Knowledge-Intensive Recruitment Learning

Joachim Diederich\(^1\)

TR-88-010

November, 1988

ABSTRACT

The model described here is a knowledge-intensive connectionist learning system which uses a built-in knowledge representation module for inferencing, and this reasoning capability in turn is used for knowledge-intensive learning. The method requires only the presentation of a single example to build a new concept representation. On the connectionist network level, the central process is the recruitment of new units and the assembly of units to represent new conceptual information. Free, uncommitted subnetworks are connected to the built-in knowledge network during learning. The goal of knowledge-intensive connectionist learning is to improve the operationality of the knowledge representation: mediated inferences, i.e. complex inferences which require several inference steps, are transformed into immediate inferences; in other words, recognition is based on the immediate excitation from features directly associated with a concept.

\(^1\)International Computer Science Institute, 1947 Center St., Suite 600, Berkeley, CA 94704-1105
Abstract

The model described here is a knowledge-intensive connectionist learning system which uses a built-in knowledge representation module for inferencing, and this reasoning capability in turn is used for knowledge-intensive learning. The method requires only the presentation of a single example to build a new concept representation. On the connectionist network level, the central process is the recruitment of new units and the assembly of units to represent new conceptual information. Free, uncommitted subnetworks are connected to the built-in knowledge network during learning. The goal of knowledge-intensive connectionist learning is to improve the operationality of the knowledge representation: mediated inferences, i.e. complex inferences which require several inference steps, are transformed into immediate inferences; in other words, recognition is based on the immediate excitation from features directly associated with a concept.
1. Introduction.

1.1 Data- and knowledge-intensive learning.

An important classification in symbolic machine learning is the distinction between data- and knowledge-intensive learning methods. Data-intensive learning techniques require a set of training examples which is used to guide and constrain the search for possible generalizations. The notion of similarity-based generalization is used by Mitchell et al. (1986) to refer to learning methods which search for similarities among positive examples, and differences between positive and negative examples in a training set.

The method described in this paper is knowledge-intensive, i.e. the technique uses knowledge which is already learned or built-in. The motivation of this work is to explore connectionist learning methods which can use already available and structured knowledge to facilitate concept learning.

The built-in knowledge base is a connectionist semantic network which is used to recognize a single example. Recruitment learning is applied to build a new conceptual structure and to connect it to the existing knowledge representation system. This acquisition of a new chunk of knowledge is called constructive generalization (also see Langley 1986). The new conceptual structure will improve the system’s capability to recognize instances as described by the single example.

The kind of learning outlined in this paper can be described by the following "real world" example: a person sees a car and forms the hypothesis that this car is a "Porsche." The hypothesis is verified by comparing the features of the perceived car with typical features of "Porsches." The person realizes that this is a special kind of "Porsche" with additional features which are not owned by "Porsches" in general. The new knowledge about this special kind of "Porsche" is added to the general knowledge about cars from this manufacturer.
1.2. Organization of the paper.

This paper is organized as follows. First, a brief introduction to connectionist recruitment learning is given. Second, the architecture of the built-in knowledge base is introduced, including a brief description of connectionist inheritance. Essential parts of the learning algorithm are outlined next, including a formal description of the method. An example of knowledge-intensive connectionist learning is given, experimental results for the learning algorithm are presented and possible methods for the learning of multiple concepts and concepts with disjunctive attributes are outlined. Finally, extensions of the learning technique are discussed.

2. Connectionist recruitment learning.

Recruitment learning is considered to be one of the most important learning methods for connectionist knowledge representation systems. It can be used for both data-intensive learning, which requires a set of training examples to be used in the search for possible generalizations, and knowledge-intensive learning, which uses already learned or built-in knowledge for generalization.

In recruitment learning, a network consists of two classes of units:

Committed Units

These are units which already represent some sort of information or function, i.e. conceptual information. Committed units are connected to other committed units and their simultaneous activation must represent a meaningful state of the network. Committed units are also connected to

Free Units

which form a kind of "primordial network" (Shastri 1985) and are connected to other free units as well as to committed units.

Recruitment learning is the strengthening of the connections between a group of committed units and one or more free units. This results in the transformation of
free units into committed units. The term "chunking" is used if "a free node becomes committed and functions as a chunking node for the cluster [of committed units, this author], i.e., the activation of nodes in the cluster results in the activation of the chunking node, and conversely, the activation of the chunking nodes activates all the nodes in the cluster" (Shastri 1985). It is possible to view the subnetwork of committed units embedded in the network of free units. Constructive generalization means the recruitment of units from the pool of free units.

