Machine Learning
Appendix to lecture 16

Reinforcement Learning for Motor Control

Michael Pfeiffer

19. 01. 2004

pfeiffer@igi.tugraz.at

Agenda

- Motor Control
- Specific Problems of Motor Control
- Reinforcement Learning (RL)
- Survey of Advanced RL Techniques
- Existing Results
- Open Research Questions

What is Motor Control?

- Controlling the <u>Movement</u> of Objects
- Biological: Understanding how the brain controls the movement of limbs
- Engineering: Control of Robots (especially humanoid)
- In this talk: Emphasis on <u>Robot Control</u>

Definition: Motor Control[1]

- Control of a nonlinear, unreliable System
- Monitoring of States with slow, low-quality Sensors
- Selection of appropriate Actions
- Translation of Sensory Input to Motor Output
- Monitoring of Movement to ensure Accuracy

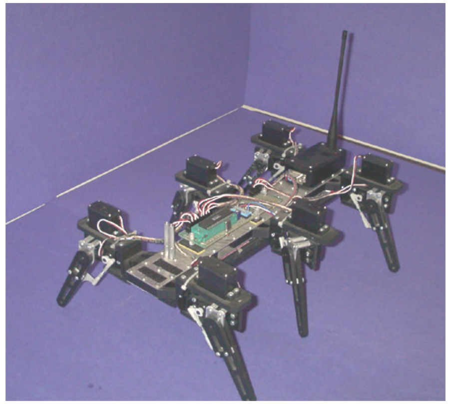
[1] R.C. Miall: Handbook of Brain Theory and Neural Networks, 2nd Ed. (2003)

Motor Learning

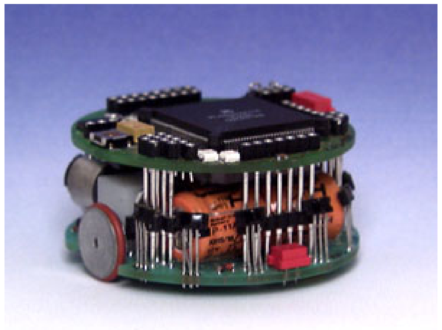
- Adaptive Control
  - <u>Monitoring</u> Performance of Controller
  - <u>Adapting</u> the Behaviour of the Controller
  - To achieve <u>better Performance</u> and <u>compensate</u> gradual Changes in the Environment

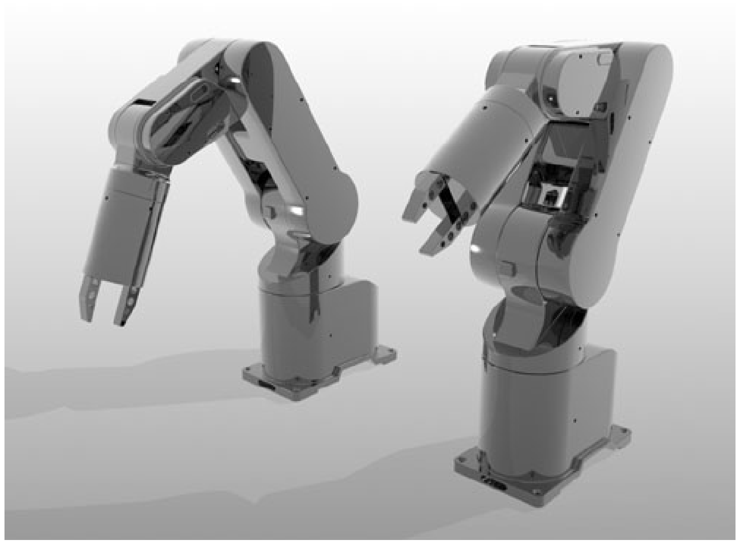
- Formulation:
  - $u = \pi(x, t, \alpha)$
  - u … Coninuous control vector
  - x … Continuous state vector
  - t … Time
  - α … Problem Specific Parameters

Interesting Robots



No





Interesting Learning Tasks

- Unsupervised Motor Learning
  - Learning Movements from Experience
- Supervised Motor Learning
  - Learning from Demonstration
- Combined Supervised and Unsupervised Learning
- Not covered: Analytical and Heuristic Solutions
  - Dynamical Systems
  - Fuzzy Controllers

Agenda

- Motor Control
- Specific Problems of Motor Control
- Reinforcement Learning (RL)
- Survey of Advanced RL Techniques
- Existing Results
- Open Research Questions

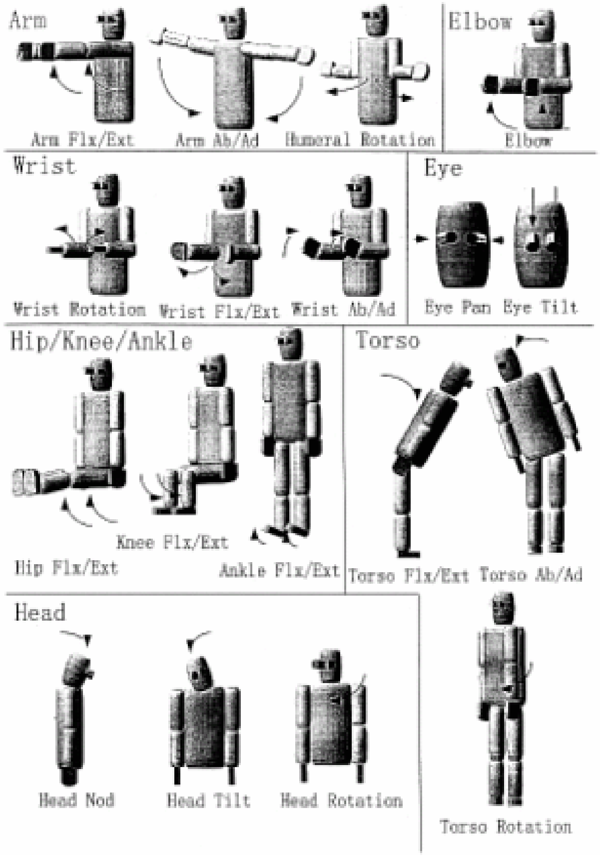
Non-linear Dynamics

- Dynamics of Motor Control Problems
  - Systems of Non-linear Differential Equations in high-dimensional State Space
- Instability of Solutions
- Analytical Solution therefore is very difficult (if not impossible) to achieve

- Learning is necessary!

Degrees of Freedom

- Every joint can be controlled separately
- Huge, continuous Action Space
  - e.g. 30 DOFs, 3 possible commands per DOF:
  - $3^{30} > 10^{14}$ possible actions in every state
- Redundancy:
  - More degrees of freedom than needed
  - Different ways to achieve a trajectory
  - Which one is optimal?
  - Optimal Policy is robust to Noise



Online Adaptation

- Unknown Environments
  - Difficult Terrain, etc.
- Noisy Sensors and Actuators
  - Commanded Force is not always the Acutal Force
- Reflex Response to strong Pertubations
  - Avoid damage to Robots

Learning Time

- Learning on real Robots is very time-consuming
- Many long training runs can damage the Robot
- Simulations cannot fully overcome these problems
  - Lack of physical Realism

- Learning „from Scratch“ takes too long

Other Issues

- Continuous Time, State and Actions
- Hierarchy of Behaviours
- Coordination of Movements
- Learning of World Models
- And many more...

Main Goals of this Talk

- Present possible Solutions for
  - Learning in Continuous Environments
  - Reducing Learning Time
  - Online Adaptation
  - Incorporating A-priori Knowledge
- Showing that Reinforcement Learning is a suitable Tool for Motor Learning

Agenda

- Motor Control
- Specific Problems of Motor Control
- Reinforcement Learning (RL)
- Survey of Advanced RL Techniques
- Existing Results
- Open Research Questions

Reinforcement Learning (RL)

- Learning through Interaction with Environment

- Agent is in State *s*
- Agent executes Action *a*
- Agent receives a *Reward r(s,a)* from the environment

- Goal: Maximize *long-term discounted Reward*

Basic RL Definitions

- Value Function: $V^{\pi}(s) = E_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s \right]$
- Action-Value Function (Q-Function):

$$Q^{\pi}(s,a) = E_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s, a_t = a \right]$$

- Bellman – Equation:

$$Q^{*}(s,a) = E\left[r_{t+1} + \gamma \cdot \max_{a'} Q^{*}(s_{t+1}, a') \middle| s_t = s, a_t = a \right]$$

Value-Based RL

- Policy Iteration:
  - Start with random policy π_0
  - Estimate Value-Function of π_i
  - Improve $\pi_i \rightarrow \pi_{i+1}$ by making it greedy w.r.t. to the learned value function
  - Exploration: Try out random actions to explore the state-space
  - Repeat until Convergence

- Learning Algorithms:
  - Q-Learning (off-policy), SARSA (on-policy)
  - Actor-Critic Methods, etc.

Temporal Difference Learning

- TD error: $\delta_t = r_{t+1} + \gamma \cdot V(s_{t+1}) - V(s_t)$
- Evaluation of Action:
  - Positive TD-Error: Reinforce Action
  - Negative TD-Error: Punish Action
- TD(λ): update value of previous action with future rewards (TD-errors)
- Eligibility Traces: Decay exponentially with λ
  - $e(s) \leftarrow \gamma \cdot \lambda \cdot e(s)$

Problems of Standard-RL

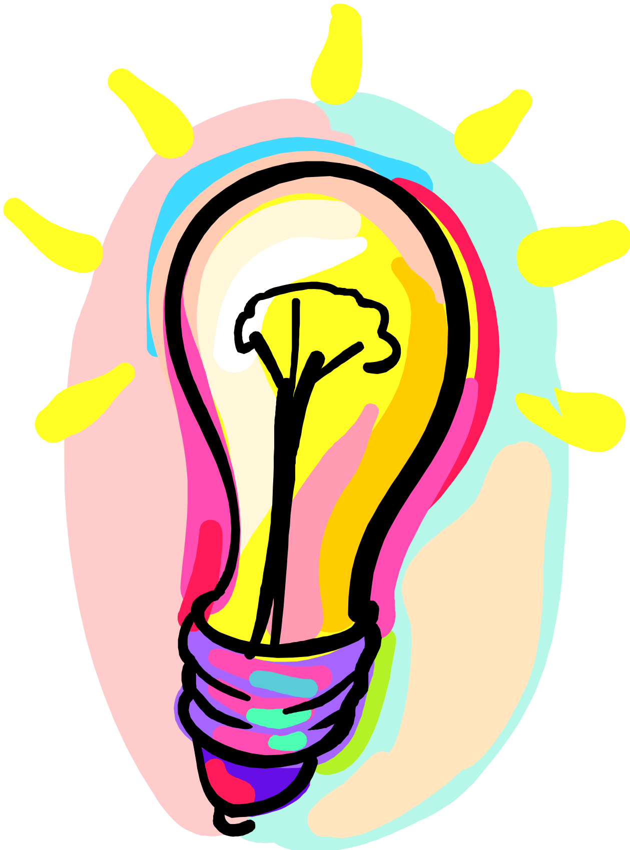
- Markov Property violated
- Discrete States, Actions and Time
- Learning from Scratch
- (Too) Many Training Episodes needed
- Convergence

Agenda

- Motor Control
- Specific Problems of Motor Control
- Reinforcement Learning (RL)
- Survey of Advanced RL Techniques
- Existing Results
- Open Research Questions

Structure of This Chapter

- Main Problems of Motor Control
- Possible RL Solutions
- Successful Applications



Problem 1

Learning in Continuous Environments

Standard Approaches for Continuous State Spaces

- Discretization of State Space
  - Coarse Coding, Tile Codings, RBF, ...
- Function Approximation
  - Linear Functions
  - Artificial Neural Networks, etc.

