

# Reinforcement Learning and Decision Making Under Uncertainty

Christos Dimitrakakis

February 20, 2024

## Contents

<b>1</b>	<b>General course information</b>	<b>3</b>
1.1	Prerequisites . . . . .	3
1.1.1	Essential . . . . .	3
1.1.2	Recommended . . . . .	4
1.2	Examination . . . . .	4
1.2.1	A . . . . .	4
1.2.2	B . . . . .	5
1.2.3	C . . . . .	5
1.2.4	D . . . . .	6
1.2.5	E . . . . .	6
1.2.6	F . . . . .	7
<b>2</b>	<b>Schedule</b>	<b>7</b>
<b>3</b>	<b>Modules</b>	<b>8</b>
3.1	Introduction . . . . .	8
3.2	Prerequisites . . . . .	8
3.2.1	Mathematics . . . . .	8
3.2.2	Programming . . . . .	8
3.3	Course Books . . . . .	9
3.4	Beliefs and decisions . . . . .	9
3.4.1	Utility theory (90') . . . . .	9
3.4.2	Probability primer . . . . .	10
3.4.3	Lab: Probability, Expectation, Utility . . . . .	11
3.4.4	Assignment. . . . .	11
3.4.5	Further Reading: . . . . .	12

3.4.6	Seminar: . . . . .	12
3.5	Decisions with observations . . . . .	12
3.5.1	Problems with Observations (45') . . . . .	12
3.5.2	Statistical decisions (45') . . . . .	13
3.5.3	Lab: Decision problems and estimation (45') . . . . .	13
3.5.4	Assignment: James Randi . . . . .	13
3.6	Bandit problems . . . . .	13
3.6.1	$n$ meteorologists as prediction with expert advice . . .	13
3.6.2	Prediction with expert advice . . . . .	14
3.6.3	The Exponentially Weighted Average . . . . .	14
3.6.4	The $n$ armed stochastic bandit problem . . . . .	15
3.6.5	Bernoulli bandits . . . . .	15
3.6.6	Policies and exploration . . . . .	15
3.6.7	The upper confidence bound . . . . .	16
3.6.8	Beta distributions as beliefs . . . . .	16
3.6.9	Assignment and exercise . . . . .	17
3.7	Markov Decision Processes: Finite horizon . . . . .	17
3.7.1	The Markov decision process . . . . .	17
3.7.2	MDP examples . . . . .	18
3.7.3	Monte Carlo Policy Evaluation . . . . .	18
3.7.4	Policy Evaluation . . . . .	19
3.7.5	Backwards induction . . . . .	19
3.8	Markov Decision Processes: Infinite horizon . . . . .	19
3.8.1	Plan . . . . .	19
3.8.2	Infinite horizon setting . . . . .	20
3.8.3	Value iteration . . . . .	20
3.8.4	Matrix notation for finite MDPs . . . . .	20
3.8.5	Convergence of value iteration . . . . .	21
3.8.6	Policy evaluation . . . . .	21
3.8.7	Optimality equations . . . . .	22
3.8.8	Value iteration convergence proof . . . . .	22
3.8.9	Speed of convergence of value iteration . . . . .	22
3.8.10	Policy Iteration . . . . .	23
3.9	RL: Stochastic Approximation . . . . .	23
3.9.1	Two reinforcement learning setting . . . . .	23
3.9.2	Learning goals . . . . .	24
3.9.3	Monte Carlo Policy Evaluation . . . . .	24
3.9.4	Monte Carlo Updates . . . . .	24
3.9.5	Temporal Differences . . . . .	25
3.9.6	Stochastic state-action value approximation . . . . .	25

3.10	Model-based RL . . . . .	26
3.10.1	Model-Based RL . . . . .	26
3.10.2	Example 1: Model-Based Value Iteration . . . . .	27
3.10.3	Asynchronous Value Iteration . . . . .	27
3.10.4	Greedy actions . . . . .	27
3.10.5	Example 2: Dyna-Q Learning . . . . .	27
3.10.6	Q-iteration . . . . .	27
3.10.7	Greedy actions . . . . .	27
3.10.8	Questions . . . . .	27
3.11	Approximate Dynamic Programming . . . . .	28
3.11.1	RL in continuous spaces . . . . .	28
3.11.2	Approximating a function $f$ . . . . .	28
3.11.3	Fitting a value function to data . . . . .	28
3.11.4	Approximate Value Iteration . . . . .	29
3.11.5	Q-learning with function approximation . . . . .	29
3.12	Policy Gradient . . . . .	29
3.13	Bayesian methods . . . . .	29
3.14	Regret bounds . . . . .	30
3.15	MCTS . . . . .	30
3.16	Advanced Bayesian Models . . . . .	30
3.17	Inverse Reinforcement Learning . . . . .	30
3.18	Multiplayer games . . . . .	30

## 1 General course information

The course will give a thorough introduction to reinforcement learning. The first 8 weeks will be devoted to the core theory and algorithms of reinforcement learning. The final 6 weeks will be focused on project work, during which more advanced topics will be introduced.

The first 6 weeks will require the students to complete 5 assignments. The remainder of the term, the students will have to prepare a project, for which they will need to submit a report.

### 1.1 Prerequisites

#### 1.1.1 Essential

- Mathematics (Calculus, Linear Algebra)
- Python programming.

### 1.1.2 Recommended

- Elementary knowledge of probability and statistics.

