Computer Science Department Technical Report
Artificial Intelligence Laboratory
University of California
Los Angeles, CA 90024-1596

FORMING GLOBAL REPRESENTATIONS WITH EXTENDED BACKPROPAGATION

Risto Miikkulainen
Michael G. Dyer

June 1988
CSD-880039

Forming Global Representations with Extended Backpropagation

Risto Miikkulainen
Michael G. Dyer

April 1988

Technical Report UCLA-AI-88-7

FORMING GLOBAL REPRESENTATIONS WITH EXTENDED BACKPROPAGATION[1]

Risto Miikkulainen
Michael G. Dyer

Artificial Intelligence Laboratory
Computer Science Department
University of California, Los Angeles, CA 90024
risto@cs.ucla.edu, dyer@cs.ucla.edu

ABSTRACT

Representing concepts distributively provides several advantages over local representation, e.g. associations and generalizations directly arise from the representations and the system is robust against damage and noise. Forming distributed representations is problematic. One approach is to classify the concept beforehand along several microfeatures and to represent each microfeature locally. The question remains as how to determine the appropriate microfeatures. This paper presents an alternative to fixed microfeature encoding. Meaningful global representations are developed automatically while learning the processing task. When the backward error propagation is extended to the input layer the representations of the input items evolve to reflect the underlying relations relevant to the processing task. No microfeatures and no discrete categorization can be seen in the resulting representation, all aspects of a concept are distributed over the whole set of units as an activity profile. The representation is determined by all the contexts where the concept has been encountered, and consequently it is also a representation of all these contexts.

1. INTRODUCTION

Representation of concepts is a key issue in natural language processing with neural networks. In the distributed approach different concepts are represented as patterns of activity over the same set of units. Desirable properties are achieved: (1) it is possible to associate similar concepts and generalize properties by sharing the same activity subpatterns, and (2) the system is robust against noise and damage [R86 Ch.3]. On the other hand, the representations are hard to build and process. In many systems they are hand coded by classifying each concept along dimensions of microfeatures such as size, softness, mass, animateness etc. Each microfeature is assigned a processing unit (or a group of units) and the classification becomes a pattern of activity over the network [R86 Ch.19], [H81]. Obvious questions are what are the relevant microfeatures and how are they determined?

The only way to avoid ad hocness completely is to let *the activity patterns be formed by the network itself.* The patterns do not necessarily have to implement a classification of the concept along any identifyable features. In the most general case they are simply profiles of continuous activity values over a set of processing units. This paper presents one such mechanism, where the representations are developed automatically *while the system is learning the processing task.* We call this process FGREP (Forming Global Representations with Extended back Propagation).

2. DESCRIPTION OF THE FGREP PROCESS

The system is based on a basic three-layer backward error propagation network, with the dynamics of [R86 Ch.8 pp.327-329]. The first goal, learning the processing task, is achieved by adapting the connection weights according to the backward error propagation principle. The second goal, developing meaningful distributed representations for the data, is achieved by *extending the error signals to the input layer* and modifying the representations *as if they were weights on connections coming in to the input layer.*

[1]This research was supported in part by an ITA Foundation grant and the JTF program of the DoD, monitored by JPL. The first author was also supported in part by grants from the Academy of Finland, the Finnish Cultural Foundation, The Jenny and Antti Wihuri Foundation and The Thanks to Scandinavia Foundation. The simulations were carried out on equipment donated to UCLA by Hewlett Packard.

In a sense, the representation of a concept serves as input activation to the input layer. The activation of an input unit is identical to the corresponding component in the representation. In this analogy, the activation function of an input unit is the identity function and its derivative is one. The error signal can now be computed for each input unit as a special case of the general error signal equation [R86 Ch.8 Eq.14]:

$$\delta_{1i} = \sum_j \delta_{2j} w_{1ij}. \tag{1}$$

where δ_{xy} stands for the error signal for unit y in layer x, and w_{1ij} is the weight between unit i in the input layer and unit j in the first hidden layer.

