Introduction to Reinforcement Learning

A. LAZARIC (*SequeL Team @INRIA-Lille*)

Machine Learning Summer School – Toulouse, France

dubbing: A. GARIVIER (Institut de Mathématique de Toulouse)

SequeL – INRIA Lille

Outline

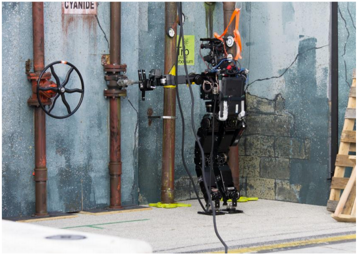
Outline

Why

Why: Important Problems

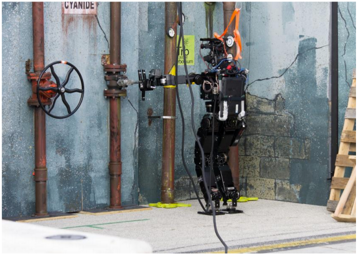
Why: Important Problems

- Autonomous robotics



Why: Important Problems

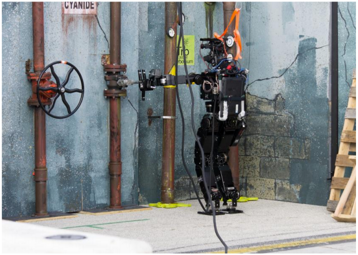
- Autonomous robotics



- *Elder care*

Why: Important Problems

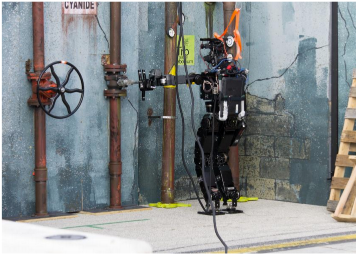
- Autonomous robotics



- *Elder care*
- *Exploration of unknown / dangerous environments*

Why: Important Problems

- Autonomous robotics



- *Elder care*
- *Exploration of unknown / dangerous environments*
- *Robotics for entertainment*

Why: Important Problems

- Autonomous robotics
- Financial applications

Why: Important Problems

- Autonomous robotics
- Financial applications

- *Trading execution algorithms*

Why: Important Problems

- Autonomous robotics
- Financial applications

- *Trading execution algorithms*
- *Portfolio management*

Why: Important Problems

- Autonomous robotics
- Financial applications

- *Trading execution algorithms*
- *Portfolio management*
- *Option pricing*

Why: Important Problems

- Autonomous robotics
- Financial applications
- Energy management

Why: Important Problems

- Autonomous robotics
- Financial applications
- Energy management

- *Energy grid integration*

Why: Important Problems

- Autonomous robotics
- Financial applications
- Energy management

- *Energy grid integration*
- *Maintenance scheduling*

Why: Important Problems

- Autonomous robotics
- Financial applications
- Energy management

- *Energy grid integration*
- *Maintenance scheduling*
- *Energy market regulation*

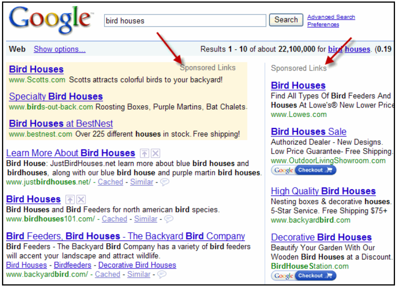
Why: Important Problems

- Autonomous robotics
- Financial applications
- Energy management

- *Energy grid integration*
- *Maintenance scheduling*
- *Energy market regulation*
- *Energy production management*

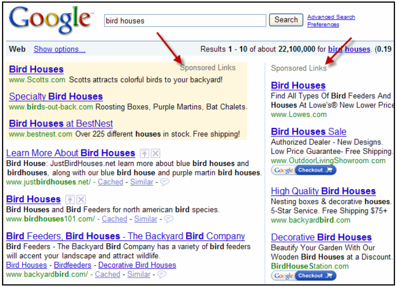
Why: Important Problems

- Autonomous robotics
- Financial applications
- Energy management
- Recommender systems



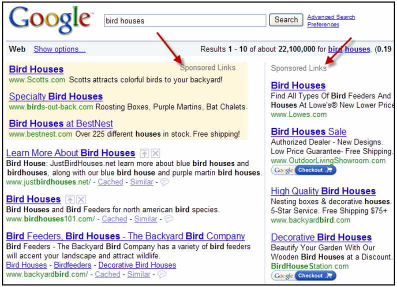
Why: Important Problems

- Autonomous robotics
- Financial applications
- Energy management
- Recommender systems



- *Web advertising*

Why: Important Problems

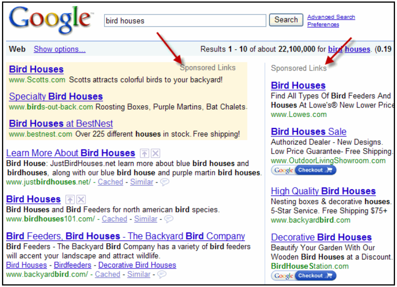


- Autonomous robotics
- Financial applications
- Energy management
- Recommender systems

- *Web advertising*
- *Product recommendation*

Why: Important Problems

- Autonomous robotics
- Financial applications
- Energy management
- Recommender systems



- *Web advertising*
- *Product recommendation*
- *Date matching*

Why: Important Problems

- Autonomous robotics
- Financial applications
- Energy management
- Recommender systems
- Social applications

Why: Important Problems

- Autonomous robotics
- Financial applications
- Energy management
- Recommender systems
- Social applications

- *Bike sharing optimization*

Why: Important Problems

- Autonomous robotics
- Financial applications
- Energy management
- Recommender systems
- Social applications

coursera
education for everyone

- *Bike sharing optimization*
- *Election campaign*

Why: Important Problems

- Autonomous robotics
- Financial applications
- Energy management
- Recommender systems
- Social applications

- *Bike sharing optimization*
- *Election campaign*
- *ER service optimization*

Why: Important Problems

- Autonomous robotics
- Financial applications
- Energy management
- Recommender systems
- Social applications

coursera
education for everyone

- *Bike sharing optimization*
- *Election campaign*
- *ER service optimization*
- *Intelligent Tutoring Systems*

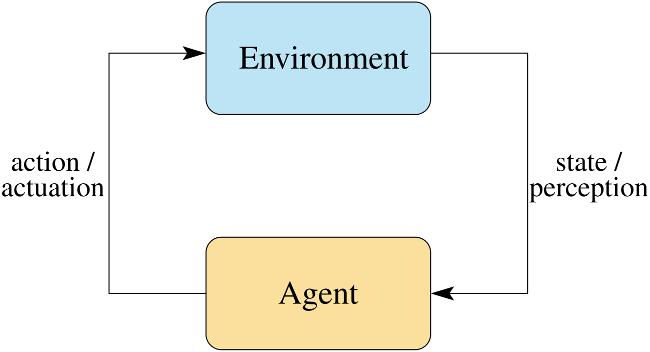
Why: Important Problems

- Autonomous robotics
- Financial applications
- Energy management
- Recommender systems
- Social applications
- And many more...

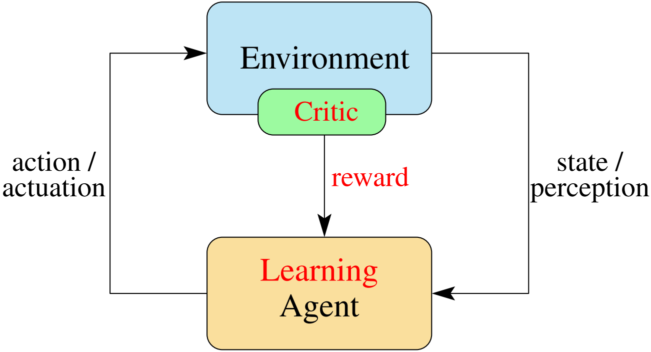
Outline

What

What: Sequential Decision-Making under Uncertainty



What: Sequential Decision-Making under Uncertainty



What: A Different Machine Learning Paradigm

- *Supervised learning:* an expert (*supervisor*) provides examples of the right strategy (e.g., classification of clinical images). *Supervision is expensive.*

What: A Different Machine Learning Paradigm

- *Supervised learning:* an expert (*supervisor*) provides examples of the right strategy (e.g., classification of clinical images). *Supervision is expensive.*
- *Unsupervised learning:* different objects are clustered together by similarity (e.g., clustering of images on the basis of their similarity). *No actual performance is optimized.*

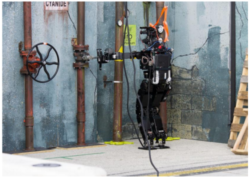
What: A Different Machine Learning Paradigm

- *Supervised learning:* an expert (*supervisor*) provides examples of the right strategy (e.g., classification of clinical images). *Supervision is expensive.*
- *Unsupervised learning:* different objects are clustered together by similarity (e.g., clustering of images on the basis of their similarity). *No actual performance is optimized.*
- *Reinforcement learning:* learning by direct interaction (e.g., autonomous robotics). *Minimum level of supervision (reward) and maximization of long term performance.*

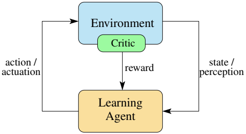
Outline

How: the Course

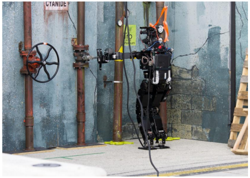
How: the Course

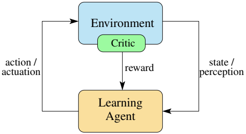


coursera
education for everyone



How: the Course





Formal and *rigorous* approach to the RL's way to sequential decision-making under uncertainty

How: the Course

- How to *model* an RL problem
- Models without states = MAB
- How to solve *exactly* an (small) MDP
- *Hands-on* session! (2h)
- How to solve *approximately* a (larger) MDP
- How to solve *incrementally* an MDP
- How to *efficiently* explore an MDP

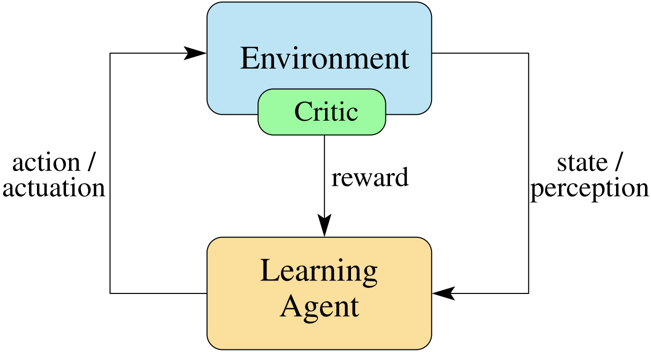
How to *model* an RL problem

The Markov Decision Process

The Model

Value Functions

The Agent-Environment Interaction Model



The Agent-Environment Interaction Model

The environment

- *Controllability*: fully (e.g., chess) or partially (e.g., portfolio optimization)
- *Uncertainty*: deterministic (e.g., chess) or stochastic (e.g., backgammon)
- *Reactive*: adversarial (e.g., chess) or fixed (e.g., tetris)
- *Observability*: full (e.g., chess) or partial (e.g., robotics)
- *Availability*: known (e.g., chess) or unknown (e.g., robotics)

The Agent-Environment Interaction Model

The environment

- *Controllability*: fully (e.g., chess) or partially (e.g., portfolio optimization)
- *Uncertainty*: deterministic (e.g., chess) or stochastic (e.g., backgammon)
- *Reactive*: adversarial (e.g., chess) or fixed (e.g., tetris)
- *Observability*: full (e.g., chess) or partial (e.g., robotics)
- *Availability*: known (e.g., chess) or unknown (e.g., robotics)

The critic

- Sparse (e.g., win or loose) vs informative (e.g., closer or further)
- Preference reward
- Frequent or sporadic
- Known or unknown

The Agent-Environment Interaction Model

The environment

- *Controllability*: fully (e.g., chess) or partially (e.g., portfolio optimization)
- *Uncertainty*: deterministic (e.g., chess) or stochastic (e.g., backgammon)
- *Reactive*: adversarial (e.g., chess) or fixed (e.g., tetris)
- *Observability*: full (e.g., chess) or partial (e.g., robotics)
- *Availability*: known (e.g., chess) or unknown (e.g., robotics)

