## **Topic 2**: Something out of nothing?

**Original questions:**

1. How to correspondingly discover new knowledge and algorithms in the inner loop?
2. Internal thought process (computation), deliberation, feedback and System 2
3. How does it relate to open-endedness, curiosity, generative processes, etc.?
4. Inference vs Learning of new processes
5. The isolated AI scientist / mathematician analogy
6. Why are we able to do it?
7. What is not sufficient on MetaGenRL (while omitting multi-agentness)?
8. Is planning a necessary component that enables creating "something out of nothing"? E.g. MCTS?
9. (Maybe similar to 2?) What are the problems being solved by this (e.g. by math, language driven thinking)? Is it the construction of new theories and the exploration of their consequences? Is it digitalization for the purpose of enabling long-distance / long-term communication with ourselves?

**continued...**

## Pre-discussion Comments & Resources

**“What something out of nothing” could mean?**

The intuitive leap in discovery. The act of internally combining knowledge into heretofore unknown structures and using these structures (either on a model of the world or experimenting on the real world) to generate new observations and knowledge. One purely mental use of this would be like saying “Imagine if gravity was at 1/10th. What are the implications?”. A more grounded usage would be “Here is some phenomenon. You have a database of facts. What combination of facts explains the phenomenon?” Iteratively using combinations of facts to elicit more observations from the environment (even if only confirming or refuting parts of the hypothesis) and storing that knowledge.

**Relation to intuition and System 2**

If the single-pass inference is system 1 (i.e ‘gut’ reasoning), then system 2 can be an iterative process where we examine the gut response and refine it. It is gut to know the applicability of some mathematical field to a problem, but it takes system 2 to actually work out what parts of the field are needed and apply them to the problem (or observations in this case)

---

*“It doesn't make a difference how beautiful your guess is. It doesn't make a difference how smart you are, who made the guess, or what his name is. If it disagrees with experiment, it's wrong.”*

Richard P. Feynman

This is a very philosophical question. Descartes tried it from “cogito ergo sum” (I think, so I exist), but as you can not falsify nor confirm it with any experiment, he could not build anything out of it, and no one have since then. I agree with Feynman: it is not possible to get something out of nothing, you need to test it in at least one experiment. After one experiment, you falsify or partially confirm your hypothesis, and then you can update your priors. This is what you get.

We should reframe the question as “**Something out of an observation**”:

- A scientist wants to learn how to prepare a new chemical product. It plans a chain of chemical products to be added to the mix, producing reactions, and ending in the desired product. He has some prior knowledge about what he will get after each reaction, so he uses it when planning. After planning a combination, he has a **hypothesis** but needs access to a lab to try it out and **check against reality**. If the result doesn’t match, the hypothesis is rejected, the **prior is updated** accordingly. Now he can try to build a new -and possibly better- hypothesis.
- In program synthesis we concatenate functions (random NNs) to produce a program that serves as a good classifier. Our **hypothesis** is that it will deliver. We need to check it **against the reality** of labeled examples to test the accuracy of the classifier. After we get our result,

good or bad, we can train the NN in the chain with gradient descent to **update our priors**. Now we can try to build a new -and possibly better- program.

**Proposal #1** is about **measuring how useful** each NN is (which product we added were more beneficial in getting the product) and roughly accounts for measuring the "heat dissipated" by each reaction -or how different the mix gets after it- as a measure of how significant this element was for the final product, so you can take it away from the final formula.

**Proposal #2** is about how to approach the case of **many experts in different areas**, solve a problem, how they communicate and learn from experiments in a coordinated way so they end up **solving the problem** (finding a good joint hypothesis that works in all areas of the problem) and **learning expertise in the process** by updating their prior knowledge.

---

**Something out of nothing**:

For me this is related to the creation of thinking tools and the generation of new hypotheses. In a way these are among the things I understand least about humans. (Mostly because I can't even express them in a Bayesian framework no matter how much I ignore the intractability).

