Three Constructive Algorithms For Network Learning

Stephen I. Gallant

College of Computer Science
Northeastern University

ABSTRACT: *Machine learning methods for connectionist models usually operate by attaching weights to a prespecified network so that a certain functionality is achieved. This is the classical credit assignment problem.*

This paper explores a constructive approach to connectionist learning where both a network and weights must be generated. It is argued that this is an easier problem to solve and is sufficient for many applications since network topology is usually not as important as functionality.

Three algorithms are presented for constructing networks from training examples. As cells are added and iterations are made, each method produces a network having optimal expected behavior (i.e. it correctly classifies the maximum number of training examples possible) with arbitrarily high probability $p < 1$.

Learning speed for these algorithms is currently being investigated.

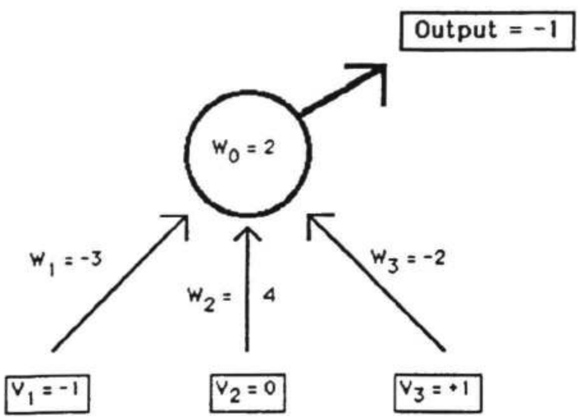
Keywords: Learning, Connectionist Models, Pocket Algorithm

Acknowledgment: This research was partially supported by a grant from the Northeastern University Research and Scholarship Development Fund. Thanks to Mark Frydenberg, Mitch Wand, Bob Futrelle and Ken Baclawski for helpful comments.

I. Introduction

A common single cell connectionist model is the linear discriminant [Fisher 1936, Duda & Hart 1973] or perceptron [Rosenblatt 1961]. A linear discriminant consists of a set of numerical weights W_i. An input vector V with components or "features" $< V_1, ..., V_n >$ is classified into one of two categories (True or False) according to whether the weighted sum of its components is greater than some threshold W_0 (see Figures 1, 2). Linear discriminants and perceptrons have been extensively studied and critiqued [Minsky & Papert 1969].

Linear discriminants are quite popular for practical applications such as machine vision, pattern recognition, and



$$\text{Output} = \begin{Bmatrix} +1 \\ -1 \\ 0 \end{Bmatrix} \text{ if } \sum V_i W_i \begin{Bmatrix} > \\ < \\ = \end{Bmatrix} W_0$$

Figure 1: Linear Discriminant

Output of top cell is -1 since
$(-1)(-3) + (0)(4) + (+1)(-2) < 2$

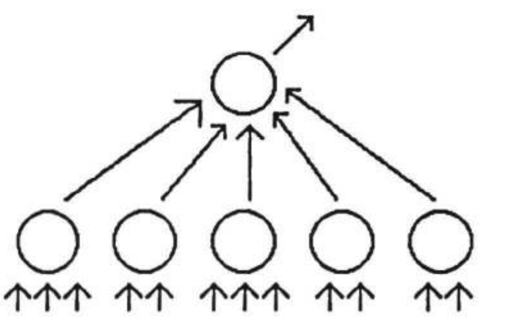


Figure 2: Linear Discriminant Network

classification problems. There are well known methods for determining good, but sub-optimal, linear discriminant weights. Recently we have developed a new method, the Pocket Algorithm, which gives optimal weights with arbitrarily high probability and appears to decrease misclassifications by roughly one-fifth over standard methods [Gallant 1985b]. The Pocket Algorithm is also important because it serves as the basis for the network growing algorithms to be discussed.

While an individual discriminant has several appealing features, it can only represent a small subset of boolean functions called separable functions (see [Gallant 1985b]). However a *network* of linear discriminants (Figure 2) can represent any boolean function on n boolean variables.

