Lecture notes - Introduction to Reinforcement learning with David Silver

Lecture 9

By - Amal Sunny

Exploration and Exploitation

- So far, we've only tried very naive approaches to the problem of exploration (ϵ-greedy)
- Here, we explore better methods.

Exploration vs. Exploitation Dilemma

- So online decision-making involves this fundamenetal choice:
  - Exploitation: Making the best decision, given the info we currently have
  - Exploration: Gathering more information (to possibly make better decisions down the line)
- Now, the best long-term strategy may involve short-term sacrifices to find better rewards down the line.
- Or alternatively we may already know the best plan, but waste actions trying to guarentee that is the best plan.
- But our goal remains the same, gather enough information to make the best overall decision.

Principles of Exploration

- Naive/Random Exploration
  - Explore random actions, introducing randomness to greedy actions
  - Eg: ϵ-greedy, softmax
- Optimism in the face of uncertainty
  - Given a situation where outcomes are unknown, we pick the more uncertain one which has higher potential for reward
  - For that, obiviously we'll have to measure uncertainity first
  - Then preference given to states/actions with highest uncertainity
- Information state space
  - In this, the information known by the agent is considered part of its state
    - Eg: A robot inside a room, and a robot inside the room knowing whats outside the room are in two different states
  - This way we can lookahead to see how the information helps our cumulative reward.

State exploration vs Parameter Exploration

- State action exploration
  - Systematically explores the state space/action space
  - Eg: Picking different actions every time we got to state S
- Parameter exploration
  - We take the parameterized policy $\pi(A|S,u)$
  - How we explore is by just modifying the parameters and seeing the agent in action
  - Advantages:
    - Consistent exploration: Will keep learning every time
  - Disadvantages:
    - No knowledge about state/action space is used. Like it can arrive at the same state, again and not benefit from knowledge of those visits as its parameter based.
- We focus on state-action exploration

Multi-Armed Bandit

- A multi-armed bandit is a simplified MDP, consisting of the tuple <A,R>
- A is the known set of actions (or the "arms" on the machine)
- $R^a(r) = P[R = r|A = a]$ is an unknown probability distribution of rewards for a given action a.
- At each step t, the agent selects an action $A_t \epsilon A$
- And the environment generates a reward for it $R_t \sim R^{A_t}$ which concludes an episode.
- Our end goal is to maximize cumulative rewards across these episodes $\sum_{r=1}^{t} R_t$

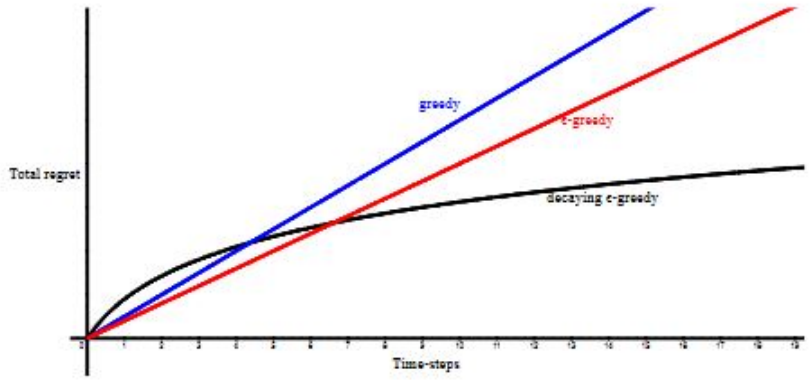
Regret

- Now, we flip our goal a bit here to calculate regret - essentially how much are we losing out by not picking the most optimal action.
- If we minimize that, that's the same as maximizing cumulative reward.
- The action-value is the mean reward for action a,
  - $$Q(a) = E[r|a]$$
- The optimal value V^* is:
  - $$V^* = Q(a^*) = max_{a \in A} Q(a)$$
- The regret is the opportunity loss for one step
  - $$I_t = E[V^* - Q(a_t)]$$
- The total regret is the total opportunity loss
  - $$L_t = E[\sum_{\tau=1}^{t} V^* - Q(a_\tau)]$$

