Syllabus: Introduction to Artificial Intelligence

TU Berlin Summer University 2019 Term 3

Week 1 July 22nd-26th

	22	23	24	25	26
	Monday	Tuesday	Wednesday	Thursday	Friday
9:00 - 10:30	**Welcome Day!** Room tbc, building tbc 10:30: Orientation session 12:30-13:15: Buffet lunch 13:30-15:30: First class session 15:30-16:15: Campus Tour 16:15-16:45: Coffee & Cake	Search Problems	Search Problems	Tutorial	No class
11:00 - 12:30		Search Problems	Search Problems	Tutorial	Cultural Program
13:30 - 15:30		Project	Cultural Program	Project	
16:00 +					

Week 2 July 29th- August 2nd

	29	30	31	1	2
	Monday	Tuesday	Wednesday	Thursday	Friday
9:00 - 10:30	Problems with Games	Problems with Games	Constraint Satisfaction Problems	Constraint Satisfaction Problems	No class
11:00 - 12:30	Problems with Games	Tutorial	Constraint Satisfaction Problems	Tutorial	Cultural Program
13:30 – 15:30	Project	Project	Cultural Program	Project	
16:00 +	Cultural Program				

Week 3 August 5th- 9th

	5	6	7	8	9
	Monday	Tuesday	Wednesday	Thursday	Friday
9:00 - 10:30	Planning Problems	Planning Problems	Machine Learning Problems	Machine Learning Problems	No class
11:00 - 12:30	Planning Problems	Tutorial	Machine Learning Problems	Tutorial	
13:30 – 15:30	Project	Project	Cultural Program	Project	
16:00 +	Cultural Program				

Week 4 August 12th-16th

	12	13	14	15	16
	Monday	Tuesday	Wednesday	Thursday	Friday
9:00 - 10:30	Problems with Uncertainty	Excursion	Project Presentations	Exam	No class
11:00 - 12:30	Problems with Uncertainty	Excursion	Project Presentations	Exam Corrections	Course Review
13:30 - 15:30	Tutorial	Excursion	Project Presentations	Exam Review	**Certificates Ceremony** Lichthof, 1st floor, TU Berlin main building
16:00 +	Cultural Program				

*The cultural program timetable will be emailed to you shortly before your course starts. For more information about the cultural program, and for examples of previous schedules, head here: https://www.tu-berlin.de/menue/summer_university/cultural_program/

Classes

The course consists of the following components: lectures, tutorials, projects, etc. In the lecture classes, students will learn about AI algorithms. In the tutorial classes, students will apply AI algorithms. In the project classes, students will implement AI algorithms. The structure of the course in terms of workload is as follows:

- Lectures – 28.5 x 60 min
- Tutorials – 11 x 60 min
- Projects – 16 x 60 min
- Exams, Presentations– 10 x 60 min
- Excursion – 6.5 x 60 min
  Total: 72h

Excursions

To be announced...

Projects

- There are 5 project tasks.
- Each task is assigned to a group of 3, 4 or 5 students.
- Each group prepares a report with the results.
- Each group presents the results at the end of the course.

Assessment

Final grade = A+(H+P+F)*points

- Attendance: min. 80% attendance, 10 points
- Homework: 30%, 90 points
  (3 Assignments x 10% or 30 points)
- Project: 50%, 90 points
  (Presentation)
- Final Exam: 40%, 90 points

Grading

All participants of the TU Berlin Summer & Winter University are required to select their grading option at the time of registration. The two options available are (i) graded or (ii) pass/fail.

All participants who select option (i) graded, will receive a grade under the German grading system. The following table provides an overview of the grading system and equivalent scores for international credit transfers:

Total mark	**German grade**	**English description**
More or equal to 95	1,0	Excellent
More or equal to 90	1,3	Very good
More or equal to 85	1,7	Good
More or equal to 80	2,0	Good
More or equal to 75	2,3	Good
More or equal to 70	2,7	Satisfactory
More or equal to 65	3,0	Satisfactory
More or equal to 60	3,3	Satisfactory
More or equal to 55	3,7	Sufficient
More or equal to 50	4,0	Sufficient
Less than 50	5,0	Failed

Credit Points

ECTS is a point system and European standard developed by the Commission of the European Community. ECTS stands for European Credit Transfer System. The aim is to provide common procedures and guarantee academic recognition of studies abroad. The credit system is based on student workload. All lectures, seminars, excursions and homework count towards the workload. One point is awarded for the equivalent of 25-30 hours of workload.

Reading

Here are reading materials which will be used or referred to during the course. You are not required to read these in advance – this is for your information and reference. All sources below are available either open source, in the TU Berlin library, or will be provided to you directly by your lecturers, during the course. TU Berlin library (https://www.ub.tu-berlin.de/en/searching-for-resources/).

The following books are recommended for this course. More precise pointers will be given during the classes.

- S. J. Russell and P. Norvig (Google Inc.). Artificial Intelligence: A Modern Approach, 3rd Edition. Pearson Education, 2010 (Chapters 3, 4, 5, 6, 10, 11, 13, 14, 16, 18, 20, 21, 25).
- B. Siciliano and O. Khatib. Springer Handbook of Robotics. Springer-Verlag New York, Inc., 2008.
- S. Thrun, W. Burgard and D. Fox. Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series). The MIT Press, 2005.
- K. Apt. Principles of Constraint Programming. Cambridge: Cambridge University Press, 2003 (Chapters 2, 3, 4, 5, 6, 7, 8).

