Structured Clique Networks as Efficient Associative Memories

Vincent Gripon

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

BRAIn

Lab-STICC

March 4th, 2021

Outline

Outline

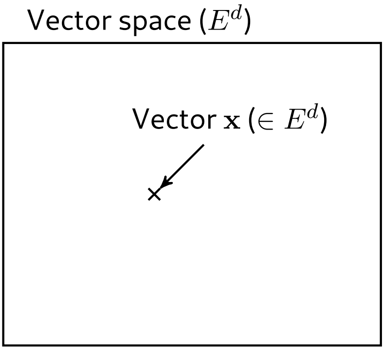
1. Context
2. Associative Memories
   - Hopfield Neural Networks
   - Willshaw Neural Networks
3. Structured Clique Networks
   - Principles
   - Theoretical results
   - Experiments
4. Example Applications
   - Applications in Hardware
   - Approximate Nearest Neighbor Search
   - DecisiveNets
5. Conclusion

Notations and problems

Vector space (E^d)

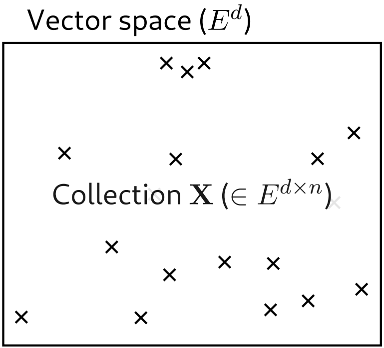
1. Supervised learning,
2. Unsupervised learning,
3. Indexing,
4. Search…

Notations and problems



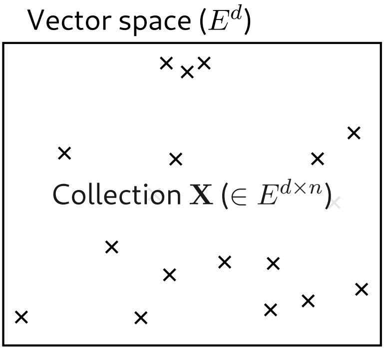
1. Supervised learning,
2. Unsupervised learning,
3. Indexing,
4. Search…

Notations and problems



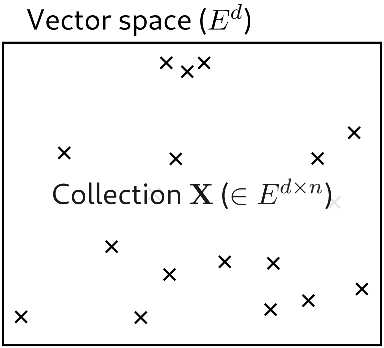
1. Supervised learning,
2. Unsupervised learning,
3. Indexing,
4. Search…

Notations and problems



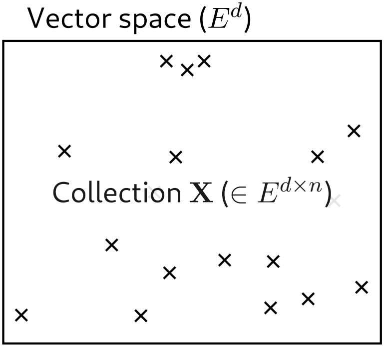
1. Supervised learning,
2. Unsupervised learning,
3. Indexing,
4. Search...

Notations and problems



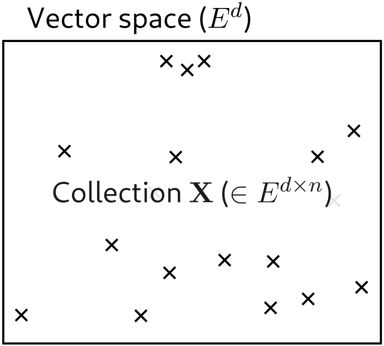
1. Supervised learning,
2. Unsupervised learning,
3. Indexing,
4. Search...

Notations and problems



1. Supervised learning,
2. Unsupervised learning,
3. Indexing,
4. Search. . .

Notations and problems



1. Supervised learning,
2. Unsupervised learning,
3. Indexing,
4. Search...

Supervised learning

Learning

To learn is to **generalize** ($\neq$ memorize),

Supervised learning

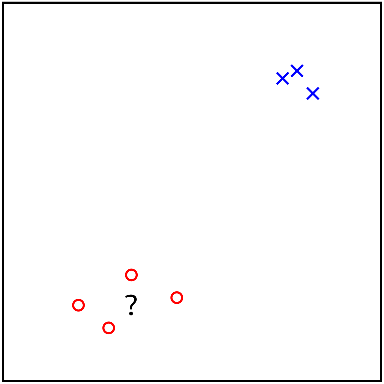
- Regression,
- Requires an expert,
- Tons of applications.

Learning

To learn is to **generalize** ($\neq$ memorize),

Supervised learning

- Regression,
- Requires an expert,
- Tons of applications.



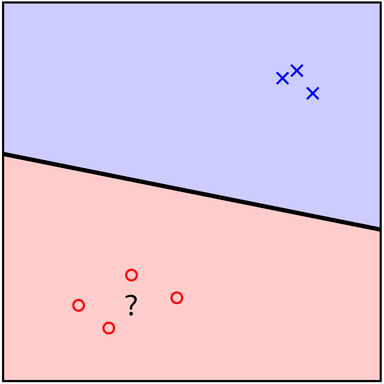
Supervised learning

Learning

To learn is to **generalize** ($\neq$ memorize),

Supervised learning

- Regression,
- Requires an expert,
- Tons of applications.



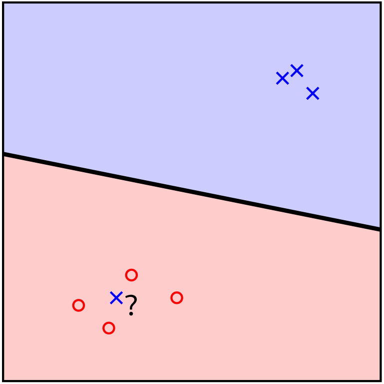
Supervised learning

Learning

To learn is to **generalize** ($\neq$ memorize),

Supervised learning

- Regression,
- Requires an expert,
- Tons of applications.



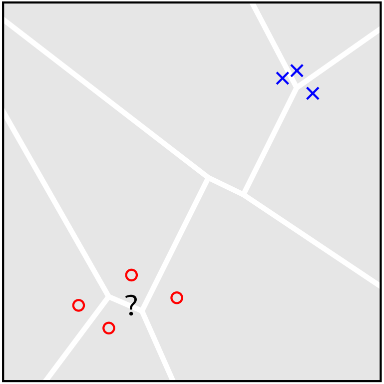
Supervised learning

Learning

To learn is to **generalize** ($\neq$ memorize),

Supervised learning

- Regression,
- Requires an expert,
- Tons of applications.



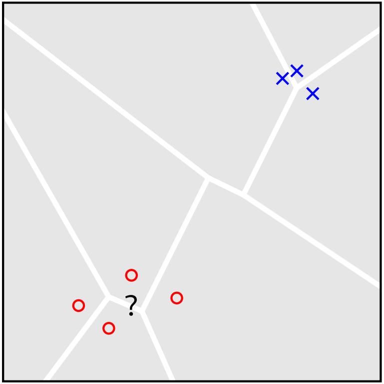
Supervised learning

Learning

To learn is to **generalize** ($\neq$ memorize),

Supervised learning

- Regression,
- Requires an expert,
- Tons of applications.



Classical methods: SVM, k-NN, Random Forests, LR, MLP, CNN...

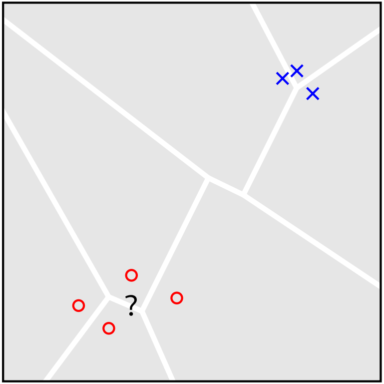
Supervised learning

Learning

To learn is to **generalize** ($\neq$ memorize),

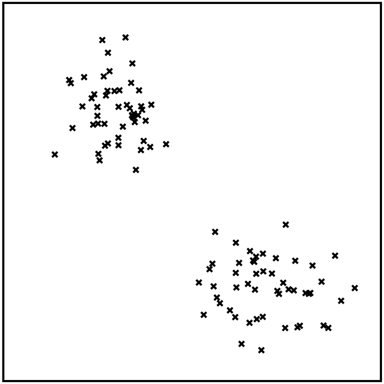
Supervised learning

- Regression,
- Requires an expert,
- Tons of applications.



Classical methods: SVM, k-NN, Random Forests, LR, MLP, CNN...

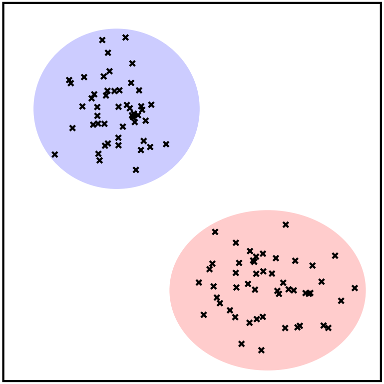
Unsupervised learning



Unsupervised learning

- Partitioning/disentangling,
- Requires priors,
- Not so many direct applications.

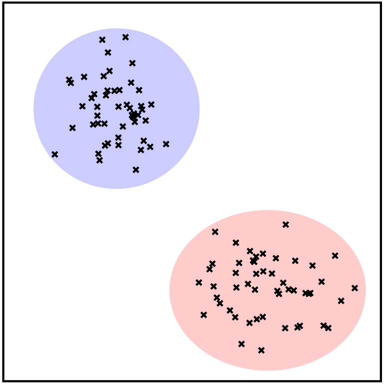
Unsupervised learning



Unsupervised learning

- Partitioning/disentangling,
- Requires priors,
- Not so many direct applications.

