Density Estimation and MDPs

Ronald Parr

Stanford University

Joint work with Daphne Koller, Andrew Ng (U.C. Berkeley) and Andres Rodriguez

What we aim to do

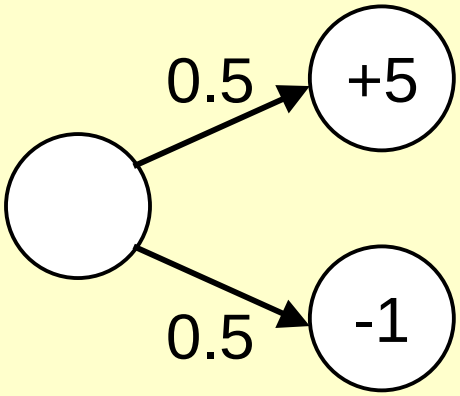
- Plan for/control complex systems
- Challenges
  - very large state spaces
  - hidden state information
- Examples
  - Drive a car
  - Ride a bicycle
  - Operate a factory
- Contribution: novel uses of density estimation

Talk Outline

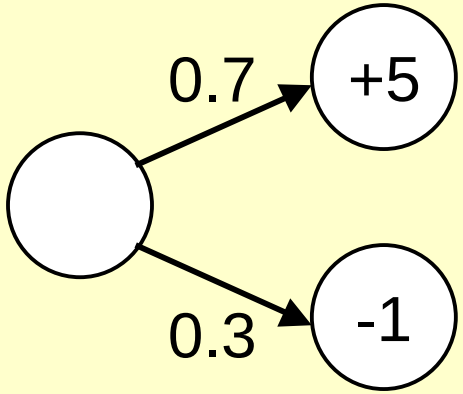
- (PO)MDP overview
- Traditional (PO)MDP solution methods
- Density Estimation
- (PO)MDPs meet density estimation
  - Reinforcement learning for PO domains
  - Dynamic programming w/function approx.
  - Policy search
- Experimental Results

The MDP Framework

- Markov Decision Process
- Stochastic state transitions
- Reward (or cost) function



Action 1



Action 2

MDPs

- Uncertain action outcomes
- Cost minimization (reward maximization)
- Examples
  - Ride bicycle
  - Drive car
  - Operate factory
- Assume that full state is known

Value Determination in MDPs

- Compute expected, discounted value of plan
- s_t - random variable for state at time t
- γ - discount factor
- $R(s_t)$ - reward for state s_t

$$V(s_0) = \sum_{t=0}^{\infty} \gamma^t R(s_t)$$

e.g. Expected value of factory output

Dynamic Programming (DP)

$$V^t(s) = R(s) + \gamma \max_a \sum_{s'} P(s' \mid s, a) V^{t+1}(s')$$

- Successive approximations
- Fixed point is V*
- $O(|S|^2)$ per iteration
- For n state variables, $|S|=2^n$

Partial Observability

- Examples:
  - road hazards
  - intentions of other agents
  - status of equipment
- Complication: true state is not known
- “state” depends upon history
- information state = dist. over true states

DP for POMDPs

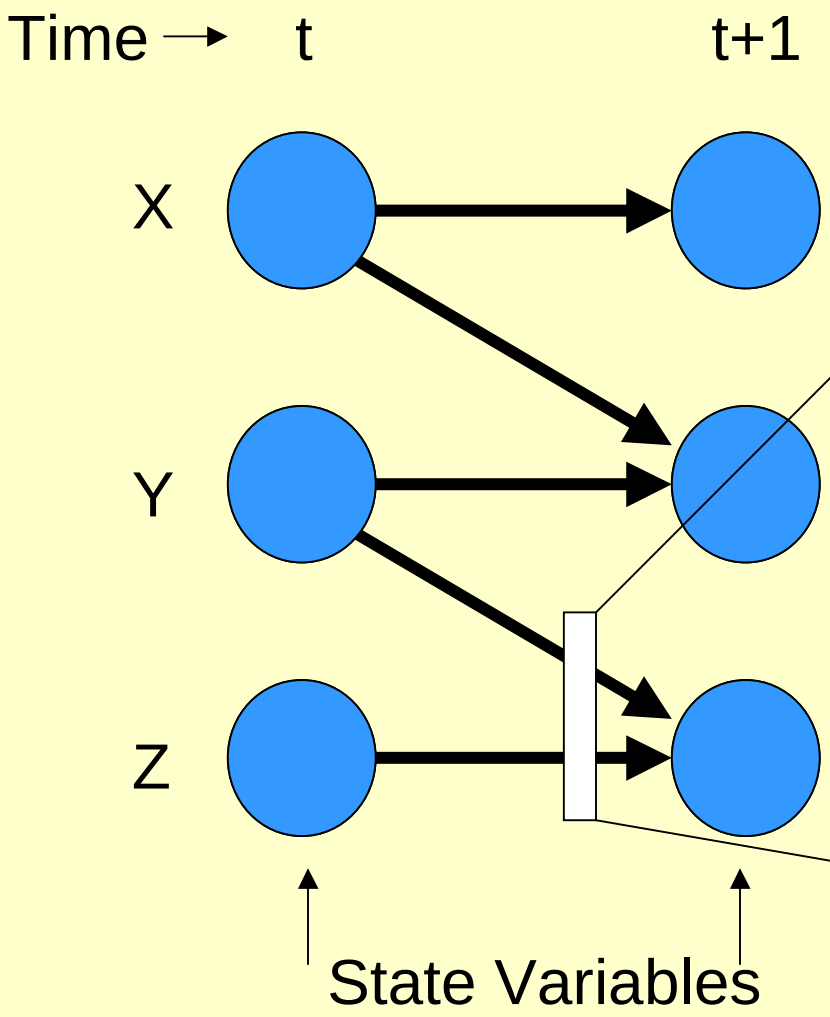
$$V^t(s) = R(s) + \gamma \max_a \sum_{s'} P(s' \mid s, a) V^{t+1}(s')$$

- DP still works, but
- s is now a belief state, i.e. prob. dist.
- For n state variables, dist. over $|S|=2^n$ states
- Representing s exactly is difficult
- Representing V exactly is nightmarish

Density Estimation

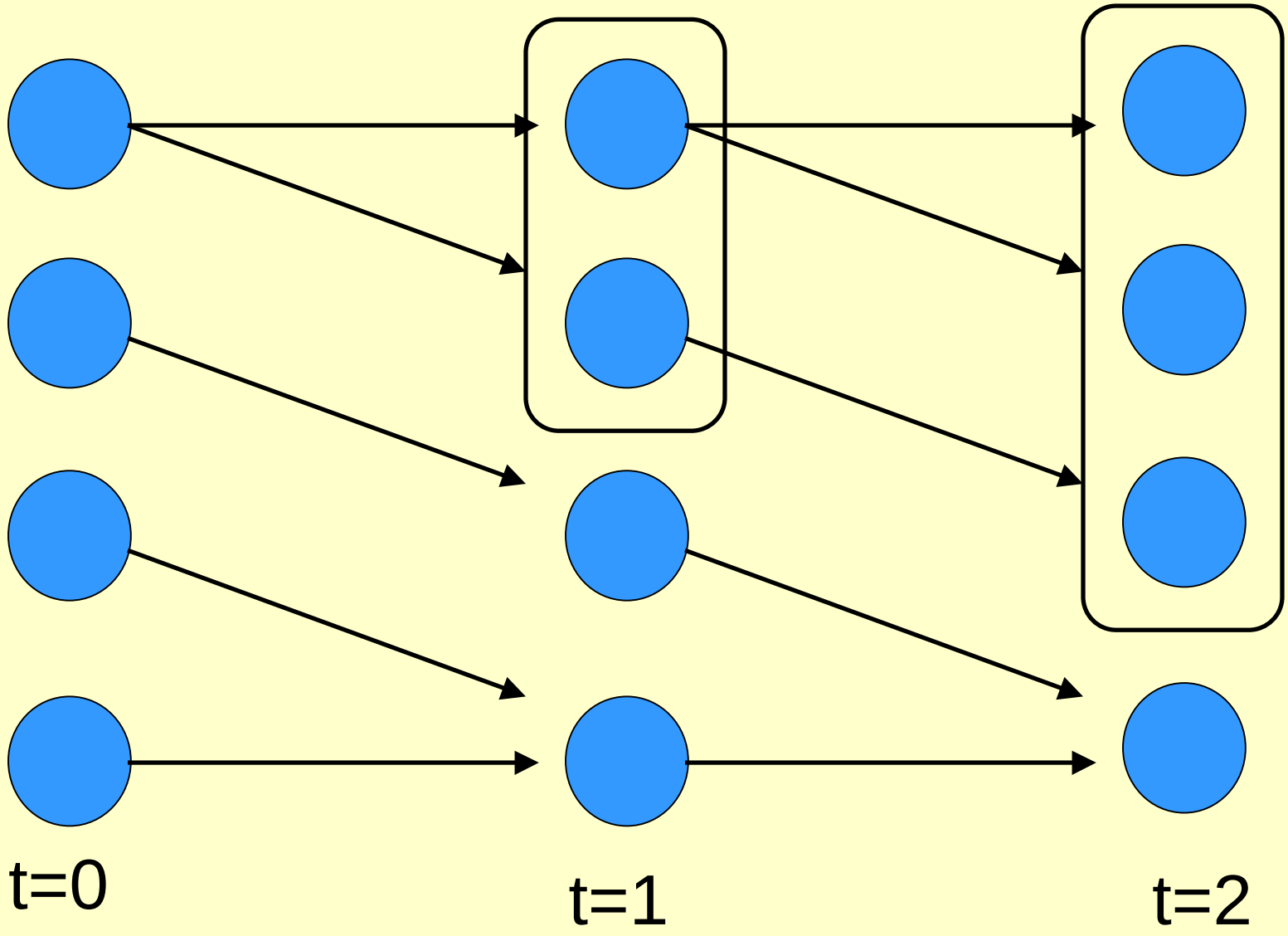
- Efficiently represent dist. over many vars.
- Broadly interpreted, includes
  - Statistical learning
    - Bayes net learning
    - Mixture models
  - Tracking
    - Kalman filters
    - DBNs

Example: Dynamic Bayesian Networks

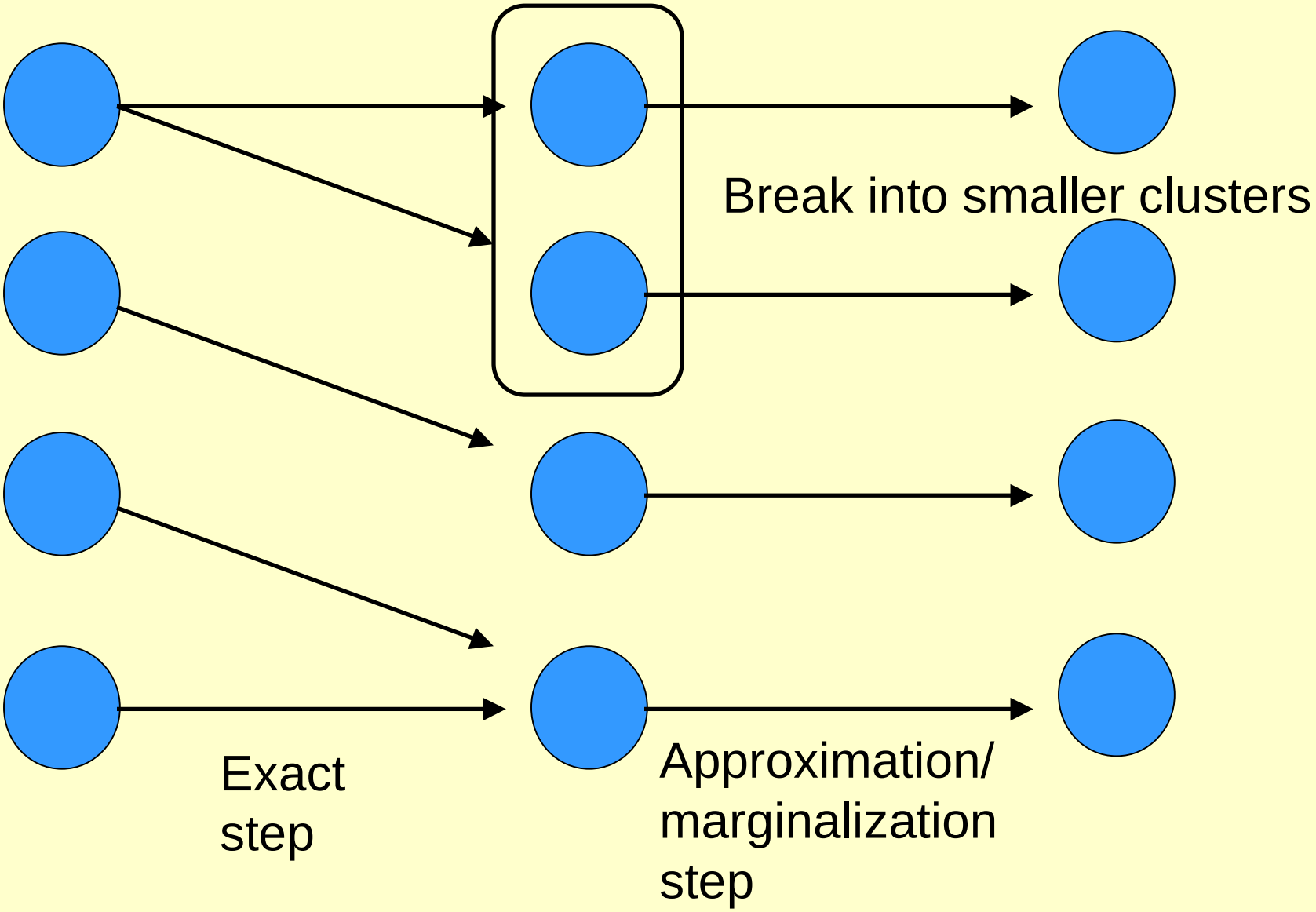


	$P(Z_{t+1})$	$P(\overline{Z}_{t+1})$
$Y_t Z_t$	$P(Z_{t+1} \mid Y_t Z_t)$	$P(\overline{Z}_{t+1} \mid Y_t Z_t)$
$Y_t \overline{Z}_t$	$P(Z_{t+1} \mid Y_t \overline{Z}_t)$	$P(\overline{Z}_{t+1} \mid Y_t \overline{Z}_t)$
$\overline{Y}_t Z_t$	$P(Z_{t+1} \mid \overline{Y}_t Z_t)$	$P(\overline{Z}_{t+1} \mid \overline{Y}_t Z_t)$
$\overline{Y}_t \overline{Z}_t$	$P(Z_{t+1} \mid \overline{Y}_t \overline{Z}_t)$	$P(\overline{Z}_{t+1} \mid \overline{Y}_t \overline{Z}_t)$

