REINFORCEMENT LEARNING I

Master's Deep Learning

Sergey Nikolenko

Harbour Space University, Barcelona, Spain
November 22, 2017

MULTIARMED BANDITS

- So far we've either had a set of "correct answers" (supervised learning) or simply nothing (unsupervised learning).
- But is it really how learning works in real life?
- How does a baby learn?

- Hence, *reinforcement learning*.
- An agent interacts with the environment.
- On every step the agent can be in state $s \in S$ and choose an action $a \in A$.
- The environment tells the agent its reward r and the next state $s' \in S$.

EXPLOITATION VS. EXPLORATION

- Exploitation vs. exploration: first learn, then apply.
- But when do we switch?
- Always a problem in reinforcement learning.

EXAMPLE

- Example: tic-tac-toe.
- How does an algorithm learn to play and *win* in tic-tac-toe?
- Example: genetic algorithm. Very slow, does not account for information.

EXAMPLE

- States are board positions.
- Value function $V(s)$ for every state.
- Reinforcement only at the end: the *credit assignment* problem.

- One version — propagate the reward back: if we got from s to s', we update
$$V(s) := V(s) + \alpha \left[V(s') - V(s) \right].$$
- This is called TD-learning (temporal difference learning), works very well in practice; we'll get to it.

- If $|S| = 1$, the agent has a fixed set of actions A and the environment has no memory.
- The multiarmed bandit model.
- No credit assignment, only exploration vs. exploitation.

- Always choose the best option, where *best* is defined with average reward so far:

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

- What's wrong with this algorithm?

- Always choose the best option, where *best* is defined with average reward so far:

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

- What's wrong with this algorithm?
- Easy to miss the optimum if we're unlucky with the initial sample.
- Useful heuristic — *optimism under uncertainty*.
- You need evidence to *reject*, not to accept.

RANDOMIZED STRATEGIES

- ϵ-*greedy strategy*: choose the best (as above) action with probability $1 - \epsilon$ and random action with probability ϵ.
- Start with large ϵ, then gradually decrease.
- Boltzmann exploration:

$$\pi_t(a) = \frac{e^{Q_t(a)/T}}{\sum_{a'} e^{Q_t(a')/T}},$$

  where ER is the expected reward, T is the *temperature*.
- Temperature usually decreases with time.

LINEAR REWARD-INACTION ALGORITHM

- For the case of binary payoffs (0-1).
- The *linear reward-inaction algorithm* adds linear reward to probability of a_i if it is successful:

$$p_i := p_i + \alpha(1 - p_i),$$

$$p_j := p_j - \alpha p_j, \quad j \neq i,$$

and nothing changes if unsuccessful.

LINEAR REWARD-INACTION ALGORITHM

- The algorithm converges with probability 1 to a vector with one 1 and the rest 0.
- Does not always converge to the optimal strategy; but by decreasing α we decrease the probability of error.
- *Linear reward-penalty*: same thing, but unsuccessful actions get punished (i.e., all the rest get a reward).

INTERVAL ESTIMATES

- One way to apply the optimism under uncertainty heuristic.
- Store the statistics n and w for every action, compute confidence interval with confidence $1-\alpha$, use the upper bound.
- Example: Bernoulli trials (coin tossing). With probability $.95$ the average lies in the interval

$$\left(\bar{x} - 1.96\frac{s}{\sqrt{n}}, \bar{x} + 1.96\frac{s}{\sqrt{n}}\right),$$

  where 1.96 is taken from Student's t distribution, n is the number of trials, $s = \sqrt{\frac{\sum(x-\bar{x})^2}{n-1}}$.
- A great method if the assumptions hold (which is often unclear).

- How do we recompute $Q_t(a) = \frac{r_1 + \ldots + r_{k_a}}{k_a}$ when new information arrives?
- Easy:

$$Q_{k+1} = \frac{1}{k+1} \sum_{i=1}^{k+1} r_i = \frac{1}{k+1} \left[r_{k+1} + \sum_{i=1}^{k} r_i \right] =$$

$$= \frac{1}{k+1} \left(r_{k+1} + kQ_k \right) = Q_k + \frac{1}{k+1} \left(r_{k+1} - Q_k \right).$$

- This is a special case of a general rule:

$$\text{NewEstimate} := \text{OldEstimate} + \text{StepSize}\left[\text{Target} - \text{OldEstimate}\right].$$

- For the average, the step size is not constant: $\alpha_k(a) = \frac{1}{k_a}$.
- Changing the sequence of steps, we can achieve other effects.

