Master's thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Architecture of Autonomous Agent Based on Cortical Learning Algorithms

Modeling human brain & mind

Marek Otáhal
Open Informatics - Artificial intelligence
otahama2@fel.cvut.cz

December 2013
Supervisor: Doc. Ing. Nahodil Pavel, CSc.

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

DIPLOMA THESIS ASSIGNMENT

Student:	Bc. Marek O t á h a l
Study programme:	Open Informatics
Specialisation:	Artificial Intelligence
Title of Diploma Thesis:	Architecture of Autonomous Agent Based on Cortical Learning Algorithms

Guidelines:

1. Study the fundamental principles of Cortical Learning Algorithms (CLA) inspired by mammalian brain.
2. Modify these algorithms to be able to produce also behaviour, aside of learning.
3. Implement CLA (or augment a current implementation) with ability to produce behaviour. This implementation should support Robotic Operating System (ROS) communication and should be as domain independent as possible.
4. Compare the efficiency of resulting learning and behaviour of agent controlled by this modification of CLA with another today used learning and decision making techniques.

Bibliography/Sources:

[1] Vítků, J.: An Artificial Creature Capable of Learning from Experience in Order to Fulfill More Complex Tasks, CTU in Prague, FEE, Diploma thesis (in English), 2011.
[2] Nahodil, P., Vítků, J.: Novel Theory and Simulations of Anticipatory Behaviour in Artificial Life Domain, in Advances in Intelligent Modelling and Simulation, Springer, pp.131-164, ISBN: 978-3-642-28887-6, 2012.
[3] Nahodil, P., Vítků, J.: How to Design an Autonomous Creature Based on Original Artificial Life Approaches, Beyond Artificial Intelligence, ISBN 978-3-642-34421-3, pp.: 161-180, Springer, 2013.
[4] Hawkins, Jeff: Hierarcical Temporal Memory including HTM Cortical Learning Algorithms, September 12, Numenta, Inc. 2011.
[5] Gerstner, W., Kistler, W.: Spiking Neuron Models. Single Neurons, Populations, Plasticity, Cambridge University Press, 2002.

Diploma Thesis Supervisor: doc. Ing. Pavel Nahodil, CSc.

Valid until: the end of the summer semester of academic year 2013/2014

[illegible]
prof. Ing. Vladimír Mařík, DrSc.
Head of Department

L.S.
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
4

[illegible]
prof. Ing. Pavel Ripka, CSc.
Dean

Prague, March 5, 2013

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student:	Bc. Marek O t á h a l
Studijní program:	Otevřená informatika (magisterský)
Obor:	Umělá inteligence
Název tématu:	Architektura autonomního agenta založená na kortikálních učících algoritmech

Pokyny pro vypracování:

1. Seznamte se s principy kortikálních učících algoritmů (CLA) inspirovaných strukturou mozku savců.
2. Modifikujte tyto algoritmy tak, aby byly schopny krom učení také produkovat chování.
3. Implementujte kortikální učící algoritmy (nebo modifikujte stávající implementaci) tak, aby byly schopny produkovat také chování. Výsledná implementace by měla podporovat komunikaci přes robotický operační systém a měla by být co nejvíce doménově nezávislá.
4. Porovnejte efektivitu výsledného učení a chování agenta kontrolovaného pomocí této modifikace kortikálních učících algoritmů s jiným dnes užívaným způsobem učení a rozhodování.

Seznam odborné literatury:

[1] Vítků, J.: An Artificial Creature Capable of Learning from Experience in Order to Fulfill More Complex Tasks, CTU in Prague, FEE, Diploma thesis (in English), 2011.
[2] Nahodil, P., Vítků, J.: Novel Theory and Simulations of Anticipatory Behaviour in Artificial Life Domain, in Advances in Intelligent Modelling and Simulation, Springer, pp.131-164, ISBN: 978-3-642-28887-6, 2012.
[3] Nahodil, P., Vítků, J.: How to Design an Autonomous Creature Based on Original Artificial Life Approaches, Beyond Artificial Intelligence, ISBN 978-3-642-34421-3, pp.: 161-180, Springer, 2013.
[4] Hawkins, Jeff: Hierarcical Temporal Memory including HTM Cortical Learning Algorithms, September 12, Numenta, Inc. 2011.
[5] Gerstner, W., Kistler, W.: Spiking Neuron Models. Single Neurons, Populations, Plasticity, Cambridge University Press, 2002.

Vedoucí diplomové práce: doc. Ing. Pavel Nahodil, CSc.

Platnost zadání: do konce letního semestru 2013/2014

[illegible]

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická -4-

L.S.

[illegible]

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 5. 3. 2013

Acknowledgement / Declaration

I would like to express my gratitude to my supervisors, doc. Ing. Nahodil Pavel, Csc. and Mgr. Jaroslav Vítků, for providing me with the opportunity to work on this interesting topic, their devoted time and endless support with advice.

My thanks belong to everybody participating in the NuPIC community and creating a great environment overflowing with new ideas, interesting research and friendly atmosphere.

My biggest thanks go to my family and friends for their continued support and overcoming the time when I wasn't much fun to be around with.

I hereby declare that this thesis is the result of my own work and all the sources I used are in the list of references, in accordance with the Methodological Instructions on Ethical Principles in the Preparation of University Theses.

Marek Otáhal
in Prague, December 30th, 2013

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Marek Otáhal
v Praze, 30. prosince 2013

Abstrakt / Abstract

Návrh autonomních agentů, kteří se pohybují v různých prostředích a vykazují prvky chování musí brát v potaz: samostatnost agenta, co největší doménovou nezávislost a odolnost vůči chybám. Výzkum těchto agentů v poslední době nabírá na významu s tím, jak se připojení k internetu stává všudypřítomné a pokroky v návrhu nových menších, levnějších, výkonějších, ale hlavně dostupnějších hw. čipů umožňují vznik mnoha "inteligentních zařízení". A většina těchto zařízení potřebuje spolupracovat, sdílet data a koordinovat svou činnost.

Podle výzkumu vedeného na kateře kybernetiky, ČVUT, v Praze, se velmi osvědčil návrh těchto agentů inspirovaný přírodou.

Nedávno uvolněná teorie HTM-CLA má silnou oporu v neurovědách, nabízí online učení, učení časových řad, a podporuje generalizaci a zapomínání. Rozhodli jsme se proto využít jejích výhod, rozšířit ji o emoce a pokusit se ji využít pro modelování agentů s chováním.

Navíc jsme implementovali podporu pro ROS, platformu umožňující integraci s dalšími metodami strojového učení, simulátory a i podporou robotického hardwaru.

Klíčová slova: kortikální učící algoritmy, hierarchická paměť, ALife, chování, mozek

Překlad titulu: Architektura autonomního agenta založená na kortikálních učících algoritmech (Modelování lidského mozku a kognitivních procesů)

Autonomous agents that exhibit any more complex behavior in various environments are required to be designed with autonomy, strong domain independence and fault tolerance in mind. The research of autonomous agents is recently gaining traction thanks to the increasing trend in recent years where internet connection starts to penetrate almost every aspect of our lives and environments and the advances in hardware design make chips more powerful, lower-factor, with longer battery life and more affordable for a huge variety of new "intelligent" gadgets. And all these devices need to cooperate, share information and synchronize their actions.

In accordance with the long-term research done at the Department of Cybernetics, of Czech Technical University, Prague, nature-inspired methods in agent design proved to be very well suited for this kind of work.

A new theory: Hierarchical temporal memory with cortical learning algorithms has recently been released to public and it looks promising for the task - thanks to its novel properties it can handle time series data, does online learning, implicitly uses generalization and forgetting, plus it has strong foundations in recent neuroscientific research.

We integrate this HTM/CLA theory with emotions to allow us construct agents with behavior, show how these additions allow us to "program" the autonomous agents and take advantages of promising features of this new theory.

In addition we implement initial support for ROS, a platform allowing wider integration with existing technologies, environments or even hardware.

Keywords: Cortical Learning Algorithms, Hierarchical Temporal Memory, ALife, Behavior, Brain

Contents /

/ Figures

Chapter 1
Introduction

A brief introduction to the structure and focus of this thesis, my motivation and interest in cortical learning applied to autonomous agents, setting the current research around NuPIC in context with related work on both fields of neural networks, neuro-science and artificial life modeling, and a description of our goals and approach we have decided for.

1.1 Motivation

I have always been interested in an interdisciplinary research, cognitive sciences as a combination of mathematical/algorithmic models, with biological focus taking inspiration in nature inspired technologies, evolution and specific focus on human brain. The psychological take on the matter modeling types of memories, how we learn, forget, certain mental disabilities and their causes, optical and mental illusions, ... The philosophical direction focuses on questions like what is required for thinking, how do we perceive the reality, or does the soul exist and can a matter-a brain create thinking as observed at human level?

For this purpose studying a neocortex inspired neural model and trying to apply it to agents to make them exhibit various forms of cognitive behavior as found in humans was a great mixture of my interests. I will try to find ways how to model certain cognitive functions human brains perform with the means of the chosen theory and then for various parts of the algorithm find biological evidence, if the algorithm adheres to the neuroscientific findings (or why not eventually).

1.2 Related work

There are two main research directions that are combined together in this work - the Hierarchical Temporal Memory and Cortical Learning Algorithm (HTM/CLA theory) and artificial life (ALife) research with focus on producing behavior and taking inspiration in nature, esp. behavior and brains of mammals.

The hierarchical temporal memory & cortical learning algorithms (HTM/CLA for short) is a learning model developed by Jeff Hawkins and his colleagues in 2003, currently supported by the Numenta foundation. It is built on findings in neuroscience and can be compared to deep-learning, recurrent neural networks (RNNs), and spinking neural networks in terms of biological principles and capabilities, although there are of course differences in functionality and use-cases. There are several existing implementations of the HTM/CLA algorithm now, and some have even been utilized by several commercial companies that make their business product on them. The focal point currently is the community around NuPIC, the “official” open-source implementation endorsed by Numenta. The situation hasn’t been that way at the beginning of writing this thesis, NuPIC hasn’t been open source, there were several semi-working implementations and documentation was lacking, to the current date, we’ve seen tremendous improvement in the usability, software support, and applications of the CLA.

The autonomous agent models and behavior modeling builds on the long term research of Doc. Ing. Nahodil Pavel, Csc. at the Department of Cybernetics at CTU and especially latest work of J. Vitku [1] that aims to create autonomous agent for universal environments with inspiration in animal species.

1.3 Our approach

With regards to the interests and current state-of the art in ALife we have decided to explore the concept of Hierarchical temporal memory and Cortical learning algorithms in general and ways to apply them to artificial agents who can produce behavior. An advantage is the fact that CLA theory is strongly building up on the findings in neuroscience.

1.3.1 Goals

Here are the goals of the thesis, the main interest is in exploring the novel concept of cortical learning and its applications for ALife agents and producing behavior. We also aim at implementing the support for Robotic Operating System (ROS) for better integration with other research:

- Study the fundamental principles of Cortical Learning Algorithms (CLA) inspired by mammalian brain
- Modify these algorithms to be able to produce also behavior, aside of learning.
- Implement CLA (or augment a current implementation) with ability to produce behavior. This implementation should support Robotic Operating System (ROS) communication and should be as domain independent as possible
- Compare the efficiency of resulting learning and behavior of agent controlled by this modification of CLA with another today used learning and decision making technique

Further, aligned with the CLA concept and our interest we try to:

- Maintain biological plausibility of the models when possible

...both at the biological/neuroscientific level and psychological, philosophical at the higher level.

1.3.2 Outline

The thesis in divided into six consecutive chapters. The required knowledge on the subject is building up, so for a new reader unaware of the HTM/CLA theory it is recommended to bear with us from the beginning. However, we put an effort to write the chapters as independent upon each other as much as possible, so if some topic is uninteresting to you, or you are searching just for a specific part of the work, it should be fine to skip directly to the required chapter.

The thesis structure is as follows:

- Chapter 1 provides thesis introduction, our motivation, and setting in the related research work.
- NuPIC is introduced in the Chapter 2, it is a recommended starting point for anyone new to the NuPIC research and development platform, you will find information about the community, the tools used, channels to ask for more help or find detailed information, or examples for work that has already been done.
- Followed by Chapter 3, where the theory of HTM/CLA and the main concept of SDRs are explained. It is possible to skip this chapter if you intent to apply CLA

to your problem quickly, it is essential, however, for better understanding of the concepts and differences to another machine learning techniques used currently, and to understand the differences for artificial agents using the HTM/CLA theory, their advantages and limitations thereof. The principle is nice and simple, but rather different so programming new agents might have a slower learning curve.

- Modeling behavior in ALife agents using CLAs is introduced in Chapter 4 for the first time, along with some implementation details and changes that needed to be taken. We discuss on several examples the usefulness and problems that come with this new approach.
- The next chapter, Chapter 5 covers design and implementation details of different solutions for ROS and behavior support that we came along, comparing them to some other standard approaches used in machine learning for that matter.
- Final recapitulation of our findings during this thesis can be found in Chapter 6 along with some possible further research ideas.

Chapter 2
NuPIC - CLA implementation, community & resources

The introductory part which briefly explains what is the NuPIC platform, draws a closer picture of the community, the development model, history of the project and most importantly ways to participate and/or ask for more information.

It is intended for new users and developers to gain a quick insight how the development works, what is possible to do with NuPIC, and what has already been achieved with it - to give you some motivation! The chapter is written in a lighter tone, more details how the theory works can be found in chapter about `SDRs, HTM/CLA theory` 3

2.1 Description of NuPIC

NuPIC [2–3] stands for *Numenta Platform for Intelligent Computing*, a collection of algorithms implementing neural network model based on the principles of the human neocortex and trying to be biologically accurate, while staying computationally effective. [1]). The ideological foundation for the NuPIC platform dates to the book On Intelligence published by Jeff Hawkins [4], and the first description of how the algorithms' implementation in the CLA Whitepaper [5].