Function Approximation in RL

- Represent State by a finite number of Features (Observations)
- Represent Q-Function as a parameterized function of these features
  - (Parameter-Vector θ)
- Learn optimal parameter-vector θ^* with Gradient Descent Optimization at each time step

Problems of Value Function Approximation

- No <u>Convergence</u> Proofs
  - Exception: Linear Approximators
- <u>Instabilities</u> in Approximation
  - „Forgetting“ of Policies
- Very high <u>Learning Time</u>
- Still it works in many Environments
  - TD-Gammon (Neural Network Approximator)

Continuous TD-Learning[1]

- Continuous State x, Continuous Actions u
- System Dynamics: $\dot{x} = f(x,u)$
- Policy π produces trajectory x(t)

$$\forall t \geq t_0 \qquad \dot{x} = f(x, \pi(x))$$

$$x(t_0) = x_0$$

- Value Function:

$$V^{\pi}(x_0) = \int_{t=t_0}^{\infty} e^{-\frac{(t-t_0)}{\tau}} r\big(x(t), \pi(x(t))\big) dt$$

[1] *K. Doya*: Reinforcement Learning in Continuous Time and Space, Neural Computation, 12(1), 219-245 (2000)

Optimality Principle

- Hamilton-Jacobi-Bellman (HJB) Equation

$$\frac{1}{\tau}V^*(x(t)) = \max_{u(t)\in U}\left[r(x(t),u(t)) + \frac{\partial V^*}{\partial x} f(x(t),u(t)) \right]$$

  - Optimal Policy must satisfy this equation

- Approximate Value Function by Parameter Vector θ
  - Find optimal θ

Continuous TD-Error

- Self-Consistency Condition:

$$\dot{V}(x(t)) = \dot{V}(t) = \frac{1}{\tau}V(t) - r(t)$$

- Continuous TD-Error:

$$\delta(t) = r(t) - \frac{1}{\tau}V(t) + \dot{V}(t)$$

- Learning: Adjust Prediction of V to decrease TD-Error (inconsistency)

Continuous TD(λ) - Algorithm

- Integration of Ordinary Diff. Equation

$$\dot{\theta} = \eta \cdot \delta(t) \cdot e(t)$$

$$\dot{e}(t) = -\left(\frac{1}{\kappa}\right)e(t) + \frac{\partial V(x(t), \theta)}{\partial \theta}$$

$$\dot{x} = f(x, \pi(x))$$

  - η … Learning Rate
  - κ … $0 < \kappa \leq \tau$, Related to λ

Policy Improvement

- Exploration: Episodes start from random initial state
- Actor-Critic:
  - Approximate Policy through another Parameter Vector θ^A
  - Use TD-Error for Update of Policy
- Choose Greedy Action w.r.t. $V(x, \theta)$
  - Continuous Optimization Problem
  - [Doya] describes more approaches

Relation to Discrete-Time RL

- Implementation with Finite Time Step

- Equivalent Algorithms can be found to
  - Residual Gradient
  - TD(0)
  - TD(λ)

Problems with this Method

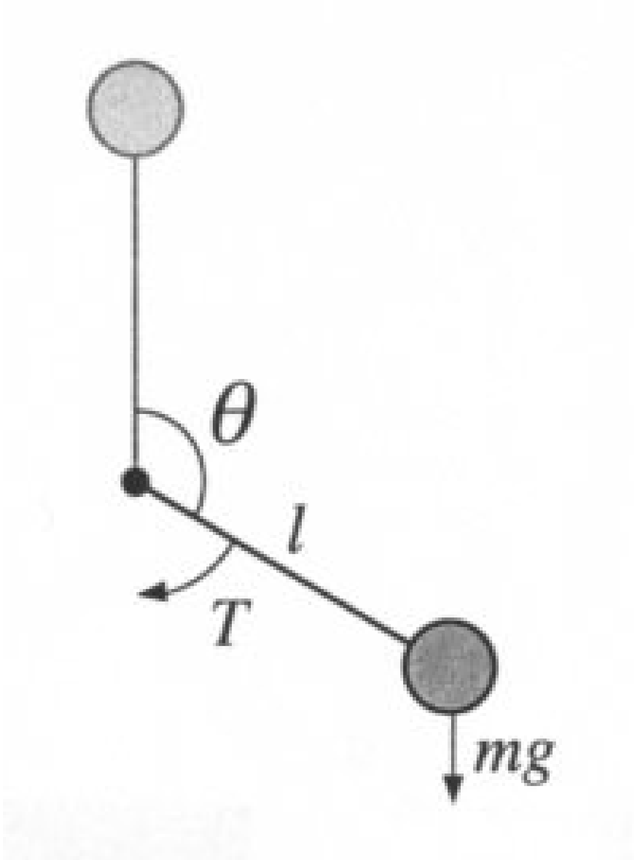
- Convergence is not guaranteed
  - Only for Discretized State-Space
  - Not with Function Approximation

- Instability of Policies

- A lot of Training Data is required

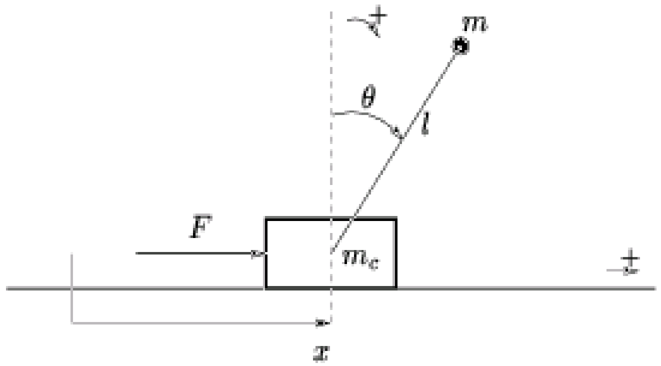
Experiments (1)

- Pendulum Up-Swing with limited Torque
  - Swing Pendulum to upright position
  - Not enough torque to directly reach goal
  - Five times faster than discrete TD



Experiments (2)

- Cart – Pole Swing-Up
  - Similar to Pole-Balancing Task
  - Pole has to be swung up from arbitrary angle and balanced
  - Using Continuous Eligibility Traces makes learning three-times faster than pure Actor-Critic algorithm



Problem 2

Reduction of Learning Time

Presented Here

- Hierarchical Reinforcement Learning
  - Module-based RL
- Model-Based Reinforcement Learning
  - Dyna-Q
  - Prioritized Sweeping
- Incorporation of prior Knowledge
  - Presented separately

1. Hierarchical RL

- Divide and Conquer Principle
  - Bring Structure into Learning Task
  - Movement Primitives

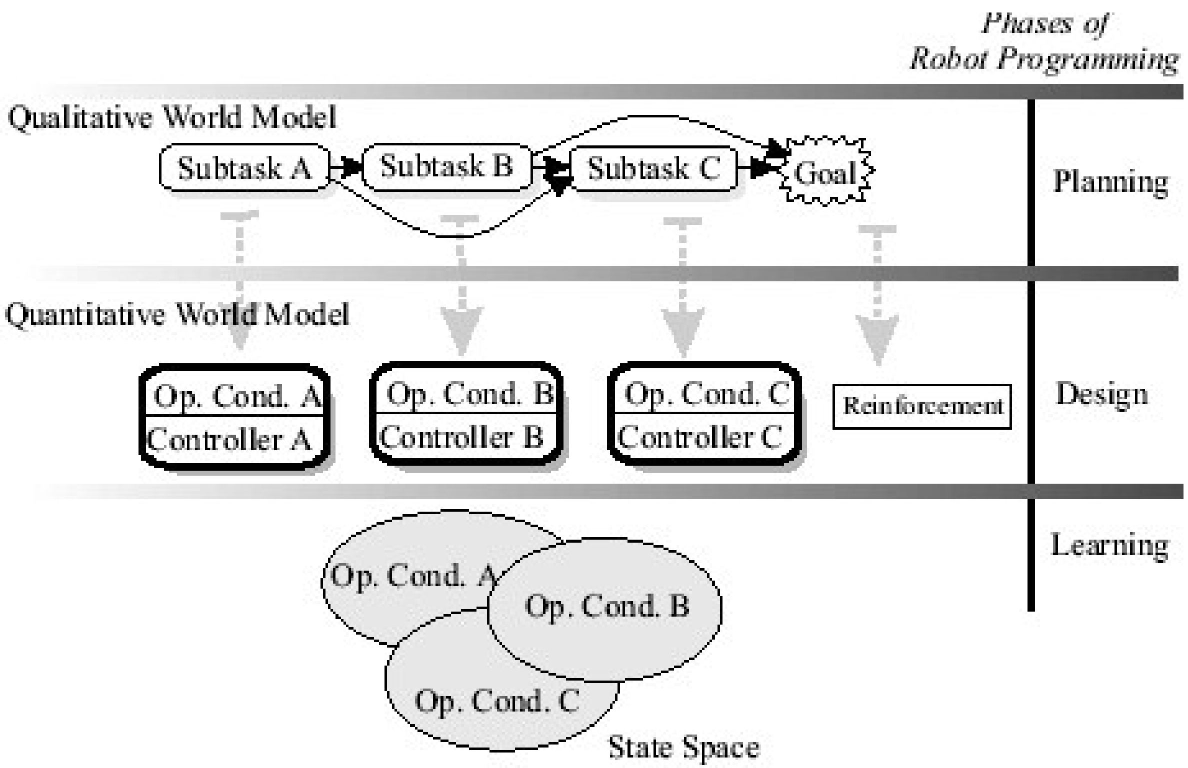
- Many Standard Techniques exist
  - SMDP Options [Sutton]
  - Feudal Learning [Dayan]
  - MAXQ [Dietterich]
  - Hierarchy of Abstract Machines [Parr]
  - **Module-based RL** [Kalmár]

