Maschinelles Lernen B

Lab Session 6
12. December 2004

Michael Pfeiffer
pfeiffer@igi.tugraz.at
<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>14th December</td>
<td>Submission for Problem Set 2</td>
</tr>
<tr>
<td>17th December</td>
<td>Presentation hour (PS2)</td>
</tr>
<tr>
<td></td>
<td><strong>13.15 - 14.45 (!)</strong></td>
</tr>
<tr>
<td>before Christmas</td>
<td>Problem Set 3 (Literature and Genetic Algorithms)</td>
</tr>
<tr>
<td>End of Semester</td>
<td>Deadline for Bonus Examples 5, 6</td>
</tr>
</tbody>
</table>
The Acrobot

- By Spong, 1994
- Apply torque at second link
- Not enough power to swing up immediately
- Questions:
  - state space
  - actions
  - rewards

Goal: Raise tip above line

Torque applied here

R. S. Sutton and A. G. Barto: Reinforcement Learning: An Introduction, modified by M. Pfeiffer
Acrobot

- **State space**
  - 4 continuous variables
  - 2 angles, 2 angular velocities
  - tilings for discretization

- **Actions**
  - positive / negative / no torque

- **Rewards**
  - -1 for all time steps
  - reach goal a.s.a.p.
  - $\gamma = 1$

Goal: Raise tip above line
Choice of Tilings

- Complete 4-dim. grid
- Hyperplanar stripes
  - slice only along one / two / three dimensions
  - e.g. both velocities
- Uneven partitioning
  - finer discretization at more important regions

- [Sutton 96]: Combination of all methods
  - 25,000 tiles
Acrobot Learning Curves for Sarsa(\(\lambda\))
Typical Acrobot Learned Behavior

R. S. Sutton and A. G. Barto: Reinforcement Learning: An Introduction, modified by M. Pfeiffer
Example 5*

• Learn a policy for the Acrobot
• Use physical model from Sutton / Barto
  – Chapter 11.3
• Decision
  – Which function approximation scheme?
    • Different methods make a big impact on performance
  – Which learning algorithm?
  – Learning Parameters

• Document success of learning
  – Points for this example according to performance
Policy Gradient RL

- Continuation of Lecture from Dec. 07
- Based on
  - R. Sutton, et.al.: *Policy Gradient Methods for RL with Function Approximation*
  - A. Ng, et.al.: *Inverted autonomous helicopter flight via RL*
  - Ng, Jordan: *PEGASUS: A policy search method for large MDPs and POMDPs*
  - Baxter, Bartlett: *Direct Gradient-based RL*

- Summary of a full-talk: *Reinforcement Learning for Motor Control*
Problem Description

- **Value-based RL** algorithms typically need a lot of training to adapt
  - Changing a Value may not immediately change the policy
  - Backup for previous actions, no change for future actions
  - Greedy Policies may change very **abruptly** (no smooth policy updates)

- True **online-learning** is often not possible!
Direct Reinforcement Learning

- **Direct Learning of Policy** without Learning of Value Functions (a.k.a. *Policy Search, Policy Gradient RL*)

- Policy is parameterized

- **Policy Gradient RL:**
  - Gradient Ascent Optimization of Parameter Vector representing the Policy
  - Optimization of Average Reward
Policy Gradient RL$^1$

- Policy is parameterized by $\theta$.
- Optimization of Average Reward

$$\eta(\theta) := \lim_{N \to \infty} \frac{1}{N} E_\theta \left[ \sum_{t=1}^{N} r(t) \right]$$

- Optimizing long-term average Reward is equivalent to optimizing discounted reward.
- Gradient Ascent on $\eta(\theta)$

$^1$Baxter, Bartlett: Direct Gradient-Based Reinforcement Learning (1999)
Gradient Ascent Algorithm

- Compute Gradient $\nabla \eta(\theta)$ w.r.t. $\theta$
- Take a step $\theta \leftarrow \theta + \gamma \nabla \eta(\theta)$

\[
\nabla \eta = d^T \nabla P \left[ I - P + ed^T \right]^{-1} r
\]

- Problems:
  - Stationary Distribution $d$ and Transition Probability-Matrix $P$ usually unknown
  - Inversion of huge Matrix

- Approximation of Gradient is necessary
Gradient Approximation

\[ \nabla \eta = \lim_{\beta \to 1} \nabla_{\beta} \eta = \lim_{\beta \to 1} d^T \cdot \nabla P \cdot V_{\beta} \]

- \( V_{\beta} \) ... Discounted State-Values
- \( \beta \in [0, 1) \) ... Discount Factor, Bias-Variance Trade-Off

- \( \beta \) close to 1:
  - good Approximation of Gradient
  - Large Variance in Estimates of \( \nabla_{\beta} \eta \nabla_{\beta} \eta \)
  - Must be set by User in advance
GPOMDP Algorithm

