Teaching Statement of Geoff Pleiss

Some of the most influential moments of my Ph.D. came from interacting with students taking their first machine learning course. One student in particular, a new graduate student in the environmental engineering department, gushed about the course I was assisting. "It has completely changed my research direction," she told me. Machine learning methods were prominently featured in her dissertation work, which focused on modeling and controlling water levels in municipal reservoirs. Experiences like this illustrate 1) the impact a single course can have on a student's academic trajectory, and 2) how teaching—especially to an academically diverse set of students—inadvertently opens the door to new applications of machine learning. As computer scientists, we are in a unique position: our field is becoming increasingly relevant to a growing cohort across various STEM disciplines. This presents an exciting opportunity and an important responsibility. My goal is to foster a welcoming and engaging environment that develops students into capable, innovative, and responsible practitioners of computer science and machine learning.

Summary of experience. At Cornell, I was a TA in two introductory machine learning courses and one graduate advanced topics course, where I had the opportunity to teach tutorials, design course materials, and advise group projects. I have given substitute/guest lectures for multiple professors, including in courses with over 600 students. As part of my postdoc appointment, I have mentored several Ph.D. students and my PI has trusted me to independently advise research projects. I also maintain a widely used open source machine learning software framework, which—though not a traditional education venue—has enabled me to disseminate tutorials and easy-to-use software to thousands around the globe. These experiences have formulated my approach and philosophy towards teaching and mentoring, which I describe in detail below.

Teaching Philosophy

Introductory courses should be accessible to anyone. Every student should leave an introductory course capable of applying the material, with excitement and motivation to do so. However, this is complicated by the growing enrollment in computational courses, as students are entering from a variety of backgrounds with different levels of mathematical and computational fluency. My office hours as a TA would often be packed with students who were stuck in all sorts of different ways—whether confused by a mathematical derivation from two lectures ago, or frustrated by a bug in their implementation of an algorithm. While the scale of these courses poses a challenge to personalized instruction, there are several strategies I will employ to maintain understanding and motivation en masse. First, I have seen that *frequent, short, and informal assessments* provide students with an opportunity to evaluate their strengths and weaknesses while at the same time providing valuable rapid feedback to me as an teacher. In the same vein, *lectures and course schedules should be fluid*, leaving ample opportunity to adapt if there is significant confusion or a desire for more in-depth instruction. Finally, courses should emphasize practical application through *assignments that use "real-world" software and datasets*. As a TA, I observed that student self-efficacy was highest after learning tools like scikit-learn or PyTorch, as these experiences were concrete manifestations of the knowledge they gained in the course.

Moreover, I believe that nontraditional teaching venues, such as open source software, will play an increasing role in computer science education. As a maintainer for GPyTorch—a library for Gaussian process models—I have developed software that has introduced a large global audience to this class of models. I also regularly design tutorials and respond to forum questions from users of all abilities, aiding in their continued learning.

Advanced courses should develop self-sufficient learners. Upper-level courses and mentorship share the same overarching goal: developing students into self-sufficient learners who are capable of mastering advanced material, identifying interesting research questions, and leading open-ended projects. In the advanced topics course I assisted at Cornell, we accomplished this by having students present papers, lead discussions, and tackle projects that simulate the research environment. Students in these settings are (to a large extent) responsible for their own learning experience, and my role is to guide them towards fruitful ideas and away from potential roadblocks.

Mentorship is a gradual process towards autonomy. My postdoc position at Columbia has prepared me to be a capable advisor, as my PI has trusted me to mentor Ph.D. research projects in a largely independent capacity. For example, I worked with two junior Ph.D. students and one masters student who were new to Gaussian process methods. I experienced the delicate balance of guiding students into a new research topic: the project must be well-scaffolded, but ownership must ultimately lie with the students. To that end, I helped scope research questions, provided relevant background literature, and suggested useful experiments—all while letting the students dictate goals and progress. This experience culminated in an ICML paper and several ongoing followup projects. With the foundations from this first project in place, I now take an increasingly hands-off role as these students pursue their own research agendas.

The transition from well-encapsulated courses to open-ended problems is challenging for many students, and my goal is to make this process exciting and rewarding rather than overwhelming. I have found that *frequent (but potentially short) check-ins* are better than longer infrequent meetings, as the former cultivate momentum and catch potential problems early. Most important is *establishing a supportive culture*, where students and I can openly discuss their strengths and areas for improvement. I have found that sharing my own stories of growth and setbacks in research has helped new students feel comfortable to reach out when they are struggling.

Courses should couple technical content and broader impacts. Machine learning and computer science should not be taught in a vacuum where technical content is divorced from its real-world impact. Computational methods are increasingly prominent throughout many applications, and consequently our teaching experiences indirectly shape the role of computer science and machine learning in society as a whole. To that end, I believe that courses should include discussions around the potential and practical consequences of these technologies, focusing on the environmental cost of computation, as well as issues of privacy, safety, and equity. I will supplement my own knowledge of these issues with materials from domain experts in these areas.

Subject Area and Potential Courses

My academic background and pedagogical approaches have prepared me to teach various courses at all levels. Specifically, I am prepared to instruct introductory computer science courses as well as deep learning, probabilistic modeling, and general machine learning courses at the undergraduate and graduate level. I am also interested in teaching specialized courses focusing on Bayesian nonparametrics and the intersection of machine learning and numerical linear algebra.