Reducing Planning to Classification

Alan Fern

Joint work w/ SungWook Yoon and Bob Givan

Electrical and Computer Engineering

Purdue University
Available actions (possibly stochastic):

Pickup(x)
PutDown(x,y)
Relational MDP / Planning Domain

State-of-the-art on AI planning benchmarks.
Learning to Act


Consider class of policies $C$. Observe $O(\log |C|)$ trajectories of target policy in $C$.

If policy $\pi$ in $C$ is consistent with trajectories then quality of $\pi$ is “probably close” to quality of target.

Suggests a type of reduction:

1) Somehow observe trajectories of a good policy.
2) Learn a classifier to (approximately) imitate the policy.

How can we observe a good policy?
Reduction 1:
Learning to Solve Small Problems

[Khardon, AIJ 1999], [Martin&Geffner, KR 2000], [Yoon,Fern & Givan, 2002]

Small Problem Distribution

Planner

Learner

Training set:
(state1,goal1,action1)
(state2,goal2,action2)

… … …
Generalizing to Large Problems

[Khardon, AlJ 1999], [Martin&Geffner, KR 2000], [Yoon,Fern & Givan, 2002]

Why expect policies to generalize to large problems?

- Select good policy language bias.
  - Restrict expressiveness to avoid overfitting.
  - But expressive enough to represent good policies.
Experimental Domains

SBW(n) (Stochastic) Blocks World

SPW(n) (Stochastic) Painted Blocks World

SLW(t,p,c) (Stochastic) Logistics World
Unsolved Problems

• Select policies without immediate access to small problems
  ▲ Can we learn directly in a large domain?

• Improving buggy policies
  ▲ All previous techniques produce policies with occasional fatal flaws.

• Our approach: use standard MDP technique of (approximate) policy iteration
Flowchart View of Policy Iteration

$V_\pi(s) =$ “value” of following $\pi$ starting at $s$

- Compute $V_\pi$ at all states
- Choose best action at each state

Current Policy $\pi$ → $V_\pi$ → Improved Policy $\pi'$

Guaranteed finite convergence to optimal policy.

Problem: too many states
Approximate Policy Iteration

**Usual Approach:** reduce to value function approximation

- Value functions can be harder to represent than policies.
- Learning a policy directly may be more effective.

Current Policy $\pi$ $\rightarrow$ Compute $V_{\pi}$ at some states by simulation $\rightarrow$ $V_{\pi}$ samples $\rightarrow$ Learn approximation of $V_{\pi}$

$\pi' = \text{greedy look ahead wrt } V_{\pi}$
Approximate Policy Iteration

Our Approach: reduce to classifier learning

Refinement: Reduce to cost-sensitive classification. Costs based on Q-values.
Rollout: Computing $\pi'$ Trajectories

For our relational planning domains we use the FF-plan plangraph heuristic.

- Use a value estimate at these states.
Initial Policy Choice

- Policy iteration requires an initial *base policy*

- Options include:
  - random policy
  - greedy policy with respect to a planning heuristic
  - policy learned from small problems
API Results

Starting with flawed policies learned from small problems

![Graph showing success rate over iterations for SBW(10) and SPW(10) with learned policies.](image)
Starting with a policy greedy with respect to a domain independent heuristic

BW(15): Starting from FF-greedy policy

iteration

SR
AL(S)/H
Ongoing and Future Work

• Explore new policy languages
  ▶ E.g. relative value functions [Dietterich & Wang, NIPS’02]

• Approximation guarantees.

• Generalize to domains that “require search”.

• Incorporating deductive reasoning.

• Generalize to games and partial observability.
  ▶ E.g. the game of Hearts.
Questions?