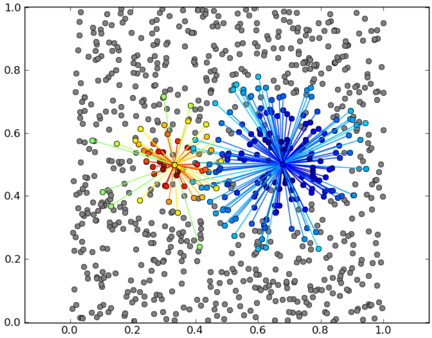


The Connection-set Algebra

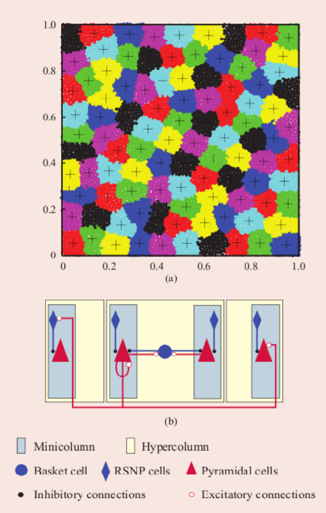
Mikael Djurfeldt, INCF/KTH

Introduction

The connection-set algebra

- Notation for description of connectivity in neuronal network models
- Connection structure - which connections exist?
- Parameters associated with connections (weights, delays, ...)
- Geometry
- Algebra for computing with connectivity
- Scalable support for setting up connectivity on serial and parallel computers

Motivation



Layer II/III cortex model
Djurfeldt et al (2008)

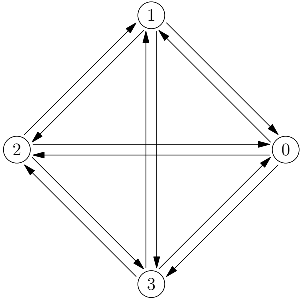
Structure at three levels:

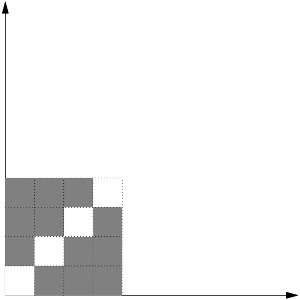
- cells
- minicolumns
- hypercolumns

$$
\begin{aligned}
C_{\mathrm{bp}} &= \bar{\rho}(0.7) \cap \mathbf{B}(h_{\mathrm{b}}, h_{\mathrm{p}})\bar{\delta} \\
C_{\mathrm{rp}} &= \bar{\rho}(0.7) \cap \mathbf{B}(m_{\mathrm{r}}, m_{\mathrm{p}})\bar{\delta} \\
C^{\mathrm{l}}_{\mathrm{pp}} &= \bar{\rho}(0.25) \cap \mathbf{B}(m_{\mathrm{p}})\bar{\delta} - \bar{\delta} \\
C^{\mathrm{g}}_{\mathrm{pp}} &= \bar{\rho}\mathbf{B}(m_{\mathrm{p}})\theta(a_e P) - \mathbf{B}(m_{\mathrm{p}})\bar{\delta} \\
C_{\mathrm{pr}} &= \bar{\rho}\mathbf{B}(m_{\mathrm{p}}, m_{\mathrm{r}})\theta(-a_i P)) - \mathbf{B}(m_{\mathrm{p}}, m_{\mathrm{r}})\bar{\delta} \\
C_{\mathrm{pb}} &= \bar{\rho}(0.7) \cap \mathbf{B}(m_{\mathrm{p}}, m_{\mathrm{b}})\text{n\_closest\_pre}(g_m, h_m, 8)
\end{aligned}
$$

Motivation

All-to-all without self-connections



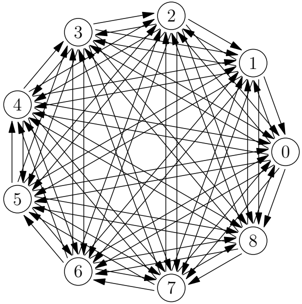


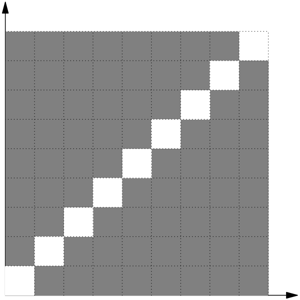
Example 1: All-to-all connectivity without self-connections

```
for i in range (0, 4):
  for j in range (0, 4):
    if i != j:
      connect (i, j)
```

Motivation

All-to-all without self-connections



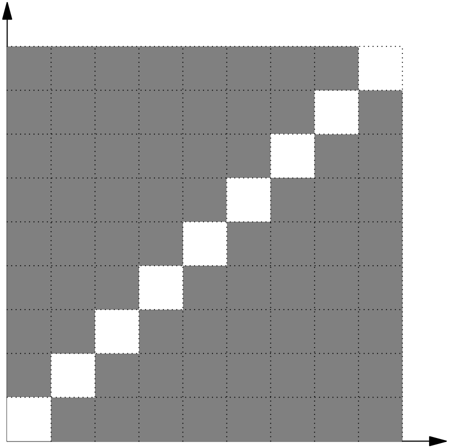


Example 1: All-to-all connectivity without self-connections

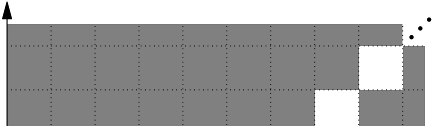
```
for i in range (0, 9):
  for j in range (0, 9):
    if i != j:
      connect (i, j)
```

