NEF and Nengo

Centre for Theoretical Neuroscience
Nengo Summer School

Chris & Terry

CTN Waterloo

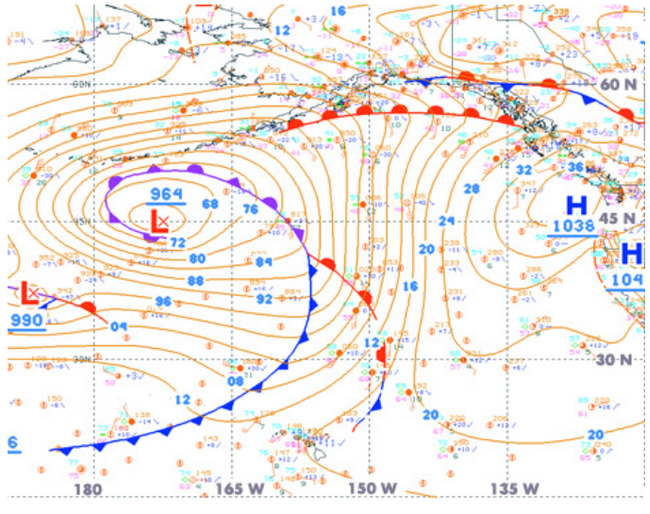
UNIVERSITY OF WATERLOO
CONCORDIA CUM VERITATE

Three views

- Symbolicism

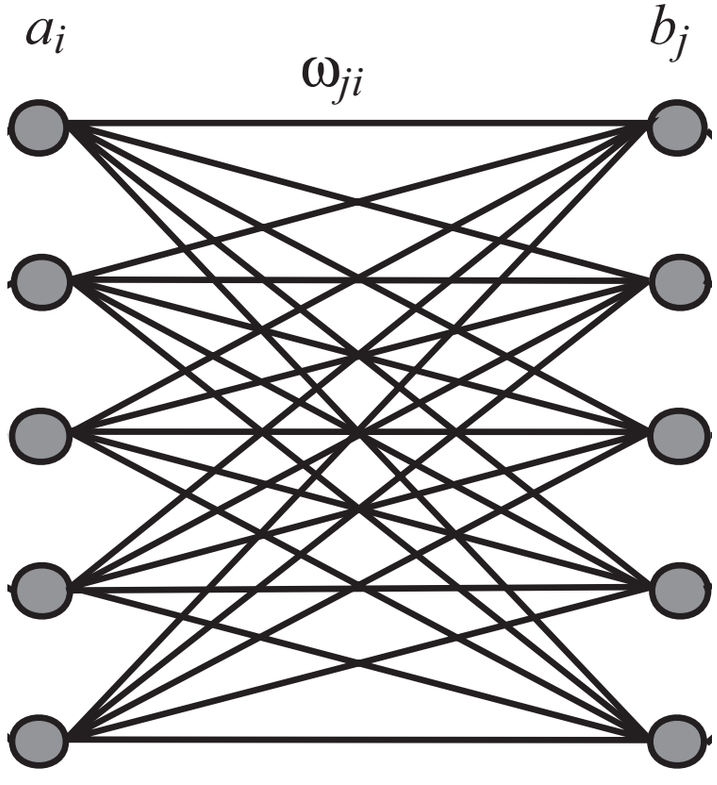
- Connectionism

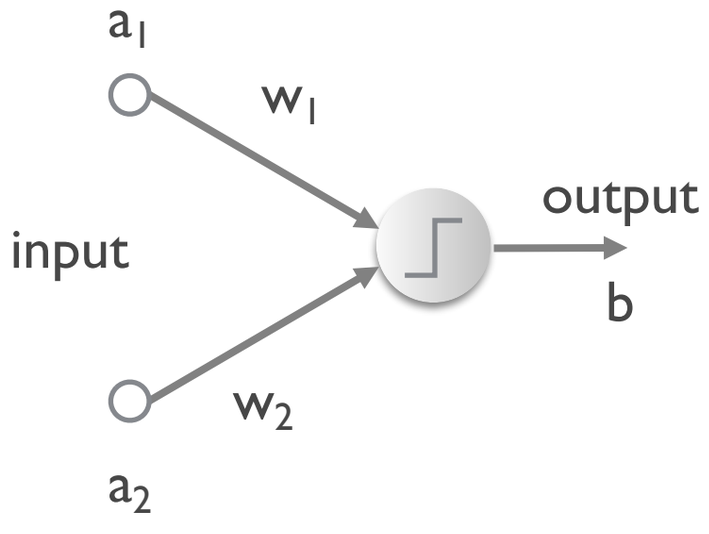
- Dynamicism



Connectionism

- ‘Brain-like’ networks





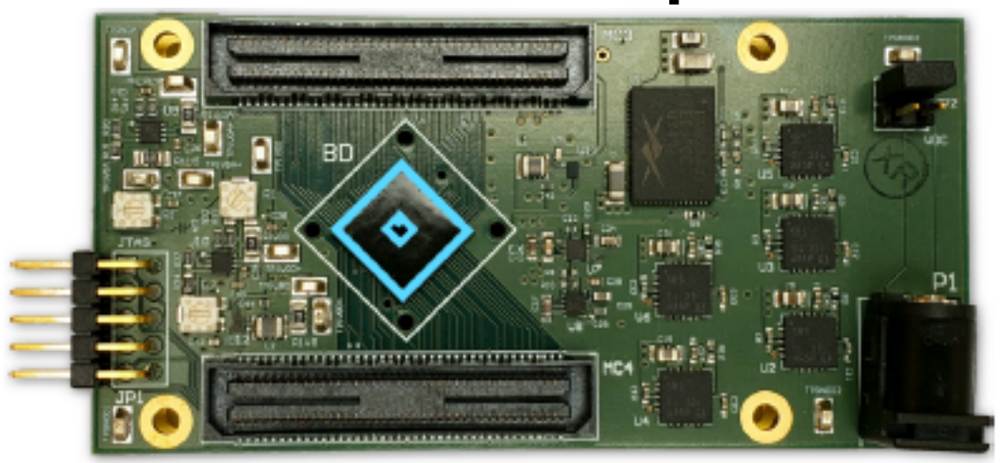
Deep Learning

- Backpropagation is an effective way to pick weights
- Challenges:
  - Relating to symbolicism (e.g., explainability, representation of concepts, high-level reasoning)
  - Incorporating time (e.g., long time scales, online learning, latency)

Neuromorphics

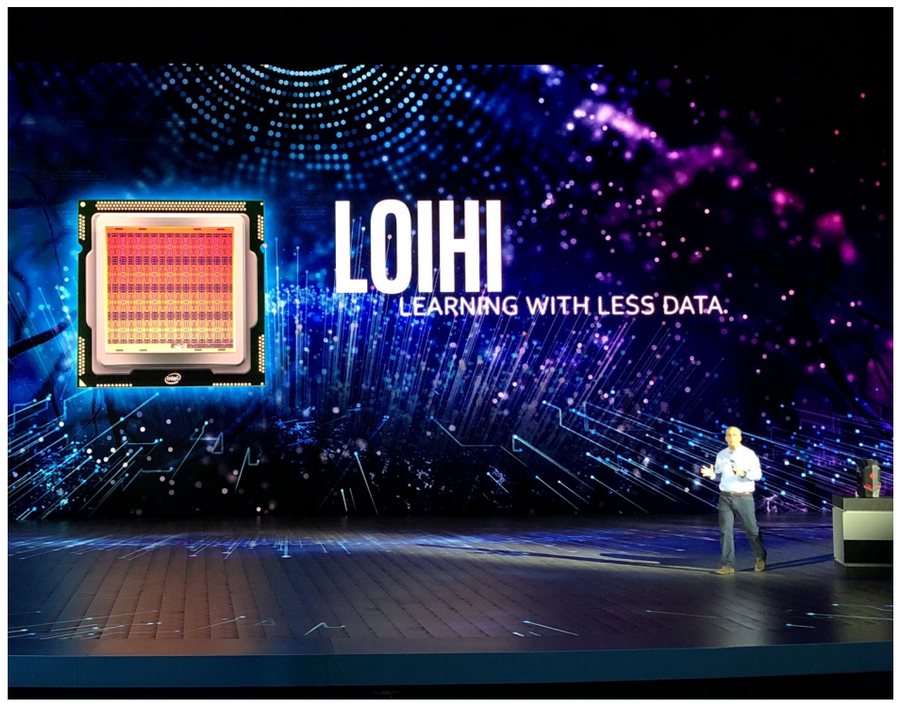
- Improve neural computing:
  - Lower power, real-time, online learning, robust

