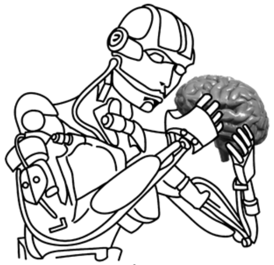
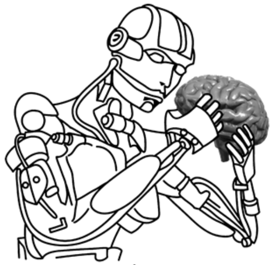


CS545— Lecture 20

- Adaptive Approaches
  - Reinforcement Learning
    - Model-free methods
    - Model-based methods
  - Control
    - Model Reference Adaptive Control
    - Self-Tuning Regulators
    - Linear Regression
- http://robotics.usc.edu/~aatrash/cs545

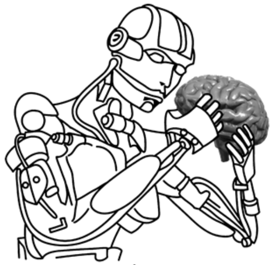
Reinforcement Learning



- Assume world is MDP but we don't have models. (Don't have *T(s,a,s')* or *R(s,a)*

- Need to determine policy (and maybe model) through execution

TD(λ)



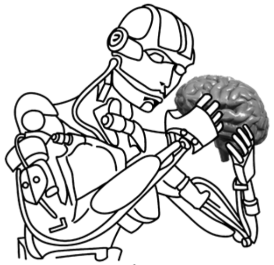
- Estimate value function

$$V(s) = V(s) + \alpha\big(R + \gamma V(s') - V(s)\big)$$

- TD(0) – only update current state
- TD(λ) – update states visited recently

- On-policy, model-free

Q-learning

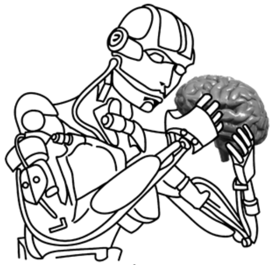


- Estimate Q value

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma\, max_{a'}\, Q(s',a') - Q(s,a))$$

- Off-policy, model-free
- Will converge to optimal policy with enough data

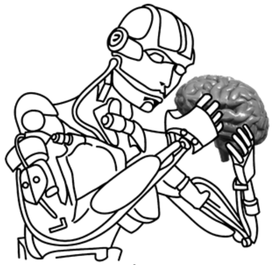
Dyna



- Model-based method. Adapts model parameters and policy during execution

1. Selection action based on *Q(s,a)*. Get *r* and *s'*
2. Update estimates of *T* and *R*
3. Update policy at state *s* (backup)
    1. $Q(s,a) = R(s,a) + \gamma(\sum_{s' \in S} T(s,a,s')V(s))$
4. Perform *k* updates to random *Q(s,a)* pairs

