23rd International Joint Conference on Artificial Intelligence IJCAI-13

Proceedings of the 9th International Workshop on Neural-Symbolic Learning and Reasoning NeSy13

Artur S. d'Avila Garcez, Pascal Hitzler and Luis C. Lamb
(eds.)

Beijing, 5 August 2013

Preface

The workshop series on Neural-Symbolic Learning and Reasoning (NeSy) started in Edinburgh, Scotland, in 2005 at IJCAI-05, and was followed by NeSy workshops at ECAI-06, IJCAI-07, ECAI-08, IJCAI-09, AAAI-10, IJCAI-11, AAAI-12, and now IJCAI-13. NeSy is an opportunity for researchers in the areas of artificial intelligence (AI), neural computation and cognitive science to get together and share experience and practices. The workshop series promotes the interplay between machine learning and knowledge representation and reasoning through the integration of neural computation and symbolic AI.

Intelligent systems are required to learn from data in order to adapt to new situations, and to represent and reason about what has been learned in order to control the accumulation of errors and take action. Towards this goal, neural-symbolic integration combines the statistical nature of learning with the logical nature of reasoning. Over the years, researchers have built sound neural-symbolic models and systems that are capable of learning several forms of reasoning, including temporal, modal, epistemic, non-monotonic, fuzzy, relational and probabilistic reasoning. Such systems seek to combine well-founded symbolic AI representations with parallel and distributed neural computation and learning to achieve efficiency, robustness and interpretability. Neural-symbolic systems have been applied in robotics, simulation, fraud prevention, semantic web, software verification and adaptation, fault diagnosis, bioinformatics, business process mining, visual information processing, music informatics and language processing.

Neural-symbolic computation offers a methodology for integrating reasoning and learning in intelligent systems, and a rich model for cognitive computation through (i) the representation of symbolic knowledge by network models, (ii) the parallel computation and adaptation of such models given numerical data, and (iii) the extraction of new knowledge and explanations from trained networks. These features allow the integrated study of symbolic and sub-symbolic AI. Further, neural-symbolic computation seeks to address important questions in cognitive science, related to the nature of reasoning, knowledge representation, and concept learning, following the computational theory of mind.

The NeSy workshop series provides a forum for the presentation and discussion of the key topics related to neural-symbolic integration, including:

1. Representation of symbolic knowledge by connectionist systems;
2. New neural-symbolic learning approaches;

3. Extraction of symbolic knowledge from trained neural networks;
4. New neural-symbolic reasoning approaches;
5. Biologically-inspired neural-symbolic models and systems;
6. Knowledge-based networks, support vector machines and deep networks;
7. Structured learning and relational learning in connectionist systems;
8. Neural-symbolic cognitive agents and applications of neural-symbolic systems.

NeSy13 has a mix of high-quality contributed papers and invited talks, all with associated research papers published in this proceedings. The papers span from the foundations of neural-symbolic computing, learning from structured data, and cognitive reasoning architectures to applications in virtual reality, relational learning, and software verification. This proceedings also includes Peter Norvig's text *On Chomsky and the Two Cultures of Statistical Learning*, used as a basis for the workshop's discussion session on the reach of Machine Learning and its inter-relationship with Knowledge Representation. We would like to thank Peter Norvig for giving us permission to publish his text in this proceedings. We would also like to thank the invited speakers, Ron Sun, Kai-Uwe Kuehnberger, Alessandro Sperduti and Danny Silver, and all the NeSy participants for making the workshop a success. We are grateful to the members of the NeSy13 Programme Committee and would like to acknowledge their key contribution to the reviewing process:

Howard Bowman, University of Kent, England
Claudia d'Amato, University of Bari, Italy
Marco Gori, University of Siena, Italy
Barbara Hammer, TU Clausthal, Germany
Steffen Hoelldobler, TU Dresden, Germany
Ekaterina Komendantskaya, University of Dundee, Scotland
Kai-Uwe Kuehnberger, University of Osnabrck, Germany
Gadi Pinkas, Center for Academic Studies, Israel
Florian Roehrbein, Albert Einstein College of Medicine, New York, U.S.A.
Rudy Setiono, National University of Singapore
Jude Shavlik, University of Wisconsin-Madison, U.S.A.
Gerson Zaverucha, Universidade Federal do Rio de Janeiro, Brazil

The workshop papers and the slides from the talks are also available online at http://www.neural-symbolic.org/NeSy13/. The 10th anniversary of the NeSy workshop series will be commemorated with a Dagstuhl Seminar from 14 to 19 September 2014. For more information, please visit http://www.dagstuhl.de/14381.

London, 1 July 2013
Artur Garcez, Luis Lamb and Pascal Hitzler[1]

[1]Artur S. d'Avila Garcez is Reader in Neural-Symbolic Computing at City University London, UK; Pascal Hitzler is Associate Professor of Computer Science at Wright State University, Dayton, OH.; Luis C. Lamb is Professor of Computer Science at UFRGS, Porto Alegre, Brazil.

Contents

Equipping Symbolic Frameworks with Soft Computing Features

Kai-Uwe Kühnberger
Institute of Cognitive Science, University of Osnabrück
kkuehnbe@uos.de

Abstract

This paper proposes to have a fresh look on the neural-symbolic distinction by focusing on the strengths and weaknesses of the two antagonistic approaches. We claim that in both worlds, the symbolic and the subsymbolic world, there is a tendency to embrace new methods borrowed from the respective other methodology. Whereas, this seems to be quite obvious from the neural perspective we focus on sketching ideas where soft computing methods are used in classical symbolic, logic-based frameworks. We exemplify this claim by some remarks concerning certain soft computing features of Heuristic-Driven Theory Projection (HDTP), a symbolic framework for analogy-making and concept blending.

1 Introduction

From a certain practical perspective the long lasting distinction between symbolic and subsymbolic approaches is rather natural. Strengths and weaknesses of these approaches are quite complementarily distributed as shown in Table 1. For example, whereas symbolic approaches have their strengths mostly in higher cognitive abilities, such as planning, reasoning, and the ability to represent knowledge explicitly, subsymbolic approaches are very successful in areas related to lower cognitive abilities, like data-driven approaches for modeling perception, learning, and adaptation to a rapidly changing environment. Therefore, it is a natural idea to develop frameworks that combine the (complementary) strengths of both approaches such that at the same time the (complementary) weaknesses of the two types of approaches are avoided.

Although, many frameworks for reconciling these two major types of models have been proposed,[1] the last decades of research in this direction has shown that the development of such integrated frameworks that show their strengths in a broad variety of classical applications is everything else than easy to achieve. To phrase this more frankly: Taking the last two decades of neural-symbolic integration into account, it is rather obvious that neither the ultimate application nor the ultimate theoretical insight has been proposed. Various frameworks have been studied theoretically and practically, strengths and weaknesses of the different accounts have been evaluated. Nevertheless, there is no agreement whether the proposed frameworks can do better than alternative (classical) approaches that are not based on the spirit of neural-symbolic reasoning and learning. Even worse many researchers would even say that neural-symbolic reasoning and learning devices have failed to demonstrate their broad applicability in theory and practice.

This overall negative claim is surprising because there are many proposals on the market. Just to mention some of these approaches towards neural-symbolic integration, the frameworks described in [Hitzler *et al.*, 2004], [Garcez *et al.*, 2002], and [Gust *et al.*, 2007] try to integrate complex higher reasoning abilities into a neural network inspired approach. Although, there are theoretically highly interesting results available, practical applications are rather rare. Besides the mentioned class of models, there is furthermore a large number of hybrid models in the field of cognitive architectures that work in similar directions towards a neural understanding of higher cognitive abilities with neurally inspired means. A good overview of hybrid systems can be found for example in [Wermter and Sun, 2000]. Unfortunately, hybrid cognitive architectures have often similar acceptance problems in parts of the scientific community as cognitive architectures in general are often confronted with: many classical researchers do not take them seriously into account, because specialized engineering solutions are just more successful in most practical applications.

We think that the situation in neural-symbolic reasoning and learning is unsatisfactory because of the described lack of theoretical and practical breakthroughs. It may be a good strategy to take a more abstract perspective on the interplay between subsymbolic and symbolic frameworks into account. In particular, it may be reasonable to focus on current tendencies of symbolic approaches to model learning and adaptation aspects one the one side, and the tendencies of subsymbolic approaches to represent complex data structures and higher cognitive processes on the other. This more abstract level of neural-symbolic integration, namely on the level of

[1] The collection [Hammer and Hitzler, 2007] gives a good overview concerning different frameworks and different methodologies in order to bridge the gap between subsymbolic and symbolic approaches.

	Symbolic Approaches	Sub–Symbolic Approaches
Methods	(Mostly) logical and/or algebraic	(Mostly) analytic
Strengths	Productivity, Recursion Principle, Compositionality	Robustness, Learning Ability, Parsimony, Adaptivity
Weaknesses	Consistency Constraints, Lower Cognitive Abilities	Opaqueness Higher Cognitive Abilities
Applications	Reasoning, Problem Solving, Planning etc.	Learning, Motor Control, Vision etc.
CogSci Relation	Not Biologically Inspired	Biologically Inspired
Other Features	Crisp	Fuzzy

Table 1: Strengths and weaknesses of symbolic and neural/subsymbolic approaches. The distributions of these strengths and weaknesses are quite complementary.

strengths and weaknesses of the respective accounts, comes together with an integration of new methods that are not considered as standard techniques in the symbolic and subsymbolic worlds. Nevertheless, the big difference to general approaches to bridge the gap between symbolic and subsymbolic models is the local character of the integration. Adding a certain feature of the subsymbolic world into symbolic frameworks (or vice versa) is not solving the general problem of the symbolic-subsymbolic distinction. The hope is that a large number of such integrations facilitates a general theory of neural-symbolic reasoning and learning.

This paper attempts to shed light on the underlying problem from a rather general perspective. First, we will describe (from a subjective perspective) current tendencies that symbolic and subsymbolic frameworks converge against each other (Section 2). By using the term relatively loosely, this convergence can be subsumed under the endeavor to develop neural-symbolic integration devices. In Section 3, we will describe informally certain soft computing features of the symbolic framework Heuristic-Driven Theory Projection (HDTP), i.e. we give an example which shows that certain symbolic approaches integrate features of the subsymbolic world quite obviously into their internal procedures. Last but not least, Section 4 adds some speculations of how the totality of these endeavors could eventually bring fresh ideas into the neural-symbolic integration research field and concludes the paper.

2 Convergence Tendencies of the Neural and Symbolic Worlds

We claim that there are certain convergence tendencies between symbolic and subsymbolic frameworks that attempt to model strengths of the respective alternative approach. In other words, there is the tendency of developing neural models that expand their applicability to higher cognitive abilities. On the other hand, there are also tendencies in the development of symbolic frameworks to model strengths of neurally inspired or soft computing approaches like learning abilities, adaptivity, and robustness with respect to noisy data. Whereas, for the neural approaches this seems to be quite obvious – e.g. the proceedings of the International Workshops for Neural-Symbolic Reasoning and Learning are a good resource for these developments[2] – this is not as clear for models that are based primarily on symbolic methodologies. Therefore, we will list informally some of these developments where symbolic approaches are intended to expand to the neural world.

- **Relational Learning:** The equipment of the framework of inductive logic programming with probabilistic features can be seen as an example of extending a classical logical (and therefore symbolic) learning approach with subsymbolic (or soft computing) methods. A standard reference for relational learning can be found in [De Raedt, 2008].
- **Ontology Repair System:** The computational modeling of scientific discovery requires relatively expressive and dynamically changing formalisms for computational approaches. An algorithmic approach for finding new insights in science, for inventing new scientific concepts, for re-interpreting old concepts in scientific theories newly and the like require a flexible formalism that allows not only extensions of a theory, but moreover changes of the underlying languages. A guiding idea of these approaches is to resolve clashes in existing scientific theories by changing domain theories and their languages in a non-trivial way. Compare [Bundy and Chan, 2008] and [Chan *et al.*, 2010] for more information concerning automated approaches towards ontology evolution in physics.
- **Dynamics of Analogy Making:** It is quite uncontroversial that a fundamental mechanism for cognition is analogy making [Hofstadter and the Fluid Analogies Research Group, 1995]. In particular, if cross-domain analogies are considered, then computational models for analogical reasoning require dynamic changes of the signatures of languages (provided we restrict our attention to symbolic approaches). Furthermore, such models compute ranked candidates for analogical relations, i.e. they add a classical soft computing feature of more or less plausible candidates for an analogy to the framework.
- **Ontologies in Text Understanding Systems:** In the context of language understanding systems there is the need to integrate not only linguistic knowledge into the systems's knowledge base, but also world knowledge, often represented in form of domain ontologies. Usually these ontologies are considered to be crisp. Nevertheless, newer approaches towards natural language understanding systems attempt to integrate soft computing elements into the symbolic representations of the integrated ontologies. A good examples for such a general multi-purpose system can be found in [Ovchinnikova, 2012].

These are just a few examples among many possible candidates showing that there are attempts to equip symbolic systems with soft computing features that allow adaptation, dynamic changes, conflict resolution, learning, and the invention of new concepts. Although most of these systems do not

[2] Cf. http://www.neural-symbolic.org/ for further reference.

implement a classical neural approach (in terms of a network of neurons, activation potentials, neurally inspired learning mechanisms and the like), these frameworks tend to integrate soft computing features from the subsymbolic world into a symbolic, logic-based framework in order to extend the range of applicability. Additionally, such approaches are often cognitively inspired and often based (at least to a certain extent) on insights from cognitive science.

3 Adaptation from a Symbolic Perspective: An Example

3.1 Heuristic-Driven Theory Projection

In this section, we give informally an example of a symbolic system that allows dynamic adaptations of representations and the learning of new concepts (and theories for these concepts) that are potentially formulated in different languages. Heuristic-Driven Theory Projection (HDTP) is an expressive formalism for the computation of candidates of analogical relations between two given first-order theories (source domain and target domain) [Gust *et al.*, 2006]. Additionally to the computation of an analogical relation the system computes a generalization of the input theories, substitutions in order to recover subtheories of the input theories from the generalization, and the re-representation of input theories (if necessary) [Schwering *et al.*, 2009]. Recently, the framework has been extended to cover also other cognitive mechanisms like concept blending, a mechanism that is important for creativity aspects of cognition [Martinez *et al.*, 2012].

HDTP's computation of an analogical relation identifies the common (structural) aspects of source and target, and exports some of these aspects from the source domain to the target domain. The generalized domain identifies common aspects of the input domains and makes these explicit, i.e. the generalized domain captures the common parts of the input domains. Figure 1 depicts this overall idea using S and T as source and target domains, respectively, and the generalization, G, represents the common parts of S and T. The *analogical transfer* not only associates S and T with each other, but also projects knowledge from S to T, resulting in an enrichment of structure in T.

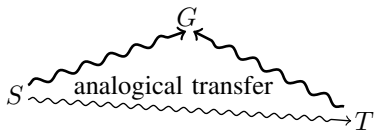


Figure 1: HDTP's overall approach to creating analogies.

The inverse operation of the generalization, namely to reconstruct the input domains from the generalization can be modeled by substitutions and therefore result in specializations of the generalization. Obviously, there are many possibilities to associate entities of two domains resulting in different candidates of analogies. Furthermore, analogies are not correct or incorrect, but more or less psychologically plausible. HDTP takes these features of analogy-making into account by the computation of candidates of analogical relations together with a ranking of these candidates.

Analogical transfer often results in structure enrichment of the target side (via the analogical transfer), which usually corresponds to the addition of new axioms to the target theory, but may also involve the addition of new or the deletion of old first-order symbols. Taking into account the whole process of analogy making, even worse, operations like the replacement of a first-order symbol of arity n by a new first-order symbol of arit m, or a combination of the mentioned operations can occur. Dynamic changes of the underlying signature of a theory are usually not considered in classical logic frameworks, because it is often hard to find a semantics for such dynamical mappings (provided we are not restricting our considerations to well-known conservative expansions or reducts in the model theoretic sense).[3]

Besides the computation of an analogical relation there are application cases in which two conceptual spaces (in our case the input theories for the source and target domains) need not only to be (partially) mapped onto each other, but partially merged in order to create a new conceptual space. In such cases, HDTP uses the computed generalization, the given source and target theories, and the analogical relation between source and target to compute a new conceptual space which is called a blend space [Goguen, 2006].

3.2 Institutions

Analogy making and concept blending as considered in HDTP can be seen as a theory integrating soft computing features into a symbolic framework. Here are some examples of these soft computing features:

- Identification of cross-domain properties and relations that cannot be associated in classical frameworks.
- Adaptation of the underlying input theories (re-representation based on logical deductions) if this is necessary for the computation of better analogies.
- Dynamic transfer of knowledge from the source to the target domain.
- Ranking of candidates by a cost function or an appropriate probability measure.
- Mapping dynamically signatures of underlying domain theories onto each other.

In particular the last issue, namely the dynamic mapping of signatures of logical theories for the analogy-making process is a difficult operation in logical systems. HDTP uses the theory of institutions [Diaconescu, 2008] to specify a semantics for this operation. Figure 2 shows the basic idea of an "institution" in a diagrammatic representation. The theory of institutions allows to represent the elementary dynamics in logical systems if a theory formalized in a language L is mapped to a theory formalized in another language L'. Furthermore, institution theory is an abstract formalism that allows to represent both syntax and semantics in the described dynamic

[3] Compare [Chang and Keisler, 1973] for more information concerning conservative expansions of languages and models of logical theories.

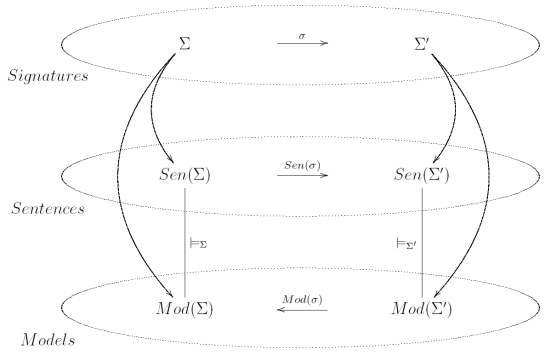


Figure 2: A graphical representation covering the basic idea of an institution.

change from one language to another language. On the signature level a morphism $\sigma : \Sigma \to \Sigma'$ maps a signature Σ to another signature Σ'. Via a functor mapping objects and morphisms in *Signature* to objects and morphisms in *Sentences*, it is possible to induce a morphism $Sen(\sigma)$ from sentences $Sen(\Sigma)$ formulated in Σ to sentences $Sen(\Sigma')$ formulated in Σ'. Similarly, it is possible to induce (via another functor) a morphism $Mod(\sigma)$ from classes of models for $Sen(\Sigma')$ to classes of models of $Sen(\Sigma)$. The important property is the contravariant relation between the morphism between model classes and the morphism between sets of sentences. More precisely, for every $\sigma : \Sigma \to \Sigma'$ every model $\mathfrak{m}' \in Mod(\Sigma'))$ and every $\rho \in Sen(\Sigma)$ it holds:

$$\mathfrak{m}' \models_{\Sigma'} Sen(\rho)(\sigma) \iff Mod(\sigma)(\mathfrak{m}') \models_{\Sigma} \rho$$

A simple example of an institution can be given in well-known terms of first-order logic (FOL). In this case, the Σ-sentences $\mathbf{Sen}(\Sigma)$ corresponds to the set of all FOL formulas that can be built using symbols from a signature Σ. For each signature Σ the collection $\mathbf{Mod}(\Sigma)$ of all Σ-models corresponds in FOL to the collection of all possible interpretations of symbols from Σ. The Σ-models and Σ-sentences are related by the relation of Σ-satisfaction, which corresponds to the classical model theoretic satisfaction relation in FOL. Institutions theory provides therefore a framework to consider the syntax and the semantics of FOL in one framework.[4]

3.3 Modeling Analogies in Institutions

The described dynamics that can be represented in institution theory is triggered by signature morphisms. In very simple situations, such mappings may suffice to model the analogical relation between a given source and a target domain in an analogical relation. Nevertheless, the general case requires a more general concept of signature morphism, because substitutions specializing terms of the generalized theory may be complex. Such cases are not covered by simple morphism in the category of signatures. Fortunately, institution theory provides such a generalized concept, namely the concept of a generalized Σ.

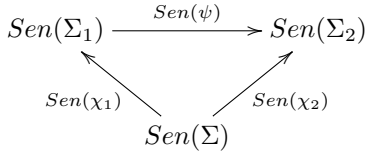
Definition 1. *For any signature Σ of an institution, and any signature morphisms $\chi_1 : \Sigma \to \Sigma_1$ and $\chi_2 : \Sigma \to \Sigma_2$, a* general Σ-substitution $\psi_{\chi_1:\chi_2}$ *consists of a pair*

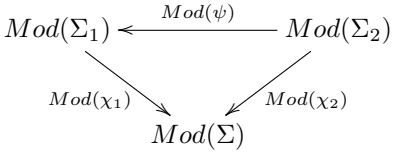
$$\langle Sen(\psi), Mod(\psi) \rangle ,$$

where

- $Sen(\psi) : Sen(\Sigma_1) \to Sen(\Sigma_2)$ *is a function*
- $Mod(\psi) : Mod(\Sigma_2) \to Mod(\Sigma_1)$ *is a functor*

such that both of them preserve Σ, i.e. the following diagrams commute:





and such that the following satisfaction condition holds:

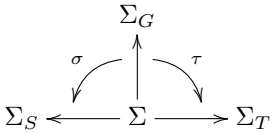
$$Mod(\psi)(\mathfrak{m}_2) \models \rho_1 \quad \textit{if and only if} \quad \mathfrak{m}_2 \models Sen(\psi)(\rho_1)$$

for each Σ_2-model $\mathfrak{m}_2$ and each Σ_1-sentence ρ_1.

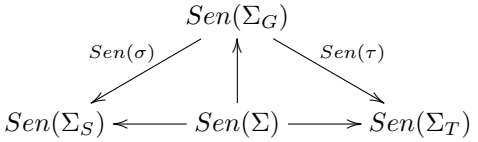
General Σ-substitutions extend the idea of a signature morphism. Although, in general there needs to be no mapping on the level of signatures between Σ_1 und Σ_2, most general Σ-substitution considered in practice are induced by some form of signature mapping. Every signature morphism can be seen as general Σ-substitution, and many other mappings, like classical first-order substitutions, second-order substitutions (for FOL), and derived signature morphisms give rise to a general Σ-substitution.

Generalized $\Sigma-$substitutions allow to define formally the concept of an analogy in the sense of HDTP.

Definition 2. *Given two signatures Σ_S and Σ_T over a common signature Σ, an* analogy $\aleph$ *is defined to be a triple $\langle \Sigma_G, \sigma, \tau \rangle$, consisting of a signature Σ_G, and general Σ substitutions σ and τ as indicated in the following diagram:*

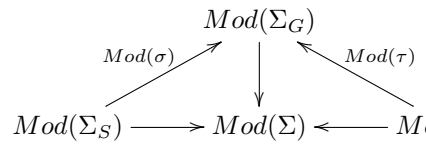


As a Σ-substitution is defined as a pair of mappings on sentence and model level, every analogy gives rise to the following diagrams:



[4] An alternative way to represent the contrainvariant interplay between syntax and semantics of logical frameworks is channel theory [Barwise and Seligman, 1997].

and



Furthermore, for every Σ_G-sentence ρ, every Σ_S-model $\mathfrak{m}_S$ and every Σ_T-model $\mathfrak{m}_T$, the following satisfiability conditions hold:

$$\begin{array}{ccc} \mathbf{Sen}(\sigma)(\rho) & \longleftarrow & \rho \\ {\scriptstyle \Sigma_S}\downarrow \models & \text{iff} & {\scriptstyle \Sigma_G}\downarrow \models \\ \mathfrak{m}_S & \rightarrowtail & \mathbf{Mod}(\sigma)(\mathfrak{m}_S) \end{array} \quad \text{and} \quad \begin{array}{ccc} \rho & \longrightarrow & \mathbf{Sen}(\tau)(\rho) \\ {\scriptstyle \Sigma_G}\downarrow \models & & {\scriptstyle \Sigma_T}\downarrow \models \\ \mathbf{Mod}(\tau)(\mathfrak{m}_T) & \twoheadleftarrow & \mathfrak{m}_T \end{array}$$

In this setting, we can introduce an analogical relation on the level of sentences as well as on the level of models in the intended sense of the computational model. To be more precise, two formulas $s \in \mathbf{Sen}(\Sigma_S)$ and $t \in \mathbf{Sen}(\Sigma_T)$ are in an analogical relation if and only if there is a formula $g \in \mathbf{Sen}(\Sigma_G)$ of the generalized theory such that s and t can be computed from g by applying the respective generalized $\Sigma-$Substitutions. A similar relations holds with respect to the model classes, hence the theory of institutions allows to model rather nicely on the syntactic and semantic level the mappings of different signatures to each other.

4 Conclusions

Neural-symbolic reasoning and learning can be approached from different directions. One direction is to directly attempt a modeling of higher cognitive abilities (like reasoning) with neural or neurally inspired means. Another possibility may be to equip symbolic frameworks with soft computing features. The claim has been made in this paper that currently several researchers follow this second line of research. We exemplified this claim by specifying some soft computing properties of HDTP, a symbolic framework for analogy-making and concept blending.

The convergence of the two worlds as described in Section 2 does not seem to follow a blueprint. It seems to result from needs of the practical application of a certain computational framework to a specific problem domain. For example, analogy-making considered as a cognitive ability, requires in its modeling a non-crisp approach, the possibility to associate aspects of theories locally, and to relate domain theories formulated in potentially different languages to each other. Such features were sketched in Section 3 resulting in an equipment of a symbolic framework with soft computing features. Here is a second example: Ontology repair systems, as mentioned in Section 2, are triggered from a cognitive perspective, simply because the system is intended to model what scientists are doing if they modify or adapt an existing theory to the needs of underlying constraints. Hence, the usage of soft computing features in such frameworks are natural extensions motivated and inspired by the needs of the application. Insofar, it is not surprising that such systems integrate only certain (seemingly isolated) aspects of the symbolic and subsymbolic world into each other, rather than trying to solve the problem in its full generality, i.e. to attempt to integrate all aspects of the two worlds into each other.

It is rather likely that the large number of approaches for integrating locally soft computing features into existing symbolic frameworks will eventually result in a better understanding of frameworks covering aspects of both worlds. We think that the various approaches for a convergence of the symbolic and subsymbolic world should enable researchers to find a broad reservoir of new and inspiring examples such that the next steps for progression with respect to the important question of neural-symbolic reasoning and learning can be anticipated.

References

[Barwise and Seligman, 1997] Jon Barwise and Jerry Seligman. *Information Flow: The Logic of Distributed Systems*, volume 44 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1997.

[Bundy and Chan, 2008] Alan Bundy and Michael Chan. Towards ontology evolution in physics. In *WoLLIC*, pages 98–110, 2008.

[Chan *et al.*, 2010] Michael Chan, Jos Lehmann, and Alan Bundy. Higher-order representation and reasoning for automated ontology evolution. In *KEOD*, pages 84–93, 2010.

[Chang and Keisler, 1973] C. C. Chang and H. J. Keisler. *Model Theory*, volume 73 of *Studies in Logic*. North-Holland Publishing Company, 1973.

[De Raedt, 2008] Luc De Raedt. *Logical and Relational Learning*. Cognitive Technologies. Springer, 2008.

[Diaconescu, 2008] Răzvan Diaconescu. *Institution-independent Model Theory*. Studies in Universal Logic. Birkhäuser, Basel, 2008.

[Garcez *et al.*, 2002] Artur D'Avila Garcez, Krysia Broda, and Dov Gabbay. *Neural-Symbolic Learning Systems: Foundations and Applications*. Springer, 2002.

[Goguen, 2006] Joseph Goguen. Mathematical models of cognitive space and time. In D. Andler, Y. Ogawa, M. Okada, and S. Watanabe, editors, *Reasoning and Cognition: Proc. of the Interdisciplinary Conference on Reasoning and Cognition*, pages 125–128. Keio University Press, 2006.

[Gust *et al.*, 2006] Helmar Gust, Kai-Uwe Kühnberger, and Ute Schmid. Metaphors and heuristic-driven theory projection (hdtp). *Theoretical Computer Science*, 354:98–117, 2006.

[Gust *et al.*, 2007] Helmar Gust, Kai-Uwe Kühnberger, and Peter Geibel. *Perspectives on Neural-Symbolic Integration*, volume 77 of *Studies in Computational Intelligence*, chapter Learning Models of Predicate Logical Theories with Neural Networks Based on Topos Theory, pages 233–264. Springer, 2007.

[Hammer and Hitzler, 2007] Barbara Hammer and Pascal Hitzler, editors. *Perspectives of Neural-Symbolic Integration*, volume 77 of *Studies in Computational Intelligence*. Springer, 2007.

[Hitzler *et al.*, 2004] Pascal Hitzler, Steffen Hölldobler, and Anthony K. Seda. Logic programs and connectionist networks. *Journal of Applied Logic*, 2(3):245–272, 2004.

[Hofstadter and the Fluid Analogies Research Group, 1995] Douglas Hofstadter and the Fluid Analogies Research Group. *Fluid Concepts and Creative Analogies. Computer Models of the Fundamental Mechanisms of Thought*. Basic Books, New York, 1995.

[Martinez *et al.*, 2012] Maricarmen Martinez, Tarek R. Besold, Ahmed Abdel-Fattah, Helmar Gust, Martin Schmidt, Ulf Krumnack, and Kai-Uwe Kühnberger. Theory blending as a framework for creativity in systems for general intelligence. In Pei Wang and Ben Goertzel, editors, *Theoretical Foundations of Artificial General Intelligence*, volume 4 of *Thinking Machines*, pages 219–239. Atlantis Press, Springer, Amsterdam, 2012.

[Ovchinnikova, 2012] Ekaterina Ovchinnikova. *Integration of World Knowledge for Natural Language Understanding*, volume 3 of *Thinking Machines*. Atlantis Press, Springer, Amsterdam, 2012.

[Schwering *et al.*, 2009] Angela Schwering, Ulf Krumnack, Kai-Uwe Kühnberger, and Helmar Gust. Syntactic principles of heuristic-driven theory projection. *Cognitive Systems Research*, 10(3):251–269, 2009.

[Wermter and Sun, 2000] Stefan Wermter and Ron Sun, editors. *Hybrid Neural Systems*. Springer, 2000.

Learning Multi-Sensory Integration with Self-Organization and Statistics

Johannes Bauer, Stefan Wermter
Department of Informatics
University of Hamburg
Germany
Email: {bauer,wermter}@informatik.uni-hamburg.de

Abstract

Recently, we have presented a self-organized artificial neural network algorithm capable of learning a latent variable model of its high-dimensional input and to optimally integrate that input to compute and population-code a probability density function over the values of the latent variables of that model. We did take our motivation from natural neural networks and reported on a simple experiment with simulated multi-sensory data. However, we focused on presenting the algorithm and evaluating its performance, leaving a comparison with natural cognition for future work. In this paper, we show that our algorithm behaves similar, in important behavioral and neural aspects, to a prime example of natural multi-sensory integration: audio-visual object localization.

1 Introduction

Imagine you are given a sheet of paper with unlabeled numbers, told that these numbers contain information about the value of some quantity, and asked what you think the value of that quantity is. This is clearly an impossible task. What if you are repeatedly shown such numbers, for different values of the quantity in question, but never given the right answer? Biological neurons face a similar situation. It is their function to produce activity corresponding to some quantity in the outside world. And all they have, to estimate that quantity, is the activity at their incoming synapses, which carries no information about its origin. A comparable situation also exists in unsupervised machine learning. Models of unsupervised neural learning have therefore found applications in general machine learning.

We have recently presented an artificial neural network (ANN) algorithm with possible applications in general machine learning based on the self-organizing map (SOM) [Bauer and Wermter, 2013]. This algorithm was inspired by the apparent ability of humans to utilize the sensory information they get in a statistically optimal fashion in many situations [Ernst and Banks, 2002; Landy *et al.*, 2011]. In particular, it aims to model how neural populations learn to make sense of uni- and cross-sensory stimuli. The result is an algorithm which takes high-dimensional data as input, learns a low-dimensional latent-variable model and computes for a given input a probability density function (PDF) over the values of the latent variables.

We have shown that our algorithm can perform near-optimally on uni-sensory input and we have shown that it can handle multiple sensory modalities with different response and noise characteristics. However, we have not fully closed the loop in 1) trying to understand a problem faced by a natural system, 2) trying to find a solution 3) comparing that solution to the one found, tried, and tested by nature [Jacobs and Kruschke, 2011; Landy *et al.*, 2011]. In this paper, we will therefore be concerned with the last step in this process, comparing our model's behavior to a biological example.

The superior colliculus (SC), a mid-brain region found in all vertebrates, is a prime candidate for this last step for a number of reasons: First, its deeper levels integrate information from vision, hearing, and touch to localize objects in space. Like our network, it thus uses high-dimensional input—the responses of uni-sensory input populations—to infer about a latent variable: the location of an object. Second, its physiology has been studied intensively and the relationship between input stimuli and overt behavior caused by SC activity is comparatively well-understood. Knowledge about and models of the SC can therefore serve as starting points for investigating other brain regions with similar tasks [Stein, 2012]. Third, as mentioned before, the SC is present in all vertebrates (being called optic tectum (OT) in non-mammals) and evolutionarily highly stable [Stein and Meredith, 1993]. The strategies it employs are therefore tried and tested indeed and likely to approach optimality.

We will compare our model's performance to two well-established principles of natural audio-visual object localization: On the neurophysiological level, we show that our network reproduces the effects of enhancement and depression depending on the spatial relation between auditory and visual stimuli found in single SC neurons [Stein and Meredith, 1993]. On the behavioral level, we demonstrate comparability with maximum likelihood estimation on the basis of uni-sensory localizations, shown for multi-sensory localization and several other cases of multi-sensory integration in humans [Alais and Burr, 2004; Ernst and Banks, 2002; Hillis *et al.*, 2004].

In the next section, we will first briefly review our algorithm, focussing on giving a good intuition of the main prin-

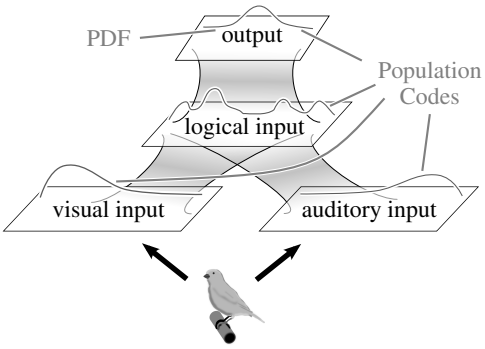


Figure 1: Visual and auditory input layers respond differently to multisensory input. They are concatenated into one logical input layer which is presented to our network, the output layer. Neurons in the output layer have no information about the origin of neurons in the logical input layer. The output layer learns to produce a population-coded PDF over the latent variable behind its input's activity.

ciples behind it, and referring to the original paper for details. We will then report on experiments we carried out in which we simulated various conditions of multi-sensory integration. We will review the known neurophysiological and psychophysical effects and compare them to our network's responses. Finally, we will interpret our results in the broader scope of computational neuroscience and its interaction with general artificial intelligence.

2 The Model

Before we start describing our solution, let us again look at the problem we are trying to solve. A population of neurons is to collaborate in learning to compute a PDF for the latent variables behind patterns of neural activity.[1] This input activity can be uni- or cross-sensory; conceptually, both can be treated the same by introducing a logical population which simply concatenates the separate input populations' activity vectors (see Fig. 1).

Our approach is to have the network learn to represent the PDF in a population code, where each neuron codes for the probability of a different value being the true value of the latent variable [Pouget *et al.*, 2003]. ANN models working with such population-coded PDFs have been proposed eg. by Cuijpers and Erlhagen [2008] and Beck *et al.* [2008]. Our model focuses on how computing such a PDF from arbitrary neural responses can be learned unsupervised and without heavy assumptions on the noise properties. The population code realized by our network is to be spatially organized, ie. close-by neurons code for similar values of the latent variable. This restriction on the more general definition of a population code seems natural and it also reflects biological evidence of topographic maps in various sensory brain areas [Stone, 2012; Hyde and Knudsen, 2000; Stein and Stanford, 2013; Kaas, 1997].

[1]From now on, we will assume wlog. a single latent variable. Note that strictly speaking a combination of latent variables can be seen as a single complex latent variable.

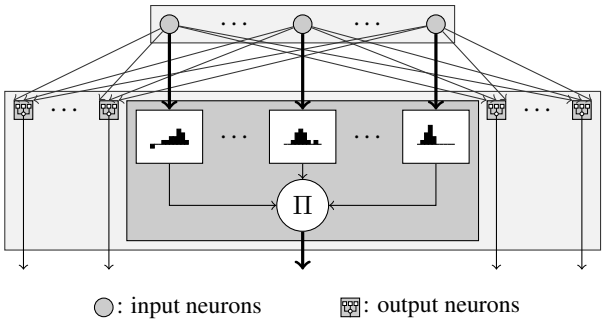


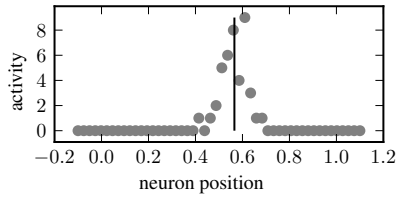
Figure 2: Structure of a single neuron within the output layer. Input neurons can be from any of the actual input populations, in no particular order. Each one is connected to all neurons in the output layer. Each output neuron learns 1) a preferred value of the latent variable, and 2) the statistics of its input given that preferred value of the latent variable. The network output is a PDF over the values of the latent variable.

Kohonen's SOM [Kohonen, 1982] was inspired by the formation of such topographic maps in the brain. It is a self-organizing ANN algorithm which has been shown to be able to learn latent-variable models [Yin, 2007; Klanke, 2007]: It learns topography-preserving mappings from points in a data space to its spatially ordered units, or neurons. Since a SOM models a population of neurons and each neuron has a response to a stimulus, it is possible to read out a population code from a SOM [Zhou *et al.*, 2011].

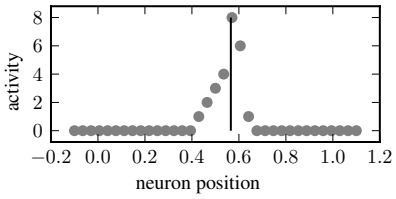
In a standard SOM, the response is just the Euclidean distance of the stimulus as a vector from the preferred stimulus of each neuron. This response is used to find the best-matching unit (BMU), the neuron with the strongest response and the neuron the stimulus is mapped to. In our algorithm, the network simultaneously learns the latent-variable model and the statistics (and noise properties) of the input (see Fig. 2). The response of each neuron then is an estimate of the probability of the neuron's preferred value being the actual value of the latent variable, given what the network knows about the noise. This is done basically by keeping weighted counts of previous activities at each input connection of each neuron. Input activities at the neurons' synapses are discrete and treated non-metrically. Therefore, the algorithm lends itself to learning problems where data points have nominal dimensions. The main benefit over previous approaches [Zhou *et al.*, 2011; Bauer *et al.*, 2012b] is that our approach does not assume any specific noise model. We refer to our original paper [Bauer and Wermter, 2013] for details on the learning algorithm and its motivation and derivation.

3 Experiments

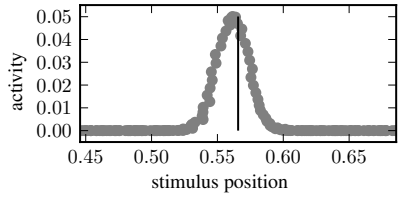
In the experiments described below, we will compare our network's performance and response properties to those found in psychophysical and neurophysiological studies. Specifically, we will examine the responses of our network in light of biological data about the SC. The SC is an evolutionarily stable midbrain structure concerned with localizing objects in space on the basis of visual, auditory, and somatosensory stimuli. It is involved in generating orienting movements on the basis



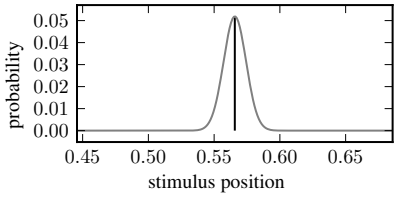
(a) Visual Input.



(b) Auditory Input.



(c) Network Output.



(d) PDF Recovered from Input.