The problem of determining weights for a predetermined network of cells is much harder than for a single cell. This is the classical credit assignment problem identified by Minsky [1961]. Recent progress on this problem has been made by Barto [1981, 1985], C. Anderson [1982], Pearl [1985a, 1985b], Sutton [1984], Hinton [1984a], and Rumelhart et. al. [1985]. Of these, Rumelhart's *Back Propagation Algorithm* appears most promising with respect to speed.

II. The Constructive Approach

Rather than assuming a fixed network and trying to determine appropriate weights for it, one of our approaches has been to determine both network and weights in order to achieve the desired functionality. There are several motivations for this *constructive approach.*

First, the constructive algorithms to be studied always produce (in theory) networks and weights which correctly classify a maximum number of training examples. (If there is no pair of contradictory examples, all examples will be correctly classified.) The constructive problem seems easier to solve than the fixed problem because the necessity of realizing a solution in a fixed network is an additional constraint on the problem. Thus the requirement of defining a network is not an added burden, it is an extra degree of freedom.

Second, for most applications the topology of the final network is not vitally important. Usually it is more important to have a network and weights implementing some desired functionality than it is to have conformity with some prespecified structure. For our work with connectionist expert systems [Gallant 1985a], functionality is much more important than network topology.

Another reason to examine constructive approaches is speed of learning. The Boltzmann learning algorithm [Hinton 1984a] has largely been abandoned because of its slowness (although work with the Boltzmann model continues to be fruitful [Touretzky & Hinton 1985, Cottrell 1985]). Rumelhart's Back Propagation Algorithm appears more promising with respect to speed. Nevertheless the prospect of a fast learning method less susceptible to convergence problems gives strong impetus for exploring constructive network algorithms.

The above arguments motivate the development of constructive algorithms and comparisons with Back Propagation where appropriate.

III. Algorithms for Generating Weights for a Single Discriminant

Generating weights for a single linear discriminant so that a maximum number of training examples are correctly classified is a classical problem reviewed in [Gallant 1985b]. To quickly summarize the situation:

1. *Perceptron learning [Rosenblatt 1961, Minsky & Papert 1969] works for separable problems, but fails for nonseparable problems.*
2. *Other standard methods solve similar but different problems. Usually they produce sub-optimal weights for non-separable problems and frequently gives sub-optimal weights for separable problems as well [Duda & Hart 1973, Kaleca 1980].*
3. *The Pocket Algorithm [Gallant 1985c, 1985b] solves separable problems in finite time and gives weights for non-separable problems with an expected quality that improves as the number of iterations increases. The probability that the generated weights will be optimal approaches 1 as the number of iterations grows. Initial comparison with a standard method (Wilks' method) gave a rough estimate of a 20% decrease in misclassifications using the Pocket Algorithm.*

The algorithms for generating single discriminants are important in their own right due to their heavy use in applications such as machine vision and pattern recognition. They are also important because they are part of algorithms that generate networks.

IV. Constructive Algorithms For Network Learning

We shall briefly describe three constructive algorithms for network generation: the Tower Construction, the Inverted Pyramid Construction, and the Distributed Method. It should be noted that there are a number of additional variations to these methods which may have important effects on performance.

1. The Tower Construction

This method constructs a linear discriminant network in the form of a tower (Figure 3). Cells are added one by one