Counting Regret

- We define two terms
  - Count $N_t(a)$ is expected number of selections for action a
  - Gap δ_a - is difference of value b/w taken action a and optimal action a*, $\delta_a = V^* - Q(a)$
- We can define regret as a function of gaps and the counts
  - $$L_t = E[\sum_{\tau=1}^{t} V^* - Q(a_\tau)]$$
    $$= \sum_{a \in AE} [Nt(a)](V^* - Q(a))$$
    $$= \sum_{a \in AE} [Nt(a)]\Delta_a$$
- Any good algorithm would try to ensure the count becomes small, as the gap becomes large.
- But that's the problem: We dont know the gaps(optimal v^* isn't known)

Linear or Sublinear Regret



- Above we have a plot of regret against time steps for a few algorithms.
- From that we can conclude,
  - If an algorithm explores forever, it will have linear regret
  - If an algorithm never explores, it will have linear regret
- We explore existing methods to see where we go wrong

Greedy Algorithm

- We take an algorithm that estimates value function for each Q(a) by MC-evaluation and picks the highest Q-value action.
- However, this one can lock onto one suboptimal action and just keep repeating it forever.
- Thus it has a linear total regret.

Greedy algorithm with Optimistic Initialisation

- Start by initializing values to max possible value, $Q(a) = r_{max}$
- Then act greedily,

    - $A_t = argmax_{a \epsilon A} Q_t(a)$
- Ensures exploration of each action atleast once
- However, a few unlucky samples can lock out optimal action forever.
- Optimistic greedy has linear total regret

ϵ-greedy algorithm

- Since we're likely to pick the non-optimal arm with probability ϵ, we can end up accumulating regret each step by pulling the wrong one.
- So regret will keep increasing linearly $\implies$ ϵ-greedy has linear total regret.
- Similarly for softmax exploration.

Decaying ϵ_t-Greedy Algorithm

- Pick a decay schedule for $\epsilon_1, \epsilon_2, ...$
- Consider the following schedule

$$c > 0$$
$$d = min_{a|\Delta_a > 0} \Delta_i$$
$$\epsilon_t = min\{1, \frac{c|A|}{d^2 t}\}$$

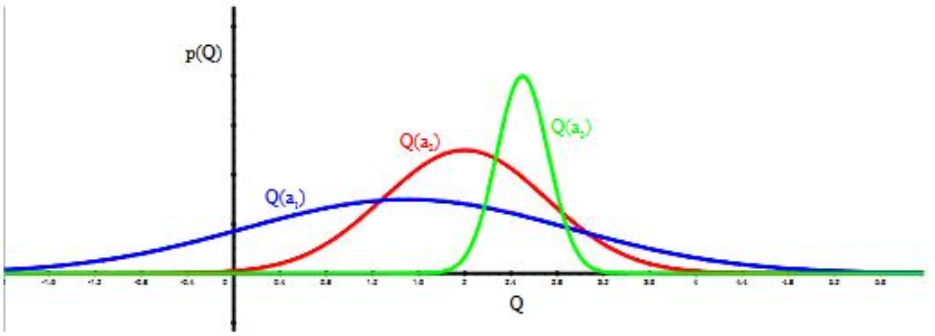
- The term for decay d, is the smallest gap that's non-zero. By having it in the denominator, we tend to explore more when gap is small and explore less when gap is big.
- However, this is not the only choice of d that may give rise to a schedule like this. Other options can also be taken which give similar results
- Decaying ϵ_t -greedy has logarithmic asymptotic total regret.
- Unfortunately, even this algorithm requires advance knowledge of gaps(which we don't have usually)
- So our goal becomes: finding an algorithm with sublinear regret for any multi-armed bandit (without knowledge of R).

