Computational Cognitive Neuroscience

Week 2 – Associative memories, Hopfield networks etc.

Hermanni Hälvä

University of Helsinki, 4th February 2021

Associative memories

- neural nets $\mathbf{y} = \mathbf{f}(\mathbf{x})$

Associative memories

- neural nets $\mathbf{y} = \mathbf{f}(\mathbf{x})$
  - if $\mathbf{x}, \mathbf{x}'$ are close to each other then usually so are $\mathbf{y}, \mathbf{y}'$, but not equal

Associative memories

- neural nets $\mathbf{y} = \mathbf{f}(\mathbf{x})$
  - if $\mathbf{x}, \mathbf{x}'$ are close to each other then usually so are $\mathbf{y}, \mathbf{y}'$, but not equal
- associative memories $\mathbf{y} = \mathbf{f}(\mathbf{x})$

Associative memories

- neural nets $\mathbf{y} = \mathbf{f}(\mathbf{x})$
  - if $\mathbf{x}, \mathbf{x}'$ are close to each other then usually so are $\mathbf{y}, \mathbf{y}'$, but not equal
- associative memories $\mathbf{y} = \mathbf{f}(\mathbf{x})$
  - if $\mathbf{x}'$ is close to $\mathbf{x}$ then they are *associated* to the same $\mathbf{y}$

Associative memories

- neural nets $\mathbf{y} = \mathbf{f}(\mathbf{x})$
  - if $\mathbf{x}, \mathbf{x}'$ are close to each other then usually so are $\mathbf{y}, \mathbf{y}'$, but not equal
- associative memories $\mathbf{y} = \mathbf{f}(\mathbf{x})$
  - if $\mathbf{x}'$ is close to $\mathbf{x}$ then they are *associated* to the same $\mathbf{y}$
- motivation: e.g. map all noisy samples into the same label

Associative memories

- neural nets $\mathbf{y} = \mathbf{f}(\mathbf{x})$
  - if $\mathbf{x}, \mathbf{x}'$ are close to each other then usually so are $\mathbf{y}, \mathbf{y}'$, but not equal
- associative memories $\mathbf{y} = \mathbf{f}(\mathbf{x})$
  - if $\mathbf{x}'$ is close to $\mathbf{x}$ then they are *associated* to the same $\mathbf{y}$
- motivation: e.g. map all noisy samples into the same label
- simple associative memory model:

$$\mathbf{y} = \mathbf{x}\mathbf{W}$$

where the vectors are row vectors.

Associative memories

- neural nets $\mathbf{y} = \mathbf{f}(\mathbf{x})$
  - if $\mathbf{x}, \mathbf{x}'$ are close to each other then usually so are $\mathbf{y}, \mathbf{y}'$, but not equal
- associative memories $\mathbf{y} = \mathbf{f}(\mathbf{x})$
  - if $\mathbf{x}'$ is close to $\mathbf{x}$ then they are *associated* to the same $\mathbf{y}$
- motivation: e.g. map all noisy samples into the same label
- simple associative memory model:

$$\mathbf{y} = \mathbf{x}\mathbf{W}$$

  where the vectors are row vectors.
- But this doesn't really make them associative yet, how could this be achieved?

Recurrent Associative memories

- associative memories

$$\mathbf{y} = \mathbf{x}\mathbf{W}$$

Recurrent Associative memories

- associative memories

$$\mathbf{y} = \mathbf{x}\mathbf{W}$$

- *recurrent* associative memories

$$\mathbf{x}_{t+1} = \mathbf{x}_t\mathbf{W}$$

Recurrent Associative memories

- associative memories

$$\mathbf{y} = \mathbf{x}\mathbf{W}$$

- *recurrent* associative memories

$$\mathbf{x}_{t+1} = \mathbf{x}_t\mathbf{W}$$

- for appropriate $\mathbf{W}$, all possible $\mathbf{x}_0$ converge eventually to a specific $\mathbf{x}_{target}$

