Today’s Agenda

- 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
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?)

The Reinforcement Learning “Agent”

![Diagram of Reinforcement Learning](image)
Early Results: Pavlov and his Dog

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

![Image](http://employees.csbsju.edu/tcreed/pb/pdoganim.html)

Predicting Reward

- Stimulus $u = 0$ or $1$
- Expected reward $\nu = wu$
- Delivered reward = $r$
- Learn $w$ by minimizing $(r - \nu)^2$
  \[ w \rightarrow w + \varepsilon (r - \nu)u \]  
  (same as the delta rule; also called Rescorla-Wagner rule)
- Prediction error $\delta = (r - \nu)$
- For small $\varepsilon$ and $u = 1$, $w \rightarrow w + \varepsilon (r - w)$
  - Average value of $w = \langle w \rangle = \langle r \rangle$
Predicting Reward during Conditioning

- Reward 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 \( t \)

\[
v(t) \approx \left( \sum_{\tau=0}^{T-t} r(t + \tau) \right)
\]
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:

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

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

For each time step $t$, do:

For all $\tau (0 \leq \tau \leq t)$, do:

\[
\delta = \sum_{\tau=0}^{t} w(\tau) u(t-\tau)
\]

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

\[
w(\tau) \rightarrow w(\tau) + \varepsilon [r(t) + v(t+1) - v(t)] u(t-\tau)
\]

- Expected future reward
- Prediction

Predicting Delayed Reward: TD Learning

- Stimulus at $t=100$ and reward at $t=200$

Prediction error $\delta$ for each time step (over many trials)

R. Rao, 528: Lecture 15
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

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

More Evidence for Prediction Error Signals

Dopaminergic cells in Ventral Tegmental Area

Negative error

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

$[r(t) + v(t + 1) - v(t)] = -v(t)$
That’s great, but how does all that math help me get food in this maze?

Using Reward Predictions to Select Actions

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

$$P(a) = \frac{\exp(\beta Q(a))}{\sum_{a'} \exp(\beta Q(a'))} \quad \text{(High } \beta \text{ selects actions with highest } Q \text{ value. Low } \beta \text{ selects more uniformly)}$$
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.

\[ Q(y) \rightarrow Q(y) + \varepsilon (r_y - Q(y)) \]
\[ Q(b) \rightarrow Q(b) + \varepsilon (r_b - Q(b)) \]

\[ P(y) = \frac{\exp(\beta Q(y))}{\exp(\beta Q(y)) + \exp(\beta Q(b))} \]
\[ P(b) = 1 - P(y) \]

Simulating Bees

- \( \beta = 1 \) (exploration possible)
- \( \beta = 50 \) (mostly exploitation)
Forget bees, just tell me how to get to the food in this maze.

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 \( v \) of each state?
Policy Evaluation

For random policy:

\[ v(B) = \frac{1}{2} \cdot 0 + \frac{1}{2} \cdot 5 = 2.5 \]
\[ v(C) = \frac{1}{2} \cdot 2 + \frac{1}{2} \cdot 0 = 1 \]
\[ v(A) = \frac{1}{2} \cdot v(B) + \frac{1}{2} \cdot v(C) = 1.75 \]

Can learn this using TD learning:

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

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 $Q$:

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

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 \( u = v(u) = w(u) \)
   
   \( w(u) \rightarrow w(u) + \varepsilon \left[ r_a(u) + v(u') - v(u) \right] \) (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 \( u \)

   \[Q_a(u) \rightarrow Q_a(u) + \varepsilon [r_a(u) + v(u') - v(u)][\delta_{a'a} - P(a'; u)]\]

3. **Interleave 1 and 2**

---

### Actor-Critic Learning in the Maze Task

**Probability of going Left at a location**

- **\( u = A \)**
- **\( u = B \)**
- **\( u = C \)**

R. Rao, 528: Lecture 15
Learning to Solve the Water Maze Task

- Rat needs to swim to platform
- Current state input (location) from place cells in hippocampus
- Rat learns to find direct path to platform

A Ratty Comparison

A Average performance of 12 rats

B Actor-Critic Model
Demo of Reinforcement Learning in a Robot
(from http://www.fe.dis.titech.ac.jp/~gen/index.html)

Things to do:
Read Chapter 9
Work on mini-project

Have a nice weekend!