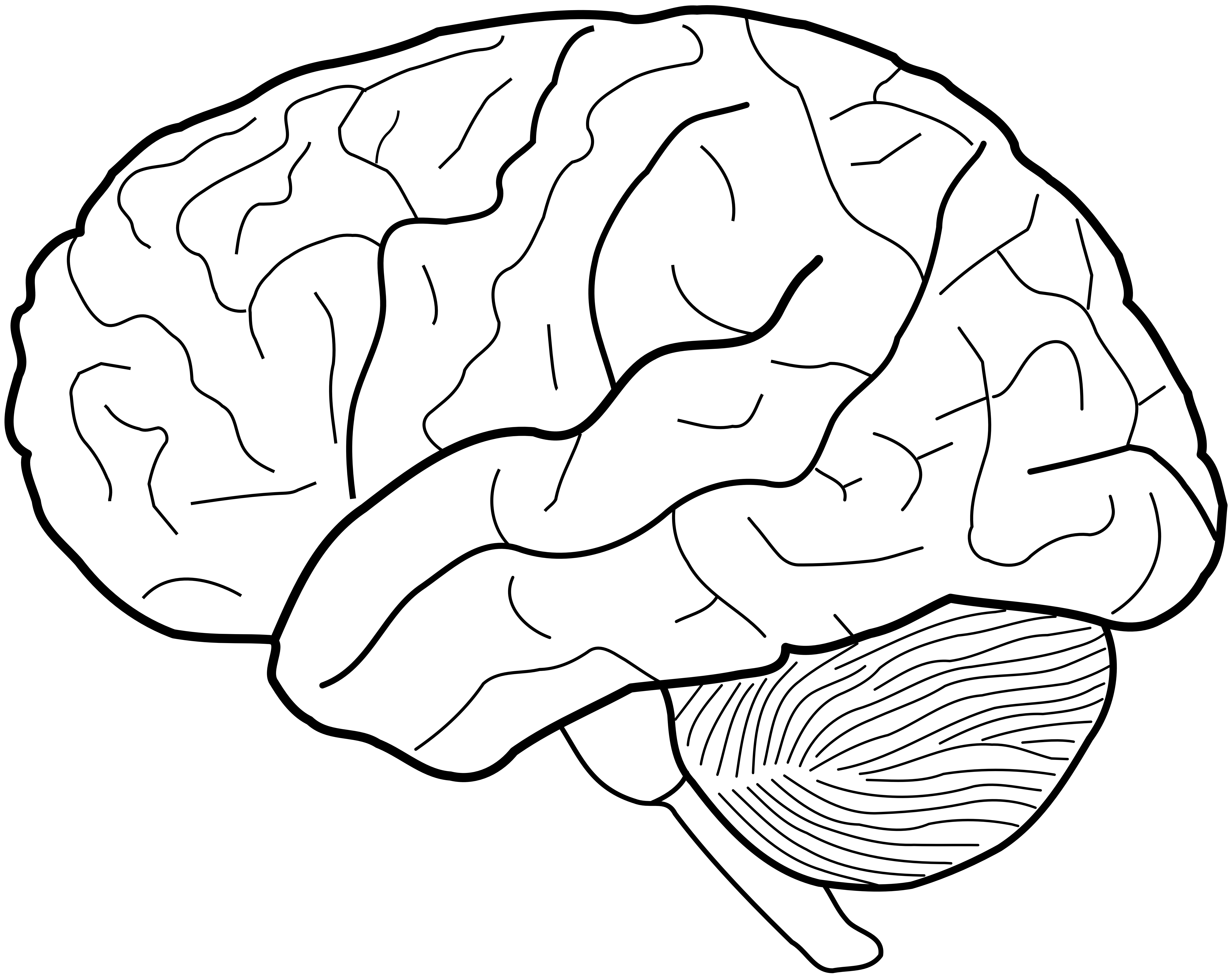
Neuroscience Introduction

hhmi | janelia Research Campus

databricks

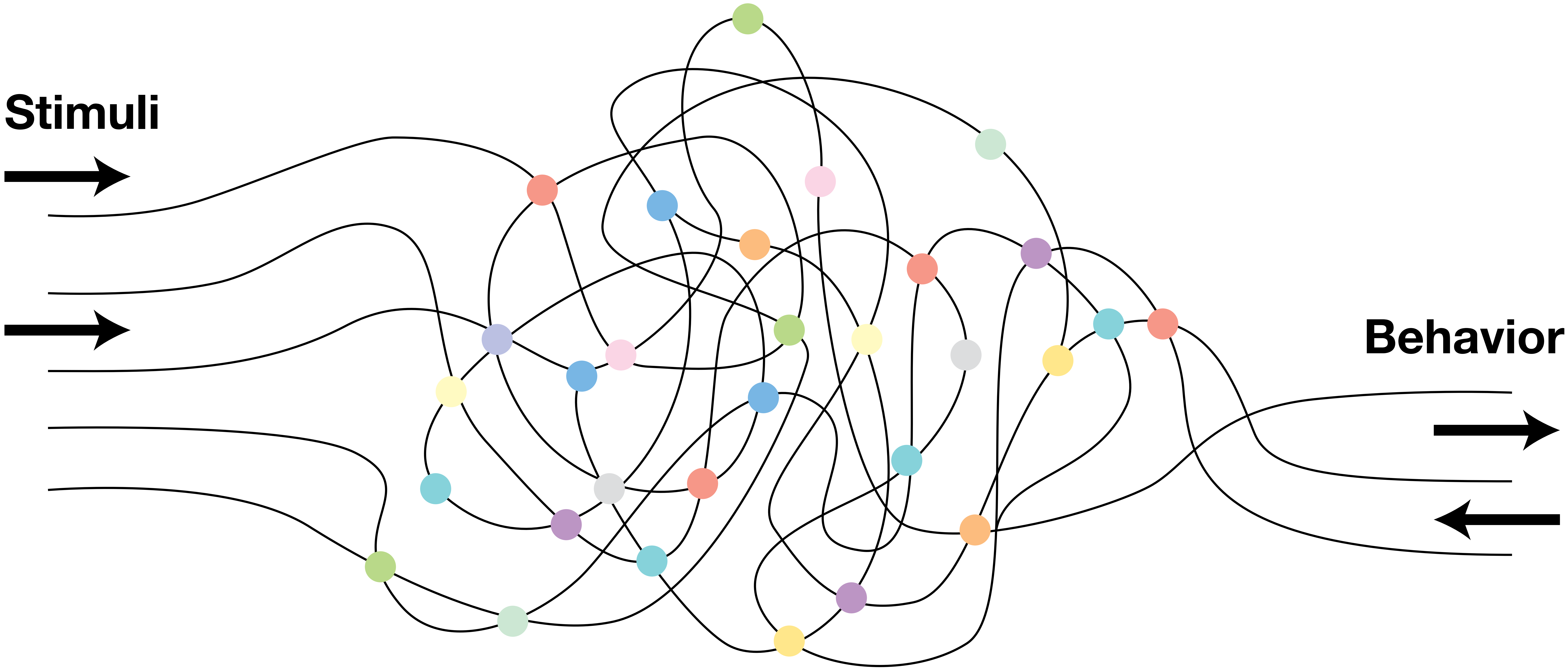
The brain



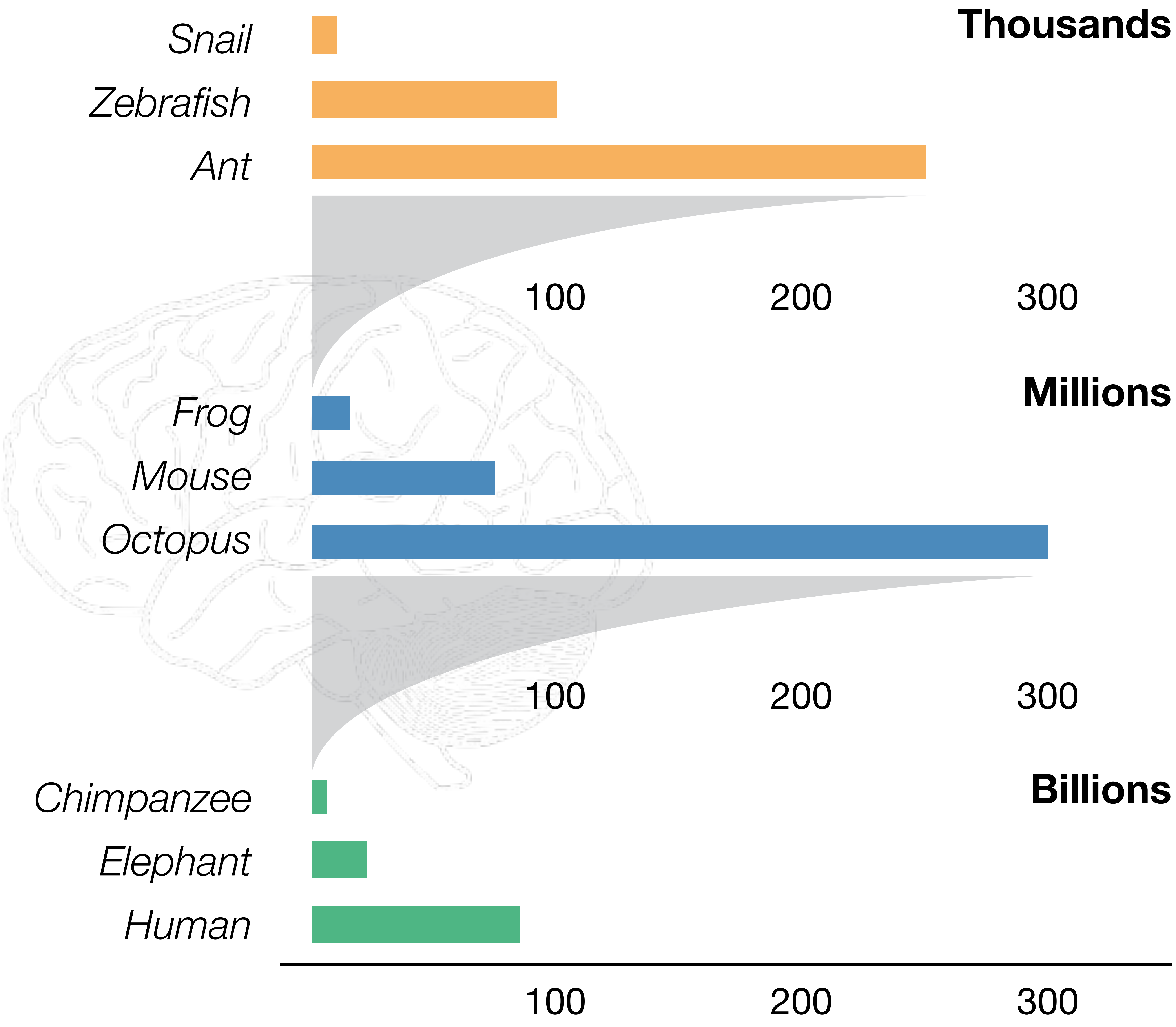
"As humans, we can identify galaxies light years away, we can study particles smaller than an atom. **But we still haven't unlocked the mystery of the three pounds of matter that sits between our ears**."

