# Binary Action Search for Learning Continuous-Action Control Policies

Jason Pazis and Michail G. Lagoudakis

Intelligent Systems Laboratory
Department of Electronic and Computer Engineering
Technical University of Crete
Chania, Crete, Greece

The 26th International Conference on Machine Learning
June 14-18, 2009
Montreal, Canada

Partially supported by Marie Curie International Reintegration Grant MIRG-CT-2006-044980.

# Motivation: Discrete agents in a continuous world

## Current Algorithms

- Can easily handle continuous state spaces
- Mostly handle discrete action spaces

# Motivation: Discrete agents in a continuous world

## Current Algorithms

- Can easily handle continuous state spaces
- Mostly handle discrete action spaces

## Real-world problems

- Many problems have continuous control variables

# Motivation: Discrete agents in a continuous world

## Current Algorithms

- Can easily handle continuous state spaces
- Mostly handle discrete action spaces

## Real-world problems

- Many problems have continuous control variables

## The problem

- Current continuous-action approaches are often inefficient
- Can we control continuous variables using discrete decisions?

# Outline

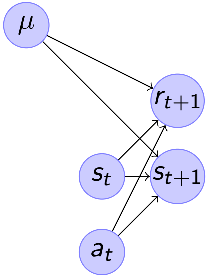
# Outline

# Markov Decision Process

## Markov Decision Process

MDP $\mu = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mathcal{D})$

- $\mathcal{S}$ is the state space
- $\mathcal{A}$ is the action space
- $P$ is the transition model: $P(s'|s, a)$
- $\mathcal{R}$ is the reward function: $\mathcal{R}(s, a)$
- $\gamma \in (0, 1]$ is the discount factor
- $\mathcal{D}$ is the initial state distribution




## Markov Property

- Transitions and rewards are independent of history

# Planning

## Optimization

Optimize the expected total discounted reward

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

# Planning

## Optimization

Optimize the expected total discounted reward

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

## Policy $\pi$

- A way of making decisions in all situations
- Deterministic policy: a mapping from states to actions
- There exists at least one deterministic optimal policy $\pi^*$

# Planning

## Optimization

Optimize the expected total discounted reward

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

## Policy $\pi$

- A way of making decisions in all situations
- Deterministic policy: a mapping from states to actions
- There exists at least one deterministic optimal policy $\pi^*$

## Algorithms

- Value iteration, policy iteration, linear programming

# Learning

## Interaction

- Transition model and reward function are unknown
- Repeated interaction with an unknown process
- Sample at time $t$: $(s_t, a_t, r_t, s_{t+1})$

# Learning

## Interaction

- Transition model and reward function are unknown
- Repeated interaction with an unknown process
- Sample at time $t$: $(s_t, a_t, r_t, s_{t+1})$

## Reinforcement Learning

- *Prediction*: learn/predict the value of a fixed policy
- *Control*: learn a good policy for controlling the process

# Learning

## Interaction

- Transition model and reward function are unknown
- Repeated interaction with an unknown process
- Sample at time $t$: $(s_t, a_t, r_t, s_{t+1})$

## Reinforcement Learning

- *Prediction*: learn/predict the value of a fixed policy
- *Control*: learn a good policy for controlling the process

## Algorithms

- Prediction: DUE, TD-Learning, LSTD, ...
- Control: $Q$-Learning, Sarsa, LSPI, FQI, ...

# The need for continuous actions in control

## Benefits

- Smoothness of motion
- Power consumption
- Mechanical stresses
- Induced power line noise

# The need for continuous actions in control

## Benefits

- Smoothness of motion
- Power consumption
- Mechanical stresses
- Induced power line noise

## Problems

- An infinite number of choices at each step
- Tabular approaches are not sufficient
- Discrete maximization is not sufficient
- Fine discretization is inefficient

# Related work

## Neural network approaches

- Gaskett et al., *AI* 1999
- Ströslin et al., *ICANN* 2003

## Monte Carlo sampling

- Lazaric et al., *NIPS* 2008
- Sallans and Hinton, *JMLR* 2004

## Single state-action approximator

- Santamaria, Sutton, Ram, *Adaptive Behavior* 1998

## Exploitation of temporal locality

- Pazis and Lagoudakis, *ADPRL* 2009
- Riedmiller, *ESANN* 1997

# Outline

1. Introduction
2. Binary Action Search
3. Experiments
4. Conclusion

# Binary Action Search

## Choosing continuous actions

- Choosing a continuous action value in a single step is hard!
- How about breaking this hard decision into many easier ones?

# Binary Action Search

## Choosing continuous actions

- Choosing a continuous action value in a single step is hard!
- How about breaking this hard decision into many easier ones?

## Idea

- Given a continuous action value in some state ...
- ... decide whether it's better to increase it or decrease it!

# Binary Action Search

## Choosing continuous actions

- Choosing a continuous action value in a single step is hard!
- How about breaking this hard decision into many easier ones?

## Idea

- Given a continuous action value in some state ...
- ... decide whether it's better to increase it or decrease it!