## 1.2 Examination

There is one project, taking up 60% of the credit. There is one written exam, for 40% of the credit. Criteria for full marks in each part of the exam are the following.

1. Documenting of the work in a way that enables reproduction.
2. Technical correctness of their analysis.
3. Demonstrating that they have understood the assumptions underlying their analysis.
4. Addressing issues of reproducibility in research.
5. Consulting additional resources beyond the source material with proper citations.

The follow marking guidelines are what one would expect from students attaining each grade.

### 1.2.1 A

1. Submission of a detailed report from which one can definitely reconstruct their work without referring to their code. There should be no ambiguities in the described methodology. Well-documented code where design decisions are explained.
2. Extensive analysis and discussion. Technical correctness of their analysis. Nearly error-free implementation.
3. The report should detail what models are used and what the assumptions are behind them. The conclusions of the should include appropriate caveats. When the problem includes simple decision making, the optimality metric should be well-defined and justified. Simiarly, when well-defined optimality criteria should given for the experiment design, when necessary. The design should be (to some degree of approximation, depending on problem complexity) optimal according to this criteria.

4. Appropriate methods to measure reproducibility. Use of an unbiased methodology for algorithm, model or parameter selection. Appropriate reporting of a confidence level (e.g. using bootstrapping) in their analytical results. Relevant assumptions are mentioned when required.
5. The report contains some independent thinking, or includes additional resources beyond the source material with proper citations. The students go beyond their way to research material and implement methods not discussed in the course.

### **1.2.2 B**

1. Submission of a report from which one can plausibly reconstruct their work without referring to their code. There should be no major ambiguities in the described methodology.
2. Technical correctness of their analysis, with a good discussion. Possibly minor errors in the implementation.
3. The report should detail what models are used, as well as the optimality criteria, including for the experiment design. The conclusions of the report must contain appropriate caveats.
4. Use of an unbiased methodology for algorithm, model or parameter selection.
5. The report contains some independent thinking, or the students mention other methods beyond the source material, with proper citations, but do not further investigate them.

### **1.2.3 C**

1. Submission of a report from which one can partially reconstruct most of their work without referring to their code. There might be some ambiguities in parts of the described methodology.
2. Technical correctness of their analysis, with an adequate discussion. Some errors in a part of the implementation.
3. The report should detail what models are used, as well as the optimality criteria and the choice of experiment design. Analysis caveats are not included.

4. Use of a possibly biased methodology for algorithm, model or parameter selection - but in a possibly inconsistent manner.
5. There is little mention of methods beyond the source material or independent thinking.

#### **1.2.4 D**

1. Submission of a report from which one can partially reconstruct most of their work without referring to their code. There might be serious ambiguities in parts of the described methodology.
2. Technical correctness of their analysis with limited discussion. Possibly major errors in a part of the implementation.
3. The report should detail what models are used, as well as the optimality criteria. Analysis caveats are not included.
4. Some effort for methodological algorithm/parameter selection.
5. There is little mention of methods beyond the source material or independent thinking.

#### **1.2.5 E**

1. Submission of a report from which one can obtain a high-level idea of their work without referring to their code. There might be serious ambiguities in all of the described methodology.
2. Technical correctness of their analysis with very little discussion. Possibly major errors in only a part of the implementation.
3. The report might mention what models are used or the optimality criteria, but not in sufficient detail and caveats are not mentioned.
4. Reproducibility is only partially addressed.
5. There is no mention of methods beyond the source material or independent thinking.

### 1.2.6 F

1. The report does not adequately explain their work.
2. There is very little discussion and major parts of the analysis are technically incorrect, or there are errors in the implementation.
3. The models used might be mentioned, but not any other details.
4. There is no effort to ensure reproducibility or robustness in the project.
5. There is no mention of methods beyond the source material or independent thinking.

cd

## 2 Schedule

Week	Topic
1	Beliefs and Decisions
2	Bayesian Decision Rules
3	Introduction to Bandit problems.
4	Finite Horizon MDPs Backwards Induction The Bandit MDP
5	Infinite Horizon MDPs Value Iteration Policy Iteration
6	Sarsa / Q-Learning
7	Model-Based RL
8	Function Approximation, Gradient Methods
9	Function Approximation Lab
10	Bayesian RL: Dynamic Programming, Sampling
11	UCB/UCRL/UCT. UCT/AlphaZero.
12	Project Lab
13	Project presentations
14	Q&A, Mock exam

## 3 Modules

### 3.1 Introduction

Reinforcement learning is the problem of learning to act through interaction with an unknown environment. It is not:

- A solution.
- Supervised learning
- Unsupervised learning.

However, algorithms for reinforcement learning can use (un)supervised learning algorithms as components.

Uncertainty and sequential decision making are central in reinforcement learning.

### 3.2 Prerequisites

No previous machine learning knowledge is needed.

#### 3.2.1 Mathematics

The following topics must be absolutely mastered, although a refresher will be given as needed.

1. Set theory and logic.
2. Probability and expectation.
3. Elementary calculus (limits, integration, differentiation)
4. Elementary linear algebra (vector and matrix manipulations)

#### 3.2.2 Programming

- Mature programming ability, preferably in python.
- Use of git or other version control system
- Use of (La)TeX.