President Obama

The brain

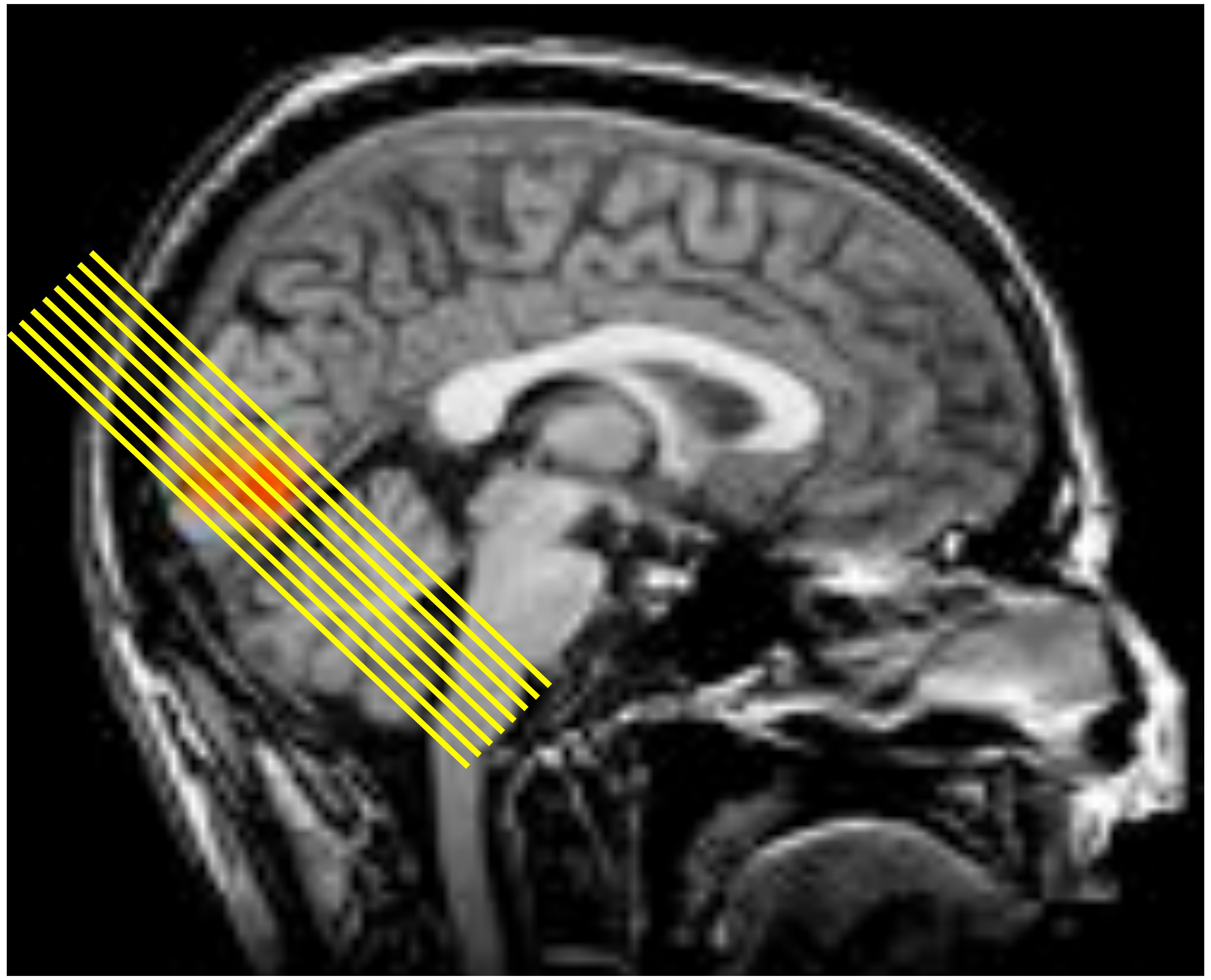


Numbers of neurons

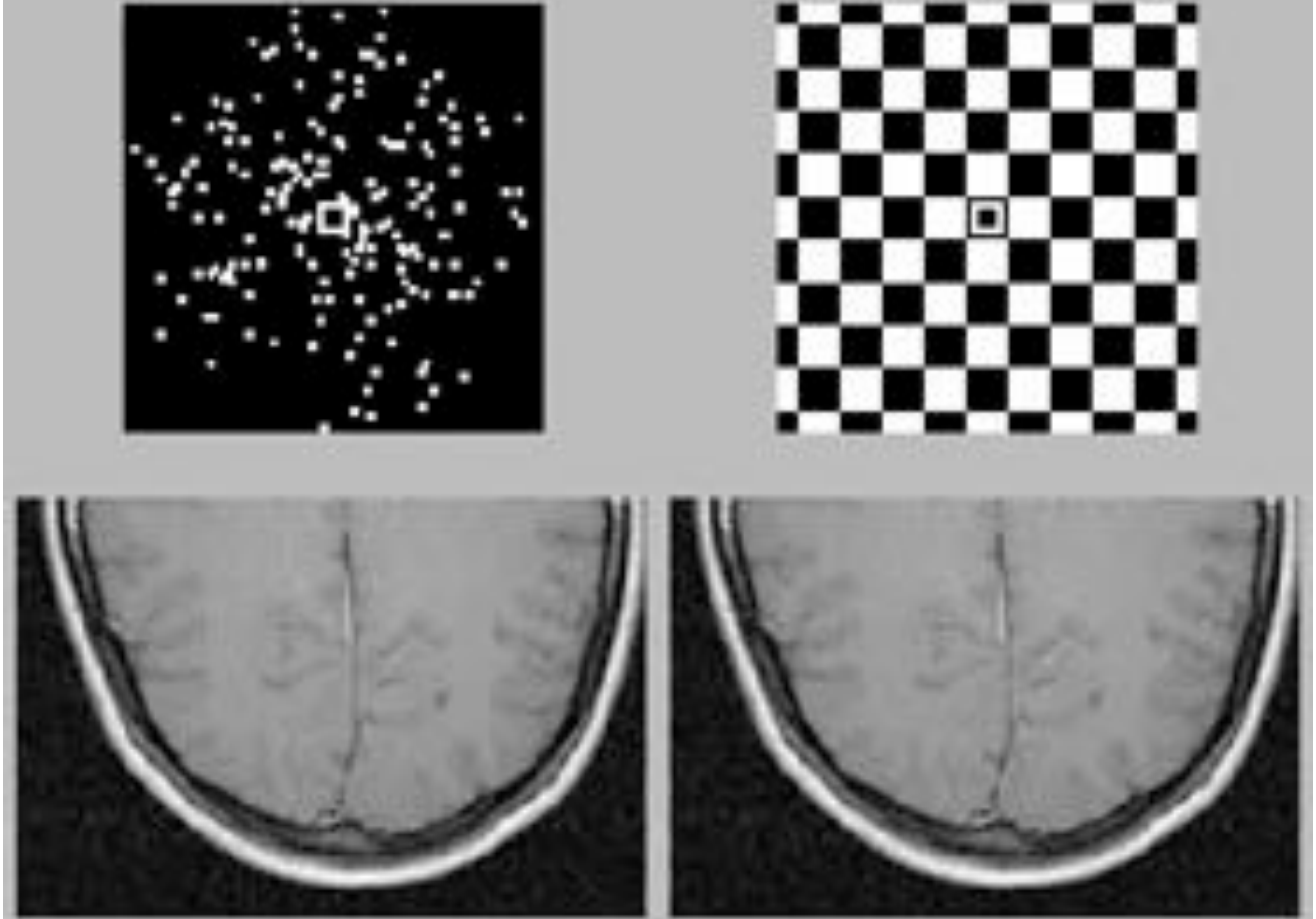


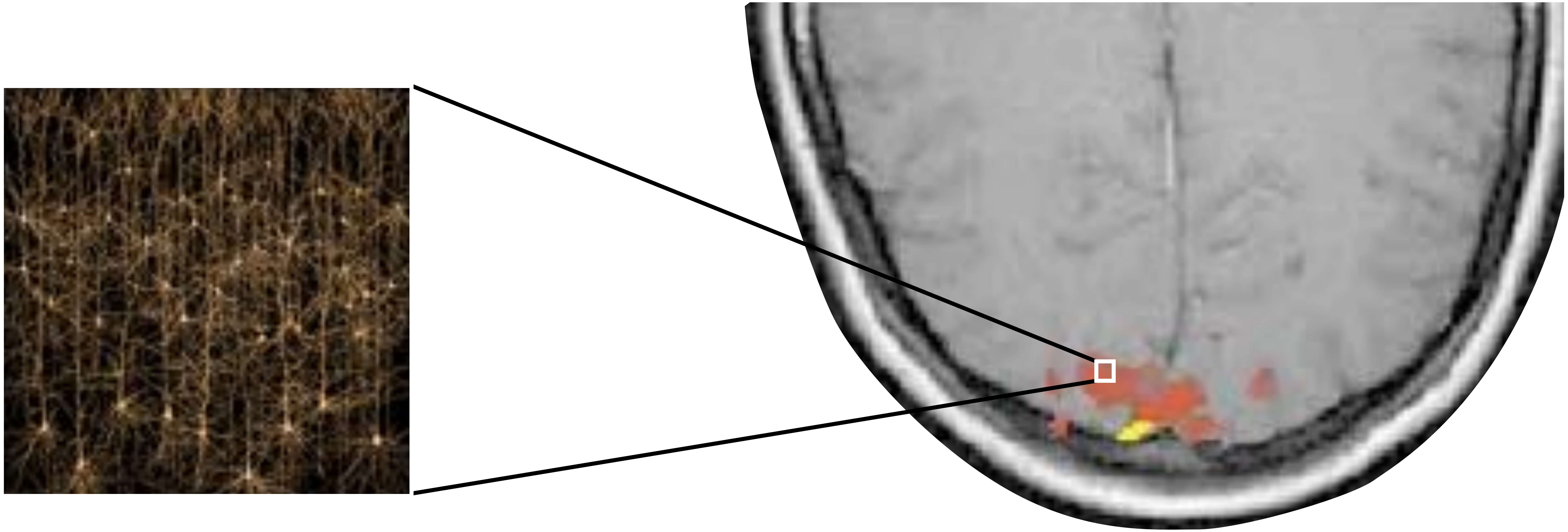
Studying the brain in humans

fMRI scanner



human brain

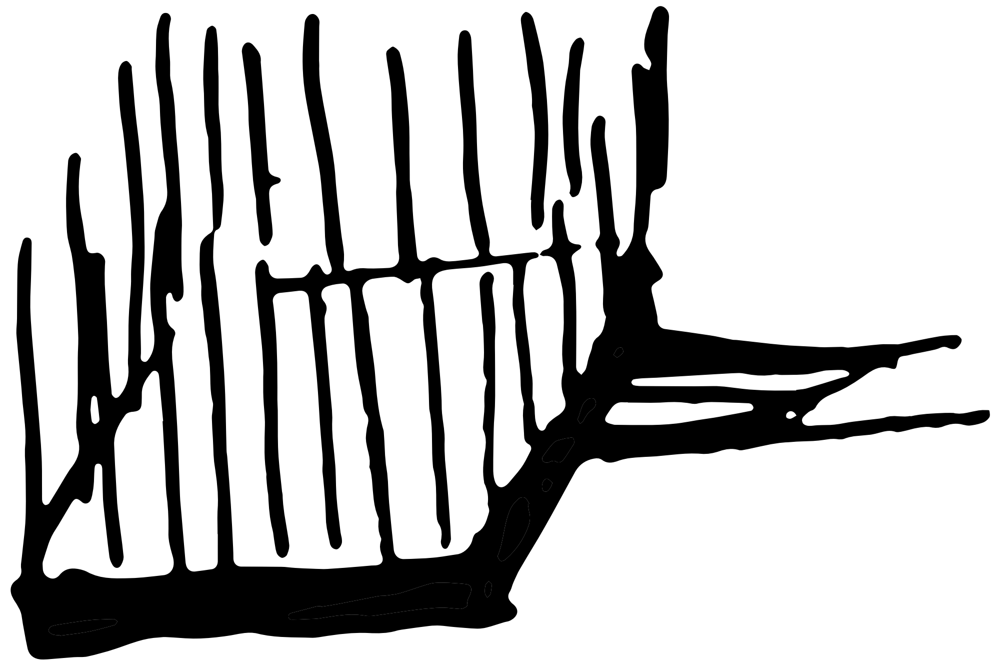


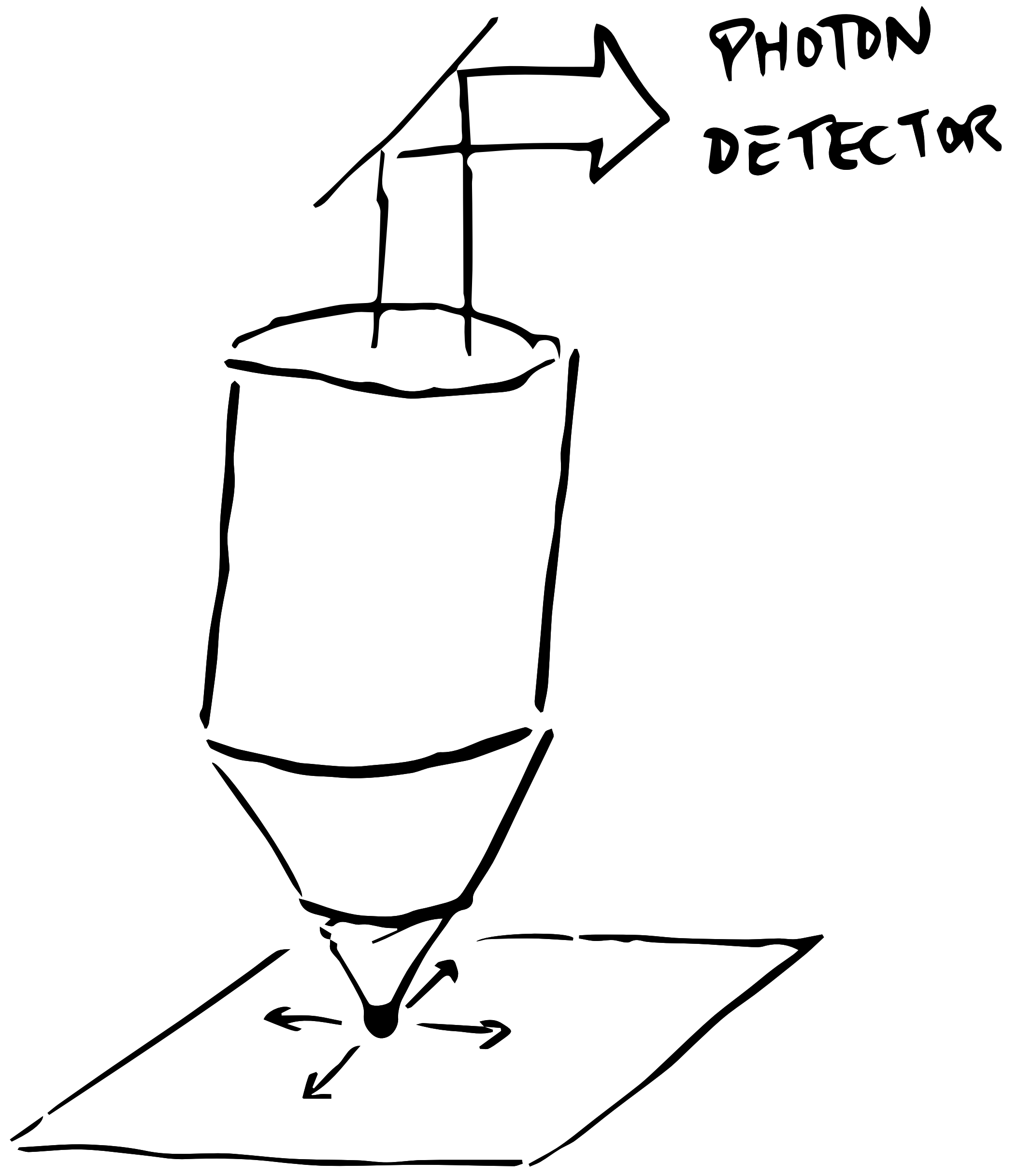


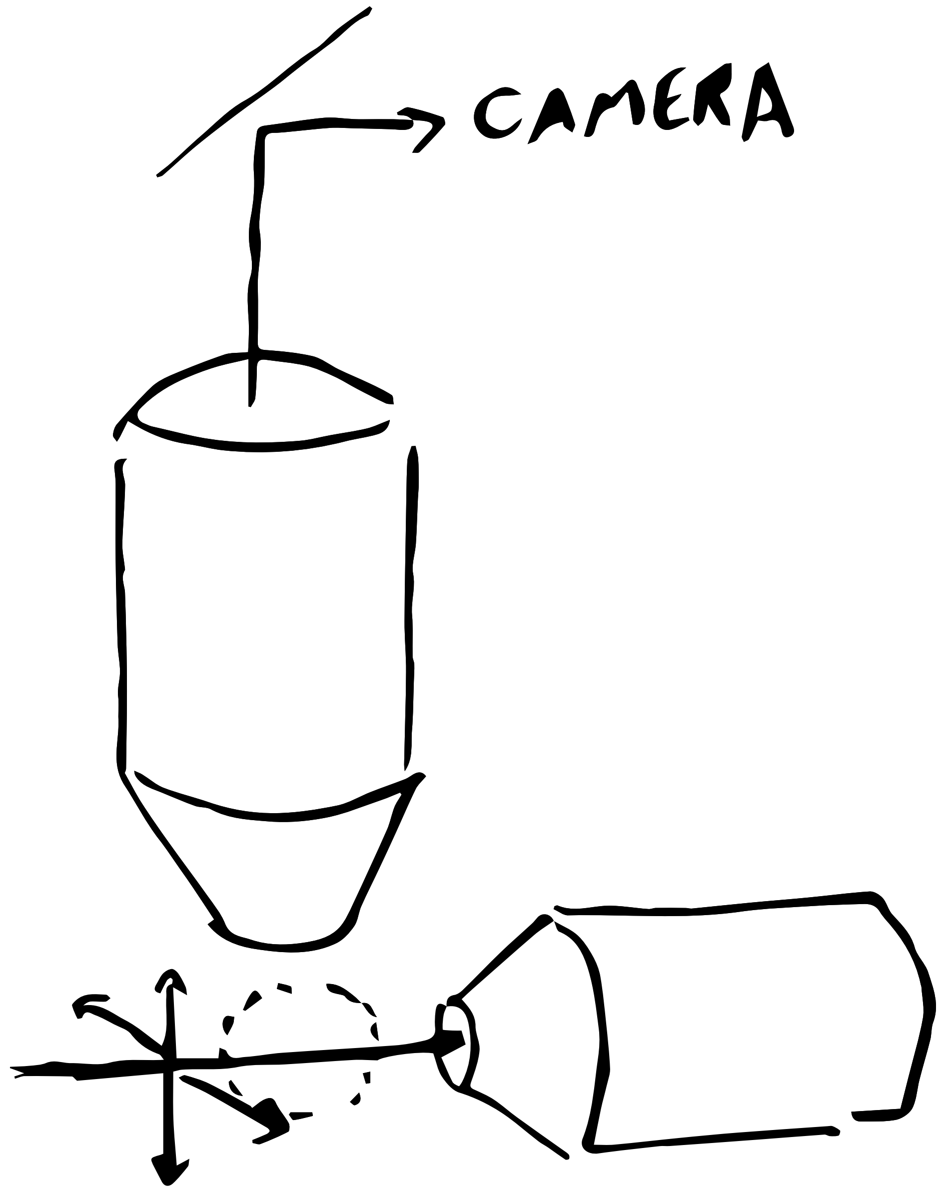
~50,000 neurons per cubic millimeter

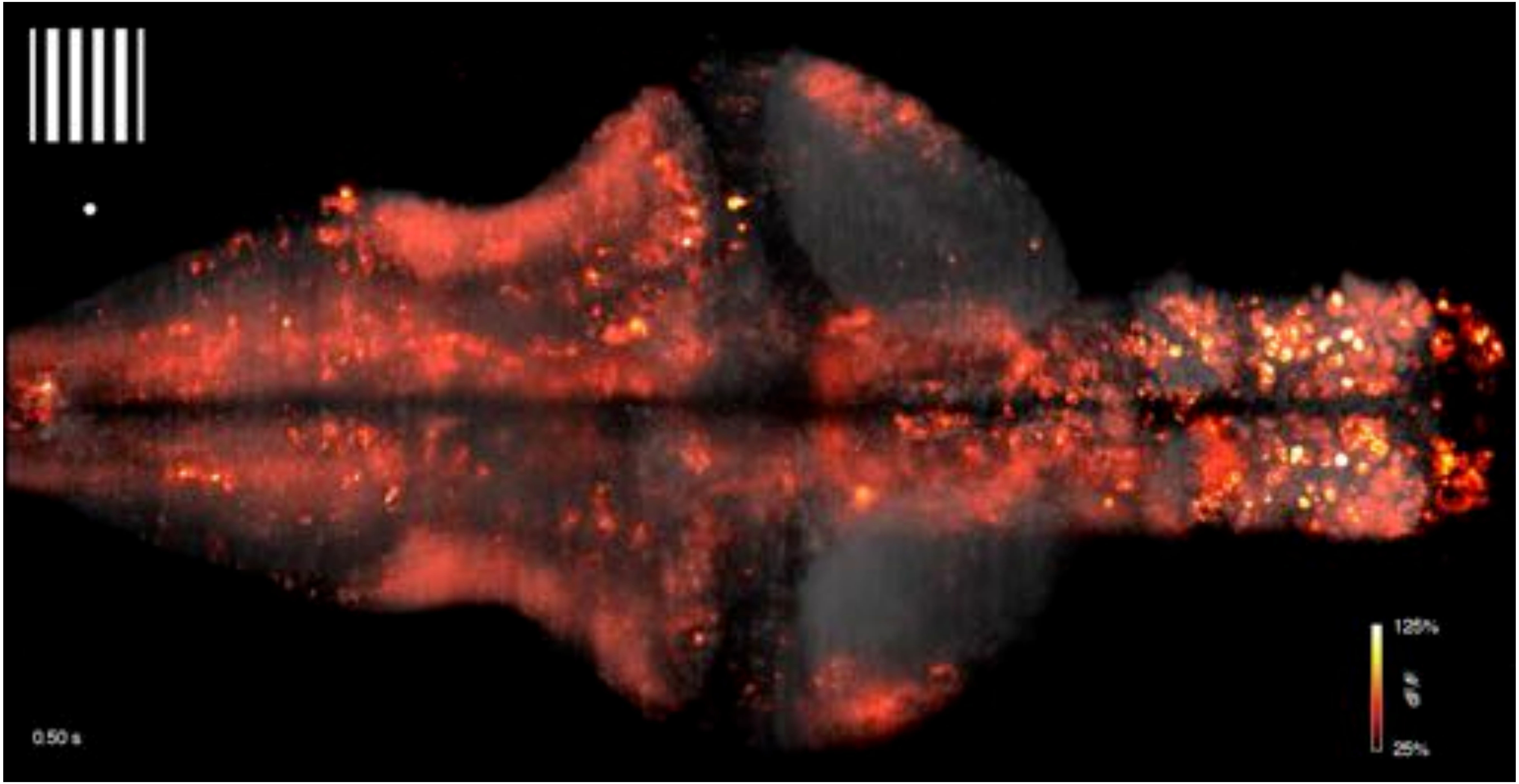
-> need higher resolution!

