When to use (or not!) a BP Neural Network Solution

A back-propagation neural network is only practical in certain situations. Following are some guidelines on when you should use another approach:

- Can you write down a flow chart or a formula that accurately describes the problem? If so, then stick with a traditional programming method.
- Is there a simple piece of hardware or software that already does what you want? If so, then the development time for a NN might not be worth it.
- Do you want the functionality to "evolve" in a direction that is not pre-defined? If so, then consider using a Genetic Algorithm (that's another topic!).
- Do you have an easy way to generate a significant number of input/output examples of the desired behavior? If not, then you won't be able to train your NN to do anything.
- Is the problem is very "discrete"? Can the correct answer be found in a look-up table of reasonable size? A look-up table is much simpler and more accurate.
- Are precise numeric output values required? NN's are not good at giving precise numeric answers.

Conversely, here are some situations where a BP NN might be a good idea:

- A large amount of input/output data is available, but you're not sure how to relate it to the output.
- The problem appears to have overwhelming complexity, but there is clearly a solution.
- It is easy to create a number of examples of the correct behavior.
- The solution to the problem may change over time, within the bounds of the given input and output parameters (i.e., today 2+2=4, but in the future we may find that 2+2=3.8).
- Outputs can be "fuzzy", or non-numeric.

One of the most common applications of NNs is in image processing. Some examples would be: identifying hand-written characters; matching a photograph of a person's face with a different photo in a database; performing data compression on an image with minimal loss of content. Other applications could be: voice recognition; RADAR signature analysis; stock market prediction. All of these problems involve large amounts of data, and complex relationships between the different parameters.

It is important to remember that with a NN solution, you do not have to understand the solution at all! This is a major advantage of NN approaches. With more traditional techniques, you must understand the inputs, and the algorithms, and the outputs in great detail, to have any hope of implementing something that works. With a NN, you simply show it: "this is the correct output, given this input". With an adequate amount of training, the network will mimic the function that you are demonstrating. Further, with a NN, it is OK to apply some inputs that turn out to be irrelevant to the solution - during the training process, the network will learn to

ignore any inputs that don't contribute to the output. Conversely, if you leave out some critical inputs, then you will find out because the network will fail to converge on a solution.

If your goal is stock market prediction, you don't need to know anything about economics, you only need to acquire the input and output data (most of which can be found in the Wall Street Journal).