Probabilistic Growth and Mining of Combinations: A Unifying Meta-Algorithm for Practical General Intelligence

Ben Goertzel(✉)

OpenCog Foundation, Hong Kong, China
ben@goertzel.org

Abstract. A new conceptual framing of the notion of the general intelligence is outlined, in the form of a universal learning meta-algorithm called Probabilistic Growth and Mining of Combinations (PGMC). Incorporating ideas from logical inference systems, Solomonoff induction and probabilistic programming, PGMC is a probabilistic inference based framework which reflects processes broadly occurring in the natural world, is theoretically capable of arbitrarily powerful generally intelligent reasoning, and encompasses a variety of existing practical AI algorithms as special cases. Several ways of manifesting PGMC using the OpenCog AI framework are described. It is proposed that PGMC can be viewed as a core learning process serving as the central dynamic of real-world general intelligence; but that to achieve high levels of general intelligence using limited computational resources, it may be necessary for cognitive systems to incorporate multiple distinct structures and dynamics, each of which realizes this core PGMC process in a different way (optimized for some particular sort of sub-problem).

1 Introduction

An open question in AGI and cognitive science is: If one's goal is the creation of an AI system with general intelligence at the human level or beyond, should one be looking for a "single learning algorithm" to carry out the core learning processes at the heart of one's system? Or, on the other hand, do the practicalities of achieving real-world general intelligence within realistic computational resource restrictions, inevitably push one toward a more heterogeneous approach, with multiple different algorithms handling different kinds of learning?

The concepts presented here constitute a sort of middle ground between single-focused and heterogeneous approaches. Instead of a single core algorithm, what is proposed is a *core process*. However, this core process is a relatively abstract one, and it is proposed that this core process can be realized via a variety of different underlying algorithms – and that, in order to achieve effective general intelligence using realistically limited resources, it may indeed be necessary to utilize a variety of different algorithms, each instantiating the same core process in a slightly different way (due to the different algorithms being optimized for different types of subproblems).

B. Steunebrink et al. (Eds.): AGI 2016, LNAI 9782, pp. 344–353, 2016.
DOI: 10.1007/978-3-319-41649-6_35

The PGMC concept presented here can be viewed as a variation of the "Universal AI" approach, but it has the property that it naturally extends and generalizes a number of practical AI algorithms in current use. It also has potential to be used to create cognitive architectures that are heterogeneous underneath, but wrap this heterogeneity behind a "universal" looking PGMC layer.

In the latter vein, after giving a semi-formal presentation of the key ideas of PGMC, we will discuss here how the OpenCog architecture in particular could be reformulated so that most of the learning algorithms in OpenCog, in spite of their apparent diversity, were explicitly implemented and displayed to the user as aspects of the PGMC "meta-algorithm." This is interesting theoretically as an illustration of the PGMC concept, and may also be interesting practically as a direction for OpenCog development. Due to space limitations, the particulars of OpenCog formalization and implementation are largely omitted here, but can be found online on the OpenCog wiki site[1].

2 Probabilistic Growth and Mining of Combinations

A High Level Conceptual View. The conceptual foundation of the "Probabilistic Growth and Mining of Combinations" notion presented here is the idea of "forward and backward growth processes" introduced in [3] and reiterated in [5,6], and proposed there as a generic framework for modeling focused cognitive processes (meaning, cognitive processes that focus their attention on a relatively small body of information within a potentially larger scope associated with a cognitive system). As defined there, roughly,

- **forward growth** is the process via which a collection of cognitive entities, the subjects of focus, combine with each other to form new entities, which will often then still be subjects of focus. Sometimes these focus entities may also combine with entities outside the focus.
- **backward growth** is the process of figuring out how a target cognitive entity might be produced via forward growth

These simple operations are actually quite generic in character, and can be seen to underlie a large variety of cognitive processes including logical inference, evolutionary learning, clustering, language processing, and a host of others.

