DD2380 Artificial Intelligence
Machine Learning 2: Reinforcement learning

Alexandre Proutiere

September 6, 2017

KTH (The Royal Institute of Technology)

Outline of today's lecture

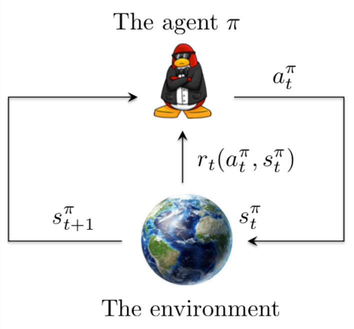
0. Introduction: Supervised vs. Unsupervised learning

1. Supervised Learning

2. **Reinforcement Learning**
    - A. **Playing against an unknown environment**
    - B. Bandit optimisation
    - C. RL in Markov Decision Provesses

Reinforcement learning

Learning optimal sequential behaviour / control from interacting with the environment (data generated as we go in an adaptive manner)



Unknown state dynamics:
$s_{t+1}^\pi = F_t(s_t^\pi, a_t^\pi)$

Reinforcement learning: Applications

- Making a robot walk
- Portfolio optimisation
- Playing games better than humans
- Helicopter stunt manoeuvres
- Optimal communication protocols in radio networks
- Display ads
- Search engines
- ...

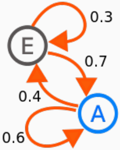
IID vs. Markovian dynamics

- IID (Independent and Identically Distributed) environment: Bandit optimisation

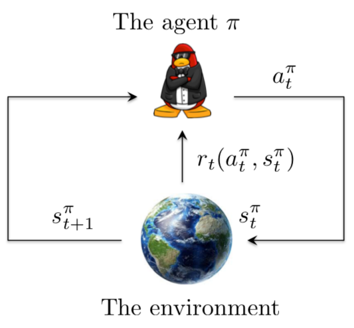
  Each *round*, select one action and observe a random reward whose distribution depends on the action only

- Markovian environment: Learning in Markov Decision Processes (MDPs)

  The new state is randomly generated depending on the former state and the selected action



Bandit optimisation

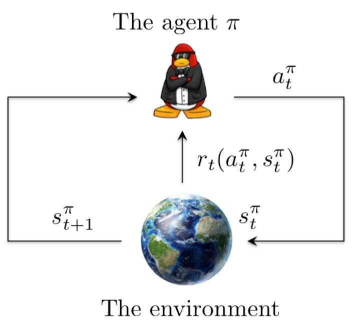


State dynamics:

$s_{t+1}^\pi = F_t(s_t^\pi, a_t^\pi)$

- Interact with an i.i.d.
- The reward is independent of the state and is the only feedback:
  - i.i.d. environment: $r_t(a, s) = r_t(a)$ random variable with mean θ_a
  - adversarial environment: $r_t(a, s) = r_t(a)$ is arbitrary!

Markov Decision Process



State dynamics:
$s_{t+1}^\pi = F_t(s_t^\pi, a_t^\pi)$

- History at t: $h_t^\pi = (s_1^\pi, a_1^\pi, \ldots, s_{t-1}^\pi, a_{t-1}^\pi, s_t^\pi)$
- Markovian environment: $\mathbb{P}[s_{t+1}^\pi = s' | h_t^\pi, s_t^\pi = s, a_t^\pi = a] = p(s'|s,a)$
- Stationary deterministic rewards (for simplicity): $r_t(a,s) = r(a,s)$

What is to be learnt and optimised?