- generating new hypotheses: even under infinite computational resources Bayesian inference tells us nothing about how to generate new hypotheses. It only tells us how to incorporate data into beliefs about the set of hypotheses that we start with. Humans have come up with hypotheses that seem not to have been in our prior (e.g. Quantum mechanics). These hypotheses are formed using abstract concepts / language but not necessarily in order to communicate with others but more as tools for thinking for oneself (inspired by the idea that the interesting thing about language isn't to be found in its use for communication with others but as a tool for a single individual).

  One theory here is that intrinsic abstractions are coarse-grainings of a dynamic world model and that an agent can learn to intervene on the coarse-grainings and evaluate the consequences of abstract assumptions in this way. In some cases we may get universal computation on the coarse-grained scale. This would require figuring out how an agent can interact with its own world model on coarse-grained scales, and how this affects the entire world model. In humans it seems that our abstract thoughts can also be used to entrain parts of our intuition to behave in certain ways. In the real world macro-scales have no effect in addition to micro-scales but in an internal model this may be different.

  In short, can World models be abused for thinking about more abstract things than they were trained to model the dynamics of? And can this abstract thinking in turn improve the agent performance?

- Computation. As mentioned below we can actually solve problems by thinking more about them. This may or may not involve language / abstraction but it seems different from e.g. motor skills that we have learned from a lot of repetition. For ANNs there also seem to be limits to how much they can profit from more time. This might be related to the above but not as wildly speculative. It doesn't require any assumptions about coarse-graining of world models being related to abstractions.

- Digitalization metaphor: maybe it isn't related but digitalization made communication over long distances possible because it allowed for amplification of the signal without amplification of the error. If we amplify an analogue signal in a cable then whatever noise got into the signal will also be amplified. This means that even if we add arbitrarily many amplifiers the signal will have basically the same noise at the end (depending mostly on distance). If we digitalize the signal however we can place amplifiers so close together that the probability for noise to switch the (still analogue) signal from the values assigned to 0 to the values assigned to 1 is arbitrarily small. This allows basically infinite length error free communication. Language is a kind of digitalization and may similarly allow long and time extended thought processes that are impossible otherwise. Is this maybe related to effects in neural networks?

**Something out of nothing:**

For me, this relates to exploiting the information contained in your priors (Your own mental model of the world, a policy/value network in an RL context, a bunch of randomly initialized NN in a program synthesis task, a GAN, the equations for a physics model, etc) by an iterative process (sampling, planning) that allows you to better understand how well it describes what they are modeling and finding non-trivial implications of your own model. For example:

- In the case of a physics model -> Make predictions of new particles
- For a GAN -> Sample faces of persons that do not exist
- Program synthesis -> Combine the NN modules to solve different tasks
- In RL -> Using an actor-critic + planning to improve performance

All of them require sampling "low probability areas" in the state space of the learned priors, or possible combinations of your previously known concepts. This is why I believe that planning is a necessary component that enables creating "something out of nothing".

What is the kind of conclusion we want to reach within these 2 days? Is the next step forward on this idea. For example:

- *Finding a common understanding of what "something out of nothing" means. What is finding "new knowledge", "new hypothesis" or "unknown structures"? Is it realistic to aim for "making intuitive leaps" or should we focus on "understanding better the knowledge of our current assumptions about the world"? -> Relates to: "can World models be abused for thinking about more abstract things than they were trained to model the dynamics of. And can this abstract thinking in turn improve the agent performance?"*
- *Is it possible to do it without planning/sampling techniques? Why or why not this is a process of finding "low probability regions" in our prior knowledge?*
- Identifying what scaling laws we expect something that satisfies 'something out of nothing' to have, framework for how to measure them? *How can we benefit from adding additional computational resources?*
- A set of tasks on which those scaling laws are/aren't meaningful?
- An implementation/architecture that we think should exhibit the desired scaling laws? *How does this relate to badger -> inner/outer loop*
- Do we want to focus on identifying a connection between these ideas and some theoretical formalism, like causality or ideas about counter-factual mutual information? In which case

we should spend the time mostly carefully walking through the math (maybe better in a different format?)

