HNN Based Neuromorphic Hardware for Combinatorial Optimization

Created	@June 5, 2023 10:35 AM
Reviewed	☐

Hopfield Neural Networks

Overview

A type of artificial neural networks specifically Recurrent Neural Networks

Widely used in pattern recognition, optimization, and associative memory applications

A fully connected network of N neurons

Can be operated with discrete or continuous variables as inputs

We focus on a HNN architecture with binary neurons operating on values {-1, 1}

Neuron Update Mechanism

x : N-dimensional **input vector** to the set of N neurons,

y : N-dimensional **output vector**

W : $N \times N$ adjacency **weight matrix** denoting the connection strength between all the neurons

b : N-dimensional **bias vector**, sometimes also known as threshold, will depend on

$$y = \text{sign}(W^T x + b)$$

where

$$\text{sign}\left(\mathbf{f}(\mathbf{z}^{(t)})\right) = \begin{cases} \mathbf{1} & \text{if } \mathbf{f}(\mathbf{z}^{(t-1)}) > 0 \\ -\mathbf{1} & \text{if } \mathbf{f}(\mathbf{z}^{(t-1)}) < 0 \\ \mathbf{z}^{(t-1)} & \text{if } \mathbf{f}(\mathbf{z}^{(t-1)}) = 0 \end{cases},$$
$$\mathbf{f}(\mathbf{z}) \text{ is a linear transformation of } \mathbf{z}$$

▼ Energy of the Network

Energy of an HNN is given by:

$$\begin{aligned} E &= -\frac{1}{2}\left(x^T W x + b^T x\right) \\ &= -\frac{1}{2}\left(\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij} x_i x_j + \sum_{i=1}^{N} b_i x_i\right) \end{aligned}$$

For a *learning type problem* like pattern recognition:

- One is given training data in form of multiple sets of input vectors x and output vectors y.
- The goal is to come up with a good model (W, b) such that the predictions made with this model match closely with those of real prediction data.

For an *optimization type problem* (in graphs):

- The node and edge weights are given, which fixes the corresponding weight matrix $W$and threshold b of the HNN. W and b can be obtained by translates the objective (+constraints) of the problem at hand into the the energy function of the network defined above.
- The goal, here, is to find the optimal solution i.e. an input vector x for which the HNN energy is globally minimized.

🤔 Realizing the *dual nature* between the above two problem types, can we say anything about the hardness of the former since we know the hardness of the latter?

▼ Natural Energy Minimization in HNN

Consider an update made to the i^{th} neuron.

$$y_i = \text{sign}\left(\left(\sum_{j=1}^{n} w_{ij} x_j\right) + b_i\right)$$

The energy change in the network due to the above *single* change is:

$$\begin{aligned}\Delta E = E^f - E^i = E_i^f - E_i^i \\ &= -\frac{1}{2}\left[\sum_{j=1}^{N} w_{ij}(y_i - x_i)x_j + b_i(y_i - x_i)\right] \\ &= -\frac{1}{2}(y_i - x_i)\left[\sum_{j=1}^{N} w_{ij} x_j + b_i\right]\end{aligned}$$

This is true under the important assumption that $w_{ii} = 0 \ \forall i \in \{1, \ldots, N\}$.

Note that the sign of $\sum_{j=1}^{N} w_{ij} x_j + b_i$ is exactly the output at the i^{th} neuron, y_i .

Now,

$$\sum_{j=1}^{N} w_{ij} x_j + b_i > 0 \quad \Rightarrow \quad y_i = 1 \quad \Rightarrow \quad y_i \geq x_i \quad \Rightarrow \quad \Delta E \leq 0$$

Similarly,

$$\sum_{j=1}^{N} w_{ij} x_j + b_i < 0 \quad \Rightarrow \quad y_i = -1 \quad \Rightarrow \quad y_i \leq x_i \quad \Rightarrow \quad \Delta E \leq 0$$

and

$$\sum_{j=1}^{N} w_{ij} x_j + b_i = 0 \quad \Rightarrow \quad y_i = x_i \quad \Rightarrow \quad \Delta E = 0$$

Thus, we always have $\Delta E \leq 0$.

