

**SYDE 556/750**

**Simulating Neurobiological Systems**  
**Lecture 3: Representations**

Andreas Stöckel

January 14 & 16 & 21, 2020



UNIVERSITY OF  
**WATERLOO**

FACULTY OF  
ENGINEERING



# Visual Cortex



## Mapping receptive fields

cell activity

behavior

overall



ongoing




# NEF Principle 1: Representation

## **NEF Principle 1 – Representation**

*Groups* (“populations”, or “ensembles”) of neurons *represent* represent values via nonlinear encoding and linear decoding.

# Lossless Codes

INTERNATIONAL ALPHABET FLAGS, PHONETIC ALPHABET, MORSE CODE AND SEMAPHORE ALPHABET															
<b>A</b> ALFA  <small>Alphabet: A Z 10 1</small>															

A  
B  
C  
D  
E  
F  
G  
H  
I  
J  
K  
L  
M  
N  
O  
P  
Q  
R  
S  
T

U  
V  
W  
X  
Y  
Z

1  
2  
3  
4  
5  
6  
7  
8  
9  
0

Encoding:  $a = f(x)$

Decoding:  $x = f^{-1}(a)$

## Binary numbers: Nonlinear encoding, linear decoding

- Represent a natural number between 0 and  $2^n - 1$  as  $n$  binary digits.

## Binary numbers: Nonlinear encoding, linear decoding

- ▶ Represent a natural number between 0 and  $2^n - 1$  as  $n$  binary digits.
- ▶ **Nonlinear encoding**

$$a_i = (f(x))_i = \begin{cases} 1 & \text{if } x - 2^i \lfloor \frac{x}{2^i} \rfloor > 2^{i-1}, \\ 0 & \text{otherwise.} \end{cases}$$

## Binary numbers: Nonlinear encoding, linear decoding

- Represent a natural number between 0 and  $2^n - 1$  as  $n$  binary digits.
- **Nonlinear encoding**

$$a_i = (f(x))_i = \begin{cases} 1 & \text{if } x - 2^i \lfloor \frac{x}{2^i} \rfloor > 2^{i-1}, \\ 0 & \text{otherwise.} \end{cases}$$

- **Linear decoding**

$$x = f^{-1}(\mathbf{a}) = \sum_{i=0}^{n-1} 2^i a_i = \mathbf{F}\mathbf{a} = \begin{pmatrix} 1 & 2 & \dots & 2^{n-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix}.$$



## Binary numbers: Nonlinear encoding, linear decoding

- Represent a natural number between 0 and  $2^n - 1$  as  $n$  binary digits.
- **Nonlinear encoding**

$$a_i = (f(x))_i = \begin{cases} 1 & \text{if } x - 2^i \lfloor \frac{x}{2^i} \rfloor > 2^{i-1}, \\ 0 & \text{otherwise.} \end{cases}$$

- **Linear decoding**

$$x = f^{-1}(\mathbf{a}) = \sum_{i=0}^{n-1} 2^i a_i = \mathbf{F}\mathbf{a} = \begin{pmatrix} 1 & 2 & \dots & 2^{n-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix}.$$

- This is a **distributed code**.

## Binary numbers: Nonlinear encoding, linear decoding

- Represent a natural number between 0 and  $2^n - 1$  as  $n$  binary digits.
- **Nonlinear encoding**

$$a_i = (f(x))_i = \begin{cases} 1 & \text{if } x - 2^i \lfloor \frac{x}{2^i} \rfloor > 2^{i-1}, \\ 0 & \text{otherwise.} \end{cases}$$

- **Linear decoding**

$$x = f^{-1}(\mathbf{a}) = \sum_{i=0}^{n-1} 2^i a_i = \mathbf{F}\mathbf{a} = \begin{pmatrix} 1 & 2 & \dots & 2^{n-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix}.$$

- This is a **distributed code**. But, **not robust** against additive noise!