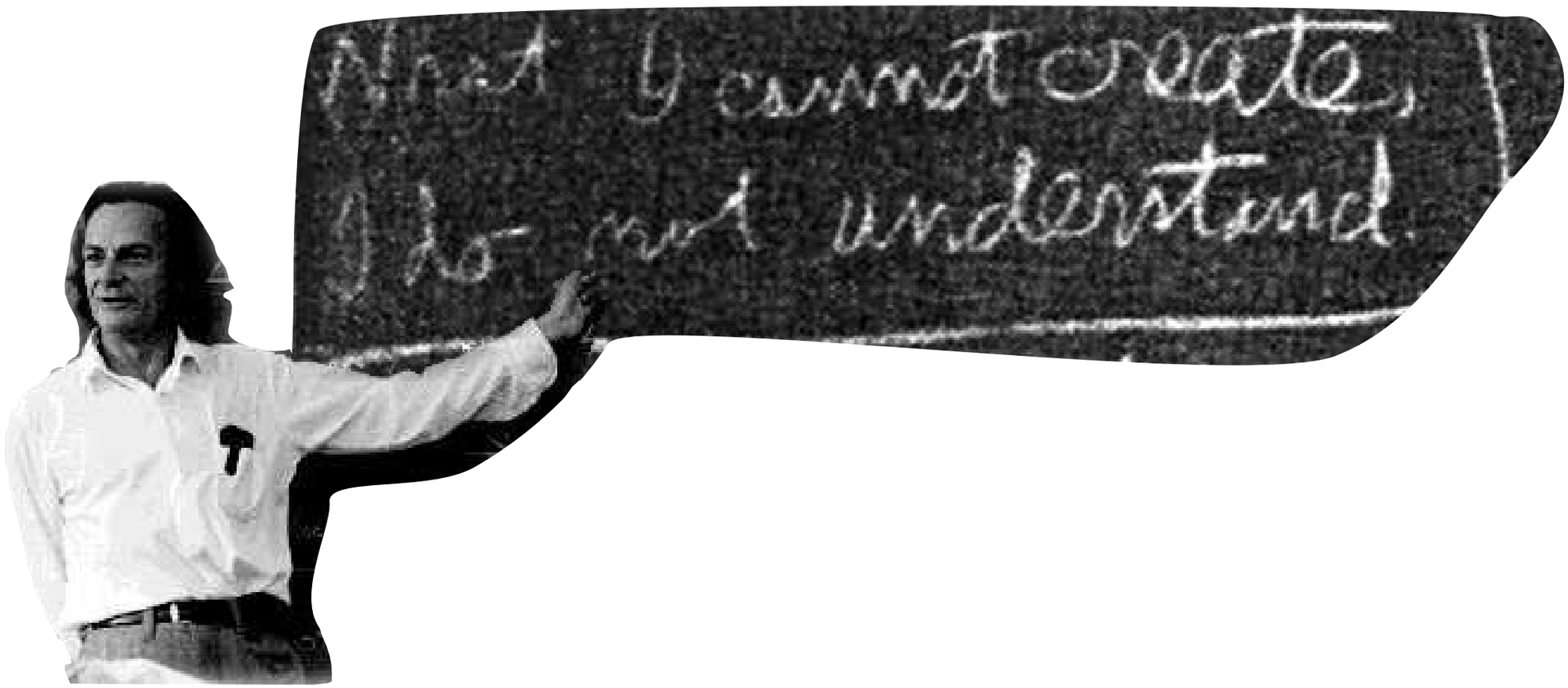
Braindrop



- Advance cognitive computing:
  - Large-scale neural modelling
  - Practical real-world, dynamic, and scalable AI

Loihi





large-scale

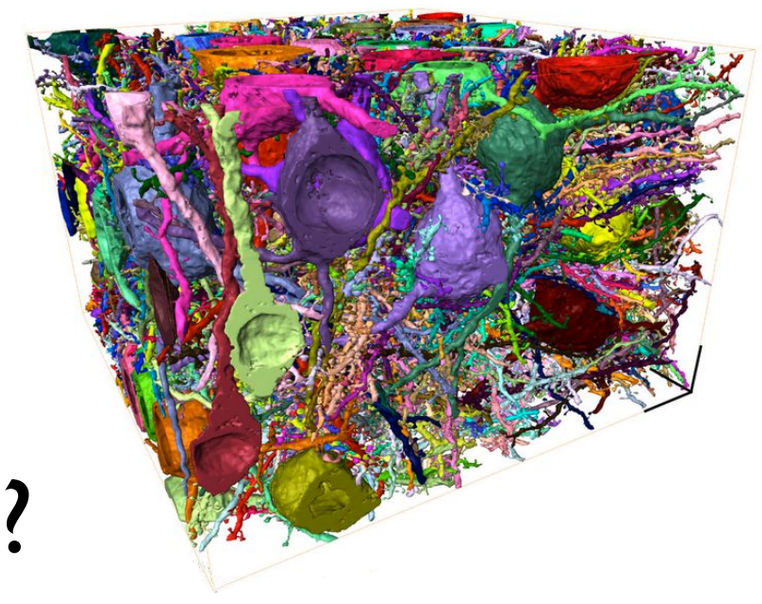
complete behaviour

A *more* synthetic approach to neuroscience (to *complement* the analytic approach)

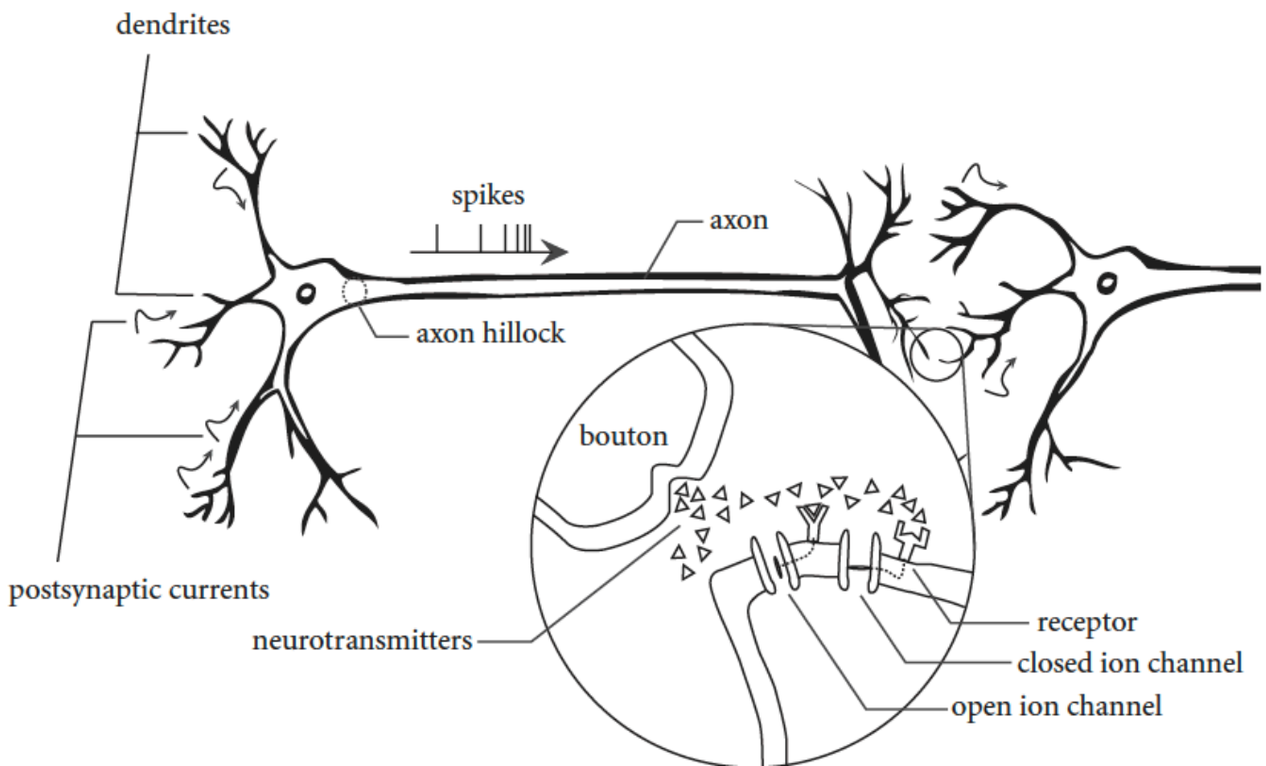
Challenges

- Include:
  - heterogeneity,
  - noise/reliability,
  - nonlinearities,
  - scaling

Neurons



- What are they good at computing?
  - Low precision, time-based, adaptive
  - Specific kinds of nonlinear functions
- How much complexity do we need?
  - As much as is useful for the questions we are asking



The NEF

Neural Engineering Framework (NEF)

- From high-level fcn to multi-layer spiking networks
- Three principles
  - Representation
  - Computation
  - Dynamics

Eliasmith & Anderson (2003) MIT Press

Three Principles of the NEF

1. *Representation*

   - Neural spike trains are nonlinear *encodings* of vector spaces that can be linearly *decoded*.

