Generalized Prioritized Sweeping

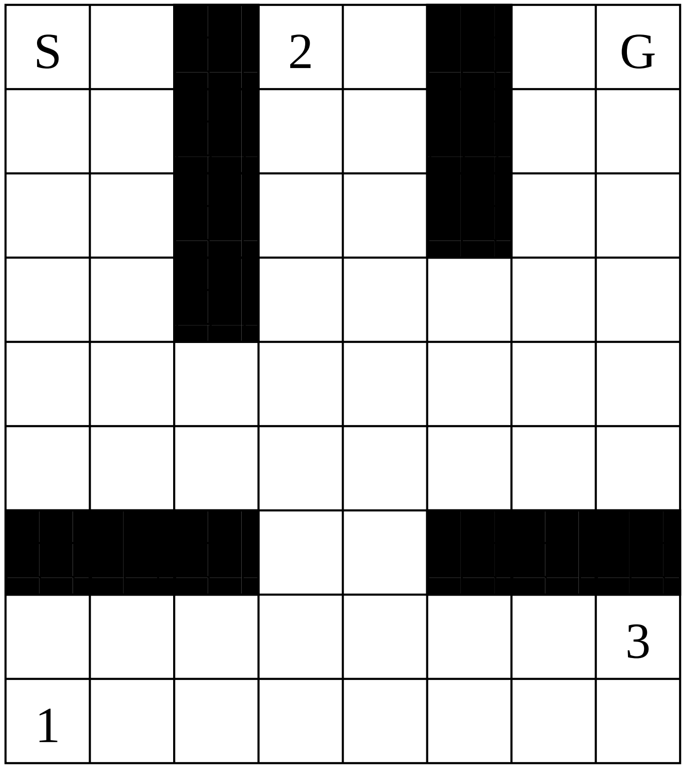
David Andre
with Nir Friedman and Ronald Parr

CMU and University of California, Berkeley
Reinforcement Learning and Friends Talk,
Oct 15th, 1997

Why Model-Based RL?

- Transfer knowledge across domains
  (Model reuse)
- Principled way to convey prior knowledge
- Generalization
- Permits off-line planning

Example



Outline

- Generalized Prioritized Sweeping
  general approach for focusing computation
- Application to explicit state representation
  results in "classic" prioritized sweeping
- Application to factored models
  exploits model structure
- Application to averagers

Markov Decision Processes

Usual MDP setting:

- $\mathcal{S}$ is a set of *states*
- $\mathcal{A}$ is a set of *actions*,
- $p(t \mid s, a)$ is a *transition model*
- $r()$ is a *reward function* mapping $\mathcal{S}$ into real-valued rewards.

Goal: maximize expected discounted total reward

Model-Based Reinforcement Learning

Approach: agent learns a model of environment.

Problem: balance planning and execution.

How to update the value fn after model update?

- Ideal: compute the optimal value function for current model.
  - Unrealistic—non-trivial computational task.
- Approximate: "repair" previous value function to account for changes.

Focusing Updates

GenPS Principle: Update states with largest *Bellman Error*: $E(s) = |\hat{V}(s) - \max_{a \in \mathcal{A}} \hat{Q}(s,a)|$

Motivation:

- Max bellman bounds $|\hat{V}(s) - V^*(s)|$ [Williams & Baird]
- $|\hat{V}(s) - V^*(s)|$ bounds *policy loss*

Policy Loss: optimal – current performance

Tracking Bellman Errors

When can the Bellman Error at s change?

- Change in model at s:
  - $\hat{r}(s)$ changes
  - $\hat{p}(s' \mid s, a)$ changes
- Change in values of successor states:
  - $\hat{V}(s')$ changes and $\hat{p}(s' \mid s, a) > 0$

Assumption: Model/Value function parameterized by θ_M and θ_V.

- Each update results in $\Delta_\theta = (\Delta_{\theta_M}, \Delta_{\theta_V})$.

Task: Estimate $E(s)$ as fn. of Δ_θ

- We need **efficient** estimates.
- Efficient $=$ cheaper than performing update

First-order approx:

$$|\Delta_{E(s)}| \approx |\nabla \max_a \hat{Q}(s,a) \cdot \Delta_\theta|$$

Problem: differentiating a $\max()$.

