Decision and game theory

Adam Boult (www.bou.lt)

July 2, 2025

Contents

Preface

This is a live document, and is full of gaps, mistakes, typos etc.

Part I

Decision theory under certainty

Chapter 1

States and accentuators

1.1 Introduction

Chapter 2

Discrete choice

2.1 Introduction

Chapter 3

Continuous choice

3.1 Axioms

3.1.1 Choice set

Economic agents face options from some set. This could be consumption choices, numbers of hours to work, or how much capital to invest in at a factory.

Utility functions

Calculating choices: Unrestricted choices

The choice of an agent is the selection which corresponds to the highest value of the utility function. Consider:

$f = 2(x-1)^2 - 10$

We can easily calculate that even if the agent can choose any real number x, they will chose 1.

This approach can be used if there are not meaningful constraints, or those constraints are implicit in the utility function. For example a firm can be modelled as profit maximising, where profit is a function of revenue and costs, with a single maximising value.

Other agents, such as a consumers, may instead have utility over consumption and leisure, and a separate constraint over this. This could be solved using simultaneous equations, but such an approach is not always desirable.

3.1.2 Complete preferences

Axiom 2: An economic agent has preferences across all pairs of elements of the choice set

Where an agent prefers one option to another, we say this choice is preferred to the other. We can show this formulaically. If a is preferred to b then:

$a \succ b$

Alternatively b may be preferred to a.

$a \prec b$

Finally an agent may be indifferent to the choices.

$a \sim b$

We use can additional symbols to denote an agent prefers or is indifferent between options:

$a \succeq b$

$a \preceq b$

Note that if:

$a \succeq b$

And:

$a \preceq b$

Then:

$a \sim b$

For a set of choice, we want the agent to have preferences defined across possible subsets – the power set.

Add symbols for choice across a set. include symbols for choice of a set, x =c(a,b,utility)

a¿b etc only for pairwise, but preferences are across more than that

Discussion of repeated. agents don't choose the same thing each time, interpretation?

3.2 Choices onto $\mathbb{R}^1$ space

3.2.1 Transitivity

Axiom 3: Transitivity

If $a \preceq b$, and $b \preceq c$, then $a \preceq c$.

For example an agent is offered apples or bananas, and prefers apples, and then bananas or carrots, and chooses bananas. If the agent is then offered apples or carrots, they axiom of transitivity says they chould choose apples.

This is not always observed. For example if a firm offers two products, a cheap a option and an expensive b option, then a preference may be:

$a \prec b$

The firm may add another product c slightly less expensive than b, with fewer features, such if there were only 2 of the goods on sale:

$a \prec b$

$a \prec c$

$b \prec c$

In particular, the last two imply the first, through the axiom of transitivity.

However with all three, the marketing effectively makes the consumer choose b, by making it look like a better deal, so we observe:

$a \prec s$

$e \prec c$

$c \prec s$

Even though the elements have not changed.

Axioms of revealed preference

Add weak axiom of revealed preference

3.2.2 Independence of irrelevant alternatives

This is not always observed. For example if a firm offers two products, a cheap a option and an expensive b option, then a preference may be:

$a \prec b$

The firm may add another product c slightly less expensive than b, with fewer features, such if there were only 2 of the goods on sale:

$a \prec b$

$a \prec c$

$b \prec c$

In particular, the last two imply the first, through the axiom of transitivity.

However with all three, the marketing effectively makes the consumer choose b, by making it look like a better deal, so we observe:

$a \prec s$

$e \prec c$

$c \prec s$

Even though the elements have not changed.

Axioms of revealed preference

Add weak axiom of revealed preference

Strong axiom. can derive both (?)

3.2.3 Utility functions

Given that an agent chooses a selection from a set, how can we model this?

As the outcome of the function is one of the options available to the agent, we can model it by putting a value number on each choice, and where the choice is the selection with the largest value. We can do this by applying a function to each element in the choice set.

This holds only if the agent is rational, that is, if they prefer A to B, they do not switch to B if C is offered. This is due to the transitive properties of real numbers. That is, if $a \geq b$ and $b \geq c$ then $a \geq c$.

If there are multiple elements with the largest value, then the agent would be indifferent between these choices.

A simple example would be choosing the number of apples to consume.

If the agent always prefers more apples, we can have a function which is always increasing when the number of apples increases.

