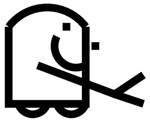
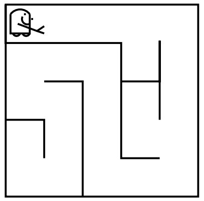
Reducing Reinforcement Learning to Classification

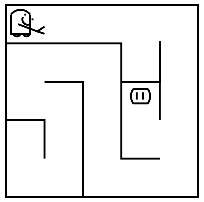
John Langford

Toyota Technological Institute at Chicago

The Standard RL Story

There once was a robot named robby. 

Who lived in Mazeworld. 

Robby was lost, and needed to find power. 

Who will help little lost Robby find his power?

Robby's Problem

1. Formalizations
   (a) What is a good model?
   (b) Markov Decision Process
   (c) Partially Observable MDP
2. A Global Reduction (RLGen)
3. Policy Search by Dynamic Programming

Formalization 1: Markov Decision Process

A = set of actions (example: {Left,Right,Up,Down})

S = set of states (example: $\{(1,1),(1,2),...,(5,5)\}$)

$P(s',r|a,s)$ = transition probability ($r \in [0,1]$ = reward)

$M = (A, S, P(s',r|a,s))$ = Markov Decision Process

Goal: Find $\pi : S \to A$ = policy maximizing:

$$\eta(\pi) = E_{(s,a,r)^T \sim M,\pi} \frac{1}{T} \sum_{t=1}^{T} r_t$$

Formalization 2: Partially Observable Markov Decision Process

O = Set of observations (example: wall ahead)

$P(o|s)$ = observation distribution

$(A, S, O, P(s', r|a, s), P(o|s))$ = POMDP

Goal: Find $\pi : O \to A$ = policy maximizing:

$$\eta(\pi) = E_{(s,o,a,r)^T \sim M,\pi} \frac{1}{T} \sum_{t=1}^{T} r_t$$

Robby's Problem

1. Formalizations

2. A Global Reduction

   (a) Generative Model

   (b) Basic Results

   (c) RLGen Algorithm (= practical variant)

3. Policy Search by Dynamic Programming

Assumption: Generative Model

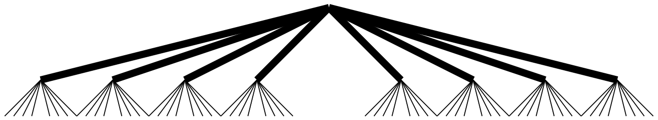
Assume: a black box $B(s,a)$ which, given a state s and an action a provides a state s', observation o, and reward r drawn from $P(s',r|a,s)$, $P(o|s')$.

Example application: You have a world dynamics simulator and want a policy which performs well in that world. Flying helicopters, walking robots, etc...

Setting Analysis

Theorem: (Sparse Sampling) Given $\left(\frac{|A|T}{\epsilon}\right)^{O(T)}$ calls to $B(s,a)$ you can execute a policy π satisfying:

$$\eta(\pi^*) - \eta(\pi) \leq \epsilon$$

Proof: 

Theorem: (Generative Lower Bound): Any ϵ-optimal solution must make $\Omega(|A|^T)$ calls to $B(s,a)$.

Proof: "combination lock" processes.

How to do Reinforcement Learning with a Generative Model B and Classification algorithm AC

Think of $\pi(a|s)$ as T classification problems: Predict the next action given the current state (or observation).

