Representation of variables and their values in neural networks

Janet Wiles
Departments of Computer Science & Psychology
University of Queensland QLD 4072 AUSTRALIA
janetw@cs.uq.oz.au

Abstract

Neural nets (*NNs*) such as multi-layer feedforward and recurrent nets have had considerable success in creating representations in the hidden layers. In a combinatorial domain, such as a visual scene, a parsimonious representation might be in terms of component features (or variables) such as colour, shape and size (each of which can take on multiple values, such as red or green, or square or circle). Simulations are described demonstrating that a multi-variable encoder network can learn to represent an input pattern in terms of its component variables, wherein each variable is encoded by a pair of hidden units. The interesting aspect of this representation is that the number of hidden units required to represent arbitrary numbers of variables and values is linear in the number of variables, but constant with respect to the number of values for each variable. This result provides a new perspective for assessing the representational capacity of hidden units in combinatorial domains.

Introduction

Cognitive models are designed to explicate the processes that underlie cognitive phenomena. However, the representations over which such processes operate must either be specified *a priori* in a model, or learnt within the model. NNs provide a concrete mechanism for thinking about ways in which such representations could be learnt in cognitive tasks. The application of NN to representation construction has produced many illustrative simulations (e.g., Rumelhart, Hinton & Williams, 1986; Elman, 1988) which in effect, provide metaphors for thinking about both the potential and limitations of the representational aspects of learning. There has been much debate concerning the representational adequacy of NN, specifically focussed on their compositional and systematic properties (e.g., see Fodor & Pylyshyn, 1988; and responses by Smolensky, 1988; van Gelder, 1990; & the special issue of *Artificial Intelligence* edited by Hinton, 1990). Although, theoretically the representational questions have been addressed, questions remain concerning ways in which a learning mechanism could induce the structure inherent in a combinatorial domain. In previous work, we showed that combinatorial structure in a simple domain (3 colours, 3 shapes and 2 locations) can be represented by a multi-layer network in terms of intersecting regions in HU space (Wiles & Ollila, 1992). In a study of generalization performance in combinatorial domains, Brousse and Smolensky (1989) report that multi-layer nets trained on subsets of a combinatorial task generalized to a large fraction of the domain, and with only a little extra training could generalize to far more. The current study explores representations as the number of values for each variable in a combinatorial domain is increased.

Review of structures in HU space

In a multi-layer net, a pattern of activation on the input units is transformed (via the first matrix of weights) to a pattern of activation on the HU layer. This pattern of activation is then transformed (via the second matrix of weights) to a pattern of activation on the output unit layer. In the simplest form, the pattern of activation in the HU layers can be viewed as a redescription or transformation of the input. If there are fewer HUs than input units, the transformation acts in such a way as to compress the input representation. If there are more HUs, then it will be

expanded. Such shaping (i.e., compression or expansion) is due to the backpropagation of errors from the output layer (Rumelhart, Hinton & Williams, 1986). Backpropagation distributes error over the HUs, causing the patterns to separate to the limits of the space available. The opposing forces of separation and containment force the HU space to structure the patterns into an efficient representation system. Potential representation systems can be viewed as geometric structures in HU space.

For networks with only 2 HUs, the structure of HU space can be seen directly by plotting the first HU on the X-axis, and the 2nd HU on the Y-axis. For networks with more HUs, there is substantial regularity in the structures formed in the hidden layer, although dimension reduction techniques, such as principal components analysis or canonical discriminant analysis (Kotz, 1982), may be required to reveal such structure (see Elman, 1989; or Bloesch & Wiles, 1991, for a review of the application of such techniques to HU space analysis).

Discrete regions: The simplest structures are formed by sets of patterns that cluster together, each cluster falling within a distinct region (all regions being disjoint). In a feedforward network, these clusters represent equivalent classes of inputs. In a recurrent network each region represents a logical state (Giles et al., 1990) and a change in the activation pattern on the hidden layer represents a transition from one logical state to another. The HU space and allowable transitions can be viewed as an embedded finite state machine.