Module-based RL

- Behaviour-based Robotics
  - Multiple Controllers to achieve Sub-Goals
  - Gating / Switching Function decides when to activate which Behaviour
  - Simplifies Design of Controllers
- Module-based Reinforcement Learning[1]
  - Learn Switching of Behaviours via RL
  - Behaviours can be learned or hard-coded

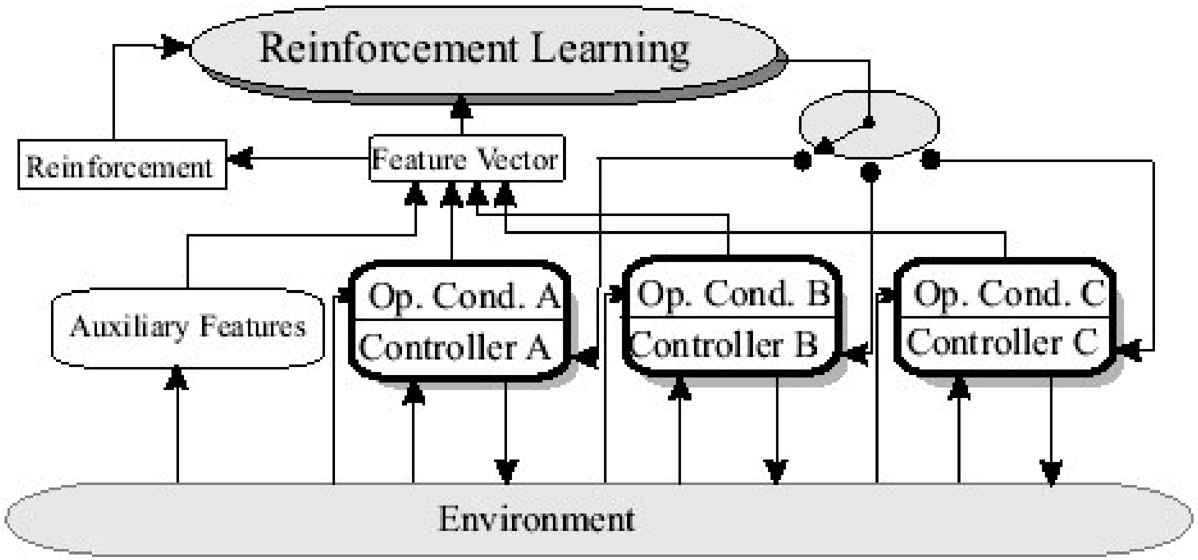
[1] *Kalmár, Szepesvári, Lörincz:* Module-based RL: Experiments with a real robot. Machine Learning 31, 1998

Module-based RL



- Planning Step introduces prior Knowledge
- Operation Conditions: When can modules be invoked?

Module-based RL



- RL learns Switching Function to resolve Ambiguities
  - Inverse Approach (learning Modules) also possible

Experiments and Results

- Complex Planning Task with Khepera
  - RL starts from scratch
  - Module-based RL comes close to hand-crafted controller after 50 Trials
  - Module-based RL outperforms other RL techniques

Other Hierarchical Approaches

- Options or Macro Actions

- MAXQ:Policies may recursively invoke sub-policies (or primitive actions)

- Hierarchy of Abstract Machines:
  - Limit the space of possible policies
  - Set of finite-state machines
  - Machines may call each other recursively

2. Model-based RL

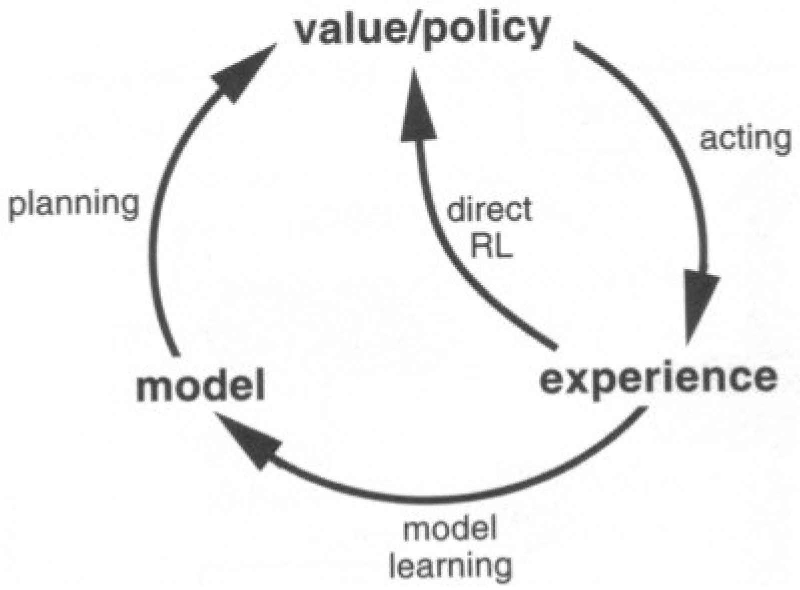
- Simultaneous Learning of a Policy and a World Model to speed-up Learning
- Learning of Transition Function in MDP
- Allows Planning during Learning

- Approaches:
  - Dyna-Q
  - Prioritized Sweeping

Planning and Learning

- Experience improves both Policy and Model

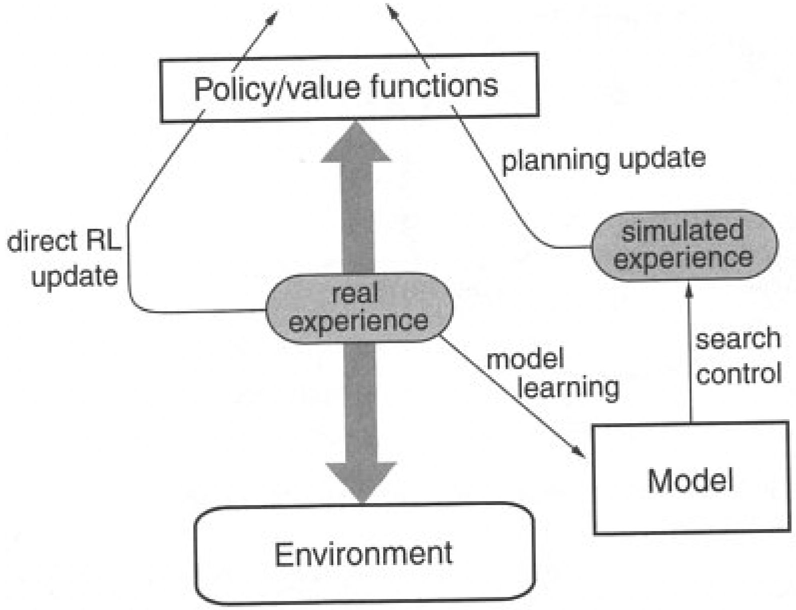
- Indirect RL:
  - Improvement of Model may also improve the Policy



Dyna-Q

- Execute a in s
- Observe s', r
  - Model(s, a) = (s', r)
  - (deterministic World)

- Make N offline update steps to improve Q-function



Prioritized Sweeping

- Planning is more useful for states where a big change in the Q-Value is expected
  - e.g. predecessor states to goal states
- Keep a Priority Queue of State-Action Pairs, sorted by the predicted TD-Error
  - Update Q-Value of highest-priority Pair
  - Insert all predecessor pairs into Queue, according to new expected TD-Error
- Problem: Mostly suitable for discrete Worlds

Pros and Cons of Model-based RL

- Dyna-Q and Prioritized Sweeping converge much faster (in Toy Tasks)

- Extension to Stochastic Worlds is possible

- Extension to Continuous Worlds is difficult for Prioritized Sweeping
  - No available results

- Not necessary in well-known Environments
  - Error-free Planning and Heuristic Search

Problem 3

Online Adaptation

Problem Description

- Environment and/or Robot Characteristics are only partially known
  - Unreliable Models for Prediction (Inverse Kinematics and Dynamics)
- 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)

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

Definitions

- Definitions in POMDP[1]:
  - State $i \in \{1, \ldots, n\}$
  - Observation $y=\nu(i) \in \{1, \ldots, M\}$
  - Controls $u \in \{1, \ldots, N\}$
  - State Transition Matrix $P(u) = [p_{ij}(u)]$
  - Stochastic, differentiable Policy $\mu(\theta,y)$
  - μ generates Markov Chain with Transition Matrix $P(\theta) = [p_{ij}(\theta)]$
  - $p_{ij}(\theta) = E_{\nu(i)}[y]\, E_{\mu(\theta,y)}\, p_{ij}(u)$
  - Stationary distribution π: $\pi^T(\theta)\, P(\theta) = \pi^T(\theta)$

[1] POMDP = Partially Observeable Markov Decision Process

Policy Gradient RL[1]

- Policy is parameterized by θ
- Optimization of Average Reward

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

  - 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. θ
- Take a step $\theta \leftarrow \theta + \gamma\, \nabla\eta(\theta)$

$$\nabla\eta = \pi^T \nabla P \left[I - P + e\pi^T \right]^{-1} r$$

- Problems:
  - Stationary Distribution π of MDP and Transition Probabilities usually <u>unknown</u>
  - <u>Inversion</u> of huge Matrix
- <u>Approximation</u> of Gradient is necessary

Gradient Approximation

$$\nabla\eta = \lim_{\beta\to 1} \nabla_{\beta}\eta = \lim_{\beta\to 1} \pi^{T} \cdot \nabla P \cdot V_{\beta}$$

