A Technical Overview of the Neural Engineering Framework

Terrence C. Stewart

Centre for Theoretical Neuroscience technical report (2012)

University of Waterloo, Canada

Outline

- Intro
- Motivation
- Representation
- Computation
- Dynamics
- Symbol Processing
- Spaun & Nengo
- Limitations

Intro

What is Neural Engineering Framework (NEF)?

➢ A general methodology to build neural models that are
- Large-scale
- Biologically plausible

➢ Acts as a neural compiler
- You specify
  - Neurons' properties, values,
  - Functions to be computed.
- NEF solves for the connection weights.

Motivation

It's already hard enough to produce realistic cognitive behavior.

❑Why put extra overhead & constraint on models?

1. Evaluate our theories
    - Produce correct behavior in the same way as real brain
        - Comparable firing patterns & neural connectivity
    - Produce same effects of neural degeneration, lesioning, deep brain simulation, drug treatments, etc.
        - Comparable timing caused by neurons' biophysical properties

    ✓End goal: Create new types of predictions

continued…

2. Explore new types of algorithms
   - You don't get exact implementation.
     - NEF forces you to use the basic operations available to neurons.
   - All classes of algorithms can't be implemented in human brain.
     - Constraints: timing, robustness, #neurons involved

✓End goal: Find plausible method for implementing symbol-like cognitive reasoning

Representation

Distributed representation

- Activity of a group of neurons
- Value being represented

Encoding:

$$a_i = G(\alpha_i \mathbf{e}_i \cdot \mathbf{x} + b_i)$$

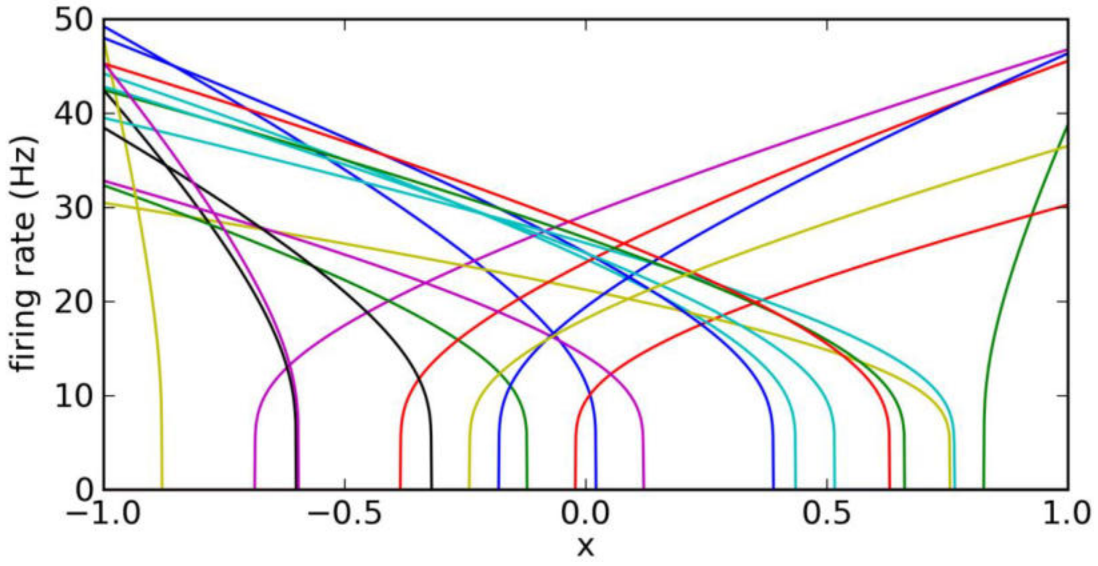
activity | model | gain | encoding vector | value | constant background bias current

Decoding:

$$\hat{\mathbf{x}} = \sum a_i \mathbf{d}_i$$

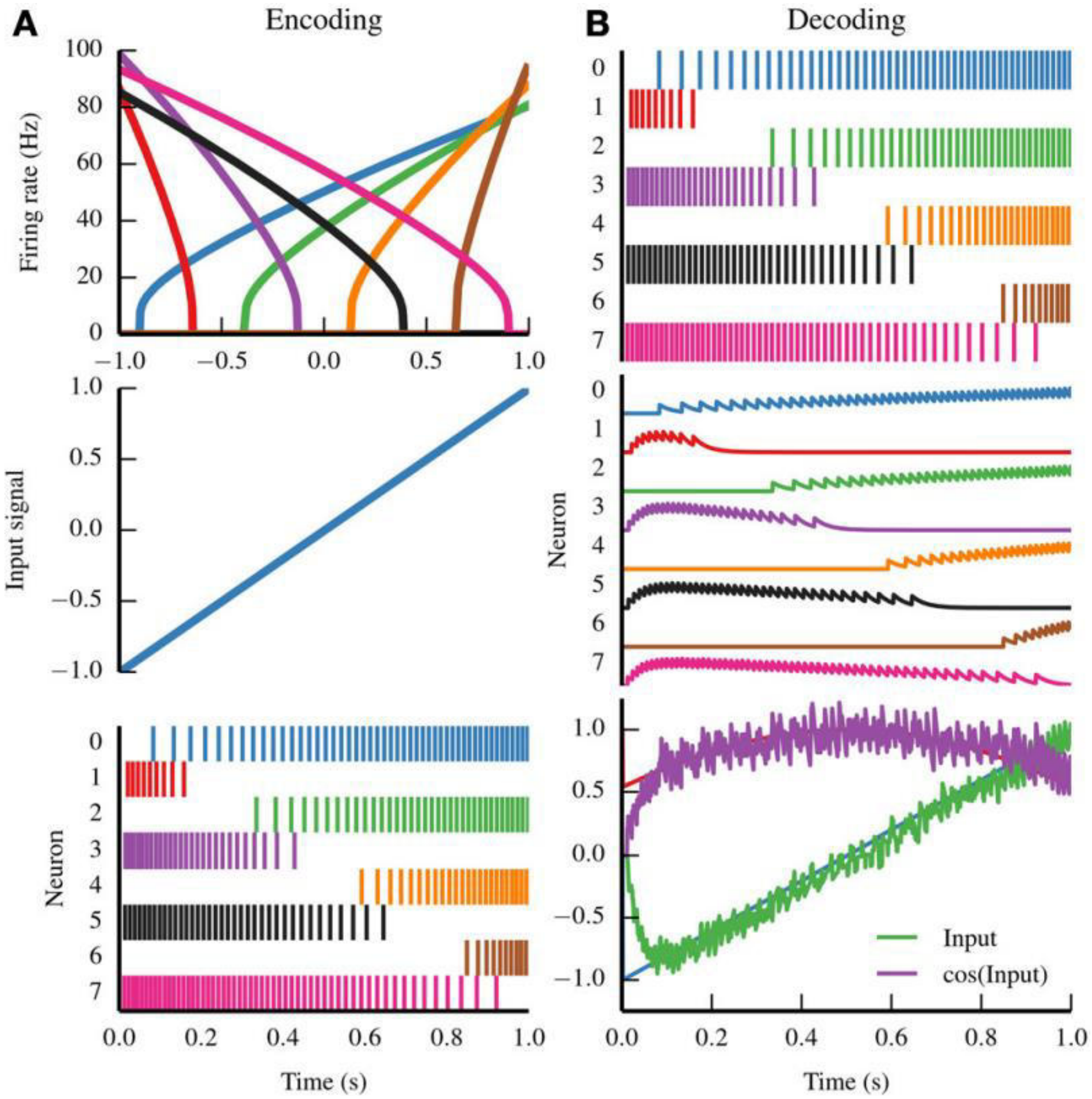
where,

$$\mathbf{d} = \Gamma^{-1}\Upsilon \qquad \Gamma_{ij} = \sum_{\mathbf{x}} a_i a_j \qquad \Upsilon_j = \sum_{\mathbf{x}} a_j \mathbf{x}$$



Average firing rates for 20 different LIF neurons. α_i and b_i are randomly chosen to give a realistic range of responses. Neurons whose firing increases with $\mathbf{x}$ have $\mathbf{e}_i = 1$, while the other neurons have $\mathbf{e}_i = -1$.

continued...


A
Encoding
B
Decoding
Firing rate (Hz)
100
80
60
40
20
0
−1.0
−0.5
0.0
0.5
1.0
Input signal
Neuron
0
1
2
3
4
5
6
7
Time (s)
0.0
0.2
0.4
0.6
0.8
1.0
Input
cos(Input)

Computation

Let’s compute $f(\mathbf{x}) = \mathbf{x}$, using two populations A & B.