The critic

- Sparse (e.g., win or loose) vs informative (e.g., closer or further)
- Preference reward
- Frequent or sporadic
- Known or unknown

The agent

- Open loop control
- Close loop control (i.e., *adaptive*)
- Non-stationary close loop control (i.e., *learning*)

Markov Decision Process

Definition (Markov decision process [1, 4, 3, 5, 2])

A ***Markov decision process*** *is defined as a tuple* $M = (X, A, p, r)$*:*

Markov Decision Process

Definition (Markov decision process [1, 4, 3, 5, 2])

*A **Markov decision process** is defined as a tuple $M = (X, A, p, r)$:*

- *X is the state space,*

Markov Decision Process

Definition (Markov decision process [1, 4, 3, 5, 2])

A ***Markov decision process*** *is defined as a tuple* $M = (X, A, p, r)$*:*

- X *is the state space,*
- A *is the action space,*

Markov Decision Process

Definition (Markov decision process [1, 4, 3, 5, 2])

*A **Markov decision process** is defined as a tuple* $M = (X, A, p, r)$*:*

- X *is the state space,*
- A *is the action space,*
- $p(y|x, a)$ *is the transition probability with*

$$p(y|x,a) = \mathbb{P}(x_{t+1} = y | x_t = x, a_t = a),$$

Markov Decision Process

Definition (Markov decision process [1, 4, 3, 5, 2])

*A **Markov decision process** is defined as a tuple $M = (X, A, p, r)$:*

- *X is the state space,*
- *A is the action space,*
- *$p(y|x, a)$ is the transition probability with*

$$p(y|x, a) = \mathbb{P}(x_{t+1} = y | x_t = x, a_t = a),$$

- *$r(x, a, y)$ is the reward of transition (x, a, y).*

Markov Decision Process: the Assumptions

Time assumption: time is discrete

$$t \rightarrow t+1$$

Possible relaxations

- Identify the proper time granularity
- Most of MDP literature extends to continuous time

Markov Decision Process: the Assumptions

Markov assumption: the current state x and action a are a sufficient statistics for the next state y

$$p(y|x,a) = \mathbb{P}(x_{t+1} = y | x_t = x, a_t = a)$$

Possible relaxations

- Define a new state $h_t = (x_t, x_{t-1}, x_{t-2}, \ldots)$
- Move to partially observable MDP (PO-MDP)
- Move to predictive state representation (PSR) model

Markov Decision Process: the Assumptions

Reward assumption: the reward is uniquely defined by a transition (or part of it)

$$r(x, a, y)$$

Possible relaxations

- Distinguish between global goal and reward function
- Move to inverse reinforcement learning (IRL) to induce the reward function from desired behaviors

Markov Decision Process: the Assumptions

Stationarity assumption: the dynamics and reward do not change over time

$$p(y|x,a) = \mathbb{P}(x_{t+1} = y | x_t = x, a_t = a) \qquad r(x,a,y)$$

Possible relaxations

- Identify and remove the non-stationary components (e.g., cyclo-stationary dynamics)
- Identify the time-scale of the changes

Question

Is the MDP formalism powerful enough?

$\Rightarrow$ *Let's try!*

Example: the Retail Store Management Problem

Description. At each month t, a store contains x_t *items* of a specific goods and the demand for that goods is D_t. At the end of each month the manager of the store can *order* a_t more items from his supplier. Furthermore we know that

- The *cost* of maintaining an inventory of x is $h(x)$.
- The *cost* to order a items is $C(a)$.
- The *income* for selling q items is $f(q)$.
- If the demand D is bigger than the available inventory x, customers that cannot be served leave.
- The *value of the remaining inventory* at the end of the year is $g(x)$.
- *Constraint*: the store has a maximum capacity M.

Example: the Retail Store Management Problem

- *State space*: $x \in X = \{0, 1, \ldots, M\}$.

Example: the Retail Store Management Problem

- *State space*: $x \in X = \{0, 1, \ldots, M\}$.
- *Action space*: it is not possible to order more items that the capacity of the store, then the action space should depend on the current state. Formally, at state x, $a \in A(x) = \{0, 1, \ldots, M - x\}$.

Example: the Retail Store Management Problem

- *State space*: $x \in X = \{0, 1, \ldots, M\}$.
- *Action space*: it is not possible to order more items that the capacity of the store, then the action space should depend on the current state. Formally, at state x, $a \in A(x) = \{0, 1, \ldots, M - x\}$.
- *Dynamics*: $x_{t+1} = [x_t + a_t - D_t]^+$.
  Problem: the dynamics should be Markov and stationary!

Example: the Retail Store Management Problem

- *State space*: $x \in X = \{0, 1, \ldots, M\}$.
- *Action space*: it is not possible to order more items that the capacity of the store, then the action space should depend on the current state. Formally, at state x, $a \in A(x) = \{0, 1, \ldots, M - x\}$.
- *Dynamics*: $x_{t+1} = [x_t + a_t - D_t]^+$.
  Problem: the dynamics should be Markov and stationary!
- The demand D_t is *stochastic and time-independent*. Formally, $D_t \overset{i.i.d.}{\sim} \mathcal{D}$.

Example: the Retail Store Management Problem

- *State space*: $x \in X = \{0, 1, \dots, M\}$.
- *Action space*: it is not possible to order more items that the capacity of the store, then the action space should depend on the current state. Formally, at state x, $a \in A(x) = \{0, 1, \dots, M - x\}$.
- *Dynamics*: $x_{t+1} = [x_t + a_t - D_t]^+$.
  Problem: the dynamics should be Markov and stationary!
- The demand D_t is *stochastic and time-independent*. Formally, $D_t \overset{i.i.d.}{\sim} \mathcal{D}$.
- *Reward*: $r_t = -C(a_t) - h(x_t + a_t) + f([x_t + a_t - x_{t+1}]^+)$.

Policy

Definition (Policy)

A *decision rule* π_t *can be*

- *Deterministic:* $\pi_t : X \rightarrow A$,
- *Stochastic:* $\pi_t : X \rightarrow \Delta(A)$,

Policy

Definition (Policy)

A *decision rule* π_t *can be*

- *Deterministic:* $\pi_t : X \rightarrow A$,
- *Stochastic:* $\pi_t : X \rightarrow \Delta(A)$,

A *policy* *(strategy, plan) can be*

- *Non-stationary:* $\pi = (\pi_0, \pi_1, \pi_2, \dots)$,
- *Stationary (Markovian):* $\pi = (\pi, \pi, \pi, \dots)$.

Policy

Definition (Policy)

A *decision rule* π_t *can be*

- *Deterministic:* $\pi_t : X \rightarrow A$,
- *Stochastic:* $\pi_t : X \rightarrow \Delta(A)$,

A *policy* *(strategy, plan) can be*

- *Non-stationary:* $\pi = (\pi_0, \pi_1, \pi_2, \dots)$,
- *Stationary (Markovian):* $\pi = (\pi, \pi, \pi, \dots)$.

Remark: MDP M + stationary policy $\pi \Rightarrow$ *Markov chain* of state X and transition probability $p(y|x) = p(y|x, \pi(x))$.

Example: the Retail Store Management Problem

- Stationary policy 1

$$\pi(x) = \begin{cases} M - x & \text{if } x < M/4 \\ 0 & \text{otherwise} \end{cases}$$

- Stationary policy 2

$$\pi(x) = \max\{(M - x)/2 - x; 0\}$$

- Non-stationary policy

$$\pi_t(x) = \begin{cases} M - x & \text{if } t < 6 \\ \lfloor (M - x)/5 \rfloor & \text{otherwise} \end{cases}$$

Outline

Outline

How to *efficiently* explore an MDP

The Exploration-Exploitation Dilemma

How to *efficiently* explore an MDP

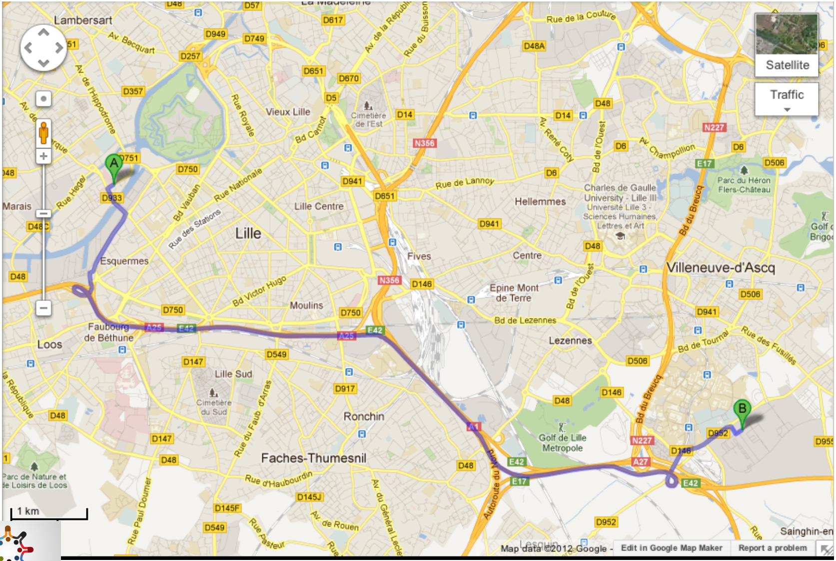
The Exploration-Exploitation Dilemma

Multi-Armed Bandit

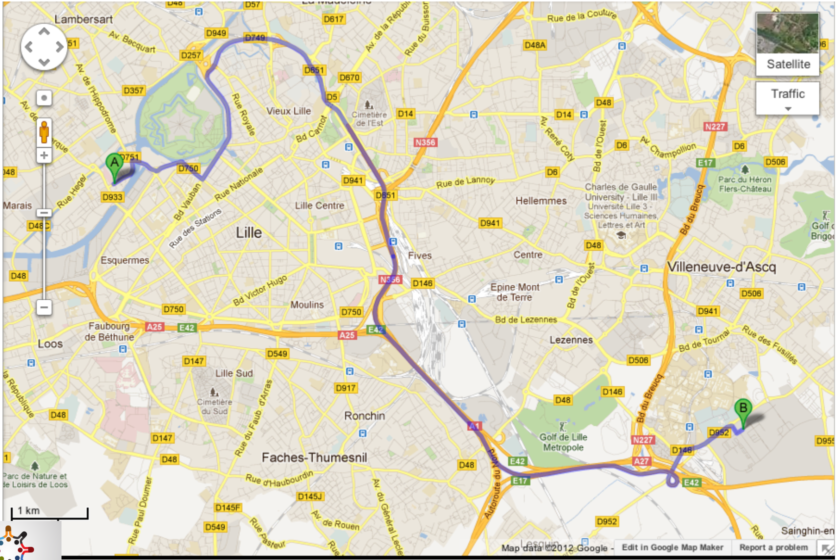
Contextual Linear Bandit

Reinforcement Learning

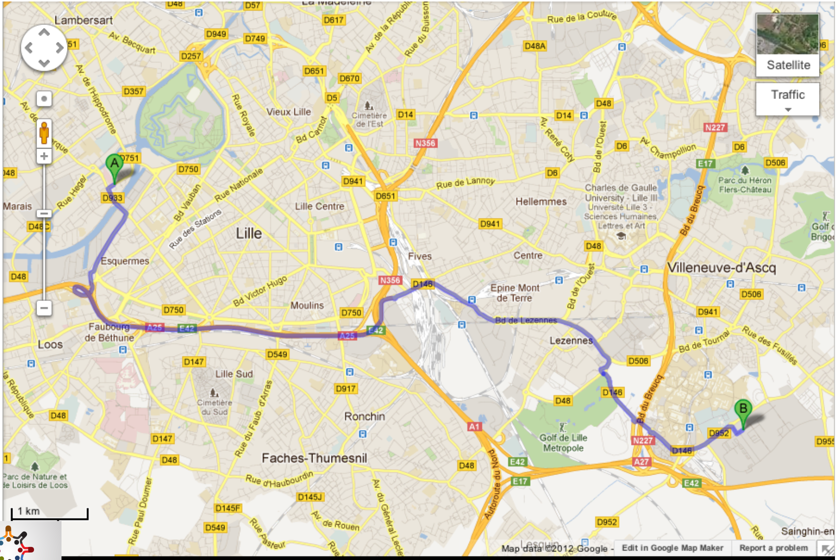
The Navigation Problem



The Navigation Problem



The Navigation Problem



The Navigation Problem

Question: which route should we take?