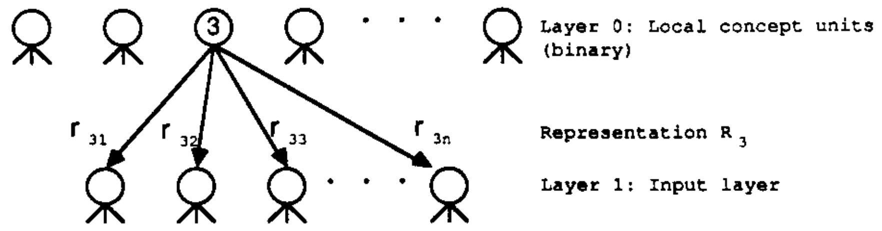

Figure 1: Interpretation of representations

Imagine a 0:th layer before the input layer, with one unit dedicated to each input concept (figure 1). In this layer exactly one unit is active with value 1 at any time (the one corresponding to the current input), the rest of the units have 0 activity. Each local unit is connected to all units in the input layer with weights equal to the input representations. *Extending back propagation weight change to these weights can be interpreted as changing the representations themselves:*

$$\Delta r_{ci} = \eta \delta_{1i} r_{ci}, \tag{2}$$

where r_{ci} is the representation component i of concept c, δ_{1i} is the error signal of the corresponding input layer unit and η is the learning rate. Using this analogy, *representation learning can be implemented as an extension of the back propagation algorithm.* There is a minor difference in the last step: while the weight values are unlimited, the representation values are limited between the maximum and minimum activation values of the units. The new activity value for the representation component i of concept c is obtained as

$$r_{ci}(t+1) = max\,[a_l,\ min\,[a_u,\ r_{ci}(t) + \eta \delta_{1i} r_{ci}(t)]], \tag{3}$$

where a_l is the lower limit and a_u is the upper limit for unit activation.

Instead of using the internal and local representation of concepts as input weights, we have chosen to separate the representations from the network and treat them as global, external objects. This way *the same representations can be used in different parts of the network,* both in the input and in the output. The input layer can be divided into assemblies so that *several concepts can be represented and modified simultaneously.* If the same concept occurs in several assemblies at the same time, the individual changes are simply summed together. It is evident that the representation values need to be continuous. Individual changes made in the process are small but converge in the course of time in a manner similar to the weights. The representation is developed by a process which tends to improve performance in the processing task. There is reason to believe that *the representation will effectively code properties of the input elements which are most crucial to the task.*

3. EXAMPLE TASK: ASSIGNING ROLES TO SENTENCE CONSTITUENTS

In [R86 Ch.19] McClelland and Kawamoto describe a system which learns to assign case roles to sentence constituents. The same task with the same data was used to test FGREP, mainly because it provides a convenient comparison to a system using fixed microfeature encoding. The architectures and goals of the two systems are compared in section 4.

Case role assignment means reading in the surface structure of the sentence and deciding on the semantic relations of the sentence constituents. For example, in `The man ate the pasta with cheese`, the sentence subject `man` is the agent of the `ate`-act, the object `pasta` is the patient and the with-clause `cheese` is a modifier of the patient. Role assigment is context dependent: e.g. in `The ball broke the window` the subject of the sentence is the instrument of the `broke` -act. In `The ball moved` the same subject is the patient of a different act. Assignment also depends on the semantic properties of the concept. In `The man ate the pasta with cheese` the with-clause modifies the patient but in `The man ate the pasta with a spoon` the with-clause is the instrument of eating. In yet other cases the assignment must remain ambiguous. In `The girl hit the boy with the ball` there is no way of telling whether `ball` is an instrument of `hit` or a modifier of `boy`.

The task was restricted to a small repertoire of sentences studied by McClelland & Kawamoto. The sentences consisted of a subject, verb, object and a with-clause. The possible case roles were agent, act, patient, instrument, and modifier-of-patient. The data generators are depicted in table 1 and the noun categories in table 2 (source: [R86 Ch.19]). The categorization is not visible to the system: it is only manifest in the combinations of nouns that occur in the input sentences. To do the case role assignment properly *the system had to figure out the underlying relations and code them into the representations.*

TABLE 1: SENTENCE GENERATORS

Gener	Sentence Frame	Correct case roles
1.	The human ate.	agent
2.	The human ate the food.	agent-patient
3.	The human ate the food with the food.	agent-patient-modif
4.	The human ate the food with the utensil.	agent-patient-instr
5.	The animal ate.	agent
6.	The predator ate the prey.	agent-patient
7.	The human broke the fragile-obj.	agent-patient
8.	The human broke the fragile-obj with breaker	agent-patient-instr
9.	The breaker broke the fragile-obj.	instr-patient
10.	The animal broke the fragile-obj.	agent-patient
11.	The fragile-obj broke.	patient
12.	The human hit the thing.	agent-patient
13.	The human hit the human with the possession.	agent-patient-modif
14.	The human hit the thing with a hitter.	agent-patient-instr
15.	The hitter hit the thing.	instr-patient
16.	The human moved.	agent-patient
17.	The human moved the object.	agent-patient
18.	The animal moved.	agent-patient
19.	The object moved.	patient