- What if the payoffs change with time?
- We should value recent information highly and outdated information low.
- Example: for an update rule

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

with constant α the weights decay exponentially:

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

$$= \alpha r_k + (1-\alpha)\alpha r_{k-1} + (1-\alpha)^2 Q_{k-2} = (1-\alpha)^k Q_0 + \sum_{i=1}^{k} \alpha (1-\alpha)^{k-i} r_i .$$

- This update rule does not necessarily converge, which is good: we want to follow new averages.
- General result – an update rule converges if the sequence of weights satisfies

$$\sum_{k=1}^{\infty} \alpha_k(a) = \infty \quad \text{and} \quad \sum_{k=1}^{\infty} \alpha_k^2(a) < \infty.$$

- E.g., for $\alpha_k(a) = \frac{1}{k_a}$ it does.

OPTIMISM AGAIN

- We can simplify the search if we begin with optimistic initial values.
- Start with large $Q_0(a)$, so that any real value is “disappointing”.
- But not too large — we need Q_0 to average out with the real r_i.

- The intuition for *reinforcement comparison* is to look for “large” payoffs; what is “large”?
- Let’s compare with average over all arms.
- These methods usually do not have action values Q_k, only preferences $p_t(a)$; probabilities can be obtained, e.g., with softmax:

$$\pi_t(a) = \frac{e^{p_t(a)}}{\sum_{a'} e^{p_t(a')}}.$$

REINFORCEMENT COMPARISON

- And on every step we update both preference and average:

$$\bar{r}_{t+1} = \bar{r}_t + \alpha \left(r_t - \bar{r}_t \right),$$
$$p_{t+1}(a) = p_t(a_t) + \beta \left(r_t - \bar{r}_t \right).$$

- Pursuit methods store both expectation estimates and action preferences, and preferences "follow" averages.
- E.g., $\pi_t(a)$ is the probability to choose a at time t; after step t we look for a greedy strategy

$$a^*_{t+1} = \arg\max_a Q_{t+1}(a)$$

and change π towards the greedy strategy:

$$\pi_{t+1}(a^*_{t+1}) = \pi_t(a^*_{t+1}) + \beta \left[1 - \pi_t(a^*_{t+1})\right],$$
$$\pi_{t+1}(a) = \pi_t(a) + \beta \left[0 - \pi_t(a)\right].$$

- Assume finite horizon of h steps.
- We use the Bayesian approach to find the optimal strategy.
- Begin with random parameters $\{p_i\}$, e.g., uniform; compute the mapping from *belief states* (after several rounds) to actions.
- A state is expressed as $\mathcal{S} = \{n_1, w_1, \ldots, n_k, w_k\}$, where each bandit i has been run n_i times with w_i positive (binary) results.

- $V^*(\mathcal{S})$ — expected remaining payoff.
- Recursion: if $\sum_{i=1}^{k} n_i = h$, $V^*(\mathcal{S}) = 0$ since there's no time left.
- If we know V^* for all states when t time slots are left, we can recompute for $t+1$:

$$V^*(n_1, w_1, \dots, n_k, w_k) = \\ = \max_i \left(\rho_i(1 + V^*(\dots, n_i + 1, w_i + 1, \dots)) + \\ (1 - \rho_i)V^*(\dots, n_i + 1, w_i, \dots)\right),$$

where ρ_i is the posterior probability of action i to be rewarded (if p_i had uniform priors then Laplace rule applies: $\rho_i = \frac{w_i+1}{n_i+2}$).

GENERAL PROBLEM SETTING

- We now go back to the multi-state model.
- Agent and environment, the agent is rewarded by the environment on every step: $r_t, r_{t+1}, \ldots$
- Two different problems:
    - learn the environment;
    - find out the optimal way to operate in it.
- We begin by solving them separately and then unite.

- But what is “good” in the long run? How do we evaluate an algorithm?
- *Episodic task*: fixed finite horizon (a game of chess), we can simply maximize the reward per episode (until terminal state).
- But what about continuous problems?

- *Finite horizon*: agent only looks at the next h steps: $E\left[\sum_{t=0}^{h} r_t\right]$.
- *Infinite horizon*: we'd like to account for the entire future but it's better to have a dollar today than tomorrow. How do we account for it?

- *Finite horizon*: agent only looks at the next h steps: $E\left[\sum_{t=0}^{h} r_t\right]$.
- *Infinite horizon*: we'd like to account for the entire future but it's better to have a dollar today than tomorrow. How do we account for it?
- $$E\left[\sum_{t=0}^{\infty} \gamma^t r_t\right],$$

  where γ is a constant *discount factor*.