The key ideas of the Cortical Learning Algorithm (CLA) theory are:

- all parts of the neocortex look the same, run the same "algorithm" in principle [6]
- information, memories are passed in a form of sparse distributed representations (SDRs ??) between the parts of the brain [4–5]
- our brains are constantly performing temporal inference in order to observe the world [4–5]
- we are learning "on the go" - that is called online learning [2]).

Figure 2.1. *Numenta, Inc*, the company behind NuPIC. (The logo represents hierarchy of regions)

[1]) Unlike the project Deep brain that focuses primarily on the precision.

[2]) Online learning, or online algorithm is a machine learning technique of unsupervised learning that is constantly processing incomming data as they come

Currently, NuPIC is an open-source project with an active community backed by Numenta, the original company that developed NuPIC for many years before it went open-source. Grok [7] is a commercial product for streaming data, an offspring of Numenta.

2.2 Community

Since NuPIC has been released as open-source for personal/academic purposes in mid-2013 [8], the community became very active and produced some interesting applications of the CLA and modifications, improvements to the code itself. The contacts person for the community is Matt Taylor [9], the "self-proclaimed open-source community flag-bearer" who has been doing an awesome job in organizing various events, the transformation to OSS, and is improving the ways community members and the company can collaborate. The communication channels stretch a variety of ways, here's a short list with description:

- *Mailing-list* is the most active place to ask questions, ask for help installing NuPIC or raise new development ideas. It is frequented by full-time NuPIC developers, enthusiasts in machine-learning, robotics, academics in cognitive sciences, neuroscience, entrepreneurs etc. The lists are separated in `nupic-devel`, for raising questions about the code, discussing pull-requests[1]) before merging, and `nupic-general` for all other discussions ranging from new ideas, interesting projects or asking for help. A lot of information can be found searching the ML archives[10].
- *NuPIC Github repository* - the complete code, new pull-request discussions and issues and requests for new features can be found on the github page [3]. The wiki is the best place to read about the project, how NuPIC works or figure out details about the algorithm. Recently a new project started and is bringing the code documentation available on-line easily [11].
- *Sprint planings, Open office-hours, Talks* The commercial product - Grok is using exactly the same code as is available online to anyone, and Numenta is trying to be as open as possible, so there are several audio-visual conferences, mostly over google hangouts : `Sprint planings` where shorter term goals, resolved or blocking bugs are discussed and tasks are assigned. `Open office-hours` are a variant of a morning coffee with the developers where any questions and issues from the community members can be discussed, and `Talks` are more like lectures, there's a given topic that is explained and later Q&A can follow. [12]
- Last, but far most popular are the *Hackathlons*, 24hrs or weekend actions where several teams work on a project, with developers ready to help them overcome some problems. Usually there are also talks on a specific topic. So far there have been two hackathlons that have been met with a great success. The event was also streamed online, so even members who couldn't attend personally could participate. [13–14]

[1]) in development with Git, pull-request is a way one developer proposes their changes to the code (bug-fix, new features, etc), for review and inclusion in upstream.

2.3 Projects and Demos implemented in NuPIC

This section is a show-case for some of the interesting projects built on HTM/CLA in NuPIC, so you can make a picture what is already possible to achieve with HTM/CLA applied in NuPIC:

- *Streaming CPU usage predictions* developed by Scott Purdy during the first hackathon, demonstrates the temporal predictions, the demo is simply streaming the current CPU usage data to the model and predicting future utilization. The source code and introductory video are available. [13]

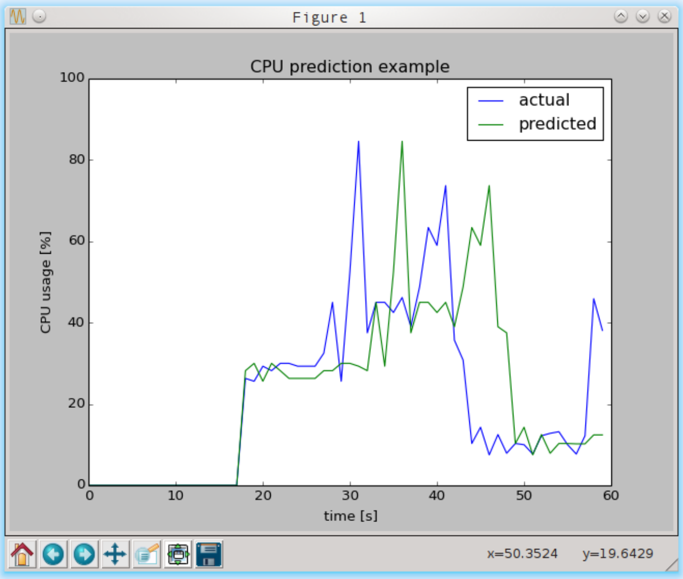
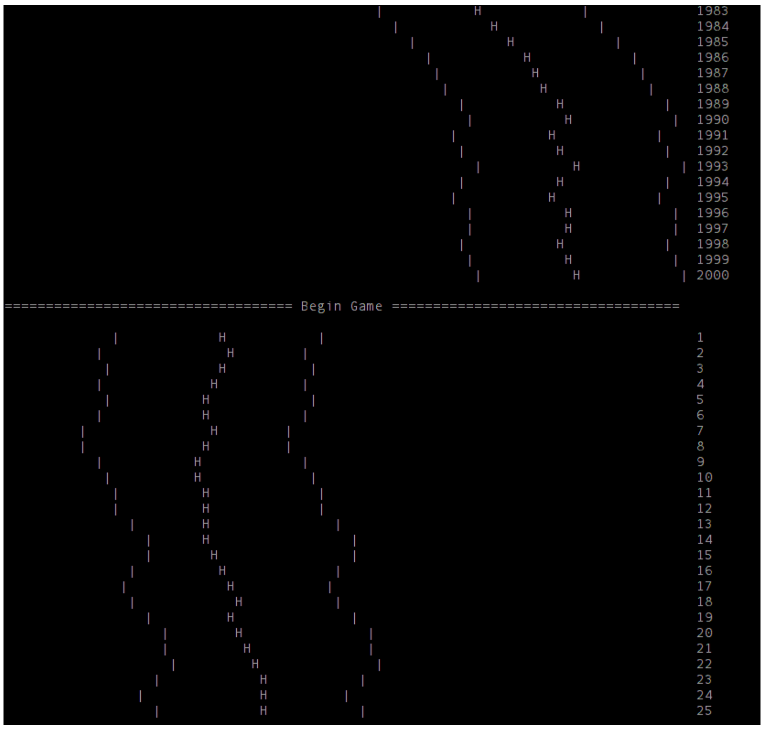


Figure 2.2. CPU utilization prediction example.

- *NTA Skier* created by Keithcom is a text based game where computer generates a random skiing slope and the algorithm learns to navigate and avoid obstacles there, after some initial guidance (learning), it is possible to "ski" the whole slope by itself alone. [15]

- *Song composition* another great outcome of the first hackathon, here the model is presented a song (in MIDI format), and later is able to follow up and produce the song even further - to unseen parts. This relates to a very interesting combination of AI and arts - computer generated music, images etc. [13]
- *Quadcopter* "This team taught the CLA to learn to control a quadcopter. Their objective was to pilot the quadcopter by controlling its vertical speed to fly to a desired altitude and hover there. The supervised learning approach involved having the CLA watch an expert flyer and learn those sequences, and use its prediction of those sequences to directly control the copter in the face of noise." [14]
- *Mice and Maze* an extension to CLA written by Eron Wright adds the ability to evaluate multiple alternatives and choose the best one. The example explains the idea on a maze where it eveluates predictions for many possible paths and chooses accordingly. [14]



```
                                             |           H            |             1983
                                               |           H            |           1984
                                                 |           H            |         1985
                                                   |           H            |       1986
                                                    |           H            |      1987
                                                     |           H            |     1988
                                                       |           H            |   1989
                                                        |           H            |  1990
                                                      |           H            |    1991
                                                       |           H            |   1992
                                                         |           H            | 1993
                                                       |           H            |   1994
                                                      |           H            |    1995
                                                        |           H            |  1996
                                                        |           H            |  1997
                                                       |           H            |   1998
                                                        |           H            |  1999
                                                         |           H            | 2000

=================================== Begin Game ===================================

             |            H           |                                             1
           |               H        |                                               2
            |             H          |                                              3
           |             H          |                                               4
            |           H            |                                              5
           |            H           |                                               6
         |               H        |                                                 7
         |              H         |                                                 8
           |           H            |                                               9
            |          H             |                                              10
             |          H             |                                             11
             |          H             |                                             12
               |        H               |                                           13
                 |      H                 |                                         14
                 |       H                |                                         15
               |          H             |                                           16
              |            H           |                                            17
               |            H           |                                           18
                 |         H              |                                         19
                   |        H               |                                       20
                   |         H              |                                       21
                    |         H              |                                      22
                  |            H           |                                        23
                |              H         |                                          24
                  |            H           |                                        25
```

Figure 2.3. Screenshot of the nta_ski game. Top half of the picture displays situation where the skier is being trained (he is placed in the middle if the track), the bottom half is where the skier is controlled by the algorithm. We can note the difference, where in the AI-controlled part the line does not perfectly pass through the middle of the track, but the skier is still able to safely complete the whole slope.

- *Linguist* is a project of Chetan Surpur [16] that falls to the category of text generation AI. It will first train on a text corpus to learn the "language features" and later can enter the prediction/generation mode to create a new text, tell you a story. We've extended it to a story-teller mode, where you can give the beginning of the sequence and the model will follow. As the model can seem just fun and childish, the use-case are for unsupervised data-mining where it can absorb knowledge from a large dataset and the user can later query it to get insights on topics, specific information etc. This is also very interesting from the research perspective, trying to replicate an impressive example made by Jeff Hinton [17] in deep-learning[1]), where grammatical features were acquired by the neural network.
- *What does the fox eat?* is a great example [14] of NLP features of CLA combined with the CEPT technology [18] for word representation during the natural language processing themed hackathon. Based on the popular song, it feeds sentence about animals and what they eat, in the end you can ask "What does the fox eat?" and get an answer, even though the algorithm has never been told "fox eats chicken"! The magic lies in pattern association done in NuPIC and the way CEPT represents words as ontologies of terms. The figure 2.4 shows visual representation of two words (dog, cat) using CEPT and their similarity. CEPT provides API to their service and online demo which is very interesting to try out!

[1]) Deeplearning is a very promising type of really huge neural networks, deep-NNs are excelling at tasks like object recognition in images, text generation, etc.

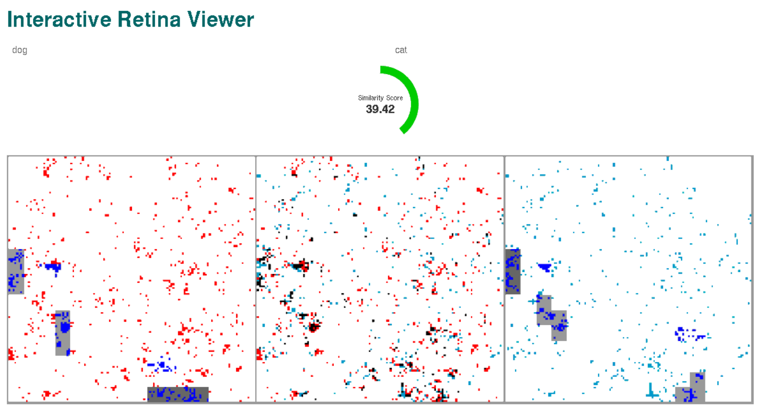


Figure 2.4. CEPT SDR representation of two words. Dog (left part), Cat (right) and their join and similarity score in the middle.

2.4 Other HTM/CLA implementations and corporate subjects

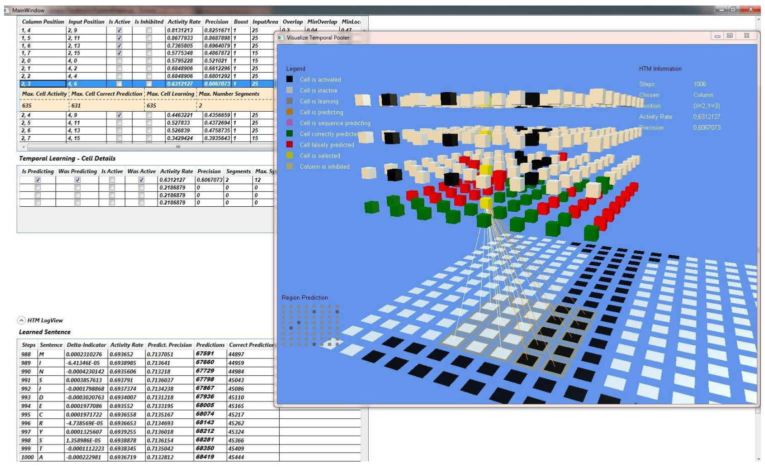
There have been several implementations of the HTM/CLA[1]) after the CLA Whitepaper [5] has been released, however their working-state and stability were varied, most of them orphaned. The situation changed after the release of NuPIC where most of the development re-united there. The list of projects known to implement CLA is listed on the NuPIC wiki, we'll describe a few notable ones:

- HTM-Camera-Toolkit is a HTM implementation[2]) based on the old NuPIC 1 version of the algorithm. Its main focus was on the visual domain - object recognition in pictures and even videos, where some nice results have been achieved.
- HTM is a simple and clean implementation[3]) of the CLA in Matlab. Based on the Whitepaper, it is useful for learning purposes.
- openHTM is a HTM/CLA implementation[4]) in .Net, a nice feature of this version is nice GUI for visualization of the neurons in the network, as we can see in 2.5.