- V_β ... Discounted State-Values
- $\beta \in [0, 1)$ ... Discount Factor, Bias-Variance Trade-Off

- β close to 1:
  - good Approximation of Gradient
  - Large Variance in Estimates of $\nabla_\beta\eta$
  - Must be set by User in advance

GPOMDP Algorithm

- Estimate Gradient from a single sample Path of the POMDP

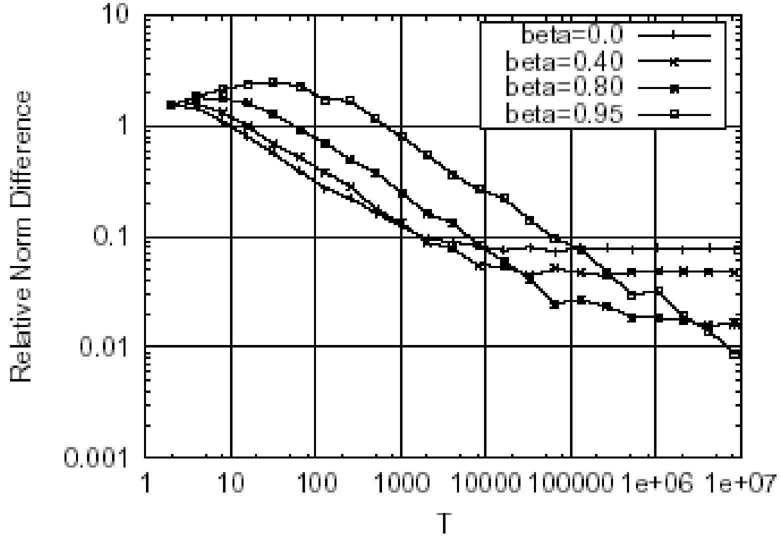
1. $z_0 = 0,\ \Delta_0 = 0$
2. FORALL observations y_t, controls u_t and subsequent rewards $r(i_{t+1})$
3. $$z_{t+1} = \beta z_t + \frac{\nabla \mu_u(\theta, y_t)}{\mu_u(\theta, y_t)}$$
4. $$\Delta_{t+1} = \Delta_t + \tfrac{1}{t+1}\left[r(i_{t+1})\, z_{t+1} - \Delta_t\right]$$
5. END

Explanation of GPOMDP

- Δ_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 β)

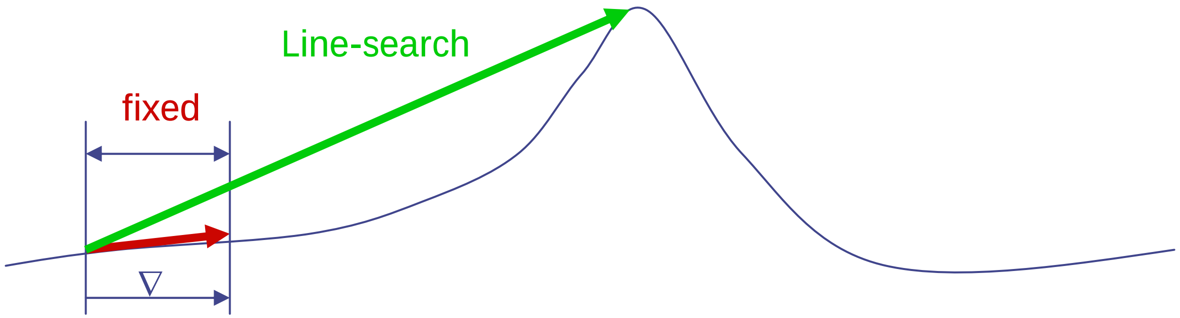
Experimental Results

- Comparing real and estimated Gradient in 3-state MDP
- Small β
  - Greater bias
- Large β
  - Later convergence



GSEARCH

- 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 θ^* between two points θ_1, θ_2
  - $GRAD(\theta_1) \cdot \theta^* > 0$, $GRAD(\theta_2)) \cdot \theta^* < 0$
  - Maximum is in $[\theta_1, \theta_2]$
  - Quadratic Interpolation to find Maximum

CONJPOMDP

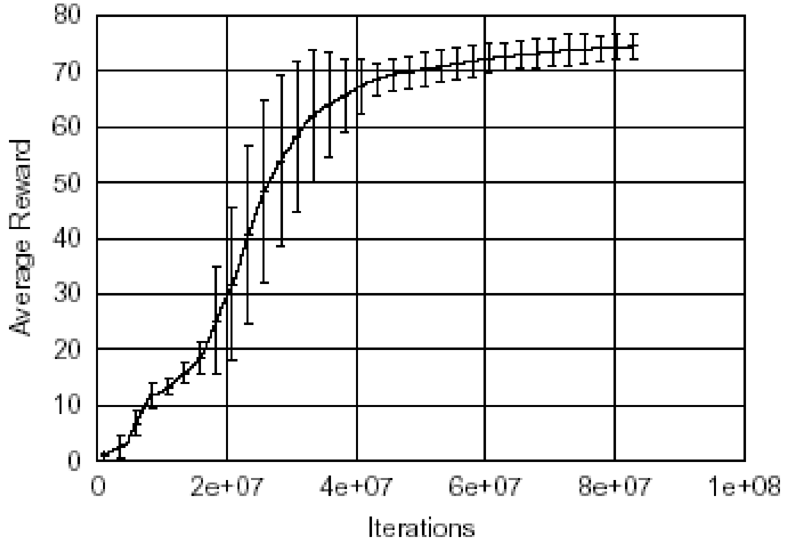
- Policy-Gradient Algorithm
  - Uses GPOMDP for Gradient Estimation
  - Uses GSEARCH for finding Maximum in Gradient Direction
  - Continues until Changes fall below threshold
  - Trains Parameters for Controllers
  - Involves many Simulated Iterations of Markov Chain for Gradient Estimations

OLPOMDP

- Directly adjust Parameter Vector during Running Time
- Same Algorithm as GPOMDP, only actions are directly executed and θ is immediately updated
- No convergence Results yet

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
- CONJPOMDP
  - 1 Mio. Runs for GPOMDP



VAPS [Baird, Moore][1]

- Value And Policy Search
- Combination of both Algorithm types
  - Allows to define Error function *e,* dependent on parameter vector θ
  - *e* determines Update rule (e.g. SARSA, Q-learning, REINFORCE (policy-search)…)
- Gradient Ascent Optimization
  - Guaranteed (local) Convergence for all function approximators

[1] Baird, Moore: Gradient Descent for General RL (1999)

Policy Gradient Theorem[1]

- **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

Gradient Estimation with Observeable Input Noise[1]

- Assume that control Noise can be measured
- Measure Eligibility of each Sample
  - $E(h) = \nabla_\pi \log P_\pi(h)$
  - How much will log-likelihood of drawing sample h change due to a change in π?
  - F(h) … Evaluation of History (Sum of Rewards)

- Adjust π to make High-scoring Histories more likely

$$\nabla_\theta \approx \frac{1}{N}\sum_{i=1}^{N} E(h_i)F(h_i)$$

[1] Lawrence, Cowan, Russell:Efficient Gradient Estimation for Motor Control Learning

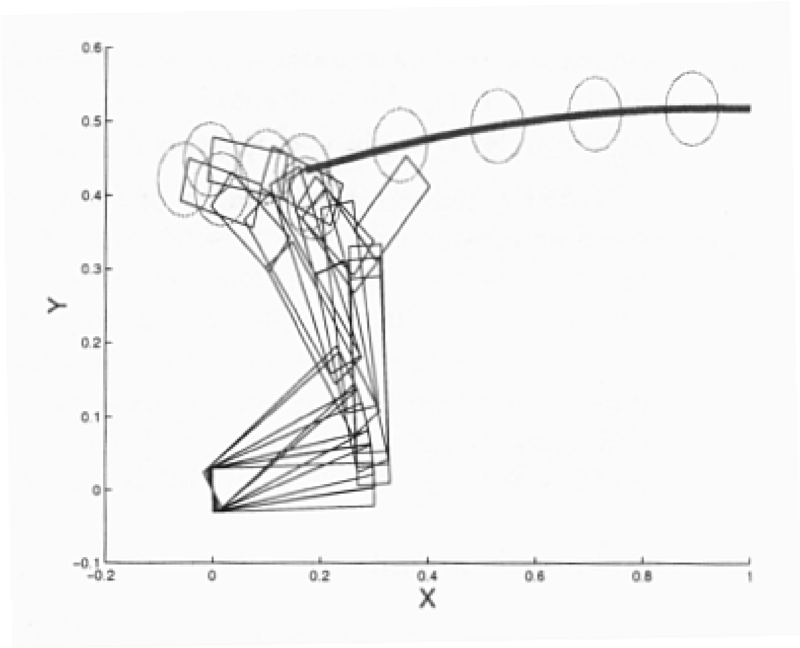
PEGASUS Algorithm[1]

- Reduce variance of gradient estimators by controlling noise
- In a simulator: Control the <u>random-number generator</u>

[1] Ng, Jordan: PEGASUS: A policy search method for large MDPs and POMDPs

Successful Application

- Dart Throwing
  - Simulated 3-link Arm
  - 1 DOF per joint
  - Goal: hit bullseye
  - Parameters: Positions of via-points for joints
  - Injection of Noise made result look more natural
  - Reliably hit near-center after 10 trials and 100 simulated gradient-estimations per step



Experiments (2)[1]

- 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, Kim, Jordan, Sastry: Autonomous Helicopter Flight via RL (unpublished draft)

Problem 4

Incorporation of Prior Knowledge

„Dilemma“ of RL

- Completely unsupervised learning from scratch can work with RL
- Some solutions may surprise humans

- Result for Real-world Tasks:
  - Everybody tries completely unsupervised learning
  - RL takes too long to find even the simplest solutions without prior knowledge
  - Makes people think: „RL does not work“
  - RL with some Guidance could work perfectly

Human and Animal Learning

- Learning without prior knowledge almost never occurs in nature!

