Exploration-Exploitation tradeoffs and multi-armed bandit problems

CS7032: AI & Agents for IET

November 24, 2015

Learning and Feedback

- Consider the ACO algorithm: how does the system learn?
- Contrast that form of "learning" with, say, our system that learns to play draughts, or to a system that learns to filter out spam mail.

Learning and Feedback

- Consider the ACO algorithm: how does the system learn?
- Contrast that form of "learning" with, say, our system that learns to play draughts, or to a system that learns to filter out spam mail.
- The RL literature often contrasts instruction and evaluation.
- Evaluation is a key component of Reinforcement learning systems:
  - Evaluative feedback is local (it indicates how good an action is) but not whether it is the best action possible
  - This creates a need for exploration

Associative vs. non-associative settings

- In general, learning is both
  - selectional: i.e. actions are selected by trying different alternatives and comparing their effects,

Associative vs. non-associative settings

- In general, learning is both
  - selectional: i.e. actions are selected by trying different alternatives and comparing their effects,
  - associative: i.e. the actions selected are associated to particular situations.

Associative vs. non-associative settings

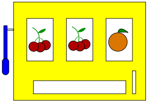
- In general, learning is both
  - selectional: i.e. actions are selected by trying different alternatives and comparing their effects,
  - associative: i.e. the actions selected are associated to particular situations.
- However, in order to study evaluative feedback in detail it is convenient to simplify things and consider the problem from a non-associative perspective.

Learning from fruit machines

- The n-armed bandit setting:
- Choice of n actions (which yield numerical rewards drawn from a stationary probability distribution)
  - each action selection called a play
- Goal: maximise expected (long term) total reward or return.
  - Strategy: concentrate plays on the best levers.

Learning from fruit machines

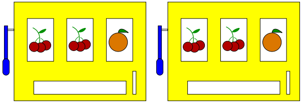
- The n-armed bandit setting:



- Choice of n actions (which yield numerical rewards drawn from a stationary probability distribution)
  - each action selection called a play
- Goal: maximise expected (long term) total reward or return.
  - Strategy: concentrate plays on the best levers.

Learning from fruit machines

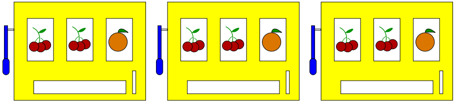
- The n-armed bandit setting:



- Choice of n actions (which yield numerical rewards drawn from a stationary probability distribution)
  - each action selection called a play
- Goal: maximise expected (long term) total reward or return.
  - Strategy: concentrate plays on the best levers.

Learning from fruit machines

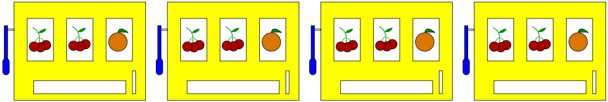
- The n-armed bandit setting:



- Choice of n actions (which yield numerical rewards drawn from a stationary probability distribution)
  - each action selection called a play
- Goal: maximise expected (long term) total reward or return.
  - Strategy: concentrate plays on the best levers.

Learning from fruit machines

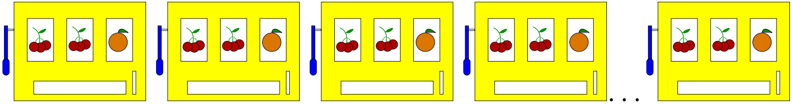
- The n-armed bandit setting:



- Choice of n actions (which yield numerical rewards drawn from a stationary probability distribution)
  - each action selection called a play
- Goal: maximise expected (long term) total reward or return.
  - Strategy: concentrate plays on the best levers.

Learning from fruit machines

- The n-armed bandit setting:



- Choice of n actions (which yield numerical rewards drawn from a stationary probability distribution)
  - each action selection called a play
- Goal: maximise expected (long term) total reward or return.
  - Strategy: concentrate plays on the best levers.

