Structured Clique Networks as Efficient Associative Memories

Vincent Gripon

March 4th, 2021
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
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 \((E^d)\)

- Supervised learning,
- Unsupervised learning,
- Indexing,
- Search...
Notations and problems

Vector space \((E^d)\)

Vector \(x (\in E^d)\)

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

Vector space \((E^d)\)

Collection \(X (\in E^{d \times n})\)

1. Supervised learning,
2. Unsupervised learning,
3. Indexing,
4. Search...
Notations and problems

Vector space $(E^d)$

Collection $X (\in E^{d \times n})$

1. Supervised learning,
2. Unsupervised learning,
3. Indexing,
4. Search...
Notations and problems

Vector space \((E^d)\)

Collection \(X (\in E^{d \times n})\)

1. Supervised learning,
2. Unsupervised learning,
3. Indexing,
4. Search...
Notations and problems

Vector space \((E^d)\)

Collection \(X (\in E^{d \times n})\)

1. Supervised learning,
2. Unsupervised learning,
3. Indexing,
4. Search...
Notations and problems

Vector space \((E^d)\)

Collection \(X (\in E^{d \times n})\)

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

Learning

To learn is to **generalize** (≠ memorize),

- 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.
Learning

To learn is to **generalize** (≠ memorize),

Supervised learning

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

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

- 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** (≠ memorize),

Supervised learning

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

Classical methods: SVM, \(k\)-NN, Random Forests, LR, MLP, CNN...
Unsupervised learning

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

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

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

Unsupervised learning
- Partitioning/disentangling,
- Requires priors,
- Not so many direct applications.
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 $X \in E^{d \times n}$ and a query vector $x$:

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

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 approximately 4 years.

Vincent Gripon (IMT-Atlantique)
Indexing

**Definition**

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

1. Is \( x \in X \)?
2. Do we have \( x' \in X \) s.t. \( x' \approx x \)?

**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 \( X \in E^{d \times n} \) and a query vector \( x \):

1. Is \( x \in X \)?
2. Do we have \( x' \in X \) s.t. \( x' \approx x \)?

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

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

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 approximately 4 years.
Indexing

Definition

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

1. Is \( x \in X \)?
2. Do we have \( x' \in X \) s.t. \( x' \approx x \)?

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.
Definition

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

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

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).
Definition

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

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

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:

\[ x \mapsto y = \sigma(Wx + b). \]

Nonlinear functions

- Sigmoids (e.g. \( x \mapsto \frac{1}{1 + \exp(-x)} \)),
- 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:

\[ x \mapsto y = \sigma(Wx + b). \]

**Nonlinear functions**
- Sigmoids (e.g. \( x \mapsto \frac{1}{1 + \exp(-x)} \)),
- ReLU (e.g. \( x \mapsto \max(0, x) \)),
- Winner-Take-All (WTA)…
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
   - Decisive Nets
5. Conclusion
Hopfield Neural Networks

Framework

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

Example:
- Storing binary vector -11-111-1-1
- Retrieve it from -11-111-1?1

- \( \mathbf{W} = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{xx}^\top = \mathbf{XX}^\top \) (except diagonal),
- \( \mathbf{y} = \text{sgn}(\mathbf{Wx}) = \mathbf{u}(\mathbf{x}) \),
- \( \mathbf{U}(\mathbf{x}) \triangleq \mathbf{u}(\mathbf{u}(\mathbf{u}(\ldots \mathbf{u}(\mathbf{x})))) \),
- 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 \(-11-111-1?1\)

\[ \mathbf{W} = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{xx}^\top = \mathbf{XX}^\top \text{ (except diagonal)}, \]

\[ y = sgn(\mathbf{Wx}) = u(x), \]

\[ U(x) \triangleq u(u(u(\ldots u(x))))), \]

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 \(-11-111-1?1\)

