THE INTRODUCTION OF HIDDEN UNITS TO OPTIMIZING NEURAL NETWORKS

Benjamin J. Hellstrom
Laveen N. Kanal

Machine Intelligence and Pattern Analysis Laboratory
Department of Computer Science
University of Maryland, College Park, Maryland, 20742, U.S.A.

Abstract

Since their resurgence in the early 1980s, classical Hopfield-type neural networks have been applied to travelling salesman, graph and set partitioning, vertex cover, maximum clique, and numerous other combinatorial optimization problems. These structurally simple problems share the common characteristics that they can all be reduced to the minimization of 0-1 Hamilton energy functions and result in networks composed of functionally homogeneous *visible units*. For certain difficult optimization problems it can be shown that *hidden units* are required. We present a systematic approach to deriving neural network algorithms that promotes the introduction of hidden units and defines their *best features*. To illustrate the method, we derive a neural network for the *NP*-hard integer knapsack-packing optimization problem.

Introduction

In 1985, Hopfield and Tank [4] embedded the travelling salesman optimization problem in a hardware model of a neural network. The approach to embedding optimization problems in neural networks that was proposed by Hopfield and Tank has since become the defacto standard and has been applied to numerous other combinatorial optimization problems [1,5]. A common characteristic of these problems is that they are structurally simple and can easily be reduced to 0-1 Hamiltonian energy minimization problems. Furthermore, the resulting neural networks consist of functionally homogeneous processing units whose activation values are directly mapped to the solutions of their respective embedded optimization problems. Since these units participate in the expression of problem solutions for external interpretation, they can be viewed as *visible units*. Let us briefly review Hopfield and Tank's reduction technique. We define $\mathbf{V}$ to be the variable space of a specific optimization problem with objective function $f:\mathbf{V}\rightarrow\Re$. Let $\mathbf{S}$ be the space defined by the volume of an n-dimensional unit hypercube for some, as of yet unspecified n. Each point in $\mathbf{V}$, or configuration of problem variables, is associated with a configuration of the states of n neuron-like units, that is, a point in $\mathbf{S}$. This association is defined by a pair of mappings $M:\mathbf{V}\rightarrow\mathbf{S}$, and $M^{-1}:\mathbf{S}\rightarrow\mathbf{V}$. We must then select an appropriate *energy function* $E:\mathbf{S}\rightarrow\Re$. Hopfield and Tank showed that in order to embed the problem in a network of neuron-like processing units, the energy function must be expressible as a *0-1 Hamiltonian*. Hamiltonian energy functions have the form:

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} s_i s_j w_{ij} + \sum_{i=1}^{n} s_i \theta_i$$

where n can be interpreted as the number of processing units in a network, s_i is the activation level of the i^{th} unit, w_{ij} is the connection strength between the i^{th} and j^{th} units, and θ_i is the activation threshold of the i^{th} unit. All connections are symmetric, i.e., $w_{ij}=w_{ji}$ for all i, j. The fundamental obstacles to embedding arbitrary optimization problems in neural networks are the discovery of a representation (M and M^{-1}) and a Hamiltonian energy function so that minimal-energy network configurations are mapped to optimal configurations of problem variables. It should be noted that $\mathbf{V}$ is a discrete space while $\mathbf{S}$ is continuous. The set of minima of a continuous 0-1 Hamiltonian must be a subset of the set of corners of the n-dimensional unit hypercube. There are few restrictions on M except those imposed by the definitions of its domain and range. If there is *no* prior knowledge about the minima of f, then care must be taken to ensure that *all* points in $\mathbf{V}$ are mapped by M to corners of the hypercube. Otherwise, we can not be certain that the minima of f can be mapped by M^{-1} to minima of E. Hopfield and Tank's method consists of the following steps:

1 A network representation is selected. n, the dimensionality of the network state space is defined. A transformation, $M:\mathbf{V}\rightarrow\mathbf{S}$, from $\mathbf{V}$, the space of problem variable configurations, to $\mathbf{S}=[0,1]^n$, the volume of an n-dimensional unit hypercube, is defined.