Figure 3: Input, Network Response, and PDF Recovered From Input.

of uni- and cross-sensory stimuli [Stein and Stanford, 2013]. We choose it as a standard to which to compare our model because its input-output behavior is relatively well-understood and because it is likely that the general principles of SC functioning are realized similarly in other brain regions with comparable tasks [Stein, 2012].

For our simulations, we used a network of 500 output neurons connected to two populations $\mathbf{i}_{a,1}, \mathbf{i}_{a,2} \ldots, \mathbf{i}_{a,35}$ and $\mathbf{i}_{v,1}, \mathbf{i}_{v,2} \ldots, \mathbf{i}_{v,50}$ of input neurons. The two input populations each represented one sensory modality. Each input neuron $\mathbf{i}_{m,k}$ had a preferred value of $\mathfrak{p}_{m,k}$ such that the $\mathfrak{p}_{m,k}$ were evenly distributed over the interval $[-0.125, 1.125]$, for $m \in \{a, v\}$. The neuron $\mathbf{i}_{m,k}$ responded to a stimulus $\mathfrak{p} \in [0, 1]$ according to a poisson-noisy Gaussian:

$$\mathbf{a}_{m,k}(\mathfrak{p}) \sim \Pr(a_m \exp(-(\mathfrak{p}_k - \mathfrak{p})^2/\sigma_m^2)), \qquad (1)$$

for $a_a = 4, \sigma_a^2 = 0.01$ and $a_v = 7, \sigma_v^2 = 0.005$. This models receptive fields of neurons in the visual and auditory layers of the SC. There, neurons respond most strongly to stimuli from some direction, depending on their position in these layers, and less or not at all to stimuli from other directions [Stein and Stanford, 2013]. This spatiotopic organization is a feature found throughout the brain, and it is present also in neural populations projecting to the SC like the retina, the lateral geniculate nucleus, and the external nucleus of the inferior colliculus [Stone, 2012; Gutfreund and King, 2012]. In the following, the neurons $\mathbf{i}_{a,k}$ and $\mathbf{i}_{v,k}, k = 1, 2, \ldots$ will be referred to as 'auditory' and 'visual', respectively, to make presentation more intuitive.

We trained the network with congruent stimuli until it had developed spatial organization and learned the simulated input noise statistics. Figs. 3a, 3b, and 3c show typical input in the visual and auditory input populations, and the network's response, respectively, after training. Fig. 3d shows a PDF we recovered from the input using knowledge of the response properties of visual and auditory input neurons. The network did not have access to this knowledge. We then started simulating our chosen psychophysical and neurophysiological experiments. For these experiments, we changed the simulated stimulus conditions as will be described below.

Enhancement/Depression. It is a well-established fact that multi-sensory SC neurons tend to react more strongly to cross-sensory stimuli in their receptive fields than to unisensory stimuli [Stein and Meredith, 1993]. This effect is called enhancement. Depression, on the other hand, occurs when stimuli are temporally or spatially incongruent: In the spatial case, this means that the reaction of a multi-sensory neuron to a visual stimulus in its receptive field will actually be weaker if that stimulus is accompanied by a sound coming from a different point in space (and vice-versa).

We simulated this condition by presenting, in each trial, one random stimulus $\mathfrak{p}_a, \in [0, 1]$ to neurons $\mathbf{i}_{a,k}$ and a different stimulus $\mathfrak{p}_v \in [0, 1]$ to neurons $\mathbf{i}_{v,k}$. We recorded the network's response to the combined, incongruent, cross-sensory input population response. Fig. 4 shows the mean response over all trials of the output neuron at whose center was the visual stimulus depending on the absolute distance of the incongruent auditory stimulus. Although somewhat noisy, the graph clearly shows that congruent stimuli elicit much stronger responses than incongruent responses, which is in accordance with the phenomena of enhancement and depression explained above.

MLE. The effects discussed so far are on the level of single multi-sensory neurons. Since these neurons are part of a sensory-motor processing circuit, it is to be expected that they manifest themselves in observable behavior. Alais and Burr found that their test subjects' performance in an audio-visual localization task was consistent with a maximum likelihood estimator (MLE) model of multi-sensory integra-

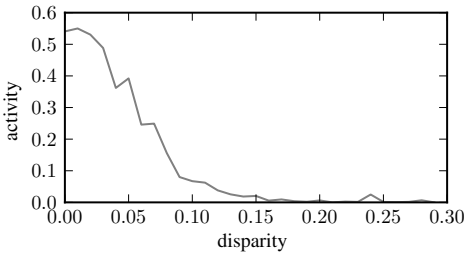


Figure 4: Response of an Output Neuron to a Visual Stimulus in its Receptive Field and an Auditory Stimulus at Various Distances from the Visual Stimulus.

tion [2004]. Other authors have found similar results for different combinations of sensory cues [Ernst and Banks, 2002; Hillis *et al.*, 2004].

The model used by Alais and Burr assumes Gaussian noise in sensory localizations. Under this assumption, MLE integrates two uni-sensory localizations l_a, l_v optimally according to a linear combination:

$$l_{MLE} = \frac{r_a}{r_a + r_v} l_a + \frac{r_v}{r_a + r_v} l_v, \tag{2}$$

where $r_m = \frac{1}{\sigma^2}$ is the reliability of a modality m, if σ is the standard error of localizations by that model, that is, the mean absolute error between the localization and the actual location of the target.

The expected reliability r_{MLE} of the combined result is given by:

$$\frac{1}{r_{MLE}} = \frac{1}{r_a} + \frac{1}{r_v}. \tag{3}$$

The distribution of errors of the combined estimator, like the assumed distribution of errors of the individual modalities' estimators, is Gaussian.

First, we determined the distributions of errors of our model given uni-sensory and cross-sensory stimuli. To do that, we fed our network with input in which either only auditory neurons $\mathbf{i}_{a,k}$, or only visual neurons $\mathbf{i}_{v,k}$ had non-zero activity (according to Eq. 1), or both, as usual. Figure 5 shows histograms of errors (mislocalizations) for uni- and cross-sensory localization, as well as Gaussian functions fitted to these errors. It can be seen that the distribution of errors is Gaussian-like, and that combined localization has much greater reliability than either of the individual localizations. Closer analysis reveals that the standard deviations of auditory-only, visual-only, and cross-modal localization are $\sigma_a = 4.594 \times 10^{-4}, \sigma_v = 1.061 \times 10^{-4}$, and $\sigma_m = 8.135 \times 10^{-5}$, respectively. The expected cross-modal localization error according to Eq. 3 would be $\sigma_{m,e} = 4.185 \times 10^{-5}$. We attribute this discrepancy to sampling error,[2] outliers, and actual learning errors. All in all, both visual inspection of the distribution of errors and this analysis demonstrate that our network effectively integrates the information in its multi-sensory input.

To test whether the behavior of our network is consistent with the MLE model described above, we conducted another experiment. As in the first experiment, we again chose one auditory stimulus $\mathfrak{p}_a$ and a visual stimulus $\mathfrak{p}_v$ in every trial. This time, each trial consisted of three conditions: an auditory, a visual, and a cross-sensory condition. In the auditory condition, we combined the normal auditory population response (Eq. 1) with a flat response of all-zero activity. The visual condition was analogous and in the cross-sensory condition, the population responses were combined as usual. In each trial n, the input was presented to the model and the auditory, visual, and cross-sensory localizations $l_{a,n}, l_{v,n}, l_{c,n}$ were recorded.

We then computed the least-squares solution to the equation

$$p_a \begin{pmatrix} l_{a,1} \\ l_{a,2} \\ \vdots \\ l_{a,N} \end{pmatrix} + p_v \begin{pmatrix} l_{v,1} \\ l_{v,2} \\ \vdots \\ l_{v,N} \end{pmatrix} = \begin{pmatrix} l_{m,1} \\ l_{m,2} \\ \vdots \\ l_{m,N} \end{pmatrix},$$

where $N = 10\,000$ is the number of trials. We found $p_a = 1.680 \times 10^{-1}$ and $p_v = 8.272 \times 10^{-1}$ which is close to the optimal values $\hat{p}_a = 1.876 \times 10^{-1}$, $\hat{p}_v = 8.124 \times 10^{-1}$ obtained by inserting σ_a and σ_v into Eq. 2.

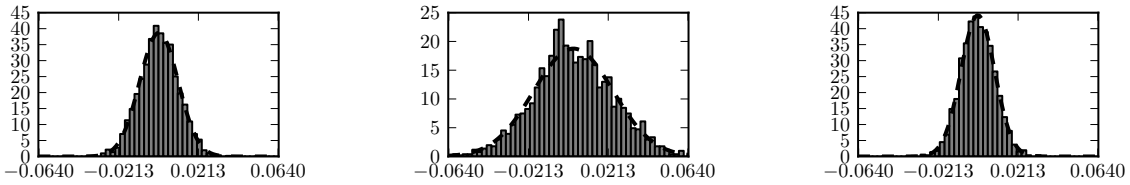
Together, these results show that our algorithm is not only statistically well-motivated and shows response characteristics similar to that of a biological information processing system, the SC, as was found in the first experiment: Its behavior on the functional level is also comparable to the optimal cue combination behavior demonstrated in human multi-sensory integration. This is especially interesting for our algorithm as a general machine learning algorithm.

4 Discussion

In this paper, we have shown that the neural learning algorithm introduced previously is able not only to integrate multi-sensory input, but also mimics biology both on the single-neuron and behavioral level. We can therefore interpret our network as a model of the SC, as it develops a representation of sensory input space, integrates percepts from different modalities depending on their reliabilities, uses the statistics of the input to learn this, and incorporates concepts known to be key in the SC, like population coding, winner-take-all, and local interactions.

We strongly believe that both fields, life sciences and artificial intelligence, will benefit from the approach of modeling observed biology to generate biological research questions, and implementing models in technical systems to validate their fitness and real-world applicability (see Fig. 6). Therefore, the next step will be validating our model's functionality and resemblance of biology in a robotic implementation. Initially, our experiments will mimic classical experiments like the ones due to Stein and Meredith [1993], which originally established the properties of multisensory integration in the cat SC: In our versions of these experiments, a robot will take the place of the feline or human subjects. It will be exposed to very similar multi-sensory stimuli as the

[2]Estimation is done by selecting the winner neuron and choosing its preferred value as the estimate. Since there are only finitely many neurons but infinitely many rationals in $[0, 1]$, estimation is bound to make sampling errors.



(a) Visual. (b) Auditory. (c) Audio-Visual.

Figure 5: Histograms of Distances between Visual, Auditory, and Audio-Visual Localization and Stimulus.

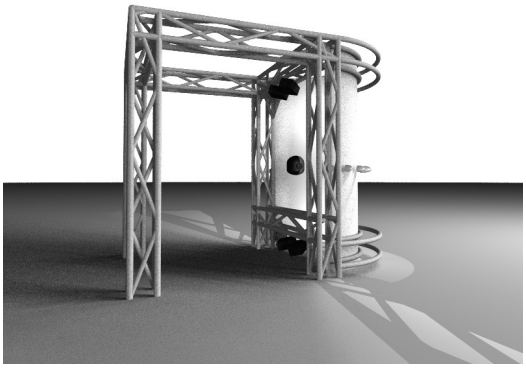


Figure 7: 3D Model of the Virtual Reality Robot Environment: A multi-purpose aluminium structure holds four projectors and a 180° projection screen. Around the screen, there is an array of speakers. The robot head is placed at the center of the half-cylinder spanned by the screen.

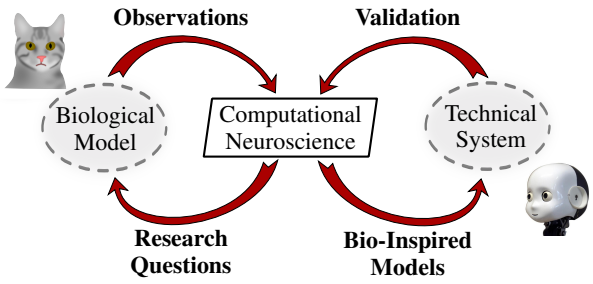


Figure 6: Research Cycle in Computational/Robotic Neuroscience

subjects in the original experiments. And its behavior and simulated neural processing will be monitored and compared to the original findings.

For these experiments, we will use the virtual reality lab infrastructure recently implemented ([Bauer *et al.*, 2012a], see Fig. 7). Designed and built as a basis for robotic sensory and cognitive experiments, this Virtual Reality for Robots Lab features a 180° projection screen and a matrix of 18 speakers. It allows us to precisely control the conditions of audio-visual localization experiments and still deliver rich and life-like stimuli to the iCub robot head which is placed at its center [Beira *et al.*, 2006]. With feedback from these experiments, we will extend our model and aim to accommodate attentional effects. This will give our model greater explanatory power and, at the same time, make it more flexible and more widely applicable in robotic and other AI systems. Again, biological experiments like those by Spence *et al.* [2004], which studied the effects of priming on multi-sensory integration, will guide our modeling efforts and serve as models for robotic experiments.

Acknowledgements

This work is funded by the DFG German Research Foundation (grant #1247) – International Research Training Group CINACS (Cross-modal Interactions in Natural and Artificial Cognitive Systems).

References

[Alais and Burr, 2004] David Alais and David Burr. The ventriloquist effect results from near-optimal bimodal integration. *Current Biology*, 14(3):257–262, February 2004.

[Bauer and Wermter, 2013] Johannes Bauer and Stefan Wermter. Self-organized neural learning of statistical inference from high-dimensional data. In *Proceedings of the International Joint Conference of Artificial Intelligence 2013*, 2013. To appear.

[Bauer *et al.*, 2012a] Johannes Bauer, Jorge Dávila-Chacón, Erik Strahl, and Stefan Wermter. Smoke and mirrors — virtual realities for sensor fusion experiments in biomimetic robotics. In *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2012 IEEE Conference on*, pages 114–119. IEEE, 2012.

[Bauer *et al.*, 2012b] Johannes Bauer, Cornelius Weber, and Stefan Wermter. A SOM-based model for multi-sensory integration in the superior colliculus. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, June 2012.

[Beck *et al.*, 2008] Jeffrey M. Beck, Wei J. Ma, Roozbeh Kiani, Tim Hanks, Anne K. Churchland, Jamie Roitman, Michael N. Shadlen, Peter E. Latham, and Alexandre Pouget. Probabilistic population codes for bayesian decision making. *Neuron*, 60(6):1142–1152, December 2008.

[Beira *et al.*, 2006] Ricardo Beira, Manuel Lopes, Miguel Praça, José Santos-Victor, Alexandre Bernardino, Giorgio

Metta, Francesco Becchi, and Roque Saltarén. Design of the robot-cub (icub) head. *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 94–100, May 2006.

[Cuijpers and Erlhagen, 2008] Raymond H. Cuijpers and Wolfram Erlhagen. Implementing Bayes' rule with neural fields. In *Proceedings of the 18th international conference on Artificial Neural Networks, Part II*, ICANN '08, pages 228–237, Berlin, Heidelberg, 2008. Springer-Verlag.

[Ernst and Banks, 2002] Marc O. Ernst and Martin S. Banks. Humans integrate visual and haptic information in a statistically optimal fashion. *Nature*, 415(6870):429–433, January 2002.

[Gutfreund and King, 2012] Yoram Gutfreund and Andrew J. King. What is the role of vision in the development of the auditory space map? In Barry E. Stein, editor, *The New Handbook of Multisensory Processing*, chapter 32, pages 473–587. MIT Press, Cambridge, MA, USA, June 2012.

[Hillis *et al.*, 2004] James M. Hillis, Simon J. Watt, Michael S. Landy, and Martin S. Banks. Slant from texture and disparity cues: Optimal cue combination. *Journal of Vision*, 4(12):967–992, December 2004.

[Hyde and Knudsen, 2000] Peter S. Hyde and Eric I. Knudsen. Topographic projection from the optic tectum to the auditory space map in the inferior colliculus of the barn owl. *The Journal of Comparative Neurology*, 421(2):146–160, May 2000.

[Jacobs and Kruschke, 2011] Robert A. Jacobs and John K. Kruschke. Bayesian learning theory applied to human cognition. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2(1):8–21, 2011.

[Kaas, 1997] Jon H. Kaas. Topographic maps are fundamental to sensory processing. *Brain Research Bulletin*, 44(2):107–112, January 1997.

[Klanke, 2007] Stefan Klanke. *Learning Manifolds with the Parametrized Self-Organizing Map and Unsupervised Kernel Regression*. PhD thesis, University of Bielefeld, March 2007.

[Kohonen, 1982] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, January 1982.

[Landy *et al.*, 2011] Michael S. Landy, Martin S. Banks, and David C. Knill. Ideal-observer models of cue integration. In Julia Trommershäuser, Konrad Körding, and Michael S. Landy, editors, *Sensory Cue Integration*, chapter 1, pages 5–29. Oxford University Press, Oxford, August 2011.

[Pouget *et al.*, 2003] Alexandre Pouget, Peter Dayan, and Richard S. Zemel. Inference and computation with population codes. *Annual review of Neuroscience*, 26(1):381–410, 2003.

[Spence *et al.*, 2004] Charles Spence, John McDonald, and Jon Driver. Exogenous spatial-cuing studies of human crossmodal attention and multisensory integration. In Charles Spence and Jon Driver, editors, *Crossmodal Space and Crossmodal Attention*, chapter 11, pages 277–320. Oxford University Press, USA, May 2004.

[Stein and Meredith, 1993] Barry E. Stein and M. Alex Meredith. *The Merging Of The Senses*. Cognitive Neuroscience Series. MIT Press, 1 edition, January 1993.

[Stein and Stanford, 2013] Barry E. Stein and Terrence R. Stanford. *Development of the Superior Colliculus/Optic Tectum*, pages 41–59. Elsevier, 2013.

[Stein, 2012] Barry E. Stein. Early experience affects the development of multisensory integration in single neurons of the superior colliculus. In Barry E. Stein, editor, *The New Handbook of Multisensory Processing*, chapter 33, pages 589–606. MIT Press, Cambridge, MA, USA, June 2012.

[Stone, 2012] James V. Stone. *Vision and Brain: How We Perceive the World*. The MIT Press, 1 edition, September 2012.

[Yin, 2007] Hujun Yin. Learning nonlinear principal manifolds by self-organising maps. In *Principal Manifolds for Data Visualization and Dimension Reduction*, Lecture Notes in Computational Science and Engineering, chapter 3, pages 68–95. Springer, Dordrecht, 2007.

[Zhou *et al.*, 2011] Tao Zhou, Piotr Dudek, and Bertram E. Shi. Self-organizing neural population coding for improving robotic visuomotor coordination. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1437–1444. IEEE, July 2011.

Linear Autoencoder Networks for Structured Data

Alessandro Sperduti
Department of Mathematics
University of Padova, Italy
Email: sperduti@math.unipd.it

Abstract

We study linear autoencoder networks for structured inputs, such as sequences, trees. We show that the problem of training an autoencoder has a closed form solution which can be obtained via the definition of linear dynamical systems modelling the structural information present in the dataset of structures. Relationship with principal directions is discussed. We also briefly discuss how autoencoder networks can be used in relation to classification tasks.

1 Introduction

With the recent development of Deep Learning (e.g., [Hinton and Salakhutdinov, 2006; Hinton *et al.*, 2006; Bengio, 2009; di Lena *et al.*, 2012]), nonlinear autoencoder networks have witnessed a resurge of attention as a tool for developing rich internal representations of input data. In fact it seems more and more evident that effective learning should be based on relevant and robust internal representations developed in autonomy by the learning system.

Since in many cases neural-symbolic integration involves structured data, such as sequences, trees, and graphs, in this paper we investigate on autoencoder networks for structured data. Specifically, we discuss the most basic version of autoencoder networks, i.e. linear systems. We show that, for linear systems, it is actually possible to devise a closed form solution for learning which relates to Principal Component Analysis of the extended vectorial representations of the structured data. Principal Component Analysis (PCA) ([Jolliffe, 2002]) constitutes one of the oldest and best known tools in Pattern Recognition and Machine Learning. It is a powerful technique for dimensionality reduction, while preserving much of the relevant information conveyed by a set of variables. It is theoretically well founded and reduces to the solution of an eigenvalue problem involving the covariance (or correlation) matrix of the available data.

Exploiting the well known kernel trick, Kernel PCA ([Schölkopf *et al.*, 1997]), which is a nonlinear form of PCA, has been proposed. Through the kernel trick, it is possible to implicitly project data into high-dimensional feature spaces. PCA is then performed in feature space, discovering principal directions that correspond to principal curves in the original data space. By defining a kernel on structured data, such as sequences, trees, and graphs, it is also possible to apply PCA to application domains where it is natural to represent data in a structured form; just to name a few, Chemistry, Bioinformatics, and Natural Language Processing.

In this paper, we are interested in understanding how a PCA-like tool can be extended to structured data without using an a priori defined kernel for structures, especially when the information attached to each vertex of the structure is a real-valued vector.

We show that linear autoencoder networks are actually closely linked to PCA not only in the case of vectorial input (see [Bourlard and Kamp, 1988; Baldi and Hornik, 1989]), but also in the case of structured data modeled via a suitable linear dynamical system. From this perspective, performing PCA of a set of structures is equivalent to discover the most compact and informative dynamical system able to map each structure into a state space vector representing as much as possible of the information conveyed by the structure itself. The state space of such dynamical system would then define a quite compact data dependent feature space, directly amenable to be used by any of the standard Machine Learning tools for tasks such as classification, regression, and clustering. This would constitute a complementary approach to kernel-based methods for structured input (see [Gartner, 2003] for a survey), where the input structures are mapped by an a priori defined nonlinear function into a very large feature space. The definition of a data dependent and compact feature space is reminiscent of the hidden activation space of Recursive Neural Networks (e.g. see [Sperduti and Starita, 1997; Frasconi *et al.*, 1998; Hammer, 2000; Baldi and Pollastri, 2003; Micheli *et al.*, 2001]), which have been successfully applied to learning tasks in Chemistry and Bioinformatics. Recursive Neural Networks, however, suffer the local minima problem, and often are not so easy to train because of the use of sigmoidal functions that introduce plateaus into the objective function.

We show that it is actually possible to define PCA-like representations for structured data via autoencoder networks, mainly for sequences and trees, since graphs can be treated indirectly as a sequence of nodes. We start by recalling PCA for vectors (Section 2.1). Then, we introduce a linear dynamical system as basic component of a linear autoencoder, and then the main result used for the definition of PCA-like rep-

resentations for sequences (Section 2.2). The application of the same ideas to trees is discussed in Section 2.3, where an extended dynamical system is introduced. In Section 3 we briefly discuss how autoencoders can be used for classification tasks. Conclusions are drawn in Section 4.

Preliminary work described in this paper and results obtained on empirical data have been presented in ([Sperduti, 2006; Micheli and Sperduti, 2007; Sperduti, 2007]).

2 Linear Autoencoder Networks and PCA for Vectors and Structures

In the following we study the relationship between solutions to linear autoencoder networks and the computation of PCA for vectors and structured data. We briefly recall the standard PCA with a perspective that will allow us to readily understand the connection with solutions to linear autoencoder networks for sequences. The suggested approach is then further extended to cover the direct treatment of trees.

2.1 Vectors

One of the aims of standard PCA ([Jolliffe, 2002]) is to reduce the dimensionality of a data set, while preserving as much as possible the information present in it. This is achieved by looking for orthogonal directions of maximum variance within the data set. The principal components are sorted according to the amount of variance they explain, so that the first few retain most of the variation present in all of the original variables. It turns out that the qth principal component is given by the projection of the data onto the eigenvector of the (sample) covariance matrix $\mathbf{C}$ of the data corresponding to the qth largest eigenvalue.

From a mathematical point of view, PCA can be understood as given by an orthogonal linear transformation of the given set of variables (i.e., the coordinates of the vectorial space in which data is embedded):

$$\mathbf{y}_i = \mathbf{A}\mathbf{x}_i$$

where $\mathbf{x}_i \in \mathbb{R}^k$ are the vectors belonging to the data set, and $\mathbf{A} \in \mathbb{R}^{k\times k}$ is the orthogonal matrix whose qth row is the qth eigenvector of the covariance matrix. Typically, larger variances are associated with the first $p < k$ principal components. Thus one can conclude that most relevant information occur only in the first p dimensions. The process of retaining only the first p principal components is known as *dimensional reduction*. Given a fixed value for p, principal components allow also to minimize the reconstruction error, i.e. the square error of the difference between the original vector $\mathbf{x}_i$ and the vector obtained by projecting its principal components $\mathbf{y}_i$ back into the original space by the linear transformation ${\mathbf{A}^{(p)}}^{\mathsf{T}}\mathbf{y}_i$:

$$\mathbf{A}^{(p)} = \arg\min_{\mathbf{M}\in\mathbb{R}^{p\times k}} \sum_i \|\mathbf{x}_i - \mathbf{M}^{\mathsf{T}}\mathbf{M}\mathbf{x}_i\|^2$$

where the rows of $\mathbf{A}^{(p)} \in \mathbb{R}^{p\times k}$ corresponds to the first p eigenvectors of $\mathbf{C}$. This can be shown by resorting to a Rayleigh quotient. In fact, let

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^{\mathsf{T}} \\ \mathbf{x}_2^{\mathsf{T}} \\ \mathbf{x}_3^{\mathsf{T}} \\ \vdots \\ \mathbf{x}_n^{\mathsf{T}} \end{bmatrix},$$

then the direction of maximum variance can be computed as

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \frac{\mathbf{w}^{\mathsf{T}}\mathbf{X}^{\mathsf{T}}\mathbf{X}\mathbf{w}}{\mathbf{w}^{\mathsf{T}}\mathbf{w}} = \arg\max_{\mathbf{w}} \frac{\mathbf{w}^{\mathsf{T}}\mathbf{C}\mathbf{w}}{\mathbf{w}^{\mathsf{T}}\mathbf{w}}.$$

By imposing with no loss in generality that $\|\mathbf{w}\| = 1$, an equivalent problem is

$$\mathbf{w}^* = \arg\max_{\|\mathbf{w}\|=1} \mathbf{w}^{\mathsf{T}}\mathbf{C}\mathbf{w}.$$

This is a constrained optimization problem that can be solved by optimizing the Lagrangian

$$\mathcal{L}(\mathbf{w},\lambda) = \mathbf{w}^{\mathsf{T}}\mathbf{C}\mathbf{w} - \lambda(\mathbf{w}^{\mathsf{T}}\mathbf{w} - 1).$$

By differentiating the Lagrangian with respect to $\mathbf{w}$ and equating to zero leads to

$$\mathbf{C}\mathbf{w} - \lambda\mathbf{w} = 0,$$

which corresponds to the following symmetric eigenvalue problem

$$\mathbf{C}\mathbf{w} = \lambda\mathbf{w}.$$

Thus the first principal direction corresponds to the eigenvector with maximum eigenvalue, while the other principal directions correspond to the other eigenvectors (pairwise orthogonal by definition), sorted according to the corresponding eigenvalues.

In [Bourlard and Kamp, 1988; Baldi and Hornik, 1989], principal directions have been related to solutions obtained by training linear autoencoder networks

$$\mathbf{o}_i = \mathbf{W}_{hidden}\mathbf{W}_{input}\mathbf{x}_i,\ i = 1,\ldots,n, \tag{1}$$

where $\mathbf{W}_{input} \in \mathbb{R}^{p\times k}$, $\mathbf{W}_{hidden} \in \mathbb{R}^{k\times p}$, $p \ll k$, and the network is trained so to get $\mathbf{o}_i = \mathbf{x}_i,\ \forall i$.

2.2 Sequences

When a temporal sequence $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t, \ldots$ of input vectors, where t is a discrete time index, is considered, the autoencoder defined in eq. (1) is extended by considering the coupled linear dynamical systems

$$\mathbf{y}_t = \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{y}_{t-1} \tag{2}$$

$$\begin{bmatrix} \mathbf{x}_t \\ \mathbf{y}_{t-1} \end{bmatrix} = \mathbf{C}\mathbf{y}_t, \tag{3}$$

It should be noticed that eq. (2) extends the linear transformation defined in eq. (2.1) by introducing a *memory term* involving the matrix $\mathbf{B} \in \mathbb{R}^{p\times p}$. This approach has been proposed, for example, in ([Voegtlin, 2005]) where an iterative procedure, based on Oja's rule, is presented. No proof of convergence for the proposed procedure is given.

Here we give a formal treatment which allows us to reach a sound solution to the above problem while returning, as a by-product, a closed form solution to the problem of training the autoencoder defined by eqs. (2) and (3).

The basic idea is to look for directions of high variance into the *state space* of the dynamical linear system (2).

Let start by considering a single sequence $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_n$ and the state vectors of the corresponding induced state sequence collected as rows of a matrix

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}_1^\mathsf{T} \\ \mathbf{y}_2^\mathsf{T} \\ \mathbf{y}_3^\mathsf{T} \\ \vdots \\ \mathbf{y}_n^\mathsf{T} \end{bmatrix}.$$

By using the initial condition $\mathbf{y}_0 = \mathbf{0}$ (the null vector), and the dynamical linear system (2), we can rewrite the matrix as

$$\mathbf{Y} = \begin{bmatrix} (\mathbf{A}\mathbf{x}_1)^\mathsf{T} \\ (\mathbf{A}\mathbf{x}_2 + \mathbf{B}\mathbf{A}\mathbf{x}_1))^\mathsf{T} \\ (\mathbf{A}\mathbf{x}_3 + \mathbf{B}\mathbf{A}\mathbf{x}_2 + \mathbf{B}^2\mathbf{A}\mathbf{x}_1)^\mathsf{T} \\ \vdots \\ (\mathbf{A}\mathbf{x}_n + \cdots + \mathbf{B}^{n-2}\mathbf{A}\mathbf{x}_2 + \mathbf{B}^{n-1}\mathbf{A}\mathbf{x}_1)^\mathsf{T} \end{bmatrix},$$

which can be factorized as

$$\mathbf{Y} = \underbrace{\begin{bmatrix} \mathbf{x}_1^\mathsf{T} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{x}_2^\mathsf{T} & \mathbf{x}_1^\mathsf{T} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{x}_3^\mathsf{T} & \mathbf{x}_2^\mathsf{T} & \mathbf{x}_1^\mathsf{T} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{x}_n^\mathsf{T} & \mathbf{x}_{n-1}^\mathsf{T} & \mathbf{x}_{n-2}^\mathsf{T} & \cdots & \mathbf{x}_2^\mathsf{T} & \mathbf{x}_1^\mathsf{T} \end{bmatrix}}_{\boldsymbol{\Xi}} \underbrace{\begin{bmatrix} \mathbf{A}^\mathsf{T} \\ \mathbf{A}^\mathsf{T}\mathbf{B}^\mathsf{T} \\ \mathbf{A}^\mathsf{T}\mathbf{B}^{2\mathsf{T}} \\ \vdots \\ \mathbf{A}^\mathsf{T}\mathbf{B}^{n-1\mathsf{T}} \end{bmatrix}}_{\boldsymbol{\Omega}}$$

where, given $s = kn$, $\boldsymbol{\Xi} \in \mathbb{R}^{n \times s}$ is a data matrix collecting all the (inverted) input subsequences (including the whole sequence) as rows, and $\boldsymbol{\Omega}$ is the parameter matrix of the dynamical system.

Now, we are interested in using a state space of smallest dimension p, i.e. $\mathbf{y}_t \in \mathbb{R}^p$, such that all information contained in $\boldsymbol{\Omega}$ is preserved. We start by factorizing $\boldsymbol{\Xi}$ using SVD, obtaining

$$\boldsymbol{\Xi} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{U}^\mathsf{T}$$

where $\mathbf{V} \in \mathbb{R}^{n \times n}$ is an unitary matrix, $\boldsymbol{\Lambda} \in \mathbb{R}^{n \times s}$ is a rectangular diagonal matrix with nonnegative real numbers on the diagonal with $\lambda_{1,1} \geq \lambda_{2,2} \geq \cdots \geq \lambda_{n,n}$ (the singular values), and $\mathbf{U}^\mathsf{T} \in \mathbb{R}^{s \times n}$ is a unitary matrix.

It is important to notice that columns of $\mathbf{U}^\mathsf{T}$ which correspond to nonzero singular values, apart some mathematical technicalities, basically correspond to the principal directions of data, i.e. PCA.

If the rank of $\boldsymbol{\Xi}$ is p, then only the first p elements of the diagonal of $\boldsymbol{\Lambda}$ are not null, and the above decomposition can be reduced to

$$\boldsymbol{\Xi} = \mathbf{V}^{(p)}\boldsymbol{\Lambda}^{(p)}{\mathbf{U}^{(p)}}^\mathsf{T} \tag{4}$$

where $\mathbf{V}^{(p)} \in \mathbb{R}^{n \times p}$, $\boldsymbol{\Lambda}^{(p)} \in \mathbb{R}^{p \times p}$, and ${\mathbf{U}^{(p)}}^\mathsf{T} \in \mathbb{R}^{p \times n}$. Now we can observe that ${\mathbf{U}^{(p)}}^\mathsf{T}\mathbf{U}^{(p)} = \mathbf{I}$ (where $\mathbf{I}$ is the identity matrix of dimension p), since by definition the columns of $\mathbf{U}^{(p)}$ are orthogonal, and by imposing $\boldsymbol{\Omega} = \mathbf{U}^{(p)}$, we can derive "optimal" matrices $\mathbf{A} \in \mathbb{R}^{p \times k}$ and $\mathbf{B} \in \mathbb{R}^{p \times p}$ for our dynamical system, which will have corresponding state space matrix

$$\mathbf{Y}^{(p)} = \boldsymbol{\Xi}\boldsymbol{\Omega} = \boldsymbol{\Xi}\mathbf{U}^{(p)} = \mathbf{V}^{(p)}\boldsymbol{\Lambda}^{(p)}{\mathbf{U}^{(p)}}^\mathsf{T}\mathbf{U}^{(p)} = \mathbf{V}^{(p)}\boldsymbol{\Lambda}^{(p)}.$$

Thus, if we represent $\mathbf{U}^{(p)}$ as composed of n submatrices $\mathbf{U}_i^{(p)}$, each of size $k \times p$, the problem reduces to find matrices $\mathbf{A}$ and $\mathbf{B}$ such that

$$\boldsymbol{\Omega} = \begin{bmatrix} \mathbf{A}^\mathsf{T} \\ \mathbf{A}^\mathsf{T}\mathbf{B}^\mathsf{T} \\ \mathbf{A}^\mathsf{T}\mathbf{B}^{2\mathsf{T}} \\ \vdots \\ \mathbf{A}^\mathsf{T}\mathbf{B}^{n-1\mathsf{T}} \end{bmatrix} = \begin{bmatrix} \mathbf{U}_1^{(p)} \\ \mathbf{U}_2^{(p)} \\ \mathbf{U}_3^{(p)} \\ \vdots \\ \mathbf{U}_n^{(p)} \end{bmatrix} = \mathbf{U}^{(p)}. \tag{5}$$

In the following, we demonstrate that there exists a solution to the above equation. We start by observing that $\boldsymbol{\Xi}$ owns a special structure, i.e. given $\boldsymbol{\Xi} = [\boldsymbol{\Xi}_1 \ \boldsymbol{\Xi}_2 \ \cdots \ \boldsymbol{\Xi}_n]$, where $\boldsymbol{\Xi}_i \in \mathbb{R}^{n \times k}$, then for $i = 1, \ldots, n-1$

$$\boldsymbol{\Xi}_{i+1} = \mathbf{R}_{k,s}\boldsymbol{\Xi}_i = \begin{bmatrix} \mathbf{0}_{1\times(n-1)} & \mathbf{0}_{1\times(s-n+1)} \\ \mathbf{I}_{(n-1)\times(n-1)} & \mathbf{0}_{(n-1)\times(s-n+1)} \end{bmatrix} \boldsymbol{\Xi}_i,$$

and

$$\mathbf{R}_{k,s}\boldsymbol{\Xi}_n = \mathbf{0}, \text{ i.e. the null matrix of size } n \times k.$$

Moreover, by eq. (4), we have

$$\boldsymbol{\Xi}_i = \mathbf{V}^{(p)}\boldsymbol{\Lambda}^{(p)}{\mathbf{U}_i^{(p)}}^\mathsf{T}, \text{ for } i = 1, \ldots, n.$$

Using the fact that ${\mathbf{V}^{(p)}}^\mathsf{T}\mathbf{V}^{(p)} = \mathbf{I}$, and combining the above equations, we get

$$\mathbf{U}_{i+t}^{(p)} = \mathbf{U}_i^{(p)}\mathbf{Q}^t, \text{ for } i = 1, \ldots, n-1, \text{ and } t = 1, \ldots, n-i$$

where $\mathbf{Q} = \boldsymbol{\Lambda}^{(p)}{\mathbf{V}^{(p)}}^\mathsf{T}\mathbf{R}_{k,s}^\mathsf{T}\mathbf{V}^{(p)}{\boldsymbol{\Lambda}^{(p)}}^{-1}$. Moreover, we have that $\mathbf{U}_n^{(p)}\mathbf{Q} = \mathbf{0}$ since

$$\begin{aligned} \mathbf{U}_n^{(p)}\mathbf{Q} &= \mathbf{U}_n^{(p)}\boldsymbol{\Lambda}^{(p)}{\mathbf{V}^{(p)}}^\mathsf{T}\mathbf{R}_{k,s}^\mathsf{T}\mathbf{V}^{(p)}{\boldsymbol{\Lambda}^{(p)}}^{-1} \\ &= (\mathbf{R}_{k,s}\boldsymbol{\Xi}_n)^\mathsf{T}\mathbf{V}^{(p)}{\boldsymbol{\Lambda}^{(p)}}^{-1}. \end{aligned} \tag{6}$$

Thus, eq. (5) is satisfied by $\mathbf{A} = {\mathbf{U}_1^{(p)}}^\mathsf{T}$ and $\mathbf{B} = \mathbf{Q}^\mathsf{T}$.

It is interesting to note that the original data $\boldsymbol{\Xi}$ can be recovered by computing

$$\mathbf{Y}^{(p)}{\mathbf{U}^{(p)}}^\mathsf{T} = \mathbf{V}^{(p)}\boldsymbol{\Lambda}^{(p)}{\mathbf{U}^{(p)}}^\mathsf{T} = \boldsymbol{\Xi},$$

which can be achieved by running the dynamical system

$$\begin{bmatrix} \mathbf{x}_t \\ \mathbf{y}_{t-1} \end{bmatrix} = \begin{bmatrix} \mathbf{A}^\mathsf{T} \\ \mathbf{B}^\mathsf{T} \end{bmatrix} \mathbf{y}_t$$

starting from $\mathbf{y}_n$, i.e. $\begin{bmatrix} \mathbf{A}^\mathsf{T} \\ \mathbf{B}^\mathsf{T} \end{bmatrix}$ is the matrix $\mathbf{C}$ defined in eq. (3).

Finally, it is important to remark that the above construction works not only for a single sequence, but also for a set of sequences of different length. For example, let consider the two sequences $(\mathbf{x}_1^a, \mathbf{x}_2^a, \mathbf{x}_3^a, \mathbf{x}_4^a)$ and $(\mathbf{x}^b{}_1, \mathbf{x}^b{}_2)$. Then, we have

$$\mathbf{\Xi_a} = \begin{bmatrix} \mathbf{x}_1^{a\mathsf{T}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{x}_2^{a\mathsf{T}} & \mathbf{x}_1^{a\mathsf{T}} & \mathbf{0} & \mathbf{0} \\ \mathbf{x}_3^{a\mathsf{T}} & \mathbf{x}_2^{a\mathsf{T}} & \mathbf{x}_1^{a\mathsf{T}} & \mathbf{0} \\ \mathbf{x}_4^{a\mathsf{T}} & \mathbf{x}_3^{a\mathsf{T}} & \mathbf{x}_2^{a\mathsf{T}} & \mathbf{x_1}^{a\mathsf{T}} \end{bmatrix}$$

and

$$\mathbf{\Xi_b} = \begin{bmatrix} \mathbf{x}_1^{b\mathsf{T}} & \mathbf{0} \\ \mathbf{x}_2^{b\mathsf{T}} & \mathbf{x}_1^{b\mathsf{T}} \end{bmatrix}$$

which can be collected together into the matrix

$$\mathbf{\Xi} = \left[\begin{array}{c|c} \mathbf{\Xi_a} & \\ \hline \mathbf{\Xi_b} & \mathbf{0}_{2\times 2} \end{array}\right]$$

and matrix

$$\mathbf{R} = \left[\begin{array}{c|c} \mathbf{R}_{k,4k} & \\ \hline \mathbf{R}_{k,2k} & \mathbf{0}_{2\times 2} \end{array}\right]$$

to define matrix $\mathbf{Q}$.

