Teaching Statement: Computer Science as a Natural Science

El Mahdi El Mhamdi

The current conjecture in machine learning is such that industrial labs are attracting young graduates from my generation by offering almost the same research scope as academia, while providing significantly more resources. Academia is, however the right place for the career I am aspiring to for two reasons: intellectual freedom, which is the key for the type of fundamental research I strive in and which I described in the research statement, and the mentoring and teaching relationship which I address in this document.

I was not trained as a computer scientist. Let me start by how I came to computer science research, how teaching made me discover the field and how it is still helping me make progress in my personal growth as an aspiring computer scientist. I then explain how this journey shaped my view on teaching computer science both to undergraduates in all departments, and to more focused audiences in CS curricula. I finally talk about my experience in mentoring master and junior PhD students in research.

My Path to Computer Science Research: Teaching

My first contact with CS was the entrance exam of the French École Polytechnique. Having prepared for that entrance from Morocco, my curriculum was not optimal compared to the one in the French classes préparatoires. In particular, there was no CS course offered. Thankfully, my grades in mathematics, physics and philosophy were high enough to compensate. At Polytechnique, I took the CS course for beginners. It was an introduction to the Java programming language. This first course was also my last course in CS as I followed a classical physics-focused curricula.

Before my PhD, and during my time as a research engineer in condensed matter physics, I started an online tutoring project called Wandida. Wandida was aimed to be a collection of short, very focused tutorials, aimed for bachelor students in mathematics and physics. Unlike MOOCs, where the content is rather heavy and often follows the lecture format, Wandida provides precise answers to questions such as "how to project vectors in mechanics"[1]. These constitute very popular web searches for undergrads who do not have the time to invest in a heavy MOOC, (they already have their lectures), but are rather searching for a self-contained explanation on a particular point they struggle with. Wandida also included *fun* and experimental content, such as a visual proof of the Bolzano-Weierestrass theorem using a sun rising in the horizon[2], or an illustration of what the curl is[3].

Michel Benard, manager of the Google university partnership, was enthusiastic about the project and helped fund its first phase and through professor Rachid Guerraoui, Wandida became a project of the computer science department. Wandida quickly became popular among EPFL undergraduate students for its physics and mathematics content. But the last thing I expected before starting this, was that I would be paid to teach computer science!

The success of Wandida convinced several leading scientists to join professor Guerraoui and me in developing it further. We were helped by French Academy of Sciences members Gerard Berry and Serge Abiteboul, ACM fellow Anne Marie Kermarrec, professor Gilles Dowek and many more. Working with these people made me realize how many foundational questions can be found in CS and ultimately pushed me into dropping my applications to PhD programs in physics and apply to CS instead. The deanship of our faculty at EPFL was convinced by our success with students and funded a full time position, so that someone could sustain the activity that was started by Wandida (which later took other forms than Wandida itself). Meanwhile, Wandida had a significant reach in my home country, Morocco, and inspired many other initiatives with larger audiences (high school, popular science etc.). In 2015 I was recognized for this by the TIZI award, given each year to about 15 promising young Moroccans. My contribution in inspiring a movement of science on the Moroccan web was recognized in the 'education' category.

Teaching CS before embarking in a PhD ultimately educated me on the very nature of computing, a fundamental science that used not to be seen as such by other disciplines. My awareness culminated during my involvement in the EPFL pan-department effort to change this old idea on computing and teach *computational thinking*, which inspires how I envision teaching undergraduate CS courses as I explain in the rest of the document.

[1]Réussir ses projections vectorielles en mécanique `https://youtu.be/UrLrReoJiH0`
[2]Bolzano Weierstrass Theorem - Rising Sun Proof `https://youtu.be/fYCAH0NzFTQ`
[3]Visualisation du rotationnel (rot) `https://youtu.be/btB4qCsOlHs`

Computer Science as Epistemology

If you take a look at a few computing classics you can find the following. (1) *Algebra* and *Algorithms* originate in a 9th century book, written by a lawyer, for lawyers. A book where *judgement* and *computation* can share the same word in Arabic (*Hissab*). (2) Turing's computability paper (1936), formalizing the universal computer, is an answer to the foundational crisis of logic, a question from mathematical philosophy, not an answer to the need for a new gadget[4]. (3) *Reinforcement Learning* (Sutton and Barto) is a venture that happened in the "Psychological Review" and other cognitive sciences, neuroscience or psychology venues. You can add to this list the paper on context-free grammars by Chomsky, a linguist who published it not in a linguistic venue, but in the IEEE Transactions on Information Theory in addition to many other multidisciplinary, yet foundational works of our field.