The Navigation Problem

Question: which route should we take?

Problem: each day we obtain a ***limited feedback***: traveling time of the *chosen route*

The Navigation Problem

Question: which route should we take?

Problem: each day we obtain a ***limited feedback***: traveling time of the *chosen route*

Results: if we do not repeatedly try different options we cannot learn.

The Navigation Problem

Question: which route should we take?

Problem: each day we obtain a ***limited feedback***: traveling time of the *chosen route*

Results: if we do not repeatedly try different options we cannot learn.

Solution: trade off between *optimization* and *learning*.

Learning the Optimal Policy

For $i = 1, \ldots, n$

1. Set $t = 0$
2. Set initial state x_0
3. **While** (x_t not terminal)

    3.1 Take action a_t ***according to a suitable exploration policy***

    3.2 Observe next state x_{t+1} and reward r_t

    3.3 Compute the temporal difference δ_t (e.g., Q-learning)

    3.4 Update the Q-function

$$\widehat{Q}(x_t, a_t) = \widehat{Q}(x_t, a_t) + \alpha(x_t, a_t)\delta_t$$

    3.5 Set $t = t + 1$

    EndWhile

EndFor

Learning the Optimal Policy

For $i = 1, \ldots, n$

1. Set $t = 0$
2. Set initial state x_0
3. **While** (x_t not terminal)

   3.1 ***Take action*** $a_t = \arg\max_a Q(x_t, a)$

   3.2 Observe next state x_{t+1} and reward r_t

   3.3 Compute the temporal difference δ_t (e.g., Q-learning)

   3.4 Update the Q-function

   $$\widehat{Q}(x_t, a_t) = \widehat{Q}(x_t, a_t) + \alpha(x_t, a_t)\delta_t$$

   3.5 Set $t = t + 1$

   EndWhile

EndFor

Learning the Optimal Policy

For $i = 1, \dots, n$

1. Set $t = 0$
2. Set initial state x_0
3. **While** (x_t not terminal)

   3.1 ***Take action*** $a_t = \arg\max_a Q(x_t, a)$

   3.2 Observe next state x_{t+1} and reward r_t

   3.3 Compute the temporal difference δ_t (e.g., Q-learning)

   3.4 Update the Q-function

$$\widehat{Q}(x_t, a_t) = \widehat{Q}(x_t, a_t) + \alpha(x_t, a_t)\delta_t$$

   3.5 Set $t = t + 1$

   EndWhile

EndFor

$\Rightarrow$ ***no convergence***

Learning the Optimal Policy

For $i = 1, \dots, n$

1. Set $t = 0$
2. Set initial state x_0
3. **While** (x_t not terminal)

   3.1 ***Take action*** $a_t \sim \mathcal{U}(A)$

   3.2 Observe next state x_{t+1} and reward r_t

   3.3 Compute the temporal difference δ_t (e.g., Q-learning)

   3.4 Update the Q-function

   $$\widehat{Q}(x_t, a_t) = \widehat{Q}(x_t, a_t) + \alpha(x_t, a_t)\delta_t$$

   3.5 Set $t = t + 1$

   EndWhile

EndFor

Learning the Optimal Policy

For $i = 1, \dots, n$

1. Set $t = 0$
2. Set initial state x_0
3. **While** (x_t not terminal)

   3.1 ***Take action*** $a_t \sim \mathcal{U}(A)$

   3.2 Observe next state x_{t+1} and reward r_t

   3.3 Compute the temporal difference δ_t (e.g., Q-learning)

   3.4 Update the Q-function

   $$\widehat{Q}(x_t, a_t) = \widehat{Q}(x_t, a_t) + \alpha(x_t, a_t)\delta_t$$

   3.5 Set $t = t + 1$

   EndWhile

EndFor

$\Rightarrow$ ***very poor rewards***

Outline

How to *efficiently* explore an MDP

The Exploration-Exploitation Dilemma

Multi-Armed Bandit

Contextual Linear Bandit

Reinforcement Learning

Reducing RL down to Multi-Armed Bandit

Definition (Markov decision process [1, 4, 3, 5, 2])

*A **Markov decision process** is defined as a tuple* $M = (X, A, p, r)$:

- ~~X *is the state space,*~~
- A *is the action space,*
- ~~$p(y|x, a)$ *is the transition probability*~~
- ~~$r(x, a, y)$ *is the reward of transition* (x, a, y)~~
  $\Rightarrow$ $r(a)$ *is the reward of action* a

Notice

For coherence with the bandit literature we use the notation

- $i = 1, \dots, K$ set of possible actions
- $t = 1, \dots, n$ time
- I_t action selected at time t
- $X_{i,t}$ reward for action i at time t

Learning the Optimal Policy

Objective: learn the optimal policy π^* ***as efficiently as possible***

Learning the Optimal Policy

Objective: learn the optimal policy π^* ***as efficiently as possible***

For $t = 1, \ldots, n$

1. ~~Set $t = 0$~~
2. ~~Set initial state x_0~~
3. ~~**While** (x_t not terminal)~~
   - 3.1 Take action a_t
   - 3.2 Observe ~~next state x_{t+1} and~~ reward r_t
   - 3.3 ~~Set $t = t + 1$~~

   ~~**EndWhile**~~

EndFor

The Multi–armed Bandit Protocol

The learner has $i = 1, \ldots, K$ arms (actions)

At each round $t = 1, \ldots, n$

The Multi–armed Bandit Protocol

The learner has $i = 1, \ldots, K$ arms (actions)

At each round $t = 1, \ldots, n$

- At the same time

The Multi–armed Bandit Protocol

The learner has $i = 1, \ldots, K$ arms (actions)

At each round $t = 1, \ldots, n$

- At the same time
  - The environment chooses a vector of *rewards* $\{X_{i,t}\}_{i=1}^{K}$
  - The learner chooses an arm I_t

The Multi–armed Bandit Protocol

The learner has $i = 1, \ldots, K$ arms (actions)

At each round $t = 1, \ldots, n$

- At the same time
  - The environment chooses a vector of *rewards* $\{X_{i,t}\}_{i=1}^{K}$
  - The learner chooses an arm I_t
- The learner receives a reward $X_{I_t,t}$

The Multi–armed Bandit Protocol

The learner has $i = 1, \ldots, K$ arms (actions)

At each round $t = 1, \ldots, n$

- At the same time
  - The environment chooses a vector of *rewards* $\{X_{i,t}\}_{i=1}^{K}$
  - The learner chooses an arm I_t
- The learner receives a reward $X_{I_t,t}$
- The environment ***does not*** reveal the rewards of the other arms

Paradigmatic Example

Imagine you are a doctor:

- patients visit you *one after another* for a given disease
- you prescribe one of the (say) *5 treatments* available
- the treatments are *not equally efficient*
- you do not know which one is the best, you *observe the effect* of the prescribed treatment on each patient

$\Rightarrow$ What do you do?

- You must choose each prescription using only the *previous observations*
- Your goal is not to estimate each treatment's efficiency precisely, but to *heal as many patients as possible*

The (stochastic) Multi-Armed Bandit Model

Environment K arms with parameters $\theta = (\theta_1, \dots, \theta_K)$ such that for any possible choice of arm $I_t \in \{1, \dots, K\}$ at time t, one receives the reward

$$X_t = X_{I_t,t}$$

where, for any $1 \le i \le K$ and $s \ge 1$, $X_{i,s} \sim \nu_i$, and the $(X_{i,s})_{i,s}$ are independent.

Reward distributions $\nu_i \in \mathcal{F}_i$ parametric family, or not. Examples: canonical exponential family, general bounded rewards

Example Bernoulli rewards: $\theta \in [0,1]^K$, $\nu_i = \mathcal{B}(\theta_i)$

Strategy The agent's actions follow a dynamical strategy $\pi = (\pi_1, \pi_2, \dots)$ such that

$$I_t = \pi_t(X_1, \dots, X_{t-1})$$

The Multi–armed Bandit Game (cont'd)

Goal: Choose π so as to maximize

$$\mathbb{E}^{\mathcal{A}}[S_n] = \sum_{t=1}^{n}\sum_{i=1}^{K}\mathbb{E}\big[\mathbb{E}[X_t\mathbb{I}\{I_t = i\}|X_1,\dots,X_{t-1}]\big]$$
$$= \sum_{i=1}^{K}\mu_i\mathbb{E}[T_{i,n}]$$

where $T_{i,n} = \sum_{t\le n}\mathbb{I}\{I_t = i\}$ is the number of draws of arm i up to time n, and $\mu_i = E(\nu_i)$.

The Multi–armed Bandit Game (cont'd)

Goal: Choose π so as to maximize

$$\mathbb{E}^{\mathcal{A}}[S_n] = \sum_{t=1}^{n}\sum_{i=1}^{K}\mathbb{E}\big[\mathbb{E}[X_t \mathbb{I}\{I_t = i\}|X_1,\dots,X_{t-1}]\big]$$
$$= \sum_{i=1}^{K}\mu_i \mathbb{E}[T_{i,n}]$$

where $T_{i,n} = \sum_{t \leq n} \mathbb{I}\{I_t = i\}$ is the number of draws of arm i up to time n, and $\mu_i = E(\nu_i)$.

$\Longrightarrow$ Equivalent to minimizing the regret

$$R_n(\mathcal{A}) = \max_{i=1,\dots,K} \mathbb{E}\Big[\sum_{t=1}^{n} X_{i,t}\Big] - \mathbb{E}\Big[\sum_{t=1}^{n} X_{I_t,t}\Big]$$

where $\mu^* \in \max\{\mu_i : 1 \leq i \leq K\}$.

The Exploration–Exploitation Lemma

Problem 1: The environment ***does not*** reveal the rewards of the arms not pulled by the learner

The Exploration–Exploitation Lemma

Problem 1: The environment ***does not*** reveal the rewards of the arms not pulled by the learner
$\Rightarrow$ the learner should *gain information* by repeatedly pulling all the arms

The Exploration–Exploitation Lemma

Problem 1: The environment ***does not*** reveal the rewards of the arms not pulled by the learner
$\Rightarrow$ the learner should *gain information* by repeatedly pulling all the arms

Problem 2: Whenever the learner pulls a ***bad arm***, it suffers some regret

The Exploration–Exploitation Lemma

Problem 1: The environment ***does not*** reveal the rewards of the arms not pulled by the learner
$\Rightarrow$ the learner should *gain information* by repeatedly pulling all the arms

Problem 2: Whenever the learner pulls a ***bad arm***, it suffers some regret
$\Rightarrow$ the learner should *reduce the regret* by repeatedly pulling the best arm

The Exploration–Exploitation Lemma

Problem 1: The environment ***does not*** reveal the rewards of the arms not pulled by the learner
$\Rightarrow$ the learner should *gain information* by repeatedly pulling all the arms

Problem 2: Whenever the learner pulls a ***bad arm***, it suffers some regret
$\Rightarrow$ the learner should *reduce the regret* by repeatedly pulling the best arm

Challenge: The learner should solve two opposite problems!

The Exploration–Exploitation Lemma

Problem 1: The environment ***does not*** reveal the rewards of the arms not pulled by the learner
$\Rightarrow$ the learner should *gain information* by repeatedly pulling all the arms
$\Rightarrow$ ***exploration***

Problem 2: Whenever the learner pulls a ***bad arm***, it suffers some regret
$\Rightarrow$ the learner should *reduce the regret* by repeatedly pulling the best arm

Challenge: The learner should solve two opposite problems!

The Exploration–Exploitation Lemma

Problem 1: The environment ***does not*** reveal the rewards of the arms not pulled by the learner
⇒ the learner should *gain information* by repeatedly pulling all the arms
⇒ ***exploration***

Problem 2: Whenever the learner pulls a ***bad arm***, it suffers some regret
⇒ the learner should *reduce the regret* by repeatedly pulling the best arm
⇒ ***exploitation***

Challenge: The learner should solve two opposite problems!