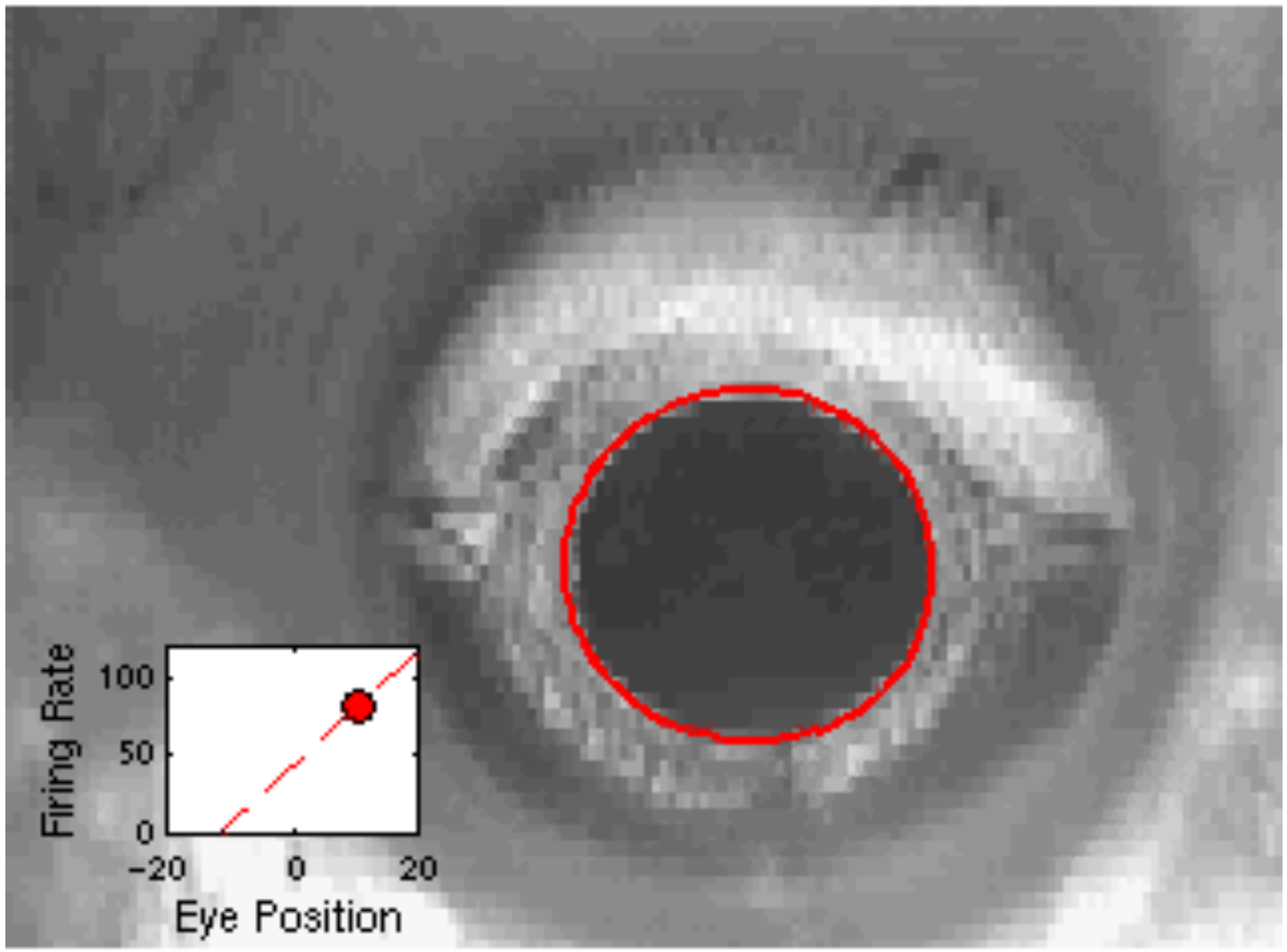
2. *Computation/Transformation*

   - Alternate linear decodings of those encodings can compute arbitrary vector functions.

3. *Dynamics*

   - Neural representations (1) are control theoretic state variables in a nonlinear dynamical system (2)