- \( W = \sum_{x \in X} xx^\top = XX^\top \) (except diagonal),

- \( y = \text{sgn}(Wx) = u(x) \),

- \( U(x) \triangleq u(u(u(u(\ldots u(x)))))) \),

- Complexity: \( O(d^2) \).
Hopfield Neural Networks

**Framework**

- \( x \in \{-1, 1\}^d, \ X \subset \{-1, 1\}^{d \times n}, \)

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

- \( W = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{x} \mathbf{x}^\top = \mathbf{XX}^\top \) (except diagonal),
- \( y = \text{sgn}(W\mathbf{x}) = u(\mathbf{x}) \),
- \( U(x) \triangleq u(u(u(u(u(u(x)))))) \),
- Complexity: \( O(d^2) \).
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 \(-11-111-1-11\)
- \( \mathbf{W} = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{xx}^\top = \mathbf{XX}^\top \) (except diagonal),
- \( y = sgn(\mathbf{Wx}) = u(x) \),
- \( U(x) \triangleq u(u(u(\ldots u(x)))) \),
- Complexity: \( O(d^2) \).
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),
  - \( \mathbf{y} = \text{sgn}(\mathbf{W}\mathbf{x}) = u(\mathbf{x}), \)
  - \( U(\mathbf{x}) \triangleq u(u(u(u(u(u(x))))))), \)
  - 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 \(-11-111-1-11\)
  - \( \mathbf{W} = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{xx}^\top = \mathbf{XX}^\top \) (except diagonal),
  - \( \mathbf{y} = sgn(\mathbf{Wx}) = u(\mathbf{x}) \),
  - \( U(\mathbf{x}) \triangleq u(u(u(u(\ldots u(\mathbf{x}))))) \),

- 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
    - $\mathbf{-11-111-1-11}$
  - Retrieve it from $\mathbf{-11-111-1-11}$
  - $\mathbf{W} = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{xx}^\top = \mathbf{XX}^\top$ (except diagonal),
  - $\mathbf{y} = \text{sgn}(\mathbf{Wx}) = u(\mathbf{x})$,
  - $U(\mathbf{x}) \triangleq u(u(u(\ldots u(\mathbf{x}))))$
  - Complexity: $O(d^2)$.
Stability of stored vectors

Theorem [1]

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

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

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 $X \approx nd$,
- 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 \( d \to \infty \), \( \mathbb{P}[\lim \inf \{ \cap_{x \in X} \{ U(x) = x \} \}] = 1 \),
- If \( \gamma > 4 \), then \( \mathbb{P}[\cap_{x} \{ U(x) = x \}] \to 1 \),
- If \( \gamma < 2 \), then \( \mathbb{P}[\cap_{x} \{ U(x) = x \}] \to 0 \).

**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 \( X \approx nd \),
- 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
Willshaw Neural Networks

Framework

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

Example:
- Storing binary vector $01011001$
- Retrieve it from $010?10?1$
- $W = \max_{x \in X} xx^\top \in \{0, 1\}^{d \times d}$
  ($= XX^\top$ for Boolean algebra),
- $y = \text{sgn}(Wx - s1) = u(x)$,
- $U(x) \triangleq u(u(u(u(\ldots u(x))))))$. 

Vincent Gripon (IMT-Atlantique)
**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
- \( W = \max_{\mathbf{x} \in \mathbf{X}} \mathbf{x}\mathbf{x}^\top \in \{0, 1\}^{d\times d} \) (= \( \mathbf{XX}^\top \) for Boolean algebra),
- \( y = \text{sgn}(Wx - s\mathbf{1}) = u(x) \),
- \( U(x) \overset{\Delta}{=} u(u(u(u(\ldots u(x)))))) \).
**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 \(010?10?1\)

\[ \mathbf{W} = \max_{\mathbf{x} \in \mathbf{X}} \mathbf{x} \mathbf{x}^\top \in \{0, 1\}^{d \times d} \]

\[ (= \mathbf{X} \mathbf{X}^\top \text{ for Boolean algebra}), \]

\[ y = sgn(\mathbf{Wx} - \mathbf{s1}) = u(\mathbf{x}), \]

\[ U(\mathbf{x}) \triangleq u(u(u(u(\ldots u(\mathbf{x})))))). \]
Willshaw Neural Networks

