Reinforcement Learning as Classification
Leveraging Modern Classifiers

Michail G. Lagoudakis and **Ronald Parr**
Department of Computer Science
Duke University
Durham, NC 27708

Machine Learning Reductions Workshop
September 11–12, 2003
TTI-Chicago, Chicago, IL

Overview

Classification Policy Iteration

An algorithm for learning good policies in sequential decision problems

Main Idea

- Focus on policy learning as opposed to value function learning
- View policy learning as supervised learning (classification)

Motivation

- Value function learning and approximation can be problematic
- Gradient-based algorithms can be inefficient

Benefits

- Direct link between reinforcement learning and classification
- Soundness and efficiency of approximate policy iteration

Background

Markov Decision Processes (MDPs)

An MDP is defined as a 6-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mathcal{D})$:

- $\mathcal{S}$: State space of the process
- $\mathcal{A}$: Action space of the decision maker
- $\mathcal{P}$: Transition model, $\mathcal{P}(s, a, s') = P(s'|s, a)$
- $\mathcal{R}$: Reward function, $\mathcal{R}(s, a)$
- γ : Discount factor, $\gamma \in (0, 1)$
- $\mathcal{D}$: Initial state probability distribution

Markov property : next state and reward independent of history

Markov Decision Processes (cont'd)

Episodes

$$s_0 \xrightarrow[r_0]{a_0} s_1 \xrightarrow[r_1]{a_1} s_2 \xrightarrow[r_2]{a_2} s_3 \quad \ldots \quad s_{h-1} \xrightarrow[r_{h-1}]{a_{h-1}} s_h$$

Expected Total Discounted Reward

$$E\left(r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \cdots + \gamma^h r_h\right)$$

