To appear in: The Proceedings of the Sixth Annual Conference of the Cognitive Science Society, Boulder Colorado, June 1984.

LEARNING SEMANTIC FEATURES

Geoffrey E. Hinton
Computer Science Department
Carnegie-Mellon University

and

Terrence J. Sejnowski
Biophysics Department
The Johns Hopkins University

May 1 1984

Introduction

An important idea within cognitive science is that much general knowledge can be represented as constraints between the slot-fillers of a schema. The central idea of "connectionism" is that knowledge is represented by the strengths of the connections in a large network of simple processing elements. The relation between these two ideas is complex.

Several ways of using connectionist networks to implement schemas have been proposed. The obvious, "localist" approach is to identify processing units in the physical network with concepts, and to treat the physical links as if they were direct implementations of the pointers that are conventionally used to represent the filling of a schema-slot by an object (Feldman and Ballard, 1982; Fahlman, 1979). An alternative, "distributed" approach is to allocate a large number of units to each slot of a schema, and to represent the filler of that slot by the pattern of activity of that set of units (Hinton, 1981). The main difference is in how the physical parallelism is used. In the distributed approach, only one instantiation of a particular schema is possible at a time because the units dedicated to each slot can only have one pattern of activity at a time. The physical links are used to implement constraints between slot-fillers. By setting the strengths of the links appropriately, it is possible to make a pattern of activity in one set of units cause (or prohibit) a pattern in another set of units. If each component of a pattern of activity is viewed as a semantic feature of the object represented by that pattern, the physical links between units allow many semantic constraints to be enforced in parallel.

A major difficulty for the distributed approach is this: Someone has to choose what pattern of activity to use to represent a particular slot-filler. If a random pattern is used, it may be hard to represent the constraints between slot-fillers because the underlying semantic features are not explicit. What is needed is an intelligent choice that makes it easy to implement the constraints. If, for example, all male fillers of slot 1 are represented by patterns that have unit 253 turned on, and all male fillers of slot 2 have unit 491 turned on, then the constraint that not both fillers can be male can be implemented by making these two units inhibit each other.

Unfortunately, it is hard to discover useful semantic features automatically. The definition of a useful feature is that it puts relatively strong constraints on the features of objects in other slots, but these other features also have to be learned and so there is a chicken-and-egg problem. This paper describes a way of learning sets of features that work well together. The learning algorithm uses some rather complicated ideas from statistical mechanics, and it runs very slowly on conventional computers, so the example given is very simple.

A very simple example

Imagine a world in which objects always occur in pairs, and only one pair occurs at a time. Each object can be paired with many but not all of the other objects. One way to characterize the structure of this world would be to simply list all the pairs and their probability of occurence. If the possible pairs were determined randomly, this method might be sensible, but if there are underlying properties of objects that influence the pairings, it will generally be much more efficient to express the probability distribution over the possible pairs by extracting these underlying features and using them to express laws of combination. Moreover, this second method will allow predictions: among the pairs that have never been observed, the ones which satisfy the laws of combination are more likely to occur than the ones which don't. The difficulty in using the second method is that the number of potential features is enormous, even if we restrict ourelves to clearcut binary features. Given n objects there are 2^n ways of picking a subset and hence 2^n potential binary features. Finding just those features which lead to good laws of combination is a formidable problem.

We use this simple example to illustrate a learning algorithm which can discover useful features. The two objects that occur together in a pair are like two slot-fillers. For each slot we have 9 units and 8 possible fillers. The different fillers are represented by turning on exactly one of the first 8 units in a slot, but we do not decide in advance whether or not the 9^{th} unit should be on. It is left to the learning algorithm to decide how to use the 9^{th} unit. The learning algorithm is therefore capable of modifying the representations that are used for the various slot fillers.