There are at least five possible applications for recruitment learning as part of connectionist knowledge representation systems:

a) Concept Binding

Shastri (1985) used a partitioned random graph to build basic knowledge structures. It consists of units representing concepts, attributes of concepts and values of attributes. The central idea is to use binder units to connect concepts and instances with attributes and values, and to make these structures accessible for retrieval and question-answering. A typical binder unit has three sites where input-lines come in, and has a specialized activation function which requires input from at least two sites for a positive output.

Figure 1: Connecting a concept-, attribute- and value unit with a binder unit.

For example, an agent sees an interesting car which is both yellow and fast, and recruits a structure equivalent to: car(B1), color(B1, yellow), speed(B1, fast). During classification, the concept car can be retrieved by clamping color
and yellow on. On the other side, it is possible to retrieve the value yellow by clamping car and color on during property retrieval.

b) Concept Formation from Several Instances

Concept formation by several instances involves presenting several examples (i.e. sets of features) to a network (either sequentially or simultaneously) and analyzing similarities among these examples. This is a special case of induction. Concept formation happens if the number of instances exceeds a certain threshold and if high similarity among the examples is determined. One or more concept units is/are then recruited, and weighted connections between these concept units and common features of the training examples are established. For example, the agent sees several instances \( B_1 \in (B_1, B_2 \ldots B_n) \) with speed\( (B_1, \text{fast}) \) and forms a concept sportscar with an attribute speed\( (\text{sportscar, fast}) \).

![Diagram of concept formation](image)

Figure 2: Simple concept formation.

(c) Higher-Level Concept Formation

Higher-level concept formation means the generation of a new super-concept triggered by the analysis of a number of concepts which are already part of the connectionist knowledge representation system. A super-concept is built and integrated into the existing organization of the knowledge representation network, and the analyzed concept units get positive weighted connections to the new super-concept. This process requires not only a positive weight change between a new knowledge structure and the already existing network of committed units, but also the unlearning of weights between the analyzed concept units and the former super-concept. Higher-level concept formation is considered to be an important and difficult form of recruitment learning.
d) Simple Specialization

Specialization means to add a new chunk of knowledge as a specialization of an existing concept to a knowledge network. The concept to-be specialized is included in the input. The new concept unit must get weights to the specialized concept and to
the features which define the newly acquired concept. No modification of the network of committed units is required. In our example, the agent knows \#wheels(car, 4) and sees one or more instances $B_i \in (B_1, B_2, ..., B_n)$ of car, i.e. car($B_i$) with speed($B_i$, fast). Consequently, the agent forms the concept sportscar and asserts car(sportscar) with speed(sportscar, fast). It is also possible that the new concept sportscar gets additional attributes inferred from domain knowledge, i.e. super-concepts of car (this is not part of this example). A connectionist realization of this process is described in Diederich (1988a,b) and in the following sections of this paper.

Figure 4: Recruitment of a unit which represents the specialization of a generic concept.

e) Specialization by Integration

Specialization by integration requires not only the enrichment of a network by a more specialized concept, but also the modification of parts of the knowledge structure. This is always necessary when the new, specialized concept has sub-concepts which must cut their connections to the former super-concept and which must establish connections to the newly recruited concept. This case is similar to higher-level concept formation, but requires the specification of a goal-concept as part of the input, as well as a training example which consists of a set of attribute/value pairs.

For instance, an agent knows car($C_i$) where car and $C_i$ are concepts and $C_i \in$ (Porsche, Ferrari, BMW). The agent sees an instance $B_i$ of car with speed($B_i$, fast) and forms a new concept sportscar which is a car with the
attribute \text{speed}(\text{sportscar}, \text{fast}). Furthermore, the agent deletes for all C_i: 
\text{car}(C_i), i.e. the connections between all \text{C}_i and \text{car} are cut in the connectionist system, and asserts for all C_i: \text{sportscar}(C_i).

Figure 5: Learning the specialization of a concept and integrating it into a taxonomic hierarchy.

This modification of existing network structures is a non-trivial process in a massively parallel connectionist system. Some parts of the network must remain unchanged while other parts undergo modifications; this is a difficult problem for connectionist systems and is excluded in approaches such as Valiant (1988). All decisions and modifications must be done in a connectionist way; there is no top-level interpretation and no programming of network structures. A possible solution to this problem is described in Diederich (1988c).

3. The architecture of the connectionist knowledge representation.

The knowledge base is constructed by compilation from a symbolic descriptive representation language or by previous learning. The built-in representation system used here consists of four parts: the concept space, the attribute space, the instance space and the space of free units. The concept space contains all learned or built-in concepts. The attribute space contains the possible values of attributes (called features). The instance space consists of single units, connected to concept units and feature units in the attribute space. The free space consists of uncommitted units which are recruited during learning.