Recurrent Associative memories

- associative memories

$$\mathbf{y} = \mathbf{x}\mathbf{W}$$

- *recurrent* associative memories

$$\mathbf{x}_{t+1} = \mathbf{x}_t\mathbf{W}$$

- for appropriate $\mathbf{W}$, all possible $\mathbf{x}_0$ converge eventually to a specific $\mathbf{x}_{target}$
- we say that $\mathbf{x}_{target}$ has been *stored*

Recurrent Associative memories

- associative memories
$$\mathbf{y} = \mathbf{x}\mathbf{W}$$
- *recurrent* associative memories
$$\mathbf{x}_{t+1} = \mathbf{x}_t\mathbf{W}$$
- for appropriate $\mathbf{W}$, all possible $\mathbf{x}_0$ converge eventually to a specific $\mathbf{x}_{target}$
- we say that $\mathbf{x}_{target}$ has been *stored*
- dominant eigenvector of $\mathbf{W}$ determines where everything converges to – *attractor*

Recurrent Associative memories

- associative memories

$$\mathbf{y} = \mathbf{x}\mathbf{W}$$

- *recurrent* associative memories

$$\mathbf{x}_{t+1} = \mathbf{x}_t\mathbf{W}$$

- for appropriate $\mathbf{W}$, all possible $\mathbf{x}_0$ converge eventually to a specific $\mathbf{x}_{target}$
- we say that $\mathbf{x}_{target}$ has been *stored*
- dominant eigenvector of $\mathbf{W}$ determines where everything converges to – *attractor*
- but only one vector $\mathbf{x}_{target}$ can be stored – not very useful

Recurrent Associative memories

- associative memories

$$\mathbf{y} = \mathbf{x}\mathbf{W}$$

- *recurrent* associative memories

$$\mathbf{x}_{t+1} = \mathbf{x}_t\mathbf{W}$$

- for appropriate $\mathbf{W}$, all possible $\mathbf{x}_0$ converge eventually to a specific $\mathbf{x}_{target}$
- we say that $\mathbf{x}_{target}$ has been *stored*
- dominant eigenvector of $\mathbf{W}$ determines where everything converges to – *attractor*
- but only one vector $\mathbf{x}_{target}$ can be stored – not very useful
- need nonlinearity to store multiple vectors

$$\mathbf{x}_{t+1} = \mathrm{sgn}(\mathbf{x}_t\mathbf{W})$$

Recurrent Associative memories

- associative memories

$$\mathbf{y} = \mathbf{x}\mathbf{W}$$

- *recurrent* associative memories

$$\mathbf{x}_{t+1} = \mathbf{x}_t\mathbf{W}$$

- for appropriate $\mathbf{W}$, all possible $\mathbf{x}_0$ converge eventually to a specific $\mathbf{x}_{target}$
- we say that $\mathbf{x}_{target}$ has been *stored*
- dominant eigenvector of $\mathbf{W}$ determines where everything converges to – *attractor*
- but only one vector $\mathbf{x}_{target}$ can be stored – not very useful
- need nonlinearity to store multiple vectors

$$\mathbf{x}_{t+1} = \operatorname{sgn}(\mathbf{x}_t\mathbf{W})$$

- i.e. want multiple attractor eigenvectors, each dominant in some input neighborhood

Hebbian learning

- Another approach to storing a specific vector $\mathbf{y}$ in associative memory model $\mathbf{y} = \mathrm{sgn}(\mathbf{xW})$

Hebbian learning

- Another approach to storing a specific vector $\mathbf{y}$ in associative memory model $\mathbf{y} = \text{sgn}(\mathbf{xW})$
- given we want to store $\mathbf{y}$, how do we find suitable $\mathbf{W}$

Hebbian learning

- Another approach to storing a specific vector $\mathbf{y}$ in associative memory model $\mathbf{y} = \text{sgn}(\mathbf{xW})$
- given we want to store $\mathbf{y}$, how do we find suitable $\mathbf{W}$
- Hebbian learning:

$$\mathbf{W}_0 = 0 \rightarrow \Delta w_{i,j} = x_i y_j \rightarrow \mathbf{W}_1 = \mathbf{x}^T \mathbf{y}$$

