Phonology 6 (1989) 305–316
Printed in Great Britain

*Feature encoding by neural nets**

Amanda Lathroum
Harvard University

While the use of categorical features seems to be the appropriate way to express sound patterns within languages, these features do not seem adequate to describe the sounds actually produced by speakers. Examination of the speech signal fails to reveal objective, discrete phonological segments. Similarly, segments are not directly observable in the flow of articulatory movements, and vary slightly according to an individual speaker's articulatory strategies. Because of the lack of a reliable relationship between segments and speech sounds, a plausible transition from feature representation to the actual acoustic signal has proven elusive. This paper utilises a theory of information processing, known as PARALLEL DISTRIBUTED PROCESSING (PDP) NETWORKS (also called neural networks), to propose a model which begins to express this transition: translating the feature bundles indicated in a broad phonetic transcription into continuous, potentially variable articulator behaviour. Following the suggestion in Ladefoged (1980), certain features are not specifically associated with certain articulator responses; rather entire feature bundles are compressed into natural classes which are associated with coordinated articulator efforts. This sort of holistic approach is encouraged by PDP theory because networks learn to associate whole patterns. Thus while there is no one-to-one feature–articulator correlation, the system can make certain generalisations about the shape of sounds encoded by sets of commonly repeated features. Furthermore, because the network allows parallel influence among learned inputs, the interplay of information in language processing, particularly the complex interaction of muscles in motor behaviour such as coarticulation, can begin to be modelled.

1 PDP: a brief overview

PDP models assume that information processing takes place through the interactions of a large number of simple processing elements called units. Instead of having one processor attack parts of a problem one after the other, information is processed in parallel by many units functioning at the same time. One benefit is that processing is likely to be faster. Another is that a network of simple processors can operate on global pieces of data even if each processor accesses only local information. Consider the

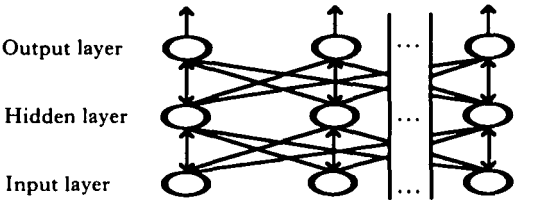

Figure 1

Each node is connected to all the other nodes on the processing level above it. The connections can maintain excitatory or inhibitory charges and may allow transmission in one or both directions. Additionally, units may be connected to themselves reflexively and/or other units on the same layer or non-adjacent layers.

human brain. A single neuron is incapable of determining what you want for dinner, but the entire brain (or significant portions of it) can retrieve information about various foods from memory and isolate a likely candidate. Units may stand for specific things such as features, or be used to refer to aspects of a situation or a particular hypothesis concerning interpretation. Usually the information is captured in patterns over a number of units, so that the system is said to rely on distributed representations.

Each unit contributes to processing through its connections to other units in the net. Sometimes each unit is connected to every other unit. More typically, units are arranged in highly interconnected layers so that each unit communicates with every other unit on the layer immediately preceding and following its own. Because these massive interconnections mimic the behaviour ascribed to synapses, and because each unit can be viewed as a neuron or group of neurons, much of the PDP research has been influenced by an understanding of the neurosciences. Indeed, part of the appeal of PDP models is their 'neurological' flavour; they are often used to model distinctly human behaviour. The architecture of a simple PDP system is pictured in Fig. 1.

In a PDP system, 'knowledge' is processed when the input nodes receive excitation that is sent through internal or 'hidden' layers to achieve an output pattern. The activation patterns among output units are controlled by the degree of excitatory and inhibitory influence of the connections between layers. When a strong excitatory connection exists between units, activating one is likely to encourage activation of the other. When a strong inhibitory connection exists between units, activating one is likely to discourage activation of the other. It is the strengths of the connections between units, not the units themselves, which store 'knowledge' or 'remember' the patterns which a network has been taught.

The task of 'learning' in a PDP framework is not the formulation of explicit rules as generativists might advocate, but the acquisition of connection strengths between a network of units which allow the network to associate patterns as though it were using an explicit rule. Because most

PDP models function as pattern associators (given a certain input, the network will produce a particular, corresponding output), the acquisition of connection strengths occurs in training sessions where the two patterns are simultaneously presented to the system and the connections between input and output are altered by specific algorithms called learning rules. The most simple learning rule, attributed to Hebb (1949), prescribes that the strength of the connection between unit A and unit B should be increased when they are simultaneously excited. Such a rule is intuitively appealing but is inherently limited by its simplicity. The model presented here relies upon a more complicated learning procedure which will be described in the next section.

