Module report on FPIA

Hans Georg Schaatnun, Que Tran, Robin T. Bye

7th September 2017

The module is organised as fifteen full-day sessions on Friday. The delivery is divided between Hans Georg (Week 1–6), Que (Week (7–9), and Robin (Week 10–14). The fifteenth week is reserved for revision and catch-up and will be negotiated later.

1 Evaluation

Evaluation data has been gather continuously and informally. With few exceptions, each teaching day day starts with an evaluation, asking every student to answer a few selected questions about the learning and the activities so far. Thus the full class has served as a reference group. Additionally, opinions are sought occassionally in chats with individual students.

The students were asked to submit a reference group reported, but opted not to do so. However, this report was presented to them on the last teaching day, and their remarks have been incorporated.

2 Intermediate report Weeks 1–6

1 A Manifest

The basic learning activity for the first part of the module is the *tutorial*, which in our case implies:

1. A blend of examples and exercises. Some steps are given as examples and some as exercises.
2. Whole-task exercises. The tutorial as a whole give a full solution to a complete and coherent problem.
3. Each individiual step in the tutorial is limited in scope, so that the students should not take very long to solve it.
4. Progression means that we start with a bias on examples or cut-and-paste exercises towards a bias on problems the students need to solve.

5. Some questions for discussion aim to promote reflection and deep processing.

6. Questions from the class and discussions are encouraged, also when they diverge from the main objectives of a session.

These principles are built on cognitive load theory **?**. The blending is not generally as structured as their paired example/problems, although there are some examples of paired problems, with the first being almost entirely cut-and-paste from the tutorial and the second leaves some key items open for completion by the student.

Whole-task exercises are intended to demonstrate relevance of the material, which is probably the main argument for proponents of whole-task learning and problem-based learning. The heavy use of cut-and-paste-able steps makes it very different from what **?** call whole-task learning. The idea is to get best of both whole-task learning, worked examples, and paired problems.

Two-way questions and discussion in class are used continuously (1) to link new material to the students' past experience, (2) to follow up on student ideas and encourage their own thinking, (3) in a Vygotskian perspective, to support the development of language, and (4) encourage collaboration and social learning.

2 Positive experience

1. The balance between practical problems and new lecture material is very good.

2. The tutorial format appears to be very effective.

3. Some students take approving note of the progression in the tutorials, which start with a lot of cut-and-paste exercises with a gradual transition to more individual work.

4. Online tutorials make it easy to catch up.

3 Challenges

1. The students use different operating systems, mainly Windows, and both software installation and adaptation to non-Unix systems are hurdles which waste unnecessary time. Without a number of Windows wizards in class it would not have worked.

2. Learning Haskell and AI in the same module is challenging for the students. It would be easier to learn AI in the context of a known programming language, and they would also be easier to learn functional programming in a context of simpler problems and algorithms.

3. The plenary discussions and talks have been too long in some cases. An extra effort has been put in to reduce this from week 2 onwards.

4. Several students keep running into obstacles which they do not get over on their own.

5. Many students lag a week or so behind. Some because they struggle and some because they don't.

6. Differentiation proved an unexpected challenge. Only one student was able to do a repeated application of the chain rule, even after a plenary review of theory.

7. There is an unexplained overlap *Swarm Intelligence*. This was not a problem last year, when FPIA ran before Swarm Intelligence. When they now run in parallel it is harder to let one module build on the other, and on top of that, the syllabus in Swarm Intelligence has changed.

3 Attendance

There are typically 10–12 students per session; about 15 students in total following the module.

4 Additional experience Week 7–9

A couple of additional remarks collected during Weeks 7–9:

1. more exercises about data preprocessing in Haskell.

2. more smaller tasks in Haskell to get used to the language before implementing neural networks.

3. (several students) other types of neural networks, such as RNN and CNN. They wanted to learn those networks, at least having the introduction showing what they are and when to use them.

5 Additional experience Week 10–14

- Complaints about too many programming languages to be learnt in different modules

- Exam questions about generic types, typeclasses, and the fat arrow, e.g. Eq a => a -> a have not been taught during semester (?)

- One students would have liked more emphasis on type system, algebraic types, etc.

- Some student who are skilled programmers repeat there dislike for Haskell.

- Some students criticise the admittance of students without programming experience.

At this stage, there is a stable group of eight students attending class.

6 Assessment results

A	B	C	D	E	F	Absent
2	1	4	–	2	4	2

Out 13 students sitting the exam, four students failed. The failure rate is disconcertingly high. However, we know that several students have struggled in several modules in the course, often because of missing prerequisite knowledge. Therefore, we will not make any conclusions based on the failure rate in this module on its own.

The passing grades show a reasonable distribution.

7 Evaluation of the module description and infrastructure

The evaluation highlights two or three challenges outside the control of the module conveners.

Firstly, learning Haskell and AI in the same module is suboptimal. This is mentioned by a number of students and we can argue the same case based on cognitive load theory **?**. It would be better to teach AI based on a language and paradigm with which all the students are familiar. The current module was originally designed to cater for students with no background in programming, and it is clear from the feedback that this is not appropriate for skilled programmers. When the entire programme now is set up to require programming background and a catch-up courses in imperative programming is included, this module no longer makes sense.

The students also object to the number of languages they have to learn for different modules. Currently every module convener has chosen language(s) completely independently of all others.

Depending on the students' own laptops wastes a lot of time, because they run operating systems and most of them do not understand the system they are using. The module convener cannot be expected to provide technical support on different systems. Adequate infrastructure can be provided either in the form of technical support for the different platforms in use, or in the form of standardised computer lab's. Virtual machines could be used, but technical support is then needed both to set up and quality assure the machines and to support students in the deployment. Virtual machines were tried this year, but the only student who tried to use it on the first day had to give up intalling it.

8 Evaluation of the delivery

The tutorial format used early in the module has proved extremely effective at that stage of learning. The practical activity both trains hands-on skills and forms a basis for theoretical discussion. Most of the class seems to catch up very well. Given the learning objectives there is no obvious way to improve the learning.

Students repeatedly needing help to get over obstacles in the exercises is intentional. We believe that the students have to experience the problem first to learn the solution.

The teaching format with a classroom booked for a full day where the convener can come and go while the students solve exercises has great potential.

There is a number of students who struggle. The continuous evaluation exercises mean that we have a good overview of exactly how much students are lagging behind. There is no reason to believe that the problem is more severe than normal for other modules. So even though this is a problem which should not be taken lightly, the solution may not lie in the delivery.