How to build a brain

Nonlinear Transformations

Terry & Chris
Centre for Theoretical Neuroscience
University of Waterloo

University of Waterloo

CTN Waterloo

CNRG computational neuroscience research group
compneuro.uwaterloo.ca

Nonlinearities via decoding

- Find an optimal linear decoder to approximate some function of the space **m**:

$$\hat{f}(\mathbf{m}) = \sum_l d_l(\mathbf{m})\phi_l^f$$

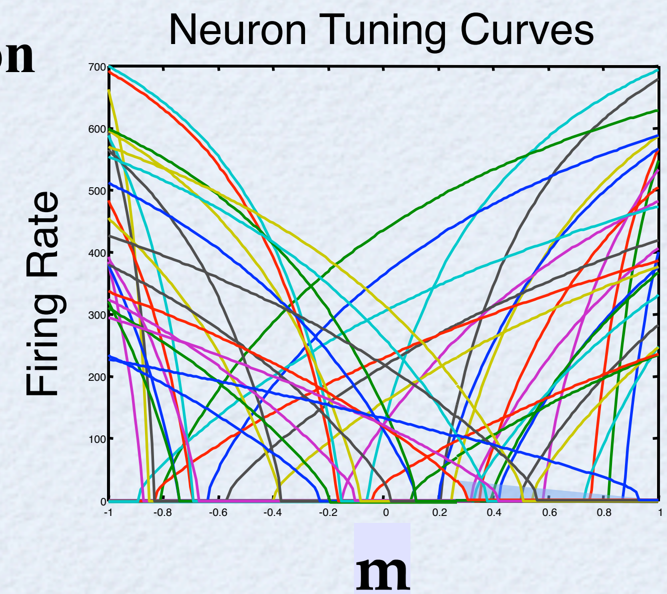
- By minimizing

$$E_f = \left\langle \left[f(\mathbf{m}) - \hat{f}(\mathbf{m}) \right]^2 \right\rangle_{\mathbf{m}}$$

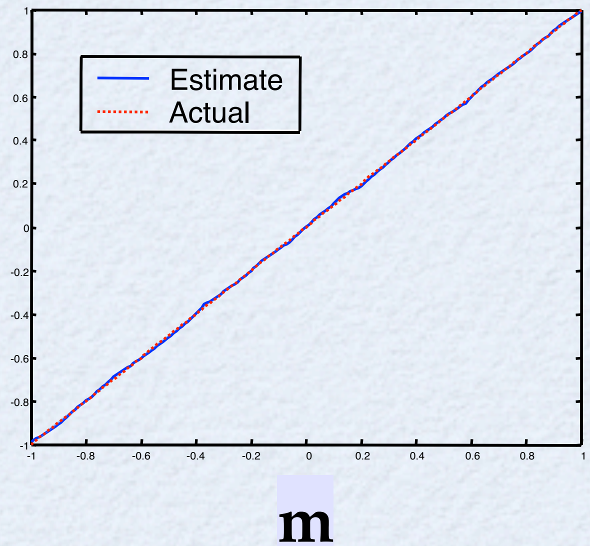
Function decoders

Representation

Encoding

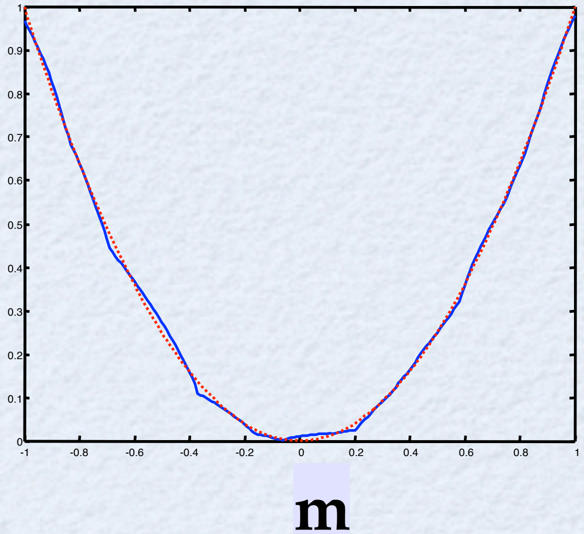


Decoding
(Linearity)