Problem: Variable Correlation



Solution: BK algorithm

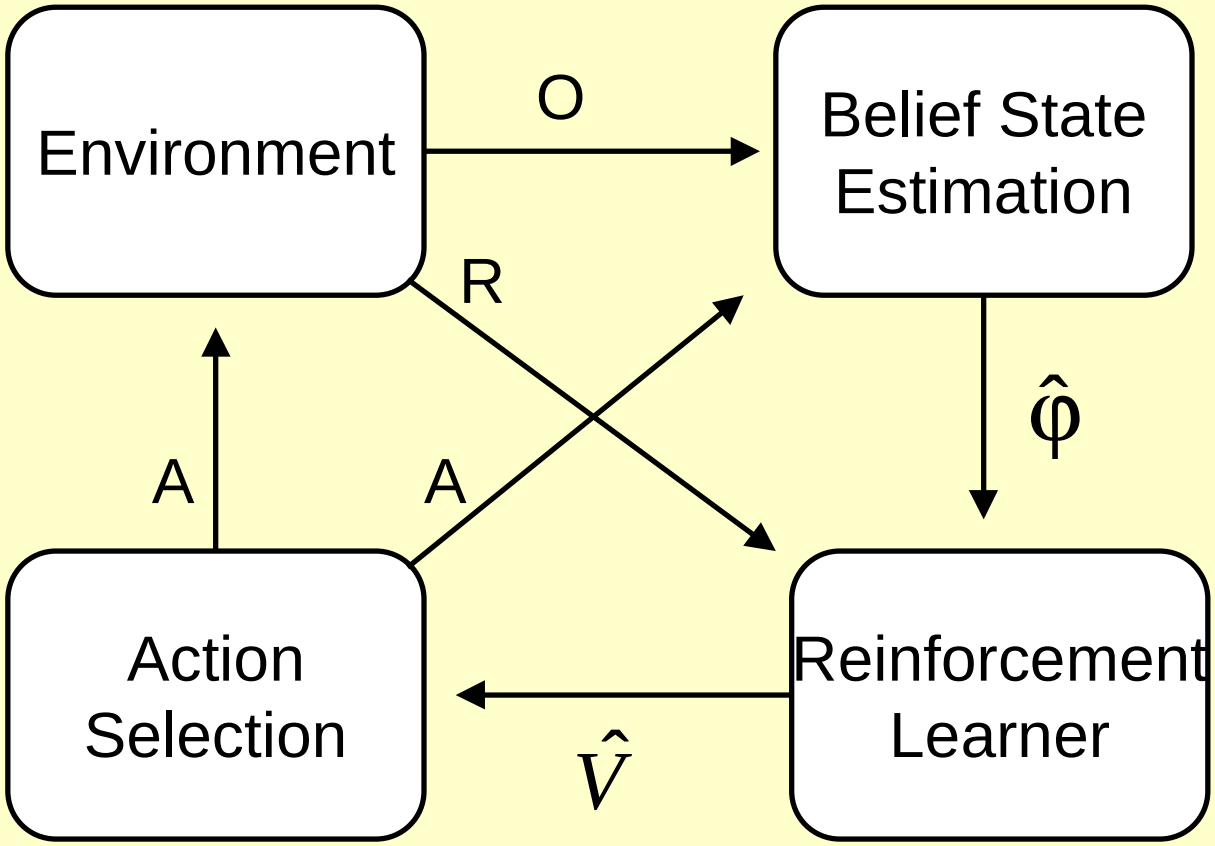


With mixing, bounded projection error: total error is bounded

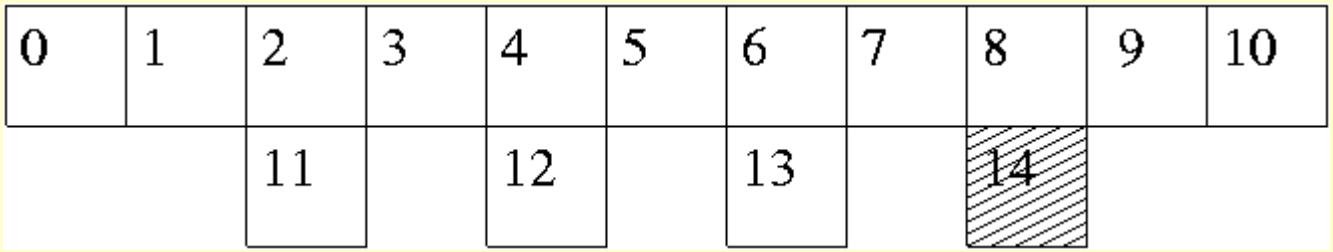
Density Estimation meets POMDPs

- Problems:
  - Representing state
  - Representing value function
- Solution:
  - Use BK algorithm for state estimation
  - Use reinforcement learning for V
    (e.g. Parr & Russell 95, Littman et al. 95)
  - Represent V with neural net
- Rodriguez, Parr and Koller, NIPS 99

Approximate POMDP RL

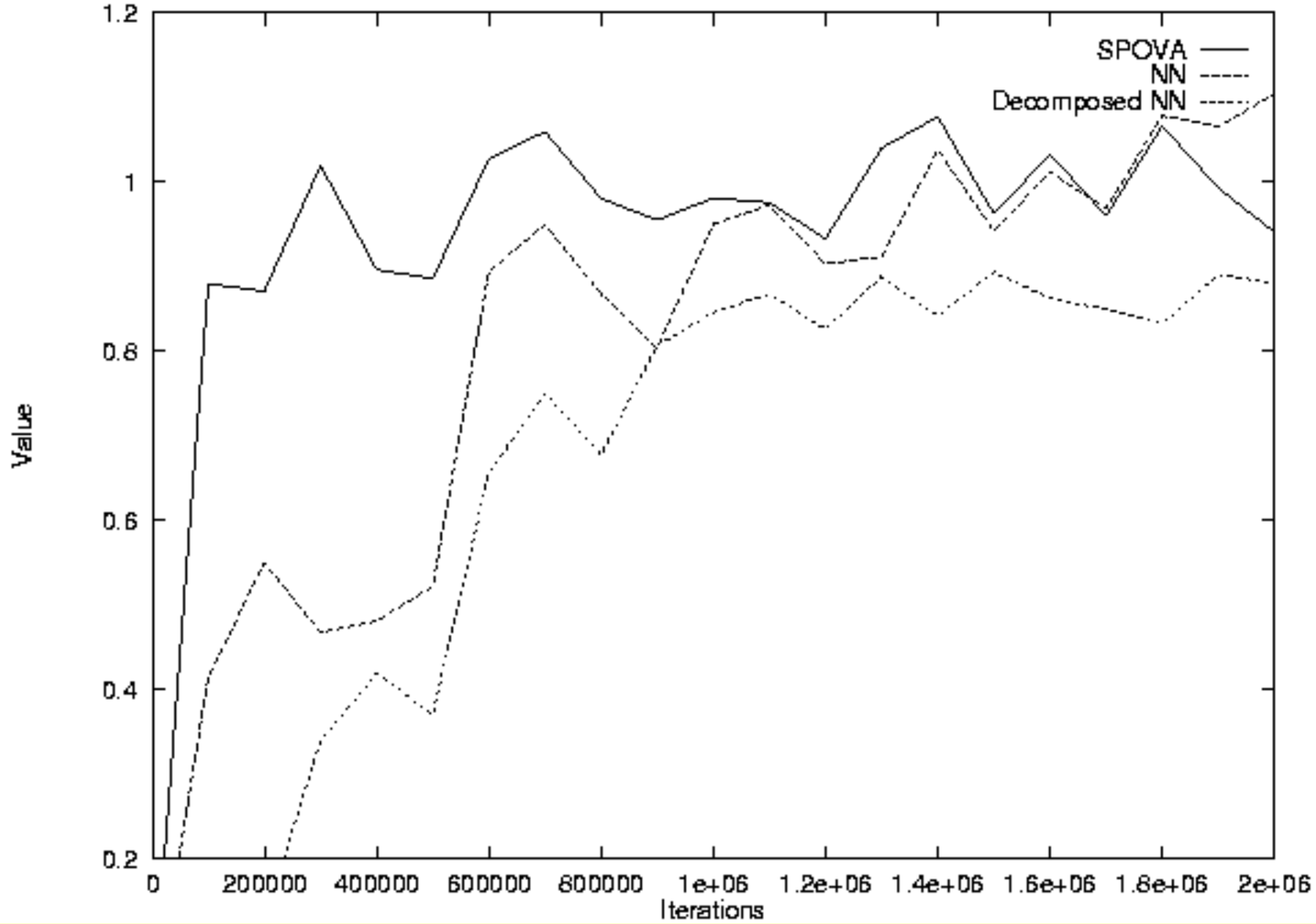


Navigation Problem

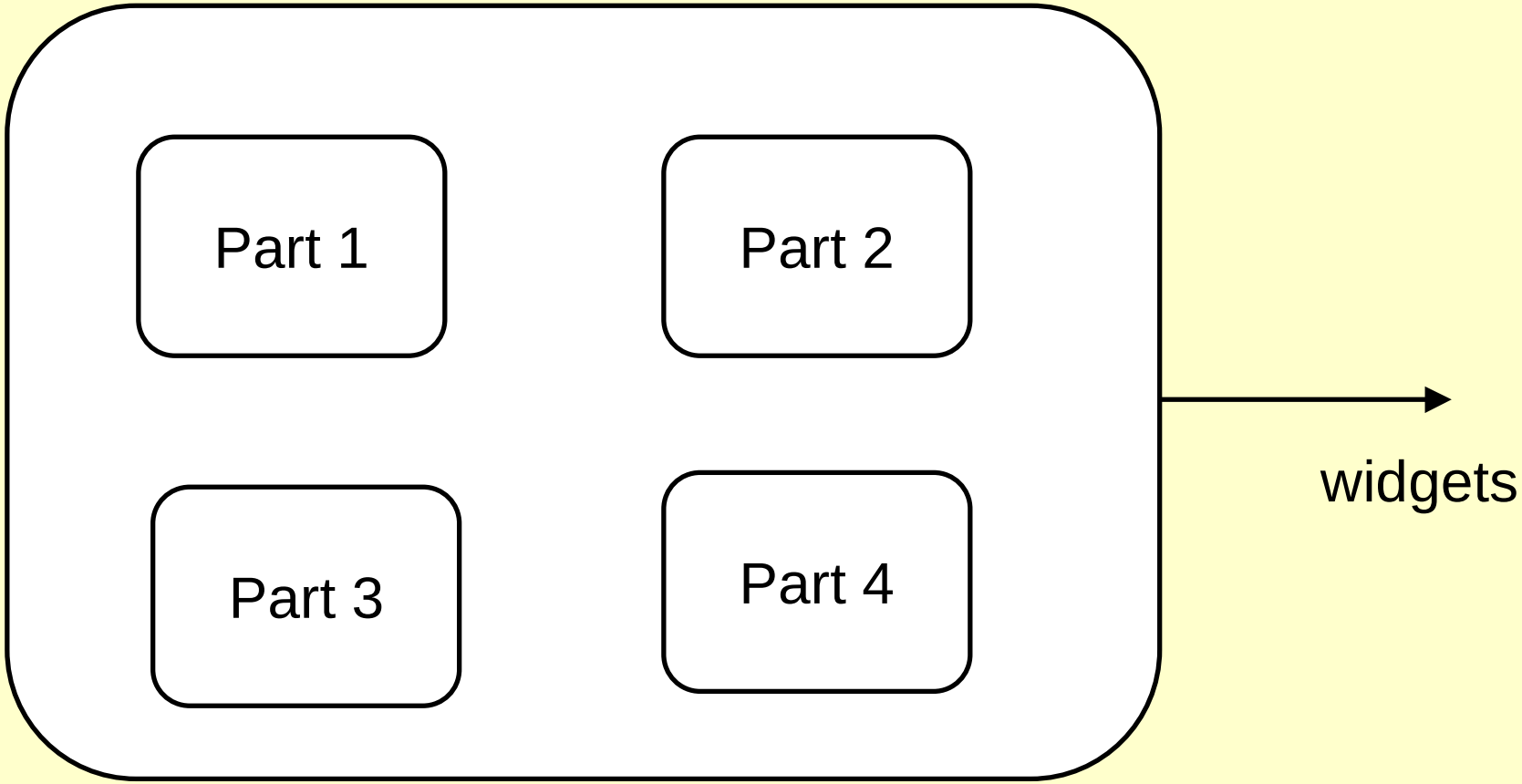


- Uncertain initial location
- 4-way sonar
- Need for information gathering actions
- 60 states (15 positions x 4 orientations)

