Personal Section:

In the summer of 2021, I got an email that would change my life. I'd been attending the UCLA math circle for almost a decade at that point, and I had a good relationship with the head of the program, Prof. Oleg Gleizer. But I was surprised when he sent me an email saying, "We may have a research opportunity at USC this Summer." Prof. Salman Avestimehr of USC (University of Southern California) was doing research into distributed machine learning techniques, and he had reached out to Prof. Gleizer looking for mathematically advanced high school students for an eight-week apprenticeship.

At the time I knew very little about the inner workings of artificial intelligence, and nothing about any of the subjects Prof. Avestimehr was researching. But I'd been interested in AI for a long while and loved seeing all the new technologies being created. This was when GPT-3 had just been released, and ChatGPT was more than a year away, so the available software was very basic by today's standards: one AI could write classical music, another could replace the background of a video, another could generate realistic faces. So, when Prof. Avestimehr reassured me that he would teach me all the background knowledge I needed to know, I couldn't pass up on the opportunity to become a part of the field I'd been observing for years, and I said yes to what I thought would be a two-month project.

My first few weeks of the program consisted mostly of lectures and exercises to catch me up to speed. Every Monday, Wednesday, and Friday I watched a data-science lecture that explained general data-science concepts, from basic linear algebra to statistics. Every Thursday I worked with one of the post-docs working in the lab, Dr. Ramy Ali, to understand their existing research, working step by step from the smallest discoveries to the biggest. Finally, after almost a month, I understood what had been discovered, and what was still missing.

The technique behind most modern Artificial Intelligence systems is called “Machine Learning,” which is an attempt to approximate the way humans learn. In simplest form, machine learning repeatedly tests and adjusts a program’s parameters to perform better and better on a dataset of examples until it can solve a desired type of problem, often performing these adjustments millions or billions of times. Machine learning begins with a template program, which is just a complex program with a lot of parameters. Most of these template programs use a “neural network,” which involves processing inputs using simulated neurons controlling other neurons via controllable parameters. What’s important is that the program, which we call a “model,” has a lot of parameters (sometimes billions of parameters). We “train” this model by repeatedly testing its performance on a set of example problems, measuring each parameter’s effect on the accuracy of the model, and adjusting each parameter accordingly. We repeat this process over and over again until the model can solve the assigned task.

The biggest problem with this technique is getting enough examples. Large machine learning models trained on complex tasks can require millions, billions, or trillions of examples to train on, and obtaining those examples often involves use of a massive amount of detailed personally identifiable information which many would argue is a serious violation of people’s privacy. That’s where my research came in. The lab had been working on improving a technique called “Federated Learning,” which has the potential to keep all the training data hidden and private even while using it to train a machine learning model. I’d never heard of Federated Learning before, and it seemed impossible. How could a computer learn the patterns in a set of data without seeing the data? But it’s surprisingly simple.

Federated learning works by dividing the training process among many devices. Instead of offering their data, users of a machine learning service offer their computational power. Every training round, each user receives a copy of the current model. They then train this copy on their device, using their own data. Finally, these individually trained models are aggregated together on the central server, using an encrypted process called “secure aggregation” so that even the individually trained models are hidden from everyone except the owner of their data.

While that idea is key to a new way of approaching privacy, it was not yet completely secure. Before my research had begun, my mentors discovered a way to reverse engineer user data just by examining the resulting, aggregated models. Methods for recovering user data by examining a trained model were already well known, but the secure aggregation should have ensured that, at the very least, no data could be paired with any person. But Prof. Avestimehr’s team found that they could approximate individual models, and thus, individual data, by exploiting the fact that not all users will be available for all training rounds. Users will randomly drop out of training because they powered off their device, uninstalled the application, or any other reason. By comparing the rounds when a user is present with rounds when that user is absent and using some linear algebra to account for the change in other users, the lab was able to discover what effect each user had on the overall model, and thus approximate the data they contributed to it.

