Robust Character Recognition using a Hierarchical Bayesian Network

John Thornton, Torbjorn Gustafsson, Michael Blumenstein, and Trevor Hine

Institute for Integrated and Intelligent Systems, Griffith University, QLD, Australia
{j.thornton,t.gustafsson,m.blumenstein,t.hine}@griffith.edu.au

Abstract. There is increasing evidence to suggest that the neocortex of the mammalian brain does not consist of a collection of specialised and dedicated cortical architectures, but instead possesses a fairly uniform, hierarchically organised structure. As Mountcastle has observed [1], this uniformity implies that the same general computational processes are performed across the entire neocortex, even though different regions are known to play different functional roles. Building on this evidence, Hawkins has proposed a top-down model of neocortical operation [2], taking it to be a kind of pattern recognition machine, storing invariant representations of neural input sequences in hierarchical memory structures that both predict sensory input and control behaviour. The first partial proof of concept of Hawkins' model was recently developed using a hierarchically organised Bayesian network that was tested on a simple pattern recognition problem [3]. In the current study we extend Hawkins' work by comparing the performance of a backpropagation neural network with our own implementation of a hierarchical Bayesian network in the well-studied domain of character recognition. The results show that even a simplistic implementation of Hawkins' model can produce recognition rates that exceed a standard neural network approach. Such results create a strong case for the further investigation and development of Hawkins' neocortically-inspired approach to building intelligent systems.

Key words: Bayesian network, cerebral cortex, character recognition

1 The Hierarchical Bayesian Network

Figure 1 presents a simplified example of the hierarchical Bayesian network used in the current study. This network is based on George and Hawkins' original implementation [3] and consequently does not follow standard Bayesian update procedures, as the paper will explain. The network input consists of a feature vector of 100 real-valued elements, with a value range from 0 to 1. The particular form of the input was set by the feature extraction method used in the neural network implementation (described in Section 2). We retained this feature vector for the Bayesian network to ensure that our results reflect differences between the two computational architectures and are not confounded by effects from differing representations of the problem domain.

The Learning Phase: In Figure 1, each level one node receives a real number input from the feature vector, which is then classified into one of eight bit patterns or as an empty pattern. In the example, a bit set in position one classifies values > 0 and < 0.125, in position two ≥ 0.125 and < 0.25 and so on, with at most one bit set in any pattern. Bit patterns from level one are then sent to level two, where they are classified again, as follows: the level one eight bit patterns are projected into level two four bit patterns, such that if there is any bit in the first or second position of a child bit pattern, then position one in the level two bit pattern is set, and so on. The network undergoes supervised learning, so when the level two bit patterns are sent to level three they are then correctly classified against the actual character input.

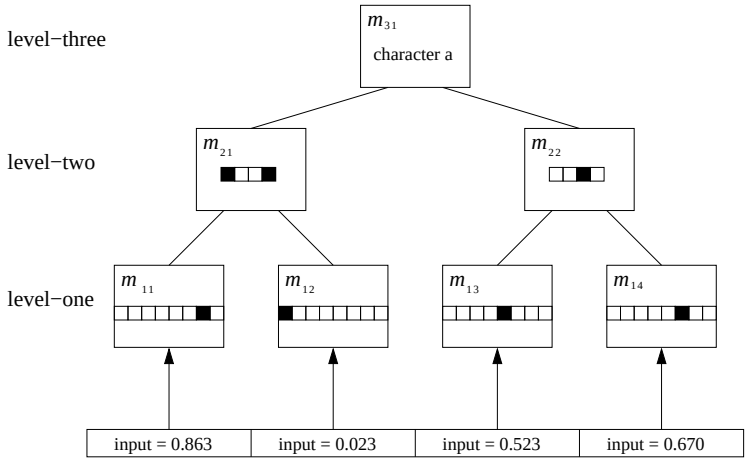

Fig. 1. A simple hierarchical network in the process of supervised learning.

In the full implementation, a similar procedure is followed, except there are 100 level one nodes, each classifying the feature vector inputs into a bit pattern with up to 100 positions. These level one nodes then connect in groups of twenty-five to one of four level two nodes. The twenty-five associated level one bit patterns are then projected into single level two bit patterns of the same length to create level two language elements. Each time a level two module receives a new set of level one bit patterns, it checks to see if these patterns are classified under an existing level two language element (i.e. that they project onto an existing level two bit pattern). If not, a new level two language element is created and given an index. In either case, the index value of the level two language element is passed back to the level one modules. Each level one module then stores how many times a particular level one bit pattern has been classified against each level two language element. This is shown in left hand matrix in Figure 2. When learning is complete, this matrix is normalised by dividing each row element by

the row sum to produce the *conditional probability distribution* (CPD). CPDs are generated in the same way for each level one and level two module (the level three root node does not generate a CPD because it does not communicate with a higher node).

