Teaching Statement

Suraj Rampure

The role of an instructor in a large university course is akin to that of an orchestra conductor, responsible for coordinating several moving parts to achieve a common goal: supporting and inspiring students while effectively conveying course material. In this statement, I will describe my perspective on key components of successful large-scale computing courses.

Lectures with a narrative

A student's intellectual journey with course concepts begins in lecture. I once read that the best teachers view teaching as a performance art and I have tried my best to embody that philosophy. I try to make lecture feel like a well-crafted story rather than a sequence of technical jargon.

I break lesson plans into four sections. First, I motivate the topic of the given lesson. In lectures that are particularly technical, it's easy for students to lose sight of the bigger picture; a high-level roadmap and real-world examples at the start remind students that the end result will be worthwhile. Second, I dive deep into the underlying ideas of the topic at hand, which becomes its own sub-narrative. Then, after struggling through the weeds, I show students how we can apply the theory we just discussed to tangible examples, such as using `LogisticRegression` in `sklearn` to classify basketball shots as makes or misses. To conclude, I summarize and provide context regarding how this lecture relates to the next lecture. I have found that this structure keeps students intrigued from start to finish, whether lessons are delivered in-person or asynchronously.

Active learning during lecture

It is crucial that active learning activities are interspersed throughout each lecture, to ensure that students are engaged and aren't simply passively listening. When teaching Introduction to Mathematical Thinking, an 80-student introductory discrete math course, I would regularly present a problem and give students a few minutes to discuss with their peers. Then, I would call upon various students to walk me through their solutions, write their thoughts directly on my slides, and work with them to correct their solutions if necessary. Not only was this more engaging than presenting pre-written solutions, but it also ensured that the slides that I posted contained solutions that students would understand, since students wrote them (see this lecture for several examples of such problems). Another consequence of this approach was that it enabled students to create study groups that could take future, larger classes together, as students built connections with others who they frequently worked with in class.

In Summer 2020 I taught Data 100, a 330-student intermediate data science course, entirely remotely. My co-instructor and I decided to present lecture content asynchronously; each lecture consisted of several videos with a target length of six minutes[1] interspersed with Quick Checks (QCs), which were short quizzes asking questions about the preceding video, with immediate feedback and detailed answers (example).

---

[1] Philip J. Guo, Juho Kim, and Rob Rubin. 2014. How Video Production Affects Student Engagement: An Empirical Study of MOOC Videos. In Proceedings of the First ACM Conference on Learning @ Scale Conference (L@S '14). Association for Computing Machinery, New York, NY, USA, 41–50. https://doi.org/10.1145/ 2556325.2566239

Most questions were multiple choice (e.g. "Which of the following lines of code correctly uses `pd.pivot_table` to create the desired result?") but some were short answer (e.g. "Compute the mean squared error of the provided model on the above training data.") and even long answer (e.g. "In your words, what is the difference between risk and empirical risk?"). A majority of students reported that QCs were helpful, as they gave them a gauge of whether or not they understood a given topic and provided them with additional resources to practice fundamentals when studying for exams.

QCs also served a complementary purpose: I regularly monitored QC aggregate responses to identify questions students commonly answered incorrectly, and used that information to prepare a supplementary synchronous lecture at the end of each week to address misconceptions with that week's lecture topics. Not many students attended these "recap" lectures – roughly 8% of the class regularly attended, though they were recorded and posted later – but those that did expressed that they were helpful in strengthening their understanding of lecture material.

Professors Fernando Pérez and Anthony Joseph, the instructors of Data 100 this fall, adopted many of our innovations from the summer, including the asynchronous lecture format with QCs and even many of the lecture videos I recorded (example). In fact, while QCs were optional in the summer, they are now mandatory in the fall.

Challenging and relevant assignments

Assignments should reinforce and extend the class narrative that is anchored by lecture. Especially in computing classes, the majority of students' learning comes from discussing and completing assignments. Therefore, it's vital that assignments are well designed.

I have found that students learn most when they apply course concepts to something that is somewhat relevant to them. In Data 100, the two assignments that students tend to enjoy the most are Trump Tweets, where they use `pandas` and `seaborn` to explore a repository of tweets by President Trump, and Spam/Ham, where they use various `sklearn` methods to create an accurate email classifier. Students particularly enjoy the latter as it contains an open-ended exploration of feature selection, in turn giving them a better understanding of the data science lifecycle. In Introduction to Mathematical Thinking, real-world examples weren't as readily available, but I chose homework problems that related current topics to what they'd learned before. For instance, I had students prove the power law for derivatives using induction, showing them that induction has the ability to prove statements they've seen in entirely separate contexts and is not some esoteric technique restricted to combinatorics and graphs.

A particular flair of assignment that I have come to enjoy creating, which I adopted from my advisor Josh Hug, is one in which students perform the same task or arrive at the same result in different ways. In Data 100, we created one such lab assignment where students found the optimal coefficients for the simple linear regression model in several ways – by using the analytical formulas from lecture, by using `scipy.optimize.minimize` on an objective function, and by using `sklearn.linear_model.LinearRegression`. Once students discover that all approaches lead to the same answer, it demystifies the black-box nature of existing utilities and instills a deeper understanding of the underlying theory.