Unsupervised learning

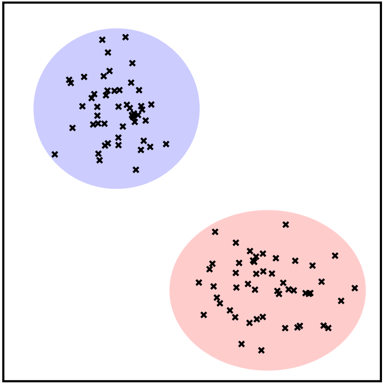


Unsupervised learning

- Partitioning/disentangling,
- Requires priors,
- Not so many direct applications.

Classical methods: k-means, db-scan, Kohonen maps, autoencoders, EM...

Unsupervised learning



Unsupervised learning

- Partitioning/disentangling,
- Requires priors,
- Not so many direct applications.

Classical methods: k-means, db-scan, Kohonen maps, autoencoders, EM...

Indexing

Definition

Given a collection $\mathbf{X} \in E^{d \times n}$ and a query vector $\mathbf{x}$:

1. Is $\mathbf{x} \in \mathbf{X}$?
2. Do we have $\mathbf{x}' \in \mathbf{X}$ s.t. $\mathbf{x}' \approx \mathbf{x}$?

Exhaustive search

- Pros: no error, simple, concurrent,
- Cons: linear with both d and n.

Example of database: SIFT1B:

- $n = 1,000,000,000$,
- $d = 128$,
- 10,000 queries,
- On my laptop, takes approximatively 4 years.

Indexing

Definition

Given a collection $\mathbf{X} \in E^{d \times n}$ and a query vector $\mathbf{x}$:

1. Is $\mathbf{x} \in \mathbf{X}$?
2. Do we have $\mathbf{x}' \in \mathbf{X}$ s.t. $\mathbf{x}' \approx \mathbf{x}$?

Exhaustive search

- Pros: no error, simple, concurrent,
- Cons: linear with both d and n.

Example of database: SIFT1B:

- $n = 1,000,000,000$,
- $d = 128$,
- 10,000 queries,
- On my laptop, takes approximatively 4 years.

Indexing

Definition

Given a collection $\mathbf{X} \in E^{d \times n}$ and a query vector $\mathbf{x}$:

1. Is $\mathbf{x} \in \mathbf{X}$?
2. Do we have $\mathbf{x}' \in \mathbf{X}$ s.t. $\mathbf{x}' \approx \mathbf{x}$?

Exhaustive search

- Pros: no error, simple, concurrent,
- Cons: linear with both d and n.

Example of database: SIFT1B:

- $n = 1,000,000,000$,
- $d = 128$,
- 10,000 queries,
- On my laptop, takes approximatively 4 years.

Indexing

Definition

Given a collection $\mathbf{X} \in E^{d \times n}$ and a query vector $\mathbf{x}$:

1. Is $\mathbf{x} \in \mathbf{X}$?
2. Do we have $\mathbf{x}' \in \mathbf{X}$ s.t. $\mathbf{x}' \approx \mathbf{x}$?

Exhaustive search

- Pros: no error, simple, concurrent,
- Cons: linear with both d and n.

Example of database: SIFT1B:

- $n = 1,000,000,000$,
- $d = 128$,
- 10,000 queries,
- On my laptop, takes approximatively 4 years.

Indexing

Definition

Given a collection $\mathbf{X} \in E^{d \times n}$ and a query vector $\mathbf{x}$:

1. Is $\mathbf{x} \in \mathbf{X}$?
2. Do we have $\mathbf{x}' \in \mathbf{X}$ s.t. $\mathbf{x}' \approx \mathbf{x}$?

Exhaustive search

- Pros: no error, simple, concurrent,
- Cons: linear with both d and n.

Example of database: SIFT1B:

- $n = 1,000,000,000$,
- $d = 128$,
- 10,000 queries,
- On my laptop, takes approximatively 4 years.

Search

Definition

Given a collection $\mathbf{X} \in E^{d \times n}$ and a query vector $\mathbf{x}$, find:

$$\mathbf{x}' = \arg\min_{\mathbf{x}' \in \mathbf{X}} \|\mathbf{x} - \mathbf{x}'\|,$$

given some metric.

Methods

- Exhaustive search again,
- Act on n and/or d:
  - On n, partition the search space (problems with high dimensions),
  - On d, quantify the collection and/or the probe (e.g. Product Quantization).

Search

Definition

Given a collection $\mathbf{X} \in E^{d \times n}$ and a query vector $\mathbf{x}$, find:

$$\mathbf{x}' = \arg\min_{\mathbf{x}' \in \mathbf{X}} \|\mathbf{x} - \mathbf{x}'\|,$$

given some metric.

Methods

- Exhaustive search again,
- Act on n and/or d:
  - On n, partition the search space (problems with high dimensions),
  - On d, quantify the collection and/or the probe (e.g. Product Quantization).

(Artificial) Neural Networks

Definition

A neural network is a mathematical function obtained by assembling simple ones, called layers, that can be written as:

$$\mathbf{x} \mapsto \mathbf{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}).$$

Nonlinear functions

- Sigmoids (e.g. $x \mapsto 1/\left(1 + \exp(-x)\right)$),
- ReLU (e.g. $x \mapsto \max(0, x)$),
- Winner-Take-All (WTA)…

(Artificial) Neural Networks

Definition

A neural network is a mathematical function obtained by assembling simple ones, called layers, that can be written as:

$$\mathbf{x} \mapsto \mathbf{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}).$$

Nonlinear functions

- Sigmoids (e.g. $x \mapsto 1/\left(1 + \exp(-x)\right)$),
- ReLU (e.g. $x \mapsto \max(0, x)$),
- Winner-Take-All (WTA)...

Outline

Framework

- $\mathbf{x} \in \{-1, 1\}^d$, $\mathbf{X} \subset \{-1, 1\}^{d \times n}$,

- Example:
  - Storing binary vector -11-111-1-11
  - Retrieve it from -11-111-1?1
- $\mathbf{W} = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{x}\mathbf{x}^\top = \mathbf{X}\mathbf{X}^\top$ (except diagonal),
- $\mathbf{y} = sgn(\mathbf{W}\mathbf{x}) = u(\mathbf{x})$,
- $U(\mathbf{x}) \triangleq u(u(u(u(\dots u(\mathbf{x})))))$,
- Complexity: $\mathcal{O}(d^2)$.

Framework

- $\mathbf{x} \in \{-1,1\}^d, \mathbf{X} \subset \{-1,1\}^{d \times n}$,

- Example:
  - Storing binary vector -11-111-1-11
  - Retrieve it from -11-111-1?1
- $\mathbf{W} = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{x}\mathbf{x}^\top = \mathbf{X}\mathbf{X}^\top$ (except diagonal),
- $\mathbf{y} = sgn(\mathbf{W}\mathbf{x}) = u(\mathbf{x})$,
- $U(\mathbf{x}) \triangleq u(u(u(u(\ldots u(\mathbf{x})))))$,
- Complexity: $\mathcal{O}(d^2)$.

Framework

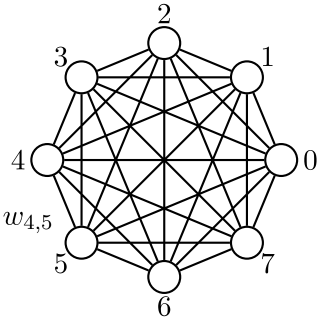
- $\mathbf{x} \in \{-1,1\}^d$, $\mathbf{X} \subset \{-1,1\}^{d \times n}$,

- Example:
  - Storing binary vector -11-111-1-11
  - Retrieve it from -11-111-1?1
- $\mathbf{W} = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{x}\mathbf{x}^\top = \mathbf{X}\mathbf{X}^\top$ (except diagonal),
- $\mathbf{y} = sgn(\mathbf{W}\mathbf{x}) = u(\mathbf{x})$,
- $U(\mathbf{x}) \triangleq u(u(u(u(\dots u(\mathbf{x})))))$,
- Complexity: $\mathcal{O}(d^2)$.

Framework

- $\mathbf{x} \in \{-1, 1\}^d$, $\mathbf{X} \subset \{-1, 1\}^{d \times n}$,

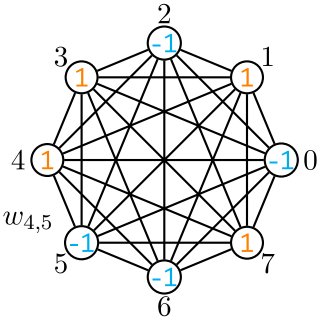
- Example:
  - Storing binary vector -11-111-1-11
  - Retrieve it from -11-111-1?1
- $\mathbf{W} = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{x}\mathbf{x}^\top = \mathbf{X}\mathbf{X}^\top$ (except diagonal),
- $\mathbf{y} = sgn(\mathbf{W}\mathbf{x}) = u(\mathbf{x})$,
- $U(\mathbf{x}) \triangleq u(u(u(u(\dots u(\mathbf{x})))))$,
- Complexity: $\mathcal{O}(d^2)$.



Framework

- $\mathbf{x} \in \{-1,1\}^d$, $\mathbf{X} \subset \{-1,1\}^{d \times n}$,

- Example:
  - Storing binary vector -11-111-1-11
  - Retrieve it from -11-111-1?1
- $\mathbf{W} = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{x}\mathbf{x}^\top = \mathbf{X}\mathbf{X}^\top$ (except diagonal),
- $\mathbf{y} = sgn(\mathbf{W}\mathbf{x}) = u(\mathbf{x})$,
- $U(\mathbf{x}) \triangleq u(u(u(u(\dots u(\mathbf{x})))))$,
- Complexity: $\mathcal{O}(d^2)$.

