PEGASUS for Helicopter Control

Presented by: Ken Alton

References

- PEGASUS: A policy search method for large MDPs and POMDPs, 2000 - Andrew Ng and Michael Jordan
- Shaping and Policy Search in Reinforcement Learning, PhD Thesis, 2003 – Andrew Ng
- Autonomous helicopter flight via reinforcement learning, 2004 – Andrew Ng, H. Jin Kim, Michael Jordan, and Shankar Sastry
- Inverted autonomous helicopter flight via reinforcement learning, 2004 – Andrew Ng and others
- An application of reinforcement learning to aerobatic helicopter flight, 2007 - Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y. Ng

PEGASUS

- Policy Evaluation-of-Goodness And Search Using Scenarios
- Markov Decision Process (MDP)

$$M = (S, D, A, \{P_{sa}(\cdot)\}, \gamma, R)$$

- Policy $\pi : S \rightarrow A$

Policy Evaluation and Optimization

- $V^\pi(s)$ is expected discounted sum of rewards for executing policy π starting from state s
- Value of a policy $V_M(\pi) = \mathbf{E}_{s_0 \sim D}[V_M^\pi(s_0)]$
- Optimal policy within class for MDP M

$$opt(M, \Pi) = \sup_{\pi \in \Pi} V_M(\pi)$$

- Find a policy

$\widehat{\pi} \in \Pi$ such that $V(\widehat{\pi})$ is close to $opt(M, \Pi)$

Deterministic Simulative Model

- For POMDPs
  - Memory-free policies that depend only on observables
  - Limited-memory policies that introduce artificial memory variables into the state
- Generative model – takes input (s,a) and outputs s' according to $P_{sa}(\cdot)$
- Deterministic simulative model

$$g : S \times A \times [0,1]^{d_P} \to S$$

Deterministic Simulative Model (2)

$$g : S \times A \times [0, 1]^{d_P} \rightarrow S$$

- Most computer implementations provide this model but need to expose interface to random number generator
- Example $S = \mathbb{R}$
  - Normal distributed $Psa(\cdot)$
  - Cumulative distribution function $F_{sa}(\cdot)$
  - Let $d_P = 1$ and choose g to be

$$g(s, a, p) = F_{sa}^{-1}(p)$$

Transformation of (PO)MDP

- Given $M = (S, D, A, \{P_{sa}(\cdot)\}, \gamma, R)$
- Construct $M' = (S', D', A, \{P'_{sa}(\cdot)\}, \gamma, R')$
  - State $S' = S \times [0, 1]^\infty$
- Deterministic transition

$$(s, p_1, p_2, \ldots) \to (s', p_2, p_3, \ldots)$$

  - Where $s' = g(s, a, p_1)$
- D’ such that s ~ D and p_i’s are i.i.d
- Reward $R'(s, p_1, p_2, \ldots) = R(s)$
- Policies $\pi'(s, p_1, p_2, \ldots) = \pi(s)$

Policy Search Method

- Transformed M to deterministic M’
- Value of policy $V_{M'}(\pi) = \mathbf{E}_{s_0 \sim D'}[V^{\pi}_{M'}(s_0)]$
- Sample of m initial states (scenarios)

$$\{s_0^{(1)}, s_0^{(2)}, \ldots, s_0^{(m)}\}$$

- Approximate value $\hat{V}_{M'}(\pi) = \frac{1}{m}\sum_{i=1}^{m} V^{\pi}_{M'}(s_0^{(i)})]$

$$\hat{V}_{M'}(\pi) = \frac{1}{m}\sum_{i=1}^{m} R'(s_0^{(i)}) + \gamma R'(s_1^{(i)}) + \cdots + \gamma^H R'(s_H^{(i)})$$

- Like generating m Monte Carlo trajectories and taking their average reward but randomization is fixed in advance

Policy Search Method (2)