# Lossy codes

- **Lossy code**

Inverse  $f^{-1}$  does not exist, instead *approximate* the represented value

Encoding:  $\mathbf{a} = f(\mathbf{x})$

Decoding:  $\mathbf{x} \approx g(\mathbf{a})$

# Lossy codes

- ▶ **Lossy code**

Inverse  $f^{-1}$  does not exist, instead *approximate* the represented value

Encoding:  $\mathbf{a} = f(\mathbf{x})$

Decoding:  $\mathbf{x} \approx g(\mathbf{a})$

- ▶ **Examples**

- ▶ Audio, image, and video coding schemes (MP3, JPEG, H.264)

- ▶ Basis transformation onto first  $n$  principal components (PCA)

# Lossy codes

- ▶ **Lossy code**

Inverse  $f^{-1}$  does not exist, instead *approximate* the represented value

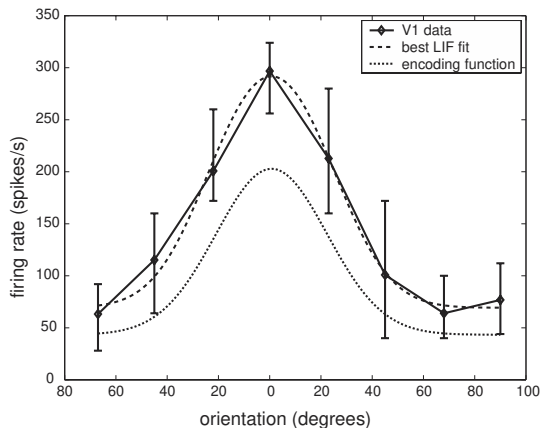
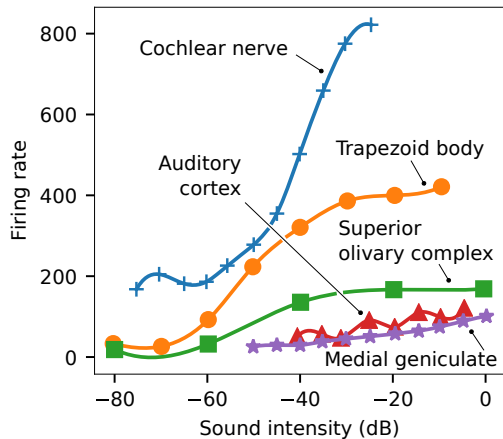
Encoding:  $\mathbf{a} = f(\mathbf{x})$

Decoding:  $\mathbf{x} \approx g(\mathbf{a})$

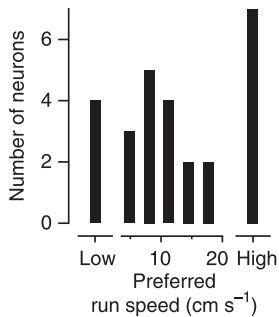
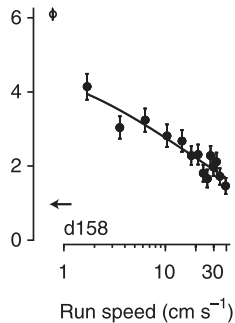
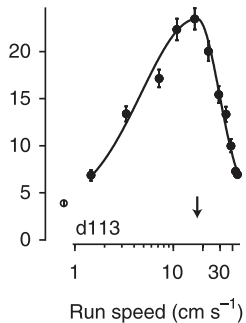
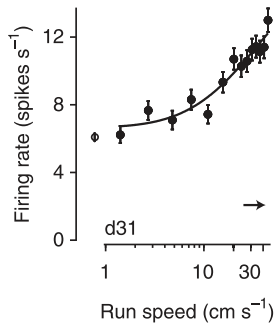
- ▶ **Examples**

- ▶ Audio, image, and video coding schemes (MP3, JPEG, H.264)
- ▶ Basis transformation onto first  $n$  principal components (PCA)
- ▶ **Neural Representations**

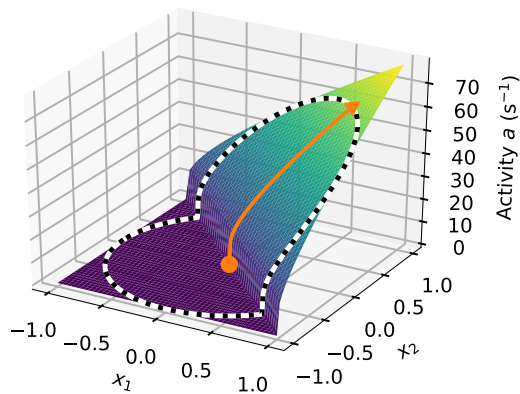
# Tuning curves (I)