This could be modelled by:

$f = x$

$f = (x-1)^2 + 1$

$f = 2(x-1)^2 - 10$

$f = cos(x)$

$f = c$

These correspond to different preferences. In the first the agent prefers more and more apples. In the second and third the agent prefers one apple. These two formulas are monotonic transformations of each other and so are identical for describing preferences. The fourth describes an infinite number of optimal numbers of apples, but is unlikely to correspond to any real preferences, and the fifth shows that agent doesn't care about apples.

The last gives a real number as output, but doesn't necessarily take in a real number. Utility functions generally take real numbers, and always give out real numbers.

Solving

Option which maximises utility

3.3 Choices from $\mathbb{R}^1$ space

3.3.1 Continuity

Transitivity

Axioms of continuity and transitivity

Continuous

Independence of irrelevant alternatives

Axiom 3: Continuity

In order to find optimal points for a utility function we want these functions to be differentiable. This requires complete sets of choices. That is, if a if preferred to b, points very close to a will also be preferred to b.

Agents often make choices discretely. How much of good x to consume, whether to go on a holiday. We treat these as continuous. This is generally not problematic as agents choices become less discrete over longer time spans, and most economic areas of interest do not rest on discrete consumption.

Marginal utility

We can differentiate our utility function with respect to a good. For example, for:

$f = 2(x-1)^2 - 10$

$\frac{\delta f}{\delta x} = 4x - 1$

This last term is the marginal utility of x, often shown as MU_x.

Solving

3.4 Choices from $\mathbb{R}^n$ space

3.4.1 Indifference curves

Marginal rate of substitution

We can consider how much of one good a consumer is willing to give up to get one of another.

Consider the utility function:

$U = f(x, y)$

We can examine the change in utility following changes in inputs by taking the total differential.

$dU = \frac{\delta f}{\delta x}dx + \frac{\delta f}{\delta y}dy$

We want to examine changes where dU is 0, so:

$\frac{\delta f}{\delta x}dx + \frac{\delta f}{\delta y}dy = 0$

$\frac{\delta f}{\delta x}dx + \frac{\delta f}{\delta y}dy = 0$

$MU_x dx + MU_y dy = 0$

$\frac{dy}{dx} = -\frac{MU_x}{MU_y}$

This is the form of the indifference curve. We refer to $\frac{MU_x}{MU_y}$ as the marginal rate of substitution.

ADD GRAPH TO SHOW

Multiple choices

If the set of choices is more complex, say there are now apples and bananas, we have to be more careful with a representative function.

The agent still prefers more apples and more bananas, but the following imply different choices:

$f = x^2.y^2$

$f = ln(x) + ln(y)$

In the first example, the agent would always swap an apple for a banana, if they had more apples, whereas the opposite is true in the second case. Functional form is important with multiple goods.

Solving

3.4.2 Specific function results

Examples of utility functions

Cobb Douglas

$U = f(x, y)$

$U = x^\alpha y^\beta$

$MU_x = \alpha x^{\alpha-1} y^\beta$

$MU_x = \alpha \frac{y^\beta}{x^{1-\alpha}}$

$MU_y = \beta \frac{x^\alpha}{y^{1-\beta}}$

So:

$\frac{MU_x}{MU_y} = \frac{\alpha \frac{y^\beta}{x^{1-\alpha}}}{\beta \frac{x^\alpha}{y^{1-\beta}}}$

$\frac{MU_x}{MU_y} = \frac{\alpha}{\beta} \frac{y}{x}$

Constant elasticity of substitution

$U = f(x, y)$

$U = (ax^b + (1-a)y^b)^{\frac{1}{b}}$

$MU_x = \frac{1}{b} abx^{b-1}(ax^b + (1-a)y^b)^{\frac{1}{b}-1}$

$MU_y = \frac{1}{b}(1-a)by^{b-1}(ax^b + (1-a)y^b)^{\frac{1}{b}-1}$

So:

$\frac{MU_x}{MU_y} = \frac{\frac{1}{b} abx^{b-1}(ax^b + (1-a)y^b)^{\frac{1}{b}-1}}{\frac{1}{b}(1-a)by^{b-1}(ax^b + (1-a)y^b)^{\frac{1}{b}-1}}$

$\frac{MU_x}{MU_y} = \frac{ax^{b-1}}{(1-a)y^{b-1}}$

Constant elasticity of substitution

$U = f(x, y)$

$U = ax + by$

$MU_x = a$

$MU_y = b$

So

$\frac{MU_x}{MU_y} = \frac{a}{b}$

Constant elasticity of substitution

$U = f(x, y)$

$U = \alpha.ln(x) + \beta.ln(y)$

$MU_x = \frac{\alpha}{x}$

$MU_y = \frac{\beta}{y}$

So:

$\frac{MU_x}{MU_y} = \frac{\alpha}{\beta}\frac{y}{x}$

This is the same as Cobb Douglas. This reflects that the logarithmic function preseres positive inflections.