2.4.1 htm-java

`Htm-java`[5]) was my implementation of the CLA, based on the code of `htm`[6]); I've followed the Whitepaper, differences were mostly in the algorithmic implementation, where I dare say my approach was cleaner, written in object-oriented style and more focused on parallelization and tuned for faster execution. Unfortunately it was a one-man project, so I wasn't able to give enough effort in debugging and support. I have abandoned the development in favor of NuPIC soon after it has been released to public.

[1]) list of the projects can be found at `https://github.com/numenta/nupic/wiki/Other-HTM-CLA-projects`
[2]) `https://github.com/binarybarry/HTM-Camera-Toolkit`
[3]) `https://github.com/ShermanMorrison/HTM`
[4]) `http://sourceforge.net/projects/openhtm/`
[5]) `https://github.com/breznak/htm-cla`
[6]) `http://code.google.com/p/htm/`



Column Position	Input Position	Is Active	Is Inhibited	Activity Rate	Precision	Boost	InputArea
1, 4	2, 9	✓	☐	0.8131213	0.8251671	1	25
1, 5	2, 11	✓	☐	0.8677933	0.8687898	1	25
1, 6	2, 13	✓	☐	0.7365805	0.6964079	1	25
1, 7	2, 15	✓	☐	0.5775348	0.4867872	1	15
2, 0	4, 0	☐	☐	0.5795228	0.521021	1	15
2, 1	4, 2	☐	☐	0.6848906	0.6612296	1	25
2, 2	4, 4	☐	☐	0.6848906	0.6801292	1	25
2, 3	4, 6	☐	☐	0.6312127	0.6067073	1	25

Max. Cell Activity	Max. Cell Correct Prediction	Max. Cell Learning	Max. Number Segments
635	631	635	2

Column Position	Input Position	Is Active	Is Inhibited	Activity Rate	Precision	Boost	InputArea
2, 4	4, 9	✓	☐	0.4463221	0.4356659	1	25
2, 5	4, 11	☐	☐	0.527833	0.4372694	1	25
2, 6	4, 13	☐	☐	0.526839	0.4758735	1	25
2, 7	4, 15	☐	☐	0.3429424	0.3935643	1	15

Temporal Learning - Cell Details

Is Predicting	Was Predicting	Is Active	Was Active	Activity Rate	Precision	Segments	Max. Sy
✓	✓	☐	✓	0.6312127	0.6067073	2	12
☐	☐	☐	☐	0.2186879	0	0	0
☐	☐	☐	☐	0.2186879	0	0	0
☐	☐	☐	☐	0.2186879	0	0	0

HTM Information

Steps:	1006
Chosen:	Column
Position:	{X=2,Y=3}
Activity Rate:	0,6312127
Precision:	0,6067073

Learned Sentence

Steps	Sentence	Delta-Indicator	Activity Rate	Predict. Precision	Predictions	Correct Prediction
988	M	0.0002310276	0.693652	0.7137051	67591	44897
989	I	-6.41346E-05	0.6938985	0.713641	67660	44959
990	N	-0.0004230142	0.6935606	0.713218	67729	44984
991	S	0.0003857613	0.693791	0.7136037	67798	45043
992	I	-0.0001798868	0.6937374	0.7134238	67867	45086
993	D	-0.0003020763	0.6934007	0.7131218	67936	45110
994	E	0.0001977086	0.693552	0.7133195	68005	45165
995	C	0.0001971722	0.6936558	0.7135167	68074	45217
996	R	-4.738569E-05	0.6936653	0.7134693	68143	45262
997	Y	0.0001325607	0.6939255	0.7136018	68212	45324
998	S	1.358986E-05	0.6938878	0.7136154	68281	45366
999	T	-0.0001112223	0.6938345	0.7135042	68350	45409
1000	A	-0.000222981	0.6936719	0.7132812	68419	45444

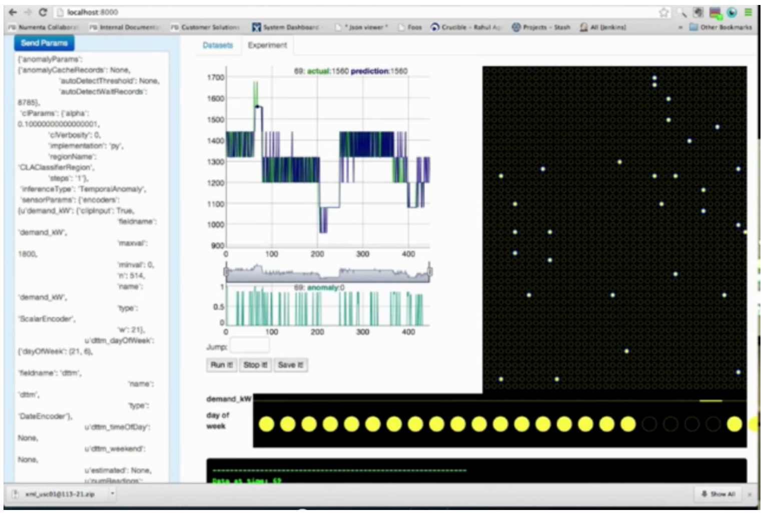
Figure 2.5. OpenHTM's visualization of the temporal pooler and letter 'I' at the input.

2.5 Tools

Several tools have been developed around NuPIC, these tools are not essential part of the NuPIC library, not needed for running. But provide means to visualize, experiment, debug and fine-tune your HTM/CLA applications better. The most important of these tools are:

- *Serialization* is not actually a stand-alone tool but a part of the NuPIC implementation[1]). As the name suggests, it allows to store and load a trained network with its weights. This is useful in many cases, eg. when you train and start off from a pretrained network on a large dataset and just want to accommodate to the new data.
- *Cerebro* is a standalone tool[2]) for visualization and debugging of a CLA. It is useful for learning how the algorithm works, debugging details in new implementations, or it can be used to generate new datasets. You can find a very explanatory video[3]) with details how to run and use Cerebro. Figure 2.6 shows Cerebro instance running a CLA on `hotgym` dataset.
- *Swarming* is an extended parameter tuning tool[4]) for NuPIC, given a dataset, it will evaluate multiple models and fine-tune parameter combinations for them. Internally the tool uses particle-swarm optimization (PSO) to find optimal combinations. Swarming is a recommended step for final deployment of your model. More extensive information can be found on the wiki or an instructive video[5]).
- *Unit tests* are part of the core NuPIC library to ensure valid and reliable behavior for (several) implementations of the core parts of the algorithms.
- *Pre-installed virtual machine (VM)* NuPIC has been ported to run on various platforms and most common OSs, should some problems occur, we have provided a preset VM [19] with a running NuPIC instance, complete development environment plus the tools mentioned in this section for a quick and easy way to experiment with the algorithms.

[1]) wiki `https://github.com/numenta/nupic/wiki/Serialization`
[2]) source repository + instructions: `https://github.com/numenta/nupic.cerebro`
[3]) `https://www.youtube.com/watch?v=WQWU1K5tE5o&feature=youtu.be`
[4]) Swarming repository and instructions: `http://numenta.org/resources/blog/Swarming_in_NuPIC.pdf`
[5]) `http://www.youtube.com/watch?v=xYPKjKQ4YZ0`



```
{'anomalyParams':
{'anomalyCacheRecords': None,
            'autoDetectThreshold': None,
            'autoDetectWaitRecords':
6785},
 'clParams': {'alpha':
0.10000000000000001,
          'clVerbosity': 0,
          'implementation': 'py',
          'regionName':
'CLAClassifierRegion',
          'steps': '1'},
 'inferenceType': 'TemporalAnomaly',
 'sensorParams': {'encoders':
{u'demand_kW': {'clipInput': True,
                              'fieldname':
'demand_kW',
                              'maxval':
1800,
                              'minval': 0,
                              'n': 514,
                              'name':
'demand_kW',
                              'type':
'ScalarEncoder',
                              'w': 21},
                    u'dttm_dayOfWeek':
{'dayOfWeek': (21, 6),

'fieldname': 'dttm',
                                'name':
'dttm',
                                'type':
'DateEncoder'},
                    u'dttm_timeOfDay':
None,
                    u'dttm_weekend':
None,
                    u'estimated': None,
```

Figure 2.6. Cerebro - a tool for visualization and debugging of the state of a CLA. We can see Cerebro showing a visualization of the hotgym dataset running in CLA, it presents a graph of predicted and actual output, anomaly score of the input (how unexpected it has been) and the black plane with a few white dots represets an SDR, output layer of the algorithm with a few cells active.

2.6 Robotic Operation System support

Support for Robot Operation System (ROS) is completely new to NuPIC, although ROS itself is a very mature and well established project [20]. ROS is a collection of machine learning algorithms, drivers for sensors and robots, and various simulators of virtual reality, and other tools. ROS has been used with success in works of J. Vítků [1] for behavior agent simulations. It does not bring any direct change to the functionality of the HTM/CLA algorithm but offers much more comfortable development and wider field of problems CLA can be applied to.

More details about how I approached the implementation[1]) are in 5, despite early stage, it seems the implementation has reached interest of some other users who are now working with it.

The resulting ROS API implementation is easy to use, transparent to existing CLA functionality and allows to connect HTM/CLA with other machine learning techniques, be it either integration of other ML algorithms into a HTM network, or using HTM/CLA as a part of a ML solution.

2.7 Domain independence

In principle hierarchical temporal memory model with cortical learning algorithm is domain independent[2]), which is a big advantage to many more specific ML techniques. Of course this needs to be taken with a grain of salt, just like eg. "classical" neural networks [3]).

[1]) the code for ROS support to NuPIC is in `https://github.com/breznak/nupic/tree/ROS-encoder` and waiting for inclusion upstream.
[2]) because of the uniformity of neocortex [6]
[3]) According to Kolmogorov's theorem, any neural network with sufficiently large hidden layer is a universal approximator; which roughly means it could learn to predict output of any function. Even if this is true, NNs are still impractical for a variety of problems (mostly the precidion-time required to learning compromise).

According to the HTM teory and with support in biological evidence, the structural[1]) and algorithmic functionality[2]) is the same in all parts of the brain.

On the implementation point of view, *encoders* are piece of software that needs to be tailored for specific use-cases (in a sense of quality - eg. different encoders for different inputs like sound, images, numbers, or words. Encoders for inputs of, say, numbers 0 to 5, and numbers 1 to 1000 would be the same). At that level, encoders represent our "senses" [3]), the senses transform from the reality-space to the internal representation understood by the brain, as mentioned earlier. An illustrative example is vision where a cat (reality) reflects light, it reaches our eye (the sensor/encoder) which projects the light on the retina where some of the light-sensitive cells become active and send out electric signal (that is the internal brain representation, SDR), as shown at fig. 2.7.

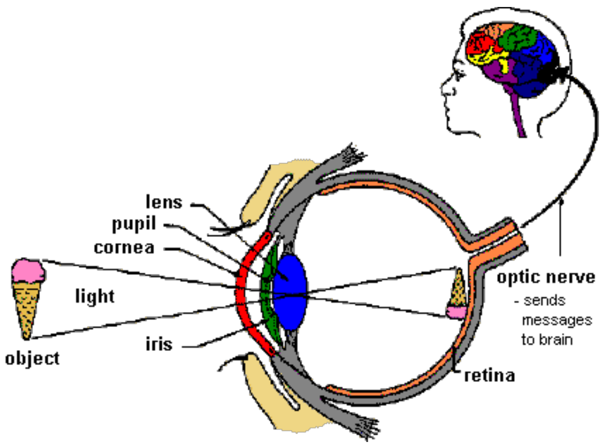


Figure 2.7. Real world object seen by the eye, projected on the retina and transfered to brain, as an analogy to sensors in NuPIC. [4])

Another degree of freedom in the algorithm are the parametric variable, respectively their values. For most cases there are defaults that work reasonably well, or it is possible to refine them using any chosen methods. One provided for NuPIC is the swarming mentioned earlier 2.6. The biological analogy for swarming is *accomodation* of our vision to darkness/light. When we are reading at night with the lights on, we are used to the light, if suddenly you turn the light off, initially you wouldn't see anything, but soon (after a couple of seconds) the eyes will accommodate and you'll start to see in the dark much better.

2.8 Things left TODO

Even though the theory has been researched for almost 10 years, and its implementation - NuPIC is in continual development for years as well, there are still some major issues (both in theory, and in the algorithmic implementation) that will need to be resolved.

Including:

[1]) Contrary to the *Neuro-evolution* approaches where quality of the solution for different problems is sought to emerge from different wiring of the neural network.

[2]) for example recurrent neural networks are suitable for one kind of problems (temporal sequence predictions) but fail for other

[3]) Even though in NuPIC, one could have different senses for rational numbers, or words.

- *hierarchy* - is the core part of the HTM theory. It was implemented in earlier versions of NuPIC [1]) that was focused on the visual domain. The code is still available, but has not been used in the current version, therefore will likely require some modifications and testing.
- *core library and stable API* after many discussions, there is a work in progress to strip down the requirements and dependencies of the NuPIC as a software project, with the aim to create a minimalistic core library (written in C++) with stable and well defined [2]) API for interfacing with other languages [3]) and simplified porting to new hardware platforms.
- *reconstruction* is a means of reversing the bottom-up computation of a layer. Normally, a layer is given an input, will process it and produce a result, reconstruction is an "inverse" to this process, more precisely, it will approximate the most likely input such that would produce the given output at the top of the layer. This is an example of another functionality that had been a part of NuPIC but has been removed.[4]) It is important to realize that in real brains this functionality is actually not required! [5])
- *sensory-motor integration* extension to HTM/CLA with functionality of the 5th region of the neocortex (V5). This part is believed to be responsible for motoric functions, generation of actions, feedback to lower layers and so on.