How to find the best plays

- Let's distinguiguish between:
  - reward, the immediate outcome of a play, and
  - value, the expected (mean) reward of a play
- But how do we estimate values?
- We could keep a record of rewards $r_1^a, \ldots, r_{k_a}^a$ for each chosen action a and estimate the value of choosing a at time t as

$$Q_t(a) = \frac{r_1^a + r_2^a + \cdots + r_{k_a}^a}{k_a} \tag{1}$$

But how do we choose an action?

- We could be greedy and exploit our knowledge of $Q_t(a)$ to choose at any time t the play with the highest estimated value.
- But this strategy will tend to neglect the estimates of non-greedy actions (which might have produced greater total reward in the long run).
- One could, alternatively, explore the space of actions by choosing, from time to time, a non-greedy action

Balancing exploitation and exploration

- The goal is to approximate the true value, $Q(a)$ for each action
- $Q_t(a)$, given by equation (1) will converge to $Q(a)$ as $k_a \to \infty$ (Law of Large Numbers)
- So balancing exploration and exploitation is necessary...
- but finding the right balance can be tricky; many approaches rely on strong assumptions about the underlying distributions
- We will present a simpler alternative...

A simple strategy

- The simplest strategy: choose action a^* so that:

$$a^* = \arg\max_a Q_t(a) \qquad \text{(greedy selection)} \tag{2}$$

- A better simple strategy: ϵ-greedy methods:
  - With probability $1 - \epsilon$, exploit; i.e. choose a^* according with (2)
  - The rest of the time (probability ϵ) choose an action at random, uniformly
  - A suitably small ϵ guarantees that all actions get explored sooner or later (and the probability of selecting the optimal action converges to greater than $1 - \epsilon$)

Exploring exploration at different rates

- The "10-armed testbed" [Sutton and Barto, 1998]:
  - 2000 randomly generated n-armed bandit tasks with $n = 10$.
  - For each action, a, expected rewards $Q(a)$ (the "true" expected values of choosing a) are selected from a normal (Gaussian) probability distribution $N(0, 1)$
  - Immediate rewards $r_1^a, \ldots, r_k^a$ are similarly selected from $N(Q(a), 1)$

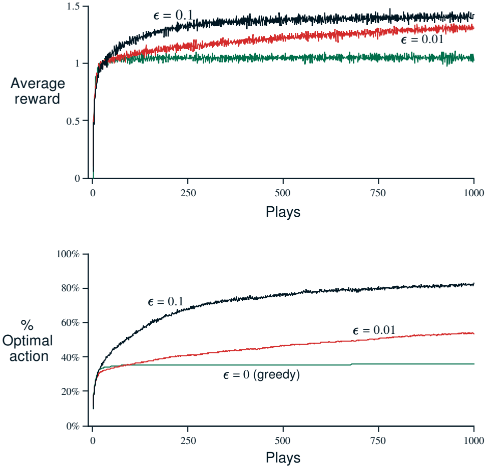
Exploring exploration at different rates

- The "10-armed testbed" [Sutton and Barto, 1998]:
  - 2000 randomly generated n-armed bandit tasks with $n = 10$.
  - For each action, a, expected rewards $Q(a)$ (the "true" expected values of choosing a) are selected from a normal (Gaussian) probability distribution $N(0, 1)$
  - Immediate rewards $r_1^a, \ldots, r_k^a$ are similarly selected from $N(Q(a), 1)$

```
nArmedBanditTask <- function(n=10, s=2000)
  {
      qtrue <- rnorm(n)
      r <- matrix(nrow=s,ncol=n)
      for (i in 1:n){
          r[, i] <- rnorm(s,mean=qtrue[i])
      }
      return(r)
  }
```

Some empirical results

- Averaged results of 2000 rounds of 1000 plays each:



Softmax methods

- Softmax action selection methods grade action probabilities by estimated values.
- Commonly use Gibbs, or Boltzmann, distribution. I.e. choose action a on the t-th play with probability

$$\pi(a) = \frac{e^{Q_t(a)/\tau}}{\sum_{b \in A} e^{Q_t(b)/\tau}} \tag{3}$$

- where τ is a positive parameter called the *temperature*, as in simulated annealing algorithms.

