AI: Principles & Techniques

Course Manual

Johan Kwisthout
Donders Center for Cognition / Artificial Intelligence
Radboud University Nijmegen
Email: johan.kwisthout@donders.ru.nl

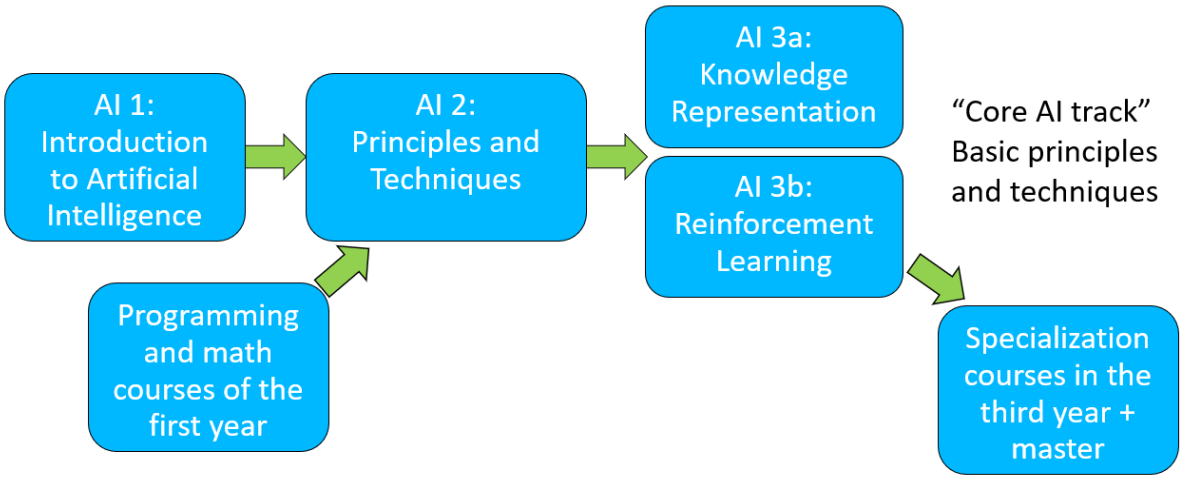
2023–2024 Fall Semester

Course manual and learning objectives

Content and place of the course

This course introduces students to several important aspects of the symbolic approach to Artificial Intelligence. During the course, students will gain insight into those important issues from the "classic" AI that are relevant to modern developments. The main themes of the course are: complexity of problems and algorithms, search strategies and algorithms, game search, optimization and scheduling algorithms, planning problems, reasoning with uncertainty (Bayesian networks), and decision making.

The figure below shows the location of the course in the AI bachelor programme and explicates what it builds on and for what courses it offers preliminaries:



Note that this course is also part of the AI pre-master programme, the AI minor of the Computing Science curriculum, and part of the international exchange courses.

An important part of this course is the actual implementation and exploration of algorithms in Python or Java (whichever is preferred) to increase understanding. Once you have completed the course, you will be able to apply these techniques to simple practical examples.

Practical information

The course consists of a number of blocks, each describing a particular topic in (classical/symbolic) AI, namely Search, Planning and Optimization, Bayesian Networks, and Decision Making. The introduction lecture describes computational complexity, which is an over-arching aspect of every problem and algorithm discussed in the course. Every block (including the introduction) starts with so-called **learning objectives** that describe what you should learn, know, and do in this block. Read these before, during, and after the block to get an idea of what you will study, what you should know and be able to demonstrate, and to prepare for the exam. The learning objectives give a fine-grained set of objectives for each block of what is expected from you at the exam. Each block has a set of **exercises** that you can use to practice these objectives. In the slides I will regularly point out relevant exercises that practice the topic just discussed. A subset of these exercises are designated as **send-in exercises** that can be submitted via Brightspace at the end of each block and for which you will receive feedback from the TAs. Submission of these exercises is optional. The deadlines are helpful to structure your study and the feedback will help you prepare for the exam.

