Spring 2020

Internal Badger Workshop

Topic 1: When is modularity, collectivity & multi-agentness beneficial?

Original questions:

1. What are the benefits of distributedness generally and in the Badger architecture?
2. Why does communication/information transfer work well in neural networks, but not necessarily in Badger?
3. Where is the transition from monolithic systems into modular/multi-agent ones?
4. How can we achieve the benefits of collective decision-making akin to the ones observed in the NASA experiment (Watson and Hall 1970)?
5. How can hidden information games help?
6. Can the benefits of distributedness be studied separately from the meta-learning part ("Something out of nothing" topic) or not? If not, why?
7. Similar to 1, but narrower: what is the benefit of dividing the computation into blocks, and how can we calculate the trade-off with respect to constricting communication between threads?
8. Is there a market like structure where agents profit even more when cooperation is successful (so that they try to convince others via communication) but can still profit from their own predictions if others can't be convinced?
9. What's possible with heterogeneous goals/rewards?

continued...

GoodAI

Pre-discussion Comments & Resources

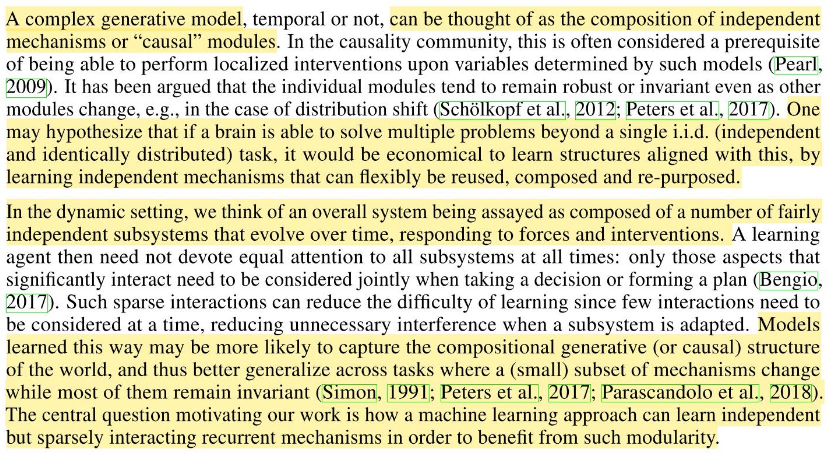
We should:

- we should specify what the problem is that we are solving
- then propose some explicit testing of the property
- propose also system biases

Structure of the following is loosely based on Maar's levels of analysis

WHY: Computational level: what does the system do and, why does it do these things?

1. **Ability to solve tasks NNs are not capable of**
   - o dynamic scaling of IO
2. **Shared weights:**
   - o Enables a good (small) outer/inner-loop parameters ratio.
   - o Shared weights enable for having small num. of outer-loop parameters (weights) vs. big num. of inner-loop parameters (many experts & their activations)
   - o Compared to this, RL^2 has a big ratio: many outer-loop parameters (weights) and too few inner-loop parameters (activations). This leads to poor generalization.
   - o A good example of a system with a small ratio is MetaGenRL, which has a small num. of outer-loop parameters and a big num. of inner-loop parameters. Therefore it has very good generalization ability.
3. **Ability to use sparse activation of modules, which:**
   a. Helps with avoiding catastrophic forgetting ~ **continual learning** [ANML]
   b. Naturally causes **modules to specialize**, which implies compositional/disentangled representations [beta-VAE, consciousness prior]. This enables **good generalization outside the training distribution** [RIM, meta-transfer]



A complex generative model, temporal or not, can be thought of as the composition of independent mechanisms or "causal" modules. In the causality community, this is often considered a prerequisite of being able to perform localized interventions upon variables determined by such models (Pearl, 2009). It has been argued that the individual modules tend to remain robust or invariant even as other modules change, e.g., in the case of distribution shift (Schölkopf et al., 2012; Peters et al., 2017). One may hypothesize that if a brain is able to solve multiple problems beyond a single i.i.d. (independent and identically distributed) task, it would be economical to learn structures aligned with this, by learning independent mechanisms that can flexibly be reused, composed and re-purposed.