- Genetic Information:
  - Young animals can walk, even without guidance from their parents
- Training:
  - Humans need Demonstration to learn complicated movements (e.g. Golf, Tennis, Skiing, …)
  - Still they improve through experience

Prior Knowledge in RL

- Dense Rewards
  - Danger of local Optimalities
- Shaping the Initial Value Functions
  - By Heuristics or by Observation
- Exploration Strategy
  - Visit interesting parts first
  - Learning from Easy Missions [Asada]

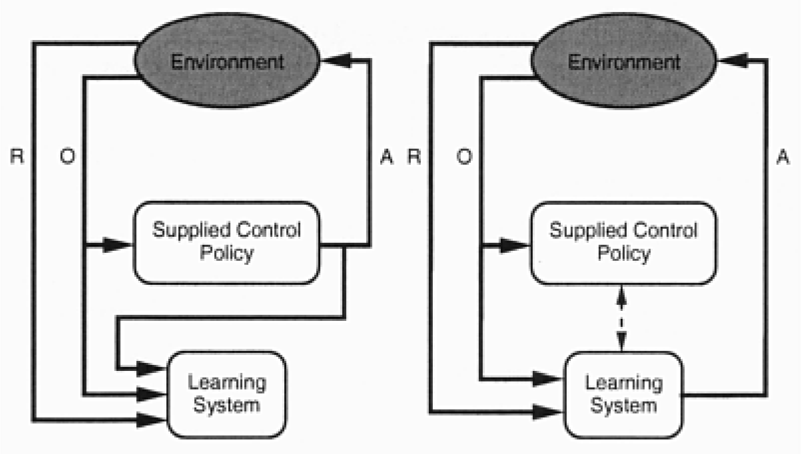
Off-policy Passive Learning[1]

- Sparse Rewards: mostly zero
  - Learning time dominated by initial „blind Search“ for sparse sources of Reward
- Off-policy Methods (e.g. Q-Learning)
  - Can learn passively from observation
- Initial Demonstration from advanced (human or coded) Controller
  - Policy is learned as if it had selected the actions supplied by the external controller

[1] Smart, Kaelbling: Effective RL for Mobile Robots

Advantage of Passive Learning

- No complete understanding of system dynamics and sensors necessary
- Only sample trajectories required
- Split in 2 Phases:
  - Supervised Training to start with sesible policy
  - Use of supplied controller in Phase 2 as advisor



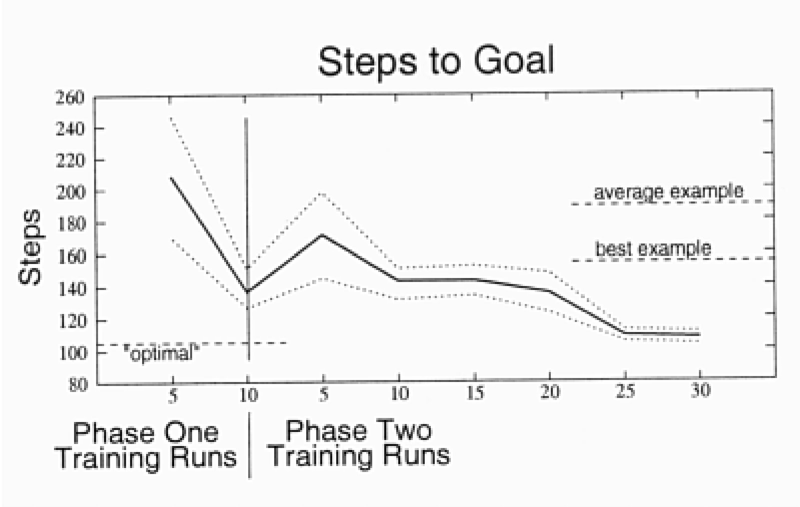
(a) Phase 1 (b) Phase 2

Experiments

- Real 2-wheeled Robot
- 2 Tasks
  - Corridor Following
  - Obstacle Avoidance
- 2 Supplied Controllers
  - Hard-coded
  - Human demonstration

Results

- Performance degrades after Supervision ends
  - Quickly recovers
  - Finds even better policy than best demonstration
- Human demonstrations are better suited
  - More Noise
  - No optimal demonstrations necessary
- Without Knowledge
  - Finding the goal once takes longer than whole training procedure



Performance in Corridor-Following Task with Human Guidance

RL from Demonstration[1]

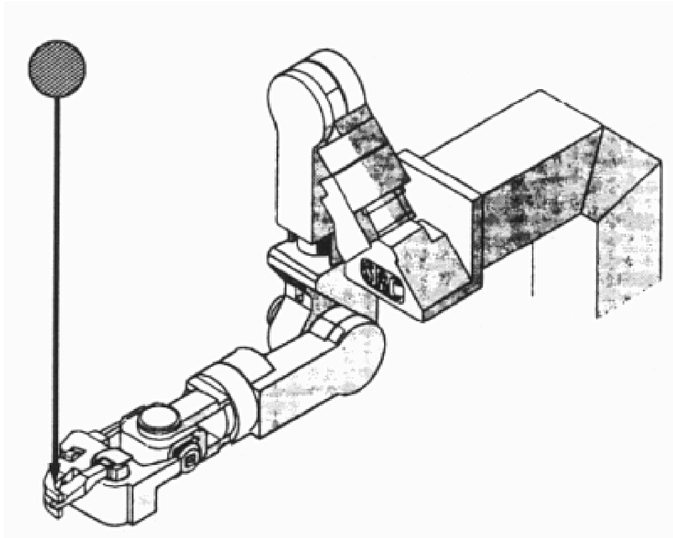
- Priming of
  - Q- or V-function
  - Policy (Actor-Critic Model)
  - World Model

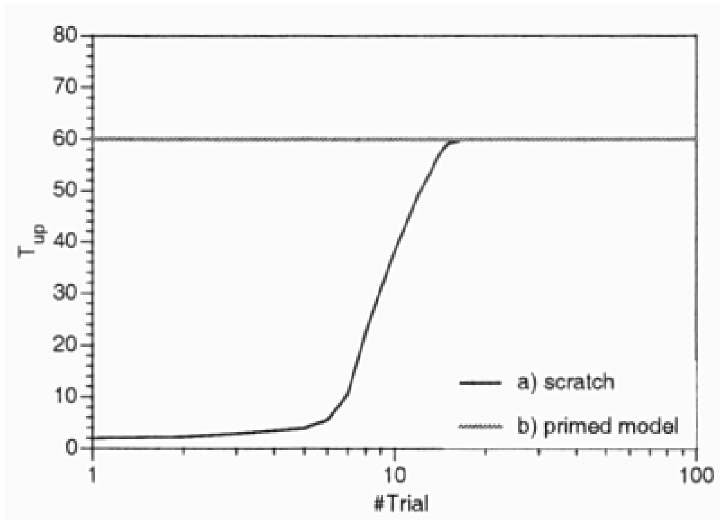
- Comparison in Different Environments
  - Pendulum Swing-up
  - Robot Arm Pole-balancing

[1] Schaal: Learning from Demonstration, NIPS 9 (1997)

Experiment 1: Real Pole-balancing

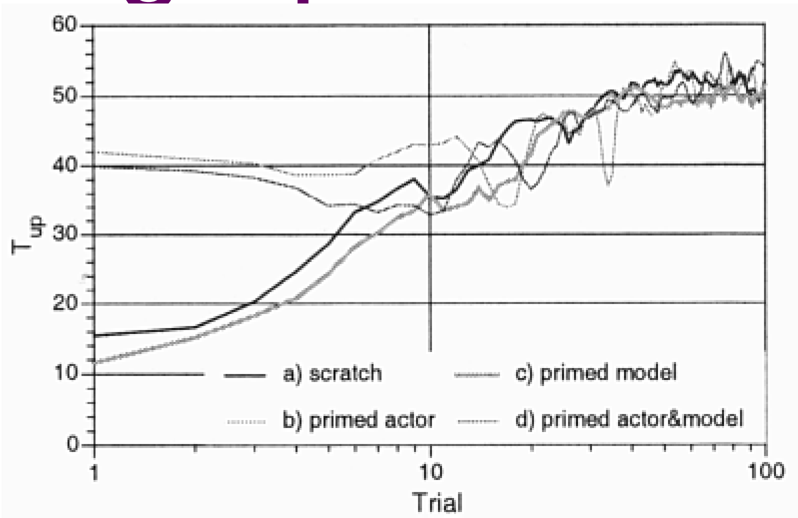
- Balance a Pole with a real Robot Arm
- Inverse Kinematics and Dynamics available
- 30 second Demonstration
  - Learning in one single Trial
- Without Demonstration
  - 10-20 trials necessary

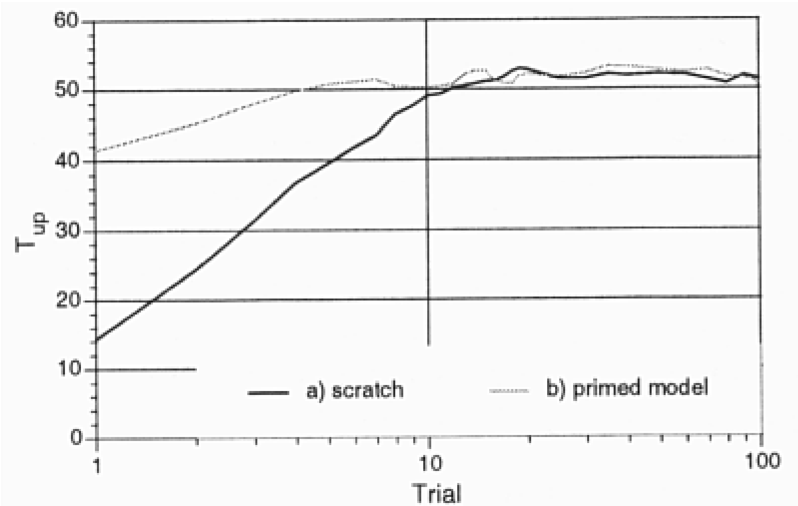




Experiment 2: Swing-up

- Value-function learning
- Primed one-step Model did not speed up learning
- Primed Actor:
  - Initial Advantage
  - Same Time necessary for convergence
- Model-based Learning:
  - Priming Model brings advantage (DYNA-Q „mental updates“)





Implicit Imitation[1]

- Observation of Mentor
  - Distribution of Search for optimal Policies
  - Guide for Exploration
- Implicit Imitation
  - No replay of actions, only additional Information
  - No communication between Mentor and Observer (e.g. commercial mentors)
  - Mentor's Actions are not observeable (allows heterogeneous Mentor and Observer)

[1] Price, Boutilier: Accelerating Reinforcement Learning through Implicit Imitation, Journal of AI Research 19 (2003)