3.5 Budget constraints

3.5.1 Budget constraints

Economic agents can choose elements from a set. We can define this set both by what is included, and what is excluded.

An agent could choose inputs from an interval of real numbers. In this case the finding the value which maximises the utility function across all real numbers may not be a valid solution. For example an agent may optimally wish to consume 20 apples, but only have 10.

Similarly an agent may be limited in combinations of choices. For example an agent could choose how much to work and how much to enjoy leisure, but be constrained by the amount of time in the day.

3.5.2 Budget optimisation

Calculating choices: Restricted choices

But what about where there is not clear maximum, like:

$f = ln(x) + ln(y)$

Here the agent would always prefer more of x and y. In practice agents are often limited in their choices by budget constraints. That is, they cannot choose all combinations of inputs.

Here we can use a Lagrangian. This maximises the value of a function subject to constraints on inputs. This may not always be appropriate. The budget constraint for an agent is often an inequality, for example consumption is less than or equal to income, but the Lagrangian takes this to be binding.

Fortunately, this can be resolved. The value of λ in the Lagrangian corresponds to the marginal effect of weakening the constraint. This is positive where the constraint is binding on the agent, but not positive if it is not. Therefore if we find the constraint is not binding, we can remove it from the optimisation.

Under some conditions, constraints will always be binding. These are useful for specific cases of agents later.

In order for the constraint to be binding we make an additional assumption:

Condition 1: Non-satiation

The marginal utility of a good is always positive.

Note that we can "do economics" without this, but we want rely on Lagrangians.

Condition 2: Decreasing marginal utility

This ensures that we do not get corner solutions, for example consuming all apples.

These two assumptions allow the use of the Lagrangian.

We know that for the Lagrangian the following is true:

$$\frac{\frac{\delta f}{\delta x}}{\frac{\delta g}{\delta x}} = \frac{\frac{\delta f}{\delta y}}{\frac{\delta g}{\delta y}}$$

$$\frac{\frac{\delta f}{\delta x}}{\frac{\delta f}{\delta y}} = \frac{\frac{\delta g}{\delta x}}{\frac{\delta g}{\delta y}}$$

Where f is the utility function and g is the budget constraint.

Budget testing

Calculating choices: Restricted choices

But what about where there is not clear maximum, like:

$f = ln(x) + ln(y)$

Here the agent would always prefer more of x and y. In practice agents are often limited in their choices by budget constraints. That is, they cannot choose all combinations of inputs.

Here we can use a Lagrangian. This maximises the value of a function subject to constraints on inputs. This may not always be appropriate. The budget constraint for an agent is often an inequality, for example consumption is less than or equal to income, but the Lagrangian takes this to be binding.

Fortunately, this can be resolved. The value of λ in the Lagrangian corresponds to the marginal effect of weakening the constraint. This is positive where the constraint is binding on the agent, but not positive if it is not. Therefore if we find the constraint is not binding, we can remove it from the optimisation.

Under some conditions, constraints will always be binding. These are useful for specific cases of agents later.

In order for the constraint to be binding we make an additional assumption:

Condition 1: Non-satiation

The marginal utility of a good is always positive.

Note that we can "do economics" without this, but we want rely on Lagrangians.

Condition 2: Decreasing marginal utility

This ensures that we do not get corner solutions, for example consuming all apples.

These two assumptions allow the use of the Lagrangian.

We know that for the Lagrangian the following is true:

$$\frac{\frac{\delta f}{\delta x}}{\frac{\delta g}{\delta x}} = \frac{\frac{\delta f}{\delta y}}{\frac{\delta g}{\delta y}}$$

$$\frac{\frac{\delta f}{\delta x}}{\frac{\delta f}{\delta y}} = \frac{\frac{\delta g}{\delta x}}{\frac{\delta g}{\delta y}}$$

Where f is the utility function and g is the budget constraint.

3.5.3 Multiple restrictions

Calculating choices: Restricted choices

But what about where there is not clear maximum, like:

$f = ln(x) + ln(y)$

Here the agent would always prefer more of x and y. In practice agents are often limited in their choices by budget constraints. That is, they cannot choose all combinations of inputs.

Here we can use a Lagrangian. This maximises the value of a function subject to constraints on inputs. This may not always be appropriate. The budget constraint for an agent is often an inequality, for example consumption is less than or equal to income, but the Lagrangian takes this to be binding.