Hebbian learning

- Another approach to storing a specific vector $\mathbf{y}$ in associative memory model $\mathbf{y} = \mathrm{sgn}(\mathbf{xW})$
- given we want to store $\mathbf{y}$, how do we find suitable $\mathbf{W}$
- Hebbian learning:

$$\mathbf{W}_0 = 0 \rightarrow \Delta w_{i,j} = x_i y_j \rightarrow \mathbf{W}_1 = \mathbf{x}^T \mathbf{y}$$

- why it works:

Hebbian learning

- Another approach to storing a specific vector $\mathbf{y}$ in associative memory model $\mathbf{y} = \mathrm{sgn}(\mathbf{x}\mathbf{W})$
- given we want to store $\mathbf{y}$, how do we find suitable $\mathbf{W}$
- Hebbian learning:

$$\mathbf{W}_0 = 0 \rightarrow \Delta w_{i,j} = x_i y_j \rightarrow \mathbf{W}_1 = \mathbf{x}^T \mathbf{y}$$

- why it works:
  - assume y_1 is positive. We know that $y_1 = \mathrm{sgn}(x_1 w_{1,1} + \cdots + x_n w_{n,1})$.

Hebbian learning

- Another approach to storing a specific vector $\mathbf{y}$ in associative memory model $\mathbf{y} = \text{sgn}(\mathbf{xW})$
- given we want to store $\mathbf{y}$, how do we find suitable $\mathbf{W}$
- Hebbian learning:

$$\mathbf{W}_0 = 0 \rightarrow \Delta w_{i,j} = x_i y_j \rightarrow \mathbf{W}_1 = \mathbf{x}^T \mathbf{y}$$

- why it works:
  - assume y_1 is positive. We know that $y_1 = \text{sgn}(x_1 w_{1,1} + \cdots + x_n w_{n,1})$.
  - wish the sum inside the $\text{sgn}(\cdot)$ to be a positive number

Hebbian learning

- Another approach to storing a specific vector $\mathbf{y}$ in associative memory model $\mathbf{y} = \text{sgn}(\mathbf{xW})$
- given we want to store $\mathbf{y}$, how do we find suitable $\mathbf{W}$
- Hebbian learning:

$$\mathbf{W}_0 = 0 \rightarrow \Delta w_{i,j} = x_i y_j \rightarrow \mathbf{W}_1 = \mathbf{x}^T \mathbf{y}$$

- why it works:
  - assume y_1 is positive. We know that $y_1 = \text{sgn}(x_1 w_{1,1} + \cdots + x_n w_{n,1})$.
  - wish the sum inside the $\text{sgn}(\cdot)$ to be a positive number
  - need weights of positive x^+ to be positive and weights of negative x^- to be negative

Hebbian learning

- Another approach to storing a specific vector $\mathbf{y}$ in associative memory model $\mathbf{y} = \text{sgn}(\mathbf{xW})$
- given we want to store $\mathbf{y}$, how do we find suitable $\mathbf{W}$
- Hebbian learning:

$$\mathbf{W}_0 = 0 \rightarrow \Delta w_{i,j} = x_i y_j \rightarrow \mathbf{W}_1 = \mathbf{x}^T \mathbf{y}$$

- why it works:
  - assume y_1 is positive. We know that $y_1 = \text{sgn}(x_1 w_{1,1} + \cdots + x_n w_{n,1})$.
  - wish the sum inside the $\text{sgn}(\cdot)$ to be a positive number
  - need weights of positive x^+ to be positive and weights of negative x^- to be negative
- But again, wish to store more than one vector.