Assumptions

- Full Observeability
  - Own state and reward
  - Mentor´s state
- Duplication of Actions
  - Observer must be able to duplicate the Mentor´s action with sequences of actions
- Similar Objectives
  - Goal of Mentor should be similar (not necessarily identical) to that of Observer

Main Ideas of Implicit Imitation

- Observer uses Mentor Information to build a better World Model
  - Related to Model-based RL
- Calculate more accurate State values through better model
- Augmented Bellman Equation:
  - Consider own and Mentor´s transition probabilities for backup

Homogeneous Case

- Observer and Mentor have same action space
- Confidence estimation for Mentor's hints
- Estimate V_{mentor}: Value of Mentor´s policy from observer´s perspective

- Action selection:
  - Either greedy action w.r.t. own $V_{observer}$
  - Or action most similar to best Mentor´s action (if V_{mentor} is higher than $V_{observer}$)
- Prioritized Sweeping

Extensions

- Inhomogeneous Case
  - Mentor has other actions than Observer
  - Feasibility Test: Can observer reproduce this state transition (otherwise ignore)
- Multiple Mentors

Experiments and Results

- Tested in „tricky“ Grid-Worlds
- Guided agents find good policies rapidly
- Standard RL often gets stuck in Traps
- Learned policies of Observers often outperforms Mentors

- No results yet with humanoid Robots

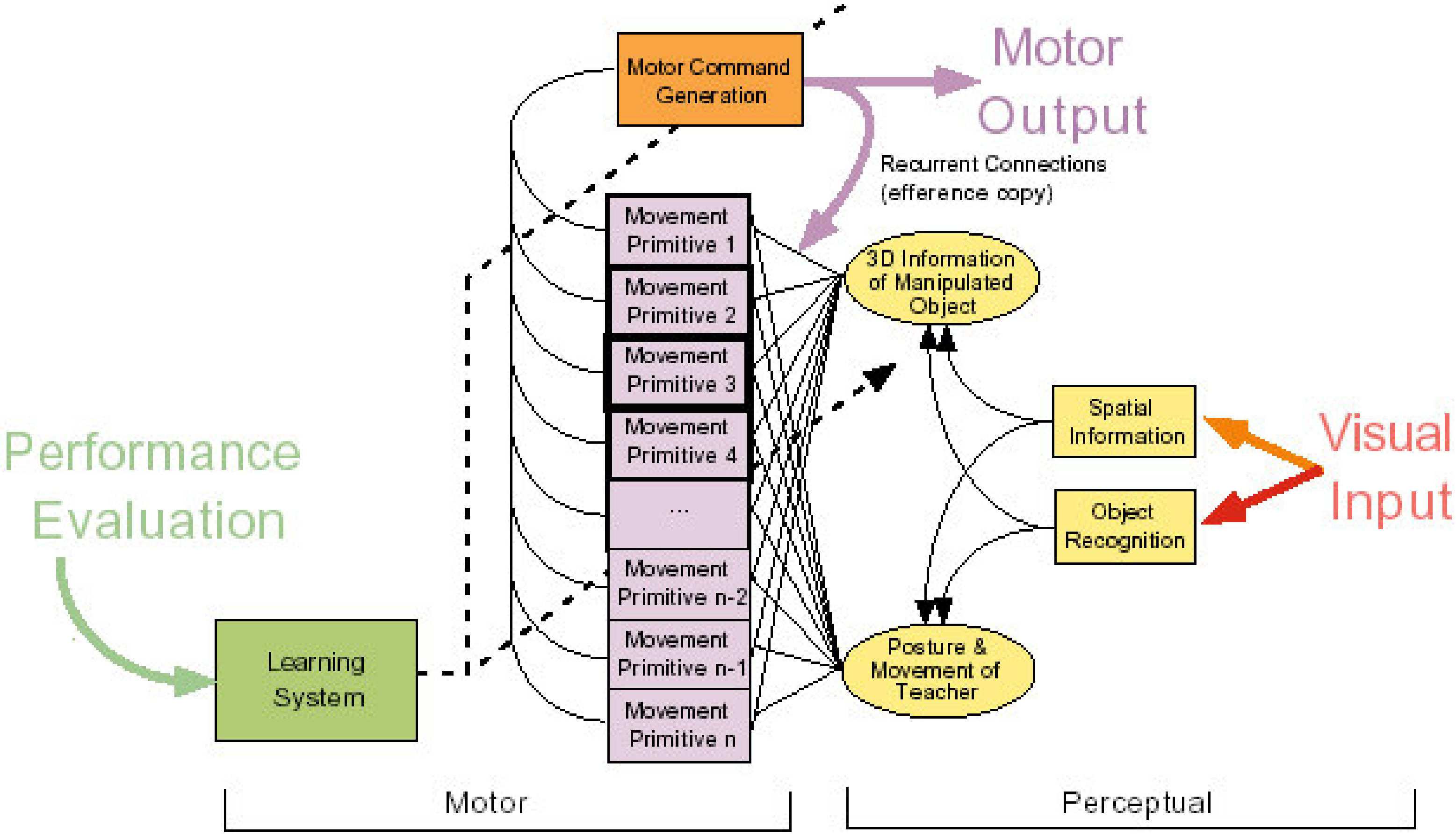
Imitation Learning[1,2]

- Other Names:
  - *Learning by Watching, Teaching by Showing, Learning from Demonstration*
- Using Demonstration from Teacher to learn a Movement
  - Speed up Learning Process
  - Later: Self-Improvement (e.g. RL)
- Highly successful Area of Robot Learning
  - Amazing results for Humanoid Robots
  - One-shot Learning of Complex Movements

[1] Schaal: Is Imitation Learning the Route to Humanoid Robots? (1999)

2 Schaal, Ijspeert, Billard: Computational Approaches to Motor Learning by Imitation (2003)

Schema: Imitation Learning

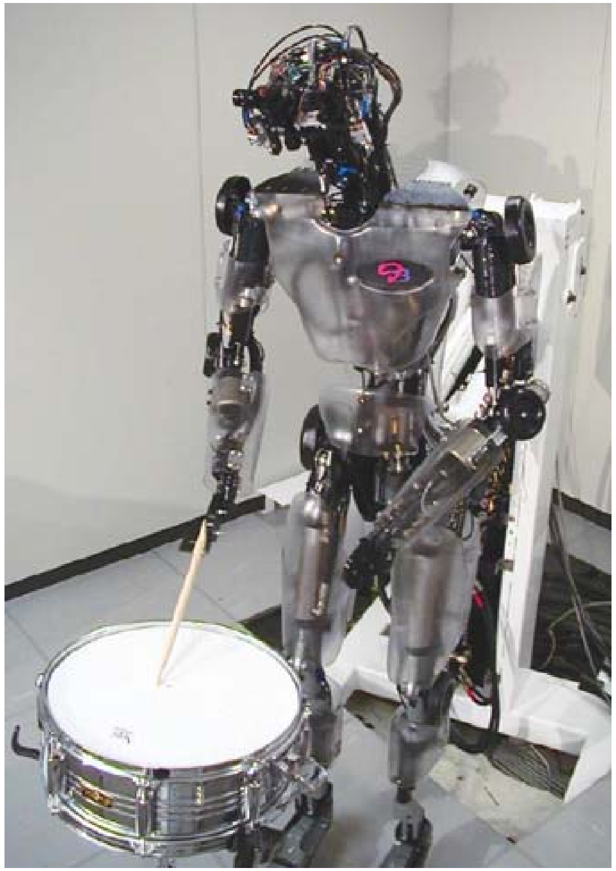


Imitation Learning Components

- Perception:
  - Visual Tracking of demonstrated Movement
- Spatial Transformation
  - Transformation of Coordinates
- Mapping to (existing) Motor Primitives
- Adjusting appropriate Primitives
- Self – improvement
  - Reinforcement Learning

Applications of Imitation Learning

- Humanoid Robots
- Learning of Motor Primitives
  - E.g. „Walking“, „Grasping“, …
- Impossible without prior Knowledge
- Also impossible to solve analytically



Supervised Motor Learning

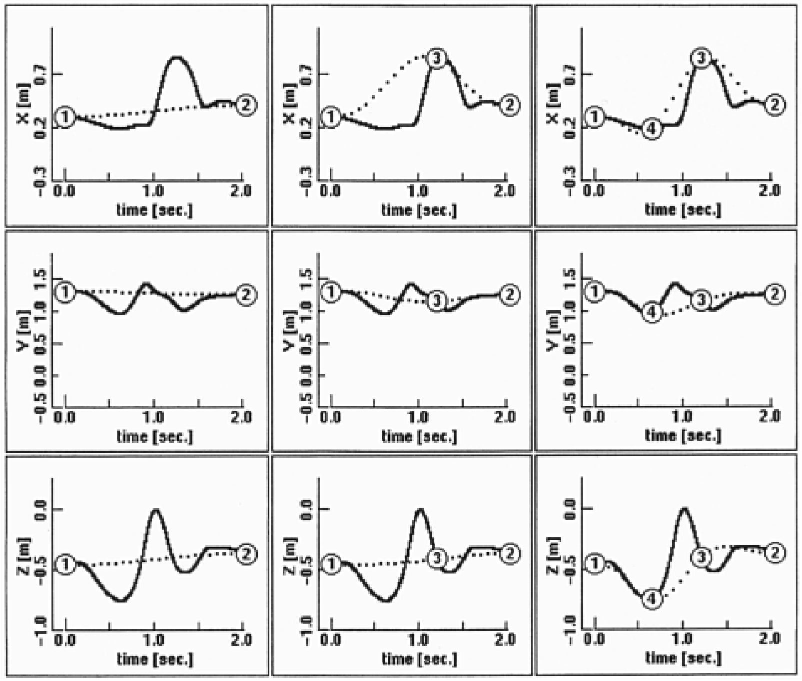
- Optimize Parameter Vector of Policy
- Evaluation Criterion
  - Difficult to design
  - What is the Goal?
    - Reaching final Position?
    - Reproducing the whole Trajectory?
    - Accomplishing Task in Presence of Noise?
    - Rhythmic Movement?

