What grade would you like to receive in this course? What grade do you honestly believe you deserve? Justify your answers.

- S
- S
- I have completed all assignments, reviewed all lecture videos, and practiced the recommended exercises and submitted these to the TA on time.

What have you learned in this course? Explain in more abstract, high-level terms (don't just list technical details from lecture slides). Has your view of AI changed from what it was before taking this class (in what way)?

I have deepened my understanding in several areas:

- Fundamental concepts behind modern neural networks
  - We can think of neural networks as a series expansion in a mathematical sense, they are able to approximate any arbitrary function, given enough layers, i.e., Cybenko and the Universal Approximation Theorem.
  - Learning is a process by which we optimize some sort of loss function. The idea is to get the loss as close to 0 as possible. Backpropagation, due to its efficiency in computing the gradient of the loss with respect to the weights of any arbitrarily sized network for each single input-output instance, has made it the favored algorithm for this process.
- The advantages of pythonic coding
  - This course has demonstrated the robustness of object-oriented programming. While python has grown in popularity due to its widely used libraries, this course demonstrated in first-principles, i.e., building piece by piece a tensor library with little to no imports, how we can utilize classes and objects in a high-level language to create dynamic computation graphs and automatically compute partial derivatives.

My view of AI has not changed much, as I have been interested in this topic before taking the class. Many of the ideas presented were familiar to me, some not. I'm grateful that this course does not handwave the mathematical concepts needed to understand neural networks, while providing a broad historical overview of the field. It has given me a better appreciation for the multidisciplinary nature of artificial intelligence.

What would you like to learn in an introductory AI course (e.g. if you take this course in a future semester)? Are you interested in more pragmatic coding with current industry libraries or in understanding the fundamental principles and theory? Do you prefer to zoom out and focus less on the mathematical/programmatic details of AI systems (instead discussing recent trends on a higher level of abstraction), or do you prefer to zoom in and focus more closely on mathematics and coding of foundational algorithms?

I would like to learn coding in more detail, and have at least one assignment where we make use of PyTorch. Students should at a minimum know how use the GitHub interface and be comfortable coding in whatever environment they prefer. Below are some additional suggestions and my justifications.

- Personally, I would like a more applied version of the assignments. For example, having students design their own learning problem (either in groups or individually) and submitting it in lieu of a final examination.

    - Our assignments were defined problems. This was advantageous because the application was clear and the datasets were pre-processed, allowing students to focus on the evaluation of their models. However, rephrasing a real-life problem into an implementable learning problem is a practical skill I would have liked to develop, namely, finding suitable data on my own and cleaning/processing it so that it could be used as training data.
- Introduce programming assignments with detailed explanations, where almost every part of the code is explained in detail.
    - The extreme detail given for the template code in Assignment 2 was enough that a studious novice could complete the assignment. Personally, I found Assignment 1 was not presented as straightforward.
    - Consider uploading videos for coding explanations even if the classes are given live. Having access to high quality videos with coding explanations was extremely useful, and the teacher/TA will not have to waste time re-explaining things covered in the lecture. This is my personal experience, but I needed to listen to lectures regarding coding several times over while taking notes to fully absorb the content. This wouldn't have been possible in a live class.