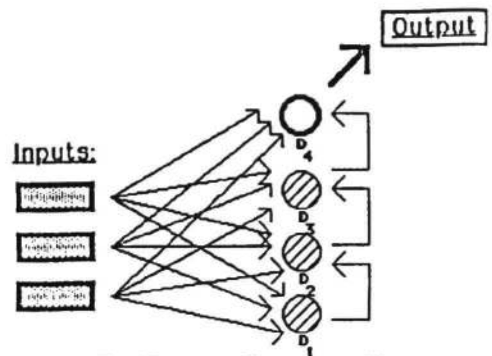


Figure 3: Tower Construction

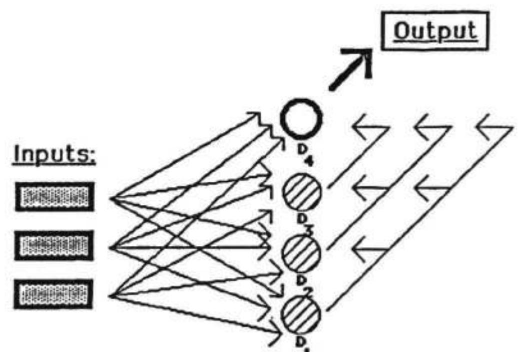
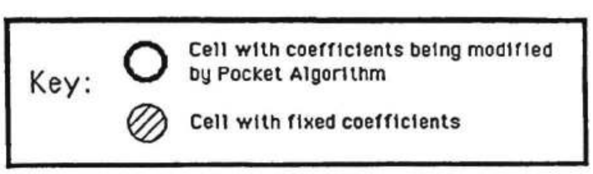


Figure 4: Inverted Pyramid Construction



and connected to the original inputs and to the previous cell as in Algorithm 1.

It can be shown that with arbitrarily high probability $p < 1$, the Tower Construction will produce a network that correctly classifies a maximum number of training examples (see [Gallant1986]). Furthermore, the output from each level will correctly classify a greater number of training examples than the output from the previous level. This guarantees theoretical convergence to a network with optimal performance for the set of training examples, as contrasted with other approaches (such as gradient descent methods) which may fail to find an optimal network.

---

1. *Construct a good (preferably optimal) set of coefficients for a single linear discriminant, D_1. If all (or the maximum number possible) of the training examples are correctly classified, we are done. Otherwise freeze the coefficients for D_1.*
2. *Construct a good (preferably optimal) set of coefficients for linear discriminant D_{n+1} having inputs from the training examples and from discriminant D_n (see Figure 3). It can be shown that D_{n+1} can correctly classify a greater number of training examples than D_n. Freeze the coefficients for D_{n+1}.*
3. *Repeat step 2 a finite number of times until all (or the maximum number possible) of training examples are correctly classified.*

Algorithm 1: *Tower Construction*

While only preliminary tests have been made with this algorithm, one of them is particularly interesting.

The Parity function is a classical non-separable function which will return true (+1) if an odd number of its inputs are +1. Otherwise the function returns false (-1). For 2 inputs the Parity function is the same as exclusive-OR. It is well known that a network of 2 cells is necessary and sufficient to represent exclusive-OR (see for example [Hinton 1984a]). We applied the Tower Construction to the Parity problem for n=3 with the expectation that a 3 level network would be

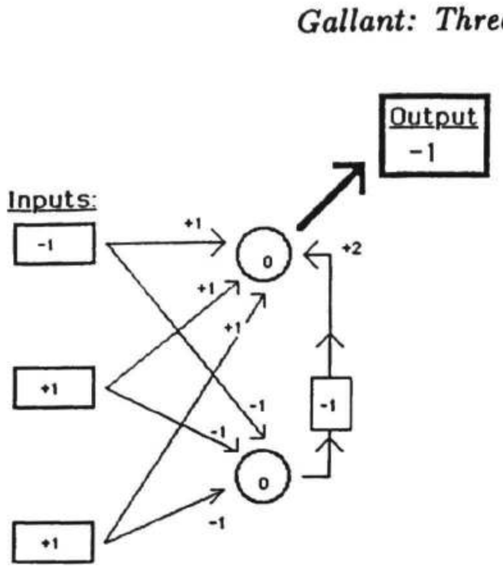


Figure 5: Tower Construction for computing parity function with three inputs.

Output is +1 if an odd number of inputs are +1.
Output is -1 if an even number of inputs are +1.
Note that all inputs and outputs are +1 and -1 (rather than +1 and 0).

