

GENERALIZED POLYATHLON: A
BENCHMARK FOR
AUTONOMOUS, GENERAL,
EMBEDDED REINFORCEMENT-
LEARNING AGENTS

Brian Tanner
University of Alberta

MAIN IDEAS

- General reinforcement learning algorithms
- Robust empirical evaluation

EXCITING CHALLENGE!

- Create algorithms:
 - Applicable without specific **domain knowledge** or **reinforcement learning experience**
 - Learn **online**/maximize cumulative reward

WE HAVE ALGORITHMS

- Algorithms require specialization and tuning
 - Policy/VF Initialization
 - Function approximation strategy
 - Exploration strategy
 - Feature selection
 - Learning rates

WE EVALUATE

- Evaluation is often not online
- Often on a single domain
 - Tuned and tweaked for particular problems
- We use tricks:
 - randomized start states, cut off bad episodes, clever initial policies?

QUESTIONS REMAIN...

- Which algorithms work well **in practice**?
- How much **tuning** is needed for strong performance?
- Which algorithm will work on **my problem**?

MORE WORK REQUIRED

- We should raise the bar for empirical evaluation
- Encourage research that increases our confidence in the wider applicability and generality of our results
- Increase long term impact of these results
- Answer practical questions

MULTI-MDP EVALUATION

- Explicitly measure generality and flexibility
 - Evaluate overall performance across a set of previously unseen problems
 - No parameter/representation/feature changes between problems
- Specialized agents will not excel in this broad setting
- Exactly what we've done in the competition!

AUTONOMOUS, EMBEDDED, GENERAL

- **Autonomous** : No human intervention. No customized representations, parameter tuning, resets, teaching, etc.
- **Embedded** : Agent interacts with the world by taking actions and receiving observations. No a priori model, trajectory sampling, look-ahead search, etc.
- **General** : The algorithm does not “identify” the environment from a known set of candidates and then load a specialized agent.

CHALLENGES

- Explore/Exploit matters (no free time to bash head against wall/eat your mouse)
- Adaptive function approximation/feature selection
- Sample-efficient learning is important but batch and model-based techniques are not immediately applicable
- Probably more here...

CONCESSIONS

- Tuned/specialized methods will do better in typical, single-problem evaluations
- No free lunch : these algorithms may have bad worst-case performance on certain problems
 - Hopefully limited to adversarially conceived counter-examples

ENVIRONMENTS AND FIRST STEPS

POSSIBLE ENVIRONMENTS

- Ideally these problems represent issues that we think are “important”
 - Large/infinite state spaces
 - Non Markov observations
 - Continuous actions
 - Risk sensitive domains

POLYATHLON

- Mostly Markov
- 6-dimensional continuous observations
- 6 discrete actions
- A few domains (so far)
 - Mountain car, acrobot, continuous grid worlds, cat-and-mouse, cart-pole, pole swing up, bandits