Methods for Imitation

- RL from Demonstration (see above)
- Via-Points Learning
  - Spline Interpolation of Movements
- Dynamical Systems
  - Assuming supplied kinematic Model
  - Shaping of Differential Equations to achieve desired Trajectories

Spline-based Imitation Learning[1]

- Learn via-points of Trajectory
  - Interpolate smoothly with Splines between these points
- Adjust location of via-points



[1] Miyamoto, Kawato: A tennis serve and upswing learning robot based on bi-directional theory (1998)

Adjustment of Via-points

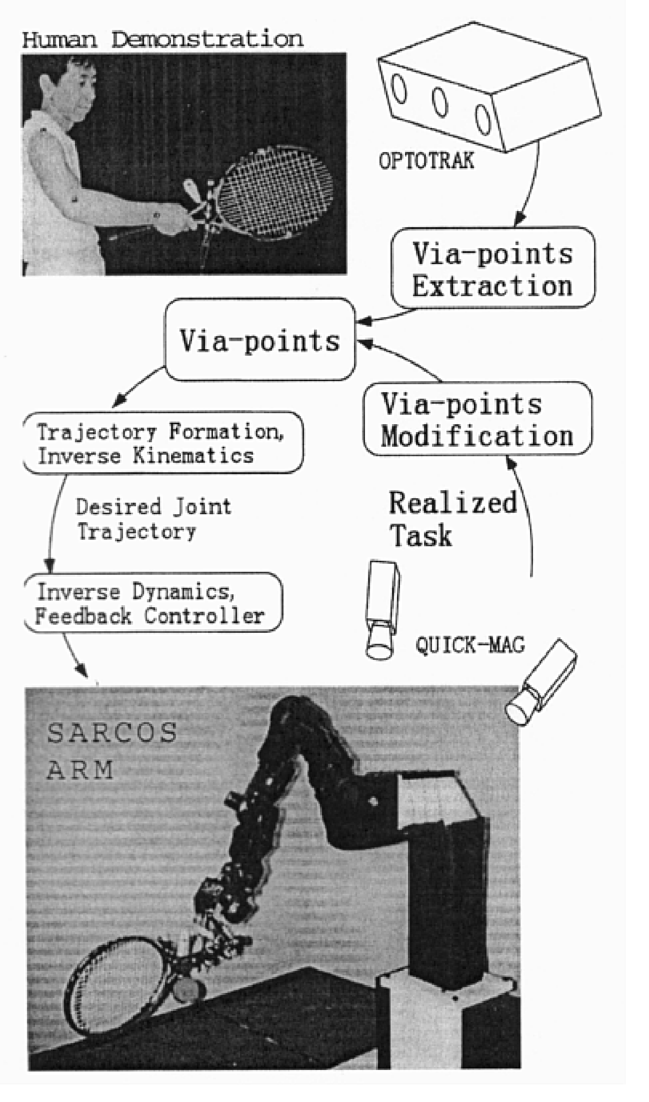
- <u>Trial-and-Error</u> Learning
  - But not real RL
- Execute Policy and Measure Error (Distance to Goal)
  - Adjust Parameters (via-point coordinates) to minimize this Error

- Newton-like Optimization
  - Estimation of Jacobi Matrix (1st partial derivations) in first Training runs
  - Estimate by applying small pertubations and measuring impact on Error

Experiment: Tennis Serve

- Robot Arms learns Tennis Serve from Human Demonstration
- Used ca. 20 trials to estimate Jacobian
- Learned to hit Goal reliably in 60 trials

- Limitations:
  - Pure feedforward Control



Problems of Via-point Learning

- Aims at explicit Imitation
  - Learned policy is time-dependent
- Difficult to generalize to other Environments
- Not robust in coping with unforeseen pertubations

Shaping of Dynamical Systems[1]

- System of ordinary Differential Equations
- y is trajectory position
- g is goal (Attractor)
- ψ_i Gaussian kernels
- x, v: internal state

$$\dot{z} = \alpha_z\left(\beta_z(g-y)-z\right)$$

$$\dot{y} = z + \frac{\sum_{i=1}^{N}\psi_i w_i}{\sum_{i=1}^{N}\psi_i} v$$

$$\dot{v} = \alpha_v\left(\beta_v(g-x)-v\right)$$

$$\dot{x} = v$$

- Attractor landscape can be adjusted by learning paramters w_i

$$\psi_i = \exp\left(-\frac{1}{2\sigma_i}\left(\frac{x-x_0}{g-x_0}-c_i\right)^2\right)$$

[1] Ijspeert, Nakanishi, Schaal: Movement Imitation with Nonlinear Dynamical Systems in Humanoid Robots (2002)

Shaping of Dynamical Systems

- g is a unique point Attractor of the system (y → g)
- v and x define an internal state that generates complex Trajectories towards g
  - These Trajectories can be shaped by learning w
- Non-linear Regression Problem
  - Adjust w to embed demonstrated trajectory
  - Locally weighted Regression

- Feedback term can be added to make on-line modifications possible (see [Ijspeert, et.al.])
- Policy Gradient RL can be used to refine behaviour[1]

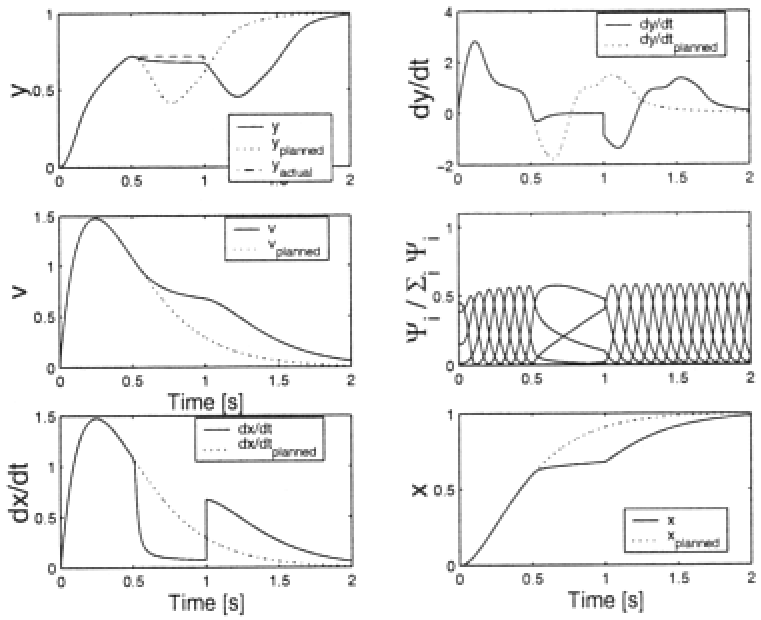
[1] Schaal, Peters, Nakanishi, Ijspeert: Learning Movement Primitives (2004)

Advantages

- Policies are not time-dependent
  - Only state-dependent
- Able to learn very complex Movements
- Learns stable Policies
  - With Feedback-Term robust to online pertubations
- Straightforward extension to rhythmic Movements (e.g. walking)
- Allows Recognition of Movements
  - Classification in Parameter Space
  - Similar Movements have similar *w* vectors

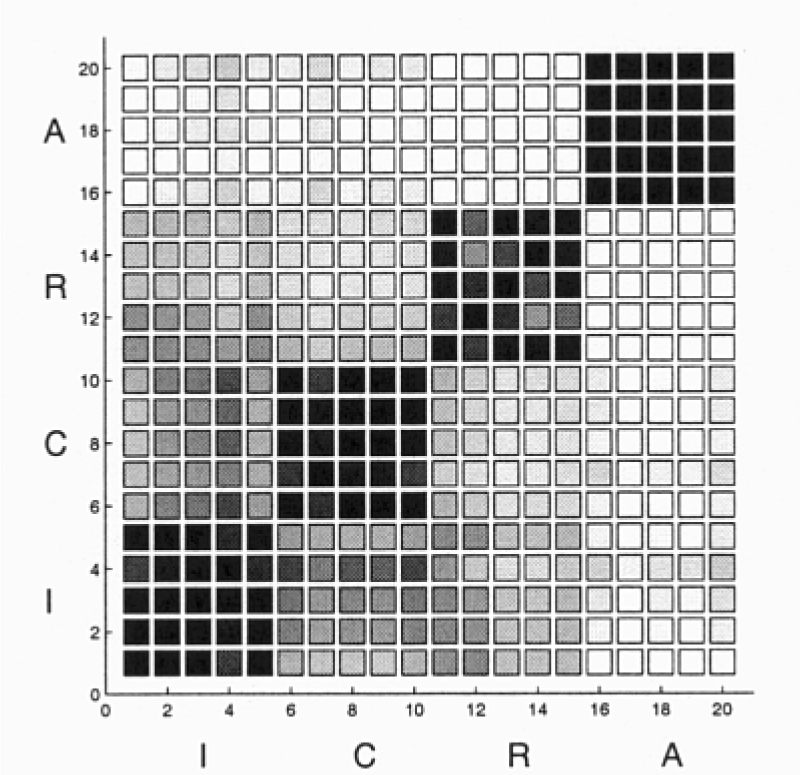
Experiments (1)

- Evolution of a dynamical system under pertubation
- Position is frozen
- System recovers from pertubation and continuous planned execution