All spaces are physically fully connected, i.e. each unit is physically connected with all other units in other spaces of the network. A 0-valued weight means "no
association." Weights represent the strength of association. Weights between the concept and attribute space (and vice versa) are subject to weight change during classification. Throughout this paper, however, only the weight change between free space and concept space, and free space and attribute space is significant.

3.1 The concept space.

The chosen representation of the concept space is similar to the one in Cottrell (1985).

The representation consists of three-unit subnetworks for the representation of each concept (called "triple subnetworks" or "3-unit networks"): the affirmative concept unit, the negative concept unit, and an additional intermediate unit. The affirmative and the negative unit have excitatory links to the intermediate unit, and the intermediate unit has inhibitory links to the affirmative and the negative unit. The intermediate unit guarantees that the subnetwork converges to one of three stable states (see Cottrell 1985). Either the affirmative unit or the negative unit is on, or all units are off. See Figure 6 for an explanation.\(^1\)

Subnetworks are connected in a hierarchy or heterarchy, reflecting the inheritance path. There are links from each affirmative unit to more general affirmative units, and from each negative unit to more specific negative units.

![Diagram](image)

Figure 6: A triple unit subnetwork.

\(^1\) It should be noted that the same effect can be achieved by a mutual inhibitory link with delay of propagation between two units only.
Units of the concept space are connected with units in the attribute space. Concepts are thereby associated with their properties. Inheritance can be described by the following example: an affirmative concept unit A is activated by some input. If A is strong enough, it becomes the winner in the 3-unit subnetwork of the concept space. It will therefore activate all units in the attribute space associated with A and the more general unit B. B itself will also activate its properties in the attribute space; therefore additional feature units become "on".

This process corresponds to simple inheritance in semantic networks. While the spreading activation process continues, more and more general concept units are activated and more and more relevant feature units are turning on. Multiple inheritance works with the same advantages and disadvantages as in standard semantic networks (see Touretzky 1986 for examples). In general, "path length" is no guide for a decision between inferences in connectionist inheritance systems. In a connectionist system, all super-concepts of a given concept will become active until the stable state is reached.

Exceptions which would realize a non-monotonic style of reasoning are built in by additional inhibitory links which modify the flow of activation from more general concept units to units in the attribute space. These links modify the inheritance process. More specific units can modify the effect that links from more general concept units have on units in the attribute space. So the connections between more specific concepts and their attributes become dominant. This realizes a basic assumption of semantic networks: the more specific should dominate the more general.

3.2 The attribute space.

The attribute space is a network divided into subnets and each subnet contains the possible values of a single attribute. Subnets are a variant of "winner take all" (WTA) networks which use mutual inhibition (see Feldman & Ballard 1982). Currently there is no explicit representation of an attribute itself; each subnetwork represents an attribute implicitly. Whenever an unit in a subnetwork of the attribute space is "on," the corresponding attribute is implicitly given.
In order to maintain likelihood information, a modification of WTA networks is used which adjusts competition among units according to the total input of a WTA network. This kind of approach is sometimes called "winner take more" (WTM) network. Our approach here is similar to the "network region" model in Chun et al. (1987): each unit in a WTM network receives the average of the total WTM network input as an inhibition signal; the result is the restriction of the dominance of winner units. All competing units with an input above average might stay active.

The total WTM network input is computed indirectly using the output values of all units in the WTM subnetwork. For each WTM network, an additional unit is built which receives the output of all units in the WTM net and computes the average. This additional unit, called "region unit," has inhibitory links to the units in the WTM net. The average signal is propagated to the WTM units via these inhibitory links.

3.3 The instance space.

Instances are single units connected with concept units. Activation of an instance unit is followed by the activation of a concept unit. If an instance has additional properties or properties which are different from the attributes of the associated concept, explicit links to feature units in the attribute space are necessary. Presenting a training example to the system is done by clamping on an instance unit which activates its concept unit and additional units in the attribute space, but only if this training instance owns features not part of the concept definition.

3.4 The free space.

The fourth space consists of free units. Parts of these free units become committed during recruitment learning. Free units are connected like concept units (triple subnetworks). The subnetworks of the free space are connected as a strict WTA network which allows only one unit of a triple subnetwork as a winner unit (with the single exception of the intermediate unit) while none of the other subnetworks in the free space has an active unit. Furthermore, there are bidirectional links between each feature unit in the attribute space and each free unit, and each affirmative or negative concept unit with each affirmative or negative free unit. These links initially have small random weights and are subject to weight change during recruitment learning.
Concept Space

Attribute Space

WTM network
region unit
Feature units

Instance Space

Free Space
(3-unit subnetworks with mutually inhibitory connections)

Instance unit/
Concept unit link
Concept unit/
Feature unit links

Free unit/
Concept unit links
(possible random units are not shown)

Instance unit/
Feature unit link
Figure 7 (previous page): The architecture of the system (not all links are shown).

