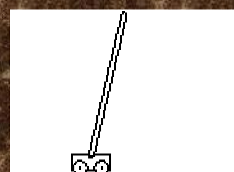
CSE/NB 528

Lecture 14: Reinforcement Learning

(Chapter 9)



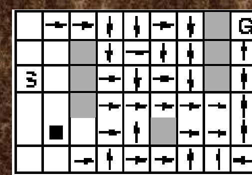


Image from http://clasdean.la.asu.edu/news/images/ubep2001/neuron3.jpg
Lecture figures are from Dayan & Abbott's book
http://people.brandeis.edu/~abbott/book/index.html

Today's Agenda

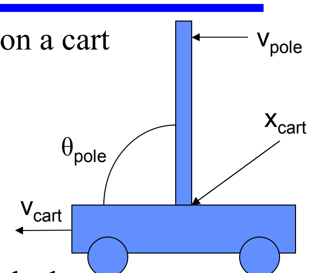
- Wrap-up of Supervised Learning
  - Demos
- Reinforcement Learning
  - What is reinforcement learning?
  - Classical conditioning
    - Learning to salivate (predicting reward)
  - Predicting Delayed Rewards
    - Temporal Difference Learning
  - Learning to Act
    - Q-learning
    - Actor-Critic Architecture

On-line Supervised Learning Demos

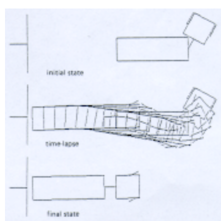
- Function Approximation:
  http://neuron.eng.wayne.edu/bpFunctionApprox/bpFunctionApprox.html
- Pattern Recognition
  http://www-cse.uta.edu/%7Ecook/ai1/lectures/applets/hnn/JRec.html
- Image Compression
  http://neuron.eng.wayne.edu/bpImageCompression9PLUS/bp9PLUS.html
- Backpropagation for Control: Ball Balancing
  http://neuron.eng.wayne.edu/bpBallBalancing/ball5.html

Demos (by Keith Grochow, 2001)

- Neural network learns to balance a pole on a cart
  - System:
    - 4 state variables: x_{cart}, v_{cart}, θ_{pole}, v_{pole}
    - 1 input: Force on cart
  - Backprop Network:
    - Input: State variables
    - Output: New force on cart



- NN learns to back a truck into a loading dock
  - System (Nyugen and Widrow, 1989):
    - State variables: x_{cab}, y_{cab}, θ_{cab}
    - 1 input: new $\theta_{steering}$
  - Backprop Network:
    - Input: State variables
    - Output: Steering angle $\theta_{steering}$



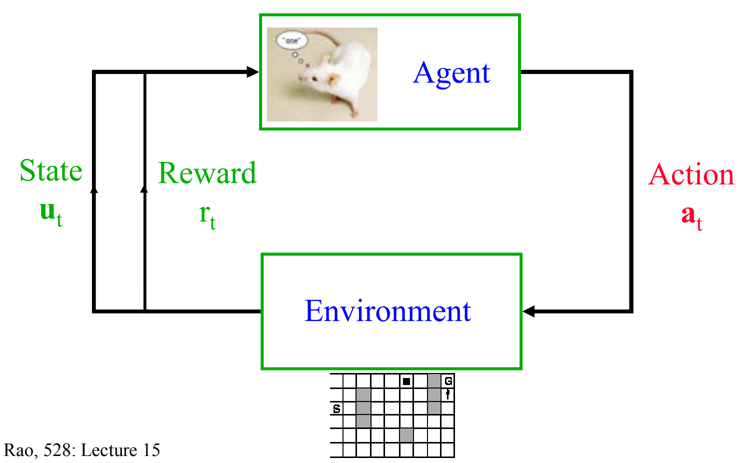
Human drivers don't usually get exact supervisory signals (commands for muscles) for learning to drive!