Navigation Results

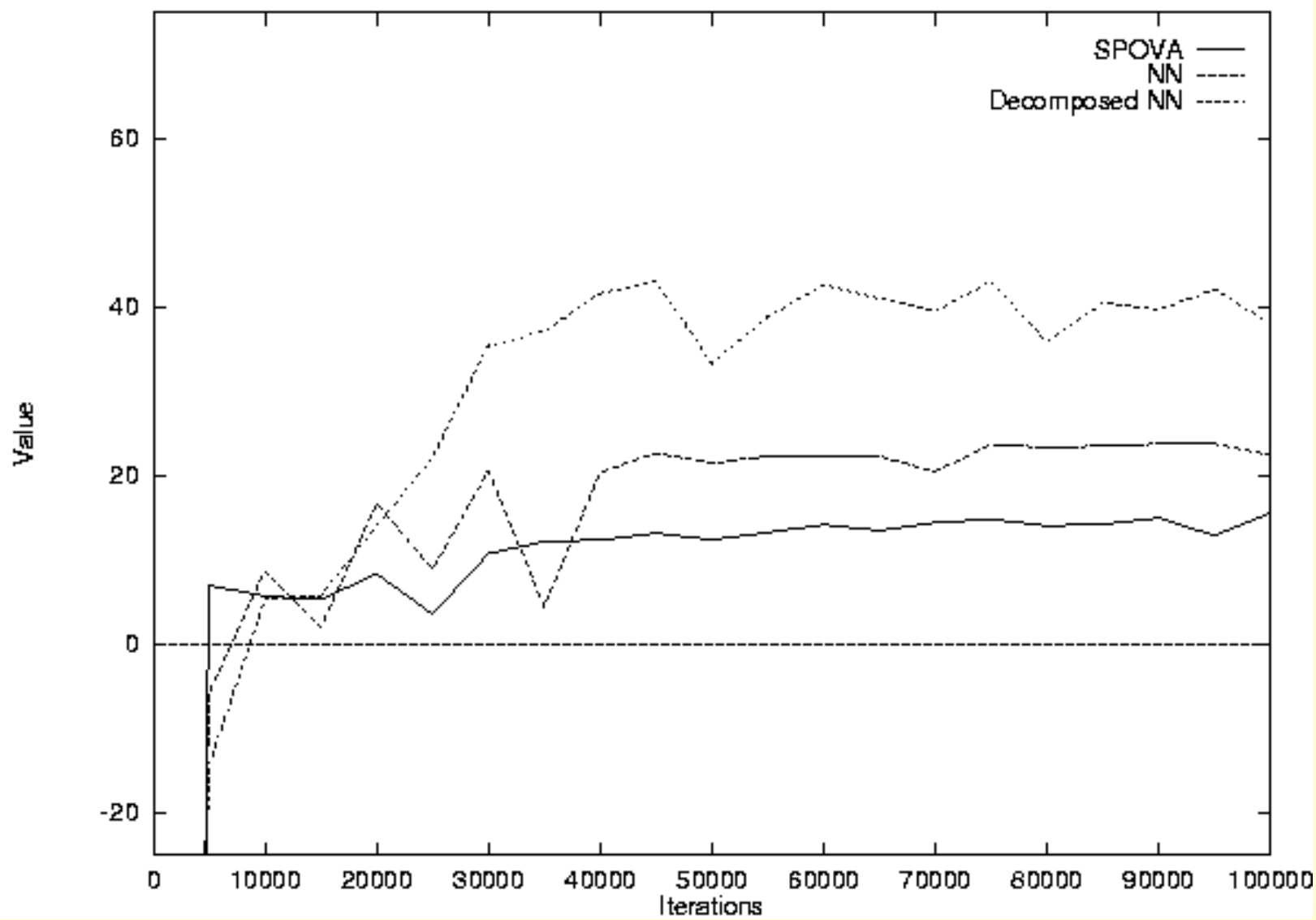


Machine Maintenance

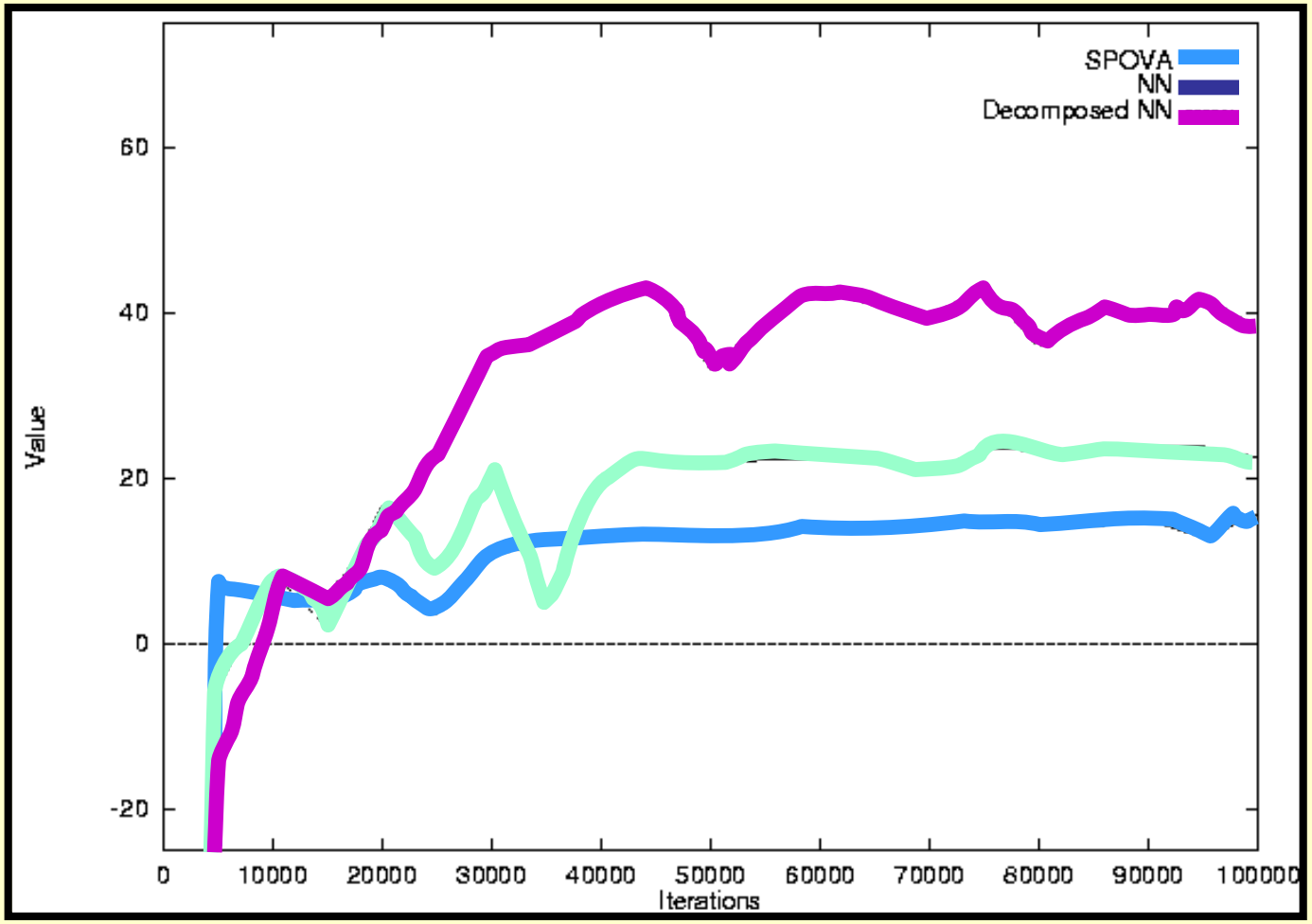


4 machine maintenance states per machine
Reward for output
Components degrade, reducing output
Repair requires expensive total disassembly

Maintenance Results



Maintenance Results (Turnerized)



Decomposed NN has fewer inputs, learns faster

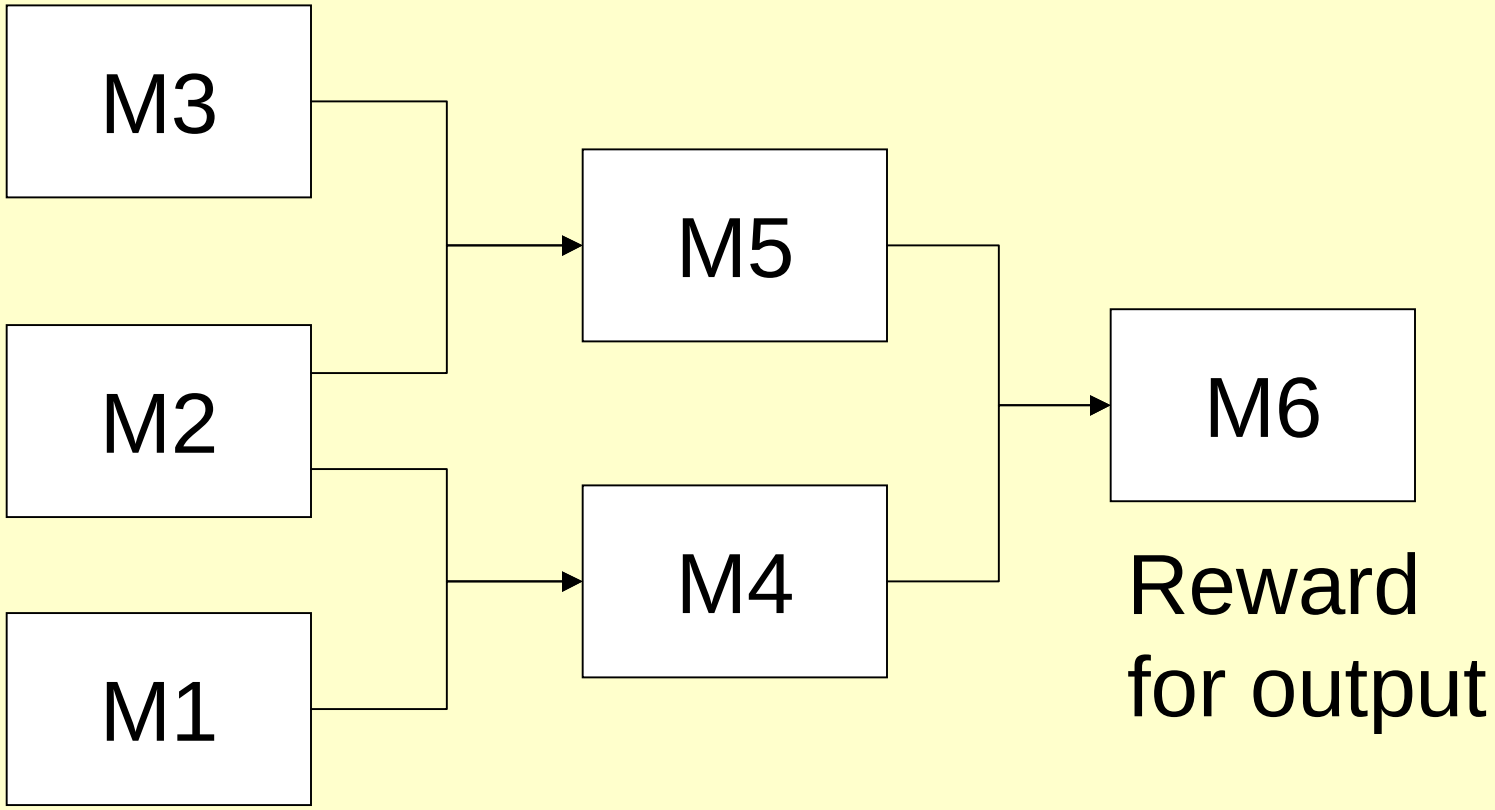
Summary

- Advances
  - Use of factored belief state
  - Scales POMDP RL to larger state spaces
- Limitations
  - No help with regular MDPs
  - Can be slow
  - No convergence guarantees

Goal: DP with guarantees

- Focus on value determination in MDPs
- Efficient exact DP step
- Efficient projection (function approximation)
- Non-expansive function approximation (convergence, bounded error)

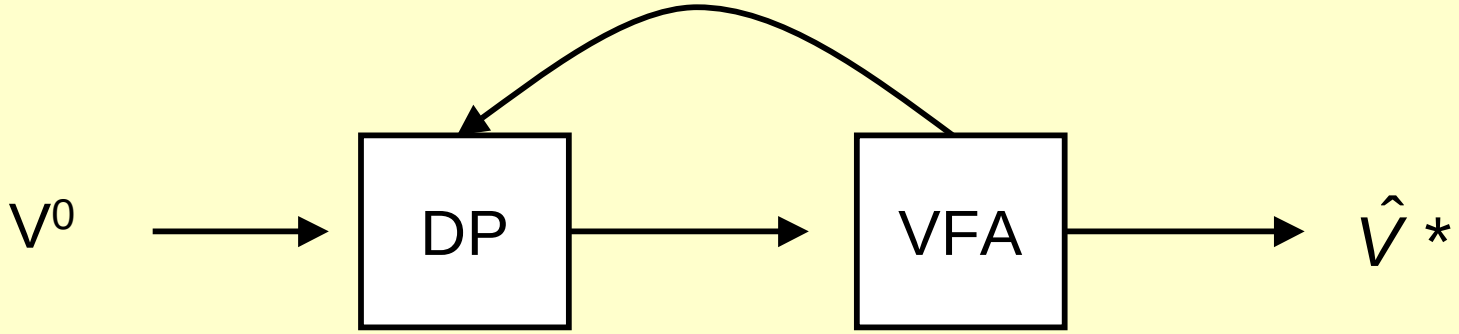
A Value Determination Problem



Machines require predecessors to work
They go offline/online stochastically

Efficient, Stable DP

Idea: Restrict class of value functions



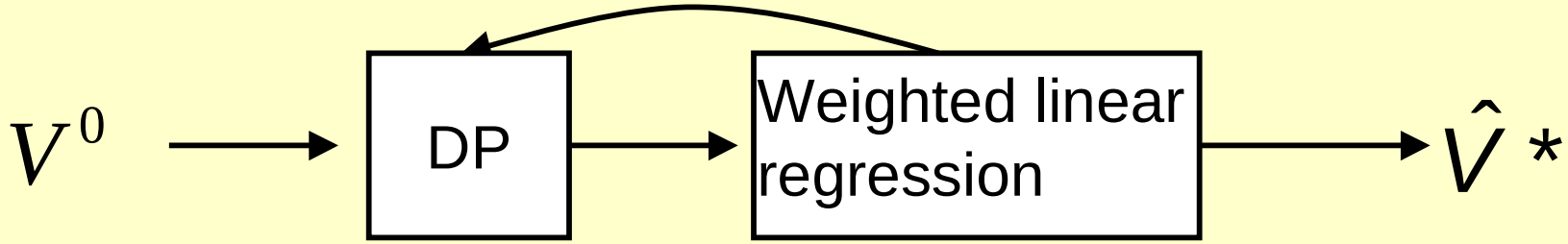
VFA: Neural Network, Regression, etc.