Simulations

Figure 1 shows some examples of pairs of objects drawn from a probability distribution over all possible pairs composed of one object from the set {A, B, ... H} and one from the set {S, T, ... Z}. Implicit within this probability distribution is a strong underlying regularity: If the first set is divided into the subsets {A B C D} and {E F G H} and the second set is divided into the subsets {S T U V} and {W X Y Z}, then there is a simple way of expressing the probability distribution: If the first object is in the set {A B C D} the other will be in the set {S T U V} with probability 0.9, and if the first object is in the set {E F G H} the second will be in the set {W X Y Z} with probability 0.9.

Figure 2 shows a network which has been exposed to the probability distribution by clamping the states of some of its units. States that represent a particular pair of objects are clamped with the appropriate probability. The network started with all its connection strengths equal to zero, and after being shown 5000 pairs of objects it has captured the regularity by setting its weights so that one of its two central units detects whether the first object is in the set {A B C D}, the other central unit detects whether the second object is in the set {W X Y Z}, and the two units inhibit each other.

DT HX HZ DV CS CT EZ BT EY CU BV
EY AS CT FT BT FW GY FX GY GW BX
CV DS AU CT HY BS AT GZ AS FY EZ
HX HX CT DZ AU CV CS FX DU AY EZ ...

Figure 1: A collection of pairs of objects. These pairs were drawn from a probability distribution that can be described relatively simply (see text). The problem is to discover ways of dividing the objects into sets that allow the simple description to be expressed.

