Symbol-Manipulation with Attractor Neural Networks

Frank van der Velde

Unit of Experimental and Theoretical Psychology, Leiden University

Wassenaarseweg 52, 2333 AK Leiden, The Netherlands, Bitnet: VDVELDE@HLERUL55

A fundamental issue in the implementation of symbol-manipulation with neural networks is the unlimited productivity of symbol-production systems. Because each realistic neural network is a finite-state system [1], unlimited productivity cannot be implemented with neural networks. Therefore, the only symbol-production which seems possible with neural networks is the production of a regular (finite-state) language. This has serious consequences for the generation of cognitive behaviour with neural networks, in particular natural language processing, which requires the productivity of non-regular production systems [2]. If it is not possible to implement such production systems in neural networks, neural network theory will remain on the level of behaviourism.

However, it should be realized that a digital computer is a finite-state machine as well. Yet, non-regular languages are successfully implemented on a digital computer. Therefore, actual symbol-production with neural networks should be compared with actual production on a digital computer. A good example is the production of a context-free language, which is the first non-regular language in the Chomsky hierarchy (many computer languages are context-free). Any actual production of a context-free language on a digital computer results in a regular subset of the context-free language because, after all, the computer is a finite-state machine. However, the subset can be increased by enlarging the working memory of the computer, without changing the production-program, that is, the context-free grammar itself.

Similar considerations should be made for symbol-production with neural networks [3]. Thus, implementation of, for instance, a context-free language with neural networks is possible if the actual set of sentences produced can be increased by changing the working memory without changing the production-scheme itself. This means that the working memory is an identifiable element of the system and can be changed without changing the system as a whole. Attempts have been made to implement a variable storage with neural networks by a feedback from the output neurons to the input neurons. Thus, information stored in the connection weights is combined with new information and again encoded in the connection weights. However, this procedure results in limited production even in practice, because the connection weights will soon become saturated.

I will show neural network memories which can be increased arbitrarily without changing the system as a whole. With these memories non-regular languages, such as context-free languages, can be produced with neural networks as well. The symbols are distributed patterns. Rules of the language (e.g., $S \rightarrow AB$, $S \rightarrow AC$, $C \rightarrow SB$, $A \rightarrow a$, $B \rightarrow b$) are stored as concatenated patterns in an attractor neural network (ANN). After presentation of the symbol (pattern) S to the left segment of the ANN, the network will settle in a pattern representing a whole rule. The other symbols of the rule (e.g., A or B) can either again be presented to the network, producing the 'words' a or b , or they can be stored in a working memory (WM), to be used later for symbol production. The WM is another ANN consisting of three segments. One segment contains the stored symbols, the other segments contain additional patterns needed to retrieve a sequence of symbols in the correct order. For instance, the stack $AABB$ is stored as r_1Ar_2 , r_2Ar_3 , r_3Br_4 , r_4Br_5 . The patterns r_i are created randomly in the WM. After presentation of r_5 to the right segment of the WM, the pattern r_4Br_5 will emerge. Shifting r_4 from the left to the right and presenting it to the WM will produce r_3Br_4 . Repeating in this fashion will retrieve the stack $BBA A$ in that order. The WM can be increased without changing the system as a whole. Thus, the competence of symbol-manipulation with neural networks equals that of classical production systems. In actual behaviour, however, there are differences between the systems, which shows the importance of the implementation in the generation of rule-like behaviour.

[1] Van der Velde, F. (1993). Is the brain an effective Turing machine or a finite-state machine? *Psychological Research/ Psychologische Forschung*, 55, 71-79.

[2] Prince, A. & Pinker, S. (1988). Rules and connections in human language. *TINS*, 11, 195-202.

[3] Van der Velde, F. (1993). Computing with patterns. (in preparation).