Issues: Stability, Closeness of $\hat{V}^*$ to V*, efficiency

Stability

- Naïve function approximation is unstable
  [Boyan & Moore 95, Bertsekas & Tsitsiklis 96]
- Simple examples where $V \rightarrow \infty$
- *Weighted* linear regression is stable
  [Nelson 1958, Van Roy 1998]
- Weights must correspond to stationary distribution of policy: ρ

Stable Approximate DP



error in final result

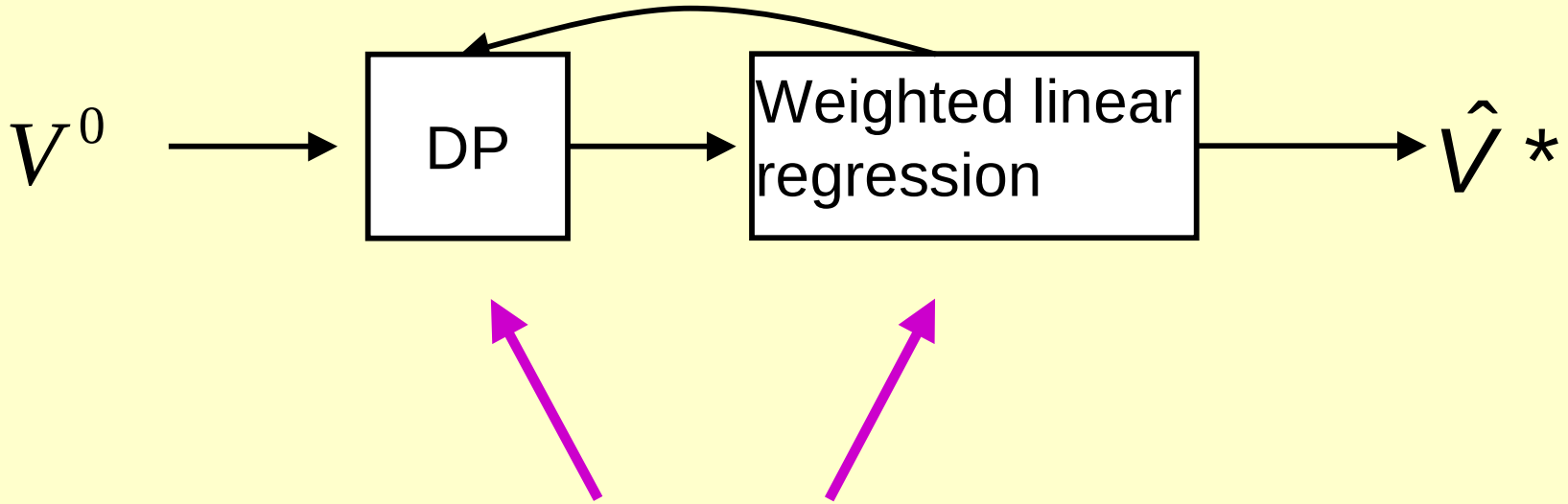
lowest error possible

$$d_\rho(V^*, \hat{V}^*) \leq \frac{1}{\sqrt{1-\kappa^2}} d_\rho(V^*, \Pi_\rho V^*)$$

κ = effective contraction rate

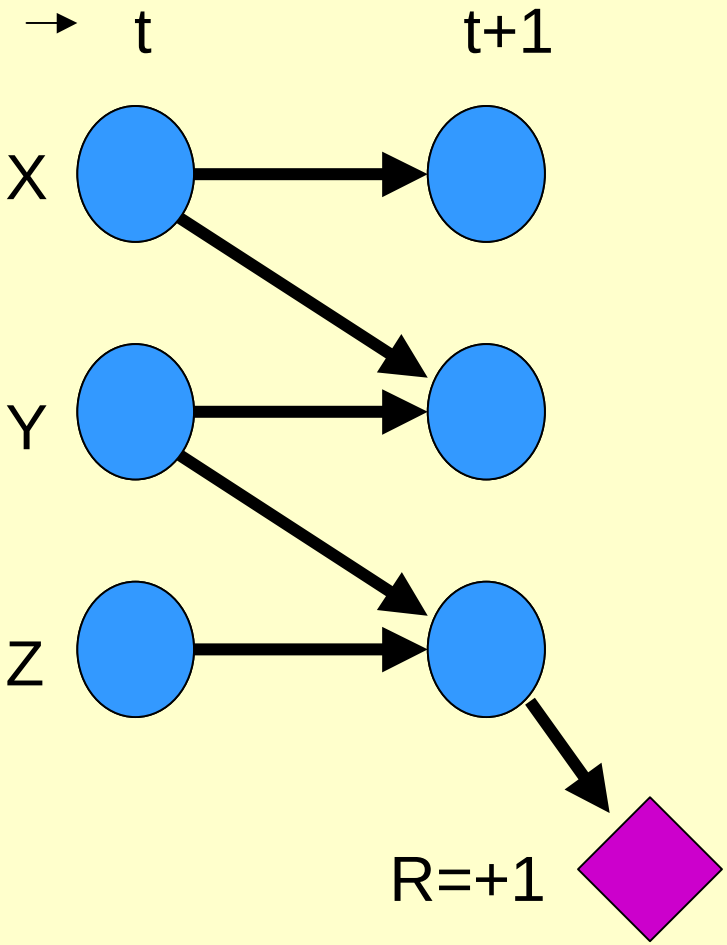
Efficiency Issues

DP, projection consider every state individually

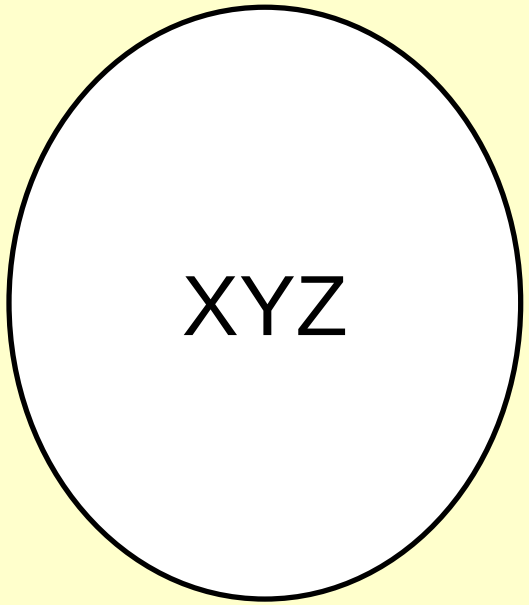


Must do these steps efficiently!!!

Compact Models = Compact V*?