Hierarchies: More complex structures can be formed by grouping regions of similar patterns together to form larger regions, effectively representing hierarchies of patterns. Using cluster analysis on a network trained on simple sentences, Elman (1989) showed that patterns representing specific instances of a word, such as "dragon", would have very similar HU patterns (i.e., cluster in a small region of HU space). Neighbouring regions represented clusters of related words such as "monster" or "lion". These patterns were part of an hierarchy of animals, animates (animals plus humans), and nouns. The major division in the space was between nouns and verbs. An interesting aspect of this hierarchy was that the HU representation was purely spatial (i.e., the relationship was defined by the topographic relationships in the HU patterns) even though no such information existed in the spatial input and output patterns. Rather, the information used to construct the spatial code was derived from the temporal relationships existing in the pairs of input and output patterns from sentences on which the network was trained.

Intersecting regions: Cluster analysis leads to the view of HU space as an hierarchy of regions. It is also possible to view the HU space as a set of overlapping regions, with each pattern being in the intersection of several intersecting regions. In a visual encoding task, Wiles & Ollila (1992) trained a network on simple scenes composed of pairs of coloured shapes. Using canonical discriminant analysis to group the patterns by colour, they showed that scenes were grouped into regions of the same colour, and that a scene with a red square and a blue triangle would be represented by a pattern that lay at the intersection of regions of red and blue shapes. Similarly, by grouping the patterns by shape alone, the same scene would be represented by a pattern at the intersection of scenes for squares and triangles. The limit to the number of independently intersecting regions depends on the number of HUs (i.e., the dimensionality of the HU space). In short, the network developed a compositional representation for features of the scene.

Continuous representations: Spatial representations are not restricted to the familiar structures of symbolic processing. For example, in compression tasks, such as the N-2-N encoder (N input, 2 hidden and N output units; see Figure 1), the 2 HUs need to take on intermediate values to represent the range of input patterns. Theoretically, with sufficient precision, weights exist for any such N (Kruglyak, 1990). For example, for an 8-2-8 encoder, patterns in the HU distribute themselves around an octagon, with each output unit active for one pattern on the octagon (Lister, 1992). These simulations show that HU space can exploit the continuous nature of the HU activation values to represent regular structures. This continuous nature has also been used to encode continuous distributions representing regular structures such as spatial scales, and irregular ones such as fractals (Pollack, 1989).

HU representations of combinatorial domains

In a combinatorial domain, such as a visual scene, a parsimonious representation of the scene might be described in terms of component

features such as colour, shape and size. The features can be viewed as variables, which take on a range of values (e.g., colours can be red, green, blue, etc). In previous work we showed that such combinatorial structure in simple scenes (3 colours, 3 shapes and 2 locations) can be represented by a multi-layer network in terms of intersecting regions in HU space (Wiles & Ollila, 1992). The current study addresses the question of how such representations change as the numbers of values for each variable increases. The task we used to study multiple values was modified from the earlier simulations to take into account recent results on representational issues in the output patterns in encoder networks, as described below.

The n–2–n encoder task as a variable/ value representation. Multi-layer feed-forward networks in which the input and output patterns are identical are called encoder tasks. In the standard encoder task, only local patterns are present in the data (i.e., in each input pattern, only one input and the corresponding output unit is non-zero, see Figure 1). In this standard formulation, all input patterns are orthogonal, and there is no structure inherent in the domain that a network can exploit in representing such data. Each input pattern in the data set is represented by a unique dimension. In an encoder task, there are fewer hidden units than input units, hence the network must find a compressed representation for the input patterns. Only 2 HUs are needed to represent an arbitrary number of input patterns which are locally coded (Kruglyak, 1990). The aspect of the n–2–n task that is of present interest is the fact that, in the input representations, only one unit is 1 for any pattern. Consequently, the set of patterns can be considered as alternative values of a single variable. The pattern that is active in the network (either at the input, or hidden layers) represents the value of the variable at that time.

In the standard form of the encoder task, learning times (in weight updates) appear to scale exponentially with the number of inputs, making it a difficult search task for backpropagation to find appropriate weights, even though theoretically they must exist (Lister, 1992). Due to such computational constraints on the learning time it is not a feasible task on which to base multiple variable/ multiple value studies.