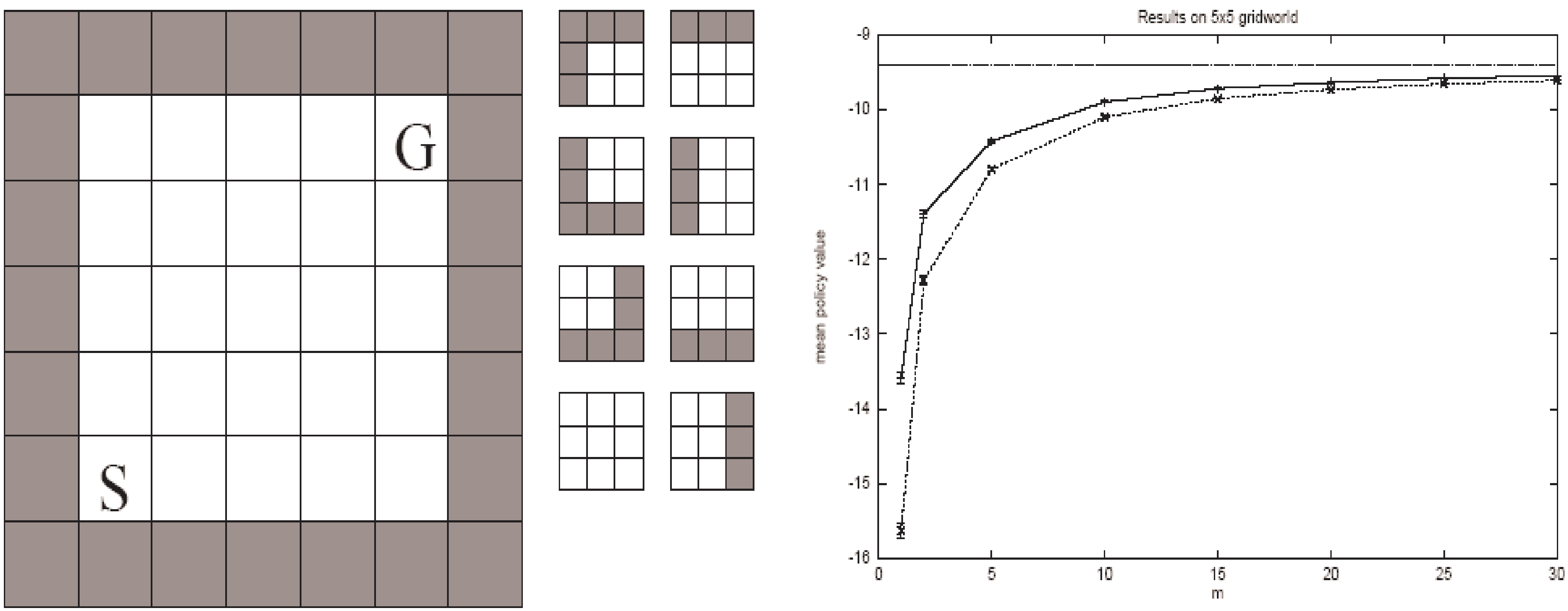
$$\widehat{V}_{M'}(\pi) = \frac{1}{m}\sum_{i=1}^{m} R'(s_0^{(i)}) + \gamma R'(s_1^{(i)}) + \cdots + \gamma^H R'(s_H^{(i)})$$

- Since objective function is deterministic can use standard optimization methods
- Gradient ascent methods
  - Smoothly parameterize family of policies

$$\Pi = \{\pi_\theta \mid \theta \in \mathbb{R}^l\}$$

  - If relevant quantities are differentiable, find gradient $\frac{d\widehat{V}_{M'}(\pi)}{d\theta}$
- Local maxima can be a problem

Example – Grid World



(a) (b)

Figure 1: (a) 5x5 gridworld, with the 8 observations. (b) PEGASUS results using the normal and complex deterministic simulative models. The topmost horizontal line shows the value of the best policy in Π; the solid curve is the mean policy value using the normal model; the lower curve is the mean policy value using the complex model. The (almost negligible) 1 s.e. bars are also plotted.

Example: Riding a Bicycle

- Randlov and Alstrom’s bicycle simulator
- Objective to ride to goal 1km away
- Action torque applied to handlebars and displacement of rider’s center of gravity
- Hand-picked 15 features of state but not fine-tuned
- Policy

$$\tau = \sigma(w_1 \cdot \vec{x})(\tau_{\mathrm{max}} - \tau_{\mathrm{min}}) + \tau_{\mathrm{min}}$$
$$\nu = \sigma(w_2 \cdot \vec{x})(\nu_{\mathrm{max}} - \nu_{\mathrm{min}}) + \nu_{\mathrm{min}}$$

- Gradient ascent
- Shaping rewards to reward progress towards the goal

Helicopter Flight

(a) (b)

Figure 1: (a) Autonomous helicopter. (b) Helicopter hovering under control of learned policy.

