

---

# Teaching Experiments and Programming for Machine Learning

---

Francois Jacquenet

FRANCOIS.JACQUENET@UNIV-ST-ETIENNE.FR

**Keywords:** Experiments, Programming Languages, Validation

## Abstract

This paper aims to be a complement to Colin de la Higuera's paper about teaching theoretical computer science for machine learning. We claim that, in the same way it is important that students get a strong theoretical background, they also need strong practical courses in order to be able to design realistic prototypes validating theoretical studies.

## 1. Introduction

Machine learning is a subdomain of computer science that has some specificities, which is the reason why this workshop exists, of course. In (de la Higuera, 2008) it is claimed that it is important for machine learning students to be taught some theoretical fundamentals but also some specific topics in that domain. We also think that this is an important feature for students in order to make them not just end users of a set of tools, but also actors of the future of machine learning. Nevertheless, we also think it is important for students first to master some methodological processes and second to get some outstanding programming language skills. What would be a good student who would know much about theory but have no idea about how to really implement a beautiful algorithm in a real programming language? Of course, such a student might perhaps have a brilliant carrier in the academic world where he could get many beautiful papers accepted in prestigious conferences. That would be in fact a good result for all of us. Nevertheless, first, all our students can't enter the academic world, second we think such students would miss something. Indeed, we think mixing good theory foundations and good practical ones might lead to better balanced students that could then fit a larger range of businesses.

In fact we are tempted to bring about some change in the figure proposed in (de la Higuera, 2008) to make our thoughts being taken into account. In Figure 1 we propose a V-cycle, inspired from the V-lifecycle from the software engineering domain.

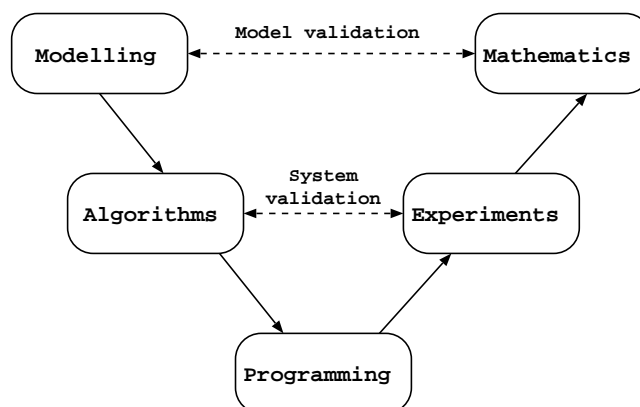


Figure 1. V-cycle of a good machine learning methodology

Modeling is the fundamental task of the whole machine learning methodology. From that first step, we design some new algorithms (or adapt some existing one). Programs are then developed in order to really implement the concepts modeled at the first step. Then a first validation step is proposed, running some experiments using the system developed in the previous step. Finally some mathematics are used to validate the formal model. Of course this process is not as linear as it is simply presented here. In fact one can see on the figure the interactions between *modeling* and *mathematics* coming from the model validation process, and between *algorithms* and *experiments* coming from the system validation process. What is interesting is that we can see we still have the links between *modeling* and *mathematics* and between *modeling* and *algorithms*, as in the figure proposed in (de la Higuera, 2008), but the third link between *algorithms* and *mathematics* is transitively obtained using *programming* and *experiments*, the two main topics of this paper.

## 2. Methodologies for experimental validation

In some domains, scientists have to make intensive use of experiments all the time. It is a crucial part of their job. Physicists, chemists, biologists are mainly used

to solve practical problems to make a step forward in their research area. In computer science, it is not systematically the same. Coming from the software engineering community the author has experienced the important gap that separates this community from the machine learning community on some experimental aspects. In the software engineering community, researchers have to design new methodologies, techniques and tools, in order to improve the quality of software. Theory is here also often important and, for example, language designers have to formally define the semantics of the language they are designing. Nevertheless, experiments have less importance in research papers because there are less comparisons to be done between the performance of tools in terms of speed or memory consumption. What software engineering researchers try to deal with is better reliability, more powerful techniques to help programmers design programs efficiently, more robustness of languages to errors, more generic tools or methodologies, etc. For example, a researcher may design a new language and then show what people can do with it, some applications of it, but there is not much experimental validation in the sense of what we intend by *experiment* in machine learning. In the machine learning community, we often want to design new learning techniques. Nevertheless, it is rarely the case that researchers design new techniques from scratch. Thus, many papers presented in conferences are related to some improvements of existing techniques. It is supposed to make the science progress, nevertheless, researchers have to prove they really do so. Thus we think it is essential for students to learn about several fields of experimental practice.

Students should know about *the main validation techniques that have to be used nowadays by researchers that want to prove their systems works better than others*. It would be a pity to design some very nice innovative algorithms and not to be able to experimentally show they work better than the state of the art, using widely admitted validation procedures.

Moreover *the theoretical foundations of various validation techniques* should be taught. Indeed, for example teaching cross validation techniques (Kohavi, 1995) is interesting, nevertheless it is even more interesting to also teach on what theoretical ideas such a method is based (Geisser, 1975). Teaching how to validate some algorithms on huge benchmarks is interesting, but it is also important to make the students understand why it is essential to observe the behavior of an algorithm when the size of the sample set tends to the infinite. If the student is not aware of such underlying theory, why should he think about testing huge amount of data?