Fortunately, this can be resolved. The value of λ in the Lagrangian corresponds to the marginal effect of weakening the constraint. This is positive where the constraint is binding on the agent, but not positive if it is not. Therefore if we find the constraint is not binding, we can remove it from the optimisation.

Under some conditions, constraints will always be binding. These are useful for specific cases of agents later.

In order for the constraint to be binding we make an additional assumption:

Condition 1: Non-satiation

The marginal utility of a good is always positive.

Note that we can "do economics" without this, but we want rely on Lagrangians.

Condition 2: Decreasing marginal utility

This ensures that we do not get corner solutions, for example consuming all apples.

These two assumptions allow the use of the Lagrangian.

We know that for the Lagrangian the following is true:

$$\frac{\frac{\delta f}{\delta x}}{\frac{\delta g}{\delta x}} = \frac{\frac{\delta f}{\delta y}}{\frac{\delta g}{\delta y}}$$

$$\frac{\frac{\delta f}{\delta x}}{\frac{\delta f}{\delta y}} = \frac{\frac{\delta g}{\delta x}}{\frac{\delta g}{\delta y}}$$

Where f is the utility function and g is the budget constraint.

3.5.4 ??

Calculating choices: Restricted choices

But what about where there is not clear maximum, like:

$f = ln(x) + ln(y)$

Here the agent would always prefer more of x and y. In practice agents are often limited in their choices by budget constraints. That is, they cannot choose all combinations of inputs.

Here we can use a Lagrangian. This maximises the value of a function subject to constraints on inputs. This may not always be appropriate. The budget constraint for an agent is often an inequality, for example consumption is less than or equal to income, but the Lagrangian takes this to be binding.

Fortunately, this can be resolved. The value of λ in the Lagrangian corresponds to the marginal effect of weakening the constraint. This is positive where the constraint is binding on the agent, but not positive if it is not. Therefore if we find the constraint is not binding, we can remove it from the optimisation.

Under some conditions, constraints will always be binding. These are useful for specific cases of agents later.

In order for the constraint to be binding we make an additional assumption:

Condition 1: Non-satiation

The marginal utility of a good is always positive.

Note that we can "do economics" without this, but we want rely on Lagrangians.

Condition 2: Decreasing marginal utility

This ensures that we do not get corner solutions, for example consuming all apples.

These two assumptions allow the use of the Lagrangian.

We know that for the Lagrangian the following is true:

$$\frac{\frac{\delta f}{\delta x}}{\frac{\delta g}{\delta x}} = \frac{\frac{\delta f}{\delta y}}{\frac{\delta g}{\delta y}}$$

$$\frac{\frac{\delta f}{\delta x}}{\frac{\delta f}{\delta y}} = \frac{\frac{\delta g}{\delta x}}{\frac{\delta g}{\delta y}}$$

Where f is the utility function and g is the budget constraint.

Part II

Intertemporal decision theory under certainty

Chapter 4

States and deterministic environments

4.1 Introduction

4.1.1 States

4.1.2 State evolution

For environment with no actors, transition model is matrix (for finite state).

Can multiply matrix n times to get state in n periods.

Transition matrix is perumation matrix. 1 or 0 only.

4.1.3 Static and dynamic environments

Does time stop to allow decisions. eg chess v driving.

4.1.4 Discrete and continuous environments

4.1.5 Episodic and sequential environments

Chapter 5

Intertemporal decision making

5.1 Introduction

5.1.1 Intertemporal decision theory

$U_T = \sum_[t = T] d_t U(x_t)$

5.2 Types of discounting

5.2.1 Exponential discounting

Introduction

We have:

$U_T = \sum_[t = T]^\infty d_t U(x_t)$

Exponential discounting

$d_t = (1 + \delta)^t$

$U_T = \sum_[t = T]^\infty (1 + \delta)^t U(x_t)$

δ is the discount rate.

5.2.2 Hyperbolic discounting

Introduction

We have:

$U_T = \sum_[t = T]^\infty d_t U(x_t)$

Hyerbolic discounting

$d_t = \frac{1}{1+kt}$

$U_T = \sum_[t = T]^\infty \frac{1}{1+kt} U(x_t)$

k is the discount parameter.

5.2.3 Quasi-hyperbolic discounting

Introduction

We have:

$U_T = \sum_[t = T]^\infty d_t U(x_t)$

Quasi-hyperbolic discounting

$d_0 = 1$

$d_t = \beta\delta^t$

$U_T = U(x_0) + \sum_[t = T+1]^\infty \beta\delta^t U(x_t)$

δ is the discount rate.