multielectrode	*two-photon*	*light-sheet*
10-100	*100-1000*	*100000*

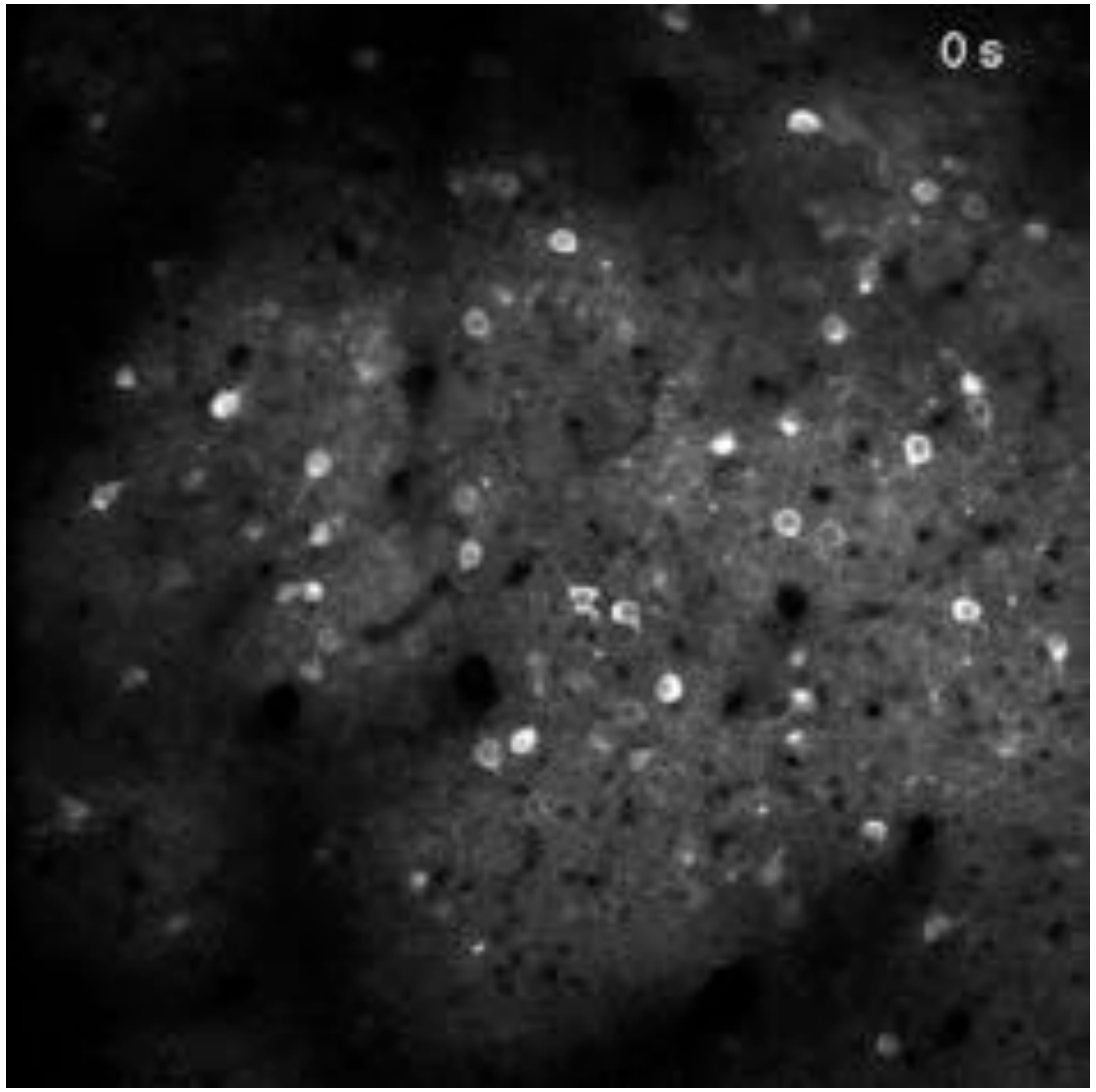


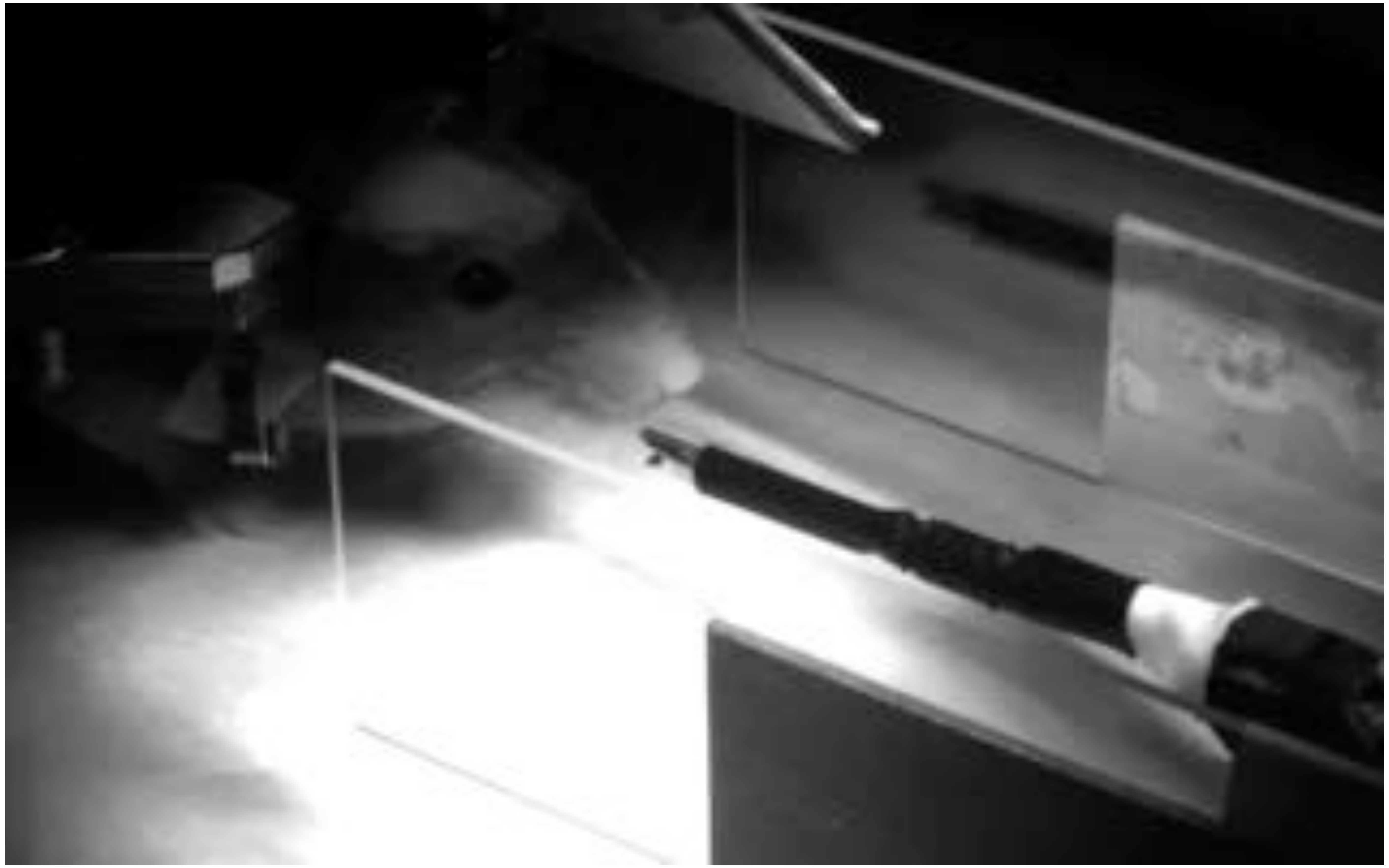






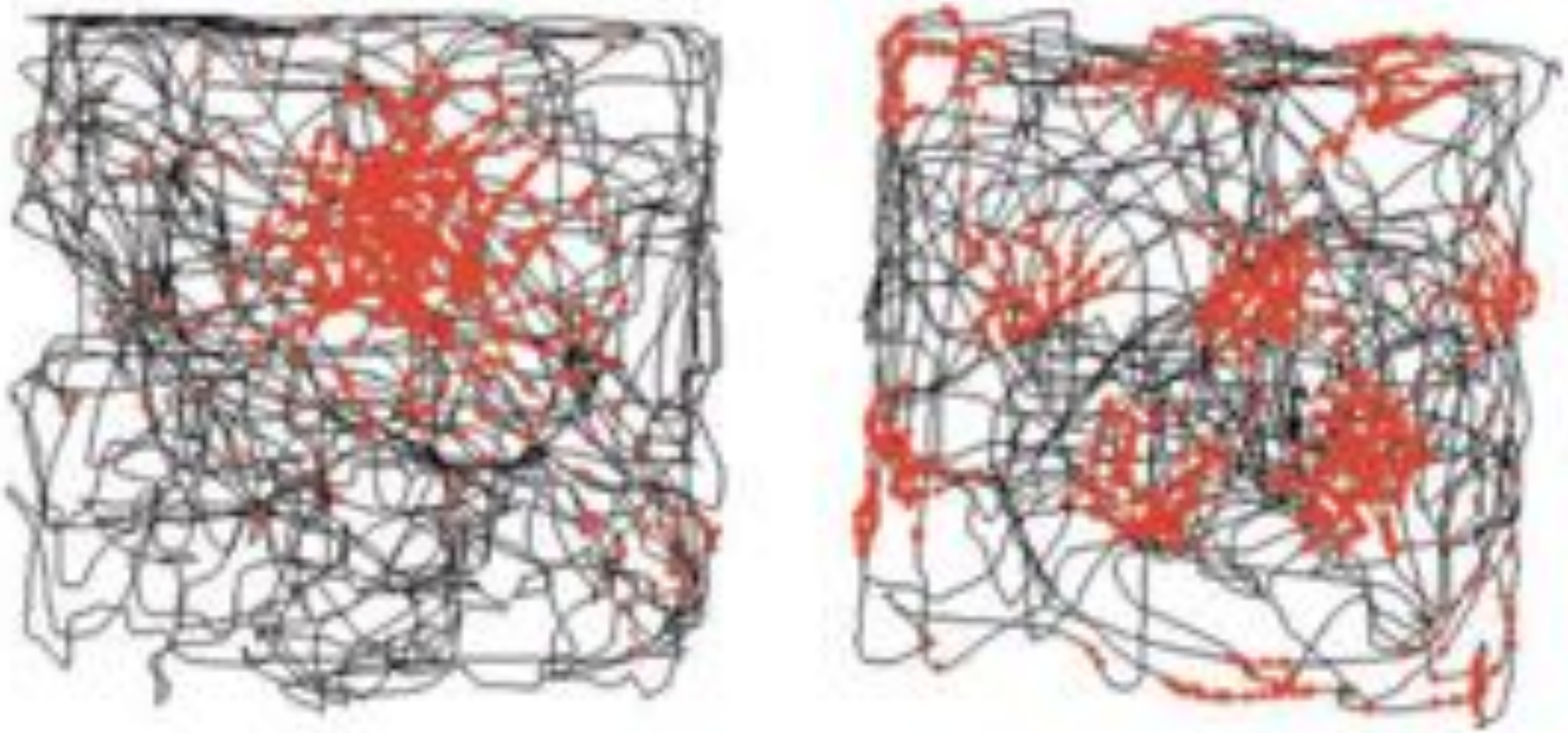
Vladimirov, et al., 2014





Sofroniew, et al., 2014

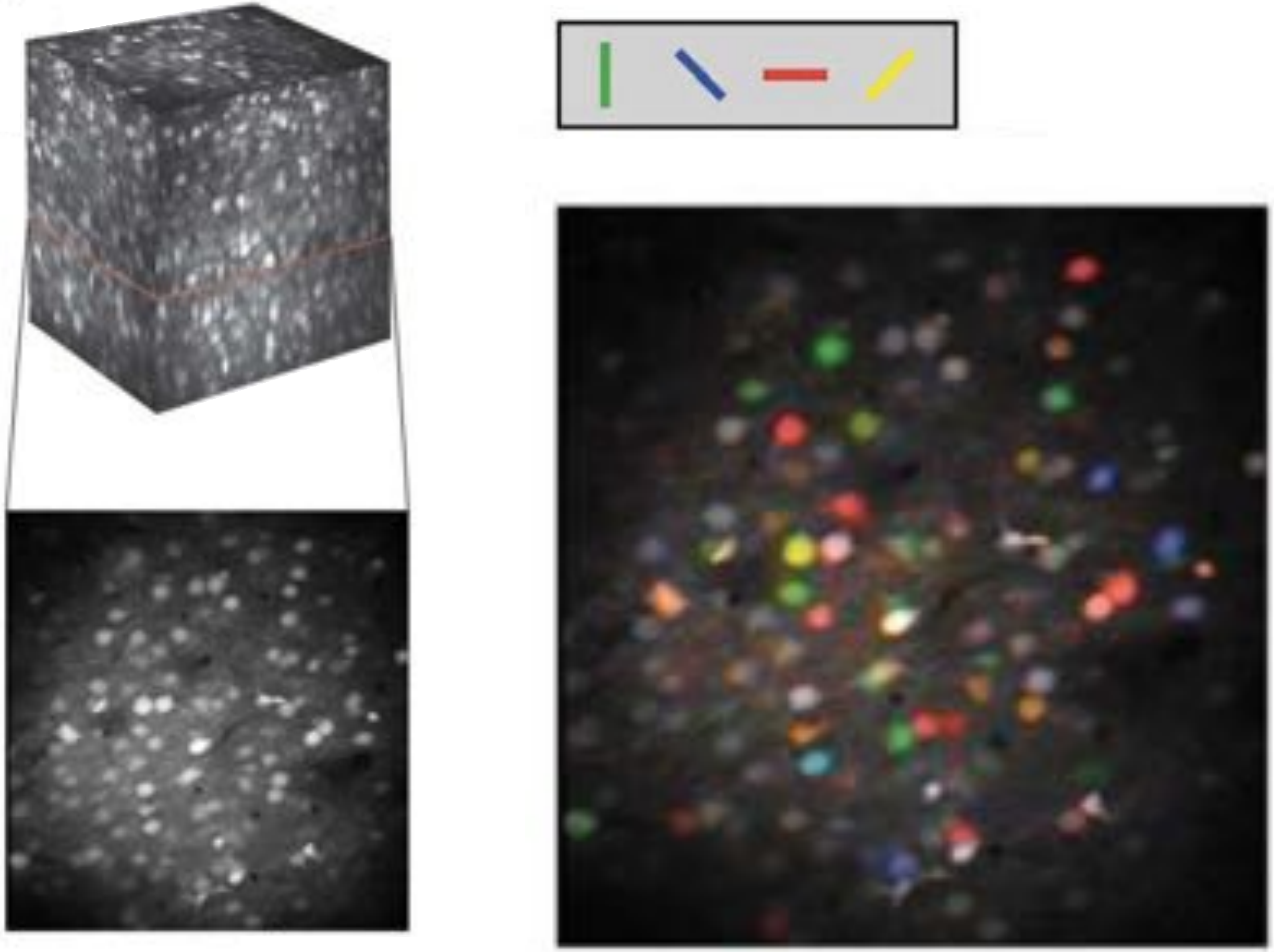
relating neuronal responses to properties of an animal and its environment



“place cell” “grid cell”

position of a mouse in maze

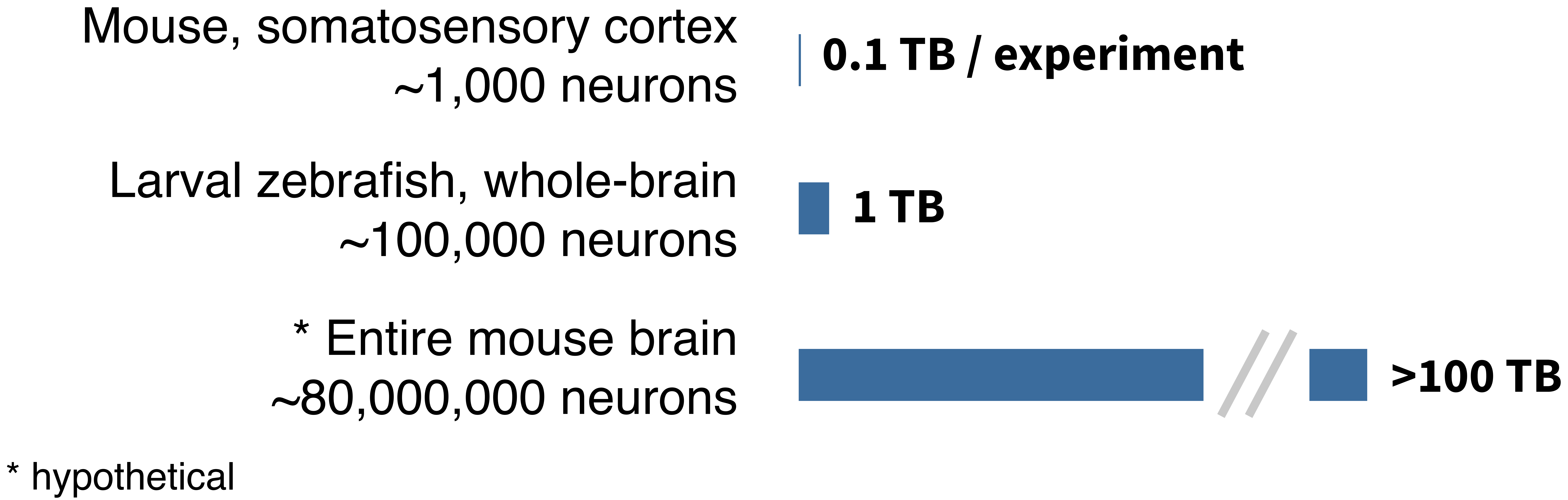
Moser et al., 2008



fine-scale
sensory
tuning

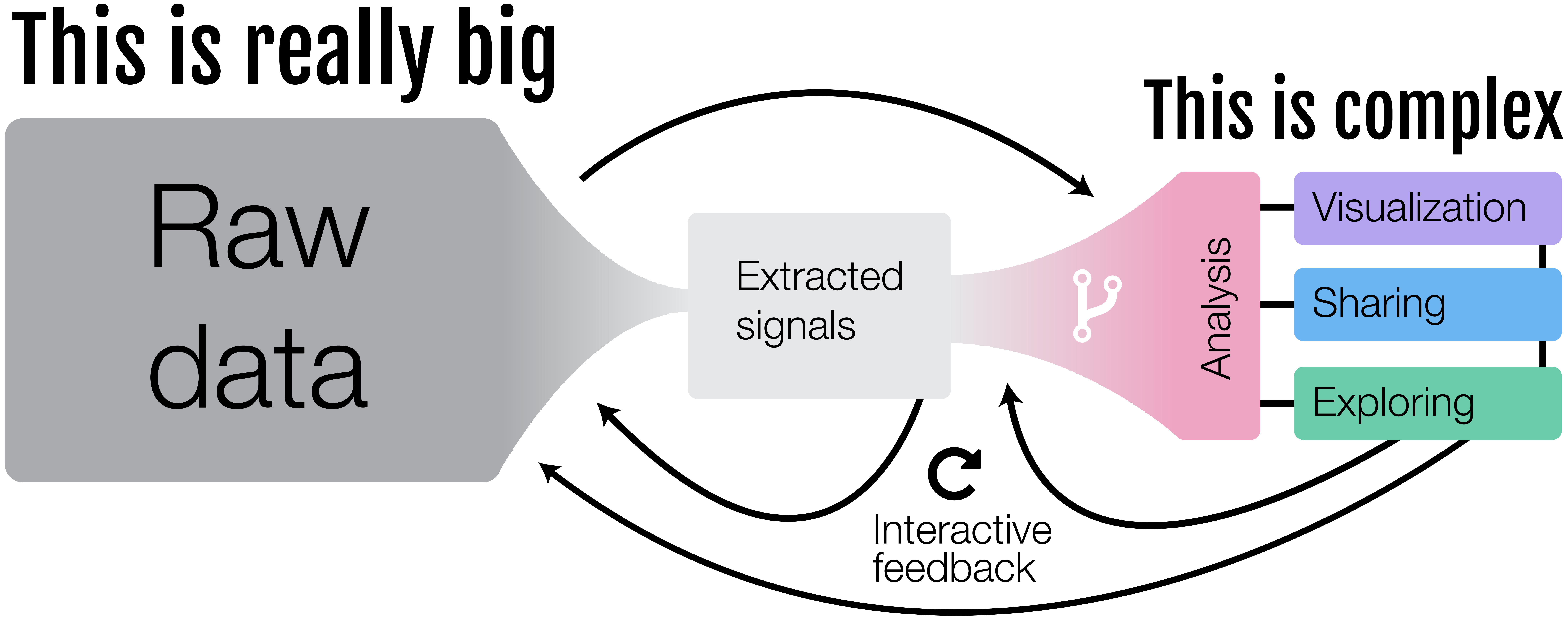
Hubel & Weisel, 1959

Ohki et al., 2006



Mouse, somatosensory cortex
~1,000 neurons
