Robustness of Tensor Product Networks Using Distributed Representations

William H. Wilson
Computer Science and Engineering
University of New South Wales
Sydney NSW 2052 Australia
billw@cse.unsw.edu.au

Graeme S. Halford
Psychology
University of Queensland
Queensland 4072 Australia
gsh@psy.uq.edu.au

ABSTRACT

This paper describes experiments on on the robustness of tensor product networks using distributed representations, for recall tasks. The results of the experiments indicate, among other things, that the degree of robustness increases with the number of binding units and decreases with the fraction of the space of possible facts that have been taught to the network. Mean recall scores decrease linearly with the proportion of binding units inactivated, and recall score variance depends linearly on number of binding units and on number of facts taught to the network.

1. Introduction

This paper describes experiments on the robustness of tensor product network of ranks between 2 and 7. In the experiments, varying numbers of randomly selected nodes in the network were "killed" by changing them so that they always produced zero output, and then the performance of the resulting network was assessed. The results indicate the effect of varying the rank of the tensor product network, the effect of varying the proportion of neurons killed, and the effect of increasing the length of the vectors used to represent concepts in the network (i.e. the length of each axis of the tensor). Most of the experiments were conducted with 85% of the binding units inactivated, and in this condition, at least for reasonable numbers of facts, facts could reliably be distinguished from non-facts.

2. Tensor Product Networks

Tensor product networks have been used as one-shot learning systems for applications like variable binding [6], cognitive modelling [2,3,4], and for memory in connectionist implementations of production system architectures [1]. A tensor product network has a *rank*: the rank of each network used in the experiments described in the papers just cited was 2 or 3; we will introduce tensor product networks mainly in terms of the rank 3 case.

* This work was supported by Australian Research Council grant A79700056

A rank 3 tensor product network has 3 dimensions - of size p, q, and r, say. Its processing units include pqr *binding units*, and input/output units grouped into 3 vectors of length p, q and r. Typically one of the I/O vectors represents a *predicate*, and the other two represent a pair of *arguments*. The information represented in the tensor (i.e. in the binding unit structure) is thus relational information, such as *larger–than(mare, foal*). With rank 2 networks, a common use is to represent a single relation (say *larger–than*), and the two axes of the tensor would thus represent the two items that, as a pair, belong to that relation, - e.g. *(mare, foal)*. The relation may be specialized, as in [6] to a functional relation, e.g. binding between members of a set of variables and a set of values.

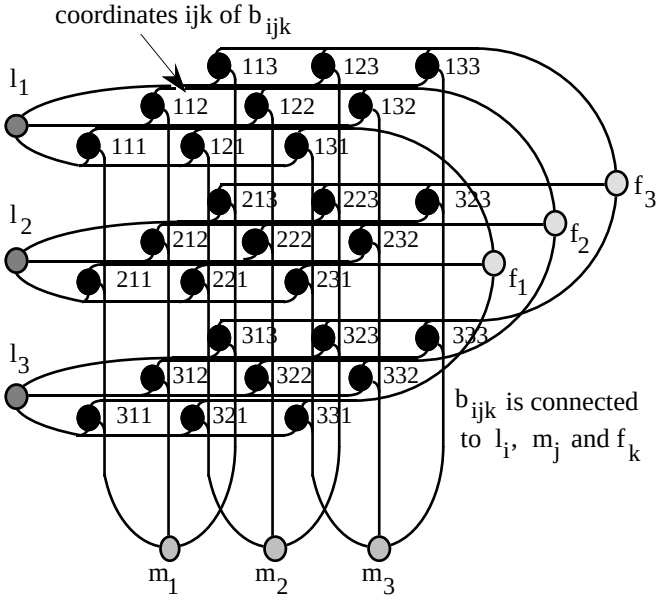

Figure 1: Network connectivity for a 3x3x3 tensor product network

To store the fact *larger–than(mare, foal)* in a rank 3 tensor, one computes the outer product *larger–than*⊗*mare*⊗*foal* of vectors representing the concepts *larger–than*, *mare*, and *foal* and adds it to the values stored in the binding units: thus if l represents *larger–than*, m represents *mare*, and f represents *foal*, one would add $l_i * m_j * f_k$ to the value stored in binding unit b_{ijk}.

Figure 1 shows binding units, input/output units, and connections in a 3x3x3 tensor product network.