Aside from directly quantitative targets, are there instead qualitative targets we want?

- 'Position paper' type of thing, which is mostly text defining a viewpoint or set of questions:
- Identify questions about cognitive artefacts produced by a 'something out of nothing' agent - are they exchangeable, are they composable, etc?
- Identify connected concepts (is there a link between something-out-of-nothing and intrinsic/self-motivated behavior, for example?)
- Identify auxiliary benefits that aren't directly task performance metrics - e.g. 'explainability' or other sorts of visualizations of an internal belief that might be easier with a something-out-of-nothing agent. E.g. in a planner you can look at plans it discarded.

Meta question: What's the purpose of doing this all together in a workshop format rather than independent exploration?

Proposed answer: Here we can establish a common understanding about the interpretation and purpose of investigations. E.g. we don't solve a task because we were told to solve it, we should solve a task because we can interpret what the solution is telling us. So we should focus time on things where we really want to be on the same page with regards to what a certain outcome would mean or how to understand the results of experiments.

1) confusing (mutual) information with (different types of) knowledge: how to clarify an *actionable notion of knowledge*?

2) about the "entanglement" through interaction with the world: what and how much interaction is necessary and sufficient to produce knowledge?

3) dreaming as a process for re-filtering information that an agent "already knows" but "cannot use yet"

4) this knowledge transfer from a domain to another feels like it should be linked to the amount of energy necessary to solve a task, taking advantage of allowing travel between domains

## Discussion Notes

Formal point: can't generate information via computation, so you have to incorporate entropy from a random variable. However, that entropy isn't 'about' anything (no MI with the world or anything else). So, if I run the algorithm for a while and then I want to store its state, we can ask: how compressible is the model state, under the constraint of preserving a level of performance? If maintaining performance requires storing more bits over time (despite the information of the world being constant) then we have 'something out of nothing'.

This leads to a hypothesis: all forms of something out of nothing must necessarily involve sampling. It seems formally true, but things about 'usable information' could violate this hypothesis.

The randomness doesn't have to be independent of all else, could be from a context switch to solving another problem or unrelated (to the problem) environmental signals like from pacing around, fidgeting, etc. I hypothesize that the source may not matter, but the structure could.

Main points discussed:

1. exploration of the internal model using noise in order to improve the model
2. usable information can be used to classify necessary computational effort / time for tasks. Way to find computationally difficult tasks?

The upper bound of what we can get out of nothing is what humans do while dreaming. While dreaming, a classifier can retrieve old observations that are augmented, deformed, mixed or noised, and then processed to better classify without adding information. Using internal representations -world model- you can also self play "what would have happened if I had taken this action instead of the one I took?" and alter your priors over the actions. But you are limited to refine your actual use (priors) from previous knowledge.

Is self-organization something-out-of-nothing? In some parts of the system entropy decreases substantially, entropy of the other parts grows. We can easily create a self-organized multi-agent system. Can we use it in learning? We can aim at a guided (or soft control) self-organized system that solves some problem (inner loop) and learn how to effectively guide the system (outer loop).

Q: Is there something as "usable entropy"? Can we specify it?

Maybe via degree to which we can auto-encode a random variable

## Testable Hypotheses

- **In meta-learning, the inner loop does inference and the outer loop learns a prior:** To look at the question of inference vs learning, maybe we can think of this as some kind of invariance to changes of the prior (similar to recent work on causality)? So if the inner loop performance depends on changes to the relative probabilities of tasks, it points towards inference and the outer loop learning the prior. Whereas if the inner loop performance only depends on changes in the support of the distribution of tasks (e.g. fundamentally new kinds of tasks added/removed) then it points more towards learning. So we could try to quantify this.

## Experiments

- The concept of 'usable information' seems important to in-place transformations, can we do something like ask a neural network to discover a sequence of representations of the same information which require incrementally more computational power to decrypt, but all contain the same thing and are all possible to decrypt in the end? The idea here is to angle towards some of the information without any new external signals being received.

