Introduction to Neural Networks

Johns Hopkins University
Engineering for Professionals Program

605-447/625-438

Dr. Mark Fleischer

Module 9.1: Hopfield Memory Capacity

What We've Covered So Far…

- Learned about the Hopfield Network
  - Hebbian Learning
    - Recurrent neural networks
    - Matrix/vector representation of a recurrent network
    - Heat death
  - Excitation and Inhibition in Hopfield Networks via bi-polar values
  - Possibility of cycling in Hopfield networks in part because of inhibition and excitation.
- In this sub-module
  - We examine how many exemplars a Hopfield network can 'reasonably' store.

H-N Function Never Increases

- This means, the *H* value can only decrease and since there is a lower bound, it will eventually converge such that

$$f_h\left(\sum_j w_{ij}x_j\right) = \mathbf{x}$$

Cannot oscillate between solutions.

The Hopfield Outerproduct

Recall $$w_{ij} = \begin{cases} \sum_{s=1}^{P} x_i^s x_j^s & i \neq j \\ 0 & i = j \end{cases}$$

$$\text{Let } \mathbf{x}^r = \left(x_1^r, x_2^r, \ldots, x_n^r\right)$$

This corresponds to the r[th] exemplar.

Memory Recall/Completion

Activity of the i^{th} neuron at the end of the first iteration.

$$a_i(1) = \sum_{\substack{j=1 \\ j \neq i}}^{n} w_{ij} x_j^r$$

Substituting in the expression for w_{ij}

$$a_i(1) = \sum_{\substack{j=1 \\ j \neq i}}^{n} \sum_{s=1}^{P} x_i^s x_j^s x_j^r$$

Memory Recall/Completion

From previous slide:

$$a_i(1) = \sum_{\substack{j=1 \\ j \neq i}}^{n} \sum_{s=1}^{P} x_i^s x_j^s x_j^r$$

Let's do some mathematical tricks and break up the double sum:

$$a_i(1) = \sum_{\substack{j=1 \\ j \neq i}}^{n} \left[\underbrace{x_i^r x_j^r x_j^r}_{\substack{s=r^{\text{th}} \\ \text{term}}} + \sum_{\substack{s=1 \\ s \neq r}}^{P} x_i^s x_j^s x_j^r \right]$$

Memory Recall/Completion

Some further manipulations:

$$a_i(1) = \sum_{\substack{j=1 \\ j \neq i}}^{n} \left[\underbrace{x_i^r x_j^r x_j^r}_{\substack{s=r^{\text{th}} \\ \text{term}}} + \sum_{\substack{s=1 \\ s \neq r}}^{P} x_i^s x_j^s x_j^r \right]$$

$$= \sum_{\substack{j=1 \\ j \neq i}}^{n} x_i^r x_j^r x_j^r + \sum_{\substack{j=1 \\ j \neq i}}^{n} \sum_{\substack{s=1 \\ s \neq r}}^{P} x_i^s x_j^s x_j^r$$

Memory Recall/Completion

Some further manipulations:

$$a_i(1) = \sum_{\substack{j=1 \\ j \neq i}}^{n} x_i^r x_j^r x_j^r + \sum_{\substack{j=1 \\ j \neq i}}^{n} \sum_{\substack{s=1 \\ s \neq r}}^{P} x_i^s x_j^s x_j^r$$

$$= x_i^r \sum_{\substack{j=1 \\ j \neq i}}^{n} x_j^r x_j^r + \sum_{\substack{j=1 \\ j \neq i}}^{n} \sum_{\substack{s=1 \\ s \neq r}}^{P} x_i^s x_j^s x_j^r$$

So …

$$a_i(1) = x_i^r (n-1) + N_i$$

Call this S_i

Memory Recall/Completion

$$a_i(1) = x_i^r \sum_{\substack{j=1 \\ j \neq i}}^{n} x_j^r x_j^r + \sum_{\substack{j=1 \\ j \neq i}}^{n} \sum_{\substack{s=1 \\ s \neq r}}^{P} x_i^s x_j^s x_j^r$$

$$= x_i^r (n-1) + N_i$$

If the noise term is 0 and we put this into the hard-limiting function

$$f_h\left(a_i(1)\right) = f_h\left(x_i^r (n-1) + N_i\right)$$

$$= f_h\left(x_i^r (n-1)\right)$$

$$= x_i^r$$

When is the *Noise* term likely to be 0?