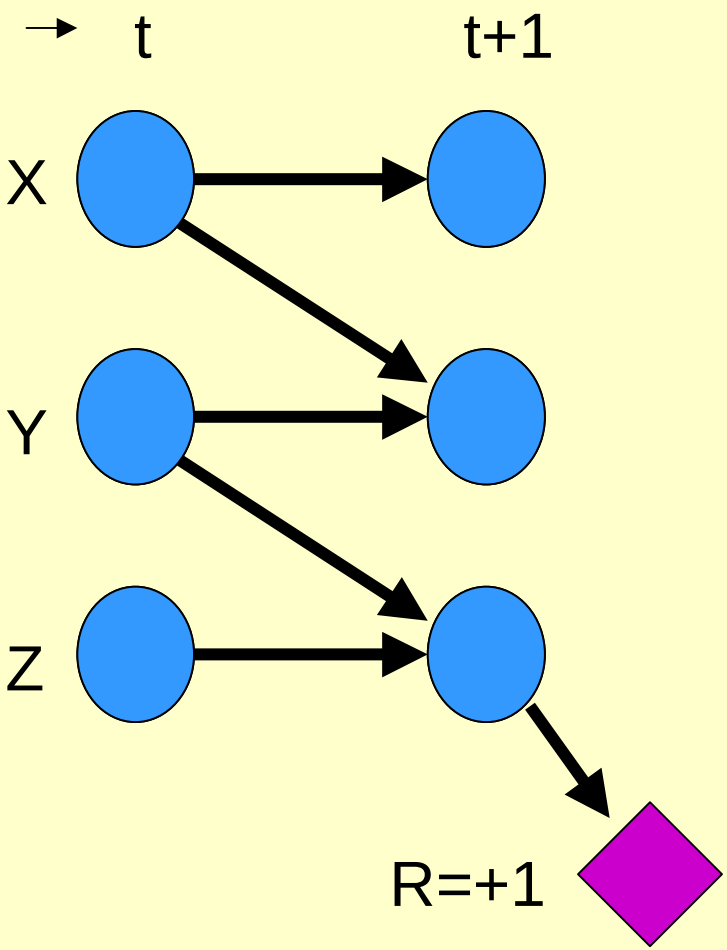
Suppose R = 1 if Z = T

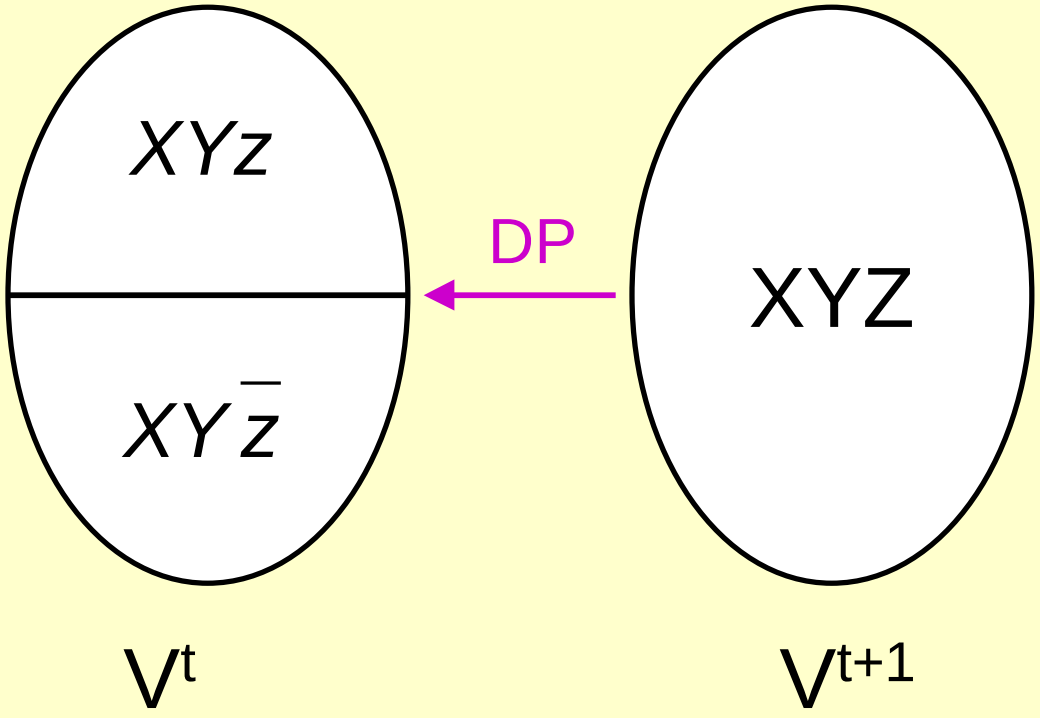


V^{t+1}

Start with a uniform value function

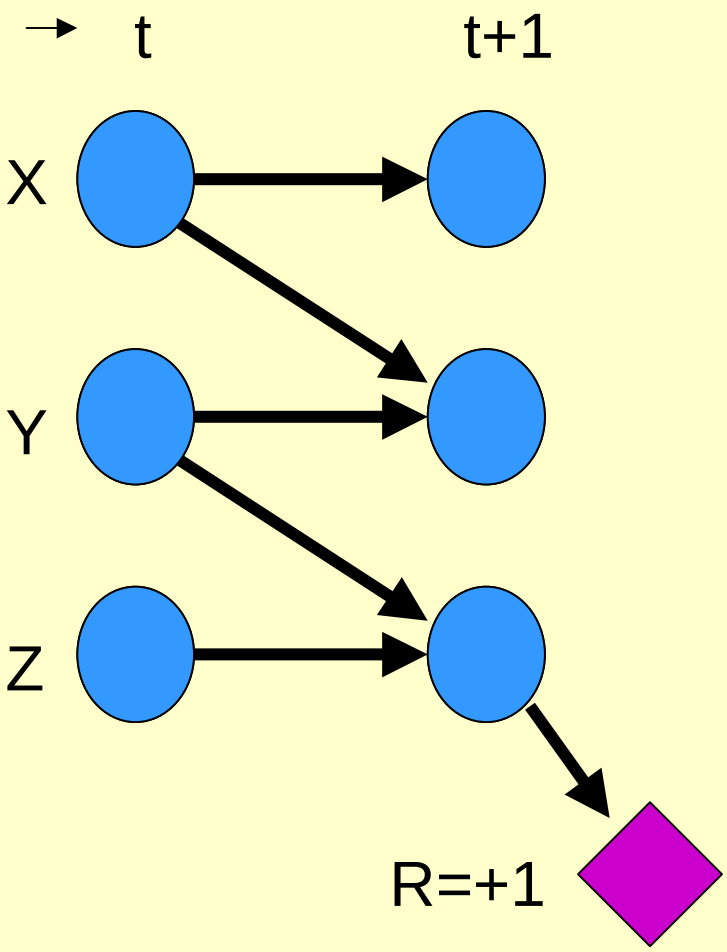
Value Function Growth

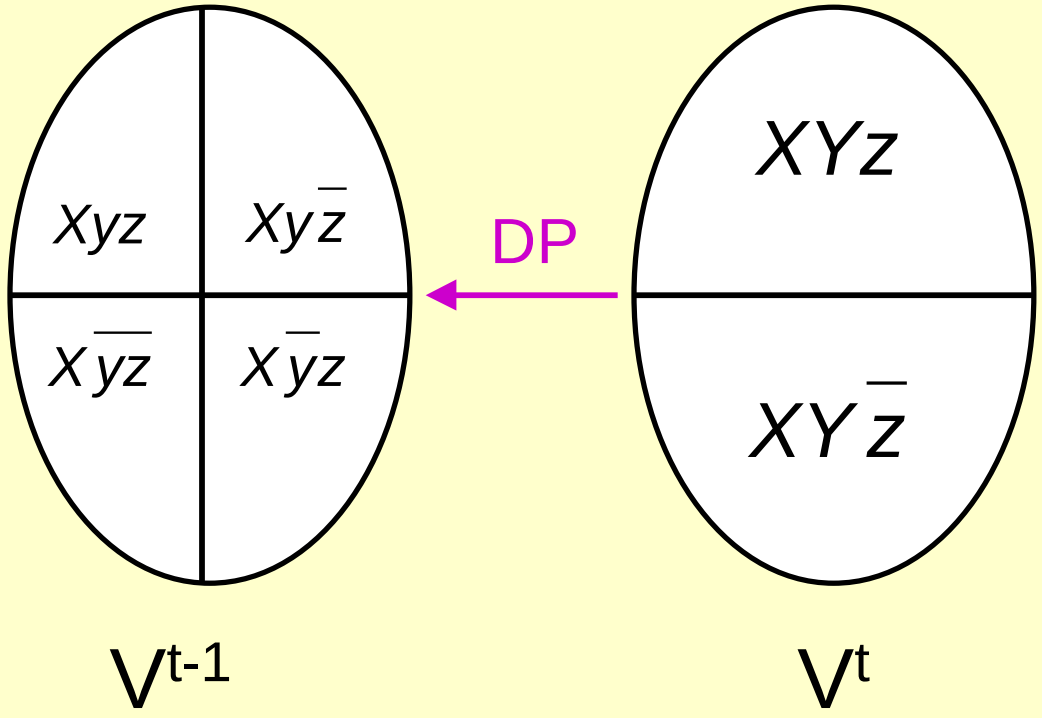




Reward depends upon Z

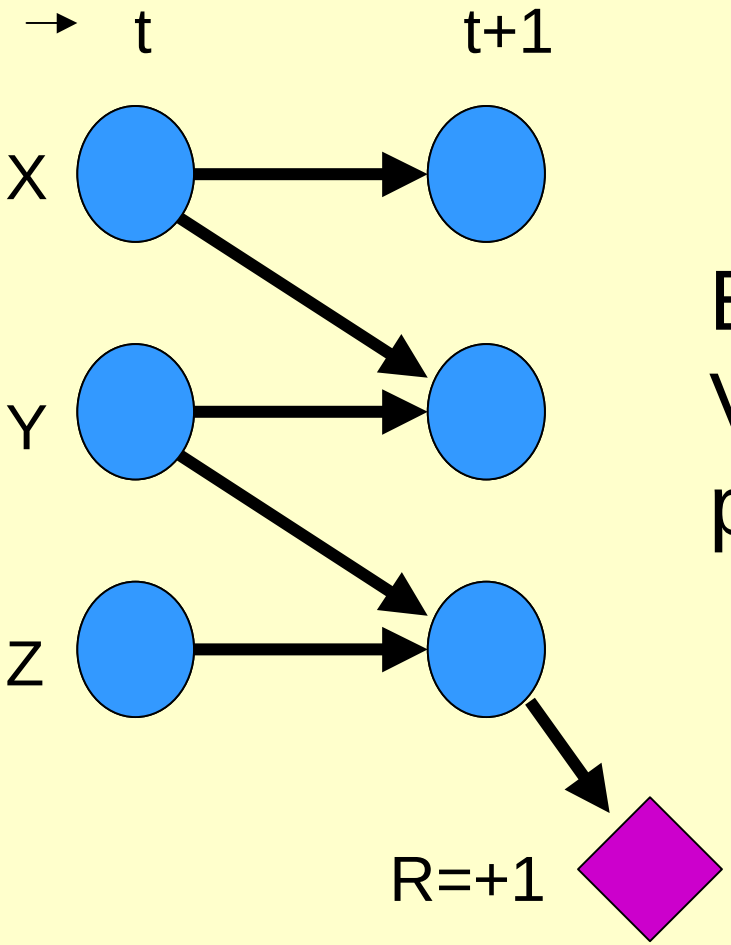
Value Function Growth





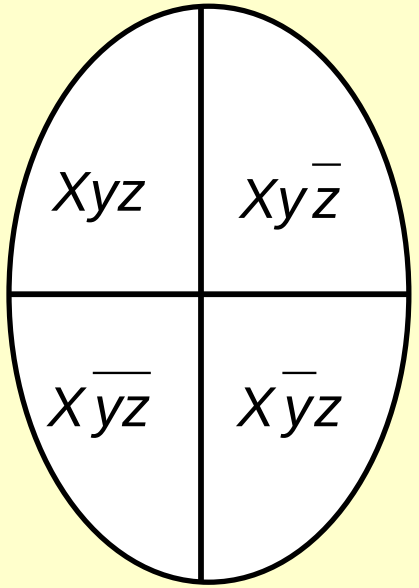
Z depends upon previous Y and Z

Value Function Growth



Eventually, V^* has 2^n partitions

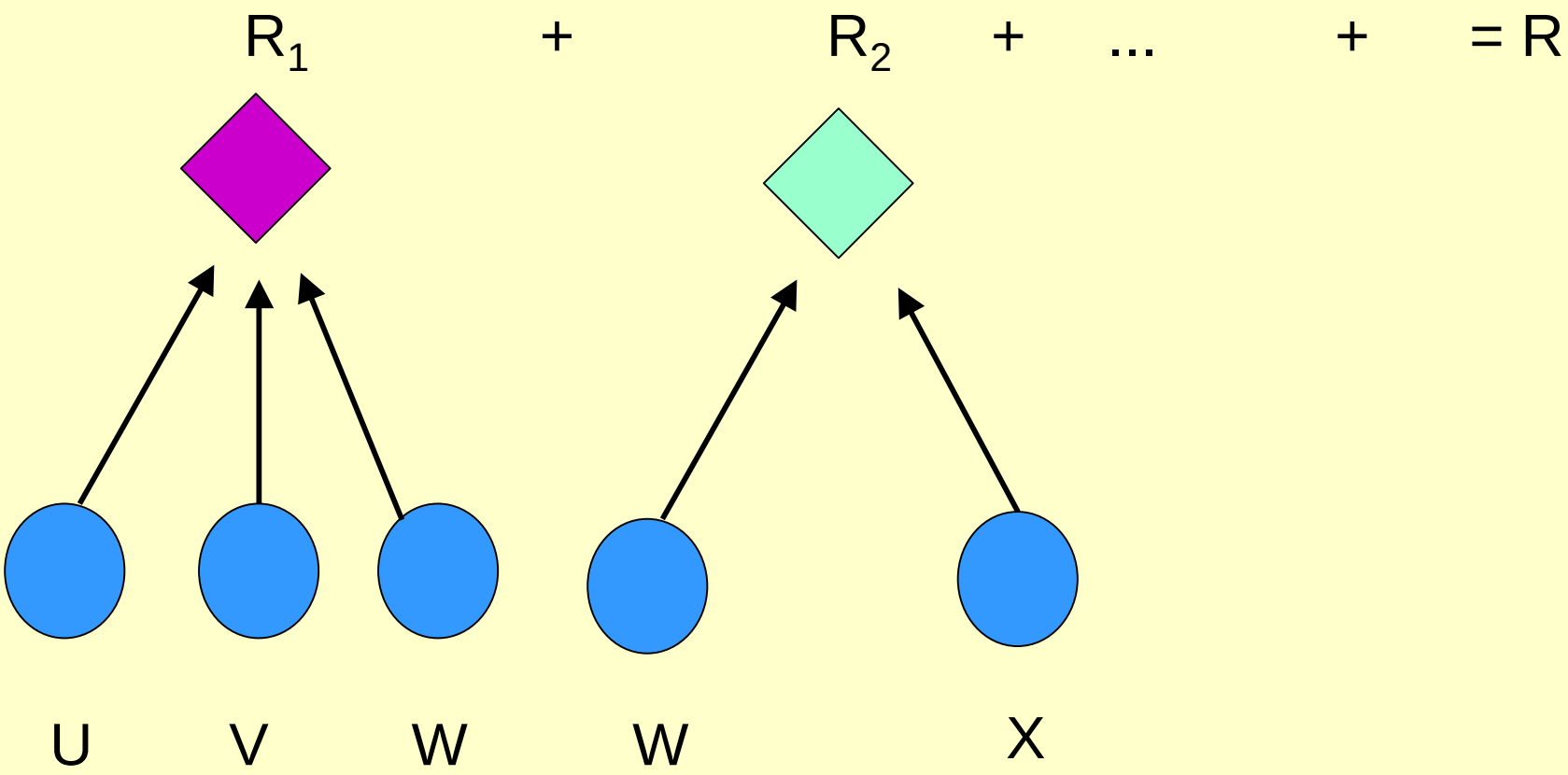
DP



V^{t-1}

See Boutilier, Dearden & Goldszmidt (IJCAI 95) for method that avoids worst case when possible.

Compact Reward Functions



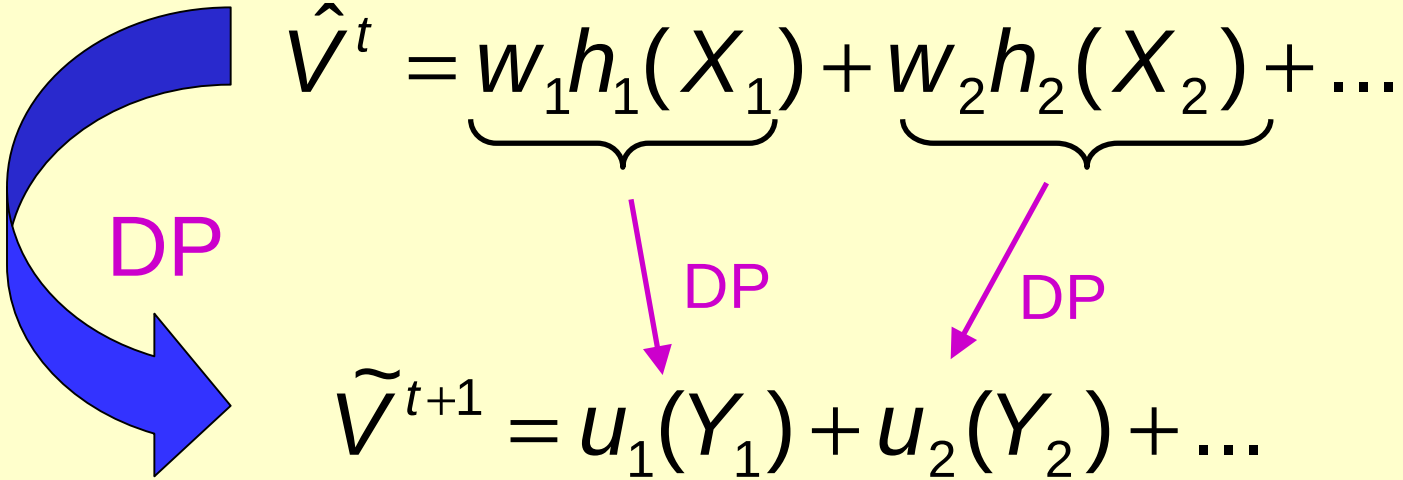
Basis Functions

- $V = w_1 h_1(\mathbf{X}_1) + w_2 h_2(\mathbf{X}_2) + \ldots$
- Use compact basis functions
- $h(\mathbf{X}_i)$ basis defined over vars in $\mathbf{X}_i$

Examples: h = function of status of subgoals
h = function of inventory in different stores
h = function of status of machines in factory