Block codes in encoder tasks. Alternative representations on the input and output units effectively change the relationships between the data patterns and as a result affect learning in the encoder task. Preliminary experiments with representations for the output patterns (Bakker, Phillips & Wiles, in preparation) gave the surprising result that if the output representations

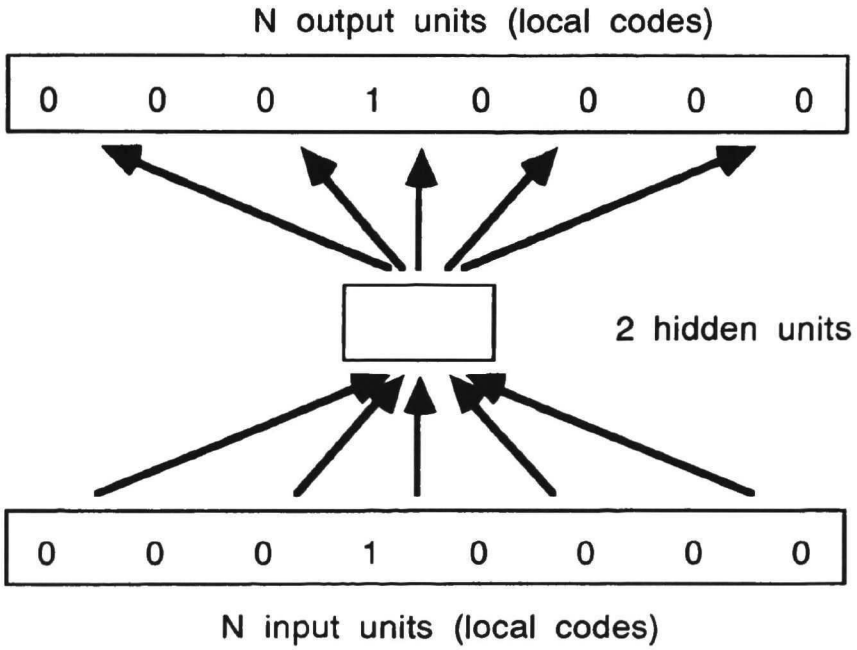

Figure 1. *n–2–n* encoder with local input and output codes.

are transformed from a local code (1 one, and n-1 zeros) to a block code (n/2 consecutive ones, and n/2 consecutive zeros, using wraparound), then encoder tasks are easy to learn (approximately linear in the number of inputs). These output codes reflect the structure that is required in the hidden unit representations, but also modify the position of the hyperplanes specified by the weights into each output unit. (Note that the modification to block output codes does not negate Kruglyak's proof of the existence of weights for arbitrary n.) The dramatic improvements in learning times in the block output codes provided the basis for simulations of multiple variables and values.

The multi-variable encoder task (nk–2k–nk encoder). The task used to test the extension of the combinatorial task to multiple values was based on the n–2–n encoder task, changing the architecture to nk input units, representing k variables, each taking on n values. Within each variable, the input values were coded using local codes, whereas the output values using block codes (see Figure 2).

The data for the k-variable encoder differ from the n–2–n encoder in their structure and number. The k-variable encoder represents a combinatorial domain, in which each variable is mapped from the input units to its corresponding output units, but is independent of the other variables. Hence, there are n^k patterns in the data set. Based on Kruglyak's proof of the existence of weights for the n–2–n encoder task, there must exist weights for the k-variable encoder task. However, it is not at all obvious that backpropagation could find such a set of weights. If it can, then it is essentially performing a decomposition of the nk input vector into the k variables that underlie the structure in the data set.

Method

We simulated the k-variable encoder task, using local input and block output codes, for n = 8, 9 & 14, and k = 2 & 3, and performed a detailed analysis of a simulation with n = 14 and k = 3 (for example, representing a scene which could contain objects in any one of 14 colours, 14 shapes and 14 sizes). The training algorithm was backpropagation with momentum of 0.9, updating the weights after every pattern presentation, initially using a tolerance margin on the target values of 0.2, a learning rate of 0.05 and a success criterion within 0.4 of the target values. After the network converged, a second stage of training followed in which the tolerance