The following lemma shows the link between matrix $\mathbf{Q}$ and the principal components corresponding to matrix $\mathbf{U}^{(p)}$

Lemma *(Relationship with Principal Directions)*

$$\mathbf{Q} = \mathbf{U}^{(p)\mathsf{T}}\mathbf{R}_{k,s}^{\mathsf{T}}\mathbf{U}^{(p)} = \sum_{i=1}^{n-1} \mathbf{U}_i^{(p)\mathsf{T}}\mathbf{U}_{i+1}^{(p)}$$

Proof: By definition $\sum_{i=1}^{n} \mathbf{U}_i^{(p)\mathsf{T}}\mathbf{U}_i^{(p)} = \mathbf{I}$ and

$$\begin{aligned} \mathbf{Q} &= \left(\sum_{i=1}^{n} \mathbf{U}_i^{(p)\mathsf{T}}\mathbf{U}_i^{(p)}\right)\mathbf{Q} = \sum_{i=1}^{n} \mathbf{U}_i^{(p)\mathsf{T}}\left(\mathbf{U}_i^{(p)}\mathbf{Q}\right) \\ &= \sum_{i=1}^{n-1} \mathbf{U}_i^{(p)\mathsf{T}}\mathbf{U}_{i+1}^{(p)} \end{aligned}$$

where we have used $\mathbf{U}_{i+1}^{(p)} = \mathbf{U}_i^{(p)}\mathbf{Q}$ and $\mathbf{U}_n^{(p)}\mathbf{Q} = \mathbf{0}$. ■

As a final remark, it should be stressed that the above construction *only* works if p is equal to the rank of $\mathbf{\Xi}$. If p is smaller, then there is no formal proof that the proposed solution is optimal, although empirical experimental results seem to be quite good (see [Sperduti, 2006; Micheli and Sperduti, 2007; Sperduti, 2007]).

2.3 Trees

When considering trees, an extension of the approach used for sequences can be used. First of all, let us illustrate what happens for an example given by a complete binary tree. Then, we will generalize the construction to (in)complete b-ary trees. For $b = 2$, we consider the following linear dynamical system (the decoding is obtained by the transposed dynamical system, as we have seen for sequences)

$$\mathbf{y}_u = \mathbf{A}\mathbf{x}_u + \mathbf{B_l}\mathbf{y}_{ch_l[u]} + \mathbf{B_r}\mathbf{y}_{ch_r[u]} \tag{7}$$

where u is a vertex of the tree, $ch_l[u]$ is the left child of u, $ch_r[u]$ is the right child of u, $\mathbf{B_l}, \mathbf{B_r} \in \mathbb{R}^{p\times p}$.

Let consider the following complete binary tree

$$\mathcal{T} \equiv \mathbf{x}_7(\mathbf{x}_5(\mathbf{x}_1, \mathbf{x}_2), \mathbf{x}_6(\mathbf{x}_3, \mathbf{x}_4)),$$

where we have used parentheses to represent the tree structure. Then we can consider the corresponding induced state elements collected as rows of a matrix

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}_1^{\mathsf{T}} \\ \mathbf{y}_2^{\mathsf{T}} \\ \mathbf{y}_3^{\mathsf{T}} \\ \vdots \\ \mathbf{y}_7^{\mathsf{T}} \end{bmatrix}.$$

By using the initial condition $\mathbf{y}_{nil} = \mathbf{0}$ (the null vector), and the dynamical linear system (7), we have

$$\mathbf{Y} = \begin{bmatrix} (\mathbf{A}\mathbf{x}_1)^{\mathsf{T}} \\ (\mathbf{A}\mathbf{x}_2)^{\mathsf{T}} \\ (\mathbf{A}\mathbf{x}_3)^{\mathsf{T}} \\ (\mathbf{A}\mathbf{x}_4)^{\mathsf{T}} \\ (\mathbf{A}\mathbf{x}_5 + \mathbf{B_l}\mathbf{A}\mathbf{x}_1 + \mathbf{B_r}\mathbf{A}\mathbf{x}_2)^{\mathsf{T}} \\ (\mathbf{A}\mathbf{x}_6 + \mathbf{B_l}\mathbf{A}\mathbf{x}_3 + \mathbf{B_r}\mathbf{A}\mathbf{x}_4)^{\mathsf{T}} \\ (\mathbf{A}\mathbf{x}_7 + \mathbf{B_l}\mathbf{A}\mathbf{x}_5 + \mathbf{B_r}\mathbf{A}\mathbf{x}_6 + \mathbf{B_l}^2\mathbf{A}\mathbf{x}_1 + \mathbf{B_l}\mathbf{B_r}\mathbf{A}\mathbf{x}_2 \\ \quad +\mathbf{B_r}\mathbf{B_l}\mathbf{A}\mathbf{x}_3 + \mathbf{B_r}^2\mathbf{A}\mathbf{x}_4)^{\mathsf{T}} \end{bmatrix},$$

which can be factorized as

$$\mathbf{Y} = \underbrace{\begin{bmatrix} \mathbf{x}_1^{\mathsf{T}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{x}_2^{\mathsf{T}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{x}_3^{\mathsf{T}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{x}_4^{\mathsf{T}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{x}_5^{\mathsf{T}} & \mathbf{x}_1^{\mathsf{T}} & \mathbf{x}_2^{\mathsf{T}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{x}_6^{\mathsf{T}} & \mathbf{x}_3^{\mathsf{T}} & \mathbf{x}_4^{\mathsf{T}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{x}_7^{\mathsf{T}} & \mathbf{x}_5^{\mathsf{T}} & \mathbf{x}_6^{\mathsf{T}} & \mathbf{x}_1^{\mathsf{T}} & \mathbf{x}_2^{\mathsf{T}} & \mathbf{x}_3^{\mathsf{T}} & \mathbf{x}_4^{\mathsf{T}} \end{bmatrix}}_{\mathbf{\Xi}} \underbrace{\begin{bmatrix} \mathbf{A}^{\mathsf{T}} \\ \mathbf{A}^{\mathsf{T}}\mathbf{B_l}^{\mathsf{T}} \\ \mathbf{A}^{\mathsf{T}}\mathbf{B_r}^{\mathsf{T}} \\ \mathbf{A}^{\mathsf{T}}\mathbf{B_l}^{2\mathsf{T}} \\ \mathbf{A}^{\mathsf{T}}\mathbf{B_r}^{\mathsf{T}}\mathbf{B_l}^{\mathsf{T}} \\ \mathbf{A}^{\mathsf{T}}\mathbf{B_l}^{\mathsf{T}}\mathbf{B_r}^{\mathsf{T}} \\ \mathbf{A}^{\mathsf{T}}\mathbf{B_r}^{2\mathsf{T}} \end{bmatrix}}_{\mathbf{\Omega}}.$$

It is not difficult to recognize that each macro component of $\mathbf{\Omega}$ corresponds to a path into a generic binary tree: $\mathbf{A}^{\mathsf{T}}$ is associated to the root, $\mathbf{A}^{\mathsf{T}}\mathbf{B_l}^{\mathsf{T}}$ is associated to the left child of the root, and so on. As for sequences, we have now to solve the following matricial equation

$$\mathbf{\Omega} = \begin{bmatrix} \mathbf{A}^{\mathsf{T}} \\ \mathbf{A}^{\mathsf{T}}\mathbf{B_l}^{\mathsf{T}} \\ \mathbf{A}^{\mathsf{T}}\mathbf{B_r}^{\mathsf{T}} \\ \mathbf{A}^{\mathsf{T}}\mathbf{B_l}^{2\mathsf{T}} \\ \mathbf{A}^{\mathsf{T}}\mathbf{B_r}^{\mathsf{T}}\mathbf{B_l}^{\mathsf{T}} \\ \mathbf{A}^{\mathsf{T}}\mathbf{B_l}^{\mathsf{T}}\mathbf{B_r}^{\mathsf{T}} \\ \mathbf{A}^{\mathsf{T}}\mathbf{B_r}^{2\mathsf{T}} \end{bmatrix} = \begin{bmatrix} \mathbf{U}_1^{(p)} \\ \mathbf{U}_2^{(p)} \\ \mathbf{U}_3^{(p)} \\ \mathbf{U}_4^{(p)} \\ \mathbf{U}_5^{(p)} \\ \mathbf{U}_6^{(p)} \\ \mathbf{U}_7^{(p)} \end{bmatrix} = \mathbf{U}^{?},$$

with respect to the three matrices $\mathbf{A}$, $\mathbf{B_l}$, and $\mathbf{B_r}$.

As for the case of sequences, we can observe that $\mathbf{\Xi}$ owns a special structure, i.e. given $\mathbf{\Xi} = [\mathbf{\Xi}_1\ \mathbf{\Xi}_2\ \cdots\ \mathbf{\Xi}_7]$ we can state that matrix $\mathbf{\Xi}_{left} = [\mathbf{\Xi}_2\ \mathbf{\Xi}_4\ \mathbf{\Xi}_5\ \mathbf{0}_{7\times 4k}]$ is equal to

$$\mathbf{\Xi}_{left} = \mathbf{R}_{k,left}\mathbf{\Xi} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \mathbf{\Xi},$$

and matrix $\mathbf{\Xi}_{right} = [\mathbf{\Xi}_3\ \mathbf{\Xi}_6\ \mathbf{\Xi}_7\ \mathbf{0}_{7\times 4k}]$ is equal to

$$\mathbf{\Xi}_{right} = \mathbf{R}_{k,right}\mathbf{\Xi} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{\Xi},$$

which combined with eq. (4) gives matrices

$$\mathbf{Q}_{left} = \mathbf{\Lambda}^{(p)}{\mathbf{V}^{(p)}}^{\mathsf{T}}\mathbf{R}_{k,left}^{\mathsf{T}}\mathbf{V}^{(p)}{\mathbf{\Lambda}^{(p)}}^{-1}$$

and

$$\mathbf{Q}_{right} = \mathbf{\Lambda}^{(p)}{\mathbf{V}^{(p)}}^{\mathsf{T}}\mathbf{R}_{k,right}^{\mathsf{T}}\mathbf{V}^{(p)}{\mathbf{\Lambda}^{(p)}}^{-1},$$

leading to the solution $\mathbf{A} = {\mathbf{U}_1^{(p)}}^{\mathsf{T}}$, $\mathbf{B_l} = \mathbf{Q}_{left}^{\mathsf{T}}$, and $\mathbf{B_r} = \mathbf{Q}_{right}^{\mathsf{T}}$.

Also in this case, we have *push* operators, i.e., $\mathbf{R}_{k,left}$ is a *push left* operator, while $\mathbf{R}_{k,right}$ is a *push right* operator. In the following, we describe how these push operators can be defined in general for complete binary trees.

Each vertex u of the binary tree is associated to a binary string $id(u)$ obtained as follows: the binary string "1" is associated to the root of the tree. Any other vertex has associated the string obtained by concatenating the string of its parent with the string "0" if it is a left child, "1" otherwise. Then, all the dimensions of the state space $\mathbb{S}$ are partitioned in s/k groups of k dimensions. The label associated to vertex v is stored into the j-th group, where j is the integer represented by the binary string $id(u)$. E.g. the label of the root is stored into group 1, since $id(root)$ ="1", the label of the vertex which can be reached by the path ll starting from the root is stored into group 4, since $id(u)$ ="100", while the label of the vertex reachable through the path rlr is stored into group 13, since $id(u)$ ="1101". Notice that, if the input tree is not complete, the components corresponding to missing vertexes are set to be equal to 0.

Matrices $\mathbf{B_l}$ and $\mathbf{B_r}$ are defined as follows. Both matrices are composed of two types of blocks, i.e. $\mathbf{I}_{k\times k}$ and $\mathbf{0}_{k\times k}$. Matrix $\mathbf{B_l}$ has to implement a *push left* operator, i.e. the tree $\mathcal{T}$ encoded by a vector $\mathbf{y}_{root(\mathcal{T})}$ has to become the left child of a new node u whose label is the current input $\mathbf{x}_u$. Thus $root(\mathcal{T})$ has to become the left child of u and also all the other vertexes in $\mathcal{T}$ have their position redefined accordingly. From a mathematical point of view, the new position of any vertex a in $\mathcal{T}$ is obtained by redefining $id(a)$ as follows: *i)* the most significative bit of $id(a)$ is set to "0", obtaining the string $id_0(a)$; *ii)* the new string $id_{new}(a)$ ="1"+$id_0(a)$ is defined, where + is the string concatenation operator. If $id_{new}(a)$ represents a number greater than s/k then this means that the vertex has been pushed outside the available memory, i.e. the vertex a is *lost*. Consequently, groups which correspond to *lost* vertexes have to be annihilated. Thus, $\mathbf{B_l}$ is composed of $(q+1)\times(q+1)$ blocks, all of type $\mathbf{0}_{k\times k}$, except for the blocks in row $id_{new}(a)$ and column $id(a)$, with $id_{new}(a) \leq s/k$, where a block $\mathbf{I}_{k\times k}$ is placed. Matrix $\mathbf{B_r}$ is defined similarly: it has to implement a *push right* operator, i.e.: *i)* the most significative bit of $id(a)$ is set to "1", obtaining the string $id_1(a)$; *ii)* the new string $id_{new}(a)$ ="1"+$id_1(a)$ is defined.

It should be noticed that the definition of the operators described above is not dependent on the number of the processed trees, but only depends on the size of the largest tree (memory size).

Generalization of the above scheme for complete b-ary trees is not difficult. The dynamical linear system becomes

$$\mathbf{y}_u = \mathbf{A}\mathbf{x}_u + \sum_{c=0}^{b-1} \mathbf{B_c}\mathbf{y}_{ch_c[u]} \tag{8}$$

where $ch_c[u]$ is the $c+1$-th child of u, and a matrix $\mathbf{B_c}$ is defined for each child. The string associated to each vertex is defined on the alphabet {"0","1",...,"b-1"}, since there are b children. The symbol $b-1$ is associated with the root and b push operations have to be defined. The new string associated to any vertex a in $\mathcal{T}$, after a c-push operation, is obtained by redefining $id(a)$ as follows: *i)* the most significative symbol of $id(a)$ is set to c, obtaining the string $id_c(a)$; *ii)* the new string $id_{new}(a)$ ="b-1"+$id_c(a)$ is defined. E.g., if $b = 5$ and c ="3", then *i)* the most significative symbol of $id(a)$ is set to "3", obtaining the string $id_3(a)$; *ii)* the new string $id_{new}(a)$ ="b-1"+$id_3(a)$ is defined. Matrix $\mathbf{B_c}$ is defined by placing blocks $\mathbf{I}_{k\times k}$ in positions $(id_{new}(a), id(a))$ only if $id_{new}(a) \leq s/k$, where $id_{new}(a)$ is interpreted as a number represented in base b. Finally, the optimal solution for a state space of dimension p is given by matrices

$$\mathbf{A} = {\mathbf{U}_1^{(p)}}^{\mathsf{T}},\ \mathbf{B}_c = \mathbf{Q}_{push_c}.$$

A problem in dealing with complete trees is that very soon there is a combinatorial explosion of the number of paths to consider, i.e. in order for the machine to deal with moderately deep trees, a huge value for s needs to be used. In practical applications, however, the observed trees tend to follow a specific generative model, and thus there may be many topologies which are never, or very seldomly, generated.

For this reason we suggest to use the following approach. Given a set of trees $\mathbf{T}$, the optimized graph $G_{\mathbf{T}}$ ([Sperduti and Starita, 1997]) is obtained by joining all the trees in such a way that any (sub)tree in $\mathbf{T}$ is represented only once. The optimized graph $G_{\mathbf{T}}$, which is a DAG, is then visited bottom-up, generating for each visited vertex v the set of id strings associated to the tree rooted in v, thus simulating all the different push operations which should be performed when presenting the trees in $\mathbf{T}$ to the machine. Repeated id strings are removed. The obtained set P is then used to define the state space of the machine: each string is associated to one group of k coordinates. In this way, only paths which appear in the set $\mathbf{T}$ (including all subtrees) are represented, thus drastically reducing the size of s, which will be equal to $|P| \times k$.

3 Classification of Structures

In this section, we briefly discuss how an autoencoder network for structures can be related to a linear dynamical system for classification. For the sake of presentation we restrict our discussion to binary classification problems involving trees.

We are interested to understand how a linear dynamical system of the form shown in eq. (8) can be used for classification. Specifically, we assume that the state $\mathbf{y}_u$, where u is the root node of tree T, is used as input to a classifier, e.g. an SVM (see [Cardin *et al.*, 2009]). Let $\{(T_i, d_i)\}_{i=1}^n$ be a training set of trees T_i, each belonging to class $d_i \in \{+1, -1\}$.

Let $\mathbf{r}(T_i)$ be any vectorial explicit representation of T_i compliant with the above dynamical system, i.e. the row of $\boldsymbol{\Xi}$ corresponding to the state obtained after presenting *all* the nodes of the tree to the above dynamical system. Let assume that the vectorial training set $\{(\mathbf{r}(T_i), d_i)\}_{i=1}^n$ is linearly separable and $\mathbf{w}^* = \sum_{j=1}^q \alpha_{i_j}^* \mathbf{r}(T_{i_j})$ be the optimal weight vector returned by an SVM, where $\alpha_{i_j}^* > 0$ are the optimal dual variables corresponding to support vectors $\mathbf{r}(T_{i_j})$.

We observe that, since $\mathbf{w}^*$ only depends on support vectors, then any linear dynamical system which only encodes the subspace spanned by the corresponding trees T_{i_j} is able to define an equivalent classifier. Specifically, let $\mathbf{T}_{sup} = \{T_{i_j}\}_{j=1}^q$ be the set of support trees and $\mathbf{T}_\perp$ the set of trees that are not support trees, i.e. all the trees in the training set are given by $\mathbf{T}_{sup} \cup \mathbf{T}_\perp$. Let $\boldsymbol{\Xi}(\mathbf{T}_{sup})$ be the data matrix generated by *all* nodes in trees belonging to $\mathbf{T}_{sup}$ and $\boldsymbol{\Xi}(\mathbf{T}_\perp)$ the corresponding matrix for $\mathbf{T}_\perp$. Notice that any $\mathbf{r}(T_{i_j})$ will be one row of $\boldsymbol{\Xi}(\mathbf{T}_{sup})$, however $\boldsymbol{\Xi}(\mathbf{T}_{sup})$ will also contain rows that correspond to subtrees belonging to any T_{i_j}. We also notice that matrix $\boldsymbol{\Xi}(\mathbf{T}_\perp)$ contains useful information for classification only with respect to the components of its rows that belong to the subspace spanned by the rows of $\boldsymbol{\Xi}(\mathbf{T}_{sup})$. We can compute this contribution by projecting the rows of $\boldsymbol{\Xi}(\mathbf{T}_\perp)$ over the rows of $\boldsymbol{\Xi}(\mathbf{T}_{sup})$. Such projection is obtained by finding the matrix solving the following optimization problem

$$\arg\min_{\mathbf{M}} \|\mathbf{M}\boldsymbol{\Xi}(\mathbf{T}_{sup}) - \boldsymbol{\Xi}(\mathbf{T}_\perp)\|^2.$$

The solution to the above problem is given by $\mathbf{M} = \boldsymbol{\Xi}(\mathbf{T}_\perp)\boldsymbol{\Xi}(\mathbf{T}_{sup})^+$, where $\boldsymbol{\Xi}(\mathbf{T}_{sup})^+$ is the pseudo-inverse of $\boldsymbol{\Xi}(\mathbf{T}_{sup})$ (easily obtainable by using its SVD decomposition). We now compute the "optimal" dynamical system with matrices $\mathbf{A}^*$, $\mathbf{B}_\mathbf{c}^*$, for matrix $\begin{bmatrix} \boldsymbol{\Xi}(\mathbf{T}_{sup}) \\ \boldsymbol{\Xi}(\mathbf{T}_\perp)\boldsymbol{\Xi}(\mathbf{T}_{sup})\boldsymbol{\Xi}(\mathbf{T}_{sup})^+ \end{bmatrix}$. Let $\mathbf{y}^*_{root(T_{i_j})}$ be the state vectors of such system corresponding to the roots of support trees T_{i_j}. Then, the reduced weight vector $\mathbf{w}_r^* = \sum_{j=1}^q \alpha_{i_j}^* \mathbf{y}^*_{root(T_{i_j})}$ returns the same classification of $\mathbf{w}^*$ over the extended representations $\mathbf{r}(T_i)$. The reason for that can be understood by recalling that $\mathbf{r}(T_{i_j}) = \mathbf{y}^*_{root(T_{i_j})} {\mathbf{U}^{(p)}}^\mathsf{T}$ and so $\mathbf{r}(T_{i_j})\mathbf{r}(T_{i_j})^\mathsf{T} = \mathbf{y}^*_{root(T_{i_j})} {\mathbf{U}^{(p)}}^\mathsf{T} \mathbf{U}^{(p)} {\mathbf{y}^*}^\mathsf{T}_{root(T_{i_j})} = \mathbf{y}^*_{root(T_{i_j})} {\mathbf{y}^*}^\mathsf{T}_{root(T_{i_j})}$ (please, remember that we are managing row vectors, so $\mathbf{r}(T_{i_j})\mathbf{r}(T_{i_j})^\mathsf{T}$ is a dot product). Moreover, for trees that are not of support, only the projection over the vectorial representation of the support trees matters, which is exactly reconstructed by the dynamical system.

Of course, the above construction can also be used when data is not linearly separable and a kernel is used. In this case, the linear dynamical system is used to generate reduced representations which are then used in substitution of the full explicit representations used by the SVM.

4 Conclusion

In this paper, we have shown that solutions to linear autoencoder networks for structured data is closely related to principal directions of explicit vectorial representations of structured data which are compliant to suitable linear dynamical systems describing the structural information. Through this approach, we showed it is possible to reach a theoretically well founded solution for linear autoencoders. We briefly also discussed how linear autoencoders can be used to define linear dynamical systems for classification.

Here we did not discuss that it is actually possible to compute such solution when a kernel map is used in the label space, although the computational burden can increase significantly when the total number of components is larger than the dimension of the label space times the depth of the memory used to represent structural information.

Experimental results are not discussed here, however in other papers ([Sperduti, 2006; Micheli and Sperduti, 2007; Sperduti, 2007]) experiments involving sequences, trees, and graphs have confirmed the feasibility of the approach and the effectiveness of the proposed methods for reducing the computational burden. From the computational point of view, further improvements, not explored in this paper, can be obtained by considering the sparsity of the $\boldsymbol{\Xi}$ matrix, and the adoption of more sophisticated numerical algorithms for computing the SVD factorization.

More work is needed to precisely relate the proposed approach with KPCA using a kernel for structures. Also a more convincing formulation for the processing of graphs is needed. In fact, while the proposed formulation is fully satisfactory for sequences and trees, it is not easy to directly address graphs, mainly because of the presence of confluent edges and cycles. It may also be interesting to investigate if it is possible to extend the proposed approach to include nonlinearities in analogy to the work proposed by Hinton & Salakhutdinov for vectorial data ([Hinton *et al.*, 2006]).

References

[Baldi and Hornik, 1989] Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53–58, 1989.

[Baldi and Pollastri, 2003] Pierre Baldi and Gianluca Pollastri. The principled design of large-scale recursive neural network architectures–dag-rnns and the protein structure prediction problem. *J. Mach. Learn. Res.*, 4:575–602, 2003.

[Bengio, 2009] Yoshua Bengio. Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[Bourlard and Kamp, 1988] H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59(4-5):291–294, 1988.

[Callan and Palmer-Brown, 1997] Robert E. Callan and Dominic Palmer-Brown. (s)raam: An analytical technique for fast and reliable derivation of connectionist symbol structure representations. *Connect. Sci.*, 9(2):139–160, 1997.

[Cardin *et al.*, 2009] Riccardo Cardin, Lisa Michielan, Stefano Moro, and Alessandro Sperduti. Pca-based representations of graphs for prediction in qsar studies. In *ICANN (2)*, pages 105–114, 2009.

[di Lena *et al.*, 2012] Pietro di Lena, Ken Nagata, and Pierre Baldi. Deep architectures for protein contact map prediction. *Bioinformatics*, 28(19):2449–2457, 2012.

[Frasconi *et al.*, 1998] P. Frasconi, M. Gori, and A. Sperduti. A general framework for adaptive processing in data structures. *IEEE Transactions on Neural Networks*, Vol 9(5):768–785, 1998.

[Gartner, 2003] Thomas Gartner. A survey of kernels for structured data. *SIGKDD Explorations*, 5(1):49–58, 2003.

[Hammer, 2000] Barbara Hammer. *Learning with Recurrent Neural Networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2000.

[Hasselmann, 1988] K. Hasselmann. Pips and pops: The reduction of complex dynamical systems using principal interaction and oscillation patters. *Geophys. Res.*, 93:11015–11021, 1988.

[Hinton and Salakhutdinov, 2006] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 2006.

[Hinton *et al.*, 2006] Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[Jolliffe, 2002] I.T. Jolliffe. *Principal Component Analysis*. Springer-Verlag New York, Inc., 2002.

[Micheli and Sperduti, 2007] Alessio Micheli and Alessandro Sperduti. Recursive principal component analysis of graphs. In *ICANN (2)*, pages 826–835, 2007.

[Micheli *et al.*, 2001] Alessio Micheli, Alessandro Sperduti, Antonina Starita, and Anna Maria Bianucci. Analysis of the internal representations developed by neural networks for structures applied to quantitative structure-activity relationship studies of benzodiazepines. *Journal of Chemical Information and Computer Sciences*, 41(2):202–218, 2001.

[N.E. *et al.*, 2001] Goljandina N.E., Nekrutkin V.V., and Zhigljavsky A.A. *Analysis of Time Series Structure: SSA and related techniques*. Chapman & Hall / CRS, 2001.

[Paccanaro and Hinton, 2001] Alberto Paccanaro and Geoffrey E. Hinton. Learning distributed representations of concepts using linear relational embedding. *IEEE Transactions on Knowledge and Data Engineering*, 13(2):232–244, 2001.

[Pollack, 1990] Jordan B. Pollack. Recursive distributed representations. *Artif. Intell.*, 46(1-2):77–105, 1990.

[Pollack, 1991] Jordan B. Pollack. The induction of dynamical recognizers. *Mach. Learn.*, 7(2-3):227–252, 1991.

[Schölkopf *et al.*, 1997] Bernhard Schölkopf, Alex J. Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *ICANN '97: Proceedings of the 7th International Conference on Artificial Neural Networks*, pages 583–588, London, UK, 1997. Springer-Verlag.

[Sperduti and Starita, 1997] A. Sperduti and A. Starita. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(3):714–735, 1997.

[Sperduti, 2006] Alessandro Sperduti. Exact solutions for recursive principal components analysis of sequences and trees. In *ICANN (1)*, pages 349–356, 2006.

[Sperduti, 2007] Alessandro Sperduti. Efficient computation of recursive principal component analysis for structured input. In *ECML*, pages 335–346, 2007.

[Voegtlin and Dominey, 2005] Thomas Voegtlin and Peter F. Dominey. Linear recursive distributed representations. *Neural Netw.*, 18(7):878–895, 2005.

[Voegtlin, 2005] Thomas Voegtlin. Recursive principal components analysis. *Neural Netw.*, 18(8):1051–1063, 2005.

Inference, Learning, and Laws of Nature

S. Frandina, M. Gori, M. Lippi, M. Maggini, S. Melacci
Department of Information Engineering and Mathematical Sciences
University of Siena, Italy
{frandina,marco,lippi,maggini,mela}@diism.unisi.it

Abstract

Although inference and learning arise traditionally from different schools of thought, in the last few years they have been framed in nice unified frameworks, in the attempt to resemble clever human decision mechanisms. In this paper, however, we support the position that a true understanding of human-based inference and learning mechanisms might arise more naturally when replacing the focus on logic and probabilistic reasoning with that of cognitive laws, in the spirit of most variational laws of Nature. To this end, we propose a strong analogy between learning from constraints and analytic mechanics, which suggests us that agents living in their own environment obey laws exactly like those of particles subjected to a force field.

1 Introduction

Inference and learning have always been the subject of curiosity and in-depth investigations in the attempt to unveil their secret and grasp their meaning. As a truly manifestation of human being, they have been studied by philosophers, logicians, psychologists, as well as by scientists in artificial intelligence. Interestingly, while inference has been early framed into logic formalisms, the process of learning has been mostly attacked by statistical approaches. Nowadays, the marriage of these methodologies has been offering nice theoretical interpretations of either inference or learning which, amongst other relevant results, leads to foundations on probabilistic reasoning. Beginning from seminal studies at the end of the Eighties (see e.g. [Pearl, 1988]), nowadays there is a huge literature in the field, from which we can see significant theoretical and experimental advances. Related studies on bridging symbolic and sub-symbolic representations in neural networks have been developing under a remarkable variety of methodologies (see e.g. the Workshop series on Neural-Symbolic Learning and Reasoning `http://www.neural-symbolic.org/`). This lack of methodological focus seems to indicate that neural symbolic integration is still looking for strongly unifying approaches, driven by solid mathematical foundations like for probabilistic reasoning. On the other side, the studies on neural symbolic integration have been opening the mind to an in-depth re-thinking of inference and learning, that is well outside the borders of probability theory.

Beginning from the biological inspiration of neural network-based intelligent agents, in this paper we support the position that a natural integration of learning and inference arises when formulating the problem within the context of intelligent agents interacting on-line with the environment under the realm of cognitive laws. As time goes by, in such an environment, the agent is giving stimuli expressed in terms of constraints amongst a set of tasks and reacts by following laws emerging from the stationary point of a functional referred to as the *cognitive action*. This is strongly inspired from analytic mechanics, where the notion of particles subjected to a force field is doomed to follow the minimization of the action functional. When considering agents embedded in their environment, we naturally associate the weights of the neural network with the coordinates of the particles, the loss related to the constraints with the potential energy, and the sum of the squares of the derivative of the weights with the kinetic energy. This leads us to study the life of the agent by means of the elegant Lagrangian and Hamiltonian frameworks. However, in order to fully grasp the above links, we need to extend these formalisms with the insurgence of dissipative processes. Basically, a newborn agent begins its life with a certain potential energy and continues by changing its parameters, thus transforming that energy into kinetic energy and by dissipating the rest. As time goes by, the velocity of the weights decreases until the agent ends into a stable configuration, where all the initial potential energy is dissipated. This results in a learning mechanism paired with an inferential process, which improves as time goes. In the next section we discuss how to bridge logic and perception, while in Section 3 we propose the laws of learning as they come out from stationary point of the cognitive action. In Section 4 we show that the weights evolve according to general energy conservation principles and, finally, in Section 5 we give a perspective view on life-long learning build up on the proposed theory.

2 Bridging logic and perception

To sketch the idea, let us consider the following example. Let $x, y \in \mathbb{R}$ and let us assume that we are given some information on functions

$$A : \mathbb{R} \rightarrow \{0, 1\}$$

$$B:\ I\!R^2 \rightarrow \{0,1\}$$

defined as follows

$$\begin{aligned} \forall x \in [0,1] : A(x) = true,\ \ \forall x \notin [0,1] :\ A(x) = false \quad (1)\\ \forall (x,y) \in [0,1]^2 :\ B(x,y) = true,\\ \forall (x,y) \notin [0,1]^2 :\ B(x,y) = false. \end{aligned}$$

Now suppose that an oracle gives the intelligent agent the following piece of knowledge

$$\forall x \forall y\ A(x) \wedge A(y) \Rightarrow B(x,y), \tag{2}$$

Let $a:\ I\!R \rightarrow [0,1]$ and $b:\ I\!R^2 \rightarrow [0,1]$ be real-valued functions associated with $A(\cdot)$ and $B(\cdot,\cdot)$, respectively. Now, suppose that an intelligent agent is living on the temporal horizon $[0,t_e]$, with $t_e > 0$. Then, we can associate the granule of knowledge (2) with

$$\mathcal{V}_1(a,b) = \int_0^{t_e} a(x(t)) \cdot a(y(t)) \left(1 - b(x(t),y(t))\right) dt,$$

which needs to be as small as possible if we want to approximate (2). Now, let us assume that the agent acquires also the additional supervised pairs $\{(x_\kappa, d^a_\kappa)\}_{\kappa=1}^{\ell_a}$ and $\left\{((x_\kappa,y_\kappa), d^b_\kappa)\right\}_{\kappa=1}^{\ell_b}$ of $A(\cdot)$ and $B(\cdot,\cdot)$. They come at time $\{t^a_\kappa\}_{\kappa=1}^{\ell_a}$ and $\{t^b_\kappa\}_{\kappa=1}^{\ell_b}$, respectively. Given $c_1 > 0$ and $c_2 > 0$, the process of learning does require to control the functional

$$\mathcal{V}(a,b) := c_1\mathcal{V}_1(a,b) + c_2\mathcal{V}_2(a,b)$$

where

$$\begin{aligned} \mathcal{V}_2(a,b) \quad := \quad & \int_0^{t_e} \sum_{\kappa=1}^{\ell_a} h(a(x_\kappa), d^a_\kappa) \cdot \delta(t - t^a_\kappa)\ dt \\ + \quad & \int_0^{t_e} \sum_{\kappa=1}^{\ell_b} h(b(x_\kappa,y_\kappa), d^b_\kappa) \cdot \delta(t - t^b_\kappa)\ dt, \end{aligned}$$

and h is a loss functions (e.g. the hinge loss). This way of bridging logic and learning has been properly formalized for the case of FOL in [Diligenti *et al.*, 2012] using kernel-based representations for the functions.

Beginning from this example, let us make the assumption that functions a and b each depend on a corresponding vector of weights $w_a \in I\!R^{m_a}$ and $w_b \in I\!R^{m_b}$, respectively. Notice that this is different with respect to the kernel-based solution of [Diligenti *et al.*, 2012], but most of the concepts are the same. In order to gain a general formulation, from now on, let $x \in X \subset I\!R^n$ and $f : X \times W \rightarrow I\!R^q$ be the notation for a multitask system in which q different interacting tasks transform inputs from space X using weights in $W \in I\!R^m$. Now, we like to think of f as a neural network which, given the input $x \in X$ returns $f(x,w)$. We can promptly see that there exists V such that

$$\mathcal{V}(a,b) = \mathcal{V}(f) = \int_0^{t_e} V(w(t))dt, \tag{3}$$

Links with Analytic Mechanics		
var.	mach. learn.	mechanics
w_i	weight	particle
$\dot{w}_i$	weight variation	particle velocity
V	constraint penalty	potential energy
T	temporal smoothness	kinetic energy

Table 1: Links between machine learning and analytic mechanics.

where

$$\begin{aligned} V(w(t)) \quad := \quad & c_1 a(x(t)) \cdot b(y(t)) \left(1 - b(x(t),y(t))\right) \\ + \quad & c_2 \sum_{\kappa=1}^{\ell_a} h(a(x_\kappa), d^a_\kappa) \cdot \delta(t - t^a_\kappa) \\ + \quad & c_2 \sum_{\kappa=1}^{\ell_b} h(b(x_\kappa,y_\kappa), d^b_\kappa) \cdot \delta(t - t^b_\kappa). \end{aligned}$$

Of course, no matter what constraints are presented during the agent's life, the equation (3) holds true when choosing the appropriate *potential energy*.

3 Lagrangian cognitive laws

We propose a unified on-line formulation of learning and inference by introducing concepts that are tightly connected with analytic mechanics (see Table 1 for a summary of connections). Interestingly, the given formulation is somewhat inspired to different ways of introducing dissipation in classic Hamiltonian systems (see e.g. [Wang and Wang, 2012; Morris, 1986; Baldiotti *et al.*, 2010; Sanjuan, 1995]). We define *cognitive action* as the functional

$$\mathcal{S} = \int_0^{t_e} \mathcal{L}_\beta\ dt = \int_0^{t_e} e^{\beta t}\mathcal{L}\ dt \tag{4}$$

where $\beta > 0$,

$$\mathcal{L}(w) = T(w) - V(w) \tag{5}$$

is the Lagrangian, and

$$T = \frac{1}{2}\sum_{i=1}^{m} \mu_i \dot{w}_i^2(t), \tag{6}$$

is the *cognitive kinetic energy*, where $\mu_i > 0$ is the *cognitive mass* associated with the particle i. The learning process consists of finding

$$w = \arg\min_{w \in W} \mathcal{S}(w). \tag{7}$$

This is a classical problem in variational calculus. Any stationary point of $\mathcal{S}$ satisfies the Euler-Lagrange equations [Giaquinta and Hildebrand, 1996]

$$\frac{d}{dt}\frac{\partial \mathcal{L}_\beta}{\partial \dot{w}_i} - \frac{\partial \mathcal{L}_\beta}{\partial w_i} = 0.$$

When considering that $\mathcal{L}_\beta = e^{\beta t}\mathcal{L}$ we get

$$\beta e^{\beta t}\frac{\partial \mathcal{L}}{\partial \dot{w}_i} + e^{\beta t}\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{w}_i} - e^{\beta t}\frac{\partial \mathcal{L}}{\partial w_i} = 0.$$

Because of the definition of the Lagrangian (5), we get

$$\ddot{w}_i + \beta \dot{w}_i + \mu_i^{-1} V'_{w_i} = 0, \tag{8}$$

where $i = 1, \ldots, m$. When adding Cauchy's conditions, that correspond with setting $w_i(0)$ and $\dot{w}_i(0)$, the above Lagrangian cognitive equations drive the evolution of the agent. It can be proven that when we enforce a strong dissipation (high values of β) the above equation yields the classic online Backpropagation algorithm, being $\eta_i = 1/(\beta\mu_i)$ the learning rate [Frandina *et al.*, 2013]. Interestingly, the learning rate turns out to be small for large cognitive masses, which nicely matches the intuition that large masses move slowly. In addition, as already pointed out, for this classic connection to arise, we need to use values of β that are large enough. In the next section, this is given a foundation using the Hamiltonian framework extended to dissipation.

4 A dissipative Hamiltonian framework

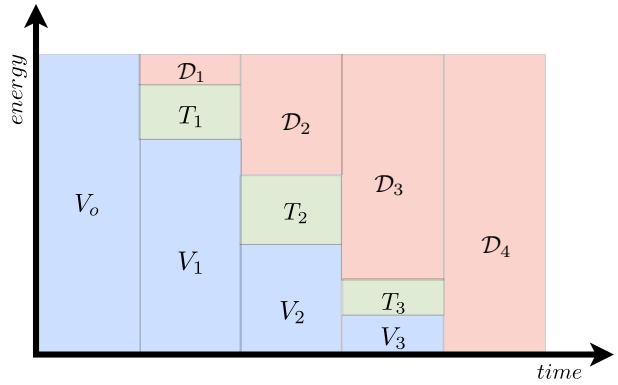


Figure 1: A sampling of energy balance. The initial value of the potential energy (inference loss) is transformed into kinetic energy and is dissipated. At the end, all the initial inference loss is dissipated.

We can start giving an interpretation of the agent evolution as follows. From equation (8) we have

$$\dot{w}_i \cdot \ddot{w}_i + \beta_i \dot{w}_i^2 + \mu_i^{-1} V'_{w_i} \cdot \dot{w}_i = 0$$

from which

$$\sum_{i=1}^{m} \frac{1}{2}\mu_i \int_0^{t_e} \frac{d}{dt}\dot{w}_i^2 dt + \sum_{i=1}^{m} \mu_i \int_0^{t_e} \beta \dot{w}_i^2 dt + \sum_{i=1}^{m} \int_0^{t_e} V'_{w_i} \dot{w}_i dt = 0.$$

Of course, we have

$$\frac{dV(w(t))}{dt} = \sum_{i=1}^{m} V'_{w_i} \dot{w}_i + \frac{\partial V}{\partial t}.$$

Then, let us define

$$D = \sum_{i=1}^{m} \beta \dot{w}_i^2 \tag{9}$$

and

$$\mathcal{D}(t) = \int_0^t D(w(\theta)) d\theta. \tag{10}$$

This is the *dissipated energy* over $[0, t_e]$. When considering definition 6, we get

$$\int_0^{t_e} \left(\frac{dT(w(t))}{dt} + \frac{dV(w(t))}{dt} + \frac{d\mathcal{D}(t)}{dt} \right) dt = \int_0^{t_e} \frac{\partial V}{\partial t} dt.$$

In case $\frac{\partial V}{\partial t} = 0$ we end up into the following *principle of conservation of cognitive energy*

$$\frac{d(T + V + \mathcal{D})}{dt} = 0 \tag{11}$$

that is the *cognitive energy*

$$\mathcal{E} = T + V + \mathcal{D}$$

is conserved during the agent's life. In Fig. 1 we can quickly grasp the meaning of the energy invariance. Basically, as time goes by, the potential V is partly transformed into kinetic energy and is partly dissipated. It is easy to see that the kinetic energy vanishes as $t_e \to \infty$. If, by contradiction $\lim_{t_e \to \infty} T(w(t)) > K > 0$ then, there exists $\bar{t}$ such that

$$\begin{aligned} \mathcal{D} &= \int_0^{t_e} \beta \dot{w}^2 dt > \int_{\bar{t}}^{t_e} \beta \dot{w}^2 dt \\ &> K(t_e - \bar{t}) \end{aligned}$$

and, therefore $\lim_{t_e \to \infty} \mathcal{D} = \infty$, from which we conclude that the kinetic energy vanishes necessarily. If the potential energy (loss term) is time-dependent (injection of stimulus) then we have the more general conservation law

$$\frac{d\mathcal{E}}{dt} = \frac{\partial V}{\partial t}. \tag{12}$$

This tells us that the cognitive energy $\mathcal{E}$ is constant whenever there is no injection of stimuli, but as the agent reacts to a new constraint there is change of cognitive energy, which is partially dissipated and transformed into kinetic energy.