There are two partial **exams** (one for the material covered in the Complexity, Search, and Planning and Optimization part; one for the Bayesian Networks and Decision Making part) and one Questions and Answers lecture per exam where we practice with a previous exam, rehearse material and answer questions. There are three **programming assignments** in the course that are designed to be made in teams of two students (individual submission is possible but will likely increase the workload!). There are weekly **lab sessions** planned where you can ask for guidance with implementation issues, ask feedback etc. The sessions are on-campus but support will also be available on-line via Discord. The programming assignments will take time - allocate that time and start early! We put them online early to have ample time available to fit working on them in your study programme (considering other courses in this semester). Use that wisely to have an evenly distributed study load. Note that the grading criteria (rubrics) will be published per assignment: use those to guide your efforts.

The following **examination criteria** hold:

- For both the two partial exams (E1,E2) and the three assignments (A1,A2,A3) you need to score at least 5.0.
- Any missing or below-threshold part (exam or assignment) needs to be retaken.
- If all parts are at least 5.0, the final grade is $F = (0.35 \times E1 + 0.35 \times E2 + 0.1 \times A1 + 0.1 \times A2 + 0.1 \times A3)$. In other words, both partial exams each count for 35%, and each of the assignments count for 10%.

- If $F < 5.5$, you *need* to retake some or all of the parts in order to get a passing grade.
- If $F \geq 5.5$, you *may* retake a part at will, but note that any retaken part supersedes the previous grade for that part and may result in failing the course altogether, so care should be taken!
- Resits for the partial exams are in Q2 and Q3 (see course schedule). Deadline for re-submission (or late submission) of assignments is listed in the course schedule. The exam resit in Q3 is not yet scheduled. Exam locations are to be decided.
- Any re-submission of a graded assignment or a late submission of an assignment will have 8.0 as a maximum grade, to compensate for the extra time or feedback as compared to assignments that were submitted before the deadline or that scored high without additional feedback.

Please make sure you fully understand these criteria and ask me if any confusion arises.

The exams are closed-book. Both exams will consist of a number (typically 6 to 7) of open questions. Practice exams are available; also, the exercises are typical for the level and content of the exam. Note that not all topics can be covered every year so some variance is to be expected. A (traditional, non-graphical) calculator is allowed at both exams.

If you took this course last year

You can reuse passing (≥ 5.0) grades from previous years. If you were enrolled last year in this course, I will add those grades to the Brightspace grade center a week after the course has started (to cover for late subscribers). In case you wish to partake in the exam again or resubmit an assignment (despite an in principle sufficient grade) those grades will override the grade in the grade center. Grades from 2020-2021 or earlier (when the course code was BKI212a rather than BKI259) can *not* be reused, as the assignments and exams have substantially changed.

No actions are needed from your side, unless you spot an error or missing grade. In either case please contact me by email. Note that these grades will only be entered one week after the course starts, so please take that into account.

Expected prior knowledge

While there are no formal prerequisites, this course assumes a basic background in programming, mathematics (notably discrete mathematics and probability theory), and introductory AI, as taught in the first year of the BSc AI programme. Students in the AI pre-master programme for life and social sciences that lack the necessary programming skills when this course is taught are allowed to submit the programming assignments later in their programme. We will ensure that online support is available for your questions. Anticipate that you will likely need to spend more time studying for this course without the above background.