0.1 TB / experiment
Larval zebrafish, whole-brain
~100,000 neurons
1 TB
* Entire mouse brain
~80,000,000 neurons
>100 TB
* hypothetical

Exploratory Data Analysis



This is really big
Raw data
Extracted signals
This is complex
Analysis
Visualization
Sharing
Exploring
Interactive feedback

Supervised methods

$$\mathbf{y} = f(\mathbf{X})$$

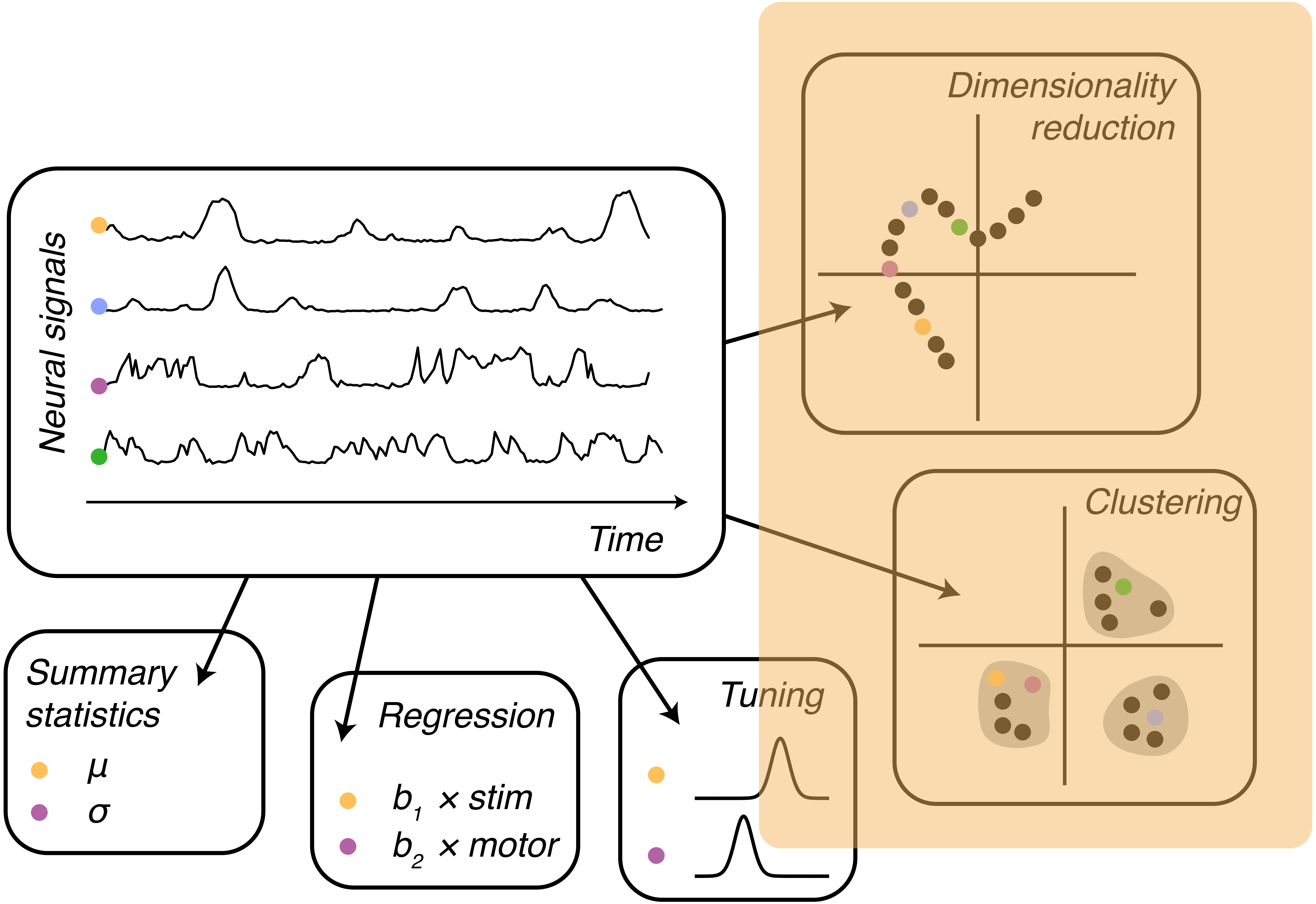
predict
our data

as a function
of other data

Unsupervised methods

$$f(\mathbf{X})$$

find structure
in the data
on its own



Neural signals
Time
Dimensionality reduction
Clustering
Summary statistics
μ
σ
Regression
$b_1 \times$ stim
$b_2 \times$ motor
Tuning