5.3 Intertemporal economics

5.3.1 Intertemporal discounting

$u_t = f(x_t)$

$U_t = \sum_{i=t} u(x_i) d^i$

5.3.2 Euler's equation

5.3.3 Hyperbolic discounting

Chapter 6

Planning

6.1 Introduction

6.1.1 Robotics

Robot exists in some configuration space By cartesian coordinates, angles and lengths.

There are legal and illegal positions

Path planning is difficult

There are many degrees of freedom for complex robots

Higher dimension space makes solutions more difficult

6.2 Generating graphs

6.2.1 Visibility graphs

Visibility graph generates nodes from points on obstacles, current position and goal. generate graph.

Use A* to solve

That algorithm is called VGRAPH

6.2.2 Robots with volume

What if robot takes up space?

Expand each obstacle by size of robot

How to grow each obstacle? draw shape of robot around each obstacle. shape is relative to point, so the growth is asymmetric if you choose a point for the robot which is not in the middle. eg a vertex.

6.2.3 Rotation of robots

What about rotation? grow robot by shape of all rotations. but this is overly conservative. will miss paths

Can do vgraph for each rotation, and switch between rotations. can choose 2 or 3 rotations

6.2.4 Voronoi path planning

Find paths as far as way from obstacles as possible

Divide plane in cells, with a cell around each obstacle

Within a cell, all points are closest to that obstacle

We move along the lines between cells

Overly conservative

Hard to compute in 3d

Small environmental changes can significantly change the graph.

6.2.5 Probabilistic Roadmap Planner

Generate random configuration of robot in space.

Find n positions which are legal

Link them using k nearest neighbours

Risk: graph not fully connected.

If so, can add extra nodes between breaks

6.3 Fields

6.3.1 Potential fields

Attracted to goal, repelled from obstacle.

Goal has 0 potential energy

Can use gradient descent to get to goal

Problem of local minimums

If in a local minimum, can pertube by walking in a random direction to get out of it.

Can use laser, sonar, to detect obstacles and get potential

Part III

Decision theory under known uncertainty

Chapter 7

Lotteries and risk aversion

7.1 Introduction

7.1.1 Lotteries

A choice may not have a certain outcome.

For example an action could have have 50

7.1.2 Von Neumann-Morgenstern utility theorem

We can model any risk preference as:

$U[L] = \sum_i p_i u(x_i)$

If the agent is risk neutral we can use:

$u(x_i) = x_i$

If the agent is risk averse:

$u(x_i) = \ln x_i$

If the agent is risk loving we can use:

$u(x_i) = x_i^2$

7.2 Absolute risk aversion

7.2.1 Absolute risk aversion

Given a utility function we can calculate the risk aversion.

$A(x) = -\frac{u''(x)}{u'(x)}$

Constant Absolute Risk Aversion (CARA) is:

$A(x) = c$

$u(x) = 1 - e^{\alpha x}$

Hyperbolic Absolute Risk Aversion (HARA) is:

$A(x) = \frac{1}{ax + b}$

Increasing and Decreasing Absolute Risk Aversion (IARA and DARA):

Risk aversion increase or decreases in x.

7.3 Relative risk aversion

7.3.1 Relative risk aversion

Absolute risk aversion is:

$A(x) = -\frac{u''(x)}{u'(x)}$

$R(x) = xA(x)$

$R(x) = -\frac{xu''(x)}{u'(x)}$

7.4 Risk

7.4.1 Expected utility

If an agent faces an uncertain world they can make decisions under uncertainty. For example, how would an agent value £10 relative to a 50

There are many different attitudes an agent could have – we need a form which can capture these. A standard approach is expected utility.

We start by taking

$E[u(x)] =$

Subjective expected utility

7.4.2 Risk aversion

HARA Hyperbolic Absolute Risk Aversion

CRRA Constant Relative Risk Aversion

7.4.3 Prospect theory

Cumulative prospect theory.

7.4.4 Knightian uncertainty

7.4.5 Bounded rationality

Chapter 8

Fully observable Markov Decision Processes (MDP)

8.1 Stochastic environments

8.1.1 Markov chain recap

In a Markov chain we move from state to state randomly, following a transtion matrix.

We have:

- S - the state space
- s_1 - the initial state
- P - the transition model

8.1.2 The value function

State payoffs

An agent gets a payoff which depends on the current state. We now have:

- S - the state space
- s_1 - the initial state
- P - the transition model
- R - the reward distribution

Discounting

We maximise the reward function using discounting.