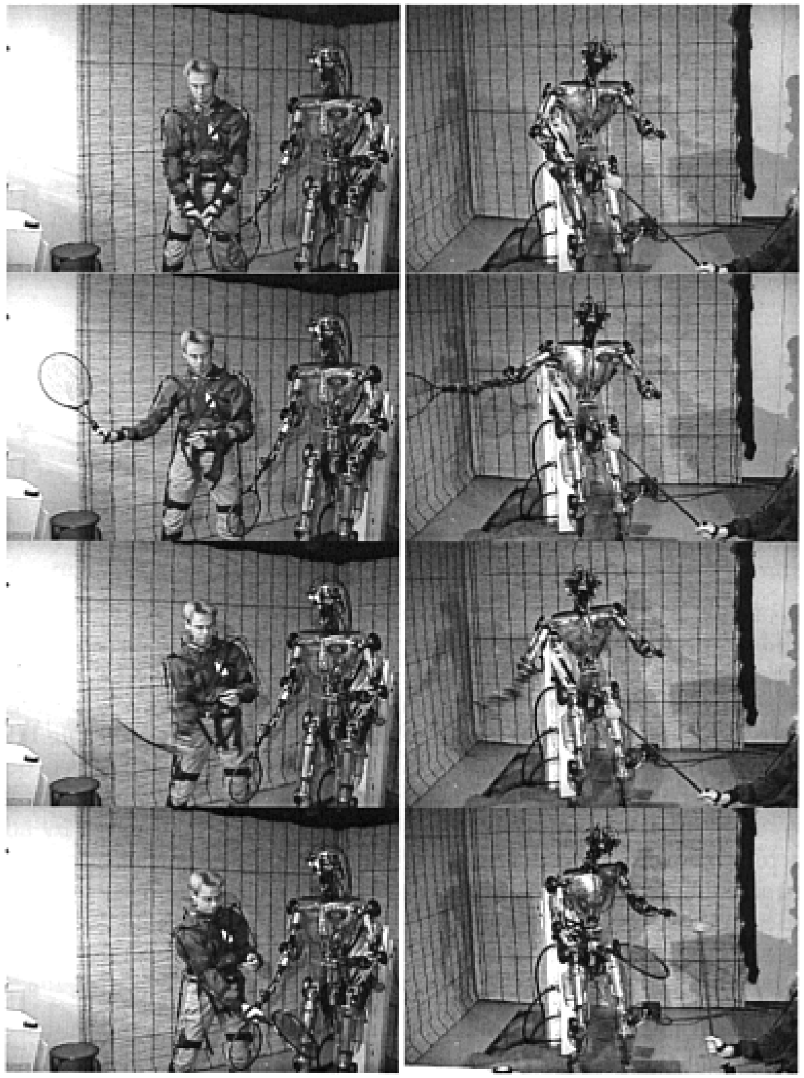
Experiments (2)

- Trajectory Comparison
- Similar Trajectories yield similar parameters
- Character Drawing
  - Measuring Correlations in five Trials
- Could be used for Recognition



Experiments (3)

- Learning Tennis Swings
  - Fore- and Backhand
- Trajectories translated with inverse dynamics
- Humanoid Robot can repeat Swing for unseen Ball Positions
  - Trajectories similar to human demonstrations



Further Results

- Imitating Rhythmic Behaviour
  - Tracing a figure of 8
  - Drumming
- Simulated Biped Walking

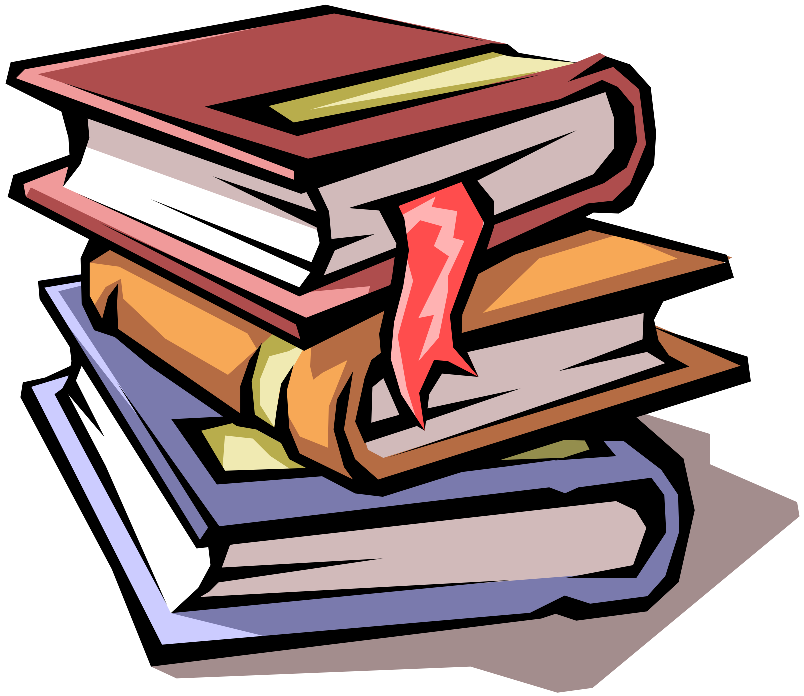
Problems of Imitation Learning

- Tracking of Demonstrations
- Hidden Variables
- Incompatibility Teacher – Student
- Generalization vs. Mimicking
- Time-dependence of learned Policy

What else exists?

- Memory-based RL
- Fuzzy RL
- Multi-objective RL
- Inverse RL
- ...

- Could all be used for Motor Learning



Memory-based RL

- Use a <u>short-term Memory</u> to store important Observations over a long time
  - Overcome Violations of Markov Property
  - Avoid storing finite histories

- Memory Bits [Peshkin et.al.]
  - Additional Actions that change memory bits
- Long Short-Term Memory [Bakker]
  - Recurrent Neural Networks

Fuzzy RL

- Learn a Fuzzy Logic Controller via Reinforcement Learning [Gu, Hu]
- Optimize Parameters of Membership Functions and Composition of Fuzzy Rules
- Adaptive Heuristic Critic Framework

Inverse RL

- Learn the Reward Function from observation of optimal Policy [Russell]
  - Goal: Understand, which optimality principle underlies a policy
- Problems:
  - Most algorithms need full policy (not trajectories)
  - Ambiguity: Many different reward functions could be responsible for the same policy
- Few results exist until now

Multi-objective RL

- Reward-Function is a <u>Vector</u>
  - Agent has to fulfill multiple tasks (e.g. reach goal and stay alive)
  - Makes design of Reward function more natural

- Algorithms are complicated and make strong assumptions
  - E.g. total ordering on reward vectors [Gabor]
  - Game theoretic Principles [Shelton]

Agenda

- Motor Control
- Specific Problems of Motor Control
- Reinforcement Learning (RL)
- Survey of Advanced RL Techniques
- Existing Results
- Open Research Questions

Learning of Motor Sequences

- Most research in Motor Learning is concerned with learning Motor Primitives

- Learning Motor Sequences is more complicated
  - Smooth switching between Primitives
  - Hierarchical RL

- Examples:
  - Playing a full game of Tennis
  - Humanoid Robot Soccer

Combinations of RL Techniques

- Explicit and Implicit Imitation
  - Use Imitation Learning for a good initial policy
  - Still use a Mentor for initial exploration phase
- RL with State Prediction
  - Any of the presented RL techniques could be improved by using a learned World Model for prediction of Movement Consequences
- Non-standard Techniques
  - Used mostly in artificial Grid-World Domains

Movement Understanding

- Imitating a Movement makes us understand the principles of biological Motor Control better

- Recognize the Goal of the Teacher by watching a Movement
  - Inverse RL (understand Reward function)

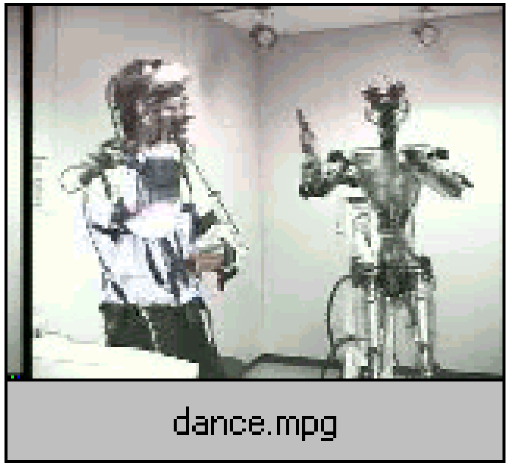
- Recognition of Movements
  - E.g. in Dynamical Systems Context
  - Computer Vision: e.g. gesture understanding

More Complex Behaviours

- There are still a lot of possibilities
  - Advanced Robots
  - Biologically Inspired Robots
  - More difficult Movements

- Useful Robots
  - Autonomous Working Robots
  - Helping Robots: for old or handicapped people, children, at home, etc.

Thank You!



References: RL

- Sutton, Barto: Reinforcement Learning: An Introduction (1998)

- <u>Continuous Learning</u>:
- Coulom: Feedforward Neural Networks in RL applied to High-dimensional Motor Control (2002)
- Doya: RL in continuous Time and Space (2000)

- <u>Hierarchical RL:</u>
- Dietterich: Hierarchical RL with the MAXQ Value Function Decomposition (2000)
- Kalmar, Szepeszvari, Lörincz: Module-based RL: Experiments with a real robot (1998)

References: Policy Gradient

- Baird, Moore: Gradient Descent for General RL (1999)
- Baxter, Bartlett: Direct Gradient-Based RL (1999)
- Baxter, Bartlett: RL in POMDP´s via Direct Gradient Ascent (2000)
- Lawrence, Cowan, Russell: Efficient Gradient Estimation for Motor Control Learning (2003)
- Ng, Jordan: PEGASUS: A policy search method for large MDPs and POMDPs (2000)
- Ng, Kim, Jordan, Sastry: Autonomous Helicopter Flight via RL (unpublished draft)
- Peters, Vijayakumar, Schaal: RL for humanoid robots (2003)
- Sutton, McAllester, Singh, Mansour: Policy Gradient Methods for RL with Function Approximation (2000)

References: Prior Knowledge

- Price, Boutilier: Accelerating RL through Implicit Imitation (2003)
- Schaal: Learning from Demonstration (1997)
- Smart, Kaelbling: Effective RL for Mobile Robots (2002)

References: Imitation Learning

- Arbib: Handbook of Brain Theory and Neural Networks, 2nd Ed. (2003)
- Ijspeert, Nakanishi, Schaal: Movement Imitation with Nonlinear Dynamical Systems in Humanoid Robots (2002)
- Ijspeert, Nakanishi, Schaal: Learning Attractor Landscapes for Learning Motor Primitives (2003)
- Miyamoto, Kawato: A tennis serve and upswing learning robot based on bi-directional Theory (1998)
- Schaal: Is Imitation Learning the Route to Humanoid Robots? (1999)
- Schaal, Ijspeert, Billard: Computational Approaches to Motor Learning by Imitation (2003)
- Schaal, Peters, Nakanishi, Ijspeert: Learning Movement Primitives (2004)

References: Non-standard Techniques

- Bakker: RL with Long Short-Term Memory (2002)
- Gabor, Kalmar, Szepesvari: Multi-criteria RL (1998)
- Gu, Hu: RL for Fuzzy Logic Controllers for Quadruped Walking Robots (2002)
- Peshkin, Meuleau, Kaelbling: Learning Policies with External Memory (1999)
- Russell:Learning Agents for Uncertain Environments (1998)
- Shelton: Balancing Multiple Sources of Reward in RL (2000)
- Sprague, Ballard: Multiple-Goal RL with Modular SARSA(0) (2003)