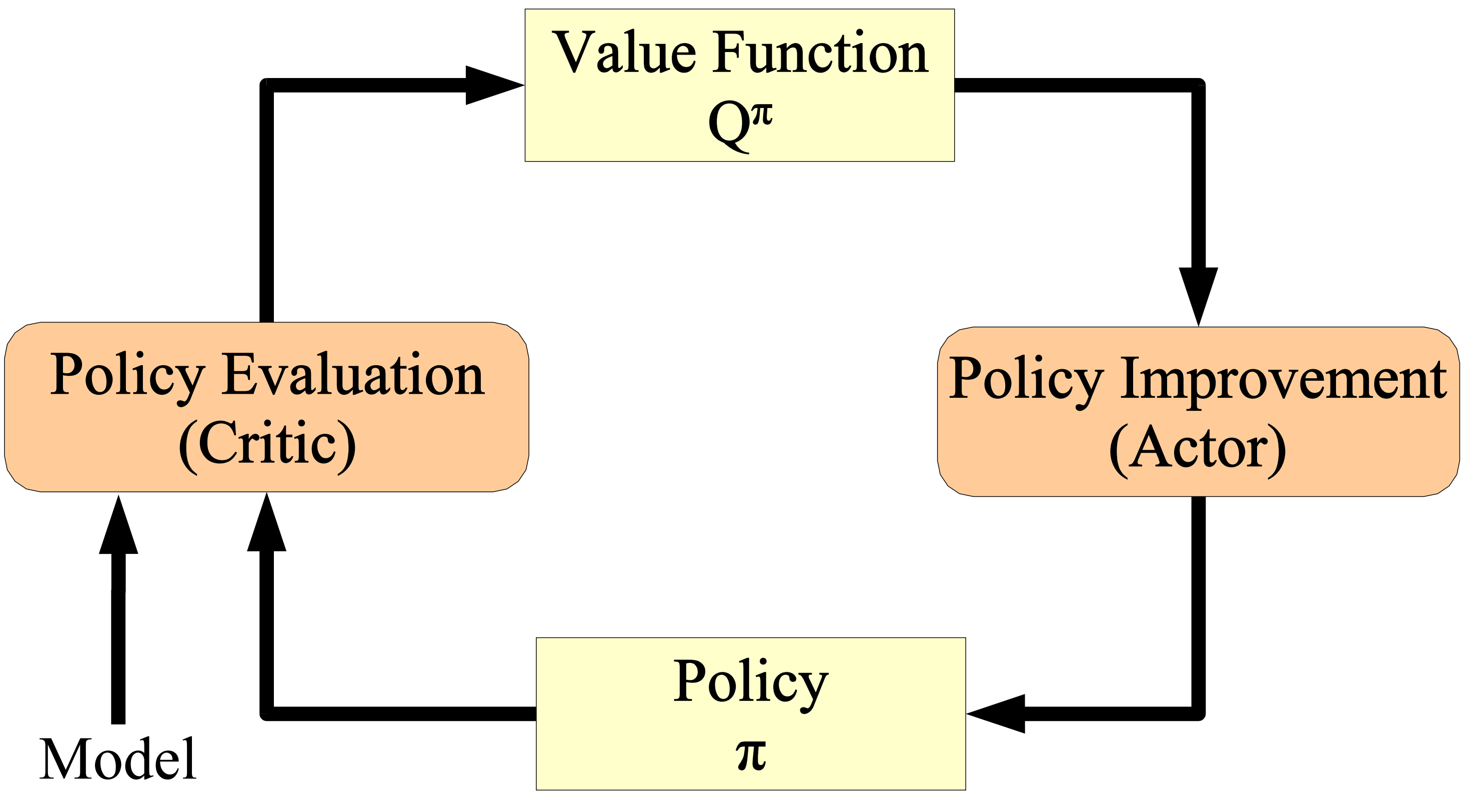
Deterministic Policy

$$\pi : \mathcal{S} \mapsto \mathcal{A}$$

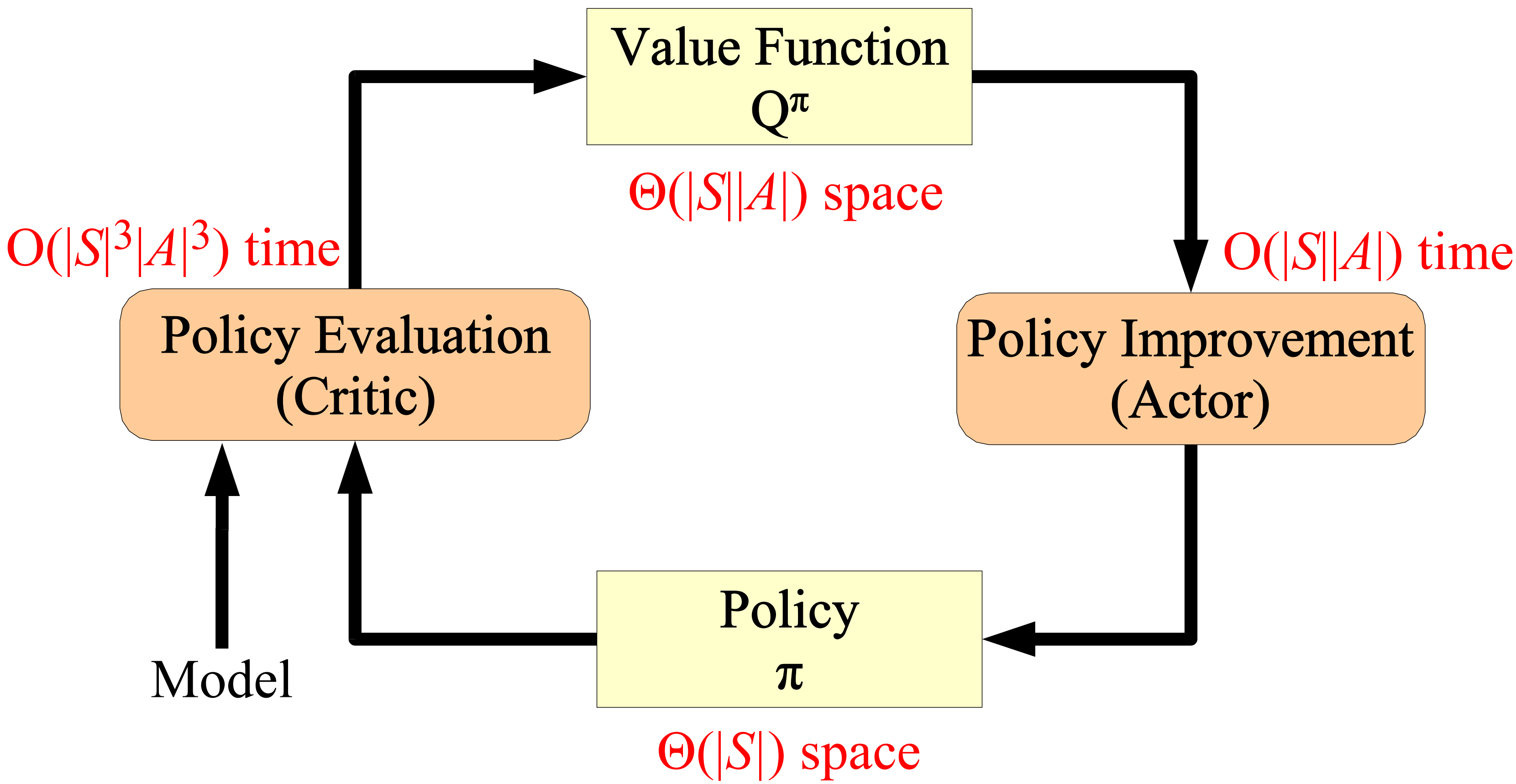
Goal: Optimal Policy

$$\pi^* = \arg\max_{\pi} E_{s \sim \mathcal{D}\ ;\ a_t \sim \pi\ ;\ s_t \sim \mathcal{P}} \left(\sum_{t=0}^{h} \gamma^t r_t \;\middle|\; s_0 = s \right)$$

