Backpropagation in a non-linear layered network: learning from past mistakes

Joshua Goh

Psych 591 Neural Network Modeling

Project #2

Spring 2007

Introduction

Human beings learn from past errors, adjusting behavior in response to feedback. This paper investigates backpropagation networks (Rumelhart, Hinton, & Williams, 1986) as models of how human feedback learning occurs in the brain. Two general questions are asked. A) What can networks learn using error feedback? B) What are the operating parameters, resource requirements and limitations of this learning method?

Relevant to A, humans can learn item categorization using response feedback (observation 1; Murphy, 2002). Of note, categorization is poorer when item features overlap more. Humans also learn generalized response rules to new stimuli based on past experiences (observation 2), which also relates to the preservation of learned information (observation 3). Relevant to B, the brain implements these behaviors using neurons that receive input from external stimuli, intermediate neurons, and neurons that produce output for behavior (observation 4; Churchland, 1992). Thus, a large amount of information is represented within limited resources (data compression; observation 5; developed from 1-4).

This study implements a series of backpropagation networks that structurally model finding 4, and that use error between produced and desired outputs to modify subsequent responses (delta rule; Widrow & Hoff, 1960). The models' performances are compared to findings 1-3, and 5 to evaluate how well backpropagation networks demonstrate actual feedback learning in the human brain.

Model Operation

Neurons are simulated as units whose output activity, a_j, is determined according to the log activation function[1]:

[1] The use of the log activation function allows non-linear decision-making capabilities for

$$a_j = \frac{1}{1 + e^{-\sum(W_{ij} a_i + b_i)}}$$

Where a_i is a vector of input neuron activity, W_{ij} is a matrix of connection strengths between input and output neurons, and b_i is the input bias weight. Unit activation is thus a log function of the sum of weighted input activations that ranges from 0 to 1 with bias as the threshold for activation (Fig. 1a, see below).

The input, intermediate, and output neurons are simulated as a layer of input units each with feed-forward connections to every unit in a layer of hidden units above that are in turn connected to output units above them (Fig. 1b). There may be more than one or no hidden layers in the network. Bias is realized as a unit, in every layer except the output layer, which always has activation of 1 and connections to every unit above but none from units below. Error is backpropagated to the layers below.

Learning proceeds by propagating an input pattern through the network layers using the activation function and initially random connection weights (range –1 to 1). The error at time *t* between the output and the desired training output, $e_{i(t)}$, is computed as:

$$e_{i(t)} = (d_{i(t)} - a_{i(t)})(a_{i(t)})(1 - a_{i(t)})$$

Where $d_{i(t)}$ is the desired output and $a_{i(t)}$, is the actual output. $(a_{i(t)})(1 - a_{i(t)})$ is the differentiated log function. Error is larger when $a_{i(t)}$ is distant from $d_{i(t)}$ but the term is adjusted to avoid error corrections inducing a 0 or 1 activation state in the units (Fig. 1c). For hidden units, the error, $e_{j(t)}$, is computed as:

$$e_{j(t)} = \sum_j w_{ij(t)} e_{i(t)} (a_{i(t)})(1 - a_{i(t)})$$

Where $w_{ij(t)}$ is the weight between the hidden unit and the unit above and $e_{i(t)}$ is the error

---

each neuron. The differentiated function is used to adjust the error based on the current unit activation, preventing binary error values in units (1 or 0), which would result in non-settling oscillations in unit activation states (Fig. 1c).

of the unit above. Thus, the hidden units' and weights' contribution to output error is backpropagated. Weight change at time *t* of the connections between the current layer of units to the units below, $\Delta w_{ij(t)}$, is instantiated as:

$$\Delta w_{ij(t)} = \eta e_{i(t)} a_{j(t)} + \Delta w_{ij(t-1)} \mu$$

Where η is the growth rate, μ is the momentum constant applied to the previous weight change that prevents settling into local error minima (see below), and $a_{j(t)}$ is the activation of the unit below. The network has total network error, e_T, between the desired and actual outputs after training on *P* patterns, computed as:

$$e_T = \sum_P \sum_i (d_i - a_i)^2$$

On each iteration, weight changes are computed for each input pattern based on the same weight matrix. After all pattern weight changes are obtained, they are summed to the weight matrix and the total network error computed, which tends to a minimum value that is a function of the number of units and input patterns. The whole training process is iterated until total network error reaches a specified criterion threshold.

Simulations

Simulation parameters were as follows: η=0.3, μ=0.7, e_T =0.001, maximum number of training iterations=100000, number of runs per simulation=10.