When two different exemplars are *statistically independent.*

What does that mean?

No significant correlations between two exemplars regarding corresponding vector elements.

Roughly, each set of corresponding vector elements has approximately a 50% chance of having the same value.

We can simply say therefore that the sum of the products of the corresponding elements have an expectation value of 0!

A = (1, 1, -1, 1, -1, 1)
B = (-1, 1, 1, -1, -1, 1)

-1 + 1 – 1 – 1 + 1 + 1 = 0

Memory Recall/Completion

What if N_i is not zero? Let's make some quick, hand-waving arguments:

Let's assume that the exemplars are "statistically independent". Therefore:

$$N_i \sim N(0, \sigma^2) \text{ that is } \mu(N_i) = 0$$

$$\text{and } \sigma^2(N_i) = (n-1)(P-1)$$

Recall that for a normally distributed random variable, 99.5% of all events lie within 3σ.

Memory Recall/Completion

We want to ensure that $|S_i| > |N_i|$

$$|S_i| = n - 1 \geq 3\sigma = 3\sqrt{(n-1)(P-1)}$$

$n - 1 \geq 3\sqrt{(n-1)(P-1)}$ and squaring both sides...

$$(n-1)^2 \geq 9(n-1)(P-1)$$

$$n - 1 \geq 9(P-1)$$

Or roughly $P \approx n/9$.

Introduction to Neural Networks

Johns Hopkins University
Engineering for Professionals Program
605-447/625-438
Dr. Mark Fleischer