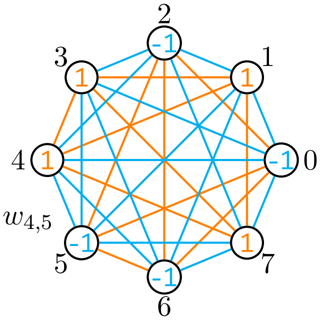


Hopfield Neural Networks

Framework

- $\mathbf{x} \in \{-1,1\}^d$, $\mathbf{X} \subset \{-1,1\}^{d \times n}$,

- Example:
  - Storing binary vector -11-111-1-11
  - Retrieve it from -11-111-1?1
- $\mathbf{W} = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{x}\mathbf{x}^\top = \mathbf{X}\mathbf{X}^\top$ (except diagonal),
- $\mathbf{y} = sgn(\mathbf{W}\mathbf{x}) = u(\mathbf{x})$,
- $U(\mathbf{x}) \triangleq u(u(u(u(\dots u(\mathbf{x})))))$,
- Complexity: $\mathcal{O}(d^2)$.

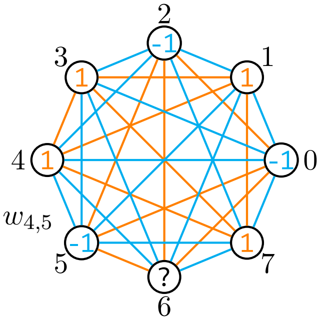


Hopfield Neural Networks

Framework

- $\mathbf{x} \in \{-1,1\}^d$, $\mathbf{X} \subset \{-1,1\}^{d \times n}$,

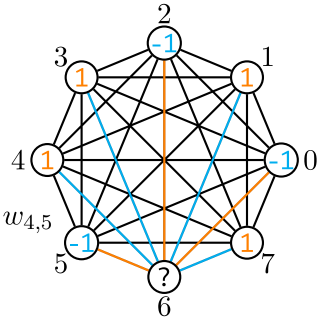
- Example:
  - Storing binary vector -11-111-1-11
  - Retrieve it from -11-111-1?1
- $\mathbf{W} = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{x}\mathbf{x}^\top = \mathbf{X}\mathbf{X}^\top$ (except diagonal),
- $\mathbf{y} = sgn(\mathbf{W}\mathbf{x}) = u(\mathbf{x})$,
- $U(\mathbf{x}) \triangleq u(u(u(u(\dots u(\mathbf{x})))))$,
- Complexity: $\mathcal{O}(d^2)$.



Framework

- $\mathbf{x} \in \{-1, 1\}^d$, $\mathbf{X} \subset \{-1, 1\}^{d \times n}$,

- Example:
  - Storing binary vector -11-111-1-11
  - Retrieve it from -11-111-1?1
- $\mathbf{W} = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{x}\mathbf{x}^\top = \mathbf{X}\mathbf{X}^\top$ (except diagonal),
- $\mathbf{y} = sgn(\mathbf{W}\mathbf{x}) = u(\mathbf{x})$,
- $U(\mathbf{x}) \triangleq u(u(u(u(\dots u(\mathbf{x})))))$,
- Complexity: $\mathcal{O}(d^2)$.

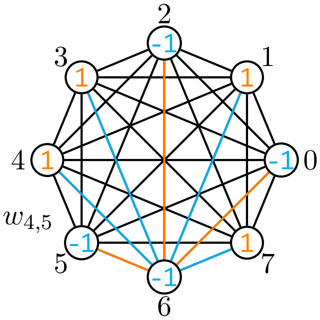


Hopfield Neural Networks

Framework

- $\mathbf{x} \in \{-1, 1\}^d$, $\mathbf{X} \subset \{-1, 1\}^{d \times n}$,

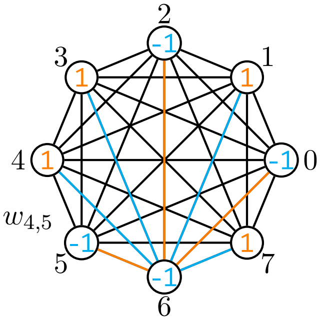
- Example:
  - Storing binary vector -11-111-1-11
  - Retrieve it from -11-111-1-11
- $\mathbf{W} = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{x}\mathbf{x}^\top = \mathbf{X}\mathbf{X}^\top$ (except diagonal),
- $\mathbf{y} = sgn(\mathbf{W}\mathbf{x}) = u(\mathbf{x})$,
- $U(\mathbf{x}) \triangleq u(u(u(u(\dots u(\mathbf{x})))))$,
- Complexity: $\mathcal{O}(d^2)$.



Framework

- $\mathbf{x} \in \{-1,1\}^d$, $\mathbf{X} \subset \{-1,1\}^{d\times n}$,

- Example:
  - Storing binary vector -11-111-1-11
  - Retrieve it from -11-111-1-11
- $\mathbf{W} = \sum_{\mathbf{x}\in\mathbf{X}} \mathbf{x}\mathbf{x}^\top = \mathbf{X}\mathbf{X}^\top$ (except diagonal),
- $\mathbf{y} = sgn(\mathbf{W}\mathbf{x}) = u(\mathbf{x})$,
- $U(\mathbf{x}) \triangleq u(u(u(u(\dots u(\mathbf{x})))))$,
- Complexity: $\mathcal{O}(d^2)$.

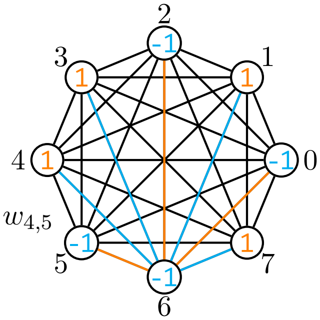


Hopfield Neural Networks

Framework

- $\mathbf{x} \in \{-1,1\}^d$, $\mathbf{X} \subset \{-1,1\}^{d\times n}$,

- Example:
  - Storing binary vector -11-111-1-11
  - Retrieve it from -11-111-1-11
- $\mathbf{W} = \sum_{\mathbf{x}\in\mathbf{X}} \mathbf{x}\mathbf{x}^\top = \mathbf{X}\mathbf{X}^\top$ (except diagonal),
- $\mathbf{y} = sgn(\mathbf{W}\mathbf{x}) = u(\mathbf{x})$,
- $U(\mathbf{x}) \triangleq u(u(u(u(\dots u(\mathbf{x})))))$,
- Complexity: $\mathcal{O}(d^2)$.

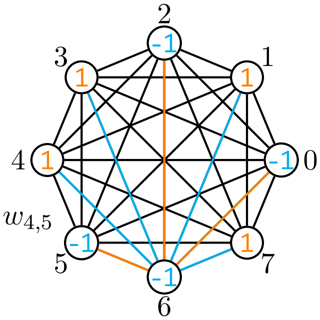


Hopfield Neural Networks

Framework

- $\mathbf{x} \in \{-1, 1\}^d$, $\mathbf{X} \subset \{-1, 1\}^{d \times n}$,

- Example:
  - Storing binary vector -11-111-1-11
  - Retrieve it from -11-111-1-11
- $\mathbf{W} = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{x}\mathbf{x}^\top = \mathbf{X}\mathbf{X}^\top$ (except diagonal),
- $\mathbf{y} = sgn(\mathbf{W}\mathbf{x}) = u(\mathbf{x})$,
- $U(\mathbf{x}) \triangleq u(u(u(u(\dots u(\mathbf{x})))))$,
- Complexity: $\mathcal{O}(d^2)$.

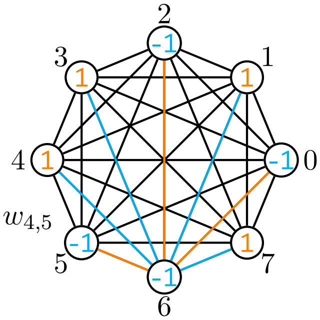


Hopfield Neural Networks

Framework

- $\mathbf{x} \in \{-1,1\}^d$, $\mathbf{X} \subset \{-1,1\}^{d \times n}$,

- Example:
  - Storing binary vector -11-111-1-11
  - Retrieve it from -11-111-1-11
- $\mathbf{W} = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{x}\mathbf{x}^\top = \mathbf{X}\mathbf{X}^\top$ (except diagonal),
- $\mathbf{y} = sgn(\mathbf{W}\mathbf{x}) = u(\mathbf{x})$,
- $U(\mathbf{x}) \triangleq u(u(u(u(\dots u(\mathbf{x})))))$,
- Complexity: $\mathcal{O}(d^2)$.



Stability of stored vectors

Theorem [1]

Consider $n = \frac{d}{\gamma \log(d)}$:

- If $\gamma > 6$, then for $d \to \infty$, $\mathbb{P}[\liminf_d \{\cap_{\mathbf{x} \in \mathbf{X}} \{U(\mathbf{x}) = \mathbf{x}\}] = 1$,
- If $\gamma > 4$, then $\mathbb{P}[\cap_{\mathbf{x}} \{U(\mathbf{x}) = \mathbf{x}\}] \to 1$,
- If $\gamma < 2$, then $\mathbb{P}[\cap_{\mathbf{x}} \{U(\mathbf{x}) = \mathbf{x}\}] \to 0$.

Memory efficiency

- $\binom{d}{2}$ connections with $n+1$ possible values each $\Rightarrow$ takes $\binom{d}{2} \log_2(n+1)$ bits without compression,
- To be compared to the entropy of $\mathbf{X} \approx nd$,
- When patterns are stable, we obtain $\eta \leq \frac{1}{\log(d)}$.