Simulation 1. To test category learning (observation 1) and the effect of hidden units (observations 4-5), networks were created with 8 inputs, 4 outputs and a hidden layer (8-x-4) varying from 0 to 20 units (2^{20} bits of information capacity). The networks were trained on 4 categories of 8 non-overlapping activation patterns (8 bits; Table 1, Set 1). With no hidden units, the network took 18796.4 (SD=6.4) iterations to settle and correctly categorized the original training inputs to the 4 output types (e_T=0). The

network took longer to learn (44460.8 iterations) with 2 hidden units (2^2 bit), but was faster with increasing hidden units (Fig. 2a-b; see data compression below). Thus, hidden units improve performance only if they have enough capacity relative to the input, otherwise, a non-hidden layer network could perform better. The networks were tested with input patterns distorted (unit activations reversed) at 0.3 and 0.5 probabilities; total network error increased with increasing distortion such that at p=0.3 distortion, the networks output on average 1 wrong activation (Fig. 2b). Although the networks learned categories, they may not be robust to noise. The same networks were trained on linearly separable inputs (Table 1, Set 2) and non-linearly separable inputs (Table 1, Set 3). For Set 2, the effect of hidden units on settling iterations and error was similar to Set 1 except that having 3 hidden units improved error at p=0.3 distortion (Fig. 2a,c). This might be due to a 2^3 bit hidden layer storing extra biasing information rendering the network robust to noise. For Set 3, the network required hidden units to settle and was faster with increasing hidden units (Fig. 2a). As such, the first network could not categorize, but for the rest, the error pattern to distortion was similar to Set 1 (Fig. 2d). Thus, unlike for linearly separable inputs, non-linearly separable inputs require hidden units for categorization.

Simulation 2. To test data compression (observation 5), an 8-3-8 network was trained on 7 patterns (Table 2). This is similar to the first 4 networks in simulation 1 where input information has to be compressed in the reduced hidden layer. The network took 99553.9 iterations to settle with the hidden layer activations reflecting 7 unique bits of information in 3 units' joint activity (Table 2) that produced the correct outputs. However, the network could not generalize (observation 2) when tested on untrained pattern 8 (Table 2), having an average output error of 1 unit activation difference.

Simulation 3. To test memory (observation 3), the trained network from simulation 2 was further trained on two new patterns. The network took 93769.7 iterations to settle and made on average 0.5 unit errors (Table 3) to previous and currently trained patterns. It still could not generalize to untrained inputs (Table 3) as well as pattern 8 from simulation 2 (error=1.913). Thus the network could not hold past training well, although it made fewer errors to old patterns compared to untrained ones.

Discussion

The backpropagation networks learned categories of varying degrees of feature similarity given the right structure and represented a larger amount of information with limited resources. The networks were sensitive to pattern distortion, previous information can get moderately erased and they could not generalize rules from past learning.

The last two findings are significant disjunctions between the model and actual brain function. Humans get interference from new experiences, but old behaviors are quickly reinstated when necessary. In the model, reinstating old behaviors involves re-initializing network weights, or re-training previous mappings, taking a longer time. Humans learn generalizations and rules, unlike the model. Thus, another type of network might be more suitable for modeling this function.

Like the human response to error, the backpropagation model adjusted its response based on output error that was fed back to the lower layers. It provides a simple framework, albeit the above disjunctions, for understanding and predicting the limitations of human responses to real errors. Further development is required to ascertain how and where this error information is processed in the first place and what constitutes an error in the brain.

References

Churchland, P. S. (1992). *The Computational Brain.* Bradford Book.

Murphy, G. L. (2002). *The Big Book of Concepts.* Bradford Books.

Rumelhart, D., Hinton, G., & Williams, R. (1986). *Learning internal representations by error propagation.* MIT Press Cambridge, MA, USA.

Widrow, B., & Hoff, M. (1960). Adaptative switching capatibility and its relation to the mechanisms of association. *Kybernetik, 12*, 204-215.

