Learning to modularize

Sameet Sreenivasan
Zoltán Toroczkai
Modularity

The presence of functional modules each of which is a “discrete entity whose function is separable from those of other modules”
Nature versus Nurture
Nature versus Nurture

Evolutionary Adaptation:

Alternating environments

“Related” modular goals in each environment
Nature versus Nurture

Evolutionary Adaptation:

Alternating environments

“Related” modular goals in each environment

Networks which evolve to successfully accomplish both goals are modular.

N. Kashtan et al.,(2005)
Can modularity arise through learning?
Can modularity arise through learning?
Necessary ingredients
Necessary ingredients

Multiple tasks
to be performed by network
Necessary ingredients

Multiple tasks
to be performed by network

Network learns through Supervised learning
Learning to perform two independent Boolean functions

\[ F(S_1) \quad G(S_2) \]

\[ S_1 \quad S_2 \]
Learning to perform two independent Boolean functions

$F(S_1)$  $G(S_2)$

$S_1$  $S_2$
Learning to perform two independent Boolean functions

F(S₁)  G(S₂)

S₁  S₂

 Modules present?

Nodes ➞ continuous sigmoidal units with thresholds
How does the network learn?

Backpropagation

Define an error $E$.

Minimize $E$ using gradient descent.
How does the network learn?

Two choices of $E$ used in this study:

Sum square error

$$E_{SS} = \sum_k (t_k - y_k)^2$$

Cross Entropy

$$E_{CE} = \sum_k [t_k \ln y_k + (1 - t_k) \ln (1 - y_k)]$$
Finding functional modules

Knockout a node

Measure performance errors

Plot the errors
Finding functional modules

Knockout a node
Measure performance errors
Plot the errors

XOR(S1)    XOR(S2)
Finding functional modules

Knockout a node

Measure performance errors

Plot the errors
Finding functional modules

XOR(S1)  XOR(S2)

Knockout a node
Measure performance errors
Plot the errors
Finding functional modules

Knockout a node

Measure performance errors

Plot the errors

\( \epsilon_F, \epsilon_G \)

Scatter

\[
\begin{align*}
Y_1 &= \text{XOR}(S_1) \\
Y_2 &= \text{XOR}(S_2) \\
Y_1 &= F(S_1) \\
Y_2 &= G(S_2) \\
\epsilon_F, \epsilon_G
\end{align*}
\]
Finding functional modules

XOR(S1)   XOR(S2)

Knockout a node
Measure performance errors
Plot the errors \( \epsilon_F, \epsilon_G \)

Histogram

\[
\begin{align*}
Y_1 &= \text{XOR}(S_1) \\
Y_2 &= \text{XOR}(S_2) \\
Y_1 &= F(S_1) \\
Y_2 &= G(S_2)
\end{align*}
\]

\[
\begin{align*}
\theta = 0 & \quad \theta = \pi/2 \\
\theta = \pi/4
\end{align*}
\]
Sum square error

Results for an ensemble of 1000 networks

8 i/p nodes, 12 hidden layer nodes

$\theta = 0, \theta, \theta = \pi/2$
Sum square error

Results for an ensemble of 1000 networks

8 i/p nodes, 12 hidden layer nodes

Nodes specialized for XOR1
Nodes specialized for XOR2

$\theta = 0$
$\theta = \frac{\pi}{2}$
Cross Entropy error

Results for an ensemble of 1000 networks

Number of hidden layer neurons = 12
Functionally specialized cortical pathways

Ventral Pathway
Locates objects

Dorsal Pathway
Identifies objects
Functionally specialized cortical pathways

- **Ventral Pathway**: Locates objects
  - WHERE

- **Dorsal Pathway**: Identifies objects
  - WHAT
Learning WHAT and WHERE

9 objects

18 output nodes
9 WHAT nodes
9 WHERE nodes

9 locations
Learning with Sum square error

Results for an ensemble of 100 networks

$H(\theta)$

45 hidden layer neurons
Learning with Sum square error

Results for an ensemble of 100 networks

$H(\theta)$

- "What" pathway
- "Where" pathway

45 hidden layer neurons
Learning with Cross Entropy error

Results for an ensemble of 100 networks

$H(\theta)$

“What” pathway

45 hidden layer neurons
What does network structure tell us?

Inferring structural modularity

Renormalize weights:

\[ W_{jk} = \frac{|w_{jk}|}{\sum_k |w_{jk}|} \]

Define quantities:

\[ M^j_1 = \sum_{k \in F} W_{jk} \]
\[ M^j_2 = \sum_{k \in G} W_{jk} \]

Structural Modularity:

\[ M = \frac{1}{NH} \sum_j \frac{|M^j_1 - M^j_2|}{M^j_1 + M^j_2} \]
What does network structure tell us?

Inferring structural modularity

Renormalize weights:

\[ W_{jk} = \frac{|w_{jk}|}{\sum_k |w_{jk}|} \]

Define quantities:

\[ M_1^j = \sum_{k \in F} W_{jk} \]
\[ M_2^j = \sum_{k \in G} W_{jk} \]

Structural Modularity:

\[ M = \frac{1}{NH} \sum_j \frac{|M_1^j - M_2^j|}{M_1^j + M_2^j} \]
What does network structure tell us?

Inferring structural modularity

Renormalize weights:

\[ W_{jk} = \frac{|w_{jk}|}{\sum_i |w_{ij}|} \]

Define quantities:

\[ M_1^j = \sum_{k \in F} W_{jk} \]
\[ M_2^j = \sum_{k \in G} W_{jk} \]

Structural Modularity:

\[ M = \frac{1}{N_H} \sum_j \frac{|M_1^j - M_2^j|}{M_1^j + M_2^j} \]
What does network structure tell us?