Must learn by trial-and-error

Might get "rewards and punishments" along the way

Enter…Reinforcement Learning

The Reinforcement Learning "Agent"

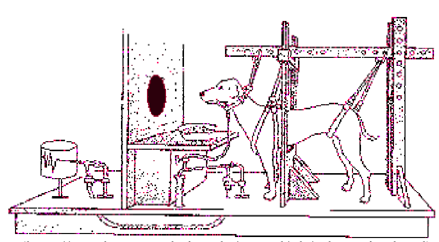


The Reinforcement Learning Framework

- **Unsupervised learning** → Learn the hidden causes of inputs
- **Supervised learning** → Learn a function based on training examples of (input, desired output) pairs
- **Reinforcement Learning** → Learn the best actions to take at any given state so as to maximize total (future) reward
  - Learn by trial and error
  - Intermediate between unsupervised and supervised learning Instead of explicit teaching signal (or desired output), you get *rewards or punishments*
  - Inspired by classical conditioning experiments (remember Pavlov's hyper-salivating dog?)

Early Results: Pavlov and his Dog

- Classical (Pavlovian) conditioning experiments
- Training: Bell →Food
- After: Bell → Salivate
- Conditioned stimulus (bell) predicts future reward (food)



(http://employees.csbsju.edu/tcreed/pb/pdoganim.html)

Predicting Reward

- Stimulus $u = 0$ or 1
- Expected reward $v = wu$
- Delivered reward $= r$
- Learn w by minimizing $(r - v)^2$

$$w \rightarrow w + \varepsilon(r - v)u$$

(same as the delta rule; also called Rescorla-Wagner rule)

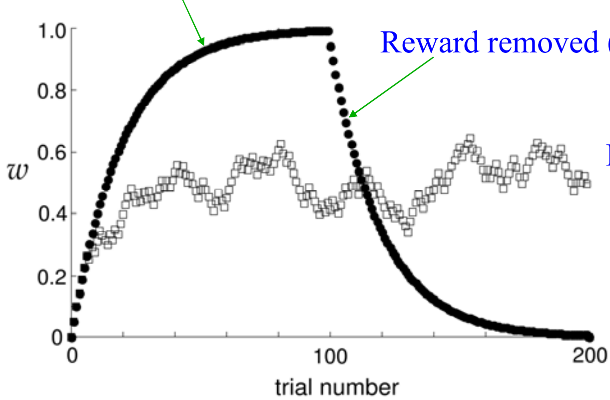
- Prediction error $\delta = (r - v)$
- For small ε and $u = 1$, $w \rightarrow w + \varepsilon(r - w)$
  - Average value of w = $\langle w \rangle \approx \langle r \rangle$

Predicting Reward during Conditioning

Reward r present (conditioning) ($r = 1$, $\varepsilon = 0.5$)

Reward removed (“extinction”)

Reward presented 50% of the trials



Predicting Delayed Rewards

- In more realistic cases, reward is typically delivered at the end (when you know whether you succeeded or not)
- Time: $0 \leq t \leq T$ with stimulus $u(t)$ and reward $r(t)$ at each time step t
- Key Idea: Make the output $v(t)$ predict *total expected future reward* starting from time t

$$v(t) \approx \left\langle \sum_{\tau=0}^{T-t} r(t+\tau) \right\rangle$$

Learning to Predict Delayed Rewards

- Use a set of modifiable weights $w(t)$ and *predict based on all past stimuli* $u(t)$:

$$v(t) = \sum_{\tau=0}^{t} w(\tau) u(t-\tau)$$

- Would like to find $w(\tau)$ that minimize:

$$\left(\sum_{\tau=0}^{T-t} r(t+\tau) - v(t) \right)^2$$

(Can we minimize this using gradient descent and delta rule?)

Yes, BUT…not yet available are future rewards

Temporal Difference (TD) Learning

- **Key Idea**: Rewrite squared error to get rid of future terms:

$$\left(\sum_{\tau=0}^{T-t} r(t+\tau) - v(t)\right)^2 = \left(r(t) + \sum_{\tau=0}^{T-t-1} r(t+1+\tau) - v(t)\right)^2$$

$$\approx \left(r(t) + v(t+1) - v(t)\right)^2$$

- **Temporal Difference (TD) Learning**:

For each time step t, do:

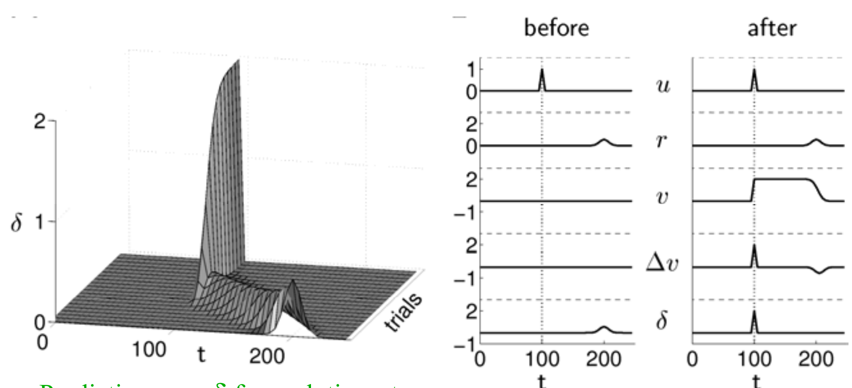
$$v(t) = \sum_{\tau=0}^{t} w(\tau)u(t-\tau)$$

For all τ ($0 \leq \tau \leq t$), do:

$$w(\tau) \to w(\tau) + \varepsilon \overbrace{[\underbrace{r(t) + v(t+1)}_{\text{Expected future reward}} - \underbrace{v(t)}_{\text{Prediction}}]}^{\delta} u(t-\tau)$$

Predicting Delayed Reward: TD Learning

Stimulus at t = 100 and reward at t = 200

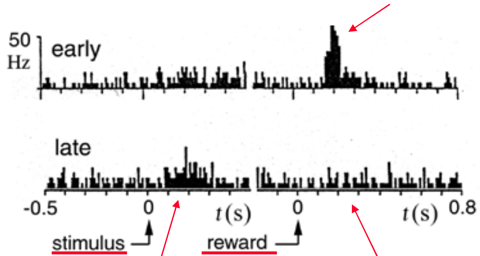


Prediction error δ for each time step
(over many trials)

Reward Prediction Error Signal in Monkeys?

Dopaminergic cells in Ventral Tegmental Area

Reward Prediction error $[r(t)+v(t+1)-v(t)]$



Before Training

After Training

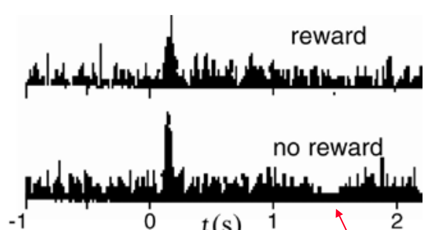
$[0+v(t+1)-v(t)]$

No error

$v(t) \approx r(t)+v(t+1)$

More Evidence for Prediction Error Signals

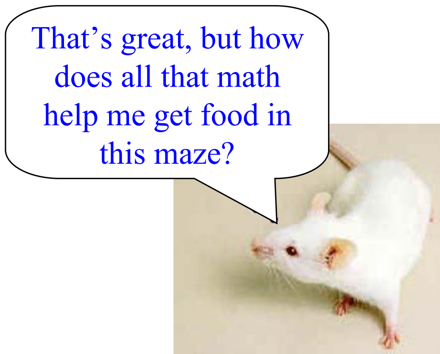
Dopaminergic cells in Ventral Tegmental Area



Negative error

$r(t)=0, v(t+1)=0$

$[r(t)+v(t+1)-v(t)]=-v(t)$



Using Reward Predictions to Select Actions

- Suppose you have computed "Values" for various actions
- $Q(a)$ = value (predicted reward) for executing action a
  - ➪ Higher if action yields more reward, lower otherwise
- Can select actions probabilistically according to their value:

$$P(a) = \frac{\exp(\beta Q(a))}{\sum_{a'} \exp(\beta Q(a'))}$$

(High β selects actions with highest Q value. Low β selects more uniformly)