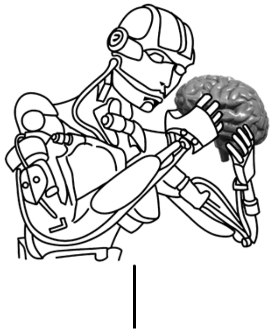
Prioritized Sweeping/ Queue-Dyna

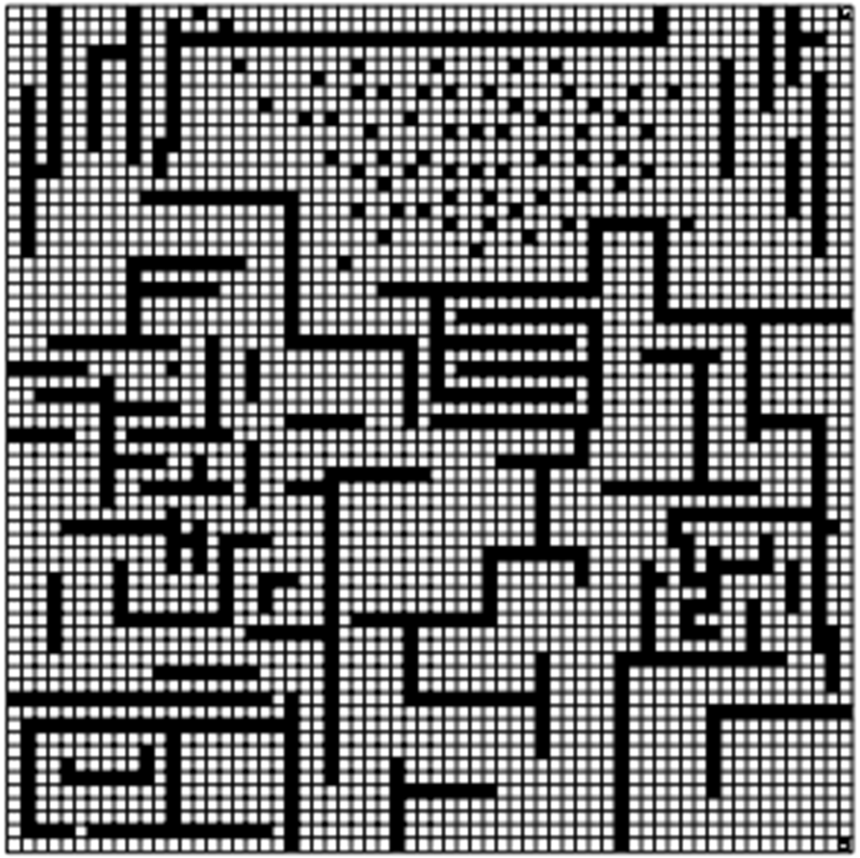


- Instead of performing backup randomly, use priorities

1. Remember old V value: $V_{old} = V(s)$
2. Update state's value
$$V(s) = \max_a(R(s,a) + \gamma \sum_{s'} T(s,a,s')V(s'))$$
3. Set state's priority to 0
4. Compute $\Delta = |V_{old} - V(s)|$ (how much change)
5. Use Δ to update priorities of predecessors
$$\Delta * T(s,a,s')$$

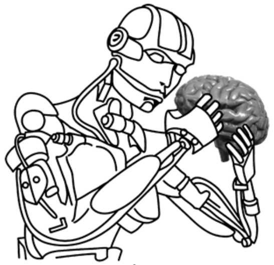
Results





	Steps before convergence	Backups before convergence
Q-learning	531,000	531,000
Dyna	62,000	3,055,000
prioritized sweeping	28,000	1,010,000

Exploration vs. Exploitation

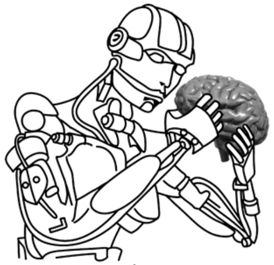


- Need to explore
- One approach: select random action with ε likelihood
- Bolztmann Exploration

$$P(a) = \frac{\frac{e^{ER(a)}}{T}}{\sum_{a' \in A} \frac{e^{ER(a')}}{T}}$$

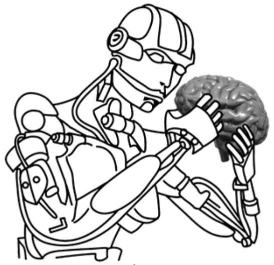
  - Decrease T over time

Application in Robotics



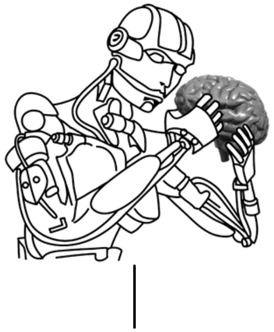
- Juggling devil-stick (Schaal 1994)
- Box-pushing (Mahadevan 1991)
- Multi-robot gathering (Mataric 1994)
- Elevator dispatching (Crites 1996)

