CS Ph.D. lessons to my younger self

PDF version, HTML version

I completed my CS Ph.D. from IIIT Delhi (Note the three Is) in March 2017. My Ph.D. was a lesson filled journey. Here's a letter from me to my younger self for CS Ph.D. lessons.

Dear younger self,

As you embark this Ph.D. journey, I wanted to share some lessons and experiences. While I do not claim to be able to follow this 100 %, I am more aware than I was before. I think I might have done a better job at my Ph.D. if I'd taken note of the points I am going to mention now (in no particular order).

Take care of your health

You're joining fresh after completing your undergraduate studies. You are full of energy. But, your body won't remain the same. You'll realise that there is a difference between the early and late 20s. You'll be debugging your code, and will feel it'll take only 5 minutes more, and after three hours realise you're well past your routine, your neck may hurt! Very soon, your body might hunch forward because of long sitting hours staring at your screen, and you may start gaining weight. Remember that when optimising for a happy life, you can't neglect your health for long. As one of my mentors would say, take time off. Set it on your calendar! Ensure that you get some form of exercise routinely.

By health, I mean not only your physical health but also your mental health. There'll be times that you'd be exhausted and dejected. No matter what you do, you don't seem to make any progress. You'll start developing doubts about your abilities. You'll start questioning your decision to pursue a Ph.D. Remember that you're not alone! Taking time off would help. It would also greatly help to have friends to talk and share. Don't keep all your anxieties and worries to yourself!

Meet well!

You'd be participating in a lot of meetings over the tenure of your Ph.D. Most often you'd be meeting your research advisor. You'd be very excited to discuss every possible detail of your work with your advisor. But, remember, that your advisor probably has some courses to take. They might have a group of students working under them, and have multiple projects running simultaneously. So, you'd need to plan and strategise to make the best use of the meeting time you get with your advisor. Here's a template that I tried to follow. First, set the context by giving a summary of previous meeting discussion. Setting the context will ensure that you both are on the same page. Present a brief agenda. Then, present the main findings. Have some thoughts on why your method works or does not work. Over the course of your Ph.D., you'll realise that your advisor is an expert in deducing why something works, and why something doesn't. It'll be precious for you to observe your advisor and learn and apply this yourself. In my earlier years, I'd often just go and show all my results to my advisor. You really shouldn't stop there. Think about what you should conclude from your findings. Come up with specific questions for your advisor. Ensure that you make the best use of the limited meeting time you get with your advisor.

Paper rejection is tough, but not the end of the world

It's likely that you'll have a few papers rejected during your Ph.D. I couldn't believe when I got my first paper rejected. How could it be! In anguish, I felt the reviewer didn't do a good job. I took it very personally! With more rejections, I got somewhat better at it. I also learned lessons along the way. One of the lessons that stuck with me was to not ponder too much on the reviews for 1-2 days after the paper was rejected. I wasn't able to take the reviews with an unbiased attitude anyway. After 1-2 days, I would often appreciate the points made by the reviewers. Sometimes, I wouldn't, but, then reviewing is complex, anyway! It should also be noted that of all my total submissions, only about one-third or so got accepted. Top CS conferences are very competitive, so paper rejections are likely to be more common than not!

Better emails go a long way

You'd be writing a ton of emails during your Ph.D., way more often than meeting in person. It's important that you learn the art and science of writing better emails. Poorly written emails are less likely to get a response. At the beginning of my Ph.D., I wrote poor emails. The subject line wasn't indicative of what to expect in the post, which made it incredibly hard to find relevant threads later! I often jumbled up multiple projects/topics in one email. I often wrote very long emails. Many of my emails were a brain dump and not organised well. Eventually, by observing how my advisor and other senior people wrote emails, I improved in this important skill. I think that many of the points about meetings are also applicable to emails. You need to set the context, discuss how you proceeded, and why you did so, summarise the observation, form some conclusions and questions, and if possible be specific on what inputs you need.

Embrace the scientific method

I was an engineer by training before I started my Ph.D. So, I'd be thinking like - "this seems cool. Let's try this approach. Should be fun to solve". I went this way for some time, till I had a meeting with a senior faculty. The first question he asked me was - "what's your hypothesis?". I stood dumbfounded. I realised that thus far I'd been a solution-first researcher without doing so in a scientific way. I'd try and pick up approaches, if they didn't work, move on. After a bit of rewiring, I improved my understanding and application of the scientific method. So, my work would now be more structured. I'd be wasting less time on random trails or unimportant problems. The key lesson is to have a checklist (mental or physical) of common pitfalls - like, improving the accuracy of a system from 95% to 99% on an unimportant or extremely contrived problem; or, starting data collection without much thought to the final experiments and analysis you'd plan to conduct to prove your hypothesis; or, the worst of all, having no hypothesis to start off with!

Mid-PhD. crisis time: you're not alone

I'd heard that every Ph.D. student goes through some crisis period. I thought I'd be an exception.