Once a network has developed a set of connection strengths, it relies upon propagation, activation and output rules to govern its response to input. The function (F) determines the activation of a unit by associating a unit and the input it receives from its connections to other units. F may be an identity function, in which case the state of activation corresponds directly to the value of the input, or it may be a threshold function where the summed total of all a unit's inputs must exceed a certain value for the unit to be activated – in much the way that the summed total of electrical activity of dendritic connections between neurons must exceed a certain threshold to fire a charge breaching a synaptic gap. Once successfully activated, a unit's activation value is used to calculate its output. Simpler architectures allow the activation value of a unit to function as its output value. Additionally, the output function may trigger a binary on–off response or a continuous value constrained by maximum and minimum boundaries. Each output value is then propagated along the unit's connections to the next layer of units in the net. Propagation rules determine how excitation is sent through a system. The most common propagation rule multiplies the output of a unit by the strength of the connection between it and the next unit and adds this to the next unit's other inputs. At the next unit, a new activation value is calculated and the entire process repeated. More detailed discussions of the philosophy and general implementations of neural networks may be found in Feldman & Ballard (1982), McClelland & Rumelhart (1986) and Rumelhart & McClelland (1986a).

2 Applications of PDP: feature to articulator encoding

Neural network theory is most successful in its treatment of those processes which separate humans from existing computers: the ability to complete familiar and novel patterns, to use associative memory, to be impervious to small disturbances in the data or processing mechanisms, and to make generalisations. Although PDP represents a collection of valuable insights into pseudo-cognitive functioning, purely parallel models (such as shown in Fig. 1) are not sufficient to accommodate the

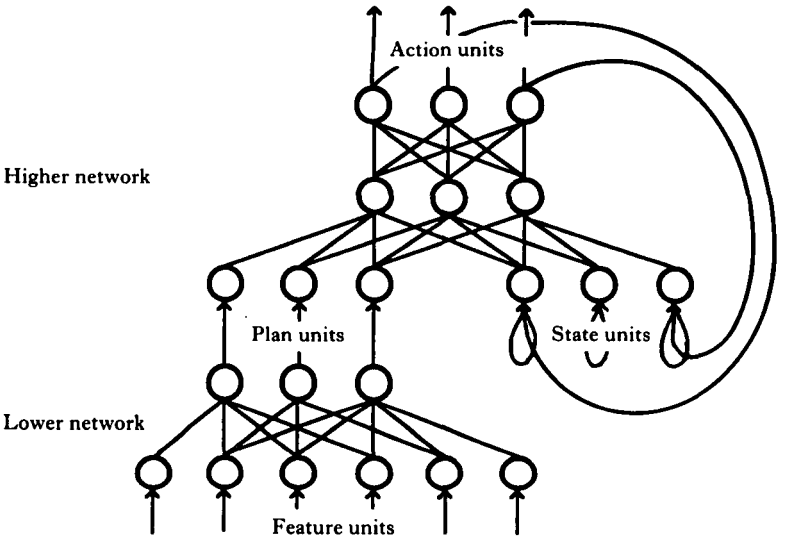

Figure 2

Basic, simplified network architecture. Reflexive connections and connections between action and state units are one-to-one; otherwise layers are fully interconnected (not all connections shown).

effects of sequential, temporally bound behaviour. Models which aim to mimic behaviour such as motion perception or the fluent speech production made possible by continuous articulator activation need to employ both sequential and parallel tools. Speech segments, represented in a sequence, exhibit a proclivity for mutual, bidirectional influence over articulators, partially captured by parallel interaction.

The following model builds upon Jordan (1986a), which describes a theoretical network synthesising sequential and parallel processing. The proposed system shown in Fig. 2 is composed of two hierarchically arranged networks. The higher network processes learned sequences of sounds to produce parallel articulator movement; the lower network compresses serial feature representations of sounds into patterns which serve as input to the higher net. It should be noted that this model is not intended to serve as a means of mimicking speech production. Realistically, it cannot: the influences and contributions of too many other factors, such as inertia, subglottal pressure and variables such as stress and rate or style of speech have not yet been considered. Instead this model attempts to present theoretical tools which might describe a plausible transition between an abstract categorical representation of information and its scalar manifestation.