My mentors had also created a solution: group the users into batches, and only use the users in a batch if every user in it is available. But that solution has a severe problem: there often aren’t enough batches available using this method, so training must pause until more people log on. My mentors hoped I could find a better, more efficient method for protecting privacy while using federated learning.

That was a daunting task. A group of experts with PhDs had failed, and they thought I, a high-schooler with no prior experience in theoretical computer science, might have a chance. But, after I had worked through my mentors’ existing research, I began where any good mathematician begins. I played around with an example. My mentors had described their Batch Partitioning method using a type of matrix (a grid of numbers), so I began by examining a specific example of that matrix using only six users at the lowest possible level of privacy. I made changes to see what those changes would break. I looked for patterns and tried to extend those patterns. I poked and I prodded until, finally, I found something. I discovered, for the specific example I was looking at, I could double the number of rows in the matrix (essentially doubling the efficiency of the algorithm) by batching the users in a second way, and it still maintained the privacy of the users. I had found somewhere to start.

This initial insight happened relatively quickly. But turning that insight into a solution required two major steps: first, I had to figure out how to turn my singular example into a general formula that would work for any number of users, and any desired level of privacy. And second, I had to prove my formula actually satisfies the level of privacy I claimed it does.

Like any big problem, in mathematics or otherwise, the way to tackle both these tasks was to take them step by step; slowly building up from my singular example to the full solution. My initial example used the lowest possible level of privacy, so I began by focusing solely on that level of privacy. The first step, extending the example into a general formula, wasn't too difficult. Because the level of privacy was the same, I just had to do the same thing with more users. That just left the second task: prove my extension would still protect the privacy of individual users.

Proving something is possible is very easy; just do it. Proving something is impossible, however, is much trickier. In order to prove the privacy of users with my technique, I had to show that it's impossible to combine aggregated models to isolate a single user. Again, I searched for patterns to use, symmetries I could follow, and tried different operations until something worked. In this case, I got to use one of my favorite proof techniques: proof by contradiction. I showed that if you did isolate down to one user, eliminating the influence of everything else, that user’s model must be multiplied by a factor of 0. In other words, you didn’t isolate down to one user, you isolated down to no users.

At this point, the eight-week program was mostly up, and there didn’t seem to be any more progress I could make in the time I had left. I made a presentation, showed my mentors my work, and was ready to be done. I had discovered an improved technique in a specific case, and I was sure my mentors could expand from there. But they had other ideas. My mentors said, if I continued working with them and figured out how to extend my technique to any desired level of privacy, they’d help me write and publish my work. This was a very exciting offer. I could be an author of published research before I even reached college. Besides, solving the simple case had gone relatively smoothly. How bad could the whole problem really be?

Thus began the worst months of my life. Like before, the first task of turning the example into a general formula wasn’t too difficult. I had to go through a few potential algorithms, each of which seemed to work at first, but later experimentation proved ineffective at preventing isolation of a single user. However, I eventually found an extension of my prior pattern which I couldn’t break.

So, I set on the challenging task of proving I couldn’t break it. And what a challenging task it was. The technique I used in my prior proof didn’t work here. Experimentation didn’t lead anywhere obvious. None of the theorems I knew seemed to help. My research into relevant (and less relevant) areas of mathematics turned up nothing applicable. Every day I’d find some new way to manipulate the equation, feel like I was finally making progress, and then discover I’d hit another dead end. I felt like I was repeatedly crashing a car, backing up, and slamming into the same wall again. And throughout those months, I just got more and more depressed. I wanted to quit, but I felt like I was perpetually almost done, just one breakthrough away. So I kept trying and kept trying, as I ran out of things to try.