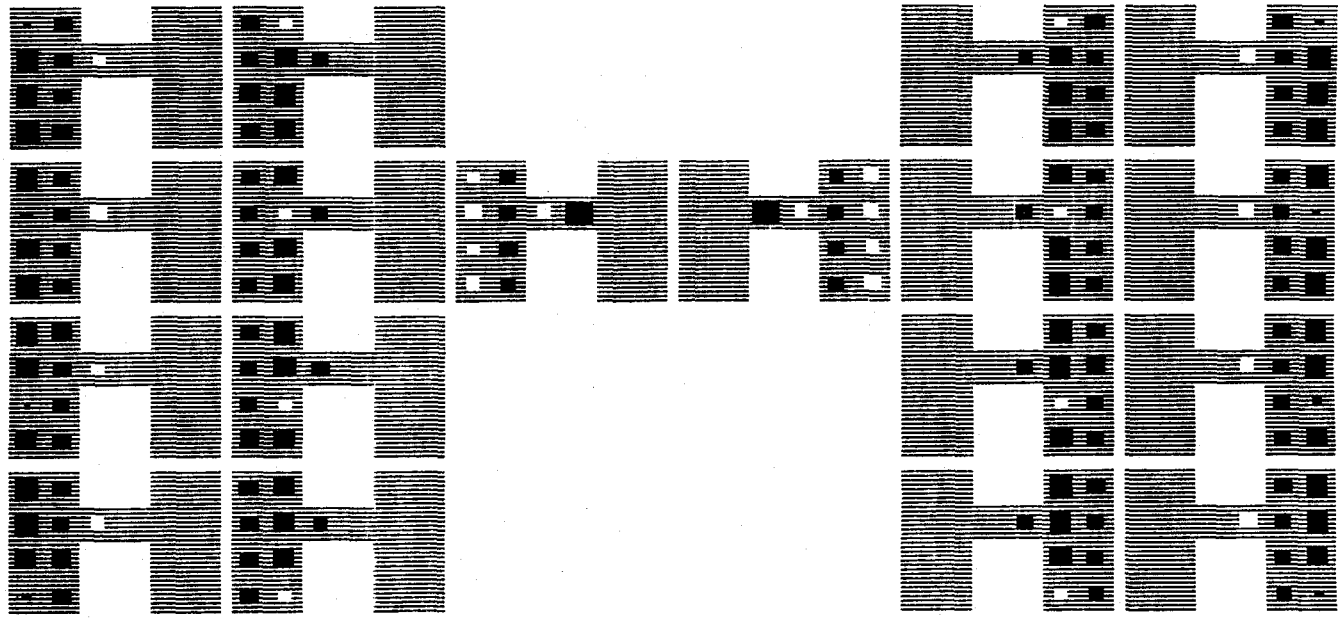

Figure 2: Each unit is represented by a gray "H" shaped region. Within this region, connections to other units are represented by white (positive weight) or black (negative weight) rectangles in the position that corresponds to the location of the other unit in the overall diagram. The size of the white or black box indicates the absolute magnitude of the weight. For example, the white rectangle in the top left hand unit represents an excitatory connection between that unit and the leftmost of the two central units. All connections between units appear twice in the diagram, once in the box for each of the two units being connected. So the white rectangle in the top left-hand corner of the leftmost central unit is the same connection as described above. Units never connect to themselves, so in the position where that connection would be displayed (e. g. the top left-hand corner of the top left-hand unit) we display the threshold using black to mean a positive threshold. The empty gray areas on the right-hand sides of the left-hand group of 8 units show that these units are not directly connected to the right-hand group of 8. Notice that the units in each group of 8 have learned to inhibit each other. This implements the within-slot constraint that only one of them should be on at a time. This constraint follows from our decision to represent each slot filler by a pattern of activity with only one of the 8 units turned on.

It is hard to learn such features because there is no information to suggest them in the fillers of either slot considered separately. All 8 fillers occur equally often and have no intrinsic similarity to each other. The *only* reason for selecting these particular features is that they allow the implicit constraint between slot-fillers to be expressed.

Finding combinations of slot fillers that satisfy existing constraints

Before describing the learning algorithm it is necessary to describe how a network with a fixed set of weights can arrive at combinations of slot-fillers that satisfy the constraints which are implemented by the weights. An arbitrary combination of slot-fillers will generally violate some constraints, and the process of finding a good combination involves an iterative search in which individual units change their states so as to minimize the violation. The stochastic iterative search procedure we use was first described by Hinton & Sejnowski (1983) and is described more briefly here.

We start by showing that networks of asynchronous, symmetrically connected, binary threshold elements obey an energy function, and that repeated iterations are guaranteed to find an energy minimum (Hopfield, 1982). This minimum corresponds to a combination of slot fillers that minimizes the constraint violation. The global potential energy of the system is defined as

$$E = -\sum_{i<j} w_{ij} s_i s_j + \sum_i \theta_i s_i \qquad (1)$$

where w_{ij} is the strength of connection (synaptic weight) from the j^{th} to the i^{th} unit, s_i is a boolean truth value (0 or 1), and θ_i is a threshold.

A simple algorithm for finding a combination of truth values that is a *local* minimum is to switch each hypothesis into whichever of its two states yields the lower total energy given the current states of the other hypotheses. If hardware units make their decisions asynchronously, and if transmission times are negligible, then the system always settles into a local energy minimum. Because the connections are symmetrical, the difference between the energy of the whole system with the k^{th} hypothesis false and its energy with the k^{th} hypothesis true can be determined locally by the k^{th} unit, and is just

$$\Delta E_k = \sum_i w_{ki} s_i \quad - \theta_k \qquad (2)$$

Therefore, the rule for minimizing the energy contributed by a unit is to adopt the true state if its total input from the other units and from outside the system exceeds its threshold. This is the familiar rule for binary threshold units.

Using probabilistic decisions to escape from local minima

The deterministic algorithm suffers from the standard weakness of gradient descent methods: It gets stuck at *local* minima that are not globally optimal. This is an inevitable consequence of only allowing jumps to states of lower energy. If, however, jumps to higher energy states occasionally occur, it is possible to break out of local minima. An algorithm with this property has recently been applied to difficult constraint satisfaction problems by Kirkpatrick, Gelatt & Vecchi (1983). We adopt a form that is suitable for parallel computation: If the energy gap between the true and false states of the k^{th} unit is ΔE_k then regardless of the previous state set $s_k = 1$ with probability

$$p_k = \frac{1}{(1 + e^{-\Delta E_k / T})} \tag{3}$$

where T is a parameter which acts like the temperature of a physical system. This parallel algorithm ensures that in thermal equilibrium the relative probability of two global states is determined solely by their energy difference, and follows a Boltzmann distribution.

$$\frac{P_\alpha}{P_\beta} = e^{-(E_\alpha - E_\beta)/T} \tag{4}$$

where P_α is the probability of being in the α^{th} global state, and E_α is the energy of that state.