Syllabus

1. Search Problems - States, Actions, Solutions, Data structures, Strategies, Analysis
2. Uninformed Search - Breadth-First Search, Uniform-Cost Search, Depth-First Search, Depth-Limited Search, Iterative Deepening Search
3. Informed Search - Greedy Best-First, Straight Line Distance, A* Search, Iterative Deepening A*, Admissible Heuristics, Composite Heuristics
4. Games - minimax search, resource limits and heuristic evaluation, α-β pruning, stochastic games, partially observable, continuous, embodied games
5. Constraint Satisfaction Problems - Definition, CSP examples, backtracking search, local search, hill climbing, simulated annealing
6. Planning - Representations, Solutions, Planning Graph, Answer Set Programming, Decision Making
7. Planning - State-Based Search, Partial Order Planning, Planning with Propositional Logic, GOLOG
8. Machine Learning - Supervised Learning, Curve Fitting, Ockham's Razor, Decision Trees
9. Machine Learning - Unsupervised Learning, Reinforcement Learning, Perceptron, Artificial Neural Networks
10. Uncertainty - Methods, Probability, Syntax and Semantics, Inference, Independence and Bayes' Rule, Bayes Networks

Project descriptions

Project 1: Classical Logics (CL) Classical logic is an intensively studied class of formal logics. Many contemporary discussions of classical logic normally only include propositional and first-order logics. Most semantics of classical logic are bivalent, meaning that all possible denotations of propositions can be categorised as either true or false. Nevertheless, classical logic is widely used in practice (e.g. coordination between industrial robots, model checking of service robots, etc.).
The aim of this project is to train students to basic reasoning techniques in classical logic. For this purpose, the project combines both theoretical and practical aspects of classical logic. From a theoretical viewpoint, students will learn how to solve problems with various techniques such as truth-table method, resolution method, SAT, etc. From a practical viewpoint, students will use state-of-the-art software to solve problems in classical logic (e.g. minisat (http://minisat.se/)).

Project 2: Knowledge Representation and Reasoning (KRR) Knowledge representation and reasoning is a field dedicated to representing information about the world in a form that a computer system can utilize to solve complex tasks such as diagnosing a medical condition, having a dialog in a natural language or improving the vision of machines. Knowledge representation incorporates findings from psychology about how humans solve problems and represent knowledge to design formalisms that will make complex systems easier to design and build, to classical logic.
In this project, the students will learn basic algorithms for several KRR reasoning frameworks such as planning, general game-playing, answer set programming, situation calculus, etc. They will also learn how to apply such algorithms to solve practical problems. For this purpose, we will use a number of state-of-the-art reasoners (e.g. Clingo (http://potassco.sourceforge.net/)).

Project 3: Constraint Satisfaction Problems (CSPs) Constraint satisfaction problems are mathematical questions defined as a set of objects whose state must satisfy a number of constraints or limitations. CSPs represent the entities in a problem as a homogeneous collection of finite number of constraints over a given (finite or infinite) number of variables, which is then solved by constraint satisfaction methods (e.g. Boolean satisfiability problem, satisfiability modulo theories and answer set programming).
The aim of this project is to enable students to use CSP solvers to represent, solve and analyze simple practical problems. For example, one such CSP solver is MiniZinc (http://www.minizinc.org/). Students will learn how to formalize constraints and thus represent CSP problems in the ``MiniZinc'' language. They will further compare the performance of different solvers (MiniZinc and others) on various CSP problems.

Project 4: Machine Learning (ML) Machine learning is a field of computer science that uses statistical techniques to give computer systems the ability to "learn" (e.g. progressively improve performance on a specific task) with data, without being explicitly programmed. Data mining (DM) is the process of discovering patterns in data involving methods at the intersection of machine learning, statistics, and database systems.
ML and DM algorithms are found useful in many practical fields. For example, a positioning system should be able to recognize an object in space and determine its location, Also, a robot should be able to distinguish between instructions such as ``open/close door'', ``turn-on/turn-off light'', etc. In particular, students will learn how to set up randomized splits of data sets into training and test sets, run k-fold cross validation to obtain average performance measures, compare the performance of different algorithms against a base-line and each other, aggregate comparative performance scores for algorithms over a range of different datasets, propose properties of algorithms and their parameters, or datasets, which may lead to performance differences being observed, and give reasons for actual observed performance differences in terms of properties of algorithms, parameter settings or datasets.

Project 5: Automated Reasoning (AR) Automated reasoning is an area of computer science and mathematical logic dedicated to understanding different aspects of reasoning. The study of automated reasoning helps produce computer programs that allow computers to reason completely, or nearly completely, automatically. AR has connections with AI, theoretical computer science, and even philosophy. Automated theorem proving (also known as ATP or automated deduction) is a subfield of automated reasoning and mathematical logic dealing with proving mathematical theorems by computer programs.
The aim of this project is to train students to common techniques for automated reasoning. For example, students will learn logical reasoning methods such as resolution and sequent calculi. They will have to implement their own theorem prover by using some logic programming language such as SWI Prolog or Strawberry Prolog. Students will also run experiments to evaluate the performance of their provers.