Clustering for preprocessing

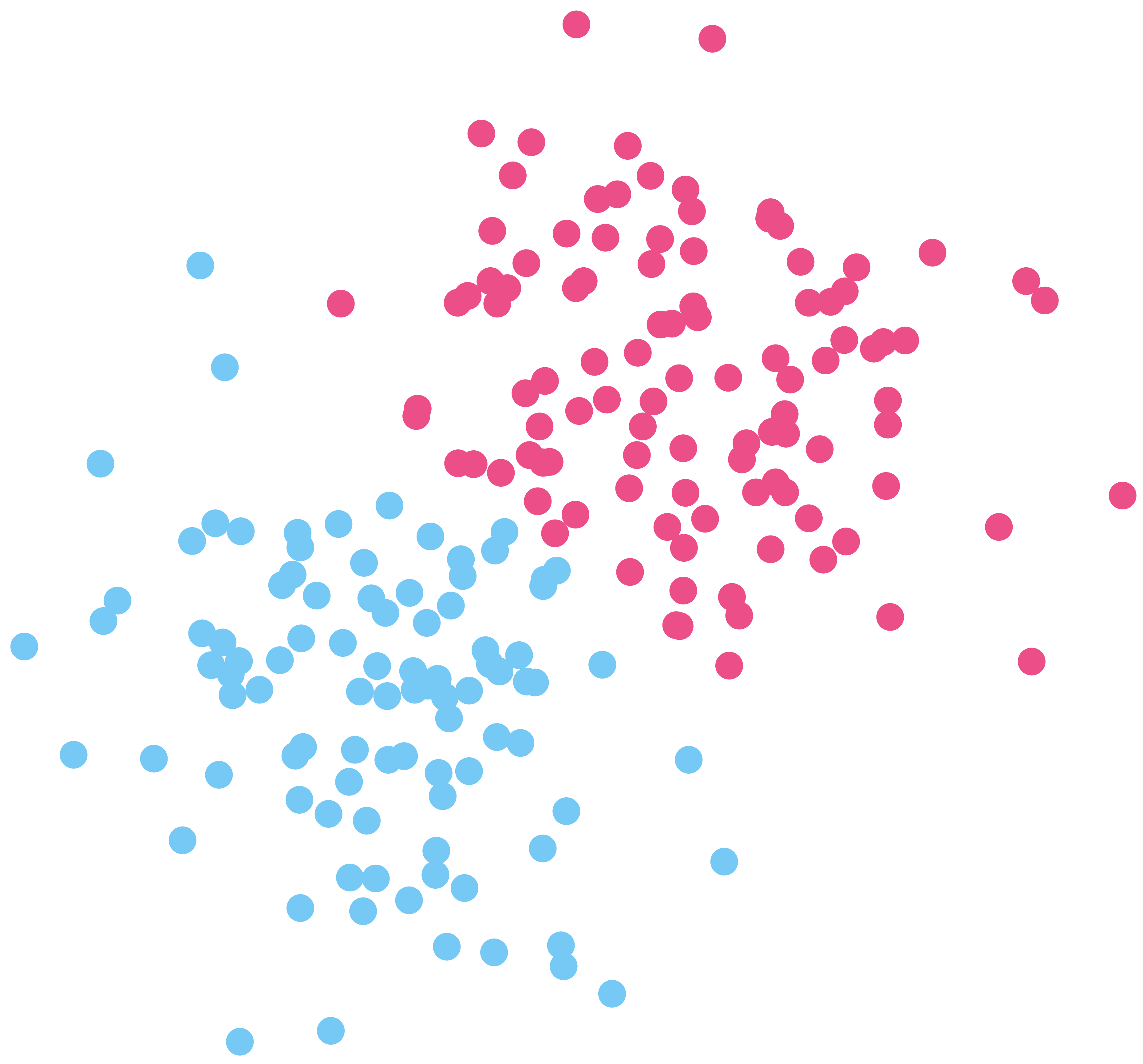
- Raw data is complex and high-dimensional

- Clustering finds collections of inputs that are similar to one another

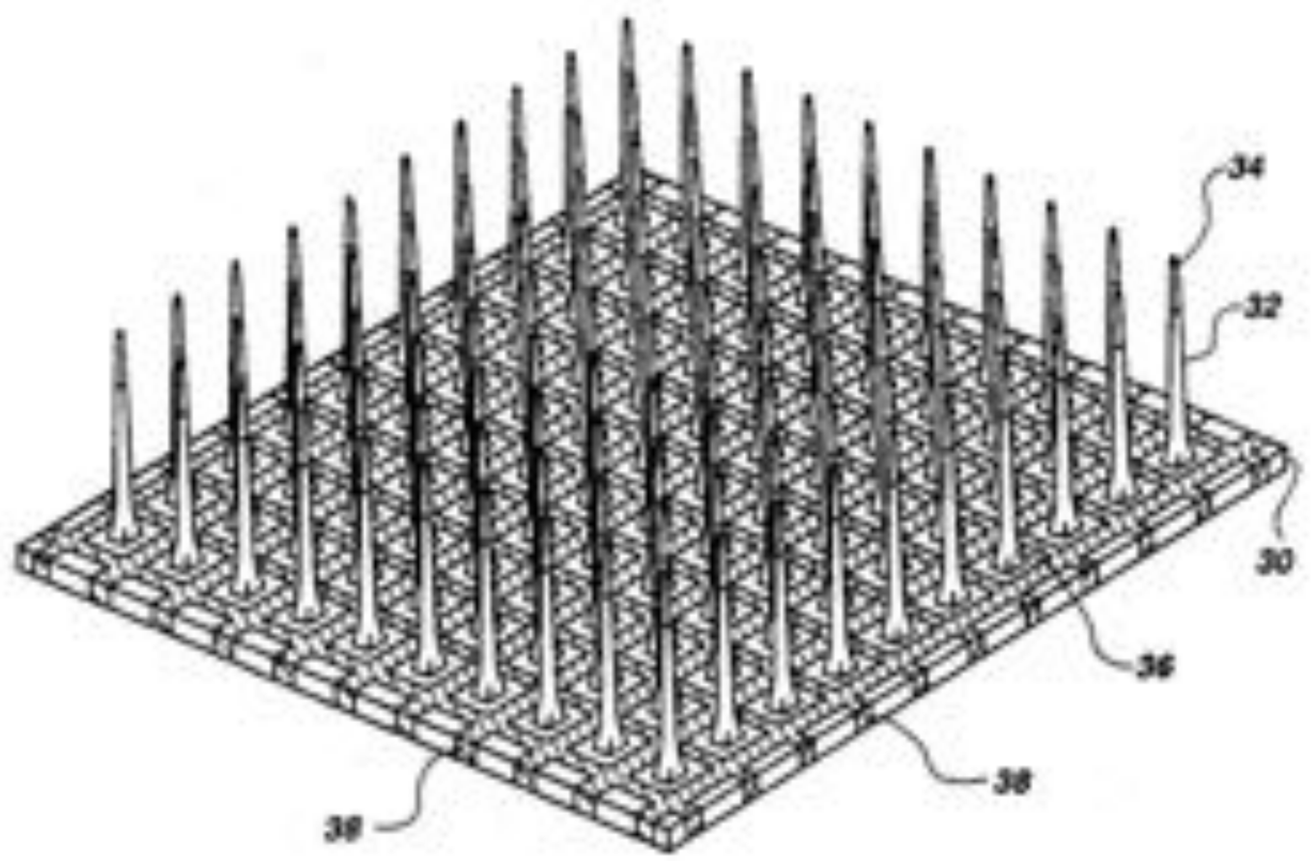
- These groups of clusters may be the more meaningful "unit" of measurement

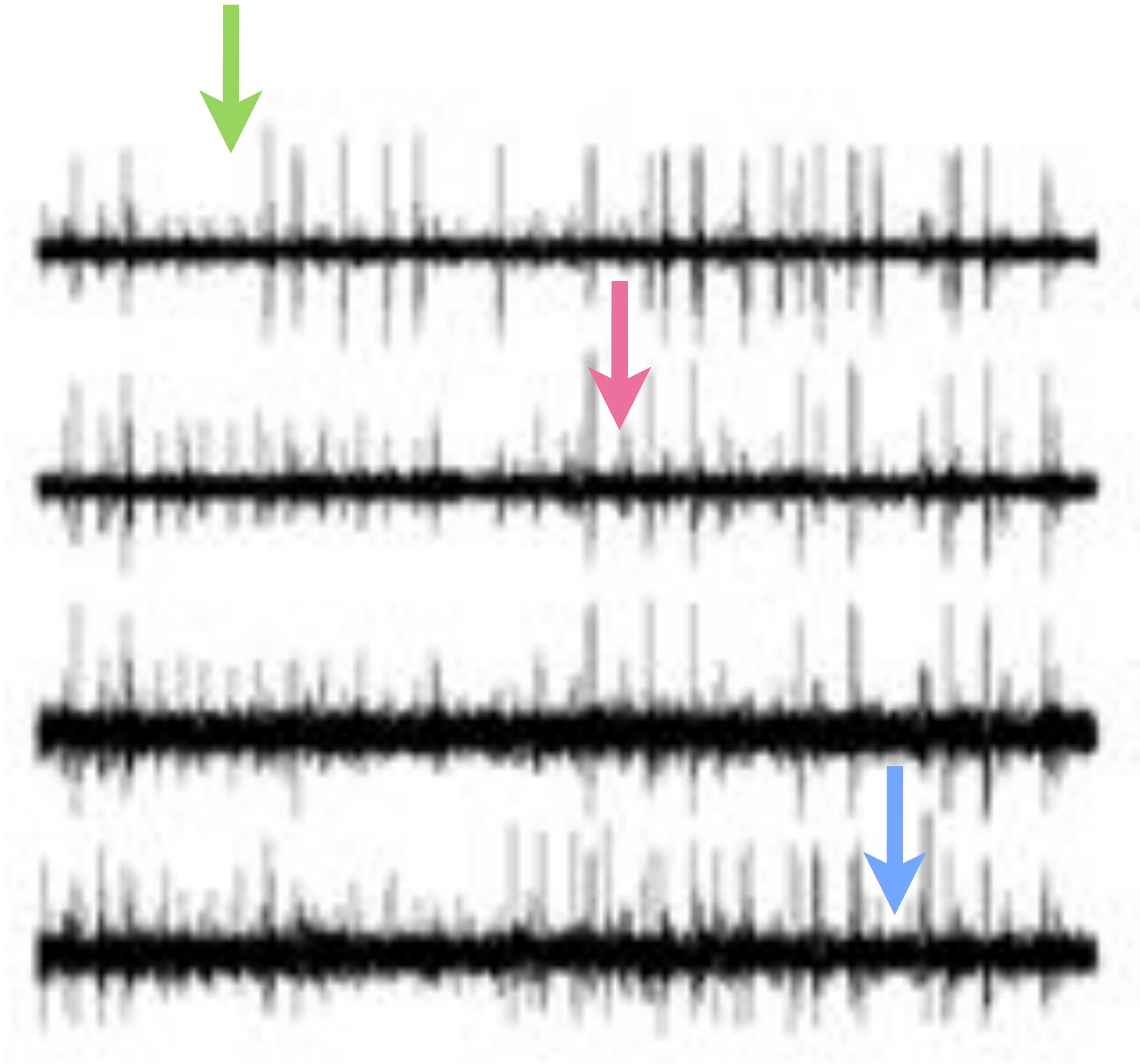
Raw data

Clustered data



Clustering to find waveforms associated with individual neurons based on their traces across multiple electrodes