The notion of PGMC presented here refines these earlier ideas considerably. Combining entities (as forward growth does) is all very well, but there are always too many combinations; so how does one decide which combinations to actually make? One can say that the entities doing the combining can decide what to combine with, which makes sense, but is not a very meaty conclusion. The idea of PGMC is that forward and backward growth processes can quite generically be executed, in a way that balances resource limitations with practical effectiveness, via a process of: (1) mining patterns in the results of prior forward and backward

[1] See http://wiki.opencog.org/wikihome/index.php/OpenCoggy_Probabilistic_Programming.

growth processes; (2) using these patterns to probabilistically guide ongoing forward and backward growth. And the pattern mining mentioned in the first item may also be done via leveraging various forward and backward growth processes, thus creating a virtuous-cycle recursion.

This rather generic process may be envisioned as a type of probabilistic programming, and I propose here that it may be considered as a general foundation for generally intelligent cognition. That any cognitive process can in principle be cast in this form is not a terribly interesting observation, since there are numerous foundations for universal computation known already. That so many cognitive processes known to be useful for narrow AI and proto-AGI systems can be conveniently cast in this form, along with so many creative processes occurring in nature, is more interesting and compelling.

A Partial Formalization. To elaborate the PGMC idea a bit further, let us consider the case of a cognitive system concerned with pursuing some complex goal in some complex environment. While goal-oriented activity is not necessarily the crux of all intelligent behavior[2], it is a significant ingredient, and also a convenient concrete focus for analysis. So let us explore the ways in which a cognitive system can maximize the degree of fulfillment of its goal functions.

Without being highly specific about the cognitive architecture of this system, let us assume that it stores knowledge in a manner founded on set of entities called Atoms, that two Atoms can often be combined with each other to form new Atoms, and that it maintains a memory of Atoms which it leverages to take both internal and external actions. Those who wish a more precise formalization may refer to the simple cognitive system model presented in [4]; the discussion here makes sense in the context of that formal agents model. The addition to that formal agents model we need here is simply that the internal operation of the formal agent contains a self-generating system as defined in [2], consisting of entities called Atoms that can combine with each other to form new Atoms.

To figure out how to achieve its goals, the cognitive system in question may look into its memory, which may contain historical information regarding which Atoms, when executed, have led to satisfaction of which goal functions to which degrees. Now, this historical information will not usually be directly relevant, because each new situation the system confronts will be a little different. But it may be indirectly relevant. Which brings us, I hypothesize, very close to the core of general intelligence.

Suppose one has a base probability distribution $\mathcal{P}$ over Atoms. Then, given a goal function G and a cognitive system S, one can define S_G as the set of Atoms in S's memory that take the form

$$[x * y] \rightarrow G < c >$$

Say that x and y are "constituents" of the Atom $[x * y] \rightarrow G < c >$.

[2] See [10] for a deep discussion of how general intelligence transcends goal-pursuit.

One can then define the following Basic Learning Process:

1. Initialize the "working memory" pool W (a set of Atoms). Possibilities include
   - Initialize $W = S_G$, for goal-directed combination formation
   - Initialize W to the system's long-term memory, for more general and exploratory combination formation
2. Repeatedly (N times, not forever) create stochastically chosen forward and backward combinations, via the following steps
   - For a forward step, choose a combination $a * b$, in which at least one of a or b is a constituent of a member of W, with probability determined by $\mathcal{P}$
   - For a backward step, choose a combination $a * b$ whose result is a member of W, with probability proportional to $\mathcal{P}$
   - Place the result of this combination in W, and place the expression describing the combination's result in the system's long-term memory
   - Tabulate the degree to which G is achieved at the current time, and insert into the long-term memory an Atom of the form $C \rightarrow G < c >$ corresponding to the current combination C

Basically, this process is doing forward and backward chaining, starting from existing knowledge about the goal G (for goal-oriented learning) or from the system's knowledge as a whole (for more general exploratory learning), and guided in its choices by the distribution $\mathcal{P}$.

