From Academia to Data Science

Heinrich Hartmann https://heinrichhartmann.com/blog/data-science.html

2018-09-15 in Stemwede, Germany

Over the last years, many people have asked me: How do I enter Data Science? I don't have a definite answer to this question, but having transitioned from an academic career in pure Mathematics to working as a Data Scientist, I have made some experiences that I can share here.

Before we dive in, let's take a step back and clarify terminology.

1 What is Data Science?

Data Science is the realization that businesses needs advanced mathematics to make sense of their data. The analysis of balance sheets and order books does not involve that much mathematics. Nowadays, companies are gathering every bit of information they can get their hands on: User data, sensor data, market data, web datasets, etc. If you want to extract information from this kind of scattered, high volume data, you need to apply some more mathematics. This is what Data Science is all about: Extracting business value from data.

2 What do Data Scientist do?

The role description is still in flux, but typically the duties of a Data Scientist involve:

1. Find, explore, transform, normalize and curate datasets.
2. Develop machine learning models in Python/R.
3. Implement, debug and operate systems that run those models.
4. Talk about those models in documentation, blogs, seminars, conferences.

In smaller companies a large chunk of your time will be spent on (3). Right now it takes up more than 80% of my time. In larger organizations, this figure will will be a much lower. In particular the operational burden might rest entirely upon another team. Nevertheless, getting your models operational is critical for the the value you deliver to the company. A good understanding of what it takes to run code in production will help you to become more effective.

3 How to get there?

Trust in your Abilities. Seasoned academics bring a great skill-set to the table that computer science (CS) majors often do not have. This includes: A high tolerance for frustration. Ability to reason formally. No fear of equations. Grasp how a high-level description is reflected in formulas. The ability to manipulate mathematical models creatively. Incorporate new ideas and extend a given model. Communicate theory and results in written and oral form. Teach colleagues and peers.

The CS majors I have worked with were not able to work with equations very well. Many are scared of matrices. Getting a firm grip on mathematical models is *really* hard for CS master students. Often times, machine learning papers are just impenetrable for them.

So while CS majors do have an edge in programming, but they also have to catch up with a whole set of equally valuable skills. It's easy to forget how far you have already made it, when you are 2 years into your PhD. Don't be afraid.

Start with Simple Models. Make sure to cover the simple things first. Linear regression is a very powerful tool, that can solve a lot of problems on the spot. 90% of your models you will write will only involve basic statistics and linear regressions. Don't apply machine sophisticated machine learning algorithms before you tried a bunch of if statements.

Also simple models can be surprisingly effective if trained on enough data. (Peter Norvig – The Unreasonable Effectiveness of Data, pdf, video)

Here is an anecdote I like to share in this context:

> I was working on a activity recognition classifier that should be able to detect human activities like walking, running, sitting, etc.

> on the basis of sensor data collected on a mobile phone. My colleagues told me that PCA, SVMs and decision trees would be the way to go here. So I spent months, learning about these methods, collecting data, doing trials, etc. This all worked OK-ish but we were far away from the 99% accuracy claimed in the papers. And you could see that if just was not working all that well when you used it on your phone.
>
> It was on a train ride, when I suddenly realized: Gosh! I could build something better by extracting a few simple features from accelerometer data and funnel that into a few if statements. It took me 30 minutes to write this little function that outperformed everything we had done up to that point.

The one thing I learned from this was: Before ever touching RapidMinder again, implement your baseline!

Another take-away is: Get a firm grip on basic statistics, before learning about advanced machine learning models.

Value Running Code. You will be measured by the value you create for the company. The models you built on csv files on your Laptop can be great and everything. If you can't get them in front of customers, they are worthless to the company.

Also, the customer will not be care if your model is 99.9% accurate and uses $HOW_NEW_DL_FRAMEWORK and runs on $HOT_NEW_CLOUD_SERVICE. All he cares about is: Is the model available? Does it work good enough? Is it fast? It's important to leave behind the academic mind set of p-values and accuracy, and focus on the customer needs.

While Data Science takes largely place in languages like R and Python, the dominant languages met in production systems are Java, C/C++, PHP, JavaScript, Python. You will have to find a way to get your models in to those environments.

