G52GRP INDIVIDUAL REPORT

Neural Network Toolbox – Alfred

Barnabas Forgo (bxf03u)

27/03/2015

gp14–jl–rij

Other group members:

James Michael Ronayne

Mohammed Ali

Mark Allen

Luke Bates

Hee Kah Ooi

Weixuan (Peyton) Tao

Supervisor: Robert John

In this report I will describe and analyse my contributions to the second year group project in a reflective manner. But why would I do this? Being reflective is very important in our everyday life. We don't even realise it but we are thinking reflectively all the time. This is how we process the events and how we learn. Also reflective thinking is the most effective way of improving ourselves, hence it is very important both in our personal and professional lives.

Project Description

Our task was to implement a neural network toolbox. This toolbox would be able to create, train, and simulate many different types of neural networks. There are many similar tools on the market already, but their use is mainly intended for advanced users who are already familiar with neural networks. Hence what makes our project different from these is that it has a much smaller learning curve – ideal for users unfamiliar with how neural networks work – yet it would make complex settings available for users who are more experienced with similar software.

My Contributions:

Group Leader:

- chairman for most informal meetings
- keeping track of and managing most tasks
- making sure that communication works well in the group
- resolving any disputes that might slow our progress
- keeping up the motivation

Developer & Technical Lead:

- research
- system design
- implementation of ART
- keeping the codebase clean and well documented
- fixing bugs
- benchmarking implemented neural networks

Challenges

On one of the first group meetings we had, I was elected to be the group leader by a uniform vote. First I wasn't too happy about this as I barely had any experience in leading a project of this magnitude. Only later did I realise that this wouldn't have been any better if anyone else had been chosen to lead the group, as none of the others had any experience with such a task either. This realisation made me more determined to the task as a good leadership is essential to the success to any project. Taking this thinking further I also understood that the experiences gained here can be very useful in my future career. I believe that since I came to this conclusion fairly early in the project's running was a key aspect to the success of it, as it made me a better, more determined leader.

On the other hand, having this responsibility had its unique challenges as well. One of which is keeping track of everything that is happening with the project all the time. This required me to be able to understand all aspects of our tasks to be carried out. Needless to say, it took me countless hours of research as our task was very complex. It also required me to be an able communicator. As I am quite reserved person and English isn't my first language – nor for some of my group members for that matter – this was one of the bigger obstacles to overcome for me. I don't think that I got perfect in this aspect, but I do believe that my communication with my fellow members improved a lot over time.

Additionally I also had to make sure that I wasn't the only one who sees where we are going with the project. For this I asked the team to use Trello[1] for task management. This service is a virtual scrumboard that is free and easy to use, with some additional features that makes group working considerably easier. It took quite a while for us to adopt it, as for it to work it has to be integrated in everyone's workflow. After the initial resistance it turned out to be a good decision to adopt it, as the group dynamics got a lot better, for it lead to a better understanding of how tasks connect to each other. It also made my job easier, because I could easily assign tasks to members, so that they would get notified over email.

This leads me to the next challenge which was to use my gained knowledge about the project to efficiently coordinate my colleagues. Probably the hardest part of this was to assign tasks accordingly to each person's capabilities, such that it doesn't overwhelm them. At first I know I was really bad at this as I hadn't met them before and I wasn't aware of their strengths and weaknesses. As the project developed and I got to know them better I could assign tasks to people to better fit them.

Another duty of a good leader is to keep his team motivated. Since most of the group thought that our project was significantly more complex than others, the overall motivation was just as low. To keep us motivated I tried to break down the problem into smaller chunks that are easier to solve for us, so that we would feel more confident. Consequently it worked out really well; as time passed each team members' confidence and motivation rose over time.

Moreover, making the software had its own issues too. Given the complexity of the task we had to do, it needed us to conduct a lot of research. This involved looking into other software – like Matlab[2] or Encog[3] – reading research papers and books. Since none of us had any real experience in implementing neural networks, this was a daunting task. To overcome this we decided to learn and build from the bottom: getting familiar with the easiest ones first leading to the more complex ones. Firstly we all got familiar with the backpropagation algorithm[4] – probably the most well–known artificial neural network algorithm. The reason why all of us had to get familiar with it is that we believed that all other neural networks were built on it. Only when we got to the other ones did we realise that this wasn't true, nevertheless I believe that this was a crucial step as backpropagation is one of the key features of our software. Also as it turns out the other architectures that we implemented weren't based on the backpropagation algorithm, but it aided the understanding of the other networks' inner workings.

After the research was done the next key step was to make a design of the software. This included rapid prototyping of the user interface and the inner system design. In this step we made a huge mistake. Since we didn't put enough research in the more complex algorithms we were lead to believe that other networks had the same basic architecture as a feedforward network[5]. As I mentioned earlier it was only

[1] trello.com

[2] uk.mathworks.com/products/matlab

[3] www.heatonresearch.com/encog

[4] The backpropagation algorithm is a neural network commonly used for classification of data. How it does this is that it tries to approximate a function that exists between the data and its classification. The backpropagation algorithm is usually used with a feedforward network.

[5] A feedforward network is a very basic neural network. It contains several layers and the layers contain nodes. Each layer's nodes are connected to the next layer's nodes. These connections have randomised weights as well. When data is presented to the network, each node's output is going to be based on its input connections' weights and its input nodes' output.

when we dived into more advanced algorithms like Self–organising Map[6] when we realised that our system design was faulty. The main issue was that the feedforward network was designed to be more modular than it needed to be as other networks have a different architecture. This has led us to a bit of 'spaghetti code', and it slowed our progress quite considerably. I think that as a leader this was one of my biggest mistakes. Had I put in a few extra hours of research and this could have been avoided.