Principle 1: Representation



Principle 1: Representation

Bob

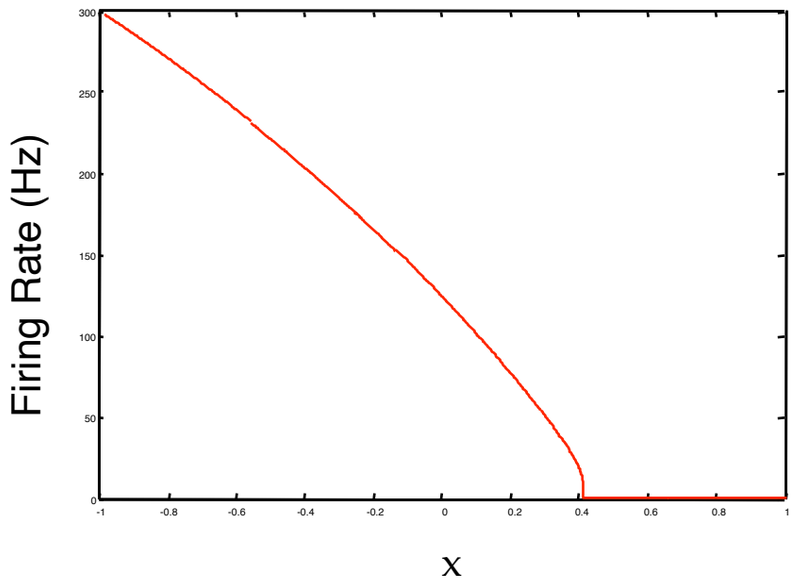
axon

Alice

axon

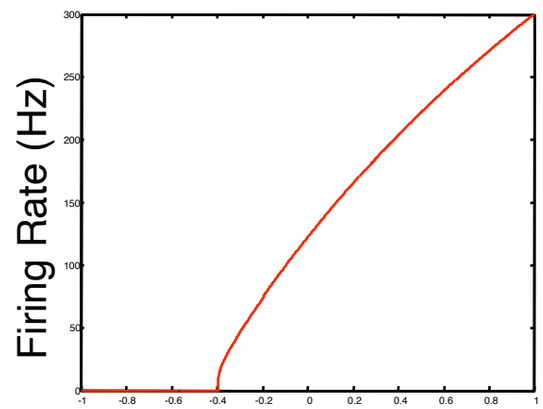
B

Neuron Tuning Curve



A

Neuron Tuning Curve



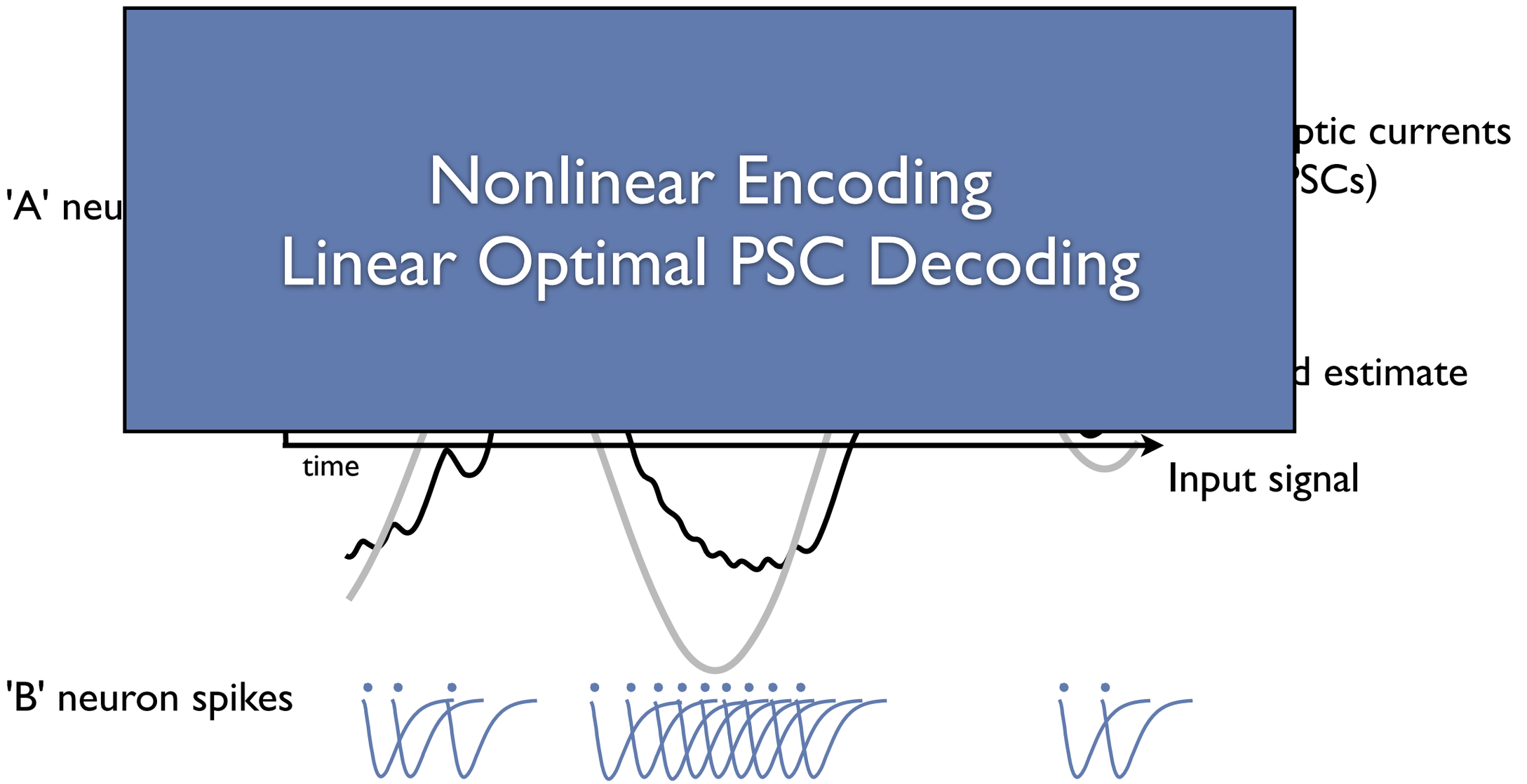
Charlie

axon

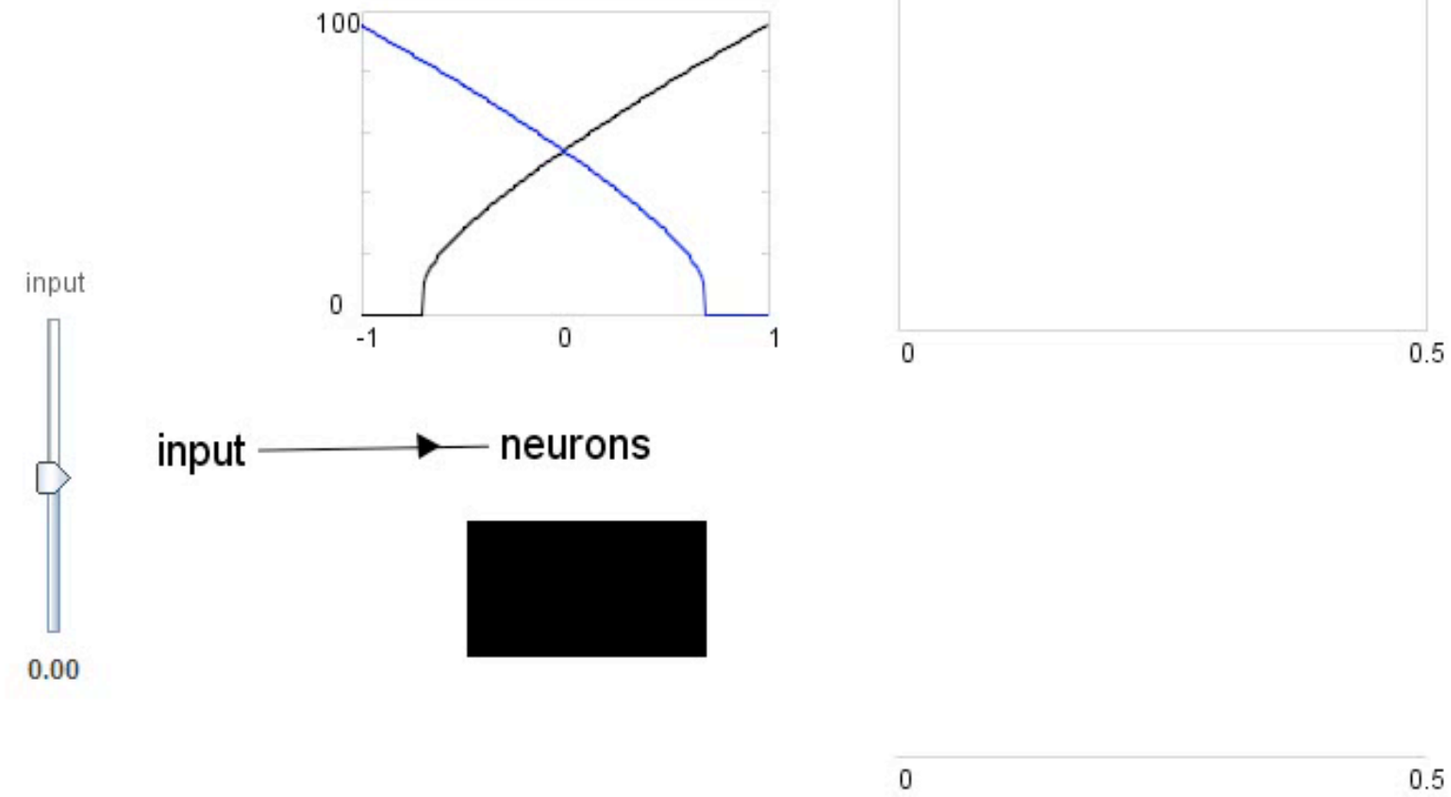
What could Charlie know about x?

Principle 1: Representation

- Encoding (stimulus -> spikes; A & B)
- Decoding (spikes -> stimulus; C)



Principle 1: Representation

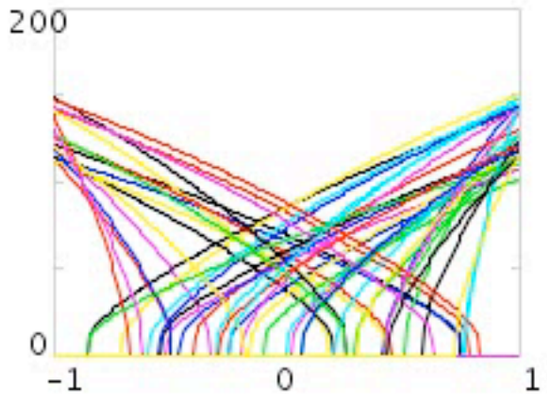


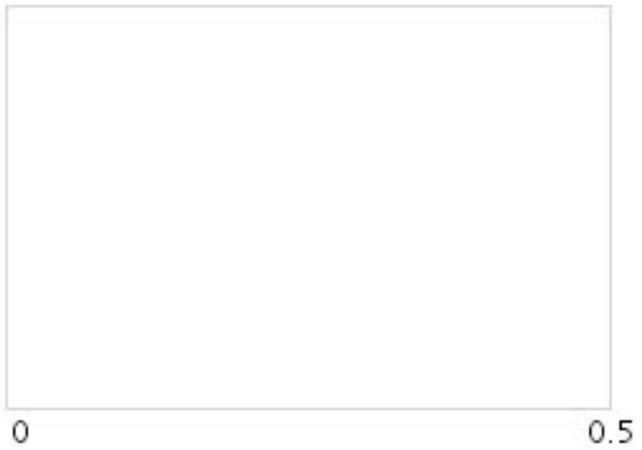
$$a_i = G_i\left[\alpha_i e_i x + J_i^{bias}\right]$$