▼ A few things to note

- We assumed zero diagonal weights for W. If we relax this assumption, there will be upper limit to the magnitude of w_{ii}, i.e. $|w_{ii}| < k \max_{i \neq j}(|w_{ij}|)$.
- We update only one neuron at a time. This is called the *synchronous* approach.
- For the *asynchronous* case, multiple neurons can be updated in a batch of say k selected neurons (k can be equal to N). Depending on the weight matrix W, this may or may not converge in the expected solution.
- Any QUBO problem can be mapped to a unique a HNN.
- The network is updated for a series of n_s steps to allow the energy to go down such that the corresponding final output vector y can be obtained.
- The whole update procedure is run for n_e epochs, and every epoch starts with a randomly initialized input vector x.
- Although the epoch for which the final energy is minimum is considered as the result. Most update paths may not converge to this solution, in which case, the energy that most epochs converge to can be considered a local minima.

Annealing Techniques

Stochastic Simulated Annealing

Now, consider the update at every neuron as follows:

$$y_i = \begin{cases} 1 & \text{with probability} \quad p_i^T(\sum_j w_{ij} x_j + b_i) \\ -1 & \text{with probability} \quad 1 - p_i^T(\sum_j w_{ij} x_j + b_i) \end{cases}$$

where p_i^T is the probability distribution which in this case is taken to be a signum function from 0 to 1, centered at $\frac{1}{2}$. For a given input x_i, The closer p_i is to 1, the higher the chances of getting an output 1. On the other hand, the closer it is to 0, the higher the chances of the state of the neuron being updated to -1.

T in superscript denotes the dependence on temperature.

$$p_i^T(z) = \frac{1}{1 + \exp(-z/T)}$$

▼ Weight Annealing

In combinatorial optimization, in order to maximize/minimize an objective function, it is common to only vary the binary variables by trial and error following some annealing technique.

Although the weight matrix (and bias) is fixed for a problem, we don't necessarily have to start with those values. In weight annealing, we start with an easy problem, say a constant weight matrix (and bias), and gradually go towards that of our required problem.

In the initial stages, the HNN would automatically converge to the lowest energy solution for the modified (easier) problem, since there won't be any possibility of getting stuck in a local minima.

Weight annealing relies on the idea that if the lowest energy solution for the previous iterations has been achieved, it is highly likely that the network will stay in the lowest energy for the current problem as well, as long as the weights in the network are changed gradually.

Specifically, we have a parameter called annealing time, τ that controls how slow the whole process of changing the weights from constant values to the required values is done.

This idea of gradual change in the network weights might be inspired from *quantum annealing*.

▼ Quantum Annealing or Quantum Adiabatic Optimization [2]

> Suppose we have a Quantum Hamiltonian H_P whose ground state encodes the solution to a problem of interest, and another Hamiltonian H_0, whose ground state is "easy" (both to find and to prepare in an experimental setup). Then, if we prepare a quantum system to be in the ground state of H_0, and then adiabatically change the Hamiltonian for a time T according to

$$H(t) = \left(1 - \frac{t}{T}\right) H_0 + \frac{t}{T} H_P$$

> if T is large enough, and H_0 and H_P do not commute, the quantum system will remain in the ground state for all times, by the adiabatic theorem of quantum mechanics. At time T, measuring the quantum state will return a solution of our problem.

▼ Approaches for Weight Annealing

Exponential Annealing

$$d_\tau(i,j) = d \cdot (1 - \exp(-1/\tau))$$

$$\tau : \infty \to 0$$

Linear (Semi-exponential) Annealing

$$d_t = d \cdot t$$

$$t : 0 \to 1$$

Exponential Smoothing [1]

$$d_\alpha(i,j) = \begin{cases} \bar{d} + \frac{\alpha}{\exp(\frac{\alpha}{d(i,j)-\bar{d}})-1} & \text{if } d(i,j) \geq \bar{d} \\ \bar{d} - \frac{\alpha}{\exp(\frac{\alpha}{\bar{d}-d(i,j)})-1} & \text{if } d(i,j) < \bar{d} \end{cases}$$

$$\begin{aligned} \alpha \to 0 \quad &\implies \quad d_\alpha(i,j) \to d(i,j) \\ \alpha \to \infty \quad &\implies \quad d_\alpha(i,j) \to \bar{d} \end{aligned}$$

$$\alpha : \infty \to 0$$

Power Smoothing [1]

$$d_\alpha(i,j) = \begin{cases} (d - \bar{d})^\alpha & \text{if } d(i,j) \geq \bar{d} \\ -(d - \bar{d})^\alpha & \text{if } d(i,j) < \bar{d} \end{cases}$$

$$\alpha : 0 \to 1$$

▼ Hybrid Approach

A combined approach that involves simultaneous **weight** as well as **stochastic** annealing.