Eventually when the design was done, we started implementing the most basic things in the system. This lead to two issues.
The first was SVN. Most of the team hadn't used version control software before and because it has a very steep learning curve it led to lot of commit conflicts and accidental code deletes. Also for the first few weeks after starting coding we didn't use branches which broke already working code more often than not. Realising this issue I tried to solve the problem by teaching others how to use SVN properly. Couple of commits later the 'repository accidents' didn't stop, but they noticeably dropped.
The second issue was figuring out how to work in a team. When designing the software we also took into consideration what software engineering methodology to use. Since we only knew the theory of the methodologies we opted to try out a few and go with the one that works best for us. After all we ended up mixing them up – a bit of XP here, some of Scrum there, with hints of a Spiral Model – which is not uncommon in industry, as far as I know, but thinking about it, had we followed only one practice could have sped up our development process.

Additionally I had the task to implement an Adaptive Resonance Theory[7] network. The main issue was that this network architecture was developed fairly recently, therefore it hadn't gained popularity, which means that hardly any open source implementations could be found. To tackle this problem I had to implement it based on the original research paper (Carpenter & Grossberg, 1987). This was particularly hard for me as translating equations to algorithms is unfortunately not in the university syllabus, but I eventually tackled the problem.

Since we had to work in a group a house coding style had to be adopted. Since we were using Java, which has well defined coding standards we went with that, with the extension of using camelCase for variable names. Although I was familiar with these standards others weren't, so I tried to gradually teach them how standards adhering code should look like. Until we all learned this I often found myself rewriting or cleaning other people's code – in some cases a whole class – which I probably shouldn't have done. Had I realised that earlier would have saved me many hours of re–coding that I could have spent on implementing new features or polishing old ones.

After we all tackled these smaller–bigger issues we eventually ended up with working neural networks. But this presented us with another task: to test these. The reason why this was particularly difficult is that our implementation is fundamentally different than the industry standard ones – which actually was sort of a requirement. Also most of the networks have a random factor in them which makes every run different than the previous. Eventually we found a way to compare our networks to Matlab's – detailed description on the methodology used in the group report.

[6] A self-organizing map is a different type of neural network that is generally used to analyse data. In its standard configuration it tries to organise similar high-dimensional data near to each other on a 2D map.

[7] An Adaptive Resonance Theory network is a neural network that tackles the plasticity-stability dilemma; namely, how a brain or machine can learn quickly about new objects and events without just as quickly being forced to forget previously learned, but still useful, memories. The way ART solves this problem is that it keeps track of two sets of weights: one top-down from the cluster layer which acts as a long-term memory and one bottom-up from the input layer which acts as a short-term memory.

Achievements

The end result of this group project is – according to our tests – a 'bulletproof' data analysis software. Since I was the technical lead I had to make sure that everything works together as expected. For example hooking up the networks to the UI – both of which are made by different team mates – was one of my tasks. I also learned quite a bit of things in several areas.

Firstly: leading a group. This at first – as I mentioned before – was a daunting task, but thanks to my very understanding and patient colleagues I adapted to the task quite quickly and was able to lead a successful project with only minor hiccups.

Secondly: how to be a self–learner. Although we used Java that we were already familiar with it had its challenges. For example I used to think that since I am not doing the GUI module I could let those who do make the UI. Later I noticed this thinking was selfish and just wrong. If I hadn't contributed to the UI's development, the project would have most likely failed.

Lastly: the practice of developing a fairly big piece of software. This at first was the most frightening of all. I remember that a handful of months earlier I had no idea how to even start with this monstrosity, and now, looking back it wasn't even that hard.

Unaccomplished goals

Since we started implementing the software quite late in the project and we got the initial system design wrong some things were left out of the original requirements. Originally we planned to have a web based interface so that the tool would be more accessible. This seemed achievable at the time, but this server–client model would have increased the complexity of our project so much that I don't think we would have finished it by now. On the other hand cross platform compatibility wasn't discarded as a requirement. For this reason we used Java to implement the software. Additionally, only later did we realise that we could extend our implementation to do some pretty amazing things like an Ensemble of Classifiers[8], but since this wasn't in our original requirements we didn't consider this as a failure.

What would I do differently?

If I had to start again from scratch with the same project I would do a great deal of things differently.

Firstly: start the implementation earlier. This is the first for a reason; for a long time I believed that there is no real chance of us finishing this on time. I think this was mostly my mistake, as I didn't push the group for a very long time. Only if I had done this earlier it could have led us to better polish features or add new ones.

Secondly: more frequent design cycles. We should have involved our supervisor more in the design process, as I am still not sure whether the implemented software matches the original requirements.

Lastly: more research and better system design. I think that the mistake we made in the software design step could have easily led to a failure of the project. Fortunately we could overcome it, but I am sure that some of the nasty bugs were only caused by the bad design.

Conclusion

Overall I am happy with the results. I think everyone put in as much work as they could and this is the reason for the success of the project. I know that each of us learned a lot, and I am sure that we will be able to use the gained experience in our careers.

[8] Ensemble of Classifiers is a type of machine learning architecture that uses multiple classifier neural networks (can be different types as well), to achieve a better accuracy.