CS221: Artificial Intelligence: Principles and Techniques

Stanford / Spring 2019-2020

[Calendar] [Coursework] [Schedule]

Logistics

COVID-19 update: CS221 will be offered online Spring 2020. You can participate real time through Zoom. The Zoom links for lecture and section will be accessible on the Canvas course home page as well as Piazza. The course videos will also be recorded and put in the "Course Videos" tab in Canvas. Office hours will be remote using Queuestatus. Project for this course is now optional, please see Exam and Project sections for updates as well.

Time/location:

- Lectures: Mon/Wed 1:30-2:50pm
- Sections: Thurs 3:30 - 4:20pm
- Office hours: CA office hours on Zoom (links will be announced in Queuestatus); see calendar for times; see [Office Hour Logistics] for logistics.

Communication: We will use Piazza for all communications: announcements and questions related to lectures, assignments, and projects. NOTE: If you enrolled in this class on Axess, you should be added to the Piazza group automatically, within a few hours. You can also register independently; there is no access code required to join the group. SCPD students, please email scpd-gradstudents@stanford.edu or call 650-204-3984 if you need assistance.

Academic accommodations: If you need an academic accommodation based on a disability, you should initiate the request with the Office of Accessible Education (OAE). The OAE will evaluate the request, recommend accommodations, and prepare a letter for faculty. Students should contact the OAE as soon as possible and at any rate in advance of assignment deadlines, since timely notice is needed to coordinate accommodations. It is the student's responsibility to reach out to the teaching staff regarding the OAE letter. Please send your letters to cs221-spr1920-staff@lists.stanford.edu by **Friday, April 24** (week 3).

Gradescope: You will submit all assignments and project milestones on Gradescope, where you will also find your grades.

Instructors:

Chelsea Finn

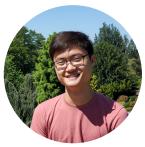
Nima Anari

Course coordinator & course assistants:

Amelie Byun
(Course Coordinator)

Cindy Jiang
(Head CA)

Andrew Tan | Bryan Kim | Dilip Arumugam | Haoshen Hong | Jenni





Jerry Qu | Marcus Pålsson | Prerna Dhareshwar | Rohan Sampath | Will De


David Lin | Sumith Kulal

Content

What is this course about? What do web search, speech recognition, face recognition, machine translation, autonomous driving, and automatic scheduling have in common? These are all complex real-world problems, and the goal of artificial intelligence (AI) is to tackle these with rigorous mathematical tools. In this course, you will learn the foundational principles that drive these applications and practice implementing some of these systems. Specific topics include machine learning, search, game playing, Markov decision processes, constraint satisfaction, graphical models, and logic. The main goal of the course is to equip you with the tools to tackle new AI problems you might encounter in life.

Prerequisites: This course is fast-paced and covers a lot of ground, so it is important that you have a solid foundation on both the theoretical and empirical fronts. You should have taken the following classes (or their equivalents):

- Programming (CS 106A, CS 106B, CS 107)
- Discrete math (CS 103)
- Probability (CS 109)
- Algorithms (CS 161)

Reading: There is no required textbook for this class, and you should be able to learn everything from the lecture notes and homeworks. However, if you would like to pursue more advanced topics or get another perspective on the same material, here are some books:

- Russell and Norvig. Artificial Intelligence: A Modern Approach. A comprehensive reference for all the AI topics that we will cover.
- Koller and Friedman. Probabilistic Graphical Models. Covers factor graphs and Bayesian networks (this is the textbook for CS228).
- Sutton and Barto. Reinforcement Learning: An Introduction. Covers Markov decision processes and reinforcement learning. Available free online.
- Hastie, Tibshirani, and Friedman. The elements of statistical learning. Covers machine learning. Available free online.
- Tsang. Foundations of constraint satisfaction. Covers constraint satisfaction problems. Available free online.

Bear in mind that some of these books can be quite dense and use different notation terminology, so it might take some effort to connect up with the material from class.