▼ Problem Solving

Through an implementation of a Hopfield Neural Network, our goal here is to solve certain NP-Hard/NP-Complete problems in combinatorial optimization such as Maximum Cut, Maximum Clique, Minimum Vertex Cover, Maximum Independent Set.

The energy functions for each of the problem below are to be minimized.

Max-Weighted-Cut

$$\min \quad \frac{1}{4}(\mathbf{1}^T W \mathbf{1} - x^T W x)$$

Max-Weighted-Clique

$$\min \quad \frac{1}{8}((x+\mathbf{1})^T (W\mathbf{1} - b) - (x+\mathbf{1})^T W (x+\mathbf{1}))$$

Min-Weighted-Vertex-Cover

$$\min \quad -\frac{1}{8}((x+\mathbf{1})^T (W\mathbf{1} + b) + (x-\mathbf{1})^T W (x-\mathbf{1}))$$

Max-Weighted-Independent-Set

$$\min \quad \frac{1}{8}((x+\mathbf{1})^T (W\mathbf{1} - b) - (x+\mathbf{1})^T W (x+\mathbf{1}))$$

▼ Results

For Max-Cut data: http://biqbin.eu/Home/Benchmarks

Baseline

dataset	optimal solution energy	obtained solution energy	time to solution (s)
mc_21_199	109	107	3.13
mc_50_156	1762	1758	3.80

mc_63_1729	956	956	4.62
mc_100_242	197	192	5.00
mc_101_5006	4322	4322	4.87
mc_800_4667	3044	2782	170.13

Stochastic Annealing

dataset	optimal solution energy	obtained solution energy	time to solution (s)
mc_21_199	109	109	12.89
mc_50_156	1762	1758	13.56
mc_63_1729	956	956	13.51
mc_100_242	197	192	15.72
mc_101_5006	4322	4322	14.87
mc_800_4667	3044	2786*	346.11

Exponential Annealing

dataset	optimal(or approx.) solution	obtained solution	time to solution (s)
mc_21_199	109	109	4.25
mc_50_156	1762	1710	9.98
mc_63_1729	956	956	10.97
mc_100_242	197	192	13.12
mc_101_5006	4322	4322	12.69
mc_800_4667	3044	2881*	357.67

* denotes a higher number of steps applied because of the large dataset.

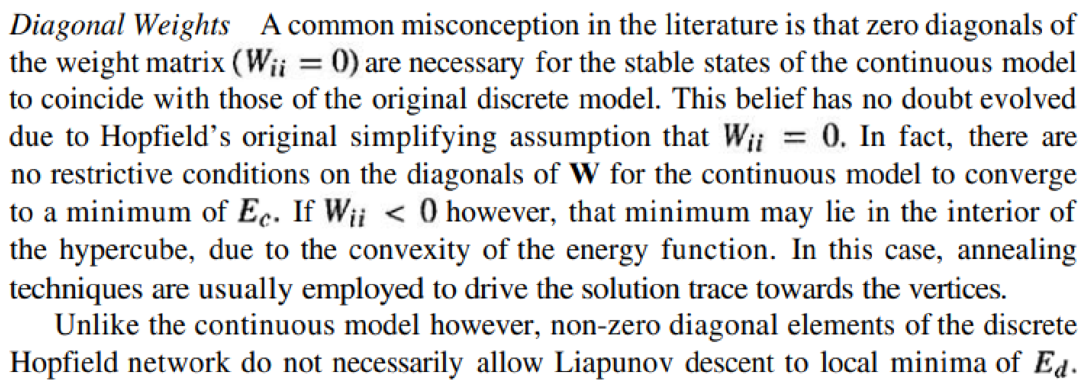
▼ References

1. J. Schneider et al., “Search-space smoothing for combinatorial optimization problems”, Physica A, 243, pp. 77-112, 1997.
2. L. Andrew, “Ising formulations of many NP problems”, Front. Phys., Vol. 2, 2014.
3. M. R. Mahmoodi et al., “Versatile stochastic dot product circuits based on nonvolatile memories for high performance neurocomputing and neurooptimization”, 2019.

