Parallel Processing in Neural Systems and Computers
R. Eckmiller, G. Hartmann and G. Hauske (Editors)
© Elsevier Science Publishers B.V. (North-Holland), 1990

Distributed Parallel Cooperative Problem-Solving with Voting and Election System of Neural Learning Networks

Byoung-Tak ZHANG and Gerd VEENKER

Division I – Artificial Intelligence
Institute for Computer Science
University of Bonn
D-5300 Bonn, FRG

In contrast to symbolic problem-solving systems, neural network systems have many salient properties such as parallelism, learnability, and fault-tolerance [1,2,5]. But it seems difficult, if not impossible, to construct an expert system in a single gigantic neural network [3,6]. In this paper we introduce a class of structured networks called VEN's (*Voting and Election Networks*) in which several different types of neural learning networks are organized into a voting and election system. The distributed parallel cooperative inference mechanism of the generic VEN system is discussed, an election scheme based on a network of sigma-pi units is presented, and the experimental results and observations on a two-person board game are reported. The VEN architecture appears to provide a framework for constructing knowledge-based problem-solving systems using neural networks as building blocks, while still maintaining the strengths of the neural computing.

1 The Voting and Election Model

The *voting and election network* (VEN) is a network whose components are neural learning networks that are organized into a voting and election system. The voting and election system consists of a *distributor*, one or more *voters*, and an *elector* (see Fig.1). The distributor distributes the input data among the voters. The voters cast their votes (hypotheses or constraints) for all candidates in parallel. It is the responsibility of the elector to determine the final solution that best satisfies the multiple hypotheses.

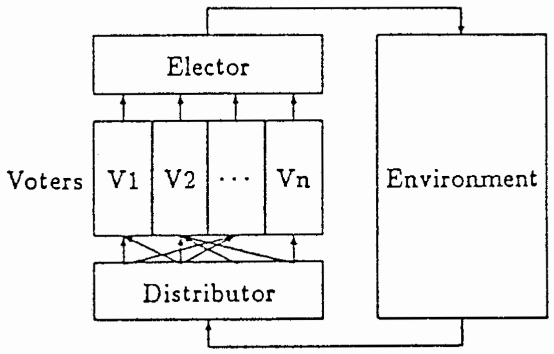

Fig.1. The organization of a VEN system with its environment

A *VEN system* is a VEN or a VEN of VEN systems. In VEN systems, the input data is partially shared among the voters, but the knowledge for solving a problem is divided into knowledge modules which are not shared among the voters in parallel voting. In general, a voter alone can not solve a problem, rather all the voters must cooperate and/or compete through the arbitration of the elector. It is noticeable that there are no constraints on the architecture and learning mechanism of the voter itself.

2 Distributed Parallel Cooperative Inference

Inferences in a voting and election system of neural networks (i.e. VEN) are performed at two different levels: at the level of neurons (*micro-inferences*) and at the level of problem states (*macro-inferences*). The problem-solving process of the system is a sequence of *conclusive* macro-inferences, each step of which corresponds to a state transition (or move) in the problem space.

At the beginning of each problem-solving step, current input data, $\mathrm{s}(t)$, describing the current situation of the problem is distributed among voting networks by the distributor $\mathcal{D}$, i.e.

$$\mathcal{D}(\mathrm{s}(t)) = \{\mathrm{x}_i\}.$$

In every conclusive inference step, a large number of *hypothetical* macro-inferences

$$\mathcal{V}_i(\mathrm{x}_i) = \mathrm{v}_i = (v_{i1}v_{i2}\dots v_{im})^T$$

are made simultaneously in each voting network where massively parallel micro-inferences are performed. The v_{ij} with $0 \le v_{ij} \le 1$ is the vote of the voter i that represents the probability of the candidate j being elected. The hypothetical macro-inference is duplicated as many times as the number of voters.

In general, election is a cooperative process that draws the conclusive macro-inferences by satisfying maximally the hypotheses of the voters. An election scheme will be described in the following section.

Continuous sequential macro-inference (i.e. problem-solving) is carried out by chaining (or feeding back) the output to the input of the next macro-inference step—possibly by means of an interaction with the environment.

3 The Strategic Election Scheme

Formally, election is a constraint-satisfaction (or optimization) process that produces a vector y

$$\mathrm{y} = \mathcal{E}(\mathrm{V}, \mathcal{S})$$

as a function of the votes V (*short-term constraints*) and the strategy $\mathcal{S}$ (*long-term constraints*), where $y_j = 1$ for the best candidate and $y_j = 0$ for the others. The function $\mathcal{E}$ and the strategy $\mathcal{S}$ can be implemented in various ways.

We present here an election network which consists of *sigma-pi* units (see Fig.2). The election involves two stages. In the first stage, the election units act as pi-units (p_j) and the votes v_{ij} for the candidate j, each v_{ij} weighted by the corresponding connection weights s_{ij}, are multiplied to a product

$$y_j(0) = \prod_{i=1}^{n} s_{ij}v_{ij} = \prod_{i=1}^{n} a_i b_j v_{ij} = b_j \prod_{i=1}^{n} a_i v_{ij} = b_j p_j \tag{1}$$

which builds the input to the next stage. In (1) the matrix

$$\mathrm{S} = (s_{ij}) = \mathrm{ab}^T = (a_i b_j)$$

represents the *privileges* that reflect the relative importance of the connections between the voters i and the candidates j. The vectors a and b will be called *privilege vectors* and the matrix S *privilege matrix* or *strategy matrix*. The separation of a and b vectors and the use of the intermediate units p_j make the strategy modification easier.