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

= do sim

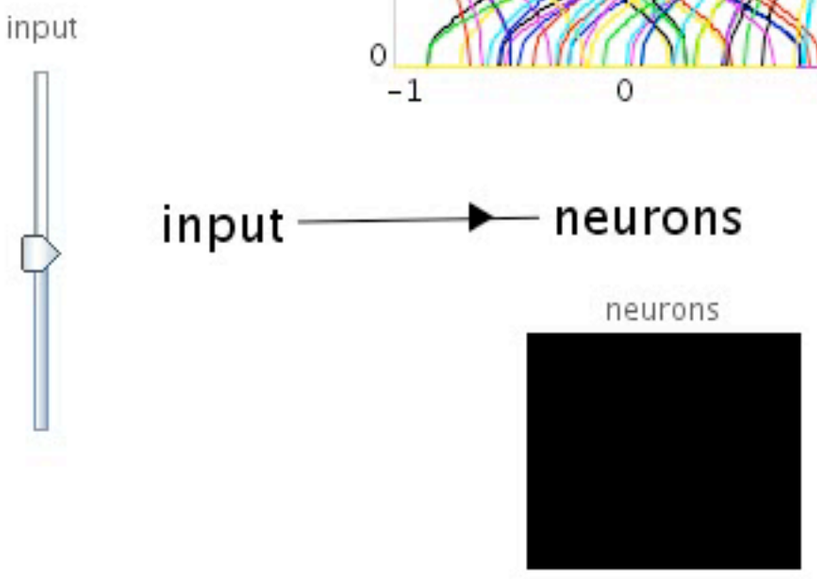
Principle 1: Representation





$$a_i = G_i\left[\alpha_i e_i x + J_i^{bias}\right]$$

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



Learn the decoders

- Use the PES (prescribed error sensitivity) rule to learn the representation
- Basically a delta rule with explicit error:

$$\Delta d_i = \kappa a_i E$$

learning rate

error

$$(\Delta \omega_{ij} = \kappa (e_j E) a_i)$$

Or, even easier

- Surprisingly simple optimization:

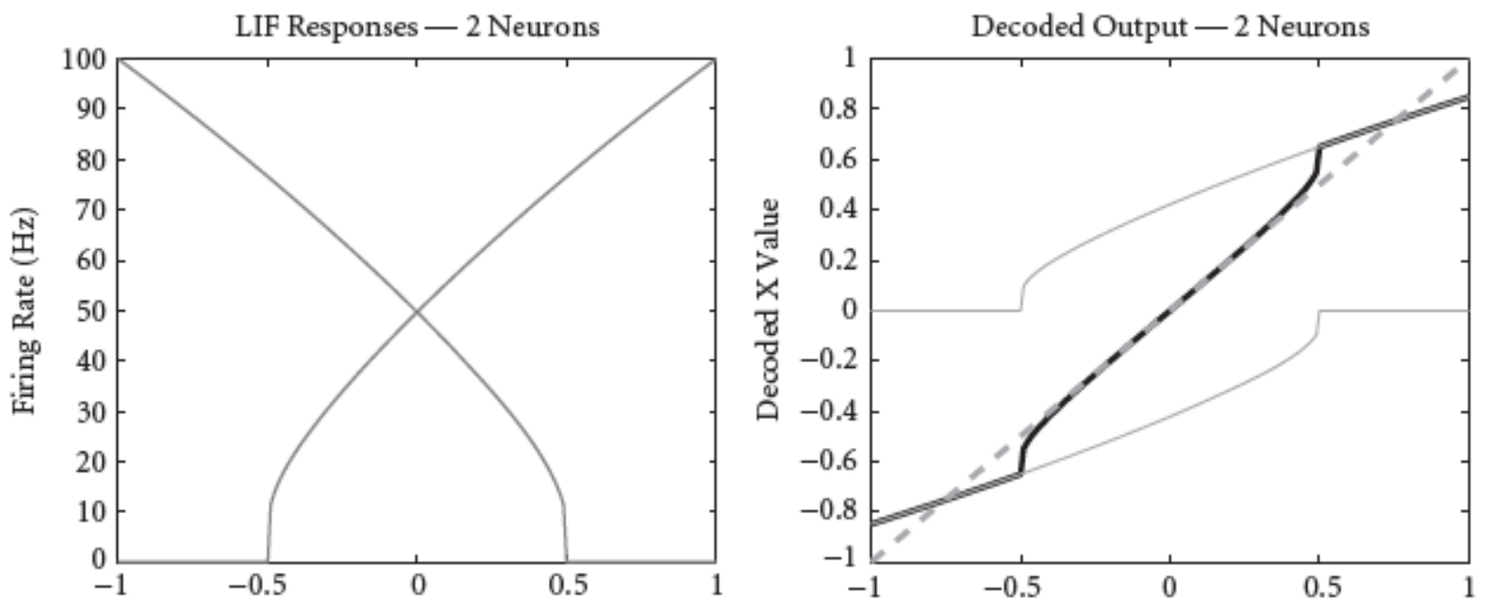
  Q: How to find population 'decoders', $\mathbf{d}_i$