Free units receive their own output as a strong positive reinforcing signal and the output of the competing free units as negative reinforcement (inhibition). This is similar to the competitive reinforcement learning in Lynne (1987). If a training example is presented to the system, the high activity in the attribute space will lead to nonspecific activation of the free space. The free unit with the highest potential will receive this activation (among others), and thereby becoming a clear winner, in part through self-reinforcement. In the case of random connectivity (this connection pattern is described below), the unit with the strongest connections receives most of the activation and becomes the winner unit which is committed during recruitment learning.

This part of the learning process is similar to competitive learning (cf. Rumelhart & Zipser 1985, Grossberg 1987) where only the winner unit gets the chance to adjust weights. However, the weight change is different in competitive learning because there is a redistribution of weights over the input lines of a winner unit. In our approach, each input link of a newly-committed unit becomes adjusted independently of other input-lines and features of the weight-vector. The weight change is based on the input-signal and the potential of the receiving unit (see section 5).

Two patterns of connectivity are explored between concept space and free space, and attribute space and free space. The first case is full connectivity. There are links between each free unit and each concept/feature unit (with the exception of the intermediate units; furthermore, affirmative free units are only connected with affirmative concept units and negative free units are only connected with negative concept units).

Obviously, full connectivity must deal with the problem of combinatorial explosion when many free units, concept units and feature units are used. Random connections are therefore used as the second connectivity pattern. Additional random units are intermediate units between attribute space and free space. There is full connectivity between attribute space and random units, and there are random links between random units and units in the free space. During network construction, there is only a
50% chance for a link between a random unit and a free unit. As in the first case, each affirmative and negative free unit has links to each feature unit.

4. Knowledge-intensive learning in a structured connectionist system.

The following sources of information are used in our system: the domain theory, which is the total knowledge base minus the free space; the goal concept, which is an affirmative concept as part of the concept space; and the training example, which is a set of attribute/value pairs, describing an instance of the goal concept. The feature units in the attribute space which represent the values of an attribute as part of the training example are continuously activated by an instance unit. The new concept is defined in terms of all attribute/value pairs of the training example and attributes from the domain theory. Furthermore, we define as an operationality criterion that mediated inferences have to be transformed into immediate inferences, i.e. a conclusion based on property values can be drawn in one inference step instead of a more complex chain of inferences (see Hollbach 1988 for details). As an additional technical restriction we assume here that each attribute has only one individual value.

The algorithm has two parts, "Recognition" and "Constructive Generalization," which are conceptual distinctions and are not separated during processing:

Recognition:

1. Let the instance unit which represents the training example activate all connected feature units in the attribute space.

2. Clamp on the goal concept unit in the concept space.

3. Run the total network until convergence is reached in order to build an activation pattern over the concept space. This resulting pattern corresponds to the chain of inference.
Constructive Generalization:

4. Recruit new triple structures for representing the new concept definition.

5. Run the total network to allow weight changes.

During recognition, an activation pattern is built in the attribute space which includes features of the training example necessary to describe the goal concept. In phase 3 of the algorithm above, less specific features from more general super-concepts of the goal concept also become active. A strong decay process is used in the concept space to restrict the influence of these less specific features. Strong activated units represent features of the training example and the goal concept. Convergence is successful, if a) all network regions in the attribute space (all WTM networks) have clear winners, otherwise (according to our technical restriction), the training example does not fit the goal concept and b) the total network stabilizes.

The constructive generalization process is simple. The wiring between free and attribute space is already done but all weights have small random values. The weighted connections between the new concept unit and features represented by feature units in the attribute space are installed by using a modified Hebb rule with slow weight change. Because of this weight change rule, the new concept will get strong connections to features of the goal concept and/or the training instance, and weak connections to more general concepts on higher-levels as the goal concept.

Furthermore, the new concept has weighted connections to units in the concept space. There will be a strong connection between the newly recruited unit and the goal-concept unit because the new concept unit is the winner unit in the free space and the goal-concept unit is clamped on. The new concept unit has small weighted connections to super-concepts of the goal-concept also, but the decay process in the concept space is used to restrict the increase of these connections. The connection between the new concept unit and the goal-concept unit will be dominant.

Step 5 of the algorithm is used to allow these weight changes. Please note that it is the weight change rule which restricts the generalization process.
5. A formal description of the method.