If precise recall of facts is needed, it can be obtained by using tensor product networks based on orthonormal sets of representation vectors, with *exact unbinding* [6]. As an aim of many neural network models is to provide a distributed representation of the concepts involved, it is desirable for the representation vectors to have a high proportion of non-zero components. This can be achieved in a systematic way by using the rows of a Hadamard matrix, suitably normalised, where an nxn Hadamard matrix is a square matrix all of whose entries are ±1, such that $HH^T = nI_n$. Figure 2 shows a 4x4 Hadamard matrix.

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

Figure 2: a 4x4 Hadamard matrix.

Hadamard matrices of size nxn are known to exist for a wide range of n divisible by 4 (including all n of the form 2^m) [5]. One can then associate with each concept that is to be represented a representation vector that is a member of an orthonormal basis for the representation space and which has no zero components. Facts involving the concepts of interest can be stored in the binding units.

A simple mechanism then allows retrieval ("unbinding") of the stored information. For example, given **m** (*mare*) and **f** (*foal*), one can retrieve the relationship(s) between them (such as **l** (*larger-than*)) that are stored in the tensor. In general, more than one relationship may hold between the arguments (e.g. *mother-of(mare, foal)*). Thus what is retrieved will be a sum of vectors representing predicates - we term this sum a *predicate bundle*. As well, given all three "pieces" one can check whether the tensor holds this relational instance, and in fact, given any one "piece", one can retrieve from the tensor a tensor of lower rank representing the relational instances involving that "piece". In the case of retrieving predicates given **m** (*mare*) and **f** (*foal*), the computation is as follows, where p_i signifies the i-th component of the predicate bundle retrieved:

$$p_i = \sum_{jk} b_{ijk} * m_j * f_k \qquad (1)$$

For a simple C program that does computations with rank 3 tensor product nets, see [7].

3 Robustness and Tensor Product Networks

In [8], Wilson & Halford considered robustness of recall and of analogical problem solving using rank 3 tensor product networks, using 2 methods of damaging the nodes in the tensor product network. The first method was to kill nodes at random. The second was to alter nodes so that they produced random noise when accessed - the amount of noise chosen in a uniform random way from the interval $[-\rho,\rho]$, where ρ was thought of as a kind of volume setting for the noise. It was found that the random noise interfered to a much greater extent with both recall and problem solving: up to 80% of nodes could be killed without destroying performance on analogical problem solving, while with $\rho = 0.1$, analogical problem solving became impossible when a little more than 50% of nodes were randomized. They considered that a fact was successfully recalled if its recall score (see equation (2), below) was greater than an arbitrarily chosen cutoff of 0.4. The paper reports that "all known facts could often be recalled until more than 30% of units [had] been destroyed." After this, the proportion of facts recalled dropped off: by the time 70% of nodes had been killed, only about 20% of facts could be recalled (by this criterion).

In the present paper, a different criterion for recall is implicitly used. When nodes are killed, the amount of information used to calculate the recall score of equation (2) below is reduced, and recall scores thus naturally drop off. The real question, in the authors' opinion, is whether one can distinguish between facts and non-facts.

The recall experiments of Wilson & Halford with randomized units found that the effect of randomization on recall performance was similar to that of killing nodes in that when up to 30% of nodes were randomized ($\rho = 0.1$) all facts could still be recalled, but that the proportion of recallable facts then dropped off much more steeply than with killed units.

Wilson & Halford also established that distributed representations were much more robust than local representations in tensor product networks.

In [9], a phenomenon related to robustness was investigated. The authors observed, with biological plausibility in mind, that normalized rows of Hadamard matrices did not seem likely to occur naturally as representation vectors. They noted the robustness results in [8], which they claimed showed that absolute orthornomality was not strictly necessary in representation vectors for the tensor product network to perform adequately, and they proposed that a certain type of random representation might work fairly well[1]. The experiments described in that paper demonstrated that two kinds of such representation, dubbed *dense* and *sparse* random representations, allowed successful analogical problem solving most of the time, provided that there were a reasonable number of components in each dense random representation vector, or a reasonable number of non-zero components in each sparse random representation vector. That paper did not look at recall tasks.

4 The Experiments

To illustrate the experiments carried out, suppose that we are dealing with a rank 3 network with 4 predicate concepts $P1, P2, P3, P4$, and 4 argument concepts, $a1, a2, a3, a4$. Networks were trained by first generating a predetermined number of "facts" at random. To generate a random fact, first a predicate would be chosen at random (using a uniform pseudo-random number generator) from $P1$ to $P4$, then two random arguments would be chosen from $a1$ to $a4$. If $P2$, $a4$, and $a1$ were chosen, then the random fact would be $P2(a4, a1)$. Once the facts, say $F^m = P^m(a1^m, b2^m)$, $m = 1,...,n$, were chosen, the network would be taught those facts by storing in binding unit b_{ijk} the sum $\sum_m P^m{}_i\, a1^m{}_j\, a2^m{}_k$. The number n of facts was one of the parameters varied in the experiments.

