Reinforcement Learning and Decision Making Under Uncertainty

Christos Dimitrakakis

May 21, 2024

Outline

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 inroduced.

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. There are two main types of projects, though a project can be hybrid.

Application project.

Application projects proposals need to contain the following:

- Domain description and goals: What is the problem, in general terms, and which aspect would you try and solve in an MDP/RL framework? Make sure to cite relevant literaure.
- Methodology: How would you formalise the problem mathematically? Which algorithms and/or models do you intend to apply at different stages of the project? Feel free to read widely about both the problem and algorithms and do cite relevant literature.
- Experiment design: How would you know that the method is working? How would you compare with existing solutions? In what context would you expect an improvement? How would you measure it? How will you test the robustness of your solution over variations in the problem instance?
- Expected results: What results do you expect to obtain, and what do you think might go wrong? In what way do you expect an improvement?

Algorithmic project.

- Algorithmic/theory problem and goals: What is the deficiency, in general terms, of current theory and algorithms that your method would try to improve? As an example, the goal could be reducing computational complexity, increasing data efficiency, improving robustness or applicability of a specific family of algorithms; or introducing a slightly different setting to existing ones. In other words, which is the open problem you will be addressing? Make sure to cite relevant literature to better identify the problem.
- Methodology: What kind of existing algorithms, theory or technical result would you rely on? Would you be combining various existing results? What would be the most significant novelty of your methodology? Do cite relevant literature.
- Experiment design (if applicable): How would you know that the method is working? How would you compare with existing solutions? In what context would you expect an improvement? How would you measure it?
- Expected results: What results do you expect to obtain, and

Grading for projets:

Grades will be adjusted based on group size with on letter grade up/down for double/half the mean group size.

- Environments: A. Complex, well described environment that captures all of the elements of the application or algorithmic cproblem. B. The environment is simple or lacks description. C. An adequate environment that captures the basic setting. D. Insufficient environment or description. E. Insuffcient environment and description.
- Algorithms: A. Significantly novel algorithms that are well described. B. Some novelty in the algorithms, with good descriptions. C. Some novelty in the algorithms, but descriptions are lacking. D. Insufficient novelty or descriptions. E. Insufficient novelty and descriptions.
- Experiments: A. Thorough experiments with ablation tests and comparisons over algorithms and environments, that are well-described. B. Somewhat incomplete experiments or descriptions. C. Sufficient experiments and descriptions. D. Insufficient experiments or descriptions. E. Insufficient

Essential

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

Recommended

- Elementary knowledge of probability and statistics.

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.

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

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.

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.

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.

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.

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

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

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.
No previous machine learning knowledge is needed.

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)

Programming

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

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

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 $a \sim^* b$

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.

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

Lab: Probability, Expectation, Utility

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

Assignment.

Exercise 7, 8, 9.

Further Reading:

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

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', $E[U|a^*] \geq E[L|a]$ 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:`

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').

The meteorologists problem

-n metereological stations $\mathcal{M} = \{1, \ldots, n\}$.

- x_t: Weather on day t (0 = dry, 1 = rain)$
- $P_\mu(x_t | x_{t-1}, x_{t-2}, \ldots)$ station μ prediction for dry/rain.

Station	Day 1	Day 2	Day 3	Day 4
1	60%	50%	40%	30%
2	30%	25%	20%	15%
3	40%	50%	50%	40%

- How should we combine these predictions?

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]

Statistical estimation

- A family of models $\{P_\mu | \mu \in \mathcal{M}\}$
- Data x.