We are defining the system in roughly the same way as Shastri (1985, pp.56). However, the objective of this work is to define recruitment learning for a structured connectionist knowledge representation system.

The system is specified on two related levels: 1. On the conceptual level where generic concepts denote classes of individuals with common features and individuals represent instances of these concepts. 2. On the network level units are organized in sets called "spaces" and interact via weighted links. Because we are using a localist approach, in most cases conceptual entities are represented by a single unit. However, it is important to distinguish between the conceptual level and the network realization, i.e. the mapping of a conceptual representation to a connectionist network. In contrast to similar approaches, such as those by Shastri (1985), Derthick (1987) and Touretzky (1987), the focus of attention here is on the enlargement of the network through recruitment learning.

The following notation is used throughout this section: capital letters (C, A, ...F) denote sets. Lower case letters denote individual members of sets, e.g. c ∈ C, a ∈ A.

5.1 The conceptual level

On the conceptual level the model is a n-tuple \( W = \{ C, A, V, I, F, \lambda, \alpha, \leftarrow \} \), where C is the set of all concepts, A is the set of all attributes, V is the set of all possible values of all elements in A (features), I is the set of all instances, F is the set of all free, uncommitted entities which become committed during learning, \( \lambda \) is a mapping from C to the power set of A, \( \alpha \) is a mapping from A to the power set of V, and \( \leftarrow \) is a partial ordering defined on C.

Furthermore, we define the function \( \beta(c) \) which returns all values of all attributes of the concept \( c \in C \):

\[
\beta(c) = \bigcup \alpha(a_i) \forall a_i, a_i \in \lambda(c)
\]
For each $c \in C$, $\lambda(c)$ is the subset of $A$ which is owned by $c$; e.g. $\lambda(\text{car})$ might be \{size, color\}. For each $a \in A$, $\alpha(a)$ is the subset of $V$ that consists of the values of $a$; e.g. $\alpha(\text{color})$ might be \{yellow, blue\} and $\alpha(\text{size})$ might be \{big, medium, small\}. If $\lambda(\text{car}) = \{\text{size, color}\}$, then $\beta(\text{car}) = \{\text{big, medium, small, yellow, blue}\}$.

$C$ has a partial ordering, organized as a taxonomic hierarchy: if $c_2$ and $c_3$ subsume $c_1$: $(c_1 \leftarrow c_2) \land (c_1 \leftarrow c_3)$, then $\beta(c_1)$ contains $\beta(c_2) \cup \beta(c_3)$. $c_{gc} \in C$ is the goal-concept.

Free entities have no attributes and values, $\forall f_j, f_j \in F: \beta(f_j) = \emptyset$.

In our system, $i_e \in I$ denotes the training example. $i_e$ must be an instance of the goal-concept $c_{gc}$, $i_e$ therefore inherits all attributes and values from $c_{gc}$ as described above. Moreover, $i_e$ might have attributes and values not shared by $c_{gc}$, but $\beta(c_{gc})$ is at least a subset of $\beta(i_e)$.

Recruitment learning means the transformation of free entities into concepts: $f_j \in F \land \neg(f_j \in C) \implies \neg(f_j \in F) \land f_j \in C$. The operator $\implies$ refers to the process of transforming a free entity into a concept. In other words, $\beta(f_j) = \emptyset$ is transformed into $\beta(f_j) = \beta(i_e) \cup \beta(c_{gc}) \cup \beta(c_{gc} \leftarrow c_n)$ after recruitment, where $c_n \in C$ are all super-concepts of $c_{gc}$. Note that $f_j$ gets only weak connections to entities in $\beta(c_{gc} \leftarrow c_n)$ because these are accessible by inheritance.

5.2 The network level

On the network level the model is a n-tuple of elements $M = \{U, W\}$. $U$ is the set of all units in the network and $W$ is the set of all weights.

Members of $U$ are sorted into four classes of elements which make up $U$. $U^C$ denotes the class of all concept units, nodes which represent a generic concept: $U^C_i$ represents $c_i \in C$. If $U^C_i$ represents $c_{gc}$ and $p_{gc} \in [0, 1000]$ is the potential of $U^C_i$, and if $(c_{gc} \leftarrow c_n)$, then $p_n < p_{gc}$ where $p_n$ is the potential of $c_n$, the super-concept of the goal-concept. On the implementation level, this is guaranteed by the fixed weighted link between $c_{gc}$ and $c_n$. Furthermore, $U^V$ is the class of all feature units in the attribute space with $U^V_i$ representing $v_i \in V$. $U^I$ is the class of all instance units...
in the attribute space with $U^i_j$ representing $i_j \in I$. The fourth and final class of units, $U^f_i$, consists of free units with $U^f_i$ representing $f_i \in F$.

