Chapter 2: Evaluative Feedback

- **Evaluating** actions vs. **instructing** by giving correct actions
- Pure evaluative feedback depends totally on the action taken. Pure instructive feedback depends not at all on the action taken.
- Supervised learning is instructive; optimization is evaluative
- **Associative vs. Nonassociative:**
  - Associative: inputs mapped to outputs; learn the best output for each input
  - Nonassociative: “learn” (find) one best output
- *n*-armed bandit (at least how we treat it) is:
  - Nonassociative
  - Evaluative feedback
The $n$-Armed Bandit Problem

- Choose repeatedly from one of $n$ actions; each choice is called a play.
- After each play $a_t$, you get a reward $r_t$, where
  \[ E\langle r_t | a_t \rangle = Q^*(a_t) \]
  These are unknown action values. Distribution of $r_t$ depends only on $a_t$.
- Objective is to maximize the reward in the long term, e.g., over 1000 plays.

To solve the $n$-armed bandit problem, you must explore a variety of actions and the exploit the best of them.
The Exploration/Exploitation Dilemma

- Suppose you form estimates
  \[ Q_t(a) \approx Q^*(a) \quad \text{action value estimates} \]

- The **greedy** action at \( t \) is
  \[ a_t^* = \arg \max_a Q_t(a) \]
  \[ a_t = a_t^* \Rightarrow \text{exploitation} \]
  \[ a_t \neq a_t^* \Rightarrow \text{exploration} \]

- You can’t exploit all the time; you can’t explore all the time
- You can never stop exploring; but you should always reduce exploring
Action-Value Methods

- Methods that adapt action-value estimates and nothing else, e.g.: suppose by the $t$-th play, action $a$ had been chosen $k_a$ times, producing rewards $r_1, r_2, \ldots, r_{k_a}$, then

$$Q_t(a) = \frac{r_1 + r_2 + \cdots r_{k_a}}{k_a}$$

“sample average”

- $\lim_{k_a \to \infty} Q_t(a) = Q^*(a)$
**ε-Greedy Action Selection**

- Greedy action selection:
  \[
  a_t = a_t^* = \arg\max_a Q_t(a)
  \]

- ε-Greedy:
  \[
  a_t = \begin{cases} 
  a_t^* & \text{with probability } 1 - \varepsilon \\
  \text{random action} & \text{with probability } \varepsilon 
  \end{cases}
  \]

... the simplest way to try to balance exploration and exploitation
10-Armed Testbed

- $n = 10$ possible actions
- Each $Q^*(a)$ is chosen randomly from a normal distribution: $\eta(0,1)$
- Each $r_t$ is also normal: $\eta(Q^*(a_t),1)$
- 1000 plays
- repeat the whole thing 2000 times and average the results
ε-Greedy Methods on the 10-Armed Testbed

![Graphs showing the performance of ε-greedy methods with different ε values.]
Softmax Action Selection

- Softmax action selection methods grade action probs. by estimated values.
- The most common softmax uses a Gibbs, or Boltzmann, distribution:

Choose action \( a \) on play \( t \) with probability

\[
\frac{e^{Q_t(a)/\tau}}{\sum_{b=1}^{n} e^{Q_t(b)/\tau}}
\]

where \( \tau \) is the “computational temperature”
Binary Bandit Tasks

Suppose you have just two actions: \( a_t = 1 \) or \( a_t = 2 \)
and just two rewards: \( r_t = \text{success} \) or \( r_t = \text{failure} \)

Then you might infer a target or desired action:

\[
d_t = \begin{cases} 
a_t & \text{if success} \\
\text{the other action} & \text{if failure}
\end{cases}
\]

and then always play the action that was most often the target

Call this the \textbf{supervised algorithm}
It works fine on deterministic tasks…
The space of all possible binary bandit tasks:
Linear Learning Automata

Let $\pi_t(a) = \text{Pr}\{a_t = a\}$ be the only adapted parameter

$L_{R-I}$ (Linear, reward - inaction)
- On success: $\pi_{t+1}(a_t) = \pi_t(a_t) + \alpha(1 - \pi_t(a_t)) \quad 0 < \alpha < 1$
  (the other action probs. are adjusted to still sum to 1)
- On failure: no change

$L_{R-P}$ (Linear, reward - penalty)
- On success: $\pi_{t+1}(a_t) = \pi_t(a_t) + \alpha(1 - \pi_t(a_t)) \quad 0 < \alpha < 1$
  (the other action probs. are adjusted to still sum to 1)
- On failure: $\pi_{t+1}(a_t) = \pi_t(a_t) + \alpha(0 - \pi_t(a_t)) \quad 0 < \alpha < 1$