2 An inverse transformation, $M^{-1}:\mathbf{S}\rightarrow\mathbf{V}$, is defined. Each network configuration in the volume of the n-dimensional hypercube is mapped by M^{-1} to a configuration of the variables of the problem space.

CH2766 – 4/89/0000 - 0127 © 1989 IEEE

3 An energy function, $E: \mathbf{S} \rightarrow \Re$, is defined on $\mathbf{S}$ so that $M^{-1}(\mathbf{s}_{min})$ is a minimum of the problem objective, f, whenever $\mathbf{s}_{min}$, is a minimum of E ($\mathbf{s}_{min} \in \mathbf{S}$). In order to utilize processing units with neuron-like behaviors, E *must* take the form of a 0-1 Hamiltonian.

4 The network is relaxed using one of a variety of unit update rules. Prior to relaxation, all network parameters must be specified.

5 After relaxation, M^{-1} is applied to the final network configuration yielding a minimum of f, the objective function of the original optimization problem.

The Knapsack Problem

Consider the integer knapsack problem given by

INSTANCE: Finite set $Q = \{1,2,\ldots,n\}$ of elements, for each $q \in Q$ a cost $w_q \in Z^+$ and a profit $p_q \in Z^+$ and a positive integer knapsack capacity, K.

OBJECTIVE: Give a subset $Q' \subseteq Q$ such that $\sum_{q \in Q'} w_q \leq K$ and $\sum_{q \in Q'} p_q$ is maximized over all subsets of Q.

The integer knapsack decision problem is known to be *NP*-complete [2]. Let s_i, $1 \leq i \leq n$, be the activation states of n neuron-like processing units such that

$$s_i = \begin{cases} 1 \text{ if } i \in Q' \\ 0 \text{ otherwise} \end{cases}$$

and $n = |Q|$. We propose the global energy function:

$$E = E_A + E_B$$

where $E_A = -A \cdot \sum_{i=1}^{n} s_i p_i$,

and $E_B = B \cdot \max \left\{ \sum_{i=1}^{n} s_i w_i - K, 0 \right\}$

E_A reflects the benefit of maximizing $\sum_{q \in Q'} p_q$ and E_B reflects the penalty for violating $\sum_{q \in Q'} w_q \leq K$. It can be shown that any neural network which embeds this energy function will require at least two different kinds of feature detectors. The knapsack problem can not be embedded in a network of functionally homogeneous visible units. *Hidden units* are necessary. In the remainder of this paper, we present a systematic method to deriving neural network algorithms for difficult optimization problems in a manner that promotes the introduction of hidden units and defines their *best features*. Our approach is applicable to many packing problems including bin packing and multiprocessor scheduling. It is easily seen that

$$\frac{\Delta E}{\Delta s_i} = \frac{\Delta E_A}{\Delta s_i} + \frac{\Delta E_B}{\Delta s_i}$$

$$\frac{\Delta E_A}{\Delta s_i} = -Ap_i$$

$$\frac{\Delta E_B}{\Delta s_i} = \begin{cases} 0 & \textbf{if } \sum_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j + w_i \leq K \\ Bw_i & \textbf{if } \sum_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j > K \\ B(w_i - K + \sum_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j) & \textbf{if } \sum_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j \leq K \\ & \textbf{and } \sum_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j + w_i > K \end{cases}$$

From the discrete differential of E, the function of each unit in a network, **G1**, of non-neural processors can be determined. Rather than adopting the strict threshold rule of a McCulloch-Pitts neuron we first employ stochastic smoothing of the type used by the units of a Boltzmann machine [3] followed by a *mean-field* transformation into a model like those of Hopfield and Tank. The function of unit i in network **G1** is given by:

$$q_i \leftarrow \sum_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j \; ;$$

$$\Delta E \leftarrow -Ap_i \; ;$$

if $(q_i > K)$ **then**
$\Delta E \leftarrow \Delta E + Bwi$;

if $(q_i \leq K)$
and $(q_i + w_i > K)$ **then**
$\Delta E \leftarrow \Delta E + B(w_i + q_i - K)$;

$$s_i \leftarrow \frac{1}{1 + e^{(-\Delta E / T)}} \; ;$$

Unit Function
Analog Network **G1**, unit i ($1 \leq i \leq n$)