At low temperatures there is a strong bias in favor of states with low energy, but the time required to reach equilibrium may be long. At higher temperatures the bias is not so favorable but equilibrium is reached faster. The fastest way to reach equilibrium at a given temperature is to start with a higher temperature and gradually reduce it.

The learning algorithm

When a network is allowed to reach thermal equilibrium using the probabilistic decision rule in Eq. 3, the probability of finding it in any particular global state depends on the energy of that state (Eq 4.). These equations allow us to derive the way in which the probability of a state changes as a weight is changed:

$$\frac{\partial \ln P_\alpha}{\partial w_{ij}} = \frac{1}{T} \left[s_i^\alpha s_j^\alpha - \sum_\beta P_\beta s_i^\beta s_j^\beta \right] \tag{5}$$

where α is a global state of the network and s_i^α is the binary state of the i^{th} unit in the α^{th} global state. Eq. 5 shows that the effect of a weight on the log probability of a global state can be computed from purely local information, because it only involves the behavior of the two units that the weight connects (the second term is just the probability of finding the i^{th} and j^{th} units on together). This makes it easy to manipulate the probabilities of global states provided the desired probabilities are known (see Hinton & Sejnowski, 1983 for

details).

Unfortunately, it is normally unreasonable to expect the environment or a teacher to specify the required probabilities of entire global states of the network. A network typically contains some "visible" units that receive the input or produce the output and it also contains some other units that we call "hidden" because they are not directly involved in representing the input or output. For example, the two central units in figure 2 are hidden units and the rest are visible. The task that the network must perform is defined in terms of the states of the visible units, and so the environment or teacher only has direct access to the states of these units (hence the name visible). The difficult learning problem is to decide how to use the hidden units to help achieve the required behavior of the visible units. A learning rule which assumes that the network is told from outside how to use *all* of its units is of limited interest because it evades the main problem which is to discover appropriate representations for a given task among the hidden units.

In statistical terms, the hidden units can be used to represent the higher-order statistical regularities that are implicit in the ensemble of vectors that the environment causes in the visible units. The learning problem is to decide how best to use the capacity of the weights to capture this higher-order statistical structure. In common-sense terms, the weights should be chosen so that the hidden units represent significant semantic features and the interactions among hidden units capture the important constraints. If we make certain assumptions it is possible to derive a measure of how effectively the weights are being used, and it is also possible to show how the weights should be changed to progressively improve this measure.

We assume that the environment "clamps" a particular vector over the visible units and it keeps it there for long enough for the network to reach thermal equilibrium with this vector as a boundary condition (i.e. to "interpret" it). We also assume (unrealistically) that the there is no structure in the sequential order of the environmentally clamped vectors. This means that the complete structure of the ensemble of environmental vectors can be specified by giving the probability, $P(V_\alpha)$, of each of the 2^v vectors over the v visible units. Notice that the $P(V_\alpha)$ do not depend on the weights in the network because the environment clamps the visible units.

A particular set of weights can be said to constitute a perfect model of the structure of the environment if it leads to exactly the same probability distribution of visible vectors when the network is running freely *with no environmental input.* Because of the stochastic behavior of the units, the network will wander through a variety of states even with no input and it will therefore generate a probability distribution, $P'(V_\alpha)$, over all 2^v visible vectors. This distribution can be compared with the environmental distribution, $P(V_\alpha)$. In general, it will not be possible to exactly match the 2^v environmental probabilities using the weights among the v visible and h hidden units because there are at most $0.5\ (v+h-1)\ (v+h)$ symmetrical weights and $(v+h)$ thresholds.

However, it may be possible to do very well if the environment contains regularities that can be expressed in the weights. An information theoretic measure (Kullback, 1959) of the distance between the environmental and free-running probability distributions is given by:

$$G = \sum_{\alpha} P(V_\alpha) \ln \frac{P(V_\alpha)}{P'(V_\alpha)} \tag{6}$$

where $P(V_\alpha)$ is the probability of the α^{th} state of the visible units when their states are determined by the environment, and $P'(V_\alpha)$ is the corresponding probability when the network is running freely with no environmental input.

