Topic 3: Challenges of the Inner Loop

Original questions:

1. How to force skill learning in the inner loop, rather than the outer loop?
2. How to scale from a fixed size inner loop, to an open-ended never ending inner loop? What kind of inner/outer loss can test for this?
3. Are hidden states/activations of a recurrent NN enough to feasibly allow for all the above?
4. Is it really important where the skills were learned? Humans are not made from general experts; we are pre-booted with experts specialized to learn language, social interaction, vision, etc.
5. Can SGD still work over millions of steps inside the inner loop? Probably no. Can this be solved via auxiliary losses?
6. Can the above be achieved using auto-generated tasks, e.g. by POET?
7. What is the importance of algorithmic choice on both loops?
8. What's the relation of this topic to the "Something out of nothing"? Do the inner loop problems limit the complexity of the experimentation of the agent/experts?

continued...

Pre-discussion Comments & Resources

Suggested session structure

We can divide the discussion into two sections. The first is more big-picture-ish, the second zooms in on more practical concerns.

1. **WHY and WHAT (Marr's computational level)**
    a. **Motivations for the outer/inner loop**
        i. How much should be learned within the inner loop, how much within the outer loop?
        ii. Motivation: we want to do learning, not inference in the inner loop (beware of the boundary between the two). This is related to the next point.
    b. **Are 2 levels/loops enough?**
        i. 3 loops: SGD outermost loop, middle loop (general inner loop), innermost loop (solves specific task).
            1. Motivation: the 2-loop setup learns skills specific to a particular task - the inner loop develops a strategy for solving instances of the particular task. If there is one more loop, then the inner loop becomes a "middle loop", or "designer of algorithms". We want this for the sake of adaptivity.
            2. Analogy:
                a. Outermost loop = programming language spec,
                b. Middle loop = program or libraries; adding new libraries could be seen as learning,
                c. Innermost loop = logical circuit elements / "atomic" algorithms.
            3. Analogical to differential learning - learning random things, expanding the palette of skills (e.g. how kids learn to play basketball by first learning other skills - running, jumping, dribbling etc.)
            4. Related also to curriculum learning
        ii. Can the middle loop change itself?
            1. If instead of learning an inner loop there is some fixed baseline algorithm that is able to collect pieces of knowledge and later act on them, we have a "middle loop" that changes itself. This could be a way out of 3 loops back to 2.
    c. **Is the inner loop the right design choice? What are the alternatives that better support the idea of lifelong learning?**
        i. Problem: computing long inner loops is not feasible computationally.
        ii. there are computational bounds, e.g. limited brain capacity
        iii. how general do we want our learner to be? Related - F. Chollet's "A Measure of Intelligence" - it lists priors relevant for human-level AI.
        iv. an open-ended search may be a way forward - MCTS, A* etc. Imagine an ever-expanding set of heuristics available to A* search - then there is just one, the infinite inner loop (the A* planner) and it expands its capabilities thanks to an increasing repertoire of heuristics/skills. Related also to AlphaZero.

v. we could support very long inner loops (millions of steps) if we then have to backprop only a bunch of steps on the winning path (program synthesis approach); program synthesis approaches already do this = an inspiration. You only want to backprop through the most fruitful direction of search. E.g. if I have 1000 modules, but only 10 are chosen into the winning policy, I backprop through the 10 modules only. Alternatively, backprop through policy evaluations that lead to the chosen solution. Currently works for 1 task, not for a sequence of tasks

vi. but do not limit our computation to ordinary PC 2 hours.

vii. but we may end up just throwing a lot of compute on stuff that would otherwise not be interesting/useful

viii. it's good if we can train long inner loops using only short inner loops - e.g. training a general for cycle using iterations up to length 4. Other example - MetaGenRL
   1. we may need some guarantees on convergence if we want to increase our inner loop lengths beyond what was seen in training (SGD has those guarantees, but do our learned inner loops have them?)
   2. fear if we limit ourselves to SGD, we may get only small improvements in learning.