The Exploration–Exploitation Lemma

Problem 1: The environment ***does not*** reveal the rewards of the arms not pulled by the learner
$\Rightarrow$ the learner should *gain information* by repeatedly pulling all the arms
$\Rightarrow$ ***exploration***

Problem 2: Whenever the learner pulls a ***bad arm***, it suffers some regret
$\Rightarrow$ the learner should *reduce the regret* by repeatedly pulling the best arm
$\Rightarrow$ ***exploitation***

Challenge: The learner should solve the *exploration-exploitation* dilemma!

The Multi–armed Bandit Game (cont'd)

Examples

- Packet routing
- Clinical trials
- Web advertising
- Computer games
- Resource mining
- ...

The Stochastic Multi–armed Bandit Problem

Definition

The environment is stochastic

- *Each arm has a distribution ν_i bounded in $[0, 1]$ and characterized by an expected value μ_i*
- *The rewards are i.i.d. $X_{i,t} \sim \nu_i$ (as in the MDP model)*

The Stochastic Multi–armed Bandit Problem (cont'd)

Notation

- Number of times arm i has been pulled after n rounds

$$T_{i,n} = \sum_{t=1}^{n} \mathbb{I}\{I_t = i\}$$

The Stochastic Multi–armed Bandit Problem (cont'd)

Notation

- Number of times arm i has been pulled after n rounds

$$T_{i,n} = \sum_{t=1}^{n} \mathbb{I}\{I_t = i\}$$

- Regret

$$R_n(\mathcal{A}) = \max_{i=1,\dots,K} \mathbb{E}\Big[\sum_{t=1}^{n} X_{i,t}\Big] - \mathbb{E}\Big[\sum_{t=1}^{n} X_{I_t,t}\Big]$$

The Stochastic Multi–armed Bandit Problem (cont'd)

Notation

- Number of times arm i has been pulled after n rounds

$$T_{i,n} = \sum_{t=1}^{n} \mathbb{I}\{I_t = i\}$$

- Regret

$$R_n(\mathcal{A}) = \max_{i=1,\dots,K} (n\mu_i) - \mathbb{E}\Big[\sum_{t=1}^{n} X_{I_t,t}\Big]$$

The Stochastic Multi–armed Bandit Problem (cont'd)

Notation

- Number of times arm i has been pulled after n rounds

$$T_{i,n} = \sum_{t=1}^{n} \mathbb{I}\{I_t = i\}$$

- Regret

$$R_n(\mathcal{A}) = \max_{i=1,\dots,K} (n\mu_i) - \sum_{i=1}^{K} \mathbb{E}[T_{i,n}]\mu_i$$

The Stochastic Multi–armed Bandit Problem (cont'd)

Notation

- Number of times arm i has been pulled after n rounds

$$T_{i,n} = \sum_{t=1}^{n} \mathbb{I}\{I_t = i\}$$

- Regret

$$R_n(\mathcal{A}) = n\mu_{i^*} - \sum_{i=1}^{K} \mathbb{E}[T_{i,n}]\mu_i$$

The Stochastic Multi–armed Bandit Problem (cont'd)

Notation

- Number of times arm i has been pulled after n rounds

$$T_{i,n} = \sum_{t=1}^{n} \mathbb{I}\{I_t = i\}$$

- Regret

$$R_n(\mathcal{A}) = \sum_{i \neq i^*} \mathbb{E}[T_{i,n}](\mu_{i^*} - \mu_i)$$

The Stochastic Multi–armed Bandit Problem (cont'd)

Notation

- Number of times arm i has been pulled after n rounds

$$T_{i,n} = \sum_{t=1}^{n} \mathbb{I}\{I_t = i\}$$

- Regret

$$R_n(\mathcal{A}) = \sum_{i \neq i^*} \mathbb{E}[T_{i,n}] \Delta_i$$

The Stochastic Multi–armed Bandit Problem (cont'd)

Notation

- Number of times arm i has been pulled after n rounds

$$T_{i,n} = \sum_{t=1}^{n} \mathbb{I}\{I_t = i\}$$

- Regret

$$R_n(\mathcal{A}) = \sum_{i \neq i^*} \mathbb{E}[T_{i,n}]\Delta_i$$

- Gap $\Delta_i = \mu_{i^*} - \mu_i$

The Stochastic Multi–armed Bandit Problem (cont'd)

$$R_n(\mathcal{A}) = \sum_{i \neq i^*} \mathbb{E}[T_{i,n}]\Delta_i$$

$\Rightarrow$ we only need to study the *expected number of pulls* of the *suboptimal* arms

Outline

The Stochastic Multi–armed Bandit Problem (cont'd)

Optimism in Face of Uncertainty Learning (OFUL)

Whenever we are *uncertain* about the outcome of an arm, we consider the *best possible world* and choose the *best arm*.

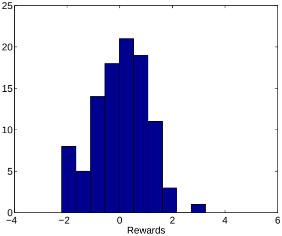
The Stochastic Multi–armed Bandit Problem (cont'd)

Optimism in Face of Uncertainty Learning (OFUL)

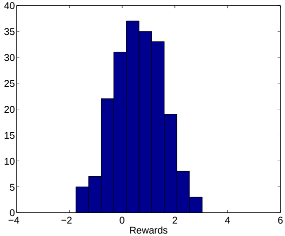
Whenever we are *uncertain* about the outcome of an arm, we consider the *best possible world* and choose the *best arm*.
Why it works:

- If the *best possible world* is correct $\Rightarrow$ *no regret*
- If the *best possible world* is wrong $\Rightarrow$ *the reduction in the uncertainty is maximized*

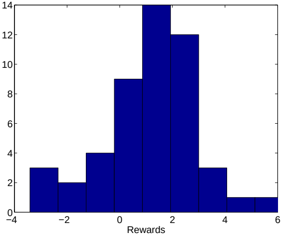
The Stochastic Multi–armed Bandit Problem (cont'd)



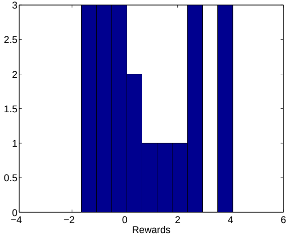
pulls = 100



pulls = 200



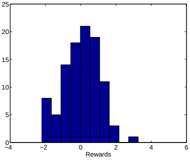
pulls = 50

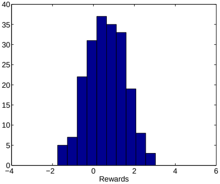


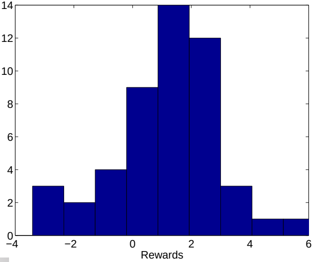
pulls = 20

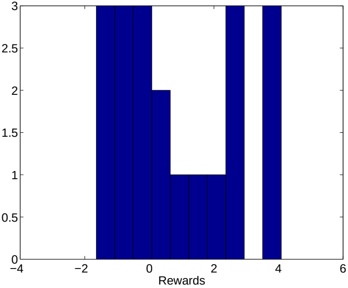
The Stochastic Multi–armed Bandit Problem (cont'd)

Optimism in face of uncertainty



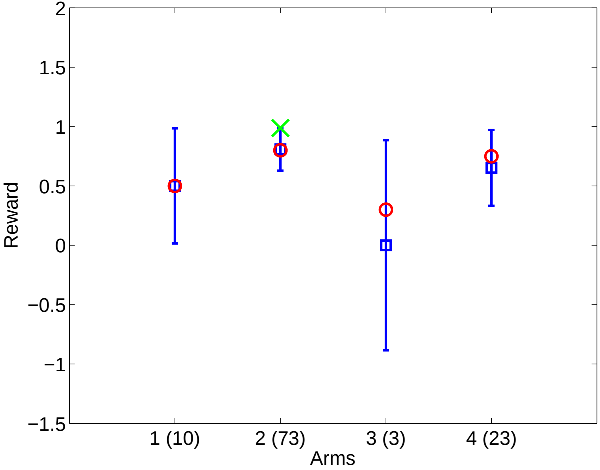






The Upper–Confidence Bound (UCB) Algorithm

The idea



The Upper–Confidence Bound (UCB) Algorithm

Show time!

The Upper–Confidence Bound (UCB) Algorithm (cont'd)

At each round $t = 1, \ldots, n$

- Compute the *score* of each arm i

$$B_i = (\textit{optimistic} \text{ score of arm } i)$$

- Pull arm

$$I_t = \arg \max_{i=1,\ldots,K} B_{i,s,t}$$

- Update the number of pulls $T_{I_t,t} = T_{I_t,t-1} + 1$ and the other statistics

The Upper–Confidence Bound (UCB) Algorithm (cont'd)

The score (with parameters ρ and δ)

$$B_i = (\textit{optimistic} \text{ score of arm } i)$$

The Upper–Confidence Bound (UCB) Algorithm (cont'd)

The score (with parameters ρ and δ)

$B_{i,s,t}$ = (*optimistic* score of arm i if pulled s times up to round t)

The Upper–Confidence Bound (UCB) Algorithm (cont'd)

The score (with parameters ρ and δ)

$B_{i,s,t} =$ (*optimistic* score of arm i if pulled s times up to round t)

Optimism in face of uncertainty:
Current knowledge: average rewards $\hat{\mu}_{i,s}$
Current uncertainty: number of pulls s

The Upper–Confidence Bound (UCB) Algorithm (cont'd)

The score (with parameters ρ and δ)

$$B_{i,s,t} = \text{knowledge} \underbrace{+}_{\textit{optimism}} \text{uncertainty}$$

Optimism in face of uncertainty:
Current knowledge: average rewards $\hat{\mu}_{i,s}$
Current uncertainty: number of pulls s

The Upper–Confidence Bound (UCB) Algorithm (cont'd)

The score (with parameters ρ and δ)

$$B_{i,s,t} = \hat{\mu}_{i,s} + \rho\sqrt{\frac{\log 1/\delta}{2s}}$$

Optimism in face of uncertainty:
Current knowledge: average rewards $\hat{\mu}_{i,s}$
Current uncertainty: number of pulls s

The Upper–Confidence Bound (UCB) Algorithm (cont'd)

At each round $t = 1, \ldots, n$

- Compute the *score* of each arm i

$$B_{i,t} = \hat{\mu}_{i,T_{i,t}} + \rho \sqrt{\frac{\log(t)}{2T_{i,t}}}$$

- Pull arm

$$I_t = \arg\max_{i=1,\ldots,K} B_{i,t}$$

- Update the number of pulls $T_{I_t,t} = T_{I_t,t-1} + 1$ and $\hat{\mu}_{i,T_{i,t}}$

The Upper–Confidence Bound (UCB) Algorithm (cont'd)

Theorem

Let $X_1, \ldots, X_n$ be i.i.d. samples from a distribution bounded in $[a, b]$, then for any $\delta \in (0, 1)$

$$\mathbb{P}\left[\left|\frac{1}{n}\sum_{t=1}^{n} X_t - \mathbb{E}[X_1]\right| > (b-a)\sqrt{\frac{\log 2/\delta}{2n}}\right] \leq \delta$$

The Upper–Confidence Bound (UCB) Algorithm (cont'd)

After s pulls, arm i

$$\mathbb{P}\left[\mathbb{E}[X_i] \leq \frac{1}{s}\sum_{t=1}^{s} X_{i,t} + \sqrt{\frac{\log 1/\delta}{2s}}\right] \geq 1 - \delta$$

The Upper–Confidence Bound (UCB) Algorithm (cont'd)

After s pulls, arm i

$$\mathbb{P}\left[\mu_i \leq \hat{\mu}_{i,s} + \sqrt{\frac{\log 1/\delta}{2s}}\right] \geq 1 - \delta$$

The Upper–Confidence Bound (UCB) Algorithm (cont'd)

After s pulls, arm i

$$\mathbb{P}\left[\mu_i \leq \hat{\mu}_{i,s} + \sqrt{\frac{\log 1/\delta}{2s}}\right] \geq 1 - \delta$$

$\Rightarrow$ UCB uses an *upper confidence bound* on the expectation

The Upper–Confidence Bound (UCB) Algorithm (cont'd)

Theorem

For any set of K arms with distributions bounded in $[0, b]$, if $\delta = 1/t$, then UCB(ρ) with $\rho > 1$, achieves a regret

$$R_n(\mathcal{A}) \leq \sum_{i \neq i^*} \left[\frac{4b^2}{\Delta_i} \rho \log(n) + \Delta_i \left(\frac{3}{2} + \frac{1}{2(\rho - 1)} \right) \right]$$

The Upper–Confidence Bound (UCB) Algorithm (cont'd)

Let $K = 2$ with $i^* = 1$

$$R_n(\mathcal{A}) \leq O\left(\frac{1}{\Delta}\rho\log(n)\right)$$

Remark 1: the *cumulative* regret slowly increases as $\log(n)$

The Upper–Confidence Bound (UCB) Algorithm (cont'd)

Let $K = 2$ with $i^* = 1$

$$R_n(\mathcal{A}) \leq O\left(\frac{1}{\Delta}\rho \log(n)\right)$$

Remark 1: the *cumulative* regret slowly increases as $\log(n)$
Remark 2: the *smaller the gap* the *bigger the regret*... why?