## Tuning curves (II)

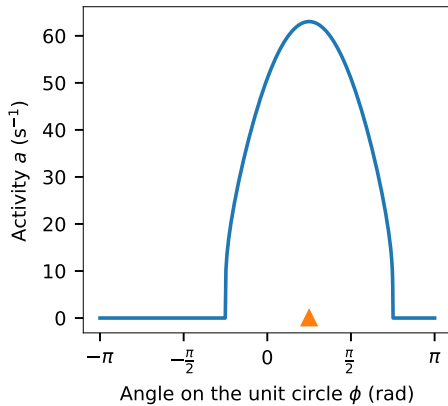
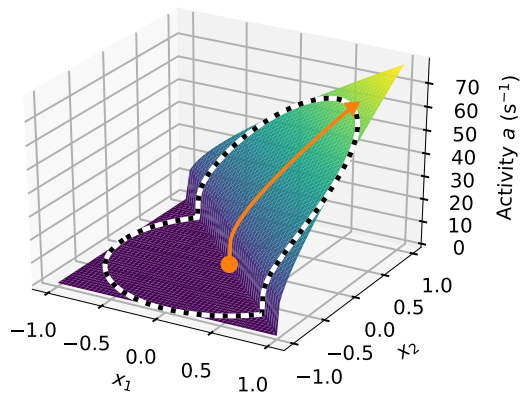


## Preferred Directions in Higher Dimensions: Representing 2D Values

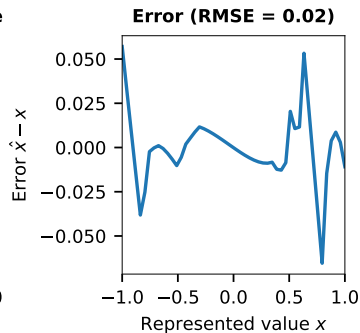
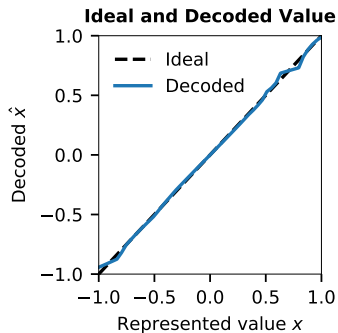
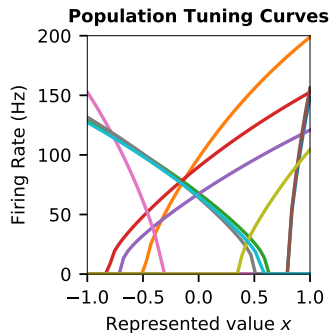




## Preferred Directions in Higher Dimensions: Representing 2D Values

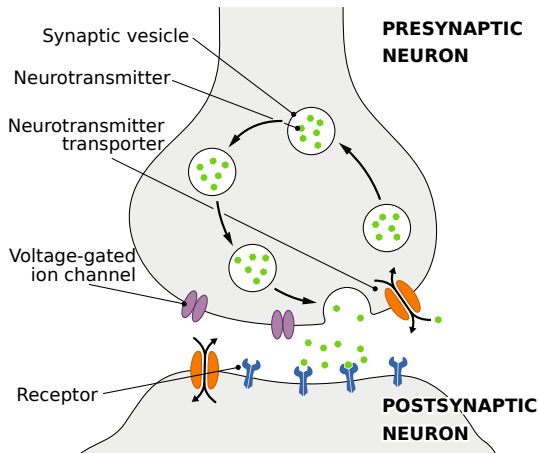


# Decoding Without Taking Noise Into Account



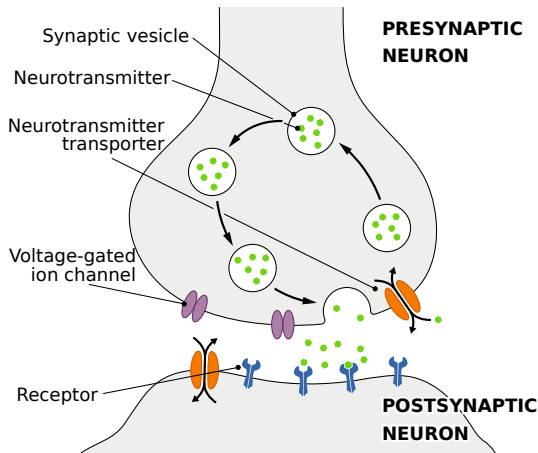
# Sources of Noise in Biological Neural Networks