- Estimate Gradient from a single sample Path of the POMDP

1. \( z_0 = 0, \Delta_0 = 0 \)
2. FORALL observations \( y_t \), controls \( u_t \) and subsequent rewards \( r(i_{t+1}) \)
3. \( z_{t+1} = \beta z_t + \nabla \pi_u (\theta, y_t) \)
\[ \pi_u (\theta, y_t) \]
4. \( \Delta_{t+1} = \Delta_t + \frac{1}{t+1} \left[ r(i_{t+1}) z_{t+1} - \Delta_t \right] \)
5. END
Explanation of GPOMDP

• $\Delta_t$ computes average of $r_{i(t)} \cdot z_t$
  - Proof in [Baxter, Bartlett]

• $\lim_{t \to \infty} \Delta_t = \nabla_\beta \eta$
  - Convergence to Gradient Estimate
  - Longer GPOMDP runs needed for exact estimation (Variance depends on $\beta$)
Experimental Results

- Comparing real and estimated Gradient in 3-state MDP
  - Small $\beta$
    - Greater bias
  - Large $\beta$
    - Later convergence
• Estimation of Gradient with GPOMDP is computationally expensive
  - Fixed search length is therefore inefficient

• Better: Do a line search in the direction of the Gradient Estimate: GSEARCH
Idea of GSEARCH

- Bracket the Maximum in direction $\theta^*$ between two points $\theta_1, \theta_2$
  - $\text{GRAD}(\theta_1) \cdot \theta^* > 0$, $\text{GRAD}(\theta_2) \cdot \theta^* < 0$
  - Maximum is in $[\theta_1, \theta_2]$
  - Quadratic Interpolation to find Maximum
CONJ POMDP

• Policy-Gradient Algorithm
  – Uses GPOMDP for Gradient Estimation
  – Uses GSEARCH for finding Maximum in Gradient Direction
  – Continues until Changes fall below threshold
  – Trains Parameters for Controllers
  – Involves many Simulated Iterations of Markov Chain for Gradient Estimations
Combining direct and value-function based RL

Theorem:
If the value-function parameterization is compatible with the policy parameterization, then the true policy gradient can be estimated, the variance of the estimation can be controlled by a reinforcement baseline, and policy iteration converges to a locally optimal policy.

Significance:
- Shows first convergence proof for policy iteration with function approximation.

1 Sutton, McAllester, Singh, Mansour: Policy Gradient Methods for RL with Function Approximation
Experiments and Results

- Mountainous Puck World
  - Similar to Mountain Car
- Navigate a Puck out of a valley to a plateau
  - Not enough power to directly climb the hill
- Train Neural-Network controllers
- CONJ POMDP
  - 1 Mio. Runs for GPOMDP
Helicopter Flying with RL$^{1,2}$

- Autonomously learning to fly a real unmanned Helicopter
  - 70,000 $ vehicle (Exploration is catastrophic!)
- Learned Dynamics Model from Observation of Human Pilot
- PEGASUS Policy-Gradient RL in Simulator
- Learned to Hover on Maiden-flight
  - More stable than Human
- Learned to fly complex Maneuvers accurately

$^1$ Ng, et.al.: Autonomous Helicopter Flight via RL
$^2$ Ng, et.al.: Autonomous inverted Helicopter Flight via Reinforcement Learning
Helicopter Dynamics

• State space
  - 12 continuous var.
  - (x,y,z) world position
  - roll, pitch, yaw angle
  - 3 velocities and 3 angular velocities

• 4-dimensional actions
  - 2 rotor-plane pitch
  - Rotor blade tilt
  - Tail rotor tilt

• Actions are selected every 20 ms
Learning the Helicopter Model

- Human pilot flies helicopter, data is logged
  - 391s training data
  - reduced to 8 dimensions (position can be estimated from velocities)
- Learn transition probabilities $P(s_{t+1}|s_t, a_t)$
  - supervised learning with linear regression
- Added Gaussian noise for stochastic model
- Implemented a simulator for model validation
Reinforcement Learning

• Direct RL of parameterized policy
  – neural network approximation of 4 controls
• Quadratic Reward Function
  – punishment for deviation of desired position and orientation
• Maximize
  \[
  U(\pi) = E\left[ \sum_{t=0}^{\infty} \gamma^t R(s_t) \bigg| \pi \right]
  \]
PEGASUS Learning Algorithm

• Estimation of $U(\pi)$
  - simulate state sequence given $\pi$
  - estimate $U$ from sample of states

• Optimization of policy $\pi$
  - Greedy Hillclimbing search for best policy
  - Problem: random sampling introduces big variation in estimation of $U$
  - e.g. the same policy may be assigned different values at different time points

• PEGASUS:
  - fix sequence of random numbers $\Rightarrow$ deterministic optimization problem
Hovering

- Learned to hover on maiden flight
- Computer is more stable than trained pilot
Inverted Flight

- Learned in 4 days!
  - from data collection to flight experiment
- Stable inverted flight controller
  - sustained position
- Very difficult for humans
Flying Maneuvers

- Additional complication
  - how to enforce following trajectories?
  - punish distance from *projected* point on trajectory
  - Reward shaping with potential functions