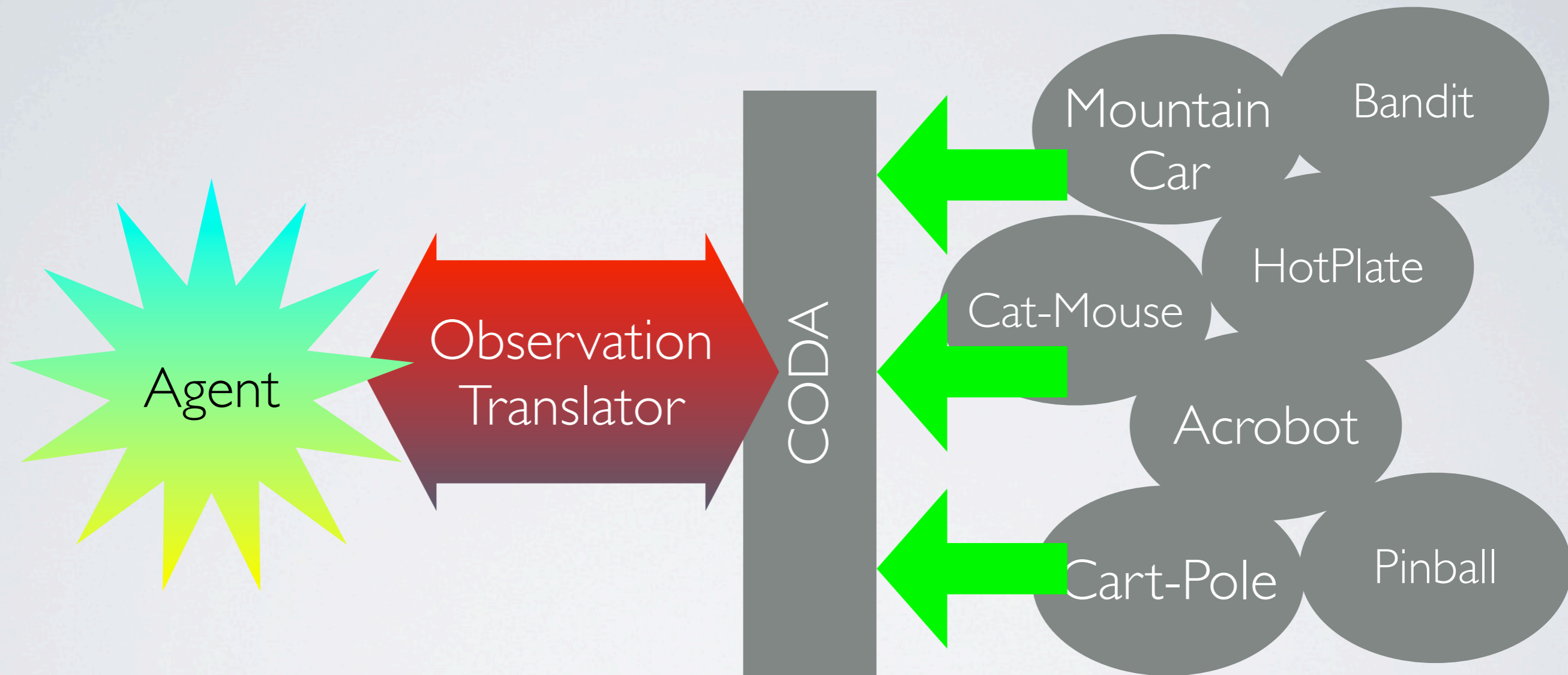
ROBUST EVALUATION

- How to do robust, independent evaluation with only 8 MDPs?
- Competition: manually “perturb” the MDPs
 - Re-order, warp the real observations
 - Add extra observations
 - Randomized, correlated with real observations
 - Add extra randomized actions

GENERALIZED POLYATHLON

- So far: manual process. Takes time and can be error-prone
- Build a parameterized MDP generator
 - **generalized domains**
- Can sample MDPs from the implied distribution!

GENERALIZING POLYATHLON



GENERALIZED POLYATHLON

- Ever-growing set of environments “behind the curtain”
 - Provides a counterweight to method overfitting
 - New environments can be chosen to deliberately plug “holes” in the existing set

BENCHMARKING

BENCHMARKING

- Should be easy to compare performance
 - Published numbers should be comparable
 - Results should be reproducible (code sharing)

BENCHMARKING : SHARED DATASETS

- Fixed public training/proving/testing sets
- Practitioners can download public data sets of training, proving, testing MDPs
- Honor system: practitioners don't use test set until final evaluation
- Easy! Results comparable between publications!
- Incentive to cheat/overfit weakens comparisons

BENCHMARKING : DYNAMIC DATASETS

- Generate new sets for each publication
 - **Publish code and MDPs with paper**
 - Run competitor (RL-Glue compatible) algorithms on the set your generate when evaluating your algorithm
 - More work, but still pretty easy!
 - Comparisons are within the publication

BENCHMARKING: SECURE BENCHMARK

- Similar to shared data sets
 - A remote machine runs final evaluations
 - Reduces overfitting; no closed loop
 - Public “leaderboard” (works for GO)
 - Easy to have community re-run top algorithms when data sets change

DISCUSSION

DISCUSSION

- Good idea?
- What domains?
- How to benchmark?
- Would you use it?