The Upper–Confidence Bound (UCB) Algorithm (cont'd)

Show time (again)!

Outline

Asymptotically Optimal Strategies

- A strategy π is said to be consistent if, for any $(\nu_i)_i \in \mathcal{F}^K$,

$$\frac{1}{n}\mathbb{E}[S_n] \to \mu^*$$

- The strategy is efficient if for all $\theta \in [0,1]^K$ and all $\alpha > 0$,

$$R_n(\mathcal{A}) = o(n^\alpha)$$

- There are efficient strategies and we consider the best achievable asymptotic performance among efficient strategies

The Bound of Lai and Robbins

One-parameter reward distribution $\nu_i = \nu_{\theta_i}, \theta_i \in \Theta \subset \mathbb{R}$.

Theorem [Lai and Robbins, '85]

If π is an efficient strategy, then, for any $\theta \in \Theta^K$,

$$\liminf_{n\to\infty} \frac{R_n(\mathcal{A})}{\log(n)} \geq \sum_{i:\mu_i<\mu^*} \frac{\mu^* - \mu_i}{\mathrm{KL}(\nu_i, \nu^*)}$$

where $\mathrm{KL}(\nu, \nu')$ denotes the Kullback-Leibler divergence

For example, in the Bernoulli case:

$$KL\big(\mathcal{B}(p), \mathcal{B}(q)\big) = d_{\text{BER}}(p, q) = p \log \frac{p}{q} + (1-p) \log \frac{1-p}{1-q}$$

The Bound of Burnetas and Katehakis

More general reward distributions $\nu_i \in \mathcal{F}_i$

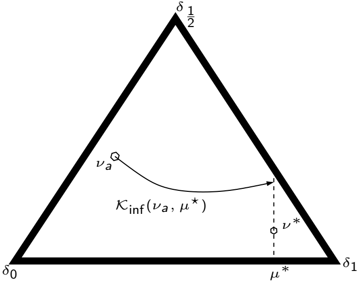
Theorem [Burnetas and Katehakis, '96]

If π is an efficient strategy, then, for any $\theta \in [0,1]^K$,

$$\liminf_{n\to\infty} \frac{R_n}{\log(n)} \geq \sum_{i:\mu_i<\mu^*} \frac{\mu^* - \mu_i}{K_{inf}(\nu_i, \mu^*)}$$

where

$$K_{inf}(\nu_i, \mu^*) = \inf\left\{K(\nu_i, \nu') : \nu' \in \mathcal{F}_i, E(\nu') \geq \mu^*\right\}$$



Intuition

- First assume that μ^* is known and that n is fixed
- How many draws n_i of ν_i are necessary to know that $\mu_i < \mu^*$ with probability at least $1 - 1/n$?
- Test: $H_0 : \mu_i = \mu^*$ against $H_1 : \nu = \nu_i$
- Stein's Lemma: if the first type error $\alpha_{n_i} \leq 1/n$, then

$$\beta_{n_i} \gtrsim \exp\big(- n_i K_{inf}(\nu_i, \mu^*)\big)$$

$\Longrightarrow$ it can be smaller than $1/n$ if

$$n_i \geq \frac{\log(n)}{K_{inf}(\nu_i, \mu^*)}$$

- How to do as well without knowing μ^* and n in advance? Not asymptotically?

Outline

The Worst–case Performance

Remark: the regret bound is *distribution–dependent*

$$R_n(\mathcal{A}; \Delta) \leq O\left(\frac{1}{\Delta}\rho \log(n)\right)$$

The Worst–case Performance

Remark: the regret bound is *distribution–dependent*

$$R_n(\mathcal{A}; \Delta) \leq O\left(\frac{1}{\Delta} \rho \log(n)\right)$$

Meaning: the algorithm is able to *adapt to the specific problem* at hand!

The Worst–case Performance

Remark: the regret bound is *distribution–dependent*

$$R_n(\mathcal{A};\Delta) \leq O\left(\frac{1}{\Delta}\rho\log(n)\right)$$

Meaning: the algorithm is able to *adapt to the specific problem* at hand!

Worst–case performance: what is the distribution which leads to the worst possible performance of UCB? what is the distribution–free performance of UCB?

$$R_n(\mathcal{A}) = \sup_{\Delta} R_n(\mathcal{A};\Delta)$$

The Worst–case Performance

Problem: it seems like if $\Delta \rightarrow 0$ then the regret tends to infinity...

The Worst–case Performance

Problem: it seems like if $\Delta \to 0$ then the regret tends to infinity...
... nosense because the regret is defined as

$$R_n(\mathcal{A}; \Delta) = \mathbb{E}[T_{2,n}]\Delta$$

The Worst–case Performance

Problem: it seems like if $\Delta \to 0$ then the regret tends to infinity...
... nosense because the regret is defined as

$$R_n(\mathcal{A}; \Delta) = \mathbb{E}[T_{2,n}]\Delta$$

then if Δ_i is small, the regret is also small...

The Worst–case Performance

Problem: it seems like if $\Delta \to 0$ then the regret tends to infinity...
... nosense because the regret is defined as

$$R_n(\mathcal{A}; \Delta) = \mathbb{E}[T_{2,n}]\Delta$$

then if Δ_i is small, the regret is also small...
In fact

$$R_n(\mathcal{A}; \Delta) = \min\left\{ O\left(\frac{1}{\Delta}\rho \log(n)\right), \mathbb{E}[T_{2,n}]\Delta \right\}$$

The Worst–case Performance

Then

$$R_n(\mathcal{A}) = \sup_{\Delta} R_n(\mathcal{A}; \Delta) = \sup_{\Delta} \min \left\{ O\left(\frac{1}{\Delta}\rho \log(n)\right), n\Delta \right\} \approx \sqrt{n}$$

for $\Delta = \sqrt{1/n}$.

Remark: Non-stochastic bandits: it is possible to ensure the same $O(\sqrt{n})$ regret even *without any stochastic asumption on the reward process*.

Tuning the confidence δ of UCB

Remark: UCB is an *anytime* algorithm ($\delta = 1/t$)

$$B_{i,s,t} = \hat{\mu}_{i,s} + \rho\sqrt{\frac{\log t}{2s}}$$

Tuning the confidence δ of UCB

Remark: UCB is an *anytime* algorithm ($\delta = 1/t$)

$$B_{i,s,t} = \hat{\mu}_{i,s} + \rho\sqrt{\frac{\log t}{2s}}$$

Remark: If the time horizon n is known then the optimal choice is $\delta = 1/n$

$$B_{i,s,t} = \hat{\mu}_{i,s} + \rho\sqrt{\frac{\log n}{2s}}$$

Tuning the confidence δ of UCB (cont'd)

Intuition: UCB should pull the suboptimal arms

- *Enough*: so as to understand which arm is the best
- *Not too much*: so as to keep the regret as small as possible

Tuning the confidence δ of UCB (cont'd)

Intuition: UCB should pull the suboptimal arms

- *Enough*: so as to understand which arm is the best
- *Not too much*: so as to keep the regret as small as possible

The confidence $1-\delta$ has the following impact (similar for ρ)

- *Big* $1-\delta$: high level of *exploration*
- *Small* $1-\delta$: high level of *exploitation*

Tuning the confidence δ of UCB (cont'd)

Intuition: UCB should pull the suboptimal arms

- *Enough*: so as to understand which arm is the best
- *Not too much*: so as to keep the regret as small as possible

The confidence $1-\delta$ has the following impact (similar for ρ)

- *Big* $1-\delta$: high level of *exploration*
- *Small* $1-\delta$: high level of *exploitation*

Solution: depending on the time horizon, we can tune how to trade-off between exploration and exploitation

UCB Proof

Let's dig into the (1 page and half!!) proof.

Define the (high-probability) event *[statistics]*

$$\mathcal{E} = \left\{ \forall i, s \ \ \left| \hat{\mu}_{i,s} - \mu_i \right| \leq \sqrt{\frac{\log 1/\delta}{2s}} \right\}$$

By Chernoff-Hoeffding $\mathbb{P}[\mathcal{E}] \geq 1 - nK\delta$.

UCB Proof

Let's dig into the (1 page and half!!) proof.

Define the (high-probability) event *[statistics]*

$$\mathcal{E} = \left\{ \forall i, s \ \left| \hat{\mu}_{i,s} - \mu_i \right| \leq \sqrt{\frac{\log 1/\delta}{2s}} \right\}$$

By Chernoff-Hoeffding $\mathbb{P}[\mathcal{E}] \geq 1 - nK\delta$.
At time t we pull arm i *[algorithm]*

$$B_{i,T_{i,t-1}} \geq B_{i^*,T_{i^*,t-1}}$$

UCB Proof

Let's dig into the (1 page and half!!) proof.

Define the (high-probability) event *[statistics]*

$$\mathcal{E} = \left\{ \forall i, s \ \left| \hat{\mu}_{i,s} - \mu_i \right| \leq \sqrt{\frac{\log 1/\delta}{2s}} \right\}$$

By Chernoff-Hoeffding $\mathbb{P}[\mathcal{E}] \geq 1 - nK\delta$.
At time t we pull arm i *[algorithm]*

$$\hat{\mu}_{i,T_{i,t-1}} + \sqrt{\frac{\log 1/\delta}{2T_{i,t-1}}} \geq \hat{\mu}_{i^*,T_{i^*,t-1}} + \sqrt{\frac{\log 1/\delta}{2T_{i^*,t-1}}}$$

UCB Proof

Let's dig into the (1 page and half!!) proof.

Define the (high-probability) event *[statistics]*

$$\mathcal{E} = \left\{ \forall i, s \ \left| \hat{\mu}_{i,s} - \mu_i \right| \leq \sqrt{\frac{\log 1/\delta}{2s}} \right\}$$

By Chernoff-Hoeffding $\mathbb{P}[\mathcal{E}] \geq 1 - nK\delta$.
At time t we pull arm i *[algorithm]*

$$\hat{\mu}_{i,T_{i,t-1}} + \sqrt{\frac{\log 1/\delta}{2T_{i,t-1}}} \geq \hat{\mu}_{i^*,T_{i^*,t-1}} + \sqrt{\frac{\log 1/\delta}{2T_{i^*,t-1}}}$$

On the event $\mathcal{E}$ we have *[math]*

$$\mu_i + 2\sqrt{\frac{\log 1/\delta}{2T_{i,t-1}}} \geq \mu_{i^*}$$

UCB Proof (cont'd)

Assume t is the last time i is pulled, then $T_{i,n} = T_{i,t-1} + 1$, thus

$$\mu_i + 2\sqrt{\frac{\log 1/\delta}{2(T_{i,n} - 1)}} \geq \mu_{i^*}$$

UCB Proof (cont'd)

Assume t is the last time i is pulled, then $T_{i,n} = T_{i,t-1} + 1$, thus

$$\mu_i + 2\sqrt{\frac{\log 1/\delta}{2(T_{i,n} - 1)}} \geq \mu_{i^*}$$

Reordering *[math]*

$$T_{i,n} \leq \frac{\log 1/\delta}{2\Delta_i^2} + 1$$

under event $\mathcal{E}$ and thus with probability $1 - nK\delta$.