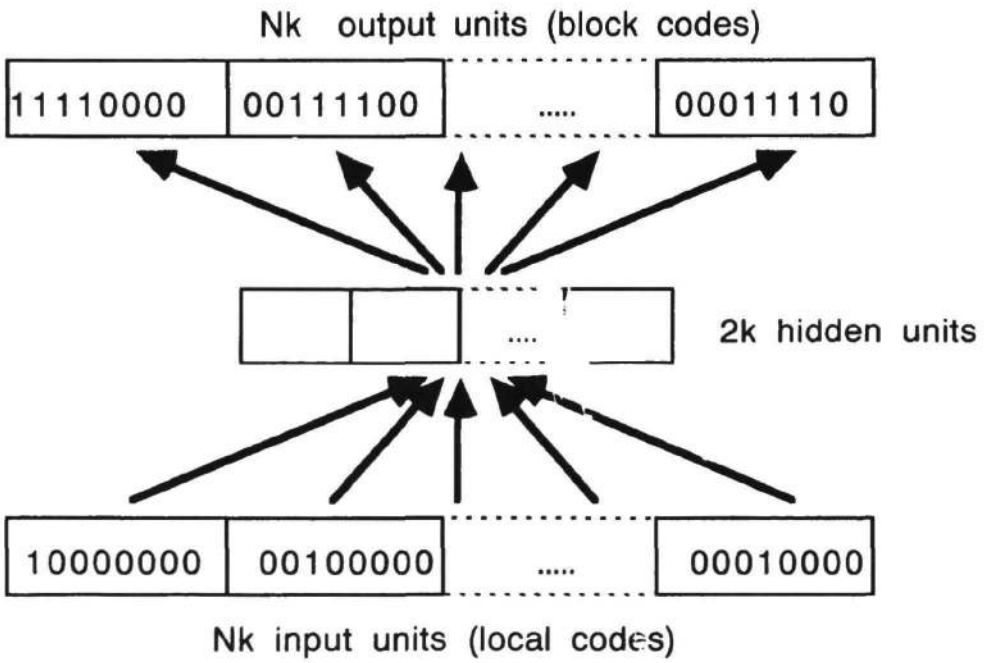

Figure 2. *nk–2k–nk* encoder with local input and block output codes.

margin, learning rate and success criterion were halved. Again, after convergence, a third stage of training proceeded with the same parameters halved again.

Results

All the networks reached the first success criterion in less than 500 passes through the training set (epochs), most in less than 100 epochs. Typical results for a n=8, k=2 (i.e., 16–4–16) encoder were 48 epochs to the first success criterion, 124 epochs to the second criterion. Typical results for an n=14, k=3 (i.e., 42–6–42) encoder were 24 epochs to the first success criterion, 178 epochs to the second criterion. In both cases, the networks continued to improve their performance, but did not reach the third success criterion even after another 500 epochs of training. Detailed simulations are needed to document the trends observed here, however it appears that the training times to reach the first criterion are linear (or even possibly sub-linear) in the training set size (i.e., $O(n^k)$). (As an aside, exploratory simulations with local output codes did not converge within the 500 epoch limits for the larger networks, and do not appear to conform to the same trends.)

After the three stages of training (the third stage proceeded for 500 epochs), one 42–6–42 network was analysed with respect to the weights formed, and the corresponding hidden unit patterns for each of the 3 input variables. After training, each variable came to be represented by a pair of hidden units: effectively the network decomposed the 42-6-42 encoder task into 3 separate 14-2-14 encoders, variable 1 represented by HUs 2 and 4, variable 2 by HUs 3 and 5, and variable 3 by HUs 1 and 6. The weights to and from each of these HUs support this analysis.

Discussion

The present study sheds light on several capabilities of NN not widely known. The first is the application of Kruglyak's result on the n–2–n encoder to the concept of variables: that is, that n independent values can be represented using just 2 hidden units, for arbitrary size n. This result is specific to sigmoid units (though it has a corollary for other types of activation functions), and is a consequence of the output units as hyperplane decision boundaries. The block codes used in the n–2–n task embody a similarity structure that reflects the spatial relations required for representation of independent values in the 2D HU space. Alternative structured codes would also be possible requiring 3D or higher dimensional structures.

The second capability concerns the improvement in learning times for block codes over local ones. It seems surprising that this result generalised to multi-variable tasks, or even that backpropagation could find the decomposition of the input vector into k coherent variables. Detailed studies are in progress investigating the learning times for combinatorial domains comparing local and block codes.

The third capability is due to Brousse and Smolensky (1989) in which they showed that learning in combinatorial tasks provides good generalisation. Further studies are proceeding in order to investigate whether the k–variable encoder nets exhibit the same generalisation phenomena reported by Brousse and Smolensky.

The original motivation for these studies was to investigate how the representations in combinatorial tasks (Wiles and Ollila, 1992) would change as the number of values for each variables increased. In the earlier studies we pointed out that tetrahedral structures are the most generally accessible structures for sigmoid (or hyperplane) output units (the dimension of a tetrahedron is the VC dimension of the space, using sigmoid output units). From the current simulations, it is clear that in the colour/shape/size combinatorial space, any combination of values must be accessible, but alternate groupings of values within a variable need not be. The implications of this understanding is that a value in one variable (e.g., red) would be accessible in combination with any single value in other variables, such as a triangle, square, circle, pentagon, etc. However, there need be no single partition of combinations of values within a variable, e.g., there may be no way of selecting red and green as a single group, or square and circle as a group.

