Autonomous Discovery of Temporal Abstractions from Interaction with an Environment

Amy McGovern* (amy@cs.umass.edu) Andrew G. Barto (barto@cs.umass.edu)

Autonomous Learning Lab (www-all.cs.umass.edu), University of Massachusetts Amherst

Motivation

- Abstraction enables an agent to:
  - Reduce the search space
  - Learn and plan at a higher level (e.g. by forming hierarchies)
  - Efficiently transfer knowledge and quickly react to new situations
  - c.f. macros, chunks, skills, behaviors
- Focus on ***temporal abstractions*** which allow an agent to ignore details of time
- Dominant approach: hand-craft abstractions
- **Goal:** *Autonomously* discover useful temporal abstractions which
  - Enable effective knowledge transfer
  - Accelerate learning, where appropriate

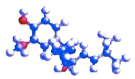
Approach: Learn from Interaction

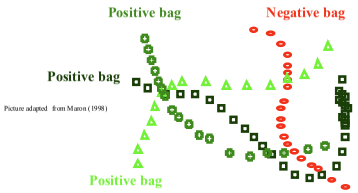
- Detect and create temporal abstractions from accumulated experience with an environment
- *Identify patterns* and create two types of temporal abstractions:
  - Subgoals of achievement
  - Action sequences
- Experience can come from many types purposeful behavior. E.g.:
  - Reinforcement learning
  - Heuristic search
  - Human tele-operation
- Use ***options*** to represent temporal abstractions (Sutton et al., 1999; Precup 2000)
  - Example: **walk-to-coffee-machine**

Identifying Useful Subgoals

- **Goal**: identify useful ***subgoals of achievement***
  - example: **go-to-doorway**
- Approach: identify ***bottleneck*** regions in sensory space: regions that are visited on multiple successful paths to a goal but not on unsuccessful ones (c.f. "Narrows" Amarel, 1968)
- Use Multiple Instance Learning framework to define the task (Dietterich, Lathrop & Lozano-Perez, 1997)
  - ***Target concept c***
  - ***Bags*** B_i: collections of instances (e.g. sensations or states) B_{ij} with one label
    - ***Positive bags*** B_i^+: contain at least one instance in target concept
      - e.g. sensations from a *successful* path to a goal
    - ***Negative bags*** B_i^-: contain no instances in target concept
      - e.g. sensations from an *unsuccessful* path to a goal
  - Subgoals = solution to multiple instance learning problem

Example: Identify molecule configuration that reacts with cell

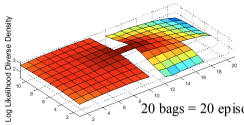




Diverse Density (Maron 1998): Best concept is at the **intersection of the positive bags minus the union of the negative bags**

Subgoal Discovery Algorithm

While solving a task
- Act purposefully, add experience to database
- Create +/− bags from sensory readings
- Search for diverse density peaks
- For each potential subgoal ***g*** found
  - Update ***g***'s running average of peak frequency
  - If ***g***'s average is above threshold and it passes the static filter
    - Create a new option to reach ***g*** and initialize using stored experience



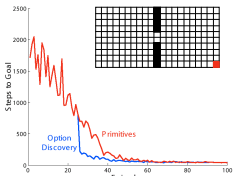
20 bags = 20 episodes

Subgoals discovered

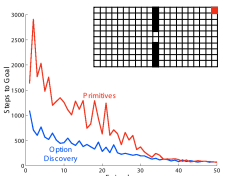
Experimental Results: Subgoals

Using reinforcement learning

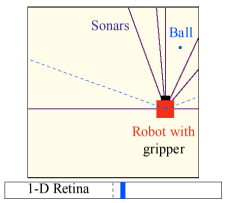
Task: Move from random state in left hand room to red square as quickly as possible



Online Learning Results

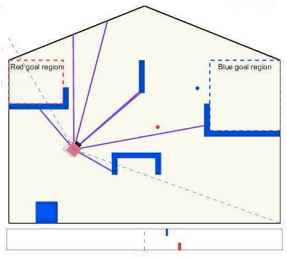


Knowledge Transfer Results



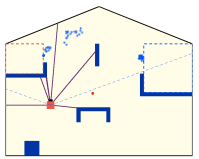
- **Task:** Move ball to within one unit of wall (without hitting wall)
- Identified subgoals in retinal space using online RI behavior
- Results:
  - **Subgoal: have ball in center of retina and close to gripper**
  - Accelerates learning in new task

Using human tele-operation data



Retinal subgoals: have ball in center of retina and close to gripper

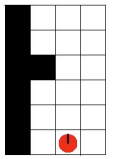
Task: Use robot to put red ball in red goal region and blue ball in blue goal region without hitting obstacles or walls



Cartesian subgoals

Identifying Useful Sequences

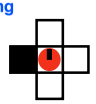
- Approach: Search for sequences of consecutive actions that occur *frequently* across successful trajectories
- Main advantages to using sequences:
  - Reduce effective depth of search
  - Create hierarchies of actions
  - Require little computational overhead
- Create two types of sequences:
  - *Action sequence*
  - *Conditionally terminating sequence*