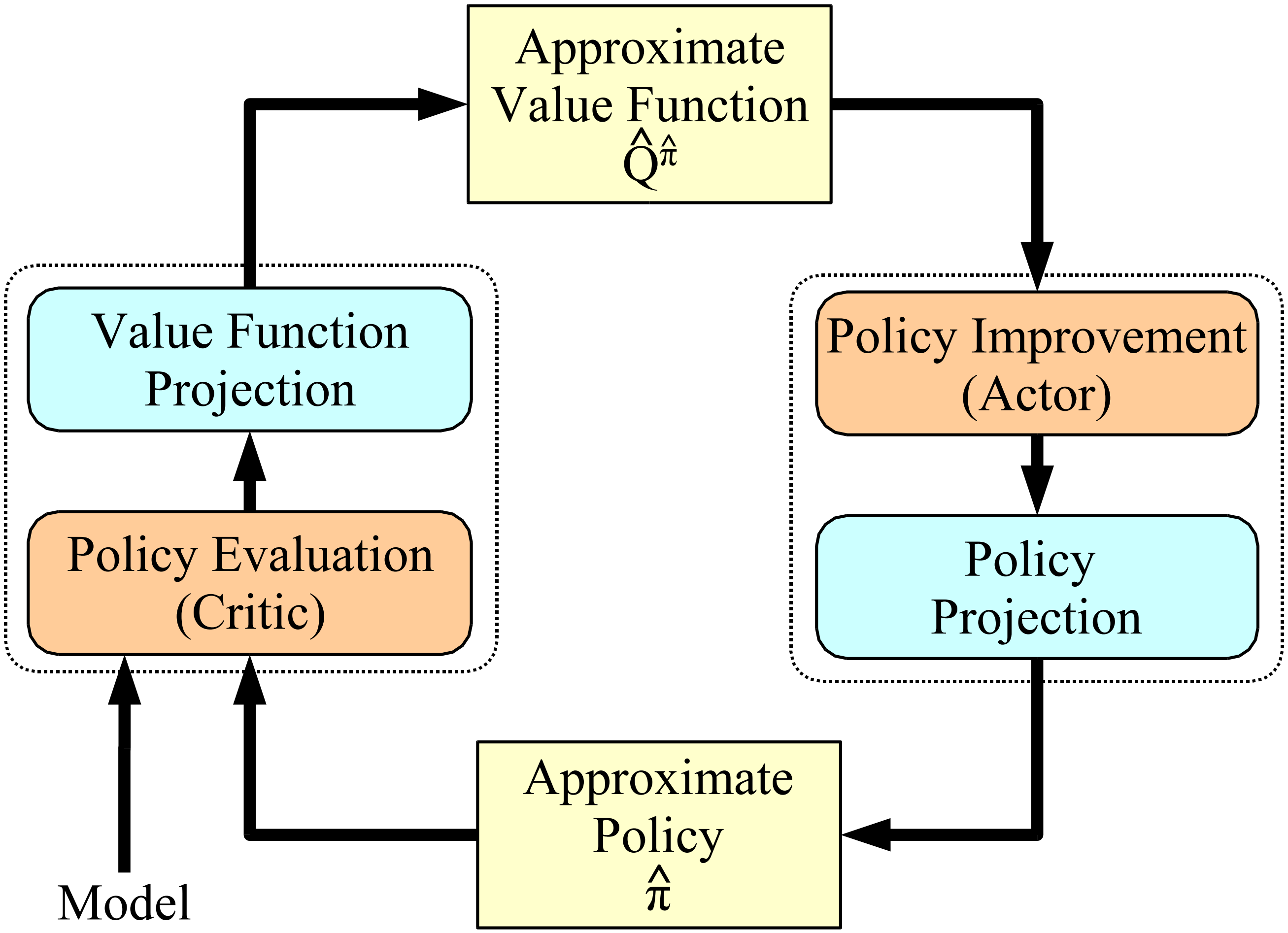
Policy Iteration



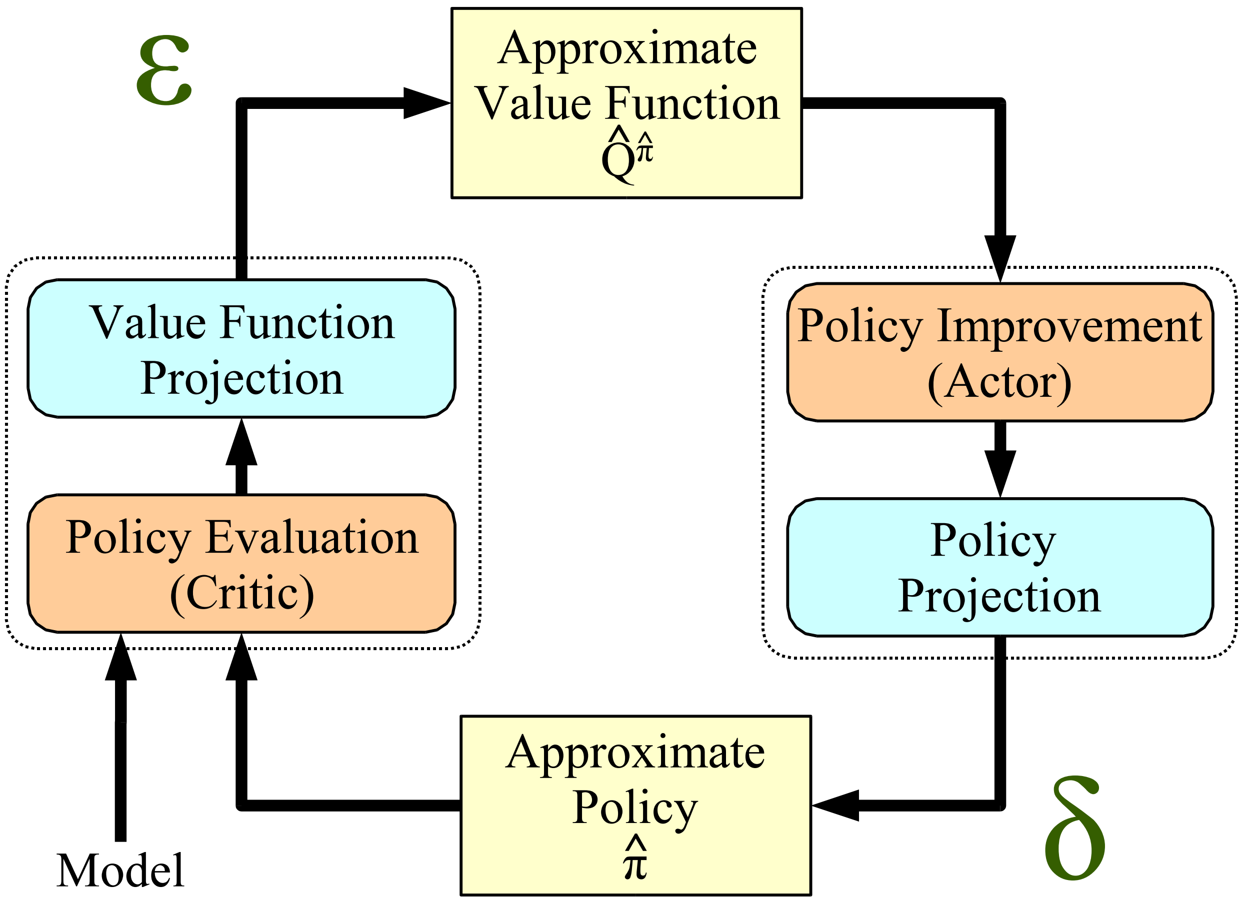
Policy Iteration



Approximate Policy Iteration



Approximate Policy Iteration Errors



Approximate Policy Iteration Bound

Theorem *If there exist positive scalars* ϵ *and* δ *such that*

$$\forall\, k = 0, 1, 2, ..., \; \|\widehat{Q}^{\widehat{\pi}_k} - Q^{\widehat{\pi}_k}\|_\infty \leq \epsilon \;\; ,$$

and

$$\forall\, k = 0, 1, 2, ..., \; \|T_{\widehat{\pi}_{k+1}} \widehat{Q}^{\widehat{\pi}_k} - T_* \widehat{Q}^{\widehat{\pi}_k}\|_\infty \leq \delta \;\; ,$$

then

$$\limsup_{k \to \infty} \|\widehat{Q}^{\widehat{\pi}_k} - Q^*\|_\infty \leq \frac{\delta + 2\gamma\epsilon}{(1-\gamma)^2} \;\; .$$

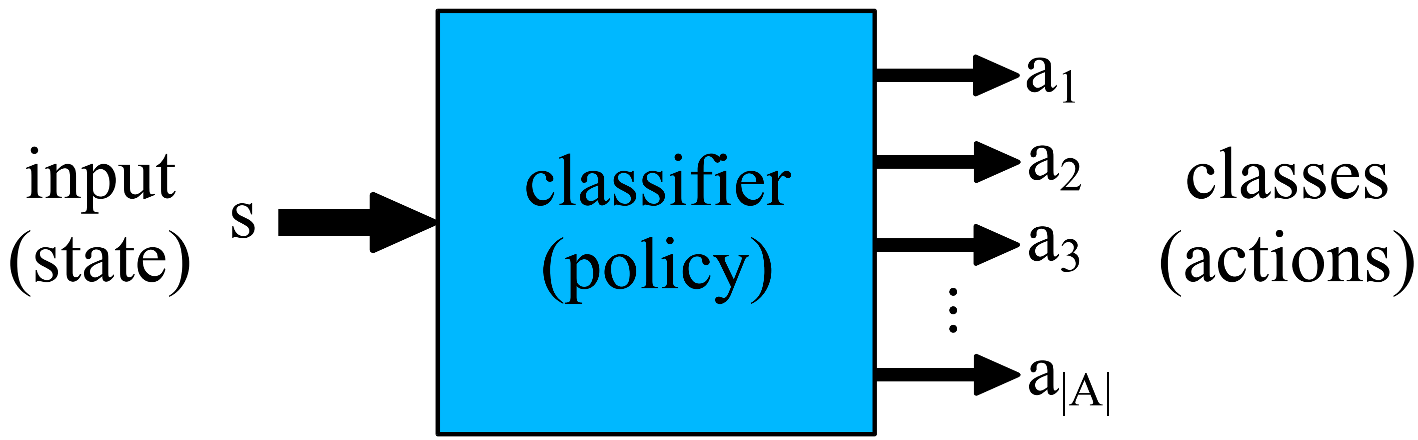
Based on [Bertsekas and Tsitsiklis, 1996]