TABLE 2: NOUN CATEGORIES

Category	Nouns
human	man woman boy girl
animal	bat chicken dog sheep wolf lion
predator	wolf lion
prey	chicken sheep
food	chicken cheese pasta carrot
utensil	fork spoon
fragile-obj	plate window vase
hitter	bat ball hatchet hammer vase paperwt rock
breaker	bat ball hatchet hammer paperwt rock
possession	bat ball hatchet hammer vase dog doll
object	bat ball hatchet hammer paperwt rock vase plate window fork spoon pasta cheese chicken carrot desk doll curtain
thing	human animal object

3.1 System Architecture and Operation

System architecture is best explained in terms of figure 2, which shows the final stage of the simulation. The figure is a snapshot of a real-time display of the process running on an HP 9000/350 workstation.

The header line displays the current input sentence. The top half of the display shows the current representations of the concepts, i.e. the current state of the lexicon. The lower half shows the current state of the network. The input and output layers of the network are divided into assemblies, each of which can hold one concept representation at a time. An input sentence consists at maximum of four concepts: subject, verb, object and a with-clause. The actual input to the network is formed by loading the lexicon entry of each concept of the sentence into the input assembly corresponding to the syntactic role of the concept. Each unit in the assembly is set to the activity value determined by the corresponding component in the lexicon entry. The network is fully connected, i.e. there is a connection from each input unit to each hidden unit and from each hidden unit to each output unit. The weights on the connections are displayed as square matrices between the layers. The output layer is divided into five assemblies indicating the case roles agent, act, patient, instrument and modifier. After the network has successfully learned the task, each output assembly produces an activity pattern which is identical to the lexicon representation of the concept that fills that role. The correct role assignment is shown at the bottom row of the display. This pattern forms the teaching input to the network. The representation layer (above the input layer) contains the new, modified representations after processing the current input. The unit activity values (in the in-