Video Access Disclaimer: This class will be given in Zoom. For your convenience, you can access recordings by logging into the course Canvas site. These recordings might be reused in other Stanford courses, viewed by other Stanford students, faculty, or staff, or used for other education and research purposes. If you have questions, please contact a member of the teaching team.

Zoom Instructions

Installing Zoom

- **Linux**:
  - Go to downloads and download appropriate Linux package from 'Zoom Client for Linux'.
  - Open package (e.g. with 'Ubuntu Software Center' or other appropriate application) and install.
- **Windows**:
  - Go to Stanford Zoom and click 'Launch Zoom'.
  - Click 'host meeting'; nothing will launch but there will a link to 'download & run Zoom'.
  - Click on 'download & run Zoom' to download 'Zoom_launcher.exe'.
  - Run 'Zoom_launcher.exe' to install.

Best Practices for Zoom
Please be aware that the lecture is being recorded and will be uploaded to the course Canvas page.

- **Set up:**
  - Download the Zoom Desktop App (Stanford IT)
  - If you have unstable internet, sometimes it helps to turn your camera off.
- **During Lecture:**
  - You will be muted for the duration of the lecture.

- Please send your questions to the chat function of the Zoom. The TA will keep track of the questions and ask the questions on your behalf.

Office Hour Logistics

CA office hours are completely remote this quarter. See the calendar for the schedule.
Students will need to register an account on QueueStatus. When you wish to join the queue, click "Sign Up" at the CS 221 queue. Be sure to enter your email when you "Sign Up"; this is a way for the CA to contact you. You will need to install Zoom to video call with the CA. See instructions above. You will be able to access the Zoom meeting through the web client (the CA hosting office hours will send you the link via the Queuestatus chat), but we recommend downloading the Zoom software as well.

Coursework

Grading

Per Stanford Faculty Senate policy, all spring quarter courses are now S/NC, and all students enrolling in this course will receive a S/NC grade. This course will still satisfy requirements as if taken for a letter grade for CS-MS requirements, CS-BS requirements, CS-Minor requirements, and the SoE requirements for the CS major.

Since the project is optional, we are providing two grading schemes:

- With project: homework 40%, midterm 30%, project 30%
- Without project: homework 60%, midterm 40%

We will take the maximum of these two so that everyone will get the better of their grades under each grading scheme.

- **Homeworks**: There will be weekly homeworks with both written and programming parts. Each homework is centered around an application and will also deepen your understanding of the theoretical concepts. Homework 5 (Pac-Man) will have a competition component; winners will receive extra credit. Here are all the homework deadlines:

  - Foundations [foundations] (zip) (template) (solutions) (due **Tue Apr 14**)
  - Sentiment classification [sentiment] (zip) (template) (solutions) (due **Tue Apr 21**)
  - Text reconstruction [reconstruct] (zip) (template) (solutions) (due **Tue Apr 28**)
  - Blackjack [blackjack] (zip) (template) (solutions) (due **Tue May 5**)
  - Pac-Man [pacman] (zip) (template) (solutions) (due **Tue May 12**)
  - Course scheduling [scheduling] (zip) (template) (solutions) (due **Tue May 19**)
  - Car tracking [car] (zip) (template) (solutions) (due **Tue May 26**)
  - From Language to Logic [logic] (zip) (template) (solutions) (due **Tue May 26**)

- **Exam**: The exam is a written exam that will test your knowledge and problem-solving skills on all preceding lectures and homeworks. The midterm exam will only cover material up to lecture in 5/20. If you have a major conflict (e.g., an academic conference), you should let us know privately on Piazza by **Thurs May 14**.

  Date: Tuesday, June 2
  Length: 3 hours, with a 15 minute leeway for scanning and/or uploading
  Format: The exam will be available for 24 hours, from 12AM PDT (midnight) to 11:59PM PDT on June 2.
  Students can choose any 3hr 15m block of time within that window to take the exam. The exam must be submitted by 11:59PM.
  The exam is open-book and will be distributed and submitted through Gradescope. It will be time stamped to ensure that all students take it within the 3 hour timeframe.