The Algorithm

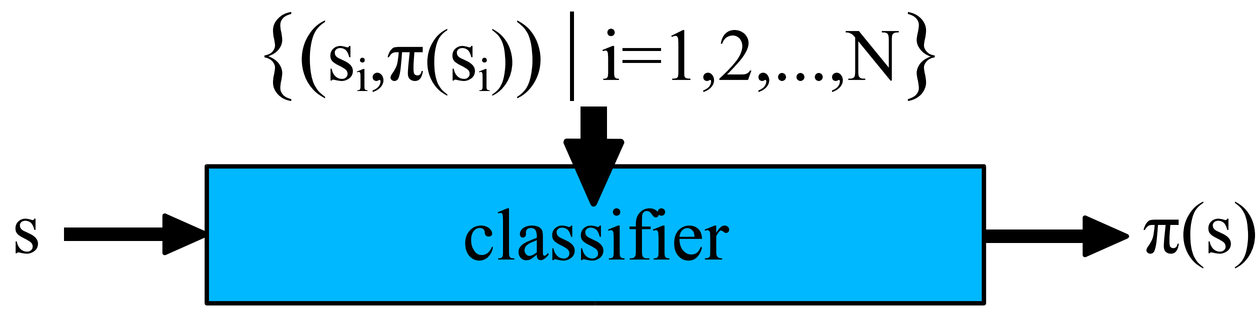
Policy as Classifier

- Deterministic policy: maps states to actions
- Multiclass classifier: maps inputs to classes

A deterministic policy can be represented/implemented as a multiclass classifier!



Policy Learning as Supervised Learning

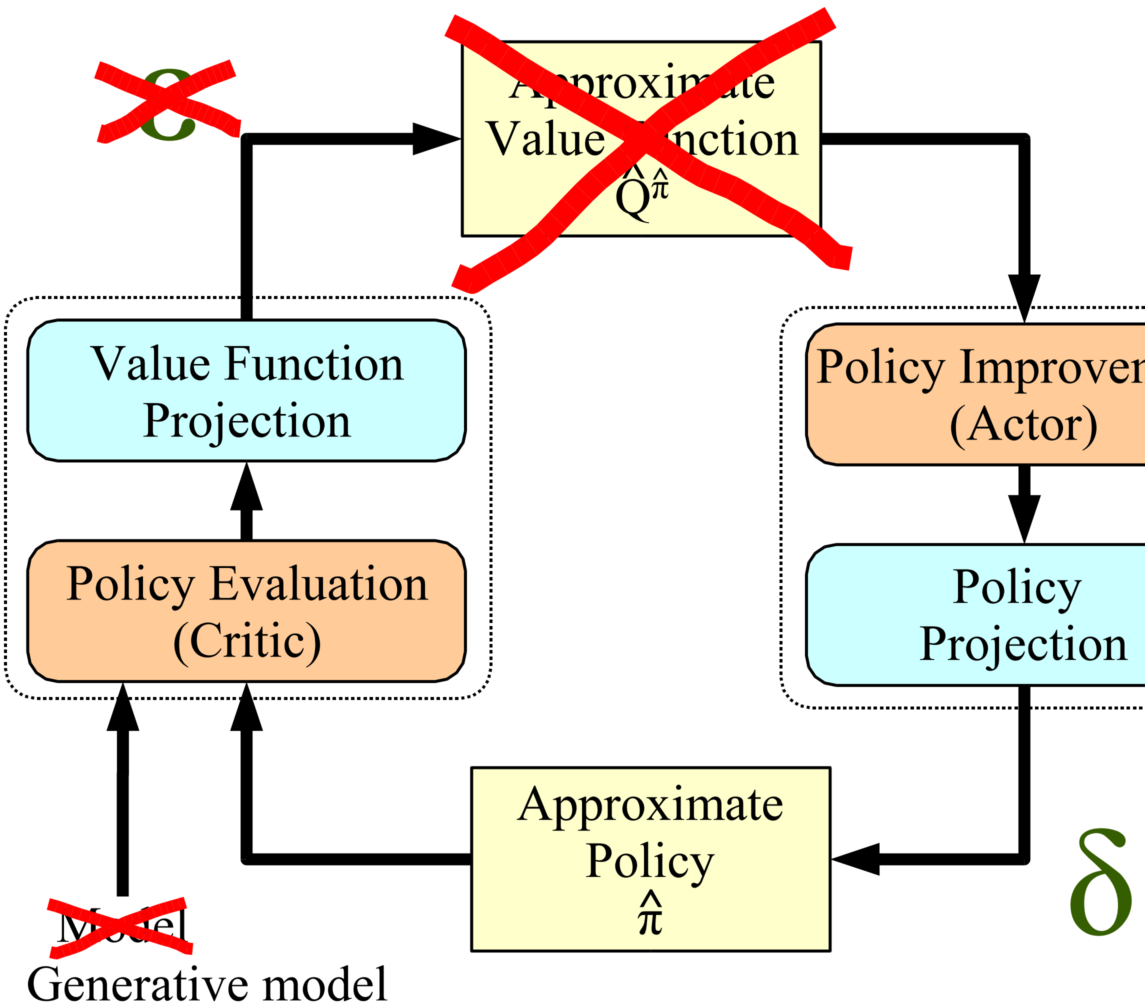


- **Input**: Examples of the target policy at a subset of states
- **Learner**: SVM, Neural net, Decision tree, ILP, ...
- **Output**: A complete policy over the entire state space