Example shows an output of -1 when two inputs are +1. The output from the lower cell is also -1 for this example.

required to represent this function. To our surprise, the Tower Construction produced a 2 level network! The network is given in Figure 5. From this construction it became clear that a tower could compute the Parity function on n inputs by an

$$\lceil (n+1)/2 \rceil$$

level tower.

2. The Inverted Pyramid Construction

The Inverted Pyramid Construction (Figure 4) is similar to the Tower Construction, except each new discriminant sees outputs from *all* previous levels, not just the immediately preceding level. One would expect that fewer levels would be needed and this would speed up network generation. However learning speed may be adversely affected, since each level must solve a problem in a space of higher dimensions than is the case with the Tower Construction. This is currently under investigation.

3. The Distributed Method

The Distributed Method is a constructive approach which may also be viewed as a fixed network approach. The basic idea of the Distributed Method is illustrated in Figure 6. A layer of many discriminants with fixed, random coefficients is specified, after which learning takes place for the single cell on top. If enough cells have been added then any function will become separable [Minsky & Papert 1969], but families of some functions may require an exponential number of added cells for a perfect representation. On the other hand our initial testing has indicated that n training examples can usually be correctly classified by a random layer with $\frac{1}{2}n$ to $2n$ added cells. The added cells produce a distributed representation for any input [Hinton 1984b, Touretzky & Hinton 1985, Bloom 1970] since the *pattern* of cells in the added layer is more important than any single cell.

An important point here is that random linear discriminants make up the added layer rather than random boolean functions. This is intended to promote robustness of the resulting system, since an input V and a close training example V^* should both produce similar patterns in the added layer. This in turn would make it likely that the top cell would give identical outputs for V and V^*. However if random functions had been used for the added layer, V and V^* would produce totally uncorrelated outputs regardless of their similarity (unless V and V^* were identical). The price to be paid for this attempt at robustness is that the added layer will require more cells to achieve separability than if strictly random boolean functions had been employed.

A major advantage of the Distributed Method is that *the same added layer can be used equally well by different top cells for computing totally different functions.* Once we invest in an added layer, it is available for every future task. Such an architecture is well suited for cells which are independent and asynchronous and it requires fewer coefficients to be learned than most other models. This structure is an old but appealing idea motivated by neural systems and the difficulty of passing large amounts of information genetically. Since we now have at least one reasonable algorithm which works with non-separable data (the Pocket Algorithm) and the ability to enhance functionality by employing the Tower or Inverted Pyramid Constructions above the added layer of fixed random cells, this idea is worth exploring again.

Space limitations preclude a more thorough examination of the Distributed Method here. The reader is referred to [Gallant 1986] for additional details.

V. Discussion and Summary

We have presented three constructive algorithms for generating connectionist networks from training examples. The Tower and Inverted Pyramid Constructions give optimal performance, after a finite number of levels have been added and after sufficient iterations, with arbitrarily high probability. Each level performs better than the previous level in terms of correct classifications.

The Distributed Method also gives optimal performance for a set of training examples with arbitrarily high probability, provided there are sufficient cells in the added layer.

All three methods are robust in the sense that an example which is sufficiently similar to a training example will be classified the same as that training example, due to the general behavior of linear discriminants.

The performance of these algorithms is just beginning to be investigated. However the Tower Construction has been incorporated into the network generation phase for the MACIE system for generating expert systems [Gallant 1985a] and has demonstrated reasonable speed and performance for several problems. More testing and practical experience is needed to fully evaluate these algorithms.