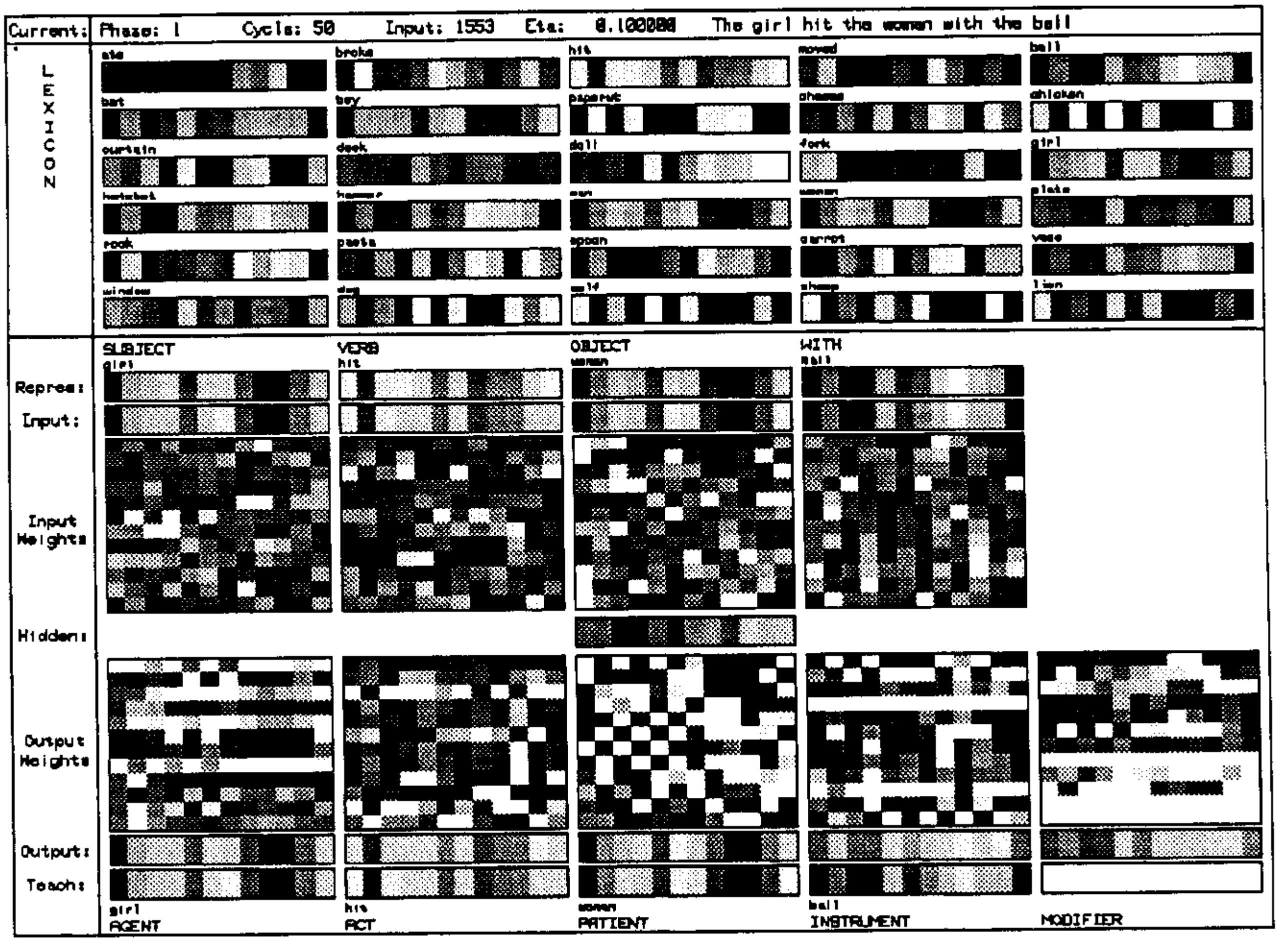

Figure 2: Final state of the system

put, hidden, and output layers) and the representation values (in the lexicon, representation layer and teaching input layer) range from zero to one and are coded as shades of gray from white to black in the figure. The weights are unlimited; the figure displays the interval [-1, 1] from white to black.

The process begins with the components of the representations uniformly distributed in the interval [0,1] and the connection weights in the interval [-1,1]. A set of input sentences and their correct role assignments are generated. The system cycles through the data set 50 times in a random order. Each sentence is presented to the network and an output activation pattern is produced. The correct activity pattern is loaded into the teaching layer and error signals are formed for each output unit. The network propagates back the error, changing weights according to the backward error propagation rules, and changing the lexicon representation of the input concepts according to equations (1) and (3). Next time the same concepts occur in a sentence, their *new* representations are used in the input and in teaching. Gradually the network converges to a set of representations and weights which facilitate appropriate performance in the case role assignment task.

In this particular task the teaching input is made up from the input sentence constituents. This is by no means necessary for learning the representations. The required output of the network could be anything and the FGREP method would work the same. As a matter of fact the convergence in a "pigeonholing" task is slower than in a general task because the *required output is changing as the representations change.*

