Spring 2020

Internal Badger Workshop

Session 2: Targeting Communication

Moderators: Jaroslav Vitku and Joseph Davidson

- Tasks/losses/architectures/etc. that target communication;
- Meta-learning on communication

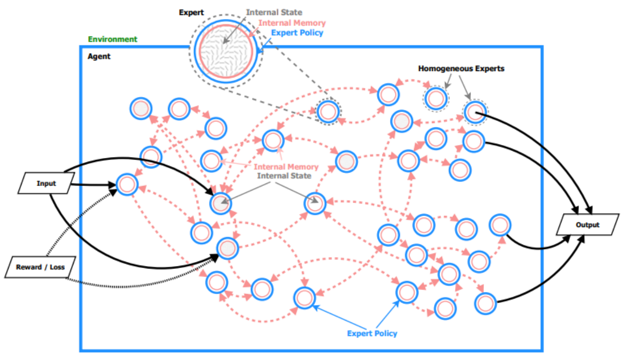
Pre-discussion Comments & Resources

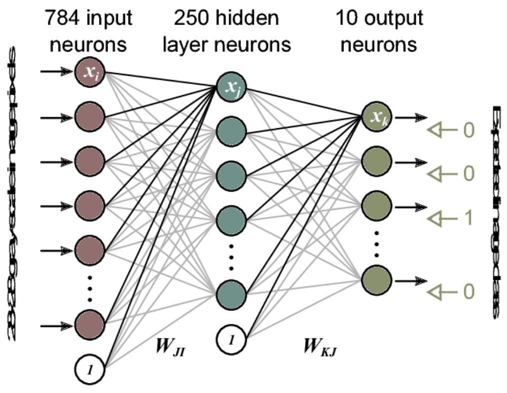
Problem description

- 1 task
- N agents
- agents have to communicate in order to solve it
- computation resources should be uniformly distributed
- this mechanism should be general (enough)

Questions:

- What are the SW requirements (what the communication protocol should do)?
- What are the HW requirements (based on the above)?





Examples of tested communication schemes

Architectures:

For our communication experiments we have made use of CommNet (1) and IC3Net (2) architectures. These offer some of the simplest baselines for communication, where CommNet averages out all the hidden states of the agents/experts, and IC3Net does the same except each agent has a binary choice of whether to communicate or not.

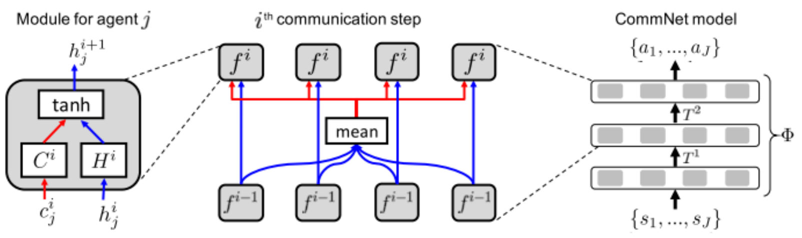


Figure 1: An overview of our CommNet model. Left: view of module f^i for a single agent j. Note

However, these architectures have scalability issues - as more experts are added, the less any one expert contributes to the communication. This becomes an issue if an expert has some private info that it needs to communicate. More complex architectures like ATOC (3) or TarMAC (4) could be reasonable approaches for scalability in communication.

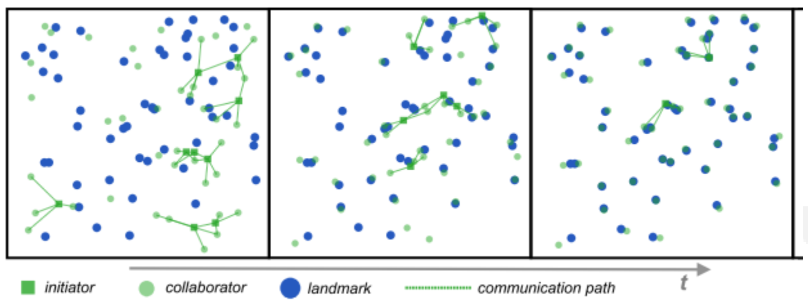


Figure 4: Visualizations of communications among ATOC agents on cooperative

Tasks:

While the guessing game with an appropriate amount of symmetry-breaking doesn't necessarily need communication, it can serve as a testbed for communication. We have made use of the particles environment and have drawn up scenarios in there to test communication. However it is challenging to create tasks which are simple, but which cannot be solved in an unintuitive fashion (such as the guessing game above)

One challenge could be in the simplicity <-> symmetry dynamic where simpler tasks have sparser observations, and sparser observations induce symmetry which inhibits communication

Discussion Notes

Possible topics for discussion

Scalability of communication / communication topologies

- Scalable/non-scalable communication schemes
- Targeted communication
- Locality definition?

