24/10/08 sjw06 / ig106 / ctm06 / jv06 / plc06

Software Engineering III - Methods - Report One

A Neural Network Modelling Tool

Key Requirements

This project can be broken down into 6 primary objectives, the fulfilment of which would constitute its success.

- GUI to represent and edit network functionality and state
- Training & Simulation
- Describe neuron linkage at a global and individual level, including various connectivity algorithms
- Saving and loading of nets, including connectivity, training state, and graphical location
- Export of data to an intermediate format for importing into external tools (e.g. Matlab source, XML for custom hardware)
- Management of training data collections for a given net

Extensions

The following collection of extension objectives are potential further directions for the project to take. Their fulfilment is not entirely necessary, but would be "nice-to-have"

- Modular training algorithms
- Modular squash and output spiking functions
- Modular neural nets – insert one network into another as a sub-network
- Network visualisation
    - Permit the user to watch a network running to see the firing patterns in hierarchical context
    - Raster plots of network firing
- Custom input and output API for loading data input and visualising output from a neural network - can also feed into training to test correctness of a given input
- Automatic permutation for N-Fold error analysis

Development Methodology

As a team we aim to practice an eXtreme Programming (XP) based methodology, which is a form of agile software development. As part of this, we will be practising pair programming and peer code review to keep group communication and code quality as high as possible. A test driven development scheme will also increase the speed of development and code correctness, and code coverage tools will be utilised to verify project progress and the robustness of our system. Finally, regular User Acceptance Testing (UAT) will ensure that our finished project meets the client's specification.
We feel that communication, documentation and project tracking are highly important aspects of project management, so we will use several programs to make these as hassle-free as

possible. A Subversion (SVN) repository will be used for source control; this lets multiple developers concurrently work on the project and even on the same files, and elegantly handles multiple revisions and streams of the project. In this way it is easy for all team members to be working on the latest versions of our code and documentation. Software documentation and team communication will be stored in a wiki, allowing each group member to quickly contribute to, and edit, the available material. The wiki implementation we have chosen is Trac, which also includes useful features such as bug tracking and tickets.
At a high level, our design will follow a Service Oriented Architecture (SOA). This allows the project to be extremely scalable and segmented into small modules, which can each act as deliverables. Doing this helps us run a concurrent development environment where single developers have the ability to work on individual modules, and code maintainability is increased as it will be trivial for other developers to add extensions in the future. The intrinsic low coupling of this architecture will keep the dependencies between modules to a minimum, letting developers work on plugins without having to rely on the progress of others.

First Iteration Plan

One early goal will be to explore our chosen frameworks and technologies to test their suitability for this project. Once this has been decided, we will be able to mock up some user interface designs using our chosen platform and iterate their design with our supervisor. At the same time, the rest of the team will evolve the use cases we have discussed with our supervisor to define the delineation of services our project will be built upon, along with standardising their interfaces. Once these are designed, we can begin to implement their behaviour. For our first iteration, we will aim to cover:

- Basic saving and loading of neural nets
- Creation of an MxN interconnected net (i.e. A network with no choice in how the neurons are connected)
- Simple training and simulation, with one type of neurone
- Basic rendering of graphs and UI interaction within the mock-up

These services will then pave the way for us to simply improve the available services without requiring any fundamental architectural changes in future iterations of the package.

Future Milestones

Beyond this first iteration, the services will be iterated upon individually, as this modular service-oriented approach allows. The important milestones for this, on a per-service basis, are outlined below.

- Saving and Loading standard file formats, e.g. In XML, Matlab, or otherwise
- More advanced GUI interaction – layered and modular networks
- Custom connectivity of neurons
- Modularity of functions and training (plugin management)
- Palette of standard available neuron functions and training algorithms
- Nestable modules
- Workspace management - including training data
- Network visualisation
- Error analysis