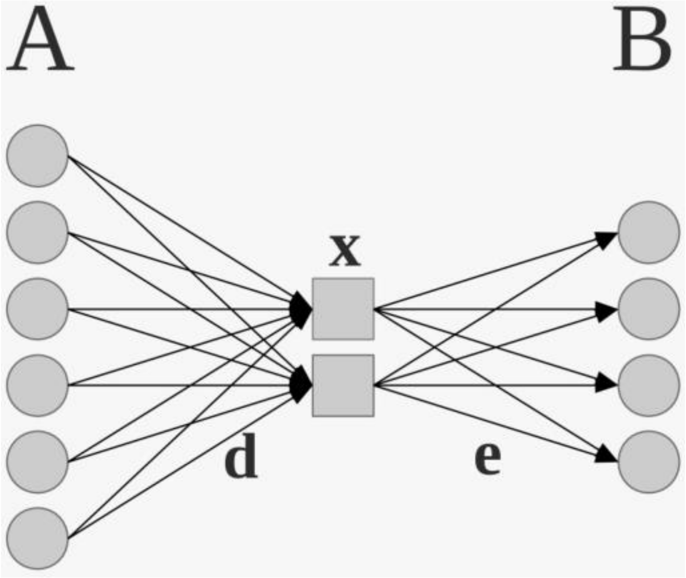
Naïve approach: Connect (i^{th} neuron of A) to (i^{th} neuron of B)

Problems:

- (#neuron in A) ≠ (#neuron in B)
- α_i and b_i might be different between A & B.
- Model G is most probably nonlinear.

Solution: Assume that we have an intermediate group of perfectly ideal linear neurons (one for each dimension).

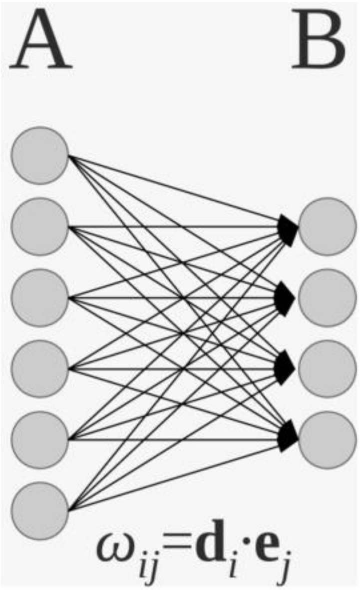
- From decoding equation, compute $\mathbf{x}$ from a_i using weights $\mathbf{d}$.
- From encoding equation, compute input current by combining $\mathbf{x}$ with $\mathbf{e}$.



continued...

But real brain doesn't have these idealized intermediate neurons.

- However, they are completely unnecessary.



We can adjust decoding equation to approximate any function $f(\mathbf{x})$.

$$\mathbf{d}^{f(\mathbf{x})} = \Gamma^{-1}\Upsilon^{f(\mathbf{x})} \qquad \Gamma_{ij} = \sum_{\mathbf{x}} a_i a_j \qquad \Upsilon_j^{f(\mathbf{x})} = \sum_{\mathbf{x}} a_j f(\mathbf{x})$$

<u>Takeaway:</u> Any nonlinear function can be approximated with a *single layer* of connection.

But the accuracy will be affected by:

- Nonlinearity and discontinuity of $f(\mathbf{x})$,
- Neuron properties and encoding scheme.

continued…

- NEF is using the same trick seen in SVM.
  - Randomly choose $\mathbf{e}_i$.
    - This is random projection.
  - Randomly choose α_i and b_i.
    - $f(\mathbf{x})$ ends up being a linear sum of tuning curves.

  Wider variety of tuning curves leads to better $f(\mathbf{x})$ approximation.
- NEF allows to add values by simply feeding inputs into the same group of neurons.
  - A → C with connection weights that compute $f(\mathbf{a})$.
  - B → C with connection weights that compute $g(\mathbf{b})$.

  C will end up with activity pattern that represents $f(\mathbf{a}) + g(\mathbf{b})$.

Dynamics

NEF provides a direct method for computing dynamic functions of the form:

$$\frac{d\mathbf{x}}{dt} = A(\mathbf{x}) + B(\mathbf{u})$$

Building this system requires you to know the neurotransmitter time constant (τ).

- τ reflects how quickly the neurotransmitter (released by a spike) is reabsorbed [2 ms ~ 200 ms].
- Once known, you can compute the desired $d\mathbf{x}/dt$ by creating a set of feedback connection weights.

continued…

A special case:

$$\frac{d\mathbf{x}}{dt} = \mathbf{u}$$

An integrator: $\mathbf{u} = 0$ → system holds current state

$\mathbf{u} > 0$ → value will increase

$\mathbf{u} < 0$ → value will decrease

In the feedback connection, neurons are passing information back to themselves.

Other complex models are also possible:

- Oscillators ($d\mathbf{x}/dt = [\mathbf{x}_2, -\mathbf{x}_1]$)
- Frequency-controlled oscillators ($d\mathbf{x}/dt = [\mathbf{x}_3\mathbf{x}_2, -\mathbf{x}_3\mathbf{x}_1]$)
- Kalman filters
- Chaotic attractors, etc.

Symbolic Processing

How can neurally realistic models possibly represent something like “Dogs chase cats” to distinguish it from “Cats chase dogs”?