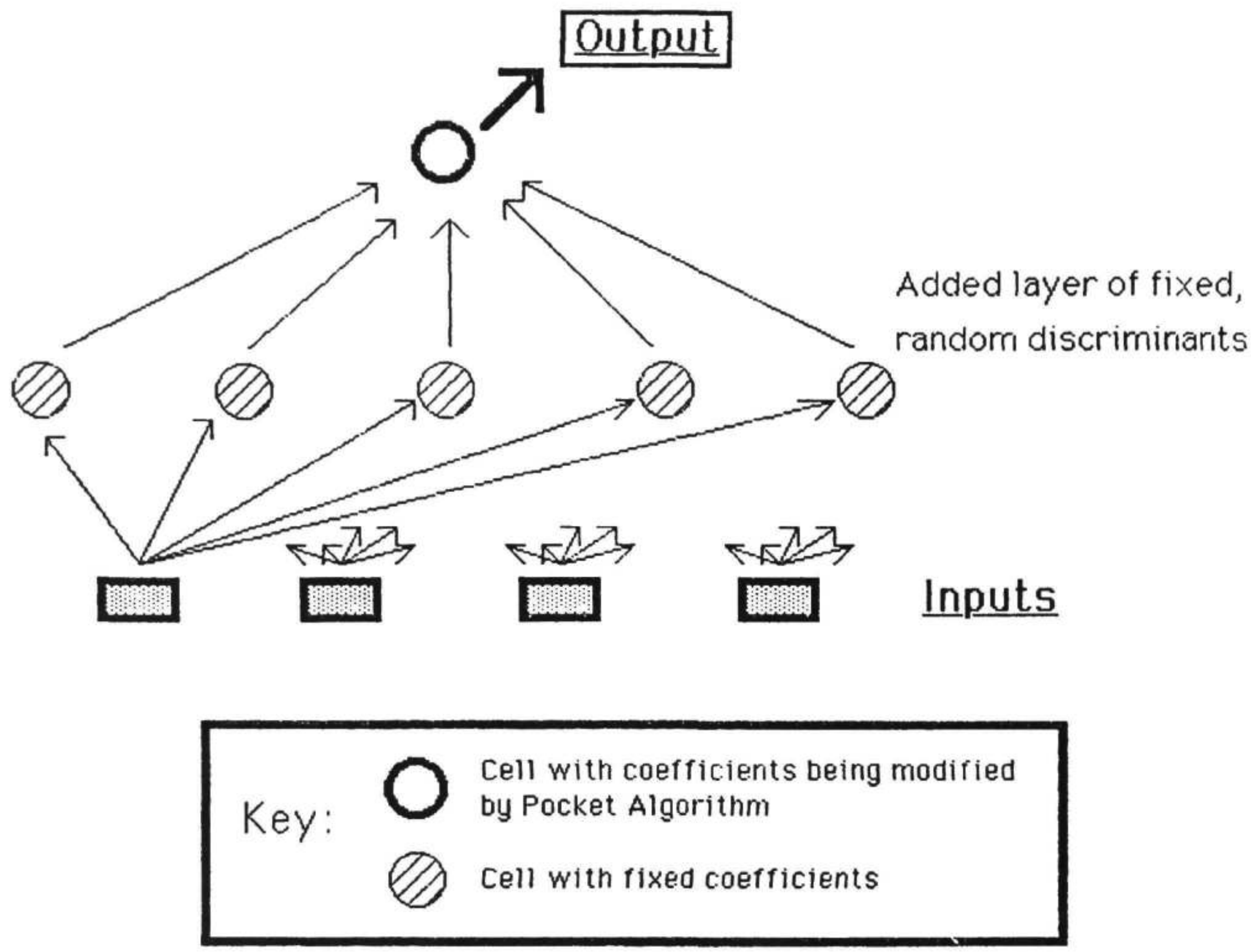


Figure 6: Distributed Method

REFERENCES

[Anderson 1982]
Anderson, C. W. Feature Generation and Selection by a Layered Network of Reinforcement Learning Elements: Some Initial Experiments. Computer and Information Science Department Technical Report 82-12, University of Massachusetts, Amherst, Massachusetts

[Barto 1981]
Barto, A. G., Sutton, R. S., & Brouwer, P. S. Associative Search Network: A Reinforcement Learning Associative Memory. Biol. Cybern. 40, 201-211 (1981)

[Barto 1985]
Barto, A. F. Learning by Statistical Cooperation of Self- Interested Neuron-like Computing Elements. Human Neurophysiology, to appear.