INFERENCE

The proposed framework places learning and inference in exactly the same framework. In the example given in Section 2, the given constraints are properly expressed by the corresponding potential energy and the weights (particle position) evolves following the general conservation principle of equation (12). Now, suppose that we communicate to the agent the additional granule of knowledge

$$\forall x, \forall y : \; C(x) \wedge C(y) \Rightarrow D(x, y)$$

and that we want the agent to infer the truth of predicate

$$(\neg B(x, y) \wedge D(x, y)) \vee (B(x, y) \wedge \neg D(x, y)) \Rightarrow A(x).$$

We can convert this predicate to the correspondent real-valued function as shown in Section 2 and check it accordingly [1]. Of course, while some constraints are given and

[1] You can perform this kind of learning and inference on batch-mode using the simulator at `https://sites.google.com/site/semanticbasedregularization/home/software`.

are responsible of the learning dynamics (12), others are only used for inference. This is somewhat coherent with the scheme proposed in ([Diligenti *et al.*, 2012]). A detailed description of the inferential mechanisms is shown in [Gori and Melacci, 2013], even though kernel based representations are used instead of neural-like models like those considered in this paper. However, the most remarkable difference is that the learning process described in this paper takes place fully on-line. This way of inferring new constraints is different to nowadays approaches of collective classification, since the inference relies on the developed weights exactly like in classic on-line backpropagation.

5 Discussion

The cognitive laws formulated in this paper can be thought of as a re-statement of variational approaches to learning, like the one recently proposed in [Gnecco *et al.*, 2013]. There is, however, a fundamental difference that involves the crucial role of time, which is in fact what gives rise to the unified on-line learning and inferential scheme. Now, as already pointed out, the stationary points of (8) are those for which $\nabla_w V = 0$, which clearly indicates the limitations of cognitive systems that are stressing the dissipation. As pointed out for the case of supervised learning, large values of β lead to a stochastic gradient descent, but this property holds in general. While this is a nice landing in a known planet, there is still the problem of local minima of the potential energy. Interestingly, equation (8) is a damping oscillator, which can get rid of suboptimal minima energy configuration when the dissipation parameter β is small. An additional role is played by the injection of noise, which leads to the Langevin equation. In particular, there is a nice duality with the dynamics of Brownian particles, which are active in the sense that they take up energy from the environment [Schweitzer, 2000; Ebeling and Schweitzer, 2001]. The interpretation by the Fokker equation might be very interesting to gain a probabilistic understanding of the learning process. Alternative intriguing connections arise when invoking the extension of the Cognitive Action as defined in this paper in the quantum framework as early pointed by R. Feyman in his Ph.D. thesis [Feyman, 1942]. Finally, regardless of the development of the idea sketched in this paper, the remarkable novelty is in the way we seek for optimal cognitive configurations by the proposed on-line scheme, which can be thought of as laws of Nature. While there is a weak connection with simulated annealing and related global optimization schemes, the equations (8) are truly embedded in the environment and, therefore, they lead naturally to on-line learning. This seems to fit the growing call for life-long learning models, where a truly intelligent agent should be capable of learning online from a lifetime of raw sensorimotor experience (see the AAAI-2011 Workshop on lifelong learning from sensorimotor experience).

Acknowledgments

The research has been supported under the PRIN 2009 grant "Learning Techniques in Relational Domains and Their Applications".

References

[Baldiotti *et al.*, 2010] M.C. Baldiotti, R. Fresneda, and D.M. Gitman. Quantization of the damped harmonic oscillator revisited. Technical report, Istituto de Fisica, Universidade de Sao Paulo, Caixa Postal 66318-CEP, 05315-970 Sao Paul, S.P., Brasil, 2010.

[Diligenti *et al.*, 2012] M. Diligenti, M. Gori, M. Maggini, and L. Rigutini. Bridging logic and kernel machines. *Machine Learning*, 86:57–88, 2012. 10.1007/s10994-011-5243-x.

[Ebeling and Schweitzer, 2001] W. Ebeling and F. Schweitzer. Active motion in systems with energy supply. In *Integrative Systems Approaches to Natural and Social Dynamics*, pages 119–142, Berlin, 2001. Springer.

[Feyman, 1942] R. P. Feyman. *Principles of Least Action in Quantum Mechanics*. Ph.D. thesis, Princeton University, 1942.

[Frandina *et al.*, 2013] S. Frandina, M Gori, M Lippi, M Maggini, and S. Melacci. Variational foundations of online backpropagation. Technical report, Department of Information Engineering and Mathematical Sciences, University of Siena, 2013.

[Giaquinta and Hildebrand, 1996] M. Giaquinta and S. Hildebrand. *Calculus of Variations I*, volume 1. Springer, 1996.

[Gnecco *et al.*, 2013] G. Gnecco, M. Gori, and M. Sanguineti. Learning with boundary conditions. *Neural Computation*, pages 1–78, 2013.

[Gori and Melacci, 2013] M. Gori and S. Melacci. Constraint verification with kernel machines. *IEEE Trans. on Neural Networks and Learning Systems*, 24(5):825–831, 2013.

[Morris, 1986] Philip J. Morris. A paradigm for joint hamilton and dissipative systems. *Physica D*, pages 410–419, 1986.

[Pearl, 1988] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kauffmann, San Francisco, California, 1988.

[Sanjuan, 1995] M.A.F. Sanjuan. Comments on the hamiltonian formulation for linear and nonlinear oscillators including dissipation. *Journal of Sound and Vibration*, 185 (4):734–736, 1995.

[Schweitzer, 2000] F. Schweitzer. Active motion of brownian particles. In *Stochastic Processes in Physics, Chemistry and Biology*, pages 97–106, Berlin, 2000. Springer, Lecture Notes in Physics, vol. 557.

[Wang and Wang, 2012] Q.A. Wang and R. Wang. Is it possible to formulate least action principle for dissipative systems? Technical report, arXiv, 2012.

Dual-Process Theories and Cognitive Architectures ·

Ron Sun
Rensselaer Polytechnic Institute, Troy, NY, USA
rsun@rpi.edu

Abstract

The distinction between intuitive and reflective thinking is arguably one of the most important distinctions in cognitive science. This talk will explore implicit processes (intuitive thinking) in a variety of tasks and domains, and their interaction with explicit processes (reflective thinking). A cognitive architecture will be used to addresses, in a mechanistic and process sense, such issues, including the relation, interaction, and competition between implicit and explicit processes. Such cognitive-psychological theories have serious implications for developing hybrid neural-symbolic models.

1 Introduction

The distinction between *intuitive* and *reflective* thinking has been, arguably, one of the most important distinctions in cognitive science. There are many dual-process theories (two-system views) out there, as they seem to have captured popular imagination nowadays. However, although the distinction itself is important, these terms have been somewhat ambiguous. Not much finer-grained analysis has been done, especially not in a precise, mechanistic, process-based way. In this article, towards developing a more fine-grained and comprehensive framework, I will adopt the somewhat better terms of implicit and explicit processes, and present a more nuanced view (centered on "cognitive architecture").

Given that there have already been an overwhelming amount of research on explicit processes ("reflective thinking"), it is important, in studying the human mind, to emphasize implicit processes (including intuition). I would argue that we need to treat them as an integral part of human thinking, reasoning, and decision-making, not as an add-on. Therefore, we need to explore implicit processes in a variety of tasks and domains, and their interaction with explicit processes (reflective thinking). A theoretical model will be presented that addresses, in a mechanistic, process-based way, such issues. Issues addressed will include different types of implicit processes, their relation to explicit processes, and their relative speeds in relation to explicit processes.

2 Some Background

There are many dual-process theories (two-system views) available. One such view was proposed early on in Sun (1994, 1995). In Sun (1994), the two systems were characterized as follows:

> "It is assumed in this work that cognitive processes are carried out in two distinct `levels' with qualitatively different mechanisms. Each level encodes a fairly complete set of knowledge for its processing, and the coverage of the two sets of knowledge encoded by the two levels overlaps substantially." (Sun, 1994)

That is, the two "levels" (i.e., two modules or two components) encode somewhat similar content. But they encode their content in different ways: Symbolic versus subsybmolic representations were used, respectively. Symbolic representation is used by explicit processes at one "level", and subsymbolic representation is used by implicit processes at another. Therefore, different mechanisms are involved at these two "levels". It was hypothesized in Sun (1994) that these two different "levels" can potentially work together synergistically, complementing and supplementing each other. This is, in part, the reason why there are these two levels.

However, some more recent two-system views are more contentious. For instance, a more recent two-system view (dual-process theory) was proposed by Kahneman (2003). The gist of his ideas was as follows: There are two styles of processing: intuition and reasoning. Intuition (or System 1) is based on associative reasoning, fast and automatic, involving strong emotional bonds, based on formed habits, and difficult to change or manipulate. Reasoning (or System 2) is slower, more volatile, and subject to conscious judgments and attitudes.

Evans (2003) espoused this view. According to him, System 1 is "rapid, parallel and automatic in nature: only their final product is posted in consciousness"; he also notes the "domain-specific nature of the learning". System 2 is "slow and sequential in nature and makes use of the central

working memory system", and "permits abstract hypothetical thinking that cannot be achieved by System 1"

But such claims seem simplistic. For one thing, intuition can be very slow (Helie and Sun, 2010; Bowers et al., 1990). For another, intuition can be subject to conscious control and manipulation; that is, it may not be entirely "automatic" (Berry, 1991; Curran and Keele, 1993; Stadler, 1995). Furthermore, intuition can be subject to conscious "judgment" (Libet, 1987; Gathercole, 2003). And so on.

To come up with more nuanced and more detailed characterization, it is important that we ask some key questions. For instance, for either type of processes, there can be the following relevant questions:

- How deep is the processing (in terms of precision, certainty, and so on)?
- How much information is involved (how broad is the processing)?
- How incomplete, inconsistent, or uncertain is the information available?
- How many processing cycles are needed considering the factors above?

And many other similar or related questions.

3 A Theoretical Framework

In order to sort out these issues and questions, below, I will present a theoretical framework that can potentially provide some clarity to these issues and questions. The framework is based on the CLARION cognitive architecture (Sun, 2002, 2003, 2014), viewed at a theoretical level, as a conceptual tool for theoretical interpretations and explanations (Sun, 2009).

The theoretical framework consists of a number of basic principles. The first major point in this theoretical framework is the division between procedural and declarative processes, which is rather uncontroversial (Anderson and Lebiere, 1998; Tulving, 1985). The next two points concerns the division between implicit and explicit processes. They are unique to this theoretical framework (and thus may require some justifications; see, e.g., Sun, 2012, 2014). The second major point is the division between implicit and explicit procedural processes (Sun et al., 2005). The third major point is the division between implicit and explicit declarative processes (Helie and Sun, 2010). Therefore, in this framework, there is a four-way division: implicit and explicit procedural processes and implicit and explicit declarative processes.

The divisions above may be related to some existing computational paradigms, for example, symbolic-localist versus connectionist distributed representation (Sun, 1994, 1995). As has been extensively argued before (Sun, 1994, 2002), the relatively inaccessible nature of implicit knowledge may be captured by distributed representation (Rumelhart et al., 1986), because distributed representational units are subsymbolic and generally not individually meaningful. This characteristic of distributed representation, which renders the representational form less accessible computationally, accords well with the relative inaccessibility of implicit knowledge in a phenomenological sense. In contrast, explicit knowledge may be captured by symbolic or localist representation, in which each unit is more easily interpretable and has a clearer conceptual meaning.

4. A Sketch of A Cognitive Architecture

Now that the basic principles have been enumerated, I will sketch an overall picture of the CLARION cognitive architecture itself, which is centered on these principles (without getting into too much technical details though).

CLARION is a generic "cognitive architecture"---a comprehensive model of psychological processes of a wide variety, specified computationally. It has been described in detail and justified on the basis of psychological data (Sun, 2002, 2003, 2014). CLARION consists of a number of subsystems. Its subsystems include the Action-Centered Subsystem (the ACS), the Non-Action-Centered Subsystem (the NACS), the Motivational Subsystem (the MS), and the Meta-Cognitive Subsystem (the MCS). Each of these subsystems consists of two "levels" of representations, mechanisms, and processes as theoretically posited earlier (see also Sun, 2002). Generally speaking, in each subsystem, the "top level" encodes explicit knowledge (using symbolic-localist representations) and the "bottom level" encodes implicit knowledge (using distributed representations; Rumelhart et al., 1986).

Among these subsystems, the Action-Centered Subsystem is responsible for procedural processes, that is, to control actions (regardless of whether the actions are for external physical movements or for internal mental operations) utilizing and maintaining procedural knowledge. Among procedural processes, implicit procedural processes are captured by MLP (i.e., Backpropagation networks; at the bottom level of the ACS within the cognitive architecture). Explicit procedural processes, on the other hand, are captured by explicit "action rules" (at the top level of the ACS).

The Non-Action-Centered Subsystem is responsible for declarative processes, that is, to maintain and utilize declarative (non-action-centered) knowledge for information and inferences. Among these processes, implicit declarative processes are captured by associative memory networks (Hopfield type networks or MLP). Explicit declarative processes are captured by explicit "associative rules".

The Motivational Subsystem is responsible for motivational dynamics, that is, to provide underlying motivations for perception, action, and cognition (in terms of providing impetus and feedback). Implicit motivational processes are comprised of drive activations, captured by MLP. Explicit motivational processes are centered on explicit goals.

The Meta-cognitive Subsystem is responsible for metacognitive functions; that is, its responsibility is to monitor, direct, and modify the operations of the other subsystems dynamically. Implicit metacognitive processes are captured by MLP, while explicit metacognitive processes are captured by explicit rules.

The two levels within each subsystem interact, for example, by cooperating in action decision-making within the ACS, through integration of the action recommendations from the two levels of the ACS, as well as by cooperating in learning (more later; Sun, 2002). See Figure 1 for a sketch of the CLARION cognitive architecture**.**

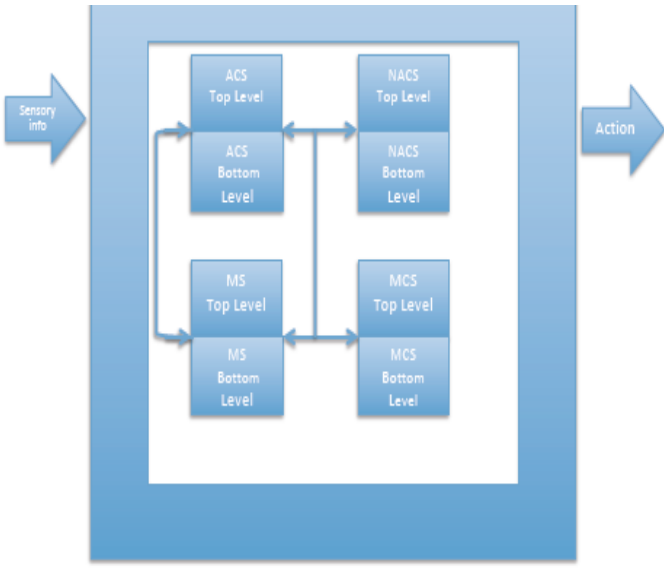


Figure 1. The four subsystems of CLARION.

5. Interpreting Psychological Notions

Based on the framework above (i.e., the CLARION cognitive architecture), we may re-interpret some folk psychological notions, to hopefully give them some clarity.

For instance, the notion of *instinct* may be made more precise by appealing to the framework of a cognitive architecture. Instinct involves mostly implicit processes and is mostly concerned with action. Within CLARION, instinct may be roughly equated with the following chain of activation: stimuli → drive → goal → action. This chain goes from stimuli received to the MS, the MCS, and eventually the ACS. That is, stimuli activate drives (especially those representing innate motives), drive activations lead to goal setting in a direct, implicit way (mostly innate), and based on the goal set, actions are selected in an implicit way to achieve the goal. Instinct is mostly implicit, but it may become more explicit, especially with regard to the part of "goal → action" (Sun et al., 2001).

For another instance, the notion of *intuition* can also be made more precise by using the CLARION framework. Intuition, according to CLARION, is roughly the following chain: stimuli → drive → goal → implicit reasoning. This chain goes from stimuli received to the MS, the MCS, and the NACS. As such, intuition mostly involves implicit declarative processes within the NACS, including the functionality of associative memory retrieval, soft constraint satisfaction, and partial pattern completion. Intuitive processes are often complementary to explicit reasoning, and the two types are used often in conjunction with each other (Helie and Sun, 2010).

Some other folk psychological notions may be re-interpreted and made more precise in a similar manner. For example, the notion of *creativity* may be captured within the CLARION framework. Creativity may be achieved through complex, multi-phased implicit-explicit interaction, that is, through the interplay between intuition and explicit reasoning, according to Helie and Sun's (2010) theory of creative problem solving---a theory derived from the CLARION cognitive architecture. It is a 3-phase model, which includes (1) the explicit phase: process given information; (2) the implicit phase: develop intuition using implicit declarative knowledge; finally the intuition emerge into explicit processes and therefore (3) the explicit phase: verify and validate the result using explicit declarative knowledge. See Helie and Sun (2010) for further details. This theory has been successful in accounting for a variety of empirical data.

What about the competition among these different types of processes, especially in terms of their relative speeds (time courses) as alluded to earlier? There was a question raised earlier concerning fast versus slow processes with regard to different dual-process theories (two-systems views). The twin divisions in CLARION, procedural versus declarative and implicit versus explicit, definitely have implications for identifying slow versus fast processes across the systems (components). For instance, we may question conventional wisdom on a number of issues in this regard (instead of simply assuming the seemingly obvious as in some of the existing views/theories):

- In terms of the division between procedural and declarative processes, can fast procedural versus slow declarative processes be posited?
- In terms of the division between implicit and explicit procedural processes, can fast implicit versus slow explicit processes be posited?
- In terms of the division between implicit and explicit declarative processes, can fast implicit versus slow explicit processes be likewise posited?
- What about relative speeds if we consider the four-way division together?

And so on. These conjectures implied by the questions above may be true to some extent, but not exactly accurate. The whole picture is not so simple.

In this regard, we may view existing models and simulations of these different types of processes as a form of theoretical interpretation concerning their time courses. In that case, we have the following potential answers:

- Fast procedural versus slow declarative: Fortunately, this is generally true if we examine many existing models and simulations (Anderson and Lebiere, 1998; Sun, 2003, 2014).
- Fast implicit versus slow explicit procedural processes: It is, again, generally true, using theoretical interpretations through modeling and simulation (Sun et al., 2001, 2005).

- Fast implicit versus slow explicit declarative processes: This, however, is generally not true. Intuition (implicit declarative processes) may (or may not) take a long time, compare with explicit declarative processes. See, for example, Helie and Sun (2010) and Bowers et al. (1990).

We need to be careful in making sweeping generalizations. Often, we need to characterize different types of processes in a more fine-grained fashion. Characteristics of different processes may also vary in relation to contexts such as task demands.

A number of empirical and simulation studies have been conducted within the framework of CLARION that shed light on these issues, and substantiate the points made above. See, for example, Sun et al. (2009), Sun and Zhang (2006), Helie and Sun (2010), and so on.

Dual-process theories (two-system views) may arguably have significant implications for neural symbolic processing and especially hybrid neural-symbolic systems. For one thing, they may serve as the theoretical basis and the justifications for some forms of hybrid neural-symbolic models that juxtapose symbolic and subsymbolic components. If cogniitive-psychological realism is what one wants to achieve in developing such models, dual-process theories must be taken into serious consideration, and be used as guides in developing such models. See, for example, Sun (2002, 2006, 2014).

Acknowledgments

The work overviewed here has been supported in part by the ONR grants N00014-08-1-0068 and N00014-13-1-0342. Thanks are due to my students and collaborators, past and present, whose work has been reviewed here. The CLARION software, along with simulation examples, may be found at: www.clarioncognitivearchitecture.com (courtesy of Nick Wilson and Mike Lynch).

References

J. R. Anderson and C. Lebiere, (1998). *The Atomic Components of Thought*. Lawrence Erlbaum Associates, Mahwah, NJ.

D. Berry, (1991). The role of action in implicit learning. *Quarterly Journal of Experimental Psychology*, 43A, 881-906.

K. Bowers, G. Regehr, C. Balthazard, and Parker, (1990). Intuition in the context of discovery. *Cognitive Psychology*. 22. 72-110.

T. Curran and S. Keele, (1993). Attention and structure in sequence learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition.* 19, 189-202.

J. St.B.T. Evans, (2003). In two minds: dual-process accounts of reasoning, *Trends in Cognitive Sciences*, 7 (10), 454-459.

S. Gathercole, (2003). *Short-term and Working Memory*. Taylor and Francis, London.

S. Helie and R. Sun, (2010). Incubation, insight, and creative problem solving: A unified theory and a connectionist model. *Psychological Review*, 117, 994-1024.

D. Kahneman (2003). A perspective on judgment and choice: mapping bounded rationality. Am Psychol. 58(9):697-720.

B. Libet, (1985). Unconscious cerebral initiative and the role of conscious will in voluntary action. *Behavioral and Brain Sciences*. 8, 529-566.

D. Rumelhart, J. McClelland and the PDP Research Group, (1986). *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, MIT Press, Cambridge, MA.

M. A. Stadler, (1995). Role of attention in implicit learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 21,* 674-685.

R. Sun, (1994). *Integrating Rules and Connectionism for Robust Commonsense Reasoning.* John Wiley and Sons, New York.

R. Sun, (1995). Robust reasoning: Integrating rule-based and similarity-based reasoning. *Artificial Intelligence* (AIJ). Vol.75, No.2, pp.241-296.

R. Sun, (2002). *Duality of the Mind: A Bottom-up Approach Toward Cognition*. Mahwah, NJ: Lawrence Erlbaum Associates.

R. Sun, (2003). *A Tutorial on CLARION 5.0.* Technical Report, Cognitive Science Department, Rensselaer Polytechnic Institute. http://www.cogsci.rpi.edu/ rsun/sun.tutorial.pdf

R. Sun, (2006). *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation*. Cambridge University Press, New York.

R. Sun, (2009). Theoretical status of computational cognitive modeling. *Cognitive Systems Research*, 10 (2), 124-140.

R. Sun, (2014). *Anatomy of Mind*. Oxford University Press, New York. In press.

R. Sun, Merrill, E., & Peterson, T. (2001). From implicit skills to explicit knowledge: A bottom-up model of skill learning. *Cognitive Science, 25*, 203-244.

R. Sun, Slusarz, P., & Terry, C. (2005). The interaction of the explicit and the implicit in skill learning: A dual-process approach. *Psychological Review, 112*, 159-192.

R. Sun and X. Zhang, (2006). Accounting for a variety of reasoning data within a cognitive architecture. *Journal of Experimental and Theoretical Artificial Intelligence*, Vol.18, No.2, pp.169-191.

R. Sun, X. Zhang, and R. Mathews (2009). Capturing human data in a letter counting task: Accessibility and

action-centeredness in representing cognitive skills . *Neural Networks*, Vol.22, pp.15-29.

R. Sun, X. Zhang, P. Slusarz, and R. Mathews, (2007). The interaction of implicit learning, explicit hypothesis testing learning, and implicit-to-explicit knowledge extraction. *Neural Networks*, Vol.20, No.1, pp.34-47.

E. Tulving, (1985). How many memory systems are there? *American Psychologist*, 40,385–398.

Knowledge Extraction from Deep Belief Networks for Images

Son N. Tran
City University London
Northampton Square, EC1V 0HB, UK
Son.Tran.1@city.ac.uk

Artur d'Avila Garcez
City University London
Northampton Square, EC1V 0HB, UK
aag@soi.city.ac.uk

Abstract

In this paper, we introduce a rule extraction method for discovering knowledge in a set of images using Deep Belief Networks. The method is shown to be useful at pruning the networks trained on image datasets, explaining the relationship between a label and the data distribution, and training the networks on related image datasets. Deep architectures have been studied intensively recently thanks to their applicability on different areas of Machine Learning. Considerable success has been achieved in a number of application areas, notably in image, video and multimodal classification and retrieval tasks. We argue that knowledge extraction from deep networks can provide further improvements in specific areas and a wider applicability of such networks through a better understanding and explanation of the relations between network layers. To the best of our knowledge, this is the first knowledge extraction system for deep networks. Keywords: Deep Belief Networks, Neural-Symbolic Integration, Knowledge Extraction, Vision and Perception.

1 Introduction

Recent publications indicate the emergence of two interesting sub-areas of Machine Learning: deep networks [Lee *et al.*, 2009b], [Lee *et al.*, 2009a] and neural-symbolic systems [Borges *et al.*, 2011], [Aha *et al.*, 2010]. While the former have been shown capable of learning and representing features effectively in different application areas, the latter has successfully proved a strong relation between logic and connectionist systems. Deep architectures have been studied intensively recently thanks to their applicability to different areas of Machine Learning [Bengio, 2009]. Considerable success has been achieved in a number of application areas, notably in image, video and multimodal classification and retrieval tasks [Ji *et al.*, 2010]. Following a neural-symbolic approach, we argue that knowledge extraction from deep networks can provide further improvements in specific problem domains and a wider applicability of such networks through a better understanding and explanation of the relations between network layers. Hence, in this paper, we focus attention on the problem of knowledge extraction [d'Avila Garcez *et al.*, 2001] from deep networks. We start by proposing a method for knowledge extraction from Restricted Boltzmann Machines (RBMs) [Smolensky, 1986], which can be seen as the basic building block of deep networks. We then unroll this method into an approach for extracting logical representations from Deep Belief Networks (DBN). We present a new extraction algorithm and experimental results on pruning the networks trained on image datasets, explaining the relationship between a label and the data distribution, and transferring the extracted knowledge to train the networks on related image datasets. To the best of our knowledge, this is the first knowledge extraction system for deep networks. The approach is inspired by and extends the idea of penalty logic [Pinkas, 1995], which was applicable originally to shallow symmetrical networks, and applied in modified form recently to recurrent temporal restricted Boltzmann machines [de Penning *et al.*, 2011]. We expect knowledge extraction to serve as a tool to help analyse how representations are built in the different layers of the network, and in the study of the relationship between the depth of DBNs and performance. In a nutshell, this paper shows that, given a trained DBN, logical rules of the forms $h_j \leftrightarrow v_1 \wedge \neg v_2 \wedge v_3$ can be extracted from each pair of layers, each rule having a *confidence* value (defined in what follows). Here, the proposition h_j represents the activation state of a unit j in a layer, and the proposition $v_i (i = 1, 2, 3)$ represents the activation state of a unit i in a layer immediately below it (hence, v_i denotes that i is activated, and $\neg v_i$ denotes that i is deactivated). Rules can be chained down all the way to the visible layer so that the extracted rule set can be compared with the DBN in terms, for example, of the images that they generate (experiments section below). The rules can be inspected or queried taking into consideration any subset of the set of layers, hopefully shedding light into the structure of the network and the role of each layer/set of neurons. This will be visualized in the case of a handwritten digit case study. Also, the chaining between layers can be analysed more explicitly with the use of the rules' confidence values. The remainder of the paper is organised as follows. In Section 2, we introduce the relevant background on DBNs and penalty logic. In Section 3, we introduce an extraction method for RBMs. In Section 4, we

introduce the extraction approach and algorithm for DBNs. In Section 5, we present and discuss the experimental results, and in Section 6 we conclude and discuss directions for future work.

2 Background

In this section, we define the notation used in the paper and briefly recall the constructs of DBNs and penalty logic used later by the extraction algorithm.

A DBN is constructed by stacking several restricted Boltzmann machines (RBMs) [Smolensky, 1986]. Theoretically, this can be done by adding complementary prior probabilities so that the joint distribution will lead to a factorial posterior [Hinton *et al.*, 2006]. In a two-layered graphical model, if the joint distribution is of the form $P(x,y) = \frac{1}{C}\exp(\sum_{i,j}\Psi_{i,j}(x_i,y_j) + \sum_i \gamma_i(x_i) + \sum_j \alpha_j(y_j))$ then there will exist complementary priors that make:

$$P(x|y) = \prod_i P(x_i|y)$$

$$P(y|x) = \prod_j P(y_j|x)$$

The joint distribution of an RBM with visible layer v and hidden layer h, when its global state reaches equilibrium, is exactly of the form above, more precisely:
$P(v,h) = \frac{1}{Z}\exp(-\sum_{i,j} v_i w_{ij} h_j - \sum_i a_i v_i - \sum_j b_j h_j)$.

The equilibrium state of an RBM can be achieved by Gibbs samplings for long iterations [Hinton *et al.*, 2006]. This process simulates the idea of creating an infinite stack of directed RBMs with fixed weights and then, from the very top layer, performing a down-pass sampling. The units in the bottom pair of layers would represent the equilibrium state of the corresponding RBM. In practice, a DBN has an undirected RBM at the top to simulate the upper-part infinite stack and several directed RBMs underneath.
Training a DBN starts from the lowest component model upwards, each component model (an RBM) is trained in the normal way using Contrastive Divergence [Carreira-Perpinan and Hinton, 2005]. This greedy training has been shown capable of approximating well enough the idea of complementary priors; however, to improve performance of DBNs as a classifier an additional back-propagation training phase is normally applied [Hinton *et al.*, 2006].
After training, each unit in the hidden layer is seen as a holder of features learned from the data. Higher layers in the hierarchy are expected to represent more concrete features such as edges, curves and shapes (in the case of image datasets as used in this paper). In [Hinton *et al.*, 2006], the exact reconstruction of the data from labels is tested by clamping a softmax unit and performing Gibbs sampling at the top layer followed by a down-pass through the network to the visible units. This "clamping" method samples visible states using the distribution $P(\mathbf{v}|h_j^l = 1)$. This is also used by [Erhan *et al.*, 2009] to visualize features in the network's second hidden layer, which is a form of "extraction". In [Erhan *et al.*, 2009], a visualization method is also introduced following the idea of finding visible states which maximize the activation of a hidden unit, $\overset{*}{\mathbf{v}} = \arg\max_{\mathbf{v}} P(h_j^l = 1|\mathbf{v})$.

In [Pinkas, 1995], symmetric connectionist systems characterised by an energy function have been shown equivalent to sets of pairs of logic formulae and real numbers $\{< p_i, f_i >\}$. The real number p_i is called a "penalty" and the set of pairs of formulas is known as a *penalty logic well-formed formula* or PLOFF. Here, extraction is explicit in that it produces symbolic knowledge rather than visualizations. In [Pinkas, 1995], a connectionist system and a rule set are said to be equivalent if and only if there exists a ranking function V_{rank} for the latter such that: $V_{rank}(\vec{x}) = E(\vec{x}) + c$, where c is a constant and E is the energy function of the system. In the case of an RBM, this can be defined as:
$E(h,v) = -\sum_{i,j} v_i w_{ij} h_j - \sum_i a_i v_i - \sum_j b_j h_j$
with the corresponding extracted rules being:
$\{\langle w_{ij}, v_i \wedge h_j\rangle | w_{ij} > 0\} \cup \{\langle -w_{pq}, \neg(v_p \wedge h_q)\rangle | w_{pq} < 0\}$
and with ranking function the sum of the penalties of the pairs whose formulae are violated given truth-values for $\mathbf{h}, \mathbf{v}$. Since RBMs are also symmetric connectionist systems, in what follows we present an extension of the penalty logic approach to rule extraction that is suitable for DBNs.

3 Knowledge Extraction from RBMs

In this section, we apply and modify the idea of penalty logic to extract knowledge from an RBM model. However, for reasons that will become clear in what follows, instead of rewriting the model as a set of formulas associated with a ranking function, a set of rules will be extracted and used for inference to evaluate how closely the rules represent the behaviour of the network model.

The propositional knowledge captured from an RBM $N = \{V, H, W\}$ can be represented as:

$$R = \{\langle w_{ij} : v_i \wedge h_j\rangle | w_{ij} > 0\} \cup \{\langle -w_{pq} : \neg(v_p \wedge h_q)\rangle | w_{pq} < 0\}$$

The extracted set of rules will capture the behaviour of an RBM, as has been discussed in the previous section. However, they are not as capable of expressing the relationships between the features represented by units in the RBM's hidden layer and raw data encoded in its visible layer (e.g. an image). Moreover, they are not capable of representing relationships among input variables. Finally, the penalty logic approach would extract $N_v \times N_j$ rules from an RBM, which is a concern in the case of a model with thousands of units such as the RBMs or DBNs used for image processing. We propose an alternative based on the concept of partial and complete model, as below.

Proposition 3.1. *For each unit h_j in the hidden layer of an RBM, it is possible to extract a single rule in the weighted-if-and-only-if (weighted-IFF) form:*

$$c_j : h_j \leftrightarrow \bigwedge_i v_i \wedge \bigwedge_t \neg v_t$$

where v_i and v_t correspond to the activated unit i and deactivated unit j in the visible layer. The rule's confidence-value is given by $c_j = \sum_{v_i} w_{ij} - \sum_{v_t} w_{tj}$

Definition 3.1. *(partial-model): A partial-model of a unit h_j in the hidden layer of an RBM is a weighted-IFF rule of the form:*

$$c_j : h_j \leftrightarrow \bigwedge_{i,w_{ij}>0} v_i \wedge \bigwedge_{t,w_{tj}<0} \neg v_t$$

Proposition 3.2. *The partial-model of a unit in the hidden layer of an RBM is the rule with the highest confidence-value possibly extracted from that unit.*

However, it is difficult to use the *partial-model* for inference because it does not capture information about the weight values between the set of input units and hidden unit h_j in the rule. This problem can be solved by having a complete model, as follows.

Definition 3.2. *(complete-model): A complete-model for a unit h_j in an RBM is a* partial-model *associated with a list of the absolute values of the weights between the units corresponding to conjuncts in the rule and h_j. It can be denoted as:*

$$c_j : h_j \overset{hv_j}{\leftrightarrow} \bigwedge_{i,w_{ij}>0} v_i \wedge \bigwedge_{t,w_{tj}<0} \neg v_t$$

with $hv_j = \{|w_{1j}|, |w_{2j}|, ..., |w_{N_v j}|\}$

In what follows, we investigate the pros and cons of partial-models and complete-models at representing RBMs. We start with an example.

Example (learning XOR function): We examine the theory by training an RBM to model a XOR function from its truth-table, where $true$ and $false$ are represented by 1 and 0, respectively. Our goal is to compare the rules extracted from trained RBMs having inputs x, y, z with different numbers of hidden units, against the rules of the XOR function as shown in Table 1.

X	Y	Z	Rules
0	0	0	$\top \leftrightarrow \neg x \wedge \neg y \wedge \neg z$
0	1	1	$\top \leftrightarrow \neg x \wedge y \wedge z$
1	0	1	$\top \leftrightarrow x \wedge \neg y \wedge z$
1	1	0	$\top \leftrightarrow x \wedge y \wedge \neg z$

Table 1: XOR problem: Truth-table and rules defining the conditions for activating hidden unit $\top$. All other truth-value assignments not specified by the rules should not activate any hidden unit.

Training $z \leftrightarrow x \oplus y$ in an RBM should also imply $x \leftrightarrow y \oplus z$ and $y \leftrightarrow x \oplus z$, given the symmetry of the network. Three RBMs having, respectively, 1, 3 and 10 hidden units were trained using Contrastive Divergence and the same learning parameters. After training, partial-model rule sets were extracted, as shown in Table 2. Notice how the rules in Table 2 that do not match a rule in Table 1 have a much smaller confidence value. In fact, if we were to remove the rules with confidence value smaller than the mean confidence for each rule set, only the rules that match the description of the XOR function from Table 1 would have been kept in Table 2. It is interesting to see that partial models have been able to capture the correct relationships between the input variables in this example, despite the inherent loss of information of partial models. Complete-models, with their confidence vectors, should be able to represent the behaviour of the network almost exactly, but partial models do not contain weight values, as discussed earlier. In what follows, we continue this analysis by applying the definitions of partial and complete models to DBNs.

#hiddens	Extracted rules
1	$17.9484 : h_1 \leftrightarrow x \wedge y \wedge \neg z$✓
	$1.499 : \top \leftrightarrow \neg x \wedge \neg y \wedge z$
3	$24.7653 : h_1 \leftrightarrow \neg x \wedge y \wedge z$✓
	$23.389 : h_2 \leftrightarrow x \wedge y \wedge \neg z$✓
	$27.1937 : h_3 \leftrightarrow \neg x \wedge \neg y \wedge \neg z$✓
	$13.4209 : \top \leftrightarrow x \wedge \neg y \wedge z$✓
10	$4.0184 : h_1 \leftrightarrow x \wedge \neg y \wedge z$✓
	$8.9092 : h_2 \leftrightarrow x \wedge \neg y \wedge z$✓
	$18.4939 : h_3 \leftrightarrow \neg x \wedge y \wedge z$✓
	$0.5027 : h_4 \leftrightarrow \neg x \wedge y \wedge \neg z$
	$7.4417 : h_5 \leftrightarrow x \wedge \neg y \wedge z$ ✓
	$5.0297 : h_6 \leftrightarrow \neg x \wedge y \wedge z$ ✓
	$7.6313 : h_7 \leftrightarrow x \wedge \neg y \wedge z$ ✓
	$22.0653 : h_8 \leftrightarrow x \wedge y \wedge \neg z$✓
	$19.6188 : h_9 \leftrightarrow \neg x \wedge \neg y \wedge \neg z$✓
	$14.6054 : h_{10} \leftrightarrow \neg x \wedge \neg y \wedge \neg z$✓
	$3.4366 : \top \leftrightarrow x \wedge y \wedge z$

Table 2: Rules extracted from RBMs trained on the XOR function (✓ means the rule matches a rule in Table 1).

4 Knowledge Extraction from DBNs

Since the structure of a DBN can be seen as a stack of RBMs, the simplest method for extracting knowledge from a DBN is to combine the rules extracted from each RBM. Hence, the rules extracted from a DBN can be represented (using *partial-models*) as:

$$R = \{c_j^{(1)} : h_j^{(1)} \leftrightarrow \bigwedge_{i,w_{ij}^{(0)}>0} v_i \wedge \bigwedge_{t,w_{tj<0}^{(0)}} \neg v_t\}$$

$$\bigcup_{l=1}^{L-1} \{c_j^{(l+1)} : h_j^{(l+1)} \leftrightarrow \bigwedge_{i,w_{ij}^{(l)}>0} h_i^{(l)} \wedge \bigwedge_{t,w_{tj<0}^{(l)}} \neg h_t^{(l)}\}$$

In the case of *complete-models* each rule is also associated with a confidence-vector, as discussed in the previous section.

Even though DBNs are stochastic systems, our experiments have shown that deterministic inference on the extracted rules can capture the behaviour of the networks. Taking a pair $\langle w_{ij}, v_i \wedge h_j \rangle$ with $w_{ij} > 0$, it is clear that if v_i is activated ($v_i = 1$), h_j should also be activated with confidence value w_{ij} (since we are interested in minimizing the energy function). Similarly, for a pair $\langle -w_{ij}, \neg(v_i \wedge h_j) \rangle$ ($w_{ij} < 0$ in this case), if v_i is activated then the energy will decrease only if h_j is not activated ($h_j = 0$) with confident value $-w_{ij}$. Therefore:

$$h_j = \begin{cases} 1 & \text{if } v_i = 1 \text{ and } w_{ij} > 0 \\ 0 & \text{if } v_i = 1 \text{ and } w_{ij} < 0 \end{cases} \qquad \text{(Eq.1)}$$

with confidence value $|w_{ij}|$.