Maximum Likelihood Estimation

- Find μ maximising $P_\mu(x)$

Maximum A Posteriori Estimation

- Prior belief ξ
- Find μ mximising $\xi(\mu)P_\mu(x)$

Bayesian Estimation

- Return function $\xi(\mu|x) = P_\mu(x)\xi(\mu)/\sum_{\mu'} P_{\mu'}(x)\xi(\mu')$

(Bayesian) MSE Estimation

- Find $\hat{\mu}$ minimising $\mathbb{E}_\xi[(\hat{\mu} - \mu)^2|x] = \sum_\mu (\hat{\mu} - \mu)^2 \xi(\mu|x)$

Maximum Likelihood Estimation

- Input: Data $x_1, \ldots, x_t$, A set of models $\{P_\mu | \mu \in \mathcal{M}\}$
- Inference: The model μ^*_{ML} maximising

$$P_\mu(x_1, \ldots, x_t)$$

- Prediction: $P_{\mu^*_{ML}}(x_{t+1} | x_1, \ldots, x_t)$.

Maximum A Posteriori Estimation

- Input: Data $x_1, \ldots, x_t$, set of models $\{P_\mu | \mu \in \mathcal{M}\}$, prior $\xi(\mu)$ on models
- Inference: The model μ^*_{MAP} maximising

$$P_\mu(x_1, \ldots, x_t)\xi(\mu)$$

- Prediction: $P_{\mu^*_{MAP}}(x_{t+1}|x_1, \ldots, x_t)$.

Bayesian Estimation

- Input: Data D, set of models $\{P_\mu | \mu \in \mathcal{M}\}$, prior $\xi(\mu)$ on models
- Inference: The posterior probability over models:

$$\xi(\mu|x_1, \ldots, x_t) = \frac{P_\mu(x_1, \ldots, x_t)\xi(\mu)}{\sum_{\mu'} P_{\mu'}(x_1, \ldots, x_t)\xi(\mu)}$$

- Prediction: $P(x_{t+1}|x_1, \ldots, x_t) = \sum_\mu P_\mu(x_{t+1}|x_1, \ldots, x_t)$.

MSE estimation

Sometimes we care about finding a point estimate for some distribution. For example, let us say we have some distribution ξ over some unknown variable μ and we need to select one μ, and we want to report a value μ^* minimising the squared error $(\mu - \mu^*)^2$ in expectation

$$\mathbb{E}_\xi[(\mu - \mu^*)^2] = \int_{\mathcal{M}} (\mu - \mu^*)^2 d\xi(\mu)$$

To find the minimising μ^* we can take the derivative

$$d/d\mu^* \, \mathbb{E}_\xi[(\mu-\mu^*)^2] = \int_{\mathcal{M}} d/d\mu^* (\mu-\mu^*)^2 d\xi(\mu) = \int_{\mathcal{M}} 2(\mu-\mu^*) d\xi(\mu)$$

Example: Beta-Bernoulli

Consider a coin with an unknown distribution of head or tails. We can model this as

$$x_t \mid \mu \sim \mathrm{Bern}(\mu)$$
$$\mu \sim \mathrm{Beta}(\alpha_0, \beta_0)$$

[Drawing on board] Data: $x_1, \ldots, x_T$ with empirical mean $\hat{\mu} = \frac{1}{t} \sum_{t=1}^{T} x_t$.

Bayesian estimate:

$\alpha_T = \alpha_0 + \sum_{t=1}^{T} x_t$, $\beta_T = \beta_0 + \sum_{t=1}^{T} (1 - x_t)$

ML estimation:

We can show that $\mu^*_{ML} = \hat{\mu}$.

MAP estimation:

We can show that $\mu^*_{MAP} = \alpha_T - 1 T$.

MSE estimation:

We can show that $\mu^*_{MSE} = \alpha_T \alpha_T + \beta_T$.

Lab: Decision problems and estimation (45')

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

Assignment: James Randi

n meteorologists as prediction with expert advice

- Predictions $p_t = p_{t,1}, \ldots, p_{t,n}$ 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

At each step t:

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

The update does not depend on a_t

Prediction with expert advice

- Advice $p_t = p_{t,1}, \ldots, p_{t,n} \in D$
- 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 $U = \sum_{t=1}^{T} r_t$.

Relation to n meteorologists

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

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

The update does not depend on a_t

The Exponentially Weighted Average

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

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

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.

