Using Label Propagation for Learning Temporally Abstract Actions in Reinforcement Learning

Pierre-Luc Bacon and Doina Precup
Reasoning and Learning Lab, McGill University

May 7, 2013
We are interested in taking a network perspective to quickly find community structures for the purpose of subgoal decomposition in reinforcement learning.
Learning is difficult

- How can we learn and plan efficiently in a complex environment?
- How can learning be transferred across tasks?
- How can prior experience be reused?
A problem formalized under the framework of Markov Decision Processes (MDP) defined as a tuple \( \langle S, A, R, P, \gamma \rangle \)

- \( S \) set of states
- \( A \) is a set of actions
- \( R \) is a reward function \( R : S \times A \rightarrow \mathbb{R} \)
- \( P \) transition probabilities \( P : S, A, S \rightarrow [0, 1] \)
- \( \gamma \) a discount factor \( \gamma \in [0, 1] \)
Scaling up Reinforcement Learning

Abstraction  Building higher-level representations from an agent’s sensory stream and low-level actions

▶ State abstraction

▶ Temporal abstraction
Temporal Abstraction

- Options framework (Sutton et al., 1999)
- Max-Q (Dietterich, 1999)
- Hierarchies of Abstract Machines (HAMs) (Parr and Russell, 1998)
We adopt the MDP assumption and use Markov options.

**Option**

A triple $\langle I, \pi, \beta \rangle$ where:

- Initiation set $I \subseteq S$. The set of states from which the option can be invoked
- Option policy $\pi : S \mapsto A$
- Termination set $\beta : S \mapsto [0, 1]$. Probability of terminating the option in a given state.
Prior Work

Many papers, yet no consensus on which method might be superior.

- A. McGovern and A. G. Barto, 2001: *Diverse density*
- M. Stolle and D. Precup, 2002: Frequency of visits
- I. Menache, S. Mannor, and N. Shimkin, 2002: Max-Flow
- O. Simsek and A. G. Barto, 2009: Betweenness centrality
- A. Rad and M. Hasler, 2010: Connection graph stability, closeness centrality
- Mathew et al., 2011: Graph clustering, PCCA+
Network Perspective

State-Transition Graph
We sample the state space and connect two states if a state transition is possible under a given action.

Community
A densely connected subsets of states connected by sparse links to other such subsets
Community members provide a natural initiation set for an option

States on the boundary (access states, bottlenecks) are seen as subgoals

A community is a dynamically stable region of the state space transitioning to other such regions through bottlenecks

Recent neural correlates consistent with this model (Cordova et al. in preparation)
Figure: State-transition graph for the 4-rooms domain
Community detection is usually expensive (as in $O(n^3)$-expensive) to compute. Raghavan et al. propose a near linear time stochastic approach in *Near linear time algorithm to detect community structures in large-scale networks* (2007).

It can be easily applied on a 100K nodes graph in a few seconds on my laptop.
Label Propagation Algorithm (LPA)

1. Assign an initial unique label to every vertices in the graph $G$
2. Arrange the vertices in a random sequence $X$
3. For every vertex $v \in X$, take the label which appears the most frequently in the neighborhood of $v$ and break ties uniformly at random.
4. Repeat steps 2 and 3 as long as there remain vertices with non-maximal labels
Weighted Label Propagation Algorithm (WLPA)

Execute the same steps as LPA but build consensus sets based on the labels with the most weight when grouped together.
Options Construction with LPA

1. Sample the state-space
2. Build the transition graph, setting as a weight the number of time a transition was taken
3. Prune edges with weight less than some threshold (optional)
4. Run weighted label propagation
5. Identify subgoals
6. Create and learn options

Figure: Four-rooms domain. Each room is a 5x5 square grid.
Figure: LPA over the sampled transition graph in the 4-room domain.
Identifying Subgoals

We have communities but how do we identify subgoals?

1. Choose edges points only to *large enough* communities. Under the normal assumption over the distribution of community sizes, fix some threshold $\sigma$.

2. A community can have multiple edge points. Only keep the edge point per community with the highest weight to another community.
Options Discovery using Betweenness Centrality

1. Compute betweenness centrality for every node
2. Only keep those nodes which are local maxima of betweenness, i.e. no other neighbor has a higher value of betweenness.
Agent Performance using Options

The graph shows the cumulative number of steps taken by different algorithms as the number of episodes increases. The algorithms compared are:

- Betweenness-8
- LPA-10
- Primitive

The y-axis represents the cumulative number of steps, while the x-axis represents the number of episodes.
1. Different graph construction approach
2. Subgoals and initialization sets must be generalized to regions of the state space
3. Value function approximation
Graph Construction

How do we now merge multiple trajectories under the same graph? Since we now work in continuous state space, a state will never be encountered exactly again.

- Delaunay Triangulation
- Euclidean minimum spanning tree
- Fixed sized ball
- k-nearest neighbors
  - Directed
  - Mutual
  - Symmetric
1. Subsample the input points uniformly at random
2. Build a kd-tree over the subsampled points
3. Connect each point to its k-nearest neighbors
4. Set the edge weights to $W(i,j) = e^{-\frac{\|x_i - x_j\|^2}{\sigma}}$
5. Check that the resulting graph is connected
Figure: Pinball is a non-linear 4-dimensional continuous environment. Acceleration actions incur -5 penalty, no action -1, terminal state 10000. Collisions are elastic and can be used strategically to navigate the ball faster to the goal.
Figure: Number of communities detected as a function of the number of nearest neighbors used in the graph construction over 30000 nodes.
Subgoal Definition

We also need to devise a proper state abstraction mechanism for generating subgoals.

**Features Construction**  Fit multivariate Gaussian over each subgoal node, kd-tree, rp-tree

**Classifier Learning**  Discriminate between subgoals states within a community

**Non-Parametric Clustering**  Partition the states within a community
Practical Challenges

- Quality (in terms of modularity) highly depends on the proximity graph.
- Proximity graph is more expensive to compute in higher dimension (approximate KNN).
- Higher dimension also means higher sample complexity.
- Only a subset of the variable seems to be relevant for certain options.
Open Questions

- What does it mean to have subgoals?
- What are the relevant quantities to optimize? Mixing time, compression?
- Are bottlenecks sufficient?
- How can structure in the reward function be taken into account?
The code developed for this project is available on Github:

- https://github.com/pierrelux/skill-acquisition
- https://github.com/pierrelux/pypinball
- https://github.com/amarack/python-rl