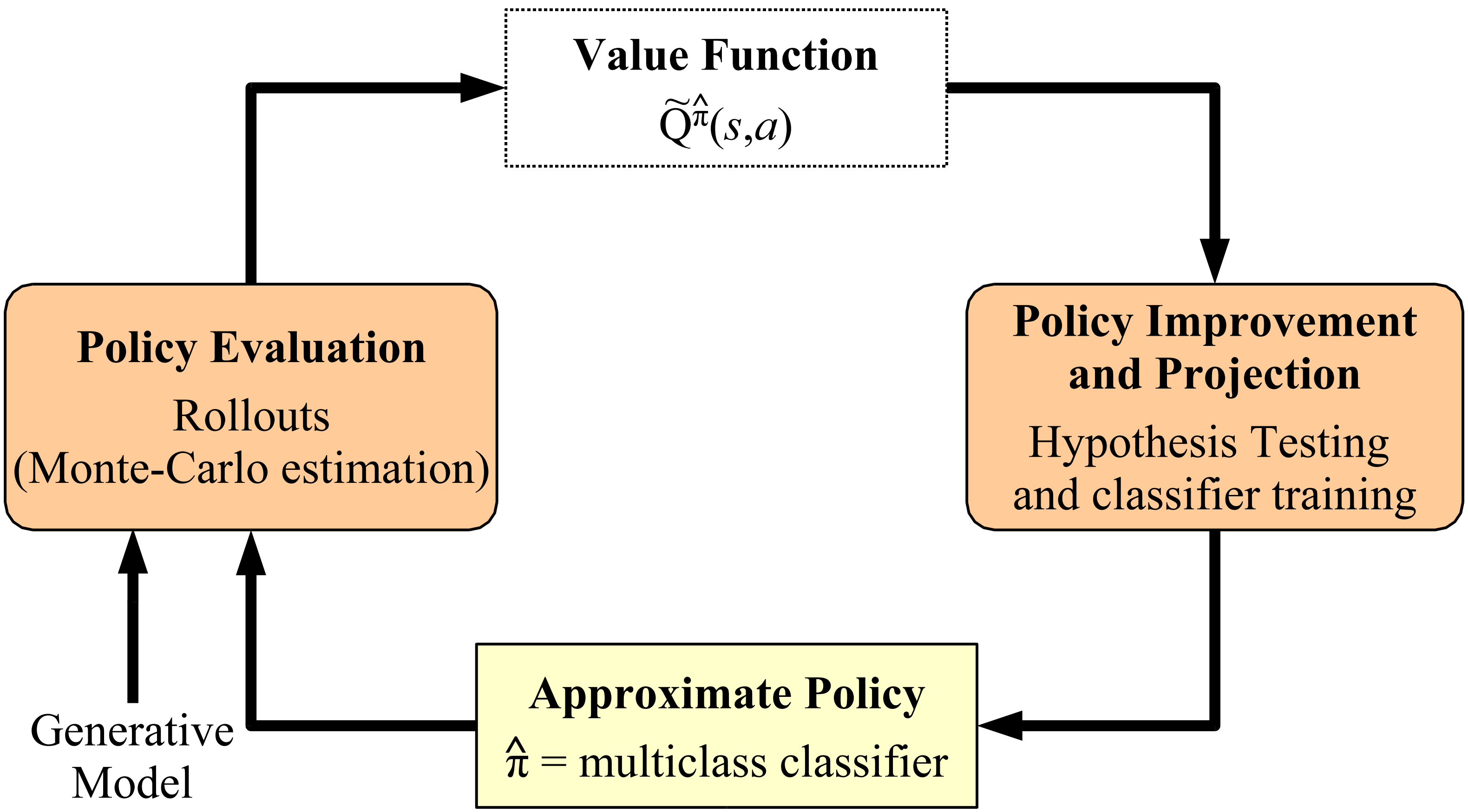
Motivation / Benefits

- Policies may be simple and easy to represent
- Value functions may be complex and hard to approximate
- Potential for discovering structure in the policy

The Key Idea - I



The Key Idea - II



Support Vector Machines, Neural Networks

Rollout: Monte Carlo Value Function Estimation

True State-Action Value Function Q

$$s \xrightarrow[r_0]{a} s_1 \xrightarrow[r_1]{\pi(s_1)} s_2 \xrightarrow[r_2]{\pi(s_2)} s_3 \xrightarrow[r_3]{\pi(s_3)} s_4 \xrightarrow[r_4]{\pi(s_4)} s_5 \quad \ldots$$

$$Q^{\pi}(s,a) = E_{a_t \sim \pi \;;\; s_t \sim \mathcal{P}} \left(\sum_{t=0}^{\infty} \gamma^t r_t \;\middle|\; s_0 = s,\; a_0 = a \right)$$

Estimated State-Action Value Function $\widetilde{Q}$

Simulate K episodes of length T, record $r_{k,t}$

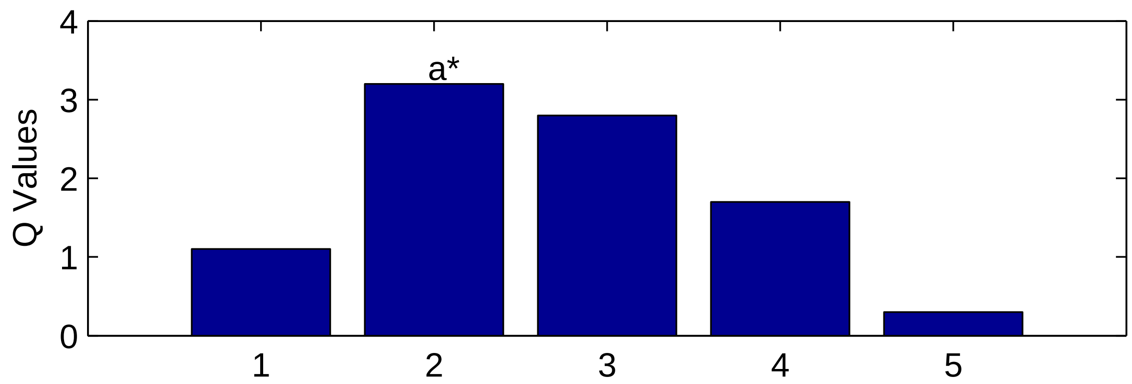
$$\widetilde{Q}^{\pi}(s,a) = \frac{1}{K} \sum_{k=1}^{K} \left(\sum_{t=0}^{T} \gamma^t r_{k,t} \;\middle|\; s_0 = s,\; a_0 = a \right)$$

Classifier Training Data for State s

Pairwise Two-Sample t-Test

$$\widetilde{Q}^{\pi}(s, a_1) \gtrsim \widetilde{Q}^{\pi}(s, a_2) \text{ with 95\% confidence}$$

Training Examples



- $a^* = \arg\max_{a \in \mathcal{A}} \widetilde{Q}^{\pi}(s, a)$
- Positive example $(s, a^*)^+$: $\widetilde{Q}^{\pi}(s, a) \lesssim \widetilde{Q}^{\pi}(s, a^*)$, $\forall\, a \neq a^*$
- Negative example $(s, a)^-$: $\widetilde{Q}^{\pi}(s, a) \gtrsim \widetilde{Q}^{\pi}(s, a^*)$

Distribution of Training (Rollout) States

Uniform distribution

- Simple
- Non-scalable

γ-Discounted future state distribution of a policy π

$$\rho_{\pi,\mathcal{D}} = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t \mathcal{D}(\mathbf{\Pi}_\pi \mathcal{P})^t \ ,$$

- Emphasis on frequently visited states and on their contribution

$$s_0 \sim \mathcal{D} \ \xrightarrow[(P=\gamma)]{\pi(s_0)} \ s_1 \ \xrightarrow[(P=\gamma)]{\pi(s_1)} \ s_2 \ \xrightarrow[(P=\gamma)]{\pi(s_2)} \ s_3 \quad \ldots \quad s_{t-1} \ \xrightarrow[(P=\gamma)]{\pi(s_{t-1})} \ s_t$$

Matching Training and Testing Distributions

Classification assumption

- Training distribution = Testing Distribution

Natural testing distribution for classifier representing $\pi^{(k+1)}$

- γ-discounted future state distribution of the target policy $\pi^{(k+1)}$

$$s_0 \sim \mathcal{D} \xrightarrow[(P=\gamma)]{\pi^{(k+1)}(s_0)} s_1 \xrightarrow[(P=\gamma)]{\pi^{(k+1)}(s_1)} s_2 \ldots \quad s_{t-1} \xrightarrow[(P=\gamma)]{\pi^{(k+1)}(s_{t-1})} s_t$$

Natural training distribution for learning $\pi^{(k+1)}$

- Can we draw states from this distribution? YES! [Fern et al., 03]
- Use the current policy $\widehat{\pi}^{(k)}$, the generative model, and rollouts to determine and execute the target policy $\pi^{(k+1)}$
- Training and testing distributions match!