The higher net consists of units encoding plan, action and state patterns. The network receives its input as a pattern of activation of the

plan and state units which triggers a series of actions, patterns indicated by the activation of the output nodes. The plan serves only to identify each series and remains constant during that time. State units vary with each action. As a result, the input values contributed by the state units control the sequential activation of the network: what action is produced at each time step.

In order to function effectively, the network must be able to form and recall an arbitrary association between action and input. Because the plan remains constant for each series, the most important input comes from the state units. In order to determine the action at each time step, state unit patterns need to be distinguishable at each step in the cycle. At the same time, sequences of states connected in time and invoking similar subsequences, like those found in coarticulation, should resemble one another. The similarity of states triggering similar actions could be expressed by defining states according to the action they produce. However, this would not allow states to contain sufficient information to identify each time step and disallow temporal ambiguity, as in the case of repeated sequences. One solution, captured by the next-state function, is to define each state as an exponentially weighted average of past outputs in addition to the action it produces. Each past state has some representation in the current state but with increasingly diminishing importance as time goes on.

The current state receives input from the previous state through reflexive connections from state units to themselves; input from the previous output comes through connections from output units to state units. Unlike other units in the net, state units use a linear activation function – the input to the state units at time n functions as its output value. The next-state function is expressed by the following equation, where $n-1$ refers to the time step prior to the present time step and o indicates the output value of the output unit connected to the state unit. The weights from output units to state units are constant at 1. Reflexive connections of the state units are also constant, being defined by (1):

$$(1) \quad s_n = \mu s_{n-1} + o_{n-1}$$

Temporally proximal states tend to be similar according to the similarity of the present and previous output and the value of μ. The larger the value of μ, the greater is the influence of the past states.

The network's net response, its action output, is a response to propagation of simple weighted sums from state and plan inputs to hidden units and from hidden to action units. The network learns output actions – not states – by using a back propagation learning rule. Teaching input supplies each output unit with a target answer. During training, input is propagated through the net to produce an output pattern. The value of each output unit is compared to its target, which triggers one of three actions. When the output units show the desired value, nothing happens. If the computed output is 0 but the target is 1, the learning rule increases

the probability that the unit will be on for the next trial by increasing the connection weights from active units by a small amount. If the situation is reversed – the computed output is 1 but the target is 0 – the weights are decreased. The change made to the weights between the output unit and the layers preceding it is roughly a function of the old weights and the error values limited by a constant learning rate. In this way, error is calculated for the action units and used to change the weights on the connection between the hidden layer and the output layer. Next, the error is calculated between the hidden layer and the input layer and used to change the weight on the connections between them. Error is propagated backwards through each layer of the system from output to input, hence the name for the learning rule. More details and justification both for back propagation and the accompanying activation rule (back propagation requires a non-linear activation/output function) can be found in Rumelhart *et al.* (1986).

Unlike traditional error-correcting schemes which anticipate a fixed value for each output unit, this system's rule has been modified to place constraints on the output which define its possible values. The constraints specify a particular value, a range of values or no value at all – a 'don't care' condition. If the activation value of an output unit falls within its set of constraints or is not subject to any particular constraint, no error corrections are created for that unit. If, on the other hand, output does not meet its constraint, an error value is calculated and used according to the back propagation algorithm to shift the system's weights toward a set of values in which the constraint will be met.

To apply this model to speech production, we can identify each element in the action pattern with a particular articulator (or more correctly as a trigger for the activity of a particular articulator), and assume that each plan represents a word and that each state creates the environment to invoke in order the particular articulator positions necessary to express the phonemes for a particular word. Drawing upon a word-based theory of the lexicon similar to Aronoff (1976), I am assuming that input to this net is taken from a lexicon of stored, learned representations for words. The information in these representations includes everything produced by the previous application of a language's phonological rules. All features which are needed to describe a segment must be indicated since the system is not capable of inventing new features. On the other hand, we shall see that the system is capable of changing the implementation of unconstrained articulators, producing effects which can be interpreted as coarticulation.

We have stated only that input constitutes a learned representation of a word without specifying the exact nature of this input. Evidence from Denker *et al.* (1987), however, suggests that a neural network of the architecture described would be unable to learn the desired response to unstructured input. Therefore, an additional network is proposed which 'precompiles' information. This net compresses a fully specified feature representation of a sound into a more generalised but still distinct pattern which serves as partial input to the higher net. Each phoneme in the word

plan is the result of one output from the lower network. The generalised pattern of this output conforms to a concept of natural class.