### 3.3 Course Books

- **Course book** *Decision Making Under Uncertainty and Reinforcement Learning*, Dimitrakakis and Ortner
- **Statistical reference** *Optimal Statistical Decisions*, De Groot.
- **MDP Reference** *Markov Decision Processes*, Puterman.
- **Basic RL Reference** *Reinforcement Learning: An Introduction*, Sutton and Barto.
- **Advanced RL Reference** *Neurodynamic Programming*, Bertsekas and Tsitsiklis.

### 3.4 Beliefs and decisions

#### 3.4.1 Utility theory (90')

1. Rewards and preferences (15')
2. Transitivity of preferences (15')
3. Random rewards (5')
4. Decision Diagrams (10')
5. Utility functions and the expected utility hypothesis (15')
6. Utility exercise: Gambling (10' pen and paper)
7. The St. Petersburg Paradox (15')
8. Preferences

We assume that, given a choice between items in a set of possible rewards  $R$ , we have a complete preference order, meaning that, for any  $a, b \in R$ , we either:

- (I) Prefer  $a$  to  $b$ , and write  $a \succ^* b$  (II) Prefer  $b$  to  $a$ , and write  $a \prec^* b$   
(III) We are indifferent between  $a$  and  $b$ , and write  $a \approx^* b$

1. Transitivity

The above assumptions do not preclude cycles. However, we can also assume that:

If  $a \succ^* b$  and  $b \succ^* c$  then  $a \succ^* c$ .

1. Random rewards.

Now consider the case where, instead of directly choosing rewards, we make a choice, and then obtain a **random** reward. Here, the reward depends in some way in our decision, but we are not sure exactly how.

Examples:

- Choosing between taking the train and a car.
- Gambling in a casino.
- Deciding how much to study for the exam.

### 3.4.2 Probability primer

1. Objective vs Subjective Probability: Example (5')
2. Relative likelihood: Completeness, Consistency, Transitivity, Complement, Subset (5')
3. Measure theory (5')
4. Axioms of Probability (5')
5. Random variables (5')
6. Expectations (5')
7. Expectations exercise (10')
8. Objective vs Subjective probability
9. Quantum Physics: There is real underlying randomness. The probabilities of all possible outcomes can be computed exactly *a priori*
10. Coin toss: We model our uncertainty about the outcome through randomness. However, the coin is not really random, and we must *experiment* to determine the proportion of each possible outcome. We simply lack the information to compute the probabilities *a priori*.

Everything that can possibly happen is contained in the universe of events  $\Omega$ .

Events  $A, B \subset \Omega$  are subsets of the universe. This can be visualised in the coin tosses example.

### 1. Relative Likelihood

$A, B$  are possible events, satisfying these properties:

- (I) Completeness:  $A \succ B$ ,  $A \prec B$  or  $A \approx B$  for any  $A, B$  (II) Transitivity: If  $A \succ B, B \succ C$  then  $A \succ C$  (III) Complement: If  $A \succ B$  then  $\neg A \prec \neg B$ . (IV) Implication:  $A \subset B \Rightarrow A \prec B$

### 1. Measure theory

We can use probability to quantify this, so that  $A > B$  iff  $P(A) > P(B)$ . But what do we mean by this?

Measure as a concept: area, length, probability  $M(A) + M(B) = M(A \cup B)$

### 1. Axioms of Probability

$P : \Sigma \rightarrow [0, 1]$   $P(\emptyset) = 0$   $P(\Omega) = 1$  If  $A \cap B = \emptyset$ ,  $P(A \cup B) = P(A) + P(B)$ .

- Exercise: Prove that  $P$  satisfies the given properties of relative likelihood.

### 2. Random variables

If  $\omega$  is distributed according to  $P$ , then the function  $f(\omega)$ , with  $f : \Omega \rightarrow \mathbb{R}$ , is a *random variable* with distribution  $P_f$ , where:

$$P_f(A) = P(\{\omega : f(\omega) \in A\})$$

### 1. Expectations

$$E_P[f] = \sum_{\omega} f(\omega)P(\omega).$$

### 3.4.3 Lab: Probability, Expectation, Utility

- Exercise Set 1. Probability introduction.
- Exercise Set 2. Sec 2.4, Exercises 4, 5.

### 3.4.4 Assignment.

Exercise 7, 8, 9.

### 3.4.5 Further Reading:

Decision Making Under Uncertainty and Reinforcement Learning. Chapter 1, 2.

### 3.4.6 Seminar:

Utility. What is the concept of utility? Why do we want to always maximise utility?

Example:

U	w1	w2
a1	4	1
a2	3	3

Regret. Alternative notion.

L	w1	w2
a1	0	2
a2	1	0

Minimising regret is the same as maximising utility when  $w$  does not depend on  $a$ . Hint: So that if  $E[L|a^*] \leq E[L|a]$  for all  $a'$ ,  $E[U|a^*] \geq E[U|a]$  for all  $a'$ ,

The utility analysis of choices involving risk: <https://www.journals.uchicago.edu/doi/abs/10.1086/256692>

The expected-utility hypothesis and the measurability of utility <https://www.journals.uchicago.edu/doi/abs/10.1086/257308>

## 3.5 Decisions with observations

### 3.5.1 Problems with Observations (45')

1. Discrete set of models example: the meteorologists problem (25')
2. Marginal probabilities (5').
3. Conditional probability (5').
4. Bayes theorem (10').