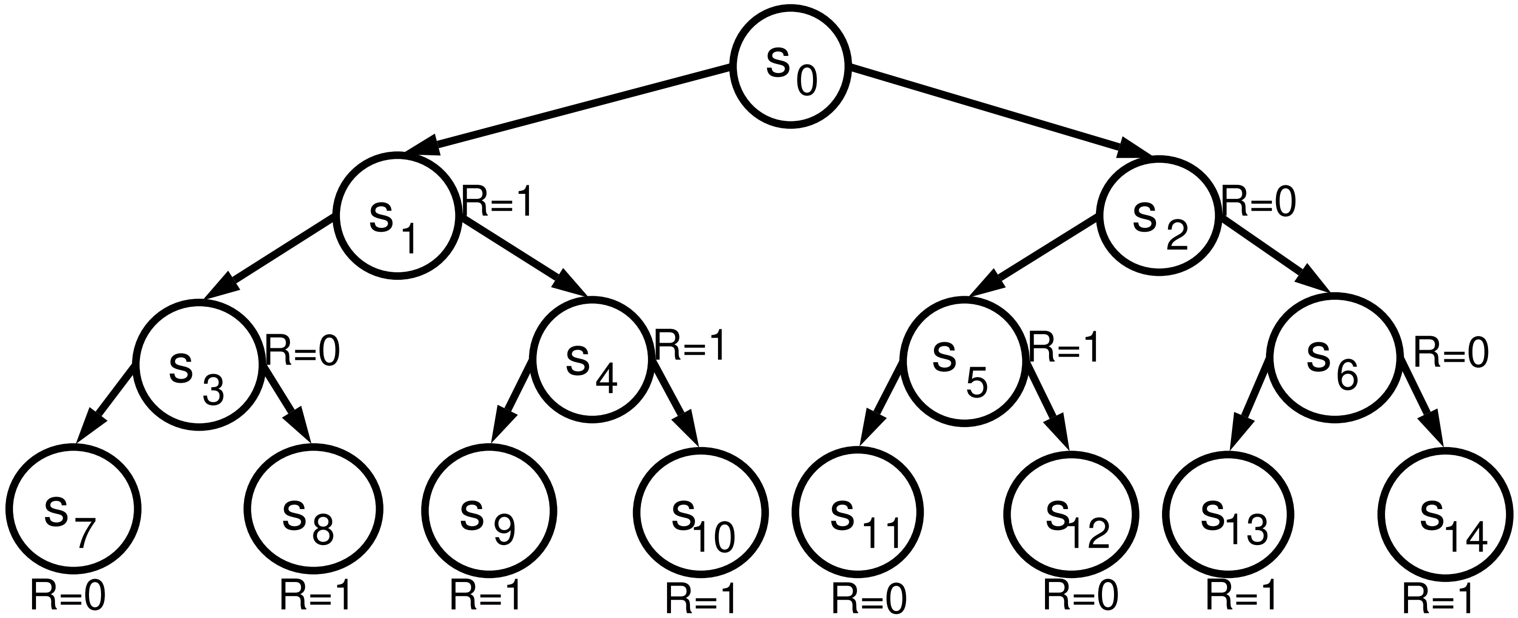
1. Construct an (importance weighted) sample set S_t from the distribution D_t for each classification problem.