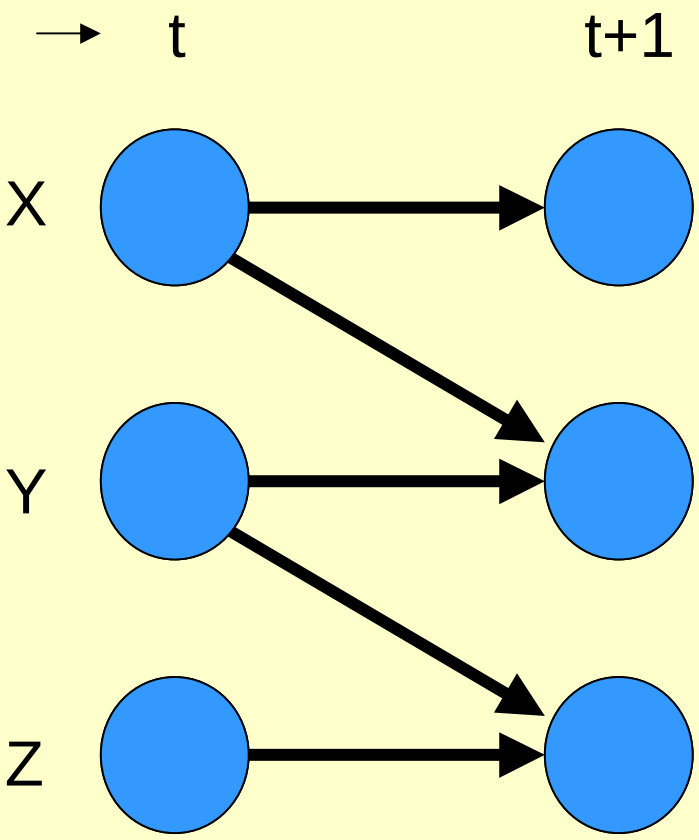
Efficient DP

Observe that DP is a linear operation:



$Y_1 = X_1 \cup \text{parents}(X_1)$

Growth of Basis Functions



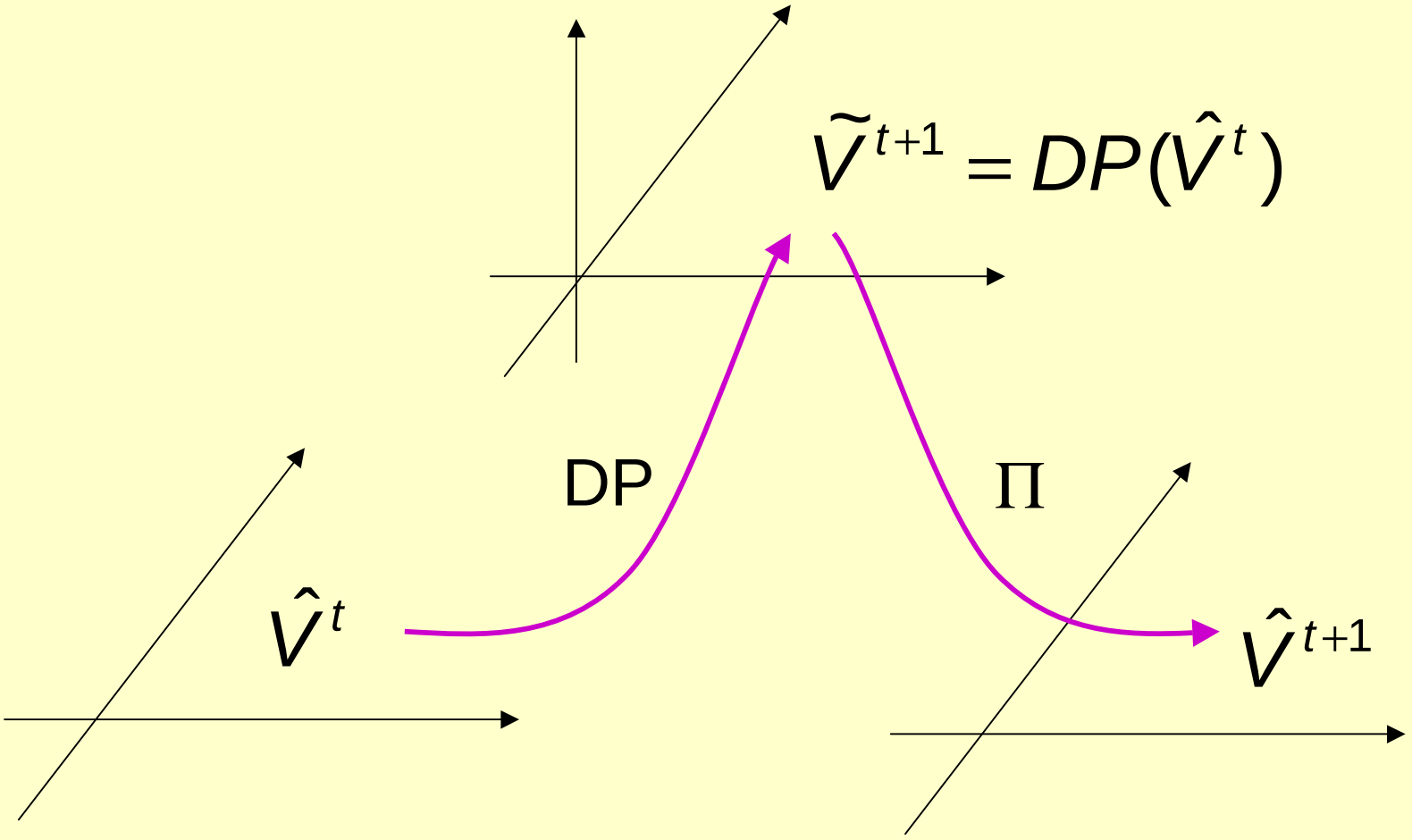
Suppose h_1=f(Y)

DP(h_1) = f(X,Y)

Each basis function is replaced by a function with a potentially larger domain

Need to control growth in function domains

Projection



Regression projects back into original space

Efficient Projection

Want to project all points:

$$W^{t+1} = (A^T A)^{-1} A^T \tilde{V}^{t+1}$$

K basis functions

$h_1(s1)$ $h_2(s1)$...
$h_1(s2)$ $h_2(s2)$…
.
.
.

2^n states

Projection matrix $(A^T A)^{-1}$ is k x k

Efficient dot product

Need to compute: $\sum_s h_i(s)h_j(s)$

Observe: no. of unique terms in summation is product of no. of unique terms in bases: $\#|\mathbf{X}_i| \times \#|\mathbf{X}_j|$

$$\sum_s h_i(s)h_j(s) = c\sum_{x_i,x_j} h_i(x_i)h_j(x_j)$$

Complexity of dot product is $O(\#|\mathbf{X}_i| \times \#|\mathbf{X}_j|)$

Compute $A^T\tilde{V}^{t+1}$ using same observation

Want: *Weighted* Projection

- Stability required *weighted* regression
- But, stationary dist. ρ may not be compact
- Boyen-Koller Approximation [UAI 98]
- Provides factored $\hat{\rho}$ with bounded error
- Dot product $\rightarrow \hat{\rho}$ weighted dot product

Weighted dot products

Need to compute: $\sum_s h_i(s)h_j(s)\rho(s)$

If ρ is factored, and basis functions are compact:

Let $Y = Clusters(Xi \cup Xj)$ i.e. all vars. in the enclosing BK clusters

$$\sum_s h_i(s)h_j(s)\rho(s) = \sum_{y \in Y} h_i(y)h_j(y)\rho(y)$$

Stability

Idea: If error in $\hat{\rho}$ not "too large", then we're OK.