## Multi-step action choice

- Need a discrete binary policy $\pi : \mathcal{S} \times \mathcal{A} \mapsto \{\text{Inc}, \text{Dec}\}$
- Perform $N$-step binary search over the action space
- Successively approximate the best continuous action value
- Anytime algorithm: more accurate action choice with larger $N$

# Binary Action Search

**Binary Action Search** $(s, \pi, N)$
// $s$ : The current state of the process
// $\pi$ : A policy making binary decisions, $+1$ or $-1$
// $N$ : The number of resolution bits

$a \leftarrow (a_{\max} + a_{\min})/2$ // Initialize $a$
$\Delta \leftarrow (a_{\max} - a_{\min})2^{N-1}/(2^N - 1)$ // Initialize $\Delta$

**for** $i = 1$ **to** $N$ **do**
  $\Delta \leftarrow \Delta/2$ // update $\Delta$
  $e \leftarrow \pi(s, a)$ // binary decision ($+1$ or $-1$)
  $a \leftarrow a + e\Delta$ // update $a$
**end for**
**return** $a$

# Learning Binary Policies

## Requirements

- continuous augmented state space $(\mathcal{S}, \mathcal{A})$
- discrete binary action space $\{\text{Increase } (+1), \text{Decrease } (-1)\}$
- most reinforcement learning algorithms can be used

# Learning Binary Policies

## Requirements

- continuous augmented state space $(\mathcal{S}, \mathcal{A})$
- discrete binary action space $\{\text{Increase }(+1), \text{Decrease }(-1)\}$
- most reinforcement learning algorithms can be used

## Learning Data

- need to get sample transitions for the transformed MDP
- for each actual transition sample with a continuous action ...
- ... generate $N$ transition samples with discrete actions

# A Simple Example

## A simplified domain

- Continuous action range $[1.0, 8.0]$
- $N = 3$ (3-bit resolution)

# A Simple Example

## A simplified domain

- Continuous action range $[1.0, 8.0]$
- $N = 3$ (3-bit resolution)

## Actual sample

$$s \xrightarrow[r]{a = 2.0} s'$$

# A Simple Example

## A simplified domain

- Continuous action range $[1.0, 8.0]$
- $N = 3$ (3-bit resolution)

## Actual sample

$$s \xrightarrow[r]{a = 2.0} s'$$

## Derived samples

$$s, 4.5 \xrightarrow[0]{-1 \ (a = 2.5)} s, 2.5 \xrightarrow[0]{-1 \ (a = 1.5)} s, 1.5 \xrightarrow[r]{+1 \ (a = 2.0)} s', 4.5$$

# Properties

## Optimality

- Searching for the single best action - no local optima

# Properties

## Optimality

- Searching for the single best action - no local optima

## Integration

- BAS can be combined with most existing RL algorithms
- The RL algorithm needs to support continuous state spaces
- Decisions over an augmented state space: $(\mathcal{S}, A)$

# Properties

## Optimality

- Searching for the single best action - no local optima

## Integration

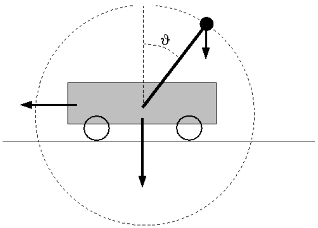
- BAS can be combined with most existing RL algorithms
- The RL algorithm needs to support continuous state spaces
- Decisions over an augmented state space: $(\mathcal{S}, A)$

## Efficiency

- Needs only a binary search policy
- Scales logarithmically with the resolution

# Outline

# Inverted Pendulum




## Balancing a pendulum at the upright position

- States: vertical angle $\theta$ and angular velocity $\dot{\theta}$
- Discrete actions: three actions [$-50$ N, 0 N, $+50$ N]
- Continuous actions: $2^8$ equally spaced in [$-50$ N, $+50$ N]
- Uniform noise in $[-10N, +10N]$ is added to all actions

# Inverted Pendulum

## Learning Setup

- Training samples collected in advance from “random episodes”
- Starting in a randomly perturbed state near equilibrium
- Following a policy that made random decisions

## Parameters

- Reward function: $-((2\theta/\pi)^2 + (\dot{\theta})^2 + (F/50)^2)$
- $|\theta| > \pi/2$ signals the end of episode and a reward of $-1000$
- Discount factor $\gamma = 0.95$
- Control interval $dt = 100$ msec