Theoretical extensions

The simulations in the previous section can be generalized to arbitrary numbers of variables and values: Since two HUs are, in principle, sufficient to encode any number of values for each variable (weights exist for arbitrary n in the n-2-n encoder task, Kruglyak, 1991), then k variables can be represented by 2k hidden units.

This provides a new mechanism for thinking about the representational capacity of hidden units. In traditional information theoretic

approaches to calculating the capacity of a representational system, a fixed n-ary logic is assumed (e.g., binary, or 3-valued units). Based on such an assumption, increasing the number of values which a variable can take on increases the minimum number of units required to represent such a variable. For example, 8 values can be represented in 3 binary units. A 9th value could not – even in principle – be represented in 3 binary units. By contrast, in the simulations presented here, a single variable with an arbitrary number of values can be represented in a constant number of HU (i.e., 2 HUs). This is an example of a functional, rather than concatenative representation system, in which the tradeoffs in representational resources lie in the precision of the representing space, rather than number of bits.

Acknowledgements

This research was supported by an NSRG from the University of Queensland. I thank Anthony Bloesch, Jeff Elman, Mark Ollila and the NN research group in CS and Psych at UQ for many discussions on structures in HU space, Paul Smolensky for discussions on the combinatorial encoder tasks, and Kate Stevens for comments on the paper.

References

Bakker, P. Phillips, S. & Wiles, manuscript in preparation.

Bloesch, A. and Wiles J. (1991). Data representation and display techniques for representations in hidden unit space, First Indiana Conference on Dynamics in Cognition: Dynamic representation in cognition, Nov, 1991, Indiana University, Bloomington, Indiana.

Brousse, O., and Smolensky, P. (1989). Virtual Memories and massive generalization in connectionist combinatorial learning. *Proceedings of the 11th Annual Conference of the Cognitive Science Society*, Lawrence Erlbaum, NJ. 380-387.

Elman, J.L. (1989). Representation and structure in connectionist models. UCSD CRL Technical Report 8903, August 1989.

Fodor, J. and Pylyshyn, Z. (1988). Connectionism and Cognitive architecture: A critical analysis, *Cognition* **28**, 3-71.

Giles, C.L., Sun, G.Z., Chen, H.H., Lee, Y.C., and Chen, D. (1990). Higher order recurrent networks and grammatical inference. In D.S.Touretzky (Ed.) *Advances in Neural Information Processing Systems 2*, Morgan Kaufmann, San Mateo.

Hinton, G.E. (1990). Mapping Part-whole hierarchies into connectionist networks. *Artificial Intelligence* **46**, 47-75.

Kotz, S., and Johnson, N.L. (1982). *Encyclopedia of Statistical Sciences*. John Wiley and Sons, NY.

Kruglyak, L. (1990). How to solve the N bit encoder problem with just two hidden units. *Neural Computation*, 2(4), 399-401.

Lister, R. (1992). Backpropagation and the N-2-N encoder problem. In *Proceedings of the Third Australian Conference on Neural Networks*, Sydney, Australia, 198-201.

Pollack, J.B. (1989). Implications of recursive distributed representations. In D. Touretzky (Ed.) *Advances in Neural Information Processing Systems*, Morgan Kaufmann, San Mateo.

Rumelhart, D.E., Hinton, G.E., and Williams, R (1986). Learning internal representations through error propagation. In D.E. Rumelhart, J.L.McClelland and the PDP Research Group (Eds.) *Parallel Distributed Processing: Experiments in the Microstructure of Cognition 1: Foundations*. MIT Press, Cambridge.

Smolensky, P. (1988). On the proper treatment of connectionism. *Behavioural and Brain Sciences*, **11**, 1-59.

van Gelder, T. (1990). Compositionality: A connectionist variation on a classical theme. *Cognitive Science* **14**, 355-384.

Wiles, J., and Ollila, M. (1992). Intersecting regions: the key to combinatorial structure in hidden unit space. *Advances in Neural Information Processing Systems 5*, Morgan Kaufmann, San Mateo.