Introducing learning into tensor networks

Steven Phillips

(stevep@etl.go.jp)
Cognitive Science Section, Electrotechnical Laboratory, Tsukuba, Japan

Abstract: *In this paper it is shown how component, role and cue representations are learnt by backpropagating error through units and connections that construct tensor representations.*

Introduction

Connectionist models of human cognition have been strongly criticized for their inability to demonstrate structure-based behaviour without implementing symbol systems [1]. Tensor networks were developed as a way of representing structured objects within a connectionist framework by units that compute the outer product of object-role pairs [4]. However, tensor representations assume, ideally, orthonormal role vectors. In this paper, I present an architecture called the *tensor-recurrent network* (TRN) that learns these representational components.

A connectionist implementation of the outer product of two vectors $\vec{V}$ (object) and $\vec{W}$ (role) is a matrix of tensor unit activations such that $T_{ij} = V_i.W_j$, where T_{ij} is the ith-row jth-column unit of a rank-2 tensor $\vec{T}$; and V_i and W_j are the ith and jth units of vectors $\vec{V}$ and $\vec{W}$, respectively. A tensor representation is the sum of the outer product of object-role pairs. For example, the structured object *John Loves Mary* has the tensor representation $\vec{T}_{JLM} = \vec{V}_J \otimes \vec{R}_s + \vec{V}_L \otimes \vec{R}_v + \vec{V}_M \otimes \vec{R}_o$, where $\vec{V}_J$, $\vec{V}_L$ and $\vec{V}_M$ represent objects *John*, *Loves* and *Mary*, respectively; and $\vec{R}_s$, $\vec{R}_v$ and $\vec{R}_o$ represent the subject, verb and object roles, respectively. In terms of connectivity, each tensor unit T_{ij} has incoming connections from units V_i and W_j of strength 1.

The inner product of a tensor $\vec{T}$ and a vector $\vec{W}$ results in vector $\vec{V}$ such that $V_i = \sum_j^N T_{ij}.W_j$, implemented as weight 1 connections from their respective units. For example, given cue vector $\vec{Q}_s$ (requesting the subject component), $\vec{T}_{JLM} \odot \vec{Q}_s = \vec{V}_J$, if $\vec{Q}_s = \vec{R}_s$, and role vectors are mutually orthonormal.

The tensor-recurrent network (TRN)

The TRN learns appropriate object, role and cue vectors by backpropagating an error signal through units that compute the outer and inner products. Using *John loves Mary Subject?* as an example, a trace of network activation and error propagation is given for the TRN, depicted in Figure 1.

Each component object is presented to the network at the *input* units (one per time step). This results in activation vectors at the *hidden* units (internal component object representation) and *roles* units (object's role representation). Role vectors are also influenced by previous roles vectors represented at the *context* units. The outer product of object and role representations is computed and added to the *tensor* units. The inner product of tensor and role representations is computed at the *inner product* units. The resulting vector is mapped to the *output* units and compared to the target output. During the first three time steps the network is required to auto-associate the input, which helps learn appropriate internal representations. At the fourth time step, a query vector is presented at the *question* units, which results in a cue vector at the *cue* units. The inner product of the tensor and cue representations is computed at the *inner product* units, which is then mapped to the *output* units for comparison against the target.

At each time step, error is backpropagated from the *output* units to *question*, *input* and *context* units. Modifiable weights (dashed lines) are updated so as to minimize the sum of squares difference between output and target vectors. Units with modifiable weights have *tanh* as their activation function.

Component representations In the case where the role vectors are mutually orthonormal, then, by the laws: $(\vec{v} \otimes \vec{w}) \odot \vec{w} = \vec{v}$, if $\|\vec{w}\| = 1$, and $(\vec{v} \otimes \vec{w}) \odot \vec{z} = \vec{0}$, if $\vec{w} \perp \vec{z}$, the vector representation at the *hidden* units will be the same as the vector representation at the *inner product* units. Consequently, the network will act like a three-layer feedforward network, permitting, in principle, the learning of internal object representations for arbitrary input-output mappings.

Role representations Suppose the network has already been given the first component $\vec{V}_J$, and is currently required to auto-associate the second component $\vec{V}_L$ (i.e., $\vec{V}_L$ is the current target). Then, the error at the output layer is:

$$E = \|\vec{V}_L - g((f(\vec{V}_J) \otimes \vec{R}_s + f(\vec{V}_L) \otimes \vec{R}_v) \odot \vec{R}_v)\|$$
$$E = \|\vec{V}_L - g(\alpha f(\vec{V}_J) + f(\vec{V}_L))\|,$$

where f is the function from *input* to *hidden* units; g is the function from *inner product* to *output* units; $\vec{V}_J$ and $\vec{V}_L$ are the input/output representations of the *John* and *Mary* components, respectively; $\vec{R}_s$ and $\vec{R}_v$ are their respective role vectors; and, α is the dot product of $\vec{R}_s$ and $\vec{R}_v$. Error E goes to zero ($E \rightarrow 0$) when $\alpha \rightarrow 0$. That is, when the two role vectors are orthogonal, assuming $g(f(\vec{V}_L)) = \vec{V}_L$ (i.e., the network has learnt to auto-associate the *Loves* component).