Keeping track of the means

- Maintaining a record of means by recalculating them after each play can be a source of inefficiency in evaluating feedback. E.g.:

```
## inefficient way of selecting actions
selectAction <- function(r, e=0) {
  n <- dim(r)[2] # number of arms (cols in r)
  if (sample(c(T,F),1,prob=c(e,1-e))) {
    return(sample(1:n,1)) # explore if e > 0 and chance allows
  }
  else { # all Q_t(a_i), i \in [1,n]
    Qt <- sapply(1:n, function(i)mean(r[,i], na.rm=T))
    ties <- which(Qt == max(Qt))
    ## if there are ties in Q_t(a); choose one at random
    if (length(ties)>1)
      return(sample(ties, 1))
    else
      return(ties)
  }
}
```

Incremental update

- Do we really need to store all rewards?

Incremental update

- Do we really need to store all rewards? NO

Incremental update

- Do we really need to store all rewards? NO
- Let Q_k be the mean of the first k rewards for an action,
- We can express the next mean value as

$$
\begin{aligned}
Q_{k+1} &= \frac{1}{k+1}\sum_{i=1}^{k+1} r_i \\
&= \dots \\
&= Q_k + \frac{1}{k+1}[r_{k+1} - Q_k]
\end{aligned} \tag{4}
$$

Incremental update

- Do we really need to store all rewards? NO
- Let Q_k be the mean of the first k rewards for an action,
- We can express the next mean value as

$$
\begin{aligned}
Q_{k+1} &= \frac{1}{k+1}\sum_{i=1}^{k+1} r_i \\
&= \dots \\
&= Q_k + \frac{1}{k+1}[r_{k+1} - Q_k]
\end{aligned} \tag{4}
$$

- Equation (4) is part of family of formulae in which the new estimate (Q_{k+1}) is the old estimate (Q_k) adjusted by the estimation error ($r_{k+1} - Q_k$) scaled by a step parameter ($\frac{1}{k+1}$, in this case)
- (Recall the LMS weight update step described in the introduction to machine learning).

Non-stationary problems

- What happens if the "n-armed bandit" changes over time? How do we keep track of a changing mean?
- An approach: weight recent rewards more heavily than past ones.
- So, our update formula (4) could be modified to

$$Q_{k+1} = Q_k + \alpha[r_{k+1} - Q_k] \tag{5}$$

  where $0 < \alpha \leq 1$ is a constant
- This gives us Q_k as an exponential, recency-weighted average of past rewards and the initial estimate

Recency weighted estimates

- Given (5), Q_k can be rewritten as:

$$\begin{aligned} Q_k &= Q_{k-1} + \alpha[r_k - Q_{k-1}] \\ &= \alpha r_k + (1-\alpha) Q_{k-1} \\ &= \ldots \\ &= (1-\alpha)^k Q_0 + \sum_{i=1}^{k} \alpha (1-\alpha)^{k-i} r_i \end{aligned} \tag{6}$$

- Note that $(1-\alpha)^k + \sum_{i=1}^{k} \alpha(1-\alpha)^{k-i} = 1$. I.e. weights sum to 1.
- The weight $\alpha(1-\alpha)^{k-i}$ decreases exponentially with the number of intervening rewards

Picking initial values

- The above methods are biased by Q_0
  - For sample-average, bias disappears once all actions have been selected
  - For recency-weighted methods, bias is permanent (but decreases over time)
- One can supply prior knowledge by picking the right values for Q_0
- One can also choose optimistic initial values to encourage exploration

Picking initial values

- The above methods are biased by Q_0
  - For sample-average, bias disappears once all actions have been selected
  - For recency-weighted methods, bias is permanent (but decreases over time)
- One can supply prior knowledge by picking the right values for Q_0
- One can also choose optimistic initial values to encourage exploration (Why?)