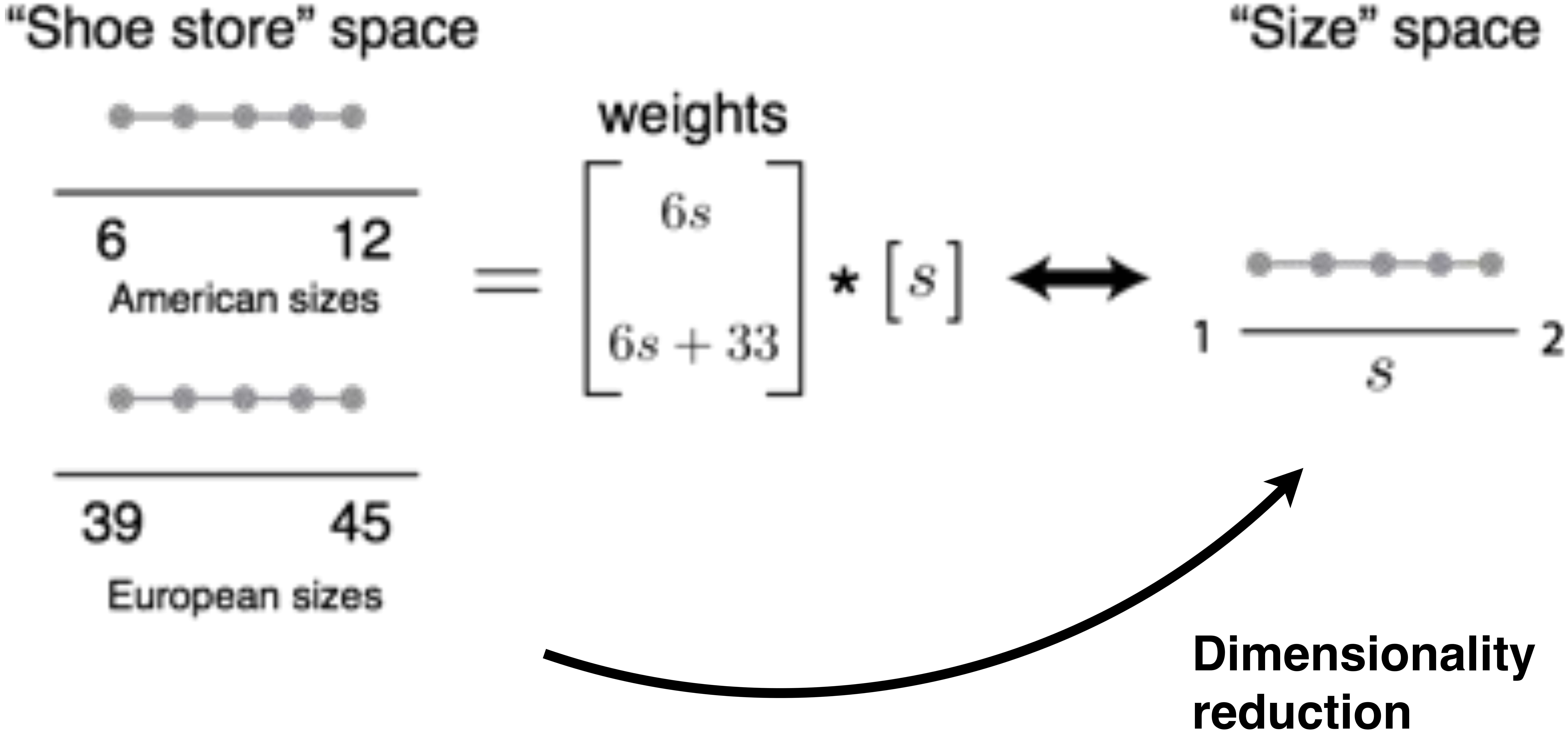


Dimensionality reduction for insight

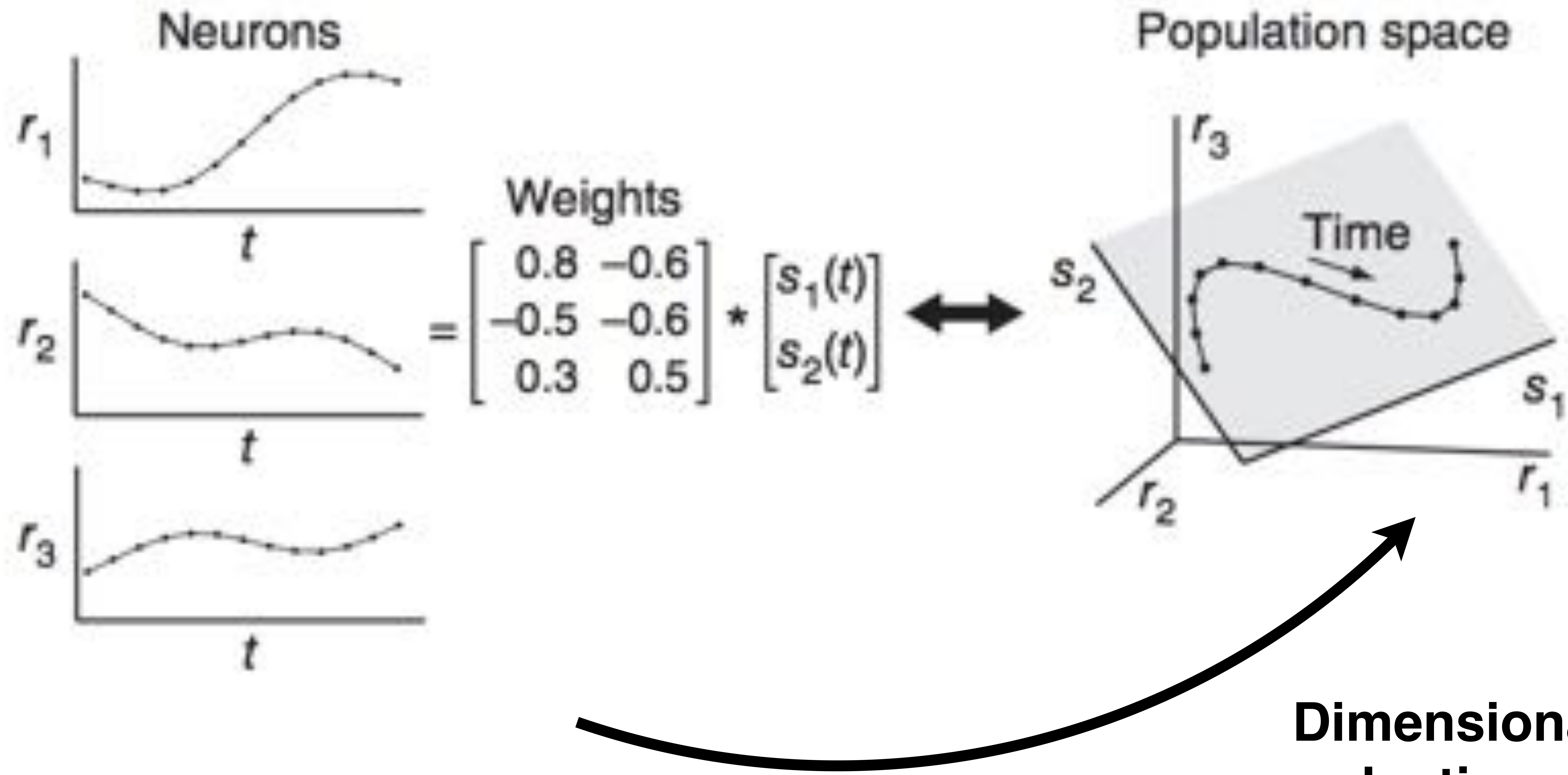
- Raw data is complex and high-dimensional

- Dimensionality reduction describes the data using a simpler, more compact representation

- This representation may make interesting patterns in the data more clear or easier to see

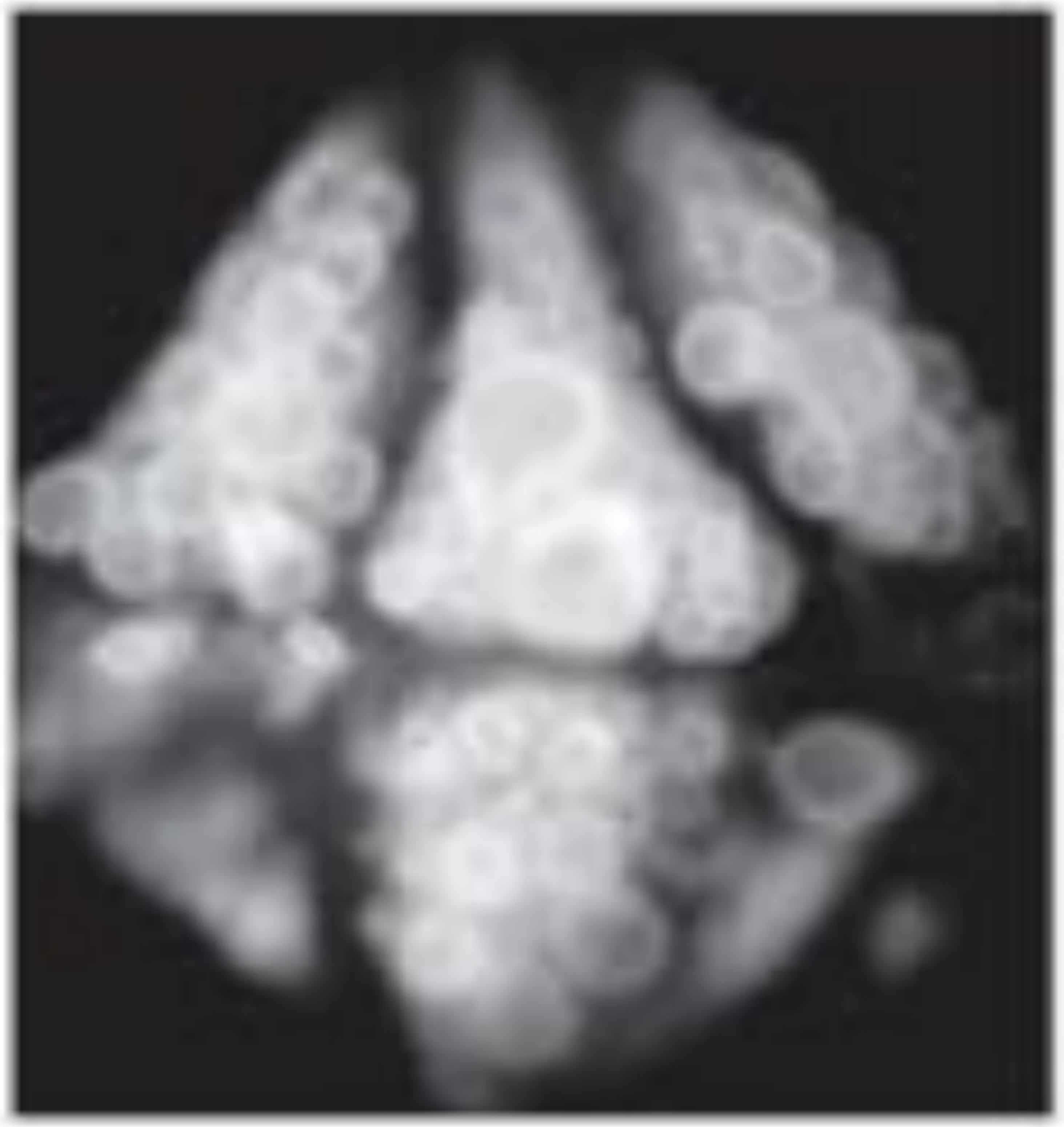


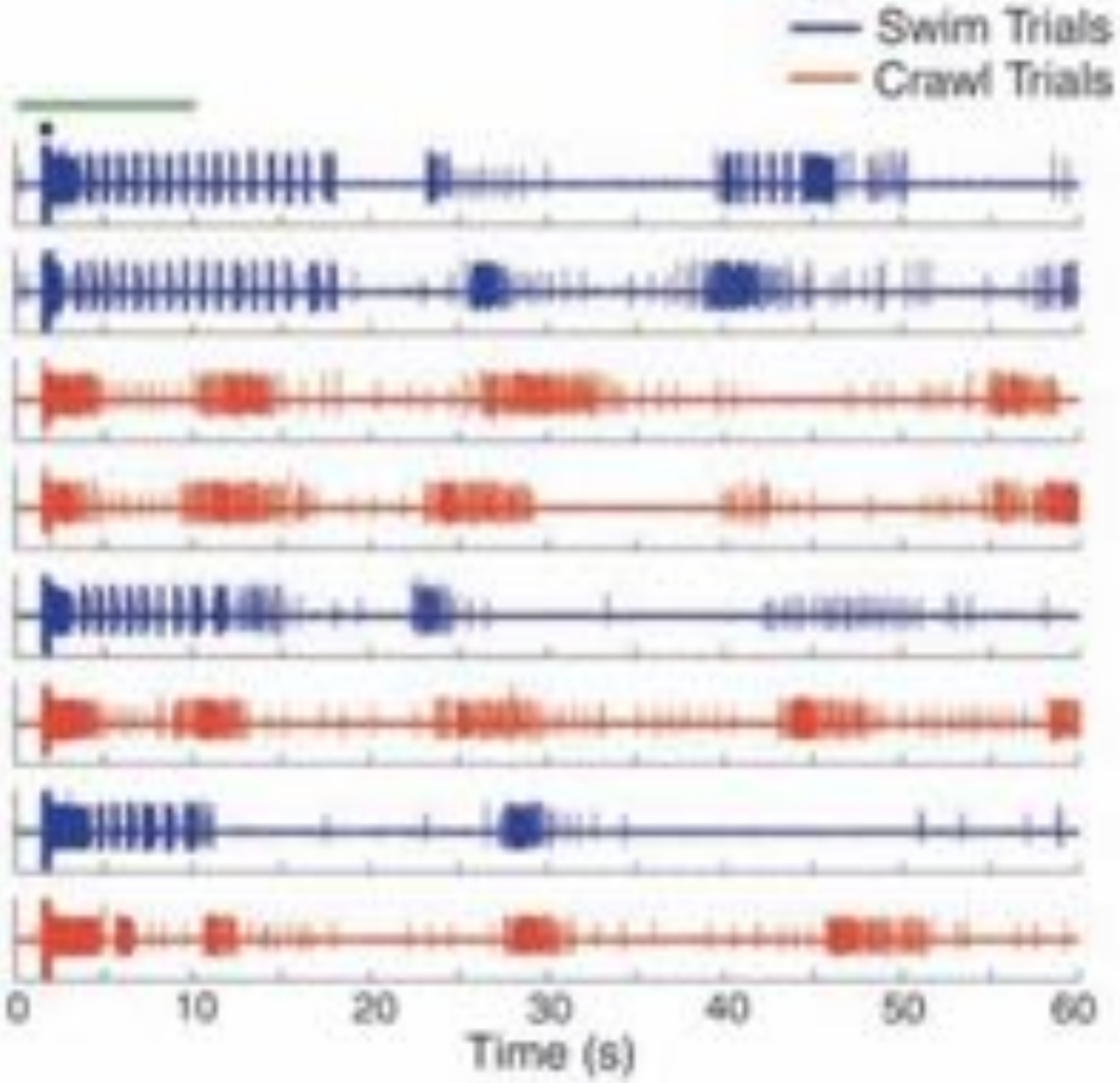
"Shoe store" space
6
12
American sizes
39
45
European sizes
weights
$= \begin{bmatrix} 6s \\ 6s+33 \end{bmatrix} \star [s]$
"Size" space
1
2
s
Dimensionality reduction



Dimensionality reduction

Yu and Cunningham, 2014

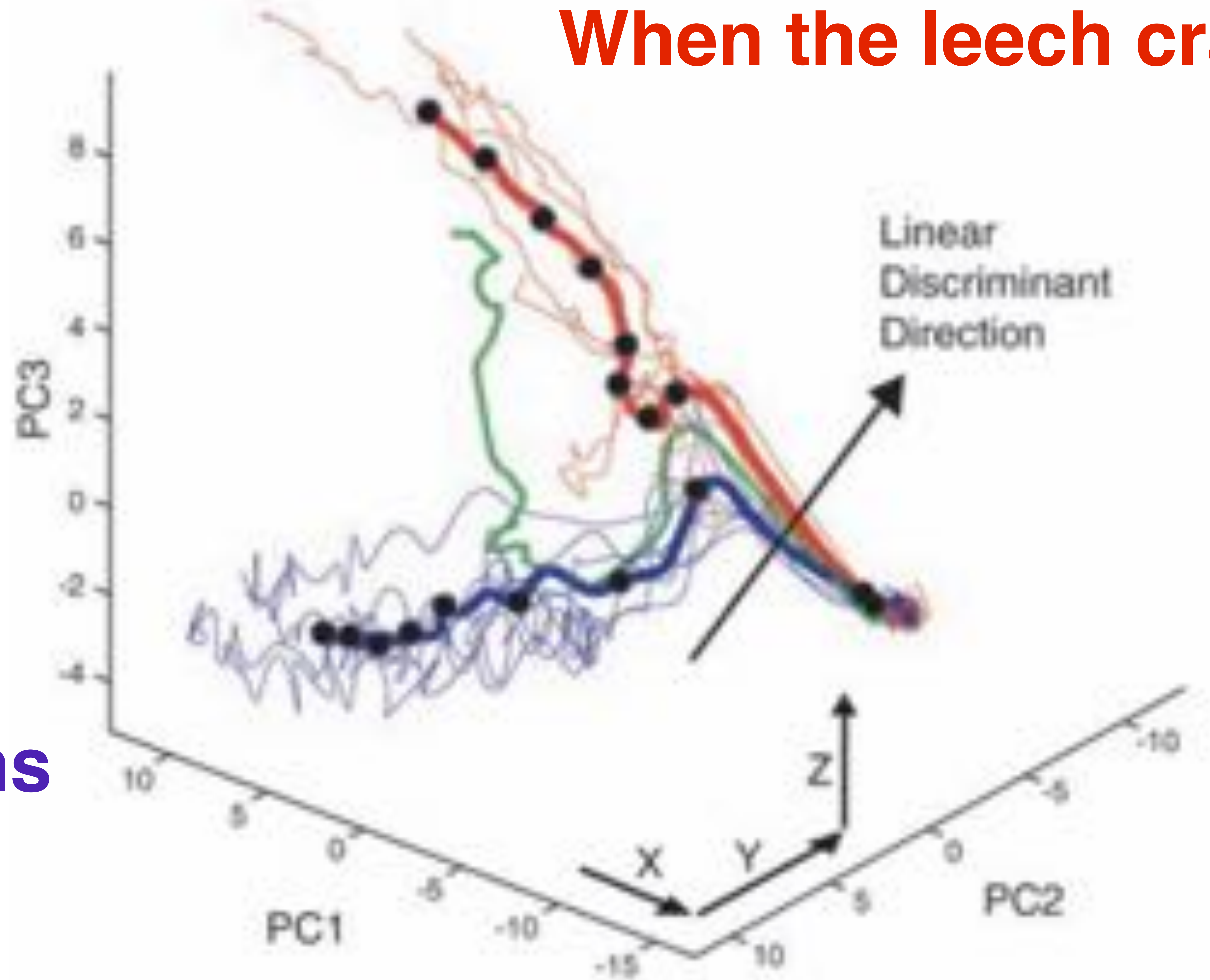




Briggman et al., 2005

When the leech changes its mind!