The lower network consists of two fully interconnected layers. Input units represent 15 feature values for the set of sounds in English (adapted from Kenstowicz & Kisseberth 1975); output units indicate eight natural classes roughly divided into place (front, palatal and back) and manner (voice, nasal, stop, fricative and sibilant). Each sound in English may be uniquely represented as a pattern over the output units so that information captured by the binary features is always retrievable by the system.

Simulation of the new net was performed as follows: weights between the two layers were initially set to 1. The system was trained on a set of patterns representing input and output pairs for half of the consonant set of English. Training began with presentation of an input pattern representing the feature values of the first sound in the set. Activation was propagated as a multiplied sum to the output layer, creating an output pattern. This output was then compared to a target value. The target was twice the threshold value if the unit was supposed to be activated; 0 otherwise. The error value was calculated by limiting the difference of the output and the target by a learning rate and then added to the strength of the connection between input and output units. This process is quite similar to back propagation but is simpler because of the lack of hidden units (see Widrow & Hoff 1960). In this way, the system 'learned' the first pattern, i.e. developed a set of connection strengths which propagated the input excitation in such a way that it produced the proper output pattern. Next, the second sound was trained on the net so that the network's newly created connection strengths were altered enough to 'remember' the first pattern but also to produce the second output pattern when presented with the second input pattern. This process was repeated until each member of the training set had been presented to the system once.

Through repeated presentations of the possible patterns, the network finally settled on weights which produced the correct output for each input, suggesting that when there is a shared structure to the input patterns, the network may be able to generalise relationships among patterns. Three complete training cycles were necessary for this system to 'learn' the first set of inputs. That the network had generalised relationships among unit activations was reinforced when the other half of the consonants were presented to the net. The correct unit response for untrained, but similar, patterns was 68 %. These new patterns were added to the original training set, and after two more complete training cycles, the net could successfully identify them all. A second net representing vowel sounds in English was created in the same fashion (though a C–V division is not theoretically necessary).

In this way, a limited set of patterns may be learned early in the speech acquisition process. Because there is no mutual influence in output, this net can process sounds as they are being joined together for the first time to form words. Sequence of input is never learned but varies randomly. The ability of the lower network to identify general patterns from feature

input addresses the problem of unstructured input to the higher net. The higher network, which learns an unlimited number of combinations of sounds, also learns to associate tendencies for articulator output with certain input patterns, that is certain classes of sounds. Thus while there is no one-to-one feature–articulator correlation, the overall system should be able to isolate generalisations about the shape of sounds encoded by sets of commonly repeated features.

Let's consider a specific example, the word *bean*. The first time the word is attempted, the lower net, which has already learned the set of associations described above, is operative. Feature values for the phoneme /b/ are presented to the input layer. The lower net propagates this excitation to produce an output pattern corresponding to the unique set of natural classes which describe the sound. In this case, the output pattern indicates a voiced stop. This pattern further acts as the first part of the plan input to the higher net. Next, the feature values for the phoneme /i/ are presented to the input layer of the lower net, producing a pattern for a high front unrounded vowel which forms the second part of the plan input to the higher net. Finally, the third sound is processed, producing a coronal voiced nasal stop as the third part of the plan input.

This plan, which triggers actions producing the phonemes in *bean*, is now active on the input units for the higher net. Because the word is novel, the network must be trained to make the proper associations and produce the proper output values over three action units for the string of phonemes, /bin/ (the number of units has been simplified). Suppose the action units each trigger the activity of a particular articulator. Unit 1 controls lip closure; unit 2 the blade of the tongue; unit 3 velum height. A rough schematic of articulator activity for the word *bean* might look something like this:

(2)

		[b]	[i]	[n]
/bin/	lip	closed/open	raised	slightly
	tongue	high		lower (as tip raises)
	velum	(raised)	lower	

The lips must be closed to form the plosive [b] and must remain open to accommodate the following sounds. The tongue blade, which is not crucial to the production of the stop, is unconstrained and partially assumes the position for producing front and high [i]. The tongue's contribution to the offglide of the stop creates coarticulation as the stop is exploded. The second phoneme is produced at the completion of the plosive and as shown in (2), is coloured by nasality. The velum, which is unconstrained in English vowel production, lowers during the [i] sound in anticipation of the following nasal, creating another instance of co-articulation. Finally, the tongue tip (not represented here) touches the alveolar ridge to form the final sound.