  - 2019 exam [solutions]
  - 2018 exam [solutions]
  - 2017 exam [solutions]

- **(Optional) Project**: The final project provides an opportunity for you to use the tools from class to build something interesting of your choice. Projects should be done in groups of up to four. The project will be something that you work on throughout the course and we have set up some milestones to help you along the way:

  - Project proposal [p-proposal] (due **Thu Apr 30**)
  - Project progress report [p-progress] (due **Thu May 21**)
  - Project final report [p-final] (due **Mon Jun 8**)

  See the project page for more details.

  Regardless of the group size, all groups must submit the same basic amount of work detailed in each milestone and will be graded on the same criteria. Although we allow 1-2 person project proposals, we encourage groups of 3+ members. The reason we encourage students to form teams of 3+ is that, in our experience, this size usually fits best the expectations for the CS 221 projects. We expect the team to submit a completed project (even for team of 1 or 2), so keep in mind that all projects require to spend a decent minimum effort towards gathering data, and setting up the infrastructure to reach some form of result. A 3-person team can be share these tasks much better, allowing the team to focus a lot more on the interesting stuff, e.g. results and discussion.

  For inspiration, we have made some previous CS221 projects available for viewing.

- **Piazza**: You will be awarded with up to 2% extra credit if you answer other students' questions in a substantial and helpful way.

Assignments

Written assignments: Homeworks should be written up clearly and succinctly; you may lose points if your answers are unclear or unnecessarily complicated. Here is an example of what we are looking for. You are encouraged to use LaTeX to writeup your homeworks (here's a template), but this is not a requirement. You will receive **one (1) bonus point** for submitting a typed written assignment (e.g. LaTeX, Microsoft Word). To receive this point, you must select the first page of your submission for the bonus question in Gradescope. We will accept scanned handwritten assignments but they will not receive the bonus point.

Programming assignments: The grader runs on Python 3.6.9, which is not guaranteed to work with older versions (Python 2.7). Please use Python 3 to develop your code.

The programming assignments are designed to be run in **GNU/Linux** environments. Most or all of the grading code may incidentally work on other systems such as MacOS or Windows, and students may optionally choose to do most of their development in one of these alternative environments. However, no technical support will be provided for issues that only arise on an alternative environment. Moreover, no matter what environment is used during development, students must confirm that their code (specifically, the student's `submission.py`) runs on the Gradescope autograder.

The submitted code will **not** be graded if it has one of the following issues:

- The original `grader.py` script (operating on the submitted `submission.py`) **may not exit normally** if you use calls such as `quit()`, `exit()`, `sys.exit()`, and `os._exit()`. Also note that Python packages outside the standard library are not guaranteed to work. This includes packages like numpy, scikit-learn, and pandas.
- The code reads external resources other than the files given in the assignment.
- The code is malicious. This is considered a violation of the honor code. The score of the assignment will be zero (0) and the incident will be reported to the Office of Judicial Affairs.

Collaboration policy and honor code: You are free to form study groups and discuss homeworks and projects. However, you must write up homeworks and code from scratch independently, and you must acknowledge in your submission all the students you discussed with. The following are considered to be honor code violations:

- Looking at the writeup or code of another student.
- Showing your writeup or code to another student.
- Discussing homework problems in such detail that your solution (writeup or code) is almost identical to another student's answer.
- Uploading your writeup or code to a public repository (e.g. github, bitbucket, pastebin) so that it can be accessed by other students.
- Looking at solutions from previous years' homeworks - either official or written up by another student or on a public repository.

When debugging code together, you are only allowed to look at the input-output behavior of each other's programs (so you should write good test cases!). It is important to remember that even if you didn't copy but just gave another student your solution, you are still violating the honor code, so please be careful. We periodically run similarity-detection software over all submitted student programs, including programs from past quarters and any solutions found online on public websites. Anyone violating the honor code will be referred to the Office of Judicial Affairs. If you feel like you made a mistake (it can happen, especially under time pressure!), please reach out to the instructor or the head CA; the consequences will be much less severe than if we approach you.

Submission

Electronic Submission: All assignments are due at **11pm (23:00, not 23:59) Pacific time** on the due date.