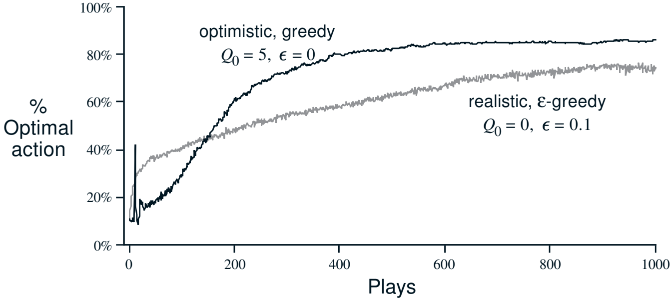
The effects of optimism

The effects of optimism

- With Q_0's set initially high (for each action)q, the learner will be disappointed by actual rewards, and sample all actions many times before converging.

The effects of optimism

- With Q_0's set initially high (for each action)q, the learner will be disappointed by actual rewards, and sample all actions many times before converging.
- E.g.: Performance of Optimistic ($Q_0 = 5, \forall a$) vs Realistic ($Q_0 = 0, \forall a$) strategies for the 10-armed testbed

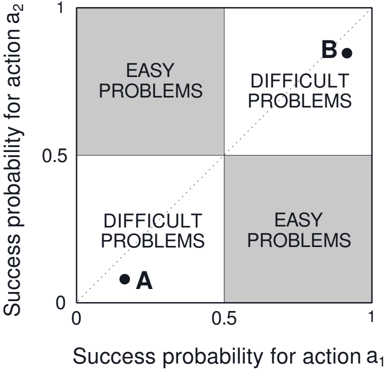


Evaluation vs Instruction

- RL searches the action space while SL searchs the parameter space.
- Binary, 2-armed bandits:
  - two actions: a_1 and a_2
  - two rewards: success and failure (as opposed to numeric rewards).
- SL: select the action that returns success most often.

Evaluation vs Instruction

- RL searches the action space while SL searchs the parameter space.
- Binary, 2-armed bandits:
  - two actions: a_1 and a_2
  - two rewards: success and failure (as opposed to numeric rewards).
- SL: select the action that returns success most often.
- Stochastic case (a problem for SL?):



Two stochastic SL schemes (learning automata)

- $L_{r\text{-}p}$, Linear reward-penalty: choose the action as in the naive SL case, and keep track of the successes by updating an estimate of the probability of choosing it in future.
- Choose $a \in A$ with probability $\pi_t(a)$
- Update the probability of the chosen action as follows:

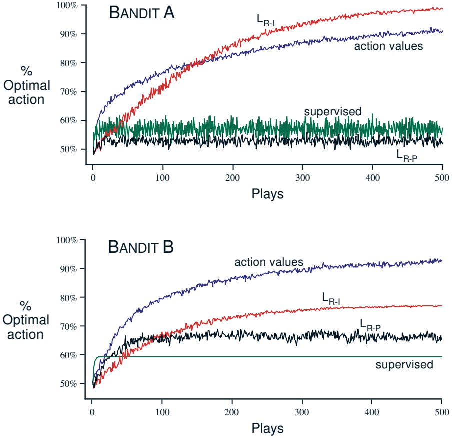
$$\pi_{t+1}(a) = \begin{cases} \pi_t(a) + \alpha(1 - \pi_t(a)) & \text{if success} \\ \pi_t(a)(1 - \alpha) & \text{otherwise} \end{cases} \tag{7}$$

- All remaining actions $a_i \neq a$ are adjusted proportionally:

$$\pi_{t+1}(a_i) = \begin{cases} \pi_t(a_i)(1 - \alpha) & \text{if } a \text{ succeeds} \\ \frac{\alpha}{|A|-1} + \pi_t(a_i)(1 - \alpha) & \text{otherwise} \end{cases} \tag{8}$$

- $L_{r\text{-}i}$, Linear reward-inaction: like L_{r-p} but update probabilities only in case of success.

Performance: instruction vs evaluation



- See [Narendra and Thathachar, 1974], for a survey of learning automata methods and results.