➢Vector Symbolic Architecture (VSA)

- Use vectors for each basic symbol
- Combine these vectors with various mathematical operations
- Produce new vectors that encode full symbol structures

VSAs are lossy.

- As symbol tree structure gets more complex, the accuracy of extracting the original vectors decreases.

Example:

- Create unit vectors for each basic symbol (DOG, CAT, $CHASE$, $SUBJECT$, $OBJECT$, $VERB$, etc.).
- To create a symbol structure, you need two operations: addition ($+$) and circular convolution ($\otimes$).
- Sentence "Dogs chase cats" would then be

$$S = DOG \otimes SUBJECT + CHASE \otimes VERB + CAT \otimes OBJECT$$

- Extract a particular component by computing

$$S \otimes SUBJECT^{-1} \approx DOG$$

Note: Both "$+$" and "$\otimes$" can be easily approximated by NEF.

Spaun

Largest cognitive model (as of 2012)

- 2.5 million spiking neurons
- A vision system (Deep Belief Network with NEF)
- Single 6-muscle 3-joint arm for output

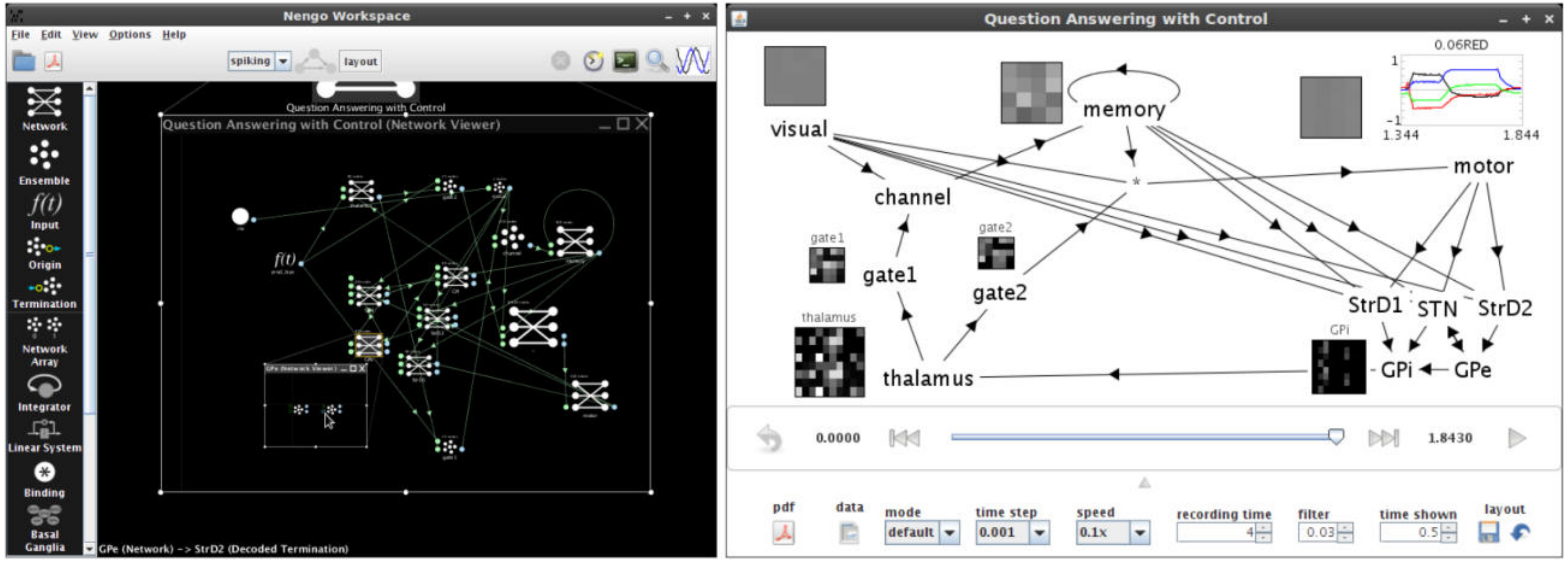
Can perform 8 different tasks, including:

- recognizing hand-written digits
- memorizing digit lists
- pattern completion
- mental addition

Nengo

Open-source cross-platform application which implements NEF

- Drag-and-drop interface / Python scripting
- $f(\mathbf{x})$ to approximate are similarly specified.
- Nengo automatically computes the optimized connection weights.



Limitations

- ➢Oversimplification of neurotransmitters
  - Neurotransmitters have many functions apart from just being a time constant.
- ➢Absence of nonlinear decoding
  - Complex neural models usually have multiple time constants and nonlinear synaptic effects.
- ➢Lack of developmental explanation
  - NEF only describes fully formed, adaptive, but non-developmental networks.

Questions?
Comments?
Concerns?