### 3.5.2 Statistical decisions (45')

1. ML Estimation (10')
2. MAP Estimation (10')
3. Bayes Estimation (10')
4. MSE Estimation (10') [not done]
5. Linearity of Expectations (10') [not done]
6. Convexity of Bayes Decisions (10') [not done]

### 3.5.3 Lab: Decision problems and estimation (45')

1. Problems with no observations. Book Exercise: 13,14,15.
2. Problems with observations. Book Exercise: 17, 18.

### 3.5.4 Assignment: James Randi

## 3.6 Bandit problems

### 3.6.1 $n$ meteorologists as prediction with expert advice

- Predictions  $p_t = p_{t,1}, \dots, p_{t,n}$  of all models for outcomes  $y_t$
- Make decision  $a_t$ .
- Observe true outcome  $y_t$
- Obtain instant reward  $r_t = \rho(a_t, y_t)$
- Utility  $U = \sum_{t=1}^T r_t$ .
- $T$  is the problem **horizon**

1. At each step  $t$ :

- (a) Observe  $p_t$ .
- (b) Calculate  $\hat{p}_t = \sum_{\mu} \xi_t(\mu) p_{t,\mu}$
- (c) Make decision  $a_t = \arg \max_a \sum_y \hat{p}_t(y) \rho(a, y)$ .
- (d) Observe  $y_t$  and obtain reward  $r_t = \rho(a_t, y_t)$ .
- (e) Update:  $\xi_{t+1}(\mu) \propto \xi_t(\mu) p_{t,\mu}(y_t)$ .

The update **does not depend** on  $a_t$

### 3.6.2 Prediction with expert advice

- Advice  $p_t = p_{t,1}, \dots, p_{t,n} \in D$
- Make prediction  $\hat{p}_t \in D$
- Observe true outcome  $y_t \in Y$
- Obtain instant reward  $r_t = u(\hat{p}_t, y_t)$
- Utility  $U = \sum_{t=1}^T r_t$ .

#### 1. Relation to $n$ meteorologists

- $D$  is the set of distributions on  $Y$ .
- However, there are only predictions, no actions. To add actions:

$$u(\hat{p}_t, y_t) = \rho(a^*(\hat{p}_t), y_t), \quad a^*(\hat{p}_t) = \arg \max_a \rho(a, y_t)$$

The update **does not depend** on  $a_t$

### 3.6.3 The Exponentially Weighted Average

#### 1. MWA Algorithm

- Predict by averaging all of the predictions:

$$\hat{p}_t(y) = \sum_{\mu} \xi_t(\mu) p_{t,\mu}(y)$$

- Update by weighting the quality of each prediction

$$\xi_{t+1}(\mu) = \frac{\xi_t(\mu) \exp[\eta u(p_{t,\mu}, y_t)]}{\sum_{\mu'} \xi_t(\mu') \exp[\eta u(p_{t,\mu'}, y_t)]}$$

#### 2. Choices for $u$

- $u(p_{t,\mu}, y_t) = \ln p_{t,\mu}(y_t)$ ,  $\eta = 1$ , Bayes's theorem.
- $u(p_{t,\mu}, y_t) = \rho(a^*(p_{t,\mu}), y_t)$ : quality of expert prediction.

### 3.6.4 The $n$ armed stochastic bandit problem

- Take action  $a_t$
- Obtain reward  $r_t \sim P_{a_t}(r)$  with expected value  $\mu_{a_t}$ .
- The utility is  $U = \sum_t r_t$ , while  $P$  is **unknown**.

1. The Regret -Total regret with respect to the best arm:

$$L \triangleq \sum_{t=1}^T [\mu^* - r_t], \quad \mu^* = \max_a \mu_a$$

- Expected regret of an algorithm  $\pi$ :

$$\mathbb{E}^\pi[L] = \sum_{t=1}^T \mathbb{E}^\pi[\mu^* - r_t] = \sum_{a=1}^n \mathbb{E}^\pi[n_{T,a}](\mu^* - \mu_a)$$

- $n_{T,a}$  is the number of times  $a$  has been pulled after  $n$  steps.

### 3.6.5 Bernoulli bandits

A classical example of this is when the rewards are Bernoulli, i.e.

$$r_t | a_t = i \sim \text{Bernoulli}(\mu_i)$$

1. Greedy algorithm

- Take action  $a_t = \arg \max_a \hat{\mu}_{t,a}$
- Obtain reward  $r_t \sim P_{a_t}(r)$  with expected value  $\mu_{a_t}$ .
- Update arm:  $s_{t,a_t} = s_{t-1,a_t} + r_t$ ,  $n_{t,a_t} = n_{t-1,a_t} + 1$ .
- Others stay the same:  $s_{t,a} = s_{t-1,a}$ ,  $n_{t,a} = n_{t-1,a}$  for  $a \neq a_t$ .
- Update means:  $\hat{\mu}_{t,i} = s_{t,i}/n_{t,i}$ .

### 3.6.6 Policies and exploration

- $n_{t,i}$ ,  $s_{t,i}$  are **sufficient statistics** for Bernoulli bandits.
- The more often we pull an arm, the more certain we are the mean is correct.

1. Upper confidence bound: exploration bonuses

- Take action  $a_t = \arg \max_a \hat{\mu}_{t,a} + O(1/\sqrt{n_{t,a}})$ .

