# Neural Networks - Log Book

Ryan Spangler

March 16, 2011

## Preamble

It was asked that this log book be a document of my *thought process*. I know that is probably intended to be more official than it will turn out to be, but for this assignment I am choosing to take the duty faithfully, and maintain this as an *unedited, stream of consciousness* account of my thought process while conducting this research project. It may contain swaring. It may contain completely unrelated tangents that have nothing to do with the experiment at hand. It may contain offensive or unpopular opinions, or pure nonsense. In accepting the charge to document such an intimate and fundamental thing as my "thought process" I take it as a personal challenge to most clearly capture with words the actual thoughts I am having while I am having them. In as far as this is actually impossible, it will reflect the nature of the channel between my raw thoughts as they occur (in whatever physical medium is really at the heart of thought) and my ability to enunciate and elucidate these incorporeal entities with language.

You have been warned.

## Log

### Before the Experiment Begins

#### The Pain of NeuralWorks

First off I must get this out of the way: I am frustrated with NeuralWorks.

There is obviously a lot of effort that went into designing and implementing the various neural network algorithms, but I am puzzled what it is the

goal of NeuralWorks truly was as intended by its makers, as none of the possible roles of software I can imagine is it even remotely suited to filling. It is dense and full of a cryptic labyrinth of menus and fields and dialog boxes, so it can't be intended as an educational tool. It is clumsy to actually accomplish a task as each act requires multiple navigations, and bulk tasks like running fifteen trials and comparing them each require multiple mechanical steps and external computation and graphing to process the information, so it can't be intended as an actual productive tool that humans are supposed to use. At the same time, there is no way to use NW as a component in other toolchains, from other programming languages for instance, without manually doing the steps yourself to process the inputs and obtain the outputs (I found the User-Defined Input and Output feature it offers, but this is only to build functions that can be run from *within* NW, you still have to navigate through the menus and interact with the GUI frontend to get any work done. Also, you must operate within NW's traditional initialization and training cycle, performing functions though callbacks that are called only at certain times under certain conditions. There is no way to use any NW functionality from a program external to NW, which is what would actually be useful). There is no way to automate tasks programmatically or access functionality thanks to its closed source proprietary nature, and the manual labor of getting results is laborious, so it can't be intended as a programming tool that could be used in conjunction with other computational environments except in the most superficial sense. Also, it does not appear to have been updated since 2001 (a whole decade!) and the owners are still charging comparable rates to a time when they were actually working on it. In the meantime many things have happened in ten years, and there are open source tools freely available which are libraries that can be included in other projects, that are readable and editable by anyone (and therefore open to academic inquiry and useful for education), and have thriving contributor bases from all over the world and are being actively maintained (FANN comes to mind, but there are others). I am baffled as to how NW can be considered a legitimate platform for anything beyond tinkering and experimentation, and am equally baffled as to why using NW is considered an engineering endeavor.

I am also frustrated that I asked to be able to develop my own algorithm for the LVQ, which would have been far more instructive and useful than merely creating something through InstaNet, and was refused. Using NW is painful and not conducive to learning or productively conducting research. I feel that the inability to programmatically conduct research and compose

functionality in novel ways is a huge obstacle to obtaining results, and renders what could be an enjoyable endeavor into a task of suffering and exasperation.

### Life After the NeuralWorks Rant

Okay, now that I have that off my chest, I can get started. The first step is to learn about the LVQ sufficiently to know what it is I am doing. The similarity between the LVQ and SOM (and common heritage) means that many of the resources for the LVQ also overlap with those for the SOM.

Now that that is done, the task is to actually run the first set of experiments with all 52 of the available features. The only consideration is in how many hidden layer PE to use. Since there are 5 outputs and the definition of LVQ is to have the same number of hidden competitive PE's per output, I choose 50, just to be safe.

This leads to a consideration of what experiments to run. Do I want to just accept 50 as a good number, or should I do some comparative runs to make sure 50 is optimal? Probably should. So here is an outline of potential experiments to get the most thorough range of results:

- All 52 features, 50 hidden PEs.
- All 52 features, 25 hidden PEs.
- All 52 features, 10 hidden PEs.
- Most salient 15 features, optimal number of PEs found before.
- Critical 5 features, optimal number of PEs found before.

## Running Experiments

After running the first set of experiments, there is no qualitative difference between using more or less competitive PE's in the Kohonen layer. I want to compare the weights before training and after training to see how the weights for the various features adapted to the inputs. If there is a correlation between certain features and the correct classification it should show up in the weights from the inputs to the competitive layer.

As expected, the relationship between the features and the classifications was encoded in the weights! After subtracting the weight vectors before training from the weight vectors after training five features were revealed to

have been positively shifted. This is fortuitous. Now I must verify that these five vectors are really the critical set. A good way to do this would be to make a training one with these five, then make a training run with a selection of five random features and compare the performance. If this really is the critical set then it should far outperform the set selected randomly.

Sure enough, the revealed set of five features trains to over 91%! The randomly selected set does not even reach 80%. Possibly the architecture of the LVQ made this relationship more obvious.

Now the task is simply to generate graphs and make screenshots.

## Reflections

### Forgiving and Accepting NeuralWorks

Ultimately, NeuralWorks does do the job of executing various neural network paradigms, so it is legitimate in that regard. It is obvious that a great amount of work went into implementing all of these paradigms efficiently. My main complaint can be boiled down to that I still must execute many mechanical repetitive steps in order to get anything done, which is what the computer is supposed to be doing for you. The advantage of being able to use NW from a programming language would be that as I learned more and used the software in real applications I could progressively automate more and more of the operations I previously had to perform manually. This would lead to my ability to harness the tool with ever increasing usefulness. The inability to perform more functionality than the designers built into the software means that there is a hard limit to what is achievable through the program alone.

Regardless, some measure of work can still be done in this limited framework, and I was able to complete the assignment as given (with the help of an external programming environment to actually process and plot the weights and the outputs for the various runs).