Inferring structural modularity

Renormalize weights:

\[ W_{jk} = \frac{|w_{jk}|}{\sum_k |w_{jk}|} \]

Define quantities:

\[ M_1^j = \sum_{k \in F} W_{jk} \]
\[ M_2^j = \sum_{k \in G} W_{jk} \]

Structural Modularity:

\[ M = \frac{1}{N_H} \sum_j \frac{|M_1^j - M_2^j|}{M_1^j + M_2^j} \]
What does network structure tell us?

Inferring structural modularity

Renormalize weights:

\[ W_{jk} = \frac{|w_{jk}|}{\sum_k |w_{jk}|} \]

Define quantities:

\[ M_1^j = \sum_{k \in F} W_{jk} \]
\[ M_2^j = \sum_{k \in G} W_{jk} \]

Structural Modularity:

\[ M = \frac{1}{N_H} \sum_j \frac{|M_1^j - M_2^j|}{M_1^j + M_2^j} \]

\[ M \to 1 \implies \text{High structural modularity} \]
\[ M \to 0 \implies \text{Low structural modularity} \]
What does structure tell us?

\[ P(M) \]
What does structure tell us?

SSE

What does structure tell us?

CE

Structurally modular?
What does structure tell us?

Structurally modular?

NO

NO

P(M)

SSE

CE

M

M
What does structure tell us?

Structurally modular?
NO

Functionally modular?
SSE
CE

P(M)

Structurally modular?
NO

Functionally modular?
NO
What does structure tell us?

**SSE**

- Structurally modular? **NO**
- Functionally modular? **NO**

**CE**

- Structurally modular? **NO**
- Functionally modular? **NO**

\[ P(M) \]
What does structure tell us?

Structurally modular?
NO

Functionally modular?
YES!

NO
Learning (or phenotypic adaptation) can lead to the emergence of functionally specialized modules.

The presence (or absence) of structural modularity is not necessarily indicative of the presence (or absence) of functional modules.
1. Start with a three layer feedforward network

   Initial weights \( w_{jk} \in (-1, 1) \)

   Initial thresholds \( \theta_j \in (0, 1) \)

2. Train the network for multiple tasks using backpropagation.

   Online training with all possible input patterns presented a prescribed number of times.
3. When training has concluded probe the network for functional modules using single node removals.

4. Repeat for networks with different initial conditions.
Boolean network

Results of the scatter
Boolean network

Results of the scatter

\[ \mathcal{E}_G, \mathcal{E}_F \]

\[ \mathcal{E}_G, 0 \leq \mathcal{E}_G \leq 10 \]

\[ \mathcal{E}_F, 0 \leq \mathcal{E}_F \leq 10 \]

Modular system

Non-Modular system

\[ \mathcal{E}_G = \frac{|w_{ij}|}{\sum_i |w_{ij}|} \]

\[ M_1(j) = \sum_{i \in I_1} W_{ij} \]

\[ M_2(j) = \sum_{i \in I_2} W_{ij} \]

\[ M = \frac{1}{N_H} \sum_j |M_1(j) - M_2(j)| \]

\[ M \rightarrow 0 \Rightarrow \text{Low Modularity} \]

\[ \theta \]

\[ \mathcal{E}_{\text{Ex}} = \sum_k (t_k - y_k)^2 \]

\[ \mathcal{E}_{\text{CE}} = \sum_k [t_k \ln y_k + (1 - t_k) \ln (1 - y_k)] \]

\[ P(M) \]

\[ y_k, t_k, \theta \]

\[ \text{ESSE} = \sum_k (t_k - y_k)^2 \]

\[ \text{ECE} = \sum_k [t_k \ln y_k + (1 - t_k) \ln (1 - y_k)] \]
Evolution of weights

For a node that contributes to the computation of \( \text{XOR}(x_1 x_2 x_3 x_4) \)

\[
W_{ij} = \frac{|w_{ij}|}{\sum w_{ij}}
\]

\[
M_{1j} = \sum_{i \in I_1} W_{ij}
\]

\[
M_{2j} = \sum_{i \in I_2} W_{ij}
\]

\[
M = 1 \cdot \frac{1}{NH} \sum_j |M_{1j} - M_{2j}|
\]

\[
M \rightarrow 1 \Rightarrow \text{High Modularity}
\]

\[
M \rightarrow 0 \Rightarrow \text{Low Modularity}
\]

\[
< M > \Rightarrow P(M)
\]
Evolution of weights

For a node that contributes to the computation of \( \text{XOR}(x_1 x_2 x_3 x_4) \)

\[
W_{ij} = \frac{|w_{ij}|}{\sum_i |w_{ij}|}
\]

\[
M_1^j = \sum_{i \in I_1} W_{ij}
\]

\[
M_2^j = \sum_{i \in I_2} W_{ij}
\]

\[
M = \frac{1}{N_H} \sum_j |M_1^j - M_2^j|
\]

\( M \rightarrow 1 \Rightarrow \text{High Modularity} \)

\( M \rightarrow 0 \Rightarrow \text{Low Modularity} \)

\( < M > \)

\( P(M) \)
Evolution of weights

For a node that contributes to the computation of $\text{XOR}(x_1 x_2 x_3 x_4)$

For a node that contributes to the computation of $\text{XOR}(x_4 x_5 x_7 x_8)$
Learning Curves

SSE

Learning rate $\eta = 1.5$

CE

Learning rate $\eta = 0.1$
The WHERE task is learnt faster than WHAT in both cases.
The WHERE task is learnt faster than WHAT in both cases.

The final performance of the WHAT task is better when CE is used.