A confidence value is the dual of the penalty in [Pinkas, 1995] and it represents how reliable (or likely to activate) a unit is when another unit connected to it is already activated. However, it is not easy to determine the activation state of a unit when it appears in different rules with positive and negative weights, respectively. For example, let $\{\langle w_{ij}, v_i \wedge h_j\rangle | w_{ij} > 0\} \cup \{\langle -w_{ij}, \neg(v_i \wedge h_j)\rangle | w_{ij} < 0\}$ be a subset of the rules containing h_j, and $Sum_j = \sum_{i,v_i=1} w_{ij}$ be the sum of the weights of the connections between h_j and all activated units v_i. If $Sum_j > 0$, we say that the *positive* rules are more likely than the *negative* rules; hence, h_j is more likely to activate. We can check this by looking at the activation of h_j : $P(h_j = 1) = \frac{1}{1+\exp(-\sum_i w_{ij} v_i)} = \frac{1}{1+\exp(-Sum_j)}$, in which the larger Sum_j is, the higher the probability is of h_j being activated. From this, we can also see that if $Sum_j < 0$ then one should have less confidence in the activation of h_j, that is, the probability of h_j being activated should be low.

In general, the inference rule for our extracted rules, where $c, \alpha_1, \alpha_2, \ldots, \alpha_n, \alpha_h$ are confidence-values, should be:

$$
\begin{array}{l}
c : h \overset{w_1, w_2, \ldots, w_n}{\leftrightarrow} belief_1 \wedge \neg belief_2 \wedge \ldots \wedge belief_n \\
\alpha_1 : belief_1 \\
\alpha_2 : belief_2 \\
\vdots \\
\alpha_n : belief_n \\
\hline
\alpha_h : h \text{ with } \alpha_h = w_1\alpha_1 - w_2\alpha_2 + \ldots + w_n\alpha_n
\end{array}
$$

In the case of partial-models, $\alpha_h = c/n$.

When rules are extracted from a deep architecture, first one should apply the above inference rule to derive hypotheses for the first RBM in the stack, and then repeat the process, assuming that those derived hypothesis are beliefs, in order to infer new hypotheses for the second RBM in the stack, and so on. For this, the confidence value of a belief should be normalized to $[0, 1]$ using, e.g., a sigmoid function. Alternatively, one might set the confidence value of a belief to 1, if the confidence value of its associated hypothesis is positive, or 0 otherwise.

5 Experiments and Results

In this section, we validate empirically the connection between logical rules and stochastic sampling established above by extracting rules and visualizing features of Deep Belief Networks. We also present new results on transfer learning using rule extraction. We have trained a system with 784, 500, 500 and 2000 units over four layers on 5000 examples from the MNIST handwritten digit dataset [Hinton *et al.*, 2006]. Some of the rules extracted from the deep network are shown below [Son Tran and Garcez, 2012].

$$
\begin{array}{l}
0.99250 : h_0^2 \leftrightarrow \neg h_0^1 \wedge h_1^1 \wedge \neg h_2^1 \wedge \ldots \neg h_{783}^1 \\
0.99250 : h_0^1 \leftrightarrow \neg v_0^0 \wedge v_1^0 \wedge \neg v_2^0 \wedge \ldots \neg v_{783}^0 \\
1.00000 : h_1^1 \leftrightarrow \neg v_0^0 \wedge \neg v_1^0 \wedge \neg v_2^0 \wedge \ldots \neg v_{783}^0
\end{array}
$$

where the confidence values have been normalized by a sigmoid function, and h_j^l represents a unit j of hidden layer l.

Let us consider the use of logical inference, as discussed above, to try and visualize the features captured by the units in the second hidden layer. In this experiment, the confidence value of a belief b associated with a hypothesis $c : h$ is set to 1 if $c > 0$, and 0 otherwise. Figure 1 shows the progression of 10 images (from left to right) generated using network sampling (shown in the top row) and the above, sign-based, logical inference (shown in the bottom-row). The results indicate information loss, as expected, but also suggest that inference with sign-based activation should be faster than network sampling; the network might have several local minima for each concept, so that each image was generated after intervals of 10 iterations, totalling 100 iterations in the network). We then

Figure 1: Images generated from network samplings (top row) and logical inference using sign-based activation/confidence value calculations (bottom row).

added labels to the network for classification. The labels are added to the visible layer of the top RBM in the hierarchy as soft-max units [Hinton *et al.*, 2006]. After training on $60,000$ samples of the MNIST dataset, the network achieved 97.63% accuracy on a test set consisting of $10,000$ samples. If we apply logical inference using a sigmoid function to normalize the confidence values, the rules achieve 93.97% accuracy on the same test set. If we then clamp the labels and reconstruct the input images, it can be verified that logical inference can generate fairly similar images, as depicted in Figure 2, where the top row shows the images generated by the network using Gibbs sampling, and the bottom row shows the images produced by inference.

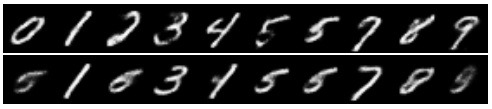


Figure 2: Images reconstructed by clamping labels of DBN and using Gibbs sampling (top row), and by setting hypotheses to *true* in the set of rules and using logical inference (bottom row).

5.1 System Pruning

The goal here is to use the confidence value of each rule to evaluate the contribution of its corresponding hidden unit in an RBM (or in the hidden layers of a DBN), removing from the system the "weak" hidden units, i.e. those with a low confidence value. We shall test the drop in network performance as a measure of usefulness of a rule's confidence value. Rules were extracted from an RBM trained on the MNIST dataset. Rules with confidence value smaller than a specified threshold were removed, and the remaining rules were re-encoded

into a smaller RBM. The performance of the new RBM can be evaluated by comparing the reconstruction of images with that done by the original RBM, as illustrated in Figure 3.

(a) (b) (c) (d)

Figure 3: The reconstructed images from original RBM trained on 60,000 samples (3a), RBM encoding 382 rules with highest confidence values (3b), RBM encoding 212 rules with highest confidence values (3c), and RBM encoding 145 rules with highest confidence values (3d).

Figure 3 indicates that it is possible to remove rules/hidden units and maintain performance, with the behaviour of the system remaining pretty much unchanged until some "important" rules start to be removed. In order to measure this more precisely, we have provided the features obtained from the RBMs as an input to an SVM classifier. Figure 4 shows the relationship between accuracy and the sizes of the RBMs. Interestingly, at first, accuracy has increased slightly, and has remained stable until some 50% of the hidden units had been removed. It then started to deteriorate rapidly, but was still at around 95% when 400 of the 500 hidden nodes were removed.

Figure 4: Classification performance of SVM on MNIST dataset with input features from RBM whose hidden layer is reduced by removing hidden units with lower confidence values.

5.2 Transferring Knowledge

Rule extraction in the context of images may find application in transfer learning [Torrey *et al.*, 2010]. To evaluate this, we have extracted rules from a network trained on an image dataset and encoded a subset of these rules on another network to be trained on a related image dataset. In particular, we have measured classification performance when transferring from the handwritten digit dataset to natural images of digits, and to writing styles, as shown in Figure 5. We have trained an RBM on 5,000 images from the MNIST dataset, extracted and pruned the rules before transferring them to RBMs that were trained on natural images of digits (ICDAR dataset[1]) and on character writing styles (TiCC dataset[Maaten, 2009]). Figure 5 shows some examples from the source and target domains.

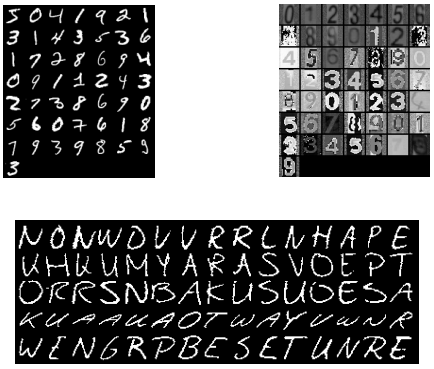


Figure 5: Visualization of MNIST images (top left), ICDAR images (top right), and TiCC character images (below) for five writers (one per line).

We started by extracting and transferring all complete-models. This is basically the same as using the RBM trained on MNIST as the starting point for training in the new domains (ICDAR and TiCC). The RBM was then augmented with newly-added hidden neurons for training in the new domains. The transferred knowledge is expected to help learning in the new domains; this knowledge was kept unchanged, with only the weights of the newly-added neurons being allowed to change. The results were put through an SVM to provide a classification measure, for the sake of comparison. As expected, in the case of transferring complete-models from MNIST to ICDAR, we have observed a small improvement in classification performance: the network trained with transfer achieved 51.55% accuracy on the ICDAR dataset, while a knowledge-free RBM achieved 50.00%. We also ran just an SVM on the raw data, which achieved 38.14% accuracy. A slightly better improvement was observed when transferring from MNIST (handwritten digits) to TiCC (handwritten characters) in order to recognize writing styles. Table 3 shows all the results.

Target	SVM	RBM	RBM with Transfer
ICDAR	38.14%	50.00%	**51.55%**
TiCC	72.94%	78.82%	**81.18%**

Table 3: Comparison with RBM with transfer learning

It is generally accepted that the performance of a model in a target domain will depend on the quality of the knowledge it

[1]http://algoval.essex.ac.uk:8080/icdar2005/index.jsp?page=ocr.html

receives and the structure of the model. We then evaluate performance also using different sizes of transferred knowledge. Figure 6 shows that if this size is too small, the model will be dominated by the data from the target domain. If, on the other hand, this size is too large, the model might not learn from the target domain, with a consequent drop in performance as the model responds mainly to the knowledge transferred. Further analysis of our rule extraction method might help guide this transfer process, which nevertheless could turn out to be a domain-dependent task.

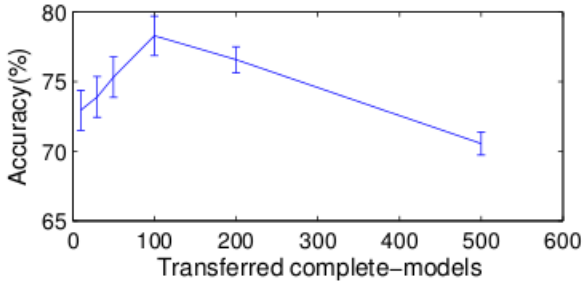


Figure 6: Performance with different sizes of transferred knowledge on the TiCC dataset: for each number of rules transferred, we have added zero, 30, 50, 100, 200, 500 and 1000 nodes to the network. Each network was tested 50 times with the graph showing the mean accuracies and standard deviations. Our results also indicate that adding 1000 nodes produce worse accuracy than adding no nodes at all, with the best performance at 200 extra nodes.

6 Conclusions And Future Work

We have proposed a knowledge extraction method and have applied it to deep belief networks trained on images. The results indicate that a deep belief network can be represented by a set of rules with logical inference serving as an alternative to stochastic sampling. In addition, by transferring the extracted rules onto a target domain, classification performance can be improved on that domain. As future work, we are interested in studying the relationships between rules' confidence values and networks in a more systematic way, and in using rule extraction based on partial models to guide transfer learning.

References

[Aha *et al.*, 2010] David W. Aha, M. Boddy, V. Bulitko, and A. S. d'Avila Garcez et al. Reports of the AAAI 2010 conference workshops. *AI Magazine*, 31(4):95–108, 2010.

[Bengio, 2009] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[Borges *et al.*, 2011] Rafael V. Borges, Artur S. d'Avila Garcez, and Luís C. Lamb. Learning and representing temporal knowledge in recurrent networks. *IEEE Transactions on Neural Networks*, 22(12):2409–2421, 2011.

[Carreira-Perpinan and Hinton, 2005] M. A. Carreira-Perpinan and G. E. Hinton. On contrastive divergence learning. *Artificial Intelligence and Statistics*, January 2005.

[d'Avila Garcez *et al.*, 2001] A.S. d'Avila Garcez, K. Broda, and D.M. Gabbay. Symbolic knowledge extraction from trained neural networks: A sound approach. *Artificial Intelligence*, 125:155–207, 2001.

[de Penning *et al.*, 2011] Leo de Penning, Artur S. d'Avila Garcez, Luís C. Lamb, and John-Jules Ch. Meyer. A neural-symbolic cognitive agent for online learning and reasoning. In *IJCAI*, pages 1653–1658, 2011.

[Erhan *et al.*, 2009] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. Technical Report 1341, University of Montreal, June 2009.

[Hinton *et al.*, 2006] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural Comp.*, 18(7):1527–1554, July 2006.

[Ji *et al.*, 2010] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. In *ICML*, pages 495–502, 2010.

[Lee *et al.*, 2009a] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 609–616, New York, NY, USA, 2009. ACM.

[Lee *et al.*, 2009b] Honglak Lee, Yan Largman, Peter Pham, and Andrew Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in Neural Information Processing Systems 22*, pages 1096–1104. 2009.

[Maaten, 2009] Laurens Maaten. Visualizing higher-layer features of a deep network. Technical Report 1341, University of Montreal, June 2009.

[Pinkas, 1995] Gadi Pinkas. Reasoning, nonmonotonicity and learning in connectionist networks that capture propositional knowledge. *Artificial Intelligence*, 77(2):203 – 247, 1995.

[Smolensky, 1986] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In *In Rumelhart, D. E. and McClelland, J. L., editors, Parallel Distributed Processing: Volume 1: Foundations*, pages 194–281. MIT Press, Cambridge, 1986.

[Son Tran and Garcez, 2012] Son Tran and Artur Garcez. Logic extraction from deep belief networks. In *ICML 2012 Representation Learning Workshop*, Edinburgh, July 2012.

[Torrey *et al.*, 2010] Lisa Torrey, Jude W. Shavlik, Trevor Walker, and Richard Maclin. Transfer learning via advice taking. In *Advances in Machine Learning I*, pages 147–170. 2010.

Combining Runtime Verification and Property Adaptation through Neural-Symbolic Integration

Alan Perotti
University of Turin
perotti@di.unito.it

Guido Boella
University of Turin
boella@di.unito.it

Artur d'Avila Garcez
City University London
aag@soi.city.ac.uk

Abstract

We propose a framework for combining runtime verification and learning in connectionist models. This framework is expected to offer an improved efficiency of the verification process through parallel computation, and a new mechanism of compliance with soft-constraints through learning (adaptation). The goal of business process adaptation is to discover, monitor and improve real processes by extracting knowledge from real-time event logs readily available in today's information systems. Within this wider framework, we have developed a run-time verification system for linear temporal logic, called RuleRunner, which we have designed explicitly with the goal of translating into a neural network for parallel computation and learning. In this paper, using results from neural-symbolic integration, we introduce the translation algorithm that enables the encoding of any RuleRunner specifications into standard CILP recurrent neural networks. The obtained network is shown to converge to the evaluation status of RuleRunner for each cell of the trace. Initial experimental results, also reported in this paper, indicate that the sparse version of the network representation scales better than the symbolic implementation of RuleRunner. The resulting system is a neural network capable of efficient runtime verification and offering well-known learning capabilities, which are being investigated. The framework will be applied to the adaptation of business processes capable of tolerating soft-violations occurring in real practice.

1 Introduction

Existing runtime verification systems are almost always built on the assumption that the specifications they are going to verify are fixed. This assumption is increasingly challenged in domains and tasks like the verification of compliance of business processes. In particular, it is relevant to distinguish two kind of violations: hard and soft. Hard violations match the classical definition of faults, i.e., errors that determine an unacceptable behaviour from the monitored system. Soft violations, on the other hand, are discrepancies between high-level directives and real implementation.

For example, in a bank, the MIFID regulation prescribes that before signing, the customer of a financial product should be informed adequately. This is a hard constraint which should not be violated. In contrast, a constraint coming from internal regulations, such as that some document must be sent in paper format rather than scanned, can be violated without major problems, if the scanned version is still compliant with laws. Thus, if a branch is considered successful (and is compliant with hard constraits) by substituting paper for scanned versions of a given document then the original specification may need to be revised.

Therefore, the problem of runtime verification must overcome the classical fault-detection schema, becoming a more flexible, fine-grained approach able to detect and report the hard violations, but also to integrate the soft violations in the encoded formal specification. A sketch of the desired system is shown in Figure 1.

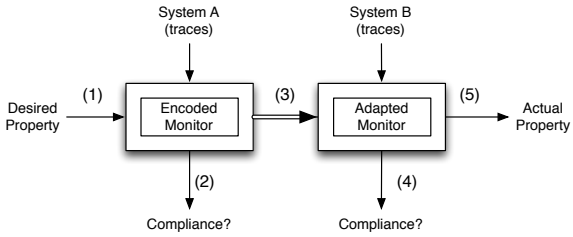


Figure 1: Sketch of the desired verification system

In this vision, the verification system should:

1. Encode the specified properties, for example in a formalism like LTL [Pnueli, 1977], and take as input traces from the observed system A.
2. Verify the compliance of system A's traces w.r.t the encoded property description, checking whether unacceptable normative violations occur.
3. If A exhibits only soft violations and its performance is better than other systems, learn its behaviour by observing it, merging A's ideal description and its actual behaviour in a new monitor.

4. Use the newly obtained monitor to verify the compliance of a second system, B. Intuitively, the idea is to verify how affine is the organisation of A and B, both concerning the normative directives and the actual process management.
5. Extract, from the adapted monitor, a new specification of the actual process management activity in A where the properties are revised according to the traces of A.

In this paper, we propose to use a neural network as the encoded monitor of Figure 1, focusing on the first two items: encoding and reasoning. We describe our rule-based runtime verification system, RuleRunner, and we introduce an algorithm to encode it in a recurrent neural network; the resulting system is a neural network capable of efficient runtime verification. We implemented all components and we obtained encouraging preliminary results in terms of performance. Furthermore, our system offers well-known learning capabilities, which are being investigated.

The paper is structured as follows: Section 2 introduces related work, Section 3 describes the RuleRunner system. Section 4 describes the neural encoding of a RuleRunner system. Section 5 discusses implementation and some performance results of the prototype, Section 6 ends the paper with conclusions and future work.

2 Background

2.1 Business Process Management

The IEEE Task Force on Process Mining aims to promote the topic of process mining. In the context of this task force, a group of more than 75 people involving more than 50 organisations created the Process Mining Manifesto. By defining a set of guiding principles and listing important challenges, this manifesto hopes to serve as a guide for software developers, scientists, consultants, business managers, and end-users. The goal is to discover, monitor and improve real processes (i.e., not assumed processes) by extracting knowledge from event logs readily available in today's (information) systems. Therefore, process mining systems aim at closing the gap between the huge (and growing) amount of stored data about processes and activities and the need for making the business process management flexible and competitive for ever-evolving contexts.

The Business Process Management provides an application domain for our framework: when it comes to big companies (such as a bank), in some of their branches the task execution often differs from the rigid protocol enforcement, due to obstacles (from broken printers to strikes) or by adaptations to specific needs (dynamic resources reallocation): these are examples of soft violations. On the other hand, law infringements concerning security and privacy issues are hard violations. For instance, having the customer of a financial product sign a contract without being adequately informed is a (hard) violation of the MIFID regulation[1]. Thus, if a branch considered successful and compliant with hard constraints, substituted paper versions of a given documents by scanned ones despite the original specifications, such specifications may be revised. The source of the modification should be the traces of the activity of the successful branch, which can be fed as input to a machine learning system.

[1]Markets in Financial Instruments Directive 2004/39/EC (link)

2.2 Runtime Verification

Runtime verification (RV) of a given correctness property ϕ (often formulated in linear temporal logic LTL [Pnueli, 1977]) aims at determining the semantics of ϕ while executing the system under scrutiny; a monitor is defined as a device that reads a finite trace and yields a certain verdict [Leucker and Schallhart, 2009]. A trace is a sequence of cells, which in turn are lists of observations occurring in a given discrete span of time. Runtime verification may work on finite (terminated) traces, finite but continuously expanding traces, or on prefixes of infinite traces. A monitor may control the current execution of a system (online) or analyse a recorded set of finite executions (offline). There are many semantics for finite traces: FLTL [Lichtenstein *et al.*, 1985], RVLTL [Bauer *et al.*, 2007], LTL3 [Bauer *et al.*, 2006], LTL [Eisner *et al.*, 2003] just to name some. Since LTL semantics is based on infinite behaviours, these semantics aim to close the gap between properties specifying infinite behaviours and finite traces. There exist several RV systems [Leucker and Schallhart, 2009], and they can be clustered in three main approaches, based respectively on rewriting, automata and rules. In this paper, we focus on rule-based system, due to their similarity with neural networks.

2.3 Neural-Symbolic Integration

The main purpose of a neural-symbolic system is to bring together the connectionist and symbolic approaches exploiting the strengths of both paradigms and, hopefully, avoiding their drawbacks. In [Towell and Shavlik, 1994], Towell and Shavlik presented the influential neural-symbolic system KBANN (Knowledge-Based Artificial Neural Network), a system for rule insertion, refinement and extraction from feedforward neural networks. KBANN served as inspiration for the construction of the Connectionist Inductive Learning and Logic Programming (CILP) system [d'Avila Garcez and Zaverucha, 1999]. CILP builds upon KBANN and [Hoelldobler and Kalinke, 1994] to provide a sound theoretical foundation for reasoning in artificial neural networks with learning capabilities. The general framework of a neural symbolic system is composed of three main phases: encoding symbolic knowledge in a neural network, performing theory revision (by means of some learning algorithm) in the network, and extracting a revised knowledge from the trained network. In particular, rules are mapped onto hidden neurons, the preconditions of rules onto input neurons and the conclusion of the rules onto output neurons. The weighs are then adjusted to express the

dependence among all these elements [Garcez *et al.*, 2002].

3 RuleRunner

RuleRunner is a rule-based online monitor observing finite but expanding traces and returning an FLTL verdict. As it scans the trace, RuleRunner mantains a state composed of rule names (for reactivating the rules), observations and formulae evaluations.
Given a finite set of observations O and a LTL formula ϕ over (a subset of) O, RuleRunner has a state composed of observations ($o \in O$), rule names ($R[\phi]s$) and truth evaluations ($[\phi]V$); $V \in \{T, F, ?\}$ is a truth value and s, called *suffix*, is used to identify formulae. Due to the lack of space, we will omit the technical details of RuleRunner, [Perotti, 2013] focusing on how the monitoring task is performed and how the rules are structured: this will be relevant in the next subsection. The state evolves according to rules: RuleRunner is composed of evaluation and reactivation rules. Evaluation rules are shaped like

$$R[\phi], [\psi^1]V, \ldots, [\psi^n]V, \rightarrow [\phi]V$$

and, intuitively, their role is to compute the truth value of a formula ϕ under verification, given the truth values of its direct subformulae ψ^i. Reactivation rules are shaped like

$$[\phi]? \rightarrow R[\phi], R[\psi^1], \ldots, R[\psi^n]$$

and, if one formula is evaluated to undecided, that formula (together with its subformulae) are scheduled to be monitored again in the next cell of the trace. This concept of rule reactivation was firstly described in [Barringer *et al.*, 2010].
A RuleRunner rule system RR is defined as $\langle R_E, R_R, S\rangle$ where R_E anr R_R are, respectively, the sets of evaluation and reactivation rules, and S is the initial state. The general algorithm for creating and running a RR is described in Algorithm 1:

Algorithm 1 Preprocessing and Monitoring Cycle

```
1: Parse the LTL formula in a tree
2: Generate evaluation rules, reactivation rules and the
   initial state
3: while new observations exist do
4:     Add observations to state
5:     Compute truth values using evaluation rules
6:     Compute next state using reactivation rules
7:     if state contains SUCCESS or FAILURE then
8:         return return SUCCESS or FAILURE
   respectively
9:     end if
10: end while
```

In a nutshell, RuleRunner's behaviour is the following: in the preprocessing phase, RuleRunner encodes an LTL formula in a rule system. The rule system verifies the compliance of a trace w.r.t. the encoded property by entering a monitoring loop, composed of observing a new cell of the trace and computing the truth value of the property in the given cell. If the property is irrevocably satisfied or falsified in the current cell, RuleRunner outputs a binary verdict. If this is not the case, another monitoring iteration is entered, and -like in [Barringer *et al.*, 2010]- undecided formulae trigger the reactivation of the corresponding monitoring rule. FLTL semantics guarantees that, if the trace ends, the verdict in the last cell of the trace is binary.

It is worth stressing how RuleRunner's approach is *bottom-up*, forwarding truth values from mere observations to the global property. RuleRunner does not keep a [multi]set of alternatives, as it is rooted in matching the encoding of the formula with the actual observations, computing the unique truth value of every subformula of the property, and carrying along a single state composed of certain information.

RuleRunner provides rich information about the 's' of a property: in any iteration the state describes which subformulae are under monitoring and what the truth value is; when the monitoring ends, the state itself explains why the property was verified/falsified.

One minimal example that shows all functionalities of RuleRunner is $\Diamond a$: we will use it as a working example throughout the paper. In the next example, we consider the RuleRunner system encoding $\Diamond a$ and its evolution over the trace $[b - a - b, END]$.

The desired behaviour of the monitoring is the following:

- In the first cell, b is observed and ignored, while non observing a makes a false. Since the end of the trace has not been reached, $\Diamond a$ is undecided, and the initial state is repeated in the second cell. Note that in the general case the initial state is not identically recomputed in the following cells.
- In the second cell, a is observed, thus satisfying, in cascade, a and $\Diamond a$; since the main formula has been satisfied, the monitoring can stop in the current cell reporting a success.

RuleRunner creates the following rule system:

EVALUATION RULES

- $R[a]$, a is observed $\rightarrow [a]T$
- $R[a]$, a is not observed $\rightarrow [a]F$
- $R[\Diamond a], [a]T \rightarrow [\Diamond a]T$
- $R[\Diamond a], [a]? \rightarrow [\Diamond a]?P$
- $R[\Diamond a], [a]F \rightarrow [\Diamond a]?P$
- $[\Diamond a]?, [END] \rightarrow [\Diamond a]F$
- $[\Diamond a]?, \sim[END] \rightarrow [\Diamond a]?$
- $[\Diamond a]T \rightarrow SUCCESS$
- $[\Diamond a]F \rightarrow FAILURE$

REACTIVATION RULES

- $[\Diamond a]? \rightarrow R[a], R[\Diamond a]$

INITIAL STATE $R[a], R[\Diamond a]$

EVOLUTION OVER $[b - a - b, END]$

state	$R[a], R[\Diamond a]$
+ obs	$R[a], R[\Diamond a], b$
eval	$[a]F, [\Diamond a]?P, [\Diamond a]?$
react	$R[a], R[\Diamond a]$
state	$R[a], R[\Diamond a]$
+ obs	$R[a], R[\Diamond a], a$
eval	$[a]T, [\Diamond a]T, SUCCESS$
STOP	PROPERTY SATISFIED

It is easy to see that RuleRunner's behaviour matches the desired one, [Perotti, 2013] provides a detailed description of RuleRunner's behaviour.

4 Neural Encoding

The first step of the neural encoding is the translation of a RuleRunner system into an equivalent logic program (Algorithm 2).

Algorithm 2 From RuleRunner to Logic Programs

function RR2LP(ϕ)
  Create $RR = \langle R_E, R_R, S \rangle$ encoding ϕ
  Create an empty logic program LP ▷ C_E
  for all $R[\phi], [\psi^1]V, \ldots, [\psi^n]V, \rightarrow [\phi]V \in R_E$ **do**
    $LP \leftarrow LP \cup [\phi]V$:-$\sim[U], R[\phi], [\psi^1]V, \ldots, [\psi^n]V$
  end for ▷ C_P
  for all $o \in R_E \cup R_R$ **do**
    $LP \leftarrow LP \cup\ o$:- $\sim[U], o$
  end for
  for all $R[\phi] \in R_E \cup R_R$ **do**
    $LP \leftarrow LP \cup\ R[\phi]$:- $\sim[U], R[\phi]$
  end for ▷ C_R
  for all $[\phi]? \rightarrow R[\phi], R[\psi^1], \ldots, R[\psi^n] \in R_R$ **do**
    $LP \leftarrow xLP \cup\ R[\phi]$:- $[U], [\phi]?$
    for all $R[\psi^i]$ **do**
      $LP \leftarrow LP \cup\ R[\psi^i]$:- $[U], [\phi]?$
    end for
  end forreturn LP
end function

The algorithm creates a single logic program LP; however, for the sake of explanation, we distinguish three kinds of clauses: evaluation, reactivation and persistence (marked as C_E, C_R, C_P in Algorithm 2). Intuitively, evaluation and reactivation clauses (in LP) mirror, respectively, evaluation and reactivation rules in RuleRunner. Persistence clauses are used to 'remember' observations and active rules by explicit re-generation: these clauses follow the pattern x :- $\sim[UPDATE], x$ (shortened to *[U]* in the algorithm).

Evaluation clauses are obtained from evaluation rules by adding one extra literal in the body, $\sim[UPDATE]$. The reactivation rules are split into several reactivation clauses, one for each literal in the head of the rule; $[UPDATE]$ is added in the body of all these rules. Finally, for all observations and truth evaluations in the rules, a persistence clause is added.

RuleRunner's monitor loop fires the evaluation and reactivation rules in an alternate fashion. We simulate that by introducing the $[UPDATE]$ literal and using it as a switch: when $[UPDATE]$ holds, only reactivation clauses can hold, while when it does not all reactivation rules are inhibited and evaluation and persistence clauses are potentially active. RuleRunner iteratively builds a state, going through partial solutions; in the same way, some sort of clamping is necessary for the partial results to be remembered by the LP. We achieve that by adding persistence clauses: as long as $[UPDATE]$ does not hold, all observations and rule names are re-obtained at each iteration of the logic program. Another way to achieve this persistence is to add the desired observations/rule names as facts: we opted for the former because persistence clauses have a standard structure, while adding facts to the LP correspond to clamping neurons in a neural network.

In our working example, the rule system encoding $\Diamond a$ is translated into the following logic program:

EVALUATION CLAUSES

- $[a]T$:- $\sim[UPDATE], R[a], a$
- $[a]F$:- $\sim[UPDATE], R[a], \sim a$
- $[\Diamond a]T$:- $\sim[UPDATE], R[\Diamond a], [a]T$
- $[\Diamond a]?P$:- $\sim[UPDATE], R[\Diamond a], [a]?$
- $[\Diamond a]?P$:- $\sim[UPDATE], R[\Diamond a], [a]F$
- $[\Diamond a]?$:- $\sim[UPDATE], [\Diamond a]?, \sim[\Diamond a]T, \sim END$
- $[\Diamond a]F$:- $\sim[UPDATE], [\Diamond a]?P, \sim[\Diamond a]T, END$
- $[SUCCESS]$:- $\sim[UPDATE], [\Diamond a]T$
- $[FAILURE]$:- $\sim[UPDATE], [\Diamond a]F$

PERSISTENCE CLAUSES

- a :- $\sim[UPDATE], a$
- $R[a]$:- $\sim[UPDATE], R[a]$
- $R[\Diamond a]$:- $\sim[UPDATE], R[\Diamond a]$

REACTIVATION CLAUSES

- $R[a]$:- $[UPDATE], [a]?$
- $R[a]$:- $[UPDATE], [\Diamond a]?$
- $R[\Diamond a]$:- $[UPDATE], [\Diamond a]?$

Summarising, we start from a formal property ϕ expressed as an LTL formula, we compute a RuleRunner rule system RR_ϕ monitoring that formula, and then we encode the rule set in a logic program LP_ϕ. By doing so, we can exploit the $CILP$ algorithm [d'Avila Garcez and Zaverucha, 1999] to translate the logic program into an equivalent neural network NN_ϕ.

Continuing with the working example, the neural network encoding a monitor for $\Diamond a$ is visualised in Figure 2. The input and output layers include neurons whose labels correspond to atoms in the logic program (and the rule system); each hidden neuron corresponds to one clause in the logic program. Active neurons are filled in grey: in Figure

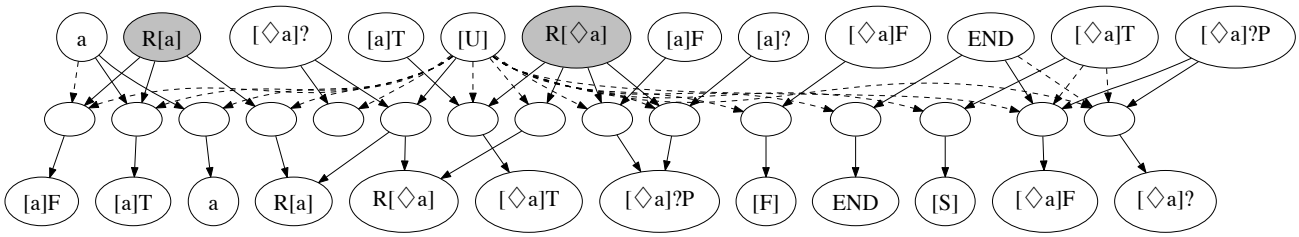


Figure 2: Neural network for monitoring $\Diamond a$

2, the initial state $(R[a], R[\Diamond a])$ is depicted. Solid lines represent connection with positive weights, dashed lines represent negative weights. For instance, the leftmost hidden neuron has ingoing connections from a (negative weight), $R[a]$, and $[U]$ (negative weight) and it has an outgoing connection towards $[a]F$; this corresponds to the clause $[a]F$:- $a, \sim a, R[a], \sim[U]$ (which is the second clause in the LP in the previous page).

Reasoning: The general algorithm for neural monitoring is described in Algorithm 3:

Algorithm 3 Preprocessing and Monitoring Cycle in NN_ϕ

```
1: function NN-MONITOR(φ,trace t)
2:     Create RR = ⟨R_R, R_E, S⟩ encoding φ (RuleRunner)
3:     Rewrite RR_φ into LP_φ (Algorithm 2)
4:     Rewrite LP_φ into NP_φ (CILP)
5:     Add S to the input layer
6:     while new observations exist in t do
7:         Add the new observations to the input layer
8:         Let the network converge
9:         if S contains SUCCESS (resp.FAILURE) then
10:            return return SUCCESS (resp.FAILURE)
11:        end if
12:        Add UPDATE to the input layer
13:        Fire the network once
14:    end while
15: end function
```

Adding x to a given layer means activating the neuron corresponding to x in that layer. In terms logic programming, this would correspond to adding the fact x to the program.

It is worth comparing how, from an operational point of view, a RuleRunner system (RR_ϕ) and its neural encoding (NN_ϕ) carry out the monitoring task. In each iteration of the main loop, RuleRunner goes through the list of evaluation rules, adding the result of each active rule to the state. When the end of the evaluation rules list is reached, the reactivation rules are allowed to fire, collecting the output in a new state. In the neural network the alternating of evaluation and reactivation is achieved by means of the $[UPDATE]$ neuron, which acts as a switch. In the evaluation phase, all evaluation rules are fired in parallel until convergence.

5 Implementation and Initial Results

We implemented RuleRunner[2] and the neural modules. Rulerunner is implemented in Java; in order to maximise the performances of the neural encoding, we experimented with several implementations, focusing on matrices manipulation.

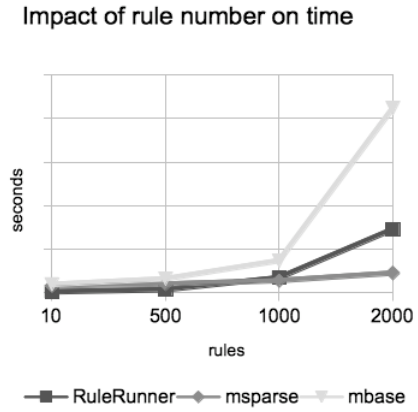


Figure 3: Compared performances

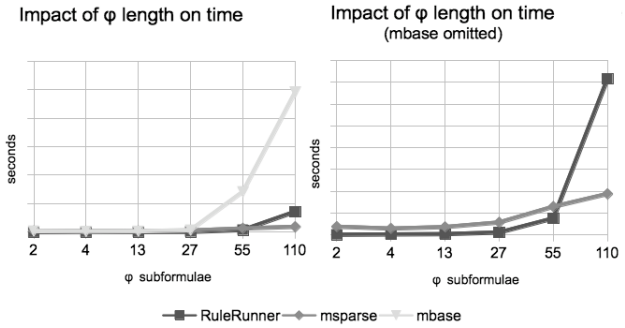


Figure 4: Compared performances

One key point in understanding how to fully exploit the parallelism of neural network models (and therefore optimise the performance of the neural encoding of RuleRunner)

[2] *www.di.unito.it/∼perotti/RV13.jnlp*

is that, even if the number of rules is large, each rule has a constant number of literals in the body, and each literal appears in (at most) a constant number of rules. Therefore, the weight matrices are very sparse.

We tested RuleRunner's performance on a number of tests and compared the results with two Matlab implementations, m_base and m_sparse: in the latter, we treated all matrices (activations, thresholds, weights) as sparse. It is worth stressing that, since we compared two prototypes implemented in different programming languages (Java and Matlab), it would not be fair to compare their performances in absolute terms; we are, instead, interested in scalability and asymptotic analysis.

These preliminary experiments show how m_base is outperformed by RuleRunner, but m_sparse scales better than both. Figure 3 shows the impact of increasing the rule numbers, in Figure 4 the size of the encoded formula is increased. In all cases the Y axis reports the average time required to process 10000 cells of randomly-generated traces.

6 Conclusions

With the goal of making the concept of compliance flexible and dynamic, we tackled the challenge of developing a framework to rigidly verify a system's compliance with a model, modelling at the same time what actually takes place in terms of process management, making it exploitable by other systems. We chose Neural-Symbolic Integration as the underlying paradigm and rule-based verification system as a bridge from the (symbolic) area of Runtime Verification and machine learning through neural networks.

As a first contribution, this paper provides a methodology for performing runtime verification within a neural network, encoding a novel rule system in a logic program and then in a recurrent neural network; we developed a prototype for our system, observing that an implementation based on sparse matrices shows better performances than RuleRunner, both in absolute terms and from the point of view of scalability. Secondly, our approach seeks to reduce the gap between Runtime Verification, Artificial Intelligence and Business Process Management, proposing a system able to perform RV tasks in a model with intrinsic learning features.

Future work involve the exploitation of standard and ad-hoc learning strategies, in relation with sequence and reinforcement learning: the goal is to analyse the adaptation capability of the model in order to integrate formal properties, encoded in the network, with the actual behaviour of the observed system, by means of feeding traces to the network as inductive learning examples.

References

[Barringer *et al.*, 2010] Howard Barringer, David E. Rydeheard, and Klaus Havelund. Rule systems for run-time monitoring: from eagle to ruler. *J. Log. Comput.*, 20(3):675–706, 2010.

[Bauer *et al.*, 2006] Andreas Bauer, Martin Leucker, and Christian Schallhart. Monitoring of real-time properties. In S. Arun-Kumar and Naveen Garg, editors, *FSTTCS*, volume 4337 of *Lecture Notes in Computer Science*, pages 260–272. Springer, 2006.

[Bauer *et al.*, 2007] Andreas Bauer, Martin Leucker, and Christian Schallhart. The good, the bad, and the ugly, but how ugly is ugly? In Oleg Sokolsky and Serdar Tasiran, editors, *RV*, volume 4839 of *Lecture Notes in Computer Science*, pages 126–138. Springer, 2007.

[d'Avila Garcez and Zaverucha, 1999] Artur S. d'Avila Garcez and Gerson Zaverucha. The connectionist inductive learning and logic programming system. *Appl. Intell.*, 11(1):59–77, 1999.

[Eisner *et al.*, 2003] Cindy Eisner, Dana Fisman, John Havlicek, Yoad Lustig, Anthony McIsaac, and David Van Campenhout. Reasoning with temporal logic on truncated paths. In *CAV'03*, pages 27–39, 2003.

[Garcez *et al.*, 2002] Artur S. d'Avila Garcez, Dov M. Gabbay, and Krysia B. Broda. *Neural-Symbolic Learning System: Foundations and Applications*. Springer-Verlag New York, Inc., 2002.

[Hoelldobler and Kalinke, 1994] Steffen Hoelldobler and Yvonne Kalinke. Towards a new massively parallel computational model for logic programming. In *ECAI94 workshop on Combining Symbolic and Connectioninst Processing*, pages 68–77, 1994.