UCB Proof (cont'd)

Assume t is the last time i is pulled, then $T_{i,n} = T_{i,t-1} + 1$, thus

$$\mu_i + 2\sqrt{\frac{\log 1/\delta}{2(T_{i,n} - 1)}} \geq \mu_{i^*}$$

Reordering *[math]*

$$T_{i,n} \leq \frac{\log 1/\delta}{2\Delta_i^2} + 1$$

under event $\mathcal{E}$ and thus with probability $1 - nK\delta$.
Moving to the expectation *[statistics]*

$$\mathbb{E}[T_{i,n}] = \mathbb{E}[T_{i,n}\mathbb{I}\mathcal{E}] + \mathbb{E}[T_{i,n}\mathbb{I}\mathcal{E}^C]$$

UCB Proof (cont'd)

Assume t is the last time i is pulled, then $T_{i,n} = T_{i,t-1} + 1$, thus

$$\mu_i + 2\sqrt{\frac{\log 1/\delta}{2(T_{i,n} - 1)}} \geq \mu_{i^*}$$

Reordering *[math]*

$$T_{i,n} \leq \frac{\log 1/\delta}{2\Delta_i^2} + 1$$

under event $\mathcal{E}$ and thus with probability $1 - nK\delta$.
Moving to the expectation *[statistics]*

$$\mathbb{E}[T_{i,n}] \leq \frac{\log 1/\delta}{2\Delta_i^2} + 1 + n(nK\delta)$$

UCB Proof (cont'd)

Assume t is the last time i is pulled, then $T_{i,n} = T_{i,t-1} + 1$, thus

$$\mu_i + 2\sqrt{\frac{\log 1/\delta}{2(T_{i,n} - 1)}} \geq \mu_{i^*}$$

Reordering *[math]*

$$T_{i,n} \leq \frac{\log 1/\delta}{2\Delta_i^2} + 1$$

under event $\mathcal{E}$ and thus with probability $1 - nK\delta$.
Moving to the expectation *[statistics]*

$$\mathbb{E}[T_{i,n}] \leq \frac{\log 1/\delta}{2\Delta_i^2} + 1 + n(nK\delta)$$

Trading-off the two terms $\delta = 1/n^2$, we obtain

$$\hat{\mu}_{i,T_{i,t-1}} + \sqrt{\frac{2\log n}{2T_{i,t-1}}}$$

UCB Proof (cont'd)

Trading-off the two terms $\delta = 1/n^2$, we obtain

$$\hat{\mu}_{i,T_{i,t-1}} + \sqrt{\frac{2\log n}{2T_{i,t-1}}}$$

and

$$\mathbb{E}[T_{i,n}] \leq \frac{\log n}{\Delta_i^2} + 1 + K$$

Tuning the confidence δ of UCB (cont'd)

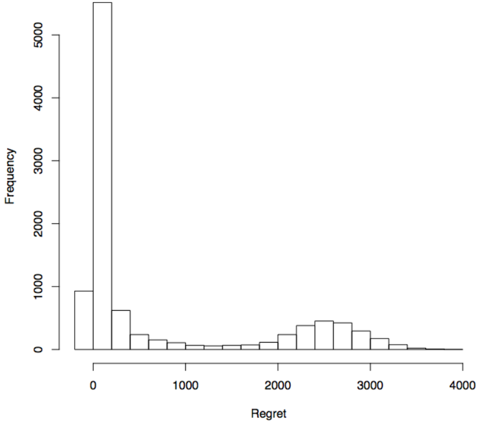
Multi–armed Bandit: the same for $\delta = 1/t$ and $\delta = 1/n$...

Tuning the confidence δ of UCB (cont'd)

Multi–armed Bandit: the same for $\delta = 1/t$ and $\delta = 1/n$...
... **almost** (i.e., in expectation)

Tuning the confidence δ of UCB (cont'd)

The value–at–risk of the regret for UCB-anytime



Tuning the ρ of UCB (cont'd)

UCB values (for the $\delta = 1/n$ algorithm)

$$B_{i,s} = \hat{\mu}_{i,s} + \rho\sqrt{\frac{\log n}{2s}}$$

Tuning the ρ of UCB (cont'd)

UCB values (for the $\delta = 1/n$ algorithm)

$$B_{i,s} = \hat{\mu}_{i,s} + \rho\sqrt{\frac{\log n}{2s}}$$

Theory

- $\rho < 0.5$, polynomial regret w.r.t. n
- $\rho > 0.5$, logarithmic regret w.r.t. n

Tuning the ρ of UCB (cont'd)

UCB values (for the $\delta = 1/n$ algorithm)

$$B_{i,s} = \hat{\mu}_{i,s} + \rho\sqrt{\frac{\log n}{2s}}$$

Theory

- $\rho < 0.5$, polynomial regret w.r.t. n
- $\rho > 0.5$, logarithmic regret w.r.t. n

Practice: $\rho = 0.2$ is often the best choice

Tuning the ρ of UCB (cont'd)

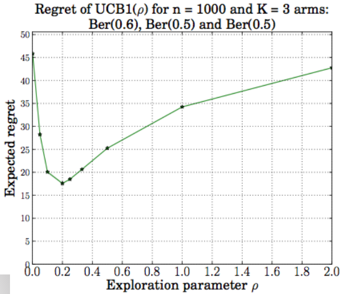
UCB values (for the $\delta = 1/n$ algorithm)

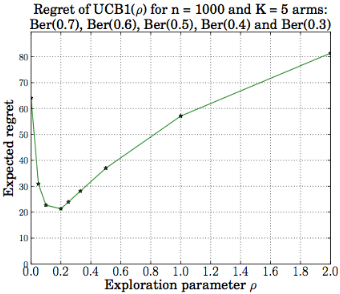
$$B_{i,s} = \hat{\mu}_{i,s} + \rho\sqrt{\frac{\log n}{2s}}$$

Theory

- $\rho < 0.5$, polynomial regret w.r.t. n
- $\rho > 0.5$, logarithmic regret w.r.t. n

Practice: $\rho = 0.2$ is often the best choice





Improvements: UCB-V

Idea: use *empirical Bernstein bounds* for more accurate c.i.

Improvements: UCB-V

Idea: use *empirical Bernstein bounds* for more accurate c.i.

Algorithm

- Compute the *score* of each arm i

$$B_{i,t} = \hat{\mu}_{i,T_{i,t}} + \rho\sqrt{\frac{\log(t)}{2T_{i,t}}}$$

- Pull arm

$$I_t = \arg\max_{i=1,\dots,K} B_{i,t}$$

- Update the number of pulls $T_{I_t,t}$, $\hat{\mu}_{i,T_{i,t}}$

Improvements: UCB-V

Idea: use *empirical Bernstein bounds* for more accurate c.i.

Algorithm

- Compute the *score* of each arm i

$$B_{i,t} = \hat{\mu}_{i,T_{i,t}} + \sqrt{\frac{2\hat{\sigma}^2_{i,T_{i,t}} \log t}{T_{i,t}}} + \frac{8 \log t}{3 T_{i,t}}$$

- Pull arm

$$I_t = \arg \max_{i=1,\dots,K} B_{i,t}$$

- Update the number of pulls $T_{I_t,t}$, $\hat{\mu}_{i,T_{i,t}}$ and $\hat{\sigma}^2_{i,T_{i,t}}$

Improvements: UCB-V

Idea: use *empirical Bernstein bounds* for more accurate c.i.

Algorithm

- Compute the *score* of each arm i

$$B_{i,t} = \hat{\mu}_{i,T_{i,t}} + \sqrt{\frac{2\hat{\sigma}^2_{i,T_{i,t}} \log t}{T_{i,t}}} + \frac{8 \log t}{3 T_{i,t}}$$

- Pull arm

$$I_t = \arg\max_{i=1,\dots,K} B_{i,t}$$

- Update the number of pulls $T_{I_t,t}$, $\hat{\mu}_{i,T_{i,t}}$ and $\hat{\sigma}^2_{i,T_{i,t}}$

Regret

$$R_n \leq O\Big(\frac{1}{\Delta} \log n\Big)$$

Improvements: UCB-V

Idea: use *empirical Bernstein bounds* for more accurate c.i.

Algorithm

- Compute the *score* of each arm i

$$B_{i,t} = \hat{\mu}_{i,T_{i,t}} + \sqrt{\frac{2\hat{\sigma}^2_{i,T_{i,t}} \log t}{T_{i,t}}} + \frac{8 \log t}{3 T_{i,t}}$$

- Pull arm

$$I_t = \arg\max_{i=1,\dots,K} B_{i,t}$$

- Update the number of pulls $T_{I_t,t}$, $\hat{\mu}_{i,T_{i,t}}$ and $\hat{\sigma}^2_{i,T_{i,t}}$

Regret

$$R_n \leq O\Big(\frac{\sigma^2}{\Delta} \log n\Big)$$

Improvements: KL-UCB

Idea: use even tighter c.i. based on *Kullback–Leibler divergence*

$$d(p, q) = p \log \frac{p}{q} + (1 - p) \log \frac{1 - p}{1 - q}$$

Improvements: KL-UCB

Idea: use even tighter c.i. based on *Kullback–Leibler divergence*

$$d(p, q) = p \log \frac{p}{q} + (1-p) \log \frac{1-p}{1-q}$$

Algorithm: Compute the *score* of each arm i (convex optimization)

$$B_{i,t} = \max \left\{ q \in [0,1] : T_{i,t} d\left(\hat{\mu}_{i,T_{i,t}}, q\right) \leq \log(t) + c \log(\log(t)) \right\}$$

Improvements: KL-UCB

Idea: use even tighter c.i. based on *Kullback–Leibler divergence*

$$d(p, q) = p \log \frac{p}{q} + (1 - p) \log \frac{1 - p}{1 - q}$$

Algorithm: Compute the *score* of each arm i (convex optimization)

$$B_{i,t} = \max \left\{ q \in [0, 1] : T_{i,t} d\left(\hat{\mu}_{i,T_{i,t}}, q\right) \leq \log(t) + c \log(\log(t)) \right\}$$

Regret: pulls to suboptimal arms

$$\mathbb{E}\left[T_{i,n}\right] \leq (1 + \epsilon) \frac{\log(n)}{d(\mu_i, \mu^*)} + C_1 \log(\log(n)) + \frac{C_2(\epsilon)}{n^{\beta(\epsilon)}}$$

where $d(\mu_i, \mu^*) > 2\Delta_i^2$

Improvements: Thompson strategy

Idea: Use a Bayesian approach to estimate the means $\{\mu_i\}_i$

Improvements: Thompson strategy

Idea: Use a Bayesian approach to estimate the means $\{\mu_i\}_i$

Algorithm: Assuming Bernoulli arms and a *Beta* prior on the mean

- Compute
$$\mathcal{D}_{i,t} = \text{Beta}(S_{i,t} + 1, F_{i,t} + 1)$$
- Draw a mean sample as
$$\widetilde{\mu}_{i,t} \sim \mathcal{D}_{i,t}$$
- Pull arm
$$I_t = \arg\max \widetilde{\mu}_{i,t}$$
- If $X_{I_t,t} = 1$ update $S_{I_t,t+1} = S_{I_t,t} + 1$, else update $F_{I_t,t+1} = F_{I_t,t} + 1$

Regret:
$$\lim_{n\to\infty} \frac{R_n}{\log(n)} = \sum_{i=1}^{K} \frac{\Delta_i}{d(\mu_i, \mu^*)}$$

How to *efficiently* explore an MDP

The Exploration-Exploitation Dilemma

Multi-Armed Bandit

Contextual Linear Bandit

Reinforcement Learning

The Contextual Linear Bandit Problem

Motivating Examples

- Different users may have different preferences
- The set of available news may change over time
- We want to minimise the regret w.r.t. the best news for each user