[1] "Étude asymptotique d'un réseau neuronal: le modèle de mémoire associative de Hopfield", Franck Vermet

Stability of stored vectors

Theorem [1]

Consider $n = \frac{d}{\gamma \log(d)}$:

- If $\gamma > 6$, then for $d \to \infty$, $\mathbb{P}[\liminf_d \{\cap_{\mathbf{x} \in \mathbf{X}} \{U(\mathbf{x}) = \mathbf{x}\}] = 1$,
- If $\gamma > 4$, then $\mathbb{P}[\cap_{\mathbf{x}} \{U(\mathbf{x}) = \mathbf{x}\}] \to 1$,
- If $\gamma < 2$, then $\mathbb{P}[\cap_{\mathbf{x}} \{U(\mathbf{x}) = \mathbf{x}\}] \to 0$.

Memory efficiency

- $\binom{d}{2}$ connections with $n+1$ possible values each $\Rightarrow$ takes $\binom{d}{2} \log_2(n+1)$ bits without compression,
- To be compared to the entropy of $\mathbf{X} \approx nd$,
- When patterns are stable, we obtain $\eta \leq \frac{1}{\log(d)}$.

[1] "Étude asymptotique d'un réseau neuronal: le modèle de mémoire associative de Hopfield", Franck Vermet

Framework

- $\mathbf{x} \in \{0,1\}^d$, $\|\mathbf{x}\|_0 \ll d$, $\mathbf{X} \subset \{0,1\}^{d \times n}$.

- Example:
  - Storing binary vector 01011001
  - Retrieve it from 010?10?1
- $\mathbf{W} = \max_{\mathbf{x} \in \mathbf{X}} \mathbf{x}\mathbf{x}^\top \in \{0,1\}^{d \times d}$ $(= \mathbf{X}\mathbf{X}^\top$ for Boolean algebra),
- $\mathbf{y} = sgn(\mathbf{W}\mathbf{x} - s\mathbf{1}) = u(\mathbf{x})$,
- $U(\mathbf{x}) \triangleq u(u(u(u(\dots u(\mathbf{x})))))$.

Framework

- $\mathbf{x} \in \{0,1\}^d$, $\|\mathbf{x}\|_0 \ll d$, $\mathbf{X} \subset \{0,1\}^{d \times n}$.

- Example:
  - Storing binary vector 01011001
  - Retrieve it from 010?10?1
- $\mathbf{W} = \max_{\mathbf{x} \in \mathbf{X}} \mathbf{x}\mathbf{x}^\top \in \{0,1\}^{d \times d}$ ($= \mathbf{X}\mathbf{X}^\top$ for Boolean algebra),
- $\mathbf{y} = sgn(\mathbf{W}\mathbf{x} - s\mathbf{1}) = u(\mathbf{x})$,
- $U(\mathbf{x}) \triangleq u(u(u(u(\dots u(\mathbf{x})))))$.

Framework

- $\mathbf{x} \in \{0,1\}^d$, $\|\mathbf{x}\|_0 \ll d$, $\mathbf{X} \subset \{0,1\}^{d \times n}$.

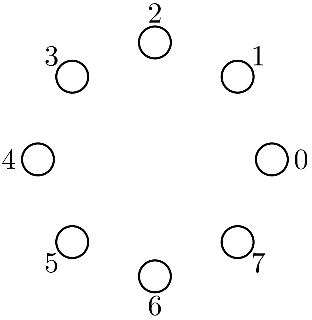
- Example:
  - Storing binary vector 01011001
  - Retrieve it from 010?10?1
- $\mathbf{W} = \max_{\mathbf{x} \in \mathbf{X}} \mathbf{x}\mathbf{x}^\top \in \{0,1\}^{d \times d}$ ($= \mathbf{X}\mathbf{X}^\top$ for Boolean algebra),
- $\mathbf{y} = sgn(\mathbf{W}\mathbf{x} - s\mathbf{1}) = u(\mathbf{x})$,
- $U(\mathbf{x}) \triangleq u(u(u(u(\dots u(\mathbf{x})))))$.

Willshaw Neural Networks

Framework

- $\mathbf{x} \in \{0,1\}^d$, $\|\mathbf{x}\|_0 \ll d$, $\mathbf{X} \subset \{0,1\}^{d \times n}$.

- Example:
  - Storing binary vector 01011001
  - Retrieve it from 010?10?1
- $\mathbf{W} = \max_{\mathbf{x} \in \mathbf{X}} \mathbf{x}\mathbf{x}^\top \in \{0,1\}^{d \times d}$ $(= \mathbf{X}\mathbf{X}^\top$ for Boolean algebra),
- $\mathbf{y} = sgn(\mathbf{W}\mathbf{x} - s\mathbf{1}) = u(\mathbf{x})$,
- $U(\mathbf{x}) \triangleq u(u(u(u(\dots u(\mathbf{x})))))$.

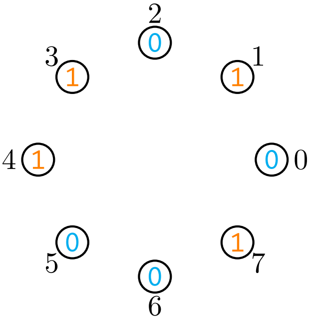


Willshaw Neural Networks

Framework

- $\mathbf{x} \in \{0,1\}^d$, $\|\mathbf{x}\|_0 \ll d$, $\mathbf{X} \subset \{0,1\}^{d \times n}$.

- Example:
  - Storing binary vector 01011001
  - Retrieve it from 010?10?1
- $\mathbf{W} = \max_{\mathbf{x} \in \mathbf{X}} \mathbf{x}\mathbf{x}^\top \in \{0,1\}^{d \times d}$ ($= \mathbf{X}\mathbf{X}^\top$ for Boolean algebra),
- $\mathbf{y} = sgn(\mathbf{W}\mathbf{x} - s\mathbf{1}) = u(\mathbf{x})$,
- $U(\mathbf{x}) \triangleq u(u(u(u(\dots u(\mathbf{x}))))).$

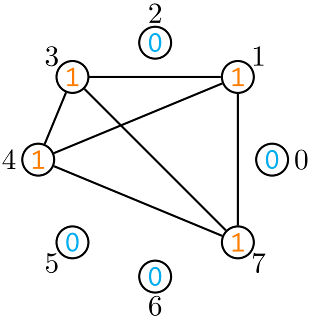


Willshaw Neural Networks

Framework

- $\mathbf{x} \in \{0,1\}^d$, $\|\mathbf{x}\|_0 \ll d$, $\mathbf{X} \subset \{0,1\}^{d \times n}$.

- Example:
  - Storing binary vector 01011001
  - Retrieve it from 010?10?1
- $\mathbf{W} = \max_{\mathbf{x}\in\mathbf{X}} \mathbf{x}\mathbf{x}^\top \in \{0,1\}^{d\times d}$ ($= \mathbf{X}\mathbf{X}^\top$ for Boolean algebra),
- $\mathbf{y} = sgn(\mathbf{W}\mathbf{x} - s\mathbf{1}) = u(\mathbf{x})$,
- $U(\mathbf{x}) \triangleq u(u(u(u(\dots u(\mathbf{x})))))$.

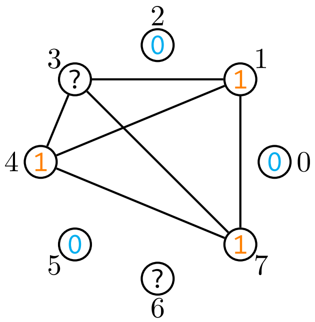


Willshaw Neural Networks

Framework

- $\mathbf{x} \in \{0,1\}^d$, $\|\mathbf{x}\|_0 \ll d$, $\mathbf{X} \subset \{0,1\}^{d \times n}$.

- Example:
  - Storing binary vector 01011001
  - Retrieve it from 010?10?1
- $\mathbf{W} = \max_{\mathbf{x} \in \mathbf{X}} \mathbf{x}\mathbf{x}^\top \in \{0,1\}^{d \times d}$ ($= \mathbf{X}\mathbf{X}^\top$ for Boolean algebra),
- $\mathbf{y} = sgn(\mathbf{W}\mathbf{x} - s\mathbf{1}) = u(\mathbf{x})$,
- $U(\mathbf{x}) \triangleq u(u(u(u(\dots u(\mathbf{x})))))$.

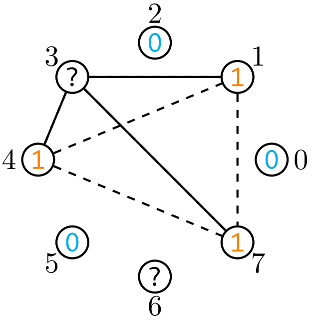


Willshaw Neural Networks

Framework

- $\mathbf{x} \in \{0,1\}^d$, $\|\mathbf{x}\|_0 \ll d$, $\mathbf{X} \subset \{0,1\}^{d \times n}$.

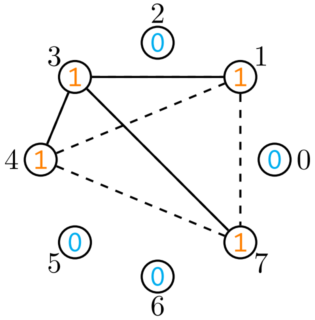
- Example:
  - Storing binary vector 01011001
  - Retrieve it from 010?10?1
- $\mathbf{W} = \max_{\mathbf{x}\in\mathbf{X}} \mathbf{x}\mathbf{x}^\top \in \{0,1\}^{d\times d}$ ($= \mathbf{X}\mathbf{X}^\top$ for Boolean algebra),
- $\mathbf{y} = sgn(\mathbf{W}\mathbf{x} - s\mathbf{1}) = u(\mathbf{x})$,
- $U(\mathbf{x}) \triangleq u(u(u(u(\dots u(\mathbf{x})))))$.



Framework

- $\mathbf{x} \in \{0,1\}^d$, $\|\mathbf{x}\|_0 \ll d$, $\mathbf{X} \subset \{0,1\}^{d \times n}$.