Cue representations Suppose we request the subject component from the object *John Loves Mary*. Then, at the fourth time step, error is:

$$E = \|\vec{V}_J - g(\vec{T}_{JLM} \odot \vec{Q}_s)\|$$

$$E = \|\vec{V}_J - g(\alpha f(\vec{V}_J) + \beta f(\vec{V}_L) + \gamma f(\vec{V}_M))\|,$$

where $\vec{Q}_s$ is the cue vector at the *cue* units; and, α, β and γ are the dot products of $\vec{Q}_s$ with $\vec{R}_s$, $\vec{R}_v$ and $\vec{R}_o$, respectively. Error E goes to zero ($E \rightarrow 0$) when $\alpha \rightarrow 1$, and $\beta, \gamma \rightarrow 0$ (i.e., when the question vector is collinear with the first role vector, assuming $g(f(\vec{V}_J)) = \vec{V}_J$).

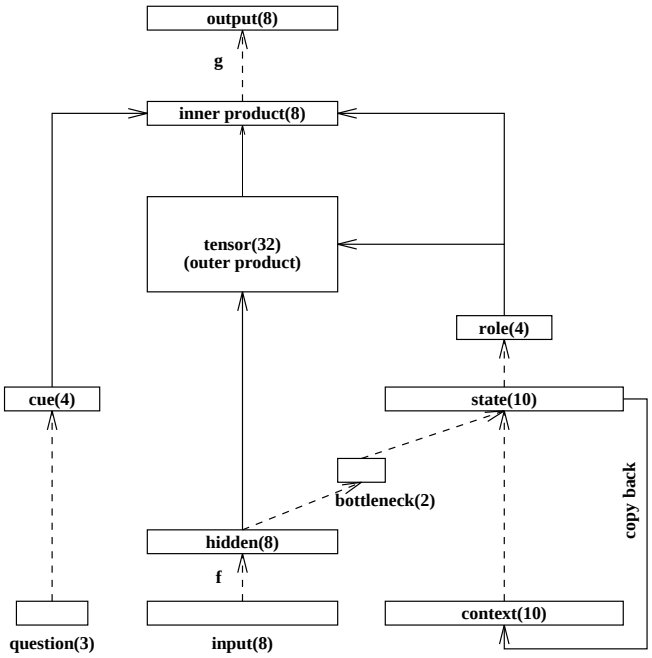

Figure1: Tensor-recurrent network architecture.

Simulation

Simple sentences conforming to the structure *subject-verb-object* are presented to the network, followed by a question requesting the subject, verb or object component. Subject and object components are drawn from the set { Bill, John, Karen, Mark, Vivian }, and the verb component is drawn from the set { calls, chases, loves }. All components are encoded locally (e.g., 01000).

Each trial, starting from a randomly initialized set of weights, consisted of training on 20 randomly generating sentence/question sequences, and testing on a separate set of 100 randomly generated sequences. Each word or question was presented one per time step with the network activations being reset to zero at the beginning of each sentence. The network was trained using standard backpropagation with the sum of squares error function and a learning rate of 0.1, until the activation of every output units was within 0.4 of the target output for all patterns in all sequences. The network was tested for correct extraction of components (all units within 0.5 of their target) on the remaining sentences. On each trial the number of correct responses was recorded. Full details of network simulations and their application to the systematicity problem appear in [2, 3].

The percentage of correct responses to the question vector on the test set averaged over 5 trials is given in Figure 2, where the x-axis indicates the number of nouns that appeared in both subject and object positions in the training set. The bottom dotted line indicates chance level response, and the error bars indicate 95% confidence intervals. The graph shows results only for noun-position combinations that did not occur in the training set.

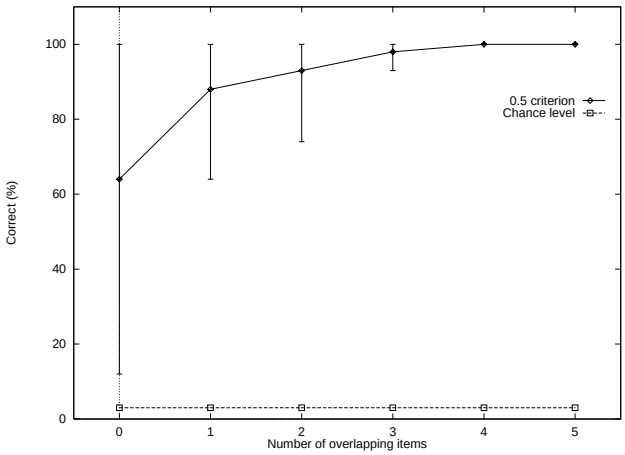

Figure2: Percentage correct versus overlap.

Discussion

Table 1 shows how with learning the role vectors become progressively more orthogonal. (The measure is one minus the magnitude of the normalized dot product so 0 implies collinear and 1 implies orthogonal.) By 1000 epochs of training the *verb* and *object* role vectors were collinear. However, by 4900 epochs of training these roles were sufficiently close to orthogonal permitting correct extraction of components.

Table1: Orthogonality between role vectors.

Update	S-V	S-O	V-O	aver.
0	0.32	0.22	0.02	0.19
1000	0.89	0.96	0.00	0.61
4900	0.70	0.59	0.71	0.67

Acknowledgments

I would like to Kazuhisa Niki for his help. The author is supported by an STA fellowship.

References

[1] J A Fodor and Z W Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28:3–71, 1988.

[2] S Phillips. Strong systematicity within a connectionist framework: Tensor-recurrent network. In *16th Conf. of Cog. Sci. Soc.*, pages 723–727, 1994.

[3] S Phillips. *Connectionism and the Problem of Systematicity.* PhD thesis, Univ. of Queensland, Australia, 1995. ftp://uqcsftp.cs.uq.oz.au/pub/pdp/theses/phillips.ps.Z.

[4] P Smolensky. Tensor product variable binding. *Artificial Intelligence*, 46:159–216, 1990.