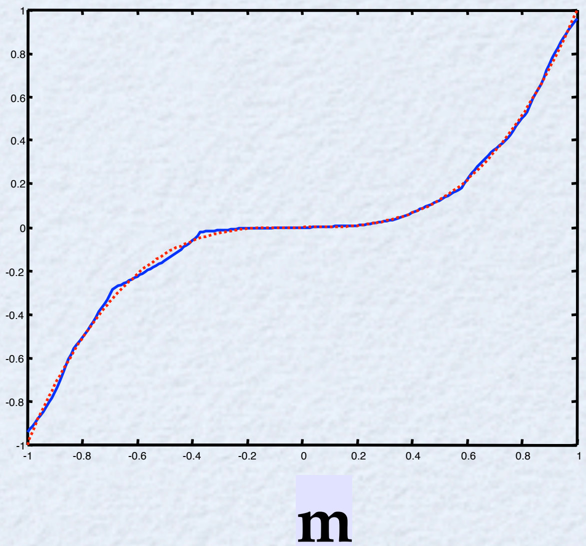


Computation

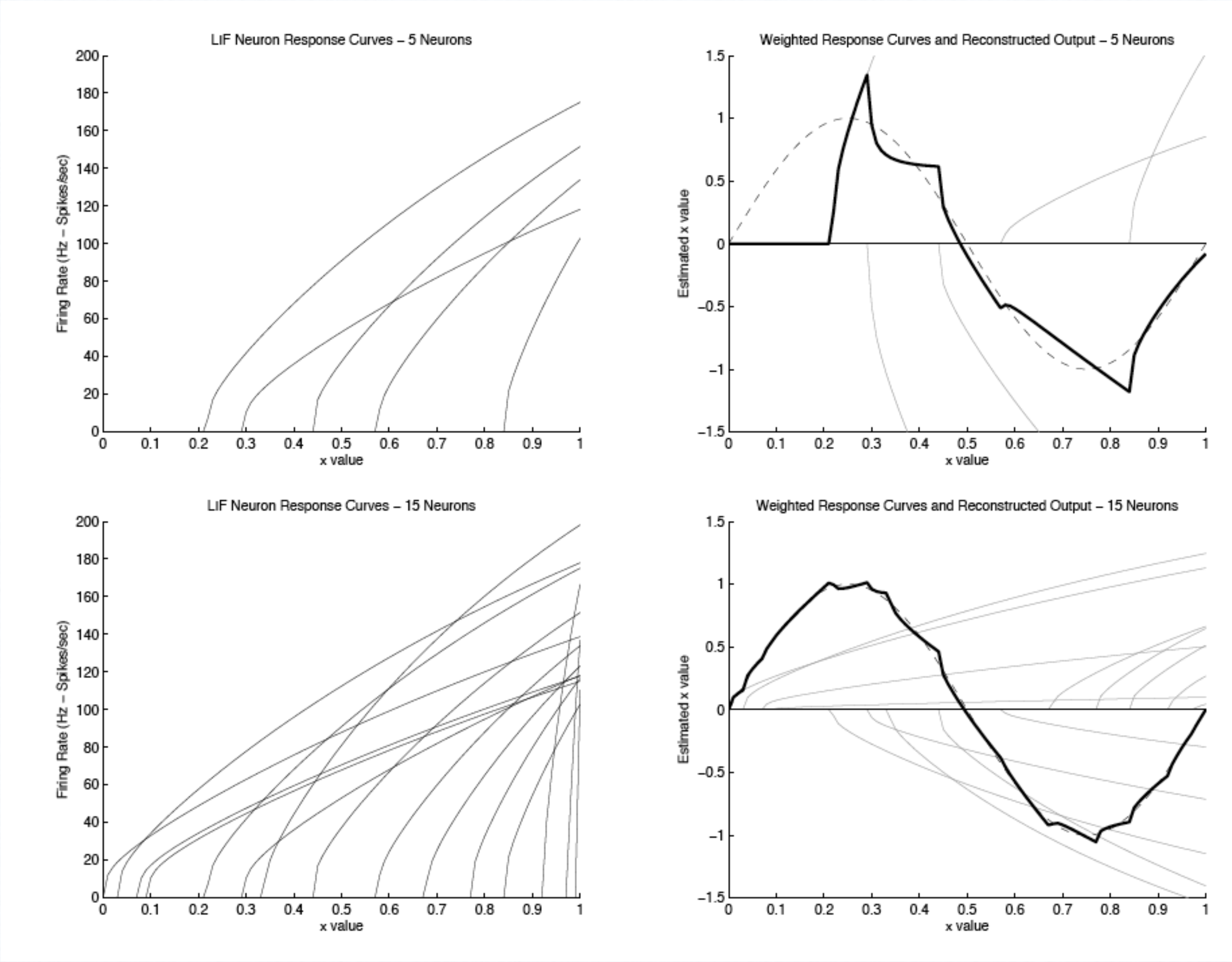
Quadratic



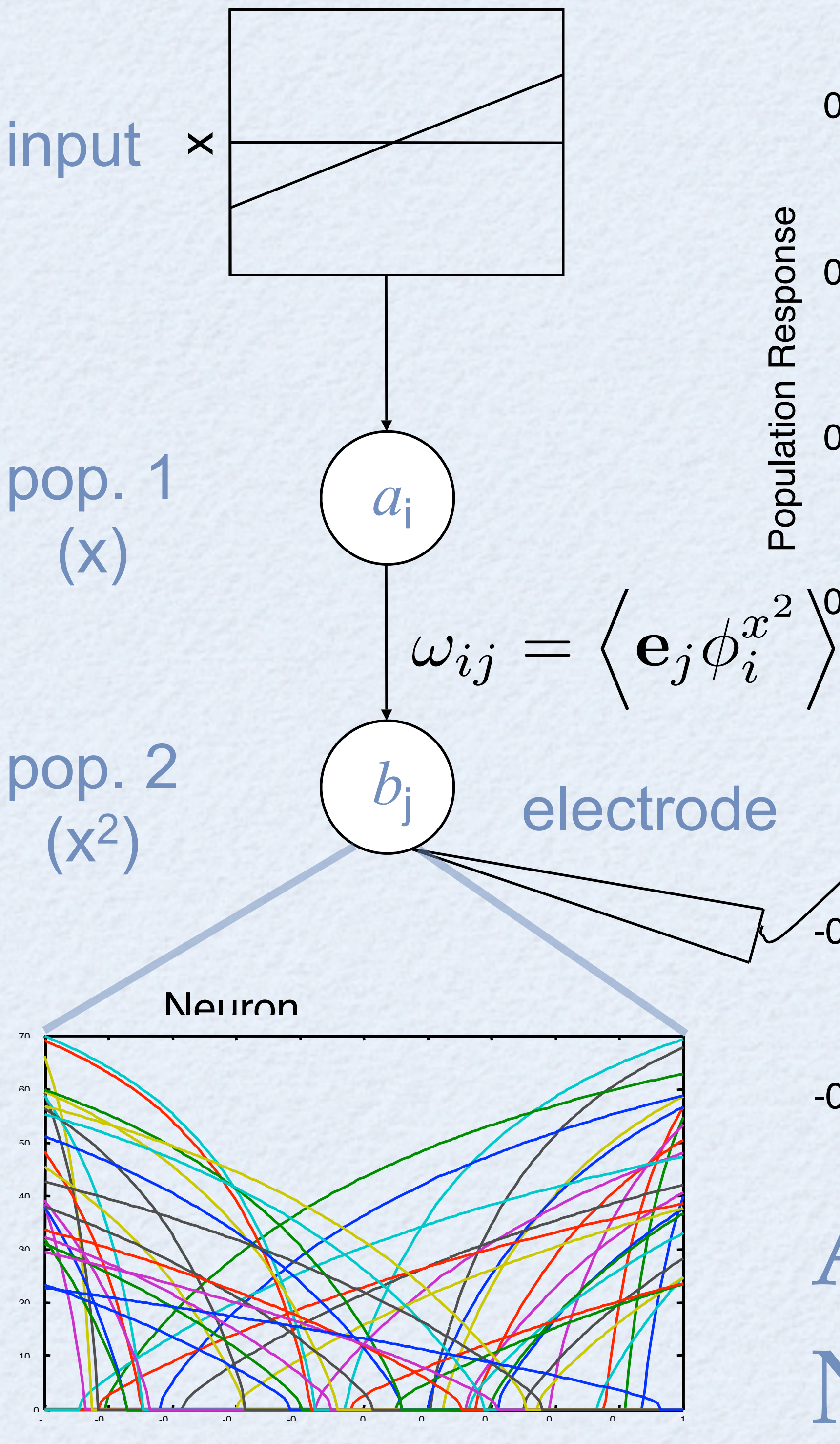
Cubic

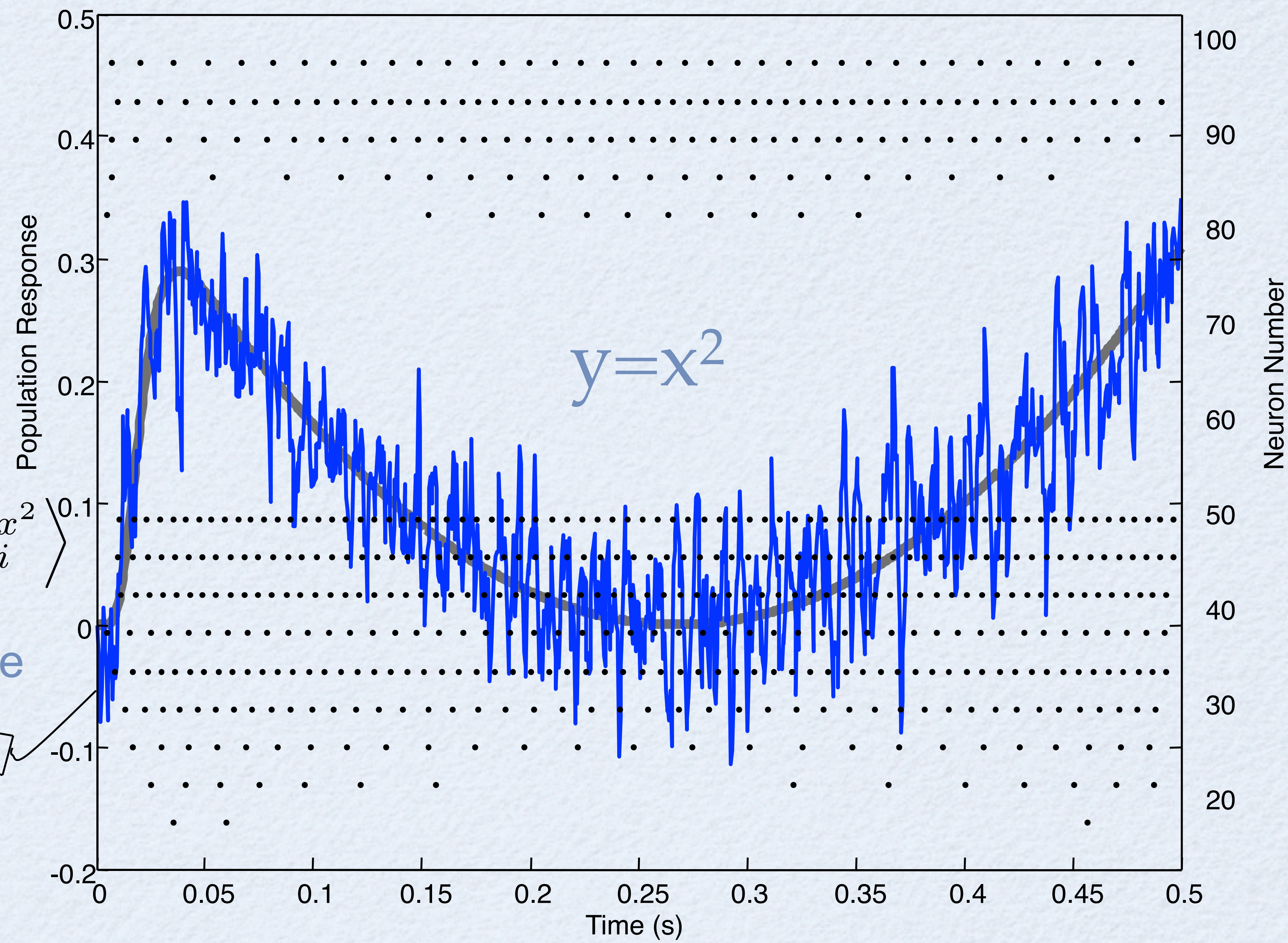


Function approximation



Nonlinearities over time*



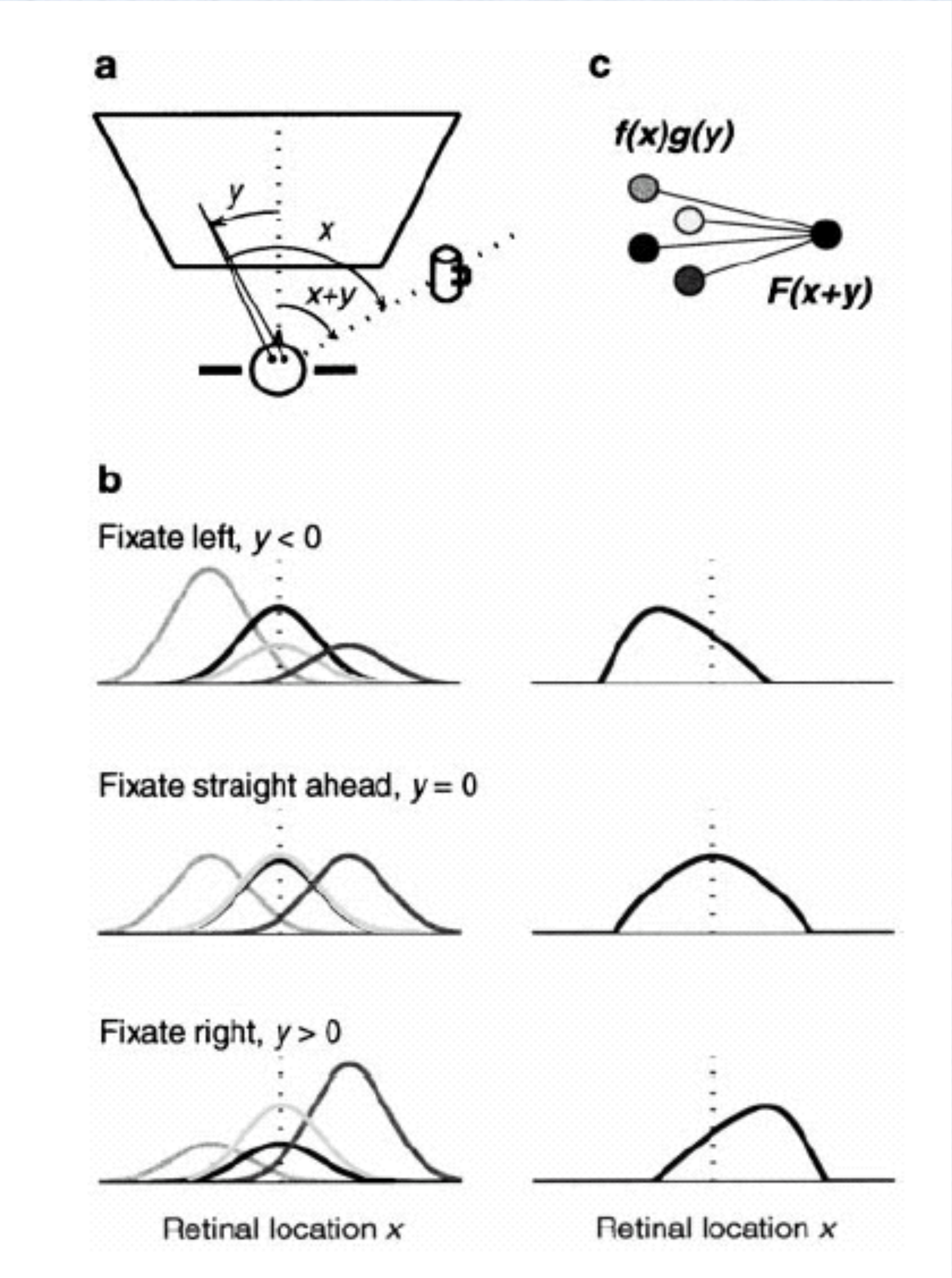


Any feedforward nonlinearity

No learning

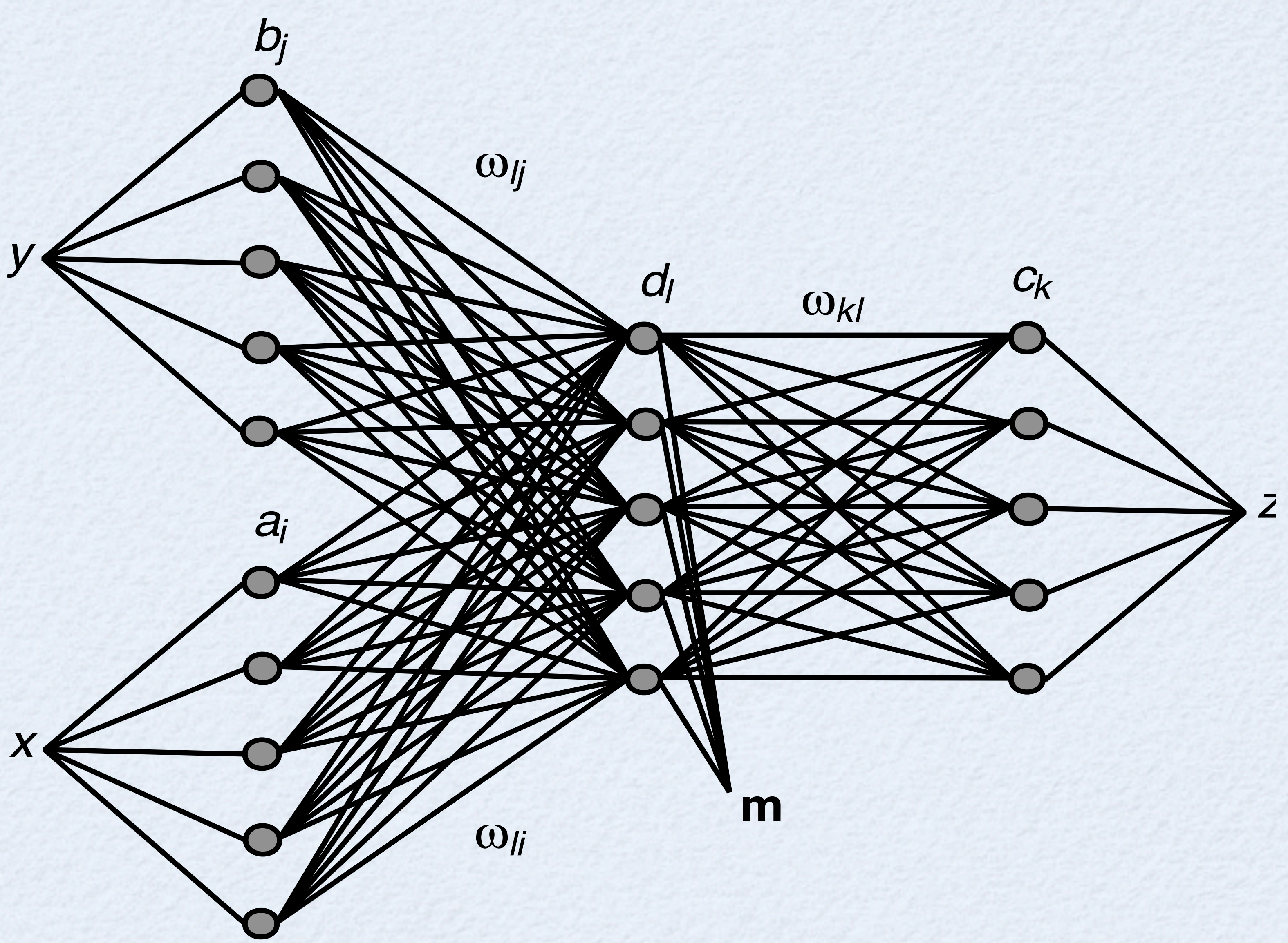