## 2. Posterior sampling: randomisation

- Given some prior parameters  $\alpha, \beta > 0$  (e.g. 1).
- $\xi_t(\mu_a) = \text{Beta}(\alpha + s_{t,a}, \beta + n_{t,a} - s_{t,a})$ .
- Sample  $\hat{\mu} \sim \xi_t(\mu)$ .
- Take action  $a_t = \arg \max_a \hat{\mu}_a$ .

### 3.6.7 The upper confidence bound

Let

$$\hat{\mu}_n = \sum_{i=1}^t r_i / n,$$

be the sample mean estimate of an iid RV in  $[0,1]$  with  $\mathbb{E}[r_i] = \mu$ . Then we have

$$\mathbb{P}(\hat{\mu}_n \geq \mu + \epsilon) \leq \exp(-2n\epsilon^2)$$

or equivalently

$$\mathbb{P}(\hat{\mu}_n \geq \mu_n + \sqrt{\ln(1/\delta)/2n} \leq \delta.)$$

### 3.6.8 Beta distributions as beliefs

- [Go through Chapter 4, Beta distribution]
- [Visualise Beta distribution]
- [Do the James Random Exercise 3]
- Note that the problem here is that this is only a point estimate: it ignores uncertainty. In fact, we can represent our uncertainty about the arms in a probabilistic way with the Beta distribution:

If our prior over an arm's mean is  $\text{Beta}(\alpha, \beta)$  then the -posterior at time  $t$  is  $\text{Beta}(\alpha + s_{t,i}, \beta + n_{t,i} - s_{t,i})$ .

- [Visualise how the posterior changes for a biased coin as we obtain more data].



### 3.6.9 Assignment and exercise

1. Implement epsilon-greedy bandits (lab, 30')
2. Implement Thompson sampling bandits (lab, 30')
3. Implement UCB bandits (home)
  1. Compare them in a benchmark (home)

## 3.7 Markov Decision Processes: Finite horizon

1. The bandit MDP (30')
2. MDP definitions (15')
3. MDP examples (15')
4. Monte Carlo Policy Evaluation (15')
5. DP: Finite Horizon Policy Evaluation (15')
6. DP: Finite Horizon Backward Induction (15')
7. DP: Proof of Backwards Induction (15')
8. DP: Implementation of Backwards Induction (30')

### 3.7.1 The Markov decision process

1. Interaction at time  $t$ 
  - Observe state  $s_t \in S$
  - Take action  $a_t \in A$ .
  - Obtain reward  $r_t \in \mathbb{R}$ .
2. The MDP model  $\mu$ 
  - Transition kernel  $P_\mu(s_{t+1}|s_t, a_t)$ .
  - Reward with mean  $\rho_\mu(s_t, a_t)$
3. Policies
  - Markov policies  $\pi(a_t|s_t)$

4. Utility Total reward up to a finite (but not necessarily fixed) horizon  $T$

$$U_1 = \sum_{t=1}^T r_t$$

### 3.7.2 MDP examples

1. Shortest path problems

- Goal state  $s^* \in S$ .
- Reward  $r_t = -1$  for all  $s \neq s^*$
- Game ends time  $T$  where  $s_T = s^*$ .

2. Blackjack against a croupier

- Croupier shows one card.
- Current state is croupier's card and your cards.
- Reward is  $r_T = 1$  if you win,  $r_T = -1$  if you lose at the end, otherwise 0.

### 3.7.3 Monte Carlo Policy Evaluation

$$\begin{aligned} V_t^\pi(s) &= \mathbb{E}^\pi[U_t | s_t = s] \\ &\approx \frac{1}{N} \sum_{n=1}^N U_t^{(n)} \end{aligned}$$

### 3.7.4 Policy Evaluation

$$\begin{aligned}
V_t^\pi(s) &= \mathbb{E}^\pi[U_t | s_t = s] \\
&= \mathbb{E}^\pi\left[\sum_{k=t}^T r_k | s_t = s\right] \\
&= \mathbb{E}^\pi[r_t | s_t = s] + \mathbb{E}^\pi\left[\sum_{k=t+1}^T r_k | s_t = s\right] \\
&= \mathbb{E}^\pi[r_t | s_t = s] + \mathbb{E}^\pi[U_{t+1} | s_t = s] \\
&= \mathbb{E}^\pi[r_t | s_t = s] + \sum_{s'} \mathbb{E}^\pi[U_{t+1} | s_{t+1} = s'] \mathbb{P}^\pi(s_{t+1} = s' | s_t = s) \\
&= \mathbb{E}^\pi[r_t | s_t = s] + \sum_{s'} V_{t+1}^\pi(s') \mathbb{P}^\pi(s_{t+1} = s' | s_t = s) \\
&= \mathbb{E}^\pi[r_t | s_t = s] + \sum_{s'} V_{t+1}^\pi(s') \sum_a \mathbb{P}(s_{t+1} = s' | s_t = s, a_t = a) \pi_t(a | s).
\end{aligned}$$

### 3.7.5 Backwards induction

Let  $v_t$  be the estimates of the backwards induction algorithm. We want to prove that  $v_t = V_t^*$ . This is true for  $t = T$ . Let us assume by induction that  $v_{t+1} > V_{t+1}^*$ . Then it must hold for  $t$  as well:

$$\begin{aligned}
v_t(s) &= \max_a r(s) + \sum_j p(j | s, a) v_{t+1}(j) \\
&\geq \max_a r(s) + \sum_j p(j | s, a) V_{t+1}^*(j) \\
&\geq \max_a r(s) + \sum_j p(j | s, a) V_{t+1}^\pi(j) \quad \forall \pi \\
&\geq V_t^\pi(s)
\end{aligned}$$