- Bandit optimisation: the average rewards of actions are unknown
  Information available at time t under π:

$$a_1^\pi, r_1(a_1^\pi), \dots, a_{t-1}^\pi, r_{t-1}(a_{t-1}^\pi)$$

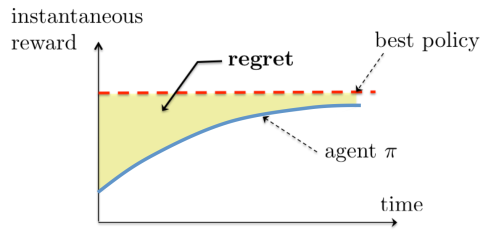
- MDP: The state dynamics $p(\cdot|s,a)$ and the reward function $r(a,s)$ are unknown
  Information available at time t under π:

$$s_1^\pi, a_1^\pi, r_1(a_1^\pi, s_1^\pi), \dots, s_{t-1}^\pi, a_{t-1}^\pi, r_{t-1}(a_{t-1}^\pi, s_{t-1}^\pi), s_t^\pi$$

- Objective: maximise the cumulative reward

$$\sum_{t=1}^{T} \mathbb{E}[r_t(a_t^\pi, s_t^\pi)] \quad \text{or} \quad \sum_{t=1}^{\infty} \lambda^t \mathbb{E}[r_t(a_t^\pi, s_t^\pi)]$$

Regret



- Difference between the cumulative reward of an "Oracle" policy and that of agent π
- Regret quantifies the price to pay for learning!
- Exploration vs. exploitation trade-off: we need to probe all actions to play the best later ...

Outline of today's lecture

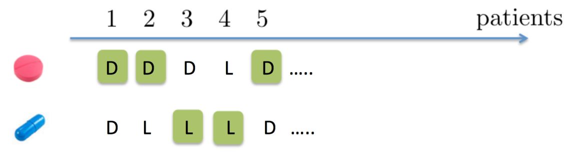
0. Introduction: Supervised vs. Unsupervised learning

1. Supervised Learning

2. **Reinforcement Learning**
   A. Playing against an unknown environment
   B. **Bandit optimisation**
   C. RL in Markov Decision Provesses

Bandit Optimisation

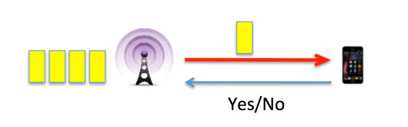
First application: Clinical trial, Thompson 1933



- A set of possible actions at each step
- Unknown sequence of rewards for each action
- Bandit feedback: only rewards of chosen actions are observed
- Goal: maximise the cumulative reward (up to step T)

Applications

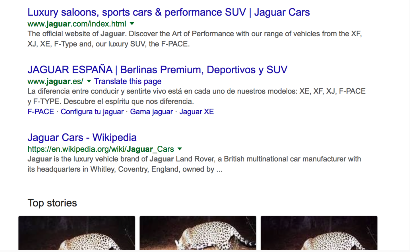
Rate adaptation in 802.11 wireless systems



- The AP sequentially sends packets to the receiver and has K available encoding rates $r_1 < r_2 < \ldots < r_K$ + MIMO modes
- The unknown probability a packet sent at rate r_k is received is θ_k
- Goal: design a rate selection scheme that learns the θ_k's and quickly converges to rate $r_{k^\star}$ maximising $\mu_k = r_k \theta_k$ over k

Applications

Search engines



- The engine should list relevant webpages depending on the request 'jaguar'
- The CTRs (Click-Through-Rate) are unknown
- Goal: design a list selection scheme that learns the list maximising its global CTRs

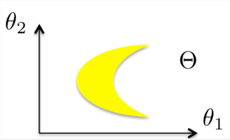
Stochastic bandit taxonomy

Unstructured problems: average rewards are not related

$\theta = (\theta_1, \dots, \theta_K) \in \Theta = \prod_k [a_k, b_k]$

Structured problems: the decision maker knows that average rewards are related. She knows Θ. The rewards observed for a given arm provides side information about the other arms.