A: Minimize $\left\langle (x - \hat{x})^2 \right\rangle_x$

Recall: $\hat{x} = \sum_i a_i \mathbf{d}_i$

$$d^x = \Gamma^{-1} \Upsilon$$

$$= \left[\int_x a_i(x) a_j(x) dx \right]^{-1} \int_x x a_i(x)$$



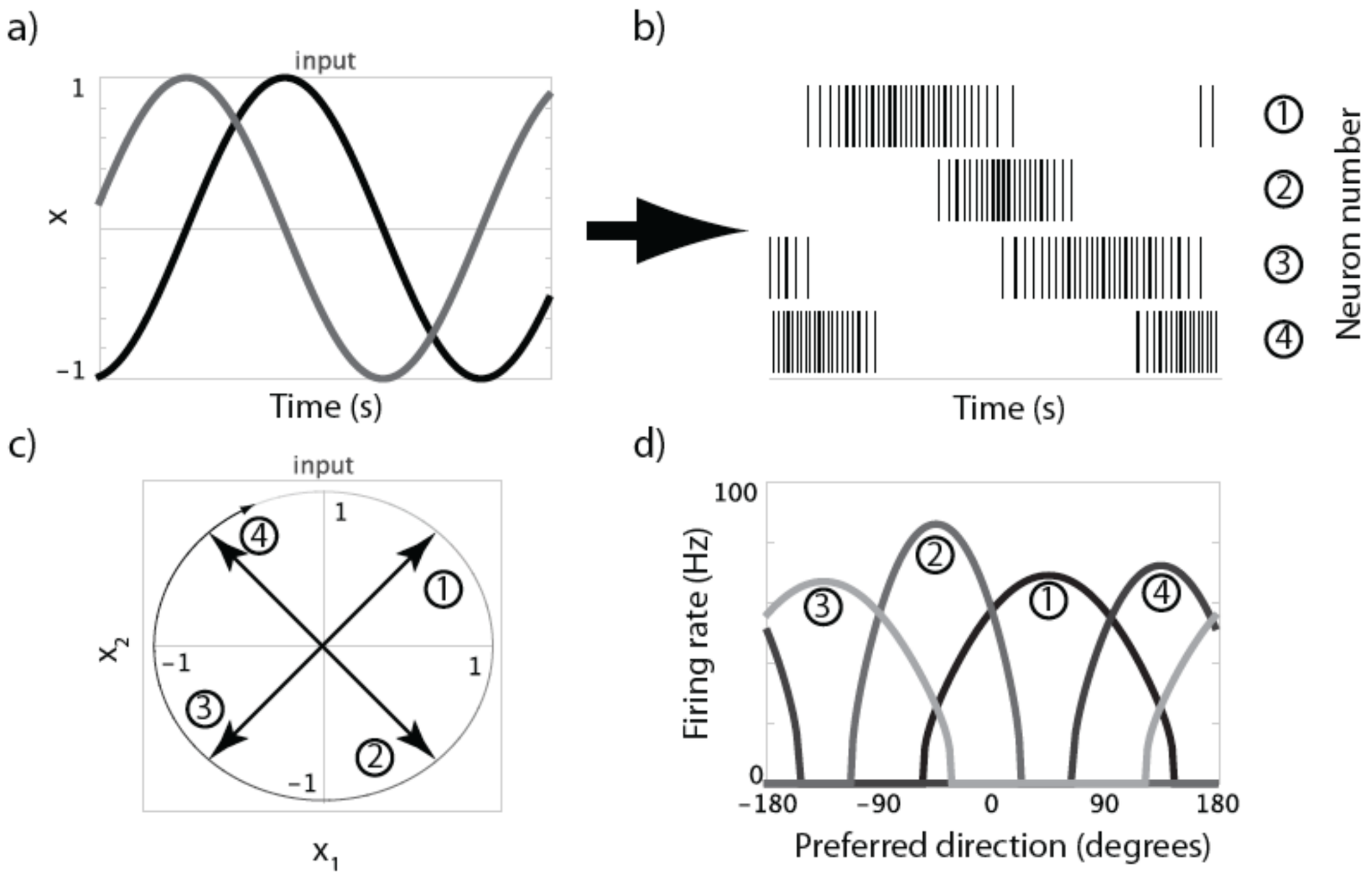
$\mathbf{d}_i$

Input X Value

Input X Value

2D Representation

- ... it's the *same thing*



Representation Hierarchy

Neural System	*Dim*	*Encoder* $(a_i(x))$	*Decoder* $(\hat{x})$

NEF and Nengo

Centre for Theoretical Neuroscience
Nengo Summer School

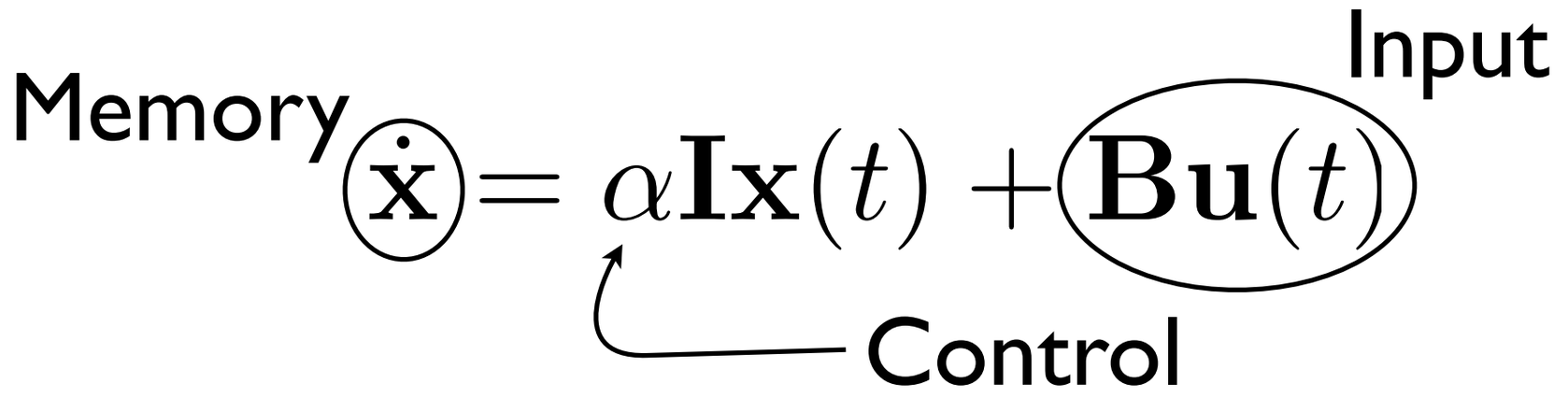
Chris & Terry

CTN Waterloo

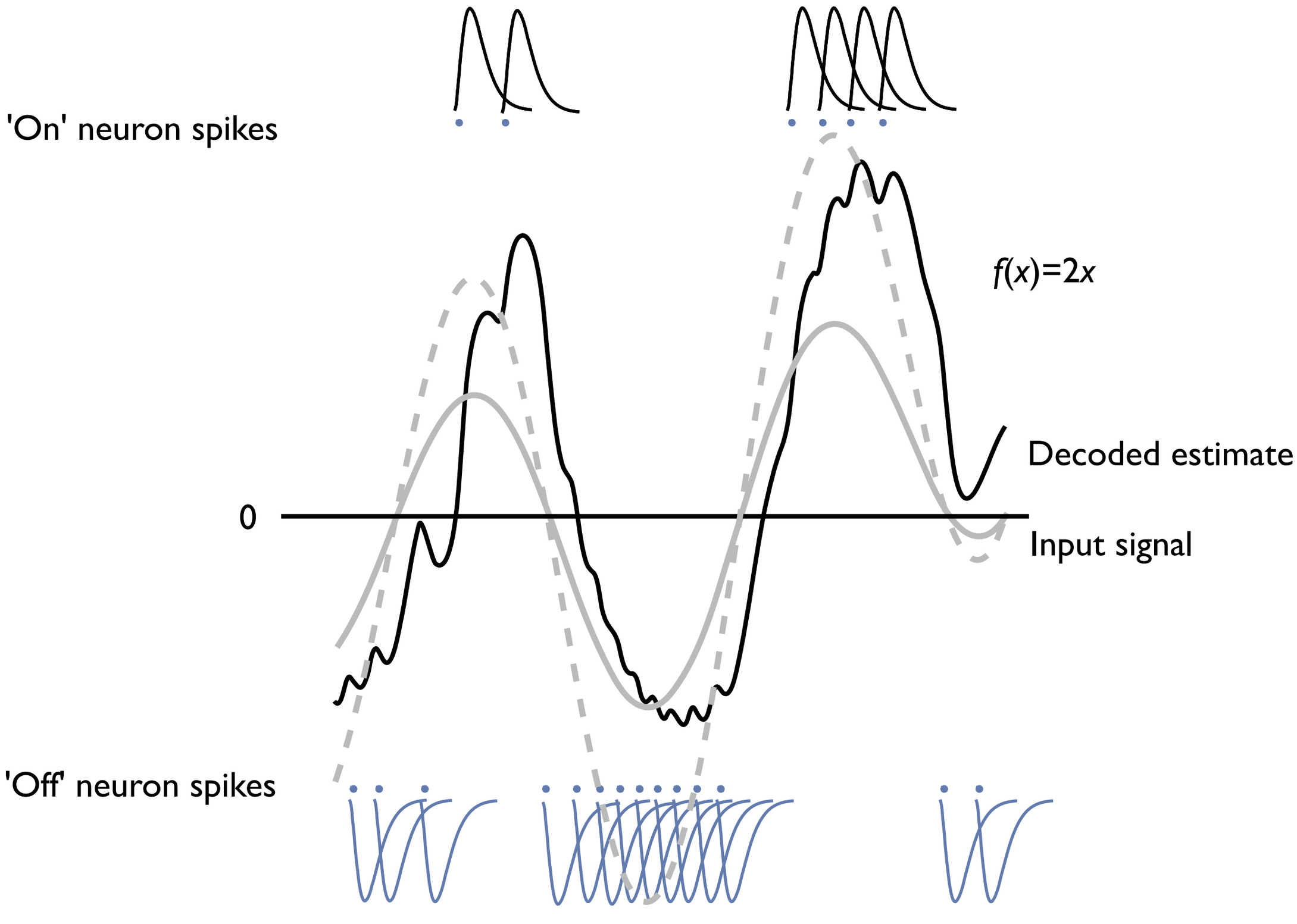
UNIVERSITY OF WATERLOO
CONCORDIA CUM VERITATE

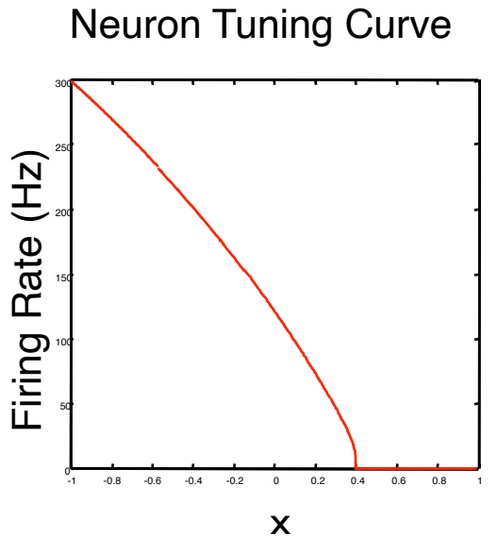
Example: Working memory

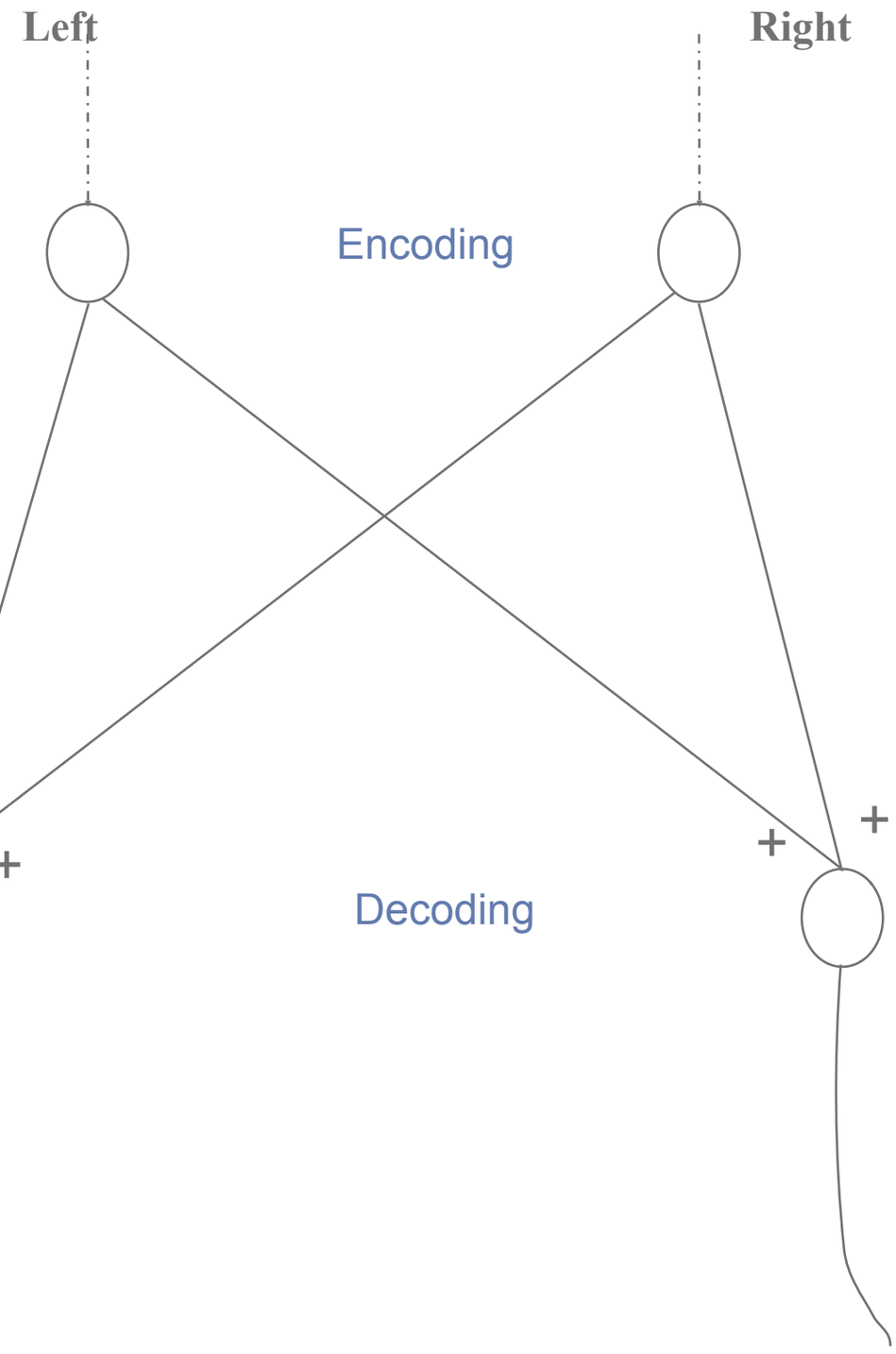
- We want a neural circuit that:
  - stores an input state
  - can be cleared (controlled storage)
- Dynamical equation (i.e., high-level program):