Pattern	Training Inputs (Category Features)										Category	Desired Output			
							Set 1								
1	0	0	0	0	0	1	0	0	0	0	A	1	0	0	0
2	0	0	0	0	0	0	0	0	0	1	A	1	0	0	0
3	0	0	0	0	1	0	0	0	1	0	B	0	1	0	0
4	0	0	0	1	0	0	0	0	0	0	B	0	1	0	0
5	0	0	0	0	0	0	0	1	0	0	C	0	0	1	0
6	0	0	1	0	0	0	0	0	0	0	C	0	0	1	0
7	0	1	0	0	0	0	1	0	0	0	D	0	0	0	1
8	1	0	0	0	0	0	1	0	0	0	D	0	0	0	1
							Set 2								
1	1	1	0	1	0	0	0	0	0	1	A	1	0	0	0
2	1	1	0	0	1	0	0	0	0	1	A	1	0	0	0
3	1	1	0	0	0	1	0	0	1	0	B	0	1	0	0
4	1	1	0	0	0	0	0	0	1	0	B	0	1	0	0
5	1	0	1	1	0	0	0	1	0	0	C	0	0	1	0
6	1	0	1	0	1	0	0	1	0	0	C	0	0	1	0
7	1	0	1	0	0	1	1	0	0	0	D	0	0	0	1
8	1	0	1	0	0	0	1	0	0	0	D	0	0	0	1
							Set 3								
1	1	1	1	1	1	0	0	0	0	0	A	1	1	0	0
2	0	0	0	0	0	1	1	1	1	1	A	1	1	0	0
3	1	1	0	1	1	0	0	1	0	0	B	0	0	1	1
4	0	0	1	0	0	1	1	0	1	1	B	0	0	1	1

Table 1. Training inputs and outputs for simulation 1. Set 1: Non-overlapping inputs binned into 4 output categories. Set 2: Overlapping but linearly separable inputs binned into 4 output categories. Set 3: Non-linearly separable inputs binned into 2 categories.

Training																
Pattern	Input								Desired Output							
1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
5	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
6	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0
7	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0

Testing											
Pattern	Input								Hidden Unit		
1	1	0	0	0	0	0	0	0	1	0	0
2	0	1	0	0	0	0	0	0	1	0	1
3	0	0	1	0	0	0	0	0	0	0	1
4	0	0	0	1	0	0	0	0	0	1	1
5	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	1	0	0	1	1	0
7	0	0	0	0	0	0	1	0	0	1	0
8	0	0	0	0	0	0	0	1	0	0	0

Table 2. Simulation 2. Training: inputs and outputs for simulation 2. The 8-3-8 network was trained on patterns 1-7. Testing: The network was then tested on patterns 1-8. A sample hidden unit activation from 1 run of simulation test is shown. The hidden units coded 7 bits of unique information, but could not resolve pattern 8 from pattern 5.

Pattern	Input								Mean (SD) Total Network Error	
1	1	1	1	0	0	0	0	0	0.600	0.516
2	0	0	0	1	1	1	0	0	0.700	0.482
3*	0	0	1	1	0	0	0	0	0.939	0.935
4*	1	0	1	0	0	0	0	0	0.967	0.512
5*	1	1	1	1	1	1	0	0	3.108	0.322
6**	1	0	0	0	0	0	0	0	0.638	0.601
7**	0	0	1	0	0	0	0	0	0.525	0.519

Table 3. The network from simulation 2 was further trained on patterns 1-2 and tested on patterns 1-7. The total network error from testing is shown. *Patterns 3-5 are novel untrained patterns. **Patterns 6-7 are previously trained patterns in simulation 2. The network could not fully learn the new patterns, and lost some of the old patterns. However, the error for trained patterns was always lower than untrained patterns.