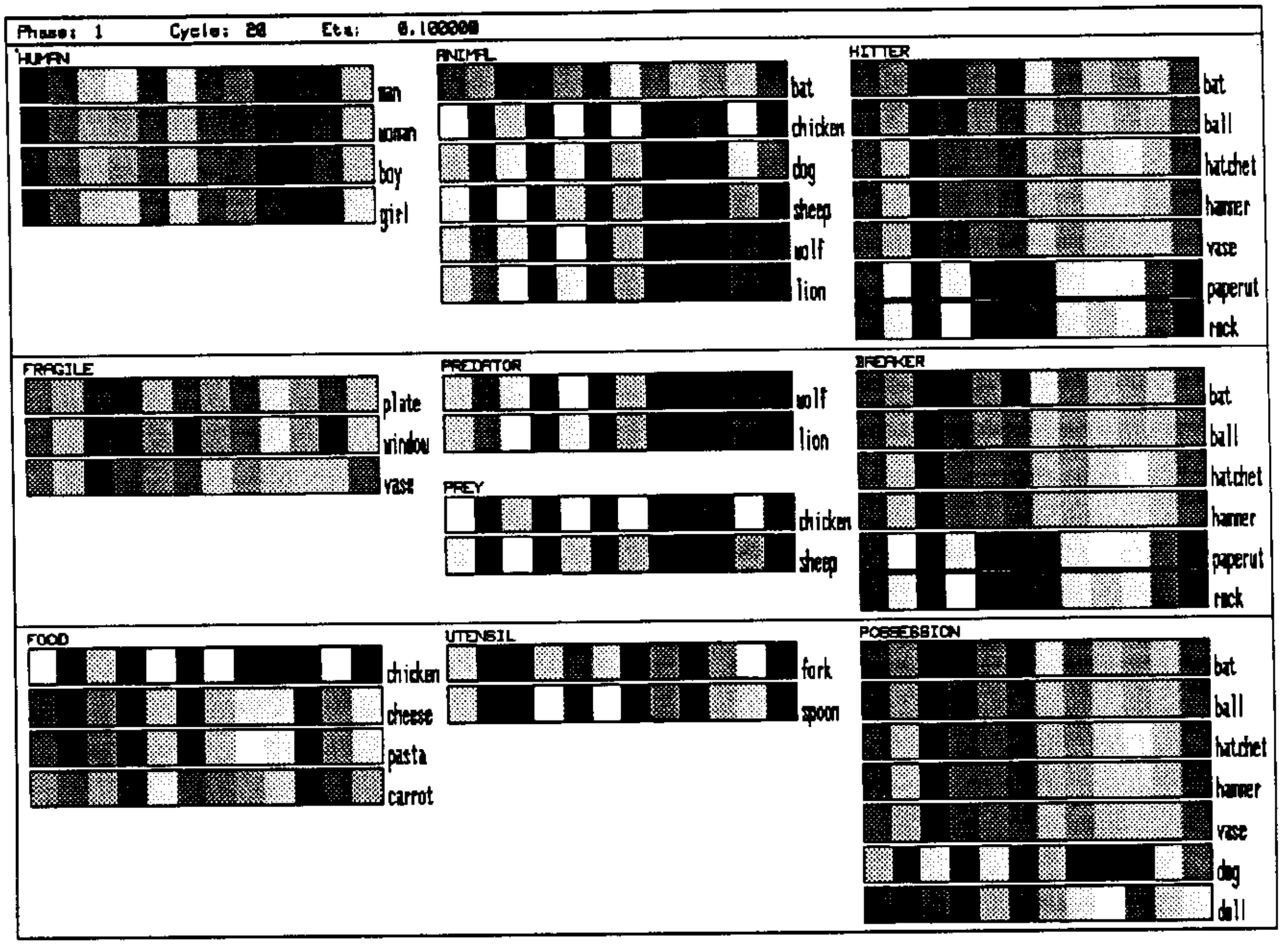

Figure 3: Representations have converged to reflect the categories

3.2 Developing the Representations

Figure 3 shows the representations developed by the system, organized according to the noun categories. Starting from random representations, the similarity of the nouns belonging to the same category increases until the changes begin to cancel out. This happens usually within the first 20 cycles. During the remaining cycles, the representations are fairly stable while the task performance still improves. With different initial representations, the final set of representations in general is different. The overall characteristics of the representations and the performance of the system is approximately the same in all cases.

Some concepts belong exactly to the same categories and consequently occur exactly in the same contexts. They are indistinguishable in the data and their representations become identical. `[man, woman, boy, girl]` forms one such equivalence class, `[fork, spoon]`, `[wolf, lion]`, `[plate, window]`, `[ball, hatchet, hammer]`, `[paperwt, rock]` and `[cheese, pasta, carrot]` others. If there is at least one difference in usage of two nouns, their representations become different. The discriminating input modifies one of the representations while the other one remains the same. Since each noun belongs to several categories its representation can be seen as evolving from the competition between the categories. This is clearest on the part of the ambiguous nouns `chicken` and `bat`, which on the other hand are both `animals`, but `chicken` is also `food` and `bat` is a `hitter`. The representation is a combination of both, weighted by the number of occurrences of each meaning. On the other hand, the fact that there is a common element in two categories tends to make all representations of the two categories more similar. In other words, the properties of one concept are generalized, to a degree, to the whole class.

A single unit does not play a crucial role in the classification of concepts. The fact that a concept belongs to a certain category is indicated by the total pattern of activity instead of particular units being on or off. *There are no identifiable microfeatures in this kind of representation*. One should rather talk about activity profile characteristics. Note that the categorization of a concept in figure 3 is formed outside the system and is independent of the task, other categories and other concepts. The system itself is not attempting categorization, it is forming the most efficient representation of each concept for the particular task. The representations of similar concepts become more alike, but this is a continuous process which occurs in all categories and between all concepts at the same time.