[Leucker and Schallhart, 2009] Martin Leucker and Christian Schallhart. A brief account of runtime verification. *J. Log. Algebr. Program.*, 78(5):293–303, 2009.

[Lichtenstein *et al.*, 1985] Orna Lichtenstein, Amir Pnueli, and Lenore D. Zuck. The glory of the past. In Rohit Parikh, editor, *Logic of Programs*, volume 193 of *Lecture Notes in Computer Science*, pages 196–218. Springer, 1985.

[Perotti, 2013] Alan Perotti. Rulerunner technical report. 2013. arXiv:1306.0810.

[Pnueli, 1977] Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 46–57, Washington, DC, USA, 1977. IEEE Computer Society.

[Towell and Shavlik, 1994] Geoffrey G. Towell and Jude W. Shavlik. Knowledge-based artificial neural networks. *Artif. Intell.*, 70(1-2):119–165, 1994.

On Common Ground: Neural-Symbolic Integration and Lifelong Machine Learning

Daniel L. Silver
Acadia University
Nova Scotia, Canada
danny.silver@acadiau.ca

Abstract

Research efforts in Neural-Symbolic Integration and Lifelong Machine Learning have taken place with limited interactions over the last 20 years. These two areas of artificial intelligence share common ground and yet have much to learn from each other. This paper provides background, particularly on Lifelong Machine Learning, and presents a number of common areas of investigation with Neural-Symbolic Integration. It then invites researchers in the two fields to better understand some of the differences in motives and objectives for the purpose of jointly advancing machine learning, knowledge representation and reasoning.

1 Introduction

Recent work on unsupervised learning using deep learning architectures has given rise to new ideas on how knowledge of the world is learned, consolidated, and then used for future learning and reasoning [Bengio, 2009; Le *et al.*, 2012]. This is bringing together research in the areas of machine learning and knowledge representation that have traditionally been pursued separately. Specifically, research efforts in Neural-Symbolic Integration [Garcez *et al.*, 2009; Bader and Hitzler, 2005] and Lifelong Machine Learning [Silver *et al.*, 2013] have taken place with only a few points of interaction over the last 20 years. This paper presents several areas of common ground for these areas where joint work has taken place and further collaboration is possible. The paper also points out fundamental differences in the motives and objectives of NSI versus LML that need to be understood by researchers as we move forward. In particular, we propose that joint research has the potential to make serious advances on a significant problem in artificial intelligence - the learning of *common background knowledge* that can be used for future learning and reasoning.

2 Background

2.1 Neural-Symbolic Integration

Neural-symbolic integration, or NSI, considers hybrid systems that integrate neural networks and symbolic logic. The goal of NSI is to take advantage of the best of symbolic and connectionist paradigms of artificial intelligence for both learning and reasoning [Garcez *et al.*, 2002; Garcez and Gabbay, 2004].

NSI seeks to make use of the learning capacities of neural network models and the reasoning capacities of logic [Garcez *et al.*, 2014]. In a NSI system, neural networks provide the machinery for parallel computation and robust learning, while symbolic logic provides knowledge representation and reasoning. The retention of symbolic knowledge of learned models can be used for transfer learning, explanation of the models to humans, and use of the knowledge by other systems. Neural-symbolic systems have application in knowledge acquisition where a system learns a complex model and then needs to reason about what has been learned in order to respond to a new situation.

NSI has at least three major areas of investigation. First there is the use of connectionist systems for symbolic knowledge representation, reasoning and learning. These systems include comparisons with purely-symbolic and purely-connectionist models, and the representation of relational, first-order and modal logics for higher-order reasoning [Garcez *et al.*, 2014]. A second area of research is the extraction of high-level concepts and knowledge from complex networks. The focus here is on efficient and effective knowledge extraction from very large networks for the purpose model comprehension, validation, maintenance and transfer learning. The third area of investigation is the design and development of applications in areas such as vision, robotics, intelligent agents, and simulation.

2.2 Lifelong Machine Learning

Lifelong Machine Learning, or LML, is concerned with the persistent and cumulative nature of learning [Thrun, 1996b]. LML considers systems that can learn many tasks over a lifetime from one or more domains. An LML system must efficiently and effectively retain the knowledge it has learned and transfer that knowledge to more efficiently and effectively learn new tasks through the transfer of knowledge [Silver *et al.*, 2013]. The following sections provide an overview of prior LML research in all areas of machine learning - supervised, unsupervised and reinforcement learning.

Supervised Learning

As early as the mid 1980s Michalski and Solomonoff had theories on *constructive inductive learning* [Michalski, 1993]

and *incremental learning* [Solomonoff, 1989]. In the mid 1990s, Thrun and Mitchell worked on *explanation-based neural networks* [Thrun, 1996a] and applied EBNN transfer learning to autonomous robot learning when a multitude of control learning tasks are encountered over an extended period of time [Thrun and O'Sullivan, 1995].

Since 1995, Silver *et al.* have proposed variants of *sequential learning and consolidation systems* using standard back-propagation neural networks [Silver and Poirier, 2004; Silver *et al.*, 2008]. A method called *task rehearsal* is an essential part of these systems. After a task has been successfully learned, its hypothesis representation is saved. The saved hypothesis can be used to generate virtual training examples so as to rehearse the prior task when learning a new task. Knowledge is transferred to the new task through the rehearsal of previously learned tasks within the shared representation of the neural network. Similarly, the knowledge of a new task can be consolidated into a large domain knowledge network without loss of existing task knowledge by using task rehearsal to maintain the functional accuracy of the prior tasks while the representation is modified to accommodate the new task.

In the late 1990s, Rivest and Schultz proposed *knowledge-based cascade-correlation* neural networks [Shultz and Rivest, 2001]. The method extends the original cascade-correlation approach, by selecting previously learned sub-networks as well as simple hidden units. In this way the system is able to use past learning to bias new learning.

Unsupervised Learning

Transfer in unsupervsied learning is almost as old as that of supervised learning. In the mid 1980s, Carpenter and Grossberg proposed ART (Adaptive Resonance Theory) neural networks to overcome the stability-plasticity problem of forgetting previous learned data concepts [Grossberg, 1987].

Raina *et al.* proposed the *Self-taught Learning* method to build high-level features using unlabeled data for a set of tasks [Raina *et al.*, 2007]. The authors used the features to form a succinct input representation for problems such as image and webpage classification.

Recent research into *deep learning architectures* of neural networks can be connected to LML [Bengio, 2009]. Layered neural networks of unsupervised Restricted Boltzman Machine auto-encoders have been shown to efficiently develop hierarchies of features that capture statistical regularities in their respective inputs. When used to learn a variety of class categories, these networks develop layers of common features similar to that seen in the visual cortex of humans. Le *et al.* have used deep learning methods to build high-level features for large-scale applications by scaling up the dataset, the model and the computational resources [Le *et al.*, 2012]. By using millions of high resolution images and very large neural networks, their system effectively discover high-level concepts like the presence of a cat's face in an image. Experimental results show that their network can use its learned features to achieve a significant improvement in image classification performance over state-of-the-art methods.

Reinforcement Learning

Several reinforcement learning researchers have considered LML systems. In 1997, Ring proposed a lifelong learning approach called *continual learning* that builds more complicated skills on top of those already developed both incrementally and hierarchically [Ring, 1997].

Tanaka and Yamamura proposed a lifelong reinforcement learning method for autonomous-robots by treating multiple environments as multiple-tasks [Tanaka and Yamamura, 1999].

Sutton *et al.* suggest that learning should continue during an agent's operations since the environment may change making prior learning insufficient [Sutton *et al.*, 2007]. An agent is proposed that adapts to different local environments when encountering different parts of its world over an extended period of time.

Moving Beyond Learning Algorithms

Many machine learning researchers are calling for a move beyond the development of inductive learning algorithms and onto the design of systems that learn, retain and use knowledge over a lifetime. In [Silver *et al.*, 2013] we cite the following reasons for a call for wider research into LML systems.

Selective Inductive Bias is Essential to Learning. The constraint on a learning system's hypothesis space, beyond the criterion of consistency with the training examples, is called *inductive bias* [Mitchell, 1980]. Utgoff wrote in 1983 about the importance of inductive bias to concept learning from practical sets of training examples and the need for learning systems to select bias [Utgoff, 1983]. The AI community has come to accept the futility of searching for a universal machine learning algorithm [Wolpert, 1996]. LML systems that retain and selectively use prior knowledge as a source of inductive bias promotes this perspective.

Theoretical Advances in ML and KR. Thrun proposed "The acquisition, representation and transfer of domain knowledge are the key scientific concerns that arise in lifelong learning" [Thrun, 1997]. Knowledge representation will play an important role in the development of LML systems. More specifically, the interaction between knowledge retention and knowledge transfer will be key to the design of intelligent agents that learn many things over an extended period.

Practical Agents/Robots Require LML. Advances in autonomous robotics and intelligent agents that run on the web or in mobile devices present opportunities for employing LML systems. Robots such as those that go into space or travel under the sea must learn to recognize objects and make decisions over extended periods of time and varied environmental circumstances. The ability to retain and use learned knowledge is very attractive to the researchers designing these systems. Similarly, software agents on the web or in our mobile phones would benefit from the ability to learn more quickly and more accurately as they are challenged to learn new but related tasks from small numbers of examples.

Increasing Capacity of Computers. Advances in modern computers provide the computational power for implementing and testing LML systems. The number of transistors that can be placed cheaply on an integrated circuit has doubled

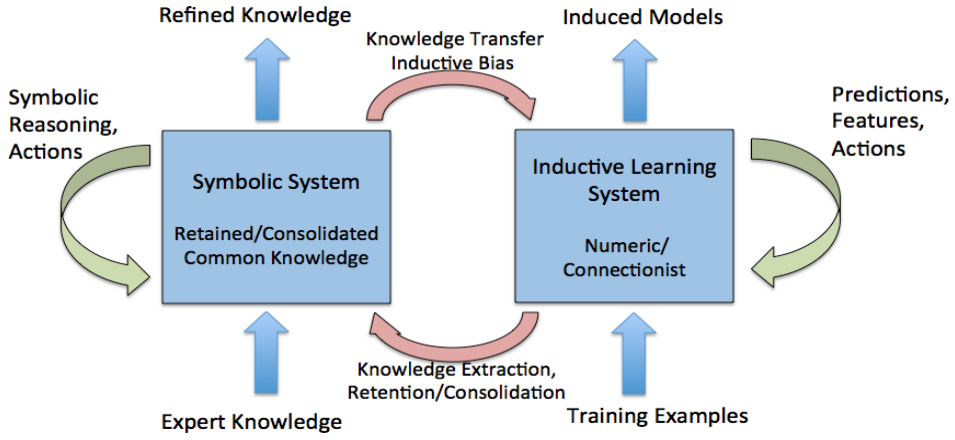


Figure 1: An integrated framework for Neural-Symbolic Integration and Lifelong Machine Learning.

approximately every two years since 1970. This trend is expected to continue into the foreseeable future, as computing systems increasingly use multiple processing cores. We are now at a point where an LML system focused on a constrained domain of tasks (*e.g.* product recommendation) is computationally tractable in terms of both computer memory and processing time [Le *et al.*, 2012].

3 Common Ground

Figure 1 is based on a diagram found in [Bader and Hitzler, 2005]. The modifications show how the NSI framework can integrate nicely with the LML framework presented in [Silver *et al.*, 2013]. The integrated framework is meant to encompass the various methods of machine learning (supervised, unsupervised, or reinforcement) and the various symbolic systems. Expert knowledge can be provided by the user to the symbolic system where it is retained and/or consolidated with existing knowledge. This store of common background knowledge can be used by the inductive learning system as a source of knowledge transfer, or inductive bias. This bias can come in representational form, such as an initial set of neural network weights from which to start learning, or in functional form, such as a set of examples for a secondary task for multiple task learning [Shavlik, 1992; Silver and Mercer, 2002]. The inductive learning system develops a hypothesis, or model, using the transferred knowledge and training examples provided by the user. The result is a more accurate model developed in a shorter period of time. The knowledge learned in these models can be extracted back to the symbolic system and retained (or consolidated) for symbolic reasoning, used by another system, or for explanation to the user.

The following sections discuss areas of common ground shared by NSI and LML research, of which there has already been some joint work.

3.1 Choice of Machine Learning to Use

An open question for both NSI and LML is which approach to machine learning or combination of approaches works best in the context of knowledge extraction and symbolic reasoning. Supervised learning continues to dominate NSI research [Bader and Hitzler, 2005], with some work being done using reinforcement and unsupervised learning including the extraction of symbolic logic from deep believe networks [Tran and Garcez, 2012]. Supervised learning has traditionally dominated LML, but over the last five years unsupervised learning has received a considerable amount of attention. For example, recent work has shown the benefit of unsupervised training using many unlabelled examples to generate new encodings (features) of the examples for use in supervised learning [Bengio, 2009]. Others feel that reinforcement learning is the only true method of developing predictive models [Sutton *et al.*, 2007]. The choice of machine learning algorithm and representation will dramatically affect the structure and function of NSI and LML systems. Combinations of approaches may be helpful for learning individual tasks but will prove challenging for knowledge consolidation and aspects of NSI.

3.2 Training Examples versus Prior Knowledge

Both NSI and LML systems must weigh the relevance and accuracy of retained knowledge along-side the information resident in the available training examples for a new task. An estimate of the sample complexity of the new task will play a role here. The relative accuracy level of prior knowledge must also be considered. Theories of how to selectively transfer common knowledge in combination with existing training examples is of significant value to NSI and LML research.

3.3 Effective and Efficient Knowledge Retention

Mechanisms that can effectively and efficiently retain knowledge over time will suggest new approaches to common knowledge representation. In particular, methods of integrating new knowledge into existing knowledge are of value to researchers in NSI and LML [Shavlik, 1992; Silver and Poirier, 2004]. Efficient long-term retention of symbolic or learned knowledge should cause no loss of prior task knowledge and increase the accuracy of such knowledge if the new task be-

ing retained is related. Furthermore, the knowledge representation approach should allow the NSI or LML system to efficiently select the most effective prior knowledge for transfer during new learning.

In general, research in NSI and LML will see theories of transfer learning and knowledge representation influence and affect each other. A combined research agenda has the potential to make serious advances on a significant AI problem - the learning of *common background knowledge* that can be used for future learning, reasoning and planning. The work at Carnegie Mellon University on NELL is an early example of such research [Carlson *et al.*, 2010].

3.4 Effective and Efficient Knowledge Transfer

The search for transfer learning methods that are able to develop accurate (effective) hypotheses rapidly (efficient) is a challenging problem for both LML and NSI. Transfer learning should produce a hypothesis for a new task that meets or exceeds the generalization performance of a hypothesis developed from only the training examples. There is evidence that functional transfer (e.g. using secondary task examples and multiple task learning) surpasses that of representation transfer in its ability to produce more accurate hypotheses [Caruana, 1997; Silver and Poirier, 2004]. Conversely, research has shown that a representational form of knowledge transfer (initializing a neural network to the weights of a related task) is typically more efficient than a functional form but it rarely results in improved model effectiveness [Silver and Poirier, 2004].

3.5 Scalability

Scalability is often the most difficult and important challenges for artificial intelligent systems. For NSI systems the extraction of symbolic knowledge is normally demanding in terms of time complexity. In LML systems the processes of retention and transfer adds time and space complexity to the challenge of learning. A NSI or LML system must be capable of scaling up to large numbers of inputs, outputs, training examples and learning tasks. Preferably, the space and time complexity of a system grows linearly in all of these factors. The move to Big Data and commercial data analytics is applying pressure on artificial intelligence approaches such as NSI and LML.

3.6 Heterogenous Domains of Tasks

Although, much of NSI and LML research has focused on retention and transfer within a single domain of tasks, an important area of research will be the development of systems that work across heterogenous domains [Yang *et al.*, 2009]. In heterogeneous transfer learning, the key idea is to leverage the feature correspondence across heterogenous domains (such as images and tags; music and lyrics) to build an effective feature mapping for transferring knowledge. Having knowledge in symbolic form would provide a number of new avenues to pursue in terms of knowledge adaptation prior to transfer from one problem domain to another.

3.7 Acquisition and Use of Meta-knowledge

Both NSI and LML systems need to collect and retain meta-knowledge of their task domains. For example, it may be critical for a NSI system to retain knowledge of the relationship or relatedness between learned tasks and an LML system may need to save the range and resolution of its input attributes [Silver *et al.*, 2008].

3.8 Shared Application Domains

Software agents and robots have provided useful test platforms for empirical studies of NSI and LML systems [Thrun, 1996a]. Agents and robots will naturally need to learn new but related tasks. This will provide opportunities to try different methods of retaining and consolidating task knowledge. The agent's fixed input and output domains provide an environment to test the impact of curriculum and the practice of tasks in a controlled manner. NSI and LML can also be used to overcome the *cold-start* problem exhibited by personal agents that employ user modeling [Lashkari *et al.*, 1994]. Retained knowledge can be used to boot-strap a new user model by transferring knowledge from a related user model.

4 Differences in NSI and LML Objectives

The following are some of the fundamental differences that NSI and LML researchers may have in their motives and objectives. These need to be appreciated by both research groups as they move to work together.

4.1 Uses of Retained Common Knowledge

LML systems focus on the use of learned knowledge for deployment in software systems or transfer during future learning. There is not as much consideration given to the transparency of learned knowledge, its use for explanation, or refinement of symbolic knowledge. In NSI systems the refinement and improvement of knowledge during learning is very important. Similarly, the ability to extract symbolic logic in a human readable form is important.

4.2 Retention versus Consolidation

LML researchers have started to consider the structure of retained knowledge for a lifelong learning system. Knowledge retention is necessary, but it may not be sufficient. In [Silver and Poirier, 2004] we propose that domain knowledge must be integrated for the purposes of efficient and effective retention and for more efficient and effective transfer during future learning. The process of integration we define as *consolidation*. The challenge for a LML system is consolidating the knowledge of a new task while retaining and possibly improving knowledge of prior tasks. An interesting aspect of this research is overcoming the *stability-plasticity* problem. The stability-plasticity problem refers to the challenge of adding new information to a system without the loss of prior information [Grossberg, 1987].

Traditional NSI systems tend to focus on methods of retaining symbolic representations of learned knowledge. Consolidation of prior knowledge with new knowledge is not often discussed. A recent survey by Lamb points out that connectionist methods of representing symbolic concepts

is destroying NSI myths that have been around for some time [Lamb, 2008]. This may provide new ground for discovery of methods of consolidating knowledge in NSI systems.

4.3 Curriculum versus Expert Knowledge

An area where NSI and LML research differs is the use of expert knowledge. NSI researchers pride themselves on the ability to prime their learning systems with knowledge provided by an expert user. Such knowledge can be used as a source of inductive bias that leverages the available training examples. In contrast, most LML researchers pride themselves on designing self-taught leaners, and investigating curriculum and training sequences that are beneficial for learning a collection of increasingly complex tasks from as little knowledge as possible.

4.4 Practicing a Task

LML considers systems that may practice one or more tasks over a lifetime. The expectation is that the accuracy of the models of these tasks, residing in common knowledge, increases over time. This is closely related to the idea of knowledge consolidation versus simple retention. A computational theory of how best to practice tasks is important to artificial intelligence, as well as psychology and education. To the best of our knowledge, this is not an area that has been studied by NSI researchers, but their differing perspectives may lead to novel approaches.

4.5 Reasoning versus Learning

NSI systems are able to learn new knowledge, retain it in symbolic form, and then reason with the symbolic representation of the knowledge. They also consider the use of symbolic knowledge for transfer learning. LML systems tend to focus exclusively on knowledge retention for the purposes of transfer learning. LML researchers could benefit from a better understanding of the constraints placed on a learning system when the knowledge acquired must be amenable to reasoning. These constraints can be consider additional inductive biases that may be informative with respect to representation and search when learning.

5 Conclusion

In this paper we have proposed that research in Neural-Symbolic Integration and Lifelong Machine Learning share common ground that can be further exploited for the advancement of artificial intelligence. We have also pointed out several differences in NSI and LML research motives and objectives for further discussion by the community. We encourage researchers to explore these differences as interesting areas for making new discoveries in machine learning, knowledge representation and reasoning. In particular, we foresee that joint research has the potential to make serious advances in the areas of learning and use of common background knowledge that is so important for all areas of artificial intelligence.

A Dagstuhl seminar on Neural-Symbolic Learning and Reasoning is planned for September, 2014 [Garcez *et al.*, 2014]. The goal of the seminar is to build bridges between symbolic and sub-symbolic reasoning and learning representations using computer vision as a catalyst application. We see this as an opportunity for a number of artificial intelligence researchers to consider the links between LML and NSI.

Acknowledgments

This research has been funded, in part, by the Natural Science and Engineering Research Council of Canada. Portions of this paper were extracted from recent joint work with Qiang Yang and Lianghao Li [Silver *et al.*, 2013].

References

[Bader and Hitzler, 2005] Sebastian Bader and Pascal Hitzler. Dimensions of neural-symbolic integration - a structured survey. In *We Will Show Them: Essays in Honour of Dov Gabbay*, pages 167–194. College Publications, 2005.

[Bengio, 2009] Yoshua Bengio. Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[Carlson *et al.*, 2010] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In Maria Fox and David Poole, editors, *AAAI*. AAAI Press, 2010.

[Caruana, 1997] Richard A. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

[Garcez and Gabbay, 2004] Artur S. d'Avila Garcez and Dov M. Gabbay. Fibring neural networks. In Deborah L. McGuinness and George Ferguson, editors, *AAAI*, pages 342–347. AAAI Press / The MIT Press, 2004.

[Garcez *et al.*, 2002] Artur S. d'Avila Garcez, Dov M. Gabbay, and Krysia B. Broda. *Neural-Symbolic Learning System: Foundations and Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[Garcez *et al.*, 2009] Artur S. d'Avila Garcez, Luís C. Lamb, and Dov M. Gabbay. *Neural-Symbolic Cognitive Reasoning*. Cognitive Technologies. Springer, 2009.

[Garcez *et al.*, 2014] Artur Garcez, Marco Gori, Pascal Hitzler, and Luis Lamb. Dagstuhl seminar - neural-symbolic learning and reasoning, September 2014.

[Grossberg, 1987] Stephen Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, 11:23–64, 1987.

[Lamb, 2008] Luís C. Lamb. The grand challenges and myths of neural-symbolic computation. In Luc De Raedt, Barbara Hammer, Pascal Hitzler, and Wolfgang Maass, editors, *Recurrent Neural Networks*, volume 08041 of *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2008.

[Lashkari *et al.*, 1994] Yezdi Lashkari, Max Metral, and Pattie Maes. Collaborative interface agents. In *In Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 444–449. AAAI Press, 1994.

[Le *et al.*, 2012] Quoc Le, Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg Corrado, Jeff Dean, and Andrew Ng. Building high-level features using large scale unsupervised learning. In *International Conference in Machine Learning*, 2012.

[Michalski, 1993] R.S. Michalski. Learning = inferencing + memorizing. *Foundations of Knowledge Acquistion: Machine Learning*, pages 1–41, 1993.

[Mitchell, 1980] Tom. M. Mitchell. The need for biases in learning generalizations. *Readings in Machine Learning*, pages 184–191, 1980. ed. Jude W. Shavlik and Thomas G. Dietterich.

[Raina *et al.*, 2007] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 759–766, New York, NY, USA, 2007. ACM.

[Ring, 1997] Mark B. Ring. Child: A first step towards continual learning. In *Machine Learning*, pages 77–104, 1997.

[Shavlik, 1992] Jude W. Shavlik. A framework for combining symbolic and neural learning. In *Machine Learning*, pages 321–331. Academic Press, 1992.

[Shultz and Rivest, 2001] Thomas R. Shultz and François Rivest. Knowledge-based cascade-correlation: using knowledge to speed learning. *Connect. Sci.*, 13(1):43–72, 2001.

[Silver and Mercer, 2002] Daniel L. Silver and Robert E. Mercer. The task rehearsal method of life-long learning: Overcoming impoverished data. *Advances in Artificial Intelligence, 15th Conference of the Canadian Society for Computational Studies of Intelligence (AI'2002)*, pages 90–101, 2002.

[Silver and Poirier, 2004] Daniel L. Silver and Ryan Poirier. Sequential consolidation of learned task knowledge. *Lecture Notes in Artificial Intelligence, 17th Conference of the Canadian Society for Computational Studies of Intelligence (AI'2004)*, pages 217–232, 2004.

[Silver *et al.*, 2008] Daniel L. Silver, Ryan Poirier, and Duane Currie. Inductive tranfser with context-sensitive neural networks. *Machine Learning*, 73(3):313–336, 2008.

[Silver *et al.*, 2013] Daniel L. Silver, Qiang Yang, and Lianghao Li. Lifelong machine learning systems: Beyond learning algorithms. In *Proceedings of the AAAI Spring Symposium on LifeLong Machine Learning, Stanford University, CA*, pages 49–55. AAAI, 2013.

[Solomonoff, 1989] Ray J. Solomonoff. A system for incremental learning based on algorithmic probability. In *Probability, Proceedings of the Sixth Israeli Conference on Artificial Intelligence, Computer Vision and Pattern Recognition*, pages 515–527, 1989.

[Sutton *et al.*, 2007] Richard S. Sutton, Anna Koop, and David Silver. On the role of tracking in stationary environments. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 871–878, New York, NY, USA, 2007. ACM.

[Tanaka and Yamamura, 1999] Fumihide Tanaka and Masayuki Yamamura. An approach to lifelong reinforcement learning through multiple environments. In *Proc. of the 6th European Workshop on Learning Robots (EWLR-6)*, pages 93–99, 1999.

[Thrun and O'Sullivan, 1995] Sebastian Thrun and J. O'Sullivan. Clustering learning tasks and the selective cross-task transfer of knowledge. *Technical Report CMU-CS-95-209*, Nov 1995.

[Thrun, 1996a] Sebastian Thrun. *Explanation-Based Neural Network Learning: A Lifelong Learning Approach*. Kluwer Academic Publishers, Boston, MA, 1996.

[Thrun, 1996b] Sebastian Thrun. Is learning the n-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems*, pages 640–646. The MIT Press, 1996.

[Thrun, 1997] Sebastian Thrun. Lifelong learning algorithms. *Learning to Learn*, pages 181–209.1 Kluwer Academic Publisher, 1997.

[Tran and Garcez, 2012] S. Tran and A. S. d'Avila Garcez. Logic extraction from deep belief networks. In *Proceedings of ICML Workshop on Representation Learning*, Edinburgh, Scotland, 2012.

[Utgoff, 1983] Paul E. Utgoff. Adjusting bias in concept learning. In *Proceedings of IJCAI-1983*, pages 447–449, 1983.

[Wolpert, 1996] David H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Comput.*, 8(7):1341–1390, October 1996.

[Yang *et al.*, 2009] Qiang Yang, Yuqiang Chen, Gui-Rong Xue, Wenyuan Dai, and Yong Yu. Heterogeneous transfer learning for image clustering via the social web. In *Proc. of the Joint Conf. of the 47th Annual Meeting of the ACL and the 4th Int. Joint Conf. on NLP of the AFNLP: Volume 1*, ACL '09, pages 1–9, Stroudsburg, PA, USA, 2009. Assoc. for Computational Linguistics.

Extending the Associative Rule Chaining Architecture for Multiple Arity Rules

Nathan Burles, James Austin, and Simon O'Keefe
Advanced Computer Architectures Group
Department of Computer Science
University of York
York, YO10 5GH, UK
{nburles,austin,sok}@cs.york.ac.uk

Abstract

The Associative Rule Chaining Architecture uses distributed associative memories and superimposed distributed representations in order to perform rule chaining efficiently [Austin *et al.*, 2012]. Previous work has focused on rules with only a single antecedent, in this work we extend the architecture to work with multiple-arity rules and show that it continues to operate effectively.

1 Introduction

Rule chaining is a common problem in the field of artificial intelligence; searching a set of rules to determine if there is a path from the starting state to the goal state. The Associative Rule Chaining Architecture (ARCA) [Austin *et al.*, 2012] performs rule chaining using correlation matrix memories (CMMs)—a simple type of associative neural network [Kohonen, 1972].

1.1 Rule Chaining

Rule chaining can be used to describe both forward and backward chaining, the choice of which to use is application specific. In this work we use forward chaining, working from the starting state towards a goal state, although there is no reason that the architecture could not be used to perform backward chaining.

In forward chaining, a search begins with an initial set of conditions that are known to be true. All of the rules are searched to find one for which the antecedents match these conditions, and the consequents of that rule are added to the current state. This state is checked to decide if the goal has been reached, and if it has not then the system will iterate. If no further matching rules can be found, then the search will result in failure.

1.2 Correlation Matrix Memories (CMMs)

The CMM is a simple neural network containing a single layer of weights. This means that the input and output neurons are fully connected, and simple Hebbian learning can be used [Ritter *et al.*, 1992].

In this work we use binary CMMs [Willshaw *et al.*, 1969], a sub-class of CMMs where the weights are binary. Learning to associate pairs of binary vectors is thus an efficient operation, and requires only local updates to the CMM. This is formalised in Equation 1, where $\mathbf{M}$ is the resulting CMM (matrix of binary weights), $\mathbf{x}$ is the set of input vectors, $\mathbf{y}$ is the set of output vectors, and n is the number of training pairs. The CMM is formed by taking the logical OR, $\bigvee$, of the outer products formed between each of the training pairs.

$$\mathbf{M} = \bigvee_{i=1}^{n} \mathbf{x}_i \mathbf{y}_i^T \tag{1}$$

A recall operation can be performed as a matrix multiplication between a transposed input vector and the CMM. This results in a non-binary output vector, to which a threshold function f must be applied in order to produce the final binary output vector, as shown in Equation 2.

$$\mathbf{y} = f(\mathbf{x}^T \mathbf{M}) \tag{2}$$

Using the fact that the input vector contains only binary components, this recall operation may be optimised. To calculate the j^{th} bit of an output vector when performing a matrix multiplication, one finds the vector dot product of the input vector $\mathbf{x}$ and the j^{th} column of the matrix $\mathbf{M}$. This vector dot product is defined as $\sum_{i=1}^{n} \mathbf{x}_i \mathbf{M}_{j,i}$, where $\mathbf{M}_{j,i}$ is the value stored in the j^{th} column of the i^{th} row of the CMM $\mathbf{M}$. As $\mathbf{x}$ is known to be a binary vector, it is clear that the result of this dot product is equal to the sum of all values $\mathbf{M}_{j,i}$ where $\mathbf{x}_i = 1$, as shown in Equation 3.

$$y_j = f\left(\sum_{i(\mathbf{x}_i=1)} \mathbf{M}_{j,i}\right) \tag{3}$$

The choice of which threshold function (f) to apply depends on the application and the data representation used. When storing fixed-weight vectors, the L-max threshold has been shown to be effective [Austin, 1996]. Alternatively, when using Baum's algorithm to generate vectors, the L-wta threshold has been shown to increase a CMM's storage capacity [Hobson and Austin, 2009]. When superimposed vectors are used, as they are in ARCA, the selection of threshold function is limited to Willshaw thresholding, where any output bit with a value at least equal to the trained input weight is set to one [Willshaw *et al.*, 1969].

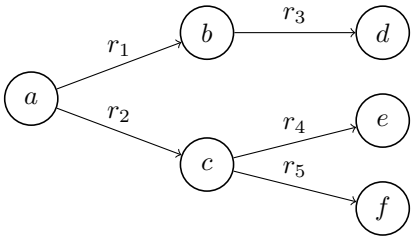


Figure 1: An example tree of rules with a maximum branching factor of 2. Each input token causes at least 1 and at most 2 rules to fire.

Token	Binary vector
a	1001000
b	0100100
c	0010010
d	1000001
e	0101000
f	0010100
g	1000010
h	0100001

Rules	
r_1	$a \to b$
r_2	$a \to c$
r_3	$b \to d$
r_4	$c \to e$
r_5	$c \to f$
r_6	$a \wedge b \to g$
r_7	$a \wedge d \wedge g \to h$
r_8	$a \wedge c \wedge d \to f$

Table 1: An example set of tokens with a binary vector allocated to each, and the set of rules used in examples.

1.3 Associative Rule Chaining

The Associative Rule Chaining Architecture uses distributed representations [Austin, 1992] to allow superposition of multiple states in a single vector, while also providing more efficient memory use and a greater tolerance to faults than a local representation [Baum *et al.*, 1988]. As an example, the superposition of two distributed vectors $\{10100\}$ and $\{10010\}$ is the vector $\{10110\}$.

ARCA performs rule chaining using superimposed representations, and hence reduces the time complexity of a tree search by searching an entire level of the tree in a single operation. A traditional method such as depth-first search would search each rule in turn, resulting in a time complexity of $O(b^d)$, where b is the branching factor, and d is the depth of the tree. By allowing superposition of states, ARCA reduces this to $O(d)$—considering the tree of rules in Figure 1 as an example, if presented with an input a, both rules r_1 and r_2 are considered simultaneously. When the system iterates the next level of rules are then searched concurrently—r_3, r_4, and r_5.

Training

ARCA separates the antecedents and consequents of a rule into two CMMs using a unique "rule vector" that exists in a different vector space to the tokens.

The antecedent CMM associates the antecedents of each rule with its rule vector—for example $a : r_1$. The rule will then fire if its antecedents are present during a later recall.

The consequent CMM then associates each rule with the consequents of that rule. In order to maintain separation between multiple, superimposed branches during a recall this requires a slightly more complex method of training. Firstly we create a tensor, or outer, product between the rule vector and the consequents of a rule—represented as $r_1 : b$. We proceed by "flattening" this tensor product, in a row-major order, with the result being a single vector with a length equal to the product of the token and rule vector lengths. The rule vector can now be associated with this "flattened" tensor product, and stored in the consequent CMM—essentially this stores $r_1 : (b : r_1)$. This means that recalling a rule from the consequent CMM will produce a tensor product that contains the output tokens bound to the rule that caused them to fire.

It should be noted that, depending on the application, it may be possible and appropriate to prune the rule tree prior to training the ARCA network. Reducing the number of rules in this fashion will help to reduce the memory requirement of the system, however will have little effect on its operation.

Recall

Figure 2 shows a single pass through the ARCA system—searching one level of the tree. An input state is initialised by forming TP_{in}—the tensor product of any input tokens (in this case a) with a rule vector (r_0).

To determine which of the rules are matched by our input tokens, we recall each column of this tensor product in turn from the antecedent CMM. Each resulting vector is used as a column in a new tensor product—TP_{rule}. These columns may contain a number of superimposed vectors, representing any rules which fired.

In order to finish this pass through ARCA, we must now find the consequents of any rules which were matched. To do this, we recall each column of TP_{rule} from the consequent CMM. Due to the way this second CMM is trained, the result of each recall is a "flattened" tensor product containing rules bound to output tokens—these can be simply reformed to recover a number of tensor products, TP_{out}. To find our final result, TP_{sum}, we can sum these tensor products and apply a threshold at a value equal to the weight of a rule vector.

2 Multiple Arity Rules

In some cases, allowing only rules with a single antecedent may be sufficient. In complex rule chaining systems or applications such as feature matching, however, rules with more than one antecedent—multiple arity—may be necessary.

Rules with different arities cannot be stored in the same CMM, due to the operation of Willshaw's threshold function and the relaxation ability of CMMs. Table 1 assigns a binary vector to a number of tokens used in our continuing example, with a vector length of 7 and weight of 2.

We can demonstrate this issue by considering a system trained with the example rules r_1 to r_6, shown in Table 1. Upon presentation of a vector containing both a and b $\{1101100\}$ we require the system to match every rule that contains a or b in the antecedents: r_1, r_2, r_3, and r_6. To match the single-arity rules correctly the threshold value used must be equal to the weight of a single input vector, a value of 2.

Using a threshold value of only 2, however, means that presentation of a vector containing only a will match every rule that contains a in the antecedents: r_1, r_2, and r_6. This occurs because of the associative memory's ability to relax and

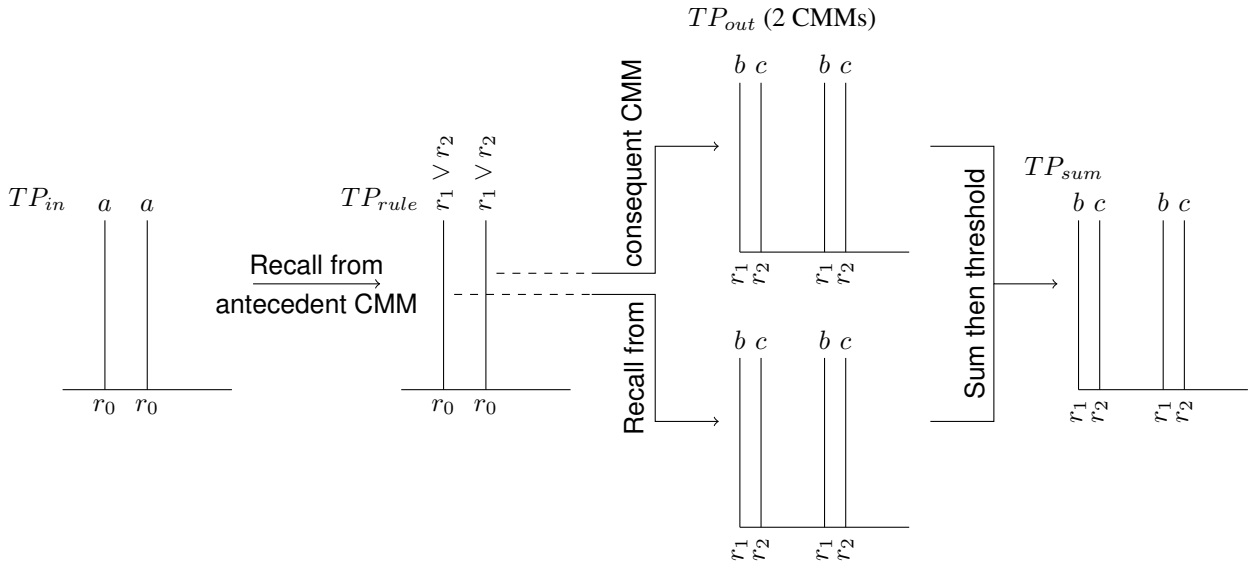


Figure 2: A visualisation of one iteration of the rule chaining process within ARCA. The process is initialised by creating a tensor product, TP_{in}, binding the input tokens (in this case a) to a rule vector (r_0). Each column of this tensor product is then recalled in turn from the antecedent CMM, to form a new tensor product, TP_{rule}, containing the rules which have fired. Each column of TP_{rule} can then be recalled in turn from the consequent CMM, resulting in a number of output tensor products (TP_{out}—one tensor product for every non-zero column of TP_{rule}). These output tensor products can be summed, to form a non-binary CMM, before a threshold is applied using a value equal to the weight of a rule vector to form TP_{sum}. The recall can continue in the same fashion, using TP_{sum} as the new input tensor product.

recognise partial inputs. In this case, however, it is undesirable behaviour as we only wish to match those rules for which all antecedents are present. Setting a threshold to resolve this case is impossible, as any value allowing the single-arity rules to match will also allow the 2-arity rule to match.

2.1 Training

One possible solution for this problem is to use arity networks, introduced in [Austin, 1996] and shown in Figure 3. Under this scheme, multiple distinct ARCA sub-networks are created—one for each arity of rules used in the system. Each rule is then trained into the correct n-arity ARCA sub-network for the number of tokens in its antecedent.

Although this scheme will help in many cases, it still does not fully solve the problem. Consider the 3-arity example rules given in Table 1. When recalling rule r_7 the superimposed tokens will form a vector $\{1001011\}$ with a weight of only 4, thus the threshold for the 3-arity network must be set as 4.

For rule r_8, the superposition of input tokens forms a vector $\{1011011\}$. It can clearly be seen that this vector is very similar to that of rule r_7, with the addition of only a single bit. Unfortunately, we have already determined that the threshold used with the 3-arity network must be at most 4. We can see, therefore, that presentation of the rule r_7 inputs ($a \wedge d \wedge g$) will cause rule r_8 to match incorrectly.

In order to resolve this we propose to modify the networks such that instead of separating rules by arity, they separate the rules by the combined weight of their superimposed antecedent tokens. This will operate in the same way as shown in Figure 3, but each ARCA network will be identified by the total weight of the superimposed antecedent tokens rather than by the number of antecedent tokens.

When training a rule its antecedents are first superimposed, and the weight of this vector determines in which of the ARCA networks the rule should be trained. This means that the threshold value for each ARCA network is well defined, while still allowing for relaxation if this is required by the application (by reducing this threshold value).