d. **What is the inner loop in humans?**

i. The benefit of multi-agentness (relevant to Topic 1):
   1. possibly faster evolution: robustness to failure, things are easy to replicate (compare bloodstream as composed of millions of identical red blood cells vs. a monolithic system of macroscopic dimensions)
   2. beware - multicellular species lose the benefits of horizontal gene transfer; they have other mechanisms for fast adaptation, though
   3. analogy with heart rate: more complex systems need to dissipate more energy to sustain themselves

ii. The analogy with innate and acquired knowledge (innate language acquisition machinery, but not Swedish) - acquired knowledge is gathered by executing the human inner loop.
   1. Why aren't we born speaking Swedish? Is it computational bounds? Time demands of evolution? Plasticity requirements due to task distribution shifts in nature? -> experiments/hypotheses

iii. The learning dynamics change over the lifetime (!)
   1. Related to switching optimizers/loss functions during runtime

iv. In nature: Similar to meta-learning, but with three loops, different from point b) above:
   1. Outermost = Evolution (architecture bias),
   2. Middle loop = life-long learning (inductive bias),
   3. Innermost = fast problem-solving. There is evidence of slow/fast learning in nature.

v. A further example from nature, relating the outer loop to System 2 and inner loop to System 1: two loops e.g. when typewriting - outer loop reads and comprehends sentence and sends word to inner loop which commands fingers to write it. The two loops are mostly independent, the inner loop is automatic and implicit. The outer loop is controlled and explicit.

e. **Question 4: Is it really important where the skills were learned?**
    i. Humans are not made from general experts; we are pre-booted with experts specialized to learn a language, social interaction, vision, etc.
    ii. If we bottleneck where things are learned, it may affect the generalization ability of the agent. Learning certain types of skills in the outer loop should be bottlenecked as much as possible so that they become as general as possible.
        1. Bottleneck example: provide such a wide range of tasks that the outer loop may only learn very general strategies
        2. A suite of meta-learning tasks? Omniglot, tiny imagenet, MuJoCo tasks -> experiments we could try; we could use very small architectures
            a. guessing game is just a sanity check
            b. next step - try it on tasks above, use a small arch.
            c. investigate to what extent we can force learning things in the inner loop / outer loop

f. **Question 8: What's the relation of this topic to the "Something out of nothing" topic?**
    i. Do the inner loop problems limit the complexity of the experimentation of the agent/experts?
    ii. Analogy with a hypothetical creature that is prepared for every possible situation right from birth - that creature does not need deliberation and can perform decisions in one step. But consider the number of situations the creature must be prepared for!
    iii. Without learning in the inner loop, you wouldn't be able to learn to read, write, build a space shuttle etc.
    iv. Humans have learned to **externalize knowledge** in external knowledge bases - those are accustomed to our inner loop learning mechanisms.
        1. Have static storage of past brain states accessible by attention? There may be other agents' brain states there etc. -> **experiment**?
        2. passing on of knowledge between generations is a helpful tool (-> relevant literature - "Jun Tani", experiment on timescales

g. **What is the right loss function for the outer loop?**
    i. Be better at solving things rather than directly solve the thing. We can train the network to become better when we add experts.
    ii. Count the number of tasks that you have high performance on (the metric from Enhanced POET - ANNECS)
    iii. Scalability - in the number of experts and the number of inner loop steps
    iv. The above implies a collection of losses rather than a single loss

2. HOW (Marr's algorithmic and implementation levels)

a. **Summary of what we've achieved so far and what limitations we have found**
    i. Solutions to the guessing game task, computational requirements of those, scaling limitations, inner loop length limitations, algorithm adaptivity limitation (scaling in inputs, different tasks, more experts)

b. **Question 1: How to force skill learning in the inner loop, rather than the outer loop?**
    i. Do not allow encoding skills within experts/architecture by the outer loop. Make it so that the experts/architecture are good at acquiring new skills.
        1. Eg. as MetaGenRL.
        2. 3 loops can also help.