Framework

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

Example:
- Storing binary vector \(01011001\)
- Retrieve it from \(010?10?1\)
- \( W = \max_{x \in X} xx^\top \in \{0, 1\}^{d \times d} \)
  \((= XX^\top\) for Boolean algebra),
- \( y = \text{sgn}(Wx - s1) = u(x) \),
- \( U(x) \triangleq u(u(u(u(\ldots u(u(x)))))))\).
Willshaw Neural Networks

Framework

- \( x \in \{0, 1\}^d, \|x\|_0 \ll d, \; X \subset \{0, 1\}^{d \times n} \).

Example:
- Storing binary vector 01011001
- Retrieve it from 010?10?1
- \( W = \max_{x \in X} xx^\top \in \{0, 1\}^{d \times d} \) (= XX^\top for Boolean algebra),
- \( y = sgn(Wx - s1) = u(x) \),
- \( U(x) \triangleq u(u(u(u(\ldots u(x)))))) \).
**Willshaw Neural Networks**

**Framework**

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

- **Example:**
  - Storing binary vector **01011001**
  - Retrieve it from **01010101**
  - \( W = \max_{x \in X} xx^\top \in \{0, 1\}^{d \times d} \)
    (= \( XX^\top \) for Boolean algebra),
  - \( y = \text{sgn}(Wx - s1) = u(x), \)
  - \( U(x) \triangleq u(u(u(\ldots u(x))))). \)
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 \(010?10?1\)

\( \mathbf{W} = \max_{\mathbf{x} \in \mathbf{X}} \mathbf{x} \mathbf{x}^\top \in \{0, 1\}^{d \times d} \)  
  \( (= \mathbf{XX}^\top \text{ for Boolean algebra}), \)

- \( y = sgn(\mathbf{Wx} - \mathbf{s1}) = u(\mathbf{x}), \)
- \( U(\mathbf{x}) \triangleq u(u(u(\cdots u(u(\mathbf{x}))))) \).
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 $010?10?1$
- $\mathbf{W} = \max_{\mathbf{x} \in \mathbf{X}} \mathbf{xx}^\top \in \{0, 1\}^{d \times d}$
  ($= \mathbf{XX}^\top$ for Boolean algebra),
- $y = \text{sgn}(\mathbf{Wx} - s1) = u(\mathbf{x})$,
- $U(\mathbf{x}) \triangleq u(u(u(\ldots u(\mathbf{x}))))$. 
**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),
- \( y = sgn(\mathbf{Wx} - s1) = u(x) \),
- \( U(x) \triangleq u(u(u(\ldots u(x)))) \)).
Example:
- Storing binary vector $0101001$
- Retrieve it from $01011001$

$W = \max_{x \in X} xx^T \in \{0, 1\}^{d \times d}$

($= XX^T$ for Boolean algebra),

$y = \text{sgn}(Wx - s1) = u(x)$,

$U(x) \overset{\Delta}{=} u(u(u(...)u(x))))$. 
**Framework**

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

**Example:**
- Storing binary vector \(01011001\)
- Retrieve it from \(01011001\)
- \(W = \max_{x \in X} xx^\top \in \{0, 1\}^{d \times d}\) 
  (= \(XX^\top\) for Boolean algebra),
- \(y = \text{sgn}(Wx - s1) = u(x),\)
- \(U(x) \triangleq u(u(u(\ldots u(x))))).\)
Example:
- Storing binary vector $01011001$
- Retrieve it from $01011001$
- $W = \max_{x \in X} xx^\top \in \{0, 1\}^{d \times d}$
  ($= XX^\top$ for Boolean algebra),
- $y = sgn(Wx - s1) = u(x)$,
- $U(x) \triangleq u(u(u(u(\ldots u(x))))))$.
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 $\alpha < 2$, $\mathbb{P}[u(\mathbf{x}) = \mathbf{x}] \to 0$.