Each weight $w_i$ is an integer $\in [-1000, 1000]$. Positive associated units have weights $w_i > 0$, negative associated units have weights $w_i < 0$, a zero weight means no association. Weights in both directions between all $U^f_i$ and all $U^v_i$ have initially small random values $w_i \in [0, 100]$. We are particularly interested in the distribution of weights after recruitment, i.e. the distribution of weights between the new recruited unit $U^f_i$ and all $U^v_i \in \beta(f_i)$. $U^f_i$ always gets a link with maximum strength to $U^c_{gc}$, because $U^f_i$ has a high potential as the winner of the competition among the free units, and because $U^c_{gc}$ is clamped on.

The following weight change rule is applied (inputs and potentials are integers):

$$\Delta w_{ij} = \begin{cases} +1 & \text{if } (i_j \geq 0 \land w_{ij} < p_j \land w_{ij} < i_j \land w_{ij} < \text{max}) \\ \text{else if } (w_{ij} > \text{min}) & \text{then -1} \end{cases}$$

where $i_j$ is the ith input to the receiving unit, $p_j$ is the potential of the receiving unit, max is an upper limit for weights ($= 990$), and min is a lower limit for weights ($= -990$).

The simulations were done with this modified version:

$$\Delta w_{ij} = \begin{cases} +10 & \text{if } (i_j \geq 250 \land w_{ij} < p_j \land w_{ij} < i_j \land w_{ij} < \text{max}) \\ \text{then +1 else if } (w_{ij} > \text{min}) & \text{then -1} \end{cases}$$

The increase of weights depends on the receiving unit's potential and on the amount of the received input. Furthermore, the upper limit for this increase is the lower value of the potential $p_j$ and the amount of transmitted activation $i_j$.

This weight change rule guarantees the appropriate distribution of weights from $U^f_i$ to all $U^v_j \in \beta(f_i)$; i.e. the new recruited unit gets strong weights to high activated
elements in $\beta(f_i)$, medium weights to medium activated elements in $\beta(f_i)$, and smaller weights to low activated elements in $\beta(f_i)$.

$U^f_j$, representing $f_j \in F$, is the winner unit in the free space and is transformed to the new concept unit. $U^f_j$ has links to all $U^V$, and all $U^V$ have links to $U^f_j$ in case of full connectivity between free and attribute space. Let $p_i$ be the potential of $U^V_i$ and $p_j$ the potential of $U^f_j$. $w_{ij}$ is the weight of the connecting link between $U^V_i$ and $U^f_j$. According to the weight change rule above, $(w_{ij} < p_i) \land (w_{ij} < p_j)$. Because $\forall v_{gc}, v_{gc} \in \beta(c_{gc}) > \forall v_n, v_n \in \beta(c_{gc} \leftarrow c_n); w_{gc \cdot i}$ between $U^f_j$ and $U^V_{gc} > w_{n \cdot j}$ between $U^f_j$ and $U^V_{gc}$. This means $U^f_j$ has stronger links to properties owned by the goal-concept than to properties of super-concepts of the goal-concept.

6. An example.

The experiments were done with six features. The goal-concept (Porsche) has two features, fast and imported. The training example (Porsche-914) has four features: fast, imported, yellow and owner:michael. Furthermore, there are two features owned by the immediate and single super-concept (Car) of the goal-concept: engine and gearbox.

We have now specified a simple example as a training instance, a goal concept and an extremely simple domain theory. Still missing is the criterion for the new concept: the new concept definition must be expressed in terms of structural attributes of the example, the goal-concept or super-concepts of the goal-concept.

The training example can easily be recognized as an instance of Porsche because of its features fast and imported. Additional features of the training example can be inferred by use of the domain theory after a successful recognition.

In full detail the method works as follows. First, the recognition of the training example is done. The feature units fast and imported receive strong activation from the instance and goal-concept unit which causes high potentials for these units. Yellow and owner:michael are stimulated by the instance unit only and get medium activation. The affirmative concept unit Porsche stimulates the concept unit Car which in turn activates other feature units in the attribute space: engine and gearbox.
The features of the example receive continuous activation from the unit representing the instance. After 50 updates of the entire network, all units in the concept space are subject to a decay process. Because of this decay process, the concept unit Car loosees activation. Note that there is only one source of stimulation for Car, namely the goal concept Porsche. The same happens to the feature units engine and gearbox in the attribute space, because these units receive their activation from the super-concept of the goal-concept.