How wrong I was. This period occurred after I'd been academically weaned. My coursework had finished. There was little barring research to keep me busy. Paper rejections stopped surprising me. Ideas not working stopped surprising me. I started questioning if I should quit Ph.D. midway. I started looking at forums on this topic and realised that this problem was common. During this period, I kept in touch with folks who'd completed their PhDs. Everyone suggested that this is normal, and a Ph.D. wouldn't be so valuable if it weren't this difficult! I feel that staying in touch with people who could relate to me was important. It was also helpful that despite the so-called inputs not converting to output, my advisors continued encouraging me, meeting regularly, and discussing scientifically correct ways of finding solutions to the Ph.D. problem. Miraculously this phase ended and led to the most successful phase of my Ph.D. where I was able to submit and get accepted a few top-tier conference papers. The key lesson is to stick it out, don't feel alone or worthless, keep talking to cheerful people and keep reporting progress to your advisor on a regular basis.

It all boils down to addition and subtraction: Blog posts

I won't lie, I often get intimidated by research papers and the huge amount of techniques in our field. Yes, I still do. One of best teachers at my university spoke - "it all boils down to addition and subtraction." This has stuck with me. I took a programming and visualisation approach to understanding concepts in my field. At a higher level, I'd be thinking that if I had to teach this concept to someone, how would I go about it. For example, if say, I wanted to study dynamic time warping, I started with some trivial problems where I'd use the concept. On such trivial problems, it would be easy to understand what the algorithm would be doing. I'd often end up writing detailed Jupyter/IPython notebooks showing how the algorithm works, programming and visualsing the various steps. All these formed a part of my technical blog, which I would update on a regular basis. The learning in writing these blog posts was immense. While these blog posts are public, I think I am the biggest beneficiary. Not only does one gain a good understanding of the concept involved, but one also gains confidence about the subject and one's ability to understand! The key lesson is to document your learnings, understandings, and try to abstract out your specific problem and think of teaching the concept to someone who doesn't know much about your problem.

Teaching is learning

A teaching assistantship is often dreaded. Why waste my valuable time on it? It turns out, I learned a lot from my TA experiences. I realised that learning with an intention to share the learning ensured that I learned well. I didn't cut corners. Moreover, with an emphasis on teaching well, I often had to think hard about making concepts relatable. This exercise helped me a lot! In my first semester, I was a TA for an introduction to programming course. Before the course, I thought I knew Python reasonably well. After the course, I realised that now I knew it way better than earlier. Since Python turned out to be the language I did most of my research coding in, it turned out to be a great learning experience. Besides, I made a lot of friends with my students in the course! The key lesson here is somewhat related to the lesson on blog posts. Thinking how to explain stuff usually always helped me get a better understanding!

Good research is only half done!

Talks are a great way to advertise your research and get some good feedback. It's easy to go very excited and fit everything from the paper into a few slides in a 15-20 minute presentation. This is a great recipe for disaster! Good presentations often leave the audience feeling thankful that they attended the talk. Bad talks ensure that people open their laptops and start answering emails they've not replied for ages! A good talk requires preparation. I repeat. A good talk requires preparation. Even if you're a pro. Over the course of the years, I developed my style learning from my advisors, other faculties, other presenters. There were a lot of lessons involved! One of the key lessons for me was to jot down a script for my slides. While I felt I could mostly do a good job speaking without speaker notes, I think I was better with them! Of course, it took a lot of effort! Another important lesson was to deliver a talk in terms of things the audience could relate with, and thus keeping them engaged. I also maintained that the slides are there to support me and not replace me. Thus, I would never put much text into them! I'd put a lot of efforts in maintaining consistency across slides, using similar figures, conventions. All of this to ensure that the viewer doesn't lose interest!

Of course, despite all these efforts, I would usually always deliver a pathetic first talk. I was lucky that most of these first pathetic talks came in front of colleagues and not the main audience. So, "Practice! Practice! And practice!" is the mantra.

Volunteer to review papers

Your advisor would likely be reviewing a lot of papers throughout the year. Many of these would probably be in your sub-area of CS. It was an excellent exercise for me when I'd help my advisor review some of the papers. Taking part in the review process makes you appreciate the time and effort put in by the TPC and the conference management team! No one is paid for all this effort! I particularly remember excitedly telling my advisor that I'd champion one of the papers he gave me to review. He smiled and said that paper wouldn't probably go through. The discussion that followed helped me learn the traits of a good and a bad paper from a reviewer's perspective. Once you get the hang of it and do the same critical analysis of your paper, you'd be able to improve your paper!