Hebbian learning

- Another approach to storing a specific vector $\mathbf{y}$ in associative memory model $\mathbf{y} = \operatorname{sgn}(\mathbf{xW})$
- given we want to store $\mathbf{y}$, how do we find suitable $\mathbf{W}$
- Hebbian learning:

$$\mathbf{W}_0 = 0 \rightarrow \Delta w_{i,j} = x_i y_j \rightarrow \mathbf{W}_1 = \mathbf{x}^T \mathbf{y}$$

- why it works:
  - assume y_1 is positive. We know that $y_1 = \operatorname{sgn}(x_1 w_{1,1} + \cdots + x_n w_{n,1})$.
  - wish the sum inside the $\operatorname{sgn}(\cdot)$ to be a positive number
  - need weights of positive x^+ to be positive and weights of negative x^- to be negative
- But again, wish to store more than one vector.
- To do this, for $\mathbf{x}_1, \ldots, \mathbf{x}_m$, do hebbian learning on each vector and then $\mathbf{W} = \mathbf{W}^1 + \cdots + \mathbf{W}^m$

Hebbian learning

- Another approach to storing a specific vector $\mathbf{y}$ in associative memory model $\mathbf{y} = \operatorname{sgn}(\mathbf{x}\mathbf{W})$
- given we want to store $\mathbf{y}$, how do we find suitable $\mathbf{W}$
- Hebbian learning:

$$\mathbf{W}_0 = 0 \rightarrow \Delta w_{i,j} = x_i y_j \rightarrow \mathbf{W}_1 = \mathbf{x}^T \mathbf{y}$$

- why it works:
  - assume y_1 is positive. We know that $y_1 = \operatorname{sgn}(x_1 w_{1,1} + \cdots + x_n w_{n,1})$.
  - wish the sum inside the $\operatorname{sgn}(\cdot)$ to be a positive number
  - need weights of positive x^+ to be positive and weights of negative x^- to be negative
- But again, wish to store more than one vector.
- To do this, for $\mathbf{x}_1, \ldots, \mathbf{x}_m$, do hebbian learning on each vector and then $\mathbf{W} = \mathbf{W}^1 + \cdots + \mathbf{W}^m$
- But this may not always work perfectly as it "mixes up" the different matrices (HW Q1.; Rojas p.317–)

Bidirectional Associative Memories (BAM)

- previously:

Bidirectional Associative Memories (BAM)

- previously:
  - associative memories

$$\mathbf{y} = \operatorname{sgn}(\mathbf{x}\mathbf{W})$$

Bidirectional Associative Memories (BAM)

- previously:
  - associative memories

$$\mathbf{y} = \operatorname{sgn}(\mathbf{x}\mathbf{W})$$

  - recurrent associative memories:

$$\mathbf{x}_{t+1} = \operatorname{sgn}(\mathbf{x}_t\mathbf{W})$$

Bidirectional Associative Memories (BAM)

- previously:
  - associative memories

$$\mathbf{y} = \operatorname{sgn}(\mathbf{x}\mathbf{W})$$

  - recurrent associative memories:

$$\mathbf{x}_{t+1} = \operatorname{sgn}(\mathbf{x}_t\mathbf{W})$$

  - *auto*associative memories:

$$\mathbf{x} = \mathbf{x}\mathbf{W}$$

Bidirectional Associative Memories (BAM)

- previously:
  - associative memories

$$\mathbf{y} = \operatorname{sgn}(\mathbf{x}\mathbf{W})$$

  - recurrent associative memories:

$$\mathbf{x}_{t+1} = \operatorname{sgn}(\mathbf{x}_t\mathbf{W})$$

  - *auto*associative memories:

$$\mathbf{x} = \mathbf{x}\mathbf{W}$$

- now, *bidirectional* associative memories

$$\begin{aligned}
\mathbf{y}_t &= \operatorname{sgn}(\mathbf{x}_t\mathbf{W}) \\
\mathbf{x}_{t+1}^T &= \operatorname{sgn}(\mathbf{W}\mathbf{y}_t^T) \\
\mathbf{y}_{t+1} &= \operatorname{sgn}(\mathbf{x}_{t+1}\mathbf{W}) \\
&\vdots
\end{aligned}$$