3.3 Performance with the Learned Representations

The system was trained with an input set where two sentences from each sentence generator were reserved. The performance in the role assignment task after 50 cycles was tested with two test sets: the first one consisted of two sentences from each generator which were included in the training set, and the second one consisted of the reserved sentences. The first test measures the association capabilities of the system, while the second test measures the capability of the system to generalize into unfamiliar situations. The robustness of the representations was tested by removing the last *n* % of the units from each input assembly and measuring the association and generalization capabilities of the damaged system. Tables 3 and 4 present results for each sentence. The leftmost entry in each row of these tables identifies the generator which produced the sentence (referring to table 1). The degrees of damage were 0, 25 and 50 percent, which means that 0, 3 and 6 units were removed from each input assembly. The figures indicate the percentage of output units whose values were within 15 percent of the correct output value. More detailed test runs are presented in [M87].

TABLE 3: FINAL PERFORMANCE (FAMILIAR SENTENCES)

Gener	Input			Damage%:	0.0	25.0	50.0
1.	man	ate			81.7	66.7	65.0
1.	girl	ate			81.7	66.7	65.0
2.	woman	ate	cheese		96.7	70.0	60.0
2.	woman	ate	pasta		98.3	73.3	60.0
3.	woman	ate	chicken	pasta	95.0	60.0	53.3
3.	man	ate	pasta	chicken	85.0	70.0	48.3
4.	girl	ate	pasta	spoon	93.3	71.7	65.0
4.	boy	ate	chicken	fork	93.3	75.0	53.3
5.	dog	ate			68.3	60.0	60.0
5.	sheep	ate			73.3	63.3	58.3
6.	lion	ate	chicken		81.7	61.7	55.0
6.	lion	ate	sheep		83.3	65.0	58.3
7.	woman	broke	window		90.0	71.7	58.3
7.	boy	broke	plate		90.0	78.3	56.7
8.	man	broke	window	bat	91.7	66.7	68.3
8.	boy	broke	plate	hatchet	88.3	68.3	65.0
9.	paperwt	broke	vase		88.3	83.3	78.3
9.	rock	broke	window		88.3	78.3	63.3
*10.	bat	broke	window		71.7	66.7	51.7
10.	wolf	broke	vase		78.3	65.0	55.0
11.	vase	broke			86.7	80.0	70.0
11.	window	broke			80.0	75.0	75.0
12.	man	hit	pasta		100.0	88.3	70.0
12.	girl	hit	boy		98.3	90.0	70.0
*13.	man	hit	girl	hatchet	83.3	70.0	51.7
13.	woman	hit	man	doll	90.0	76.7	56.7
14.	woman	hit	bat	hammer	100.0	85.0	71.7
*14.	girl	hit	woman	ball	80.0	80.0	70.0
15.	hatchet	hit	pasta		98.3	90.0	86.7
15.	hammer	hit	vase		100.0	95.0	86.7
16.	man	moved			88.3	73.3	75.0
16.	woman	moved			85.0	73.3	71.7
17.	woman	moved	plate		98.3	73.3	70.0
17.	girl	moved	pasta		98.3	83.3	70.0
*18.	chicken	moved			71.7	61.7	71.7
18.	lion	moved			75.0	60.0	70.0
19.	doll	moved			90.0	80.0	71.7
19.	desk	moved			95.0	81.7	80.0
AVERAGE%					87.8	73.6	65.4

TABLE 4: FINAL PERFORMANCE (UNFAMILIAR SENTENCES)