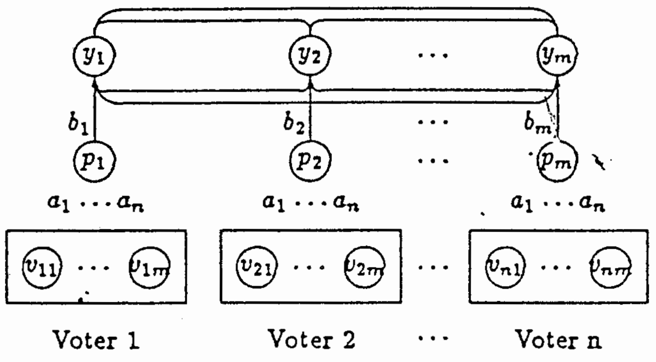

Fig.2. The election network

In the second stage, the election units act as sigma-units (y_j) and the winner-take-all [1] competition process

$$y_j(t) = \epsilon \sum_{k=1}^{m} c_{jk} y_j(t-1) \tag{2}$$

is repeated until a candidate (or l candidates in case of l-winner-take-all) takes all the votes. It is worth noting from Equations (1) and (2) that a candidate has a very low probability of winning if a voter casts him a strictly negative vote (near 0). This is the effect of the characteristic of the pi-units.

4 An Example: Playing a Two-Person Board Game

To test the voting and election model and the strategic election scheme, we have constructed a VEN system which learns to play a two-person board game. In the game, the two players move their figures alternately on a 3 by 3 game board. The figure can move only one square horizontally or vertically at a time. A player wins the game when he conquers the home of the opponent or captures the opponent's figure. The home of the player is the position from which the player started the game. A player can capture the opponent's figure if it is in the position to which the player can move legally.

Attempting to move a step, in the mind of the player, at least the five *little minds* are cooperating and/or fighting: reaching the opponent's home, capturing the opponent's figure, not being captured, avoiding loops, and moving legally. We have implemented these little minds as the voters in the VEN system. Each voter was implemented as a three-layer (or two-connection-layer) feedforward network. The election network described above was used in election. The five voting networks were trained with examples from typical situations by a back-propagation learning algorithm [5]. A result of voting and election for board positions to move to is shown in Fig.3 (see [7] for details).

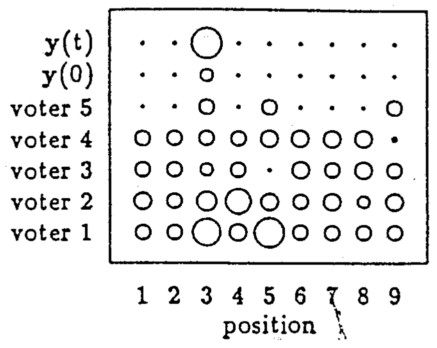

Fig.3. The voting and election for board positions.
The diameter of circles reflects the strength of hypotheses.

5 Concluding Remarks

The game-playing VEN system has predicted optimal moves for the situations, for which it is not trained explicitly as a whole. From the experiments with the system we have made the following observations. The voting and election formulation of problem-solving makes it possible to realize a synchronic distribution of the entire burden among voting networks. This renders parallel processing feasible and the macro-level distribution of knowledge possible. The knowledge distribution helps further in acquiring knowledge easily in terms of a single neural network, which also results in the enhancement of reusability of the knowledge. Besides the cooperative election, the indirect interaction and partial duplication of knowledge among voting networks through the sharing of data and training examples also contribute to the robustness of the inference. Although these observations should be proved more rigorously in different problem domains, the VEN architecture appears to provide a framework for constructing distributed parallel cooperative problem-solving systems for Artificial Intelligence. In the future, attempts will be made on the self-organization of the VEN systems using the (co-)evolutionary learning concepts, e.g. [3,4].

Acknowledgements

We would like to thank Heinz Mühlenbein, Jörg Kindermann, Knut Möller, and the members of the Arbeitsgemeinschaft Neuronale Netze for useful discussions and comments. Thanks are also extended to the POSCO Scholarship Society for the financial support.

References

[1] Feldman, J.A. and Ballard, D.H., Connectionist models and their properties, *Cognitive Science 6*, pp. 205-254 1982.

[2] Hinton, G.E., Connectionist learning procedures, *Artificial Intelligence 40*, pp. 185-234, 1989.

[3] Mühlenbein, H. and Kindermann, J., *Distributed problem solving by coevolution*, Internal Report, German National Research Center for Computer Science (GMD), 1989.

[4] Mühlenbein, H. and Kindermann, J., The dynamics of evolution and learning—Towards genetic neural networks, In: Pfeifer, R. et al.(eds.), *Connectionism in Perspective*, Elsevier, 1989.

[5] Rumelhart, D.E and McClelland, J.L. (eds.), *Parallel Distributed Processing*, MIT Press, 1986.

[6] Smolensky, P., Connectionist AI, symbolic AI, and the brain, *Artificial Intelligence Review 1*, pp. 95-109, 1987.

[7] Zhang, B.T., *Playing a two-person board game with a voting and election system of neural learning networks*, AGNN-Memo, Inst. for Comp. Sci., Div.I-AI, Univ. of Bonn, 1989.