- In the previous meeting, there was an example of training a conv-net on simulations from Conway's Game of Life or other synthetic data sources. Maybe we can make an explicit experiment out of this? E.g. we have a training process with two-time dimensions - one is epochs on a synthetic infinite-data source, the other is epochs on a finite real-world dataset. This is

related to a recent paper on 1M augmentations of a single image being enough to train the first two layers of an ImageNet classifier to the same performance as training on ImageNet itself. **Question for Badger relevance:** is there an 'inner loop' version of this that would actually work? E.g. if we apply our meta-learning approaches in the same way, allowing them to adapt to a data generator, would any existing approach actually benefit? This might point at the problem with our conception of inner loop learning.

## Architectures

In terms of what it means to 'discover a new algorithm', we should think about the process by which someone (e.g. a human programmer) receives and chooses to utilize an algorithm. Proposed architecture using Fractal AI/MCTS framework is:

- Let's have a bunch of experts/agents which each have a repertoire of modules or functions inside them.
- Let's use MCTS/Fractal to decide when to use these modules
- Based on how the modules are used, let's let experts/agents exchange modules with each-other
- Hypothesis: performance should scale (maybe logarithmically) with the number of agents in the 'society' all trying to solve problems together, as which modules are useful/not useful end up being filtered out by the knowledge exchange process.
- How to set up the experiment: Need a task distribution (RL tasks, supervised tasks, etc) where an individual task can be solved to some level relatively quickly by Fractal (a few minutes at most). Scale the cognitive depth (# of walkers) and see performance improve. Scale the cognitive width (# of simultaneous agents exchanging modules) and see if performance improves. Scale the social age (how long agents have been exchanging modules) and see if learning to solve new tasks becomes faster. Try to characterize the scaling laws of these three parameters, re-architect for faster scaling.

Considering how we think, we can see ourselves as having trains of thought that run concurrently (one train runs at a time) as we 'context switch' between them like in a modern single-core processor. If an expert was given the above ability to create and test a hypothesis against a mental model, then a group of experts could each do this (trains of parallel thought) and come to some consensus about which hypothesis is most probable. This could speed up thinking by an order of N, as well as strongly demonstrate the benefit of multi-agentness.

In a way, the above scenario is a microcosm of this workshop, we experts with different experiences and expertise are attacking a single problem, but have different ideas. We are coming together now to exchange them and find the most probable solutions.

- **Diagrams to support the discussion**
- Where can something out of nothing happen?
- Are programs an example of "noise"? How can we measure "usable information"?
- How can we measure the value of a learned function? Can this be done independently from a program?
- **Scaling properties**: where can we benefit from adding more computational resources?
    - General:
        - Run the system for longer
        - Add experts

        - Add tasks
        - Search for longer programs
    - Functions:
        - Add functions
        - Make functions bigger -> More layers/neurons
        - Train for more epochs
    - Planning:
        - Depth
        - Width
        - Train planner for longer
- **Dreaming**: planning using learned priors in Alphazero-like fashion with no contact with the environment?
- **Multiple trains of thought**: related to task distribution? One agent should focus on one task or all agents use the same batch of tasks?
- Longer programs related to “more computational effort to decrypt”?
- Is passing learned functions between experts a way of symbolic communication? How should we share the functions? (related to self-organization)
- Where “learning to plan” should happen: shared planner learner or one learner per expert?

## References

[1] Pascanu et al., 2017, Agents that imagine and plan
*Summary: Sokoban via planning using a GAN-like setup*

[2] Guez et al., 2019, An Investigation of Model-Free Planning
*Summary: Argument about ‘behaviorist’ perspective on planning - what signatures should an architecture have in its performance to say that it might be planning?*

[3] Xu et al., 2020, A Theory of Usable Information Under Computational Constraints
*Summary: Usable information - information theory for computational difficulty*

[4] Pierrot et al., 2019, Learning Compositional Neural Programs with Recursive Tree Search and Planning
*Summary: a novel reinforcement learning algorithm, AlphaNPI, that incorporates the strengths of Neural Programmer-Interpreters (NPI) and AlphaZero.*