The teaching vectors for the output of the action nodes may look like the

following (numbers indicate the specific value for which that unit is constrained; asterisks represent a 'don't care' condition):

(3)

	[b]	[i]	[n]
unit 1	1	*	*
unit 2	*	1	*
unit 3	*	*	1

The initial value for unit 1 indicates the lips' fully opened position on release of the stop while the following values are unconstrained. Unit 2 is unconstrained until the second phoneme, when its action is specifically defined: high and front. Unit 3 is unconstrained until the third sound when its action, lowered velum, is specifically defined. At each step, defined as a new activation sequence on the state units, errors are sent back only from constrained output units. The pattern of activation over constrained units becomes associated with the current state. In this example, associations are learned from s1 (s = state) to activation of the first output unit, from s2 to activation of the second output unit, and from s3 to activation of the third output unit.

A by-product of this kind of learning is a parallel influence among actions. One action pattern primes the activation of action units for subsequent outputs in a sequence. After learning the sequence shown above, the network will activate the first unit when s1 is presented on the state units. It will, in addition, partially activate the second and third output units even though there are no learned associations between s1 and these units. The reason for this effect is due to the formulation of the next-state function: temporally proximal states resemble each other. Because of the next-state function, s1 and s2 are similar and therefore tend to trigger outputs that are similar. Because the second and third units in s1 are not constrained, being in a 'don't care' situation, they are free to generalise from the pattern triggered by s2. The pattern triggered by s2 is itself partially influenced by the pattern of output units in s3, providing an instance of anticipatory influence. A learned representation of the word *bean* would exhibit some parallelism and may look like the following:

(4)

	[b]	[i]	[n]
unit 1	1	0·8	0·4
unit 2	0·8	1	0·8
unit 3	0·4	0·8	1

Because output values translate to articulator behaviour, the decimal figures indicate points of coarticulation. The serial nature of the network accommodates the sequential production of sounds. The parallelised output describes the influence one phoneme has on the sounds which surround it. Because of the parallel influence action units exert upon one another, articulators that are constrained while learning would not be subject to perturbations from surrounding sounds. Unconstrained articulators would allow some influence from other sounds. Although the output constraints discussed in the example above were fixed values, a

more effective alternative is to encode constraints as a range of potential values. Another way to represent constraints which very effectively captures the synergistic relationship between some articulators is to require that the combined output of two units be within a certain range. For information on various simulations of the higher network see Jordan (1986b).

Notice that in this example, constraints played a very important role in accurately modelling coarticulatory behaviour. Typically, coarticulation is defined as the overlap of articulatory gestures, either through anticipating the next sound or perseverating the previous sound. It is not, however, a random process, and only those aspects of a sound that do not interfere with its identification may change as a result of coarticulatory influence. For example, in examining the English word *bean*, we expect the vowel to be nasalised due to the influence of the following nasal consonant. In a language such as Akan where nasality functions as a distinctive feature for vowels, we would expect this articulator to be constrained so that it could exhibit no or very limited coarticulatory influences in a similar CVC pattern (Huffman 1987). Jordan (1986a) suggests that assignment of articulator constraints functions independently of context (unlike actual articulator behaviour) and can be derived from a definition of distinctive feature values. Conversely, initial acquisition of what articulators are constrained in which environments could occur as the speaker (or system) learns to associate sound with articulator position. The speaker may be aided by universal principles which dictate defaults for articulator settings, all or some of which are incorporated by different languages (Keating 1985). The strategy of default setting provides a nice basis for explaining individual language variation of phonetic processes, and by extending the definition of default to include a range of potential values, we can also begin to explain variation among articulations of individual speakers noted by Lindau *et al.* (1972). Taken together, these approaches seem quite compatible with the suggestion of Jakobson (1941) that speakers first learn to make the sounds of their language and then isolate distinctive features. Additional experience is necessary to see if this type of model can successfully accommodate such a learning sequence.

Finally, despite the fact that the intended input to this network uses phonemes, just how much of the transition from a traditional phonological to phonetic interpretation this model can accommodate is an empirical question without answer at the moment. Problems regarding timing – whether simply the duration of segments or the results of phonetic 'rules' such as the lengthening of vowels before voiced consonants or the simplification of heavy consonant clusters – seem particularly difficult. However, by expressing phonetic processes such as coarticulation in terms of the articulatory system, we do capture an inherent relationship between production and phonetic concerns.