After the decay process becomes effective, the second step of the method continues with the recruitment of the new structure. As mentioned above, triplets of units for the representation of concepts are already built in the free space. In the beginning, the connections have small random weights. The increase of weights between the new concept structure and features represented by active feature units in the attribute space are installed by using the Hebb rule described above. The new concept therefore gets strong connections to the feature units fast and imported, and small- or medium-weighted connections to the feature units engine and gearbox, as well as to yellow and owner:Michael, features of the training example which received no activation from the concept space.

The learning process is finished after installation of the weighted links. The new concept definition contains features of the training example as well as inferred features from domain knowledge. The new concept definition will improve the system's ability to recognize examples similar to the training example because of the distribution of weights between the new concept and its features in the attribute space.

7. Experimental results.

The example above was tested and implemented by use of the Rochester Connectionist Simulator (RCS; Goddard et al. 1987). Simulations were done using fair asynchrony (units are randomly chosen for update; in our simulations, each unit was updated at least once after 243 steps). See Diederich (1988a) for experiments on recruitment learning with random connections.
7.1 The generation of a stable activation pattern in the attribute space.

Figure 8 shows the development of the potentials of feature units in the attribute space during the first 200 steps (full connectivity). Note the effect of the decay process after 50 update steps. Features such as engine and gearbox loose activation after 50 steps because they are connected to concepts higher than the goal concept in the inheritance hierarchy. The results show a small oscillation in the potentials of the feature units caused by the ongoing competition of the winner units in the attribute space.

![Graph showing the generation of a stable activation pattern in the attribute space.](image)

Figure 8: The generation of a stable activation pattern in the attribute space.

It is possible to distinguish among three classes of units in attribute space after the network becomes stable. The first class consists of feature units which are stimulated through the instance unit and units in the concept space. These units receive a high potential, and the new concept unit gets strong connections to these units. Fast and imported are members of the first class in our example. Members of the second class of feature units receive activation from the instance unit only. These have medium activation. The third class of feature units receives activation from super-
concepts of the goal-concept. These have low activation and the new concept unit gets links with low weights to these units (engine and gearbox in the example).

7.2 Weight developments between a newly recruited free unit and the attribute space after the presentation of a single example.

Figure 9 shows the weight development between the newly committed unit and feature units in the attribute space when both spaces are fully connected. Note the decrease of some weights after 50 updates caused by the decay process in the concept space.

![Graph showing weight development over time steps](image)

Figure 9: Weight development between the free space winner unit and feature units in the attribute space (full connectivity).

It makes sense to investigate recruitment learning when several examples are presented in a sequence. When more and more units of the free space become committed, the behavior of the system has the potential to change significantly, because newly committed units have well established weights to units in the attribute space and the concept space.
7.3 **Weight development for a second example with high similarity to the first instance.**

It is important to understand the behavior of the free space when parts of this space are already recruited and therefore part of the concept space, because the network should have the ability to learn several concepts. The recruitment of a triple-sub-network might influence the competition among the remaining units in the free space and interfere with further recruitment.

It is, of course, possible to free the new recruited unit out of the free space by changing the inhibitory connections from and to this unit to 0-valued connections. This is no problem from a technical viewpoint, and will be necessary for large applications. However, there is at least one strike against this procedure: the new recruited unit must change incoming and outgoing links (and not only the input-lines). The rest of this chapter investigates recruitment learning without such a procedure.

**We are starting with the case where an instance has triggered recruitment learning and a new example is presented immediately after the end of this acquisition. This second example has high similarity with the first one: it shares all features with the first instance with the exception of one (which is not owned by the goal-concept). In order to understand the network's ability to acquire a new concept immediately after a first one, the following experimental procedure was used.**

An instance is presented to the system, a goal concept is chosen, and the network runs for 200 update steps in the same way as described above. After 200 update steps all potentials and all output values in all spaces of the network are set to 0. Next a new example is presented, a new goal concept is chosen, and the network runs for another 200 update steps. The weight development for the second example is recorded.

**Figure 10 shows a typical result for the acquisition of a second new concept structure immediately after the first one. Note the weight development between the second recruited unit and features units in the attribute space.**
Figure 10: Weight development between the free space winner unit and feature units in the attribute space (full connectivity) after the presentation of a second, similar example.

This experiment was done 50 times (i.e. a second instance was presented immediately after a recruitment of a new concept), and the acquisition of a new unit failed only once: the system tried to recruit an already committed unit.