2.2 Recall

A block level diagram of the recall operation is shown in Figure 3. To initialise the recall any input tokens are superimposed, for example a, d, and g. The superimposed input vector is then recalled from each of the individual ARCA networks. As each ARCA network is distinct, this recall may happen in parallel where this is supported by the infrastructure. Given this particular input, the rules r_1 and r_2 will match in the weight-2 ARCA network, and r_7 will match in the weight-4 ARCA network. Rule r_8 will not be matched, as it is stored in the weight-5 ARCA network and so requires all 5 set bits to be present in the input.

After recall from each of the ARCA networks, the result is a number of vectors containing the consequent tokens for each of the matched rules. In order to be able to iterate we can simply superimpose these outputs.

Testing for search completion can operate in the same fashion as the single-arity ARCA [Austin *et al.*, 2012]. If the su-

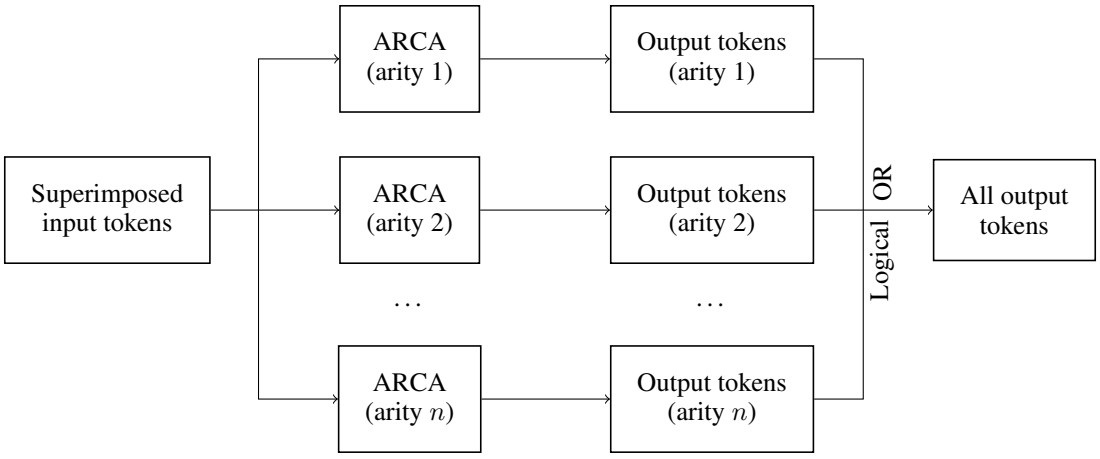


Figure 3: Using arity networks with ARCA. The input tokens are recalled from each arity network in turn, and the output tokens from each network are all superimposed to form the final result.

perimposed output consists solely of zeros then no rules have been matched and hence the search is completed without finding a goal state, if it is not empty then we must check whether all of the goal state's consequent tokens are present in the output—if they are, then the search has been successful.

3 Experiments

In order to show that our proposed solution is effective we implemented it using the Extended Neural Associative Memory Language (ENAMeL), a domain specific language created to simplify the development of applications using binary vectors and CMMs.

We generated a tree of rules, with a depth of 8 and a branching factor of 3 (where the number of children of a given node was uniformly randomly sampled from the range [1, 3]). The number of tokens in the antecedent of each rule, n, was randomly sampled from the range [1, 5], allowing a rule to share at most $\lceil n/2 \rceil$ antecedent tokens with any others. An example of this is shown in Figure 4. We selected these parameters to ensure that this experimentation would remain computationally feasible.

In the worst case these parameters will generate 3^8 rules. To ensure that any potential recall errors were not caused by undesired overlap between vectors, we decided to use a vector length of 5120 and weight of 10—extrapolating from results obtained during previous work [Austin *et al.*, 2012], this should comfortably allow over 10000 rules to be stored.

We then trained the multiple ARCA systems with each of the rules, generating a code for each unique token using Baum's algorithm [Baum *et al.*, 1988]. As we wished to test the correct operation and reliability of the multiple ARCA system, and not ARCA itself, we proceeded as follows:

1. Form the superposition of all tokens in the root of the rule tree to use as the starting state.
2. Recall the current state from each of the ARCA networks in turn, comparing the result to that which was trained—if the result differs from the expected value, then record the error.
3. Superimpose the results from each of the ARCA networks to form a new current state, and repeat from the previous step until all 8 levels of the tree have been searched.

This experimental design meant that if a recall error did occur, we would be able to easily determine the point of failure. We ran this experiment 100 times, using a different random seed each time, to ensure that different rule trees were used.

Our results showed that segregating rules by the weight of their superimposed antecedent tokens is effective, and allows correct operation of the rule chaining system in the presence of multiple-arity rules. In all of the experimental runs the recall was successful, with each ARCA system's output containing the correct tokens and no others.

In one of the runs, however, the output of the weight-40 ARCA network after the third iteration contained 8 set bits in addition to those which were expected. Upon iterating, this did not cause the erroneous recall of further incorrect bits, however with more iterations this could potentially occur again and cause an incorrect recall. The weight-40 ARCA network stored rules with 4 antecedent tokens that have no overlap when superimposed. Upon further analysis of the randomly generated rule tree, it seems that a disproportionately high number of rules were generated with this configuration. This caused the CMMs to become saturated, which means that too many bits in the matrix were set. This can result in erroneous bits being present after a recall operation, due to interference within the matrix.

4 Conclusions and Further Work

This paper has described an important extension to the ARCA, allowing it to perform forward chaining using rules that are not limited to a single antecedent token. We have shown that our proposed solution operates correctly, and is able to successfully store and recall large sets of rules.

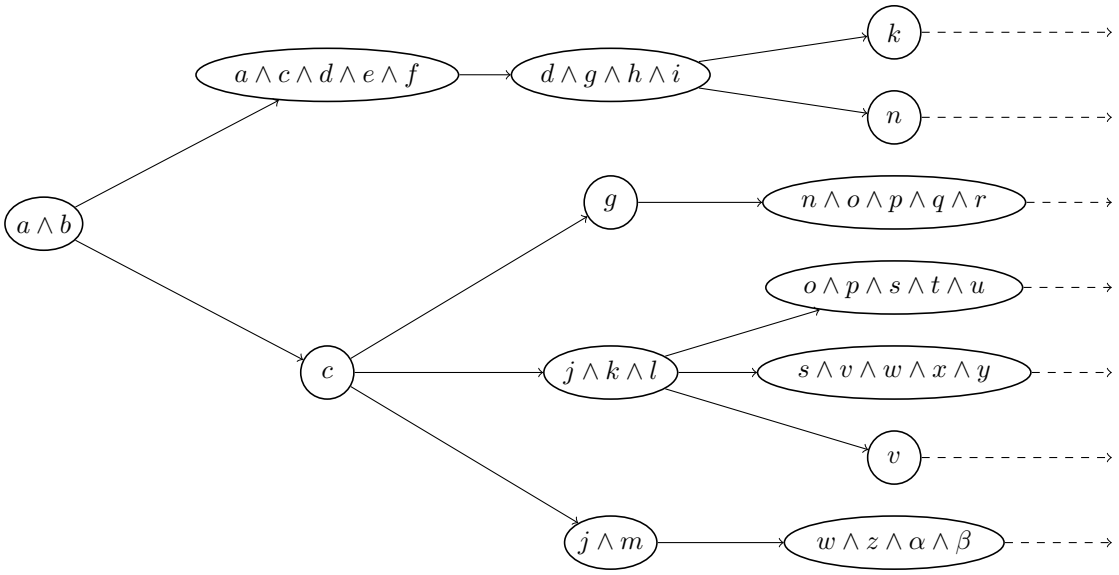


Figure 4: An example tree of rules with a maximum branching factor of 3. Each set of input tokens causes at least 1 and at most 3 rules to fire. Each rule has between 1 and 5 antecedents and consequents and shares at most $\lceil n/2 \rceil$ antecedent tokens with other rules, where n is the number of antecedent tokens of a given rule.

Our initial experimentation was limited to testing only trees of rules with a depth of 8 and branching factor of 3. Having shown that the system is able to operate successfully, a full investigation of these parameters can be undertaken in order to understand the effect that they have on network operation and capacity. Further to this, experimenting with these parameters will also allow a sensitivity analysis to be performed. This is important given the random nature of our experimental inputs, and the potentially large variance between sets of vectors that can occur for a number of reasons—such as the vector generation algorithm used.

In the experimentation we chose to use a vector length of 5120 and weight of 10, as we extrapolated from previous results to determine that this should provide ample capacity. All previous results used rules containing a single token in the antecedent and consequent, so they cannot necessarily be used to predict the capacity when storing rules with greater arities. As the number of tokens involved increases, so do the number of bits set when training a rule—leading to quicker CMM saturation, as we found in one of the experimental runs.

Further investigation into the effect of changing parameters such as the vector length and weight is required, in order to understand how the capacity of each of the ARCA subnetworks changes as well as the memory requirement for the complete system.

Finally, further work is required to understand how the network reacts when reaching saturation; in the experiments performed so far a number of the ARCA sub-networks remained unused, as the weight of the superimposed antecedent tokens was never equal to their value. It may be possible to allocate binary vectors using a different algorithm, in order to either spread the rules more evenly through the sub-networks, or possibly to guarantee that some of them are unnecessary.

References

[Austin *et al.*, 2012] James Austin, Stephen Hobson, Nathan Burles, and Simon OKeefe. A rule chaining architecture using a correlation matrix memory. *Artificial Neural Networks and Machine Learning–ICANN 2012*, pages 49–56, 2012.

[Austin, 1992] James Austin. Parallel distributed computation in vision. In *Neural Networks for Image Processing Applications, IEE Colloquium on*. IET, 1992.

[Austin, 1996] James Austin. Distributed associative memories for high-speed symbolic reasoning. *Fuzzy Sets and Systems*, 82(2):223–233, 1996.

[Baum *et al.*, 1988] Eric B Baum, John Moody, and Frank Wilczek. Internal representations for associative memory. *Biological Cybernetics*, 59(4):217–228, 1988.

[Hobson and Austin, 2009] Stephen Hobson and Jim Austin. Improved storage capacity in correlation matrix memories storing fixed weight codes. *Artificial Neural Networks–ICANN 2009*, pages 728–736, 2009.

[Kohonen, 1972] Teuvo Kohonen. Correlation matrix memories. *Computers, IEEE Transactions on*, 100(4):353–359, 1972.

[Ritter *et al.*, 1992] Helge Ritter, Thomas Martinetz, Klaus Schulten, Daniel Barsky, Marcus Tesch, and Ronald Kates. *Neural computation and self-organizing maps: an introduction*. Addison Wesley Longman Publishing Co., Inc., 1992.

[Willshaw *et al.*, 1969] David J Willshaw, O Peter Buneman, and Hugh Christopher Longuet-Higgins. Non-holographic associative memory. *Nature*, 1969.

Evolution of Connections in SHRUTI Networks

Joe Townsend, Ed Keedwell and Antony Galton
College of Engineering, Mathematics and Physical Sciences
University of Exeter
North Park Road, Exeter
jt231@ex.ac.uk, E.C.Keedwell@ex.ac.uk, A.P.Galton@ex.ac.uk

Abstract

SHRUTI is a model of how predicate relations can be represented and reasoned upon using a network of spiking neurons, attempting to model the brain's ability to perform reasoning using as biologically plausible a means as possible. This paper extends the biological plausibility of the SHRUTI model by presenting a genotype representation of connections in a SHRUTI network using indirect encoding and showing that working networks represented in this way can be produced through an evolutionary process. A multi-objective algorithm is used to minimise the error and the number of weight changes that take place as a network learns.

1 Introduction

Neural-symbolic integration concerns the representation of symbolic information in neural networks [Bader and Hitzler, 2005; Hammer and Hitzler, 2007]. Two motivations of this field are to combine the interpretability of symbolic systems with the adaptability of neural networks, and to produce models that point towards how symbols may be represented in biological neural networks. SHRUTI is a neural-symbolic network which models reasoning in the brain in a way that is claimed to be *biologically plausible* [Shastri and Ajjanagadde, 1993; Shastri, 1999]. The developers of SHRUTI discus the idea that the prerequisite structure required to enable it to learn logical relations can be realised in a way which is itself biologically plausible [Wendelken and Shastri, 2003]. They suggest that a SHRUTI network could be developed by a genotypic representation of rules that direct its growth; this is known as *indirect encoding*. However, the representation of SHRUTI networks using indirect encoding has not been implemented to our knowledge. We show that developmental genomes for creating connections between neurons in SHRUTI networks can be produced through evolution using *artificial development* [Chavoya, 2009], a form of evolutionary computing which uses indirect encoding.

We also want the evolved networks to yield minimal error when presented with test questions and to learn to do so with as few weight updates as possible, thus reducing the workload of the learning algorithm. A multi-objective algorithm was therefore chosen to minimise both of these properties.

2 Background

2.1 SHRUTI

SHRUTI [Shastri and Ajjanagadde, 1993; Shastri, 1999] is a neural-symbolic model of reflexive reasoning which has a number of biologically plausible traits: spiking neurons [Kumar *et al.*, 2010], temporal coding [Kumar *et al.*, 2010], and Hebbian learning [Hebb, 1949]. This section only covers the fundamentals required for SHRUTI to represent quantifier-free predicate logic, perform reasoning and learn relations, but a more complete explanation of SHRUTI can be found in the literature.

SHRUTI takes a localist approach to knowledge representation in that each concept is represented by its own ensemble of spiking neurons. Although a more distributed representation in which each neuron participates in the representation of multiple concepts is a more popular view in neuroscience, the localist view has not been completely ruled out [Bowers, 2009]. The bare minimum SHRUTI requires to work is one neuron per concept, but even by localist standards this lacks biological plausibility, and ensembles are preferred.

Each node in figure 1 represents an ensemble of spiking neurons. A predicate in the logic program is represented by a cluster of these nodes (e.g. $Buy(x, y)$), which contains a role node for each argument (e.g. $buyer$ and $object$) and collector (labelled '+' and '-') and enabler (labelled '?') nodes which direct the path of inference. Separate from the predicate clusters are entity nodes that represent entities which can fulfil the roles of predicate arguments ($John$, $Mary$, $Paul$ and $book$). A dynamic binding between a predicate argument and an entity can be formed by firing the nodes representing them so that their spike trains are in synchrony with each other. This method of temporal coding enables SHRUTI to overcome the variable binding problem. A predicate instance is composed of a set of argument-entity bindings for that predicate. For example, in order to ask the network in figure 1 'Does Mary own the Book?', the predicate $Own(x, y)$ is instantiated as $Own(Mary, Book)$ by firing the *owner* and *object* nodes of *Own* in synchrony with *Mary* and *Book* respectively. Activation of a predicate's enabler triggers a search for the truth of the current predicate instance, and activation of the positive or negative collector nodes asserts the truth or falsity of that instance. If neither collector is activated within a fixed time window, then the truth is regarded as unknown.

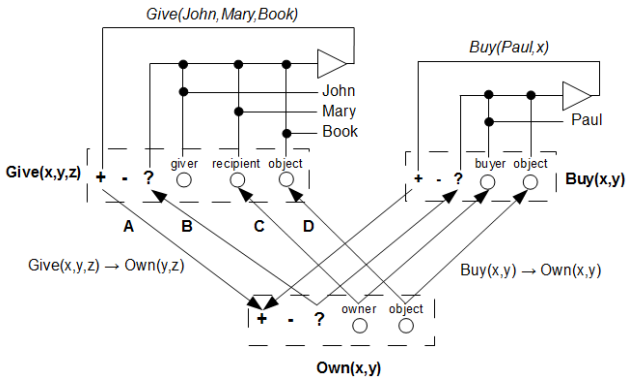


Figure 1: A simple SHRUTI network for the relations *Give(x,y,z)* $\rightarrow$ *Own(y,z)* and *Buy(x,y)* $\rightarrow$ *Own(x,y)*.

Connections between nodes represent relations between the predicates. Instantiating $Own(Mary, Book)$ as described above and firing the enabler of *Own* propagates the bindings to $Give(x, y, z)$ and $Buy(x, y)$ so that *recipient* and *buyer* are now bound to Mary, and the object nodes of each are bound to *Book*. The network is now asking 'Did somebody give Mary the Book?' $(Give(x, Mary, book))$ and 'Did Mary buy the book?' $(Buy(Mary, x))$. These bindings are then propagated further into sub-circuits representing long-term facts which can confirm or deny the truth of the corresponding predicate instance. Each such fact is composed of a set of inhibitory connections (shown as filled circles) and a fact node (shown as a triangle). The fact node only fires when the propagated set of dynamic bindings matches the static bindings encoded by the inhibitory connections. The facts in figure 1 state that John gave Mary the book $(Give(John, Mary, Book))$ and Paul bought something $(Buy(Paul, x))$. Because the dynamic bindings $Give(x, Mary, book)$ match the static bindings for the fact $Give(John, Mary, Book)$, the fact node is activated and in turn activates the positive collector of *Give*. This activation propagates to the positive collector of *Own* to assert that $Own(Mary, Book)$ is true.

Using Hebbian learning [Hebb, 1949], SHRUTI can learn relations between predicates [Wendelken and Shastri, 2003]. The training data takes the form of a sequence of events in the form of predicate instances that reflects causal relations between the predicates. If $P(a, b)$ is observed shortly before $Q(a, b)$, the weights of the connections supporting $P(x, y) \rightarrow Q(x, y)$ are strengthened according to equation 1 in order to reflect the likelihood that P is a cause of Q. However, if $Q(a, b)$ is not observed within a fixed time window of $P(a, b)$ occurring, the same weights are weakened according to equation 2 to reflect the likelihood that P is not a cause of Q. In both cases, the learning rate α is defined according to equation 3, which ensures that when a large amount of evidence has been found to support a relation it becomes more difficult to change its weights.

$$\omega_{t+1} = \omega_t + \alpha * (1 - \omega_t) \quad (1)$$

$$\omega_{t+1} = \omega_t - \alpha * \omega_t \quad (2)$$

$$\alpha = 1/NumberOfUpdates \quad (3)$$

New predicates can also be learned using recruitment learning [Feldman, 1982]. Nodes are divided into two groups: recruited and free. A free node becomes recruited when its connections to other nodes become strong enough, and when enough nodes have been recruited a new predicate is learned.

SHRUTI's learning model means that predicates and relations can be learned from a network of interconnected neurons. However, a fully interconnected network is impractical and lacks biological plausibility. The developers of SHRUTI argue that some pre-organisation of the network is required, and that this pre-organisation could be encoded in a biologically plausible way through genotypic representation [Wendelken and Shastri, 2003]. The idea of genome-instructed development of neural networks is possible through artificial development, but we have not found any literature that presents any attempts to implement the artificial development of SHRUTI networks.

2.2 Artificial Development

Artificial development is a form of evolutionary computing in which the genome encodes the phenotype indirectly by encoding a set of rules for its gradual development [Chavoya, 2009]. This is known as *indirect encoding*, the alternative of which is *direct encoding*, in which the genome encodes the structure of the phenotype explicitly. Indirect encoding is the more biologically plausible of the two, because DNA encodes instructions for the gradual development of an organism. Furthermore, indirect encoding has the advantage of scalability in that the size of the genotype is independent of that of the phenotype, in contrast to direct encoding. Such scalability is acheived through the compact representation of repeated substructures, and therefore indirect encoding is very suitable for the representation of SHRUTI networks, since each relation and fact is represented by a similar sub-circuit.

Artificial development has a number of applications, but the application of relevance to this paper is that of constructing neural networks. Some models for the artificial development of neural networks use graph grammars [Kitano, 1994; Gruau, 1994], which are adapted from Lindenmayer systems [Lindenmayer, 1968]. A number of other approaches to indirect encoding which do not involve graph grammars and exhibit more biological plausibility also exist [Eggenberger, 1997; Hotz *et al.*, 2003; Khan *et al.*, 2010]. In these models, the connections between neurons are often referred to by their biological counterpart of axons, and have positional attributes. A neuron's axons can grow in Euclidean grid space according to the genotypic instructions and connections between neurons are formed when axons meet [Eggenberger, 1997; Hotz *et al.*, 2003; Khan *et al.*, 2010].

3 Evolving SHRUTI Networks

This paper aims to demonstrate that simple SHRUTI networks can be produced through artificial development, supporting the claim of SHRUTI's developers that the prerequisite structure required to learn relations can be realised in a biologically plausible way. However only certain elements of the SHRUTI architecture have been accounted for. Though the current genome model supports the Hebbian learning of

relations, it is not yet capable of developing networks that can produce new predicates through recruitment learning. In general, this genome assumes the pre-existence of neuron clusters representing facts and predicates and therefore does not develop neurons, only the connections between them. Furthermore, each node is implemented with only one neuron and these experiments have yet to be tested on SHRUTI models that represent nodes as ensembles of neurons.

3.1 The SHRUTI Genome

This section presents an existing genome model produced by the authors for developing connections in a SHRUTI network [Townsend *et al.*, 2012]. Figure 2 shows an example set of rules for developing a working SHRUTI network both as they appear in the genome and as they appear in the form of a decision tree, where each rule is represented by a path from the root node to a leaf node. Leaf nodes represent actions to be performed and all other nodes represent conditions necessary for that action to take place. The network is presented with a temporal sequence of predicate instances supporting a set of causal relations. At each time t, all predicate instances occurring at t are observed and any existing connection weights are updated according to SHRUTI's Hebbian learning algorithm. The rules in the genome are then assessed for each possible node pair and the actions of any satisfied rules are executed. These actions may be the addition of a connection with a specified weight, or the removal of an existing connection. Each node in a possible node pair has a different label in the genome. 'SELF' refers to the node for which an input connection is being considered, and the node from which the connection is being considered is labelled 'P_INPUT' (possible input) if the connection does not exist or 'E_INPUT' (existing input) if it does exist.

The genome is a string of elements that describes the structure of the decision tree. Each sub-string describes one node (condition or action) so that each node is referenced by the index of the corresponding sub-string. Each condition's sub-string contains an element for the attribute to be tested, the operator ($<$,$\leq$,$=$,$\geq$,$>$,$\neq$), the value to test that attribute against, and the indices of the action or condition to branch to when the current condition is evaluated as true or false. For example, the first sub-string in the genome represents condition 1, and can be interpreted as follows: if the activity of SELF is above 0.5, branch to condition (sub-string) 4, otherwise branch to condition 3. If an index of 0 is specified, then the search terminates. Genomes may also contain conditions or actions which although not currently expressed (e.g. condition 2 in figure 2), may come to be if an index gene is mutated to branch to it.

Attributes which can be tested in this model are the node's current level of activity, its type (role, enabler or collector), the total number of inputs, the remaining time before the time window for coactivation closes, and the firing delay, which corresponds to a node's current phase. Additional attributes may be tested for existing inputs: the weight and the number of updates (how many times a connection has been strengthened or weakened).

The genome in figure 2 contains two rules. Rule 1 removes redundant connections and rule 2 establishes connections between two active neurons of the same type. The genome was tested on a number of different event sequences, each supporting sets of relations of different sizes. Each network developed was tested with a set of 'true or false' questions in the form of predicate instances and was found to answer all of these questions correctly. Statistics for the networks developed from these event sequences are presented in table 1. Even though sets may contain the same number of relations and predicates (e.g. sets 5 to 7), the number of arguments in each predicate may differ, resulting in different statistics for these sets. Although the size of the genome is fixed, the size of each developed network is different, showing that the size of the genotype is independent of the size of the phenotype and that the genome is therefore scalable.

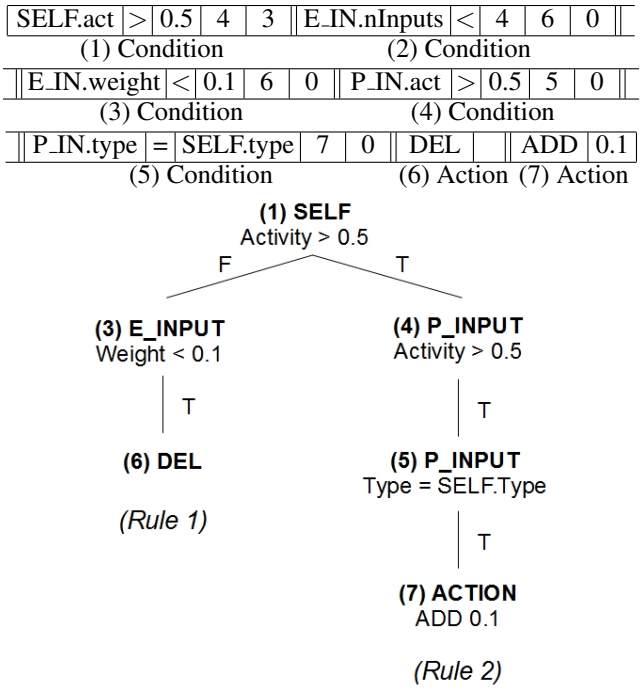


Figure 2: Rules for developing SHRUTI networks as they appear in the genome and as they appear in a decision tree.

Table 1: Number of connections and weight updates for networks produced by the genome in figure 2.

Set	Relations	Predicates	Connections	Updates
1	2	3	22	139
2	3	4	64	719
3	4	5	86	1056
4	4	7	53	535
5	5	6	65	721
6	5	6	80	852
7	5	6	83	1064
8	6	7	80	1168
9	6	7	73	730

3.2 Fitness Function

The aim of evolving SHRUTI networks was to minimise both the area beneath the error-time graph and the number of weight updates performed in training the network.

As the network develops, error will change during development as it learns more relations. Therefore the area beneath the error-time graph (which will henceforth be referred to as *e-area*) was chosen as an objective in order to encourage the algorithm to converge not only towards networks of minimum error but towards networks that can achieve minimum error as early as possible. To calculate an approximation of the e-area, error was measured at five intervals during the development of the network.

Error was measured as the difference between maximum accuracy and the accuracy obtained. Accuracy is based on how many questions the network is capable of answering correctly, and therefore maximum accuracy is achieved by answering all questions correctly. A question takes the form of a predicate instance, for example $P(a, b)$, which a developed network must assert as true, false, or unknown. However, rather than simply counting the number of correct answers, accuracy is a function on the number of correct collector activations. Each answer to a question consists of a positive and negative collector state, and therefore takes one of four values: [1,0] (true), [0,1] (false), [0,0] (unknown) and [1,1] (contradiction).

Answers are contained in a matrix a, where each row a_i represents one answer and contains an element for each collector activation. Matrix a is compared against a target matrix t. Each row a_i in the answer matrix is assigned a score and accuracy is measured as the sum of these scores as in equation 4. In the original scoring function S_1, a score of 1 was assigned to each correct collector activation as shown in equation 5. However, this resulted in high accuracies even when all questions were answered as unknown. All correct answers contain at least one inactive collector, and therefore by simply guessing [0,0] (unknown) for all questions the total accuracy could be a high percentage of the maximum accuracy as shown in table 2 (6 compared to a maximum accuracy of 8). Furthermore, this is higher than the accuracy obtained by guessing [1,1] for all questions, which may be considered 'better' from an evolutionary perspective since such networks are at least trying to answer questions. To overcome this problem, the score function was modified to create a new function S_2 as shown in equation 6, which gives a total accuracy of 0 to networks which always answer [0,0], and adds more weight to correct 'true' or 'false' questions by assigning them a score of 3 as opposed to just 2. Table 2 shows that by using S_2, networks only answering [0,0] are now assigned the absolute minimum accuracy, and that the accuracy of networks answering all questions correctly is even greater than before.

$$Accuracy(a) = \sum_{i=0}^{n} S(a_i) \tag{4}$$

$$S_1(a_i) = \sum_{j=0}^{m} 1 - |t_{i,j} - a_{i,j}| \tag{5}$$

$$S_2(a_i) = \begin{cases} 0 & \text{if } \sum_{i=0}^{n} \sum_{j=0}^{m} a_{i,j} = 0 \\ 3 & \text{if } \sum_{j=0}^{m} t_{i,j} = 1 \text{ and } S_1(a_i) = 2 \\ S_1(a_i) & \text{otherwise} \end{cases} \tag{6}$$

Table 2: A comparison of the outputs of two different scoring functions. Numbers given in bold denote total accuracies.

Expected answer		Score 1 (S_1)			Score 2 (S_2)		
+	-	All 0,0	All 1,1	All correct	All 0,0	All 1,1	All correct
1	0	1	1	2	0	1	3
0	1	1	1	2	0	1	3
0	0	2	0	2	0	0	2
0	0	2	0	2	0	0	2
		6	**2**	**8**	**0**	**2**	**10**

In addition to minimising e-area, the second objective was to minimise the number of weight updates performed on the network. Minimising this reduces the workload of the learning algorithm and constrains the number of connections in the network, because as the number of connections in a network increases, so does the number of the weights the algorithm has to update.

3.3 Evolutionary Algorithm

Given that this is a multi-objective problem, NSGA-II [Deb *et al.*, 2002] was chosen to evolve the networks. The algorithm was run with an initial population size of 100 over 500 generations, repeated over 50 trials. Genomes were fixed at a size of twelve conditions or actions and binary tournament selection with replacement was used to select genomes for recombination. Crossover was performed at a rate of 90% by randomly swapping a sub-tree from each parent. Children were then mutated by selecting sub-strings representing nodes with a probability of 10%, randomly choosing one of the elements from each and choosing new values according to a uniform distribution.

Fitness was calculated on a set of training questions and the performance of the final population was tested against a set of test questions. Set 7 from table 1 was chosen as the event sequence used for training and testing evolved networks. The set of training questions contained the minimum set of questions required to demonstrate that the desired relations had been learned correctly. In this case, the set contained three questions for which the expected answer was 'true', three expected to be 'false' and seven expected to be 'unknown', totalling thirteen questions and a maximum accuracy of 32 according to the scoring function. The test questions contained all other possible questions with the exception of predicate instances which were already encoded as facts, since these would always be answered correctly.

4 Results

4.1 Performance on Training Questions

Figure 3 shows samples obtained from 50 trials when minimising the e-area and the number of weight updates performed by a network during the learning process. Points marked with a dot in figure 3 indicate genomes which developed networks capable of answering all training questions correctly. 224 zero-error networks were found across 49 of the trials. In general, three very distinct groups of networks emerged, each of which was found to employ a different strategy for answering questions. These groups are indicated by the three boxes in figure 3. Each group's behaviour was analysed by sampling ten genomes from each.

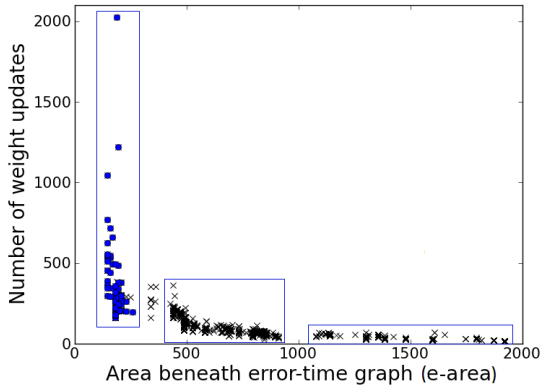


Figure 3: Points obtained from 50 trials on training data. Points marked with a dot represent genomes that answered all questions correctly.

First group: 0 < e-area < 250

Errors in this group range between 0 and 2, though the error of the majority is 0. Genomes which construct these networks behave like rule 2 from the genome constructed in section 3.1, in that connections between SELF and P_INPUT are produced when both nodes are active and of the same type. However, equivalent conditions or combinations of conditions also emerged, rather than testing activity or type directly (figure 4(a)).

Where the number of updates is greater, at least one of the conditions that restrict the number of connections may be missing from the rules. For example, if the genome in figure 2 bypassed condition 4, i.e. did not require that P_INPUT was active in order for a connection to be created, a maximum accuracy network would still be developed. However the resulting network would contain a number of superfluous connections, therefore increasing the number of weight change operations the network has to perform.

Second group: 400 < e-area < 900

Errors in this group range between 5 and 15. However only one genome is found for errors of both 5 and 15, so the boundaries of any real interest are networks with errors of 6 and 14. Answers given by networks in this range depend only on the predicate queried without reference to its arguments. For example, the network might always answer 'true' for predicate P, 'false' for predicate Q and 'unknown' for predicate R. As error increases, questions for a greater number of predicates were always answered with 'unknown', from networks with an error of 6 and up to networks with an error of 14. 14-error networks always answer 'true' or 'false' for only one of the predicates and 'unknown' for everything else. Anything less will result in all questions being answered 'unknown' and the genome's performance will be penalised to the maximum error of 32. For all of these genomes, connections were only formed between enablers and collectors, but there was some restriction on the number of inputs that a node could have before a connection could be added. Figure 4(b)) shows a genome for constructing networks that yield an error of six.

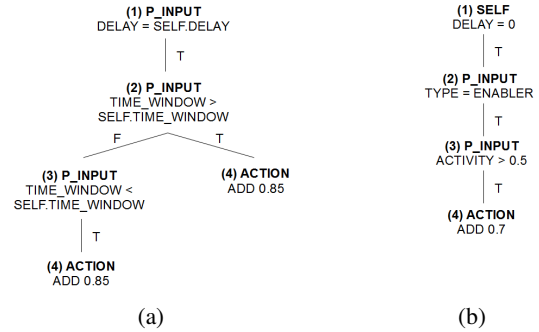


Figure 4: Example genomes for constructing networks from the first (a) and second (b) groups.

Third group: 1050 < e-area < 1900

Errors in this group range between 11 and 32, though the error of the majority is 32, i.e. the maximum possible error. In the maximum-error genomes in this cluster with the largest e-areas, any *add* actions they contained were never expressed. Therefore connections are never created, resulting in empty networks which always answer [0,0] (unknown) for every question at any time. As a result, the error and the e-area are always maximum whilst the number of weight updates is minimal. Maximum-error networks with a smaller e-area construct some connections and temporarily yield a non-maximal error, but remove these connections after a time and yield maximum error at the end of development.

4.2 Performance on Test Questions

The evolved genomes were tested by asking the developed networks a set of test questions which was larger than the set of training questions. Figure 5 shows how the networks performed on the test set. The general shapes of the Pareto fronts remain roughly the same for both sets. In particular, every genome capable of developing networks which answered all training questions correctly could also answer every test question correctly. Because most of the evolved networks perform as well in the test questions as they do on the training questions, these results demonstrate that the performance of evolved genomes is robust to unseen test questions.

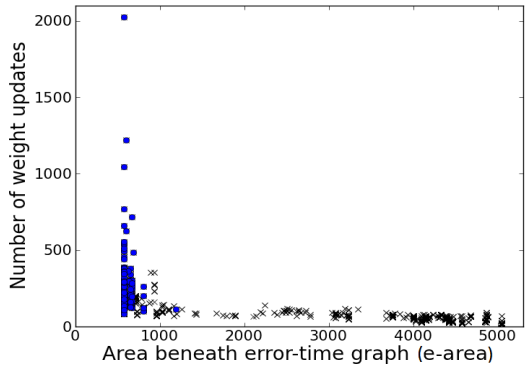


Figure 5: Points obtained by running genomes from 50 trials on test data.

5 Conclusions

Using NSGA-II, three groups of genomes for developing connections in SHRUTI networks emerged, each with its own distinct strategy for answering questions. One of these groups was successful in producing networks that behaved like regular SHRUTI networks, answering all questions correctly based on evidence learned from observed events. One claim of SHRUTI's developers was that reasoning can be the spontaneous and natural outcome of a system of neurons, and the findings in this paper support the idea that this itself can be the outcome of an evolutionary process. This supports another claim of the SHRUTI developers that the prerequisite structure required to enable the learning of relations in SHRUTI can be realised through a model of biological development, adding another dimension of biological plausibility to the model. However, the results only account for the basic SHRUTI model. The genome does not yet support nodes formed of multiple neurons, the creation of new neurons and the formation of new predicates through recruitment learning. To support the idea of a developmental SHRUTI model even further, we propose to investigate this.

References

[Bader and Hitzler, 2005] Sebastian Bader and Pascal Hitzler. *Dimensions of Neural-Symbolic Integration: A Structured Survey*, volume 1, pages 167–194. College Publications, London, 2005.

[Bowers, 2009] Jeffrey S Bowers. On the biological plausibility of grandmother cells: Implications for neural network theories in psychology and neuroscience. *Psychological Review*, 116(1), 2009.

[Chavoya, 2009] Arturo Chavoya. *Artificial Development*, volume 1 of *Studies in Computational Intelligence*, pages 185–215. Springer, 2009.

[Deb *et al.*, 2002] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[Eggenberger, 1997] Peter Eggenberger. Creation of neural networks based on developmental and evolutionary principles. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 337–342. Springer-Verlag, 1997.

[Feldman, 1982] Jerome A. Feldman. Dynamic connections in neural networks. *Biological Cybernetics*, 46:27–39, 1982.

[Gruau, 1994] Frédéric Gruau. Automatic definition of modular neural networks. *Adaptive Behaviour*, 3(2):151–183, 1994.

[Hammer and Hitzler, 2007] Barbara Hammer and Pascal Hitzler. *Perspectives of Neural-Symbolic Integration*. Springer, Berlin, 2007.

[Hebb, 1949] Donald Olding Hebb. *The Organization of Behavior: A neuropsychological theory*. Wiley, New York, 1949.

[Hotz *et al.*, 2003] Peter Eggenberger Hotz, Gabriel Gómez, and Rolf Pfeifer. Evolving the morphology of a neural network for controlling a foveating retina - and its test on a real robot. In *Proceedings of The Eighth International symposium on artificial life*, pages 243–251, 2003.

[Khan *et al.*, 2010] Gul Muhammad Khan, Julian F. Miller, and David M. Halliday. *Intelligent agents capable of developing memory of their environment*, pages 77–114. UEFS, 2010.

[Kitano, 1994] Hiroaki Kitano. Neurogenetic learning: an integrated method of designing and training neural networks using genetic algorithms. *Physica D: Nonlinear Phenomena*, 75(1-3):225–238, 1994.

[Kumar *et al.*, 2010] Arvind Kumar, Stefan Rotter, and Ad Aertsen. Spiking activity propagation in neuronal networks: reconciling different perspectives on neural coding. *Nature Reviews Neuroscience*, 11(9), 2010.

[Lindenmayer, 1968] Aristid Lindenmayer. Mathematical models for cellular interactions in development. *Journal of Theoretical Biology*, 18(3):280–299, 1968.

[Shastri and Ajjanagadde, 1993] Lokendra Shastri and Venkat Ajjanagadde. From simple associations to systematic reasoning. *Behavioral and Brain Sciences*, 16(3):417–494, 1993.

[Shastri, 1999] L. Shastri. Advances in SHRUTI - a neurally motivated model of relational knowledge representation and rapid inference using temporal synchrony. *Applied Intelligence*, 11:79–108, 1999.

[Townsend *et al.*, 2012] Joe Townsend, Ed Keedwell, and Antony Galton. A scalable genome representation for neural-symbolic networks. Birmingham, 2012. Proceedings of the First Symposium on Nature Inspired Computing and Applications (NICA) at the AISB/IACAP World Congress 2012.

[Wendelken and Shastri, 2003] Carter Wendelken and Lokendra Shastri. Acquisition of concepts and causal rules in shruti. In *Proceedings of the Twenty Fifth Annual Conference of the Cognitive Science Society*, Boston, MA, 2003. Proceedings of the Twenty Fifth Annual Conference of the Cognitive Science Society.

On Chomsky and the Two Cultures of Statistical Learning

Peter Norvig

Director of Research, Google Inc., USA
pnorvig@google.com

At the Brains, Minds, and Machines symposium held during MIT's 150th birthday party, Technology Review reports that Prof. Noam Chomsky

> derided researchers in machine learning who use purely statistical methods to produce behavior that mimics something in the world, but who don't try to understand the meaning of that behavior.

The transcript is now available, so let's quote Chomsky himself:

> It's true there's been a lot of work on trying to apply statistical models to various linguistic problems. I think there have been some successes, but a lot of failures. There is a notion of success ... which I think is novel in the history of science. It interprets success as approximating unanalyzed data.

This essay discusses what Chomsky said, speculates on what he might have meant, and tries to determine the truth and importance of his claims.

Chomsky's remarks were in response to Steven Pinker's question about the success of probabilistic models trained with statistical methods.

1. What did Chomsky mean, and is he right?
2. What is a statistical model?
3. How successful are statistical language models?
4. Is there anything like their notion of success in the history of science?
5. What doesn't Chomsky like about statistical models?

What did Chomsky mean, and is he right?