This Basic Learning Process summarizes a basic and simplistic form of "probabilistic growth of combinations", and also stores some data along the way, which will be useful for mining. The initial growth of combinations is governed by a prior distribution, so unless this distribution is highly appropriate for the goals in question, the process will take a very long time. The step needed to get from this to PGMC is to mine the data stored, so as to figure out how to grow more appropriate combinations with a higher probability.

To take this next step, suppose one uses the above process to calculate the probability of G being fulfilled to a given degree, within a certain time period following execution of an executable Atom E. This may be used to define another probability distribution $\mathcal{P}_G(E)$ over the space of Atoms E. This distribution may be used to guide the above learning process.

We may then define our universal learning meta-algorithm, PGMC (Probabilistic Growth and Mining of Combinations) as the following loop:

1. begin with a goal G, and an initial distribution $\mathcal{P} = \mathcal{P}_0$
2. apply the Basic Learning Process using $\mathcal{P}$, leading to an estimate of $\mathcal{P}_G$
3. set $\mathcal{P} = \mathcal{P}_G$ and return to the previous step

In essence, this is a form of "reinforcement learning" driven by generic forward and backward chaining processes. One reason this is interesting is that the generic processes of backward and forward chaining can be used to conveniently formulate a wide variety of different learning processes; thus this framework can

unify a variety of different, useful learning algorithms within a common meta-algorithm.

This learning process will be very slow (i.e. require many repetitions or very large N) if $\mathcal{P}$ is not chosen intelligently. Here we come to a very deep and interesting aspect of learning. If the Atom language chosen is rich enough, then $\mathcal{P}$ may itself be represented at an Atom $x_{\mathcal{P}}$, i.e. a nested Atom expression. This may be achieved e.g. if there are basic Atoms that make random choices. One then has a framework capable of supporting *meta-learning*, meaning the process of learning $x_{\mathcal{P}}$ so that $\mathcal{P}_G$ causes the system to choose actions with high goal-achievement value.

In probabilistic programming vernacular, Step 3 in the Basic Learning Process indicated above can be interpreted as a kind of *fitness-proportionate sampling* over the Atom space, conditioned on the distribution $\mathcal{P}$. This is related to, though different from, the use of *optimization queries* in probabilistic programming, as outlined in [1] Fitness-proportionate sampling may also be performed based on weighted combinations of system goals, rather than a single goal. Why are fitness-proportionate samples appropriate here? Because a full distribution over Atom space, conditioned on $\mathcal{P}_G$, need not be computed, since what the system really cares about is finding something good to do according to its goals, not accurately estimating the degree of badness of each potential bad action it could take. So sampling from the space in a way that takes more samples from the "good" parts is the right thing to do.

If $\mathcal{P}_G$ has the property of favoring simpler Atoms (e.g. Atoms corresponding to smaller Atom expressions), then what we have here is conceptually a variant of Solomonoff induction [8,9]. It seems likely to us that in this case, PGMC could be proved to be an instance of Solomonoff induction based on a certain special computational model. And it seems likely that, in most cases, if the initial $\mathcal{P}$ embodies Occam's Razor, then ensuing $\mathcal{P}_G$ will also do so. But PGMC is not just another reformulation of Occam's Razor; the key point is that this computational model, based on forward and backward chaining, is especially cognitively natural, comprising a mathematical abstraction of a host of well-studied and demonstrably-useful cognitive processes.

The potential flexibility and power of this sort of framework is considerable. For instance, one possibility is to stock S_G with Atoms embodying *logical inference rules*. In this case, application of the logical inference rules will be part of the forward and backward chaining process. Supposing S_G also contains Atoms that are specifically tied to particular situations in which G has been achieved in the past. Then combining the Atoms embodying logical inference rules, with Atoms regarding specific past situations involving G, will lead to *inferential extrapolation* from past situations to present and future situations. The quality of this extrapolation will depend on the quality of the inference rules – and on the quality of the choices made regarding which inference rules to apply, in what order, in each situation. These choices regarding *inference control* may be made by yet more Atoms embodying "inference control rules" – or, more interestingly, they may be made by the probabilistic sampling process outlined above. In this

case the probabilistic sampling process is mining historical patterns regarding what inference steps have previously been helpful in what situations. And the inference rules themselves may be useful in performing extrapolations pertinent to these historical patterns and their present applications.