BAM stable state & learning

- just like with recurrent associative memories, want BAM to converge to stable state:

$$\mathbf{y}^T = \operatorname{sgn}(\mathbf{x}\mathbf{W}) \quad \mathbf{x}^T = \operatorname{sgn}(\mathbf{W}\mathbf{y}^T)$$

BAM stable state & learning

- just like with recurrent associative memories, want BAM to converge to stable state:

$$\mathbf{y}^T = \mathrm{sgn}(\mathbf{x}\mathbf{W}) \quad \mathbf{x}^T = \mathrm{sgn}(\mathbf{W}\mathbf{y}^T)$$

- we want to store some $(\mathbf{x}, \mathbf{y})$, how do we find weight $\mathbf{W}$ to give this pair as stable state

BAM stable state & learning

- just like with recurrent associative memories, want BAM to converge to stable state:

$$\mathbf{y}^T = \operatorname{sgn}(\mathbf{x}\mathbf{W}) \quad \mathbf{x}^T = \operatorname{sgn}(\mathbf{W}\mathbf{y}^T)$$

- we want to store some $(\mathbf{x}, \mathbf{y})$, how do we find weight $\mathbf{W}$ to give this pair as stable state
- Hebbian learning again $\mathbf{W}_0 = \mathbf{0} \to \mathbf{W} = \mathbf{x}^T\mathbf{y}$ since:

$$\mathbf{y} = \operatorname{sgn}(\mathbf{x}\mathbf{W}) = \operatorname{sgn}(\mathbf{x}\mathbf{x}^T\mathbf{y}) = \operatorname{sgn}(\|\mathbf{x}\|^2\,\mathbf{y}) = \mathbf{y}$$
$$\mathbf{x}^T = \operatorname{sgn}(\mathbf{W}\mathbf{y}^T) = \operatorname{sgn}(\mathbf{x}^T\mathbf{y}\mathbf{y}^T) = \operatorname{sgn}(\mathbf{x}^T\|\mathbf{y}\|^2) = \mathbf{x}^T$$

notice that $\mathbf{W}$ is symmetric.

BAM stable state & learning

- just like with recurrent associative memories, want BAM to converge to stable state:

$$\mathbf{y}^T = \operatorname{sgn}(\mathbf{x}\mathbf{W}) \quad \mathbf{x}^T = \operatorname{sgn}(\mathbf{W}\mathbf{y}^T)$$

- we want to store some $(\mathbf{x}, \mathbf{y})$, how do we find weight $\mathbf{W}$ to give this pair as stable state
- Hebbian learning again $\mathbf{W}_0 = \mathbf{0} \rightarrow \mathbf{W} = \mathbf{x}^T\mathbf{y}$ since:

$$\begin{aligned}\mathbf{y} &= \operatorname{sgn}(\mathbf{x}\mathbf{W}) = \operatorname{sgn}(\mathbf{x}\mathbf{x}^T\mathbf{y}) = \operatorname{sgn}(\|\mathbf{x}\|^2\,\mathbf{y}) = \mathbf{y}\\ \mathbf{x}^T &= \operatorname{sgn}(\mathbf{W}\mathbf{y}^T) = \operatorname{sgn}(\mathbf{x}^T\mathbf{y}\mathbf{y}^T) = \operatorname{sgn}(\mathbf{x}^T\|\mathbf{y}\|^2) = \mathbf{x}^T\end{aligned}$$

  notice that $\mathbf{W}$ is symmetric.
- To store multiple pairs $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)$, do hebbian learning on each vector and then $\mathbf{W} = \mathbf{W}^1 + \cdots + \mathbf{W}^m$

Energy function

- BAM stable state

$$
\begin{aligned}
\mathbf{y}_0 &= \operatorname{sgn}(\mathbf{x}_0 \mathbf{W}) \\
\mathbf{x}_0^T &= \operatorname{sgn}(\mathbf{W}\mathbf{y}_0^T) \\
&= \operatorname{sgn}(\mathbf{e}^T)
\end{aligned}
$$

Energy function

- BAM stable state

$$\begin{aligned} \mathbf{y}_0 &= \mathrm{sgn}(\mathbf{x}_0 \mathbf{W}) \\ \mathbf{x}_0^T &= \mathrm{sgn}(\mathbf{W}\mathbf{y}_0^T) \\ &= \mathrm{sgn}(\mathbf{e}^T) \end{aligned}$$

- so want vectors $\mathbf{e}$ and $\mathbf{x}_0$ to be as close to each other as possible.