            3. Bottlenecking (see above)
                a. homogeneity of the communication policy
                b. amount of computation
                c. outer loop parameters never having access to the sensory data from within inner loop (MetaGenRL inspiration)
                d. restriction on the level of hidden states (e.g. experiments with regularization of hidden states)
                e. invariant risk minimization - minimize the amount of information carried about specific tasks
            4. Additionally, reframing the problem to a single infinite-length inner loop can also help.

c. **Question 2: How to scale from a fixed size inner loop to an open-ended never-ending inner loop? What kind of inner/outer loss can test for this?**
    i. Inner loop should have the following features that are important in brain networks:
        1. networks are hierarchical,
        2. have small-world properties (creates local clusters),
        3. are fractal (self-similar),
        4. have scale-free topology and
        5. be close to critical state (chaotic vs stable).

d. **Question 3: Are hidden states/activations of a recurrent NN enough to feasibly allow for all the above?**
    i. Fast weights (Schmidhuber, Hinton etc.) - good for algorithmic tasks, but not shown more effective than LSTMs for general / RL tasks
        1. May have been superseded by multiplicative interactions (multiplicative LSTMs, replacing linear layers by bi-linear layers)

e. **Question 5: Can SGD still work over millions of steps inside the inner loop? Probably not. Can this be solved via auxiliary losses?**
    i. Synthetic gradient? Why do we need millions of steps? Do we want to have millions of steps or work more like (few-shot) meta-learning?

f. **Question 6: Can the above be achieved using Auto-generated tasks, e.g. by POET?**

g. **Question 7: What is the importance of algorithmic choice on both loops?**

Discussion Notes

To test whether the inner loop is 'inference only', let's say we have a distribution of tasks p(T). If the performance of the model depends on changes in the relative distribution p(T) -> p(T)*q(T) then it's likely to just be inference. Can we directly ask for invariance with respect to this transform (e.g. via an adversary on the task distribution?).

Maybe we should discuss if and how should experts evolve during the inner loop, is hidden state enough? E.g. We can make the topology between the experts trainable in the inner loop. Or update the weights locally (something like kickback, synthetic gradient?).

many notes were incorporated in the text above

Testable Hypotheses

1. There is an optimal ratio of outer/inner loop learning that is determined by:
    a. Computational bounds (amount of parallel compute, amount of execution time)
    b. Difference between past and future (e.g., the mutual information between seen and unseen task distributions)

Experiments

Some possible areas of experimentation:

1. Necessity (or not) of the inner loop, balance outer/inner loop
2. Forcing inner loop learning
3. Scaling up of inner loop
    a. outer loop loss function that encourages scaling
    b. inner loop auxiliary losses
    c. auto-generated tasks

What tasks do we need for such experiments?

Possible experiments:

1. Using rollouts of a small number of steps: does it lead to a stable strategy that runs for an infinite time, without exploding or vanishing?
2. Can we learn a "composite" policy by running short rollouts instead of one long rollout?
3. 3 loops training: how much time is needed for learning a general, adaptive algorithm trained on a multitude of tasks? What will those tasks be / what is the desired adaptive behavior? Is the required time feasible?
4. Show how optimal ratio of outer/inner loop training varies with computational bounds, past-future KL divergence, compare to training with only outer loop (what's the simplest setting that can do this?)

Architectures

1. Combination of MCTS/A* with “memory unit” experts that hold heuristic information used by the search. Given a specific situation, the search asks what experts may help and those that can help decide the best search strategy.
    a. Alternative version: the search algorithm is learned within the experts.
    b. Advanced version: make the planning component hierarchical; goals can be hierarchical, actions too. The number of actions should not be limited; if one wants to define a new action, they should specify what states it applies to and what states it leads to.
2. Inspired by Finding online neural update rules: Outer loop is a meta-learning update policy that updates weights and hidden states of experts during the inner loop. Weight w_{ij} is not a scalar but a small vector instead (so slow/fast learning can be expressed by update policy). Outer loop backprop is not through the whole inner_loop, but just from a sampled window.
3. Inspired by Incremental embodied chaotic exploration of self-organized motor behaviors with proprioceptor adaptation: Autonomous open-ended learning of goal-directed, self-organized complex behaviors. Applied to multi-agent scenarios.