Reinforcement comparison

- Instead of estimates of values for each action, keep an estimate of overall reward level $\bar{r}_t$:

$$\bar{r}_{t+1} = \bar{r}_t + \alpha[r_{t+1} - r_t] \tag{9}$$

- and action preferences $p_t(a)$ w.r.t. reward estimates:

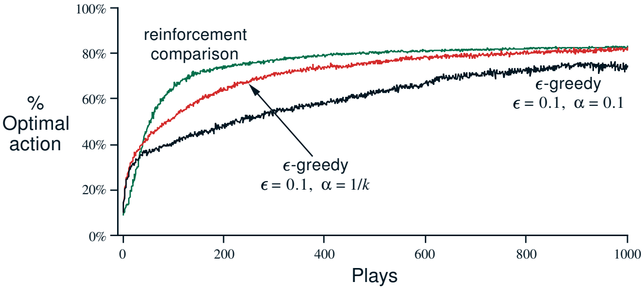
$$p_{t+1}(a) = p_t(a) + \beta[r_t - \bar{r}_t] \tag{10}$$

- The softmax function can then be used as a PMF for action selection:

$$\pi_t(a) = \frac{e^{p_t(a)}}{\sum_{b \in A} e^{p_t(b)}} \tag{11}$$

- ($0 < \alpha \leq 1$ and $\beta > 0$ are step-size parameters.)

How does reinforcement comparison compare?



- Reinforcement comparison ($\alpha = 0.1$) vs action-value on the 10-armed testbed.

Pursuit methods

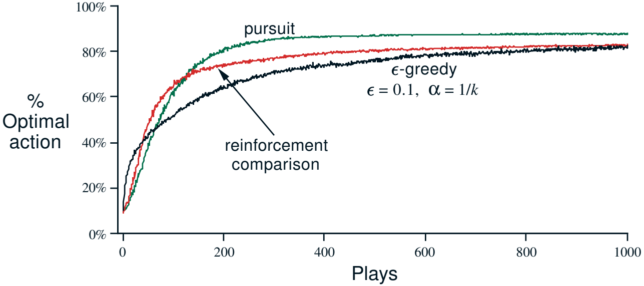
- maintain both action-value estimates and action preferences,
- preferences continually "pursue" greedy actions.
- E.g.: for greedy action $a^* = \arg\max_a Q_{t+1}(a)$, increase its selection probability:

$$\pi_{t+1}(a^*) = \pi_t(a^*) + \beta[1 - \pi_t(a^*)] \tag{12}$$

- while decreasing probabilities for all remaining actions $a \neq a^*$

$$\pi_{t+1}(a) = \pi_t(a) + \beta[0 - \pi_t(a)] \tag{13}$$

Performance of pursuit methods



- Pursuit ($\alpha = 1/k$ for Q_t update, $\pi_0(a) = 1/n$ and $\beta = 0.01$) versus reinforcement comparison ($\alpha = 0.1$) vs action-value on the 10-armed testbed.

Further topics

- Associative search:
  - suppose we have many different bandit tasks and the learner is presented with a different one at each play.
  - Suppose each time the learner is given a clue as to the which task it is facing...

Further topics

- Associative search:
  - suppose we have many different bandit tasks and the learner is presented with a different one at each play.
  - Suppose each time the learner is given a clue as to the which task it is facing...
  - In such cases, the best strategy is a combination of *search* for the best actions and association of actions to situations

Further topics

- Associative search:
  - suppose we have many different bandit tasks and the learner is presented with a different one at each play.
  - Suppose each time the learner is given a clue as to the which task it is facing...
  - In such cases, the best strategy is a combination of *search* for the best actions and association of actions to situations
- Exact algorithms for computation of Bayes optimal way to balance exploration and exploitation exist [Bellman, 1956], but are intractable.

References

Bellman, R. (1956).
A problem in the sequential design of experiments.
Sankhya, 16:221–229.

Narendra, K. and Thathachar, M. (1974).
Learning automata - a survey.
IEEE Transactions on Systems, Man and Cybernetics, SMC-4(4):323–334.

Sutton, R. S. and Barto, A. G. (1998).
Reinforcement Learning: An Introduction.
MIT Press, Cambridge, MA.