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 $\alpha < 2$, $\mathbb{P}[u(\mathbf{x}) = \mathbf{x}] \to 0$.

Stability of stored vectors

Theorem [2]

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

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

Retrievability of stored vectors

**Theorem [2]**

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

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

**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 \( X \): \( \approx ndH_2(\log(d)/d) \),
- When patterns are stable, we obtain \( \eta \geq \frac{\alpha d \log \log(d)}{\log(d)} \to +\infty \).

Retrievability of stored vectors

Theorem [2]

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

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

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 $X$: $\approx ndH_2(\log(d)/d)$,
- When patterns are stable, we obtain $\eta \geq \frac{\alpha d \log \log(d)}{\log(d)} \to +\infty$.

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

**On the difficulty of computing the memory efficiency**

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

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 \( x \) are erased to obtain \( \tilde{x} \). Then:

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

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 \( X: \approx ndH_2(\log(d)/d) \),
- When patterns are stable, we obtain \( \eta \geq \frac{\alpha d \log \log(d)}{\log(d)} \rightarrow +\infty \).

Associative memories

Store
Associative memories

Some piece of information

Store
Associative memories

Another piece of information
Associative memories

Another piece of information

Store
Associative memories

![Graph showing storage capacity versus amount of stored messages (M)](image)

- **Store**
- **Storage capacity**

---

Vincent Gripon (IMT-Atlantique)
Associative memories

Retrieve

Storage capacity

Noisy version of previously stored piece of information
Associative memories

Retrieve

Storage capacity

Noisy version of previously stored piece of information

Corresponding previously stored piece of information
<table>
<thead>
<tr>
<th>Hopfield</th>
<th>Willshaw</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Framework</strong></td>
<td><strong>Framework</strong></td>
</tr>
<tr>
<td>$x \in {-1, 1}^d$</td>
<td>$x \in {0, 1}^d, |x|_0 \ll d$</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td><strong>Memory</strong></td>
</tr>
<tr>
<td>$xx^\top$</td>
<td>$xx^\top$</td>
</tr>
<tr>
<td><strong>Aggregation</strong></td>
<td><strong>Aggregation</strong></td>
</tr>
<tr>
<td>$W = \sum_{x \in X} xx^\top$</td>
<td>$W = \max_{x \in X} xx^\top$</td>
</tr>
<tr>
<td>using classical linear algebra</td>
<td>using Boolean algebra</td>
</tr>
<tr>
<td><strong>Search</strong></td>
<td><strong>Search</strong></td>
</tr>
<tr>
<td>$\text{sgn}(W\tilde{x})$</td>
<td>$\text{sgn}(W\tilde{x} - s1)$</td>
</tr>
</tbody>
</table>

**Summary**

<table>
<thead>
<tr>
<th>Framework</th>
<th>Memory</th>
<th>Aggregation</th>
<th>Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hopfield</td>
<td>Willshaw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x \in {-1, 1}^d$</td>
<td>$x \in {0, 1}^d, |x|_0 \ll d$</td>
<td>$W = \sum_{x \in X} xx^\top$</td>
<td>$\text{sgn}(W\tilde{x})$</td>
</tr>
<tr>
<td>$xx^\top$</td>
<td>$xx^\top$</td>
<td>$W = \max_{x \in X} xx^\top$</td>
<td>$\text{sgn}(W\tilde{x} - s1)$</td>
</tr>
</tbody>
</table>