- ▶ **Axonal jitter**  
Active axonal spike propagation
- ▶ **Vesicle release failure**  
10-30% of pre-synaptic events cause post-synaptic current
- ▶ **Neurotransmitter per vesicle**  
Varying amounts of neurotransmitter
- ▶ **Ion channel noise**  
Ion-channels are “binary”, stochastic
- ▶ **Thermal noise**
- ▶ **Network effects**  
Simple, noise-free inhibitory/excitatory networks produce irregular spike trains



# Sources of Noise in Biological Neural Networks

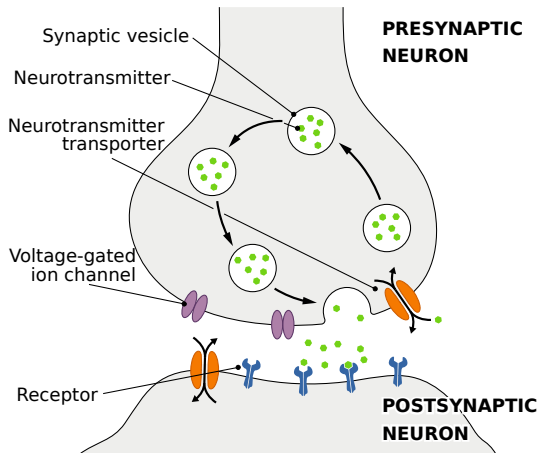
- ▶ **Axonal jitter**  
Active axonal spike propagation
- ▶ **Vesicle release failure**  
10-30% of pre-synaptic events cause post-synaptic current
- ▶ **Neurotransmitter per vesicle**  
Varying amounts of neurotransmitter
- ▶ **Ion channel noise**  
Ion-channels are “binary”, stochastic
- ▶ **Thermal noise**
- ▶ **Network effects**  
Simple, noise-free inhibitory/excitatory networks produce irregular spike trains



▶ **How to model?**

# Sources of Noise in Biological Neural Networks

- ▶ **Axonal jitter**  
Active axonal spike propagation
- ▶ **Vesicle release failure**  
10-30% of pre-synaptic events cause post-synaptic current
- ▶ **Neurotransmitter per vesicle**  
Varying amounts of neurotransmitter
- ▶ **Ion channel noise**  
Ion-channels are “binary”, stochastic
- ▶ **Thermal noise**
- ▶ **Network effects**  
Simple, noise-free inhibitory/excitatory networks produce irregular spike trains



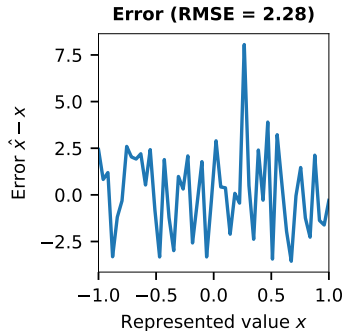
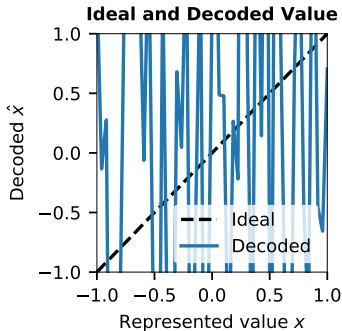
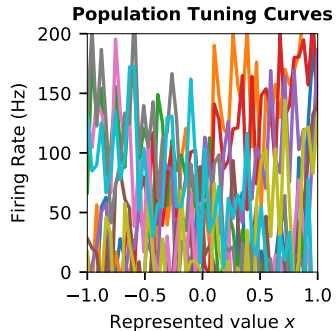
- ▶ **How to model?** Gaussian noise