The n armed stochastic bandit problem

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

The Regret

-Total regret with respect to the best arm:

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

- Expected regret of an algorithm π:

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

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

Bernoulli bandits

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

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

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 μ_{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: $\hat{\mu}_{t,i} = s_{t,i}/n_{t,i}$.

Priors for Bernoulli distribution

The standard prio

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.

Upper confidence bound: exploration bonuses

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

Posterior sampling: randomisation

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

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

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

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 $\mathrm{Beta}(\alpha, \beta)$ then the -posterior at time t is $\mathrm{Beta}(\alpha + s_{t,i}, \beta + n_{t,i} - s_{t,i})$.
- [Visualise how the posterior changes for a biased coin as we obtain more data].

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)

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')

The Markov decision process

Interaction at time t

- Observe state $s_t \in S$
- Take action $a_t \in A$.
- Obtain reward $r_t \in \mathbb{R}$.

The MDP model μ

- Transition kernel $P_\mu(s_{t+1}|s_t, a_t)$.
- Reward with mean $\rho_\mu(s_t, a_t)$

Policies

- Markov policies $\pi(a_t|s_t)$

Utility

Total reward up to a finite (but not necessarily fixed) horizon T

$$U_1 = \sum^{T} r_t$$

MDP examples

Shortest path problems

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

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.

Monte Carlo Policy Evaluation

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

Policy Evaluation

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

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:

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

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

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

Plan

1. DP: Value Iteration (45')
2. DP: Policy Iteration (45')

Infinite horizon setting

Utility

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

Discount factor $\gamma \in (0, 1)$

Tells us how much we care about the future. Note that

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

Value iteration

Idea: Run backwards induction, discounting by γ until convergence.

Algorithm

- Input: MDP μ, discount factor γ, threshold ϵ
- $v_0(s) = \rho_\mu(s)$ for all s
- For $n = 1, \ldots$

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

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

Norms

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

Matrix notation for finite MDPs

- r: reward vector.
- P_π: transition matrix.
- v: value function vector.

Stationary policies

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

Matrix formula for value function

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

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

Convergence of value iteration

Proof idea

1. Define the VI operator L so that $v_{n+1} = Lv_n$.
2. Show that if $v = V^*$ then $v = Lv$.
3. Show that $\lim_{n\to\infty} v = V^*$.

Further questions

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

Policy evaluation

Policy evaluation theorem

For any stationary policy π, the unique solution of

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

Proof

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

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

Interpretation: $X = (I - P)^{-1}$

Is the total discounted number of times reaching a state

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

Optimality equations

Policy operator

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

Bellman operator

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

Bellman optimality equation

$$v = Lv$$

Value iteration convergence proof

Contraction mappings

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

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

Banach fixed point theorem

If M is a contraction mapping

1. There is a unique x^* so that $Mx^* = x^*$.
2. If $x_{n+1} = Mx_n$ then $x_n \to x^*$.

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^*.

Speed of convergence of value iteration

Theorem

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

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

Proof

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

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

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

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

Policy Iteration

Algorithm

- Input: MDP μ, discount factor γ, initial policy π_0.
- For $n = 0, 1, \ldots$
- $v_n = (I - \gamma P_{\pi_n})^{-1} r = V^{\pi_n}$.
- $\pi_{n+1} = \arg\max_\pi \{r + \gamma P_\pi v_n$.
- Until $\pi_{n+1} = \pi_n$.

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.
- $v_n = \max_\pi \{r + \gamma P_\pi v_n\}$

1. Sarsa (45')

2. Q-learning (45')

Two reinforcement learning setting

Online learning

- Observe state s_t
- Take action a_t
- Get reward r_{t+1}
- See next state s_{t+1}

Simulator access

- Select a state s_t
- Take action a_t
- Get reward r_{t+1}
- See next state s_{t+1}

Learning goals

Value function estimation

$$v_t^\pi \to V^\pi \qquad q_t^\pi \to Q^\pi$$
$$v_t^* \to V^* \qquad q_t^* \to Q^*$$

Optimal policy approximation

$$\pi_t \to \pi^*$$

Bayes-optimal policy approximation

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

Monte Carlo Policy Evaluation

Direct Monte Carlo

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

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

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.