Simple Example: Bee Foraging

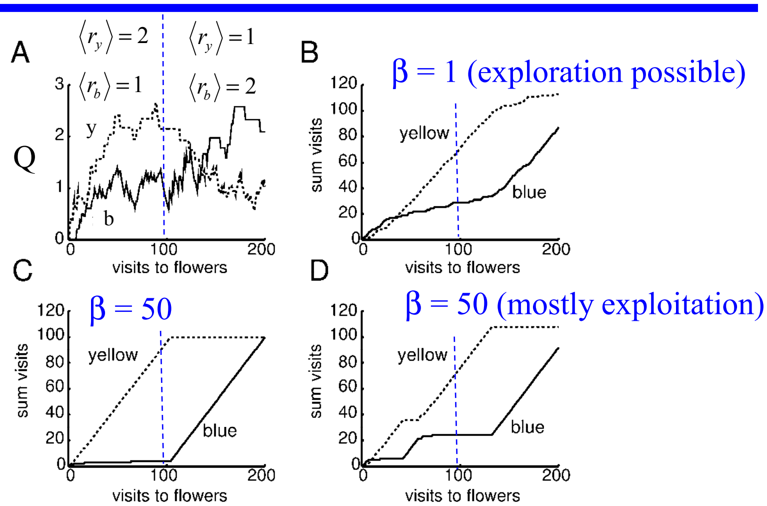
- Experiment: Bees select either yellow (y) or blue (b) flowers based on nectar reward
- Idea: Value of yellow/blue = average reward obtained so far

$$\left.\begin{aligned} Q(y) &\to Q(y) + \varepsilon(r_y - Q(y)) \\ Q(b) &\to Q(b) + \varepsilon(r_b - Q(b)) \end{aligned}\right\} \text{delta rule}$$

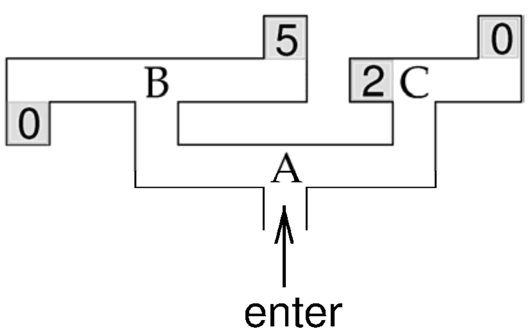
$$P(y) = \frac{\exp(\beta Q(y))}{\exp(\beta Q(y)) + \exp(\beta Q(b))}$$

$$P(b) = 1 - P(y)$$

Simulating Bees



Selecting Actions when Reward is Delayed

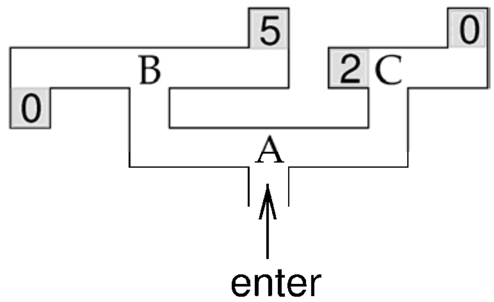


States: A, B, or C

Possible actions at any state: Left (L) or Right (R)

If you randomly choose to go L or R (random "policy"), what is the *value v of each state*?

Policy Evaluation



For random policy:

$$v(B) = \frac{1}{2}\cdot 0 + \frac{1}{2}\cdot 5 = 2.5$$

$$v(C) = \frac{1}{2}\cdot 2 + \frac{1}{2}\cdot 0 = 1$$

$$v(A) = \frac{1}{2}\cdot v(B) + \frac{1}{2}\cdot v(C) = 1.75$$

(Location, action) ⇨ new location

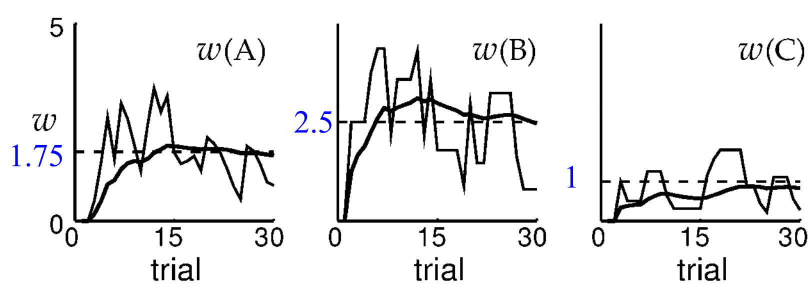
(u,a) ⇨ u'