$\theta = (\theta_1, \dots, \theta_K) \in \Theta$ not an hyper-rectangle



Stochastic bandit taxonomy

K/T

(Decision set/Time horizon)

0 $\longrightarrow$ ∞

Lai-Robbins 1985

Bandits with structure:
Combinatorial bandits
Linear bandits
Convex bandits
Unimodal bandits
Agrawal 1995
Kleinberg-Slivkins 2005
Bubeck et al. 2011 ...

Infinite Bandits:
Berry et al. 1997
Wang et al. 2008
Bonald-P. 2013

Lai
Agrawal-Teneketzis-Anantharam 1989
Kumar 1985
Borkar-Varaiya 1979
Optimal control of Markov Chains

Regret scalings

- Discrete unstructured bandits $A = \{1, \dots, K\}$
  Regret $O(K \log(T))$ (stochastic), $O(\sqrt{KT})$ (adversarial)
- Infinite Bandits $A = \mathbb{N}$, Bayesian setting
  Regret $O(\sqrt{T})$
- Continuous Bandits $A \subset \mathbb{R}^d$
  Structures: $a \to \theta_a$ is convex, Lipschitz, linear, unimodal (quasi-convex) etc.
  Regret: $O(poly(d)\sqrt{T})$

Unstructured stochastic bandits – Robbins 1952

- Finite set of actions $A = \{1, \dots, K\}$
- (Unknown) rewards of action $a \in A$: $(X_t(a), t \geq 0)$ i.i.d. Bernoulli with $\mathbb{E}[X_t(a)] = \theta_a$
- Optimal action $a^\star \in \arg\max_a \theta_a$
- Online policy π: select action a_t^π at time t depending on $a_1^\pi, X_1(a_1^\pi), \dots, a_{t-1}^\pi, X_{t-1}(a_{t-1}^\pi)$
- Regret up to time T: $R^\pi(T) = T\theta_{a^\star} - \sum_{t=1}^{T} \theta_{a_t^\pi}$

Regret lower bound

Uniformly good algorithms: An algorithm π is uniformly good if for all $\theta \in \Theta$, for any sub-optimal arm a, the number of times $n_a(t)$ arm a is selected up to round t satisfies: $\mathbb{E}[n_a(t)] = o(t^\alpha)$ for all $\alpha > 0$.

Fundamental performance limits: (Lai-Robbins1985)
For any uniformly good algorithm π:

$$\liminf_T \frac{R^\pi(T)}{\log(T)} \geq \sum_{a \neq a^\star} \frac{\theta_{a^\star} - \theta_a}{KL(\theta_a, \theta_{a^\star})}$$

where $KL(a,b) = a\log(\frac{a}{b}) + (1-a)\log(\frac{1-a}{1-b})$ (KL divergence)

. Regret linear in K, and scaling as $1/(\theta_{a^\star} - \theta_a)$

Algorithms

Empirical reward of arm a: $\hat{\theta}_a(t) = \frac{1}{n_a(t)} \sum_{n=1}^{t} X_n(a) 1_{a(n)=a}$

- ϵ-**greedy.** In each round t:
  - with probability $1-\epsilon$, select the best empirical arm $a^\star(t) \in \arg\max_a \hat{\theta}_a(t)$
  - with probability ϵ, select an arm uniformly at random

  The algorithm has linear regret (not uniformly good)

- ϵ_t-**greedy.** In each round t:
  - with probability $1-\epsilon_t$, select the best empirical arm $a^\star(t) \in \arg\max_a \hat{\theta}_a(t)$
  - with probability ϵ_t, select an arm uniformly at random

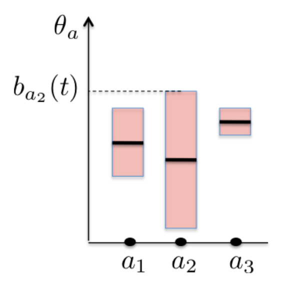
  The algorithm has logarithmic regret for Bernoulli rewards and $\epsilon_t = \min(1, \frac{K}{t\delta^2})$ where $\delta = \min_{a \neq a^\star}(\theta_{a^\star} - \theta_a)$