Learning Objectives

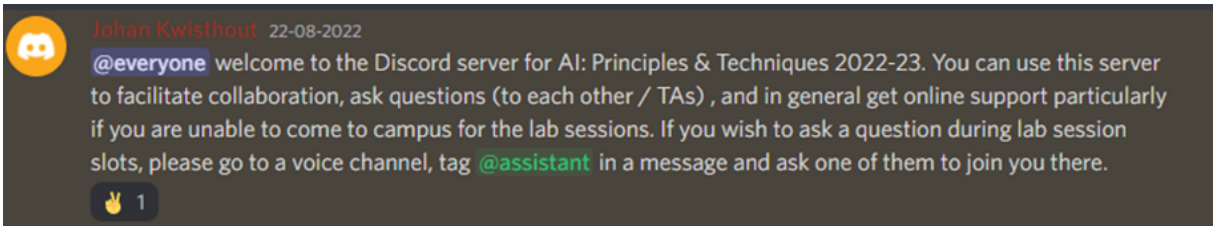
After completing this course you:

- Understand, and are able to reason with, basic aspects of computational complexity theory such as NP-hardness and reductions.
- Understand, and are able to apply and implement, the following search algorithms/strategies:
  - Prim's and Kruskal's algorithm for finding a minimum spanning tree;
  - The Floyd-Fulkerson algorithm for computing the maximum flow in a network;
  - The Simulated Annealing strategy for local search.
  - The minimax strategy and alpha-beta pruning approach in game search.
  - The Monte Carlo tree search strategy for stochastic game search;
- Understand, and be able to apply and reason with, the following aspects from planning and optimization:
  - Constraint satisfaction, including heuristics and complexity;
  - Planning representation and solution strategies such as feature-based, STRIPS, forward and means-end planning;
  - Linear combinatorial optimization, linear programming, *optional: Satisfiability solving)*;
- Understand, and be able to compute and reason with probabilities, independences, preferences, and utilities.
- Understand, and be able to apply and implement, the Variable Elimination algorithm for inference and MAP in Bayesian Networks and decision making in Decision Networks;
- Understand approximation strategies, Hidden Markov Models, Naive Bayes classifiers, and learning in Bayesian Networks;
- Understand, and be able to reason with Decision Networks and (Partially Observable) Markov Decision Processes;
- Understand Bellman equations, value and policy iterations, the Markov property, discount factor, and reward function;
- Are able to write a scientific report describing algorithm implementations and parameter explorations.

See also the detailed learning objectives per topic later in these notes!

Using Discord

For this course the online support is facilitated using Discord.



Enroll in the Discord server here: `https://discord.gg/ttRCrFVpzN`.

Mutual expectations

At Radboud University we adhere to particular *codes of conduct* which, among other things, describe mutual expectations on the appropriate behaviour among students and between students and staff (including TAs). The shortest summary of this behaviour may be: be respectful and inclusive. Radboud University is an international university and particularly the AI programme hosts many international students with a diverse background. Being respectful and inclusive includes that we do not belittle, address inappropriately, harass, or make jokes about (e.g.) gender, sexual orientation, ethnic background, social-economic status; even if this seems innocent to you, it can have big impact to the addressee. Students may expect from staff that they treat them without prejudice and that assessment is fully and only based on test and assignment results. Staff may expect from students that they take their own responsibility in mastering the material and that they are approached respectfully. If you feel any of these mutual expectations are violated, please contact a confidential advisor: `https://www.ru.nl/en/students/services/guidance-and-advice/counsellors/radboud-university-confidential-advisors`.

Communication

All formal information from staff to students is posted via Brightspace. Preferred channels of communication are as follows.

- Content-related questions about the lecture: to Johan, in the break / after the lecture or by email;
- Admin-related questions about registration, use of systems etc.:
    1. check the STIP website (`https://www.ru.nl/socialsciences/stip/`) to see whether your question is answered there;
    2. if question is about registration or exam: ask STIP by email or in person;

3. otherwise: ask Johan by email or in person.