## NEF Principle 0: Noise

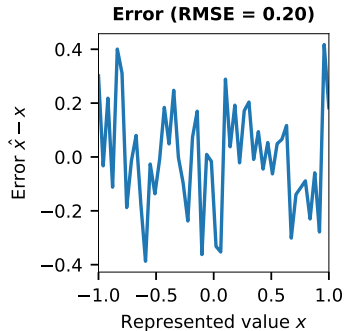
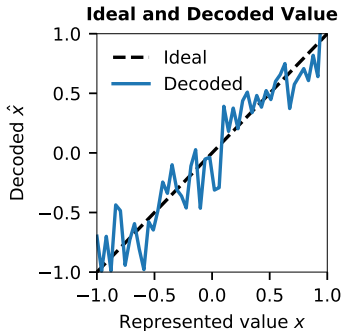
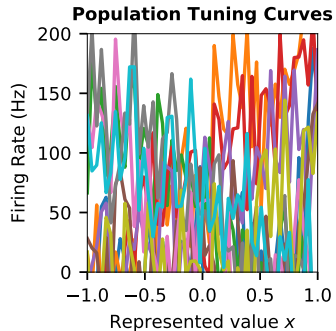
### **NEF Principle 0 – Noise**

Biological neural systems are subject to significant amounts of noise from various sources. Any analysis of such systems must take the effects of noise into account.

# Decoding Noisy $\mathbf{A}$ Without Taking Noise Into Account



# Decoding Noisy $\mathbf{A}$ Accounting for Noise



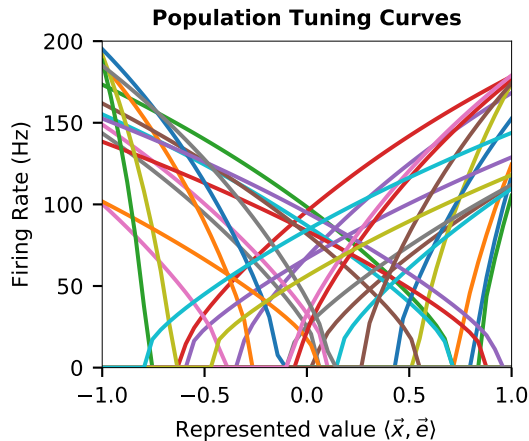


# Summary: Building a model of neural representation (Encoding)

## Encoding

- ▶ Select  $d$ , possible range  $\mathbf{x} \in \mathbb{X}$ , usually  $\mathbb{X} = \{\mathbf{x} \mid \|\mathbf{x}\| \leq r, \mathbf{x} \in \mathbb{R}^d\}$  ( $r = 1$ )
- ▶ Select number of neurons  $n$
- ▶ Select tuning curves, maximum rates  $\Rightarrow \mathbf{e}_i, \alpha_i, J_i^{\text{bias}}$ 
  - ▶ Sample  $\mathbf{e}_i$  from unit-sphere
  - ▶ Uniformly distribute  $\alpha_i, J_i^{\text{bias}}$
- ▶ Encoding equation:

$$a_i(\mathbf{x}) = G[\alpha_i \langle \mathbf{e}_i, \mathbf{x} \rangle + J_i^{\text{bias}}]$$



## Summary: Building a model of neural representation (Decoding)

### Decoding

- ▶ Uniformly sample  $N$  samples from  $\mathbb{X}$ ,  
 $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$

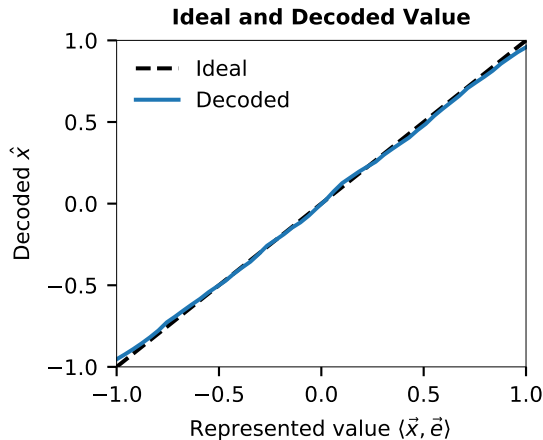
- ▶ Compute  $\mathbf{A}$ , where  $(\mathbf{A}) = a_i()$