Algorithms

Optimism in front of Uncertainty

Upper Confidence Bound algorithm: Auer et al., 2002

$$b_a(t) = \hat{\theta}_a(t) + \sqrt{\frac{2\log(t)}{n_a(t)}}$$



$\hat{\theta}(t)$: empirical reward of a up to t

$n_a(t)$: nb of times a played up to t

In each round t, select the arm with highest index $b_a(t)$

Under UCB, the number of times $a \neq a^\star$ is selected satisifies:

$$\mathbb{E}[n_a(T)] \leq \frac{8\log(T)}{(\theta_{a^\star} - \theta_a)^2} + \frac{\pi^2}{6}$$

Algorithms

KL-UCB algorithm: Lai 1987, Garivier et Cappe 2011

$$b_a(t) = \max\{q \le 1 : n_a(t) KL(\hat{\theta}_a(t), q) \le f(t)\}$$

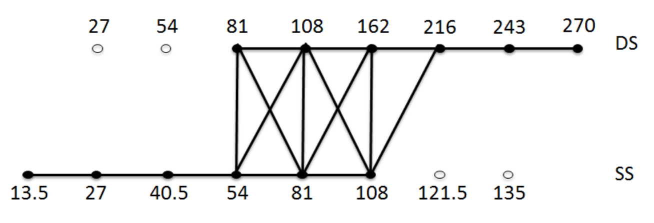
where $f(t) = \log(t) + 3\log\log(t)$ is the *confidence* level.
In each round t, select the arm with highest index $b_a(t)$

Under KL-UCB, the number of times $a \neq a^\star$ is selected satisifies: for all $\delta < \theta_{a^\star} - \theta_a$, for all T,

$$\mathbb{E}[n_a(T)] \le \frac{\log(T)}{KL(\theta_a + \delta, \theta_{a^\star})} + C\log\log(T) + \delta^{-2}$$

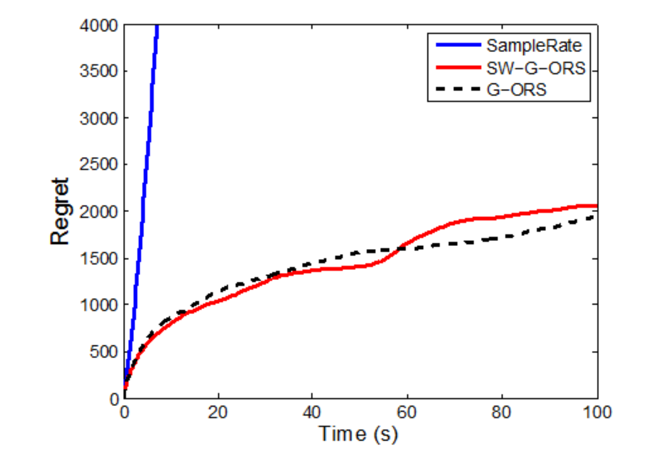
Structured stochastic bandits

- $a \mapsto \theta_a$ is (lipschitz, convex, unimodal, multimodal, ...)
- $\theta \in \Theta$, Θ encodes the known structure
- 802.11 rate adaptation example. Graphical unimodality: there is a graph (vertices = arms) such that from any vertex, there is a path to the optimal arm with increasing average reward