Module 9.2: Binary Associative Memories

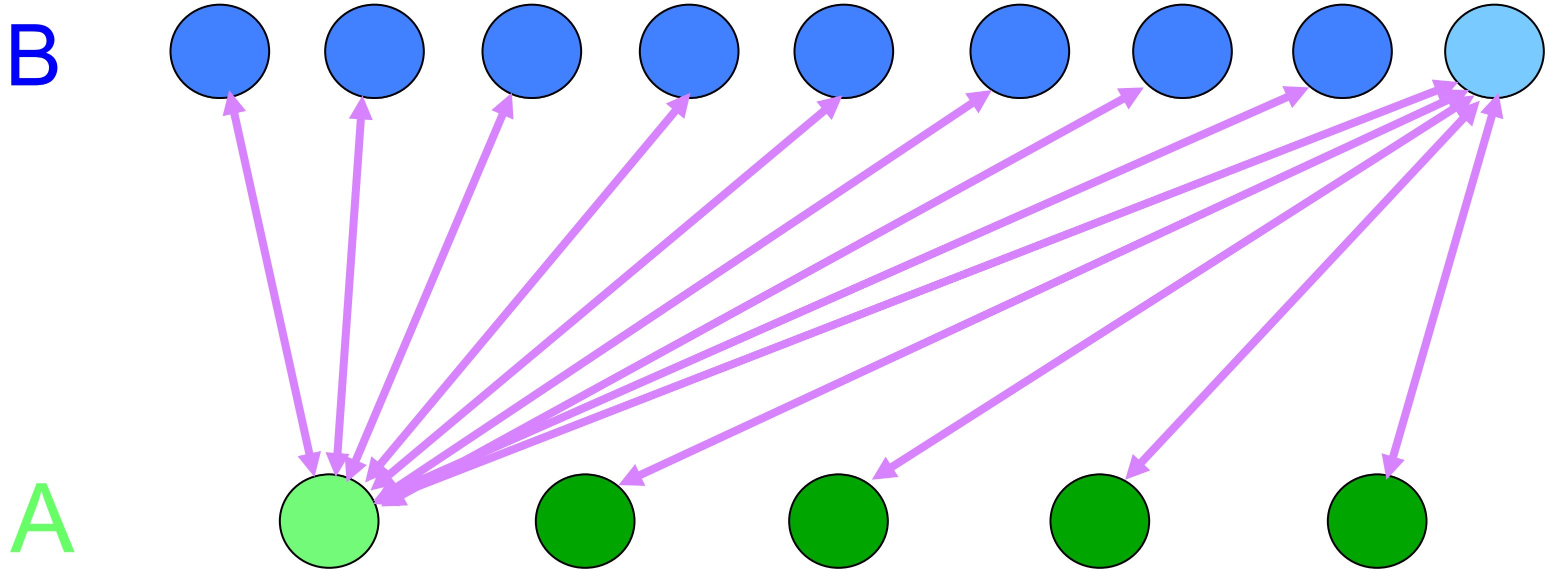
In this sub-module…

- We will learn about Binary Associative Memories (BAMs)
  - Based on *Adaptive Resonance Theory*
  - Another example of *unsupervised learning*
  - Restricted form of a Hopfield network
- We will also learn about
  - Concepts such as *Feature Detectors*
  - Matrix/vector analysis of BAMs.

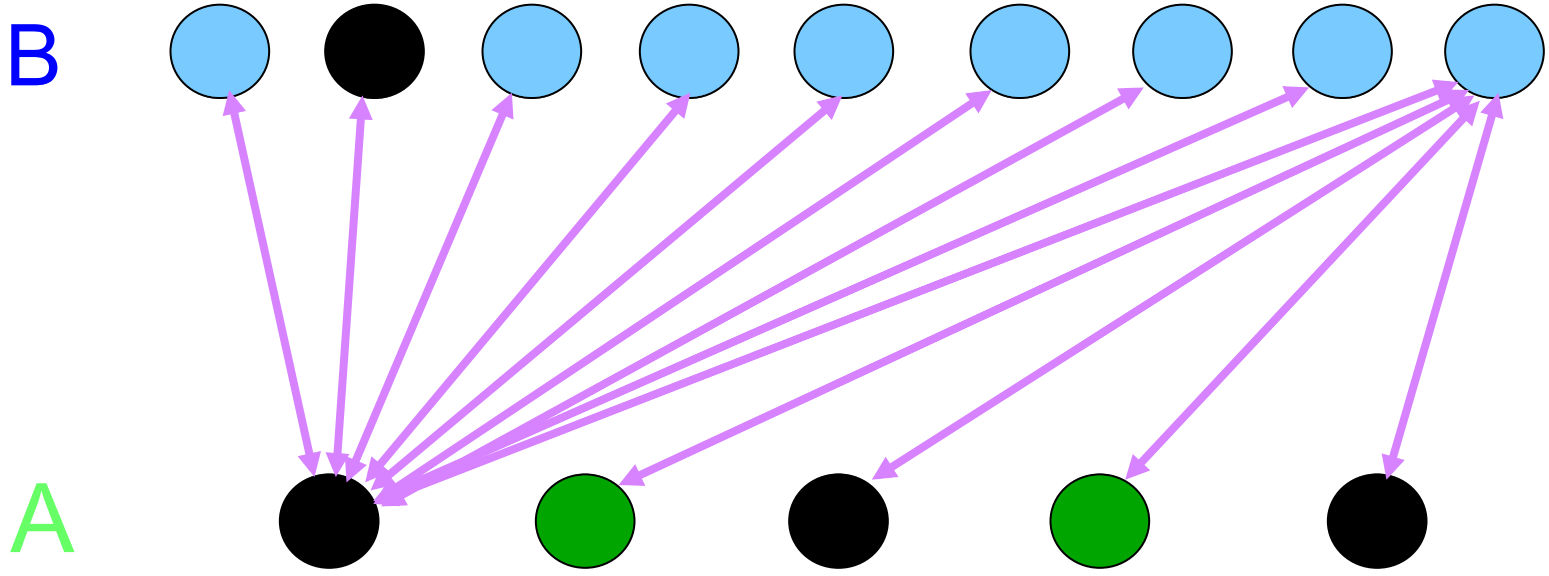
Binary Associative Memories

- Based on a bi-partite graph
- All nodes from one layer connect to all nodes in a second layer.
- No nodes in the same layer connect to each other.
- Connections are bi-directional.
- Same weights in each direction (more general examples don't have this.)