In sum, what we have described here is a "universal learning meta-algorithm" as follows:

1. Combine Atoms in memory via forward and backward chaining
2. Choose which combinations to form, via optimization queries conditioned on a distribution formed by extrapolating from which combinations have achieved relevant goals in the past
3. The "extrapolation" in Step 2 is carried out via forward and backward chaining, guided by optimization queries conditioned on a goal-driven probability distribution
4. The goal-driven probability distribution in Step 3, is formed or improved via Steps 2 and 3 ("meta-learning")

Or, to rephrase once more:

1. Combine stuff in memory, via forward combination (put stuff together and see what comes out) and backward combination (given a target, find stuff that can be combined to yield that target)
2. Since there are too many possible combinations to try them all, choose which ones to form based on estimating which ones have the highest probability of leading to goal achievement. This estimation is based on memory of which combinations have been tried in the past and how much success they've had.
3. Since past combinations and situations were a bit different from the current reality, figuring out how to use them to estimate probabilities regarding current situation requires some "reasoning." This reasoning can be done by the same old backward and forward combination process that we're now in the middle of describing. In other words, the "reasoning rules" or "reasoning processes" involved should be embodied in atomic entities that get combined with other entities in the course of the forward and backward combination processes.
4. To perform all this probability estimation we need some heuristics to guide which combinations have a higher "a priori" probability. This prior distribution had better favor simple combinations, for instance. And this prior distribution can be adapted over time, i.e. "meta-learning."

As emphasized above, this is intended not to summarize the full cognitive activity of a mind, but rather the focused, goal-directed portion of a mind's activity. "Background cognitive activity" is also assumed to occur. This background activity may often be modelable via ongoing forward and backward chaining driven by non-goal-oriented stochastic processes, but we will not enlarge on this point here.

3 Explicitly Implementing PGMC in OpenCog

The PGMC formalism and concept may be used as a general mode of describing and thinking about cognitive processes. It may also, however, be used as a means of explicitly structuring cognitive processes, i.e. of guiding and structuring practical AGI implementation. This is a direction we are now exploring within the OpenCog project. For a review of the OpenCog concepts utilized here, please see [5,6][3]. But we emphasize that the applicability of the PGMC idea is not restricted to OpenCog; the latter is used here as an illustrative example.

The Simplest OpenCog Implementation of PGMC. First we explore a relatively simple, not necessarily practical method of instantiating PGMC within OpenCog[4]

Within OpenCog, knowledge is represented in terms of:

- Atoms for specific sensory data. As illustrative examples: RGB values associated with particular pixels coming from a particular camera at a particular point in time; and characters coming from a particular terminal at a particular point in time.
- Atoms carrying out internal or external actions (SchemaNodes)
- Atoms for combining Atoms to form new Atoms.[5]
- As well as ListLinks, SetLinks for grouping together Atoms into sets, which may then be acted on by SchemaNodes – this is a mechanism enabling SchemaNodes to take whole "nested Atom expressions" as inputs

The standard assemblage of OpenCog Atoms is highly expressive and arguably sufficient to form the representational framework for a general intelligence interacting in the everyday human world.

OpenCog has a Unified Rule Engine (URE) which carries out generic forward and backward chaining processes like the ones abstractly described above[6]. The URE contains a FocusSet which can be used like the "working memory" W in the above-described process. In the URE, the role of the operator $*$ is played by the BindLink construct, which when it sees $a*b$ seeks to bind the VariableNodes contained in a to the Atom b. This is a somewhat specialized formalism, yet fits within the general mathematical framework described above, and has both general computational power and flexible pragmatic usability. For example, this approach is used to apply probabilistic logic rules (formulated as Atom expressions themselves, via the PLN, Probabilistic Logic Networks, framework) to Atoms representing concrete data or data patterns.

[3] Or see http://wiki.opencog.org/w/CogPrime_Overview for an informal online overview.