- Questions about the assignments: at first check with your TA, by email or via Discord or during working groups; if they cannot answer, email Johan;
- Suggestions, criticism, or feedback on the lecture: feel free to approach Johan; if you feel uncomfortable to do so directly, ask your TA to relay your feedback;
- Course evaluation: your anonymous feedback on all aspects of the course at the course evaluation (after the course) is *really* appreciated! Please help us, and future students, to continuously improve the course based on your feedback. The degree programme committee (DPC) will formulate an advice to the course coordinator based on the evaluation. You can see what happened with that advice;
- Study delays, missing background, personal situations: contact your student advisor and discuss this with them;
- Confidential complaints: if you experience inappropriate behaviour in whatever sense, please contact the confidential advisor: `https://www.ru.nl/en/students/services/guidance-and-advice/counsellors/radboud-university-confidential-advisors`.

Other practical stuff

The lectures are scheduled on-campus and will be recorded and available after class, without restrictions. Particularly in the first week, seating in the lecture hall may be tight.

course coordinator & examiner

Dr Johan Kwisthout	`johan.kwisthout@donders.ru.nl`

teaching assistants (TAs)

Tim Hiemstra	`tim.hiemstra@ru.nl`
Max de Boer-Blazdell	`max.deboer-blazdell@ru.nl`
Herman Adriaensen	`herman.adriaensen@ru.nl`
Teun Hanraets	`teun.hanraets@ru.nl`
Siena Vergeer	`siena.vergeer@ru.nl`
Razvan Ciopraga	`razvan.ciopraga@ru.nl`
Dionique van Haren	`dionique.vanharen@ru.nl`
Luc van der Gun	`luc.vandergun@ru.nl`
Mats Robben	`mats.robben@ru.nl`
Justin Snelders	`justin.snelders@ru.nl`
Anh Khoa Nguyen	`anhkhoa.nguyen@ru.nl`

Learning task 0: writing a programming report

Background

As part of this course you will implement algorithms, experiment with them, and report about your implementation and your experimental findings. This chapter gives you some pointers. There is an exercise available for you to write a short report on given implementations (of two related sorting algorithms) and some experimental results. You can use this exercise to practice, and get feedback, on how to write a programming & experimentation report.

Learning objectives

After completing this task you have a good understanding of the purpose and contents of a programming & experimentation report, as will be required in the three programming assignments.

Instruction

In the lecture notes on complexity theory you'll see two $\mathcal{O}(n^2)$ sorting algorithms: Insertion Sort and Selection Sort. While their worst-case asymptotic behaviour is similar, they differ in their best-case behaviour. In the gitlab folder `https://gitlab.socsci.ru.nl/j.kwisthout/sorting-simulations` you will find implementations in C++, Java, and Python, as well as a .csv input file with arrays and an output file with results. In this exercise you will write a report based on the software and the results. A (good) example programming report from a different assignment in previous years is available. Note that this report does not follow the below section headers in all details and the reflection and work division is missing – these are later additions to the assignments.

A programming and experimentation report

A programming and experimentation report is typically written using the so-called IMRAD method (abbreviating Introduction, Method, Results, Analysis, Discussion) for empirical papers, but fine-tuned as the method is not so much an experimental design but a piece of software. Furthermore, in software experimentation there is typically no statistical analysis of the results, as the results are typically derived from benchmark inputs. So, the Method section is normally further subdivided into Specification, Design, Implementation, and Testing; and the Analysis section normally is missing. Finally, the report is concluded with a Conclusion and Reflection section which, among others, gives a specification of the work division (if you did the assignment in a team), a reflection on the process, and possible improvements for future work.

Introduction

Sketch the context of the work so that the reader understand what they are reading. In this case: write that you will empirically compare two sorting algorithms on a set of inputs to get an idea on the difference between worst-case and best-case running times for both algorithms. This is quite similar as the Introduction in the example report.

Specification

Specify the task at hand, with the assignment text as your guideline. Note that the algorithms are already given in this example exercise, but normally you would describe what the software should do (in this case: read in a comma-separated values (.csv) file with arrays, one per row of the file, sort the array using Selection Sort and Insertion Sort, count the array assignments and comparisons, and return the sorted array as well as the measures). This specification is in the first part of the Methods in the example report.