## Basis Functions

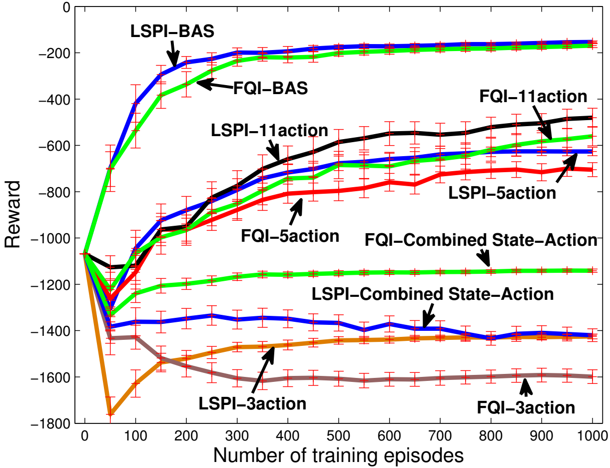
- Augmented state vector $s = (\theta, \dot{\theta}, F)$
- Block of 28 basis functions for each discrete action
- 1 constant term and 27 radial basis functions (Gaussians)
- Arranged in a $3 \times 3 \times 3$ grid

$$\phi = \Big(1 \quad , \quad e^{-\frac{\sqrt{(\theta/n_\theta - \theta_1)^2 + (\dot{\theta}/n_{\dot{\theta}} - \dot{\theta}_1)^2 + (F/n_F - F_1)^2}}{2\sigma^2}},$$

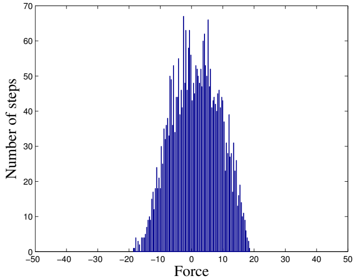
$$\cdots \quad , \quad e^{-\frac{\sqrt{(\theta/n_\theta - \theta_3)^2 + (\dot{\theta}/n_{\dot{\theta}} - \dot{\theta}_3)^2 + (F/n_F - F_3)^2}}{2\sigma^2}} \Big)^\top ,$$

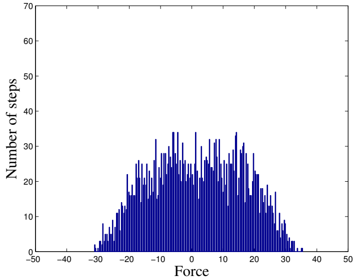
- $\theta_i$'s, $\dot{\theta}_i$'s and $F_i$'s are in $\{-1,\ 0,\ +1\}$
- $n_\theta = \pi/2$, $n_{\dot{\theta}} = 2$ and $n_F = 50$

# Inverted Pendulum: Total accumulated reward



# Inverted Pendulum: $10N$ (left) and $20N$ (right) noise




- $10N$ noise BAS mean force magnitude: 6.65N
- $10N$ noise 3-action mean force magnitude: 17.91N
- $20N$ noise BAS success rate: 99.64%
- $20N$ noise 3-action success rate: 39.49%

# Double Integrator

## Control a car moving on a one-dimensional flat terrain

- States: position $p$ and velocity $v$
- Actions: Control the acceleration $a$
- Linear dynamics: $\dot{p} = v$ and $\dot{v} = a$

## Setup

- Reward function: $-(p^2 + a^2)$
- Constraints: $|p| \leq 1$, $|v| \leq 1$ and $|a| \leq 1$
- $-50$ reward for constraint violation
- Discount factor $\gamma = 0.98$
- Control interval $dt = 500$ msec

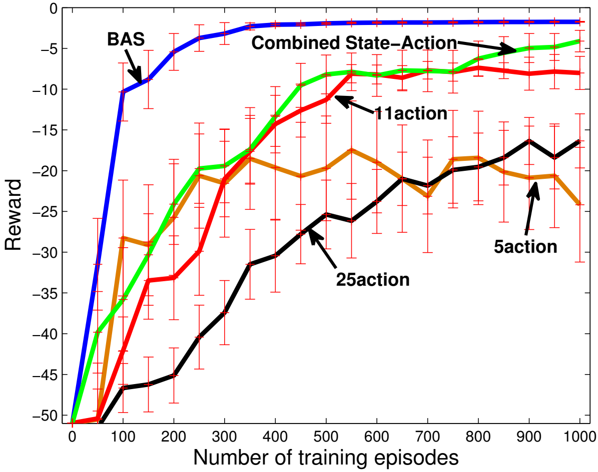
# Double Integrator

## Basis Functions

- Augmented state vector $s = (p, v, a)$
- Simple polynomial approximator with 10 terms

$$\phi = (1, p, v, a, p^2 a, v^2 a, a^2, pv, pa, va, a^2 p, a^2 v)^\top$$

# Double Integrator: Total accumulated reward



# Outline

1. Introduction
2. Binary Action Search
3. Experiments
4. Conclusion

## Strengths

- Simplicity
- Requires no tuning
- Requires only 2 actions from the discrete policy
- Achieves resolutions impossible to reach with discrete actions
- Can be used in conjunction with any RL algorithm
- Can be used in an online, offline, on-policy or off-policy setting

## Strengths

- Simplicity
- Requires no tuning
- Requires only 2 actions from the discrete policy
- Achieves resolutions impossible to reach with discrete actions
- Can be used in conjunction with any RL algorithm
- Can be used in an online, offline, on-policy or off-policy setting

## Weaknesses

- The state space of the problem is now more complex
- More samples have to be processed by the learning algorithm

# Future Work

## Ongoing Research

- High-dimensional action spaces
- Increasing learning and execution efficiency

## Future Research

- Planning with BAS
- Skewing functions over action range

# Acknowledgments

- Thank you for your attention!