If  $\pi^*$  is the policy returned by backwards induction, then  $v_t = V^{\pi^*}$ . Consequently

$$V^* \geq V^{\pi^*} = v \geq V^* \Rightarrow v = V^*.$$

## 3.8 Markov Decision Processes: Infinite horizon

### 3.8.1 Plan

1. DP: Value Iteration (45')

2. DP: Policy Iteration (45')

### 3.8.2 Infinite horizon setting

1. Utility

$$U = \sum_{t=0}^{\infty} \gamma^t r_t$$

2. Discount factor  $\gamma \in (0, 1)$  Tells us how much we care about the future.  
Note that

$$\sum_{t=0}^{\infty} \gamma^t = \frac{1}{1 - \gamma}$$

### 3.8.3 Value iteration

Idea: Run backwards induction, discounting by  $\gamma$  until convergence.

1. Algorithm

- Input: MDP  $\mu$ , discount factor  $\gamma$ , threshold  $\epsilon$
- $v_0(s) = \rho_\mu(s)$  for all  $s$
- For  $n = 1, \dots$

$$v_{n+1}(s) = \rho_\mu(s) + \gamma \sum_j P_\mu(j|s, a) v_n(j).$$

- Until  $\|v_{n+1} - v_n\|_\infty \leq \epsilon$ .

2. Norms

- $\|x\|_1 = \sum_t |x_t|$
- $\|x\|_\infty = \max_t |x_t|$
- $\|x\|_p = (\sum_t |x_t|^p)^{1/p}$

### 3.8.4 Matrix notation for finite MDPs

- $r$ : reward vector.
- $P_\pi$ : transition matrix.
- $v$ : value function vector.

1. Stationary policies

$$\pi(a_t|s_t) = \pi(a_k|s_k)$$

2. Matrix formula for value function

$$v^\pi = \sum_{t=0}^{\infty} \gamma^t P_\pi^t r.$$

Note that  $(P_\pi r)(s) = \sum_j P_\pi(s, j)r(j)$ .

### 3.8.5 Convergence of value iteration

1. Proof idea

- (a) Define the VI operator  $L$  so that  $v_{n+1} = Lv_n$ .
- (b) Show that if  $v = V^*$  then  $v = Lv$ .
- (c) Show that  $\lim_{n \rightarrow \infty} v = V^*$ .

2. Further questions

- How fast does it converge?
- When is the policy actually optimal?

### 3.8.6 Policy evaluation

1. Policy evaluation theorem For any stationary policy  $\pi$ , the unique solution of

$$v = r + \gamma P_\pi v \quad \text{is} \quad v^\pi = (I - \gamma P_\pi)^{-1} r$$

2. Proof If  $\|A\| < 1$ , then  $(I - A)^{-1}$  exists and

$$(I - A)^{-1} = \lim_{T \rightarrow \infty} \sum_{t=0}^T A^t.$$

3. Interpretation:  $X = (I - P)^{-1}$  Is the total discounted number of times reaching a state

$$X(i, j) = \mathbb{E} \sum_{t=0}^{\infty} \gamma^t \mathbb{I}\{s_t = j | s_0 = i\}$$

### 3.8.7 Optimality equations

1. Policy operator

$$L_\pi v = r + \gamma P_\pi v.$$

2. Bellman operator

$$Lv = \max_\pi \{r + \gamma P_\pi v\}.$$

3. Bellman optimality equation

$$v = Lv$$

### 3.8.8 Value iteration convergence proof

1. Contraction mappings  $M$  is a contraction mapping if there is  $\gamma < 1$  so that

$$\|Mx - My\| \leq \gamma \|x - y\| \quad \forall x, y.$$

2. Banach fixed point theorem If  $M$  is a contraction mapping

(a) There is a unique  $x^*$  so that  $Mx^* = x^*$ .

(b) If  $x_{n+1} = Mx_n$  then  $x_n \rightarrow x^*$ .

3. Value iteration

- Since  $L$  is a contraction mapping, it converges to  $v^* = Lv^*$  (Theorem 6.5.7)
- If  $v = Lv$  then  $v = \max_\pi v^\pi$  (Theorem 6.5.3)
- Hence, value iteration converges to  $v^*$ .

### 3.8.9 Speed of convergence of value iteration

1. Theorem If  $r_t \in [0, 1]$ ,  $v_0 = 0$ , then

$$\|v_n - v^*\| \leq \gamma^n / (1 - \gamma).$$

2. Proof Note that  $\|v_0 - v^*\| = \gamma^0 / (1 - \gamma)$ , and

$$\|v_{n+1} - v^*\| = \|Lv_n - Lv^*\| \leq \gamma \|v_n - v^*\|.$$

Induction:  $\|v_n - v^*\| \leq \gamma^n / (1 - \gamma)$

$$\|v_{n+1} - v^*\| \leq \gamma \|v_n - v^*\| \leq \gamma^{n+1} / (1 - \gamma).$$