Design

Specify here your software design. Again, the algorithm is already given, but here you would specify the different classes and data structures you use, packages used for I/O and visualisation etc., and the pseudo code for existing algorithms that you will implement. In this case you would specify that you separate the sorting from the I/O (reading in the .csv file, writing down the results), you would give the pseudo-code for the sorting algorithms, etc. The middle part of the Methods in the example report contains this information.

Implementation

Give (only) the crucial part of the implementation here, preferably with line numbers, and explain what it does without literally repeating the code. Highlight important aspects and

don't repeat trivial translations from the pseudo-code. This is in the last part of the Methods in the example report.

Testing

Explain how you will test and experiment on the implementations in order to get a good idea of 1) their correct working and 2) the empirical results you are interested in. In this case you would describe the contents of the .csv file (extremes, such as already sorted arrays, arrays sorted in opposite order, uniform arrays; as well as several randomly sorted arrays) and how you would interpret the results. Oftentimes the assignment will ask you for particular empirical questions that you can summarize here.

Results

Write up the results in a clear and easy to follow way. Use a table and if meaningful, a graph, to summarize results. Give the raw facts here, not the interpretation of the facts. For example, a table like this could be useful:

	Insertion		Selection	
	Assignments	Comparisons	Assignments	Comparisons
Already sorted	…	…	…	…
…	…	…	…	…

(by the way, this is the LaTeX code for this table):

```
\begin{tabular}{c|c|c|c|c}
& \multicolumn{2}{c|}{Insertion} & \multicolumn{2}{c}{Selection} \\
& Assignments & Comparisons & Assignments & Comparisons \\
\hline
Already sorted & \ldots & \ldots & \ldots & \ldots \\
\ldots & \ldots & \ldots & \ldots & \ldots \\
\end{tabular}
```

Discussion

In the discussion you interpret the results. What do you see and what does it mean (spoiler: Selection Sort has the same number of comparisons and assignments for every n-sized array, independent of the order, whereas Insertion Sort is sensitive to the order). The example report could still be improved somewhat by more clearly separating raw results (running times) and their interpretation.

Conclusion and Reflection

Conclude the report. Summarize the main findings, mirroring the introduction (in this case: Selection Sort has the same worst-case and best-case run-time complexity, whereas Insertion Sort is linear time in the best case). Reflect on the assignment: what obstacles did you encounter? Any interesting observations? What could still be improved?

From this year on it is **c**ompulsory to give an overview of the work division. Any specific information can go here (e.g., building on existing partial implementation of one of the team members, someone dropping out or joining late, etc.). As a general rule: all team members should contribute to both coding and writing the report!

Plagiarism & Fraud

By submitting your report, including the work division, you declare that the work you take credit for (explicitly or implicitly, by not mentioning sources) is your own. Usage of AI tools such as ChatGPT to assist with writing reports or code is explicitly prohibited and is considered to be plagiarism as per the Radboud policy. Violations of this (or suspects of violation) will be handled to the examination board. Make sure to read and understand the faculty's definition of plagiarism: `https://www.ru.nl/socialsciences/stip/faculty-study-information/study-information/fraud-plagiarism/`. If uncertain, ask the TAs for guidelines.

Products

- A practice report, to be handed in (recommended) to receive feedback on content and writing.

Learning task 1: Complexity, P and NP

Background

A limiting factor in designing artificial intelligence (or understanding natural intelligence) is the *computational complexity* of the problems that are to be solved. This complexity (in time, space, or other resources) is a fundamental constraint on what can and what cannot be tractably done in any computing device, be it a brain or a computer. In this lecture we focus on some of the basic aspects of the complexity of problems and algorithms. This is essential for understanding how to cope with intractability (by constraining the applicability of the algorithm to a subset of all possible instances, by approximation / heuristics, or otherwise).