As a physicist at heart, I find joy in practising and teaching computer science as a natural science. Computing is a sort of quantitative epistemology, the science of how much can be done and how much can be known, that is still rarely seen as such (at least not by the physics curricula I came from).

In my undergraduate teaching, I would be highly motivated by introducing the students to this aspect of CS regardless of their background and major, because our era requires an even more multidisciplinary population of scientists and practitioners. This is what I already took part in, when I helped professors at EPFL use Wandida to introduce concepts from information theory, to computability and decidability or complexity theory.

Foundations of Machine Learning

For a course that can be both given in a lighter undergraduate level, or a more substantial graduate level, I am very motivated by teaching the basics of machine learning. In 2018, I built and taught (as a sole instructor) a course for the inaugural class of PhD students in the newly created Polytechnic University in Bengrir, Morocco. In 2019, I helped professor Alexandre Maurer build the second edition of the same course as he joined that University. This course was aimed for a PhD audience. But given the extreme heterogeneity of profiles, I had to adapt to a wide spectrum and added a lot of undergraduate-level content, from probabilities, algebra and convex optimization.

At EPFL, I was a teaching assistant for the first edition of *Optimization for Machine Learning*, taught by professor Martin Jaggi. This was a Masters' course. However, some of its content could as well serve as the basis for a possible undergraduate, larger audience course, if I have to aim for undergrads.

Distributed Systems Theory

For three years in a row, I was a teaching assistant for my supervisor's course on distributed algorithms. This course covers the basics of distributed systems: communication channels, broadcast abstractions, consensus and shared memory. I also gave guest lectures in the course on topics from my thesis, mainly distributed algorithms for machine learning. Besides delivering the exercise session, I was also in charge of setting the exams and correcting over 140 copies as the course gained in popularity. After TAing the course for the first time, I decided to include a session on logic and the basics of writing a proof and (theoretically) analysing algorithms, as I realized that students coming from different Bachelor places do not necessarily have the same level of familiarity with theory.

Topics in Robust Machine Learning

Finally, I would also envision teaching a graduate, seminar-based course on topics that are directly related to my current research. As mentioned earlier, I did that in the context of guest lectures for my supervisor's course on distributed algorithms, but as I mentioned in my research statement, I also did that in the context of the working group I was animating on AI safety research. This topic is moving very fast, especially in questions such as high dimensional robust statistics. Organizing such a graduate level course could be a very good way for me and my group to keep up with the literature. This could also be part of a larger-scope seminar with other professors.

Teaching Readiness

An indicative list of courses I could teach immediately: introduction to machine learning, optimization for machine learning, distributed algorithms, introduction to algorithms, discrete mathematics, calculus and algebra, probabilities and statistics.

An indicative list of courses I could teach but would require additional preparation: learning theory (graduate), theory of deep learning (graduate), information theory (undergraduate and graduate), introduction to databases (undergraduate), concurrent algorithms (graduate).

Courses that are out my teaching readiness would be those in the computer architecture, operating systems, software engineering or programming languages categories.

[4]Another overlooked fact is that Turing's most cited paper (until last year), was the work on the chemical basis for morphogenesis.

Mentoring

During my PhD, I supervised several master thesis students and some junior PhD students who became my co-authors: Sébastien Rouault, Arsany Guirguis, Vladislav Tempez, Sergei Volodin, Isabela Constantin, Hadrien Hendrikx, Philippe Yazdani and Henrik Aslund. This was possible thanks to the freedom that my supervisor gave me, as I could initiate many research projects on my own. My advising style varied according to the student and ranged from giving a high amount of freedom in defining the research direction (and actually learning from the student), to low-level supervision where I would provide conjectures, proof ideas or sketch experiments which were further conducted by the student.

Computer Science is defining our era and I am looking forward to becoming an educator in this field.