3 Conclusion

By adopting the theoretical framework of PDP networks, we can visualise a model which translates groups of feature bundles into articulator positions, sidestepping the problem of using rule-governed manipulation of discrete units to describe continuous acoustic information. This model exploits parallel activation of articulators to produce coarticulation among learned sequences of phonemes; the association of generalised sound patterns with constraints on articulator behaviour reinforces the notion of natural classes. Future refinements to this model may revolve around its ability to include other, hierarchically arranged networks. The question of syllable membership, for example, has not been addressed. At this point, the plan pattern represents each phoneme in a word. The plan, therefore, is of variable length, which is highly problematic because the degree of excitation entering the network will also vary: the longer the word, the more input units will be excited. Such extreme variation in excitation should be too great to allow the network to find an appropriate set of weights which can match input/output pairs. For this reason, I am assuming that a system with a more sophisticated vocabulary will require a syllable network feeding this one. An additional area of research involves the manipulation of features earlier in the speech production system. Each phonological rule in a language, for example, may be best expressed as another net whose output functions as the input to another net culminating as input to the model outlined here.

This model, rather than providing a definitive solution to problems concerning speech production or the interaction of phonetics and phonology, is meant to suggest a new field of work which will provide interested researchers with a different way of examining old questions. Each network in the model offers a different perspective. The higher network addresses the question of mutual influence, which may be instructive in examining phonological process such as assimilations. It also describes long distance relationships useful when examining processes such as vowel harmony. The lower network functions as a pattern associator without mutual influence, which could be instructive when considering systems of underspecification or phonological rules that describe a change or addition of features. A similar pattern associator paradigm was used by Rumelhart & McClelland (1986b) to model past tense formation of English verbs. Certainly, there is much more work to be done in refining neural network applications. The model presented here may be too simple to be of practical value. It is hoped, however, that it serves to point the way toward a potentially rich theory of language.

NOTES

* Many thanks to Jill Carrier for her constant encouragement and to Sam Bogoch and Phil Lesourd for their comments on drafts of this paper. Of course, I assume full responsibility for its final contents.

REFERENCES

Aronoff, M. (1976). *Word formation in generative grammar*. Cambridge, Mass.: MIT Press.

Denker, J., D. Schwartz, B. Wittner, S. Solla, R. Howard, L. Jackel & J. Hopfield (1987). Automatic learning, rule extraction and generalization. *Complex Systems* **1**. 877–922.

Feldman, J. A. & D. H. Ballard (1982). Connectionist models and their properties. *Cognitive Science* **6**. 205–254.

Hebb, D. O. (1949). *The organization of behavior*. New York: Wiley.

Huffman, M. K. (1987). Timing of contextual nasalization in two languages. *JASA* **82**. S115–S116.

Jakobson, R. (1941). *Kindersprache, Aphasie und allgemeine Lautgesetze*. Uppsala: Uppsala University Aarskrift.

Jordan, M. I. (1986a). Attractor dynamics and parallelism in a connectionist sequential machine. Proceedings of the 8th Annual Conference of the Cognitive Science Society.

Jordan, M. I. (1986b). Serial order: a parallel distributed processing approach. Technical Report 8604, Institute for Cognitive Science, UCSD.

Keating, P. A. (1985). Universal phonetics and the organization of grammars. In V. Fromkin (ed.) *Phonetic linguistics: essays in honor of Peter Ladefoged*. Orlando, Fl.: Academic Press. 115–132.

Kenstowicz, M. & C. Kisseberth (1975). *Generative phonology*. New York: Academic Press.

Ladefoged, P. (1980). What are linguistic sounds made of? *Lg* **56**. 485–502.

Lieberman, P. (1976). Phonetic features and physiology: a reappraisal. *JPh* **4**. 91–112.

Lindau, M., R. Jacobson & P. Ladefoged (1972). The feature advanced tongue root. *UCLA Working Papers in Phonetics* **22**. 76–94.

McClelland, J. L. & D. E. Rumelhart (eds.) (1986). *Parallel distributed processing: explorations in the microstructure of cognition*. Vol. 2: *Psychological and biological models*. Cambridge, Mass.: MIT Press.

Rumelhart, D. E., G. E. Hinton & R. J. Williams (1986). Learning internal representations by error propagation. In Rumelhart & McClelland (1986a). 318–362.

Rumelhart, D. E. & J. L. McClelland (eds.) (1986a). *Parallel distributed processing: explorations in the microstructure of cognition*. Vol. 1: *Foundations*. Cambridge, Mass.: MIT Press.

Rumelhart, D. E. & J. L. McClelland (1986b). On learning the past tense of English verbs. In McClelland & Rumelhart (1986). 216–271.

Widrow, B. & M. E. Hoff (1960). Adaptive switching circuits. *WESCON Convention Record*. Part 4. 96–104.