The Adaptive Control Problem

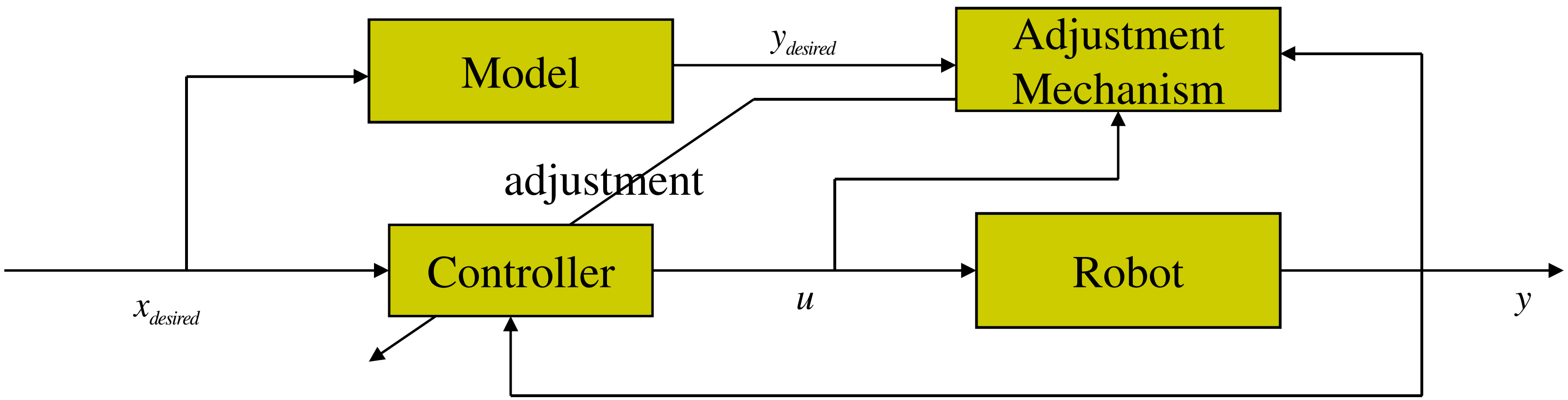


- Characterize the desired behavior of the closed loop system
- Determine a suitable control law with adjustable parameters
- Find a mechanism for adjusting the parameters
- Implement the control law

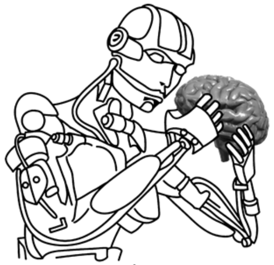
Model-Reference Adaptive Control (Direct Learning)



- Performance is given to correspond to a particular reference model
  - E.g. $$m\ddot{x} + b\dot{x} + c = u$$
- Adjustment of controller is done directly
- E.g., adjust controller parameter by gradient descent



Model-Reference Adaptive Control – Example



- Consider the generic control system

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u}$$

- For this example, make this an even simpler system

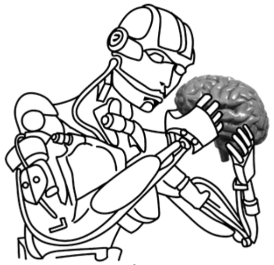
$$\dot{x} = f(x) + u$$

- Assume that *f* is unkown and needs to be estimated by a learning process. Thus, we can formulated a control law:

$$\begin{aligned} u &= -\hat{f}(x) + \dot{x}_d - k(x - x_d) \\ &= -x\ \hat{\theta} + \dot{x}_d - k(x - x_d) \end{aligned}$$

  - Where we replaced *f* with a simple linear function

Model-Reference Adaptive Control – Example



- The goal of model-reference adaptive control is to adjust the open parameter and the control law such that the system is ALWAYS stable
- The system dynamics are now

$$\dot{x} = x\ \theta - x\ \hat{\theta} + \dot{x}_d - k\left(x - x_d\right)$$

- Define errors

$$e = x_d - x$$

$$\tilde{\theta} = \theta - \hat{\theta}$$

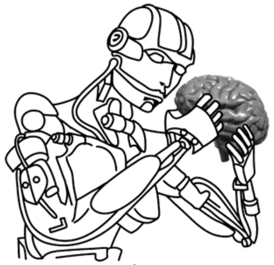
- Thus:

$$\dot{x} = x\ \tilde{\theta} + \dot{x}_d + ke$$

$$0 = x\ \tilde{\theta} + \dot{e} + ke$$

$$\dot{e} = -x\ \tilde{\theta} - ke$$

Model-Reference Adaptive Control – Example



- Define a Lyapunov function

$$
\begin{aligned}
V &= \frac{1}{2}e^2 + \frac{1}{2}\tilde{\theta}\,\Gamma^{-1}\tilde{\theta} \\
\dot{V} &= e\dot{e} + \tilde{\theta}\,\Gamma^{-1}\dot{\tilde{\theta}} \\
&= e\dot{e} - \tilde{\theta}\,\Gamma^{-1}\dot{\hat{\theta}} \\
&= e\left(-x\,\tilde{\theta} - ke\right) - \tilde{\theta}\,\Gamma^{-1}\dot{\hat{\theta}} \\
&= -ex\tilde{\theta} - ke^2 - \tilde{\theta}\,\Gamma^{-1}\dot{\hat{\theta}}
\end{aligned}
$$