[1]) reffered to as NuPIC 1, this old version of NuPIC is completely different to today's NuPIC (2) which uses the CLA learning algorithm. NuPIC 1 used Zeta learning algorithm, the work on NuPIC 1 has been focused around the visual domain - in recognition of objects in video streams. It has been discontinued by Numenta because of speed reasons, the current version of NuPIC2 in teh commercial product Grok is focused on work that can be done in a single CLA region.

[2]) Another ongoing task is to create online documentation to the code

[3]) currently only Python and C++ are supported

[4]) replaced with *Classifier* a code which is not a part of the HTM/CLA theory, but for usability reasons is included in NuPIC. This code pairs the inputs with the outputs.

[5]) The difference between human brain and our usecases in AI is that brain always uses its internal representation to cummunicate between different parts of the brain (say an SDR) - so once a picture of the reality (we see a red car) enters our senses (eg. vision) the retina of the eye converts it to the internal representation, which is passed and processed in several regions of the brain (V1,...,V5 vision regions, or any other parts of the brain). The message probably changes, but the encoding (SDR) stays the same. On the contrary to our common usecases for the algorithm, where we need to interpret the internal representation back in the input-space of the reality - take another example, we want to predict stock price of the USD/EUR pair, the inputs are values of several currencies at given times of a day, and we want to predict the price in future. This is where some way of conversion back from SDR to USD is required.

Chapter 3
Applied Theory of Hierarchical Temporal Memory & Cortical Learning Algorithms

The aim of the following chapter is to introduce the model used - Hierarchical temporal memory (HTM) and its learning algorithm - Cortical learning algorithm (CLA) which together combine to the HTM/CLA theory. The HTM/CLA theory tries to describe how the neocortex works and transfer this knowledge to a new machine learning algorithm. HTM/CLA has a very sound biological foundation supported by findings in neuroscience, but still it is a machine learning algorithm which has simplicity and real-world use in mind[1]).

The chapter offers a quick reiteration of the theory from my side, the more in-depth resources are available at [4–5] or video talks [21–22].

Now let us look how neocortex is structured and how the information is processed in it:

3.1 Hierarchical temporal memory

Hierarchical temporal memory (HTM) is a machine learning concept inspired by the structure and algorithmic functionality of the neocortex. It is a type of a neural network that is programmed by exposure to streaming sensory data (that is called on-line learning). The core unit in HTMs are `cells` [2]), the cells are arranged in columns (there is the same topological structure in the whole neocortex [6]), layers, regions, and hierarchy. The specific structure plays an important role and is what differentiates HTMs from other neural networks[3]) [23].

HTM is composed of smaller units, called Regions, and these regions[4]) are ordered in a *hierarchy*. That has 2 important consequences: smaller networks are trained faster and consume less memory - because the patterns can be reused, combined to describe new patterns; and at the higher layers of HTM the patterns become more stable[5]) (the cause is explained later on in this chapter in spatial pooler), which leads to *generalization* and improved learning. The figure 3.1 depicts a HTM with two sensory inputs, consisting of 6 regions arranged in 4 layers.

[1]) unlike the project Blue Brain `http://bluebrain.epfl.ch/` which seeks to model brain functions precisely up to the smallest level (chemical reaction in synapses) - this is because primary goal of BlueBrain is to research brain diseases, while NuPIC (Grok) is used for commercial applications of streaming web data

[2]) Cells are usually called neurons in real brain and other neural networks

[3]) The topological importance of the network and impact on the performance is not a new idea (neuro-evolution), but HTM does not do a “blind” evolution but rather takes inspiration in how specifically it is done in mamalian brain (neocortext) and wires accordingly.

[4]) note that each region is like a small neural network itself

[5]) A stability of a pattern means how much is it changing with new inputs in time. For illustration, imagine inputs as: a big tiger, a cute kitten, an old dog,... These are varying laregely in most aspects, yet at some level of abstraction, all of them can be described as `an animal`, a stable pattern.

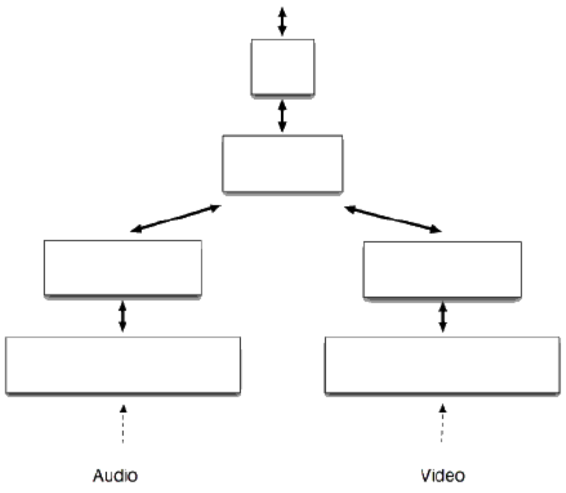


Figure 3.1. Regions (white rectangles) connected to a hierarchical structure form a four layer HTM (with 2 inputs). [1])

3.2 CLA region

A region is the main building block for each HTM structure. Basically a region is what we call a neural network in classical machine learning, but with a specific structure that is interestingly same for all parts of the neocortex [6, 24].

A region[2]) composes of many cells ordered in a 2D fashion (a layer) but also in a vertical structure - columns, as we can see on the figure 3.2.

A column (here composed of 4 cells) maps a part of the input space[3]) by connecting a (shared for all cells in a column) proximal dendrite to a subset of the input synapses.Thus forming a randomized feature. Figure 3.3 shows a neocortical column with 4 cells (C1-C4) with a shared proximal dendrite linking them to a set of 10 potential synapses. The set of *potential synapses* is a pool of total synapses available for the column. Whether they are considered as relevant (active) for the column is decided by the *permanence value*[4]).

A synapse is considered *active* if its permanence value[5]) reaches a certain threshold (these are depicted as filled dots), while inactive synapses have a low permanence value below this threshold (empty dots). Please do note that as the requirements of the algorithm adapt during the run, synapses may become active or inactive. Note the difference to classical neural networks, what is called an output of a neuron there is in CLA actually output of the column, not a particular cell!

In terms of functionality, a region performs functions of a spatial pooler (SP) and a temporal pooler (TP), both of them will be explained now. Although their functionality is merged in a real physical region, for the purposes of the algorithm SP, and TP are separated, as they perform different tasks and actually either of them, or both can be used in different scenarios.

[2]) Regions are shown in 3.1 as the rectangles.

[3]) By "input space" here we mean either the real input, or actually an output of a layer directly below this region.

[4]) Eg. noisy, "broken", or irrelevant (uncorrelating, inactive) synapses for a features will not be considered.

[5]) Permanence value means, roughly speaking, an infogain of the synapse - if it correlates well with the activity of this column, the synapse is selected.

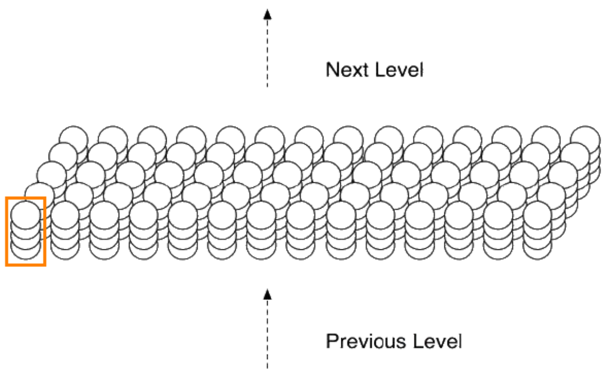


Figure 3.2. Each region of neocortex (and in HTM/CLA) composes of a number of cells (a cell is depicted as a circle) and the cells additionally form columns (the vertical structures; one of them being highlighted).

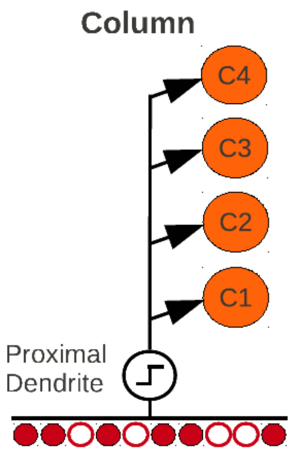


Figure 3.3. Cells forming a columnar structure, a shared proximal dendrite connects them to the potential synapses where the synapses that correlate with the overall activity of the column are selected as active (filled dots) and considered for computing columns activity (On/Off state).

3.3 Spatial Pooler and Sparse Distributed Representations (SDRs)

A spatial pooler (SP) is a part of each cortical region and an (optional) part of a NuPIC region. As it physically is a region, its structure is the same as described in 3.2 where we talked about a region. The main functionality of the spatial pooler is to create sparse distributed representations of the input.

To achieve the SDR representation, the cells in a region are not only connected in a vertical fashion by proximal dendrites (as shown in 3.3), cells also connect in a horizontal meaning by distal dendrites to other cells within the same layer. This kind of horizontal connections allow the SP to perform inhibition[1]) [25]. So if a column

[1]) Inhibition is a process when active column forces (inhibits) the neighboring columns to be inactive. This is important property, otherwise all columns within the network would be active at once - this is called an "explosion"

becomes active (thanks to one of its cells), it inhibits all of its neighboring cells. The functionality is shown in figure 3.4, where the cell C2 is connected with 5 distal dendrites (the 5 rows of blue filled/empty dots, each dot represents a connection - a synapse - to a cell within the dendrite's reach. Same as in 3.3, filled dots represent active synapses and empty the inactive ones. This is called a lateral input of the cell.) A logical OR from the distal dendrites represents the predictive state for the cell. The cell's output value is a boolean `OR` of the predicted state (driven by neighboring cells) and the column activation (by proximal dendrite, a red line on the left side).

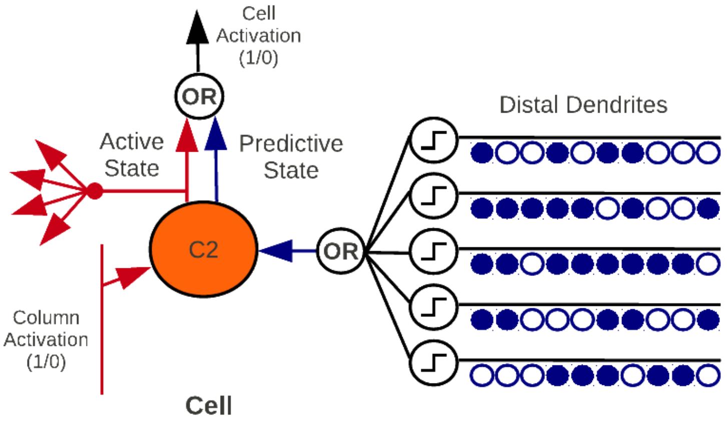


Figure 3.4. Computation of the output state for a single cell: If the feed-forward input (related by the proximal dendrite, shown as a red line) is On, the cell's output is set as On. Or if any of the (5) distal dendrites (the lines of 10 dots) that connect to cells in another columns but within the same layer has sufficient input from the active synapses (filled dots), the cell enters a predictive state and is also marked is On. Otherwise the output of the cell is Off.

Sparse distributed representations (SDRs for short) are the key concept of how the CLA works. The main idea of Jeff Hawkins's HTM/CLA theory is that all parts of the brain operate on the same principle and thus can communicate with same messages. And an SDR is the message type regions share with each other.

On the figure 3.5 you see a grid of neurons (cells in NuPIC) where the active cells are highlighted.

To explain the principle behind the SDRs, we'll analyze its main features.

3.3.1 Sparsity

This means only a small portion of the total number of columns is active at a time. It is a fixed percentage (2% as observed by neuroscience [4], and that value is also used as a default in NuPIC) of active neurons. Such stable value is interesting to find (and seems wasteful, but we will show nice implications coming out of this requirement), obviously two cases occur: the number of active neurons is higher than 2% and thus has to be reduced, or the other case where insufficient number of neurons fired at a same time (active).

Inhibition is a well known mechanism for the first case, where the output is too dense and some columns need to be chosen to be `turned off`. It ensures a dense representation becomes sparse.

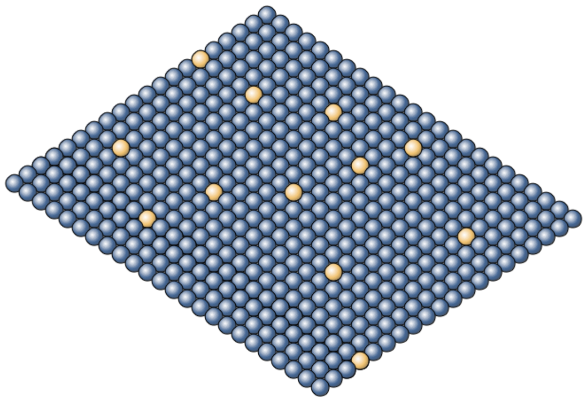


Figure 3.5. Sparse distributed representation (of a spatial pooler) with active neurons highlighted. Credit: F.Byrne

In the algorithmic implementation, inhibition can be either local, or global. Local inhibition is the one how mammalian brains work: neurons have horizontal inhibitory synapses connecting neighboring neurons. Whenever one neuron's activation potential reaches the threshold (neuron becomes active, fires), the inhibitory signal is sent along the inh. synapses to block possible activations of the other neighbors. This is crucial for the network to avoid a state of chaos where almost all neurons fire, known as `network explosion`[1]) from the spiking neural fields theory.

Global inhibition does not exist in biological networks, but in principle is similar to local inhibition, and is easier to implement algorithmically, that is why it is commonly used. Just like its used in NuPIC. The principle of the global inhibition is simple: if we want to ensure 2% sparsity (only 2% of cells active at a time from the whole spatial pooler) we will represent s "quality" of the activation and choose just such a number of the total active columns to make for a 2% of the total number of columns.