When the leech crawls



When the leech swims

Briggman et al., 2005

Principal Component Analysis (PCA) Overview

Raw data can be Complex, High-dimensional

To understand a phenomenon we measure various related quantities

If we knew what to measure or how to represent our measurements we might find simple relationships

But in practice we often *measure redundant signals*, e.g., US and European shoe sizes

We also *represent data via the method by which it was gathered*, e.g., pixel representation of brain imaging data

Dimensionality Reduction

Issues

- *Measure redundant signals*
- *Represent data via the method by which it was gathered*

Goal: Find a ‘better’ representation for data

- To visualize and discover hidden patterns
- Preprocessing for supervised task, e.g., feature hashing

How do we define ‘better’?

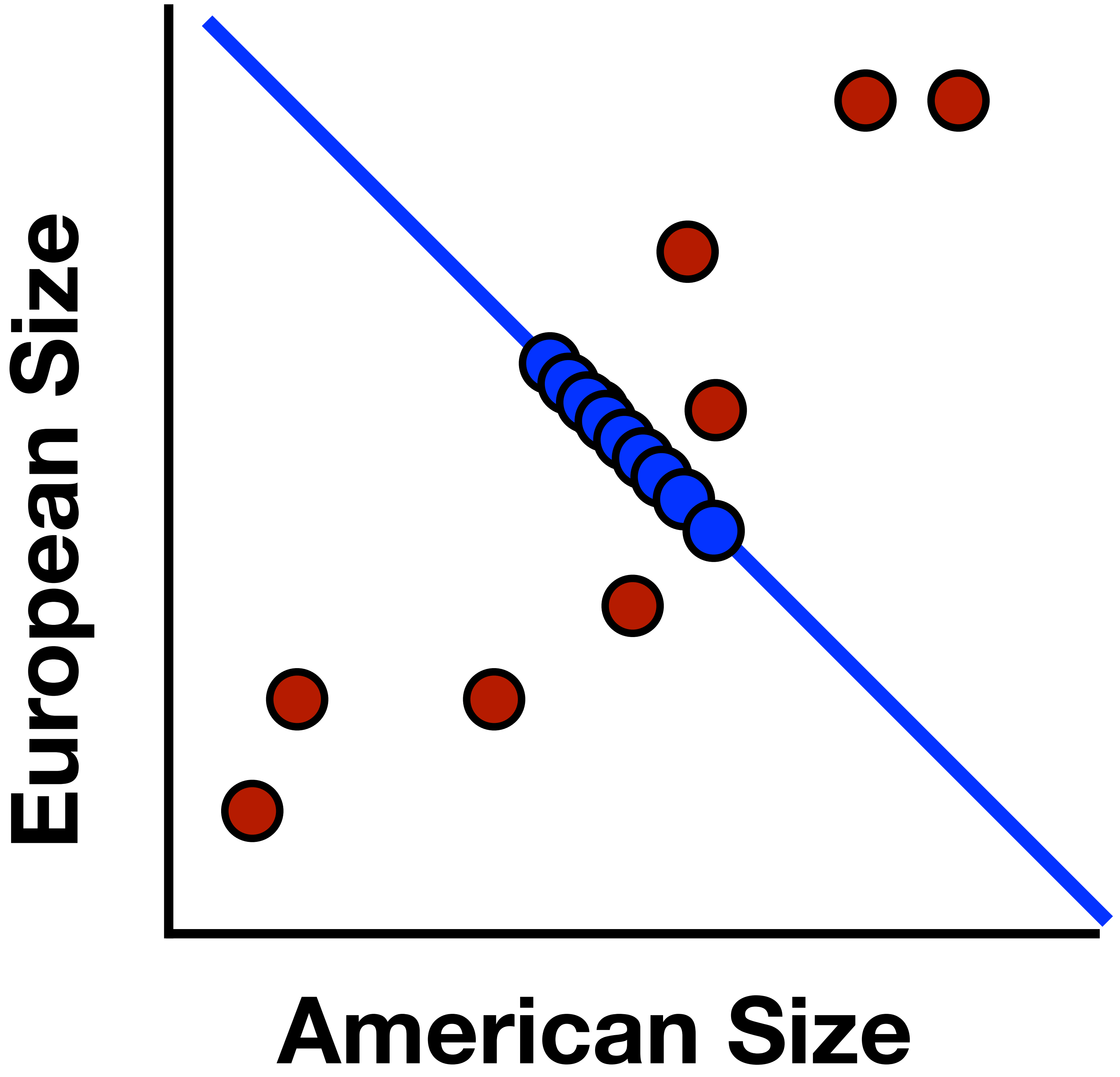
E.g., Shoe Size

We take noisy measurements on European and American scale

- Modulo noise, we expect perfect correlation

How can we do ‘better’, i.e., find a simpler, compact representation?

- Pick a direction and project onto this direction



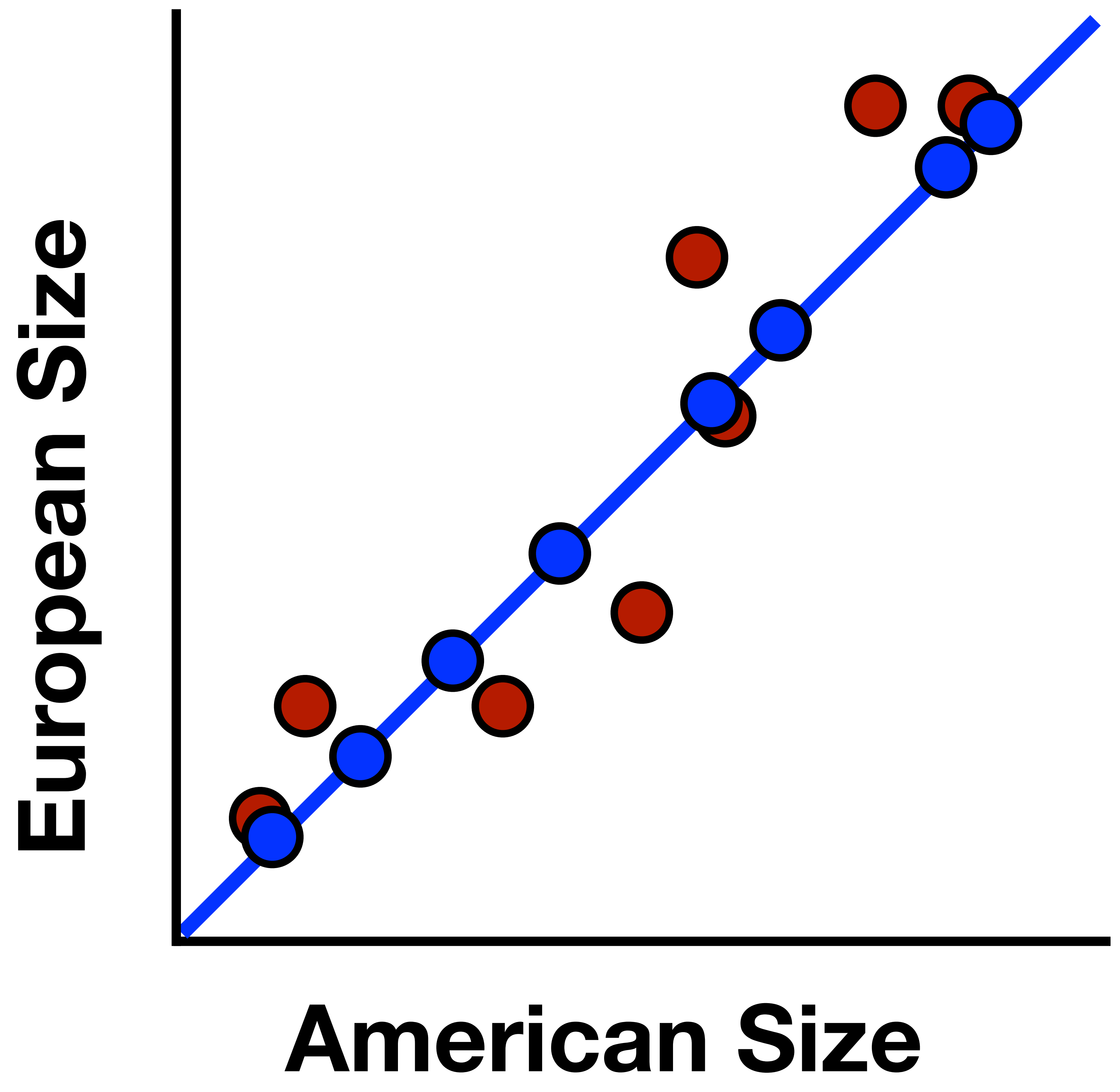
E.g., Shoe Size

We take noisy measurements on European and American scale

- Modulo noise, we expect perfect correlation

How can we do 'better', i.e., find a simpler, compact representation?

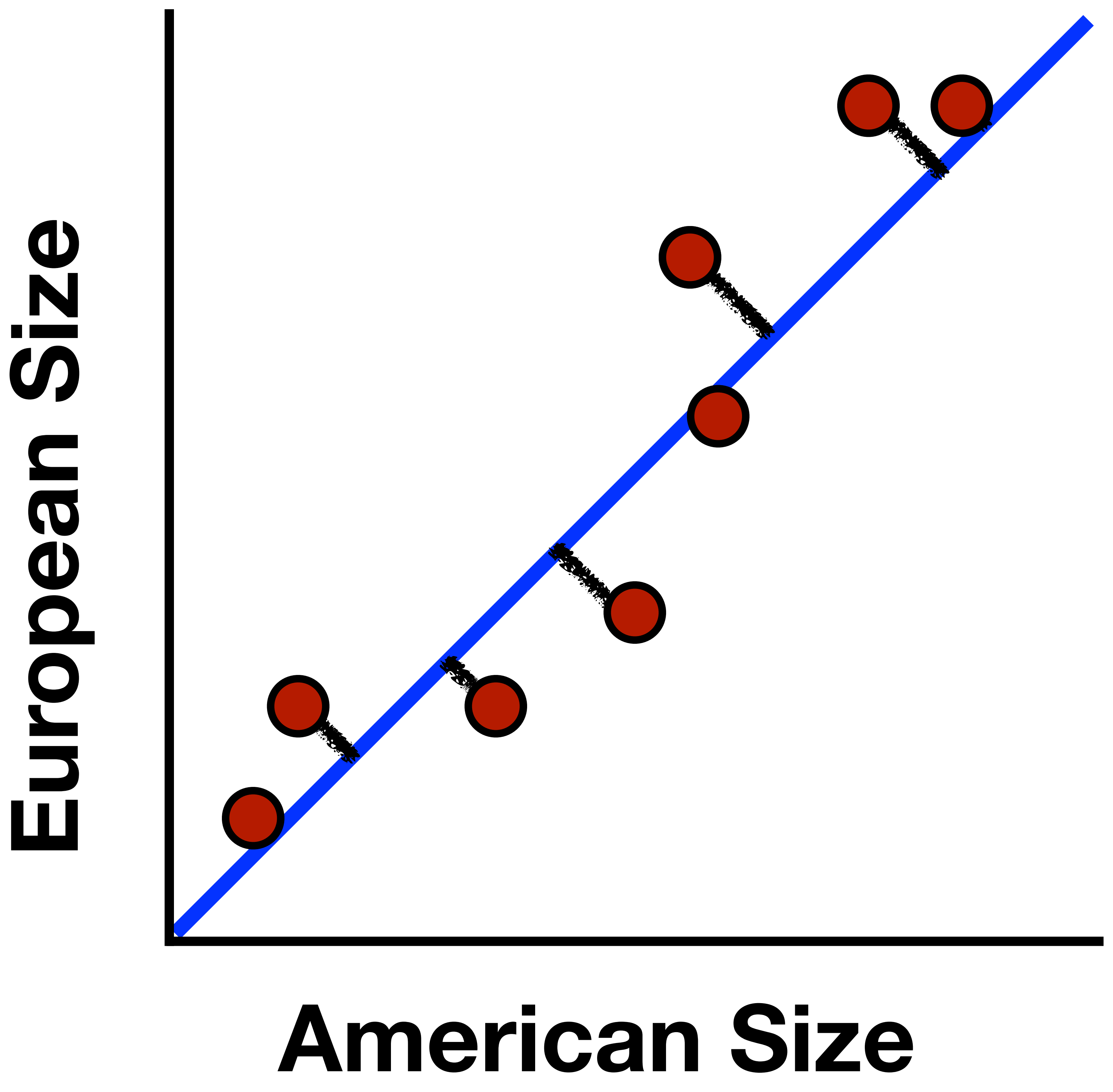
- Pick a direction and project onto this direction