After the network had been trained in this way, binding units in the tensor product were chosen for killing. Units to be killed were chosen in a random way: for each axis of the tensor a coordinate number was chosen in a uniform

[1] Random representations may also not occur naturally, but are somewhat more plausible than Hadamard representations.

random way - the assemblage of coordinate numbers designated a particular binding unit. If the unit so selected happened to be already "dead", then another unit was chosen in the same way. The percentage of units to kill was also a parameter varied in the experiments.

The methods used to test the effect of damaging the network were to kill a certain percentage of the units and then to measure the performance of the network on recalling the facts that it had been taught, and also in attempting to "recall" a collection of non-facts. The non-facts were chosen at random in the way described above for the facts (subject to checking that the non-facts did not coincide with any fact!) It would be possible, in principle, to check every possible non-fact, but in practice in large fact spaces, this would be prohibitively time-consuming, so in the experiments, one non-fact was chosen for every fact.

The measure of recall of a known fact in an intact network is 1.0, computed as shown below for the rank 3 case:

$$recall(P(a1,a2)) = \Sigma_{ijk}\, b_{ijk} * P_i * a1_j * a2_k \qquad (2)$$

The other parameters varied in the experiments included the rank r of the tensor, and the length d of the representation vectors. These two together determined the total number d^r of binding units. Mostly there was a goal of keeping the total number of binding units, if not the same, then at least of the same order of magnitude. In one sequence of experiments, however, the rank was held constant and the length d of the representation vectors was varied to see the effect of this variable.

The practice of generating facts at random is not entirely desirable: the pattern of real facts in the cartesian product space $P \times A_1 \times ... \times A_{r-1}$ (P = set of predicate symbols; A_i = set of possible ith arguments) is not normally random. On the other hand, it is difficult to see how to generate successively larger sets of real facts in a natural way, and if some of the fact sets consisted of real facts and some were artificial, this would introduce worse problems. Some results with small, fairly natural sets of facts are reported in [8], where natural sets of facts were essential for the analogical reasoning tasks.

For each set of random facts generated, 10 runs were done, randomly "killing" a predetermined proportion of the binding units.The information computed in the course of the experiments included the mean and variance of the recall scores for each fact over the 10 runs, the smallest and largest recall scores over the 10 runs, and information to allow a histogram of all recall scores over the 10 runs to be produced.

5 Experimental Results

The term *length* is used below to refer to the number of components in a representation vector. The number of binding units in a tensor product network is the *length* raised to the power indicated by its rank. Figure 3 shows the effect of length on recall. It can be seen that length has a significant effect on the spread of recall scores. In each case the average recall scores for facts was about 0.15 (range 0.1499 to 0.1599), and for non-facts was about 0.0 (range –0.00027 to 0.00927). The effect of length is confounded with proportion of the fact space used, as while in each case there were 250 facts generated, the number of *possible* facts (that is, the number of atomic propositions expressible using the given predicate and argument symbols) varies from $16^3 = 4096$ to $128^3 = 2097152$. In fact, as Table 1 shows, for a fixed number of facts, the number of binding units more or less determines the variance of the recall scores. However, as in practice there is usually a limit on the size of memory - and thus on the number of binding units, and this number is jointly determined by rank and length of representation vectors, so for fixed rank, length has an effect on the spread of recall scores.

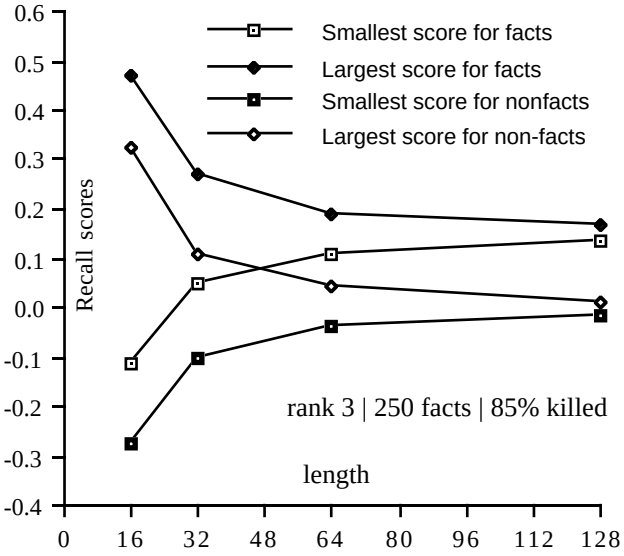

Figure 3: Effect of length on recall scores for facts and non-facts.