Helicopter

- Inertial Navigation System – accelerometers and gyroscopes
- Differential GPS and digital compass
- Kalman filter integrates sensor information
- State $s = (x, y, z, \phi, \theta, \omega, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\omega})$
- Actions
  - a_1, a_2 : longitudinal (front-back) and latitudinal (left-right) pitch control
  - a_3 : main rotor collective pitch control
  - a_4 : tail rotor collective pitch control

Model Identification

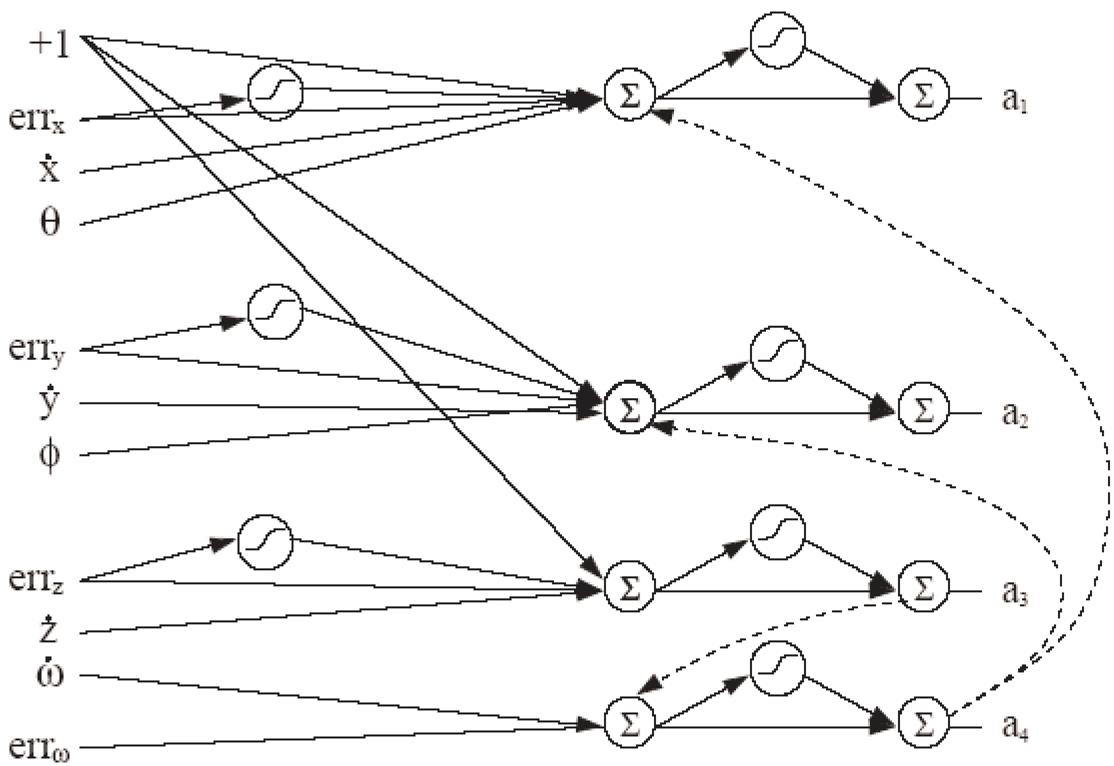
- Body coordinates

$$s^b = (\phi, \theta, \dot{x}^b, \dot{y}^b, \dot{z}^b, \dot{\phi}, \dot{\theta}, \dot{\omega})$$

- Locally-weighted linear regression with (s_t, a_t) as inputs and one-step differences $s_{t+1} - s_t$ as outputs
- Some parameters in the regression hard-coded
- Extra unobserved variables to model latency in response to controls
- Used human pilot flight data to fit and test the model

Policy Search

- PEGASUS
- Neural network for policy class



Policy Search (2)

- Quadratic state cost

$$R(s) = -(\alpha_x(x-x^*)^2+\alpha_y(y-y^*)^2+\alpha_z(z-z^*)^2+\alpha_{\dot{x}}\dot{x}^2+\alpha_{\dot{y}}\dot{y}^2+\alpha_{\dot{z}}\dot{z}^2+\alpha_\omega(\omega-\omega^*)^2)$$