Lower Bound

- The performance of any algorithm is determined by similarity between optimal arms and other arms, i.e how their reward distributions look like.
- Hard problems would have similar looking arms, but with different means - so it becomes much harder to pick based on samples.
- This is described formally by the gap δ_a and the similarity in distributions $KL(R^a || R^a \star)$

> Theorem (Lai and Robbins)
> Asymptotic total regret is at least logarithmic in number of steps
>
> $$\lim_{t \to \infty} L_t \geq \log t \sum_{a|\Delta_a > 0} \frac{\Delta_a}{KL(R^a || R^{a^*})}$$

Optimism in the Face of Uncertainity



- Given a distribution of arms like this, which action should we pick ?
- So now, by Optimism in the Face of Uncertainity, we would pick the action-value which has higher potential, that we're uncertain about - until it is no longer the highest potential one.
  - i.e. blue one here
- We explore the uncertain one, and after gaining enough information - we decide whether to keep going or explore/exploit a different arm.
- This way, we'll just end up picking the most optimal one as enough time passes.

Upper confidence Bounds (UCB)

- The basic idea for UCB is similar to that of confidence intervals. We basically assign an upper bound for each action-value based on some confidence level that it'll occur(also can be read as high probability)
- So now the UCB, is the upper bound of that distribution given a certain confidence level.
- This is basically quantifying potential of possible arms to pick, such that we can explore more effectively.
- Essentially, estimate upper confidence $U_t(a)$ for each action value
  - Such that q(a) $\leq Q_t(a) + U_t(a)$ with high probability
- Now, obviously this UCB depends on N(a) - number of times that action has been picked
  - Small N(a) => Large U(a) (estimate uncertain)
  - Large N(a) => Small U(a) (estimate more accurate)
- And then we pick the action with the highest UCB.
  - $$A_t = argmax_{a \epsilon A} Q_t(a) + U_t(a)$$
- There are two ways to approach this problem now
  - Making no assumptions about the distribution
  - Making some assumptions (Bayesian)
- We start off with the former (no assumptions)

Hoeffding's Inequality - Frequentist Approach

> **Theorem (Hoeffding's Inequality)**
> Let $X_1, ..., X_t$ be i.i.d. random variables in [0,1], and let $\bar{X}_t = 1/\tau \sum_{\tau=1}^{t} X_\tau$ be the sample mean. Then

$$P[E[X] > \bar{X}_t + u] \leq e^{-2tu^2}$$

- We will apply Hoeffding's Inequality to rewards of the bandit
- Condition on selecting action a
  - $$P[Q(a) > \bar{Q_t}(a) + U_t(a)] \leq e^{-2N_t(a)U_t(a)^2}$$

Calculating Upper Confidence Bounds - UCB1 Algorithm

- We pick a probability p that true value exceeds UCB (confidence interval %)
- And then solve for $U_t(a)$

$$e^{-2N_t(a)U_t(a)^2} = p$$
$$U_t(a) = \sqrt{\frac{-\log p}{2N_t(a)}}$$

- Now we don't exactly set a static p, we reduce it as we observe more rewards
  - Eg: p = t^{-4}
- Ensures we select optimal action as $t \to \infty$

$$U_t(a) = \sqrt{\frac{2\log t}{N_t(a)}}$$

- Which leads to the **UCB1 algorithm**

$$a_t = argmax_{a\epsilon A}Q(a) + \sqrt{\frac{2\log t}{N_t(a)}}$$

> Theorem
> The UCB algorithm achieves logarithmic asymptotic total regret
>
> $$\lim_{t\to\infty} L_t \leq 8\log t \sum_{a|\delta_a>0} \delta_a$$

Other Upper confidence Bounds

- After Hoeffding's, other inequalities started being appplied to UCBs to get tighter bounds, different bounds for different distributions, etc. like:
  - Bernstein's inequality
  - Chernoff inequality
  - Azuma's inequality