Rank	Number of binding units	Variance of fact recall scores, 500 facts	Variance of fact recall scores, 4000 facts
2	1048576	0.000062	0.000454
3	2097152	0.000030	0.000245
4	1048576	0.000062	0.000475
5	1048576	0.000062	0.000493
6	262144	0.000245	0.001946
7	2097152	0.000030	0.000242

Table 1: Effect of number of binding units on variance of fact recall. Variances for non-fact recall are not shown, but are almost the same as for fact recall for the same number of facts.

The *average* recall score was a more or less linear function of the proportion of units killed, as shown in Figure 4. The distribution of recall scores for facts was

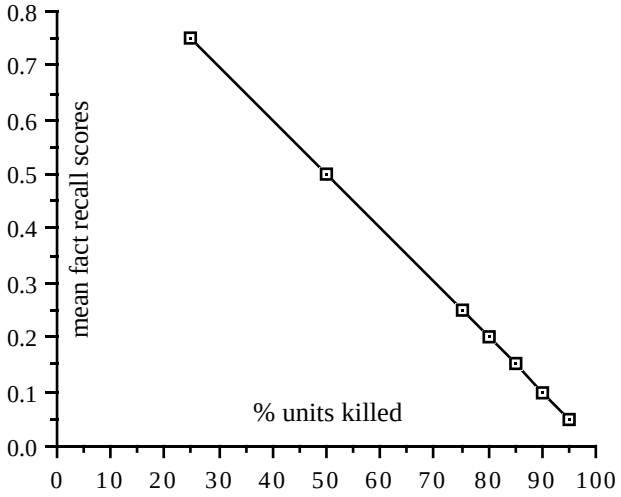

Figure 4: Effect of proportion of binding units killed on average recall score for facts.

more or less bell-shaped, as exemplified in Figure 5 (upper panel). Recall scores for non-facts are also roughly bell-shaped, though with a higher peak and narrower spread.

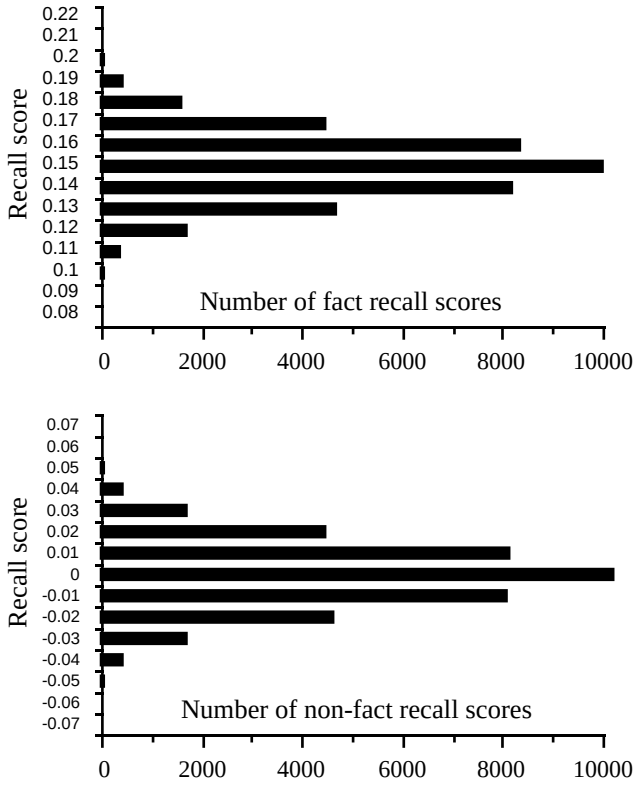

Figure 5: Histogram of pooled recall scores for 10 rank 3 tensor product networks trained on 4000 facts, and with 85% of units killed, for facts (upper panel) and non-facts (lower panel).

The issue of distinguishing between facts and non-facts is addressed in Figure 6, which graphs the largest recall score for a non-fact along with the smallest recall score for a fact, for two different numbers of binding units: if there are enough binding units in relation to the number of facts taught to the tensor, then facts and non-facts can be distinguished. In the runs summarised in Figure 6, 85% of binding units were killed: panel (a) shows rank 3 / 1048576 binding units and panel (b) shows rank 7 / 2097152. As demonstrated in Table 1 and the text that refers to it, it is the number of binding units that makes the difference.

6 Conclusions and Discussion

Tensor product networks using orthonormal distributed representations for the activations projected into the network along the input axes are relatively robust to destruction of individual neurons.