### 3.8.10 Policy Iteration

1. Algorithm
  - Input: MDP  $\mu$ , discount factor  $\gamma$ , initial policy  $\pi_0$ .
  - For  $n = 0, 1, \dots$
  - $v_n = (I - \gamma P_{\pi_n})^{-1} r = V^{\pi_n}$ .
  - $\pi_{n+1} = \arg \max_{\pi} \{r + \gamma P_{\pi} v_n\}$ .
  - Until  $\pi_{n+1} = \pi_n$ .
2. Policy iteration terminates with the optimal policy in a finite number of steps.
  - $v_n \leq v_{n+1}$  (Theorem 6.5.10)
  - There is a finite number of policies.
  - $v_n = \max_{\pi} \{r + \gamma P_{\pi} v_n\}$

## 3.9 RL: Stochastic Approximation

1. Sarsa (45')
2. Q-learning (45')

### 3.9.1 Two reinforcement learning setting

1. Online learning
  - **Observe** state  $s_t$
  - Take action  $a_t$
  - Get reward  $r_{t+1}$
  - See next state  $s_{t+1}$
2. Simulator access
  - **Select** a state  $s_t$
  - Take action  $a_t$
  - Get reward  $r_{t+1}$
  - See next state  $s_{t+1}$

### 3.9.2 Learning goals

1. Value function estimation

$$\begin{aligned} v_t^\pi &\rightarrow V^\pi & q_t^\pi &\rightarrow Q^\pi \\ v_t^* &\rightarrow V^* & q_t^* &\rightarrow Q^* \end{aligned}$$

2. Optimal policy approximation

$$\pi_t \rightarrow \pi^*$$

3. Bayes-optimal policy approximation

$$\pi_t \approx \arg \max_{\pi} \int_{\mu} \xi_t(\mu)$$

### 3.9.3 Monte Carlo Policy Evaluation

1. Direct Monte Carlo

- For all states  $s$
- For  $k = 1, \dots, K$
- Run policy  $\pi$ , obtain  $U^{(k)} = \sum_{t=1}^T r_t^{(k)}$

$$v_K(s) = \frac{1}{K} U^{(k)}$$

2. Online update

- For each  $k$

$$v_k(s) = v_{k-1}(s) + \alpha_k [U^{(k)} - v_{k-1}(s)]$$

- For  $\alpha_k = 1/k$ , the algorithm is the same as direct MC.

### 3.9.4 Monte Carlo Updates

1. Every-visit Monte Carlo

- Observe trajectory  $(s_t, r_t)_t$ , set  $U = 0$ .
- For  $t = T, T-1, \dots$
- $U = U + r_t$
- $n(s_t) = n(s_t) + 1$



- $v(s_t) = v(s_t) + \frac{1}{n(s_t)}[U - v(s_t)]$ .

## 2. First-visit Monte Carlo

- Observe trajectory  $(s_t, r_t)_t$ , set  $U = 0$ .
- For  $t = T, T - 1, \dots$
- $U = U + r_t$
- If  $s_t$  not observed before
- $n(s_t) = n(s_t) + 1$
- $v(s_t) = v(s_t) + \frac{1}{n(s_t)}[U - v(s_t)]$ .

### 3.9.5 Temporal Differences

- Idea: Replace actual  $U$  with an estimate:  $r_t + \gamma v(s_{t+1})$ .
- Temporal difference error:  $d_t = r_t + \gamma v(s_{t+1}) - v(s_t)$ .

#### 1. Temporal difference learning

$$v(s_t) = v(s_t) + \alpha_t d_t$$

#### 2. TD ( $\lambda$ )

$$v(s_t) = v(s_t) + \alpha_t \sum_{\ell=t}^{\infty} (\gamma \lambda)^{\ell-t} d_t$$

#### 3. Online TD ( $\lambda$ )

- $n(s_{t+1}) = n(s_{t+1}) + 1$
- For all  $s$

$$v(s_t) = v(s_t) + \alpha_t n(s) d_t$$

- $n = \lambda n$

### 3.9.6 Stochastic state-action value approximation

#### 1. SARSA

- Input policy  $\pi$
- Generate  $s_t, a_t, r_t, s_{t+1}, a_{t+1}$
- Update value

$$q(s_t, a_t) = q(s_t, a_t) + \alpha[r_t + \gamma q(s_{t+1}, a_{t+1}) - q(s_t, a_t)]$$

## 2. QLearning

- Observe  $s_t, a_t, r_t, s_{t+1}$
- Update value

$$q(s_t, a_t) = q(s_t, a_t) + \alpha[r_t + \gamma \max_a q(s_{t+1}, a) - q(s_t, a_t)]$$

$$q(s_t, a_t) += \alpha[r_t + \gamma \max_a q(s_{t+1}, a) - q(s_t, a_t)]$$

$$q(s_t, a_t) = (1 - \alpha)q(s_t, a_t) + \alpha[r_t + \gamma \max_a q(s_{t+1}, a)]$$

## 3. QLearning( $\lambda$ )

- Observe  $s_t, a_t, r_t, s_{t+1}$
- $e_{s_t, a_t} += 1$
- Update value

For every state-action  $s, a$ :

$$q(s, a) += (e_{s, a} \alpha)[r_t + \gamma \max_a q(s_{t+1}, a) - q(s, a)]$$

- $e = \lambda e$  ,  $\lambda < 1$ .

When  $\lambda \rightarrow 1$ , then you have Monte-Carlo.

## 4. Experience Replay

Run any of these algorithm repeatedly on a dataset you have collected so far.