[Bloom 1970]
Bloom, B. H. Space/Time Trade-offs in Hash Coding with Allowable Errors. CACM 13, No. 7, July 1970, 422-426.

[Cottrell 1985]
Cottrell, G. W. Connectionist Parsing. Seventh Annual Conference of the Cognitive Science Society, Irvine, Ca. 1985.

[Duda & Hart 1973]
Duda, R. O. & Hart, P. E. Pattern Classification and Scene Analysis. (1973) John Wiley & Sons, New York.

[Fisher 1936]
Fisher, R. A. The use of multiple measurements in taxonomic problems. Ann. Eugenics, 7, Part II, 179-188. Also in *Contributions to Mathematical Statistics* (1950) John Wiley, New York.

[Gallant 1985a]
Gallant, S. I. Automatic Generation of Expert Systems From Examples. Proceedings of Second International Conference on Artificial Intelligence Applications, sponsored by IEEE Computer Society, Miami Beach, Florida, Dec. 11-13, 1985.

[Gallant 1985b]
Gallant, S. I. Optimal Linear Discriminants. Technical Report SG-85-30, Northeastern University College of Computer Science. (to appear, Eighth International Conference on Pattern Recognition, Paris, France, Oct. 28-31, 1986)

[Gallant 1985c]
Gallant, S. I. The Pocket Algorithm for Perceptron Learning. Technical Report SG-85-19, Northeastern University College of Computer Science.

[Gallant 1986]
Gallant, S. I. Three constructive Algorithms for Network Learning. Technical Report SG-86-37, Northeastern University College of Computer Science. (note: the previous working title for this paper was "Credit Assignment, Intermediate States, and the Power of Towers.")

[Hinton 1984a]
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 Department of Computer Science

[Hinton 1984b]
Hinton, G. E. Distributed Representations. Technical Report CMU-CS-84-157, Carnegie-Mellon University, Department of Computer Science

[Minsky 1961]
Minsky, M. Steps Toward Artificial Intelligence. Proceedings IRE, 1961, 49, 8-30. Reprinted in Feigenbaum, E. A. & Feldman, J., Eds., *Computers and Thought* McGraw- Hill, New York, 1963.

[Minsky & Papert 1969]
Minsky, M. & Papert, S. *Perceptrons: An Introduction to Computational Geometry* (1969) MIT Press, Cambridge, Ma.

[Nilsson 1965]
Nilsson, N. J. *Learning Machines* (1965) McGraw-Hill, New York, NY.

[Pearl 1985a]
Pearl, J. Fusion, Propagation, and Structuring in Bayesian Networks. UCLA Computer Science Department Technical Report CSD-850022, R-42, June 1985.

[Pearl 1985b]
Pearl, J. How to Do with Probabilities What People Say You Can't. Proceedings of Second International Conference on Artificial Intelligence Applications, sponsored by IEEE Computer Society, Miami Beach, Florida, Dec. 11-13, 1985.

[Rosenblatt 1961]
Rosenblatt, F. *Principles or neurodynamics: Perceptrons and the theory of brain mechanisms* Spartan Press, Washington, DC.

[Rumelhart 1985]
Rumelhart, D. E., Hinton, G. E., & Williams, R. J. Learning Internal Representations by Error Propagation. ICS Report 8506, Institute for Cognitive Science, University of California, San Diego, Sept. 1985.

[Sutton 1984]
Sutton, R. S. Temporal Credit Assignment in Reinforcement Learning. Computer and Information Science Technical Report 84-02, University of Massachusetts, Amherst, Ma. (1984)

[Touretzky & Hinton 1985]
Touretzky, D. & Hinton, G. Symbols Among the Neurons: Details of a Connectionist Inference Architecture. Ninth International Joint Conference on Artificial Intelligence, Los Angeles, Ca. (IJCAI 85)