Jacob Andreas

Teaching Statement

I've been excited about teaching computer science for nearly as long as I've been excited about computer science. One of my favorite memories from my undergraduate days is of teaching the diagonalization lecture in the introductory theory class while the instructor was traveling. The prospect of continuing to teach is a large part of the appeal of an academic career. Since coming to Berkeley, I've taken advantage of a variety of different teaching opportunities:

- Running a course: In the summer of 2016, I served as the instructor of record for Berkeley's *Introduction to Artificial Intelligence* (CS 188). I co-taught lectures four days a week, designed a new unit on optimization, managed a staff of six graduate and undergraduate teaching assistants, and interacted with campus administration about space booking, enrollment, and grading.
- Managing scale: The summer before I taught it myself, I was a teaching assistant for *Intro to AI* during the 650-student fall offering. As a teaching assistant, I led discussion sections, held office hours, designed exams, led a redesign of one of the six course projects, and helped manage the large software infrastructure used to support the course.
- Teaching across disciplines: Throughout my time at Berkeley, I've been invited to give guest lectures in the linguistics department, the school of information sciences, and in computer science classes on computer vision and robotics. I prepared new lectures for audiences without a background in one of either language or computer science.
- Mentoring researchers: I worked closely with several younger graduate students in my group to guide them through their first research projects. This involved discussing possible project ideas, training them in the required technical material, designing models and experiments, editing paper drafts, and giving feedback on practice talks.

From these experiences I've learned how to be an effective educator for many different kinds of students and educational settings. But there are unifying themes: the most effective teaching is *personal* and *concrete*. Even in large lecture courses, individual attention is the key to keeping students engaged; well-designed examples, demonstrations, and projects are more effective than even the most lucid presentation of abstract material.

Making teaching personal

When I taught *Intro to AI*, a group of five or six students in the 160-person summer session would consistently stick around after every lecture to ask questions. Often these questions were about the course material; other times they ranged from current research topics to the broader impacts of artificial intelligence on society. Though these Q&A sessions were not part of the official class schedule, I would sometimes stay an hour past the end of lecture to host them. The students who participated had a wide variety of comfort levels with the course material—some had already mastered the content on the syllabus and wanted to talk about how to generalize it to different problems; other students still struggled with the core concepts and needed extra help understanding.

Several of the people in this group went on to do AI research while still undergraduates, some publishing at top conferences. The student I'm most proud of really struggled through the first few weeks of the course, but by the end had such good command of the material that he was picked by the notoriously competitive selection process to work as an undergraduate teaching assistant the following spring.

These individual interactions also made me a more effective instructor for the rest of the class. It's not possible to provide one-on-one attention to every person in a class of six hundred. But seeing in detail what a small subset of students already understand and what they struggle with makes it possible to respond with changes to the lesson plan for the larger group in near-real-time. Watching how students react to particular examples or explanations up-close helps to present them effectively in a lecture format to the full class. Ultimately, some students will always need more help than others; identifying these students and making sure they're comfortable seeking help is essential to making sure they learn.

Making teaching concrete

One of the best pieces of advice I've received about preparing talks and papers is to always put concrete examples before discussions of the underlying abstraction. A version of this advice applies on all scales: Within individual lectures, it's always best to show pictures and work through specific cases before presenting the underlying math. Within courses, projects should be released before all the lectures needed to finish them, so that students have a sense for the purpose of the tools they're being given. At the level of whole curricula, classes that provide hands-on problem-solving experience are most effective before more theory- and analysis-focused classes.

One of the most important functions of an undergraduate artificial intelligence curriculum is to make sure that students have a firm grasp of fundamental concepts to help structure their later thinking. At the same time, the availability of high-quality public software packages for learning and modeling means that students can start working on practical problems right away, and gradually learn the underlying theory in parallel with the intuition they develop from writing real code.

This kind of concreteness is a powerful tool for getting students involved in computer science early, and for building inclusive classrooms. People who are not initially excited about core algorithmic tools often become excited once they see what these tools can do; judicious use of libraries and scaffolding avoids privileging students who arrive with extra engineering experience.

Future teaching plans

I'm excited to teach about both the applications and the techniques that are core to my research: natural language processing problems of all kinds, as well as methodology for both deep learning and structured prediction. I know from experience how much fun it is to teach a general undergraduate artificial intelligence lecture, and I'm enthusiastic about staying involved in undergraduate teaching. I enjoy presenting on a range of other subjects—both core CS topics like discrete math and algorithms, and interdisciplinary interactions with linguistics, the humanities, and other AI application areas. Teaching has been an important part of my graduate experience, and I look forward to continuing it in the future.