Energy function

- BAM stable state

$$
\begin{aligned}
\mathbf{y}_0 &= \operatorname{sgn}(\mathbf{x}_0\mathbf{W}) \\
\mathbf{x}_0^T &= \operatorname{sgn}(\mathbf{W}\mathbf{y}_0^T) \\
&= \operatorname{sgn}(\mathbf{e}^T)
\end{aligned}
$$

- so want vectors $\mathbf{e}$ and $\mathbf{x}_0$ to be as close to each other as possible.
- can measure the angle between two vectors with the dot product:

$$
\mathbf{x}\mathbf{e}^T = \|\mathbf{x}\|\|\mathbf{e}\| \cos\theta_{\mathbf{x},\mathbf{e}} \tag{1}
$$

so by maximizing $\mathbf{x}\mathbf{e}^T$ we minimize θ ($\cos\theta$ goes up)

Energy function

- BAM stable state

$$
\begin{aligned}
\mathbf{y}_0 &= \operatorname{sgn}(\mathbf{x}_0 \mathbf{W}) \\
\mathbf{x}_0^T &= \operatorname{sgn}(\mathbf{W}\mathbf{y}_0^T) \\
&= \operatorname{sgn}(\mathbf{e}^T)
\end{aligned}
$$

- so want vectors $\mathbf{e}$ and $\mathbf{x}_0$ to be as close to each other as possible.
- can measure the angle between two vectors with the dot product:

$$\mathbf{x}\mathbf{e}^T = \|\mathbf{x}\|\|\mathbf{e}\| \cos\theta_{\mathbf{x},\mathbf{e}} \tag{1}$$

  so by maximizing $\mathbf{x}\mathbf{e}^T$ we minimize θ ($\cos\theta$ goes up)
- $\max \mathbf{x}\mathbf{e}^T = \min -\mathbf{x}\mathbf{e}^T = \min -\mathbf{x}\mathbf{W}\mathbf{y}^T$

Energy function

- BAM stable state

$$\begin{aligned}\mathbf{y}_0 &= \mathrm{sgn}(\mathbf{x}_0\mathbf{W})\\ \mathbf{x}_0^T &= \mathrm{sgn}(\mathbf{W}\mathbf{y}_0^T)\\ &= \mathrm{sgn}(\mathbf{e}^T)\end{aligned}$$

- so want vectors $\mathbf{e}$ and $\mathbf{x}_0$ to be as close to each other as possible.
- can measure the angle between two vectors with the dot product:

$$\mathbf{x}\mathbf{e}^T = \|\mathbf{x}\|\|\mathbf{e}\|\cos\theta_{\mathbf{x},\mathbf{e}} \tag{1}$$

  so by maximizing $\mathbf{x}\mathbf{e}^T$ we minimize θ ($\cos\theta$ goes up)
- $\max \mathbf{x}\mathbf{e}^T = \min -\mathbf{x}\mathbf{e}^T = \min -\mathbf{x}\mathbf{W}\mathbf{y}^T$
- thus Energy, $E(\mathbf{x},\mathbf{y}) = -\mathbf{x}\mathbf{W}\mathbf{y}^T$, is minimized at stable state

Energy function

- BAM stable state

$$\begin{aligned} \mathbf{y}_0 &= \operatorname{sgn}(\mathbf{x}_0 \mathbf{W}) \\ \mathbf{x}_0^T &= \operatorname{sgn}(\mathbf{W}\mathbf{y}_0^T) \\ &= \operatorname{sgn}(\mathbf{e}^T) \end{aligned}$$