Gener	Input			Damage%:	0.0	25.0	50.0
1.	boy	ate			85.0	66.7	63.3
1.	woman	ate			83.3	66.7	63.3
2.	woman	ate	chicken		93.3	65.0	58.3
2.	man	ate	chicken		95.0	66.7	58.3
3.	woman	ate	chicken	carrot	91.7	66.7	55.0
3.	boy	ate	carrot	pasta	91.7	66.7	50.0
4.	man	ate	chicken	fork	93.3	73.3	51.7
4.	woman	ate	carrot	fork	91.7	73.3	58.3
5.	bat	ate			71.7	56.7	56.7
5.	chicken	ate			75.0	65.0	61.7
6.	wolf	ate	chicken		78.3	66.7	56.7
6.	wolf	ate	sheep		80.0	65.0	61.7
7.	girl	broke	plate		90.0	71.7	58.3
7.	woman	broke	plate		88.3	70.0	56.7
8.	man	broke	vase	ball	90.0	73.3	66.7
8.	girl	broke	vase	hatchet	95.0	76.7	75.0
9.	hammer	broke	vase		93.3	85.0	71.7
9.	ball	broke	vase		95.0	86.7	76.7
*10.	bat	broke	vase		75.0	68.3	61.7
10.	dog	broke	plate		73.3	68.3	53.3
11.	plate	broke			81.7	80.0	73.3
11.	plate	broke			81.7	80.0	73.3
12.	boy	hit	girl		95.0	91.7	68.3
12.	girl	hit	carrot		100.0	86.7	70.0
*13.	man	hit	boy	hammer	80.0	71.7	46.7
13.	boy	hit	woman	doll	91.7	76.7	56.7
14.	girl	hit	curtain	ball	100.0	85.0	68.3
14.	girl	hit	spoon	rock	98.3	78.3	63.3
15.	paperwt	hit	chicken		86.7	85.0	81.7
15.	rock	hit	plate		95.0	81.7	70.0
16.	boy	moved			85.0	73.3	73.3
16.	girl	moved			85.0	73.3	73.3
17.	man	moved	window		100.0	75.0	70.0
17.	girl	moved	hammer		100.0	81.7	71.7
18.	wolf	moved			71.7	61.7	76.7
18.	sheep	moved			65.0	61.7	71.7
19.	paperwt	moved			75.0	68.3	58.3
19.	hatchet	moved			96.7	83.3	78.3
AVERAGE%					87.3	73.5	64.7

The system learned the correct assignment of most sentences. Difficulties arise in ambiguous cases (indicated by '*'), where the system develops an intermediate output between the two possible interpretations, as shown in figure 2. The system also performs poorly with the `animal` meanings of `bat` and `chicken`. Because a vast majority of the occurrences of `bat` are `hitters`, and `chicken` is more frequently food than `animal`, their representations become better `hitter` and `food` than `animal`.

Strikingly, there seems to be very little difference in performance between the familiar and unfamiliar cases. *The generalization capabilities of the system are excellent.* This is a result of the system's tendency to develop similar representations to similarly behaving concepts. In a precoded, fixed microfeature -based system, such as McClelland and Kawamoto's, even though two concepts are equivalent in the input set, their representations remain different. With FGREP approach the representations become more similar, which means that generalization is necessarily stronger. *Damage resistance is also very good.* Even with *half* the input units removed, the system gets 65% of the output within 15% of the correct value. This is partly due to the general robustness of hidden-layer networks but also partly due to the fact that the representation is not coded into specific microfeature-units, but is distributed over all units.

The behaviour of the system is fairly insensitive to the learning parameters and system configuration parameters. Several different values of the learning rate η within the range 0.01 - 5.0 were tried and also increasing it in six steps from 0.01 to 0.5 (see [P86]). All these cases lead to essentially identical results. The number of units in the representation and the number of hidden units are not crucial either: values as low as 5 and as high as 100 were tested. If more hidden units are used, the task performance, generalization capability and damage resistance improves slightly and the learning in general is faster. Decreasing the number of hidden units on the other hand lays more pressure on the representations, and a more uniform set of representations results. In general, the best results are obtained when the number of hidden units is somewhat less than half the number of units in the input layer.

It must be noted that direct comparison of performance to McClelland and Kawamoto's system is hard because of the different architectures and goals. Percentage of correct units is not a very good performance measure for their system since in all cases most of the output units should be off. Also, to learn the representations properly the training set should be as large as possible, whereas if the representations are predetermined, smaller training sets can well be used. Generalization capabilities as a function of training set size have not been tested.

