As a computer science PhD student, I am a disciple of big data. I see no ground too sacred for statistics: I have used it to study everything from sex to Shakespeare, and earned angry retorts for these attempts to render the ineffable mathematical. At Stanford I was given, as a teenager, weapons both elegant and lethal—algorithms that could pick out the terrorists most worth targeting in a network, detect someone's dissatisfaction with the government from their online writing. Computer science is wondrous. The problem is that many people in Silicon Valley believe that it is all that matters. You see this when recruiters at career fairs make it clear they're only interested in the computer scientists; in the salary gap between engineering and non-engineering students; in the quizzical looks humanities students get when they dare to reveal their majors. I've watched brilliant computer scientists display such woeful ignorance of the populations they were studying that I laughed in their faces. I've watched military scientists present their lethal innovations with childlike enthusiasm while making no mention of whom the weapons are being used on. There are few things scarier than a scientist who can give an academic talk on how to shoot a human being but can't reason about whether you should be shooting them at all. The fact that so many computer scientists are ignorant or disdainful of non-technical approaches is worrisome because in my work, I'm constantly confronting questions that can't be answered with code. When I coded at Coursera, an online education company, I developed an algorithm that would recommend classes to people in part based on their gender. But the company decided not to use it when we discovered it would push women away from computer science classes. It turns out that this effect—where algorithms entrench societal disparities—is one that occurs in domains from criminal justice to credit scoring. This is a difficult dilemma: In criminal justice, for example, you're confronted with the fact that an algorithm that fulfills basic statistical desiderata is also a lot more likely to rate black defendants as high-risk even when they will not go on to commit another crime. I don't have a solution to this problem. I do know, however, that I won't find it in my algorithms textbook; I'm far more likely to find relevant facts in Ta-Nehisi Coates's work on systemic discrimination or Michelle Alexander's on mass incarceration. My personal coding projects have presented similarly thorny ethical questions. Should I write a computer program that will download the communications of thousands of teenagers suffering from eating disorders posted on an anorexia advice website? Write a program to post anonymous, suicidal messages on hundreds of college forums to see which colleges offer the most support? My answer to these questions, incidentally, was "no". But I considered it. And the glory and peril of computers is that they magnify the impact of your whims: an impulse becomes a program that can hurt thousands of people.

Perhaps it's more efficient to allow computer scientists to do what we're best at—writing code—and have other people regulate our products? This is insufficient. Coders push products out at blinding speed, often cloaked in industry secrecy; by the time legislation catches up, millions of people could be harmed. Ethics training is required for professionals in other fields in part because it's important for doctors and lawyers to be able to act ethically even when no one's looking over their shoulders. Further, computer scientists need to help craft regulations because they have the necessary technical expertise; it's hard to regulate algorithmic bias in word embeddings if you have no idea what a word embedding is. Here are some steps forward. Universities should start with broader training for computer science students. I contacted eight of the top undergraduate programs in computer science, and found that most do not require students to take a course on ethical and social issues in computer science (although some offer optional courses). Such courses are hard to teach well. Computer scientists often don't take them seriously, are uncomfortable with non-quantitative thinking, are overconfident because they're mathematically brilliant, or are convinced that utilitarianism is the answer to everything. But universities need to try. Professors need to scare their students, to make them feel they've been given the skills not just to get rich but to wreck lives; they need to humble them, to make them realize that however good they might be at math, there's still so much they don't know. A more socially focused curriculum would not only make coders less likely to cause harm; it might also make them more likely to do good. Top schools squander far too much of their technical talent on socially useless, high-paying pursuits like algorithmic trading. As Andrew Ng, a Stanford computer scientist, admonished a roomful of Stanford students he was trying to recruit to Coursera: "You have to ask yourself, why did I study computer science? And for a lot of students, the answer seems to be, so I can design the latest social media app...I believe we can build things that are more meaningful than that." There are many steps tech companies should take as well. Organizations should explore the social and ethical issues their products create: Google and Microsoft deserve credit for researching algorithmic discrimination, for example, and Facebook for investigating echo chambers. Make it easier for external researchers to evaluate the impacts of your products: be transparent about how your algorithms work and provide access to data under appropriate data use agreements. (Researchers also need to be allowed to audit algorithms without being prosecuted.) Ask social or ethical questions in hiring interviews, not just algorithmic ones; if hiring managers asked, students would learn how to answer them. (Microsoft's CEO was once asked, in a technical interview, what he would do if he saw a baby lying in an intersection: the obvious answer to pick up the baby did not occur to him). Companies should hire the people harmed or excluded by their products: whose faces their computer vision systems don't recognize and smiles their emojis don't capture, whose resumes they rank as less relevant and whose housing options they limit, who are mobbed by online trolls they helped organize and do little to control. Hire non-computer-scientists, and bring them in for lunchtime talks; have them challenge the worldviews of the workforce. It's possible that listening to non-computer scientists will slow the Silicon Valley machine: Diverse worldviews can produce argument. But slowing down in places where reasonable people can disagree is a good thing. In an era where even elections are won and lost on digital battlefields, tech companies need to move less fast and break fewer things.