**Structured Clique Networks**

Vincent Gripon (IMT-Atlantique)  
March 4th, 2021  
15/25
<table>
<thead>
<tr>
<th></th>
<th>Hopfield</th>
<th>Willshaw</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Framework</strong></td>
<td>$x \in {-1, 1}^d$</td>
<td>$x \in {0, 1}^d, |x|_0 \ll d$</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td>$xx^\top$</td>
<td>$xx^\top$</td>
</tr>
<tr>
<td><strong>Aggregation</strong></td>
<td>$W = \sum_{x \in X} xx^\top$</td>
<td>$W = \max_{x \in X} xx^\top$</td>
</tr>
<tr>
<td></td>
<td>using classical linear algebra</td>
<td>using Boolean algebra</td>
</tr>
<tr>
<td><strong>Search</strong></td>
<td>$\text{sgn}(W \tilde{x})$</td>
<td>$\text{sgn}(W \tilde{x} - s1)$</td>
</tr>
<tr>
<td></td>
<td>Hopfield</td>
<td>Willshaw</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>Framework</td>
<td>$\mathbf{x} \in {-1, 1}^d$</td>
<td>$\mathbf{x} \in {0, 1}^d$, $|\mathbf{x}|_0 \ll d$</td>
</tr>
<tr>
<td>Memory</td>
<td>$\mathbf{xx}^\top$</td>
<td>$\mathbf{xx}^\top$</td>
</tr>
<tr>
<td>Aggregation</td>
<td>$\mathbf{W} = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{xx}^\top$</td>
<td>$\mathbf{W} = \max_{\mathbf{x} \in \mathbf{X}} \mathbf{xx}^\top$</td>
</tr>
<tr>
<td></td>
<td>using classical linear algebra</td>
<td>using Boolean algebra</td>
</tr>
<tr>
<td>Search</td>
<td>$\text{sgn}(\mathbf{W} \tilde{\mathbf{x}})$</td>
<td>$\text{sgn}(\mathbf{W} \tilde{\mathbf{x}} - s\mathbf{1})$</td>
</tr>
</tbody>
</table>
## Summary

<table>
<thead>
<tr>
<th>Hopfield</th>
<th>Willshaw</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Framework</strong></td>
<td>$\mathbf{x} \in {-1, 1}^d$</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td>$\mathbf{xx}^\top$</td>
</tr>
<tr>
<td><strong>Aggregation</strong></td>
<td>$\mathbf{W} = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{xx}^\top$</td>
</tr>
<tr>
<td></td>
<td>$\mathbf{W} = \mathbf{XX}^\top$</td>
</tr>
<tr>
<td><strong>Search</strong></td>
<td>$\text{sgn}(\mathbf{W}\hat{\mathbf{x}})$</td>
</tr>
</tbody>
</table>

*using classical linear algebra*  
*using Boolean algebra*
### Summary

<table>
<thead>
<tr>
<th>Hopfield</th>
<th>Willshaw</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Framework</strong></td>
<td>( x \in {-1, 1}^d )</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td>( xx^\top )</td>
</tr>
<tr>
<td><strong>Aggregation</strong></td>
<td>( W = \sum_{x \in X} xx^\top )</td>
</tr>
<tr>
<td>&amp; ( W = XX^\top )</td>
<td>( W = XX^\top )</td>
</tr>
<tr>
<td><strong>Search</strong></td>
<td>( \text{sgn}(W\tilde{x}) )</td>
</tr>
</tbody>
</table>

- Using classical linear algebra
- Using Boolean algebra
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
Adding structure to Willshaw networks

\( c \) clusters

\( \ell \) neurons/cluster
Leveraging the structure

Willshaw rule for retrieval

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

New rule for retrieval

- \[ u(x)_{ij} = 1 \iff \sum_{i'} \max_{j'} W_{i'j',ij} x_{i'j'} \text{ is maximal in cluster } i, \]
- \[ u(x)_{ij} = 1 \iff \bigwedge_{i'} \bigvee_{j'} W_{i'j',ij} \bigwedge x_{i'j'}, \]
  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