4. RELATED AND FUTURE WORK

In McClelland and Kawamoto's system, the representations were hand coded as a collection of microfeatures and remained fixed throughout the experiments. The network consisted of two layers of binary-valued units with the input and output layer both divided into assemblies. A concept was represented in an input assembly as the outer product of the concept's microfeature vector with itself. Each unit in an input assembly stood for a conjunction of two microfeatures of the concept, and each unit stochastically determined whether it was going to be on or off, biasing the decision according to the activation the unit received from the two microfeature units. In an output assembly a concept was represented according to the same principle, except that the outer product was formed of the concept's microfeature vector and the microfeature vector of its head concept. The weights between the fully connected input and output layers were adjusted according to the perceptron convergence procedure. The primary goal of the experiment was to teach the system to assign correct semantic roles to the concepts, based on their syntactic role and the semantic context within the sentence [R86 Ch.19]. The system described in this paper, on the other hand, is a three layer backward error propagation network with the error propagation extended to the input layer. The units produce deterministic activation values between zero and one. Input and output concepts are represented directly as vectors of these units. The initial representation is random and adapts continuously to the input data. The main goal of the experiments was to show that *microfeature coding is not necessary. Meaningful global representations can be developed by the system itself while it is learning the processing task.*

In [H86] Hinton describes a system which develops internal distributed representations of concepts. This back propagation network consists of five layers: input, output and three hidden layers. The input and output layers have exactly one unit dedicated to each input/output concept. The hidden layers next to the input and output layers contain considerably fewer units, which forces these layers to form compressed distributed activity patterns for each input/output concept. Since both penultimate layers develop their activity patterns independently, each concept has two different representations: one as input and another one as output. It is possible to impose linear constraints on the input and output activity patterns and force these to become the same [H87]. This kind of system would develop representations very similarly to our model. However, we have chosen to treat *the representations as separate objects global to the system, instead of local activity patterns in the internal layers.* Our architecture reflects the eventual aim of using the system as a building block in larger language understanding systems. Looking at the input representations, Hinton is able to identify some features of the input with particular units, while others do not have any clear interpretation. Our results suggest, that in a more complex task, interpreting individual units is not possible. The features of the concepts are in general distributed over the set of units, which also seems to make the representation more robust against damage.

The prime direction of further work is to use the learning of distributed representations as a building block in higher level language understanding systems. Major issues include: (1) invariance versus adaptability of the representations over changing data, (2) efficiency in the task performance versus portability of representations between building blocks, (3) hierarchical versus independent building of deeper meanings, and (4) hierarchical versus competition-based creation of sequentiality [M87].

5. CONCLUSION

The FGREP method provides an alternative to fixed microfeature encoding of concept representations. With backward error propagation extended to the input layer, meaningful global representations develop automatically while the system is learning the processing task. There are no identifyable microfeatures nor discrete categorizations in the resulting representation. All aspects of a concept are distributed over the whole set of units as an activity profile, making the system particularly robust against damage. The representation evolves to improve the system's performance in the processing task and therefore efficiently codes the underlying relations relevant to the task. This results in excellent association and generalization capabilities. Using the learned representations, the system is able to assign case roles correctly, indicate degree of confidence when the sentence is ambiguous, and generalize correctly for unfamiliar sentences.

Our approach is based on the philosophy that *concepts are defined by the way they are used.* Learning a language is learning the usage of the language elements; *language is a skill.* The meaning of a concept is encoded in its representation. This is defined by all the contexts where the concept has been encountered, and it determines how the concept behaves in different contexts. The representation as well as the meaning evolves continuously as more experience is gained. On the other hand, *the representation carries a memory trace of all the contexts that defined it.* The more frequent the context, the stronger is the trace. When a concept is encountered and its representation activated, a large number of expectations about the context are immediately active with different degrees of confidence, corresponding to the memory traces. As more concepts are input, the expectations created by the memory traces are amalgamated and the set of possible interpretations narrows down. All this emerges automatically and continuously from the input concept representations, and should turn out to be useful in larger language understanding systems.

REFERENCES

[H81] G. Hinton. Implementing Semantic Networks in Parallel Hardware, in Hinton & Anderson (eds.), *Parallel Models for Associative Memory*, LEA Press, 1981.

[H86] G. Hinton. Learning Distributed Representations of Concepts, in *Proc. of CogSci-86*, 1986.

[H87] G. Hinton. Personal communication, 1987.

[M87] R. Miikkulainen & M. Dyer. Building Distributed Representations without Microfeatures, Technical Report UCLA-AI-87-17, UCLA 1987.

[P86] D. Plaut, S. Nowlan, G. Hinton. Experiments on Back-Propagation, Technical Report CMU-CS-86-126, CMU 1986.

[R86] D. Rumelhart, J. McClelland and the PDP Research Group. *Parallel Distributed Processing*, MIT Press, 1986.