$E[\sum_{t=1}^{\infty} \gamma^{t-1} r_t | s_1]$

8.2 Markov Decision Processes (MDPs)

8.2.1 Markov Decision Processes (MDPs)

Introduction

Actions

In a MDP the agent can choose an action in each state. The action affects the transition matrix.

This means that the rewards depends on the actions taken. We now have:

- S - the state space
- s_1 - the initial state
- A - the action space
- P - the transition model
- R - the reward distribution

8.2.2 Policies

The decision maker needs to decide which action to take. For a MDP we assume that the agent knows:

- The current state
- The transition matrix
- The payoffs

If the current state is not known, we have a Partially Observable Markov Decision Process.

If the transition matrix or payoffs are not known we have a reinforcement learning problem.

8.2.3 The value function of a policy

Other

$P(s_{t+1}|H) = P(s_{t+1}|s_t = s, a_t = a) = P_{s,a}$

$E[r_t|H] = E[r_t|s_t = s, a_t = a] = R_{s,a}$

Optimal policies

Expected return is greater or identical to any other policy.

8.3 Identifying policies

8.3.1 Solving Markov Decision Processes with linear programming

8.3.2 The Bellman equation for Markov Decision Processes

8.3.3 Policy iteration

Policy iteration method

We start with a random policy.

We then loop:

- Evaluate the policy, using the Bellman equations.
- Update the policy

We update the policy by changing a to maximise:

$v_pi'(s) = r_\pi + \gamma P_\pi v_\pi(s)$

For example, if we have a policy of doing a in state s, we would see if we increase the value if we change a to a'.

8.3.4 Value iteration

Value iteration method

Doesn't directly calculuate policy. Doesn't require inverting matrix.

8.4 Markov Decision Processes with infinite states

8.4.1 Markov Decision Processes with infinite states

Part IV

Decision theory under unknown uncertainty

Chapter 9

Decision making with unknown uncertainty

9.1 Introduction

9.1.1 Introduction

In practice we don't know distribution.

9.1.2 Minimax

9.1.3 inimax regret

9.1.4 Robust decision making

9.1.5 Info-gap decision theory

9.1.6 Wald's maximin model

Chapter 10

Sensors and Partially Observable Markov Decision Processes (POMDP)

10.1 Partially observable environments

10.1.1 Hidden Markov Model (HMM) recap

10.1.2 Belief states

Inc initial knowledge of environment, page on storing observations

eg if position is right or left, at start belief state is both, if we try to move right, we are definitely right.

what if actions differ between states? can consider only actions available to all states if cost of illegal action is high

goal test: need to test all belief states

10.2 Partially Observable Markov Decision Processes

10.2.1 Partially Observable Markov Decision Processes (POMDP)

10.2.2 Policies for POMDPs

10.2.3 Value functions for POMDPs

max of linear terms?

10.2.4 Belief Markov Decision Processes

10.3 Identifying policies

10.3.1 Value iteration for POMDPs

Exponential in complexity for actions, observations?

10.3.2 Policy iteration for POMDPs

10.4 POMDPs with infinite states

10.4.1 POMDPs with infinite states

Chapter 11

Reinforcement learning

11.1 Reinforcement learning for Markov Decision Processes

11.1.1 Unknown MDP transition matrix

11.1.2 Exploration to understand the MDP

11.2 Temporal difference learning for Markov Decision Processes

11.2.1 Temporal difference learning

11.3 Q-learning for Markov Decision Processes

11.3.1 Q-functions

We can't use value functions, so we use Q-values instead.

$Q(s,a)$ is the value of taking action a in state s.

$Q(s,a) = R_{s,a} + \gamma E_{s'}[max_{a'}Q(s',a')]$.

assigned to every action/state combination), to real. h on q tables, q-values

11.3.2 Value iteration for Q

11.3.3 ϵ-greedy

11.4 Q-learning for large state Markov Decision Processes

11.4.1 Function approximation for Q

eg NN from state/action to R

11.4.2 Quantisation of states

Shrink the number of states.

11.4.3 End-to-end reinforcement learning

11.4.4 Deep Q-Network (DQN)

11.5 Reinforcement learning for POMDPs

11.5.1 Unknown POMDP transition matrix

11.5.2 Stacking frames

11.5.3 Deep Recurrent Q-Network

Chapter 12

Control theory

12.1 Introduction

Chapter 13

Active learning

13.1 Introduction

Part V

Game theory under certainty

Chapter 14

Simultaneous games

14.1 Introduction

14.1.1 Interaction between agents

Introduction

Previously actions map to states. Not now.