Learning objectives

After completing this task you

- Understand the difference between the complexity of an algorithm and the complexity of a problem;
- Understand the difference between best case, worst case and average case complexity measures;
- Understand the basic notion of the complexity classes P and NP, their relation, and NP-hardness;
- Be aware of intractability issues when dealing with problems and algorithms met in artificial intelligence and cognitive science;
- Understand the notion of a polynomial-time reduction to relate problems complexity-wise and be able to explain the consequences of the existence of such a reduction from problem A to problem B;
- Appreciate the relevance of complexity analysis for both practical AI engineering and theorizing about cognitive computational models;
- Have some insight in what to do when a problem turns out to be NP-hard.

Instruction

- There is no chapter in Poole and Mackworth related to this lecture. I wrote some lecture notes (available on Brightspace) to remedy this, which are part of the exam material. Additional material to read is widely available on the Internet and in the Library. Here are some pointers:
  - Russell and Norvig Appendix A.1 (Complexity Analysis and O() Notation)
  - Garey and Johnson (1979). Computers and Intractability: A Guide to the Theory of NP-completeness. Old, but still canonical introduction.
  - Wikipedia: `http://en.wikipedia.org/wiki/Computational_complexity_theory`
- Make exercises 1: Complexity, P and NP and (optional) hand in the homework subset of them before the deadline.

Products

- Answers to the exercises.

Reflection

Can you:

- Estimate the (best, worst, average) case complexity of an algorithm using the Big O notation?
- Decide if a problem belongs to the classes P or NP?
- Explain what it means that a problem is NP-hard?
- Describe the notions NP, NP-hard, NP-complete, polynomial-time reduction?
- Sketch a Venn-diagram illustrating the possible relationships between the various classes?

Learning task 2: Search

Background

In this block we will expand on the search algorithms you encountered in Intro to AI (i.e., uninformed search and informed search - see the book, section 3.5 and 3.6.1). We will introduce two important search algorithms in graphs that are intended for often-occurring real world problems: finding a minimum spanning tree (the most efficient tree connecting all nodes in a network) and finding the maximum flow (the maximal throughput over a network with limited capacity). Furthermore we will elaborate more on local heuristic search, focusing on simulated annealing, and on game search, focusing on the MiniMax algorithm and Monte Carlo tree search.

Learning objectives

After completing this task you will be able to

- Understand and apply the algorithms of Prim and of Kruskal for constructing a MST from a graph;
- Understand and apply the Ford-Fulkerson method for finding the maximum flow in a flow graph;
- Understand and apply local search with simulated annealing;
- Understand and apply MiniMax, alpha-beta pruning, and Monte Carlo tree search;
- Understand and deal with some complexity aspects of the algorithms.

Instruction

- Study the lecture slides for the Search block;
- Study section 4.7 on local search, particularly the section on simulated annealing;

- (optional) Read the corresponding chapters (24, 27) in Introduction to Algorithms, 3rd edition;
- (optional) Search the internet for texts on Minimal Spanning Trees, Maximum Flow, Game Search, and the data structures they use;
- Watch the knowledge clip by John Levine: `https://www.youtube.com/watch?v=zp3VMe0Jpf8` (Alpha-beta pruning)
- Make the exercises: Network Flow, MST, local search and game search and (optional) hand in the homework subset of them before the deadline;
- Carefully read the instructions on how to write a programming and experimentation report (see Brightspace);
- Make Assignment 1 (Four in a Row) before the deadline in the schedule.

Products

- Answers to the (hand-in) exercises (optional);
- Implementation + report of Four in a Row (Assignment 1);

Reflection

Can you explain and apply:

- The notions of a 'cut that respects A', and of a 'light safe edge'?
- Prim's and Kruskal's algorithm for solving the Minimum Spanning Tree problem?
- The *min cut, max flow* theorem?
- The Floyd-Fulkerson method for computing maximum flows?
- The notions of local search, hill climbing, random restarts, random walks, simulated annealing?
- The basics of game search, i.e., MiniMax, alpha-beta pruning, and Monte Carlo game search?