We rewrite this function replacing implicit binary threshold functions with explicit "smooth threshold" functions. Let

$$\text{CTHRESH}(x) = \frac{1}{1+e^{(-x/T)}}$$

The function of unit i $(1 \le i \le n)$ in the modified network, **G1'**, is given by:

$$q_i \leftarrow \sum_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j \; ;$$

$$c1_i \leftarrow \text{CTHRESH}(\, q_i - K \,);$$

$$c2_i \leftarrow \text{CTHRESH}(\, K - q_i + \varepsilon \,);$$

$$c3_i \leftarrow \text{CTHRESH}(\, q_i - K + w_i \,);$$

$$c23_i \leftarrow \text{CTHRESH}(\, Dc2_i + Dc3_i - \tfrac{3D}{2} \,);$$

$$\Delta E \leftarrow -Ap_i + c1_i \cdot Bw_i + c23_i \cdot B(w_i + q_i - K \,);$$

$$s_i \leftarrow \frac{1}{1+e^{(-\Delta E/T)}} \; ;$$

Unit Function
Modified Analog Network G1', unit i $(1 \le i \le n)$
Explicit representation of Smoothed Binary-Threshold Functions

where $D>0$ is a constant network parameter (similar to A or B) and $0<\varepsilon<<1$ is utilized to detect the condition "$q_i=K$". This function deserves a brief explanation. Since, in the low-temperature limit, $c1_i=1$ whenever $q_i-K>0$ and $c1_i=0$ whenever $q_i-K\le 0$, by adding $c1_i \cdot Bw_i$ to ΔE we preserve the function of the statement:

$$\textbf{if } (q_i > K) \textbf{ then } \Delta E \leftarrow \Delta E + Bwi \; ;$$

Since $c23_i=1$ if, and only if $c2_i=1$ and $c3_i=1$, it is clear that $c23_i=1$ whenever $(q_i \le K)$ and $(q_i+w_i>K)$ and $c23_i=0$ otherwise. Multiplication of $B(w_i+q_i-K)$ by $c23_i$ prior to summation with ΔE suffices to increment ΔE by $B(w_i+q_i-K)$ whenever $(q_i \le K)$ and $(q_i+w_i>K)$. This change preserves the function of the statement:

$$\textbf{if } (q_i \le K) \textbf{ and } (q_i + w_i > K) \textbf{ then } \Delta E \leftarrow \Delta E + B(\, w_i + q_i - K \,) \; ;$$

We are now in a position to make an interesting observation. Since the values s_i, $c1_i$, $c2_i$, $c3_i$, and $c23_i$ are computed continuously, if we constrain these values from changing too rapidly, then $c1_i$, $c2_i$, $c3_i$, and $c23_i$ can be computed in parallel. We simply replace the entire unit with a network of 5 simpler units as shown in figure 1. We refer to these units as *c1-*, *c2-*, *c3-*, *c23-*, and *s*-units and the resulting network as **G2**. The astute reader will notice that we have introduced neuron-like behavior to each unit. The only anomaly of network **G2** is in the function of the *s*-unit which must compute a weighted product of the incoming activation signals from other *s*-units and its *c23*-unit. This special case is represented by a *higher-order interaction* or *conjunctive synapse*. The activation value of the *c23*-unit must moderate the transmission of the activation values of all external *s*-units to the *s*-unit in its cluster. It is well known that the function of conjunctive synapses can be approximated, in the low-temperature limit, by conjunctive units. Each conjunctive synapse is simply replaced by a separate unit that detects the conjunction of the *c23*-unit and the *s*-unit. This order reduction increases the total number of units in the network from $O(n)$ to $O(n^2)$ but results in a complete network, of neuron-like processing units connected with simple synapses.