More intro

Previously we modelled utility as a function of variables in control of the agent, or constants. We now add another type of term: variables controlled by other agents.

Consider a simple pair of agents:

$u_a = f_a(x_a, y_a)$

$actions_a = \{x_a, y_a\}$

$u_b = f_b(x_b, y_b)$

$actions_b = \{x_b, y_b\}$

Each agent's decision does not affect the other agent. Consider now a utility function:

$u_a = f_a(x_a, y_a)$

$actions_a = \{xoffer; yoffer\}$

$u_b = f_b(x_b, y_b)$

$actions_b = \{accept; reject\}$

Where a offers a trade to b and b accepts or rejects. This is an example of a sequential game. There are many types of game, with differing implications.

14.2 Simultaneous games

14.2.1 Simultaneous games

One round simultaneous games

Economic agents face options from some set. This could be consumption choices, numbers of hours to work, or how much capital to invest in at a factory.

Consider the prisoner's dilemma game: table.table.table-bordered thead tr th th Silent th Tell tbody tr td Silent td (5,5) td (10,0) tr td Tell td (0,10) td (8,8)

In this game we have two agents who simultaneously choose

Let's compare the decision to "tell" to the decision to be "silent".

No matter what the other agent does, you are always better off choosing "tell". As a result we say the strategy "tell" strictly dominates "silent".

If under some circumstances the agent is indifferent to the strategy and another, then the strategy only weakly dominates.

So one way to solve a game is to choose dominating strategies. However an agent may not have strictly dominating strategies. Another method it to rule out strategies. If one strategy is strictly dominated for an agent, we can rule out them choosing it. This may reveal strategies which are dominant one actions of another agent can be ruled out.

If after iterations of this process we are left with only one strategy for each agent, we say this is a Von Neumann solution, an analytic solution.

But what if there are still multiple options? Consider

table.table.table-bordered thead tr th th Opera th Tell tbody tr td Opera td (10,5) td (0,0) tr td Football td (0,0) td (5,10)

And

table.table.table-bordered thead tr th th Rock th Paper th Scissors tbody tr td Rock td (0,0) td (-1,1) td (1,-1) tr td Paper td (1,-1) td (0,0) td (-1,1) tr td Scissors td (-1,1) td (1,-1) td (0,0)

In both of these there is no strategy which is always better to follow, even weakly. But these games are very different. In the former, if agents agree to both got to football, or both to opera, neither would be better off by defecting. In the second example there is no such "Nash equilibrium".

This is relevant for considering how to expand the game. In the former example a couple can talk to each other and coordinate actions. For example one agent could commit to going to the football, and the other agent would rationally join.

In the latter no such coordination is beneficial.

In the context of the game, the player can instead of choosing a pure strategy such as "rock", which may not always be appropriate, choose a mixed strategy.

For example a player could choose each of the 3 moves $\frac{1}{3}$ of the time.

Nash equilibrium

Minimax strategy

Pure/mixed strategy

14.3 Hedonic games

14.3.1 Hedonic games

Introduction

Core stability

Chapter 15

Sequential games

15.1 Sequential games

15.1.1 Sequential games

Introduction

A game can have multiple rounds. For example an agent could offer a trade, and the other agent could choose to accept or reject the trade. As later agents know the other choices, and the earlier agents know their choices will be observed, the games can change.

This doesn't change games with pure strategies, but does affect those with mixed strategies. For example, even if prisoners could see earch other in the prisoners dilemma we would still get the same outcome. The last agent still prefers to "tell", and earlier agents know this and have no reason to not also "tell".

But consider the football/opera game. Here the first mover is better off, and there is a pure strategy, while in the rock paper scissors game the first mover loses.

We can solve more complex games backwards. As the actions in the last round of a game have no impact on others, they can be solved separately. Agents then know what the outcome will be if an outcome is arrived at, and can treat that "subgame" as a pay-off.

This method is known as backwards induction.

One-round sequential games

Backwards induction

Zero-sum games

Subgame perfect equilibrium

Nash equilibrium

Chapter 16

Deep sequential games

16.1 Efficient search

16.1.1 Alpha-beta pruning

Pruning

Alpha-beta pruning

We can use pruning to reduce the number of nodes we search.

Alpha-beta pruning is a method for pruning. For each node we maintain two values:

- Alpha. The lower bound on the maximum value of the children.
- Beta. The upper bound on the minimum value of the children.