Bayesian Bandits

- Bayesian Bandits exploits prior knowledge about reward distribution, $p[R^a]$
- Consider a distribution p[Q|w] over action-value function, parametrized by w
  - Eg: independent gaussian: $w = [\mu_1, \sigma_1^2, \vdots, \mu_k, \sigma_k^2]$ for $a\epsilon[1, k]$

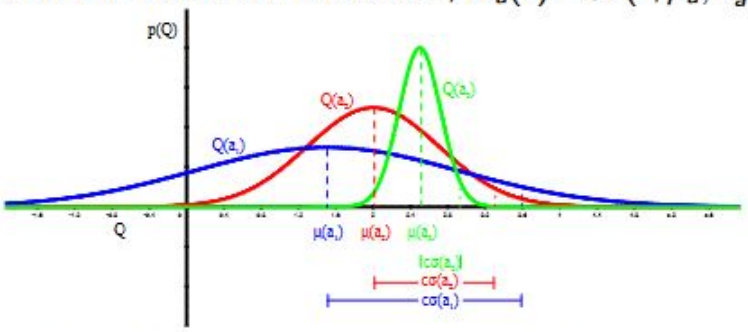
    - where 1,...,k, refer to each of the arms
- Bayesian methods compute the posterior distribution of rewards over w.

$$p[w|R_1, \vdots, R_t]$$

- And then uses this posterior to guide exploration:
    - Upper confidence bounds (Bayesian UCB)
    - Probability matching (Thompson Sampling)
- Gives better performance, if the prior knowledge is accurate (about the distribution)

For Bayesian UCB, example -

- Assume reward distribution is Gaussian, $\mathcal{R}_a(r) = \mathcal{N}(r; \mu_a, \sigma_a^2)$



- Compute Gaussian posterior over μ_a and σ_a^2 (by Bayes law)

$$p\left[\mu_a, \sigma_a^2 \mid h_t\right] \propto p\left[\mu_a, \sigma_a^2\right] \prod_{t \mid a_t = a} \mathcal{N}(r_t; \mu_a, \sigma_a^2)$$

- Pick action that maximises standard deviation of $Q(a)$

$$a_t = \text{argmax}\ \mu_a + c\sigma_a/\sqrt{N(a)}$$

Probability Matching

- We pick actions proportionally to the probability with which they're the best
    - Eg: If i have two actions, and with my knowledge so far - action 1 has 70% chance of being the best and action 2 has 30%. Then I pick them by those proportions.
- Probability matching selects action a according to probability that a is the optimal action

$$\pi(a|h_t) = P[Q(a) > Q(a\prime), \forall a\prime \neq a|h_t]$$

    - h_t is the history sampled so far.
- Probability matching is optimistic in the face of uncertainity by default
    - Uncertain actions with high potential get picked often
- And this is put into effect by **Thompson Sampling**.

Thompson Sampling

- Thompson sampling implements probability matching

$$\begin{aligned}\pi(a|h_t) &= P[Q(a) > Q(a\prime), \forall a\prime \neq a|h_t] \\ &= E_{R|h_t}[1(a = argmax_{a \in A} Q(a))]\end{aligned}$$

- So, first we compute posterior distribution using Bayes law $p[R|h_t]$
- Sample a reward distribution from posterior that we just made.
- Compute action-value function $Q(a) = E[R_a]$
- Select the action that maximisizes value on sample, $a_t = argmax_{a \epsilon A} Q(a)$
- This algorithm, somehow achieves Lai and Robbins lower bound even though it predates.

Information State Search

Value of Information

- Exploration is useful because it gains information.
- Essentially, every step we're exploring we're losing out on rewards from our known optimal. So that raises the question.
- Can we quantify the value of the information we gain, over blindly exploiting what we know ?
  - How much reward a decision maker is ready to sacrifice in order to get the information, prior to making decisions
  - We can write it as : (Long term reward after that information - immediate reward)
- We can immediately observe information gain is variable, it is higher in uncertain situations over more certain ones
- Thus it makes sense to explore uncertain situations more, but we can't just do it willy-nilly.
- We need to manage the tradeoffs optimally, and for that we need to know the value of information.