In the dynamic setting, we think of an overall system being assayed as composed of a number of fairly independent subsystems that evolve over time, responding to forces and interventions. A learning agent then need not devote equal attention to all subsystems at all times: only those aspects that significantly interact need to be considered jointly when taking a decision or forming a plan (Bengio, 2017). Such sparse interactions can reduce the difficulty of learning since few interactions need to be considered at a time, reducing unnecessary interference when a subsystem is adapted. Models learned this way may be more likely to capture the compositional generative (or causal) structure of the world, and thus better generalize across tasks where a (small) subset of mechanisms change while most of them remain invariant (Simon, 1991; Peters et al., 2017; Parascandolo et al., 2018). The central question motivating our work is how a machine learning approach can learn independent but sparsely interacting recurrent mechanisms in order to benefit from such modularity.

   c. Might help with **reducing the noise in the comm.** channel [e.g. CommNet vs IC3Net]
4. **Other modules can be used as a source of auxiliary signal**
   - o this can be taken into account for better learning (lateral context)
   - o causal influence between agents [Social Influence]
   - o can social influence help with training? [Social Influence]
5. **Better robustness; in ensemble-like meaning:**
   - o the whole system does not get trapped in local optimum

- always there are agents that are doing something else than others

6. **Drawback: single-agent problems vs multi-agent (modular) Badger**
    a. MARL papers are for MARL problems, not for normal single-agent ones
    b. Need to decompose a single problem amongst experts, then compose the solution back
    c. An example related paper: Separation of Concerns in RL:
        i. multiple agents used to solve a single-agent task by (hierarchical) decomposition of agents with different objectives.
        ii. Specifically moves toward specialized agents.
7. **Drawback: modular systems are less efficient than distributed ones (see above)**
    a. local vs. holographic representations
8. **Drawback: more complicated to learn modular systems**
    - Need to find how to decompose the task and compose the solution
    - Should it be done in a distributed manner?
    - I.e.: shared weights, local optimum
    - Help of POET curriculum, some kind of [ESN] component and/or "write-everything" memory?

WHAT: Algorithmic level: how does the system do what it does?

1. **Ability to solve tasks NNs are not capable of**
2. **Shared weights**
    a. Badger should employ shared weights between agents so that the ratio outer/inner loop parameters are small
    b. There might be a problem with local optimas (see below), might need for some diversification of expert behavior (specialization in time)
3. **Ability to use sparse activation of modules**
    a. There should be a mechanism which will make only some subset of experts active during inference (e.g. gating in ANML)
    b. During learning, only the active experts are updated
        i. should help specialization and avoid catastrophic forgetting
    c. The resulting system should be able to generalize outside the training tasks well
    d. Questions:
        i. how the information should be routed to the active experts?
            1. Separation of concerns, dynamic routing between capsules, predictive coding mechanisms?
        ii. how the information should be routed from active experts to the output?

4. **Other modules can be used as a source of auxiliary signal**
5. **Better robustness; in ensemble-like meaning**
6. **Drawback: single-agent problems vs multi-agent (modular) Badger**
7. **Drawback: modular systems are less efficient than distributed ones**
8. **Drawback: more complicated to learn modular systems**
    a. The ideal solution would be to leverage multiagent-ness to make the learning easier (compared to a monolithic system)

b. **Separation of Concerns in RL**: multiple agents used to solve single-agent tasks by decomposition of agents with different objectives. Specifically moves toward specialized sub-agents and generalized hierarchical decomposition (options).
    i. The learning objective does not always result in the best performance objective; work on *intrinsic motivation* aims to address this

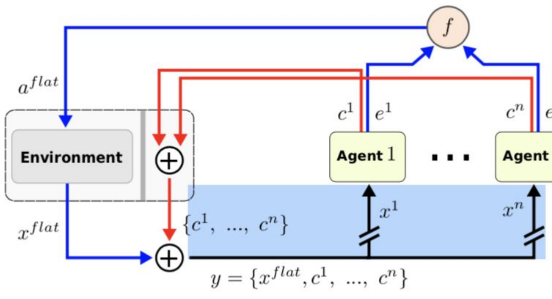


Figure 2. SoC model.

    ii. But this may lead to the Attractor Phenomenon in Decomposed RL: local objectives may lead to bad attractors, states where the optimal local policy is bad overall system behavior. This paper gives conditions on when local policies may or may not lead to bad attractors—although unfortunately hard to design around these conditions.
    iii. Similar but simpler work with explicit soft state partitioning via auxiliary exploration bonus

b. The field of *mechanism design* studies how to set up a game of self-interested agents to ensure good system behavior (depending on application)
    - In short, create incentives (design reward function) so that attempts to cheat or deceive is suboptimal
    - Groves and VCG mechanisms are popular, efficient, and guarantee truthful communication is optimal in social choice games
    - There is surprisingly little work on this? Something like mechanism design (incentive shaping) should be used for decomposed RL