The opposite case is when insufficient number of columns is active. The method to fight this is called *boosting*. We can imagine this as a round-based game. In each round some fixed number of columns are the winners. When a column hasn't won for a long time (it is becoming useless) it will try to "adapt and be more eager". Becoming eager means a) lowering (some of) its existing synapses' permanences, thus becoming more competitive and increasing a chance to fire first for a certain pattern and winning. Or b) the neuron can grow new synapses (and/or drop the old ones) from the pool of potential synapses of the belonging dendrite thus covering a new (and hopefully) unexplored part of the input space.

One could say it's wasteful to represent patterns in the sparse way. Yet, unlike computers, brains are heavily parallel so this is not a burden. Quite the opposite, useful features of `SDR arithmetic` appear.

An example of sparse encoding is when an input vector of 30.000 elements is converted to a representation of 10.000 bits where only 2% of the bits are ON at a time (that is 200 out of 10.000). Same would also apply for a smaller input, say 50 elements, this would also be sampled to 10.000 bits representation with a 2% sparsity.

3.3.2 Distributiveness

[1]) State when all of the neurons fire because their neighbors fire and the horizontal connections made them active too.

The other important property of an SDR is that they are distributed. That is a way how the information is encoded to the number of bits. The encoding can have each bit independent - this concept is called a Grandmother cell[1]), the other pole is an encoding where a bit does not bear a meaning, only the whole set of bits conveys the meaning - example of such an encoding is the binary code. The obvious disadvantage of Grandmother-cell encoding is its memory requirement for increasing number of possible input patterns and inability to determine if two patterns have "something in common". On the other hand, the disadvantage of dense (eg. binary) encoding is the sensitivity to noise[2]), and also the completely different representations for similar concepts.

The response to these issues is the distributed representation which is a compromise from the both words. Each bit conveys some (fuzzy) meaning, but the bits need to be interpreted in context of a large number of other bits to describe a certain pattern. For example the 200 out of 10.000 bits as we discussed above.

3.3.3 Summary of Features of SDRs

Key ideas of SDRs are that they are sparse and distributed representations of the input pattern, that means a small subset of bits(columns) is active at a time (2%). The mechanisms that regulate the number of active neurons are inhibition and boosting.

Recapitulation of features that are derived from these properties:

- *Robustness to noise*
  Because SDRs are distributed representations, there must always be a big number (200) of bits to encode a pattern. Even if a number of bits is "broken", the remainder of the ON-bits pattern fits the original pattern with a very high probability - Of 10.000 bits, some 200 bits are selected to represent a pattern, still knowledge of only 50 of these 200 active bits gives us a good chance that the pattern is the original one.
- *Capacity*
  Even though the sparse representation may seem wasteful, the number of combinations, thus patterns the SDR is able to represent is huge. For an SDR with 10.000 bits with 2% sparsity it is (10.000 choose 200) !
- *Similarity*
  Similar concepts (input patterns) will have similar[3]) SDRs[4]). This is a very useful property - it allows us to tell if "cat" is similar to a "tiger", is the cause of resistance to noise, allows merging of concepts - SDRs for "a pear" and "an apple" can be merged, or even sub-sampled.

3.4 Temporal Pooler

As the spatial pooler's main role is to form sparse distributed representations of the patterns, the role of temporal pooler (TP) is to represent the notion of time, or sequential order of events. The temporal pooler fits patterns in context of previously

[1]) In "Grandmother cell" encoding each bit conveys its meaning as a stand-alone unit. In principle it is an enumeration, for example, to represent 10.000 words, we'll make a list of the words and each bit in our representation would match one word. So if 163rd word is 'Grandmother', the 163rd bit means 'Grandmother'.

[2]) Changing a single bit in 4=(100)2 can lead to (000)=0!

[3]) Similar in a way of Hamming distance

[4]) Because the way how proximal dendrites sub-sample the input space to form features. ??

presented (because HTM/CLA does online learning, data are continuously streamed into it) inputs.

A good example could be when I say a "dog", you partly know what I mean (a dog, not a cat, of course), but all in all it does not make much sense. If you knew I said "My new dog Zag", or "The wild dog that bit me", the message would be much more clear. And this is exactly what the TP does. It allows the SDR for "dog" to always look the same, because I said literally "dog", but also adds more information to tell the difference between many contexts of the word.

The functionality is achieved by the number of cells in each column. Remember that the output of the column (logical OR of any of its cell's outputs) is what forms bits in the SDR. As there are more cells in a column and the cells connect to other cells with distal (predictive) connections, CLA is able to represent: when a "dog cell"[1]) is active, CLA can learn that usually either the "Zag cell" or "that" cell will be active in the next step (and form the predictive link). As shown in figure 3.6 where SDRs for the dog are the same, but the context is different (represented by the colored cell - either orange for "My dog Zag", or blue for "The dog that bit me").

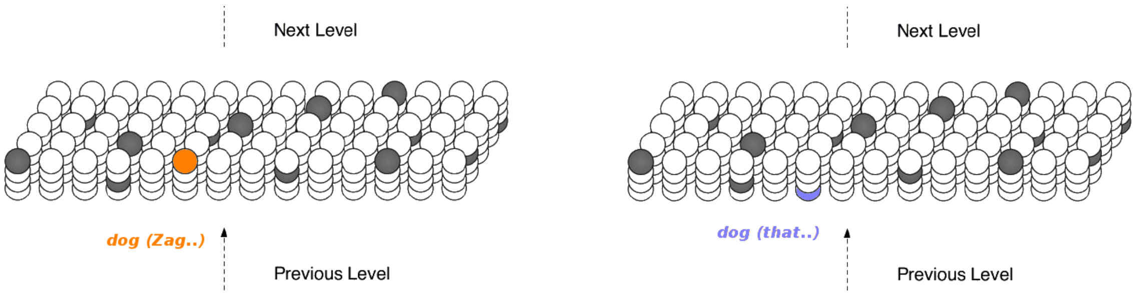


Figure 3.6. The temporal pooler represent word "dog" in two contexts. The SDRs (active columns) are the same, difference is in the cell(s) (here only 1 colored for simplicity) that make the columns active. Cells form predictive connections by distal dendrites between other cells at the same level to represent context.

The temporal pooler is responsible for the sequence memory, a type of memory that is crucial for most of our cognitive tasks. Very nice examples are described in On Intelligence [4], to name a few:

Our ability to remember and recognize songs (even if the pitch does not perfectly fit). By giving you the first tone, you would know nothing about the song, after the second note the number of possibilities is reduced, so with every other consecutive note, and already after 3 - 5 notes we could be able to recall a melody of the whole song!

Another, not so obvious example is the sense of touch. If I place an object in your open hand and you are not allowed to move, you probably would not recognize what the object is. To do so, one will need to swipe on its surface to make a feeling of the structure, move the hand around the object to create and idea of the shape and size, after that, it would be possible to tell whether it is an apple, or a car.

[1]) Actually a "dog pattern of cells" because of distributed representations.

3.5 Summary: Key Concepts of HTM/CLA

The most important findings to take from this chapter are:

- Neocortex has a same structure and performs same "algorithm" in all of its parts. This suggests existence of a unified algorithm that is capable of performing all of the tasks human neocortex is responsible for. That is the goal of HTM/CLA.
- Our perception and thinking happens in a continuous stream of input data, the brain is performing on-line learning and is constantly making predictions, representing things in context of previous events. 3.5
- The patterns in brain are represented as sparse distributed representations (SDRs) and have their useful properties - similarity, robustness to noise, invariance, possibility to subsample or merge patterns.

Chapter 4 Producing Behavior in a Biologically Plausible Way

The following chapter aims to bring the concept of *behavior*[1] [27, 26] to cortical learning algorithms, a fusing of the cortical learning (CLA) represented by NuPIC and modeling of behavior in artificial agents. The concept of behavior is completely new to NuPIC[2] therefore a number of modifications needed to be done. From the side of artificial life (ALife)/behavior - the work is based on research of Vitku [1], Kadlecek [26], and Svatos [28], however, because of the diference of HTM/CLA the implementation is rather different. The novelty and limitations of my approach come from the goal to stay biologically plausible as much as possible. As we will see, some things which are considered as easily done by to-date machine-learning (ML)[3] are hard with this new (and willingly limited) approach, on the other hand, new problems that are currently difficult[4] are possible with the CLA theory.

The text is separated into sections representing different aspects[5] which we expect from an agent exhibiting behavior. Role of emotions [29] will be discussed and whether/why they would be needed for an agent that exhibits behavior but still on much lower level than feeling *in love* with somebody; problems passing our intentions to the agent, or how to force to make our agent robo-car run through fire and not stay safe in the depo? How does a baby learn to walk? And could we learn to use telepathy, or night vision as a 6th sense? What are dreams? Or why does your child always gaze at a handicapped person in the subway? These interesting questions apparently do not fall to the field of machine learning nor neuroscience, and still should be answered in the following text.

[1]) Definition of behavior:
"Behavior refers to the actions or reactions of an object or organism, usually in relation to the environment. Behavior can be conscious or unconscious, overt or covert, and voluntary or involuntary. In animals, behavior is controlled by the endocrine system and the nervous system. The complexity of the behavior of an organism is related to the complexity of its nervous system. Generally, organisms with complex nervous systems have a greater capacity to learn new responses and thus adjust their behavior. Behavior can be viewed from various views and thus is studied in different disciplines.", quotation from [26]

[2]) Pull-request with code and discussion: `https://github.com/numenta/nupic/pull/407`

[3]) like a simple planning taks with breadth-first search.

[4]) like agent that learns and operates with a language (NLP domain), all sorts of vision tasks

[5]) like cognitive functions, ability to carry out similar mental tasks as we can observe in mammals, or even only on humans (young babies will be very common target of our focus during this chapter because their brain is the most plastic and they are just beginning to explore everything, create patterns in mind and in brain literally.)

4.1 Behavior in NuPIC

Based loosely on the Belief-Desire-Intention (BDI) model[1]), [30], there is an active entity, the agent, and the environment it operates in, the world. The world can be of any nature, it just requires its "laws"[2]) to be defined.

For the examples here and for easier visualization let's assume a "classical" 2D-map based *world* (see figure 4.1) which is composed of tiles, and each tile can have specific features (contain food, be occupied by another agent, cause damage, etc.) In multi-agent scenarios the environment and resources are shared among all the agents (who can compete on them).

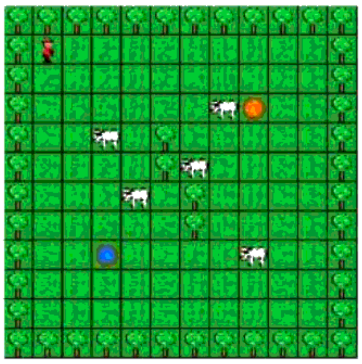


Figure 4.1. An example of a 2D environment (agent's world) with two resources (food, water) and several tiles with other properties (cows, trees).

Belief of the agent represents its *knowledge* of the world. Depending on the model, either the whole world map is known to the agent since the beginning, or the agent has to explore and reveal the properties. In that case, the map can be incomplete (missing, unexplored areas), or even inaccurate (a faulty sensor disturbs the measurement, or the reality changed while the agent's map haven't been updated - eg. another agent takes all the food from a tile we have marked as "hasFood" before and haven't revisited ever since). These limitations require that any planning based on this belief is fault tolerant to some degree. This would be analogical to memory in humans, with a slight difference that for multi-agent setups, the memory could be shared (a variant of us humans having telepathy and sharing thoughts), but that is just an implementation detail. Throughout the code, this is referred to as a *utility map*. Figure 4.2 shows a utility map of a agent living in a 2D world and the utility corresponds to "attractiveness" of each tile (the lower, the better).

Another property of agents we use in our models is an *internal state*, a set of variables that are not shared with other agents and correspond to "personal properties" of each agent instance. For example these would include: agent's name, its current position on the map, its target location (the goal), how hungry the agent is (how much battery it has left), etc. Some of these stats stay over time (the goal, name), some change with each action (the position), and some can

[1]) Agent has its beliefs, desires, intentions, avail. actions and an environment

[2]) like physics, but in a general sense. For example there could be a world where states are letters, and rule is: whenever 'X' is encountered, next state would be 'A'.

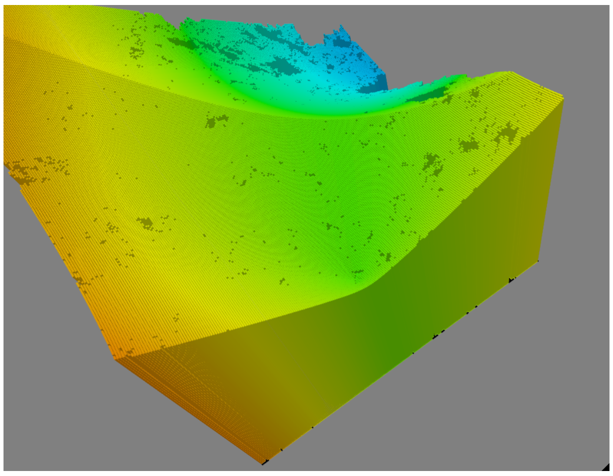


Figure 4.2. Utility map showing an incomplete knowledge. It is a visualization of a 2D-plane with utility being the 3rd (up) dimension. The agent is minimizing the utility, so the lower, the better. Missing places represent unvisited, or obsoleted states - we do not have any knowledge about them.

change only in certain condition (the hunger increases when walking, and decreases when the agent takes action “eat food”).

That leaves us to the final properties of an agent, its *senses*[1]) and *actuators*. A set of actions which agent uses to perceive the world (a GPS for location on the 2D map, “eyes” or a camera allows to tell what properties (food, danger, cow) are on each tile, ...), and a set of actions that interact with the world (allow the agent to manipulate the world (eg move box to the next tile), some actions manipulate the inner states (action eat will increase the food agent has - reduces hunger state).