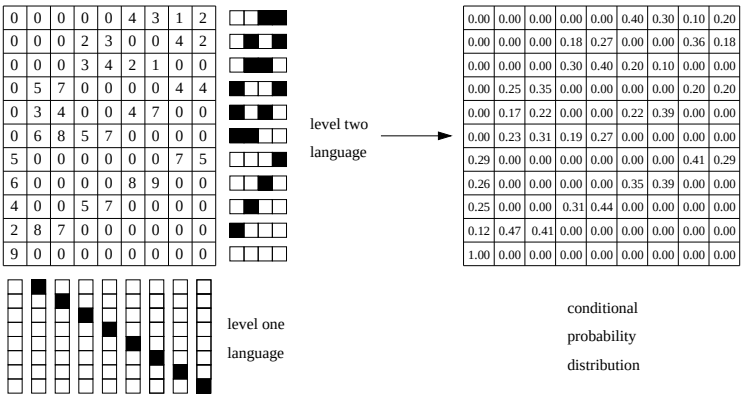

Fig. 2. Calculating a level one conditional probability distribution (CPD).

The Recognition Phase: The details of the recognition phase calculations are shown in Figure 3 for an internal node in the hierarchy. Both the level one nodes and the root node are special cases in that they only receive input either from above or below. In the root node case, it initially sends down a π vector containing m normalised equal probabilities, where m is the number of character types the network has been trained to recognise. In addition, a level one node needs to generate a λ vector containing the probabilities for each of the n possible language elements in the current input.

If we consider node m_{11} in relation to Figure 3, m_{11} will initially receive a π vector from level two (in the top left of the diagram) with all elements set to one (this means the level two node has no belief about which of its language elements are active). Then each column of the CPD matrix is multiplied by the corresponding λ vector element and each row by the corresponding π vector element (as all π elements equal one, only the λ elements will have an effect in the first iteration). This produces the array on the right of Figure 3. We then take the maximum likelihood value for each row to form the λ^{k+1} vector that is then sent to level two. We also take the maximum likelihood for each column to form the π^{k-1} vector. As we are at level one, this vector cannot be sent down, although it can be used to condition the probabilities of the next λ input vector (assuming we are seeing a meaningful temporal sequences of inputs).

Following the path of the λ^{k+1} vector upwards, the level two node will receive one such vector from each of its children. These vectors are combined by multiplying together all elements in corresponding positions to produce a new vector with the same number of elements. This process is repeated at level two,

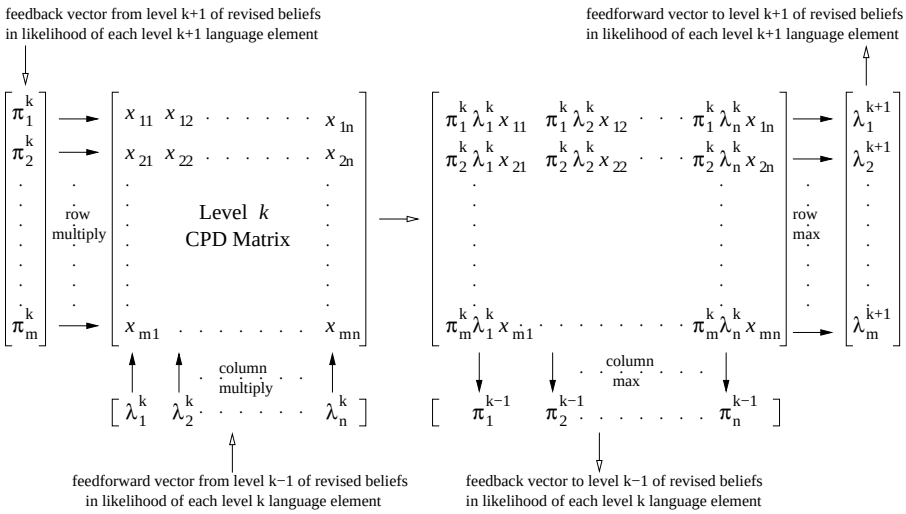

Fig. 3. Belief revision calculations for recognition.

producing a new λ vector at level three, containing the combined beliefs of all level two nodes about which level three language element is active. As the level three language elements are the actual characters we are trying to recognise, the solution to the character recognition problem is the λ element at this level with the highest probability.

2 Experimental Study

For the neural network implementation we used a modification to the backpropagation algorithm known as the Resilient Version (RPROP) [4]. RPROP is a locally adaptive learning scheme for batch learning in feed-forward neural networks. Because it only examines the direction of the gradient and uses a local adaptive approach to determine the size of the weight change, it generally converges faster than a standard backpropagation approach and has the added advantage that it does not require the constant modification of parameters to achieve near-optimal performance and convergence. The transition feature vector used to represent the input was calculated using the location of transitions from background to foreground pixels in the vertical and horizontal directions of a given character image (see [5] for more details).