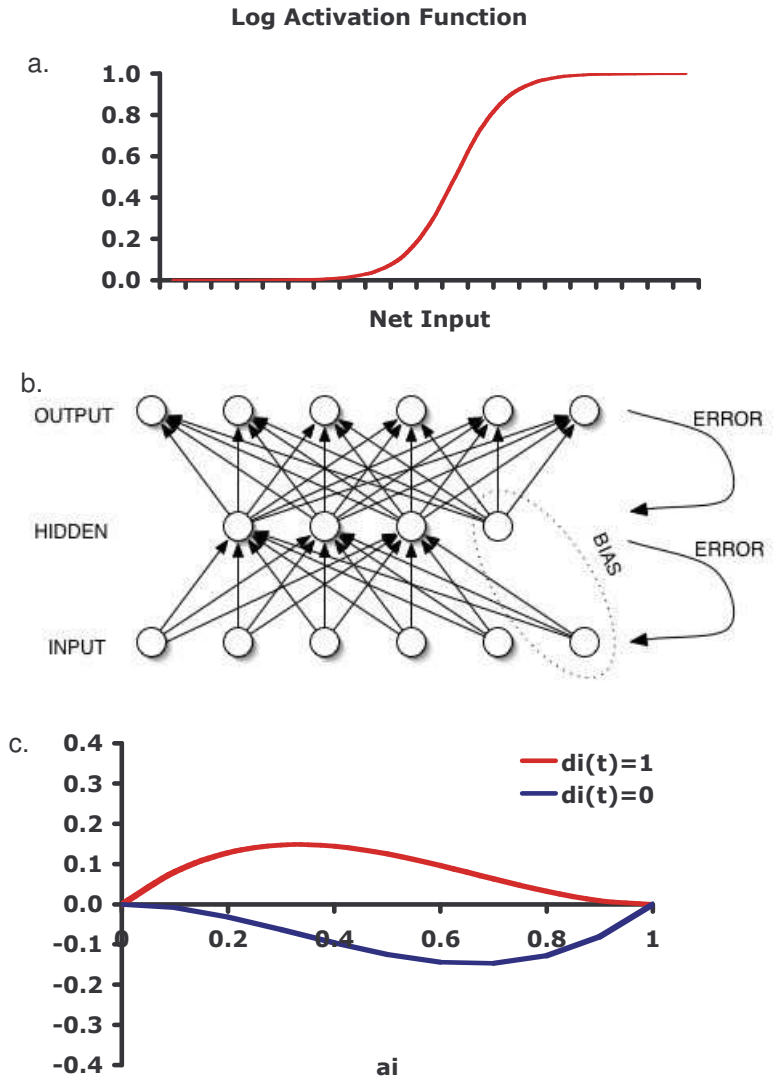

Fig. 1. a) Unit activation varies with the log function of the net input to each unit ($\Sigma W_{ij}a_i$) and the bias threshold, which determines the net input value to obtain a_j=0.5. b) The backpropagation network model. c) The effect of the differentiated log function term in error calculation. When a_i is small and d_i is 1, the error is large except when a_i approaches 0. This ensures that the backpropagated error does not induce binary oscillation states in unit activations. The converse applies to d_i=0.

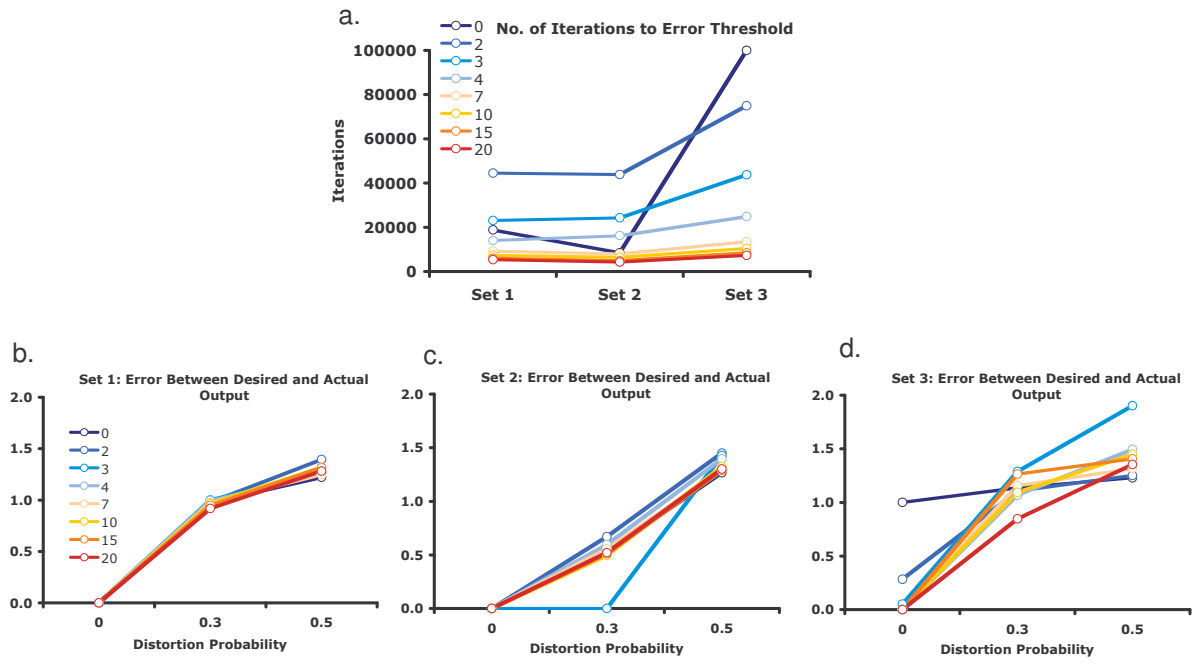

Fig. 2. Results for simulation 1. a) Number of training iterations required for networks to reach error threshold. Networks have increasing number of hidden units (key colors). b-c) Total network error for each network type for each level of test pattern distortion and for each training set of inputs.