4.2 Memory

Some sort of memory is essential for all but very simple reactive agents. As HTM/CLA is a memory based system, it already has this feature and allows to learn and predict sequences very well[2]) The ability of CLAs to do online learning and accommodate the sequence to their capacity makes them ideal for analogy of *long term* and *short term* memory known from psychology and cognitive sciences.[3]) So to answer the first of the questions: `How are memories stored in brain?` - it seems memories are represented by SDRs and are highly context sensitive, so they would be stored is sequences in CLA regions.

4.2.1 Generalization

Generalization is an ability to learn from experience and apply the knowledge to new, unencountered situations. By definition, generalization of a concept is

[1]) Senses are represented by the Encoders in CLA; Encoders transform the reality (input space) to something the CLA understands (arrays of 1/0).

[2]) This is where simple neural nets are insufficient, and more complex recurrent NNs must be deployed. A disadvantage of RNNs is that they train relatively bad, resp. it’s time consuming. Not like online learning of CLAs.

[3]) Another successful ML technique to deal with these types of memory is TLST-mem.

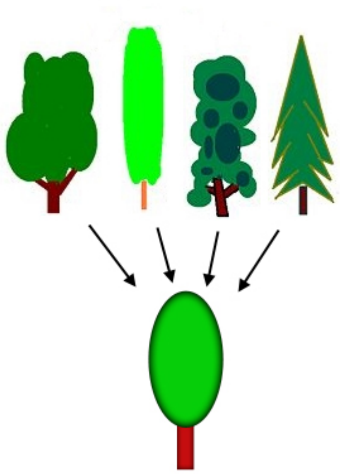


Figure 4.3. Generalization finds the common, defining features in a set of discrete objects and creates a condensed representation with only the essential features.

"extension of the concept to less-specific criteria". It is an important element of human(-like) reasoning.

The *important-feature extraction*[1]) property of generalization (as on 4.3) is the exact thing that happens when spatial pooler creates a SDR of the input pattern. And the resulting "essence" is the reason why higher levels in HTM theory represent more stable patterns and allow *abstract thinking* - thinking about concepts, or concepts of concepts.

Another interesting thing about generalization in human mind is that it is strongly concept specific [2]), this fits perfectly in the CLA's sequence memory - where same object ("a dog") can be interpreted in many different contexts (my dog, dog as an animal, dog as a slang term, ...)

4.2.2 Forgetting and Dreaming

Forgetting and dreaming are interesting properties of human (and animals') memories. It has been shown that forgetting is not a negative effect of our minds (within reasonable amounts), it is even essential for our brains to operate effectively[3]). Otherwise we would become bloated with random details of memories, rendering any reasonable thinking difficult or impossible.

It is an interesting fact that there are three "types" of forgetting implicitly available in the CLA:

- *Outliners*: outliners, errors[4]), or any statistically insignificant data are remembered at first but forgotten soon during the process of learning, as the CLA is a statistically based learning model.
- *Feature selection and missing values* as a property of the sparse distributed representations, the model will pick only the features from the inputs that carry

[1]) `https://en.wikipedia.org/wiki/Feature_selection`

[2]) From one point of view, dogs and cats are very much the same - both are mammals, have four legs and we raise them as pets in our homes. From another point they are completely different - different species, dogs chase cats, Ben is a dog, but not a cat!

[3]) `http://faculty.washington.edu/chudler/sleep.html`

[4]) can be caused by faulty sensors, a mistake in the environment situation, user input, ...

significant information and will discard the useless features. Another property of SDRs is ability to handle incomplete input (missing or otherwise corrupted values) well.

- *Dealing with limited capacity*: a nice thing about CLAs is that there is no need to specify to which details we want to remember [1]) and what should be considered unimportant and forgotten. Like with the outliners 4.3 the CLA will balance the "information storing" capacity of the CLA (determined by the number of columns), the details remembered from the inputs (what is already considered an outliner in the current data), and adapt to new trends emerging in the input[2]).

And what is the role of dreaming? Recent research shows that sleeping is crucial for our brains: during the sleep (and dreaming) chemical reactions occur in brain and help neurons and synapses revitalize[3]), forgetting takes place (and brain is throwing out useless information), and also the contrary happens: the brain replays (emotionally) important events to structure and strengthen the knowledge[4]) - dreaming is a time when brain is idle, not overwhelmed with multiple sensory inputs as usual, and uses this time to improve the quality of stored information, or just meaninglessly generates some activity as it's used to, probably a mix of both is what happens.

4.3 Emotions

Emotions express how we feel, some researchers even consider them as superfluous in the concept of artificial intelligence. In my opinion, emotions are a key idea for expressing goals, desires in agents, allowing us to make the agent do what we need in a biologically plausible way, because without them, there would be no reason for the agent to do anything.

This is not unique only to machines, it of course applies to humans as well. Why do you do anything at all? Why do you go to work every day? Why did you stop at a supermarket on the way home to buy a chicken and salad? And why are you considering to get married? These questions could go on and on. Your answers, motivations, urges and reasons may vary, but down there (at the bottom of the Maslow's pyramid on figure 4.4) there is an answer, because you've been programmed to do so! And emotions are the messengers who relate the needs of the body and mind. Don't get me wrong, I do not imply that our destiny is written: we have no free will etc. I'm saying that in evolution (of species and of

[1]) or respectively, how many columns does the model require? We just allocate some number (usually 2048 columns in a region) and the CLA will try to utilize them to the best.

[2]) For example, if a CLA is encountering numbers with precision to 2 decimal digits (3.14) in range from 0 to 10, it is able to learn and recall them with the fill precision. now that it encounteres a large value (1900) for the first time, it is considered an outliner. As more and more values in range 1000-5000 arrive, the CLA will adapt to represent them too, at the expense of reduced precision for small numbers (say it will not be able to remember the decimal precision, only values like 1, 2, 3, ...). Should the shift in trend continue and only data in values 1000 - 5000 keep arriving, the initial small numbers (¡10) become marginal, an anomaly and the model will forget them.

[3]) that's why bigger mammals with physically bigger and more complex brains need to sleep more.

[4]) All of us dream some of the time (actually many times) at night, but we only remember the dreams when we wake up at the right moment. And the dream is the state of the brain (the sequence of memories) which we have just realized as we woke up and became aware. That is also the reason why we often dream about stressful, worrying, or happy moments of the past day(s). And why dreams sometimes seem "crazy, not making sense", because two or more unrelated information can be processed at a time and they become mixed together during the dream.

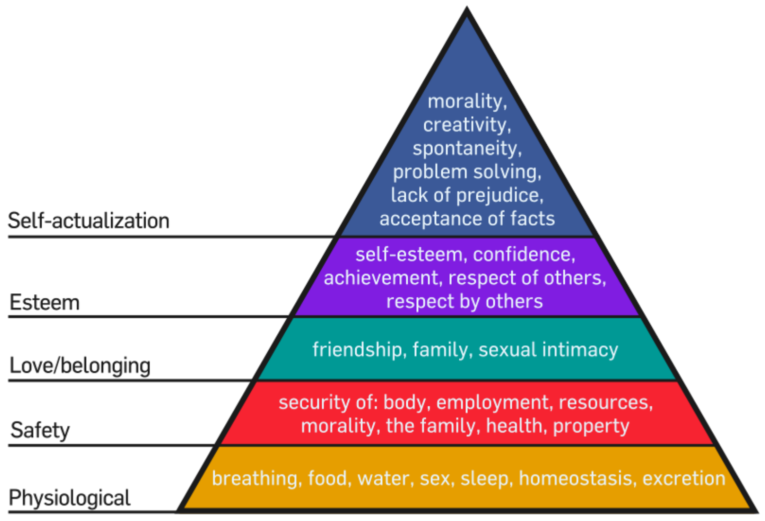

morality, creativity, spontaneity, problem solving, lack of prejudice, acceptance of facts
Self-actualization
self-esteem, confidence, achievement, respect of others, respect by others
Esteem
friendship, family, sexual intimacy
Love/belonging
security of: body, employment, resources, morality, the family, health, property
Safety
breathing, food, water, sex, sleep, homeostasis, excretion
Physiological

Figure 4.4. An interpretation of Maslow's hierarchy of needs, represented as a pyramid with the more basic needs at the bottom.[1])

individuals as well) those who survive and reproduce succeed. That seems to be an objective fitness function for a long time.

Low level emotions relate directly to body functions, some examples are:

Need to eat, to sleep, reproduce, stay in optimal living conditions, etc. These emotions express a state of the body and are related to brain by levels of certain hormones the organs are able to produce. For example when you are hungry, the body produces Ghrelin hormone[2]) and you get a `I'm feeling hungry` feeling in your brain.

We could say senses are represented by emotions as well. For example when you touch a red hot iron, a sensation of pain (haptic sense) flows through your body (the nervous system, to be more exact) to the brain.

These emotions are tacit knowledge, we don't need to learn them, we have them instantly. This is important, because emotions are not learned by the brain during our lives (the brain uses them heavily in its work, but does not initiate them). This assumption allows us to say that when we aim to create artificial intelligence (strong AI) by taking inspiration from how the brain works, it is OK to hard-code emotions directly. That is, implement them in a non-biologically plausible way and still we can be sure none of the important concepts of how the brain works is missed. Compare this to a case where we would `emulate` long-term memory by a (classic relational) database, such model would not be plausible and it's possible some important part of the brain's functionality would be missed and we wouldn't succeed in constructing a strong AI.

For the purposes of the algorithm some other, artificial, emotions can be constructed. For example if you need to make your agent go to a target location (say point [5,2]) we would create an agent with an artificial emotion dragging it to that point. Now you should say `That's wrong, it's not how the brain works!`, and you'd be right. Doing so, however, is not wrong either. It's completely true there is no emotion in your head dragging you to the point [5,2], or to buy chicken

[2]) `https://en.wikipedia.org/wiki/Ghrelin`

on the way home. However, you buy the chicken, so that you can cook it later, give one half to your girlfriend and eating the other. Thus avoiding a death from starvation and the state of sadness from death of your significant other. These, rather complex, consequences bubble through the hierarchy of the (real) brain regions and you've spend years of practice to create the knowledge of relations, consequences, experiences of states - you've been mapping the state-time-action dimension space for a very long time. In this light, creating an emotion for `go to the target` is a concession for the simplicity. But the result is the same, and even the way the flow of events is processed in brain is the same too. It's important to note that emotions are related to the environment, the body and the goals (lifestyle) of the agent, so it is perfectly ok to omit some of the emotions that are not relevant (for example hunger in a world where there is no death and creatures don't eat; or emulating sensation of pain where the only goal of the agent is to get to the target location and the only limit is distance - aka shortest path search).

4.4 Action-learning

Action learning is the process of adopting new actions, or to be more precise, re-learning their prerequisites and effects once either the environment's laws, or the actuators have changed. Some real life examples of these include: drinking in the space-station[1]) (`new physics` of the environment), using an artificial limp after operation (new actuator), or the action can be more abstract, for example `go to work` changes when you relocate to a new city and are unable to make a 5 minute walk to work every day, instead you need to take a bus or drive a car.

The time when we experience most of the action-learning, and also the time when our brains are the most flexible to learning new things is of course the babyhood and early childhood (same applies to all mammals and animals in general). Some of the most prominent examples of such behavior are when a new born baby is adopting its senses. From the prenatal time when it was in the mothers belly, the baby is used to her tone of voice and thus is able to recognize the mother. Eyes, however, have not been used before and in the early days the vision is only blur, unable to recognize further objects nor the parents' faces. This is the reason why we place rattles close to the child's bed and the baby likes to watch them and train the vision. Same applies to haptic coordination of hands and fingers, the baby tries to grab and move small toys, practicing fine-motorics.

In artificial intelligence, as well as for humans, a common example is learning to walk. Humans start to learn walking around the 10th month and it takes them several months to achieve this ability. On the contrary, some animals have to learn it almost instantly to survive (young giraffes know how to walk in a few hours). If you think it's an easy task, I suggest you try the QWOP-game[2]), where you learn to walk again, by fingers. In machine learning reinforcement learning (RL) and genetic programming (GP) are most commonly used to achieve action-learning. For example the walking robot in GP[3]).

The experiments with action learning agents who use CLA are in the *ALife* repository [31]. The two primary concepts of CLA are in use for this process - sequence memory and generalization. The simpler approach represents actions

[1]) Video of an astronout drinking in space: `https://www.youtube.com/watch?v=Fg1RMEIP6i4`

[2]) Game where you learn to "walk" by your fingers: `http://www.foddy.net/Athletics.html`

[3]) Walking robot learns by genetic programing `https://www.youtube.com/watch?v=LL0ajYq8A_0`

and states as a same entity, so the CLA is learning a sequence of these states: `S1--A1--S2--A2--S3...`, where `S(i), A(i)` are distinct states, resp. actions.

The other approach is similar to the one that will be used in the Planning problem 4.6, we make the space of a Cartesian product of the:
`(world's state) x (agent's internal state) x (parametrized action) x (context)` but we modify the `utility-encoder` to write the best action (instead of utility score) to the `knowledge` of the agent.

The obvious drawback of this approach is the curse of dimensionality[1]), but because of the CLA's generalization and ability to deal with missing values, it makes the proposal feasible, and most importantly, we believe this is the way we train ourselves too.

4.5 Planning

Planning is a very important task in machine learning, basically it means executing right actions at the right time and under right conditions. CLA's ability for temporal predictions (when to execute) and generalization (what action) make it a good candidate for a planning agent.

There are two ways how to apply the planning, one of them is relatively simple, the other would be biologically plausible but is complicated.

The first, simple, approach is to use a CLA as a "helper", a module to store and predict next state in a sequence, or suggest most suitable action given certain conditions. Then some other, non biological [2]) algorithm will evaluate utility of the select state/action and from a set of available actions pick the most suitable one.