Information State Space

- And to that end, we do so by converting the bandit problem into a sequential decision one - an MDP.
- At each step, we'll have the information state $\bar{s}$ summarize the information accumulated so far.
- Each action a causes a transition to a new information state $\bar{s'}$ (by adding info) with probability $P^{\bar{a}}_{\bar{s},\bar{s'}}$
- This defines a new MDP $\bar{M}$ in the augumented information state space

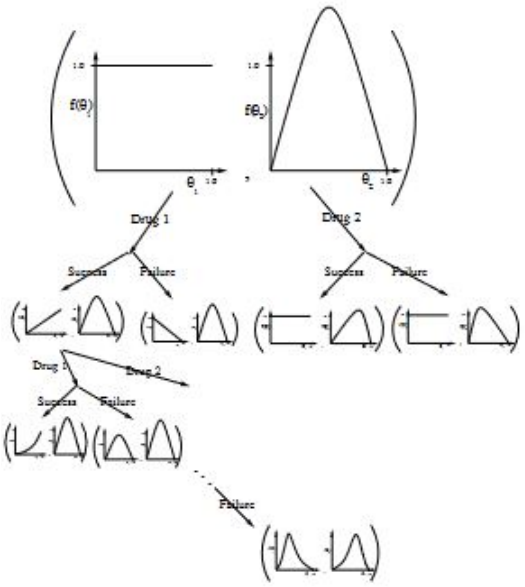
$$\bar{M} = \langle \bar{S}, A, \bar{P}, R, \gamma \rangle$$

Eg: Bernoulli Bandits

- Consider a Bernoulli bandit, such that $R^a = B(\mu_a)$
- e.g. Win or lose a game with probability μ_a
- Want to find which arm has the highest μ_a
- The information state is $\bar{s} = \langle \alpha, \beta \rangle$
  - α_a counts the pulls of arm a where reward was 0
  - β_a counts the pulls of arm a where reward was 1

Solving information State Space Bandits

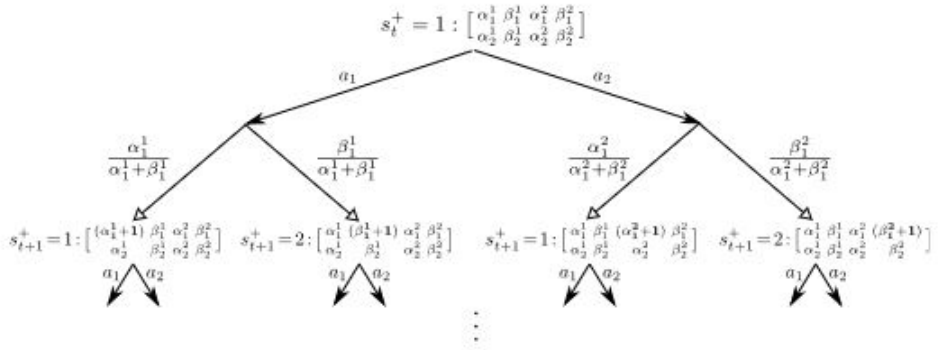
- We now have an infinite MDP over information states
- This MDP can be solved by reinforcement learning
- Model-free reinforcement learning
  - e.g. Q-learning (Duff, 1994)
- Bayesian model-based reinforcement learning
  - e.g. Gittins indices (Gittins, 1979)
    - Gittins indices is a dynamic programming solution to the lookahead tree like MDP discussed



- If we're charecterizing the information we've obtained by a posterior distribution - that's called Bayes-adaptive RL.
  - Eg: Gittings indices
- Finds Bayes-optimal exploration/exploitation trade-off w.r.t prior distribution.