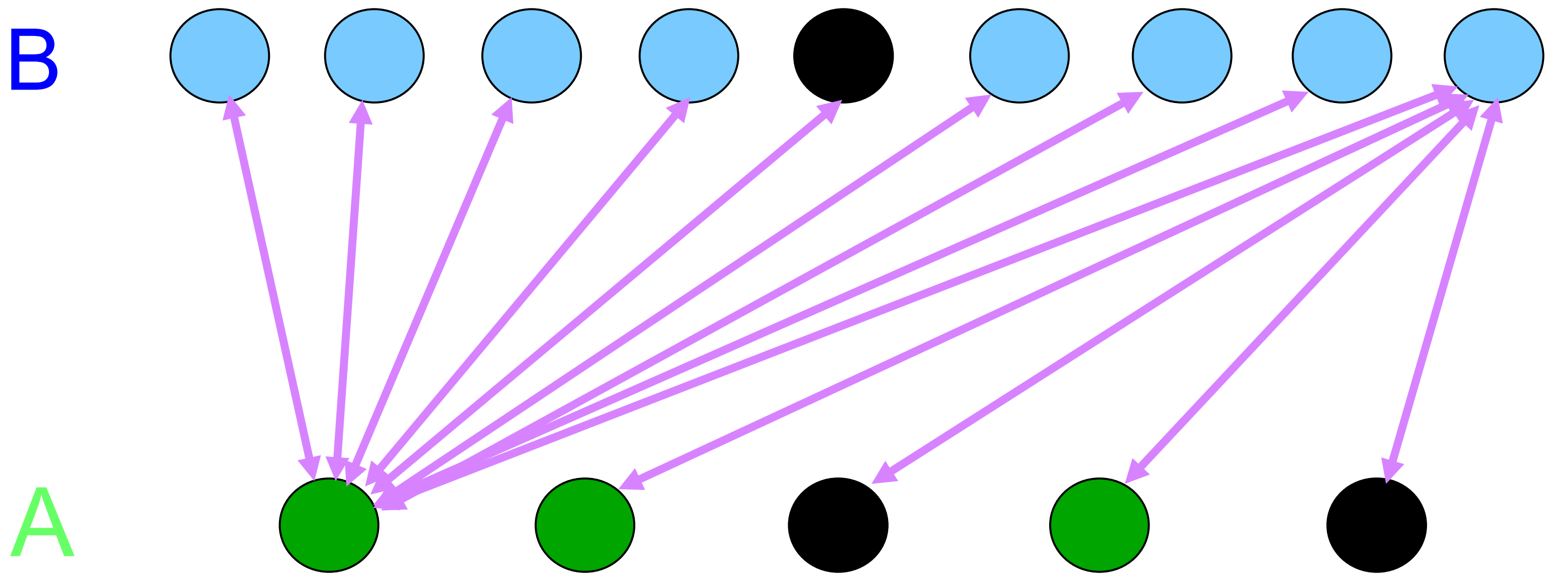
Binary Associative Memories