HOW: Implementational/physical level: how is the system physically realized

1. **Ability to solve tasks NNs are not capable of**
2. **Shared weights**
    a. Test: the smaller the ratio of outer/inner loop parameters, the better the generalization ability?
    b. Test: smaller num of parameters has slower learning at the beginning (saddle point) but then faster convergence (more data like in conv nets?)
3. **Ability to use sparse activation of modules**
    a. Test: sparse activation of experts leads to natural specialization in the inner loop?
        i. use some decomposable task, measure independence of the experts and their MI with some part of the task (generative factors)
        ii. example: bouncing balls in the RIM paper
        iii. compare with modular system without sparsity constraint
    b. Test: sparse activation of experts helps with lifelong learning?
        i. see the ANML tasks
        ii. one expert per task (no conflicting gradients, more **stable learning**)

        c. Test: generalization outside the training distribution due to composition of parts
            i. beta-VAE experiments? Something better?
4. **Other modules can be used as a source of auxiliary signal**
5. **Better robustness; in ensemble-like meaning**
    - Test: agents randomly switching to random policies
    - Test: communication dropout
    - Test: adding/removing experts runtime
    - Test: adversarial agents (train time only)
6. **Drawback: single-agent problems vs multi-agent (modular) Badger**
    a. Test: decomposition: the problem is meaningfully distributed amongst (a subset) agents
    b. Test: composition: solutions from multiple agents are correctly aggregated to the output
7. **Drawback: modular systems are less efficient than distributed ones**
    a. Test: probably not worth here
8. **Drawback: more complicated to learn a modular system (?)**
    a. Test: is probably not needed here

Collectivity & Multi-Agentness - The dream vs. reality

- There really seem to be at least **two views** on the topic of multi-agentness/collectivity in badger, depending on whether we are talking about the **potential** of badger vs. the **technical** details of how learning in badger occurs, i.e. the WHY and WHAT vs the HOW
    - WHY & WHAT - seems to be primarily viewed via **analogies** to either biological systems, such as collections of neurons in the brain and their collective computation, or social systems where groups of agents act collectively as we humans do, for example
    - HOW - this view is more technological as it's more about the **substrate** in which badger lives in our simulations, i.e. a collection of RNNs connected up in some sense (statically or dynamically) and the impact of this substrate on the ability to learn within it at all, let alone some relevant structures, such as learning algorithms
- I think the above two views are potentially so different that they might warrant very **different discussions**, e.g. mathematical optimization view vs social science/economy view
- Both views are very important, but before we are able to resolve the technical issue, the "dream" view might stay as such
- Contrary to the above view, and a reason why the MARL view could be important - **rethinking the technological view** might drastically change the substrate and hence eliminate current training issues, but this hasn't happened yet, especially with the fact the many MARL-specific properties do not apply in badger due to the much more forgiving setup in badger that's much more akin to a NN setup

The Dream - the potential of multi-agentness

- There is a huge collection of relevant resources and literature from various fields.
    - Economics
    - Social Science
    - Biology
    - Cognitive Science

    - Psychology
- Ideas from all of the above can be relevant and interesting, but are not trivial to implement and within a system like Badger might not always be directly applicable

The Reality - the substrate and its impact on learnability and achieving the Dream

- As discussed at the last workshop, multi-agentness might be a **necessary evil**
- But can we find where it can help at the substrate level?
    - Distributed Optimization
    - Ensembles
    - Niches
    - Herding
    - Swarm Intelligence
    - Distributed/Decentralized Artificial Intelligence (see section 1.1.3 in [15])
    - Distributed Problem Solving
    - Federated/Collaborative Learning
- Many multi-agent problems and settings try to approximate scenarios in which the physical /mental barriers can be removed by modelling/approximating the type of structure and information that is missing due to the distributed nature of the problem, the architecture or the world (Stone and Veloso, 2000)
- The power of masses vs the power of individuality (ref)
    - i.e. when do collective system benefit from masses (e.g. Ensembling) rather than useful fusion of disparate information by individuals with collectively beneficial information/knowledge (aka NASA experiment)
- We need to remember the difference between a multi-agent system and its properties and **learning** in multi-agent systems. Similarly collective computation and collective learning are different things and most likely should be treated as such. Some ideas on MAS learning can be found in [16].
- The notion of a Multiplicative/Divisive and Interaction Mechanism in MAS learning

Notes on other questions

1. What are the benefits of distributedness generally and in the Badger architecture?
    - see above, mainly RIM
2. Why does communication/information transfer work well in neural networks, but not necessarily in Badger?
    - because we are not training it correctly (to invent the communication)
    - good communication properties emerge from training on multiple different tasks [the emergence of grounded compositional language] (or in case of tasks specifically designed to do so [the emergence of discrete compositional language])
3. Where is the transition from monolithic systems into modular/multi-agent ones?
    - see the *disentangled* and *separation of concerns* above?