DIFFERENT MODELS

- *Finite horizon*: agent only looks at the next h steps: $E\left[\sum_{t=0}^{h} r_t\right]$.
- *Infinite horizon*: we'd like to account for the entire future but it's better to have a dollar today than tomorrow. How do we account for it?
- $$E\left[\sum_{t=0}^{\infty} \gamma^t r_t\right],$$

  where γ is a constant *discount factor*.
- *Average reward model*:

$$\lim_{h \to \infty} E\left[\frac{1}{h}\sum_{t=0}^{h} r_t\right].$$

- The infinite horizon model is the most common.
- Besides, it can be generalized to episodic problems: let $\gamma = 1$ and add one extra terminal state with reward 0 which is closed to itself.
- So we will always assume

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}.$$

- Estimating quality of an algorithm:
  - convergence itself;
  - rate of convergence (to a fixed share of optimality of after fixed time);
  - regret (the best but also the hardest measure).

- *Markov decision process*:
  - set of states S;
  - set of actions A;
  - reward function $R : S \times A \to \mathbb{R}$; expected reward when passing from s to s' after action a is $R^a_{ss'}$;
  - transition functions between states $p^a_{ss'} : S \times A \to \Pi(S)$, where $\Pi(S)$ is the set of probability distributions over S; the probability to get from s to s' after a is $P^a_{ss'}$.
- The model is *Markov* if transitions do not depend on the history of transitions.

THE MARKOV PROPERTY

- The Markov property:

$$p(s_{t+1} = s', r_{t+1} = r \mid s_t, a_t, \dots, s_0, a_0) = p(s_{t+1} = s', r_{t+1} = r \mid s_t, a_t).$$

- Does not seem to be relevant: don't we almost always need to account for history?
- Well, yes, but we can still consider Markov processes. Why?

THE MARKOV PROPERTY

- The Markov property:

$$p(s_{t+1} = s', r_{t+1} = r \mid s_t, a_t, \ldots, s_0, a_0) = p(s_{t+1} = s', r_{t+1} = r \mid s_t, a_t).$$

- Does not seem to be relevant: don't we almost always need to account for history?
- Well, yes, but we can still consider Markov processes. Why?
- Because we will simply define states S so that each state holds all relevant information.
- What is the state in chess? in poker?

VALUE FUNCTION AND BELLMAN EQUATIONS

- Main difference compared to bandits – the difference between *reward function* (immediate reinforcement) and *value function* (total expected reinforcement if we start from this state).
- The essence of many reinforcement learning algorithms is to estimate and optimize the value function.
- For Markov processes we can formally define

$$V^{\pi}(s) = \mathrm{E}_{\pi}\left[R_t \mid s_t = s\right] = \mathrm{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right].$$

- Or in even more detail – general reinforcement which is expected if we start from state s and action a:

$$Q^{\pi}(s,a) = \mathrm{E}_{\pi}\left[R_t \mid s_t = s, a_t = a\right] =$$

$$= \mathrm{E}_{\pi}\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right].$$

- Functions V and Q are exactly what we need to estimate; if we knew them we could simply choose a that maximizes $Q(s,a)$.

- For a known strategy π, V satisfies Bellman equations:

$$\begin{aligned} V^\pi(s) &= \mathrm{E}_\pi\left[R_t \mid s_t = s\right] = \mathrm{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right] \\ &= \mathrm{E}_\pi\left[r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_t = s\right] \\ &= \sum_a \pi(s,a) \sum_{s'} P^a_{ss'}\left(R^a_{ss'} + \gamma \mathrm{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_{t+1} = s'\right]\right) \\ &= \sum_a \pi(s,a) \sum_{s'} P^a_{ss'}\left(R^a_{ss'} + \gamma V^\pi(s')\right). \end{aligned}$$

TWO MAIN PROBLEMS

- If we know the model, the problem is to find the optimal strategy.
- In a real situation, we don't know the model, and we don't know the strategy.
- We begin with the first (easier) problem.

OPTIMAL STATE VALUES

- *Optimal state value* is the expected total reward for an agent that starts in this state and follows the optimal strategy:

$$V^*(s) = \max_\pi \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right].$$

- Also satisfies similar Bellman equations:

$$\begin{aligned}
V^{\pi}(s) &= \max_a Q^{\pi^*}(s,a) = \max_a \mathrm{E}_{\pi^*}\left[R_t \mid s_t = s, a_t = a\right] \\
&= \max_a \mathrm{E}_{\pi^*}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right] \\
&= \max_a \mathrm{E}_{\pi^*}\left[r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_t = s, a_t = a\right] \\
&= \max_a \mathrm{E}\left[r_{t+1} + \gamma V^*(s_{t+1}) \mid s_t = s, a_t = a\right] \\
&= \max_a \sum_{s'} P^a_{ss'}\left(R^a_{ss'} + \gamma V^*(s')\right).
\end{aligned}$$

OPTIMAL STATE VALUES

- I.e., $V^*(s)$ can be defined as a solution of the system

$$V^*(s) = \max_a \sum_{s' \in S} P_{ss'}^a \left(R_{ss'}^a + \gamma V^*(s') \right),$$

and then choose the optimal strategy as

$$\pi^*(s) = \arg\max{}_a \sum_{s' \in S} P_{ss'}^a \left(R_{ss'}^a + \gamma V^*(s') \right).$$

- How can we solve these equations?