For two actions, a stochastic, incremental version of the supervised algorithm
Performance on Binary Bandit Tasks A and B

**BANDIT A**

<table>
<thead>
<tr>
<th>Plays</th>
<th>0</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>500</th>
</tr>
</thead>
<tbody>
<tr>
<td>% Optimal action</td>
<td>50%</td>
<td>60%</td>
<td>70%</td>
<td>80%</td>
<td>90%</td>
<td>100%</td>
</tr>
<tr>
<td>Action values</td>
<td>LR-I</td>
<td>LR-P</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**BANDIT B**

<table>
<thead>
<tr>
<th>Plays</th>
<th>0</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>500</th>
</tr>
</thead>
<tbody>
<tr>
<td>% Optimal action</td>
<td>50%</td>
<td>60%</td>
<td>70%</td>
<td>80%</td>
<td>90%</td>
<td>100%</td>
</tr>
<tr>
<td>Action values</td>
<td>LR-I</td>
<td>LR-P</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

R. S. Sutton and A. G. Barto: Reinforcement Learning: An Introduction
Incremental Implementation

Recall the sample average estimation method:

The average of the first $k$ rewards is (dropping the dependence on $a$):

$$Q_k = \frac{r_1 + r_2 + \cdots + r_k}{k}$$

Can we do this incrementally (without storing all the rewards)?

We could keep a running sum and count, or, equivalently:

$$Q_{k+1} = Q_k + \frac{1}{k+1}[r_{k+1} - Q_k]$$

This is a common form for update rules:

$$NewEstimate = OldEstimate + StepSize[Target - OldEstimate]$$
Tracking a Nonstationary Problem

Choosing $Q_k$ to be a sample average is appropriate in a stationary problem,
    i.e., when none of the $Q^*(a)$ change over time,

But not in a nonstationary problem.

Better in the nonstationary case is:

$$Q_{k+1} = Q_k + \alpha[r_{k+1} - Q_k]$$

for constant $\alpha$, $0 < \alpha \leq 1$

$$= (1-\alpha)^k Q_0 + \sum_{i=1}^{k} \alpha(1-\alpha)^{k-i} r_i$$

*exponential, recency-weighted average*
Optimistic Initial Values

- All methods so far depend on $Q_0(a)$, i.e., they are biased.
- Suppose instead we initialize the action values optimistically, i.e., on the 10-armed testbed, use $Q_0(a) = 5$ for all $a$. 

---

% Optimal action

optimistic, greedy $Q_0 = 5$, $\epsilon = 0$  
realistic, $\epsilon$-greedy $Q_0 = 0$, $\epsilon = 0.1$
Reinforcement Comparison

- Compare rewards to a reference reward, \( \bar{r}_t \), e.g., an average of observed rewards
- Strengthen or weaken the action taken depending on \( r_t - \bar{r}_t \)
- Let \( p_t(a) \) denote the preference for action \( a \)
- Preferences determine action probabilities, e.g., by Gibbs distribution:
  \[
  \pi_t(a) = \Pr\{a_t = a\} = \frac{e^{p_t(a)}}{\sum_{b=1}^{n} e^{p_t(b)}}
  \]

- Then:
  \[
  p_{t+1}(a_t) = p_t(a) + \left[ r_t - \bar{r}_t \right] \quad \text{and} \quad \bar{r}_{t+1} = \bar{r}_t + \alpha \left[ r_t - \bar{r}_t \right]
  \]
Performance of a Reinforcement Comparison Method

\[ \epsilon\text{-greedy} \quad \epsilon = 0.1, \quad \alpha = 0.1 \]

\[ \epsilon\text{-greedy} \quad \epsilon = 0.1, \quad \alpha = 1/k \]

\[ \text{reinforcement comparison} \]
Pursuit Methods

- Maintain both action-value estimates and action preferences
- Always “pursue” the greedy action, i.e., make the greedy action more likely to be selected
- After the $t$-th play, update the action values to get $Q_{t+1}$
- The new greedy action is $a^*_{t+1} = \arg \max_a Q_{t+1}(a)$

Then:

$$\pi_{t+1}(a^*_{t+1}) = \pi_t(a^*_{t+1}) + \beta[1 - \pi_t(a^*_{t+1})]$$

and the probs. of the other actions decremented to maintain the sum of 1
Performance of a Pursuit Method

\[ \epsilon\text{-greedy} \quad \epsilon = 0.1, \quad \alpha = 1/k \]
Imagine switching bandits at each play
Conclusions

- These are all very simple methods
  - but they are complicated enough—we will build on them
- Ideas for improvements:
  - estimating uncertainties . . . interval estimation
  - approximating Bayes optimal solutions
  - Gittens indices
- The full RL problem offers some ideas for solution . . .