- Assignments are submitted through Gradescope. **Do not submit your assignment via email.** If anything goes wrong, please ask a question on Piazza or contact a course assistant.
  If you need to sign up for a Gradescope account, please use your @stanford.edu email address.
- You can submit as many times as you'd like until the deadline: we will only grade the last submission.
- **Submit early to make sure your submission runs properly on the Gradescope servers.**
- Gradescope will run `grader.py` on the programming questions and give you feedback on non-hidden test cases. **You are responsible for checking that your program runs properly on these cases. You will not get credit otherwise.**
- Partial work is better than not submitting any work.

For assignments with a programming component, we will automatically sanity check your code in some basic test cases, but we will grade your code on additional test cases. **Important**: just because you pass the basic test cases, you are by no means guaranteed to get full credit on the other, hidden test cases, so you should test the program more thoroughly yourself!

Unless the assignment instructs otherwise, all of your code modifications should be in `submission.py` and all of your written answers in `<assignment ID>.pdf`. Upload the former to Gradescope under the "Programming" section, and the latter under the "Written" section.
For the project milestones, make sure **all members** of your group are included in the submission.

Late days: An assignment is $\lceil d \rceil$ days late if it is turned in d days past the due date (note that this means if you are 1 second late, $d = 1$ and it is 1 day late). You have **seven (7) late days** in total that can be distributed among the assignments (except for p-poster and p-final) without penalty. After that, the maximum possible grade is **decreased by 25% each day** (so the best you can do with $d = 1$ is 75%). As an example, if you are out of late days and submit one day late, a 90 will be capped at a 75, but a 72 will not be changed. Note that we will only allow a max of $d = 2$ late days *per assignment*, though, so if $d > 2$ then we will not accept your submission. Gradescope is set up to accept written and programming submissions separately. Late days are calculated per-assignment, with the number of late days used depending on the later submission. So, if the programming part is 1 day late and the written part is 2 days late, then that will count as using 2 late days.

Regrades: If you believe that the course staff made an objective error in grading, then you may submit a regrade request for the written part of your assignment. Remember that even if the grading seems harsh to you, the same rubric was used for everyone for fairness, so this is not sufficient justification for a regrade. It is also helpful to cross-check your answer against the released solutions. If you still choose to submit a regrade request, click the corresponding question on Gradescope, then click the "Request Regrade" button at the bottom. Any requests submitted over email or in person will be ignored. Regrade requests for a particular assignment are due one week after the grades are returned. Note that we may regrade your entire submission, so that depending on your submission you may actually lose more points than you gain.

Schedule

Day	Topic	Slides	Events	Deadlines
	[Introduction (Chelsea)] *What is this class about?*			
Mon Apr 6 (week 1)	Overview of course Optimization	[one page] [text outline] [pdf:1pp,6pp] [code] [supplementary]	[foundations (zip) (template)] **out**	
	[Machine learning (Chelsea)] *Don't manually code it up, learn it from examples...*			
Wed Apr 8	Linear classification Loss minimization Stochastic gradient descent	[one page] [text outline] [pdf:1pp,6pp] [code] [supplementary]		
Thu Apr 9	Section: optimization, probability, Python (review)	[slides]		
Mon Apr 13 (week 2)	Features and non-linearity Neural networks, nearest neighbors	[one page] [text outline] [pdf:1pp,6pp] [supplementary]	[sentiment (zip) (template)] **out**	
Tue Apr 14				[foundations (zip) (template)] **due**
Wed Apr 15	Generalization Unsupervised learning, K-means	[one page] [text outline] [pdf:1pp,6pp] [code] [supplementary]		
Thu Apr 16	Section: Backpropagation, nearest neighbors and past exam problems	[slides] [annotated slides]		
	[Search (Nima)] *Problem solving as finding paths in graphs...*			
Mon Apr 20 (week 3)	Tree search Dynamic programming, uniform cost search	[one page] [text outline] [pdf:1pp,6pp] [code] [supplementary]	[reconstruct (zip) (template)] **out** p-proposal (survey) **out**	
Tue Apr 21				[sentiment (zip) (template)] **due**
Wed Apr 22	A*, consistent heuristics Relaxation	[one page] [text outline] [pdf:1pp,6pp] [code] [supplementary]		
Thu Apr 23	Section: UCS, Dynamic Programming, A*	[slides] [annotated slides]		

[Markov decision processes (Chelsea)]

When nature intervenes randomly...