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

New rule for retrieval

- \( u(x)_{ij} = 1 \iff \sum_{i'} \max_{j'} W_{i'j',ij} x_{i'j'} \) is maximal in cluster \( i \),
- \( u(x)_{ij} = 1 \iff \land_{i'} \lor_{j'} W_{i'j',ij} \land x_{i'j'} \), 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

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

New rule for retrieval

- $u(x)_{ij} = 1 \iff \sum_{i'} \max_{j'} W_{i'j',ij} x_{i'j'}$ is maximal in cluster $i$,
- $u(x)_{ij} = 1 \iff \land_{i'} \lor_{j'} W_{i'j',ij} \land x_{i'j'}$, 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**

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

**New rule for retrieval**

- \( u(x)_{ij} = 1 \iff \sum_{i'} \max_{j'} W_{i'j',ij} x_{i'j'} \) is maximal in cluster \( i \),
- \( u(x)_{ij} = 1 \iff \land_{i'} \lor_{j'} W_{i'j',ij} \land x_{i'j'}, \) if neurons in erased clusters are all initialized active.
- Both storage and retrieval are performed using Boolean algebra.
Approaching $\log(2)$

- Let us choose: $\alpha c = 2 \log_2(\ell)$,
- $\eta \sim \frac{nc \log_2(l)}{(c^2)\ell^2} \sim \frac{\alpha n}{\ell^2}$,
- 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: $P_e \approx p^{(c^2)}$,
  none of them: $P_{e*} \leq P_e \ell^c$,
  $P_{e*} \leq \exp\left(\frac{c^2}{2} \left[ \log_2(p) + \alpha \right]\right) \rightarrow 0 \text{ if } \alpha = -\beta \log_2(p), \beta < 1$.
- Conclusion: $\eta \sim \beta \log_2(1 - p) \log_2(p) \log(2)$
Approaching $\log(2)$

- Let us choose: $\alpha c = 2 \log_2(\ell)$,
- $\eta \sim \frac{nc \log_2(\ell)}{\left(\frac{c}{2}\right)^2 \ell^2} \sim \frac{\alpha n}{\ell^2}$,
- 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:
  $P_e^* \leq P_e \ell^c$,
  $P_e^* \leq \exp \left( \frac{c^2}{2} \left[ \log_2(p) + \alpha \right] \right) \to 0\text{ if } \alpha = -\beta \log_2(p), \beta < 1.$
- Conclusion: $\eta \sim \beta \log_2(1 - p) \log_2(p) \log(2)$
Memory efficiency (with some approximations)

Approaching $\log(2)$

- Let us choose: $\alpha c = 2 \log_2(\ell)$,
- $\eta \sim \frac{nc \log_2(\ell)}{(\frac{c}{2})\ell^2} \sim \frac{\alpha n}{\ell^2}$,
- 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: $P_e \approx p^{(\frac{c}{2})}$, none of them:
  $P_{e^*} \leq P_e \ell^c$,
  $P_{e^*} \leq \exp\left(\frac{c^2}{2} \left[\log_2(p) + \alpha\right]\right) \to 0$ if $\alpha = -\beta \log_2(p)$, $\beta < 1$.
- Conclusion: $\eta \sim \beta \log_2(1 - p) \log_2(p) \log(2)$
Approaching \( \log(2) \)

- Let us choose: \( \alpha c = 2 \log_2(\ell) \),
- \( \eta \sim \frac{nc \log_2(\ell)}{(c/2)\ell^2} \sim \frac{\alpha n}{\ell^2} \),
- 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: \( P_e \approx p^{(c \choose 2)} \), none of them:
  \[ P_e^* \leq P_e \ell^c, \]
  \[ P_e^* \leq \exp\left(\frac{c^2}{2} \left[ \log_2(p) + \alpha \right]\right) \rightarrow 0 \text{ if } \alpha = -\beta \log_2(p), \beta < 1. \]
- Conclusion: \( \eta \sim \beta \log_2(1 - p) \log_2(p) \log(2) \)
Memory efficiency (with some approximations)