The pseudo-code looks like this:

```
in the current state;
for-loop:  from all possible actions:
--evaluate utility of outcome of the action applied in this state;
pick the best action;
```

That is the classical approach, the for-loop for all actions, and picking the best one are not implemented in CLA, that means brain couldn't perform it that way. We have found a solution to this problem by shifting the problem to a weird, multi-dimensional space where a state is represented as `(state in the environment) x (parametrized action) x (inner state) x (state of the CLA -the weird part!- it depends on previous context)`. And in this space we implement simply an optimization rule (override with better, eg. lower value). The tricky part is that this space depends on the context of the agent, so if you come to me from the right side and say 'Hello', my reaction could be different to when you come from the left and say 'Hello'.

However, there could still a problem of transition between two sequences. Let's assume the agent has learned to follow the path of points A–B–C–D (the D is important [3])), and another path C–X–Y. And we want to go from A to Y. But the problem is how to merge the knowledge? We are at state `C`, the sentence 1 suggests to go to 'D', the sentence 2 suggests to go to 'X'. And as CLA is doing

[1]) `https://en.wikipedia.org/wiki/Curse_of_dimensionality`
[2]) that is how current planning algorithms work.
[3]) if the first sentence ended with C instead (the end-point of first one and starting-point of the other were the same), the problem will not occur and the transition would be clear.

online learning (experience, statistical based), if we had walked path A–D 100 times, and C–Y 10 times, the odds would be 10:1 in favor of going to 'D', instead of taking path to 'X' which is the correct one and will leave us to 'Y'. Even adding hierarchy and look-ahead predictions does not help much.

We have come up with a hypothesis that predictions' probabilities could be altered in favor of path containing the desired destination with emotions (a mix of reinforcement learning and emotions applied to CLA), but we still investigate details of the implementation and if there is an analogy in brain's functionality. This is one of the open problems of CLA application with focus on biological plausibility.

4.6 Attention

Attention to the details, ability to recognize what's new, what has changed, the feeling something is suspicious! ...all of these are crucial for living creatures that operate and compete in complex environments. *Anomaly detection*[1])[2]) is the principle behind these features that seem like a complex behavior. And it's happening automatically with the CLA - you can always say how much some input was unexpected in the previous context and learning experience.

The importance of attention is well known in the psychological literature [32], however it has rarely been used in ML because anomaly detection is a hard task in itself. Imagine the astronomical amounts of data coming through your senses at every moment[3]). Such amount of data would overwhelm even such a highly parallel system as a brain is, not to mention such a poor approximations as computers where the data makes the bottleneck for usability of AI in some domains (vision, ...).

4.7 Higher-level cognitive concepts, thoughts, Abstraction

I have found a very interesting definition of happiness, which says: "the brain is happy when it finds a better description of the reality, allowing it to store the same (or higher) amount of data with less required information."[4]) That means lowering the entropy, more effectively storing the information with lesser energy demands. And that goes in hands with the hierarchical principle of the brain and the HTM theory.

A simple example can be found *feature selection* mentioned above. This means the brain is trying to create higher-level concepts, abstractions.

An interesting conclusion for this is that we are `programmed` to learn, it makes us happy. The humans, our brains, are curious. And especially babies..

The biological support for abstraction can be observed on young babies as well. During the process of adopting new abilities - senses, not only we learn how to `operate` the hardware, as we improve our ability to perceive the world, we also extend and improve our map of it, the mind and memory.

[1]) `https://en.wikipedia.org/wiki/Anomaly_detection`

[2]) Anomaly detection in HTM/CLA: `https://github.com/numenta/nupic/wiki/Anomaly-Score-Memo`

[3]) Resolution of eyes is 576Mpix! `http://www.clarkvision.com/imagedetail/eye-resolution.html`

[4]) `http://www.davidpublishing.org/DownLoad/?id=13874`

A game babies love is “Peek-a-boo!”[1]). The baby and a parent are sitting and facing each other. Then the parent will close their eyes, cover them with hands and open them in a moment later (a few seconds) and smile at the baby. The baby will follow (babies also love mirroring), it will look afraid, surprised when it closes the eyes, and so happy and relieved when open again, you’ll be rewarded with a happy smile on the baby’s face. I bet you know this game and you have played it as a baby yourself, and have/will play it with your child when you have one, no matter what culture, country you are from. Does it sound like a silly game to you? The baby is just exploring and learning one of the key properties of our physical world - that is: the (macroscopic) objects have duration and exist (in reasonable sizes) in time and space. In fact, your smart baby is just philosophizing the old question “Does a falling tree make a noise if there’s no-one to hear it?” That is the reason of the happy smile on the baby’s face, it’s genuinely happy that you haven’t disappeared to the void, or haven’t turned into a dragon. It’s not only babies who think about that, for example an Australian Indian tribe believes you live another life when you close your eyes and dream at night[2]). And it’s not foolish to think so, there are worlds where this `longevity` property does not exist, for example in the quantum `world`: there is something, then you look at it... and you’ve just broken it.

Another fine game to play with a baby is “Where is the toy?”. Here you’ll need some of the baby’s favorite toys, and some object to hide it, a blanket or a box will be just fine. Show the baby the toy and then put it under the blanket, so the toy is not visible. The baby will be puzzled where the toy could be, uncover the blanket and the baby is happy again. Later it will learn to put the blanket away and recover the toy. So, what is the baby learning here? For example the perspective, a property of the vision domain, that a bigger object placed between us and our target hides the target. Or simple the concept of `A is behind B`.

How does this translate to cortical learning? In HTM, abstraction is happening automatically with new, more stable patterns forming in the higher levels. And the way the spatial and temporal pooler learn in the cortical learning algorithms. Even though we cannot exactly say “this is a pattern for –being behind–”, the more common (in context) patterns are learned.

One such practical example, which was successfully realized with HTM/CLA in NuPIC is from the visual domain, where the HTM was thought to understand invariance to rotation and translation. That means the algorithm is aware the observed object is the same, even though it’s rotating or moving in the picture. It is a well understood tracking problem, however the way the CLA is thought is different. Details of the invariance vision experiment can be found at the wiki[3]). This example is an exact analogy of how the young child is becoming aware of `persistance of objects`, as mentioned above.

[1]) Peek-a-boo game for children: `https://en.wikipedia.org/wiki/Peekaboo`

[2]) `http://dreamhawk.com/dream-encyclopedia/australian-aborigine-dream-beliefs/`

[3]) `https://github.com/numenta/nupic/wiki/Vision:-Object-Recognition-Using-NuPIC`

Chapter 5
Discussion of the Implementation of CLA-based Agents

The description and discussion of some of the details we have faced in our approach to the implementation of an artificial agent controlled by the HTM/CLA algorithm, and relation to other machine learning approaches currently used.

5.1 Design of the CLA-based Agent

Our premise was that we want to explore if the HTM/CLA algorithm we acquainted ourselves with is suitable for construction of an autonomous agent. Our motivation for such a decision is in the features of the HTM/CLA theory. It is a nature-inspired agent based on the functioning of the mammalian neocortex which is capable of performing very complex and, even more importantly, a wide variety of cognitive processing and behavior.

As the HTM/CLA is a memory based temporal learning model constantly performing online learning and predictions able to deal with anomalies, adapt to new trends in inputs, the core functionality is simply covered by this algorithm.

The agent is constantly improving its representation of the world - a knowledge map. Depending on the implementation described later, the map actually really exists as a 2D array holding a utility map and then a reinforcement learning (RL) like method is used to guide the agent on a known utility map terrain, for example minimizing the utility. The other method stores even the utility map in the CLA memory, therefore even the 2D map similarity with Q-learning disappears.

Technically, aside of that, the simulation environment (the `World`) and the `Agent` were only needed[1]). The relevant code can be found in the ALife repository [31] [2]). The environment provides the simulation for the agent plus it defines the laws applicable in that world[3]). The agent must consist of a set of actions applicable to the World - the actuators, sensors to perceive (parts of) the environment, the HTM/CLA control unit, and optionally a utility function to evaluate certain “states in time-space”, emotions to form goals and characteristics of the behavior[4]) of the agent. More details how the utility function and changing levels of hormones (the emotions) can alter the plans of the agent can be found in the chapter 4 - Modeling Behavior.

[1]) As the NuPIC implementation is still being developed in an active pace, a number of modifications were needed to iron things out and be able to perform experiments we wanted.

[2]) Plus additional branches to the main NuPIC repository: ROS-encoder & utility-encoder from my repository https://github.com/breznak/, if these were not yet mainlined.

[3]) Such as what happens if the agents steps outside of the World’s borders? - It dies, stays at the same spot, wraps around to the other side of the map?

[4]) For example an “excited” agent who performs exploration when reaches an unknown state, or a lazy agent who does nothing, or a coward agent who returns to the last know state.

Because of the nature of the HTM/CLA and the range of its features, the classification of the model used for our CLA-based agent is slightly complicated. The agent with utility function and emotion whose behavior is controlled by the HTM/CLA algorithm falls into these categories:
It is a cognitive architecture, a model-based, utility-based learning agent, an input-based agent [1]), a processing agent [2]) and finally a world agent. [33–34]

5.2 Comparison of HTM/CLA with other Machine Learning Techniques

To evaluate the important properties, limitation and learning methods it is useful to try to study (a partial) classification of the HTM/CLA algorithm itself.

HTM/CLA is a neural network model inspired by the mammalian neocortex, in the context of neurobiological models it is similar to convolutional neural networks, HMAX or deep-learning neural networks recently proposed. In regards to ability to represent temporal memory, HTM resembles recurrent neural networks or temporal logic.

When we look at general purpose probabilistic models, HTMs belong to the category with Bayesian networks, Energy based models, Hierarchical Hidden Markov Models or Bolzman Machines that analyze statistical relationships between variables.

Although it is not the primary purpose, it is possible to emulate supervised learning with HTMs, just like discriminative models do. These models avoid trying to find a mapping of inputs to the desired output (not exactly of the same structure as the original model) where are the well-known Support Vector Machines and “classical”[3]) neural networks.

5.3 Features of CLA-based Agents

A recapitulation of features of the HTM/CLA with regards to the demands of autonomous agents.

- *Domain independence*: as the CLA is quite universal algorithm, the core functionality of the agent is domain independent. We have designed the agent to be parametrized by the utility function which allows to define desired goals and characteristics of the behavior. This makes agents reusable for different tasks, but of course some code modification is needed for new environments and tasks. Also new set of actions and sensors are required for various environments.
- *Contextual memory and notion of time* The agent is able to perform different actions in different context and create plans with time perspective in mind.
- *Generalization and invariants* Agents are able to form ideas about concepts like “is similar to”, “was unexpected”, and so on.
- *Biological plausibility* As the model adheres to nature inspired model, some other cognitive functions are possible to model, as a side-effect. For example forgetting, ignoring always-present sensory inputs, modeling of emotions, convergence to forming low-energy state representations of the memory.

[1]) Neural network based agents.
[2]) That solve signal processing problems, eg. speech recognition.
[3]) Using back-propagation.

5.4 ROS Implementation

For the purposes of other research done at our department, and for interoperability of this new solution, we preferred to be able to abstract the CLA algorithm as a node in Robot Operating System (ROS). This approach has been pioneered by J. Vitku [1], where both “talker/listener”[1]) and client/server[2]) were implemented. For NuPIC, I have decided for the listener/talker based approach in Python API for ROS because the core API for NuPIC is still not clearly separated, and in this approach we implement the ROS support as a new Encoder to NuPIC, which presents just a plug-n-play functionality. It is completely transparent to existing CLA based applications, the support can be added by creating a new instance of the respective encoder and attaching it to intended part of the pipe-line, just like in flow-based programming. A publisher ROS-encoder will broadcast all data passing through it to specified ROS topics, and ROS listener encoder will listen to all given topics and send received messages further through the CLA. Details for ROS specific functionality can be found at the Tutorials page [20].

5.5 Memory Representation

As HTM is a memory based system, the agent’s memory representation of the world can be stored completely inside the HTM/CLA, which is a novel approach to most of the agent architectures.

We have implemented two approaches to this topic: a “classical” memory where the knowledge of the agent about the world is stored in a map of the world - for example a 2D map where each position holds some information about the world, and also a utility for doing certain action at that position. In [1] this approach is successfully used. They assume agent to have a set of (possibly divergent) goals each goal with a utility map for given position in the map. And based on agent’s changing! internal state (“need for food, sleep, ...”) the overall utility is computed as a linear combination of these sub-goals, so goals can shift around as needed.

The process we take with CLA is quite different, although it looks almost the same. There is just one utility map shared for all goals. The solution assumes that for each objective there exists an optimal solution as a sequence of actions, states, parameters and timing. As the CLA is context aware, we are able to have a different quality of a state for the same position, but different sequence of steps leading to it. So we will create a `adaptive utility map` that stores a utility and the best action for each position depending on the optimal sequence of events that lead to it. As the CLA is context aware (the path how we have reached the point) so we cannot use a simple utility map as is used in Q-learning, instead, we store the sequence x action pair’s utility in the CLA’s context memory. Normally this would create unfeasible dimension expansion but because of CLA is able to recognize similarities in parts of patterns or sequences, the solution seems feasible (at least for reasonable problem instances).

In our second approach we even avoid the 2D array structure for storing the values and attempt to model everything, including the sequence memory and the utility map in the CLA. This has an advantage of even simpler implementation of

[1]) Distributed message-passing, topic-based architecture; please refer to ROS for details.

[2]) Classical client-server, request-response design.

the agent and possibly more general capabilities, of course at the cost of increased demands on the CLA algorithm and its recall capability.