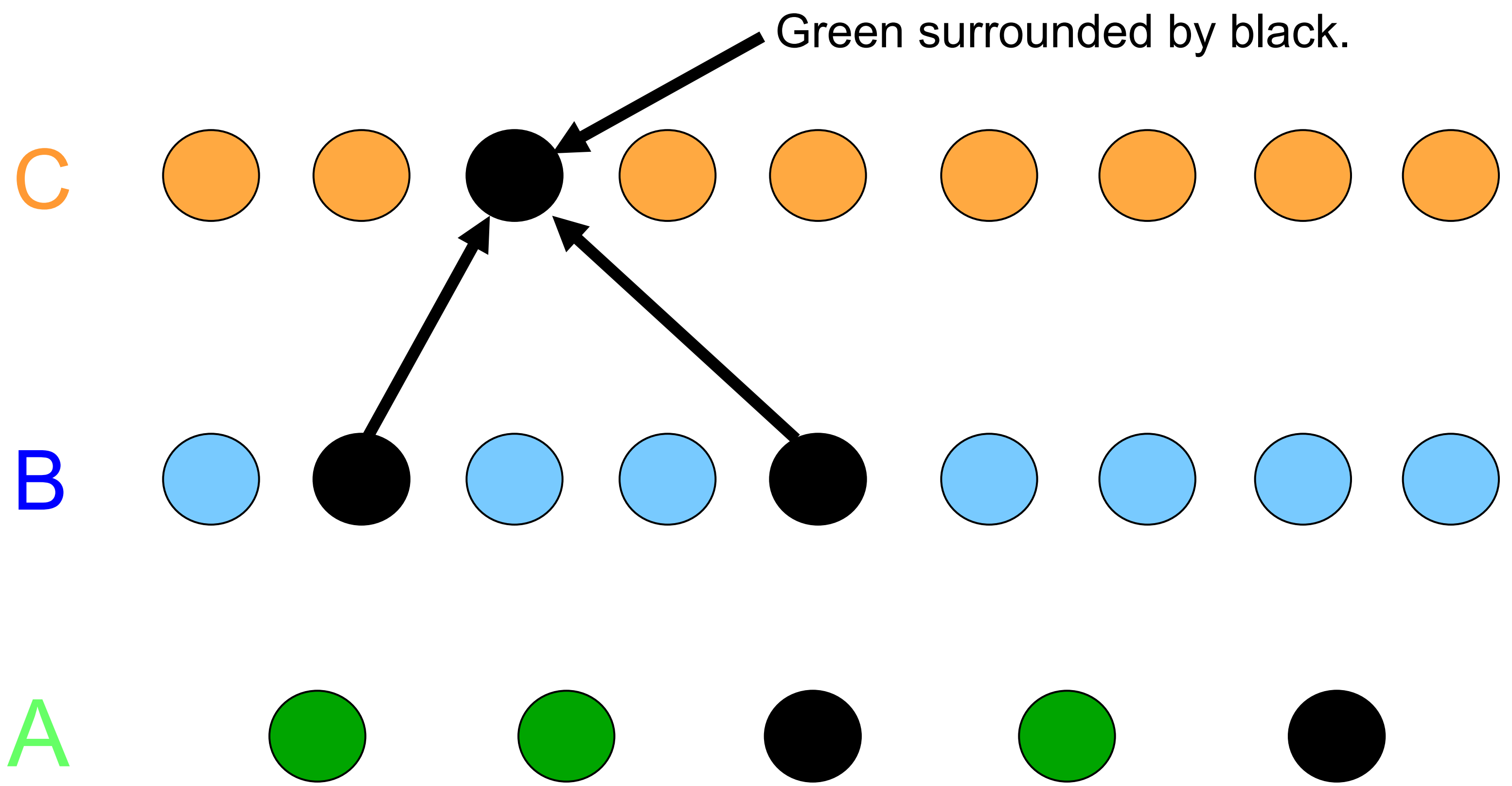
Binary Associative Memories as Feature Detectors