Learning task 3: Planning and optimization

Background

Good planning is essential for good performance of operating factories, handling logistics for military campaigns, robots finding their way in complex environments, etc. Planning is an important research topic in AI. Planning is a problem with high complexity. Heuristic approaches are needed to make planning more efficient.

Learning objectives

After completing this task you

- Know why planning is complex and what factors determine the complexity; can work with feature-based representations and STRIPS based representations for classical planning problems;
- Have experience with search-based planning and heuristics for some benchmark problems for classical planning problems;
- Have applied forward and regression planning on small planning problems;
- Know how to construct a partial-order plan and to use it in scheduling;
- Understand the basics of combinatorial optimization problems, including basic complexity;
- Can translate an optimization problem into an integer linear program;
- Can solve a simple constraint satisfaction problem and know concepts as arc consistency, MRV and LCV heuristics;
- Understand the notion of tree-width as a measure of complexity of many problems and can answer simple questions about tree-width.

Instruction

- Read and study
  - Poole and Mackworth, Section 3.8.1 (Branch & Bound), 4.3-5 (Constraint satisfaction), Chapter 8 (Planning);
  - The slides.
  - (optional) read Russell and Norvig, Chapter 10, Section 11.3;
  - (optional) watch the knowledge clips by John Levine: `https://www.youtube.com/watch?v=_e64FiDWvqs` (constraint satisfaction intro) and `https://www.youtube.com/watch?v=4cCS8rrYT14` (AC3 algorithm)
- Make the exercises on Planning and (optional) hand in the homework subset of them before the deadline;
- Make Assignment 2 (Sudoku solving using arc consistency) before the deadline announced in the schedule;

Products

- Answers to the (hand-in) exercises (optional);
- Implementation + report of Sudoku (Assignment 2)

Reflection

Can you:

- Explain and show how forward planning / regression planning works?
- Apply scheduling principles (e.g. early/late start, slack) to a partial order plan?
- Translate a (verbal) description of a problem into a combinatorial optimization / integer linear program / CSP?
- Solve a constraint satisfaction problem and apply heuristics to speed up computation?

Learning task 4: Bayesian Networks

Background

All of the time, agents are forced to make decisions based on incomplete information. Even when an agent senses the world to find out more information, it rarely finds out the exact state of the world. A robot does not know exactly where an object is. A doctor does not know exactly what is wrong with a patient. A teacher does not know exactly what a student understands. When intelligent agents must make decisions, they have to use whatever information they have.

Important architectures and techniques for dealing with uncertainty are Bayesian networks (BNs), the variable elimination algorithm for exact inference in BNs and sampling methods for approximate inference in BNs. Naive Bayes classifiers and Noisy-or models simplify specification and reasoning; Markov chains and hidden Markov models are special BNs for reasoning with time and uncertainty.

Learning objectives

After completing this task you understand:

- That a Bayesian network is a (sparse) representation of the joint probability distribution of a set of random variables;
- The relation between independence relations in a probability distribution and their graphical depiction;
- How the joint probability distribution can be factored in a Bayesian network;
- Inference in Bayesian networks (naive and variable elimination);
- That the complexity of inference in a general BN is NP-hard;
- The role of the tree-width of a moralisation of a Bayesian network in complexity analysis;
- Naive Bayes classifiers and Noisy-or models as a simplified type of Bayesian networks;
- Hidden Markov models as a type of Bayesian networks;

- The idea of forward, rejection, and importance sampling and particle filtering as approximation techniques;
- Basic aspects of learning of Bayesian networks (structure and parameters);
- Challenges with constructing a Bayesian network with help of domain experts;
- The MAP problem in Bayesian networks.

and you can:

- Define and use marginal and conditional (in)dependence;
- Carry out variable elimination (VE) by using factor representation and using the factor operations reduction, marginalisation, multiplication, maximization;
- Learn/estimate the probability tables in a given BN when you have (enough) samples with no missing data.