In networks: Gain fields

- The rate of the (idealized) neurons in b) are R=f(x)g(y)
- x+y is needed to reach target
- Adding these multiplicatively modulated 'gain fields' gives an estimate
- From Salinas and Their, 2000, Neuron



Nonlinearities via networks

- Form an intermediate representation in a `middle layer' of neurons with dimensionality D_m=D_x+D_y



Network nonlinearities

- First find the weights to put the inputs into this D_m-dimensional space

$$
\begin{aligned}
d_l(\mathbf{m} = [x\, y]) &= G_l\left[\alpha_l \left\langle \tilde{\phi}_l \mathbf{m} \right\rangle + J_l^b\right] \\
&= G_l\left[\alpha_l \left(\tilde{\phi}_l^{m_1}\hat{x} + \tilde{\phi}_l^{m_2}\hat{y}\right) + J_l^b\right] \\
&= G_l\left[\sum_i \omega_{li}^{m_1} a_i(x) + \sum_j \omega_{lj}^{m_2} b_j(y) + J_l^b\right],
\end{aligned}
$$

$$
\omega_{li}^{m_1} = \alpha_l \phi_i^x \tilde{\phi}_l^{m_1} \qquad\qquad \omega_{lj}^{m_2} = \alpha_l \phi_j^y \tilde{\phi}_l^{m_2}
$$

Network nonlinearities

- Then use the transformation decoders we found earlier to 'extract' the product

$$
\begin{aligned}
c_k(f(\mathbf{m})) &= G_k\left[\alpha_k\left(\tilde{\phi}_k f(\mathbf{m})\right) + J_k^b\right] \\
&= G_k\left[\alpha_k\left(\tilde{\phi}_k \sum_l d_l(\mathbf{m})\phi_l^f\right) + J_k^b\right] \\
&= G_k\left[\sum_l \omega_{kl} d_l(\mathbf{m}) + J_k^b\right]
\end{aligned}
$$

$$\omega_{kl} = \alpha_k \tilde{\phi}_k \phi_l^f$$

Comments

- Notice the similarities between 1D and *n*D case
- Essentially, we have derived the `hidden layer' typical in ANNs. ANNs with such a layer can compute any function of the input
- Could makes nonlinearities internal to the cell (evidence is controversial, but mounting): would need far fewer cells to do multiplication

Comments*

- Choosing encoding vectors can improve the specific nonlinearity being done
  - E.g. thresholding
  - E.g. quadratic function of x
  - E.g. placing encoders along the diagonal of variables being multiplied