4. Z. Fahimi et al., “Combinatorial Optimization by Weight Annealing in Memristive Hopfield Networks”, 2021.

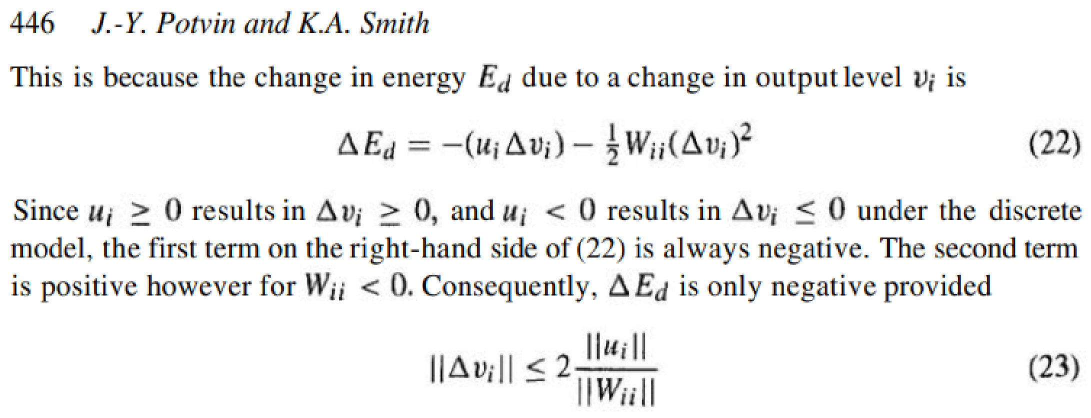
Appendix

▼ Issues with negative diagonal weights in the Weight Matrix



Diagonal Weights A common misconception in the literature is that zero diagonals of the weight matrix ($W_{ii} = 0$) are necessary for the stable states of the continuous model to coincide with those of the original discrete model. This belief has no doubt evolved due to Hopfield's original simplifying assumption that $W_{ii} = 0$. In fact, there are no restrictive conditions on the diagonals of $\mathbf{W}$ for the continuous model to converge to a minimum of E_c. If $W_{ii} < 0$ however, that minimum may lie in the interior of the hypercube, due to the convexity of the energy function. In this case, annealing techniques are usually employed to drive the solution trace towards the vertices.

Unlike the continuous model however, non-zero diagonal elements of the discrete Hopfield network do not necessarily allow Liapunov descent to local minima of E_d.



446 *J.-Y. Potvin and K.A. Smith*

This is because the change in energy E_d due to a change in output level v_i is

$$\Delta E_d = -(u_i \Delta v_i) - \tfrac{1}{2} W_{ii} (\Delta v_i)^2 \tag{22}$$

Since $u_i \geq 0$ results in $\Delta v_i \geq 0$, and $u_i < 0$ results in $\Delta v_i \leq 0$ under the discrete model, the first term on the right-hand side of (22) is always negative. The second term is positive however for $W_{ii} < 0$. Consequently, ΔE_d is only negative provided

$$||\Delta v_i|| \leq 2 \frac{||u_i||}{||W_{ii}||} \tag{23}$$

Book: PotvinSmith_NeuralNetworks-Corrected.pdf (uab.es), page: 446

▼ Other Resources to look at

- [ ] Hopfield network - Wikipedia
- [ ] PotvinSmith_NeuralNetworks-Corrected.pdf (uab.es) (page 446)
- [ ] Hopfield Network - an overview | ScienceDirect Topics
- [ ] Problem Solving with Hopfield Networks and Adiabatic Quantum Computing
- [ ] Quantum Hopfield neural network

▼ Questions to Explore

1. Operating HNN near the critical point?

Critical point: the edge b/w order and disorder.

> Remember, the stochastic neuron simulation also gave some sort of boundary between a poor/noisy solution till a certain value of **T** and then better solution relatively suddenly after that value.

Critical phenomena occurs at a lot of places.

- phase change
- spins in lattices
- brain (claimed here)

Critical dynamics are best demonstrated in a simplified system known as the Ising model.

2. Neural Networks: Dual Nature of learning and solving?

…