- ▶ Decoder computation:

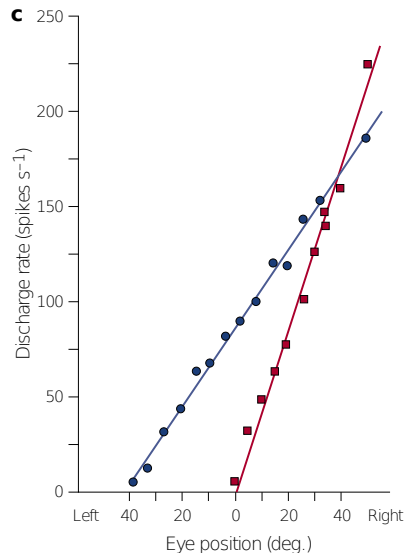
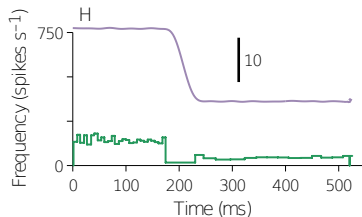
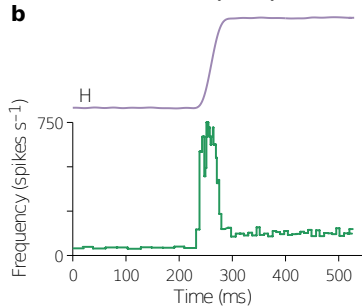
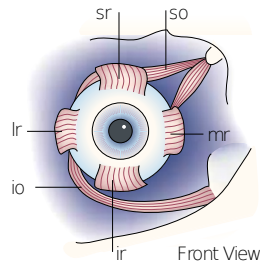
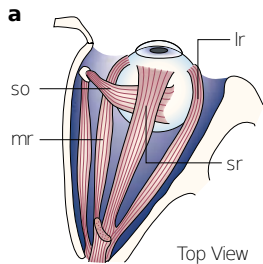
$$\mathbf{D} = (\mathbf{A}\mathbf{A}^T + N\sigma^2\mathbf{I})\mathbf{A}\mathbf{X}^T$$

- ▶ Decoding equation:

$$\hat{\mathbf{X}} = \mathbf{D}\mathbf{A}$$



## Example: Horizontal Eye Position (1D)



## Example: Horizontal Eye Position (1D) (cont.)

### ► Step 1: System Description

- What is being represented?
  - $x$  is the horizontal eye position
- What is the tuning curve shape?
  - Linear, low  $\tau_{\text{ref}}$ , high  $\tau_{\text{RC}}$
  - $e_i \in \{1, -1\}$
  - Firing rates up to  $300 \text{ s}^{-1}$

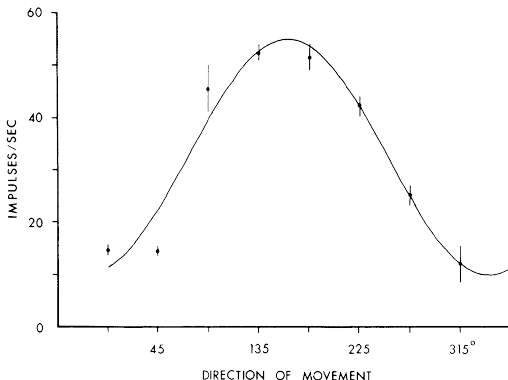
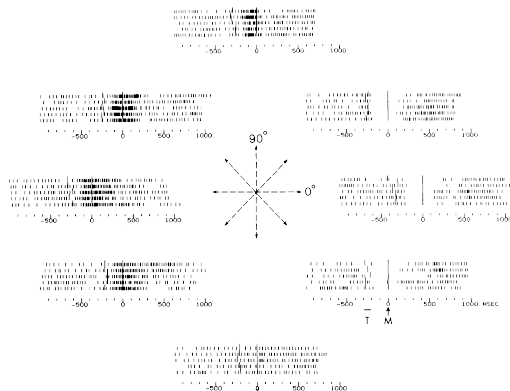
### ► Step 2: Design Specification