Theorem: If

$$(1-\varepsilon)\hat{\rho}(x) \le \rho^*(x) \le (1+\varepsilon)\hat{\rho}(x)$$

and

$$g = k\frac{(1+\varepsilon)}{(1-\varepsilon)} < 1$$

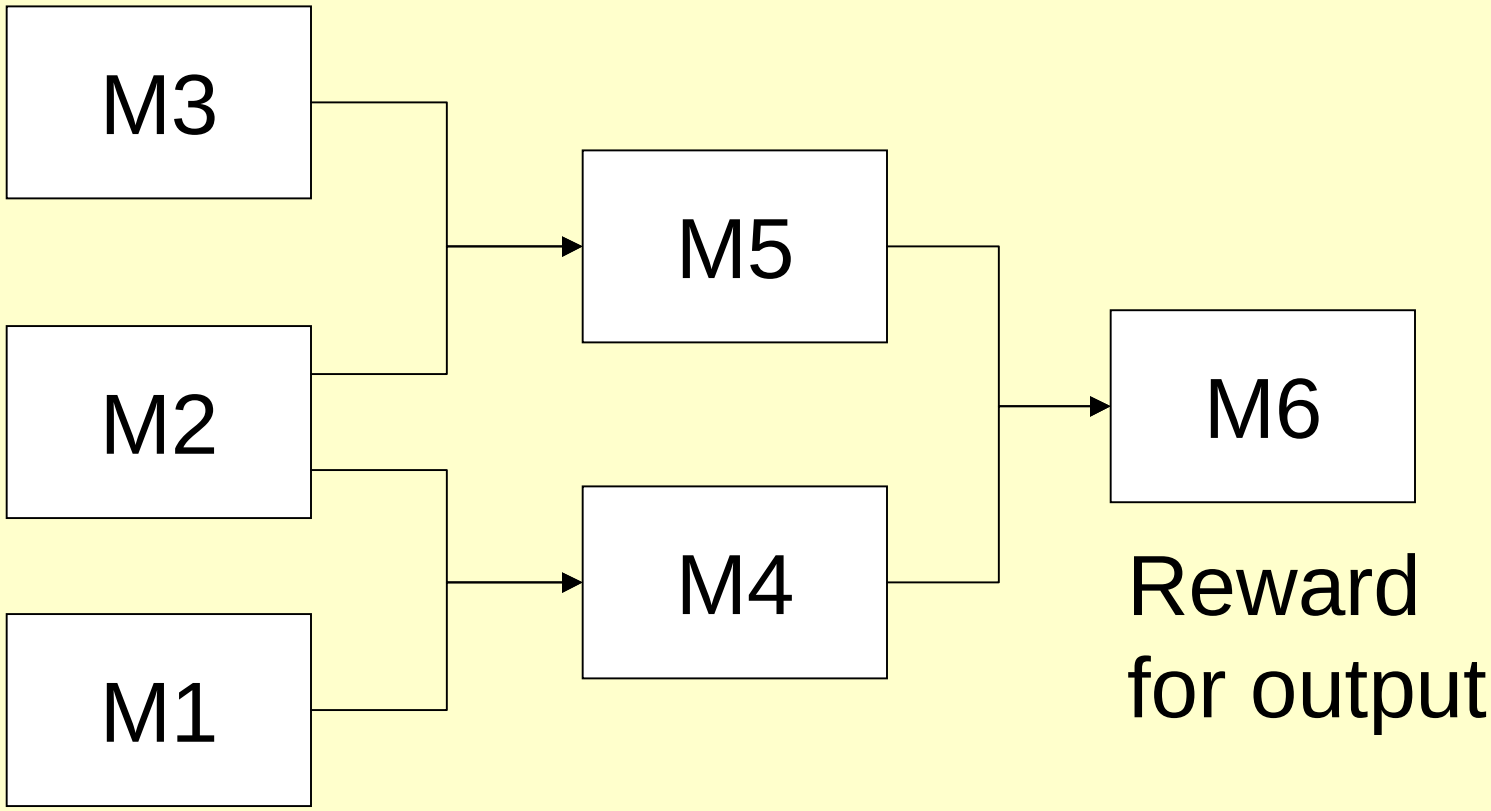
then

$$d_\rho(V^*, \hat{V}^*) \le \frac{1}{\sqrt{1-g^2}} d(V^*, \Pi_\rho V^*)$$

Approximate DP summary

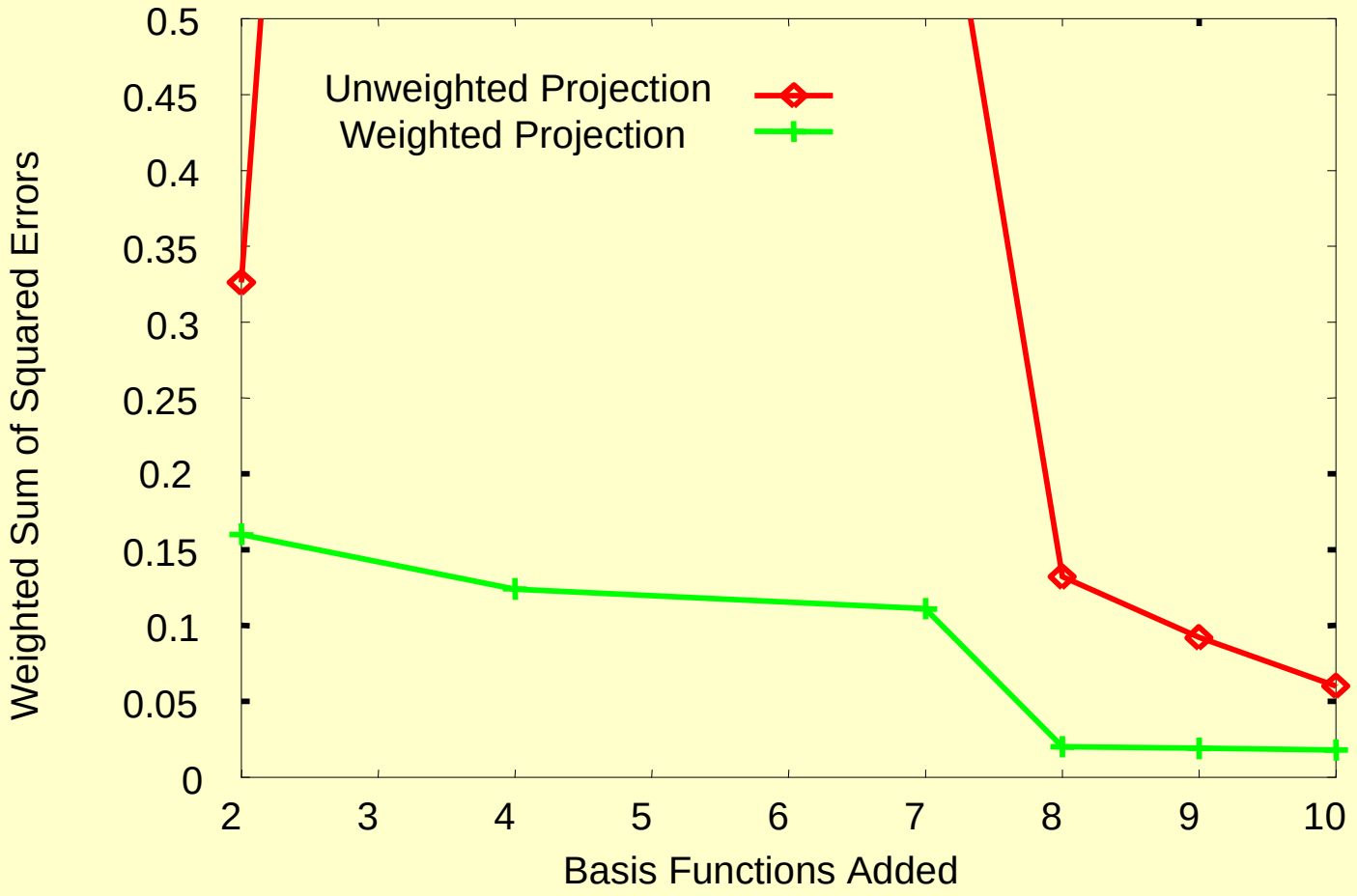
- Get compact, approx. stationary distribution
- Start with linear value function
- Repeat until convergence:
  - Exact DP replaces bases with larger fns.
  - Project value function back into linear space
- Efficient because of
  - Factored transition model
  - Compact basis functions
  - Compact approx. stationary distribution

Sample Revisited

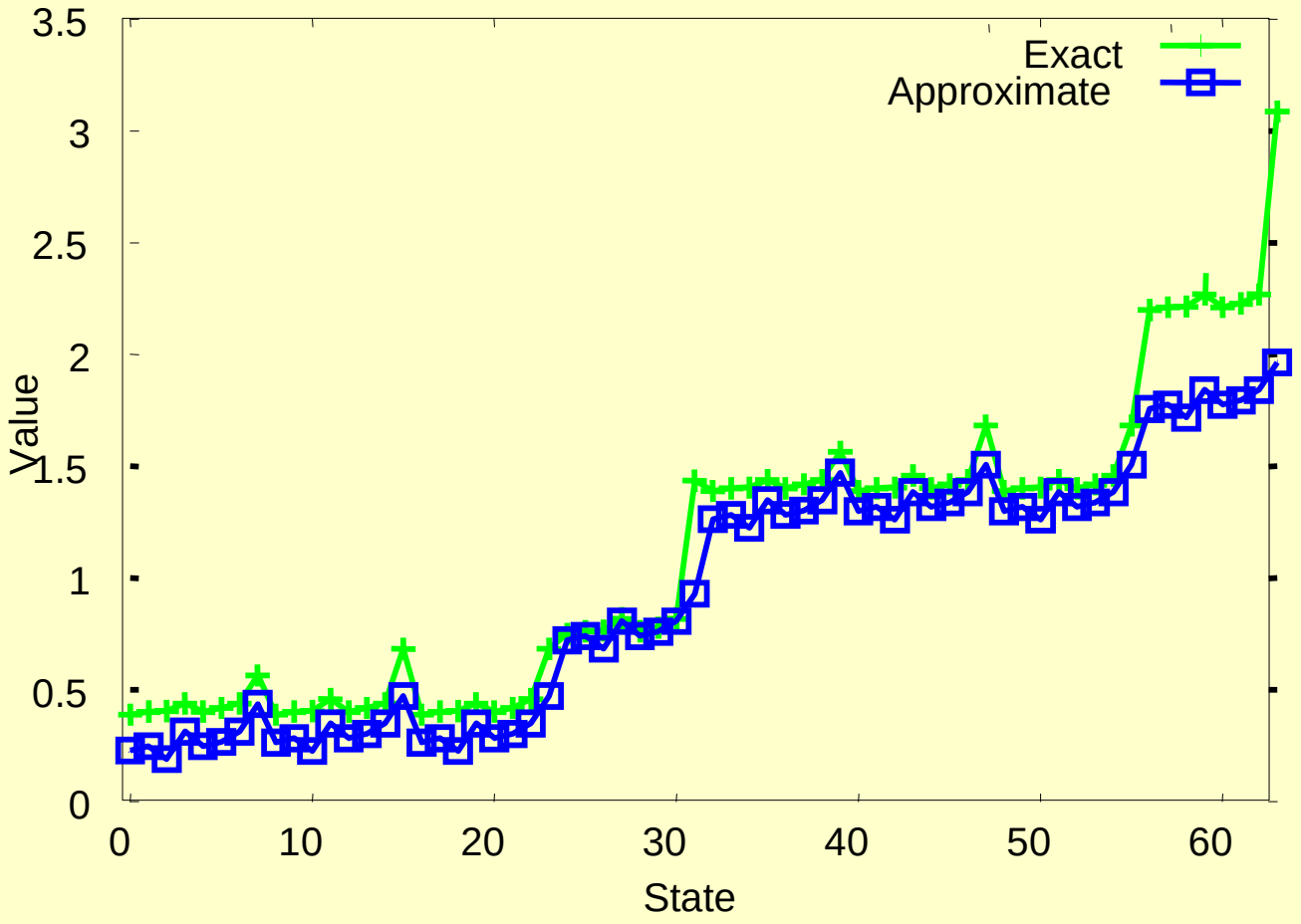


Machines require predecessors to work,
Fail stochastically

Results: Stability and Weighted Projection



Approximate vs. Exact V



Summary

- Advances
  - Stable, approximate DP for large models
  - Efficient DP, projection steps
- Limitations
  - Prediction only, no policy improvement
  - non-trivial to add policy improvement
  - Policy representation may grow

Direct Policy Search

Idea: Search smoothly parameterized policies

Policy function: $\pi(s, \theta)$

Value function (wrt starting dist.): $V(\theta)$

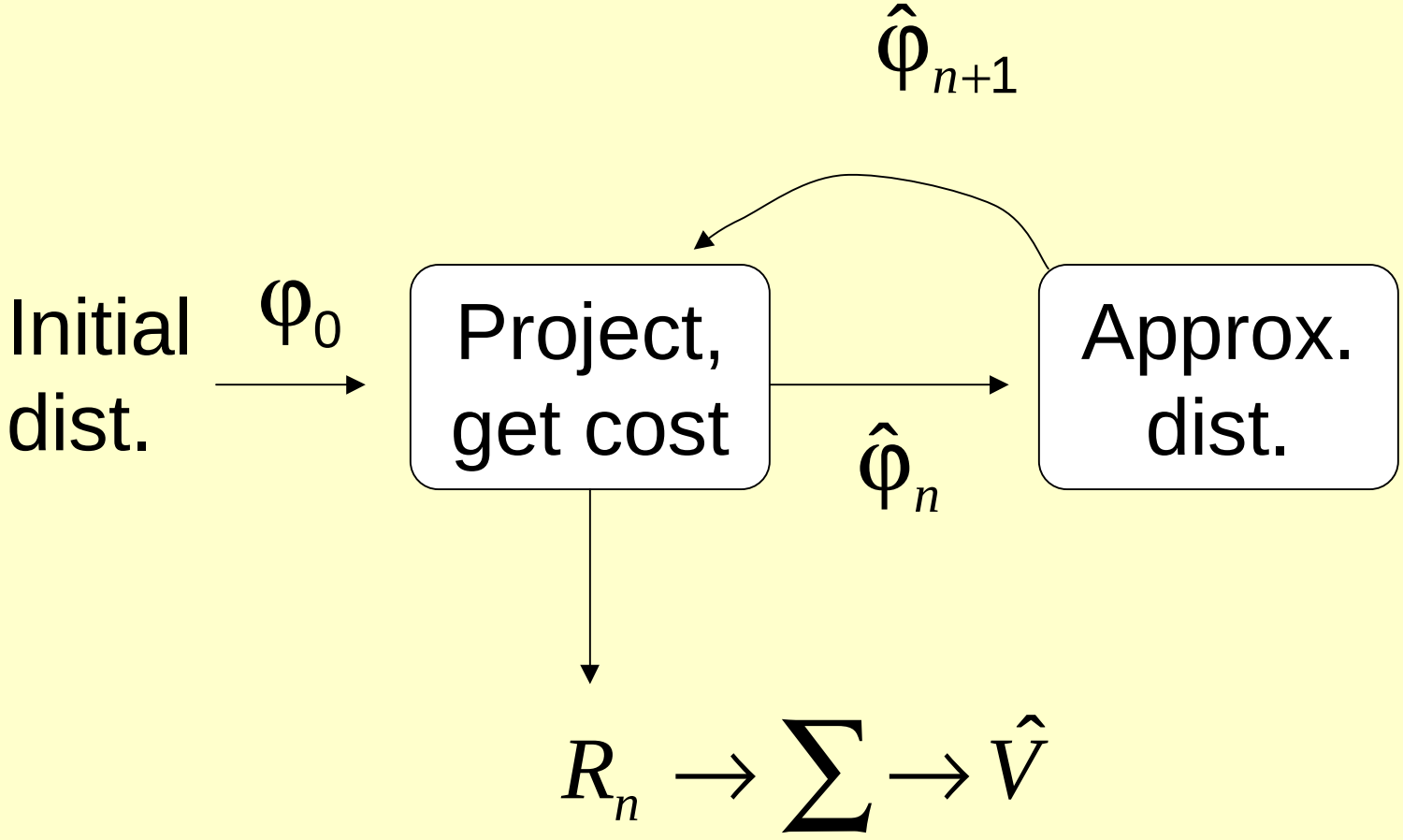
See: Williams 83, Marbach & Tsitskilis 98, Baird & Moore 99,
Meauleau et al. 99, Peshkin et al. 99,
Konda & Tsitsiklis 00, Sutton et al. 00

Policy Search with Density Estimation

- Typically compute value gradient
- Works for both MDPs and POMPDs
- Gradient computation methods
  - Single trajectories
  - Exact (small models)
  - Value function
- Our approach:
  - Take all trajectories simultaneously
  - Ng, Parr & Koller NIPS 99

Policy Evaluation

Idea: Model rollout



Rollout Based Policy Search

Idea: Estimate $\hat{V}(\theta)$
Search θ space
e.g. using simplex search

Theorem:

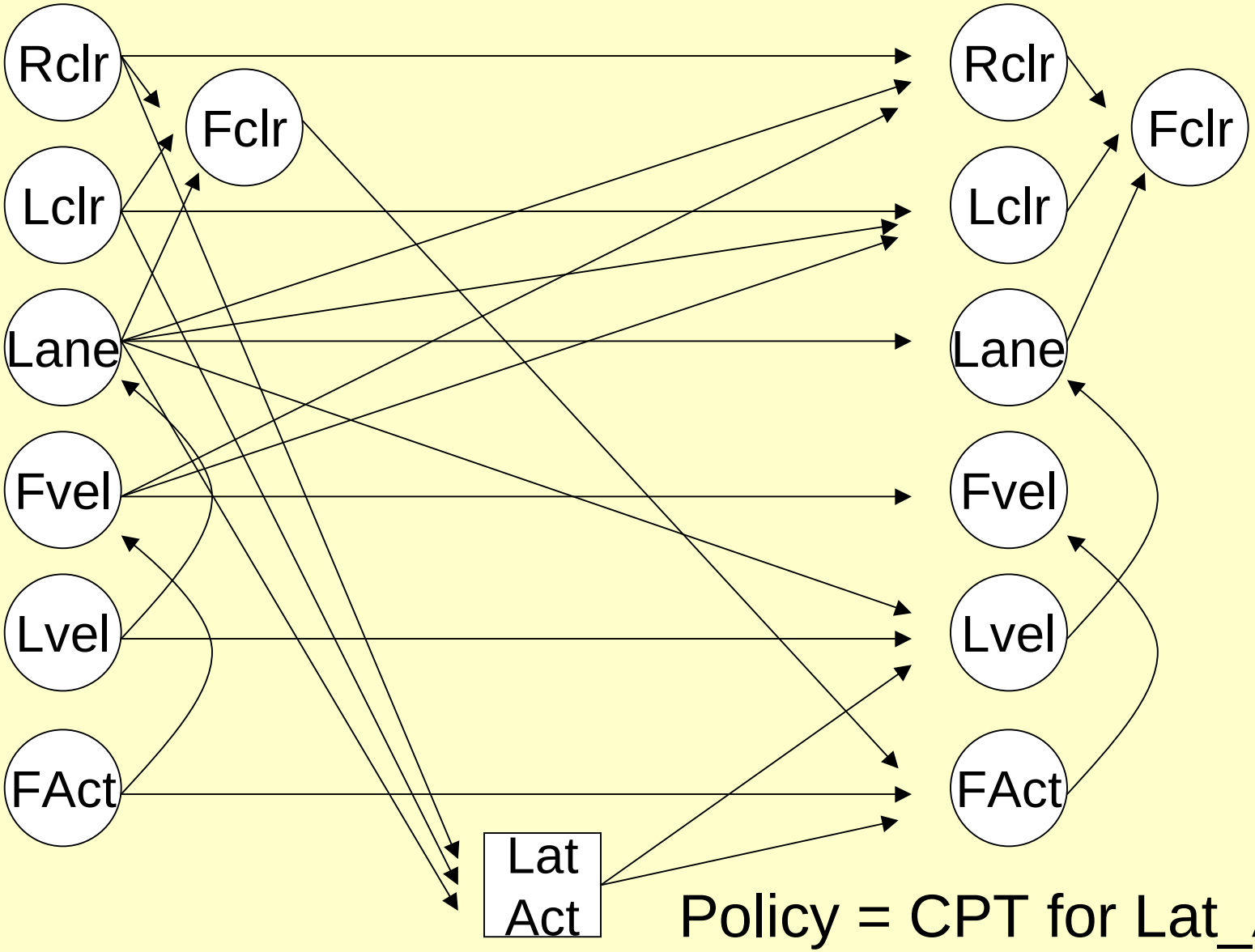
Suppose: $|V(\theta) - \hat{V}(\theta)|_\infty \leq \varepsilon$