References

Literature search

Some areas of interest:

1. Innate vs. acquired skills — what controls the innate/acquired balance in humans and other organisms?
2. Losses that help inner-loop scalability
3. Auxiliary losses
4. Expressivity / limitations of RNN activation states
5. Fast weights
6. Lifelong learning / learning without forgetting / memory augmentation
7. Task generation

Learning update rules and losses

[1] Gregor, 2020, Finding online neural update rules by learning to remember
Summary: about learning the learning rules in the outer loop

[2] Bechtle et al., 2019, Meta-Learning via Learned Loss
Summary: meta-learning method for learning parametric loss functions that can generalize across different tasks and model architectures.

Task generation

POET and Enhanced POET

[3] Wang et al. 2019, Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. arXiv preprint arXiv:1901.01753.

[4] Wang et al., 2020, Enhanced POET: Open-Ended Reinforcement Learning through Unbounded Invention of Learning Challenges and their Solutions
Summary: creating open-ended algorithms, which generate their own never-ending stream of novel and appropriately challenging learning opportunities
blog post, video, github

[5] Portelas et al., 2020, Automatic Curriculum Learning For Deep RL: A Short Survey
Summary: these methods shape the learning trajectories of agents by challenging them with tasks adapted to their capacities.

Auxiliary losses, intrinsic motivation

[6] van den Oord et al., 2018, Representation learning with contrastive predictive coding
Summary: universal unsupervised learning approach to extract useful representations from high-dimensional data

[7] Etcheverry, 2020, Intrinsically Motivated Discovery of Diverse Patterns in Self-Organizing Systems
Summary: motivation: exploration of self-organizing systems

[8] Trinh et al., 2018, Learning Longer-term Dependencies in RNNs with Auxiliary Losses
Summary: simple method that improves the ability to capture long term dependencies in RNNs

Memory

[9] Zhang et al, 2019, Equilibrated Recurrent Neural Network: Neuronal Time-Delayed Self-Feedback Improves Accuracy and Stability
Summary: learning long-term dependencies in RNNs

[10] Matsuki and Shibata, 2018, Reinforcement Learning of a Memory Task Using an Echo State Network with Multi-layer Readout
Summary: Echo State Network with multi-layer readout allows to solve memory task without BPTT

[11] Hamid and Braun, 2019, Reinforcement Learning and Attractor Neural Network Models of Associative Learning
Summary: explains the importance of linking events in temporal order

Optimization Viewpoint

[12] Wichrowska, et al., 2017, Learned Optimizers that Scale and Generalize

Summary: learned gradient descent optimizer that generalizes well to new tasks

[13] Metz et al., 2018, Understanding and correcting pathologies in the training of learned optimizers
Summary: dynamically weighting two unbiased gradient estimators for a variational loss on optimizer performance, allowing us to train neural networks to perform optimization of a specific task faster than tuned first-order methods.

[14] Rabinowitz, 2019, Meta-learners' learning dynamics are unlike learners'
Summary: once meta-trained, LSTM Meta-Learners aren't just faster learners than their sample-inefficient deep learning (DL) and reinforcement learning (RL)

[15] Beatson and Adams, 2019, Efficient Optimization of Loops and Limits with Randomized Telescoping Sums
Summary: optimization problems in which the objective requires an inner loop

Meta-learning

[16] Ortega et al., 2019, Meta-learning of Sequential Strategies
Summary: memory-based meta-learning agents do Bayesian filtering

[17] Bechtle, 2019, Meta-Learning via Learned Loss
Summary: Parallel work to MetaGenRL with similar objective

[18] Kirsch et al., 2019, Improving Generalization in Meta Reinforcement Learning using Learned Objectives
Summary: MetaGenRL

Credit assignment

[19] Arjona-Medina, et al., 2018, RUDDER: Return Decomposition for Delayed Rewards
Summary: credit reassignment by jumping back in time

Expressive power of RNNs

[20] Merrill, 2020, A Formal Hierarchy of RNN Architectures
Summary: how RNN architectures relate to Chomsky's hierarchy

Priors

[21] Chollet, 2019, On the Measure of Intelligence
Summary: a new formal definition of intelligence based on Algorithmic Information Theory

Relevant Researchers

Karol Gregor

Jascha Sohl Dickstein

Luke Metz

Pierre-Yves Oudeyer

Mayalen Etcheverry

Jeff Clune

Kenneth O. Stanley

Quoc V. Le

Francois Chollet