Monte Carlo Updates

Every-visit Monte Carlo

- Observe trajectory $(s_t, r_t)_t$, set $U = 0$.
- For $t = T, T-1, \ldots$
- $U = U + r_t$
- $n(s_t) = n(s_t) + 1$
- $v(s_t) = v(s_t) + \frac{1}{n(s_t)}[U - v(s_t)]$.

First-visit Monte Carlo

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

Temporal Differences

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

Temporal difference learning

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

TD (λ)

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

Online TD (λ)

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

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

Stochastic state-action value approximation

SARSA

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

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

QLearning

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

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

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

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

QLearning(λ)

Model-Based RL

Model $\hat{\mu}_t$

Built using data $h_t = \{(s_1, a_1, r_1), \ldots, (s_t, a_t, r_t)\}$.

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

Algorithm

At time t

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

Example 1: Model-Based Value Iteration

Model

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

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

Asynchronous Value Iteration

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

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

Greedy actions

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

Example 2: Dyna-Q Learning

Why do value full iteration at every step?

Model

P_t, ρ_t

Q-iteration

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

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

Greedy actions

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

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?

1. Fitted Value Iteration (45')

2. Approximate Policy Iteration (45')

RL in continuous spaces

- From Tables to Functions

Value Function Representations

- Linear feature representation

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

Policy Representations

- Linear-softmax (Discrete Actions)

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

Approximating a function f

Approximation error of a function g

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

The optimisation problem

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

Fitting a value function to data

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 $\theta = \text{Regression}(\Theta, (s_n, \hat{V}_n))$

Regression (linear, with SGD)

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

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.

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

Q-learning with function approximation

Standard Q-update:

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

Gradient Q-update

Minimise the squared TD-error

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

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

Approximate policy iteration

- $\pi_k = \arg\min_{\pi \in \Pi} \|\hat{L} v_{k-1} - \hat{L} v_{k-1}\|$
- $v_k = \arg\min_{v \in V} \|v - \hat{V}^{\pi_k}\|$

Bellman error methods

$$\|v - Lv\| = \sum_s D(s)^2$$

$$D(s) = v(s) - \max_a \int_S v(j) dP(j|s, a)$$

1. Direct Policy Gradient, i.e. REINFORCE (45')

2. Actor-Critic Methods, e.g. Soft Actor Critic (45')

Policy gradient

We want to solve the problem

$$\max_{\theta} \mathbb{E}_\theta[U], \qquad \mathbb{E}_\theta[U] = \int_S dy(s) \int_H p_\theta(h \mid s_1 = s) U(h),$$

where

- θ parametrises a policy
- $y(s)$ is a starting-state distribution
- $h = (s_t, a_T, r_t)_{t=1}^T$ is a trajectory and
- $U(h) = \sum_t r_t \gamma^{t-1}$ is its utility

Policy Gradient Theorem I: Analytic Gradients

$$V = (I - \gamma P)^{-1} r$$

First of all, $\nabla A^{-1} = -A^{-1} \nabla A A^{-1}$ and so

$$\nabla_\theta V = \gamma (I - \gamma P)^{-1} \nabla P (I - \gamma P)^{-1}$$

Finally,

$$\nabla P_{ij} = \nabla \sum_a P(s' = j | s = i, a) \pi(a \mid s)$$

Policy Gradient Theorem II: State-Visitations

- $X = (I - \gamma P)^{-1}$ discounted state-visitation matrix
- $x = y^\top X$ expected state visitations from starting state distribution

Then

$$\nabla \mathbb{E}[U] = \sum_x x(s) \sum_a \nabla \pi(a|s) Q^\pi(s, a)$$

- We can approximate x and Q for the gradient update.

