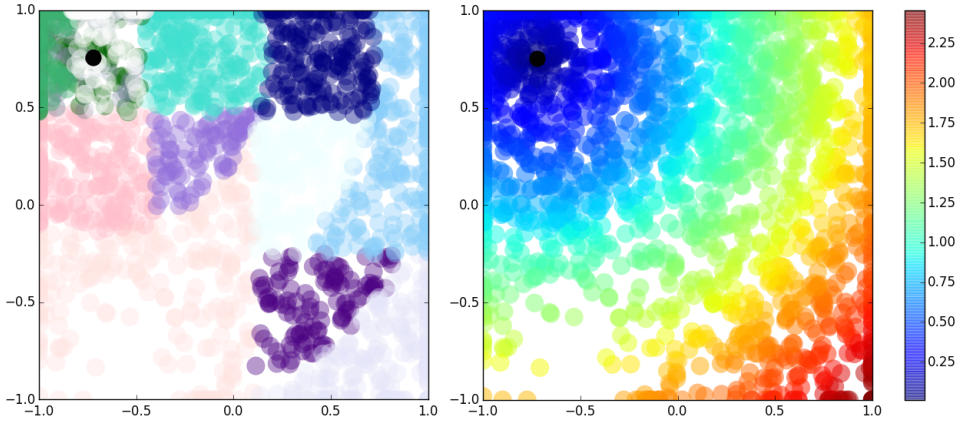


Figure 4.1: Example of sensorimotor space split into regions (left) and predicting toy distance (right). The black dot denotes the toy's coordinates. In both figures, the axis correspond to (x, y) coordinates of the robot in the room. They are taken from the $\boldsymbol{SM}(t)$. In the region split figure (left), colours represent various regions governing the areas. In the toy distance prediction figure (right), the colour represents the predicted distance taken from $\widehat{\boldsymbol{S}}(t+1)$.

The prediction error of various regions slowly decreases over time, although it may increase temporarily as the robot encounters novel situations or upon region splitting, when the two child regions are suddenly poorer in the number of exemplars they have at their disposal.

Due to its simplicity, we used this environment to test and debug our IAC implementation. Afterwards we moved on to the next one, where the robot learned to play with a toy.

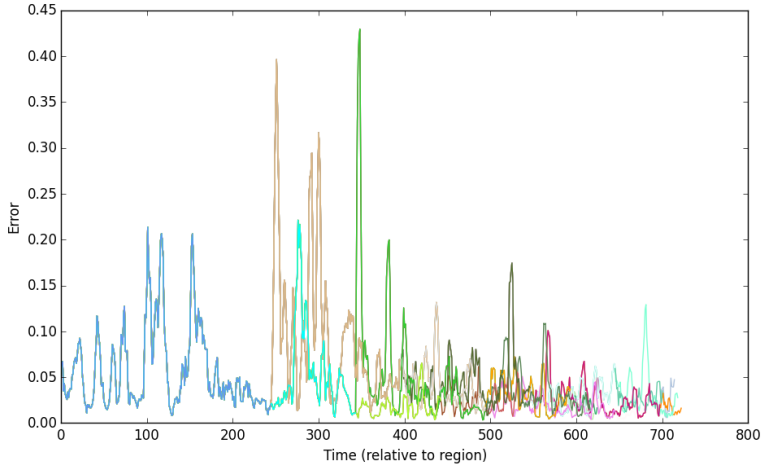


Figure 4.2: Evolution of smoothed error in various regions over time. Error for each region has its own colour. Time in this figure is relative to the region.

4.2.2 Living Toy Environment

Moving on from the static world, this environment features a few changes, the first one being a live toy placed inside. The robot can try to communicate with the toy by whistling, and the toy responds depending on the whistled tune. To see results of the interaction more clearly we disabled the robot's location provider, only allowing it to sense the distance to the toy via the "infrared sensor". To summarize

$$\boldsymbol{S}(t) = (d)$$
$$\boldsymbol{M}(t) = (\delta x, \delta y, f)$$

where f is the frequency of the whistled tune. The toy responds to the frequency by either moving randomly, staying in one place, or jumping towards the robot (making the distance 0).

- The random toy movement reaction represents a task that cannot be

learned. Thus, IAC should in turn avoid whistling tunes that lead up to this toy behaviour.