- It has been argued that multi-agent systems with perfect communication are analogous to a single agent system with multiple effectors (Stone and Veloso, 2000)

- “an **information accumulation phase**, in which (in this study) pairs of individuals gather information about their fighting abilities and make decisions about their dominance relationships, and an **information aggregation phase**, in which these decisions are combined to produce a collective computation.” (Brush, Krakauer and Flack, (2018))
    - *Is this something that does/should and can be in Badger?*

Discussion Notes

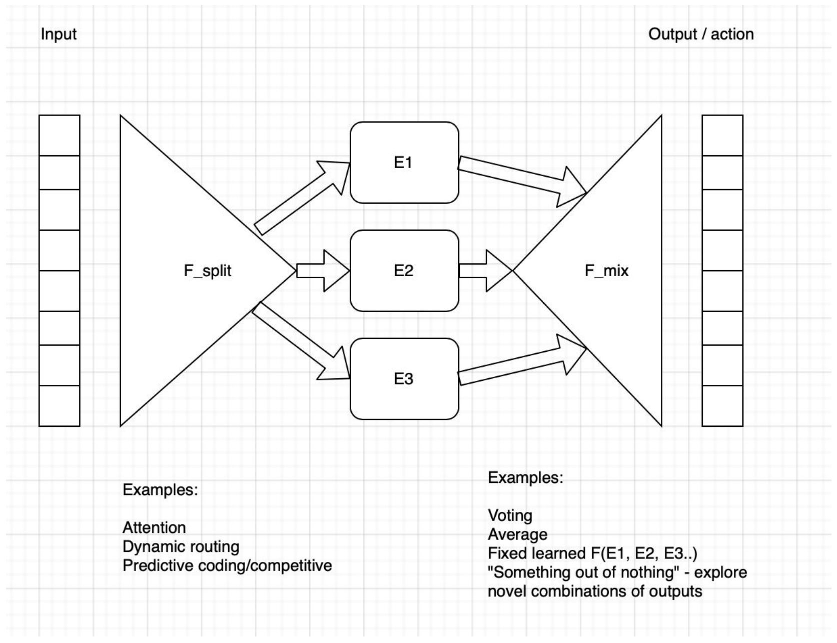
- It would be nice (instead of saying “this is nice, this is nice”) to pose the question as follows:
    - is there a problem which cannot be solved without modularity??
        - example: generalization ability of disentangled representations
- Modularity in this way: similar to program synthesis
- There is this subsymbolic/symbolic debate going on (what is better etc..)
- Important to find out how to decompose the task
    - The disentangled representations is a rabbit hole (but an interesting one)
    - What about non-decomposable tasks
    - Potential for “something out of nothing” topic, combining knowledge in modules in novel ways -> something new

- Another possible advantage of modularity: possibly faster problem solving (with partitioned domain, faster exploration), e.g. Q-Learning Acceleration via State-space Partitioning, Separation of Concerns in Reinforcement Learning
    - What cannot be solved simply without modularity:
        - Agents are trying to find gold in a maze, the maze is not completely connected (some agents cannot reach the gold), but the gold can be “smelled” through the walls when close enough. The need to communicate if they smell the gold but cannot reach it.
        - Large boundary value problem (solving partial differential equations). Large domain means a huge system of linear equations = impossible to solve on a computer with one agent. We divide the domain into many smaller domains and solve many smaller boundary problems - each agent solves one (one cpu). They need to communicate (iterate) to solve the whole problem.

- How do we decide where & how to introduce a bottleneck to aid/enforce generalization?
- What natural bottlenecks does multi-agentness/collectivity pose?
- Conflicting gradients
    - Does the multi-agent nature of Badger make this issue more prominent?
    - Parameter sharing might be making it more difficult to avoid/have such gradients

Challenges:

Modular system - high-level description:


Input
Output / action
E1
E2
E3
F_split
F_mix
Examples:
Attention
Dynamic routing
Predictive coding/competitive
Examples:
Voting
Average
Fixed learned F(E1, E2, E3..)
"Something out of nothing" - explore novel combinations of outputs

- how the splitting function should be designed (non-trivial)?
- how the aggregation function should be designed?
- communication (between expert) constraints, e.g. for covering partial dependency?