Consider a player node with a score of 2, which has a neighbour. This neighbour has a

We can know for sure we don't need to explore some paths. for example consider the mini player. there is a node we know the minimax value of a node is 2, and there is a neighbour with one child with a value of 3, then

like DFS, but keep track of:

- alpha (largest value for max across viewed children)
- initialise to $+/-\infty$
- initialise to ∞

Propagate α and β down during search. prune where $\alpha \geq \beta$

Update α and β by propagating upwards.

Only update alpha on max nodes, beta on min node

Ordering matters. if it's worst, then no pruning. want an ordering with lots of pruning p can: p + do shallow nodes p + order node so best are done first p + domain knowledge heuristics (chess: capture, threaten, forward, backwards)

In practice can get time down to $O(b^{\frac{m}{2}})$.

Use heuristic. deep blue uses 6000

16.1.2 Late Move Reductions (LMR)

Introduction

If the tree is explored in an efficient order alpha-beta pruning is more effective.

16.2 Partial evaluations

16.2.1 Heuristics

16.2.2 Identifying heuristics

16.3 Monte-Carlo Tree Search

16.3.1 Monte-Carlo Tree Search

Introduction

In search tree, each node has wins/total.

So start with just root in 0/0.

Algo:

- Start at root
- Take n choices to arrive at node which has not been explored (or until w/l state)
- play randomly from there
- Back prob up (eg if win, then 1/1 for path back to root, or 0/1 if loss)

Wins/simulation count

So deterministic when choosing paths, up until play randomly.

16.3.2 Upper Confidence bound 1 applied to Trees (UCT)

Introduction

Way to choose paths

$$\frac{w}{n} + c\sqrt{\frac{\ln N}{n}}$$

w is number of wins from node chosen

n is number of simulations from node chosen

N is number of simulations that have happened this many layers in

16.3.3 Defining board games

A board is described by:

- The layout of the board
- Whose turn it is
- Move history

These can be represented as nodes the root node being the start of the game, and each branch being a move taken.

The number of possible nodes can quickly become very large, as in chess and go.

16.4 Training with reinforcement learning

16.4.1 Reinforcement learning for combinatorial games

16.5 Other

16.5.1 Processing visual observations using neural networks

Introduction

We can use CNNs.

The output can be the values for each action.

16.5.2 Experience replay and catastrophic interference

16.5.3 Training with supervised learning

eg human chess games

16.6 Other

16.6.1 Minimax

Games

Games are different to search. In a search algorithm we are looking for a sequence of actions. In a game we are looking for a reaction function. Unlike in seach, there are other players.

We can use iterative deepening search.

Heuristics

The search space can be too big to look through all the nodes.

Rather than look for win states, we evaluate a non-terminal state using heuristics.

Stochastic games

Can use expectminimax

For max node, return highest expectminimax of children

For min node —

For chance node, average of children weightted

Minimax: two players, max min

Max goes first, maximises results. min minimises results

A node's minimax value is the best outcome against best player.

To find optimal strategy, depth first search of game tree.

Propagate minimax values up tree as terminal nodes are discovered

If a state is terminal, its value is utility of state

If a state is max node, highest value of children

If a state is min node, lowest value of children

Minimax is optimal, complete (for finite trees)

Chapter 17

Repeated episodic games

17.1 Finitely repeated games

17.1.1 Discounting

17.2 Infinitely repeated games

17.2.1 Strategies for the iterated prisoner's dilemma

Tit-for-tat

Grim trigger

Chapter 18

Tree search

18.1 Tree search algorithms

18.1.1 Depth-limited search

Depth-first search with a limit. This is useful if we know the solution is shallower than limit l.

Informed: No

Time:

Space:

Complete:

Optimal:

18.1.2 Iterative deepening depth-limited search

Does a depth-limited search to a layer L, increases the layer and starts again. The repeats are a waste, but earlier layers are much cheaper.

Informed: No

Time:

Space:

Complete:

Optimal:

Part VI

Game theory under uncertainty

Chapter 19

Games with imperfect or incomplete information

19.1 Games with incomplete information

19.2 Games with imperfect information

19.3 Bayesian games

Chapter 20

Stochastic games

20.1 Bayesian games

20.1.1 Bayesian games

20.1.2 The common prior assumption

20.1.3 Signalling

20.1.4 Bayesian Nash equilibrium

20.1.5 Perfect Bayesian equilibrium

20.2 Stochastic games

20.2.1 Stochastic games

Generalisation of Markov Decision Processes.

20.2.2 Markov perfect equilibrium

20.2.3 Expectiminimax

20.3 Partially observable stochastic games

20.3.1 Partially observable stochastic games

Generalisation of reinforcement learning.

Part VII

Multi-agent systems