### 3.10 Model-based RL

#### 3.10.1 Model-Based RL

1. Model  $\hat{\mu}_t$  Built using data  $h_t = \{(s_1, a_1, r_1), \dots, (s_t, a_t, r_t)\}$ .

$$P_t(s'|s, a) \triangleq P_{\hat{\mu}_t}(s'|s, a)$$

2. Algorithm At time  $t$

- $\hat{\mu}_t = f(h_t)$
- $\pi_t = \arg \max_{\pi} V_{\hat{\mu}}^{\pi}$ .

### 3.10.2 Example 1: Model-Based Value Iteration

1. Model

$$P_t(s'|s, a) = \frac{\sum_t \mathbb{I}\{s_{t+1} = s' \wedge s_t = s \wedge a_t = a\}}{\sum_t \mathbb{I}\{s_t = s \wedge a_t = a\}} = \frac{N_t(s, a, s')}{N_t(s, a)}$$

$$\rho_t(s, a) = \frac{\sum_t r_t \mathbb{I}\{s_t = s, a_t = a\}}{N_t(s, a)}$$

### 3.10.3 Asynchronous Value Iteration

For  $n = 1, \dots, n_{max}$ , all  $s$

$$v(s) := \max_a \rho_t(s, a) + \gamma \sum_{s'} P_t(s'|s, a) v(s')$$

### 3.10.4 Greedy actions

$$a_t = \arg \max_a \rho_t(s, a) \gamma \sum_{s'} P_t(s'|s, a) v_{n_{max}}(s'|s, a)$$

### 3.10.5 Example 2: Dyna-Q Learning

Why do value full iteration at **every** step?

1. Model  $P_t, \rho_t$

### 3.10.6 Q-iteration

For some  $s \in S$ , e.g.  $s_t$

$$q_t(s, a) = \rho_t(s, a) + \gamma \sum_{s'} P_t(s'|s, a) v_{t-1}(s') v_t(s, a) = \max_a q_t(s, a)$$

### 3.10.7 Greedy actions

$$a_t = \arg \max_a q_t(s, a)$$

### 3.10.8 Questions

- Is a point-estimate of the MDP enough?
- How fully do we need to update the value function?
- Which states should we update?
- How fast should the policy change?

### 3.11 Approximate Dynamic Programming

1. Fitted Value Iteration (45')
2. Approximate Policy Iteration (45')

#### 3.11.1 RL in continuous spaces

- From Tables to Functions

1. Value Function Representations

- Linear feature representation

$$v_{\theta}(s) = \sum_i \phi_i(s) \theta_i$$

2. Policy Representations

- Linear-softmax (Discrete Actions)

$$\pi_{\theta}(a|s) = \exp \sum_i \phi_i(s) \theta_i$$

#### 3.11.2 Approximating a function $f$

1. Approximation error of a function  $g$

$$\|f - g\| \triangleq \int_x |f(x) - g(x)| dx$$

2. The optimisation problem

$$\min_g \|f - g\|$$

#### 3.11.3 Fitting a value function to data

1. Monte-Carlo fitting

- Input  $\pi, K, N, \gamma, \epsilon$
- Sample  $N$  states  $s_n$
- Calculate  $\hat{V}_n$  through  $K$  rollouts of depth  $T > \ln_{1/\gamma}[1/\epsilon(1 - \gamma)]$
- Call  $\theta = \text{REGRESSION}(\Theta, (s_n, \hat{V}_n))$

2. Regression (linear, with SGD)

- 
- Initialise  $\theta \in \Theta$ .
- For  $n = 1, \dots, N$
- $\theta$

**3.11.4 Approximate Value Iteration**

- For  $s \in S$
- Calculate  $u(s) = \max_a r(s, a) + \gamma \int_S dP(s'|s, a)v_\theta(s')$  for all  $s \in \hat{S}$ .
- $\min_\theta \|v_\theta - u\|_{\hat{S}}$ , e.g.

$$\|v_\theta(s) - u\|_{\hat{S}} = \sum_{s \in \hat{S}} |v_\theta(s) - u(s)|^2$$

**3.11.5 Q-learning with function approximation**

1. Standard Q-update:

$$q_{t+1}(s_t, a_t) = (1 - \alpha_t)q_t(s_t, a_t) + \alpha_t[r_t + \gamma \max_a q_t(s_{t+1}, a)]$$

2. Gradient Q-update Minimise the squared TD-error

$$d_t = r_t + \gamma \max_a q_t(s_{t+1}, a) - q_t(s_t, a_t)$$

$$\nabla_\theta d_t^2 = 2d_t \nabla_\theta q_t(s_t, a_t)$$

**3.12 Policy Gradient**

1. Direct Policy Gradient, i.e. REINFORCE (45')
2. Actor-Critic Methods, e.g. Soft Actor Critic (45')

**3.13 Bayesian methods**

1. Thompson sampling (25')
2. Bayesian Policy Gradient (20')
3. BAMDPs (25')
4. POMDPs (20')

### **3.14 Regret bounds**

1. UCB (45')
2. UCRL (45')

### **3.15 MCTS**

1. UCT (45')
2. Alphazero (45')

### **3.16 Advanced Bayesian Models**

1. Linear Models (20')
2. Gaussian Processes (25')
3. GPTD (45')

### **3.17 Inverse Reinforcement Learning**

1. Apprenticeship learning (45')
2. Probabilistic IRL (45')

### **3.18 Multiplayer games**

Bayesian games (90')