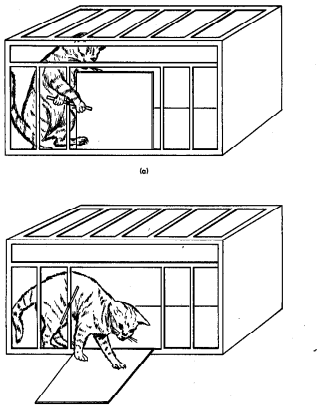
Lecture 2: Learning from Evaluative Feedback

or

"Bandit Problems"

Edward L. Thorndike (1874-1949)



Puzzle Box

Learning by Trial-and-Error

Law of Effect:

> "Of several responses to the same situation, those which are accompanied or closely followed by satisfaction to the animal will, other things being equal, be more firmly connected with the situation, so that, when it recurs, they will be more likely to recur; those which are accompanied or closely followed by discomfort to the animal will, other things being equal, have their connections with that situation weakened, so that when it recurs, they will be less likely to recur."
>
> Thorndike, 1911

Some Distinctions

- **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. **Non-associative**:
  - Associative: inputs mapped to outputs; learn the best output **for each** input
  - Non-associative: "learn" (find) one best output
- *n*-armed bandit (at least how we treat it) is:
  - Non-associative
  - Evaluative feedback

4

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 the **exploit** the best of them

5

The Exploration/Exploitation Dilemma

- Suppose you form estimates

$$Q_t(a) \approx Q^*(a)$$ **action value estimates**

- The **greedy** action at *t* is:

$$a_t^* = \arg\max_a Q_t(a)$$

$$a_t = a_t^* \Rightarrow \text{exploitation}$$

$$a_t \neq a_t^* \Rightarrow \text{exploration}$$

- 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

6

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"

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

7

ε-Greedy Action Selection

- Greedy action selection:

$$a_t = a_t^* = \arg\max_a Q_t(a)$$

- ε-Greedy:

$$a_t = \begin{cases} a_t^* & \text{with probability } 1 - \varepsilon \\ \text{random action} & \text{with probability } \varepsilon \end{cases}$$

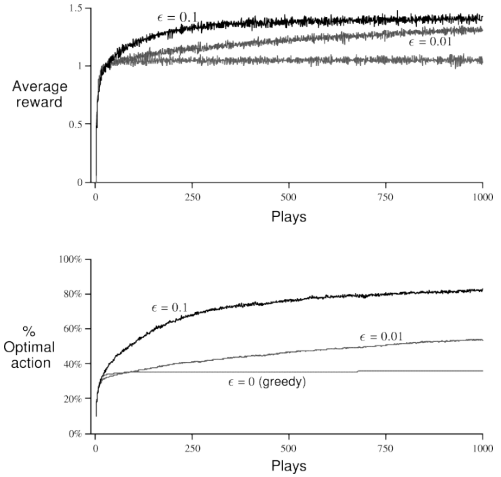
. . . the simplest way to try to balance exploration and exploitation

8

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



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 τ is the "computational temperature"

Binary Bandit Tasks

Suppose you have just **two** actions: $a_t = 1$ or $a_t = 2$

and just **two** rewards: $r_t =$ *success* or $r_t =$ *failure*

Then you might infer a **target** or **desired action**:

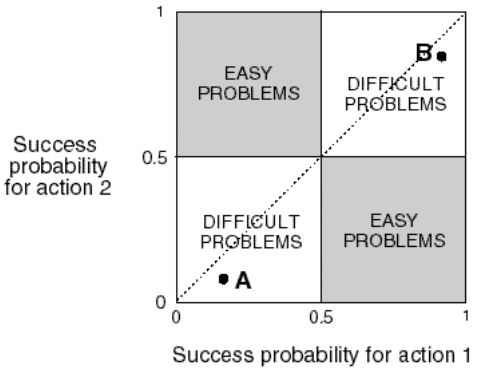
$$d_t = \begin{cases} a_t & \text{if } \textit{success} \\ \text{the other action} & \text{if } \textit{failure} \end{cases}$$

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

Call this the **supervised algorithm**
It works fine on deterministic tasks...