- Example:
  - Storing binary vector 01011001
  - Retrieve it from 01011001
- $\mathbf{W} = \max_{\mathbf{x}\in\mathbf{X}} \mathbf{x}\mathbf{x}^\top \in \{0,1\}^{d\times d}$ ($= \mathbf{X}\mathbf{X}^\top$ for Boolean algebra),
- $\mathbf{y} = sgn(\mathbf{W}\mathbf{x} - s\mathbf{1}) = u(\mathbf{x})$,
- $U(\mathbf{x}) \triangleq u(u(u(u(\dots u(\mathbf{x})))))$.

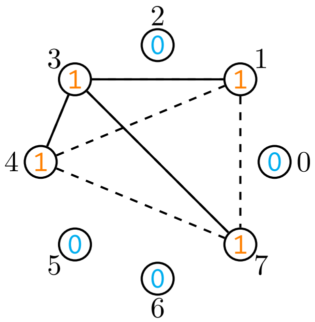


Willshaw Neural Networks

Framework

- $\mathbf{x} \in \{0,1\}^d$, $\|\mathbf{x}\|_0 \ll d$, $\mathbf{X} \subset \{0,1\}^{d \times n}$.

- Example:
  - Storing binary vector 01011001
  - Retrieve it from 01011001
- $\mathbf{W} = \max_{\mathbf{x} \in \mathbf{X}} \mathbf{x}\mathbf{x}^\top \in \{0,1\}^{d \times d}$ ($= \mathbf{X}\mathbf{X}^\top$ for Boolean algebra),
- $\mathbf{y} = sgn(\mathbf{W}\mathbf{x} - s\mathbf{1}) = u(\mathbf{x})$,
- $U(\mathbf{x}) \triangleq u(u(u(u(\dots u(\mathbf{x})))))$.

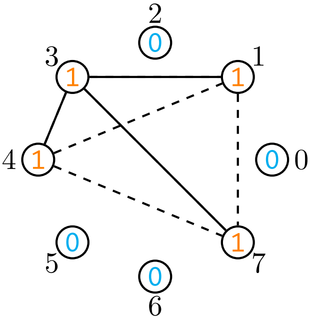


Willshaw Neural Networks

Framework

- $\mathbf{x} \in \{0,1\}^d$, $\|\mathbf{x}\|_0 \ll d$, $\mathbf{X} \subset \{0,1\}^{d \times n}$.

- Example:
  - Storing binary vector 01011001
  - Retrieve it from 01011001
- $\mathbf{W} = \max_{\mathbf{x}\in\mathbf{X}} \mathbf{x}\mathbf{x}^\top \in \{0,1\}^{d\times d}$ (= $\mathbf{X}\mathbf{X}^\top$ for Boolean algebra),
- $\mathbf{y} = sgn(\mathbf{W}\mathbf{x} - s\mathbf{1}) = u(\mathbf{x})$,
- $U(\mathbf{x}) \triangleq u(u(u(u(\dots u(\mathbf{x}))))).$

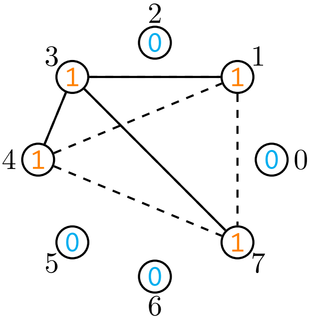


Willshaw Neural Networks

Framework

- $\mathbf{x} \in \{0,1\}^d$, $\|\mathbf{x}\|_0 \ll d$, $\mathbf{X} \subset \{0,1\}^{d \times n}$.

- Example:
  - Storing binary vector 01011001
  - Retrieve it from 01011001
- $\mathbf{W} = \max_{\mathbf{x} \in \mathbf{X}} \mathbf{x}\mathbf{x}^\top \in \{0,1\}^{d \times d}$ ($= \mathbf{X}\mathbf{X}^\top$ for Boolean algebra),
- $\mathbf{y} = sgn(\mathbf{W}\mathbf{x} - s\mathbf{1}) = u(\mathbf{x})$,
- $U(\mathbf{x}) \triangleq u(u(u(u(\dots u(\mathbf{x})))))$.



Stability of stored vectors

Theorem [2]

Consider $\mathbf{X}$ generated with $\|\mathbf{x}\|_0 = \lfloor \log(d)/d \rfloor$ and $\mathbf{x}$ chosen at random such that $\|\mathbf{x}\|_0 = \lfloor \log(d)/d \rfloor$. With $n = \alpha d^2 \log\log(d)/\log^2(d)$:

- If $\alpha > 2$, $\mathbb{P}[u(\mathbf{x}) = \mathbf{x}] \to 1$,
- If $\alpha = 2$, $\exists \gamma > 0$, for d large enough, $\mathbb{P}[u(\mathbf{x}) = \mathbf{x}] \geq \gamma$,
- If $\alpha < 2$, $\mathbb{P}[u(\mathbf{x}) = \mathbf{x}] \to 0$.

[2] "A Comparative Study of Sparse Associative Memories", Jour. Stat. Phys.

Stability of stored vectors

Theorem [2]

Consider $\mathbf{X}$ generated with $\|\mathbf{x}\|_0 = \lfloor \log(d)/d \rfloor$ and $\mathbf{x}$ chosen at random such that $\|\mathbf{x}\|_0 = \lfloor \log(d)/d \rfloor$. With $n = \alpha d^2 \log\log(d)/\log^2(d)$:

- If $\alpha > 2$, $\mathbb{P}[u(\mathbf{x}) = \mathbf{x}] \to 1$,
- If $\alpha = 2$, $\exists \gamma > 0$, for d large enough, $\mathbb{P}[u(\mathbf{x}) = \mathbf{x}] \geq \gamma$,
- If $\alpha < 2$, $\mathbb{P}[u(\mathbf{x}) = \mathbf{x}] \to 0$.

[2] "A Comparative Study of Sparse Associative Memories", Jour. Stat. Phys.

Stability of stored vectors

Theorem [2]

Consider $\mathbf{X}$ generated with $\|\mathbf{x}\|_0 = \lfloor \log(d)/d \rfloor$ and $\mathbf{x}$ chosen at random such that $\|\mathbf{x}\|_0 = \lfloor \log(d)/d \rfloor$. With $n = \alpha d^2 \log\log(d)/\log^2(d)$:

- If $\alpha > 2$, $\mathbb{P}[u(\mathbf{x}) = \mathbf{x}] \to 1$,
- If $\alpha = 2$, $\exists \gamma > 0$, for d large enough, $\mathbb{P}[u(\mathbf{x}) = \mathbf{x}] \geq \gamma$,
- If $\alpha < 2$, $\mathbb{P}[u(\mathbf{x}) = \mathbf{x}] \to 0$.

[2] “A Comparative Study of Sparse Associative Memories”, Jour. Stat. Phys.

Retrievability of stored vectors

Theorem [2]

Consider $n = \alpha d^2 / \log^2(d)$, $\rho \in [0,1[$ such that $\lfloor \rho \log(d) \rfloor$ of 1s in $\mathbf{x}$ are erased to obtain $\tilde{\mathbf{x}}$. Then:

- If $\alpha < -\log(1 - \exp(-1/(1-\rho)))$, then $\mathbb{P}[u(\tilde{\mathbf{x}}) = \mathbf{x}] \to 1$,
- If $\alpha > -\log(1 - \exp(-1/(1-\rho)))$, then $\mathbb{P}[u(\tilde{\mathbf{x}}) \neq \mathbf{x}] \to 1$.

On the difficulty of computing the memory efficiency

- $\binom{d}{2}$ connections with 2 possible values each $\Rightarrow$ takes $\binom{d}{2}$ bits without compression,
- To be compared to the entropy of $\mathbf{X}$: $\approx ndH_2(\log(d)/d)$,
- When patterns are stable, we obtain $\eta \geq \frac{\alpha d \log\log(d)}{\log(d)} \to +\infty$.

[2] "A Comparative Study of Sparse Associative Memories", Jour. Stat. Phys.

Retrievability of stored vectors

Theorem [2]

Consider $n = \alpha d^2 / \log^2(d)$, $\rho \in [0,1[$ such that $\lfloor \rho \log(d) \rfloor$ of 1s in $\mathbf{x}$ are erased to obtain $\tilde{\mathbf{x}}$. Then:

- If $\alpha < -\log(1-\exp(-1/(1-\rho)))$, then $\mathbb{P}[u(\tilde{\mathbf{x}}) = \mathbf{x}] \to 1$,
- If $\alpha > -\log(1-\exp(-1/(1-\rho)))$, then $\mathbb{P}[u(\tilde{\mathbf{x}}) \neq \mathbf{x}] \to 1$.

On the difficulty of computing the memory efficiency

- $\binom{d}{2}$ connections with 2 possible values each $\Rightarrow$ takes $\binom{d}{2}$ bits without compression,
- To be compared to the entropy of $\mathbf{X}$: $\approx ndH_2(\log(d)/d)$,
- When patterns are stable, we obtain $\eta \geq \frac{\alpha d \log \log(d)}{\log(d)} \to +\infty$.

[2] "A Comparative Study of Sparse Associative Memories", Jour. Stat. Phys.

Retrievability of stored vectors

Theorem [2]

Consider $n = \alpha d^2/\log^2(d)$, $\rho \in [0,1[$ such that $\lfloor \rho \log(d) \rfloor$ of 1s in $\mathbf{x}$ are erased to obtain $\tilde{\mathbf{x}}$. Then:

- If $\alpha < -\log(1-\exp(-1/(1-\rho)))$, then $\mathbb{P}[u(\tilde{\mathbf{x}}) = \mathbf{x}] \to 1$,
- If $\alpha > -\log(1-\exp(-1/(1-\rho)))$, then $\mathbb{P}[u(\tilde{\mathbf{x}}) \neq \mathbf{x}] \to 1$.