**Approaching \( \log(2) \)**

- Let us choose: \( \alpha c = 2 \log_2(\ell) \),
- \( \eta \sim \frac{nc \log_2(\ell)}{(c/2)\ell^2} \sim \frac{\alpha n}{\ell^2} \),
- 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: \( P_e \approx p_{(2)}^{(c)} \), none of them:
  \( P_e^* \leq P_e \ell^c \),
  \( P_e^* \leq \exp \left( \frac{c^2}{2} \left[ \log_2(p) + \alpha \right] \right) \rightarrow 0 \) if \( \alpha = -\beta \log_2(p), \beta < 1 \).
- **Conclusion**: \( \eta \sim \beta \log_2(1 - p) \log_2(p) \log(2) \)
Approaching $\log(2)$

- Let us choose: $\alpha c = 2 \log_2(\ell)$,
- $\eta \sim \frac{nc \log_2(\ell)}{(\frac{c}{2})\ell^2} \sim \frac{\alpha n}{\ell^2}$,
- 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: $P_e \approx p^{(c)}$, none of them:
  $P_e^* \leq P_e \ell^c$,
  
  $P_e^* \leq \exp \left( \frac{c^2}{2} \left[ \log_2(p) + \alpha \right] \right) \to 0$ 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 \( \rho c \) initial neurons, then for \( \alpha < -\log(1 - \exp(-1/(1 - \rho))) \), probability to retrieve the vector goes to 1.

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 \( \rho c \) initial neurons, then for \( \alpha < - \log(1 - \exp(-1/(1 - \rho))) \), probability to retrieve the vector goes to 1.

### Performance in search

<table>
<thead>
<tr>
<th>Amari (Hopfield)</th>
<th>Willshaw</th>
<th>Structured</th>
</tr>
</thead>
<tbody>
<tr>
<td>No structure</td>
<td>No structure</td>
<td>Clusters</td>
</tr>
<tr>
<td>Weights</td>
<td>No weights</td>
<td>No weights</td>
</tr>
</tbody>
</table>

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


![Graph showing performance in search](chart.png)
False positive rate for various number of clusters $c$ and $\ell = 512$ neurons per cluster.

With 1% of error, memory efficiency is 137.1%
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
Application to Implementation of Search Engines

\[ v^*_i(f, i, j) = \left( \bigwedge_{i'=0}^{\psi-1} \bigvee_{j'=0}^{\gamma-1} u^{(f, i, j)}(f', i', j') \bigwedge v(f, i, j) \right) \]

Field 0  Field 1  Field 2  Field 3
(Eg. Keyword) (Eg. Year) (Eg. Name) (File ID)

Entry: (010...001)-(001...011)-(001...101)-(100...000)

Extract: (00...01)-(00...11)-(00...01)-(00...11)

Segment: (0,1)-(13)-(0,1)-(1,16,15)

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

Application to Approximate Nearest Neighbor Search

**Offline processing:**

Step 1: initialize matrices and counters

\[
W^1 = x^1 x^1 \mathsf{T} \\
W^2 = x^2 x^2 \mathsf{T} \\
W^3 = x^3 x^3 \mathsf{T} \\
\vdots \\
W^n = x^n x^n \mathsf{T}
\]

\[
u^1 = 1 \\
u^2 = 1 \\
u^3 = 1 \\
\vdots \\
u^n = 1
\]

Step 2: allocate each remaining vector \(x^\theta\):

```
\begin{array}{c}
x^{\theta+1} \\
x^{\theta+2} \\
x^{\theta+3} \\
x^{\theta+4} \\
x^{\theta+5} \\
x^{\theta+6} \\
x^{\theta+7} \\
x^{\theta}
\end{array}
```

```
\begin{array}{c}
W^1 \\
W^2 \\
W^3 \\
\vdots \\
W^n
\end{array}
```