Feedback mechanisms

It's important to give students regular mechanisms to provide feedback about a course. In Data 100 in Summer 2020, we made weekly feedback surveys mandatory, in which students would self-report their understanding of the previous week's lectures and assignments and provide input on the usefulness of various course components. We used the information students provided to make on-the-fly modifications to the course. For instance, we added more comprehensive Quick Check explanations once students said they were using them as a study tool and removed an experimental "discussion recap" section (where students would watch a worksheet walkthrough video beforehand and come with lingering questions) in favor of an additional live lab section (where students and TAs would collaboratively work on that day's lab assignment). Similarly, students appreciate when course design decisions are made transparent (e.g. "This course has a final exam instead of a final project since we didn't feel that a final project was appropriate for such a foundational, theoretical course."). Being transparent and receptive to feedback makes students feel like their voices are heard and helps keep class morale high.

Similarly, it's important from the staff perspective to provide students with immediate and frequent feedback pertaining to their progress in the course. In Summer 2020, we implemented a robust new autograding system that allowed us to return grades to students dramatically quicker than in previous semesters. In data science, it's important for students to learn how to create visualizations and interpret data and have a human give them feedback on their work; as such, Data 100 complements autograder scores with feedback on written problems by tutors.

Invested and confident teaching staff

Teaching assistants are common in computing courses of all sizes, but are particularly plentiful in large courses. One benefit of having many TAs is that everyone has a slightly different teaching style, allowing students to find the TA that best fits their learning style. Accordingly, I encourage TAs to put their own spin on material in section, while sharing resources as needed. Another benefit is that they provide several viewpoints on what students may be struggling with, from their experiences in sections and office hours. I like to start each week's TA meeting by reflecting on the previous week, giving everyone a chance to share what went well and what areas of the course need attention. Such meetings are crucial to ensuring that the course is effectively serving its students and that TAs feel a sense of ownership in the course.

First-time TAs often lack the confidence to lead a classroom discussion and feel as though they aren't qualified to be a teacher. Both as a leader of the First-Time CS TA conference and informally as a veteran on staff, I have helped new TAs overcome these hurdles by sharing with them moments in which I was vulnerable and made mistakes, and gave them tips to overcome these challenges. After taking down the course website for hours in my first semester as a TA and receiving a calm lesson on Git and GitHub Pages from the head TA, I learned to treat every mistake as a teaching and learning opportunity and to give new TAs autonomy to explore how they could impact the course. I value my mentorship of other TAs; there are few feelings in the world like hearing students praise a TA that you helped train and mentor.

In order to further their pedagogical skills and ultimately benefit students, I share with TAs best practices I have developed for interacting with students one-on-one. For instance, when interacting with students in office hours, I ask them to pinpoint exactly what they're stuck on; in the context of a programming

assignment, I ask them what they've tried and what they think their issue may be. After helping students with a very specific bug or slightly nudging them in the direction of a correct approach, students tend to feel that they arrived at the solution on their own, giving them a sense of accomplishment and independence that is important for them to maintain.

Extracurricular impact

Each semester, I strive to ensure that students leave my class enthusiastic about what they've learned and determined to apply their newfound skills to projects of their own undertaking. I believe enthusiasm is contagious; I try to share my excitement for course material in each lecture and in other interactions with students. I also make sure that students work on practical assignments in class, as discussed earlier. The impact on students can be significant, as evidenced by the following email I received from a student in Summer 2020:

> I started off the semester as an angsty Data Science major. I couldn't reach the GPA cap required to declare Computer Science as a major, and Data Science was my next best option. I was pretty upset at the prospect of data analysis courses, as I assumed data science was a bunch of stupid, labrinthyne formulas, unintelligible numbers, and ugly Excel pie charts... But in retrospect, I think this class has been one of the most rewarding courses I've ever taken in my career as a student. It was such a perfect combination of programming and mathematics, but I was blown away by the sheer amount of application that was showcased. There's rarely a lecture in my life that I've discussed with my friends, but I would find myself talking about DS100 quite often.

Since computing has applications in virtually all domains, it is important to discuss the societal implications of computing in our courses. In Data 100, I have worked with Berkeley's Human Context & Ethics program to incorporate pertinent examples and thought-provoking questions into our lectures and assignments. This helped students realize the potential impact of their work and the ethical considerations necessary when using the computing skills they've learned in the real world.

Finally, I strive to be a mentor for students outside of the classroom. I have been quite open about my interest in both the teaching community at Berkeley and pursuing teaching as a career. As a result, numerous students reach out each semester asking for advice on how to get started with undergraduate teaching and for tips on improving their teaching; many such students have eventually become teaching assistants. I have also helped students navigate the application processes for clubs, internships, and graduate schools; in doing so I have strengthened my understanding of what students are looking to get out of coursework and supported my community.

Conclusion

Teaching at scale is a juggling act of sorts. It's the role of an instructor to ensure lecture content is coherent, assignments are polished and relevant, exams are fair, TAs are effective and not overworked, and students are engaged and inspired. I have fallen in love with the act of coordinating each of these components just as much as I love the end result, which – when everything is executed correctly – is that students feel a sense of achievement.

I recognize that I have just barely scratched the surface of teaching at the university level. As an aspiring teaching faculty member, I want to continue to experiment with new classroom approaches while learning from other teachers. I also want to continue to share what I have learned about pedagogy with the broader community, in order to help advance the field of computing education.