Optimize to reach: $\hat{\theta}^*$

$$V(\theta^*) - \hat{V}(\hat{\theta}^*) < 2\varepsilon$$

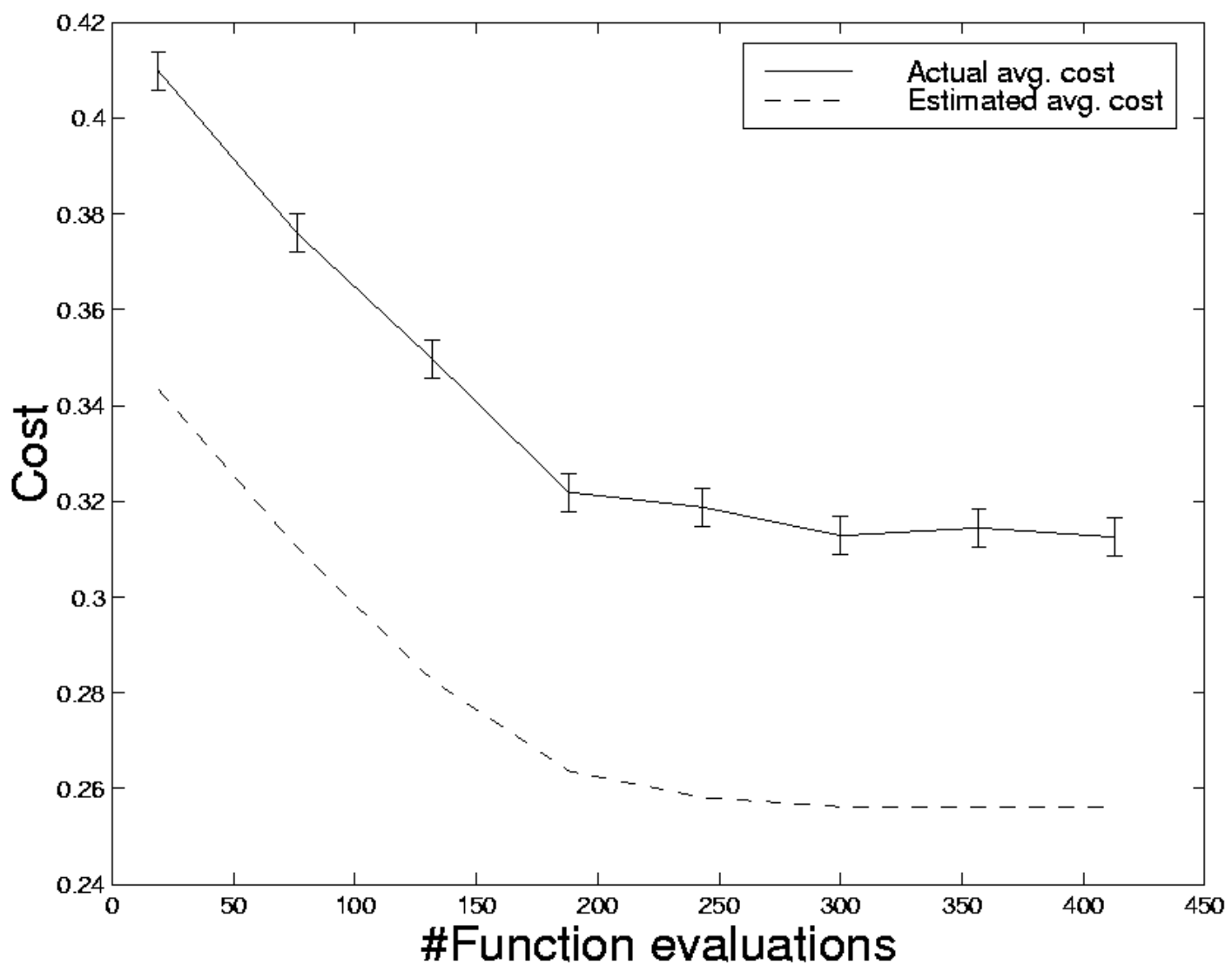
N.B.: Given density estimation, this turns policy search into simple function maximization

Simple BAT net



Policy = CPT for Lat_Act

Simplex Search Results



Gradient Ascent

Simplex is weak: better to use gradient ascent

Assume differentiable model, approximation

estimated density

$$\hat{\varphi}^{(t+1)} = \hat{\Psi}(\theta_0, \hat{\varphi}^{(t)})$$

Combined propagation/estimation operator

Apply the Chain Rule

Rollout

$$\frac{d\hat{\varphi}^{(t+1)}}{d\theta}(\theta_0) = \frac{d}{d\theta}\hat{\Psi}(\hat{\varphi}^{(t)})$$

$$= \frac{\partial \hat{\Psi}}{\partial \theta}(\theta_0, \hat{\varphi}^{(t)}) + \frac{\partial \hat{\Psi}}{\partial \hat{\varphi}}(\theta_0, \hat{\varphi}^{(t)}) \frac{d\hat{\varphi}^{(t)}}{d\theta}(\theta_0)$$

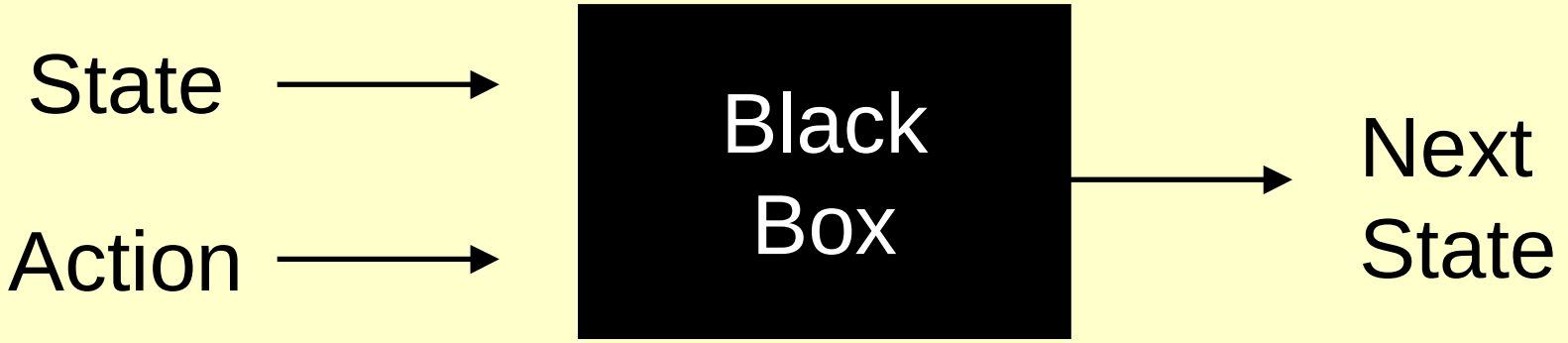
Recursive formulation

Differentiation

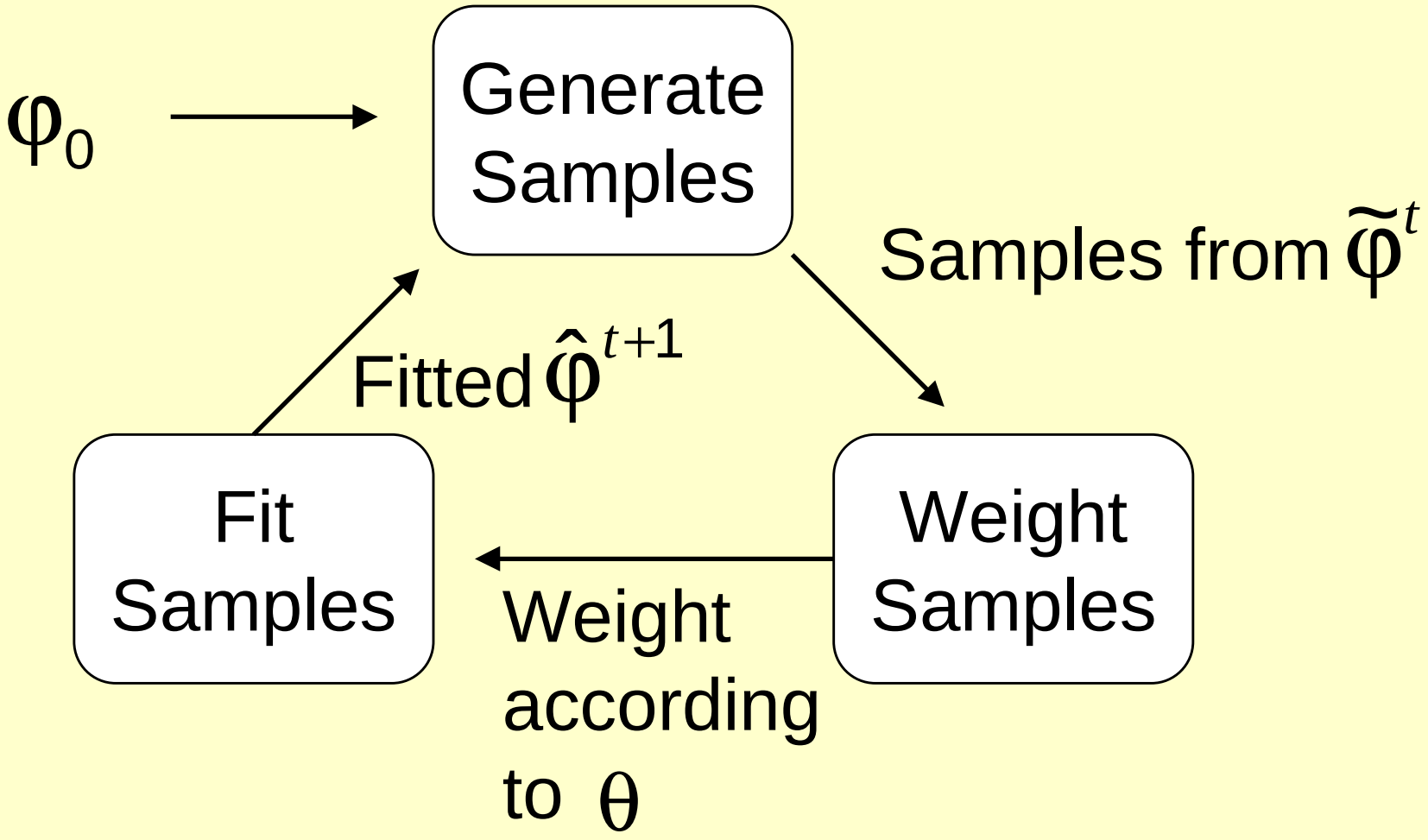
c.f. Neural Networks

What if full model is not available?

Assume generative model:



Rollout with sampling



Gradient Ascent & Sampling

If model fitting is differentiable, why not do:

$$\frac{d\hat{\varphi}^{(t+1)}}{d\theta}(\theta_0) = \frac{d}{d\theta}\hat{\Psi}(\hat{\varphi}^{(t)})$$

$$= \frac{\partial \hat{\Psi}}{\partial \theta}(\theta_0, \hat{\varphi}^{(t)}) + \frac{\partial \hat{\Psi}}{\partial \hat{\varphi}}(\theta_0, \hat{\varphi}^{(t)}) \frac{d\hat{\varphi}^{(t)}}{d\theta}(\theta_0)$$

Problem: Samples are from wrong distribution

Thought Experiment

Consider a new θ_1

Redo estimation, reweighting old samples:

$$w_i(\hat{\varphi}, \theta_1) = \frac{\varphi(s_i^{\,t})\pi_{\theta_1}(a_i \mid s_i^{\,t})}{\varphi^t(s_i^{\,t})}$$

$$\frac{\partial \Psi(\varphi)}{\partial \Phi} = \frac{\partial \Psi(\varphi)}{\partial w}\frac{\partial w}{\partial \Phi}$$

$\Phi =$ everything else

Notes on reweighting

- No samples are actually reused!
- Used for differentiation only
- Accurate, since differentiation considers an infinitesimal change in θ_0

Bicycle Example

- Bicycle simulator from Randlov & Astrom 98
- 9 actions for combinations of lean and torque
- 6 dimensional state + absorbing goal state
- Fitted to 6D multivariate Gaussian
- Used horizon of 200 steps, 300 samples/step
- softmax action selection
- Achieved results comparable to R&A
  - 5 km vs. 7km for “good” trials
  - 1.5km vs. 1.7km for “best” runs

Conclusions

- 3 new uses for density estimation in (PO)MDPs
- POMDP RL
  - Function approx. with density estimation
- Structured MDPs
  - Value determination with guarantees
- Policy search
  - Search space of parameterized policies