Binary Associative Memories as Feature Detectors



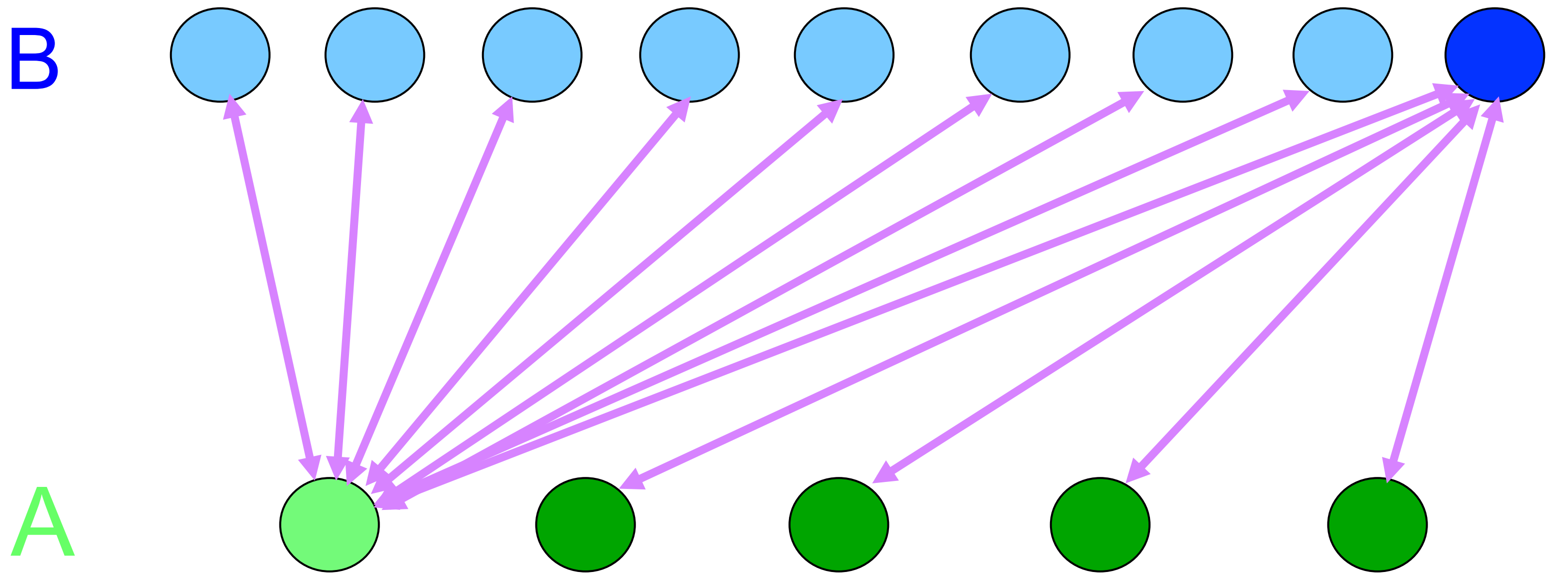
Binary Associative Memories Features of Features



Binary Associative Memories

- More layers, more feature generalizations, more abstractions.
- More layers, more versatility, more weights to train.
- For now we’ll only consider two layers.

Binary Associative Memories



We can use this topology to train the network with two sets of **associated** exemplars.

Binary Associative Memories

$$A_1 \longleftrightarrow B_1$$
$$A_2 \longleftrightarrow B_2$$
$$\vdots \qquad\qquad \vdots$$
$$A_n \longleftrightarrow B_n$$

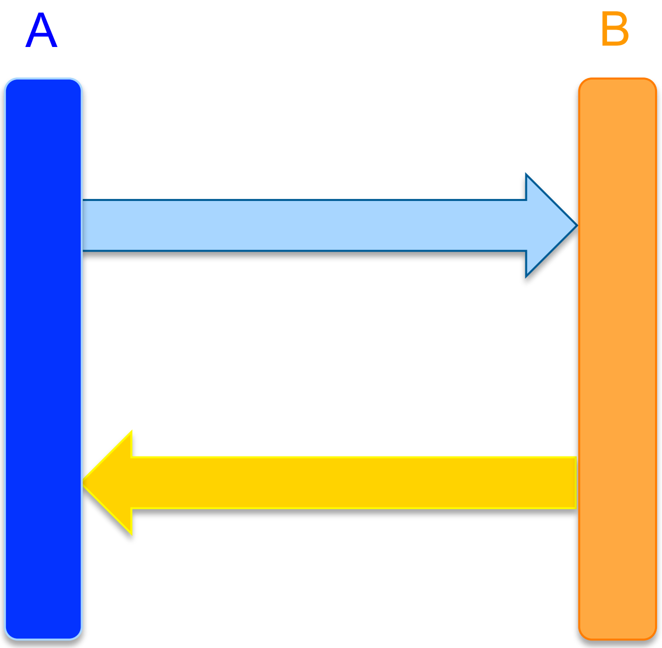
Goal: Noisy A_1 produces a correct B_1 which then produces a correct A_1.

$$\widetilde{A}_1 \rightarrow B_1 \rightarrow A_1$$

Binary Associative Memories

- Present a noisy A as input to the A nodes.
- The A nodes produce outputs and are presented to the B nodes.
- The B nodes produce outputs and are presented back to the A nodes.