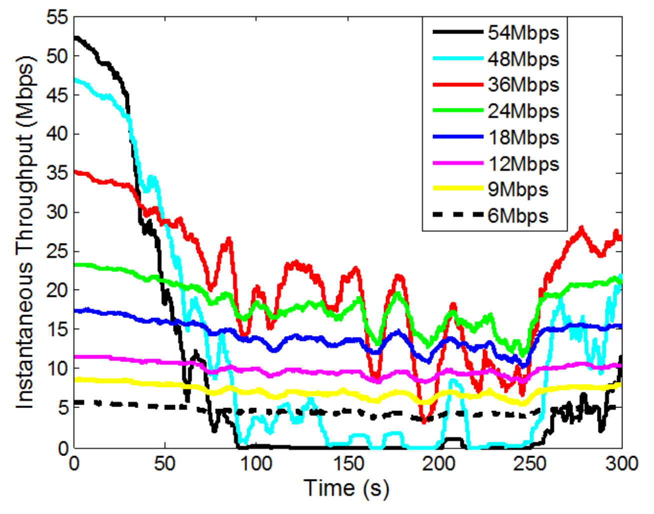
Unimodal bandits

- Optimal exploration: Only neighbors of the optimal arm generate log regret
- Optimal algorithm[1]: maximises throughput over a time window

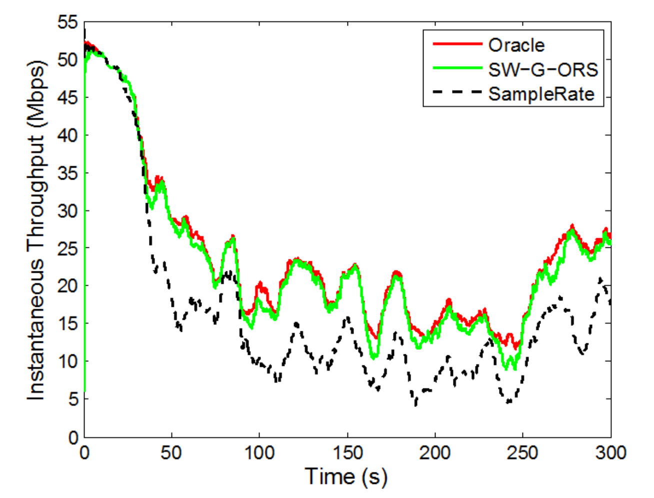


[1]Optimal Rate Sampling in 802.11, Combes at al., IEEE Infocom 2014

Non-stationary environment



SW version of optimal algorithms



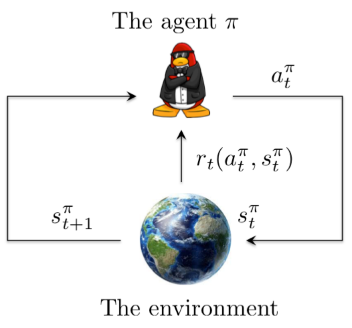
Outline of today's lecture

0. Introduction: Supervised vs. Unsupervised learning

1. Supervised Learning

2. **Reinforcement Learning**
    - A. Playing against an unknown environment
    - B. Bandit optimisation
    - C. **RL in Markov Decision Provesses**

Markov Decision Process (MDP)



State dynamics:
$s_{t+1}^\pi = F_t(s_t^\pi, a_t^\pi)$

- Markovian environment: $\mathbb{P}[s_{t+1}^\pi = s' | h_t^\pi, s_t^\pi = s, a_t^\pi = a] = p(s'|s,a)$
- Stationary deterministic rewards (for simplicity): $r_t(a,s) = r(a,s)$
- $p(\cdot|s,a)$ and $r(\cdot,\cdot)$ are unknown initially

Example

Playing pacman (Google Deepmind experiment, 2015)

State: the current displayed image
Action: right, left, down, up
Feedback: the score and its increments + state

Bellman's equation

Objective: max the average discounted reward $\sum_{t=1}^{\infty} \lambda^t \mathbb{E}[r(a_t^{\pi}, s_t^{\pi})]$
Assume the transition probabilities and the reward function are known

- Value function: maps the initial state s to the corresponding maximum reward $v(s)$
- Bellman's equation:

$$v(s) = \max_{a \in A} \left[r(a, s) + \lambda \sum_j p(j|s, a) v(j) \right]$$

- Solve Bellman's equation. The optimal policy is given by:

$$a^{\star}(s) = \arg\max_{a \in A} \left[r(a, s) + \lambda \sum_j p(j|s, a) v(j) \right]$$

Q-learning

What if the transition probabilities and the reward function are unknown?