- The toy staying in one place is a situation that can be learned, although the learning might be complicated due to the fact that whistling the "wrong tune" effectively resets the learning.
- The toy jumping towards the robot is the easiest situation to learn by far, and the robot quickly learns how predict the distance. It gets bored with it after a while though, as there is nothing new to learn.

Qualitative Assessment

To evaluate progress in this scenario, we take a look at actions taken by the robot during the simulation. In the beginning, we should see a purely reflexive behaviour similar to that observed in newly born infants – the robot will try all actions indiscriminately, merely assimilating new data. After a developmental milestone occurs (the first region split, in our case), the robot intentionally verifies outcomes of its actions.

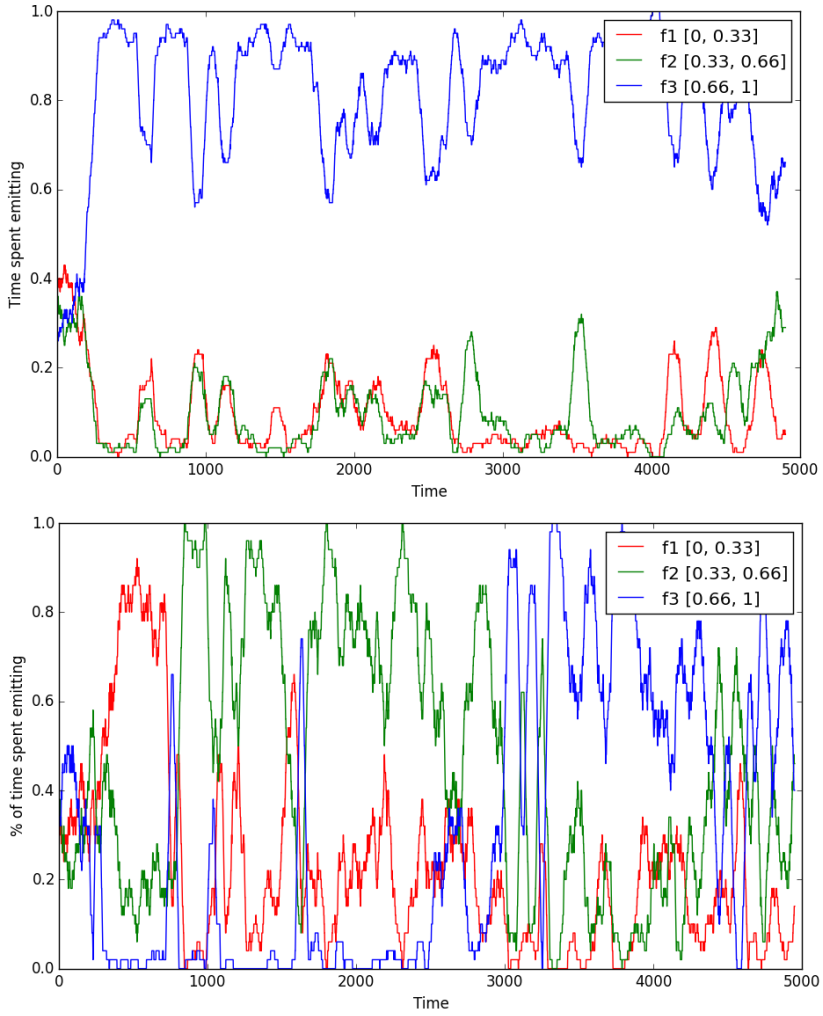


Figure 4.3: Action selection over time in the dynamic environment. There are three plots in both figures, representing the percentage of time spent whistling the tune (data is obtained by smoothing the selected frequency range in a small moving window). Before the region space differentiates (first 250 iterations), all 3 actions are selected roughly with the same probability. Once the first region is split, a more intentional behaviour emerges. In the first figure, the algorithm chooses (and sticks with) the easy path. The development in the second figure is much more interesting. After the first region split, the robot tries to learn the impossible task, and only after realizing its futility, it switches to the learn-able tasks.

Let us look further into two cases from Figure 4.3. In one, the robot decides to pursue the easiest action from the from the very beginning and sticks with it. This probably happened, because the random motor babbling done in the beginning of the experiment thoroughly hid the fact that the toy-standing-still reaction is also learn-able. The actions triggering it were inherited by a region with low learning progress.