Termination Criteria

Policy Performance

- Monte-Carlo estimation of η_π

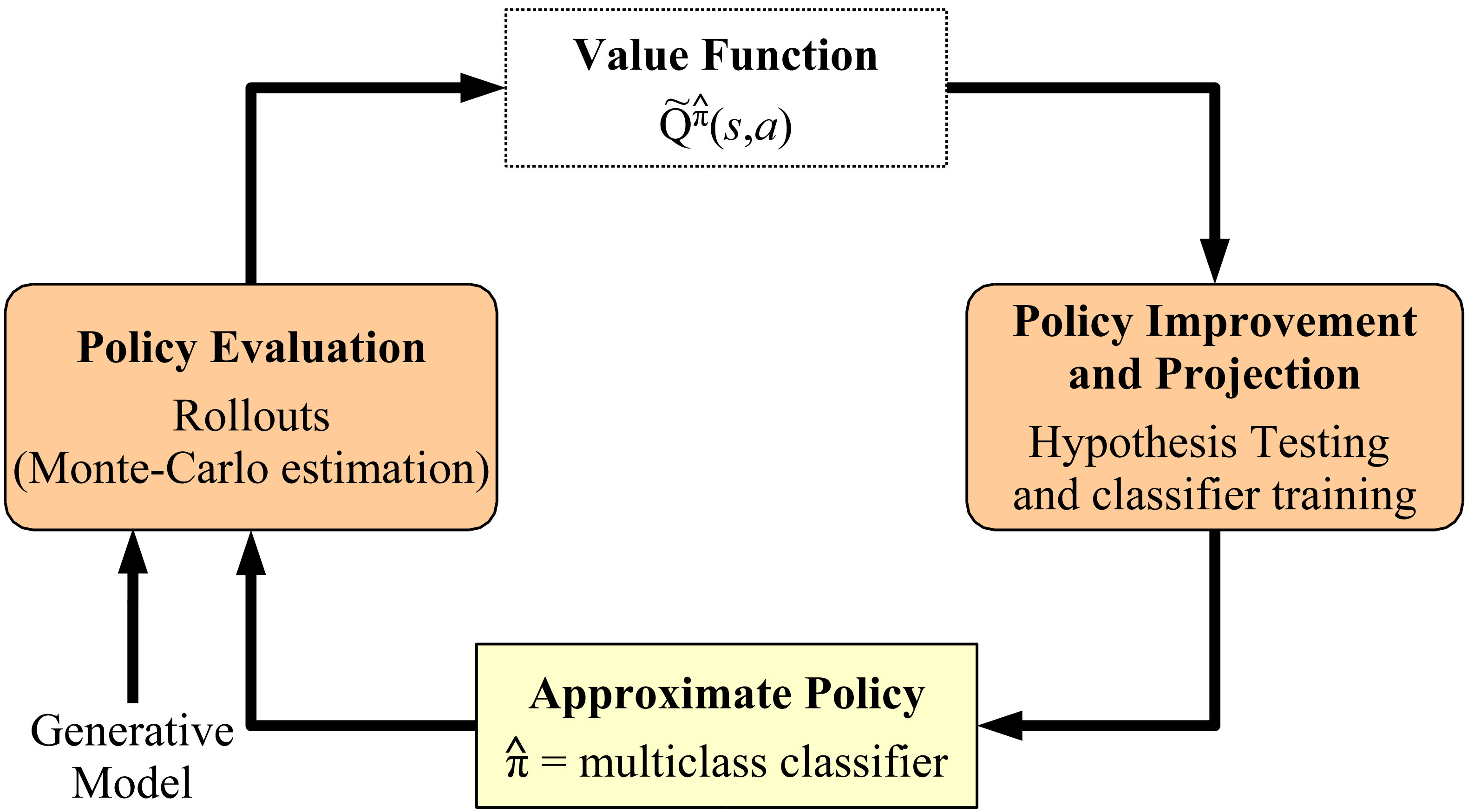
$$\eta_\pi = E_{s\sim\mathcal{D}\ ;\ a_t\sim\pi\ ;\ s_t\sim\mathcal{P}} \left(\sum_{t=0}^{h} \gamma^t r_t \;\middle|\; s_0 = s \right)$$

- Terminate if $\eta_{\pi^{(k-1)}} \geq \eta_{\pi^{(k)}}$

Policy Representation

- Similarity between classifiers
- Terminate if the classifiers for $\pi^{(k-1)}$ and $\pi^{(k)}$ are similar

Summarizing the Algorithm



Support Vector Machines, Neural Networks

The Algorithm

CPI $(\mathcal{M},\ D_\rho,\ \gamma,\ \pi_0,\ K,\ T)$ // Learns a good policy from a generative model

// $\mathcal{M}$: Generative model
// D_ρ : Source of rollout states
// γ : Discount factor
// π_0 : Initial policy (default: uniformly random)

$\pi' = \pi_0$
repeat
  $\pi = \pi'$; TS $= \varnothing$
  for each $s \in D_\rho$
    $\widetilde{Q}^\pi(s,a) \leftarrow$ **Rollout**$(\mathcal{M},\ s,\ a,\ \gamma,\ \pi,\ K,\ T)$, $\forall\, a \in \mathcal{A}$
    $a^* = \arg\max_{a \in \mathcal{A}} \widetilde{Q}^\pi(s,a)$
    if $\forall\, a \in \mathcal{A},\ a \neq a^* :\ \widetilde{Q}^\pi(s,a) \lesssim \widetilde{Q}^\pi(s,a^*)$
      TS $\leftarrow$ TS $\cup \{(s,a^*)^+\}$
    for each $a \in \mathcal{A} :\ \widetilde{Q}^\pi(s,a) \lesssim \widetilde{Q}^\pi(s,a^*)$
      TS $\leftarrow$ TS $\cup \{(s,a)^-\}$
  $\pi' =$ **Classifier-Training**(TS)
until $(\pi \approx \pi')$
return π

Complexity: $O\left(|D_\rho|\left(T_{\mathcal{M}}(KT) + |\mathcal{A}|^2\right) + T_{\text{Train}}(|D_\rho|)\right)$ time/iteration, $O\left(|D_\rho|\right)$ space

Properties of the Algorithm

Advantages

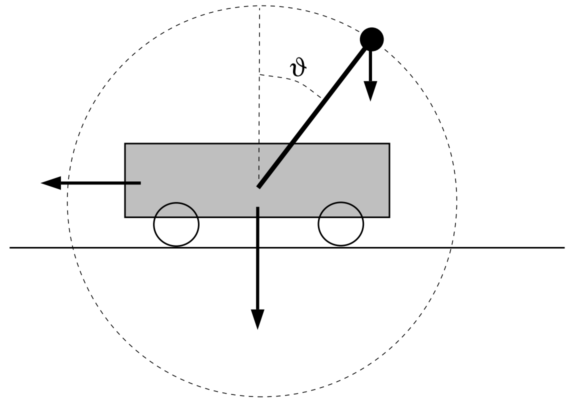
- Is stable; does not diverge
- Eliminates value function approximation
- Simplifies feature engineering with modern classifiers
- Is simple and easy to implement