Solution: upper bound change

$$\left|\nabla(\max_a \hat{Q}(s,a)) \cdot \Delta_\theta\right| \leq \max_a \left|\nabla \hat{Q}(s,a) \cdot \Delta_\theta\right|.$$

The algorithm, updating priorities

procedure update-priorities (Δ_θ)

for all $s \in \mathcal{S}$

pri$(s) \leftarrow$ *pri*$(s) + \mathrm{max}_a \, |\nabla \hat{Q}(s,a) \cdot \Delta_\theta|$.

The algorithm, main loop

loop

  perform an action

  update the model

  update-priorities(Δ_θ).

  while there is available computation time

    $s^{\max} \leftarrow \arg\max_s \textit{pri}(s)$

    perform value-propagation for $\hat{V}(s^{\max})$

    update-priorities(Δ_θ).

    $\textit{pri}(s^{\max}) \leftarrow |\hat{V}(s^{\max}) - \max_a \hat{Q}(s^{\max}, a)|$

Explicit-State Representation

θ consists of:

- $\theta_{\hat{r}(s)}$—estimated reward at s.
- $\theta_{\hat{V}(s)}$—estimated value at s,
- $N_{s,a,t}$—the number of times the execution of a at s lead to state t.

$$\hat{p}(t \mid s, a) = \frac{N_{s,a,t} + N^0_{s,a,t}}{\sum_{t'} N_{s,a,t'} + N^0_{s,a,t'}}$$

Reduction to Classic Prioritized Sweeping

We compute the necessary rules for the two possible updates.

- Model update:
  After $s \xrightarrow{a} s'$ only priority of s changes.
- Value update:
  After changing $\hat{V}(s)$,
  $\Delta_{\text{priority}}(s) = \gamma \hat{p}(s \mid s', a) \Delta\theta_{\hat{V}(s)}$

Matches "classic" prioritized sweeping [Moore and Atkeson].

Value Function Approximation

We can apply similar analysis to value function approximations.

Averagers [Gordon]:

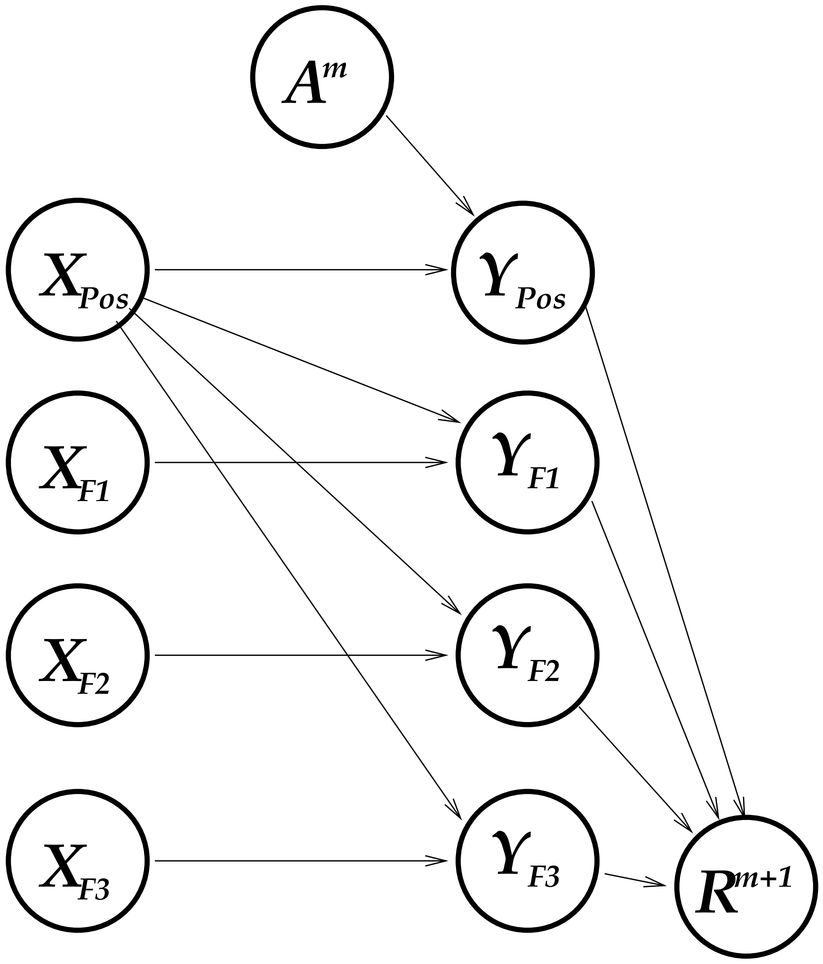
$$\hat{V}(s) = \beta_s k_s + \sum_u \beta_{s,u} \hat{V}(u)$$

Idea: only store values of u's.