- Weights scale terms to same order of magnitude
- Quadratic action cost

$$R(a) = -(\alpha_{a_1}a_1^2 + \alpha_{a_2}a_2^2 + \alpha_{a_3}a_3^2 + \alpha_{a_4}a_4^2)$$

- Overall reward is $R(s,a) = R(s) + R(a)$
- Both gradient ascent and random walk worked

Competition Maneuvers

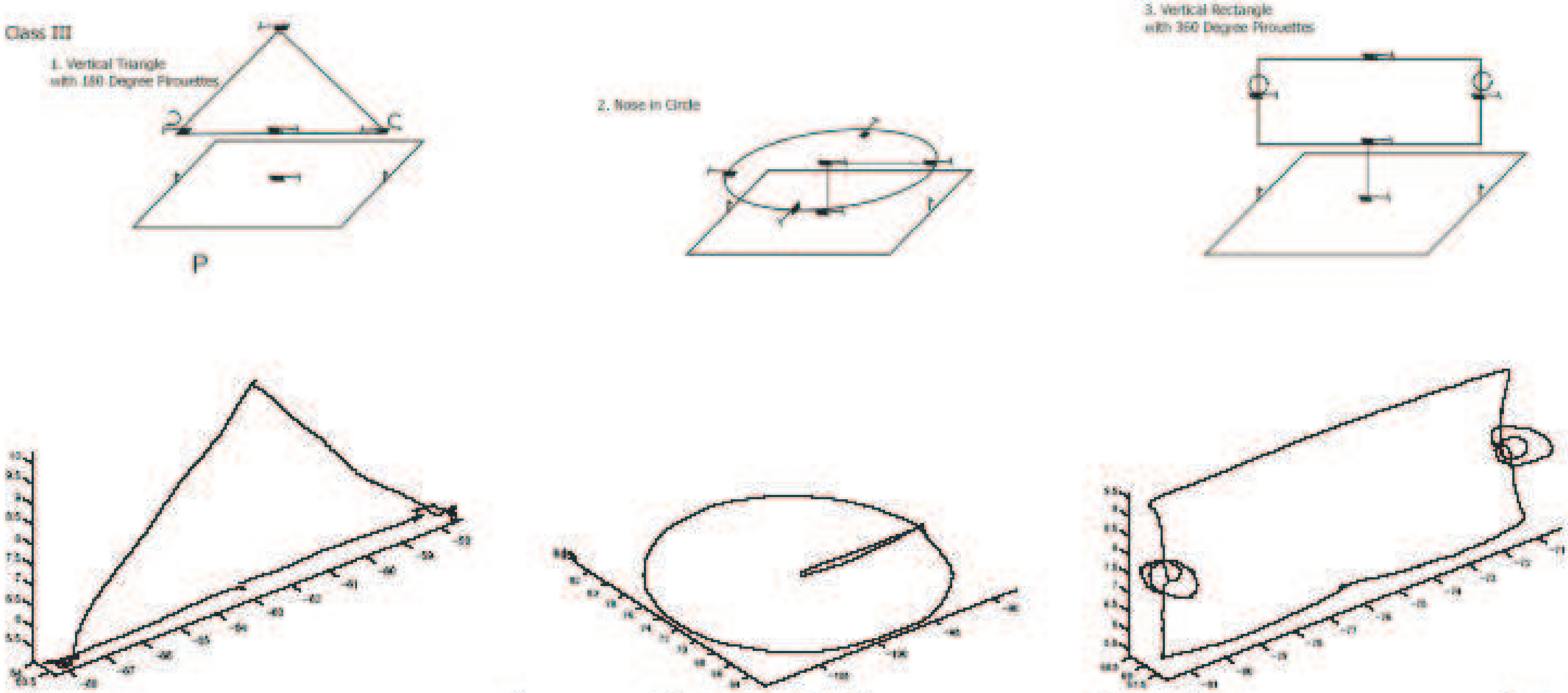


Figure 4: Top row: Maneuver diagrams from RC helicopter competition. [Images courtesy of the Academy of Model Aeronautics.] Bottom row: Actual trajectories flown using learned controller.

Competition Maneuvers (2)

- Vary over desired trajectory $(x^*, y^*, z^*, \omega^*)$
- Augment policy class to consider coupling between helicopter's subdynamics
- Use deviation from a projection onto desired trajectory $(x_p, y_p, z_p, \omega_p)$
- Use a potential function which increases along the trajectory