- Range of values
  - $\mathbb{X} = [-60, 60]$
- Amount of noise
  - About 20% of  $\max(\mathbf{A})$

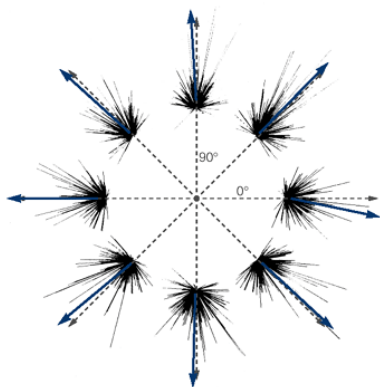
### ► Step 3: Implementation

- Choose tuning curve parameters
- Compute decoders

# Example: Arm Movements (2D)



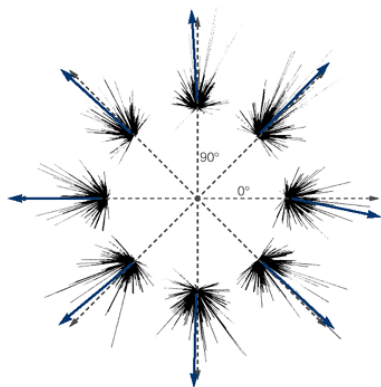
## Example: Arm Movements (2D) (cont.)



- ▶ Experiment by Georgopoulos et al., 1982
- ▶ Preferred arm movement directions  $\mathbf{e}_i$
- ▶ **Idea:** *Population Vectors*, decode using

$$\hat{\mathbf{x}} = \sum_{i=1}^n a_i(\mathbf{x}) \mathbf{e}_i = \mathbf{E} \mathbf{A}$$

## Example: Arm Movements (2D) (cont.)

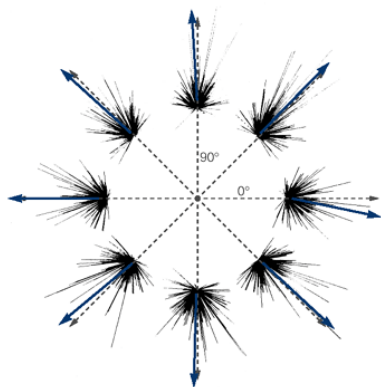


- ▶ Experiment by Georgopoulos et al., 1982
- ▶ Preferred arm movement directions  $\mathbf{e}_i$
- ▶ **Idea:** *Population Vectors*, decode using

$$\hat{\mathbf{x}} = \sum_{i=1}^n a_i(\mathbf{x}) \mathbf{e}_i = \mathbf{E}\mathbf{A}$$

- ⊕ Good direction estimate

## Example: Arm Movements (2D) (cont.)



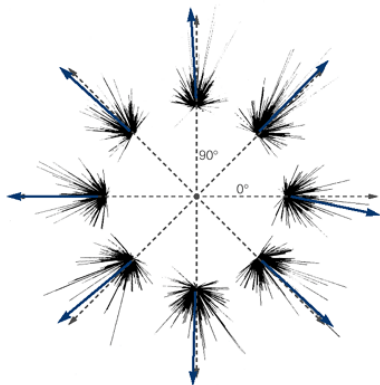
- ▶ Experiment by Georgopoulos et al., 1982
- ▶ Preferred arm movement directions  $\mathbf{e}_i$
- ▶ **Idea:** *Population Vectors*, decode using

$$\hat{\mathbf{x}} = \sum_{i=1}^n a_i(\mathbf{x}) \mathbf{e}_i = \mathbf{E} \mathbf{A}$$

- ⊕ Good direction estimate
- ⊖ Cannot reconstruct magnitude



## Example: Arm Movements (2D) (cont.)



- ▶ Experiment by Georgopoulos et al., 1982
- ▶ Preferred arm movement directions  $\mathbf{e}_i$
- ▶ **Idea:** *Population Vectors*, decode using

$$\hat{\mathbf{x}} = \sum_{i=1}^n a_i(\mathbf{x}) \mathbf{e}_i = \mathbf{E} \mathbf{A}$$