After updating $\hat{V}(u)$, priority update for u' is:

$$\gamma \Delta \hat{V}(u) \cdot \sum_s \hat{p}(s \mid u', a) \beta_{s,u}$$

Factored Representation



In general,a graph specifies the *factorization* of the conditional distribution:

$$P(Y_1, \ldots, Y_n \mid X_1, \ldots, X_n, A_1, \ldots, A_k) = \prod_{i=1}^{n} P(Y_i \mid \mathit{Pa}_i)$$

Assumption: learner is supplied with the structure.

We estimate probabilities using **Dirichlet** priors:

$$\theta_{i,y,z} = \frac{N_{i,y,z} + N^0_{i,y,z}}{N_{i,\cdot,z}}$$

- $N_{i,y,z}$—number of transitions s.t. $Y_i = y$ and $\mathit{Pa}_i = z$.
- $N_{i,\cdot,z} = \sum_{y'} N_{i,y',z} + N^0_{i,y',z}$.

Priority Updates in Factored Models

After a transition, only one count per family changes:

$$\nabla Q(s,a) \cdot \Delta_\theta = \sum_i \frac{\partial Q(s,a)}{\partial N_{i,y_i^*,z_i^*}}$$

Notation: $(s,a,t) \models (X = x)$, if X is assigned the value x in $s \xrightarrow{a} t$.

Gradient equation for Factored Models

If $(s,a) \models \textit{Pa}_i = z_{i^*}$ then

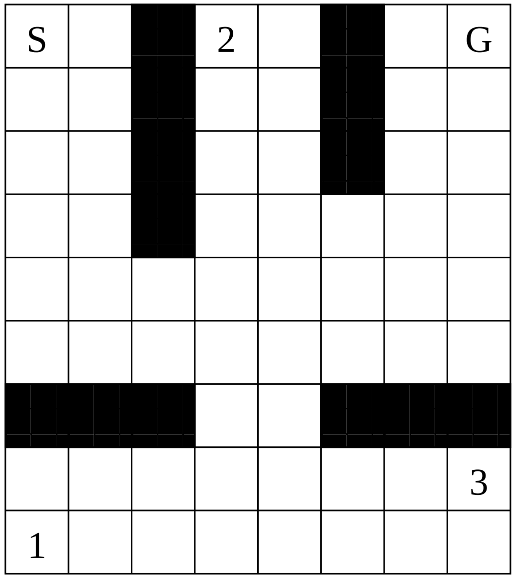
$$\frac{\partial Q(s,a)}{\partial N_{i,y_i^*,z_i^*}} =$$

$$\frac{\gamma}{N_{i,\cdot,z_i^*}} \left\{ \frac{1}{\theta_{i,y_i^*,z_i^*}} \sum_{t:(s,a,t)\models y_i^*,z_i^*} \hat{p}(t \mid s,a)\hat{V}(t) \right.$$

$$\left. - \sum_{t:(s,a,t)\models z_i^*} \hat{p}(t \mid s,a)\hat{V}(t) \right\}$$

Updates in the Flag Word

After a transition from (pos,$F_1 \ldots F_n$):
Update priority for all states, (pos, $F'_1 \ldots F'_n$)



Results

Complications

Problem – for most domains, we do not want to sum over all states.

Solution: We limit the states by using an upper bound on the gradient.

Then, we can use simpler similarity metrics to determine priorities.

These details are topic for current research.

Conclusion

- Generalization of prioritized sweeping
- Works with general model representations
- Works with value function approximation
- Reduces to “classic” PS for simple cases

Future Work

- Fast approximate updates in large models
- Different types of val fn approximators
- CPT function approximation
- Empirical results combining
  value fn approximation *and* factored models
- Learning DBN structure