**Online processing (request \(y^v\)):**

Step 1: compute scores:

\[
y^v W^1 y^v \\
y^v W^2 y^v \\
y^v W^3 y^v \\
\vdots \\
y^v W^n y^v
\]

Step 2: for the \(p\) largest scores, exhaustively search:

```
\begin{array}{c}
y^v x^{\theta_1} \\
y^v x^{\theta_2} \\
y^v x^{\theta_3} \\
\vdots \\
y^v x^{\theta_p}
\end{array}
```

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Scan Time</th>
<th>Recall@1</th>
<th>Scan Time</th>
<th>Recall@1</th>
<th>Scan Time</th>
<th>Recall@1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random kd-trees [1]</td>
<td>0.04</td>
<td>0.6</td>
<td>0.22</td>
<td>0.8</td>
<td>3.1</td>
<td>0.95</td>
</tr>
<tr>
<td>K-means trees [1]</td>
<td>0.06</td>
<td>0.6</td>
<td>0.25</td>
<td>0.8</td>
<td>2.8</td>
<td>0.99</td>
</tr>
<tr>
<td>Proposed method (hybrid)</td>
<td>0.17</td>
<td>0.6</td>
<td>0.25</td>
<td>0.8</td>
<td>1.1</td>
<td>0.99</td>
</tr>
<tr>
<td>ANN [16]</td>
<td>3.7</td>
<td>0.6</td>
<td>8.2</td>
<td>0.8</td>
<td>24</td>
<td>0.95</td>
</tr>
<tr>
<td>LSH [22]</td>
<td>6.4</td>
<td>0.6</td>
<td>11.1</td>
<td>0.8</td>
<td>28</td>
<td>0.98</td>
</tr>
</tbody>
</table>
DecisiveNets: from DNNs to DAMs

\[ \hat{y}[\ell(i - 1) : \ell i] = \sigma_t(x[\ell(i - 1) : \ell i]), \forall i, \text{ where} \]

\[ \sigma_t(z) = \frac{\text{softmax}(t \cdot \sigma(z))}{\max(\text{softmax}(t \cdot \sigma(z)))} \sigma(z). \]

<table>
<thead>
<tr>
<th>( \ell )</th>
<th>Resnet18 and CIFAR-10</th>
<th>Resnet50 and CIFAR-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (baseline)</td>
<td>95.21%</td>
<td>78.50%</td>
</tr>
<tr>
<td>2</td>
<td>95.25%</td>
<td>79.23%</td>
</tr>
<tr>
<td>4</td>
<td>94.65%</td>
<td>76.58%</td>
</tr>
<tr>
<td>8</td>
<td>92.95%</td>
<td>70.46%</td>
</tr>
<tr>
<td>16</td>
<td>88.95%</td>
<td>64.36%</td>
</tr>
<tr>
<td>32</td>
<td>84.90%</td>
<td>61.07%</td>
</tr>
<tr>
<td>64</td>
<td>78.28%</td>
<td>53.05%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \ell )</th>
<th>Resnet18 and CIFAR10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>clean data</td>
</tr>
<tr>
<td>1</td>
<td>95.21%</td>
</tr>
<tr>
<td>2</td>
<td>95.25%</td>
</tr>
<tr>
<td>4</td>
<td>94.65%</td>
</tr>
<tr>
<td>8</td>
<td>92.95%</td>
</tr>
<tr>
<td>16</td>
<td>88.95%</td>
</tr>
<tr>
<td>32</td>
<td>84.90%</td>
</tr>
<tr>
<td>64</td>
<td>78.28%</td>
</tr>
</tbody>
</table>


Vincent Gripon (IMT-Atlantique) | Structured Clique Networks | March 4th, 2021 | 24/25
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