Open source is the right way

CS research often involves writing code for our approach, some baselines, data management and analysis. This set of code leads to the generation of results and analysis in our papers. Now, when I started my Ph.D. and started working on a problem, I thought that it should be trivial to compare our work to previous literature. It turns out; I was wrong. Reproducibility or the act of generating results from previous literature is a huge challenge. Making your code reproducible, such that others can use it for their analysis and paper takes a lot of effort. I felt that it was the right thing to make my code open source and easy for others to use. One of my most cited paper came as a result of introducing more transparency, comparability, and reproducibility. I'm also not surprised that I'm the biggest benefactor by making my code good enough to be made public. The steps I took to make the code more readable, and reproducible, helped me a lot going back to my code to tweak or re-run some experiments. In the age of Github, there aren't really many excuses for not putting efforts towards - "Let's make scientific articles comparable again."

Funding for conferences/PhD scholarships

While we may crib about our stipends not being comparable to industry folks, let's not forget that a Ph.D. can be very expensive. Coming from India, the travel costs to conferences would be high! Yes, very few of the international conferences I attended happened in India. However, some research labs like MSR and Google and some government agencies provide funding for "good" conferences. Many conferences also provide economic support. I was lucky that I could cut some of the costs for my advisor and department by getting travel fundings. Various organisations also offer Ph.D. fellowships. I was fortunate to receive one from TCS. These fellowships not only allow for some industry mentors but also include financial support. It's one less thing to worry if we can get some of the finances worked out. The key lesson isn't a particularly surprising one - apply for fellowship and grants whenever you can!

Good paper writing takes time and practice

Before joining grad school, I was a great writer. After completing it, I could manage some writing. Confused? When I entered grad school did I realise how pathetic my scientific writing skills were. While I'd been speaking and writing English since I was four or so, it wasn't my first language. The red coloured paper returned by my advisor on my first draft was an eye opener (technically, it was `\color` in `LaTeX` and not a hard copy!). For someone like me who loves to learn by observation, the process of taking my first draft and seeing it turn into a well-written document courtesy my advisor was wonderful. The key lesson is to observe what makes a good draft. I came up with a structure that I would follow in almost all of my papers:

- What is the general problem?
- How can we convert it to a CS problem?
- What's the related work and where does it fail to address the problem?
- Why would our approach work? and what's the gist of it?
- What's the experimental setup I've used for evaluation?
- How well does our approach perform?
- In light of above, what can be conclude about the problem given our approach?

I'd write using such a structure in the abstract and just expand it into different sections in the paper.

Oh, and yes, don't forget Grammarly and some scripts from Matt Might! Something that really helped me was to run my drafts, even before the final version with my colleagues to get some reviews.

Accept limitations of your work and be honest about them

We've all seen those few points on the graph that make our algorithm look bad. Only if we didn't have those points, our technique would look so amazing! There may be a tendency to "avoid" the limitations. My key learning on this front has been to accept the limitations of my work and be very honest about them. Science loses if we "hide" facts. Rather, it's the limitations which make things interesting. They force us to go back and review everything. Is this an issue with our approach, or implementation, or dataset, or something more fundamental? Maintaining this integrity was always a top priority! There've been instances where paper reviewers have shown appreciation for us being clear and honest about the limitations.

Sometimes it helps to disconnect

We live in an age of data deluge. There are so many forums to network and brand your work. There are so many forums like Reddit and HackerNews to remain on top of the latest in the field. While I tried to remain up to date with the latest, it helped when I would sometimes disconnect. This used to be the way I studied in primary and secondary school. So, if any idea or interesting problem comes to mind, I would sometimes try and think it through before googling it. Similarly, if the code gives some error, I found that my programming and understanding improved if I would spend some time thinking before googling! The key lesson is to think before you search.

Is the grass really greener on the other side

As a Ph.D. student sitting in India, I'd often think how great it would have been to be a Ph.D. student at say MIT, Stanford, or some other top CS university! Maybe, I would have a "better" publication record. Procrastinating in such situations is easy. My key learning in this aspect came from a discussion I had about 15 years back in my high school when my teacher told me that it's the students who make the school. So, the key lesson was to accept that I may not be from the most well-known school in the world, but nothing stops me from doing world-class research. So, accepting what I had, and trying to change what I could was the main learning. Of course, research is highly collaborative these days. Eventually, by the time I had graduated, I had worked with people from Imperial London, Univ. of Southampton, University of Virginia, UCLA and CMU.

Invest in good hardware

I get it. Our Ph.D. stipends aren't super amazing. I chose to buy the "best" laptop I could in a reasonable budget. Why bother with Apple-like expensive laptops. I could do with a regular laptop and just install Linux on it. It turns out, I was wrong. Within two years, my laptop had a broken hinge, I had driver issues (on Linux), the laptop wasn't super light, the battery wasn't all that great. Then, it took me a hard time to move to a superior laptop. It was expensive. But, it more than made up in terms of the increased productivity. The battery would often last a work day; my back pain got better due to a lighter laptop. I was no longer afraid of updating my system. So, while I am talking about a laptop here, the lesson is more general. The cost of a "good" piece of hardware can be easily recovered many times over in terms of increased productivity.

Take care!

More to come

- Networking at conferences, discussion buddy, elevator pitch, attending thesis and faculty job talks..