A significant factor in the degree of robustness exhibited is the density of facts in “fact space” - that is, what proportion of all expressible propositions have been taught to the network as facts.

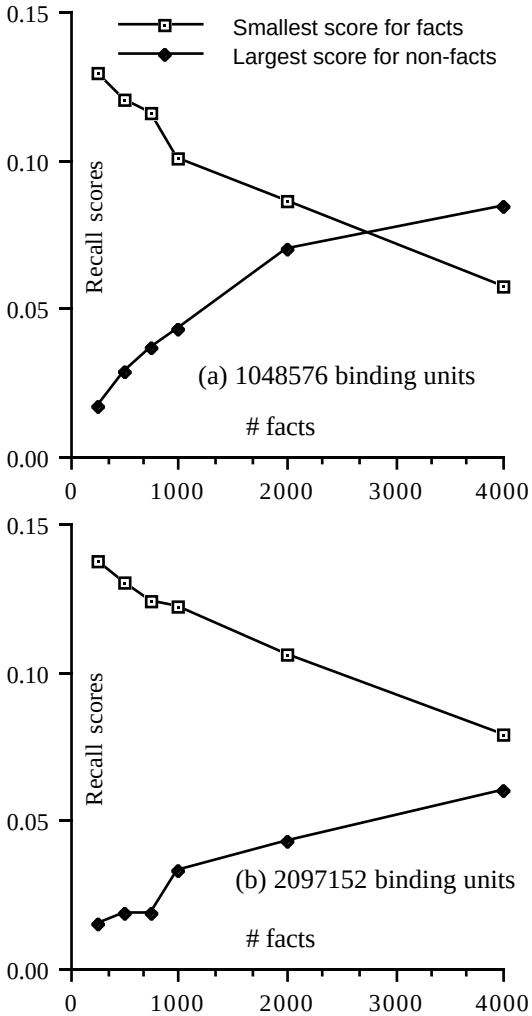

Figure 6: Distinguishing Between Facts And Non-Facts On The Basis Of Recall Scores

The variance of fact recall scores over a number of runs (deleting different binding units at random) is inversely proportional to the number of binding units, and the number of binding units is proportional to the cube of the length of the vectors used for representing concepts (or to the product of the lengths if different lengths are used for

different axes of the tensor) so that using longer representation vectors will increase the ability of the network to distinguish between facts and non-facts.

References

[1] C.P. Dolan and P. Smolensky, Tensor product production system: A modular architecture and representation, *Connection Science* **1** (1989) 53-68.

[2] G.S. Halford, J. Wiles, M.S. Humphreys, & W.H. Wilson, Parallel distributed processing approaches to creative reasoning: Tensor models of memory and analogy,in *AI and Creativity* ed.T Dartnall, S.Kim, and F. Sudweeks, AAAI Technical Report (1993)

[3] G.S. Halford, W.H. Wilson, J. Guo, J. Wiles, and J.E.M. Stewart (1994), Connectionist implications for processing capacity limitations in analogies, in K.J. Holyoak and J. Barnden (editors) *Advances in Connectionist and Neural Computation Theory: Volume 2: Analogical Connections*, Norwood, NJ: Ablex.

[4] G.S. Halford, W.H. Wilson & S. Phillips, Processing capacity defined by relational complexity: Implications for comparative, developmental and cognitive psychology, to appear as a target article in *Behavioral and Brain Sciences.*

[5] J. Seberry & M. Yamada, Hadamard matrices, sequences, and block designs, pp. 431-560 in *Contemporary Design Theory: A Collection of Surveys*, ed. J.H. Dinitz and D.R. Stinson, John Wiley, 1992

[6] P. Smolensky, Tensor product variable binding and the representation of symbolic structures in connectionist systems, *Artificial Intelligence* **46** (1990) 159-216

[7] William H. Wilson, A C program to perform analogical problem solving (simple proportional analogies) using tensor product networks. [URL:ftp://ftp.cse.unsw.edu.au/pub/users/billw/star.c] 17KBytes.

[8] William H. Wilson and Graeme S. Halford, Robustness of tensor product networks using distributed representations, in *Proceedings of the Fifth Australian Conference on Neural Networks, ACNN'94*, edited by A.C. Tsoi & T. Downs, Brisbane, 31 January-2 February 1994, 258-261.

[9] William H. Wilson, Deborah J. Street, and Graeme S. Halford, Solving Proportional Analogy Problems using Tensor Product Networks with Random Representations, *1995 IEEE International Conference on Neural Networks Proceedings, ICNN'95*, Perth, November 1995, 2971-2975. ISBN 0 646 26352 8