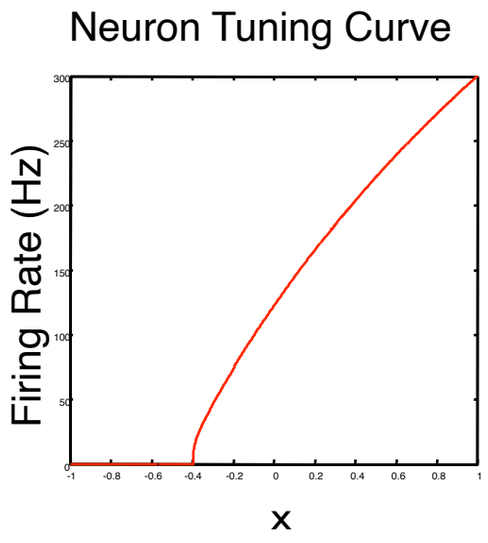


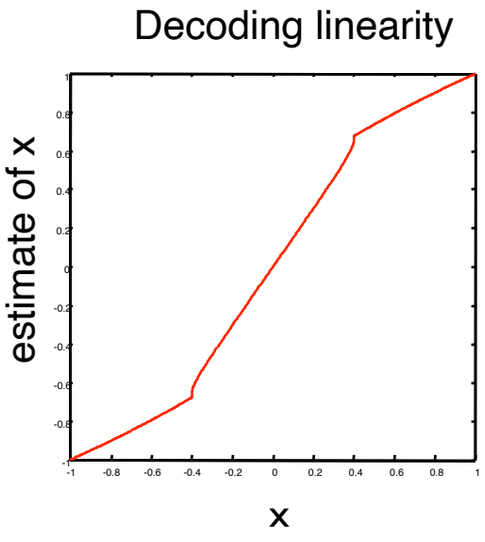
Principle 2: Transformation



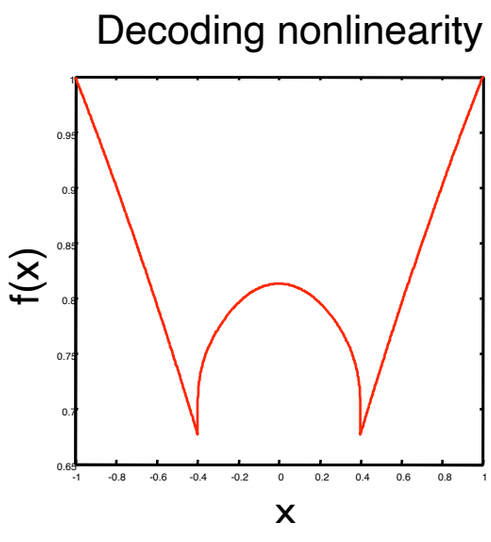




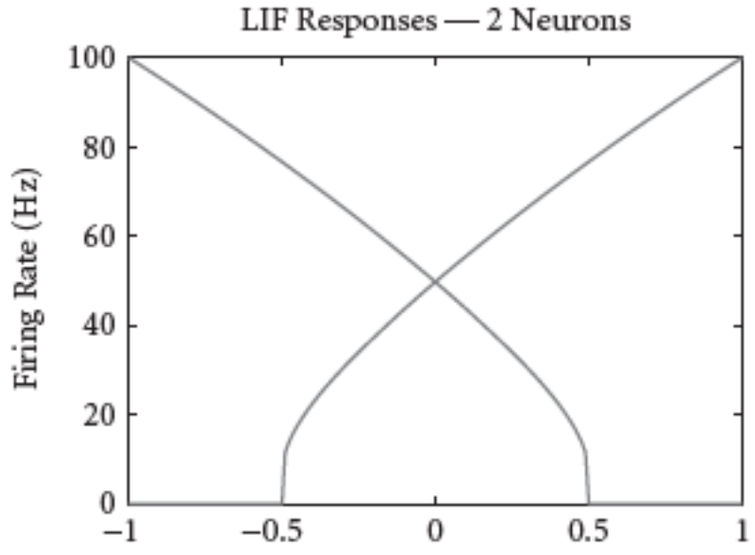


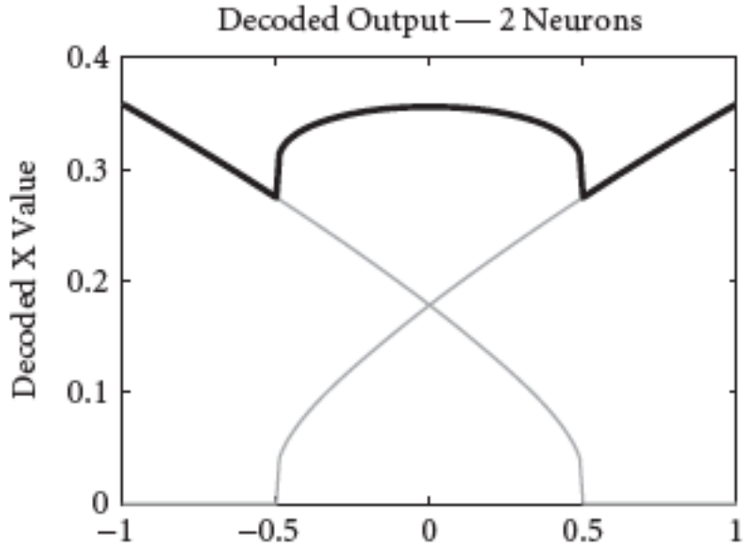


Representation



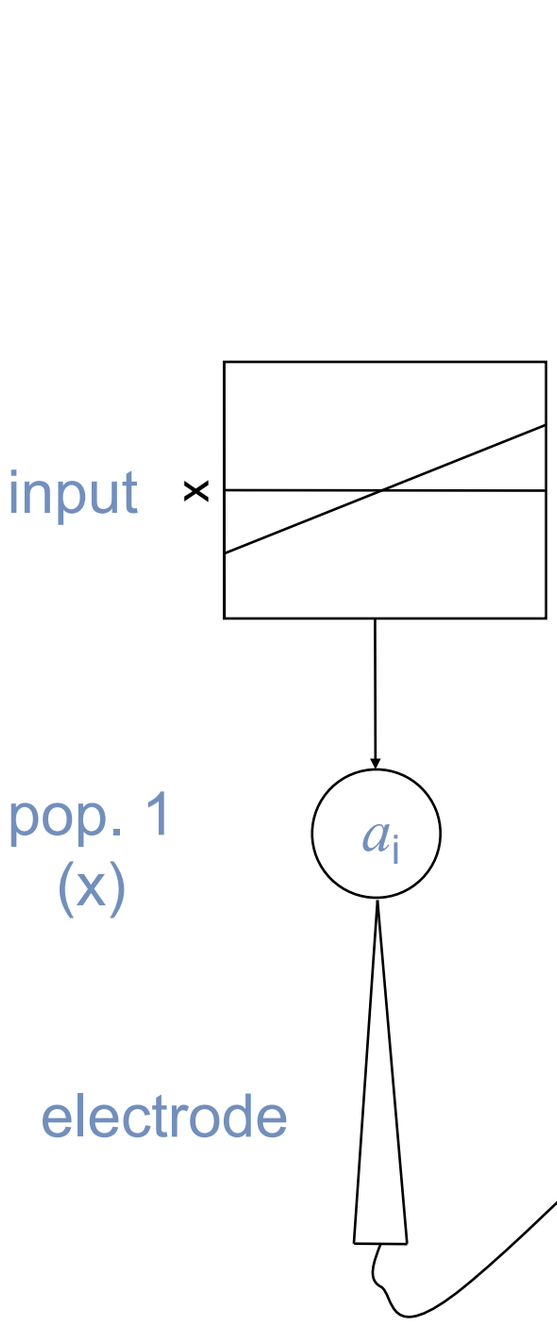
Transformation

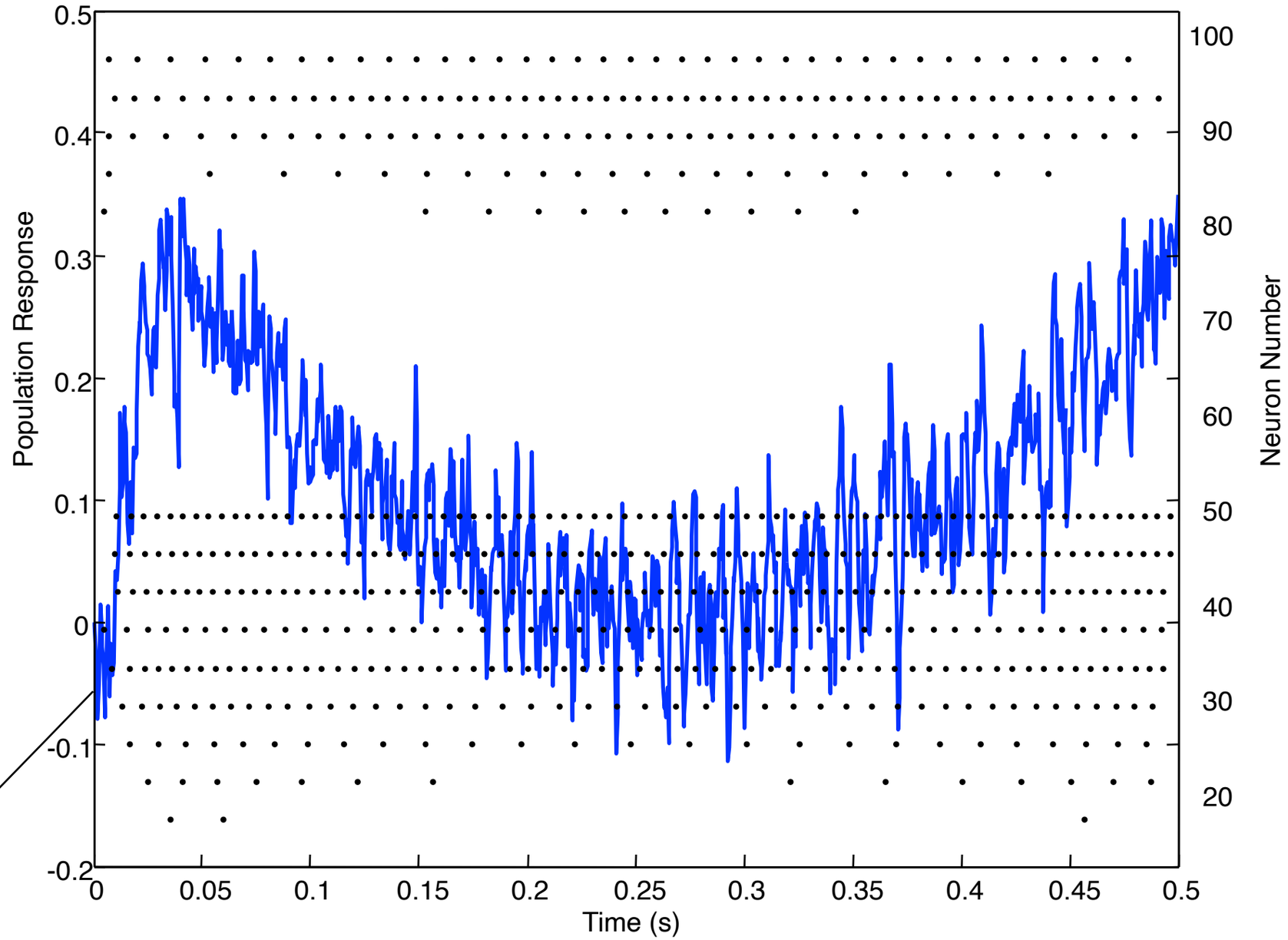

LIF Responses — 2 Neurons
Firing Rate (Hz)
100
80
60
40
20
0
−1
−0.5
0
0.5
1
Input X Value


Decoded Output — 2 Neurons
Decoded X Value
0.4
0.3
0.2
0.1
0
−1
−0.5
0
0.5
1
Input X Value

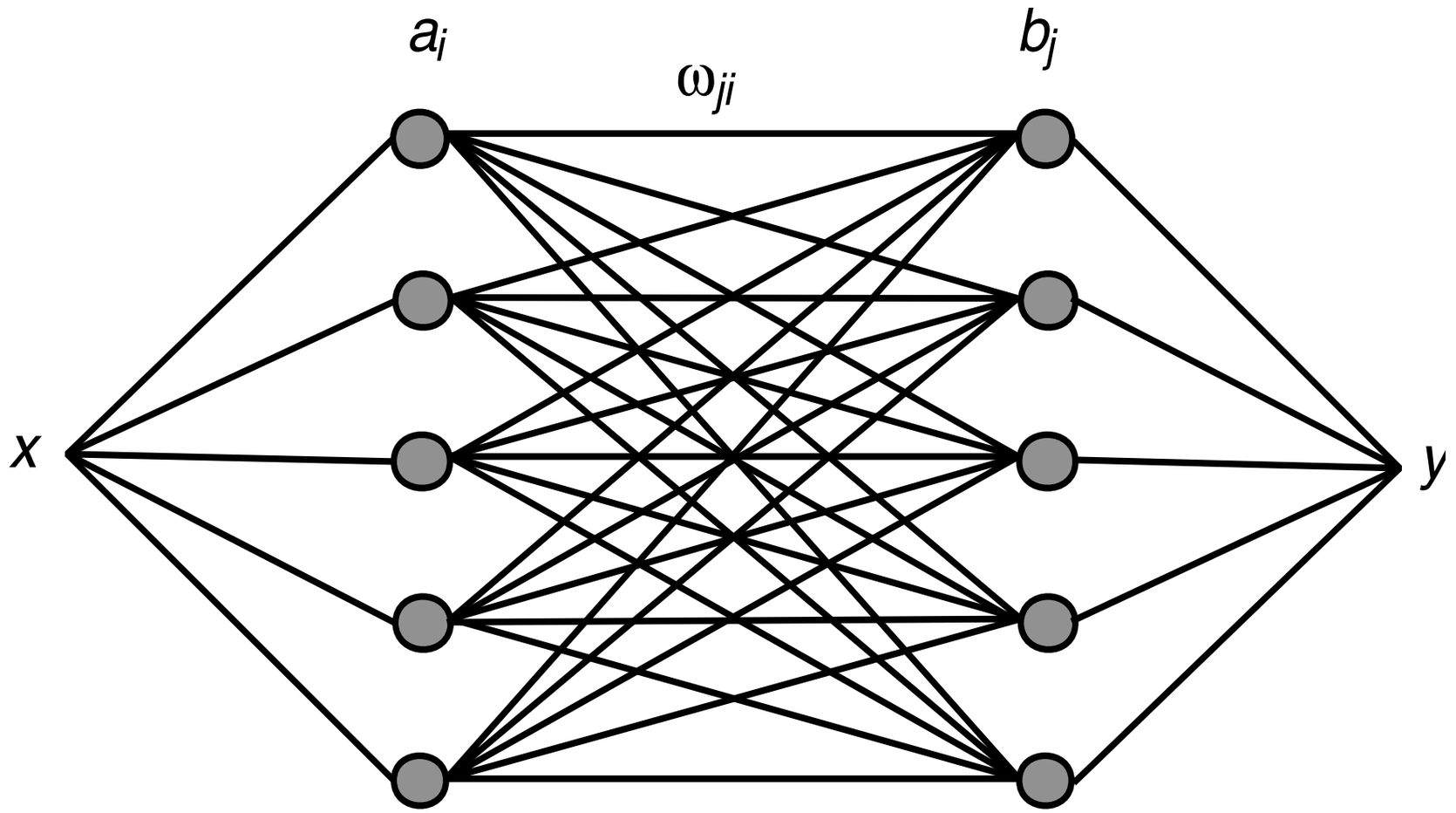
$\mathbf{d}f_i$

Principle 2: Computation





Building Circuits - A communication channel



- Define the representations of both pops:

Encoding

$$\begin{aligned} a_i &= G_i\left[J_i(x)\right] & b_j &= G_j\left[J_j(y)\right] \\ &= G_i\left[\alpha_i \mathbf{e}_i x + J_i^{bias}\right] & &= G_j\left[\alpha_j \mathbf{e}_j y + J_j^{bias}\right] \end{aligned}$$

$$\hat{x} = \sum_i a_i \mathbf{d}_i \qquad \hat{y} = \sum_j b_j \mathbf{d}_j$$

Decoding

- Define the computation: *y=x*
- Substitute our estimate of *x* into *b*

- Substituting:

$$y = x \approx \hat{x}$$

$$\begin{aligned} b_j &= G_j \left[\alpha_j \mathbf{e}_j x + J_j^{bias} \right] \\ &= G_j \left[\alpha_j \mathbf{e}_j \sum_i a_i \mathbf{d}_i + J_j^{bias} \right] \\ &= G_j \left[\sum_i \omega_{ji} a_i + J_j^{bias} \right] \end{aligned}$$

$$\omega_{ji} = \alpha_j \mathbf{e}_j \mathbf{d}_i$$

NEF connection weights

x

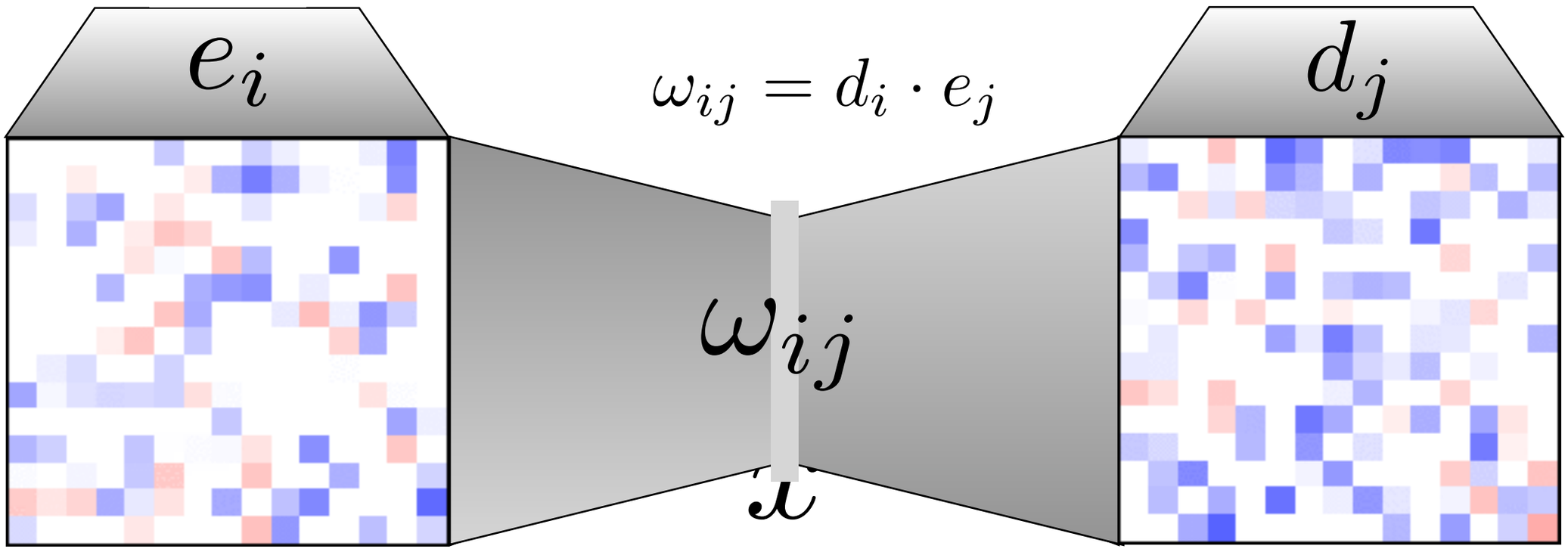
0.65

e_i

$\omega_{ij} = d_i \cdot e_j$

0.65

d_j



ω_{ij}

x

a_i

$\hat{x} = \sum_i a_i d_i^x \leftarrow f(x)$

b_j

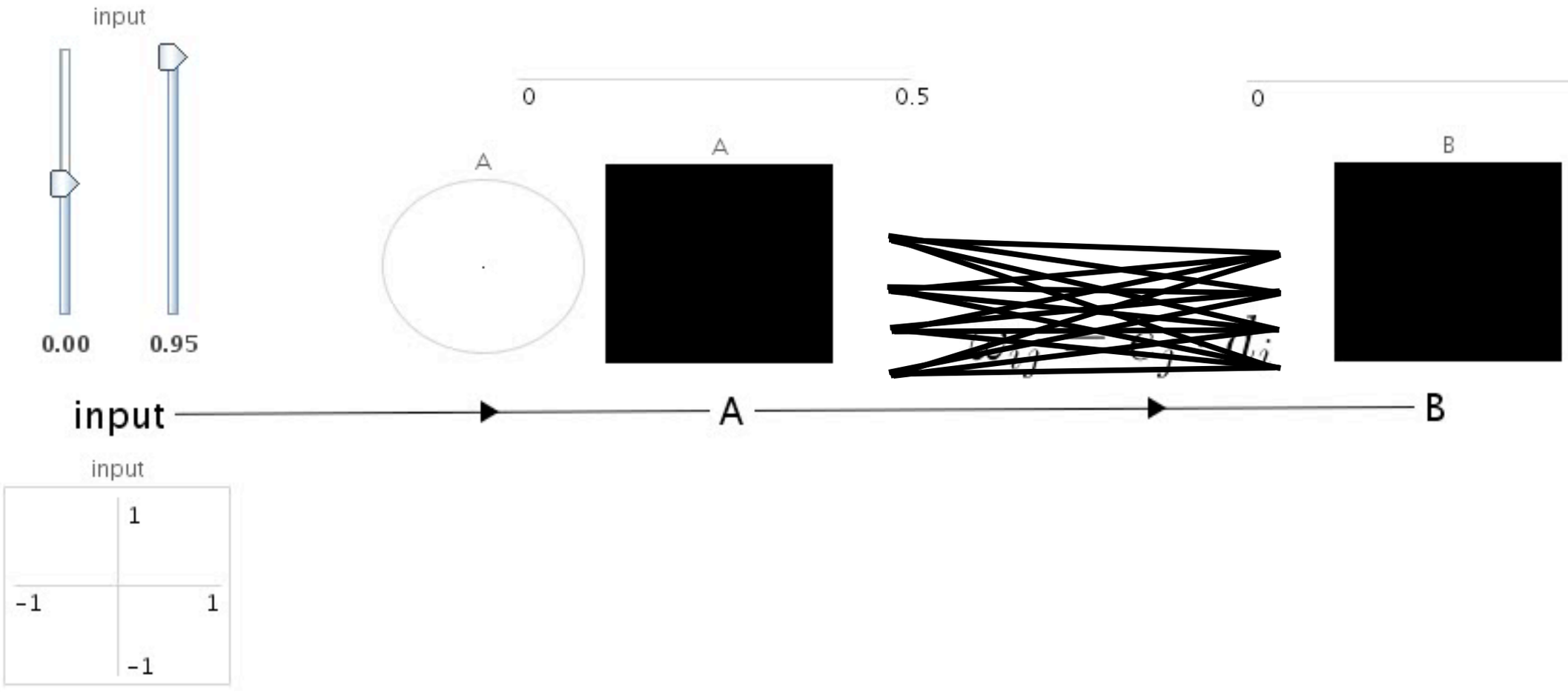
$J_i = x \cdot e_i$

$a_i = G_i\left[J_i(x)\right]$

$J_j = \sum_i \omega_{ij} a_i$

$J_j = \hat{x} \cdot e_j$

$b_j = G_j[J_j]$


input
0.00
0.95
A
0
0.5
A
A
B
B
input
A
B

input
1
-1
1
-1

Pick a function

- Any function... build a feedforward approximation

Now learn it

$$\Delta d_i = \kappa a_i E$$

learning rate

error

$$(\Delta \omega_{ij} = \kappa (e_j E) a_i)$$

Summary

- Built a feedforward computation in spiking neural networks
- Principles used are very general (vector space representation, nonlinear computation)
- Dealt with heterogeneity, nonlinearities, noise
- ... dynamics & scaling... next

Previous Lecture

Next Lecture

Playlist

Centre for Theoretical Neuroscience
Nengo Summer School

Chris & Terry