The Contextual Linear Bandit Problem

The problem: at each time $t = 1, \ldots, n$

- User u_t arrives and a set of news $\mathcal{A}_t$ is provided
- The user u_t together with a news $a \in \mathcal{A}_t$ are described by a feature vector $x_{t,a}$
- The learner chooses a news a_t and receives a reward r_{t,a_t}

The optimal news: at each time $t = 1, \ldots, n$, the optimal news is

$$a_t^* = \arg\max_{a \in \mathcal{A}_t} \mathbb{E}[r_{t,a}]$$

The regret:

$$R_n = \mathbb{E}\Big[\sum_{t=1}^{n} r_{t,a_t^*}\Big] - \mathbb{E}\Big[\sum_{t=1}^{n} r_{t,a_t}\Big]$$

The Contextual Linear Bandit Problem

The linear assumption: the reward is a linear combination between the context and an unknown parameter vector

$$\mathbb{E}[r_{t,a}|x_{t,a}] = x_{t,a}^\top \theta_a$$

The Contextual Linear Bandit Problem

The linear regression estimate:

- $\mathcal{T}_a = \{t : a_t = a\}$
- Construct the design matrix of all the contexts observed when action a has been taken $D_a \in \mathbb{R}^{|\mathcal{T}_a| \times d}$
- Construct the reward vector of all the rewards observed when action a has been taken $c_a \in \mathbb{R}^{|\mathcal{T}_a|}$
- Estimate θ_a as

$$\hat{\theta}_a = (D_a^\top D_a + I)^{-1} D_a^\top c_a$$

The Contextual Linear Bandit Problem

Optimism in face of uncertainty: the LinUCB algorithm

- Chernoff-Hoeffding in this case becomes

$$\left|x_{t,a}^\top \hat{\theta}_a - \mathbb{E}[r_{t,a}|x_{t,a}]\right| \leq \alpha \sqrt{x_{t,a}^\top (D_a^\top D_a + I)^{-1} x_{t,a}}$$

- and the UCB strategy is

$$a_t = \arg\max_{a \in \mathcal{A}_t} x_{t,a}^\top \hat{\theta}_a + \alpha \sqrt{x_{t,a}^\top (D_a^\top D_a + I)^{-1} x_{t,a}}$$

The Contextual Linear Bandit Problem

The evaluation problem

- Online evaluation: too expensive
- Offline evaluation: how to use the logged data?

The Contextual Linear Bandit Problem

Evaluation from logged data

- Assumption 1: contexts and rewards are i.i.d. from a stationary distribution

$$(x_1, \ldots, x_K, r_1, \ldots, r_K) \sim D$$

- Assumption 2: the logging strategy is random

The Contextual Linear Bandit Problem

Evaluation from logged data: given a bandit strategy π, a desired number of samples T, and a (infinite) stream of data

Algorithm 3 Policy_Evaluator.

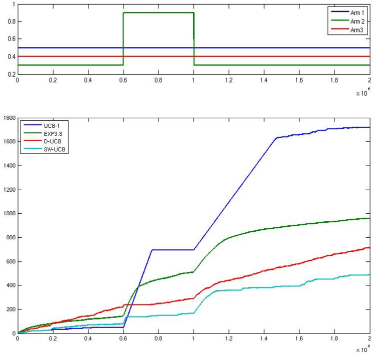
```
Inputs: T > 0; policy π; stream of events
h_0 ← ∅ {An initially empty history}
R_0 ← 0 {An initially zero total payoff}
for t = 1, 2, 3, ..., T do
    repeat
        Get next event (x_1, ..., x_K, a, r_a)
    until π(h_{t−1}, (x_1, ..., x_K)) = a
    h_t ← CONCATENATE(h_{t−1}, (x_1, ..., x_K, a, r_a))
    R_t ← R_{t−1} + r_a
end for
Output: R_T / T
```

Outline

Outline

Non-stationary Bandits

- Changepoint : reward distributions change *abruptly*
- Goal : *follow the best arm*
- Application : scanning tunnelling microscope



- Variants D-UCB et SW-UCB including a progressive *discount* of the past
- Bounds $O(\sqrt{n \log n})$ are proved, which is (almost) optimal

Generalized Linear Bandits

- Bandit with contextual information:

$$\mathbb{E}[X_t | I_t] = \mu(m'_{I_t}\theta_*)$$

where $\theta_* \in \mathbb{R}^d$ is an unkown parameter and $\mu : \mathbb{R} \to \mathbb{R}$ is a link function

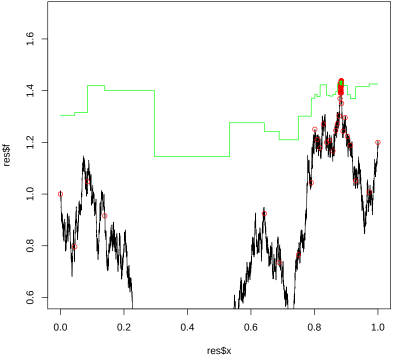
- Example : binary rewards

$$\mu(x) = \frac{\exp(x)}{1 + \exp(x)}$$

- Application : targeted web ads
- GLM-UCB : regret bound depending on dimension d and not on the number of arms

Stochastic Optimization

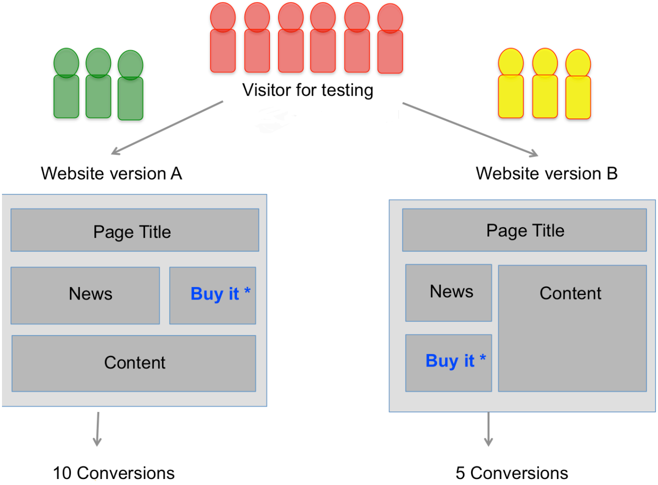
- Goal : Find the maximum of a function $f : C \subset \mathbb{R}^d \to \mathbb{R}$ (possibly) observed in noise
- Application : DAS



- Model : f is the realization of a Gaussian Process (or has a small norm in some RKHS)
- GP-UCB : evaluate f at the point $x \in C$ where the confidence interval for $f(x)$ has the highest upper-bound

Outline

Motivation


Visitor for testing
Website version A
Page Title
News
Buy it *
Content
10 Conversions
Website version B
Page Title
News
Content
Buy it *
5 Conversions

Goal $\neq$ regret minimization

Improve performance:

- fixed number of test users $->$ smaller probability of error
- fixed probability of error $->$ fewer test users

Tools: sequential allocation and stopping

The model

A two-armed bandit model is

- a set $\nu = (\nu_1, \nu_2)$ of two probability distributions ('arms') with respective means μ_1 and μ_2
- $a^* = \text{argmax}_a \, \mu_a$ is the (unknown) best am

To find the best arm, an agent interacts with the bandit model with

- a *sampling rule* $(A_t)_{t \in \mathbb{N}}$ where $A_t \in \{1, 2\}$ is the arm chosen at time t (based on past observations) $- >$ a sample $Z_t \sim \nu_{A_t}$ is observed
- a *stopping rule* τ indicating when he stops sampling the arms
- a *recommendation rule* $\hat{a}_\tau \in \{1, 2\}$ indicating which arm he thinks is best (at the end of the interaction)

In classical A/B Testing, the sampling rule A_t is uniform on $\{1, 2\}$ and the stopping rule $\tau = t$ is fixed in advance.

Two possible goals

The agent's goal is to design a strategy $\mathcal{A} = ((A_t), \tau, \hat{a}_\tau)$ satisfying

Fixed-budget setting	Fixed-confidence setting
$\tau = t$	$\mathbb{P}_\nu(\hat{a}_\tau \neq a^*) \leq \delta$
$p_t(\nu) := \mathbb{P}_\nu(\hat{a}_t \neq a^*)$ as small as possible	$\mathbb{E}_\nu[\tau]$ as small as possible

An algorithm using uniform sampling is

Fixed-budget setting	Fixed-confidence setting
a classical test of $(\mu_1 > \mu_2)$ against $(\mu_1 < \mu_2)$ based on t samples	a sequential test of $(\mu_1 > \mu_2)$ against $(\mu_1 < \mu_2)$ with probability of error uniformly bounded by δ

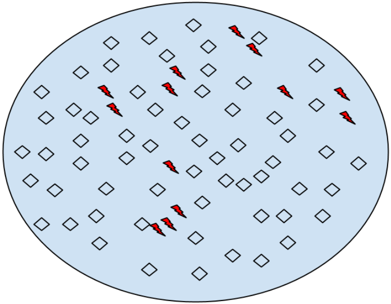
[Siegmund 85]: sequential tests can save samples !

Outline

The model

Optimal Discovery with Probabilistic Expert Advice: Finite Time Analysis and Macroscopic Optimality, JMLR 2013
joint work with S. Bubeck and D. Ernst

- Subset $A \subset \mathcal{X}$ of important items
- $|\mathcal{X}| \gg 1$, $|A| \ll |\mathcal{X}|$
- Access to $\mathcal{X}$ only by probabilistic experts $(P_i)_{1 \leq i \leq K}$: sequential independent draws

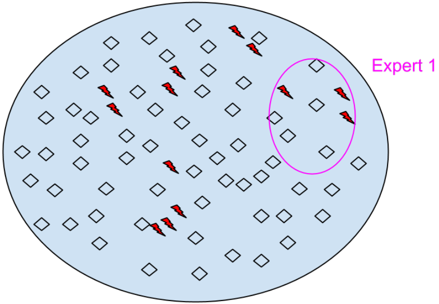


: discover rapidly the elements of A

The model

Optimal Discovery with Probabilistic Expert Advice: Finite Time Analysis and Macroscopic Optimality, JMLR 2013
joint work with S. Bubeck and D. Ernst

- Subset $A \subset \mathcal{X}$ of important items
- $|\mathcal{X}| \gg 1$, $|A| \ll |\mathcal{X}|$
- Access to $\mathcal{X}$ only by probabilistic experts $(P_i)_{1 \leq i \leq K}$: sequential independent draws

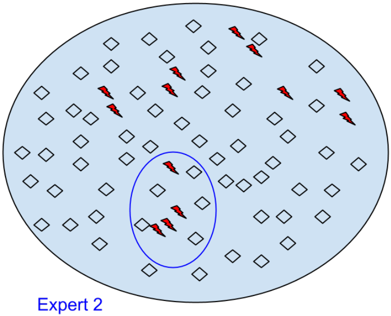


: discover rapidly the elements of A

The model

Optimal Discovery with Probabilistic Expert Advice: Finite Time Analysis and Macroscopic Optimality, JMLR 2013
joint work with S. Bubeck and D. Ernst

- Subset $A \subset \mathcal{X}$ of important items
- $|\mathcal{X}| \gg 1$, $|A| \ll |\mathcal{X}|$
- Access to $\mathcal{X}$ only by probabilistic experts $(P_i)_{1 \leq i \leq K}$: sequential independent draws

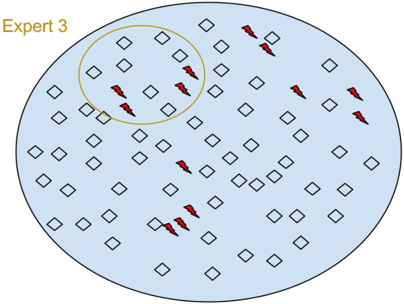


: discover rapidly the elements of A

The model

Optimal Discovery with Probabilistic Expert Advice: Finite Time Analysis and Macroscopic Optimality, JMLR 2013
joint work with S. Bubeck and D. Ernst

- Subset $A \subset \mathcal{X}$ of important items
- $|\mathcal{X}| \gg 1$, $|A| \ll |\mathcal{X}|$
- Access to $\mathcal{X}$ only by probabilistic experts $(P_i)_{1 \leq i \leq K}$: sequential independent draws



: discover rapidly the elements of A

Optimal Exploration with Probabilistic Expert Advice

Search space : $A \subset \Omega$ discrete set

Probabilistic experts : $P_i \in \mathcal{M}_1(\Omega)$ for $i \in \{1, \ldots, K\}$

Requests : at time t, calling expert I_t yields a realization of $X_t = X_{I_t,t}$ independent with law P_a

Goal : find as many distinct elements of A as possible with few requests :

$$F_n = \text{Card}\left(A \cap \{X_1, \ldots, X_n\}\right)$$

Goal

At each time step $t = 1, 2, \dots$:

- pick an index $I_t = \pi_t\big(I_1, Y_1, \dots, I_{s-1}, Y_{s-1}\big) \in \{1, \dots, K\}$ according to past observations
- observe $Y_t = X_{I_t, n_{I_t,t}} \sim P_{I_t}$, where

$$n_{i,t} = \sum_{s \le t} \mathbb{I}\{I_s = i\}$$

Goal: design the strategy $\pi = (\pi_t)_t$ so as to maximize the number of important items found after t requests

$$F^\pi(t) = \Big| A \cap \big\{ Y_1, \dots, Y_t \big\} \Big|$$

Assumption: non-intersecting supports

$$A \cap \mathrm{supp}(P_i) \cap \mathrm{supp}(P_j) = \emptyset \text{ for } i \neq j$$

Is it a Bandit Problem ?