7.4 Learning several dissimilar instances.

The input examples used in these experiments shared no features. The structure of the domain theory and the operationality criterion remained unchanged. An instance was presented to the system, a goal concept was chosen and the network was updated for 200 steps. After 200 steps all potentials and outputs were set to 0 to destroy activation patterns in concept, attribute, and free space. Next, a new example was presented, a new goal concept chosen, and the network was updated again for 200 steps. In total, this procedure was performed three times. After each presentation of an instance, the weight change of links between the free and the attribute space was recorded.
Figures 11 and 12 show that the weight development is very similar for all instances independent of the time of presentation. However, because of the small size of the free space in our simulations (only five triple-unit networks) the network tends to pick an already-committed unit for the representation of a concept after the presentation of the third instance. This indicates that the free space must have a minimum size, or that the recruitment of new units is affected by previous recruitment processes.

![Graph showing weight development over time for different concepts]

Figure 11: Weight development between the free space winner unit and feature units in the attribute space (full connectivity) after the presentation of a second, dissimilar example.

8. Multiple concepts and disjunctive attributes.

The example above describes a simulation with a single attribute space. In this case, obviously, several new concepts can be recruited simultaneously, but these must share the same attribute/value structure. If the goal is the recruitment of several new concepts with different attributes and values, two procedures are possible. First, a single attribute space is used and learning for each concept is done in dif-
ferent time intervals while all attributes and values shared by these concepts remain active. Second, more than one attribute space is used.

![Graph showing weight development]

Figure 12: Weight development between the free space winner unit and feature units in the attribute space (full connectivity) after the presentation of a third, dissimilar example.

In many cases, disjunctive combinations of attributes are of interest. In order to learn these combinations, a direct connection between attribute and concept space, as proposed, is not sufficient and thus "hidden units" are necessary. See Fanty (1988) for experiments on concept learning in structured connectionist systems and the use of sequential processing in massively parallel systems.

9. Identifying necessary features for concept formation.

The most serious shortcoming of our system is the inability to distinguish between features of the training example, which are necessary for the definition of the new concept, and those which are only relevant for the particular input instance presented to the system. In the example above, the features *yellow* and *owner:michael* are equally connected to the new concept, with respect to strength, although only one of them might be necessary for the definition of the new concept.
Before Recruitment Learning

Feature Units

Concept Units

Goal-Concept

Super-Concept

Free Units

After Recruitment Learning

Feature Units

Concept Units

Goal-Concept

Super-Concept

New Concept

Free Units
Figure 13 (previous page): Classification for and after recruitment. A set of features is the input in both cases. In the unmodified network before recruitment, the input feature units activate a generic concept, and additional feature units are getting on by stimulation of this concept unit or its super-concepts. After recruitment, the more specific new recruited concept is getting the strongest input from the feature units and then primes additional feature units directly.

There are several ways to overcome this shortcoming:

1. A similarity-based learning component is added to the system to realize a combined knowledge- and data-intensive system which also captures the statistics of the input environment.

2. Another possibility is to use more knowledge to decide how specific or general a feature is. This would require detailed and explicit definitions of features.

3. It is possible to make the method supervised by telling the system which features are necessary for concept formation. The feature yellow in our example above would be clamped on and the new recruited unit would have strong connections to this feature.

4. Introduce a "generality" measurement in the same way that certainty factors are used in evidential reasoning systems. Those features of the input example which are only owned by this particular instance would get small initial potentials and other features would get higher values.

10. Conclusion.

A structured connectionist learning system is introduced, which uses a built-in knowledge representation module for inferencing. On the implementation level, the system is an application of recruitment learning (Feldman 1982). The new and essential contribution of the approach presented here is the interaction between constraints on the "knowledge level" and the connectionist implementation level. On
the knowledge level the approach introduces the notion of "goal concepts", restrictions for the new concept definition, and the use of a single instance only. These are constraints for recruitment learning on the network level, and allow the connection of a new concept to an existing connectionist inheritance system.

The recruitment learning will make a connectionist inheritance system more operational. The system's ability to recognize instances as described by the training example will improve because of the new recruited unit and the new concept it represents. During recognition the presentation of another instance which belongs to the same class as the training example will cause the immediate activation of the newly recruited unit and therefore very efficient recognition (see Figure 13 for explanation). Hollbach (1988) has introduced the distinction between immediate and mediated inferences. An inference is immediate if a conclusion is made based on a feature which is directly connected to a concept. A mediated inference is made when several inference steps are necessary for a final decision. The machine learning system described above will make a knowledge representation more operational by allowing more immediate inferences instead of mediated conclusions. The newly learned concept has weighted connections to its features, and activating these features results in the direct activation of the newly acquired concept.

The presented method should be applicable for connectionist inheritance networks in general. The necessary element is the distinction in concept, attribute, and free space.

11. References.