Fri Apr 24			Drop date	
Mon Apr 27 (week 4)	MDPs, policy evaluation, value iteration	[one page] [text outline] [pdf:1pp,6pp] [code] [supplementary]	[blackjack (zip) (template)] **out**	
Tue Apr 28				[reconstruct (zip) (template)] **due**
Wed Apr 29	Reinforcement learning Monte Carlo, SARSA, Q-learning Exploration/exploitation, function approximation	[one page] [text outline] [pdf:1pp,6pp] [supplementary]		
Thu Apr 30	Section: MDPs and Reinforcement Learning	[slides]		p-proposal (survey) **due**

[Game playing (Nima)]

When an adversary intervenes...

Mon May 4 (week 5)	Minimax, expectimax Evaluation functions Alpha-beta pruning	[one page] [text outline] [pdf:1pp,6pp] [code] [supplementary]	[pacman (zip) (template)] **out**	
Tue May 5				[blackjack (zip) (template)] **due**
Wed May 6	TD learning Game theory	[one page] [text outline] [pdf:1pp,6pp] [supplementary]		
Thu May 7	Section: Games	[slides]		

[Constraint satisfaction problems (Nima)]

Problem solving as assigning variables (with constraints)...

Mon May 11 (week 6)	Factor graphs Backtracking search Dynamic ordering, arc consistency	[one page] [text outline] [pdf:1pp,6pp] [demo] [supplementary]	[scheduling (zip) (template)] **out** p-progress **out**	
Tue May 12				[pacman (zip) (template)] **due**
Wed May 13	Beam search, local search Conditional independence, variable elimination	[one page] [text outline] [pdf:1pp,6pp] [supplementary]		
Thu May	Section: CSPs	[slides]		

14		[annotated slides]		

[Bayesian networks (Nima)]

Representing uncertainty with probabilities...

Mon May 18 (week 7)	Probabilistic inference Hidden Markov models	[one page] [text outline] [pdf:1pp,6pp] [supplementary]	[car (zip) (template)] **out** [logic (zip) (template)] **out**	
Tue May 19				[scheduling (zip) (template)] **due**
Wed May 20	Forward-backward Particle filtering Gibbs sampling	[one page] [text outline] [pdf:1pp,6pp] [supplementary]		
Thu May 21	Section: Bayesian networks	[slides] [annotated slides]		p-progress **due**
Mon May 25 (week 8)	(Memorial Day — no class)			
Tue May 26				[car (zip) (template)] **due** [logic (zip) (template)] **due**
Wed May 27	Learning Bayesian networks Laplace smoothing Expectation Maximization	[one page] [text outline] [pdf:1pp,6pp] [code] [supplementary]		
Thu May 28	Section: Exam review 1 Reflex and State Based Models Skilling Aud 3:30-4:20pm			
Fri May 29	Section: Exam review 2 Variable Based Models Skilling Aud 3:30-4:20pm	[slides] [annotated slides]		
Sat May 30	Extra: Review of Variable Elimination and Treewidth	[slides]		

[Logic (Nima)]

More expressive models...

Mon Jun 1 (week 9)	Syntax versus semantics Propositional logic Horn clauses	[one page] [text outline] [pdf:1pp,6pp] [supplementary]	p-final **out**	
Tue Jun 2			**Exam**	
Wed Jun 3	First-order logic Resolution	[one page] [text outline] [pdf:1pp,6pp] [supplementary]		

	[Conclusion (Chelsea)] *Reflections and prospects...*		
Mon Jun 8 (week 10)	Deep learning autoencoders, CNNs, RNNs	[one page] [text outline] [pdf:1pp,6pp]	p-final **due**
Wed Jun 10	Summary, future of AI		