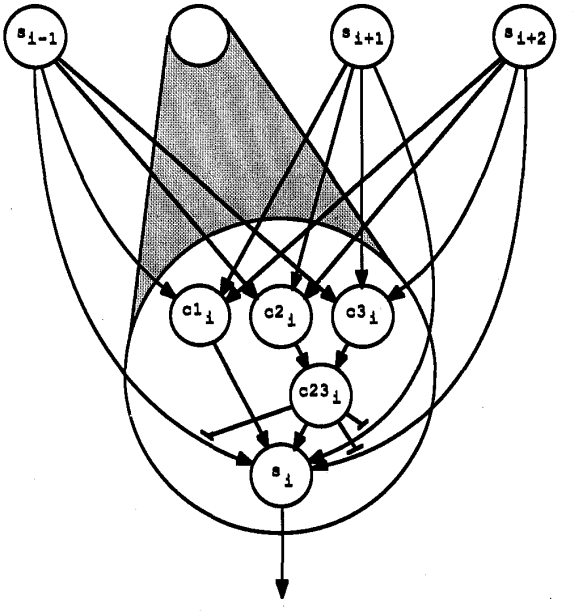

Figure 1
Analog Knapsack-Packing Network **G2**
with conjunctive synapses

Summary of the Derivation Method

Let us summarize the steps that were used to arrive at the knapsack-packing network.

1 A network representation is selected. n, the dimensionality of the network state space, is defined. A transformation, M:**V**→**S'**, from **V**, the space of problem variable configurations, to $\mathbf{S}'=\{0,1\}^n$, the corners of the n-dimensional unit hypercube, is defined.

2 An inverse transformation, $M^{-1}: \mathbf{S} \rightarrow \mathbf{V}$, is defined. Each network configuration in the volume of the n-dimensional hypercube is mapped by M^{-1} to a configuration of the variables of the problem space.

3 A discrete energy function, $E: \mathbf{S}' \rightarrow \Re$, is defined on $\mathbf{S}'$ so that $M^{-1}(\mathbf{s}_{min})$ is a minimum of the problem objective, f, whenever $\mathbf{s}_{min}$, is a minimum of E ($\mathbf{s}_{min} \in \mathbf{S}$). E need not take the form of a 0-1 Hamiltonian.

4 The discrete partial differential, $\frac{\Delta E}{\Delta s_i}$, is derived giving the gradient of E at the corners of the hypercube.

5 $\frac{\Delta E}{\Delta s_i}$ defines the function of a non-neural, analog unit (network **G1**) to which stochastic smoothing and the mean-field transformation is applied. This transformation provides an interpolation of $\frac{\Delta E}{\Delta s_i}$ in the interior of the hypercube, **S**.

7 Implicit binary-threshold functions are identified and rewritten explicitly via the CTHRESH function. Auxiliary binary variables (e.g. *c23*) are assigned to represent boolean combinations (***and***, ***or***, etc.) of simple conditions giving network **G1'**.

9 The entire non-neural analog unit is replaced with a collection of hidden units together with a single visible unit giving network **G2**. A hidden unit is introduced for each condition, that is, each application of CTHRESH.

This derivation technique differs from Hopfield's technique in two important aspects. The initial proposed energy function E need not take the form of a Hamiltonian and need not even be continuously differentiable on the interior of the hypercube. Consequently, we will not have neuron-like unit behavior, i.e., summation and thresholding. Neuron-like behavior must be re-introduced by steps 6-9. More importantly, explicit recognition of implicit binary thresholds that are present in $\frac{\Delta E}{\Delta s_i}$ allows us to eventually replace these functions with hidden units. Our constructive method simplifies the conceptual "leap" that must be taken when reducing non-trivial discrete optimization problems to analog neural-network algorithms.

Simulation

A number of simulations were performed on networks **G1** and **G2**. In all simulations an asynchronous, sequential update rule like that of the Boltzmann machine was used. Prior to relaxation, activation values of all units were initialized to $\mathbf{U}(0.4,0.6)$ (continuous uniform on (0.4,0.6)) random numbers. Simple fixed-length exponential annealing schedules of 10 or 20 fixed-temperature "chains" were used.

At each temperature, a fixed number of *update cycles* were executed. For each update cycle, n random unit probes were made. Each time a unit was probed, its state was updated according to its prescribed local function. The constant ε, used to bias the *c23-* units, was arbitrarily set to 0.01 and the threshold of M^{-1}, χ, was set to 0.5.