Bayes-Adaptive Bernoulli Bandits

- Start with Beta(α_a, β_a) prior over reward function R^a
- Each time a is selected, update posterior for R^a
  - $Beta(\alpha_a + 1, \beta_a)$ if r = 0
  - $Beta(\alpha_a, \beta_a + 1)$ if r = 1
- This defines transition function $\tilde{P}$ for the Bayes-adaptive MDP
- Information state $\langle \alpha, \beta \rangle$ corresponds to reward model Beta(α,β)
- Each state transition corresponds to a Bayesian model update



Contextual Bandits

- A contextual bandit is a tuple ⟨A,S,R⟩
- A is a known set of actions (or “arms”)
- $S = P[s]$ is an unknown distribution over states (or “contexts”)
  - Contexts in this case refer to the fact that is more information on user here that influences the distribution
- $R_s^a(r) = P[r|s,a]$ is an unknown probability distribution over rewards
- At each step t
  - Environment generates state $s_t \sim S$
  - Agent selects action $a_t \in A$
  - Environment generates reward $r_t \sim R_{s_t}^{a_t}$
- Goal is to maximise cumulative reward $\sum_{\tau=1}^{t} r_T$
- We can see Contextual bandits can be solved via the UCB algorithm factoring in a function approximator.

Exploration/Exploitation to MDPs

- The principles we learnt above can also be applied to MDPs

UCB for MDPs

- The UCB approach can be generalised to full MDPs
- To give a model-free RL algorithm
- So now when we have uncertainity in some Q-value, we add on some bonus(U(a)) to encourage exploration of actions tried the least.
- So we pick actions like:

$$A_t = argmax_{a \epsilon A} Q(S_t, a) + U(S_t, a)$$

- This has been applied in:
  - Kaelbling's interval estimation (table lookup)
  - LSTD with confidence ellipsoid (linear function approximation)
- However this approach has issues, as it doesnt account for the policies improving and error during policy evaluation (and accounting for those is hard to do)

Optimistic Initialisation: Model-Based RL - Rmax algorithm

- One approach to exploration/exploitation in model-based RL is this:
  - Construct a model of the MDP

    - For unknown/poorly estimated states, replace reward function with r_{max}
    - i.e. Becomes optimistic in face of uncertainity - and ends up exploring all such states
    - Solve optimistic MDP using any planning algo
- This approach is also called Rmax algorithm

Info state space in MDPs

- MDPs can be augumented to included information state.
- i.e. we saved augumented state as $\langle s, \bar{s} \rangle$
    - where s is original state within MDP
    - and $\bar{s}$ is statistic of the history (accumulated info)
- Each action a causes a transition
    - to a new state s′ with probability $P^a_{s,s\prime}$
    - to a new information state $\bar{s}\prime$
- Defines MDP $\bar{M}$ in augmented information state space
    - $$\bar{M} = \langle \bar{S}, A, \bar{P}, R, \gamma \rangle$$

Problems of the week

Problem 1: Taken from Sutton and Barto

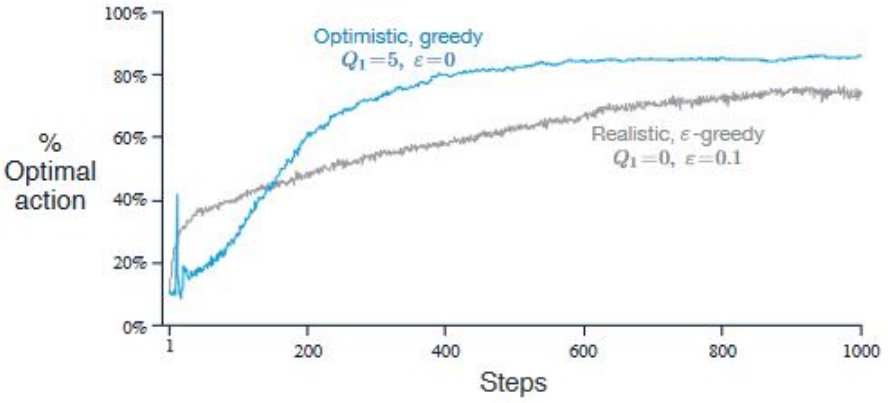


Figure 2.3: The effect of optimistic initial action-value estimates on the 10-armed testbed. Both methods used a constant step-size parameter, $\alpha = 0.1$.