Fundamentally there are two way how to do it:

1. Write your models in a language used in production.
2. Export your models from R/Python. Load them into your production environment.

If you run Python in production, route one is particularly attractive. I have mostly worked with C based production systems, which have an embedded lua scripting layer. So I implemented my models in lua.

Route two is commonly taken in larger organizations, which have a team to run a data platform, and a separate team of Data Scientists who provide the models. Tools like Deeplearning4j (Java) allow you to import models trained in e.g. Keras (Python) and deploy them on the JVM.

Which route you take will depend on the environment you end up working for. In any case, it is your responsibility to make sure your models make it to production, and continue to provide value to the company.

Avoid perfection. This might be the hardest lesson for an academic to take. The software industry had to learn that planning is extremely hard and error prone. Agile Software Development and the DevOps Movement are the reactions of the community to epic failures of many large software projects. Instead of writing down requirements and estimating engineering effort, the most effective way to provide value is:

Release early. Release Often. Learn from your mistakes.

For Data Science applications this means:

1. Once your model is barely usable, get it in front of users.
2. Evaluate (measure) it's usefulness in practice.
3. Refine your model.
4. Iterate. Quickly.

It's much more important to be able to iterate quickly, than to have a 99%-accurate model to start with. But iterating quickly is not easy. There are a lot of delicate steps involved in getting a new model rolled out into a production environment. Make sure to understand the steps involved, and automate them as far as possible. Leading companies have fully automated deployment pipelines, that allow them to update (and revert) production code within minutes.

Take programming seriously. A large part of the day to day work of a Data Scientist, consists of writing code. In many circumstances you are effectively a Software Engineer with broad Statistical know-how. This means, you should treat programming as a profession.

The good news is: It's not all that hard. If you made it through grad school in pure maths, you are more than smart enough to become a decent programmer. After all, a proof is very similar to a program.

The bad news is: It's a huge mess. Approaching programming from first principles (Touring Machines, Lambda Calculus) is a rabbit hole with no end. In practice you will have to deal with large piles of undocumented code, that surfaces subtle bugs in the worst possible moments.

The only thing that really helps is exercise. Read code. Write code. Ship code ... and watch it burn. Debug code. Learn from your mistakes. You will eventually get better.

Just get started. Finding an attractive Data Science position with a decent company and a high salary is not easy. Job descriptions usually require industry experience, and list a large number of technologies you should be an expert in. While you might actually be a good match for those position (at least after a few months of training), it will be very hard to get such a position if you don't fulfill the requirements.

To understand the problem one must realize that hiring in the Data Science space is extremely difficult for companies. There are tons of young engineers from

all over the world, who can do some Python coding and have worked through some TensorFlow tutorials, which apply for Data Science positions. Often times these people can produce certificates and the best grades from schools or universities, yet in practice they are not able to do much. Tool-knowledge does not take you very far if you lack basic understanding on what they are doing.

The best strategy that I have encountered, is to just start somewhere. Take any job that roughly matches your interest, and will let you develop your skill-set. Software engineering roles are a great start if you have some programming experience already. Compared to the average CS major, you should have no problems to catch up quickly and exceed expectations.

Once you got some industry experience on your CV and earned some "street cred", your next job search will be much easier.

Engage with the community. Get in touch with other people who work in Data Science. Talking to other people makes learning much easier and is a great source of inspiration for things to try out. In addition, reputation and a network gained from these interactions will allow you to find interesting positions.

Read blogs. Leave comments. Ask questions. Take some online courses. Watch out for Meetups or Data Science conferences in cities near you. Start a side project that is of personal interest and talk about it at a Meetup. Write blog posts about things that you learn or find interesting. Teach other people what you have learned. Contribute to open source projects. There are many ways to get in touch!

4 Conclusion

In my opinion Data Science presents an attractive opportunity for academics in the STEM fields. Thorough mathematical education and fearless manipulation of statistical models are essential skills that most CS majors are lacking.

The biggest huddle to jump is to close the gap in programming skills. Here only practice will help. Once that step is taken, the door is open for many very attractive positions in an industry which is craving for analytical talent.

5 Acknowledgements

Thanks to Kevin Lin, Max Pumperla and Thiago de Faria for inspiration and feedback to earlier versions of this writeup.