It looks like a bandit problem. . .

- sequential choices among K options
- want to maximize cumulative rewards
- exploration vs exploitation dilemma

. . . but it is **not a bandit problem** !

- rewards are not i.i.d.
- destructive rewards: no interest to observe twice the same important item
- all strategies eventually equivalent

The oracle strategy

Proposition: Under the non-intersecting support hypothesis, the greedy oracle strategy selecting the expert with highest 'missing mass'

$$I_t^* \in \arg\max_{1 \le i \le K} P_i\left(A \setminus \{Y_1, \dots, Y_t\}\right)$$

is optimal: for every possible strategy π, $\mathbb{E}\big[F^\pi(t)\big] \le \mathbb{E}\big[F^*(t)\big]$.

Remark: the proposition if false if the supports may intersect

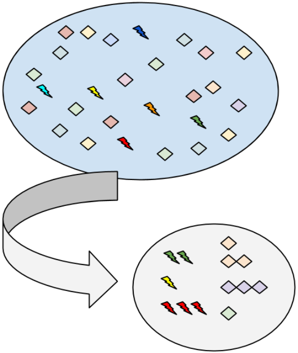
$\Longrightarrow$ **estimate the "missing mass of important items"!**

Missing mass estimation

Let us first focus on one expert i: $P = P_i, X_n = X_{i,n}$

$X_1, \ldots, X_n$ independent draws of P

$$O_n(x) = \sum_{m=1}^{n} \mathbb{I}\{X_m = x\}$$



How to 'estimate' the total mass of the *unseen* important items

$$R_n = \sum_{x \in A} P(x) \mathbb{I}\{O_n(x) = 0\} \ ?$$

The Good-Turing Estimator

Idea: use the **hapaxes** = items seen only once (linguistic)

$$\hat{R}_n = \frac{U_n}{n}, \quad \text{where } U_n = \sum_{x \in A} \mathbb{I}\{O_n(x) = 1\}$$

Lemma [Good '53]: For *every* distribution P,

$$0 \leq \mathbb{E}\big[\hat{R}_n\big] - \mathbb{E}\big[R_n\big] \leq \frac{1}{n}$$

Proposition: With probability at least $1 - \delta$ for *every* P,

$$\hat{R}_n - \frac{1}{n} - (1+\sqrt{2})\sqrt{\frac{\log(4/\delta)}{n}} \leq R_n \leq \hat{R}_n + (1+\sqrt{2})\sqrt{\frac{\log(4/\delta)}{n}}$$

See [McAllester and Schapire '00, McAllester and Ortiz '03]:

- deviations of $\hat{R}_n$: McDiarmid's inequality
- deviations of R_n: negative association

The Good-UCB algorithm [Bubeck, Ernst & G.]

Optimistic algorithm based on Good-Turing's estimator :

$$I_{t+1} = \underset{i \in \{1,\dots,K\}}{\arg\max} \left\{ \frac{H_i(t)}{N_i(t)} + c\sqrt{\frac{\log(t)}{N_i(t)}} \right\}$$

- $N_i(t)$ = number of draws of P_i up to time t
- $H_i(t)$ = number of elements of A seen exactly once thanks to P_i
- c = tuning parameter

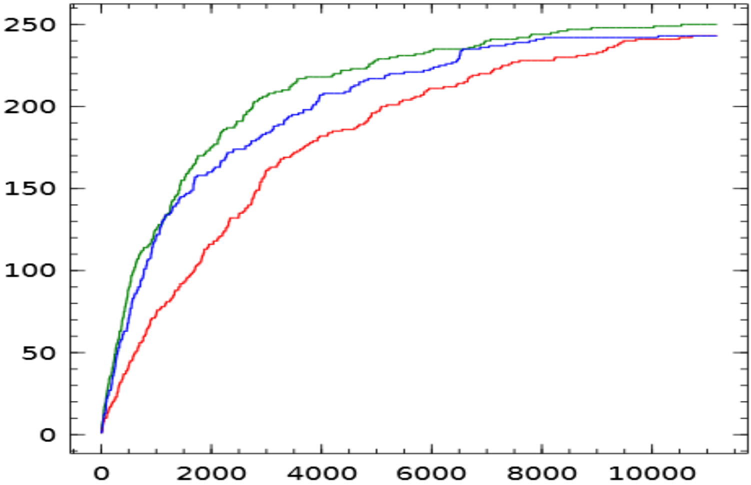
Classical analysis

Theorem: For any $t \geq 1$, under the non-intersecting support assumption, Good-UCB (with constant $C = (1+\sqrt{2})\sqrt{3}$) satisfies

$$\mathbb{E}\left[F^*(t) - F^{UCB}(t)\right] \leq 17\sqrt{Kt\log(t)} + 20\sqrt{Kt} + K + K\log(t/K)$$

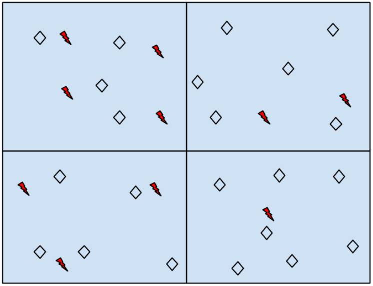
Remark: Usual result for bandit problem, but not-so-simple analysis

A Typical Run of Good-UCB



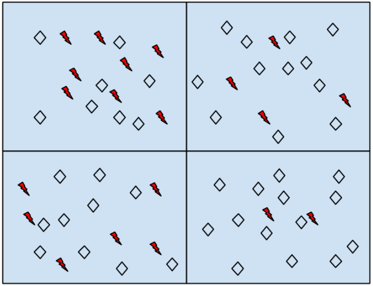
The macroscopic limit

- Restricted framework: $P_i = \mathcal{U}\{1, \ldots, N\}$
- $N \to \infty$
- $|A \cap \mathrm{supp}(P_i)|/N \to q_i \in (0, 1)$, $q = \sum_i q_i$



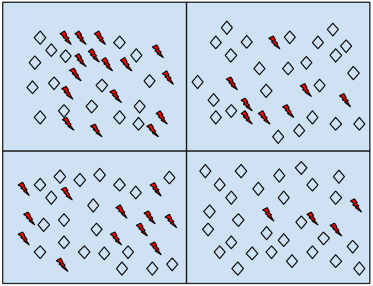
The macroscopic limit

- Restricted framework: $P_i = \mathcal{U}\{1, \dots, N\}$
- $N \to \infty$
- $|A \cap \text{supp}(P_i)|/N \to q_i \in (0, 1)$, $q = \sum_i q_i$



The macroscopic limit

- Restricted framework: $P_i = \mathcal{U}\{1, \ldots, N\}$
- $N \to \infty$
- $|A \cap \text{supp}(P_i)|/N \to q_i \in (0, 1)$, $q = \sum_i q_i$



The Oracle behaviour

The limiting discovery process of the Oracle strategy is *deterministic*

Proposition: For every $\lambda \in (0, q_1)$, for every sequence $(\lambda^N)_N$ converging to λ as N goes to infinity, almost surely

$$\lim_{N \to \infty} \frac{T_*^N(\lambda^N)}{N} = \sum_i \left(\log \frac{q_i}{\lambda} \right)_+$$

Oracle vs. uniform sampling

Oracle: The proportion of important items not found after Nt draws tends to

$$q - F^*(t) = I(t)\underline{q}_{I(t)} \exp\left(-t/I(t)\right) \leq K \underline{q}_K \exp(-t/K)$$

with $\underline{q}_K = \left(\prod_{i=1}^K q_i\right)^{1/K}$ the geometric mean of the $(q_i)_i$.

Uniform: The proportion of important items not found after Nt draws tends to $K \bar{q}_K \exp(-t/K)$

$\Longrightarrow$ Asymptotic ratio of efficiency

$$\rho(q) = \frac{\bar{q}_K}{\underline{q}_K} = \frac{\frac{1}{K}\sum_{i=1}^k q_i}{\left(\prod_{i=1}^k q_i\right)^{1/K}} \geq 1$$

larger if the $(q_i)_i$ are unbalanced

Macroscopic optimality

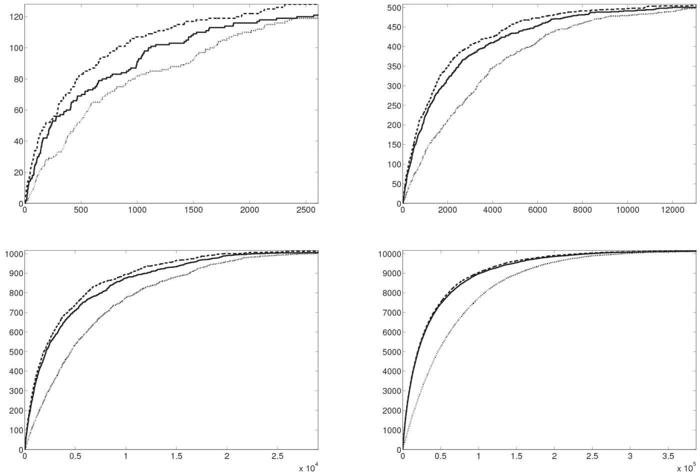
Theorem: Take $C = (1+\sqrt{2})\sqrt{c+2}$ with $c > 3/2$ in the Good-UCB algorithm.

- For every sequence $(\lambda^N)_N$ converging to λ as N goes to infinity, almost surely

$$\limsup_{N\to+\infty} \frac{T^N_{UCB}(\lambda^N)}{N} \leq \sum_i \left(\log \frac{q_i}{\lambda}\right)_+$$

- The proportion of items found after Nt steps F^{GUCB} converges *uniformly* to F^* as N goes to infinity

Simulation



Number of items found by Good-UCB (line), the oracle (bold dashed), and by uniform sampling (light dotted) as a function of time, for sample sizes $N = 128$, $N = 500$, $N = 1000$ and $N = 10000$, in an environment with 7 experts.

Bibliography I

R. E. Bellman.
Dynamic Programming.
Princeton University Press, Princeton, N.J., 1957.

D.P. Bertsekas and J. Tsitsiklis.
Neuro-Dynamic Programming.
Athena Scientific, Belmont, MA, 1996.

W. Fleming and R. Rishel.
Deterministic and stochastic optimal control.
Applications of Mathematics, 1, Springer-Verlag, Berlin New York, 1975.

R. A. Howard.
Dynamic Programming and Markov Processes.
MIT Press, Cambridge, MA, 1960.

M.L. Puterman.
Markov Decision Processes: Discrete Stochastic Dynamic Programming.
John Wiley & Sons, Inc., New York, Etats-Unis, 1994.

Reinforcement Learning

Alessandro Lazaric
alessandro.lazaric@inria.fr

sequel.lille.inria.fr

dubbing: *Aurélien Garivier*
aurelien.garivier@math.univ-toulouse.fr