Exercise 2.6: Mysterious Spikes The results shown in Figure 2.3 should be quite reliable because they are averages over 2000 individual, randomly chosen 10-armed bandit tasks. Why, then, are there oscillations and spikes in the early part of the curve for the optimistic method? In other words, what might make this method perform particularly better or worse, on average, on particular early steps?

Ans: The oscillations and spikes are caused by the optimistic exploration of the unknown states, given they're set to high inital values so it randomly picks actions. Setting higher initial values would make the initial oscillation worse as it'll take longer for the Q-values to update back to reflect their actual values.

Problem 2: Taken from Sutton and Barto

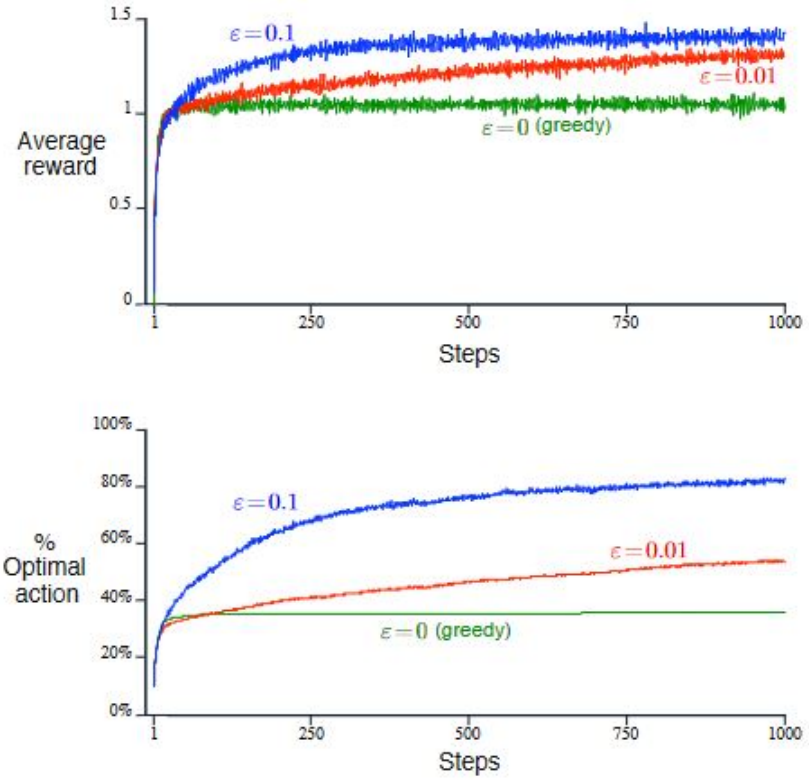


Figure 2.2: Average performance of ε-greedy action-value methods on the 10-armed testbed. These data are averages over 2000 runs with different bandit problems. All methods used sample averages as their action-value estimates.

Q: Exercise 2.3 In the comparison shown in Figure 2.2, which method will perform best in the long run in terms of cumulative reward and probability of selecting the best action? How much better will it be? Express your answer quantitatively.

Ans: Looking at 1000 step mark, we can clearly see the method with $\epsilon = 0.01$ is clearly better than the rest, with the graph steadying at that point.
With that in mind, we can clearly state that it will be atleast 50% better than fully greedy method seeing as steadied into a straight line. Both in terms of reward and choosing the optimal action.