[4] See http://wiki.opencog.org/wikihome/index.php/OpenCoggy_Probabilistic_Programming for more details.

[5] E.g. the easiest way to do this in terms of OpenCog's current assemblage of Atom types, is simply to consider polymorphic, higher-order-functional SchemaNodes – i.e. SchemaNodes whose inputs may be SchemaNodes and whose outputs may be SchemaNodes.

[6] https://github.com/opencog/atomspace/tree/master/opencog/rule-engine.

A first pass at a combination-choice distribution $\mathcal{P}$ would be a distribution over patterns (representable as Atoms) obtained by applying OpenCog's *Pattern Miner* to the Atomspace. The Pattern Miner finds combinations of Atoms that occur frequently, or surprisingly often (in the sense of information theory), across the Atomspace. A product of frequency or surprisingness, with a measure of combination size, gives an apparently quite reasonable $\mathcal{P}$ (though further investigation in this direction will certainly be valuable).

Using the Pattern Miner to infer $\mathcal{P}$ is essentially a way of doing frequency-based reinforcement learning for goal-driven selection of combinations to use in forward or backward chaining. One is choosing combinations that match template patterns that have frequently been useful in the past. Of course, this is nowhere near maximally intelligent, in itself, initially. But it's a basis to start from. Pattern mining can be used to choose specific actions, but it can also be used to choose more abstract rules for generating specific actions – which can lead indirectly to higher levels of intelligence. Using this counting-based approach, one can choose, for example, *inference rules* that seem to work frequently. Using these inference rules in future, based on their previously calculated utility, one will then be deploying more intelligent reasoning than is directly implied by pattern mining.

Using the URE and the Pattern Miner, and an expressive set of Atoms including VariableNodes and associated quantifiers (i.e. at least including basic predicate and/or term logic constructs), one could construct a fairly minimal OpenCog based cognitive architecture, which learns *everything* from experience based on reinforcement, aided by spontaneous background self-organization. In an approach like this, meta-learning would be relied upon to ramp up intelligence from an initially low level, where "counting" (probability estimation; information-theoretic pattern mining) is used as the core learning algorithm, relied upon to discover Atom-combinations that comprise more intelligent learning algorithms. The processes of chaining and history-based chain-selection are the core ones, and are carried out with increasing intelligence as the system's experience allows it to improve the tools (the chains of combinations) it uses to build the probability distributions used in chain-selection. The Atomspace would be getting utilized purely as a single, large, self-reprogramming functional program.

Furthermore, it is interesting and important to observe that OpenCog's PLN probabilistic inference engine can be used to improve the various steps of this "simplistic OpenCog PGMC" i.e. pattern mining and credit assignment. Using PLN to augment greedy pattern mining with heuristic probabilistic inference based pattern mining, and to augment simplistic credit assignment with heuristic probabilistic causal inference, makes the OpenCog-based PGMC process much less simplistic. But adaptive pattern mining based inference control, as described in this section, is necessary to get PLN to perform scalably in these roles. So we have a certain circularity, but it's a virtuous cycle rather than a vicious one – the smarter PLN gets, the better it can do pattern mining and credit assignment; and the better pattern mining and credit assignment are done, the smarter PLN will be.

Framing OpenCog Cognitive Processes in Terms of PGMC. While the previously published formulations of OpenCog's cognitive algorithms do not use the PGMC formalism or terminology, in fact every significant cognitive process in OpenCog can be cast fairly straightforwardly in PGMC terms. Here we will illustrate this point by briefly running through a few examples[7],

- MOSES [7], OpenCog's evolutionary learning algorithm, could be reimplemented in such a way that:
  - Candidate "programs" being evolved by MOSES are wrapped in GroundedSchemaNodes, and the fitness function is also wrapped in a GSN
  - Fitness evaluation of candidate GSNs in an evolving population, is recorded in the Atomspace as the degree to which the GSN implies the GSN goal (fitness function)
  - A few "program generation" rules are implemented, each of which creates new GSNs representing new programs to be added to the "evolving population." E.g. a crossover rule could be implemented, as could a "local search" rule.
- Clustering could be implemented via an agglomerative algorithm, so that when the "clustering rule" Atom acts on a pair of Atoms, it decides whether to fuse them into a proto-cluster or not[8]
- Concept blending could be implemented in a similar way to clustering, leveraging the existing cog-blend command[9]

