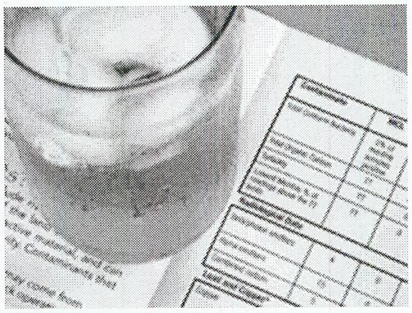
No, We Need a Neural Network

Friday, December 01, 2006 (137 Comments)
by Alex Papadimoulis



M.A. is one of the world's foremost experts on neural networks. His undergraduate specialty was artificial intelligence, his master's thesis was about genetic algorithms, and his doctoral dissertation covered evolutionary programming. Such an extensive computer science education opened up a wide range of career options, ranging from a professor at a university to ... a professor at another university. When someone outside of academia sought out his expertise for a project, he jumped at the opportunity.

The company that wanted to hire M.A. was a small programming firm that developed and maintained software used by the Bureau of Water Management. They were recently awarded a large contract to redo a rather inefficient part of the system and were convinced that implementing a neural network was the way to go. After the initial interview, M.A. told them that a neural network was the wrong tool for the job and that they should use a traditional approach. Management disagreed with his assessment and insisted that he come aboard to help rebuild the system. Had they not offered such a generous salary, he might have recognized this as a first warning sign.

M.A's first few weeks were spent analyzing the existing system and its processes. Water testing equipment throughout the region sent data in a variety of different formats to the central processing server. The data was logged in a single database table with three columns: Station Number, Recording Date, and Raw Data. Each week, a team of seven technicians would spend a day manually parsing the Raw Data -- be it in CSV, fixed-width, XML, etc. -- and inserting the formatted data into another database table. A script would then parse the standardized data and create the weekly water reports.

Management wanted to create a Neural Network that would learn how to parse the raw data and free up the technicians so they could do their regular job.

Armed with all the details of the process, M.A. explained that this was a very bad idea. He argued that:

Neural Networks are not designed to handle tasks like this
If developed, such a Neural Network would need a lot of training
An algorithmic version could be developed in a fraction of the time and do the job better

Management disagreed. They insisted that Neural Networks were the wave of the future and wanted M.A. to go forward with developing it. He grudgingly agreed and got to work designing the system. There would have to be a different Neural Network for each type of equipment, a server to keep track of the networks' activities, a learner application for all networks, and an encapsulation around each network to allow it to reset in order to adapt to changes once it was fully trained.

M.A. explained the mess that would be required in implementing a Neural Network approach and tried once again to convince them to go with at traditional approach. They disagreed and actually complimented him on coming up with such a "clever" approach. He started development.

Nearly a year later, the system was complete. And it was a horrible mess. It required a server cluster to run the analysis, produced and executed horribly ugly and slow code at runtime, consumed a ridiculous amount of RAM, and needed at least a full week of training for each node. But it worked. Before heading back to academia, M.A. tried one last time to convince them to drop the neural network. They wouldn't budge.

Fast forward three years, and the programming firm was bought and given a completely new management team. The new CTO got wind of the behemoth that M.A. created and called him up to find out the reason behind such an "eccentric" implementation. M.A. explained everything and even forwarded the CTO a proposal that he created for a traditional solution long before starting the neural network.

The new CTO was more than pleased and wanted to bring M.A. in once again to develop the traditional solution. A few weeks later -- and after receiving another generous paycheck -- the new solution runs on a single server and is able to calculate, alert, and report on any issues within two minutes of a station updating their log file.

Not wanting to throw away a several hundred thousand dollar investment, the CTO decided to put the monstrous neural network to use doing something else: prose. The developers spent a few hours re-training it to work from a dataset of phrases and output actual sentences instead of tokenized log lines. The reports it generates included in a section of the company's newsletter. Following is one of its more recent creations.

The pig go. Go is to the fountain. The pig put foot. Grunt. Foot in what? ketchup. The dove fly. Fly is in sky. The dove drop something. The something on the pig. The pig disgusting. The pig rattle. Rattle with dove. The dove angry. The pig leave. The dove produce. Produce is chicken wing. With wing bark. No Quack.

Rather Zen, isn't it?