In the other case, the robot gets bored with one learn-able reaction, switches to the other for a while, then starts switching back and forth. This ties in neatly with what Kaplan says: "An agent motivated by maximizing learning progress constructs its behavior in order to go from unpredictable situations to predictable ones. Instead of focusing on situations that it predicts well (minimizing prediction error) or on situations it does not predict at all (maximizing prediction error) it focuses on the frontier that separates mastered know-how from unmastered know-how" [35].

4.2.3 3D Simulations and Real Robots

In theory, we should be able to take IAC further than simple virtual simulations. Oudeyer et. al. [3] performed experiments on a physical robot representing a child in a crib. This robot could direct its gaze and execute preprogrammed, high-level motor actions (biting, bashing) on toys. The physical environment in these experiments was in fact stateless, as the toys were tied up with strings and the robot always waited for them to return to the original state before acting again.

The motor primitives in the crib experiment were high-level for illustrative reasons – the author wanted to show how the robot discovered sensorimotor affordances (relationships between actions, objects, and effects). It would be very interesting to see a version of IAC driving low-level sensorimotor interactions, for instance the self-modelling starfish presented by Bongard et. al. [36].

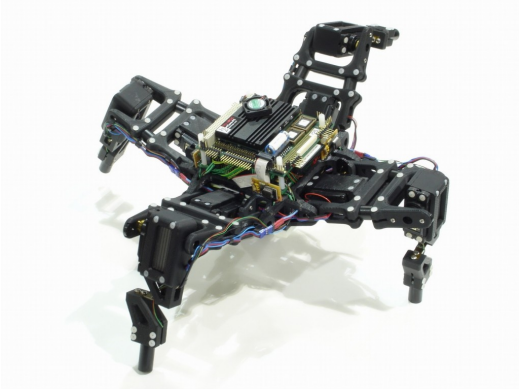


Figure 4.4: The Self-modelling Starfish robot. Source: Bongard et. al. [36]

Discussion

Cognitive Science features an interesting self-breeding cycle, where we use technology to better understand cognition, and in turn apply that new understanding to create more exciting technology. Even as this cyclic process gains ever more traction, we feel that it will never actually conclude. Rather, our understanding can only improve asymptotically as time progresses. IAC research fits right into this cycle. The fact that we can now operationalize concepts such as curiosity and motivation unlocks more research topics.

Our experiments confirmed some of the results of Oudeyer et. al. [3], showing a curiosity driven model behaving in a human-like manner from a certain perspective. Properties of constant objects quickly became boring, once learned. On the other hand, being irritated by chaos, the robot only pursued endeavours it thought were learn-able (even if difficult). This behaviour fits well into various education and entertainment theories [17], and could be considered human.

Lets not get ahead of ourselves, though. We should reflect on what we learned and ask ourselves, whether we are not just hiding number-crunching behind fancy words. It is important to note that we made the mistake of oversimplifying, so it is difficult to tell without trying something more realistic. It would also be interesting to see, how would a system with IAC's inner organizational ability scale beyond the sensorimotor world.

The major (trivially fixable) regret at this point is about not following the work further. Particularly exciting were the notions of embodying IAC into either more realistic virtual simulations or outright real robots, such as the

limping starfish [36]. The merits of intrinsically motivated exploration could have been better tested there, as the embodiment lessons remind us [9]. Also, plugging in more complex models as experts (e.g. recurrent architectures) or implementing real reinforcement learning could produce interesting results.

Bibliography

[1] Robert W White. Motivation reconsidered: the concept of competence. *Psychological review*, 66(5):297, 1959.

[2] Karl J Friston and Klaas E Stephan. Free-energy and the brain. *Synthese*, 159(3):417–458, 2007.

[3] P-Y Oudeyer, Frédéric Kaplan, and Verena Vanessa Hafner. Intrinsic motivation systems for autonomous mental development. *Evolutionary Computation, IEEE Transactions on*, 11(2):265–286, 2007.

[4] Barry J Wadsworth. *Piaget's theory of cognitive and affective development: Foundations of constructivism* . Longman Publishing, 1996.

[5] Thomas R Shultz. *Computational developmental psychology*. MIT Press, 2003.