2. Learn T classifiers, $c_t = AC(S_t)$

3. Return $\pi(a|s,t) = c_t(s)$

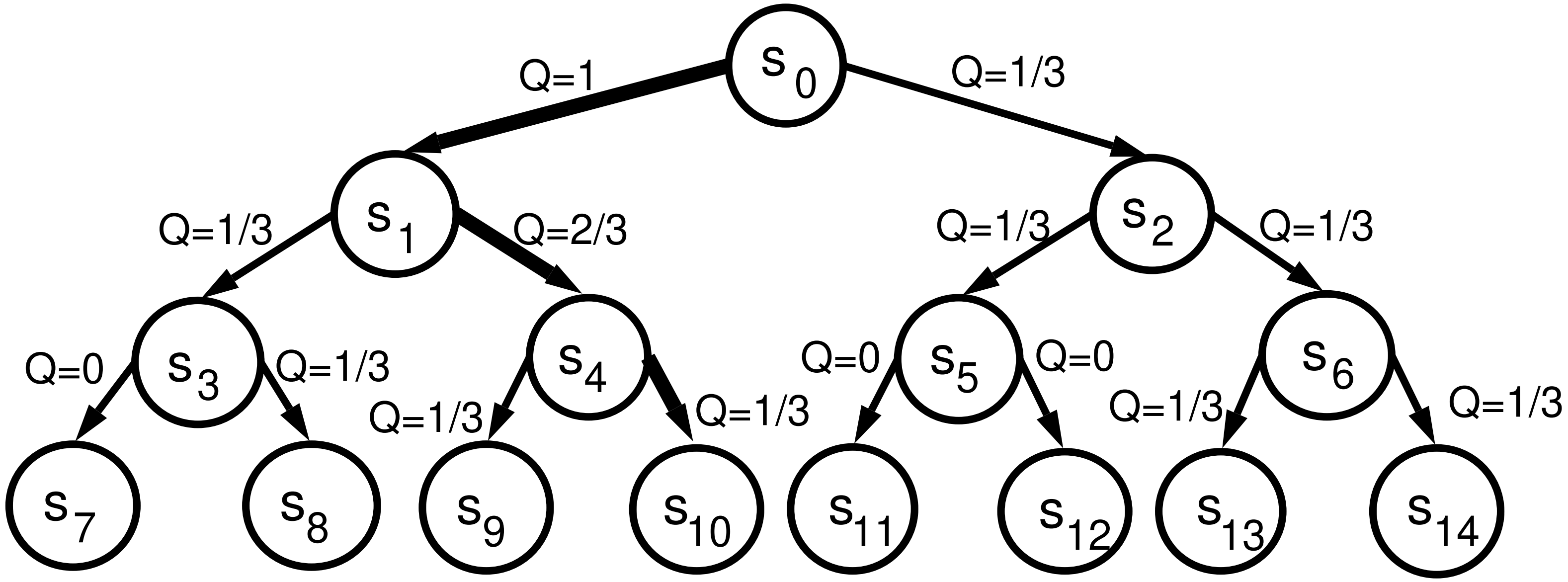
Sample Set Construction I

Build a Trajectory Tree:

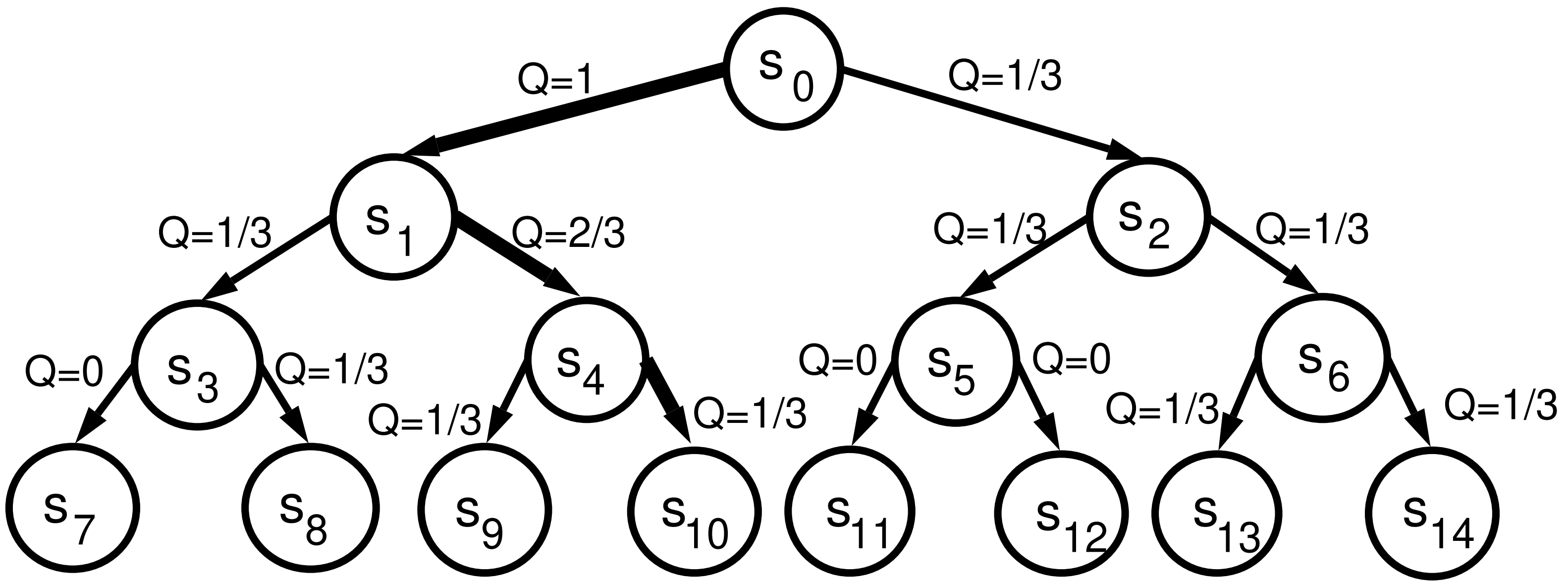


Sample Set Construction II

Find Empirical Values:



Sample Set Construction III



Extract Examples $(s_0, \mathrm{Left}, 1)$, $(s_1, \mathrm{Right}, \frac{2}{3})$, $(s_4, \mathrm{Right}, \frac{1}{3})$

Basic Analysis

Theorem: For every POMDP, if every learned classifier has error rate at most ϵ, the policy π satisfies:

$$\eta(\pi^*) - \eta(\pi) \leq \frac{T+1}{2}\epsilon$$

where π^* is the optimal nonstationary T-step policy.

Proof

Let: $N_t = E_{s,a,i \sim D_t} i$ = expected importance of tth decision

No errors $\Rightarrow$ $\eta(\pi) = N_1$ = the reward of the prescient policy which chooses the right action given random numbers.

Error rate ϵ $\Rightarrow$ loss rate at most ϵN_t (via "costing" reduction).

Worst case error $\Rightarrow$ 0 reward

$\Rightarrow$ Errors by c_t reduce reward by at most ϵN_t.

$\eta(\pi) \geq N_1 - \epsilon \sum_{t=1}^{T} N_t$

$\geq N_1 - \epsilon \sum_{t=1}^{T} \frac{T-t+1}{T}$ (Since $N_t \leq \frac{T-t+1}{T}$)

$\geq \eta(\pi^*) - \epsilon \frac{T+1}{2}$ (Since π^* not prescient + computing sum)

A More Practical Variant:

RLGen(classification algorithm AC, B, initial policy π)

1. Pick $R =$ set of times $\subseteq \{1, ..., T\}$, $k \in \{1, ...\}$, $m \in \{1, ...\}$
2. For each $t \in R$, let $S_t = m$ k-cooled classification examples made with B
3. For each $t \in R$, let $c_t = \text{AC}(S_t)$
4. If π improved (using B) when for all $t \in R$, $\pi_t \leftarrow c_t$, update π

Output: (anytime) nonstationary policy π

1. Pick $R =$ set of times $\subseteq \{1, ..., T\}$, $k \in \{1, ...\}$, $m \in \{1, ...\}$

k = cooling (noise removal) parameter. Deterministic world $\Rightarrow$ $k = 1$ is optimal.

m = training sample size for classification problems.

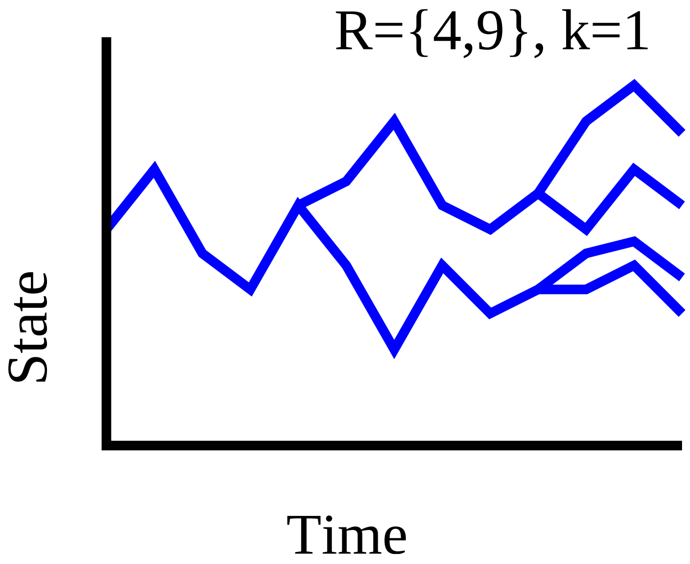
R = set of times to simultaneously optimize.

Number of calls to B is $O(mTk^{|R|}|A|^{|R|})$.

Simple heuristic: optimize low complexity choices of R, k, m while measurable improvement is made. Increase complexity of R, k, m when low complexity choices are exhausted.