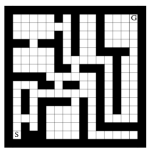
Experimental Results: Sequences

Using reinforcement learning

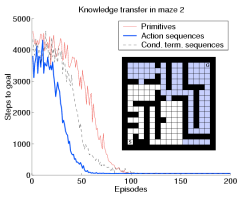
- **Task**: Identify both types of sequences in maze environment
- Reuse sequences in two new environments

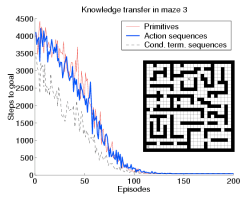


Observations



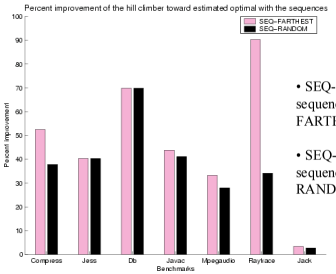
Maze 1





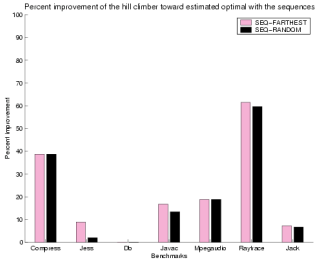
Using heuristic search in instruction scheduling task

- ***Instruction scheduling task***: Reorder assembly code within basic blocks to execute as quickly as possible (while preserving semantics)
  - Related work: Moss et al. (1997), McGovern, Moss, & Barto (2002)
  - Use Jikes RVM platform to schedule Java bytecode
- Autonomous methods for scheduling are useful because:
  - Can make a significant difference in running time on multi-pipelined machines
  - Current methods are labor-intensive
  - Architects can more fully explore the design space of a machine
- ***Sequence task***: Identify sequences from behavior of heuristic schedulers that can:
  - Improve the performance of the heuristics within a processor
  - Transfer knowledge to similar processors



- SEQ-FARTHEST: sequences created from FARTHEST heuristic
- SEQ-RANDOM: sequences created from RANDOM heuristic

Knowledge transfer in the instruction scheduling task



Sequences also improved performance on a similar processor.

Conclusions

- Introduced two methods for autonomous discovery of two types of temporal abstractions
  - Multiple Instance Learning/Diverse Density
    - Subgoals of achievement
  - Frequently occurring sequences of actions
    - Action sequences
    - Conditionally terminating sequences
- Agent's experience used to identify and create the abstractions
  - Frequent patterns in an agent's successful experience
  - Experience can come from purposeful agents
    - RL, heuristic search, human tele-operation

Future Work

- Extending diverse density
  - Other concept spaces
    - E.g. linearly separable, temporally linked
  - Noisy negative bags
    - Can negative bags contain some positive data?
  - Use relational data
- Learning from unsuccessful experience
  - Identifying places or situations to avoid
  - Focus exploration in better/safer areas
- Learning from less data
- Application to robots, with a particular focus on robots assisting a human presence in space

Acknowledgements

This work was supported by the National Physical Science Consortium, Lockheed Martin, Advanced Technology Labs, and the National Science Foundation under grant ECS-9980062 and EIA 9703217.

References

- Amarel, S. (1968). On representations of problems of reasoning about actions. In Michie, D., editor, *Machine Intelligence 3*, volume 3, pages 131–171. Elsevier, North Holland, Amsterdam, London, New York.
- Dietterich, T. G., Lathrop, R. H., and Lozano-Perez, T. (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71.
- Iba, G. A. (1989). A heuristic approach to the discovery of macro-operators. *Machine Learning*, 3:285–317.
- Maron, O. (1998). *Learning from Ambiguity*. PhD thesis, Massachusetts Institute of Technology.
- McGovern, A., Moss, E., and Barto, A. G. (2002). Building a basic block instruction scheduler with reinforcement learning and rollouts. *Machine Learning*, 49(2/3):141–160.
- McGovern, A. (2002) *Autonomous Discovery of Temporal Abstractions from Interaction with an Environment*. PhD thesis, University of Massachusetts Amherst.
- Moss, J. E. B., Utgoff, P. E., Cavazos, J., Precup, D., Stefanovi´c, D., Brodley, C. E. & Scheeff, D. T. (1997). Learning to schedule straight-line code. In *Proceedings of Advances in Neural Information Processing Systems 10 (Proceedings of NIPS'97)*. MIT Press.
- Precup, D. (2000). *Temporal Abstraction in Reinforcement Learning*. PhD thesis, University of Massachusetts Amherst.
- Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.
- Sutton, R. S., Precup, D., and Singh, S. (1999). Between MDPs and Semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211.

* Currently at the Knowledge Discovery Lab, University of Massachusetts Amherst