Chapter 6 Conclusions

We have been able to create an autonomous agent exhibiting behavior (in [31] and chap. 4) that is controlled by the CLA. Most significant, although expected finding is how completely different it is to "program" the agent using the CLA-only technology, and also that it brings some very nice benefits (sequence memory, generalization, forgetting, robustness to unknown inputs), and also has its limitations (more complicated way to program the agent).

We have thought-through various interesting analogies between cognitive functions of human brains and their realizations within the HTM/CLA framework - such as how is forgetting realized, what happens while dreaming, ability to learn to control completely new kind of limbs, planning the way from work home, providing motivation to the agent, exploring unknown space and so on.

The agent controlled by CLA takes advantage of the HTM/CLA theory main features - anomaly detection, generalization, self-accommodation to new trends in input patterns, fault tolerance, sequence memory - some of which are uncommon in classical machine learning.

Two kinds of learning have been implemented, `mirroring` where a sequence of action-state-action is learned. Here CLA performs very well because it has a strong sequence memory. For some problems (4.6) this approach is not well suited and a more general, multi-dimensional learning has to be used. With this attitude we face the curse of dimensionality, but CLA is able to deal with it to some degree.

6.1 Diploma thesis contributions

Only a listing of goals of the thesis that have been achieved and some additional features I consider note-worthy.

- ROS support has been added ([35])
- joined development to advance the CLA implementation
- functionality needed for agents exhibiting behavior is possible with CLA ([31])
- further researched into analogies between workings of HTM/CLA theory and cognitive processes in mammals (at the neural, behavioral, psychological level)
- designed thought experiments to further test abilities of CLA based agents
- compared the differences of autonomous agents controlled by HTM/CLA and other approaches
- identified missing functionality in the current implementation of NuPIC

6.2 Remaining open issues

This is a list of issues we have encountered and which either blocked or complicated our experimenting with CLA-driven autonomous agents.

- *simple core library*: this is a work in progress now, the NuPIC code-base is very complex, written in combination of C++ and Python withs callbacks in Swig. It still contains obsolete code, multiple implementation of similar functionality and the core CLA code is not clearly separated from other supportive functionality. That presents a burden for some more adventurous redesign aims. This issue has been very well discussed and a pursuit to make a stripped down, tiny, portable and understandable library is underway.
- *reintroduce reconstruction* so that "input" patterns can be obtained for each component of the region in a top-down manner[1],[2].
- *Planning with CLA only* - the research problem, how to evaluate and choose multiple hypothesis with a CLA-only approach.
- *Improve speed/memory requirements*: according to our experiments, for some interesting experiments huge datasets would be required (like all texts of Wikipedia - 50GB, or video processing), to carry on these experiments currently is very difficult or almost impossible due to memory and speed consumption of the CLA. We have proposed some solutions and these are also issues for Grok, so they will be definitely worked on.

6.3 Future research

I definitely want to continue my programming tasks on NuPIC, seeing to some of my open pull-requests to get merged and participate on the open issues mentioned above.

Right after this thesis I plan to join Fergal Byrne on his great idea of improved spatial pooler with predictive (distal) connections[3]. This would mean improved predictions and anomaly detection, allowing for reconstruction in the spatial-pooler and of course again a more biologically plausible model.

I want to spend some more time investigating an even more biologically accurate implementation, somewhere even at the cost of reduced functionality/convenience or speed. I'm expecting some of these changes would be acceptable to the up-stream, some probably wouldn't and would stay just for research purposes.

Other plan is to execute some of the big-dataset projects (domains of NLP, vision come on my mind), maybe including hierarchy, and see if some qualities emerge just from the sheer amount of data. This will likely uncover another algorithmic limitations, mostly speed issues.

Once hierarchical support in the NuPIC takes better shape, it would be possible to extend our existing agents based on CLA (4) to HTM-based agents and gain advantages that (theoretically) come with HTM[4].

Interesting idea would be experimenting with pre-trained CLAs, their sharing, distribution etc. This is related to serialization (??)

And last but not least, I would like to spend my future research on brain-machine interaction. More precisely, brain-CLA interfacing/extensions. That is the reason I have set the ever-reoccurring "biologically-plausible" limitation on myself. The

[1] `https://github.com/numenta/nupic/wiki/Reconstruction`

[2] ML discussion `https://www.mail-archive.com/nupic@lists.numenta.org/msg01695.html`

[3] `http://inbits.com/2013/11/adding-prediction-to-the-nupic-spatial-pooler/`

[4] esp. this would be more stable patterns at higher levels of the hierarchy and abstract patterns.

technology for brain-computer interfaces (BCI) exists, the CLA model seems to adhere to the brain principles, there would be some limitations to be overcome [1]).

[1]) eg. the difference between CLA's model producing '1' and '0', while real neurons (and eg. spiking NNs) use frequency encoding. Some maybe connecting CLA with a SpikingNN (through ROS?) could be the first milestone!

References

[1] Jaroslav Vitku. *An Artificial Creature Capable of Learning from Experience in Order to Fulfill More Complex Tasks.* CTU, Prague, 2011. Diploma Thesis.

[2] Numenta. Numenta, inc website, 2013.
`http://numenta.org/`.

[3] Numenta. Nupic github repository, 2013.
`https://github.com/numenta/nupic`.

[4] Jeff Hawkins and Sandra Blakeslee. *On Intelligence.* oct 2003.
`http://www.citeulike.org/user/prolog/article/705837`.

[5] Numenta. *CLA Whitepaper*, 2013.
`http://numenta.org/cla-white-paper.html`.

[6] AJ Rockel, RW Hiorns, and TP Powell. *The basic uniformity in structure of the neocortex.* Brain, Jun 1980.
`http://www.ncbi.nlm.nih.gov/pubmed/6772266`.

[7] Numenta and Grok. Grok, 2013.
`https://www.groksolutions.com/`.

[8] Numenta. Nupic open source project and community established at numenta.org, jul 2013.
`http://www.groksolutions.com/prs/pr-07-22-13.html`.

[9] Matt Taylor aka rhyolight. self-proclaimed open-source community flag-bearer.
`https://github.com/rhyolight`.

[10] NuPIC. Nupic mailing list. and archives,
`http://lists.numenta.org/mailman/listinfo/nupic_lists.numenta.org`.

[11] NuPIC. Nupic online api documentation, 2013.
`http://numenta.org/docs/`.

[12] Numenta. Sprint plannings, talks and conferences on numenta google+.
`https://plus.google.com/+NumentaOrg/posts`.

[13] NuPIC. Nupic first hackathon, jun 2013.
`http://numenta.org/blog/2013/06/25/hackathon-outcome.html`.

[14] NuPIC. Nupic nlp hackathon, nov 2013.
`http://numenta.org/blog/2013/11/06/2013-fall-hackathon-outcome.html`.

[15] Keithcom. Skier game controlled by cla, 2013.
`https://github.com/keithcom/nta_ski`.

[16] Chetan Surpur. Linguist, a story-teller, text generating cla.
`https://github.com/chetan51/linguist`.

[17] Jeff Hinton. Lecture on text-generation using deep nns.
`https://class.coursera.org/neuralnets-2012-001/lecture/91`.

[18] CEPT. Nlp technology.
`http://www.cept.at/index.html`.

[19] M. Otahal. Virtual machine with nupic and tools preinstalled, 2013.
`https://mega.co.nz/#!fQomTTqI!ffY_QBQQH5oGagylqpgScTzKPHF8Ef7yLFo3LboHzuk`.

[20] ROS. Robot operating system website, 2013.
`http://www.ros.org/`.

[21] M. Thaylor and P. Scott. Nupic at oscon, 2013.
`https://www.youtube.com/watch?v=5r1vZ1ymrQE`.

[22] J. Hawkins. Intelligence and the brain: Recent advances in understanding how the brain works with jeff, 2012.
`https://www.youtube.com/watch?v=qZM9JREjnp4`.

[23] Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies, 2001.

[24] M. EBDON. Is the cerebral neocortex a uniform cognitive architecture?, 1993.
`http://onlinelibrary.wiley.com/doi/10.1111/j.1468-0017.1993.tb00291.x/abstract`.

[25] James A. Reggia, C. Lynne D'Autrechy, III Granger G. Sutton, and Michael Weinrich. A competitive distribution theory of neocortical dynamics.

[26] D. Kadlecek. *MOTIVATION DRIVEN REINFORCEMENT LEARNING AND AUTOMATIC CREATION OF BEHAVIOR HIERARCHIES*. CTU, 2008. Doctoral Thesis.

[27] P. Nahodil and J. Vítků. *Novel Theory and Simulations of Anticipatory Behaviour in Artificial Life Domain, in Advances in Intelligent Modelling and Simulation*. Springer, 2012.

[28] V. Svatos. *Návrh a aplikace hybridního řídícího systému mobilních robotů na bázi Umělého života*. CTU, 2007. Doctoral Thesis, in Czech language.

[29] E. A. Blechman. *Moods, Affect, and Emotions*. Lawrence Erlbaum Associates: Hillsdale, NJ, 1990.

[30] A. S. Rao and M. P. Georgeff. *Modeling Rational Agents within a BDI-Architecture*. In Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning, 1991.

[31] M. Otahal. Agents with cla, 2013. repository
`https://github.com/breznak/ALife`.

[32] *Cognitive psychology and its implications (6th ed.)*. Worth Publishers., 2004.

[33] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

[34] Intelligent agent.
`https://en.wikipedia.org/wiki/Intelligent_agent`.

[35] M. Otahal. Ros support to nupic, 2013. branch
`https://github.com/breznak/nupic/tree/ROS-encoder`.

Appendix A
Acknowledgment

I would like to express my special gratitude to the following members, institutions and authors of tools which we have found helpful during the project.

- *CTUstyle* a very elegant template for thesis at CTU created by Petr Olšák[1]) and which has been used to typeset this document. Thank you very much Mr. Olšák, and especially for the additional help with the template.
- *Metacentrum* an organization[2]) providing access to high-performance clusters to universities for research purposes. Your project is really great and allows us to experiment with datasets and algorithms which would otherwise be completely out of our scope. Thank you.

[1]) `http://petr.olsak.net/ctustyle.html`
[2]) `http://metavo.metacentrum.cz/`

Appendix B
List of Abbreviations and Names

B.1 Abbreviations:

BCI	Brain-Computer Interface
CLA	Cortical Learning Algorithm
CTU	also ČVUT Czech Technical University in Prague, my University
HTM	Hierarchical Temporal Memory
IDE	Integrated Development Environment
LTST	Long-Term Short-Term memory model
ML	Mailing list
NN	Neural network
OSS	open-source (software)
PSO	particle swarm optimization
RNN	recurrent neural network
ROS	Robot Operating System[1])
RL	Reinforcement Learning
SDR	Sparse Distributed Representation
SP	spatial pooler, core part of CLA
TP	temporal pooler
VM	virtual machine

B.2 Names:

ALife	also A-life, discipline studying agents and artificial life
NuPIC	a development platform founded by Numenta, now an open-source project, implementation of HTM/CLA theory
Whitepaper	or CLA Whitepaper, refers to HTM/CLA theory pseadocode and documentation published in [5]
Numenta	the company behind NuPIC development, [2]
Grok	commercial product developed by [2, 7] (it uses same HMT/CLA algorithm as the one available in NuPIC)
CEPT	a company and technology to represent words in a visualize-able and computational friendly way (as a feature vector, ontology) [18]
Cerebro	visualization and debugging extension tool for NuPIC
Swarming	tool used for parameter tuning of the CLAs

[1]) "The Robot Operating System (ROS) is a set of software libraries and tools that help you build robot applications. From drivers to state-of-the-art algorithms, and with powerful developer tools, ROS has what you need for your next robotics project. And it's all open source." [20]

B.3 Other software

Python	Python language interpreter
GCC	GNU C Compiler Collection (also C++)
Netbeans	Java IDE with optional python and C++ support
ROS	ROS platform
VirtualBox	a virtualization solution
CTUstyle	TeX template used to typeset this thesis
Linux	OS supported by NuPIC, optionaly also OSX
pip	python module installer

Appendix C Electronic content

The enclosed DVD contains the following files and directories:

- *DP-CLA-agent-otahama2.pdf* this thesis in an pdf document.
- *tex/* folder contains the source files (.TeX) and images to generate the pdf above, the CTUstyle template is included
- *NuPIC.ova*: NuPIC development platform on Ubuntu 13.04 64bit [19]
    - (pre)installed NuPIC (core HTM CLA)
    - Cerebro (CLA visualisation and debugging)
    - swarming (parameter tuning for Nupic)
    - Netbeans IDE with added C++/Python support
    - all dependencies for development (python 2.6,2.7, gcc 4.7, git)
    - XChat IRC (set to #nupic channel)
    - cppcheck
    - ROS (robotic OS)
    - username/password: nupic/nupic

  all tested and working out-of-the box;

  As a user, you can tune these settings:

    - change you name & email in git (for commits under your name)
    - change nick on IRC (so we're not all 'nupic' ;))
    - in VM settings, you can allocate more RAM/CPUs (but this is a reasonable default)
    - in VM settings, set up shared folders

  When you boot the VM image the following repositories are readily available:
- *repositories*:
    - *nupic-source/* a clone of the official NuPIC git repository, also contains branches *ROS* and *utility-encoder* pulled from my repo. These are required for some of the experiments mentioned in this thesis and have not yet been merged to upstream.
    - *ALife/* clone of my repository for experiments with agents, behavior and emotions. This code depends on the nupic repo and some of the experimental branches.

Usage: the NuPIC.ova file is a virtual machine (VM) image that can be conveniently imported to any standard virtualization program [1]), it will create a new instance of Ubuntu Linux with the preinstalled software and users can just hit "Play" to run the VM.

[1]) for example Virtualbox, VMWarePlayer, kvm - these programs are available for all the main platforms - Win, Linux, OSX