Motivation

Finite — structure + size



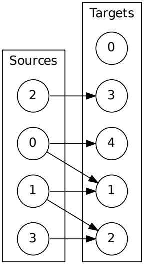
Infinite — only structure



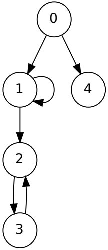
Connection-set

- Mask $\overline{M} : \mathcal{I} \times \mathcal{J} \to \{\mathcal{F}, \mathcal{T}\}$ or $\{(i_0, j_0), (i_1, j_1), \ldots\}$

  Example: $\{(0,1),(1,1),(1,2),(3,2),(2,3),(0,4)\}$



Separate source and target enums



Same source and target enums

- Value set $V : \mathcal{I} \times \mathcal{J} \to \mathbb{R}^N$
- Connection-set $\langle \overline{M}, V_0, V_1, \ldots \rangle$

CSA

Snippets of CSA formalism

- Index sets

  $\mathcal{I} = \mathbb{N}_0$ infinite index set (natural numbers)

  $\mathcal{I} = \{m..n\}$ finite index set

- Cartesian product on index sets

  $$\mathcal{I} \times \mathcal{J} = \{(i,j) | i \in \mathcal{I}, j \in \mathcal{J}\}$$

- Elementary masks

  $\bar{\Omega}$ the set of all connections (index pairs)

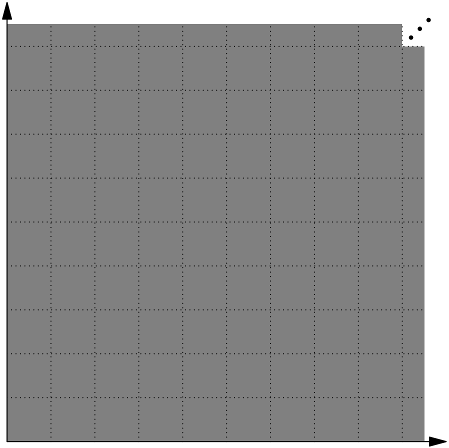
  $\bar{\delta}$ the set of all (i, i)

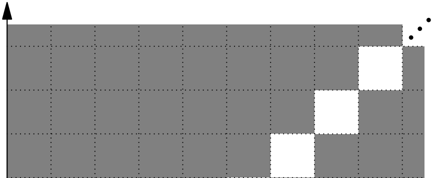
- Operators on connection-sets

  $\cap$ intersection

  $-$ set difference

All-to-all without self-connections





The Connection-set Algebra

Mikael Djurfeldt

Python demo

- Python demo implementation beta-released under GPL at INCF Software Center: `http://software.incf.org/software/csa`
- Distribution contains a tutorial for hands-on-learning
- Part of Debian/Squeeze
- Supported in PyNN (CSAConnector)
- NEST connect can use native CSA objects (csanest branch)
- Support in NineML (experimental branch)

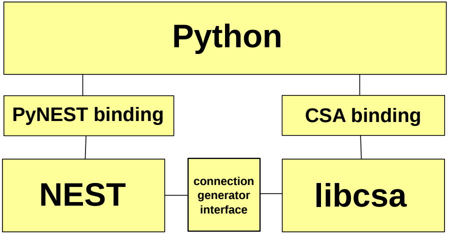
Demo

Hands on demo

C++ library

- C++ library under development
- Planned release autumn 2012

ConnectionGenerator



`int arity()`: Return the number of values associated with this iterator. Values can be parameters like weight, delay, time constants, or others.

`int size()`: Return the number of connections represented by this iterator.

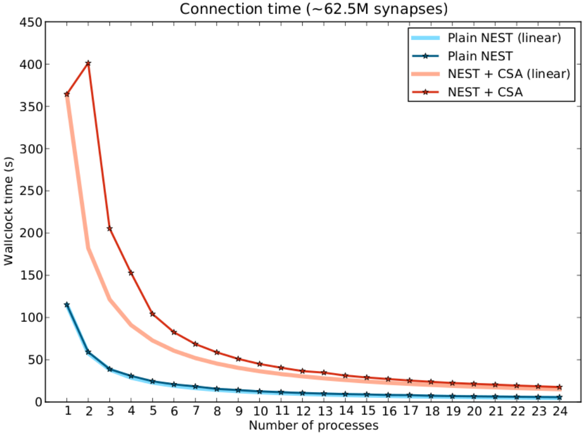
`void setMask(Mask& mask)`: Inform the generator about which source and target indices exist. A mask represents a subset of the nodes in the network.

`void setMask(std::vector<Mask>& masks, int local)`: Parallel case.

`void start()`: Start an iteration.

`bool next(int& source, int& target, double* value)`: Advance to the next connection. Return false if no more connections.

ConnectionGenerator



Eppler et al (2011) INCF congress poster