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
Try it....

- Arm 1
- Arm 2
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

\[
E \{ r_t \mid a_t \} = Q^*(a_t)
\]

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 **exploit** the best of them
The Exploration/Exploitation Dilemma

- Suppose you form estimates

\[ Q_t(a) \approx Q^*(a) \]

action value estimates

- The **greedy** action at \( t \) is \( a_t \)

\[ a_t^* = \arg \max_a Q_t(a) \]

- \( a_t = a_t^* \implies \text{exploitation} \)

- \( a_t \neq a_t^* \implies \text{exploration} \)

- 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. Maybe.
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 balance exploration and exploitation
10-Armed Testbed

- $n = 10$ possible actions
- Each $Q^*(a)$ is chosen randomly from a normal distrib.: $\eta(0, 1)$
- 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
Supervised Learning Attempt

- Suppose either returned 0 or 1 rewards….

\[
\pi_{t+1}(d_t) = \pi_t(d_t) + \alpha[1 - \pi_t(d_t)]
\]
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”
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:

$$\text{NewEstimate} = \text{OldEstimate} + \text{StepSize}[\text{Target} - \text{OldEstimate}]$$
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 $a$.
Conclusions

- These are all very simple methods
  - but they are complicated enough—we will build on them
  - we should understand them completely
Conclusions

- These are all very simple methods
  - but they are complicated enough—we will build on them
  - we should understand them completely

- Ideas for improvements:
  - estimating uncertainties . . . interval estimation
  - new “action elimination” methods (see ICML’03)
  - approximating Bayes optimal solutions

- The full RL problem offers some ideas for solution . . .