Limitations

- May yield poor policies with badly distributed rollout states
- Is practical only for small action spaces
- Needs a generative model

Experiments

Inverted Pendulum Balancing

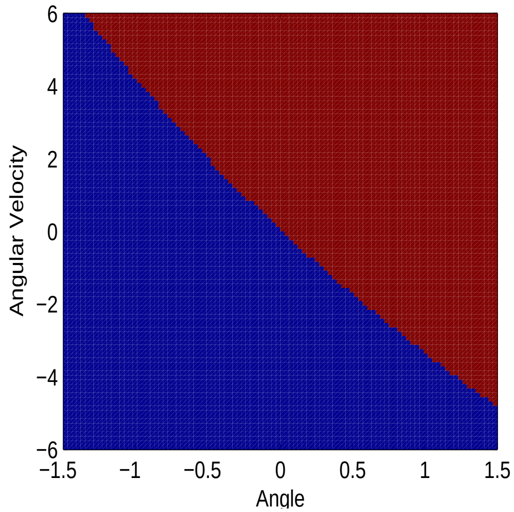


Balance the pendulum at the upright position!

- $\mathcal{S} = \{(\text{angle } \theta, \text{angular velocity } \dot{\theta})\}$
- $\mathcal{A} = \{-50 \text{ N},\ 0 \text{ N},\ +50 \text{ N}\}$
- Model: non-linear dynamical system [Wang et al., 1996]
- Noise: Input $u = a + 10n$, n uniform in $[-1, +1]$
- Reward: -1 if $|\theta| > \frac{\pi}{2}$, 0 otherwise
- $\gamma = 0.95$

Pendulum: Policies Learned

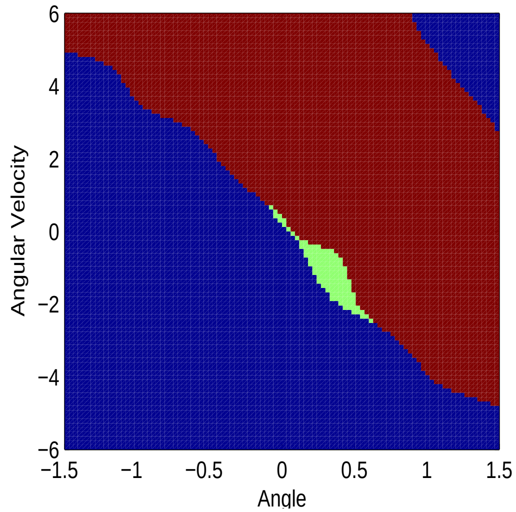
blue: left force, green: no force, red: right force



SVM

polynomial

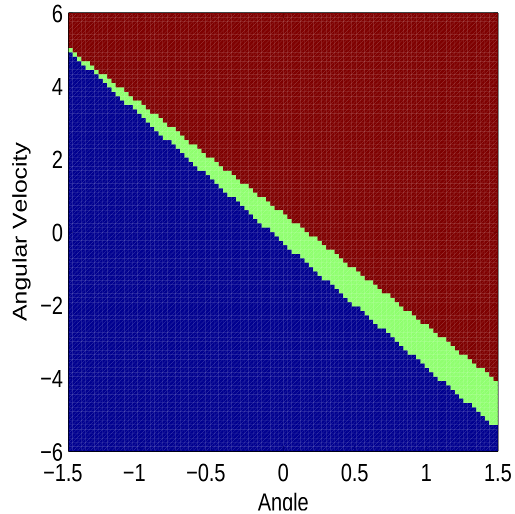
(deg 2) kernel



SVM

gaussian

kernel



Neural Network

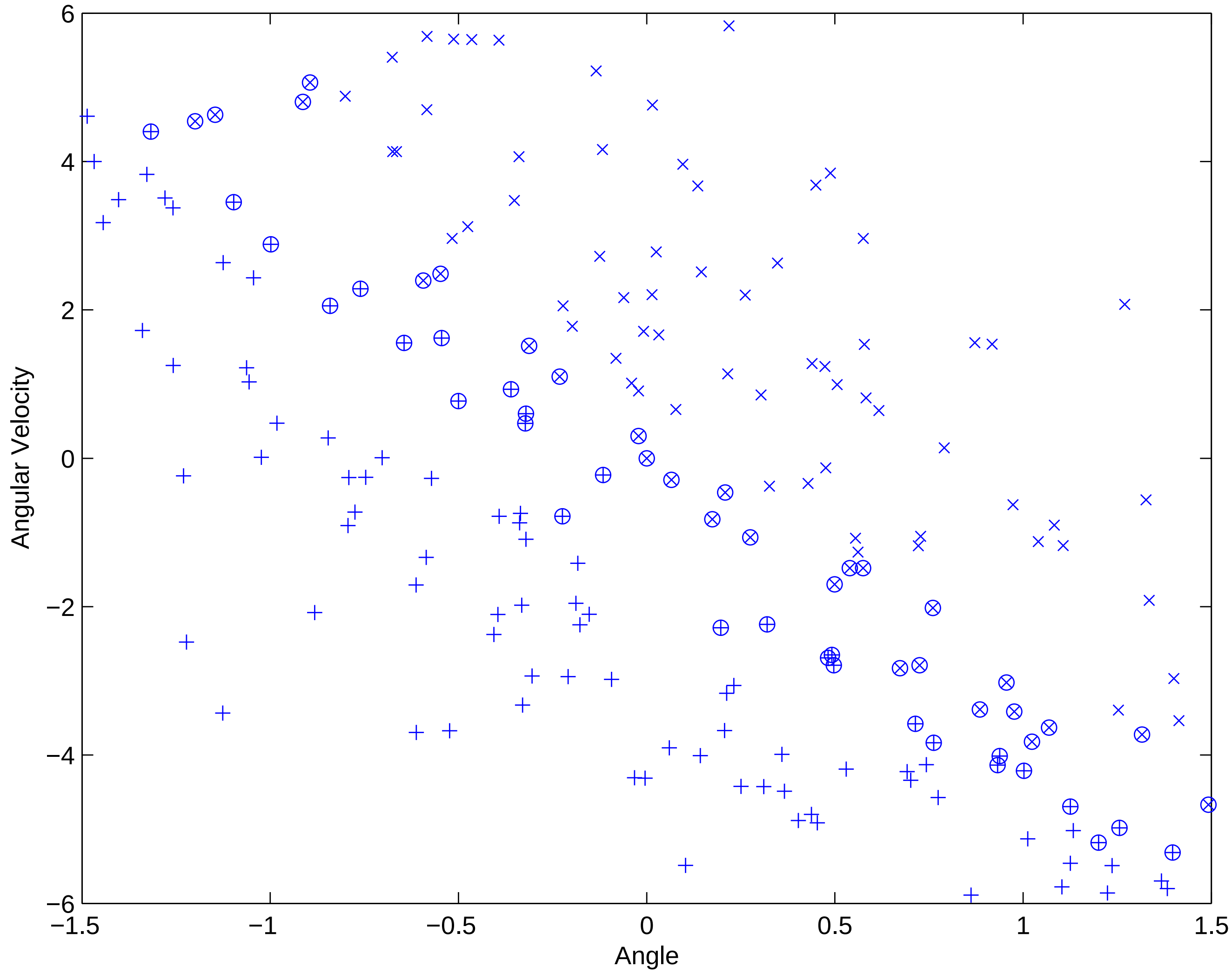
5 hidden

units

200 rollout states, uniform distribution

Learned in one iteration starting from the random policy

Pendulum: Data for the LeftForce Action



Positive (+), negative (x) examples and support vectors (o)