On the difficulty of computing the memory efficiency

- $\binom{d}{2}$ connections with 2 possible values each $\Rightarrow$ takes $\binom{d}{2}$ bits without compression,
- To be compared to the entropy of $\mathbf{X}$: $\approx ndH_2(\log(d)/d)$,
- When patterns are stable, we obtain $\eta \geq \frac{\alpha d \log\log(d)}{\log(d)} \to +\infty$.

[2] "A Comparative Study of Sparse Associative Memories", Jour. Stat. Phys.

Retrievability of stored vectors

Theorem [2]

Consider $n = \alpha d^2 / \log^2(d)$, $\rho \in [0, 1[$ such that $\lfloor \rho \log(d) \rfloor$ of 1s in $\mathbf{x}$ are erased to obtain $\tilde{\mathbf{x}}$. Then:

- If $\alpha < -\log(1 - \exp(-1/(1-\rho)))$, then $\mathbb{P}[u(\tilde{\mathbf{x}}) = \mathbf{x}] \to 1$,
- If $\alpha > -\log(1 - \exp(-1/(1-\rho)))$, then $\mathbb{P}[u(\tilde{\mathbf{x}}) \neq \mathbf{x}] \to 1$.

On the difficulty of computing the memory efficiency

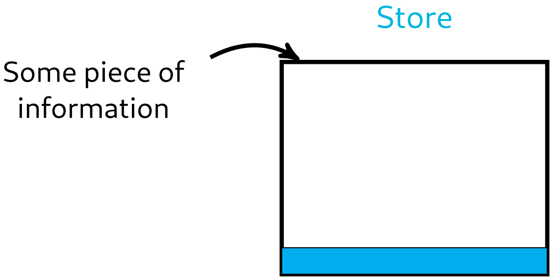
- $\binom{d}{2}$ connections with 2 possible values each $\Rightarrow$ takes $\binom{d}{2}$ bits without compression,
- To be compared to the entropy of $\mathbf{X}$: $\approx ndH_2(\log(d)/d)$,
- When patterns are stable, we obtain $\eta \geq \frac{\alpha d \log\log(d)}{\log(d)} \to +\infty$.

[2] “A Comparative Study of Sparse Associative Memories”, Jour. Stat. Phys.

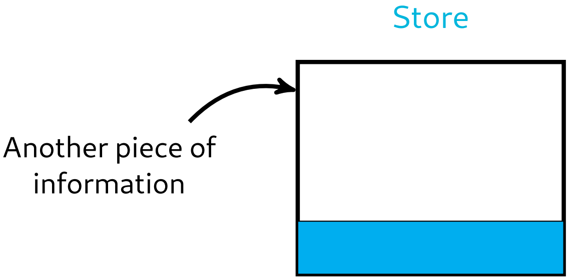
Associative memories

Store

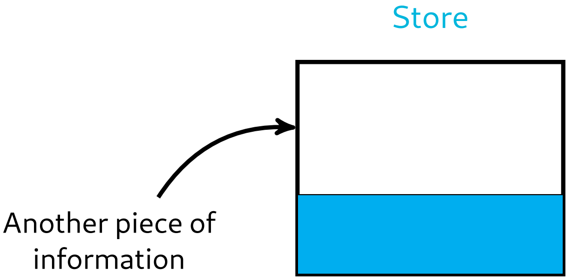
Associative memories



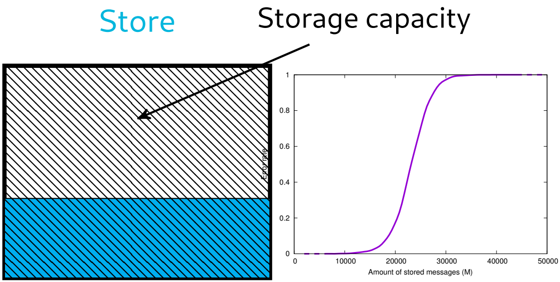
Associative memories



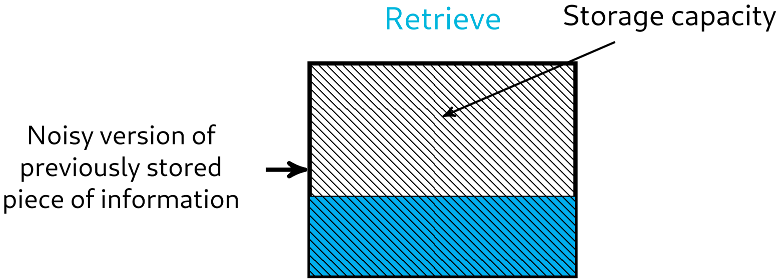
Associative memories



Associative memories

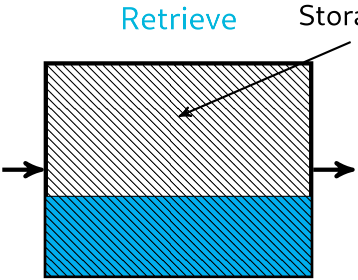


Associative memories



Associative memories

Noisy version of previously stored piece of information



Corresponding previously stored piece of information

Summary

	Hopfield	Willshaw
Framework	$\mathbf{x} \in \{-1,1\}^d$	$\mathbf{x} \in \{0,1\}^d$, $\|\mathbf{x}\|_0 \ll d$
Memory	$\mathbf{x}\mathbf{x}^\top$	$\mathbf{x}\mathbf{x}^\top$
Aggregation	$\mathbf{W} = \sum_{\mathbf{x}\in\mathbf{X}} \mathbf{x}\mathbf{x}^\top$ $\mathbf{W} = \mathbf{X}\mathbf{X}^\top$ using classical linear algebra	$\mathbf{W} = \max_{\mathbf{x}\in\mathbf{X}} \mathbf{x}\mathbf{x}^\top$ $\mathbf{W} = \mathbf{X}\mathbf{X}^\top$ using Boolean algebra
Search	$sgn(\mathbf{W}\tilde{\mathbf{x}})$	$sgn(\mathbf{W}\tilde{\mathbf{x}} - s\mathbf{1})$

Summary

	Hopfield	Willshaw
Framework	$\mathbf{x} \in \{-1, 1\}^d$	$\mathbf{x} \in \{0, 1\}^d$, $\|\mathbf{x}\|_0 \ll d$
Memory	$\mathbf{x}\mathbf{x}^\top$	$\mathbf{x}\mathbf{x}^\top$
Aggregation	$\mathbf{W} = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{x}\mathbf{x}^\top$ $\mathbf{W} = \mathbf{X}\mathbf{X}^\top$ using classical linear algebra	$\mathbf{W} = \max_{\mathbf{x} \in \mathbf{X}} \mathbf{x}\mathbf{x}^\top$ $\mathbf{W} = \mathbf{X}\mathbf{X}^\top$ using Boolean algebra
Search	$sgn(\mathbf{W}\tilde{\mathbf{x}})$	$sgn(\mathbf{W}\tilde{\mathbf{x}} - s\mathbf{1})$

Summary

	Hopfield	**Willshaw**
Framework	$\mathbf{x} \in \{-1,1\}^d$	$\mathbf{x} \in \{0,1\}^d$, $\|\mathbf{x}\|_0 \ll d$
Memory	$\mathbf{x}\mathbf{x}^\top$	$\mathbf{x}\mathbf{x}^\top$
Aggregation	$\mathbf{W} = \sum_{\mathbf{x}\in\mathbf{X}} \mathbf{x}\mathbf{x}^\top$ $\mathbf{W} = \mathbf{X}\mathbf{X}^\top$ using classical linear algebra	$\mathbf{W} = \max_{\mathbf{x}\in\mathbf{X}} \mathbf{x}\mathbf{x}^\top$ $\mathbf{W} = \mathbf{X}\mathbf{X}^\top$ using Boolean algebra
Search	$sgn(\mathbf{W}\tilde{\mathbf{x}})$	$sgn(\mathbf{W}\tilde{\mathbf{x}} - s\mathbf{1})$

Summary

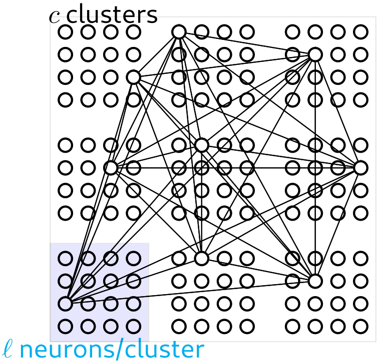
	Hopfield	**Willshaw**
Framework	$\mathbf{x} \in \{-1, 1\}^d$	$\mathbf{x} \in \{0, 1\}^d$, $\|\mathbf{x}\|_0 \ll d$
Memory	$\mathbf{x}\mathbf{x}^\top$	$\mathbf{x}\mathbf{x}^\top$
Aggregation	$\mathbf{W} = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{x}\mathbf{x}^\top$ $\mathbf{W} = \mathbf{X}\mathbf{X}^\top$ using classical linear algebra	$\mathbf{W} = \max_{\mathbf{x} \in \mathbf{X}} \mathbf{x}\mathbf{x}^\top$ $\mathbf{W} = \mathbf{X}\mathbf{X}^\top$ using Boolean algebra
Search	$sgn(\mathbf{W}\tilde{\mathbf{x}})$	$sgn(\mathbf{W}\tilde{\mathbf{x}} - s\mathbf{1})$

Summary

	Hopfield	**Willshaw**
Framework	$\mathbf{x} \in \{-1,1\}^d$	$\mathbf{x} \in \{0,1\}^d$, $\|\mathbf{x}\|_0 \ll d$
Memory	$\mathbf{x}\mathbf{x}^\top$	$\mathbf{x}\mathbf{x}^\top$
Aggregation	$\mathbf{W} = \sum_{\mathbf{x}\in\mathbf{X}} \mathbf{x}\mathbf{x}^\top$ $\mathbf{W} = \mathbf{X}\mathbf{X}^\top$ using classical linear algebra	$\mathbf{W} = \max_{\mathbf{x}\in\mathbf{X}} \mathbf{x}\mathbf{x}^\top$ $\mathbf{W} = \mathbf{X}\mathbf{X}^\top$ using Boolean algebra
Search	$sgn(\mathbf{W}\tilde{\mathbf{x}})$	$sgn(\mathbf{W}\tilde{\mathbf{x}} - s\mathbf{1})$

Outline

Adding structure to Willshaw networks



Leveraging the structure

Willshaw rule for retrieval

- $\mathbf{x}_{ij}$ denotes the j-th neuron in the i-th cluster,
- $u(\mathbf{x})_{ij} = 1 \Leftrightarrow \sum_{i'} \sum_{j'} \mathbf{W}_{i'j',ij}\mathbf{x}_{i'j'} \geq s.$
- Storage is performed using Boolean algebra, retrieving is performed using classical linear algebra.