Instruction

- Read and study
  - Poole and Mackworth, Chapter 8 and 10.3;
  - The slides;
  - The knowledge clip on variable elimination;
  - (Optional) Russel & Norvig, Chapter 13 and 14;
  - Interesting extra material:
    - AISpace: `http://aispace.org/bayes/` and its successor using Jupyter notebooks rather than javascript: `https://aispace2.github.io/AISpace2/index.html`
    - Bayes theorem with LEGO: `http://www.countbayesie.com/blog/2015/2/18/bayes-theorem-with-lego`
- Make the exercises on Bayesian Networks and (optional) hand in the homework subset of them before the deadline;
- Make Assignment 3 (Variable Elimination) before the deadline in the schedule.

Products

- Answers to the (hand-in) exercises (optional);
- Implementation + report of Variable Elimination (Assignment 3)

Reflection

Do you know the notions

- random variable?
- probability, marginal, conditional; probability distribution?
- joint probability distribution?
- Bayesian network (what is a / definition)?
- The Inference and MAP problems in a Bayesian network?
- independence, conditional independence, d-separation?
- D-Map, I-Map, P-Map?
- causal chain, common cause, common effect?
- factor; factor operators (product, summing out/marginalization, reduction, multiplication);
- Markov chain, Markov property;
- Hidden Markov model;
- Naive Bayes classifier;
- Noisy-or model;
- parameter and structure learning

Can you

- Explain and use Bayes' rule, product rule, chain rule?
- Answer questions on (conditional) (in)dependence of variables in a BN (both compute and read from network structure)?
- Explain and apply the variable elimination algorithm? Also for MAP (and decision networks; next chapter)? What operations on factors are needed?
- Compute the effect of elimination order on the number of computations needed?
- Explain and apply forward, rejection, importance sampling, particle filtering?

Learning task 5: Decision Making

Background

Decision making is at the heart of AI. Agents and robots should ACT upon the information they have. A sequence of actions (i.e. a plan) CAN lead to the desired goal state, but if there is uncertainty (e.g. in actions failing with some probability) the whole plan is useless. A policy is a mapping from states to actions and is more suitable for such problems. Computing optimal policies is often done via so-called value functions which represent the amount of reward which can be expected when following some policy from a particular state. Many algorithms can be found in the literature, and here we discuss two core algorithms of two particular families of algorithms: value iteration and policy iteration. Both are heavily based on MDP models and Bellman equations, which we will also cover.

Learning objectives

After completing this task you understand

- How to model simple decisions using decision networks and how to use them to compute optimal actions;
- How sequential decision making tasks can be modeled using Markov decision processes, and you can explain the main characteristics of those models;
- How uncertainty about the states can be incorporated in MPDs to obtain partially observable MPDs (POMPDs);
- What policies, value functions, optimal policies and state values are, and how they are constrained by the so-called Bellman (optimality) equations;

and you can

- Compute the utility of decisions and optimize a policy for a given decision network using a variant of variable elimination;
- Apply value iteration and policy iteration on a small example MDP

Instruction

- Read and study and use
  - Poole and Mackworth, Section 9.1, 9.2, 9.3, and 9.5;
  - The slides;
  - (optional) Russel & Norvig, Section 17.4
  - AISpace / AISpace2;
- Make the exercises on Decision Making and (optional) hand in the homework subset of them before the deadline.

Products

- Answers to the (hand-in) exercises (optional);

Reflection

Can you explain

- one-off decisions (versus sequential decisions);
- the elements of a decision network;
- variable elimination for decision networks;
- summing out and maximizing variables in a decision network;
- what policies are and how many there are for a given decision network;
- factored utility and additive value functions;
- Markov decision process (MDP) and POMDP;
- how reward functions influence behaviors;
- discount factor;
- policies and value functions (any vs. optimal versions);
- Bellman equations;
- value iteration;
- policy iteration and its two steps (evaluation and improvement).