The neural network and Bayesian implementations were tested on the well-known CEDAR database using the Buffalo Designed (BD) dataset of handwritten digits. [1] From the training set, six subsets were created containing 10, 25, 50, 100, 200 and 300 training digits per category and for testing, all (544) of the digits from the test set were used.

[1] See http://www.cedar.buffalo.edu/Databases/CDROM1/

RPROP: Table 1 shows the best RPROP recognition rate was 92.28% achieved using 32 hidden layer neurons and 500 iterations. The results are for the best of five runs and show the effects of varying the number of iterations and the number hidden units.

Table 1. RPROP character recognition results as percentages.

Number of Hidden Units	Number of Iterations				
	100	200	300	400	500
18	88.97	90.99	91.17	91.17	90.99
24	91.36	90.44	90.99	91.54	89.34
32	90.63	91.91	90.44	91.91	**92.28**

Bayesian Network: As previously described, we constructed a hierarchical Bayesian network in three levels with 100 level one nodes. During the initial experimentation we found that a tree structure with four level two nodes consistently produced better results, so we retained this structure for all the following experiments. However, it was unclear what size training set produced the best result, so we tried five different values: $[25, 50, 100, 200, 300]$. We also varied the number of level one elements between 26, 51, 76 and 101 elements and experimented with putting an upper limit on the size of the level two language. Once this limit is reached, any new language elements are classified under an existing element according to the minimum Hamming distance.

Table 2 shows that the best Bayesian recognition rate of 96.32% was achieved for a training set category size of 300, a level one language size of 76 and a level two language limit of $12,000$. However, recognition rates that exceeded the best RPROP results were achievable with a smaller level two language limit of $2,600$ and a faster mean recognition time of 4.39 seconds per character.

Table 2. Hierarchical Bayesian network character recognition results as percentages. Results that exceeded the best RPROP recognition rate are in bold, of these the 92.65% result had the best mean recognition time of 4.39 seconds per character and an overall training time of 12.19 seconds and the best result of 96.32% had a mean recognition time of 7.83 seconds per character and an overall training time of 15.99 seconds. All experiments were performed on an AMD Athlon 64 3000+ processor with 1 GB RAM.

Level One Alphabet Size	Training Set Size / Level Two Alphabet Limit					
	25/1000	50/2000	100/2600	200/2600	300/2600	300/12000
26	87.13	89.34	91.36	89.70	90.80	n/a
51	87.50	90.80	**92.65**	90.63	89.15	n/a
76	88.24	90.07	89.89	89.89	87.32	**96.32**
101	88.24	90.44	91.36	87.50	83.46	n/a

3 Conclusions

The final best recognition rate of 96.32% for the hierarchical Bayesian network compares very favourably with the 92.28% rate for RPROP. If we bear in mind that character recognition has been the province of neural network research for many years, and that the Bayesian network was doing recognition on a feature vector structure specifically developed for a neural network approach, then our results strongly recommend the new hierarchical Bayesian approach. In addition the results show that the Bayesian network can still achieve good recognition rates on quite small training sets. This is a distinct advantage in real world situations where only a small subset of the possible input is likely to be available for training. The main disadvantage of the hierarchical Bayesian approach is that it involves considerable computational overhead. While in the neocortex this overhead is counteracted by the massively parallel nature of neocortical processing, the matrix arithmetic required to propagate belief revision throughout the Bayesian network can result in recognition times lasting several seconds, in contrast to the virtually instantaneous recognition times for RPROP (this is partly balanced by the Bayesian training times being at least ten times faster than for RPROP). There are several ways of approaching this speed problem. One is to construct hardware that allows the matrix operations to occur in parallel (George and Hawkins are already exploring this option). A second more immediate approach is to develop a selective method for updating beliefs, such that strong beliefs tend to suppress similar, weaker beliefs. In this way the flow of information in the network could be considerably reduced. It would also be worth investigating problem representations that are more "natural" for the hierarchy, such as going back to original character images rather than using feature vectors. It should also be noted that this Bayesian network is only a partial implementation of Hawkins' model of the neocortex. Further improvements can be expected over a neural network approach in the area of detecting *moving* objects that allow the feedback properties of the network to influence recognition.

References

1. Mountcastle, V.: An organizing principle for cerebral function: the unit model and the distributed system. In Edelman, G., Mountcastle, V., eds.: The Mindful Brain. MIT Press, Cambridge, Mass (1978)
2. Hawkins, J., Blakeslee, S.: On intelligence. Henry Holt, New York (2004)
3. George, D., Hawkins, J.: A hierarchical Bayesian model of invariant pattern recognition in the visual cortex. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN-05). (2005)
4. Riedmiller, M., Braun, H.: A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In: Proceedings of the IEEE International Conference on Neural Networks, San Francisco, CA (1992) 586–591
5. Gader, P.D., Mohamed, M., Chiang, J.H.: Handwritten word recognition with character and inter-character neural networks. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics **27** (1997) 158–164