- Q-value function: the max expected reward starting from state s and playing action a:

$$Q(s,a) = r(a,s) + \lambda \sum_j p(j|s,a) \max_{b \in A} Q(j,b)$$

  Note that: $v(s) = \max_{a \in A} Q(s,a)$

- Algorithm: update the Q-value estimate sequentially so that it converges to the true Q-value

Q-learning

1. **Initialisation:** select $Q \in \mathbb{R}^{S\times A}$ arbitrarily, and s_0
2. **Q-value iteration:** at each step t, select action a_t (each state-action pair must be selected infinitely often)
   Observe the new state s_{t+1} and the reward $r(s_t, a_t)$
   Update $Q(s_t, a_t)$:

$$\begin{aligned} Q(s_t, a_t) :=& Q(s_t, a_t) \\ &+ \alpha_t \left[r(s_t, a_t) + \lambda \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t) \right] \end{aligned}$$

It converges to Q if $\sum_t \alpha_t = \infty$ and $\sum_t \alpha_t^2 < \infty$

Q-learning: demo

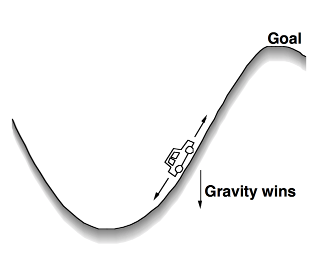
The crawling robot ...

https://www.youtube.com/watch?v=2iNrJx6IDEo

Scaling up Q-learning

Q-learning converges very slowly, especially when the state and action spaces are large ...
State-of-the-art algorithms (optimal exploration, ideas from bandit opt.): regret $O(\sqrt{SAT})$

What if the action and state are continuous variables?
Example: Mountain car demo[2]



[2]See Sutton tutorial, NIPS 2015

Q-learning with function approximation

Idea: restrict our attention to Q-value functions belonging to a family of functions $\mathcal{Q}$

Examples:

1. Linear functions: $\mathcal{Q} = \{Q_\theta, \theta \in \mathbb{R}^M\}$,

$$Q_\theta(s,a) = \sum_{i=1}^{M} \phi_i(s,a)\theta_i = \phi^\top \theta$$

   where for all i, ϕ_i is linear. The ϕ_i's are linearly independent.
2. Deep networks: $\mathcal{Q} = \{Q_\mathbf{w}, \mathbf{w} \in \mathbb{R}^M\}$, $Q_\mathbf{w}(s,a)$ given as the output of a neural network with weights $\mathbf{w}$ and inputs (s,a)

Q-learning with linear function approximation

1. **Initialisation:** select $\theta \in \mathbb{R}^M$ arbitrarily, and s_0
2. **Q-value iteration:** at each step t, select action a_t (each state-action pair must be selected infinitely often)
   Observe the new state s_{t+1} and the reward $r(s_t, a_t)$
   Update θ:

$$\begin{aligned} \theta &:= \theta + \alpha_t \Delta_t \nabla_\theta Q_\theta(s_t, a_t) \\ &= \theta + \alpha_t \Delta_t \phi(s_t, a_t) \end{aligned}$$

   where $\Delta_t = r(s_t, a_t) + \lambda \max_{a \in A} Q_\theta(s_{t+1}, a) - Q_\theta(s_t, a_t)$

For convergence results, see "An analysis of Reinforcement Learning with Function Approximation", Melo et al., ICML 2008

Q-learning with function approximation

Success stories:

- TD-Gammon (Backgammon), Tesauro 1995 (neural nets)
- Acrobatic helicopter autopilots, Ng et al. 2006
- Jeopardy, IBM Watson, 2011
- 49 atari games, pixel-level visual inputs, Google Deepmind 2015

Questions?

Alexandre Proutiere

alepro@kth.se