- Thus, choose:

$$
-ex\tilde{\theta} - \tilde{\theta}\,\Gamma^{-1}\dot{\hat{\theta}} = 0
$$

Model-Reference Adaptive Control – Example

- Thus:

$$-ex\tilde{\theta} - \tilde{\theta}\;\Gamma^{-1}\dot{\hat{\theta}} = 0$$

$$\tilde{\theta}\left(-ex - \Gamma^{-1}\dot{\hat{\theta}}\right) = 0$$

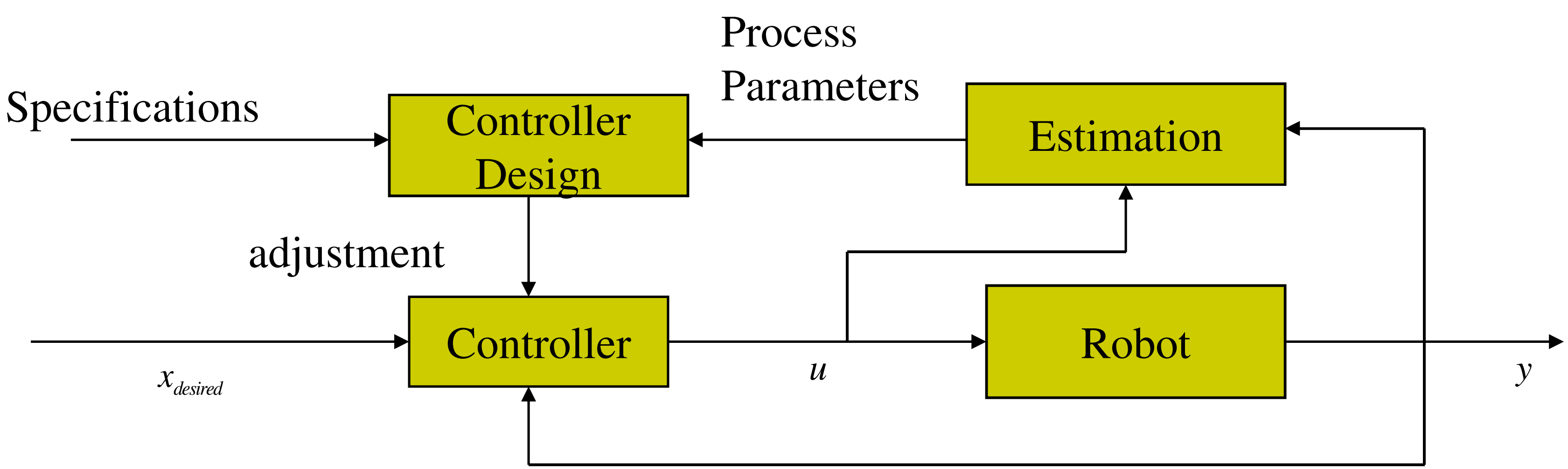
$$\dot{\hat{\theta}} = -\Gamma ex$$

- I.e., the guaranteed stable parameter adapation law is

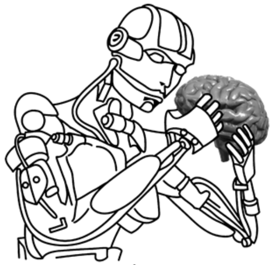
$$\dot{\hat{\theta}} = -\Gamma ex$$

Self-tuning Regulators (Indirect Learning)

- Controller is redesigned based on some estimated parameters
  - “certainty equivalence principle”
  - E.g., LQR controller is redesigned based on estimated model
  - This corresponds to an indirect update of the controller



Example: Estimate the Robot Model From Data



- How to obtain data?
  - Try "random" commands u, observe the state and change of state
  - Don't destroy the robot …
- How to estimate the model?
  - Model is nonlinear
    - Need nonlinear estimation techniques (e.g., neural networks)
  - Model is linear
    - Use linear regression or recursive least squares
- Essential ingredients of estimation:
  - A cost criterion:
    - Usually least squares $J = \frac{1}{2}\sum_{i=1}^{N}(t_i - y_i)^2$
  - Some adjustable parameter ("a data generating model")