Bicycle Balancing and Riding

Balance and ride a bicycle at a target location 1 Km away!

- $\mathcal{S} = \{(\theta, \dot{\theta}, \omega, \dot{\omega}, \ddot{\omega}, \psi)\}$

  θ: angle of the handlebar, ω: vertical angle, ψ: angle to the goal

- $\mathcal{A} = \{(\tau, \upsilon)\}$

  $\tau \in \{-2, 0, +2\}$: torque, $\upsilon \in \{-0.02, 0, +0.02\}$: displacement

- Model: non-linear dynamical system [Randløv and Alstrøm, 1998]

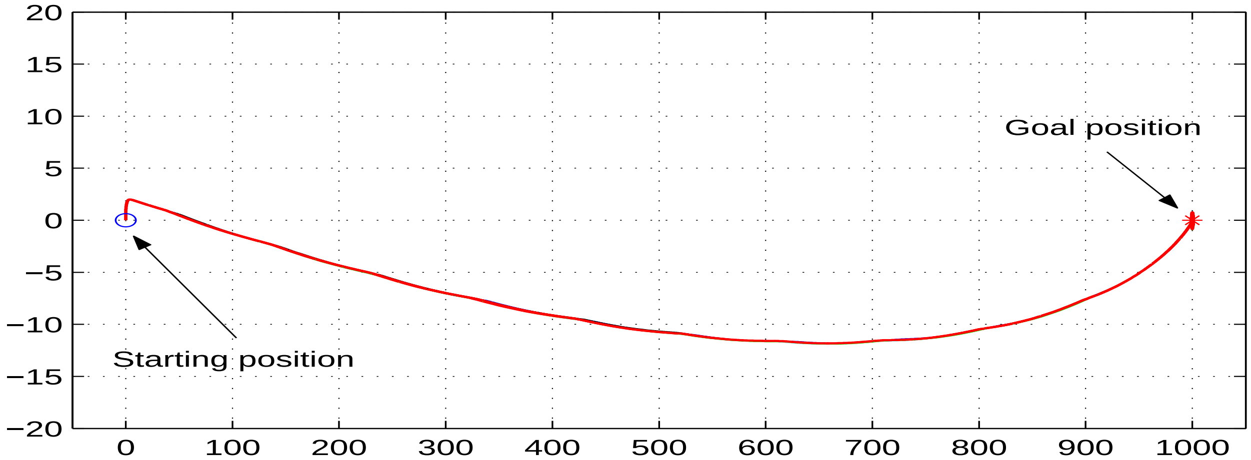
- Noise: input $(\tau, \upsilon + n)$, $n \in [-0.02, +0.02]$

- Reward:
  - 1 for balancing + 1% of the net change in distance to goal
  - 0 for crashing

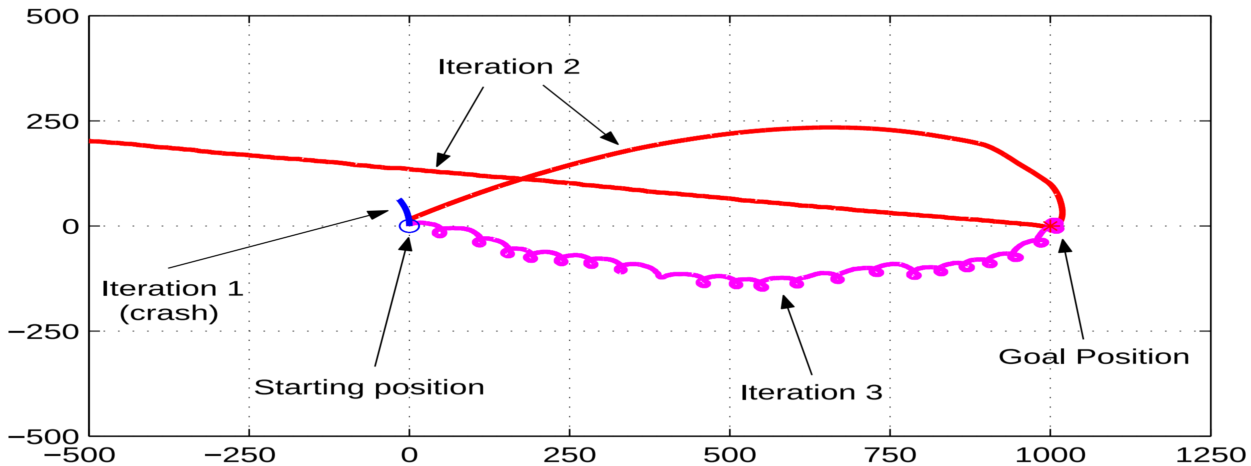
- $\gamma = 0.95 - 0.99$

Bicycle: Policies from Sample Runs

SVM (polynomial kernel of degree 3, 4000 rollout states)



NN (30 hidden units, 8000 rollout states)



Related Work

The talks in this session of the workshop!

[Fern, Yoon, and Givan, 2003]

- Decision lists, cost-sensitive classification, policy language bias

[Bagnell, Kakade, Ng, and Schneider, 2003]

- Binary action MDPs, linear decision boundaries for classification

[Langford and Zadrozny, 2003]

- T-step non-stationary policies, one classifier per step, (forked) traces

Future Work and Conclusion

Future Work

- Sparse sampling techniques for value function estimation
- Alternative distributions for training states
- Multi-agent learning: Zero-Sum Markov games, Team MDPs
- Probabilistic classification methods for stochastic policies

Conclusion

- A reinforcement learning algorithm in policy space
- No value function approximation, No policy gradient
- Soundness and efficiency of approximate policy iteration
- Direct link between reinforcement learning and classification

Acknowledgments

Thanks!

- *John Langford, David McAllester* (TTI-Chicago)
- *Alan Fern, Bob Givan* (Purdue)
- *Ryan Deering* (Duke), *Carlos Guestrin* (Stanford)
- *National Science Foundation* (NSF grant 0209088)

... and to all of you for your attention!