To show the scalability of his algorithm, perhaps, but this should not be the only reason. As another example, consider precision and recall criteria intensively used in information retrieval (Swets, 1963): they may be taught as such, just as formulas. But they may also be linked to some more sophisticated statistical issues, such as type I and II errors (Neyman & Pearson, 1967) giving the student a higher point of view on his validation task.

Students should learn as soon as in Master's degree *to be honest and ethical when trying to prove the efficiency of a new algorithm*. Of course, theoreticians would state that the best proof is a theoretical proof, because there is no possible challenge. In situations where the complexity of algorithms may be formally calculated, this is indeed the best proof of efficiency. Nevertheless, there are many situations where we cannot compute the exact complexity of algorithms, for example when they integrate some heuristics or when they depend on users interactions. Then, when we have to experimentally compare an algorithm with others, we should not only choose the best dataset that will do so. If we think that comparing objectively our system will not be in favor of it, then that is because we did not do some good job before, or maybe in fact we did some good job proving one way of thinking was not the good one. We think conferences should accept papers showing how some tracks have been explored and *how they have been proved* to be unsuccessful. Maybe, if students got some advance courses on how to correctly run some experiments, they would be able to write some interesting papers about some failures they encountered during their researches.

Finally, students should learn *the way experiments have to be commented*. There is nothing more annoying than a paper that exhibits large amount of charts or tables and that does not provide any intelligent comments on them. Moreover, using results of experiments we may exhibit all and nothing. Analyzing results is a difficult task that need to be taught and practiced. Professional statisticians are used to that kind of task, they know where bad ways of reasoning may appear while reading statistics and they know how to take care of it (or not sometime, depending on who is granting their work...).

### 3. Outstanding programming skills

There are several reasons for students to develop outstanding programming skills. As we have said previously, machine learning is a domain where experiments play a very important role. Thus, we think it is essential for students to be able to make their methodolog-

ical process rely on solid programming skills.

First we think it is important for students to *have a good knowledge of a wide variety of languages*. There are problems that may be more easily solved using some languages rather than others. It is important to be able to switch from one language to another, depending on the problem to be solved and its context. It is also sometimes important to be able to design a tool that will integrate several different programming languages. For example, it may be interesting to combine JAVA to be able to easily deal with user interface, Prolog to design some sophisticated reasoning parts of the system, XML and PERL to deal with data structured in files to be read or written. In fact this knowledge is not to be taught at master's level but rather it has to be stated as a prerequisite to all the courses of a Master's degree in machine learning.

Then students should have *an outstanding knowledge of a particular programming language*. It does not necessarily have to be the same language for each student. A student should be able to have a passion for his preferred language and show some professionalism while using it. He should know all the tricks and hacks that can be done with it and be always aware of new ones. One may think that when several languages can be chosen in the context of a given problem, the researcher (student or not), and the industrial practitioner would prefer to choose the one he masters the best. With such a language, in a given situation, he should be able to immediately use the specific APIs that will be the most efficient, the correct tricks to implement such particular data structure and control structure, etc. Within the era of the Internet, many resources are already available, many APIs have been designed worldwide, thus the main problem remains to know they exist and how to use them for our particular needs. Such kind of expertise can only be achieved while programming hardly everyday using the same programming language.

An important proficiency students should acquire is the ability to *design some powerful interfaces to their systems*. It's a pity to see brilliant algorithms that can deal with huge amount of data but that are not brought to the fore by some professional user interfaces. For good algorithms or systems to be widely adopted by the community, designers should provide the possibility to encapsulate their tools in some elegant interface. This would benefit to other potential users, but also to the dissemination of ideas, which is an important feature of researchers task, but also of industrial engineers.

Finally, students should be able to *validate their ideas*

*efficiently*. As experiments are a key feature in the domain of machine learning it is obvious that researchers and industrial practitioners will have to develop a huge amount of code in order to validate their innovative techniques (unless they have the possibility to theoretically validate them). Thus, being able to develop bug-less prototypes and scripts for running huge amount of experiments and get (semi) automatically some charts and comparative studies, is something essential to prevent completing the design of an innovative tool while competitors have already done a better job than you. In that sense, to program and to validate rapidly is an essential point to come through the international competition.

#### 4. Conclusion

This paper presents some thoughts about some practical issues we think teachers should deal with in some machine learning curricula. We think theoretical computer science is important, as said in (de la Higuera, 2008), and that such a theory may also be brought to the fore as much as possible in the experimental part of machine learning courses. Without such a practical background we think the students will not be well equipped to make their job progress over time while being later in the academic world or in the industrial one. That may penalize them, as well as their employer and, at the end, the end users of the (bad) products they may appear to have designed.

#### References

- de la Higuera, C. (2008). Theoretical computer science for machine learning. *Proceedings of the workshop on Teaching Machine Learning*. This volume.
- Geisser, S. (1975). The predictive sample reuse method with applications. *Journal of the American Statistical Association*, 70, 320–328.
- Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence* (pp. 1137–1145). Morgan Kaufmann.
- Neyman, J., & Pearson, E. (1967). The testing of statistical hypotheses in relation to probabilities a priori. *Joint Statistical Papers* (pp. 186–202). Cambridge University Press. (originally published in 1928).
- Swets, J. (1963). Information retrieval systems. *Science*, 141, 245–250.