The network parameter D is used to control the "decisiveness" of the conjunctive *c23-* units. Since

$$e^{-\frac{(Dc2_i + Dc3_i - \frac{3D}{2})}{T}} = e^{-\frac{(c2_i + c3_i - \frac{3}{2})}{TD^{-1}}}$$

D^{-1} can be interpreted as the ratio of the temperature of the *c23-* units to the temperature of the other units In order for the *c23-* units to run at roughly the same temperature as all other units, we defined D as a function of A and B, specifically, D is set to the average absolute value of all connection weights and biases excluding both the connections directed into the *c23-* units and the biases of the *c23-* units.

The remaining network parameters A and B are related. A and B are coefficients of energy terms that represent conflicting ground states. By increasing A relative to B we emphasize profitable packings but increase the likelihood of exceeding the knapsack capacity. By increasing B relative to A, we emphasize valid packings (i.e. packings for which $\sum_{q \in Q'} w_q \leq K$) and give less importance to the profit, $\sum_{q \in Q'} p_q$, of the packing. In our simulations A was fixed at 6.0 and different values of B were tested.

Problem instances were randomly generated as a function of K. Specifically, $n \sim \mathbf{U}(K-5, K+5)$, $w_i \sim \mathbf{U}(1,K)$, and $p_i \sim \mathbf{U}(1,K)$ for $1 \leq i \leq n$. Several values of K (10, 20, 35, and 80) were tested. For each combination of network parameters, and for each problem instance, 10 simulations were performed. After a network was constructed for each problem instance, the network was relaxed using the 10 or 20 step annealing schedule. The resulting approximate solutions were compared with those produced by the standard "greedy" algorithm - a first-fit-decreasing packing by value/density ratios $\frac{p_i}{w_i}$. The performance ratios, $\frac{\text{NETWORK(I)}}{\text{GREEDY(I)}}$, are shown in table 1. On the average, the knapsack-packing network performed slightly better than the greedy algorithm.

B	% Valid Packings	NETWORK(I) / GREEDY(I)
4	60	0.9660
5	80	1.0582
6	100	1.0718
8	90	1.0069
9	95	1.0578
10	100	1.0593
12	100	1.0701
14	100	1.0380
20	100	1.0822

Table 1
Effect of Varying Network Parameter *B*
Network **G2**, $K = 20$, 20 step annealing schedule, $A = 6.0$

Out of curiosity, 20 simulations of network **G1'** were executed with $|Q| \sim 200$ elements. This would correspond to a **G2** network of 1000 units. No effort was made to tune the network parameters, *A* and *B*, to improve the results of this particular simulation ($A = 6$ and $B = 14$ were used). Nevertheless, a performance ratio of 0.9721 was achieved. This result suggests that degradation due to increased frustration when scaling may not be as serious a problem as we expected

Conclusion

The approach we have adopted allows us to start with very simple network representations of the variable spaces of difficult optimization problems with non-Hamiltonian energy functions and to transform them via a sequence of rewrites into analog networks of neuron-like units. Preliminary simulation results have been promising, especially in light of the ease with which we have been able to find viable network parameters. In the course of subsequent research, we have found the method to be applicable to bin-packing, multiprocessor scheduling, and job-sequencing problems as well. We hope to simulate networks for these problems in the near future.

References

[1] Dahl, E.D., Neural Network Algorithm for an *NP*-complete problem: Map and Graph Coloring, *IEEE ICNN conf. proc.*, Vol. 3, pp 113-120, June, 1988.

[2] Garey, M., and Johnson, D., Computers and Intractability; A Guide to the theory of *NP*-completeness, Freeman, 1979.

[3] Hinton, G.E., Sejnowski, T.J., Ackley, D.H., Boltzmann Machines: Constraint Satisfaction Networks that Learn, *Tech. Rep. No. CMU-CS-84-119*, Carnegie-Melon University, Pittsburgh, PA, May, 1984.

[4] Hopfield, J.J., and Tank, D.W., "Neural" Computation of Decisions in Optimization Problems, *Biological Cybernetics*, Vol. 52, pp. 141-152, Springer-Verlag, 1985.

[5] Ramanujam, J., Sadayappan, P., Optimization by Neural Networks, *IEEE ICNN conf. proc.*, Vol. 2, pp 325-332, June, 1988.