Testable Hypotheses

- Having sparsely activated experts [RIM paper]:
  - leads to specialization of experts
  - prevents one expert dominating and modeling complex behavior
  - small nets/experts provide better generalization
  - good properties of modular/disentangled representation (recombine learned experts)
- Some kind of competition for the input is helpful for splitting the input amongst experts, which should help with:
  - specialization of experts
  - convergence (problems with conflicting gradients inhibited)
  - competitive mechanism could be tested on the guessing game task?
- Smaller outer/inner loop parameters help generalization
  - can be modular not modular solution
  - in case of modular system: the “division of work” amongst modules might need to be done properly first
- Can we pose Badger in the sense of single layer RNNs as done in (Turek et al., (2019)) with stacked RNNs?

   - By doing this, can the formulation reveal why badger is difficult to train compared to standard RNNs?

Experiments

- Implement Badger through single layer RNN as in (Turek et al., (2019)) to investigate if learning dynamics are better than in current implementation.

See above for more

References

[1] Beaulieu et al., 2020, Learning to Continually Learn
Summary: Avoiding catastrophic forgetting due to sparse activations

[2] Vickrey–Clarke–Groves Mechanism
Summary: Good design of the system, agents do not cheat

[3] Stone and Veloso, (2000) Multiagent Systems: A Survey from a Machine Learning Perspective
Summary: A classical and seminal overview of multi-agent systems

[4] David Sussilo, Echo State Networks
Summary: Great tweet storm on ESNs and their behaviour. Relevant as some pointed out similarity of ESNs and badger

[5] Turek et al., (2019), A single-layer RNN can approximate stacked and bidirectional RNNs, and topologies in between
Summary: Possible way of mapping badger to RNNs via specially crafting the weight matrix and delaying signals within

[6] Giving Up Control: Neurons as Reinforcement Learning Agents
Summary: A preliminary framework where individual neurons are RL agents

[7] Chalk, Tkacik and Marre, (2020), Training and inferring neural network function with multi-agent reinforcement learning
Summary: Neuron as an agent from the point of view of neuroscience

[8] Lu and Yan, (2020), Algorithms in Multi-Agent Systems: A Holistic Perspective from Reinforcement Learning and Game Theory
Summary: A unifying view of Multi-agent systems through the lens of deep RL and game theory

[9] Brush, Krakauer and Flack, (2018), Conflicts of interest improve collective computation of adaptive social structures
Summary: Looking at how collective computation is performed in nature and how conflicts of interest can help decision-making

[10] Khadka et al., (2019), Collaborative Evolutionary Reinforcement Learning

[11] Krafft, (2018), A Simple Computational Theory of General Collective Intelligence
Summary: "General collective intelli-gence arises from groups achieving commitment to group goals, accurate shared beliefs, and coordinated actions"

[12] Decker, (1987), Distributed Problem-Solving Techniques: A Survey
Summary: "Distributed artificial intelligence (DAI) is concerned with solving problems by applying both artificial intelligence techniques and multiple problem solvers. It differs from the more general area of distributed processing because it is concerned with distributing control as well as data and can involve extensive cooperation between processing entities."

[13] Jiang et al., (2019) Improving Federated Learning Personalization via Model Agnostic Meta Learning
Summary: a link between MAML and federated learning which deals with training a model in a distributed fashion where input data is provided only locally at each agent

[14] Arcas (2020), Social Intelligence
Summary: A NeurIPS talk about the creeping in of Social Intelligence into Machine Learning

[15] Bond and Gasser, (1988), Readings in Distributed Artificial Intelligence
Summary: A book about distributed AI with chapters on topics relevant to the benefits of distributed AI

[16] Weiss and Dillenbourgh, (2007), What is 'multi' in multi-agent learning?
Summary: Delves into what multi-agentness is and its benefits compared to a single-agent system

Relevant Researchers

Peter Stone (*Multi-Agent Systems*)

Angeliki Lazaridou (*Emergent Communication & Social Influence*)

Jessica Flack (*Collective Computation*)

David Krakauer (*Collective Computation*)

Drew Fudenberg (*Game Theory*)

James Crutchfield (*Statistical Complexity, Computational Mechanics*)

Melanie Mitchell (*Complexity Science*)

Pierre Lévy (*Collective Intelligence*)

Iain Couzin (*Collective Behaviour*)

Keith Decker (*Distributed Problem Solving*)

Blaise Agüera y Arcas (*Federated Learning*)

Jakub Konecky, Petr Richtarik (*Federated Learning*)

Dave Bacon (*Quantum Computation & Federated Learning*)

Gerhard Weiss (*Multi-Agent Systems*)