- To compute value functions for a given strategy, we can simply iteratively recompute Bellman equations:

$$V(s) := \sum_a \pi(s,a) \sum_{s' \in S} P^a_{ss'} \left(R^a_{ss'} + \gamma V(s') \right),$$

until convergence.

- Accordingly, for the optimal V^* we solve equations with $\max$:

$$V^*(s) := \max_a \sum_{s' \in S} P^a_{ss'} \left(R^a_{ss'} + \gamma V^*(s') \right).$$

- The same for Q — repeat until convergence:

$$Q(s,a) := \sum_{s' \in S} P^a_{ss'} \left(R^a_{ss'} + \gamma \sum_{a'} \pi(s,a') Q(s,a') \right).$$

- And then simply set $V(s) := \max_a Q(s,a)$.
- It is also easy to find optimal $Q^*(s,a)$:

$$Q^*(s,a) := \sum_{s' \in S} P^a_{ss'} \left(R^a_{ss'} + \gamma \max_{a'} Q^*(s,a') \right).$$

- The recomputation in the previous algorithm uses information from all preceding states:

$$Q(s,a) := \sum_{s' \in S} P^a_{ss'} \left(R^a_{ss'} + \gamma \max_{a'} Q(s,a') \right).$$

- We can make a stochastic version: for a current transition we update

$$Q(s,a) := Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a)).$$

- It works (theorem) if every pair (s,a) occurs an infinite number of times, s' is chosen from distribution $P^a_{ss'}$, and r is sampled with mean $R(s,a)$ and bounded variance.

WHAT'S THE PROBLEM?

- We have seen a simple algorithm that works well and converges rapidly.
- What's the problem? Have we solved reinforcement learning (in a known model)?

WHAT'S THE PROBLEM?

- We have seen a simple algorithm that works well and converges rapidly.
- What's the problem? Have we solved reinforcement learning (in a known model)?
- Problem: huge number of states, even larger number of state-action pairs.
- So we will have to approximate these equations and try to limit search space.
- But first let's talk some more about general approaches.

- How can we improve a strategy? For a strategy π, is it beneficial to change the action for a given state s?
- If we choose a new action a at state s and then follow π, we get

$$Q^\pi(s,a) = \sum_{s'} P^a_{ss'} \left(R^a_{ss'} + \gamma V^\pi(s') \right).$$

- Policy improvement theorem: if, for some strategies π and π', for all s

$$Q^\pi(s, \pi'(s)) \geq V^\pi(s),$$

  then π' is better: for all s $V^{\pi'}(s) \geq V^\pi(s)$.
- Proof: left as an exercise for rewriting Bellman equations.

- This gives a natural greedy algorithm (policy iteration):
  - compute V^{π};
  - update $\pi'(s) = \arg\max_a Q^{\pi}(s, a)$;
  - repeat.
- We will improve the strategy until it stops improving.
- Why does it stop, by the way?

- Two-step process: $V \to \pi \to V \to \ldots.$
- It may be hard to estimate V. Let us stochastically stop after one step; *value iteration*:

$$V_{k+1}(s) = \max_a \mathrm{E}\left[r_{t+1} + \gamma V_k(s_{t+1}) \mid s_t = s, a_t = a\right] = \\ = \max_a \sum_{s'} P^a_{ss'} \left(R^a_{ss'} + \gamma V_k(s')\right).$$

- It's the same as solving Bellman equations with the $\max$.

- We can look for the optimal strategy with a simple iterative algorithm.
- `PolicyIteration`:
    - Initialize π.
    - Repeat:
        - compute state values for strategy π by solving a system of linear equations

$$V_\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} P_{ss'}^{\pi(s)} V_\pi(s'),$$

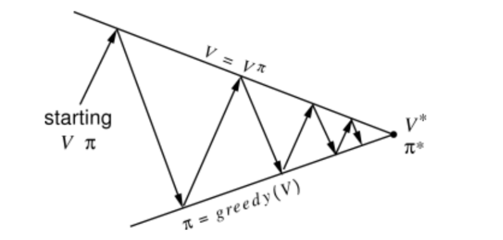
        - improve the strategy on every state:

$$\pi'(s) := \arg\max_a \left(R(s, a) + \gamma \sum_{s' \in S} P_{ss'}^{a} V_\pi(s') \right);$$

    - until $\pi \neq \pi'$.

- Why does it converge?

- Why does it converge?
- On every step it strictly improves the objective function, and there is a finite number ($|A|^{|S|}$) of strategies.
- There are other versions; the common theme is to recompute π and V until convergence.



THANK YOU!

Thank you for your attention!