And then, while lying on the sofa after a few particularly unproductive hours of work, an idea came into my head. Could it really be that simple? I ran back to my desk, and within an hour I had the solution. Remember when I said turning the example into a general algorithm wasn't too difficult? It turns out I was wrong, because the general algorithm I thought I found was the wrong one. I still, to this day, have no idea whether that algorithm would work or not, but it doesn't matter because I found a significantly simpler, and much better, method. My new algorithm, which I called Double Partitioning, worked by partitioning the users in two different ways, and it also doubled the number of rows. Once the idea for this new algorithm popped into my head, proving it worked was almost trivial: it was in fact the exact same proof as for the simpler case I'd done all the way back at the beginning. I had solved it, and all I had to do was look from a new perspective.

My work wasn't done, of course. Being a scientist doesn't just require doing science, you also have to explain that science to others. I had to take my proof, and all the work I'd done, and figure out how to write it down in a way that made sense. My mentors were so helpful in this phase; they understood the conventions of academic journals and academic articles, so they ensured everything I wrote was clear and understandable by other computer scientists. After another month or so of work writing and rewriting and simplifying my proofs, I had written my first ever academic paper, and I was ready to submit it to journals.

Unfortunately, our first submission to a journal was rejected during peer review. They felt that my improvement was not significant enough to be worth publishing, as the measured improvement on performance time was somewhat small. In addition, they said our standard of “privacy” was too weak; our paper assumed that a group of users was private if it was impossible to eliminate the influence of all other users, but peer review pointed out that reducing the influence of other users to some really small factor, such as one in one million, would still count as “private.” While our second submission to a different publication was accepted, those criticisms stuck in my head. Could I make the algorithm even better?

So, I got on a zoom call with one of my mentors, and we spent several hours (split over around a week) figuring out how to improve our definition of privacy without being too restrictive. My experimentation suggested that our existing algorithm followed a stronger form of privacy than I’d proven, but I needed to figure out how to articulate and prove that form. I’d pose a suggestion, my mentor would find an issue and suggest a solution, and I’d alter that solution to be better. After a few iterations, we had a definition that we felt satisfactorily represented a common understanding of “privacy,” without being too restrictive. Proving Double Partitioning also satisfied this new definition of privacy was a breeze.

We also wanted to improve the performance even more, or at least try. My mentor suggested that I try to calculate exactly how well the current method, Double Partitioning, performed, instead of just measuring it on examples. Calculating this performance ended up being very difficult. I particularly struggled with one pattern; while every row in my matrix satisfied the pattern, there were gaps which, if filled, would make the necessary calculations much simpler.

And then, I thought, why not fill those gaps? I examined my proof of Double Partitioning’s privacy, and no step required those gaps to be present. So, I removed them, and created Mixed Partitioning, my final algorithm. By filling the gaps in the pattern, I had improved the performance even more, and this time I could say exactly how much more of an improvement it was.

This final iteration of my algorithm is what got me selected as a finalist for the Regeneron STS. As you can see, getting there was a long journey full of wrong turns and dead ends. I learned so much about the research process, and I’m so thankful to my mentors at USC for giving me this opportunity and helping me through every setback.

So, if you ever have the chance to do mathematical research, learn from my mistakes, and take my advice. Start small and build up to the full problem. If you’re stuck, just play around with what you know, looking for patterns to extend or break. If you’re trying to improve one aspect of something, understand exactly why that aspect is at the level it is. And, most importantly, never assume you’re on the correct path. Even when something seems to work, keep trying other ideas. Even after you’re done and your work was published, there might be a way to make it better.

One last thing, to cap this story off. As I was going back through my old emails and notes, I discovered something funny. That improved method, Mixed Partitioning? I actually discovered it almost right away, and then forgot about it. In my week 3 presentation, when I was showing my example solution for six users, I also showed another example solution, which coincidentally is the mixed partitioning matrix for six users. I promptly forgot about that, because it was much less obvious how to extend that pattern, but it was there. So, if you’re stuck, read through your old notes. Maybe you’ve already hit on something and forgotten.