[6] David Vernon, Giorgio Metta, and Giulio Sandini. A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents. *Evolutionary Computation, IEEE Transactions on*, 11(2):151–180, 2007.

[7] Gary L Drescher. *Made-up minds: a constructivist approach to artificial intelligence*. MIT press, 1991.

[8] Paul R Cohen, Tim Oates, Marc S Atkin, and Carole R Beal. Building a baby. In *Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society*, pages 518–522, 1996.

[9] Linda Smith and Michael Gasser. The development of embodied cognition: Six lessons from babies. *Artificial life*, 11(1-2):13–29, 2005.

[10] Jay S Rosenblatt, Gerald Turkewitz, and TC Schneirla. Division of psychology: Development of home orientation in newly born kittens*,†. *Transactions of the New York Academy of Sciences*, 31(3 Series II):231–250, 1969.

[11] Eric I Knudsen. Instructed learning in the auditory localization pathway of the barn owl. *Nature*, 417(6886):322–328, 2002.

[12] Jeffrey L Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99, 1993.

[13] Linda Smith. How space binds words and referents, 2003.

[14] Richard M Ryan and Edward L Deci. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary educational psychology*, 25(1):54–67, 2000.

[15] Edward L Deci and Richard M Ryan. *Intrinsic motivation and self-determination in human behavior*. Springer Science & Business Media, 1985.

[16] Daniel E Berlyne. Conflict, arousal, and curiosity. 1960.

[17] Mihaly Csikszentmihalyi. *Finding flow: The psychology of engagement with everyday life*. Basic Books, 1997.

[18] Wolfram Schultz, Peter Dayan, and P Read Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, 1997.

[19] Peter Dayan and Bernard W Balleine. Reward, motivation, and reinforcement learning. *Neuron*, 36(2):285–298, 2002.

[20] Sham Kakade and Peter Dayan. Dopamine: generalization and bonuses. *Neural Networks*, 15(4):549–559, 2002.

[21] k-d tree wikipedia entry. `http://en.wikipedia.org/wiki/K-d_tree`. [Online; accessed 2015-05-29].

[22] Pierre-Yves Oudeyer, Adrien Baranes, and Frédéric Kaplan. Intrinsically motivated exploration for developmental and active sensorimotor learning. In *From motor learning to interaction learning in robots*, pages 107–146. Springer, 2010.

[23] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[24] The python programming language. `http://www.python.org`. [Online; accessed 2015-05-18].

[25] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed 2015-05-18].

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[27] Olivier Michel. Webotstm: Professional mobile robot simulation. *arXiv preprint cs/0412052*, 2004.

[28] Numpy indexing manual. `http://docs.scipy.org/doc/numpy/reference/arrays.indexing.html`. [Online; accessed 2015-05-29].

[29] Francis Crick. The recent excitement about neural networks. *Nature*, 337(6203):129–132, 1989.

[30] Wikibooks entry on activation functions. `http://en.wikibooks.org/wiki/Artificial_Neural_Networks/Activation_Functions`. [Online; accessed 2015-05-29].

[31] Multilayer perceptron for activity recognition. `http://ape.iict.ch/teaching/MLBD2014/MLBD_Labo/MLBD_2014_lab3`. [Online; accessed 2015-05-29].

[32] Randall C O'Reilly. Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm. *Neural computation*, 8(5):895–938, 1996.

[33] Peter S Eriksson, Ekaterina Perfilieva, Thomas Björk-Eriksson, Ann-Marie Alborn, Claes Nordborg, Daniel A Peterson, and Fred H Gage. Neurogenesis in the adult human hippocampus. *Nature medicine*, 4(11):1313–1317, 1998.

[34] Ján Tóth. Intelligent adaptive curiosity github repository. `https://github.com/yanchith/iac`. [Online].

[35] Frédéric Kaplan and Pierre-Yves Oudeyer. Maximizing learning progress: an internal reward system for development. In *Embodied artificial intelligence*, pages 259–270. Springer, 2004.

[36] Josh Bongard, Victor Zykov, and Hod Lipson. Resilient machines through continuous self-modeling. *Science*, 314(5802):1118–1121, 2006.

[37] Andrew G Barto, Satinder Singh, and Nuttapong Chentanez. Intrinsically motivated learning of hierarchical collections of skills. In *Proc. 3rd Int. Conf. Development Learn*, pages 112–119, 2004.