New rule for retrieval

- $u(\mathbf{x})_{ij} = 1 \Leftrightarrow \sum_{i'} \max_{j'} \mathbf{W}_{i'j',ij}\mathbf{x}_{i'j'}$ is maximal in cluster i,
- $u(\mathbf{x})_{ij} = 1 \Leftrightarrow \wedge_{i'} \vee_{j'} \mathbf{W}_{i'j',ij} \wedge \mathbf{x}_{i'j'}$,
  if neurons in erased clusters are all initialized active.
- Both storage and retrieval are performed using Boolean algebra.

Leveraging the structure

Willshaw rule for retrieval

- $\mathbf{x}_{ij}$ denotes the j-th neuron in the i-th cluster,
- $u(\mathbf{x})_{ij} = 1 \Leftrightarrow \sum_{i'} \sum_{j'} \mathbf{W}_{i'j',ij} \mathbf{x}_{i'j'} \geq s$.
- Storage is performed using Boolean algebra, retrieving is performed using classical linear algebra.

New rule for retrieval

- $u(\mathbf{x})_{ij} = 1 \Leftrightarrow \sum_{i'} \max_{j'} \mathbf{W}_{i'j',ij} \mathbf{x}_{i'j'}$ is maximal in cluster i,
- $u(\mathbf{x})_{ij} = 1 \Leftrightarrow \wedge_{i'} \vee_{j'} \mathbf{W}_{i'j',ij} \wedge \mathbf{x}_{i'j'}$,
  if neurons in erased clusters are all initialized active.
- Both storage and retrieval are performed using Boolean algebra.

Leveraging the structure

Willshaw rule for retrieval

- $\mathbf{x}_{ij}$ denotes the j-th neuron in the i-th cluster,
- $u(\mathbf{x})_{ij} = 1 \Leftrightarrow \sum_{i'} \sum_{j'} \mathbf{W}_{i'j',ij}\mathbf{x}_{i'j'} \geq s$.
- Storage is performed using Boolean algebra, retrieving is performed using classical linear algebra.

New rule for retrieval

- $u(\mathbf{x})_{ij} = 1 \Leftrightarrow \sum_{i'} \max_{j'} \mathbf{W}_{i'j',ij}\mathbf{x}_{i'j'}$ is maximal in cluster i,
- $u(\mathbf{x})_{ij} = 1 \Leftrightarrow \wedge_{i'} \vee_{j'} \mathbf{W}_{i'j',ij} \wedge \mathbf{x}_{i'j'}$,
  if neurons in erased clusters are all initialized active.
- Both storage and retrieval are performed using Boolean algebra.

Leveraging the structure

Willshaw rule for retrieval

- $\mathbf{x}_{ij}$ denotes the j-th neuron in the i-th cluster,
- $u(\mathbf{x})_{ij} = 1 \Leftrightarrow \sum_{i'} \sum_{j'} \mathbf{W}_{i'j',ij} \mathbf{x}_{i'j'} \geq s$.
- Storage is performed using Boolean algebra, retrieving is performed using classical linear algebra.

New rule for retrieval

- $u(\mathbf{x})_{ij} = 1 \Leftrightarrow \sum_{i'} \max_{j'} \mathbf{W}_{i'j',ij} \mathbf{x}_{i'j'}$ is maximal in cluster i,
- $u(\mathbf{x})_{ij} = 1 \Leftrightarrow \wedge_{i'} \vee_{j'} \mathbf{W}_{i'j',ij} \wedge \mathbf{x}_{i'j'}$,
  if neurons in erased clusters are all initialized active.
- Both storage and retrieval are performed using Boolean algebra.

Memory efficiency (with some approximations)

Approaching $\log(2)$

- Let us choose: $\alpha c = 2\log_2(\ell)$,
- $\eta \sim \frac{nc\log_2(\ell)}{\binom{c}{2}\ell^2} \sim \frac{\alpha n}{\ell^2}$,
- Probability a given connection exists (i.i.d. uniform vectors): $p = 1-(1-\ell^{-2})^n \Rightarrow n \sim -\ell^2\log(1-p)$,
- Probability to accept a random vector: $P_e \approx p^{\binom{c}{2}}$, none of them : $P_e^* \leq P_e \ell^c$,
  - $P_e^* \underset{+\infty}{\leq} \exp\left(\frac{c^2}{2}\left[\log_2(p)+\alpha\right]\right) \to 0$ if $\alpha = -\beta\log_2(p)$, $\beta < 1$.
- Conclusion : $\eta \sim \beta\log_2(1-p)\log_2(p)\log(2)$

Memory efficiency (with some approximations)

Approaching $\log(2)$

- Let us choose: $\alpha c = 2\log_2(\ell)$,
- $\eta \sim \frac{nc\log_2(\ell)}{\binom{c}{2}\ell^2} \sim \frac{\alpha n}{\ell^2}$,
- Probability a given connection exists (i.i.d. uniform vectors): $p = 1 - (1-\ell^{-2})^n \Rightarrow n \sim -\ell^2 \log(1-p)$,
- Probability to accept a random vector: $P_e \approx p^{\binom{c}{2}}$, none of them : $P_e^* \leq P_e \ell^c$,
  - $P_e^* \underset{+\infty}{\leq} \exp\left(\frac{c^2}{2}\left[\log_2(p) + \alpha\right]\right) \to 0$ if $\alpha = -\beta\log_2(p)$, $\beta < 1$.
- Conclusion : $\eta \sim \beta \log_2(1-p)\log_2(p)\log(2)$

Memory efficiency (with some approximations)

Approaching $\log(2)$

- Let us choose: $\alpha c = 2\log_2(\ell)$,
- $\eta \sim \frac{nc\log_2(\ell)}{\binom{c}{2}\ell^2} \sim \frac{\alpha n}{\ell^2}$,
- Probability a given connection exists (i.i.d. uniform vectors): $p = 1 - (1 - \ell^{-2})^n \Rightarrow n \sim -\ell^2 \log(1-p)$,
- Probability to accept a random vector: $P_e \approx p^{\binom{c}{2}}$, none of them : $P_e^* \leq P_e \ell^c$,
  - $P_e^* \underset{+\infty}{\leq} \exp\left(\frac{c^2}{2}\left[\log_2(p) + \alpha\right]\right) \to 0$ if $\alpha = -\beta\log_2(p)$, $\beta < 1$.
- Conclusion : $\eta \sim \beta \log_2(1-p)\log_2(p)\log(2)$

Memory efficiency (with some approximations)

Approaching $\log(2)$

- Let us choose: $\alpha c = 2\log_2(\ell)$,
- $\eta \sim \frac{nc\log_2(\ell)}{\binom{c}{2}\ell^2} \sim \frac{\alpha n}{\ell^2}$,
- Probability a given connection exists (i.i.d. uniform vectors): $p = 1-(1-\ell^{-2})^n \Rightarrow n \sim -\ell^2\log(1-p)$,
- Probability to accept a random vector: $P_e \approx p^{\binom{c}{2}}$, none of them : $P_e^* \leq P_e \ell^c$,
  - $P_e^* \underset{+\infty}{\leq} \exp\left(\frac{c^2}{2}\left[\log_2(p)+\alpha\right]\right) \to 0$ if $\alpha = -\beta\log_2(p)$, $\beta < 1$.
- Conclusion : $\eta \sim \beta\log_2(1-p)\log_2(p)\log(2)$

Memory efficiency (with some approximations)

Approaching $\log(2)$

- Let us choose: $\alpha c = 2\log_2(\ell)$,
- $\eta \sim \frac{nc\log_2(\ell)}{\binom{c}{2}\ell^2} \sim \frac{\alpha n}{\ell^2}$,
- Probability a given connection exists (i.i.d. uniform vectors): $p = 1 - (1 - \ell^{-2})^n \Rightarrow n \sim -\ell^2 \log(1-p)$,
- Probability to accept a random vector: $P_e \approx p^{\binom{c}{2}}$, none of them : $P_e^* \leq P_e \ell^c$,
  - $P_e^* \underset{+\infty}{\leq} \exp\left(\frac{c^2}{2}\left[\log_2(p) + \alpha\right]\right) \to 0$ if $\alpha = -\beta\log_2(p)$, $\beta < 1$.
- Conclusion : $\eta \sim \beta\log_2(1-p)\log_2(p)\log(2)$

Memory efficiency (with some approximations)

Approaching $\log(2)$

- Let us choose: $\alpha c = 2\log_2(\ell)$,
- $\eta \sim \frac{nc\log_2(\ell)}{\binom{c}{2}\ell^2} \sim \frac{\alpha n}{\ell^2}$,
- Probability a given connection exists (i.i.d. uniform vectors): $p = 1-(1-\ell^{-2})^n \Rightarrow n \sim -\ell^2\log(1-p)$,
- Probability to accept a random vector: $P_e \approx p^{\binom{c}{2}}$, none of them : $P_e^* \leq P_e \ell^c$,
  - $P_e^* \underset{+\infty}{\leq} \exp\left(\frac{c^2}{2}\left[\log_2(p)+\alpha\right]\right) \to 0$ if $\alpha = -\beta\log_2(p)$, $\beta < 1$.
- Conclusion : $\eta \sim \beta\log_2(1-p)\log_2(p)\log(2)$

Asymptotic behavior

Storage diversity

Theorem: consider $n = \alpha \log(c)\ell^2$, with $c = \log(\ell)$, then:

- For $\alpha > 2$, a random vector is accepted with probability that goes to 1,
- For $\alpha = 2$, probability is strictly positive,
- For $\alpha < 2$, probability goes to 0.