- so want vectors $\mathbf{e}$ and $\mathbf{x}_0$ to be as close to each other as possible.
- can measure the angle between two vectors with the dot product:

$$\mathbf{x}\mathbf{e}^T = \|\mathbf{x}\| \|\mathbf{e}\| \cos \theta_{\mathbf{x},\mathbf{e}} \tag{1}$$

  so by maximizing $\mathbf{x}\mathbf{e}^T$ we minimize θ ($\cos\theta$ goes up)
- $\max \mathbf{x}\mathbf{e}^T = \min -\mathbf{x}\mathbf{e}^T = \min -\mathbf{x}\mathbf{W}\mathbf{y}^T$
- thus Energy, $E(\mathbf{x}, \mathbf{y}) = -\mathbf{x}\mathbf{W}\mathbf{y}^T$, is minimized at stable state
- each iteration that is not at local stable state / minimum, reduces energy (Rojas pp.343-)

Hopfield networks

- BAM units are synchronous as $\text{sgn}(\mathbf{xW})$ uses $\mathbf{W}$ in one go

Hopfield networks

- BAM units are synchronous as $\mathrm{sgn}(\mathbf{x}\mathbf{W})$ uses $\mathbf{W}$ in one go
- *Asynchronous* networks run only one unit at a time

$$y_i = \mathrm{sgn}(\mathbf{x}\mathbf{w}_i^T)$$

for some $i \in \{1, \ldots, K\}$ chosen randomly and uniformly.

Hopfield networks

- BAM units are synchronous as $\text{sgn}(\mathbf{xW})$ uses $\mathbf{W}$ in one go
- *Asynchronous* networks run only one unit at a time

$$y_i = \text{sgn}(\mathbf{x}\mathbf{w}_i^T)$$

  for some $i \in \{1, \ldots, K\}$ chosen randomly and uniformly.
- Hopfield network is asynchronous BAM with with fully coupled units (nonzero weights between units) except each unit is disconnected from itself:

$$\mathbf{W} = \begin{pmatrix} 0 & w_{1,2} & w_{1,3} \\ w_{2,1} & 0 & w_{2,3} \\ w_{3,1} & w_{3,2} & 0 \end{pmatrix}$$

Hopfield networks

- BAM units are synchronous as $\operatorname{sgn}(\mathbf{xW})$ uses $\mathbf{W}$ in one go
- *Asynchronous* networks run only one unit at a time

$$y_i = \operatorname{sgn}(\mathbf{x}\mathbf{w}_i^T)$$

  for some $i \in \{1, \ldots, K\}$ chosen randomly and uniformly.
- Hopfield network is asynchronous BAM with with fully coupled units (nonzero weights between units) except each unit is disconnected from itself:

$$\mathbf{W} = \begin{pmatrix} 0 & w_{1,2} & w_{1,3} \\ w_{2,1} & 0 & w_{2,3} \\ w_{3,1} & w_{3,2} & 0 \end{pmatrix}$$

- threshold λ often included in the nonlinearity

Hopfield networks

- BAM units are synchronous as $\text{sgn}(\mathbf{x}\mathbf{W})$ uses $\mathbf{W}$ in one go
- *Asynchronous* networks run only one unit at a time

$$y_i = \text{sgn}(\mathbf{x}\mathbf{w}_i^T)$$

  for some $i \in \{1, \ldots, K\}$ chosen randomly and uniformly.

- Hopfield network is asynchronous BAM with with fully coupled units (nonzero weights between units) except each unit is disconnected from itself:

$$\mathbf{W} = \begin{pmatrix} 0 & w_{1,2} & w_{1,3} \\ w_{2,1} & 0 & w_{2,3} \\ w_{3,1} & w_{3,2} & 0 \end{pmatrix}$$

- threshold λ often included in the nonlinearity
- energy function $-\frac{1}{2}\mathbf{x}\mathbf{W}\mathbf{x}^T + \lambda\mathbf{x}^T$