Let $v(u) = w(u)$

Can learn this using TD learning:

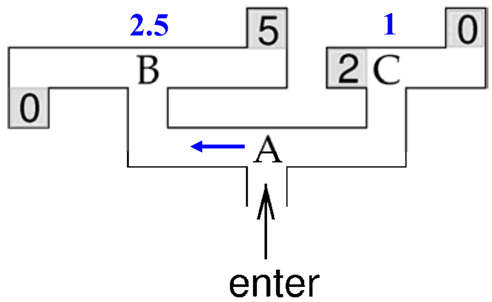
$$w(u) \rightarrow w(u) + \varepsilon[r_a(u) + v(u') - v(u)]$$

Maze Value Learning for Random Policy



Once I know the values, I can pick the action that leads to the higher valued state!

Selecting Actions based on Values



Values act as surrogate immediate rewards → Locally optimal choice leads to globally optimal policy (for Markov environments)

Related to *Dynamic Programming* in CS (see appendix in text)

Q learning

- A simple method for action selection based on action values (or Q values) $Q(x,a)$ where x is a state and a is an action

1. Let u be the current state. Select an action a according to:

$$P(a) = \frac{\exp(\beta Q(u,a))}{\sum_{a'} \exp(\beta Q(u,a'))}$$

2. Execute a and record new state u' and reward r. Update Q:

$$Q(u,a) \rightarrow Q(u,a) + \varepsilon(r + \max_{a'} Q(u',a') - Q(u,a))$$

3. Repeat until an end state is reached

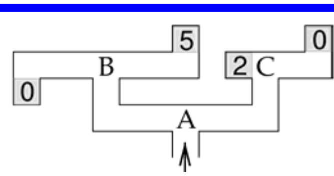
Actor-Critic Learning

- Two separate components: Actor (maintains policy) and Critic (maintains value of each state)

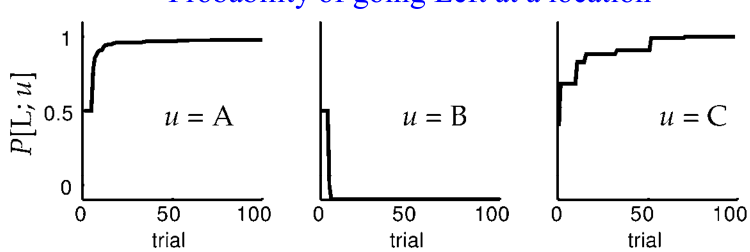
1. Critic Learning ("Policy Evaluation"):
   Value of state $u = v(u) = w(u)$
   $$w(u) \rightarrow w(u) + \varepsilon\,[r_a(u) + v(u') - v(u)]$$ (same as TD rule)
2. Actor Learning ("Policy Improvement"):
   $$P(a;u) = \frac{\exp(\beta Q_a(u))}{\sum_b \exp(\beta Q_b(u))}$$ Use this to select an action a in u

   For all a':
   $$Q_{a'}(u) \rightarrow Q_{a'}(u) + \varepsilon[r_a(u) + v(u') - v(u)](\delta_{aa'} - P(a';u))$$
3. Interleave 1 and 2

Actor-Critic Learning in the Maze Task

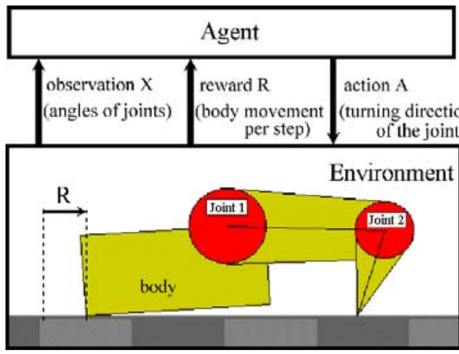


Probability of going Left at a location



Demo of Reinforcement Learning in a Robot

(from http://sysplan.nams.kyushu-u.ac.jp/gen/papers/JavaDemoML97/robodemo.html)



Reinforcement Learning Applications

Example: Flying a helicopter via reinforcement learning (videos)
(work of Andrew Ng, Stanford)

Things to do:

Read Chapter 9

Work on mini-project

Have a nice weekend!