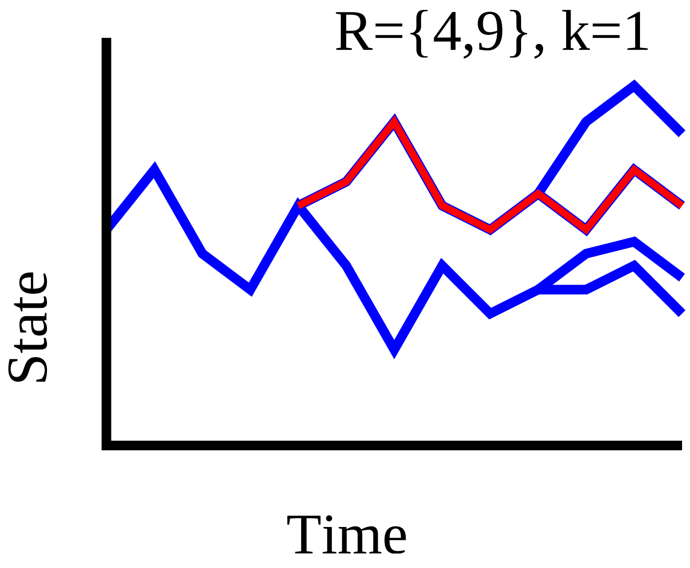
$\Rightarrow$ For any necessary complexity, the algorithmic complexity is reasonable.

2. For each $t \in R$, let $S_t = m$ k-cooled classification examples made with B



Make a tree with node = state drawn from B and policy π with k branches for every action at timesteps in R.

2. For each $t \in R$, let $S_t = m$ k-cooled classification examples made with B



Pick highest (average over k) reward path. Classification example at timestep $t = (s_t, a_{\text{path}}, \text{Importance on path})$.

3. For each $t \in R$, let $c_t = \mathsf{AC}(S_t)$

4. If π improved (using B) when for all $t \in R$, $\pi_t \leftarrow c_t$, update π

To measure "improved", use B to sample from rewards given policy.

Occam's razor bound on stochastic rewards $\Rightarrow$ good choice made with high probability.

(Note the inefficient statisticians alternative.)

Guarantees

1. Anytime: property of the algorithm

2. Monotonic improvement: At iteration i, pick $\delta_i = \frac{\delta}{i(i+1)}$. Theorem: With probability $1 - \delta$, the policy reward bound always improves and is never violated.

3. ϵ-verifiability: For $R = \{1, ..., T\}$, $k = 1$, if each learned classifier has error rate ϵ, $\pi = (c_1, ..., c_t)$ satisfies $\eta(\pi^*) - \eta(\pi) \leq \frac{T+1}{2}\epsilon$

4. ϵ-optimality: For $R = \{1, ..., T\}$, $k=$ large Sparse Sampling implies first step of policy is ϵ-near optimal.

Robby's Problem

1. Formalizations

2. A Global Reduction (RLGen)

3. Policy Search by Dynamic Programming

Policy Search by Dynamic Programming

Input: Classifier learning algorithm AC, D_t = distribution of optimal policy at timestep t, B = generative model

Algorithm: For $t = T$ down to $t = 1$ do

Construct samples S_t using $c_{t+1}, ..., c_T$, D_t and B

$c_t = S_t$

done

return $\pi(a|s, t) = c_t(s)$

Sample construction:

1. Draw s from D_t
2. For each a, act according to $\pi_{t+1}, \ldots, \pi_T$
3. Let example $= (s, \arg\max_{a \in A} \sum_{i=t+1}^{T} r_i, \sum_{i=t+1}^{T} r_i)$

Note: $O(|A|T^2)$ rather than $O(|A|^T)$

PSDP: Basic Analysis

Theorem: For every POMDP, if every learned classifier has error rate at most ϵ, the policy π satisfies:

$$\eta(\pi^*) - \eta(\pi) \leq T\epsilon$$

where π^* is the optimal nonstationary T-step policy.

Proof: Similar to RLGen

Results

Can solve various standard mazes.

Can solve double pole balancing problems.

Can solve walking robot problems.

Things to Remember

Classification $+$ Generative model lets you solve Reinforcement Learning

However, the reduction does a lot of work. ($\Rightarrow$ Reinforcement Learning is more than just classification)