G is never negative and is only zero if the distributions are identical. It is possible to improve the network's model of the structure of its environment by changing the weights so as to reduce G. It can be shown that:

$$\frac{\partial G}{\partial w_{ij}} = -\frac{1}{T} [p_{ij} - p'_{ij}] \tag{7}$$

where p_{ij} is the probability, averaged over all environmental inputs and measured at equilibrium, that the i^{th} and j^{th} units are both on when the network is being driven by the environment, and p'_{ij} is the corresponding probability when the network is free running.

One surprising feature of Eq. 7 is that it does not matter whether the weight is between two visible units, two hidden units, or one of each. The same rule applies for the gradient of G. An even more surprising fact is that the gradient involves only locally available information, even though G is a global property of the whole set of weights and the effect of one weight on G therefore depends on the current values of all the other weights. Fortunately, the other weights affect p_{ij} and p'_{ij} in just the right way to make the dependence locally available.

Parameters for the learning algorithm

The ability to discover the partial derivative of G by observing p_{ij} and p'_{ij} does not completely determine the learning algorithm. It is still necessary to decide how much to change each weight, how long to collect co-occurence statistics before changing the weight, how many weights to change at a time, and what temperature schedule to use during the annealing searches. Reasonable values for these parameters were found by trial and error. Further discussion of the effects of these parameters can be found in Hinton, Sejnowski & Ackley (1984). A "sweep" consisted of annealing 16 times with environmentally determined vectors clamped on the visible units, and 16 times with no clamping. After each sweep, each of the weights was updated with a probability of 0.5. This partially asynchronous updating helps avoid oscillations in the weights. When a weight was updated, it was always increased or decreased by the same fixed amount. The sign of the increment was determined by the sign of $p_{ij} - p'_{ij}$. The magnitude of the weight-step was 0.2.

The annealing schedule started by randomizing the state and then ran for the following times at the following temperatures: 2@2.0, 2@1.5, 2@1.2, 2@1.0, where one unit of time means running the network for long enough so that the expected number of times each unit is picked is 1. After this annealing, the network was assumed to be at equilibrium at a temperature of 1.0, and was run for a further time of 10 while co-occurence statistics were collected.

Conclusion

One of the major problems with using distributed patterns of activity as representations is to choose the patterns. Some choices work much better than others because they make important underlying features explicit and thus they allow the physical links in the network to capture the constraints that characterize the domain. We have presented a learning algorithm for choosing representations, and shown that it can create semantic features that are useful for expressing the constraints between the fillers of two slots.

Acknowledgements

The research reported here was supported by grants from the System Development Foundation. We thank Dave Ackley, Mark Derthick, Scott Fahlman, Jay McClelland, Dave Rumelhart, and Paul Smolensky for helpful discussions.

References

Fahlman, S. E., *NETL: A system for representing and using real world knowledge.* Cambridge, Mass.: MIT Press, 1979.

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

Hinton, G. E. Implementing semantic networks in parallel hardware. In G. E. Hinton & J. A. Anderson (Eds.) *Parallel Models of Associative Memory.* Hillsdale, NJ: Lawrence Erlbaum Associates, 1981, 161-187.

Hinton, G.E., & Sejnowski, T.J. Analyzing cooperative computation. *Proceedings of the Fifth Annual Conference of the Cognitive Science Society.* Rochester, NY, May 1983.

Hinton, G.E., Sejnowski, T.J., & Ackley, D. H. Boltzmann Machines: Constraint satisfaction networks that learn. Technical report CMU-CS-84-119, Carnegie-Mellon University, May 1984.

Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences USA*, 1982, *79* pp 2554-2558.

Kirkpatrick, S. Gelatt, C. D. & Vecchi, M. P. Optimization by simulated annealing. *Science*, **220**, 671-680.

Kullback, S. *Information Theory and Statistics.* New York: Wiley, 1959.