Stability and error correction

Theorem: Consider $n = \alpha\ell^2/c^2$ vectors. Deactivate ρc initial neurons, then for $\alpha < -\log(1 - \exp(-1/(1-\rho)))$, probability to retrieve the vector goes to 1.

"A comparative study of sparse associative memories," Jour. Stat. Phys.

Asymptotic behavior

Storage diversity

Theorem: consider $n = \alpha \log(c)\ell^2$, with $c = \log(\ell)$, then:

- For $\alpha > 2$, a random vector is accepted with probability that goes to 1,
- For $\alpha = 2$, probability is strictly positive,
- For $\alpha < 2$, probability goes to 0.

Stability and error correction

Theorem: Consider $n = \alpha \ell^2 / c^2$ vectors. Deactivate ρc initial neurons, then for $\alpha < -\log(1 - \exp(-1/(1-\rho)))$, probability to retrieve the vector goes to 1.

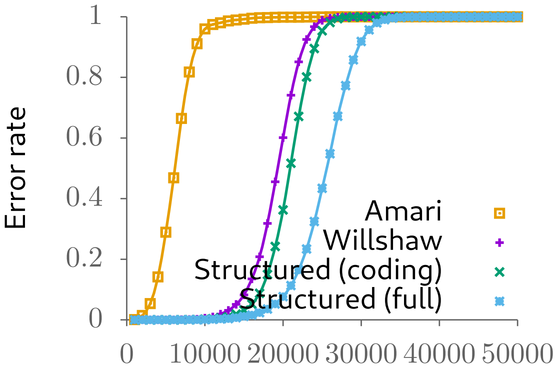
"A comparative study of sparse associative memories," Jour. Stat. Phys.

Performance in search

Amari (Hopfield)
- No structure
- Weights

Willshaw
- No structure
- No weights

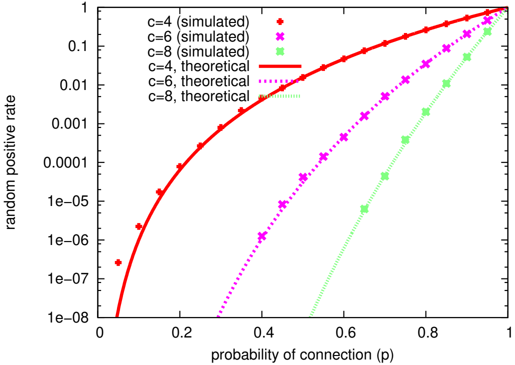
Structured
- Clusters
- No weights



- 2048 neurons total,
- 8 neurons per vector,
- 4 initially activated neurons,
- ($\ell = 256$).

"A comparative study of sparse associative memories," Jour. Stat. Phys.

Performance in indexing



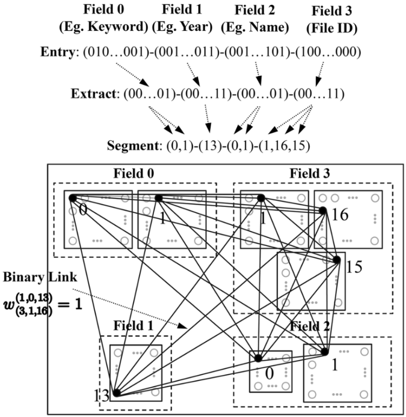
False positive rate for various number of clusters c and $\ell = 512$ neurons per cluster.

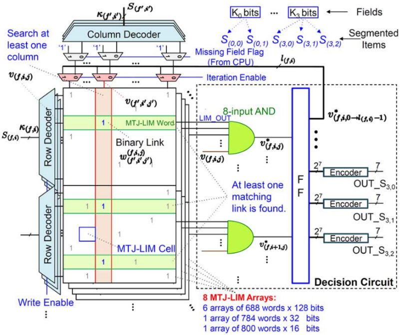
With 1% of error, memory efficiency is 137.1%

Outline

Application to Implementation of Search Engines

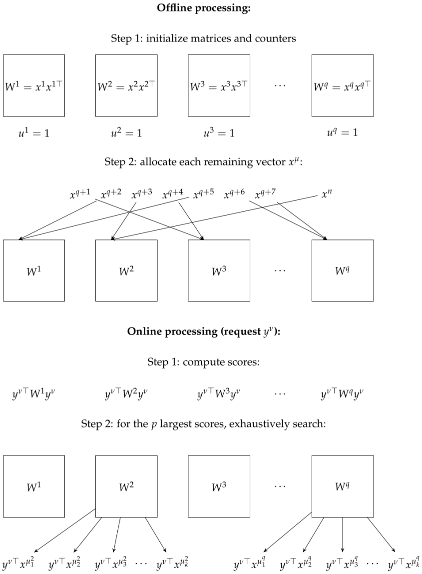
$$v^*_{(f,i,j)} = \left(\bigwedge_{\substack{i'=0 \\ (f',i')\neq(f,i)}}^{\psi-1} \bigvee_{j'=0}^{\gamma-1} w^{(f,i,j)}_{(f',i',j')} v_{(f',i',j')} \right) \bigwedge v_{(f,i,j)}$$





Conclusion: 13.6x memory reduction and 89% energy saving compared to classical CAMs.

"A Nonvolatile Associative Memory-BasedContext-Driven Search Engine Using 90 nmCMOS/MTJ-Hybrid Logic-in-Memory Architecture," IEEE Journal on Emerging and Selected Topics in Circuits and Systems



"Associative Memories to Accelerate Nearest Neighbor Search," Applied Science

Application to Approximate Nearest Neighbor Search

Recalls on SIFT1M dataset.

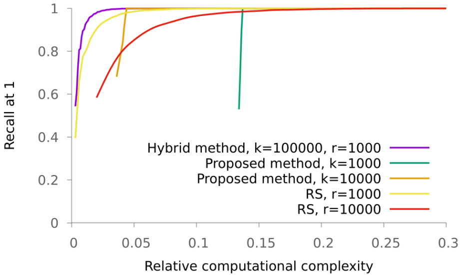


Table 1. Comparison of recall@1 and computation time for one scan (in ms) of the proposed method, kd-trees, K-means trees [1], ANN [16] and LSH [22] on the SIFT1M dataset for various targeted recall performances.

	Scan Time	Recall@1	Scan Time	Recall@1	Scan Time	Recall@1
Random kd-trees [1]	**0.04**	0.6	**0.22**	0.8	3.1	0.95
K-means trees [1]	0.06	0.6	0.25	0.8	2.8	0.99
Proposed method (hybrid)	0.17	0.6	0.25	0.8	**1.1**	0.99
ANN [16]	3.7	0.6	8.2	0.8	24	0.95
LSH [22]	6.4	0.6	11.1	0.8	28	0.98

"Associative Memories to Accelerate Nearest Neighbor Search," Applied Science

DecisiveNets: from DNNs to DAMs

$$\hat{\mathbf{y}}[\ell(i-1):\ell i] = \sigma_t\left(\mathbf{x}[\ell(i-1):\ell i]\right), \forall i, \ \ \texttt{where}$$

$$\sigma_t(\mathbf{z}) = \frac{\texttt{softmax}\left(t \cdot \sigma(\mathbf{z})\right)}{\max\left(\texttt{softmax}\left(t \cdot \sigma(\mathbf{z})\right)\right)}\sigma(\mathbf{z}).$$

	Resnet18 and CIFAR-10		Resnet50 and CIFAR-100	
ℓ	accuracy	multiplications	accuracy	multiplications
1 (baseline)	95.21%	5,070,848	78.50%	1,297,809,408
2	**95.25%**	3,125,248	**79.23%**	861,601,792
4	94.65%	2,152,448	76.58%	643,497,984
8	92.95%	1,666,048	70.46%	534,446,080
16	88.95%	1,422,848	64.36%	479,920,128
32	84.90%	1,301,248	61.07%	452,657,152
64	78.28%	1,240,448	53.05%	439,025,664

Resnet18 and CIFAR10

ℓ	clean data	Gaussian noise	Shot noise	Impulse noise
1	95.21%	46.40%	59.50%	51.75%
2	**95.25%**	47.59%	60.21%	53.45%
4	94.65%	49.98%	61.99%	52.33%
8	92.95%	46.34%	58.07%	53.54%
16	88.95%	50.51%	60.26%	48.95%
32	84.90%	**56.56%**	**64.34%**	**54.78%**
64	78.28%	48.72%	55.60%	40.03%

"DecisiveNets: Training Deep Associative Memories to Solve Complex Machine Learning Problems," in review

Outline

Conclusion

Take-away message

- Structured Clique Networks are very efficient associative memories,
- They can help in many problems,
- They are very efficient for specific hardware.

Interesting directions of research

- Improving explanability, robustness, transferability of knowledge in DNNs,
- DNNs on edge,
- Intricating storing and learning in neural networks,
- Continual learning.

email: vincent.gripon@imt-atlantique.fr