Linear Regression for One Output

- The data generating model

$$y = \tilde{\mathbf{w}}^T \tilde{\mathbf{x}} + w_0 + \varepsilon = \mathbf{w}^T \mathbf{x} + \varepsilon$$

$$\text{where } \mathbf{x} = \left[\mathbf{x}^T, 1\right]^T, \mathbf{w} = \begin{bmatrix} \tilde{\mathbf{w}} \\ w_0 \end{bmatrix}, E\{\varepsilon\} = 0$$

- Least Squares Cost Function

$$J = \frac{1}{2}(\mathbf{t} - \mathbf{y})^T (\mathbf{t} - \mathbf{y}) = \frac{1}{2}(\mathbf{t} - \mathbf{X}\mathbf{w})^T (\mathbf{t} - \mathbf{X}\mathbf{w})$$

$$\text{where : } \mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ \cdots \\ t_n \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \cdots \\ \mathbf{x}_n^T \end{bmatrix}$$

- Minimize Cost

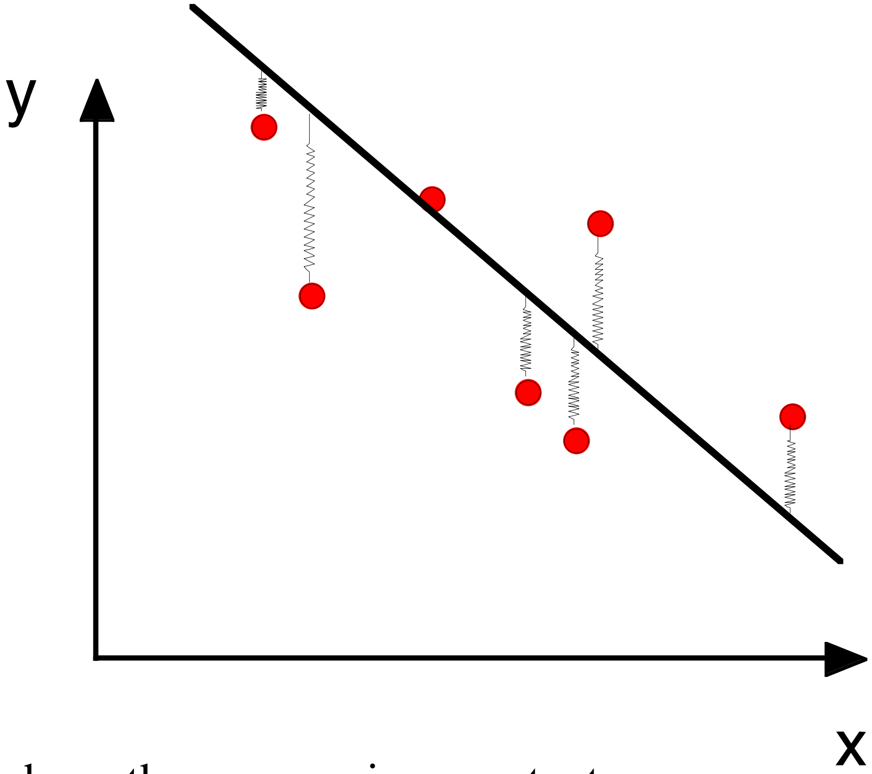
$$\frac{\partial J}{\partial \mathbf{w}} = 0 = \frac{\partial J}{\partial \mathbf{w}}\left(\frac{1}{2}(\mathbf{t} - \mathbf{X}\mathbf{w})^T (\mathbf{t} - \mathbf{X}\mathbf{w})\right) = -(\mathbf{t} - \mathbf{X}\mathbf{w})^T \mathbf{X}$$

$$= -\mathbf{t}^T \mathbf{X} + (\mathbf{X}\mathbf{w})^T \mathbf{X} = -\mathbf{t}^T \mathbf{X} + \mathbf{w}^T \mathbf{X}^T \mathbf{X}$$

$$\textit{thus}: \quad \mathbf{t}^T \mathbf{X} = \mathbf{w}^T \mathbf{X}^T \mathbf{X} \quad \textit{or} \quad \mathbf{X}^T \mathbf{t} = \mathbf{X}^T \mathbf{X}\, \mathbf{w}$$

$$\textit{result}: \quad \mathbf{w} = \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{t}$$

Physical Interpretation of Least Squares



– all springs have the same spring constant

– points far away generate more “force” (danger of outliers)

– springs are vertical

– solution is the minimum energy solution achieved by the springs