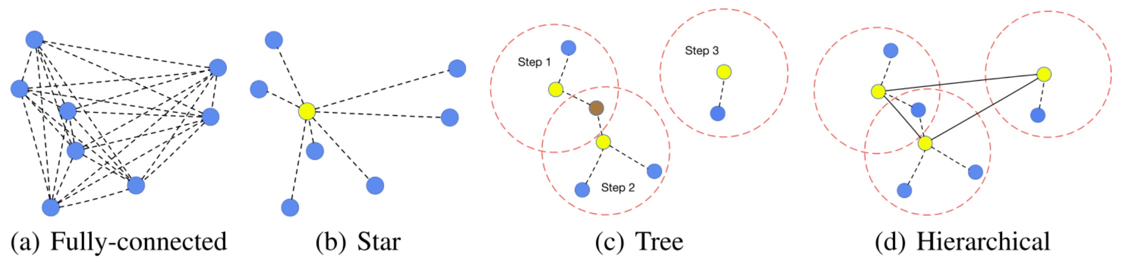


Figure 1: Topology of different communication structures and LSC falls into the hierarchical one.

source: [LSC]

Decentralized problem solving

How should the communication protocol look like?

E.g. if the task is known:

- Count the number of agents = N
- Task decomposition into N subtasks (coordinative communication)
- Negotiation which agents solve what sub-problems (mapping subtasks to agents)
- Establishing the communication topology
- Each agent solves sub-task (communication information transmission)
- Solution composition

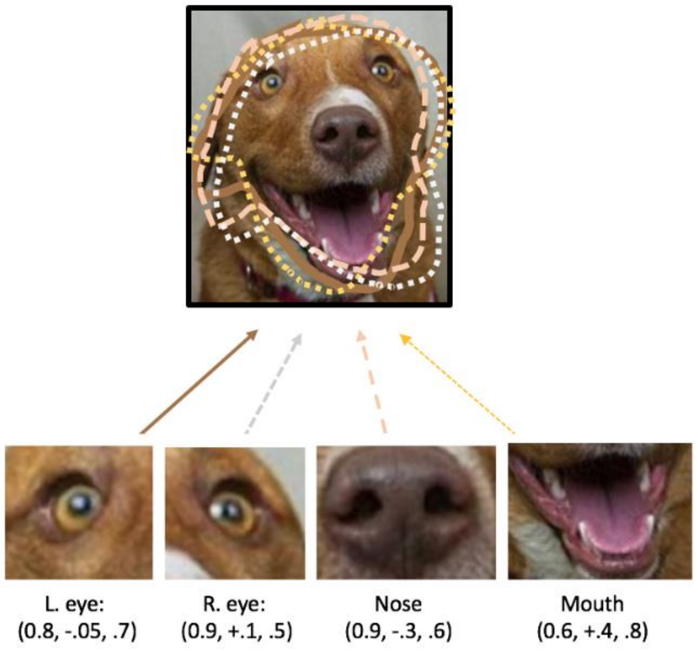
Communication related to inner/outer loop

What should be the property of the communication in the meta-learning setting?

Inner loop, maybe:

- at the beginning, the experts should negotiate some communication topology/structure
- then keep the structure until the end of the rollout

Possible inspiration: dynamic routing between capsules [blog, blog]



Notes from the meeting

- Different types of communication:
    - coordination
    - control
    - information exchange
- Adding more experts is beneficial
    - introduce specialization
    - since in case of the targeted communication they are doing the same
- Question:
    - what is the property of communication?
        - bottleneck? communication is then task independent
- How does communication differ from data/information transfer in NNs, NAS, Routing?
    - It's all about scalability and generality - comms allows this
    - But isn't that rather the multi-agentness?
- One view on what comms is - a way to introduce a beneficial bottleneck to the system. Something that enforces generalization and an API for scalability. Ideally comms should include task independent constructs/symbols/objects/refs that can be used across any task
- Rather than communicating task-specific data, we want to learn to communicate as a learning algorithm

- Seems analogous to the transmission of programs between experts learned in the inner loop
- Why does `comms` work in NNs, but not yet in Badger? How do they differ and why?
    - NN 'comms' have associated weights-per-bit, but in some of the methods we have tried, those weights aren't there, and communication is over a fixed-width vector which aggregates possibly many bits without weighting them. One of our Badger versions showed a hardwired comm scheme working in the first badger experiments.
- Information Transmission, Units and Filters
- Task is more important than architecture - example - routing task
- Hidden information games - a natural fit for multi-agent testing as it requires comms
    - How come it is hard even for an LSTM to learn those?
- There could be two perspectives on communication: Spreading important information (say only one expert getting the error), and getting a holistic view of the task (seeing what your neighbour sees and what they are doing).
    - The act of communicating is good only when you have something diverse and useful to say, so we need a way for the expert to judge if it is going to contribute anything meaningful. But there should be a distinction between 'important info you don't have' and 'here is what I'm going to try'