- + Good direction estimate
- Cannot reconstruct magnitude

**The NEF does not use population vectors!**

## Example: Arm Movements (2D) (cont.)

### ► Step 1: System Description

- What is being represented?
  - $\mathbf{x}$  the movement direction (or hand position)
- What is the tuning curve shape?
  - Bell-shaped
  - Encoders are randomly distributed along the unit circle
  - Firing rates up to  $60\text{ s}^{-1}$

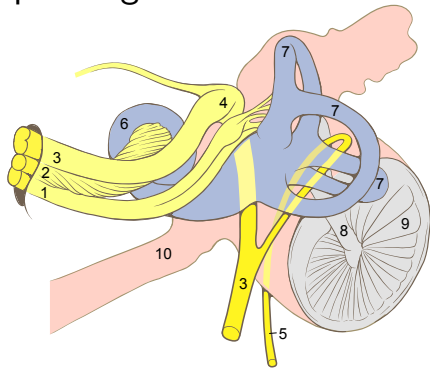
### ► Step 2: Design Specification

- Range of values
  - $\mathbb{X} = \{\mathbf{x} \mid \|\mathbf{x}\| \leq r, \mathbf{x} \in \mathbb{R}^2\}$
- Amount of noise
  - About 20% of  $\max(\mathbf{A})$

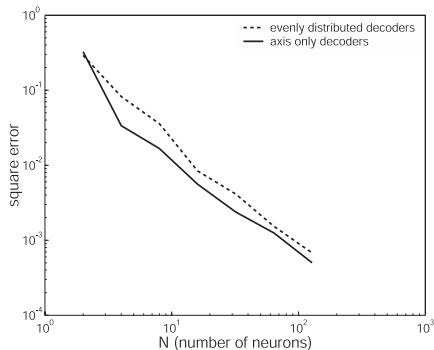
### ► Step 3: Implementation

- Choose tuning curve parameters
- Compute decoders

## Example: Higher Dimensional Representation

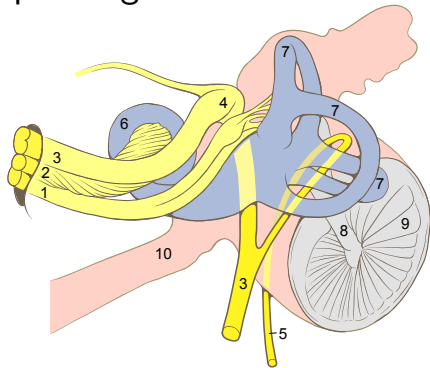


- ▶ Vestibular system senses head acceleration in 3D
- ▶ Axis aligned, must choose  $\mathbf{e}_i \in \{[1, 0, 0], [-1, 0, 0], \dots, [0, 0, -1]\}$

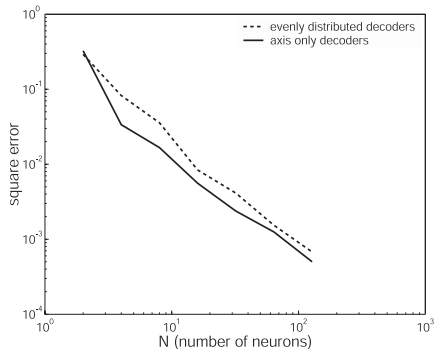


- ▶ Same as three 1D populations
- ▶ Slightly lower precision

## Example: Higher Dimensional Representation



- ▶ Vestibular system senses head acceleration in 3D
- ▶ Axis aligned, must choose  $\mathbf{e}_i \in \{[1, 0, 0], [-1, 0, 0], \dots, [0, 0, -1]\}$



- ▶ Same as three 1D populations
- ▶ Slightly lower precision
- ▶ **Encoders affect accuracy**

# Administration

- ▶ **Assignment 1 has been released.**

The due date has been adjusted to January, 30.

- ▶ Some new potential times for office hours

Mon 15:30–16:30, Mon 16:30–17:30, Tue 15:00–16:00,

Thu 11:30–12:30 (current slot), Thu 12:30–13:30

# Image sources

## **Title slide**

“The Ultimate painting.”

Author: Clark Richert.

From Wikimedia.