I take Chomsky's points to be the following:

A. Statistical language models have had engineering success, but that is irrelevant to science.
B. Accurately modeling linguistic facts is just butterfly collecting; what matters in science (and specifically linguistics) is the underlying principles.
C. Statistical models are incomprehensible; they provide no insight.
D. Statistical models may provide an accurate simulation of some phenomena, but the simulation is done completely the wrong way; people don't decide what the third word of a sentence should be by consulting a probability table keyed on the previous two words, rather they map from an internal semantic form to a syntactic tree-structure, which is then linearized into words. This is

done without any probability or statistics.

E. Statistical models have been proven incapable of learning language; therefore language must be innate, so why are these statistical modelers wasting their time on the wrong enterprise?

Is he right? That's a long-standing debate. These are my answers:

A. I agree that engineering success is not the goal or the measure of science. But I observe that science and engineering develop together, and that engineering success shows that something is working right, and so is evidence (but not proof) of a scientifically successful model.

B. Science is a combination of gathering facts and making theories; neither can progress on its own. I think Chomsky is wrong to push the needle so far towards theory over facts; in the history of science, the laborious accumulation of facts is the dominant mode, not a novelty. The science of understanding language is no different than other sciences in this respect.

C. I agree that it can be difficult to make sense of a model containing billions of parameters. Certainly a human can't understand such a model by inspecting the values of each parameter individually. But one can gain insight by examing the *properties* of the model—where it succeeds and fails, how well it learns as a function of data, etc.

D. I agree that a Markov model of word probabilities cannot model all of language. It is equally true that a concise tree-structure model without probabilities cannot model all of language. What is needed is a probabilistic model that covers words, trees, semantics, context, discourse, etc. Chomsky dismisses all probabilistic models because of shortcomings of particular 50-year old models. I understand how Chomsky arrives at the conclusion that probabilistic models are unnecessary, from his study of the generation of language. But the vast majority of people who study *interpretation* tasks, such as speech recognition, quickly see that interpretation is an inherently probabilistic problem: given a stream of noisy input to my ears, what did the speaker most likely mean? Einstein said to make everything as simple as possible, but no simpler. Many phenomena in science are stochastic, and the simplest model of them is a probabilistic model; I believe language is such a phenomenon and therefore that probabilistic models are our best tool for representing facts about language, for algorithmically processing language, and for understanding how humans process language.

E. In 1967, Gold's Theorem showed some theoretical limitations of logical deduction on formal mathematical languages. But this result has nothing to do with the task faced by learners of natural language. In any event, by 1969 we knew that probabilistic inference (over probabilistic context-free grammars) is not subject to those limitations (Horning showed that learning of PCFGs is possible). I agree with Chomsky that it is undeniable that humans have some innate capability to learn natural language, but we don't know enough about that capability to rule out probabilistic language representations, nor statistical learning. I think it is much more likely that human language learning involves something like probabilistic and statistical inference, but we just don't know yet.

Now let me back up my answers with a more detailed look at the remaining questions.

What is a statistical model?

A **statistical model** is a mathematical model which is modified or trained by the input of data points. Statistical models are often but not always probabilistic. Where the distinction is important we will be careful not to just say "statistical" but to use the following component terms:

- A **mathematical model** specifies a relation among variables, either in functional form that maps inputs to outputs (e.g. $y = m\,x + b$) or in relation form (e.g. the following (x, y) pairs are part of the relation).
- A **probabilistic model** specifies a probability distribution over possible values of random variables, e.g., $P(x, y)$, rather than a strict deterministic relationship, e.g., $y = f(x)$.
- A **trained model** uses some training/learning algorithm to take as input a collection of possible models and a collection of data points (e.g. (x, y) pairs) and select the best model. Often this is in the form of choosing the values of parameters (such as m and b above) through a process of statistical inference.

For example, a decade before Chomsky, Claude Shannon proposed probabilistic models of communication based on Markov chains of words. If you have a vocabulary of 100,000 words and a second-order Markov model in which the probability of a word depends on the previous two words, then you need a quadrillion (10^{15}) probability values to specify the model. The only feasible way to learn these 10^{15} values is to gather statistics from data and introduce some smoothing method for the many cases where there is no data. Therefore, most (but not all) probabilistic models are trained. Also, many (but not all) trained models are probabilistic.

As another example, consider the Newtonian model of gravitational attraction, which says that the force between two objects of mass m_1 and m_2 a distance r apart is given by

$$F = G\,m_1\,m_2 / r^2$$

where G is the universal gravitational constant. This is a trained model because the gravitational constant G is determined by statistical inference over the results of a series of experiments that contain stochastic experimental error. It is also a deterministic (non-probabilistic) model because it states an exact functional relationship. I believe that Chomsky has no objection to this kind of statistical model. Rather, he seems to reserve his criticism for statistical models like Shannon's that have quadrillions of parameters, not just one or two.

(This example brings up another distinction: the gravitational model is **continuous** and **quantitative** whereas the linguistic tradition has favored models that are **discrete**, **categorical**, and **qualitative**: a word is or is not a verb, there is no question of its degree of verbiness. For more on these distinctions, see Chris Manning's article on Probabilistic Syntax.)

A relevant probabilistic statistical model is the ideal gas law, which describes the pressure P of a gas in terms of the the number of molecules, N, the temperature T, and Boltzmann's constant, K:

$$P = N\,k\,T / V.$$

The equation can be derived from first principles using the tools of statistical mechanics. It is an uncertain, incorrect model; the *true* model would have to describe the motions of individual gas molecules. This model ignores that complexity and *summarizes* our uncertainty about the location of individual molecules. Thus, even though it is statistical and probabilistic, even though it does not completely model reality, it does provide both good predictions and insight—insight that is not available from trying to understand the *true* movements of individual molecules.

Now let's consider the non-statistical model of spelling expressed by the rule "*I before E except after*

C." Compare that to the probabilistic, trained statistical model:

```
P(IE) = 0.0177          P(CIE) = 0.0014         P(*IE) = 0.163
P(EI) = 0.0046          P(CEI) = 0.0005         P(*EI) = 0.0041
```

This model comes from statistics on a corpus of a trillion words of English text. The notation P(IE) is the probability that a word sampled from this corpus contains the consecutive letters "IE." P(CIE) is the probability that a word contains the consecutive letters "CIE", and P(*IE) is the probability of any letter other than C followed by IE. The statistical data confirms that IE is in fact more common than EI, and that the dominance of IE lessens wehn following a C, but contrary to the rule, CIE is still more common than CEI. Examples of "CIE" words include "science," "society," "ancient" and "species." The disadvantage of the "I before E except after C" model is that it is not very accurate. Consider:

```
Accuracy("I before E") = 0.0177/(0.0177+0.0046) = 0.793
Accuracy("I before E except after C") = (0.0005+0.0163)/
(0.0005+0.0163+0.0014+0.0041) = 0.753
```

A more complex statistical model (say, one that gave the probability of all 4-letter sequences, and/or of all known words) could be ten times more accurate at the task of spelling, but offers little **insight** into what is going on. (Insight would require a model that knows about phonemes, syllabification, and language of origin. Such a model could be trained (or not) and probabilistic (or not).)

As a final example (not of statistical models, but of insight), consider the Theory of Supreme Court Justice Hand-Shaking: when the supreme court convenes, all attending justices shake hands with every other justice. The number of attendees, *n*, must be an integer in the range 0 to 9; what is the total number of handshakes, *h* for a given *n*? Here are three possible explanations:

A. Each of *n* justices shakes hands with the other *n* - 1 justices, but that counts Alito/Breyer and Breyer/Alito as two separate shakes, so we should cut the total in half, and we end up with $h = n \times (n - 1) / 2$.
B. To avoid double-counting, we will order the justices by seniority and only count a more-senior/more-junior handshake, not a more-junior/more-senior one. So we count, for each justice, the shakes with the more junior justices, and sum them up, giving $h = \Sigma_{i = 1 \,..\, n} (i - 1)$.
C. Just look at this table:

n:	0	1	2	3	4	5	6	7	8	9
h:	0	0	1	3	6	10	15	21	28	36

Some people might prefer A, some might prefer B, and if you are slow at doing multiplication or addition you might prefer C. Why? All three explanations describe *exactly the same theory* — the same function from *n* to *h*, over the entire domain of possible values of *n*. Thus we could prefer A (or B) over C only for reasons other than the theory itself. We might find that A or B gave us a better understanding of the problem. A and B are certainly more useful than C for figuring out what happens if Congress exercises its power to add an additional associate justice. Theory A might be most helpful in developing a theory of handshakes at the end of a hockey game (when each player shakes hands with players on the opposing team) or in proving that the number of people who shook an odd number of hands at the MIT Symposium is even.

How successful are statistical language models?

Chomsky said words to the effect that statistical language models have had some limited success in some application areas. Let's look at computer systems that deal with language, and at the notion of "success" defined by "making accurate predictions about the world." First, the major application areas:

- **Search engines:** 100% of major players are trained and probabilistic. Their operation cannot be described by a simple function.
- **Speech recognition:** 100% of major systems are trained and probabilistic, mostly relying on probabilistic hidden Markov models.
- **Machine translation:** 100% of top competitors in competitions such as NIST use statistical methods. Some commercial systems use a hybrid of trained and rule-based approaches. Of the 4000 language pairs covered by machine translation systems, a statistical system is by far the best for every pair except Japanese-English, where the top statistical system is roughly equal to the top hybrid system.
- **Question answering:** this application is less well-developed, and many systems build heavily on the statistical and probabilistic approach used by search engines. The IBM Watson system that recently won on Jeopardy is thoroughly probabilistic and trained, while Boris Katz's START is a hybrid. All systems use at least some statistical techniques.

Now let's look at some components that are of interest only to the computational linguist, not to the end user:

- **Word sense disambiguation:** 100% of top competitors at the SemEval-2 competition used statistical techniques; most are probabilistic; some use a hybrid approach incorporating rules from sources such as Wordnet.
- **Coreference resolution:** The majority of current systems are statistical, although we should mention the system of Haghighi and Klein, which can be described as a hybrid system that is mostly rule-based rather than trained, and performs on par with top statistical systems.
- **Part of speech tagging:** Most current systems are statistical. The Brill tagger stands out as a successful hybrid system: it learns a set of deterministic rules from statistical data.
- **Parsing:** There are many parsing systems, using multiple approaches. Almost all of the most successful are statistical, and the majority are probabilistic (with a substantial minority of deterministic parsers).

Clearly, it is inaccurate to say that statistical models (and probabilistic models) have achieved *limited* success; rather they have achieved a *dominant* (although not exclusive) position.

Another measure of success is the degree to which an idea captures a community of researchers. As Steve Abney wrote in 1996, "In the space of the last ten years, statistical methods have gone from being virtually unknown in computational linguistics to being a fundamental given. ... anyone who cannot at least use the terminology persuasively risks being mistaken for kitchen help at the ACL [Association for Computational Linguistics] banquet."

Now of course, the majority doesn't rule -- just because everyone is jumping on some bandwagon, that doesn't make it right. But I made the switch: after about 14 years of trying to get language models to work using logical rules, I started to adopt probabilistic approaches (thanks to pioneers like Gene Charniak (and Judea Pearl for probability in general) and to my colleagues who were early adopters,

like Dekai Wu). And I saw everyone around me making the same switch. (And I didn't see anyone going in the other direction.) We all saw the limitations of the old tools, and the benefits of the new.

And while it may seem crass and anti-intellectual to consider a financial measure of success, it is worth noting that the intellectual offspring of Shannon's theory create several trillion dollars of revenue each year, while the offspring of Chomsky's theories generate well under a billion.

This section has shown that one reason why the vast majority of researchers in computational linguistics use statistical models is an *engineering* reason: statistical models have state-of-the-art performance, and in most cases non-statistical models perform worst. For the remainder of this essay we will concentrate on *scientific* reasons: that probabilistic models better represent linguistic facts, and statistical techniques make it easier for us to make sense of those facts.

Is there anything like [the statistical model] notion of success in the history of science?

When Chomsky said "*That's a notion of [scientific] success that's very novel. I don't know of anything like it in the history of science*" he apparently meant that the notion of success of "accurately modeling the world" is novel, and that the only true measure of success in the history of science is "providing insight" — of answering *why* things are the way they are, not just describing *how* they are.

A dictionary definition of science is "the systematic study of the structure and behavior of the physical and natural world through observation and experiment," which stresses accurate modeling over insight, but it seems to me that both notions have always coexisted as part of doing science. To test that, I consulted the epitome of doing science, namely *Science*. I looked at the current issue and chose a title and abstract at random:

> Chlorinated Indium Tin Oxide Electrodes with High Work Function for Organic Device Compatibility
>
> In organic light-emitting diodes (OLEDs), a stack of multiple organic layers facilitates charge flow from the low work function [~4.7 electron volts (eV)] of the transparent electrode (tin-doped indium oxide, ITO) to the deep energy levels (~6 eV) of the active light-emitting organic materials. We demonstrate a chlorinated ITO transparent electrode with a work function of >6.1 eV that provides a direct match to the energy levels of the active light-emitting materials in state-of-the art OLEDs. A highly simplified green OLED with a maximum external quantum efficiency (EQE) of 54% and power efficiency of 230 lumens per watt using outcoupling enhancement was demonstrated, as were EQE of 50% and power efficiency of 110 lumens per watt at 10,000 candelas per square meter.

It certainly seems that this article is much more focused on "accurately modeling the world" than on "providing insight." The paper does indeed fit in to a body of theories, but it is mostly reporting on specific experiments and the results obtained from them (e.g. efficiency of 54%).

I then looked at all the titles and abstracts from the current issue of *Science*:

- Comparative Functional Genomics of the Fission Yeasts
- Dimensionality Control of Electronic Phase Transitions in Nickel-Oxide Superlattices
- Competition of Superconducting Phenomena and Kondo Screening at the Nanoscale

- Chlorinated Indium Tin Oxide Electrodes with High Work Function for Organic Device Compatibility
- Probing Asthenospheric Density, Temperature, and Elastic Moduli Below the Western United States
- Impact of Polar Ozone Depletion on Subtropical Precipitation
- Fossil Evidence on Origin of the Mammalian Brain
- Industrial Melanism in British Peppered Moths Has a Singular and Recent Mutational Origin
- The Selaginella Genome Identifies Genetic Changes Associated with the Evolution of Vascular Plants
- Chromatin "Prepattern" and Histone Modifiers in a Fate Choice for Liver and Pancreas
- Spatial Coupling of mTOR and Autophagy Augments Secretory Phenotypes
- Diet Drives Convergence in Gut Microbiome Functions Across Mammalian Phylogeny and Within Humans
- The Toll-Like Receptor 2 Pathway Establishes Colonization by a Commensal of the Human Microbiota
- A Packing Mechanism for Nucleosome Organization Reconstituted Across a Eukaryotic Genome
- Structures of the Bacterial Ribosome in Classical and Hybrid States of tRNA Binding

and did the same for the current issue of *Cell*:

- Mapping the NPHP-JBTS-MKS Protein Network Reveals Ciliopathy Disease Genes and Pathways
- Double-Strand Break Repair-Independent Role for BRCA2 in Blocking Stalled Replication Fork Degradation by MRE11
- Establishment and Maintenance of Alternative Chromatin States at a Multicopy Gene Locus
- An Epigenetic Signature for Monoallelic Olfactory Receptor Expression
- Distinct p53 Transcriptional Programs Dictate Acute DNA-Damage Responses and Tumor Suppression
- An ADIOL-ERβ-CtBP Transrepression Pathway Negatively Regulates Microglia-Mediated Inflammation
- A Hormone-Dependent Module Regulating Energy Balance
- Class IIa Histone Deacetylases Are Hormone-Activated Regulators of FOXO and Mammalian Glucose Homeostasis

and for the 2010 Nobel Prizes in science:

- Physics: *for groundbreaking experiments regarding the two-dimensional material graphene*
- Chemistry: *for palladium-catalyzed cross couplings in organic synthesis*
- Physiology or Medicine: *for the development of in vitro fertilization*

My conclusion is that 100% of these articles and awards are more about "accurately modeling the world" than they are about "providing insight," although they all have some theoretical insight component as well. I recognize that judging one way or the other is a difficult ill-defined task, and that you shouldn't accept my judgements, because I have an inherent bias. (I was considering running an experiment on Mechanical Turk to get an unbiased answer, but those familiar with Mechanical Turk told me these questions are probably too hard. So you the reader can do your own experiment and see if you agree.)

What doesn't Chomsky like about statistical models?

I said that statistical models are sometimes confused with probabilistic models; let's first consider the extent to which Chomsky's objections are actually about probabilistic models. In 1969 he famously wrote:

> But it must be recognized that the notion of "probability of a sentence" is an entirely useless one, under any known interpretation of this term.

His main argument being that, under any interpretation known to him, the probability of a novel sentence must be zero, and since novel sentences are in fact generated all the time, there is a contradiction. The resolution of this contradiction is of course that it is not necessary to assign a probability of zero to a novel sentence; in fact, with current probabilistic models it is well-known how to assign a non-zero probability to novel occurrences, so this criticism is invalid, but was very influential for decades. Previously, in *Syntactic Structures* (1957) Chomsky wrote:

> I think we are forced to conclude that ... probabilistic models give no particular insight into some of the basic problems of syntactic structure.

In the footnote to this conclusion he considers the possibility of a useful probabilistic/statistical model, saying "I would certainly not care to argue that ... is unthinkable, but I know of no suggestion to this effect that does not have obvious flaws." The main "obvious flaw" is this: Consider:

1. **I** never, ever, ever, ever, ... **fiddle** around in any way with electrical equipment.
2. **She** never, ever, ever, ever, ... **fiddles** around in any way with electrical equipment.
3. * **I** never, ever, ever, ever, ... **fiddles** around in any way with electrical equipment.
4. * **She** never, ever, ever, ever, ... **fiddle** around in any way with electrical equipment.

No matter how many repetitions of "ever" you insert, sentences 1 and 2 are grammatical and 3 and 4 are ungrammatical. A probabilistic Markov-chain model with *n* states can never make the necessary distinction (between 1 or 2 versus 3 or 4) when there are more than *n* copies of "ever." Therefore, a probabilistic Markov-chain model cannot handle all of English.

This criticism is correct, but it is a criticism of Markov-chain models—it has nothing to do with probabilistic models (or trained models) at all. Moreover, since 1957 we have seen many types of probabilistic language models beyond the Markov-chain word models. Examples 1-4 above can in fact be distinguished with a finite-state model that is not a chain, but other examples require more sophisticated models. The best studied is probabilistic context-free grammar (PCFG), which operates over trees, categories of words, and individual lexical items, and has none of the restrictions of finite-state models. We find that PCFGs are state-of-the-art for parsing performance and are easier to learn from data than categorical context-free grammars. Other types of probabilistic models cover semantic and discourse structures. Every probabilistic model is a superset of a deterministic model (because the deterministic model could be seen as a probabilistic model where the probabilities are restricted to be 0 or 1), so any valid criticism of probabilistic models would have to be because they are too expressive, not because they are not expressive enough.

In *Syntactic Structures*, Chomsky introduces a now-famous example that is another criticism of finite-state probabilistic models:

> Neither (a) 'colorless green ideas sleep furiously' nor (b) 'furiously sleep ideas green colorless', nor any of their parts, has ever occurred in the past linguistic experience of an English speaker. But (a) is grammatical, while (b) is not.

Chomsky appears to be correct that neither sentence appeared in the published literature before 1955. I'm not sure what he meant by "any of their parts," but certainly every two-word part had occurred, for example:

- "It is neutral green, **colorless green**, like the glaucous water lying in a cellar." The Paris we remember, Elisabeth Finley Thomas (1942).
- "To specify those **green ideas** is hardly necessary, but you may observe Mr. [D. H.] Lawrence in the role of the satiated aesthete." The New Republic: Volume 29 p. 184, William White (1922).
- **"Ideas sleep** in books." Current Opinion: Volume 52, (1912).

But regardless of what is meant by "part," a statistically-trained finite-state model *can* in fact distinguish between these two sentences. Pereira (2001) showed that such a model, augmented with word categories and trained by expectation maximization on newspaper text, computes that (a) is 200,000 times more probable than (b). To prove that this was not the result of Chomsky's sentence itself sneaking into newspaper text, I repeated the experiment, using a much cruder model with Laplacian smoothing and no categories, trained over the Google Book corpus from 1800 to 1954, and found that (a) is about 10,000 times more probable. If we had a probabilistic model over trees as well as word sequences, we could perhaps do an even better job of computing degree of grammaticality.

Furthermore, the statistical models are capable of delivering the judgment that both sentences are *extremely* improbable, when compared to, say, "Effective green products sell well." Chomsky's theory, being categorical, cannot make this distinction; all it can distinguish is grammatical/ungrammatical.

Another part of Chomsky's objection is "we cannot seriously propose that a child learns the values of 10^9 parameters in a childhood lasting only 10^8 seconds." (Note that modern models are much larger than the 10^9 parameters that were contemplated in the 1960s.) But of course nobody is proposing that these parameters are learned one-by-one; the right way to do learning is to set large swaths of near-zero parameters simultaneously with a smoothing or regularization procedure, and update the high-probability parameters continuously as observations comes in. And noone is suggesting that Markov models by themselves are a serious model of human language performance. But I (and others) suggest that probabilistic, trained models are a better model of human language performance than are categorical, untrained models. And yes, it seems clear that an adult speaker of English does know billions of language facts (for example, that one says "big game" rather than "large game" when talking about an important football game). These facts must somehow be encoded in the brain.

It seems clear that probabilistic models are better for judging the likelihood of a sentence, or its degree of sensibility. But even if you are not interested in these factors and are only interested in the grammaticality of sentences, it still seems that probabilistic models do a better job at describing the linguistic facts. The *mathematical* theory of formal languages defines a language as a set of sentences. That is, every sentence is either grammatical or ungrammatical; there is no need for probability in this framework. But natural languages are not like that. A *scientific* theory of natural languages must account for the many phrases and sentences which leave a native speaker uncertain about their grammaticality (see Chris Manning's article and its discussion of the phrase "as least as"), and there are

phrases which some speakers find perfectly grammatical, others perfectly ungrammatical, and still others will flip-flop from one occasion to the next. Finally, there are usages which are rare in a language, but cannot be dismissed if one is concerned with actual data. For example, the verb *quake* is listed as intransitive in dictionaries, meaning that (1) below is grammatical, and (2) is not, according to a categorical theory of grammar.

1. The earth quaked.
2. ? It quaked her bowels.

But (2) actually appears as a sentence of English. This poses a dilemma for the categorical theory. When (2) is observed we must either arbitrarily dismiss it as an error that is outside the bounds of our model (without any theoretical grounds for doing so), or we must change the theory to allow (2), which often results in the acceptance of a flood of sentences that we would prefer to remain ungrammatical. As Edward Sapir said in 1921, "All grammars leak." But in a probabilistic model there is no difficulty; we can say that *quake* has a high probability of being used intransitively, and a low probability of transitive use (and we can, if we care, further describe those uses through subcategorization).

Steve Abney points out that probabilistic models are better suited for modeling language change. He cites the example of a 15th century Englishman who goes to the pub every day and orders "Ale!" Under a categorical model, you could reasonably expect that one day he would be served eel, because the great vowel shift flipped a Boolean parameter in his mind a day before it flipped the parameter in the publican's. In a probabilistic framework, there will be multiple parameters, perhaps with continuous values, and it is easy to see how the shift can take place gradually over two centuries.

Thus it seems that grammaticality is not a categorical, deterministic judgment but rather an inherently probabilistic one. This becomes clear to anyone who spends time making *observations* of a corpus of actual sentences, but can remain unknown to those who think that the object of study is their own set of *intuitions* about grammaticality. Both observation and intuition have been used in the history of science, so neither is "novel," but it is observation, not intuition that is the dominant model for science.

Now let's consider what I think is Chomsky's main point of disagreement with statistical models: the tension between "accurate description" and "insight." This is an old distinction. Charles Darwin (biologist, 1809–1882) is best known for his insightful theories but he stressed the importance of accurate description, saying "False facts are highly injurious to the progress of science, for they often endure long; but false views, if supported by some evidence, do little harm, for every one takes a salutary pleasure in proving their falseness." More recently, Richard Feynman (physicist, 1918–1988) wrote "Physics can progress without the proofs, but we can't go on without the facts."
On the other side, Ernest Rutherford (physicist, 1871–1937) disdained mere description, saying "All science is either physics or stamp collecting." Chomsky stands with him: "You can also collect butterflies and make many observations. If you like butterflies, that's fine; but such work must not be confounded with research, which is concerned to discover explanatory principles."

Acknowledging both sides is Robert Millikan (physicist, 1868–1953) who said in his Nobel acceptance speech "Science walks forward on two feet, namely theory and experiment ... Sometimes it is one foot that is put forward first, sometimes the other, but continuous progress is only made by the use of both."

The two cultures

After all those distinguished scientists have weighed in, I think the most relevant contribution to the current discussion is the 2001 paper by Leo Breiman (statistician, 1928–2005), Statistical Modeling: The Two Cultures. In this paper Breiman, alluding to C.P. Snow, describes two cultures:

First the **data modeling culture** (to which, Breiman estimates, 98% of statisticians subscribe) holds that nature can be described as a black box that has a relatively simple underlying model which maps from input variables to output variables (with perhaps some random noise thrown in). It is the job of the statistician to wisely choose an underlying model that reflects the reality of nature, and then use statistical data to estimate the parameters of the model.

Second the **algorithmic modeling culture** (subscribed to by 2% of statisticians and many researchers in biology, artificial intelligence, and other fields that deal with complex phenomena), which holds that nature's black box cannot necessarily be described by a simple model. Complex algorithmic approaches (such as support vector machines or boosted decision trees or deep belief networks) are used to estimate the function that maps from input to output variables, but we have no expectation that the *form* of the function that emerges from this complex algorithm reflects the true underlying nature.

It seems that the algorithmic modeling culture is what Chomsky is objecting to most vigorously. It is not just that the models are statistical (or probabilistic), it is that they produce a form that, while accurately modeling reality, is not easily interpretable by humans, and makes no claim to correspond to the generative process used by nature. In other words, algorithmic modeling describes what *does* happen, but it doesn't answer the question of *why*.

Breiman's article explains his objections to the first culture, data modeling. Basically, the conclusions made by data modeling are about the model, not about nature. (Aside: I remember in 2000 hearing James Martin, the leader of the Viking missions to Mars, saying that his job as a spacecraft engineer was not to land on Mars, but to land on the model of Mars provided by the geologists.) The problem is, if the model does not emulate nature well, then the conclusions may be wrong. For example, linear regression is one of the most powerful tools in the statistician's toolbox. Therefore, many analyses start out with "Assume the data are generated by a linear model..." and lack sufficient analysis of what happens if the data are not in fact generated that way. In addition, for complex problems there are usually many alternative good models, each with very similar measures of goodness of fit. How is the data modeler to choose between them? Something has to give. Breiman is inviting us to give up on the idea that we can uniquely model the true underlying *form* of nature's function from inputs to outputs. Instead he asks us to be satisfied with a function that accounts for the observed data well, and generalizes to new, previously unseen data well, but may be expressed in a complex mathematical form that may bear no relation to the "true" function's form (if such a true function even exists). Chomsky takes the opposite approach: he prefers to keep a simple, elegant model, and give up on the idea that the model will represent the data well. Instead, he declares that what he calls *performance* data—what people actually do—is off limits to linguistics; what really matters is *competence*—what he imagines that they should do.

In January of 2011, television personality Bill O'Reilly weighed in on more than one culture war with his statement "*tide goes in, tide goes out. Never a miscommunication. You can't explain that*," which he proposed as an argument for the existence of God. O'Reilly was ridiculed by his detractors for not knowing that tides can be readily explained by a system of partial differential equations describing the

gravitational interaction of sun, earth, and moon (a fact that was first worked out by Laplace in 1776 and has been considerably refined since; when asked by Napoleon why the creator did not enter into his calculations, Laplace said "I had no need of that hypothesis."). (O'Reilly also seems not to know about Deimos and Phobos (two of my favorite moons in the entire solar system, along with Europa, Io, and Titan), nor that Mars and Venus orbit the sun, nor that the reason Venus has no moons is because it is so close to the sun that there is scant room for a stable lunar orbit.) But O'Reilly realizes that it doesn't matter what his detractors think of his astronomical ignorance, because his supporters think he has gotten exactly to the key issue: *why?* He doesn't care *how* the tides work, tell him *why* they work. *Why* is the moon at the right distance to provide a gentle tide, and exert a stabilizing effect on earth's axis of rotation, thus protecting life here? *Why* does gravity work the way it does? *Why* does anything at all exist rather than not exist? O'Reilly is correct that these questions can only be addressed by mythmaking, religion or philosophy, not by science.

Chomsky has a philosophy based on the idea that we should focus on the deep *whys* and that mere explanations of reality don't matter. In this, Chomsky is in complete agreement with O'Reilly. (I recognize that the previous sentence would have an extremely low probability in a probabilistic model trained on a newspaper or TV corpus.) Chomsky believes a theory of language should be simple and understandable, like a linear regression model where we know the underlying process is a straight line, and all we have to do is estimate the slope and intercept.

For example, consider the notion of a pro-drop language from Chomsky's Lectures on Government and Binding (1981). In English we say, for example, "I'm hungry," expressing the pronoun "I". But in Spanish, one expresses the same thought with "Tengo hambre" (literally "have hunger"), dropping the pronoun "Yo". Chomsky's theory is that there is a "pro-drop parameter" which is "true" in Spanish and "false" in English, and that once we discover the small set of parameters that describe all languages, and the values of those parameters for each language, we will have achieved true understanding.

The problem is that reality is messier than this theory. Here are some dropped pronouns in English:

- "Not gonna do it. Wouldn't be prudent." (Dana Carvey, impersonating George H. W. Bush)
- "Thinks he can outsmart us, does he?" (Evelyn Waugh, The Loved One)
- "Likes to fight, does he?" (S.M. Stirling, The Sunrise Lands)
- "Thinks he's all that." (Kate Brian, Lucky T)
- "Go for a walk?" (countless dog owners)
- "Gotcha!" "Found it!" "Looks good to me!" (common expressions)

Linguists can argue over the interpretation of these facts for hours on end, but the diversity of language seems to be much more complex than a single Boolean value for a pro-drop parameter. We shouldn't accept a theoretical framework that places a priority on making the model simple over making it accurately reflect reality.

From the beginning, Chomsky has focused on the *generative* side of language. From this side, it is reasonable to tell a non-probabilistic story: I *know* definitively the idea I want to express—I'm starting from a single semantic form—thus all I have to do is choose the words to say it; why can't that be a deterministic, categorical process? If Chomsky had focused on the other side, *interpretation*, as Claude Shannon did, he may have changed his tune. In interpretation (such as speech recognition) the listener receives a noisy, ambiguous signal and needs to decide which of many possible intended messages is most likely. Thus, it is obvious that this is inherently a probabilistic problem, as was recognized early

on by all researchers in speech recognition, and by scientists in other fields that do interpretation: the astronomer Laplace said in 1819 "Probability theory is nothing more than common sense reduced to calculation," and the physicist James Maxwell said in 1850 "The true logic for this world is the calculus of Probabilities, which takes account of the magnitude of the probability which is, or ought to be, in a reasonable man's mind."

Finally, one more reason why Chomsky dislikes statistical models is that they tend to make linguistics an empirical science (a science about how people actually use language) rather than a mathematical science (an investigation of the mathematical properties of models of formal language). Chomsky prefers the later, as evidenced by his statement in *Aspects of the Theory of Syntax* (1965):

> Linguistic theory is mentalistic, since it is concerned with discovering a mental reality underlying actual behavior. Observed use of language ... may provide evidence ... but surely cannot constitute the subject-matter of linguistics, if this is to be a serious discipline.

I can't imagine Laplace saying that observations of the planets cannot constitute the subject-matter of orbital mechanics, or Maxwell saying that observations of electrical charge cannot constitute the subject-matter of electromagnetism. It is true that physics considers idealizations that are abstractions from the messy real world. For example, a class of mechanics problems ignores friction. But that doesn't mean that friction is not considered part of the subject-matter of physics.

So how could Chomsky say that observations of language cannot be the subject-matter of linguistics? It seems to come from his viewpoint as a Platonist and a Rationalist and perhaps a bit of a Mystic. As in Plato's allegory of the cave, Chomsky thinks we should focus on the ideal, abstract forms that underlie language, not on the superficial manifestations of language that happen to be perceivable in the real world. That is why he is not interested in language performance. But Chomsky, like Plato, has to answer where these ideal forms come from. Chomsky (1991) shows that he is happy with a Mystical answer, although he shifts vocabulary from "soul" to "biological endowment."

> Plato's answer was that the knowledge is 'remembered' from an earlier existence. The answer calls for a mechanism: perhaps the immortal soul ... rephrasing Plato's answer in terms more congenial to us today, we will say that the basic properties of cognitive systems are innate to the mind, part of human biological endowment.

It was reasonable for Plato to think that the ideal of, say, a horse, was more important than any individual horse we can perceive in the world. In 400BC, species were thought to be eternal and unchanging. We now know that is not true; that the horses on another cave wall—in Lascaux—are now extinct, and that current horses continue to evolve slowly over time. Thus there is no such thing as a single ideal eternal "horse" form.

We also now know that language is like that as well: languages are complex, random, contingent biological processes that are subject to the whims of evolution and cultural change. What constitutes a language is not an eternal ideal form, represented by the settings of a small number of parameters, but rather is the contingent outcome of complex processes. Since they are contingent, it seems they can only be analyzed with probabilistic models. Since people have to continually understand the uncertain. ambiguous, noisy speech of others, it seems they must be using something like probabilistic reasoning. Chomsky for some reason wants to avoid this, and therefore he must declare the actual facts of language use out of bounds and declare that true linguistics only exists in the mathematical realm,

where he can impose the formalism he wants. Then, to get language from this abstract, eternal, mathematical realm into the heads of people, he must fabricate a mystical facility that is exactly tuned to the eternal realm. This may be very interesting from a mathematical point of view, but it misses the point about what language is, and how it works.

Thanks

Thanks to Ann Farmer, Fernando Pereira, Dan Jurafsky, Hal Varian, and others for comments and suggestions on this essay.

Annotated Bibliography

1. Abney, Steve (1996) Statistical Methods and Linguistics, in Klavans and Resnik (eds.) *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*, MIT Press.

   An excellent overall introduction to the statistical approach to language processing, and covers some ground that is not addressed often, such as language change and individual differences.

2. Breiman, Leo (2001) Statistical Modeling: The Two Cultures, *Statistical Science*, Vol. 16, No. 3, 199-231.

   Breiman does a great job of describing the two approaches, explaining the benefits of his approach, and defending his points in the vary interesting commentary with eminent statisticians: Cox, Efron, Hoadley, and Parzen.

3. Chomsky, Noam (1956) Three Models for the Description of Language, *IRE Transactions on Information theory* (2), pp. 113-124.

   Compares finite state, phrase structure, and transformational grammars. Introduces "colorless green ideas sleep furiously."

4. Chomsky, Noam (1967) Syntactic Structures, Mouton.

   A book-length exposition of Chomsky's theory that was the leading exposition of linguistics for a decade. Claims that probabilistic models give no insight into syntax.

5. Chomsky, Noam (1969) Some Empirical Assumptions in Modern Philosophy of Language, in *Philosophy, Science and Method: Essays in Honor or Ernest Nagel*, St. Martin's Press.

   Claims that the notion "probability of a sentence" is an entirely useless notion.

6. Chomsky, Noam (1981) Lectures on government and binding, de Gruyer.

   A revision of Chomsky's theory; this version introduces Universal Grammar. We cite it for the coverage of parameters such as pro-drop.

7. Chomsky, Noam (1991) Linguistics and adjacent fields: a personal view, in Kasher (ed.), *A Chomskyan Turn*, Oxford.

I found the Plato quotes in this article, published by the Communist Party of Great Britain, and apparently published by someone with no linguistics training whatsoever, but with a political agenda.

8. Gold, E. M. (1967) Language Identification in the Limit, *Information and Control*, Vol. 10, No. 5, pp. 447-474.

Gold proved a result in formal language theory that we can state (with some artistic license) as this: imagine a game between two players, guesser and chooser. Chooser says to guesser, "Here is an infinite number of languages. I'm going to choose one of them, and start reading sentences to you that come from that language. On your N-th birthday there will be a True-False quiz where I give you 100 sentences you haven't heard yet, and you have to say whether they come from the language or not." There are some limits on what the infinite set looks like and on how the chooser can pick sentences (he can be deliberately tricky, but he can't just repeat the same sentence over and over, for example). Gold's result is that if the infinite set of languages are all generated by context-free grammars then there is no strategy for guesser that guarantees she gets 100% correct every time, no matter what N you choose for the birthday. This result was taken by Chomsky and others to mean that it is impossible for children to learn human languages without having an innate "language organ." As Johnson (2004) and others show, this was an invalid conclusion; the task of getting 100% on the quiz (which Gold called language identification) really has nothing in common with the task of language acquisition performed by children, so Gold's Theorem has no relevance.

9. Horning, J. J. (1969) A study of grammatical inference, Ph.D. thesis, Stanford Univ.

Where Gold found a negative result—that context-free languages were not identifiable from examples, Horning found a positive result—that probabilistic context-free languages are identifiable (to within an arbitrarily small level of error). Nobody doubts that humans have unique innate capabilities for understanding language (although it is unknown to what extent these capabilities are specific to language and to what extent they are general cognitive abilities related to sequencing and forming abstractions). But Horning proved in 1969 that Gold cannot be used as a convincing argument for an innate language organ that specifies all of language except for the setting of a few parameters.

10. Johnson, Kent (2004) Gold's Theorem and cognitive science, *Philosophy of Science*, Vol. 71, pp. 571-592.

The best article I've seen on what Gold's Theorem actually says and what has been claimed about it (correctly and incorrectly). Concludes that Gold has something to say about formal languages, but nothing about child language acquisition.

11. Lappin, Shalom and Shieber, Stuart M. (2007) Machine learning theory and practice as a source of insight into universal grammar., *Journal of Linguistics*, Vol. 43, No. 2, pp. 393-427.

An excellent article discussing the poverty of the stimulus, the fact that all models have bias, the difference between supervised and unsupervised learning, and modern (PAC or VC) learning theory. It provides alternatives to the model of Universal Grammar consisting of a fixed set of binary parameters.

12. Manning, Christopher (2002) Probabilistic Syntax, in Bod, Hay, and Jannedy (eds.), *Probabilistic Linguistics*, MIT Press.

A compelling introduction to probabilistic syntax, and how it is a better model for linguistic facts than categorical syntax. Covers "the joys and perils of corpus linguistics."

13. Norvig, Peter (2007) How to Write a Spelling Corrector, unpublished web page.

Shows working code to implement a probabilistic, statistical spelling correction algorithm.

14. Norvig, Peter (2009) Natural Language Corpus Data, in Seagran and Hammerbacher (eds.), *Beautiful Data*, O'Reilly.

Expands on the essay above; shows how to implement three tasks: text segmentation, cryptographic decoding, and spelling correction (in a slightly more complete form than the previous essay).

15. Pereira, Fernando (2002) Formal grammar and information theory: together again?, in Nevin and Johnson (eds.), *The Legacy of Zellig Harris*, Benjamins.

When I set out to write the page you are reading now, I was concentrating on the events that took place in Cambridge, Mass., 4800 km from home. After doing some research I was surprised to learn that the authors of two of the three best articles on this subject sit within a total of 10 meters from my desk: Fernando Pereira and Chris Manning. (The third, Steve Abney, sits 3700 km away.) But perhaps I shouldn't have been surprised. I remember giving a talk at ACL on the corpus-based language models used at Google, and having Fernando, then a professor at U. Penn., comment "I feel like I'm a particle physicist and you've got the only super-collider." A few years later he moved to Google. Fernando is also famous for his quote "The older I get, the further down the Chomsky Hierarchy I go." His article here covers some of the same ground as mine, but he goes farther in explaining the range of probabilistic models available and how they are useful.

16. Plato (c. 380BC) The Republic.

Cited here for the allegory of the cave.

17. Shannon, C.E. (1948) A Mathematical Theory of Communication, *The Bell System Technical Journal*, Vol. 27, pp. 379-423.

An enormously influential article that started the field of information theory and introduced the term "bit" and the noisy channel model, demonstrated successive n-gram approximations of English, described Markov models of language, defined entropy with respect to these models, and enabled the growth of the telecommunications industry.