Contingency Space

The space of all possible binary bandit tasks:



Linear "Learning Automata"

Let $\pi_t(a) = \Pr\{a_t = a\}$ 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)

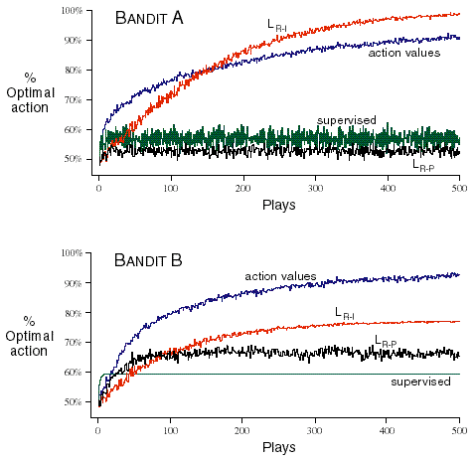
On *success*: $\pi_{t+1}(a_t) = \pi_t(a_t) + \alpha(1 - \pi_t(a_t))$ $\quad 0 < \alpha < 1$

(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



Incremental Implementation

Recall the sample average estimation method:

The average of the first k rewards is (dropping the dependence on a): $Q_k = \dfrac{r_1 + r_2 + \cdots r_k}{k}$

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

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

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

This is a common form for update rules:

NewEstimate = *OldEstimate* + *StepSize*[*Target* – *OldEstimate*]

Tracking a Non-stationary 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 non-stationary problem.

Better in the non-stationary case is:

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

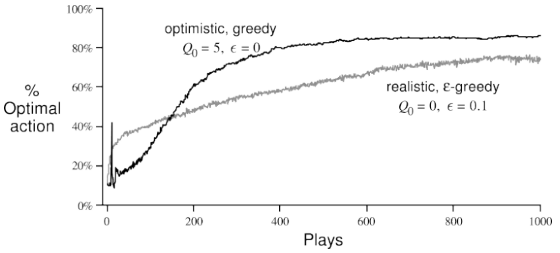
for *constant* α, $0 < \alpha \leq 1$

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

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



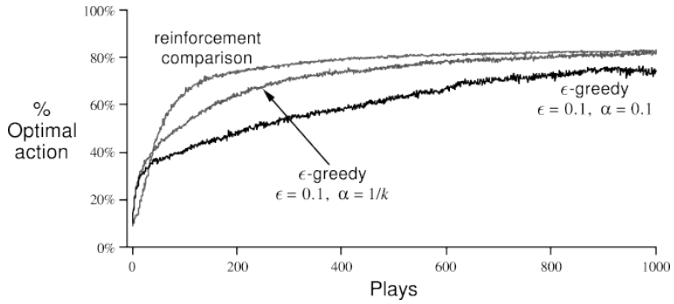
Reinforcement Comparison

- 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: $p_{t+1}(a_t) = p_t(a) + \left[r_t - \bar{r}_t\right]$ and $\bar{r}_{t+1} = \bar{r}_t + \alpha\left[r_t - \bar{r}_t\right]$

Performance of a Reinforcement Comparison Method



Pursuit Methods

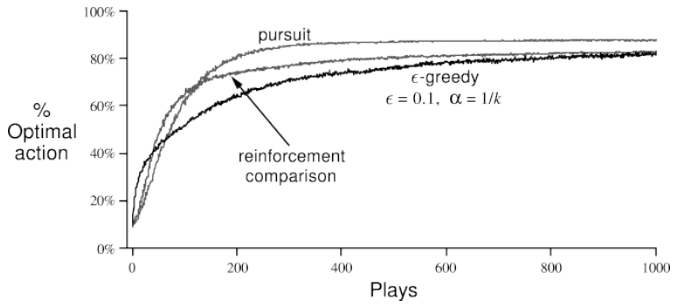
- Maintain both action-value estimates and action preferences
- Always "pursue" the greedy action, i.e., make the greedy action more likely to be selected
- After the t-th play, update the action values to get Q_{t+1}
- The new greedy action is $a^*_{t+1} = \arg\max_a Q_{t+1}(a)$
- Then:

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

and the probs. of the other actions decremented to maintain the sum of 1

21

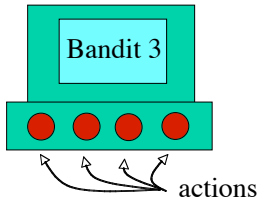
Performance of a Pursuit Method



22

Associative Search

Imagine switching bandits at each play



23

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 . . .

24

Next Class

- Read Chapter 3