Policy Gradient Theorem III: Reinforce

$$\nabla \mathbb{E}[U] = \sum_h U(h) \nabla P(h) = \sum_h U(h) P(h) \nabla \ln P(h).$$

This allows us to use the approximation

$$\nabla \mathbb{E}[U] \approx \frac{1}{K} \sum_{k=1}^{K} \sum_h U(h^{(k)}) \nabla \ln P(h^{(k)}), \qquad h^{(k)} \sim P(h)$$

One-shot normal-form games

Given a utility function $U(a, \omega)$, we need to find the value of the game

$$U^* = \max_a \min_\omega U(a, \omega) = \max_a \min_\omega U(a, \omega).$$

There is no guarantee that there is a solution. However, when we define

$$U(\pi, \xi) = \sum_a \sum_\omega \pi(a) U(a, \omega) \xi(\omega)$$

a solution is guaranteed by the *minimax theorem*

Extensive-form alternating-move zero sum games

- At time t:
- The state is s_t, players receive rewards $\rho(s_t), -\rho(s_t)$
- Player chooses action a_t, which is revealed.
- The state changes to s_{t+1}, and is revealed.
- Players receive reward $\rho(s_{t+1}), -\rho(s_{t+1})$
- Player chooses action ω_{t+1}.
- The state changes to s_{t+2}.
- Player a receives $\rho(s_t)$ and b receives $-\rho(s_t)$.

The utility for player a is

$$U^1 = \sum_t \rho(s_t),$$

while for ω it is

$$U^2 = -\sum_t \rho(s_t)$$

Backwards induction for Alternating Zero Sum Games

Let π_1 and π_2 be the policies of each player and π the joint policy.

The value function of a policy $\pi = (\pi_1, \pi_2)$

For the utility of player 1, we get:

$$V_t^{1,\pi}(s) \triangleq \mathbb{E}_\pi[U_t^1 \mid s_t = s] = \rho(s) + \mathbb{E}[U_{t+1}^1 \mid s_t = s] \tag{1}$$

$$= \rho(s) + \sum_a \pi(a \mid s) \sum_j V_{t+1}^{1,\pi}(j) P(j \mid s, a) \tag{2}$$

$$V_{t+1}^{1,\pi}(j) = \rho(j) + \sum_b \pi(b \mid j) \sum_j V_{t+2}^{1,\pi}(j) P(k \mid j, b) \tag{3}$$

We can define the optimal value function analogously to MDPs:

$$V_t^{1,*}(s) = \max_{\pi_1} \min_{\pi_2} \mathbb{E}_\pi[U_t^+ \mid s_t = s] \tag{4}$$

$$= \rho(s) + \max_a \sum_j V_{t+1}^{1,*}(j) P(j \mid s, a) \tag{5}$$

$$V_{t+1}^{1,*}(j) = \rho(j) + \min_\omega \sum_j V_{t+1}^{1,*}(j) P(k \mid j, \omega) \tag{6}$$

Example: Chicken

ρ^1, ρ^2	turn	dare
turn	0, 0	-5, -1
dare	1, -5	-10, -10

Example: Prisoner's dilemma

ρ^1, ρ^2	cooperate	defect
cooperate	0, 0	-5, -1
defect	1, -5	-10, -10

Example: penalty shot

ρ^1, ρ^2	kick left	kick right
dive left	1, -1	-1, 1
dive right	-1 1	1, -1

Extensive-form general sum games

- At time t:
- The state is s_t, players receive rewards $\rho^i(s_t)$.
- Player $i = I(s_t)$ chooses an action.
- The state changes to s_{t+1}, and is revealed.

The utility for each player is

$$U^i = \sum_t \rho^i(s_t)$$

Backwards induction for Alternating General Sum Games

Let π_i be the policy of the i-th player and π the joint policy.