4 Conclusion and Next Steps

We have outlined PGMC, a new framing of the process of general intelligence. Conceptually, PGMC presents general intelligence as a synthesis of probabilistic pattern mining with generic "growth processes" as one finds at the core numerous existing AI algorithms, and throughout the physical and natural world. PGMC can also be viewed as a formulation of the general concept of "universal AGI learning using brute force"; but it differs considerably from prior formulations in that it connects closely conceptually with a variety of practically useful learning algorithms, and with dynamical processes occurring in nature.

Preliminarily, PGMC appears useful as a way of formulating diverse concrete learning algorithms within a common framework. Some examples of this have been given within the OpenCog framework (which would however require various small tweaks to the current OpenCog framework to function as desired). "The OpenCog example is generally instructive regarding the pragmatics of applying PGMC to real-world AGI systems. Different components or aspects of a complex AGI systems may end up manifesting PGMC in their own different ways.

[7] Discussed in more depth at http://wiki.opencog.org/wikihome/index.php/OpenCoggy_Probabilistic_Programming).

[8] See http://wiki.opencog.org/wikihome/index.php/Agglomerative_Clustering_in_Atomspace_using_the_URE on the OpenCog wiki site for specifics.

[9] See https://github.com/opencog/opencog/tree/master/opencog/python/blending.

But PGMC may be a valuable way to model components with a view toward simplifying and comprehending inter-component interactions.

Natural next steps in this research direction would be (unordered): Create a full mathematical formalization of the PGMC framework, along the lines outlined here; mathematically explore the relationship between PGMC, probabilistic programming, type theory and other areas of AI mathematics; add SampleLink and any other needed tools to OpenCog, to make PGMC elegantly and concisely implementable in OpenCog; implement a crude, non-scalable version of PGMC-based reinforcement learning in OpenCog, and test it on very simple problems; implement PGMC-based PLN inference control and credit assignment and test as appropriate; formalize the mapping of further OpenCog cognitive algorithms (e.g. MOSES, clustering) into the PGMC framework; explore the mapping of other, non-OpenCog AI approaches into the PGMC framework.

References

1. Potapov, A., Batishcheva, V., Rodionov, S.: Optimization framework with minimum description length principle for probabilistic programming. In: Bieger, J., Goertzel, B., Potapov, A. (eds.) AGI 2015. LNCS, vol. 9205, pp. 331–340. Springer, Heidelberg (2015)
2. Goertzel, B.: Chaotic Logic. Plenum, New York (1994)
3. Goertzel, B.: A system-theoretic analysis of focused cognition, and its implications for the emergence of self and attention. Dynamical Psychology (2006)
4. Goertzel, B.: Toward a formal definition of real-world general intelligence. In: Proceedings of AGI 2010 (2010)
5. Goertzel, B., Pennachin, C., Geisweiller, N.: Engineering General Intelligence, Part 1: A Path to Advanced AGI via Embodied Learning and Cognitive Synergy. Atlantis Thinking Machines. Springer, Heidelberg (2013)
6. Goertzel, B., Pennachin, C., Geisweiller, N.: Engineering General Intelligence, Part 2: The CogPrime Architecture for Integrative, Embodied AGI. Atlantis Thinking Machines. Springer, Heidelberg (2013)
7. Looks, M.: Competent Program Evolution. Ph.D. Thesis, Computer Science Department, Washington University (2006)
8. Solomonoff, R.: A formal theory of inductive inference part I. Inf. Control **7**(1), 1–22 (1964)
9. Solomonoff, R.: A formal theory of inductive inference part II. Inf. Control **7**(2), 224–254 (1964)
10. Weinbaum, D.W., Veitas, V.: Open-ended intelligence (2015). http://arXiv.org/abs/1505.06366