Binary Associative Memories



Binary Associative Memories

An example:

$$\mathbf{A}_1^{\mathbf{T}} = \begin{pmatrix} 1, & -1, & 1, & -1, & 1, & 1 \end{pmatrix}$$

$$\mathbf{A}_2^{\mathbf{T}} = \begin{pmatrix} 1, & 1, & 1, & -1, & -1, & -1 \end{pmatrix}$$

$$\mathbf{B}_1^{\mathbf{T}} = \begin{pmatrix} 1, & 1, & -1, & 1 \end{pmatrix}$$

$$\mathbf{B}_2^{\mathbf{T}} = \begin{pmatrix} 1, & -1, & 1, & 1 \end{pmatrix}$$

Want to associate $A_1 \Leftrightarrow B_1$ and $A_2 \Leftrightarrow B_2$

Binary Associative Memories

Create a weight matrix ala Hopfield thusly:

$$\mathbf{W}_{6\times 4} = \mathbf{A}_1\mathbf{B}_1^{\mathrm{T}} + \mathbf{A}_2\mathbf{B}_2^{\mathrm{T}}$$

$$= \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & -1 & 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{pmatrix} \begin{pmatrix} 1 & -1 & 1 & 1 \end{pmatrix}$$

Binary Associative Memories

$$\mathbf{W}_{6\times4} = \begin{bmatrix} 2 & 0 & 0 & 2 \\ 0 & -2 & 2 & 0 \\ 2 & 0 & 0 & 2 \\ -2 & 0 & 0 & -2 \\ 0 & 2 & -2 & 0 \\ 2 & 0 & 0 & 2 \end{bmatrix}$$

$$\mathbf{W}_{6\times4} = \mathbf{A}_1\mathbf{B}_1^{\mathbf{T}} + \mathbf{A}_2\mathbf{B}_2^{\mathbf{T}}$$

[1 × 6] × [6 × 4] = [1 × 4]

Binary Associative Memories

$$\hat{\mathbf{A}}_1^{\mathrm{T}}\mathbf{W} = \begin{pmatrix} -1, & -1, & 1, & -1, & 1, & 1 \end{pmatrix} \begin{bmatrix} 2 & 0 & 0 & 2 \\ 0 & -2 & 2 & 0 \\ 2 & 0 & 0 & 2 \\ -2 & 0 & 0 & -2 \\ 0 & 2 & -2 & 0 \\ 2 & 0 & 0 & 2 \end{bmatrix}$$

$$\begin{pmatrix} 4, & 4, & -4, & 4 \end{pmatrix} = \hat{\mathbf{b}}^{\mathrm{T}}$$

$$f_h\left(\hat{\mathbf{b}}^{\mathrm{T}}\right) = \begin{pmatrix} 1, & 1, & -1, & 1 \end{pmatrix} = \mathbf{B}_1^{\mathrm{T}}$$

Binary Associative Memories

Just some integer value!

$$\hat{\mathbf{A}}_1^T\mathbf{W} = \hat{\mathbf{A}}_1^T\mathbf{A}_1\mathbf{B}_1^T + \hat{\mathbf{A}}_1^T\mathbf{A}_2\mathbf{B}_2^T$$

How do we analyze this multiplication?
Why does this work the way it does?

So what do these integers evaluate to?

Binary Associative Memories

When $\hat{\mathbf{A}}_1^{\mathbf{T}}$ is not too different from $\mathbf{A}_1^{\mathbf{T}}$, then most of the vector elements will be the same and $\hat{\mathbf{A}}_1^{\mathbf{T}}\mathbf{A}_1$ will be a positive number while $\hat{\mathbf{A}}_1^{\mathbf{T}}\mathbf{A}_2$ will tend to be ... ?

$$(1) \quad f_h(x) = \begin{cases} 1 & x \geq 1 \\ 0 & -1 < x < 1 \\ -1 & x \leq -1 \end{cases}$$

$$(2) \quad f_h(x) = \begin{cases} 1 & x \geq n/2 \\ 0 & -n/2 < x < n/2 \\ -1 & x \leq -n/2 \end{cases}$$

where *n* is the vector length