The value function of a policy $\pi = (\pi_i)_{i=1}^n$

For any player i, we can define their value at time t as:

$$V_t^{i,\pi}(s) \triangleq \mathbb{E}_\pi[U_t^i \mid s_t = s] \tag{7}$$

$$= \rho^i(s) + \sum_{a \in A} \pi_{I(s)}(a \mid s) \sum_j V_{t+1}^{1,\pi}(j) P(j \mid s, a) \tag{8}$$

Optimal policies

For perfect information games, we can use this recursion:

$$a_t^*(s) = \arg\max_{a \in A} \sum_j V_{t+1}^{I(s),*}(j) P(j \mid s, a) \tag{9}$$

$$V_t^{i,*} = \rho^i(s) + \sum_j V_{t+1}^{i,\pi}(j) P(j \mid s, a_t^*(s)) \qquad \forall i \tag{10}$$

This ensures that we update the values of all players at each step.

General architecture

- Board representation
- Rollouts
- Monte-Carlo Tree Search
- Value approximation
- Policy approximation
- Model approximation

Example: DeepMind engines

1. Alpha Go (MCTS, value approximation)
2. Alpha Go Zero (MCTS, value and policy approximation)
3. Alpha Zero (almost the same)
4. Mu Zero (MCTS, value, policy and model approximation)

Monte-Carlo Tree Search

For each state s, select moves according to:

- Number of visits
- Move probability
- Expected value

Rollouts

Plain MCTS

- Start from s_t
- Generate $a_t = \arg\max_a Q_\theta(s, a) + U(s_t, a)$ where

$$U(s, a) = P(s, a)/N(s, a)$$

- Update \$Q$_\theta$(s,a) $\rightarrow$ 1/N(s,a) $\sum_{s' \in s_T}$ V(s')
- Loses

1. Thompson sampling (25')

2. Bayesian Policy Gradient (20')

3. BAMDPs (25')

4. POMDPs (20')

Priors

- Maximising expected utility

Policy types

- Memoryless policies
- Adaptive policies

Counterexamples

- Mixed MDP
- Many bandit problems

Bayesian update

- Just Bayes's theorem
- Example: Discrete MDPs

Gradient ascent

$$\nabla_\pi E_\xi^\pi[U] = \int_{\mathcal{M}} \nabla_\pi \mathbb{E}_\mu^\pi[U] d\xi(\mu) \approx \frac{1}{K} \sum_{k=1}^{K} \nabla_\pi \mathbb{E}_\mu^\pi{}^k[U], \sim \mu^k \sim \xi$$

Bayesian value function

$$U^*(\xi) = \max_\pi U(\pi, \xi)$$

$$U(\pi, \xi) \leq U^*(\xi) \leq \int_{\mathcal{M}} \max_\pi U(\pi, \mu) d\xi(\mu)$$

Hyperstates

- Exact solution: Bayes-Adaptive Markov Decision Processesx

Branch and bound

POMDPs

- In POMDPs, the state is unknown, so we must infer both μ, s.

1. UCB (45')

2. UCRL (45')

1. UCT (45')

2. Alphazero (45')

Generic Monte-Carlo Planning

MCPlanner(state)

- *repeat* {
- search(state, 0)
- } *until* TimeOut
- *return* BestAction(state, 0)

search(state, depth)

- *if* Terminal(state) *return* 0
- *if* Leaf(state) *return* Evaluate(state)
- a = SelectAction(state, depth)
- (NextState, reward) = Simulate(state, action)
- q = reward + γ search(NextState, depth + 1)
- UpdateValue(state, action, q, depth)
- return q

Bandit-Based Monte-Carlo Planning

SelectAction(s, d)

- $N_{s,d}(t)$: Visits of state s
- $N_{s,a,d}(t)$: Number of times s, a is selected
- $c_{x,y} = C\sqrt{\ln(x)/y}$

$$a^* = \arg\max_a Q_t(s,a,d) + C\sqrt{\ln(N_{s,d}(t))/N_{s,a,d}(t)}$$

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

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