Goal: Minimize Reconstruction Error

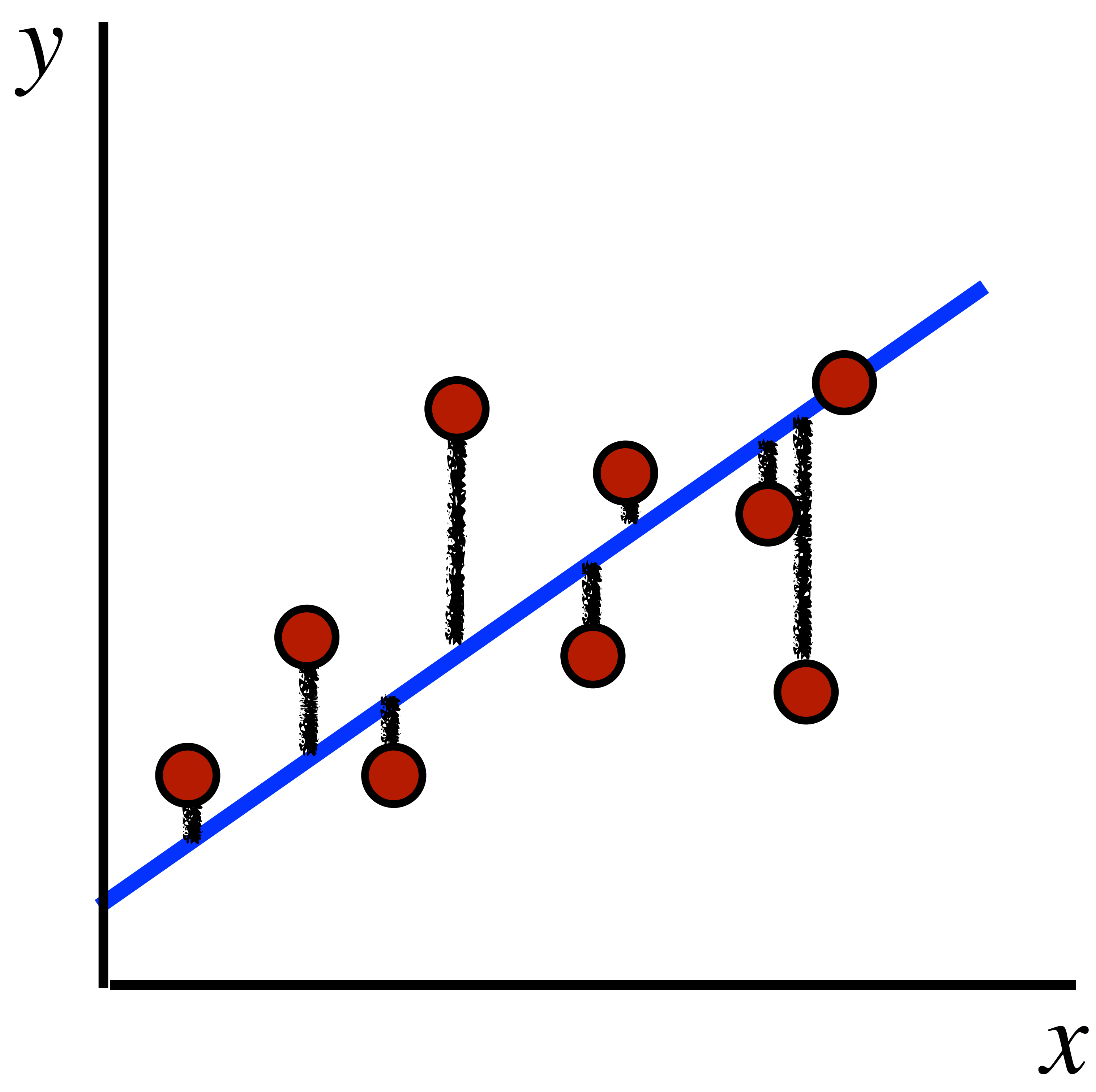
Minimize Euclidean distances between original points and their projections

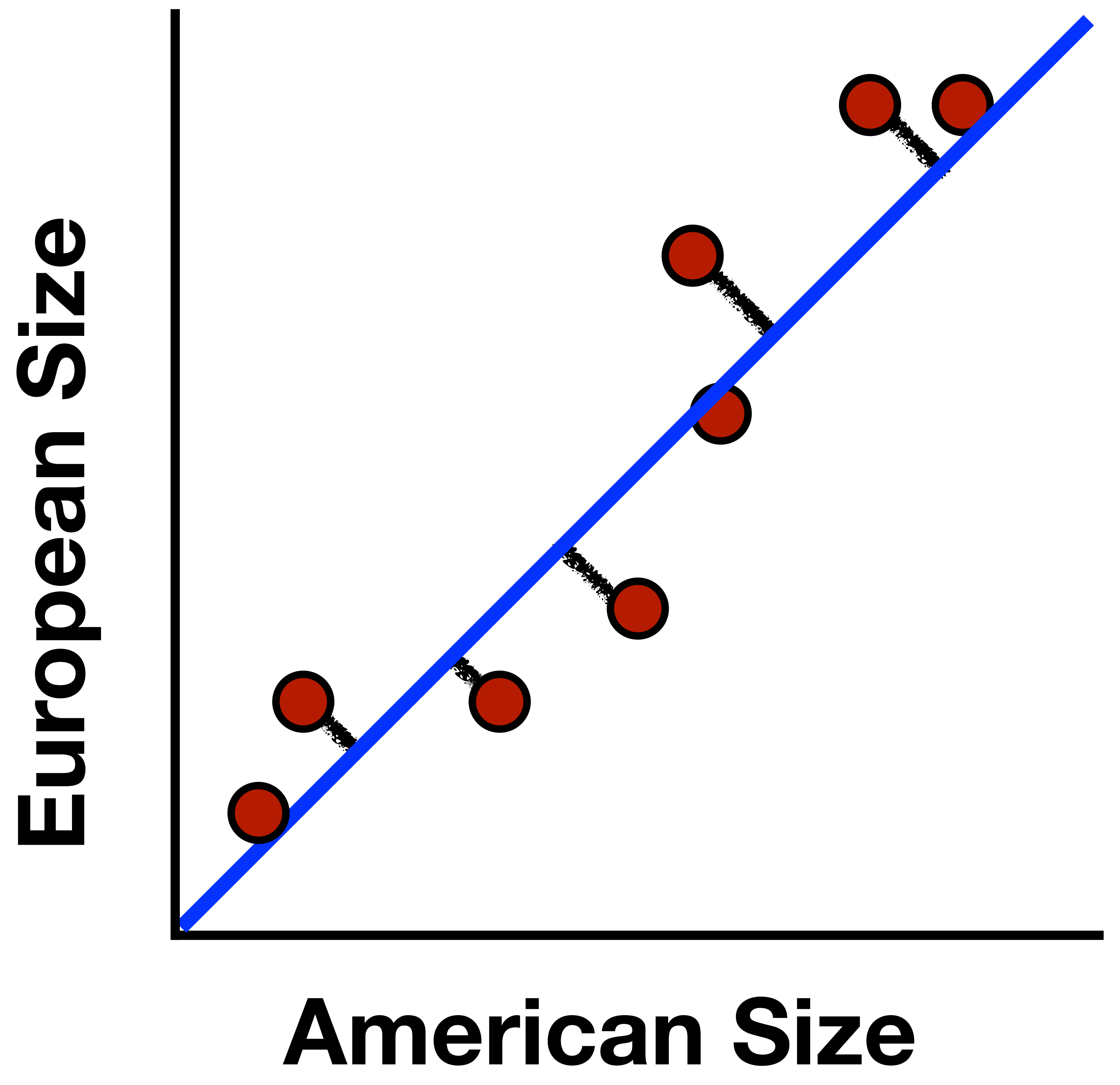
PCA solution solves this problem!



Linear Regression — predict y from x. Evaluate accuracy of predictions (represented by blue line) by **vertical** distances between points and the line

PCA — reconstruct 2D data via 2D data with single degree of freedom. Evaluate reconstructions (represented by blue line) by **Euclidean** distances

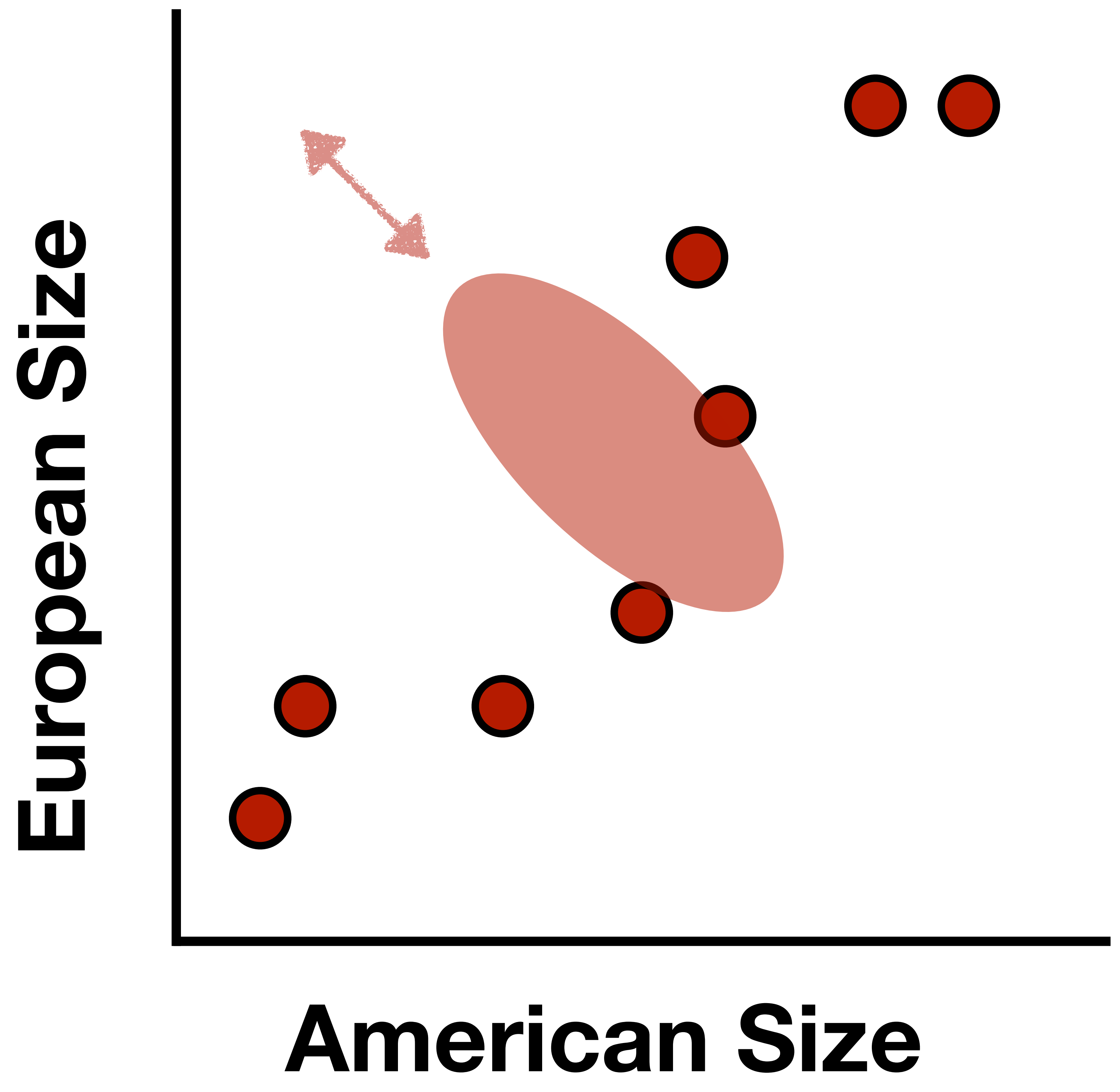




Another Goal: Maximize Variance

To identify patterns we want to study variation across observations

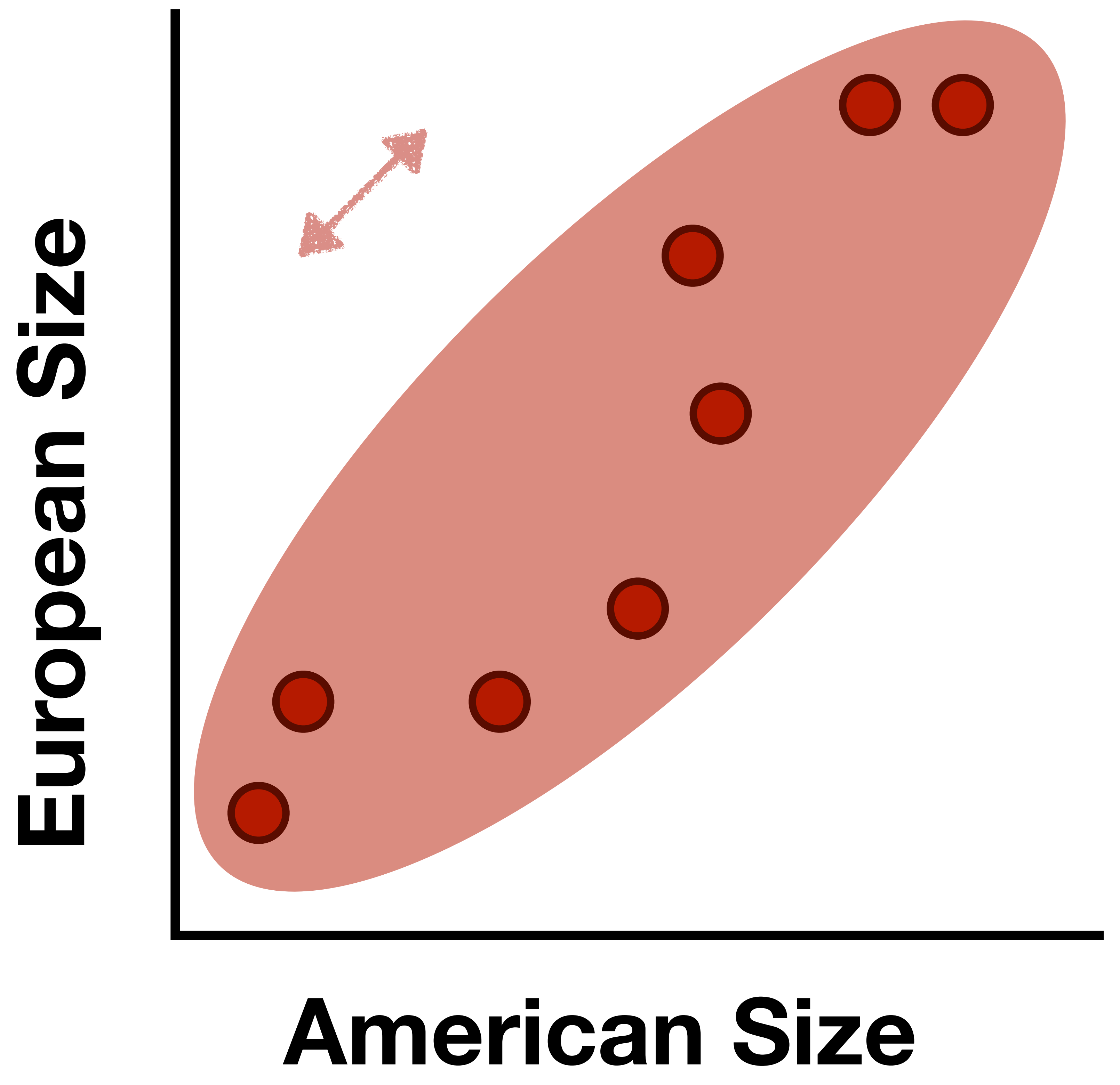
Can we do ‘better’, i.e., find a compact representation that captures variation?



Another Goal: Maximize Variance

To identify patterns we want to study variation across observations

Can we do ‘better’, i.e., find a compact representation that captures variation?

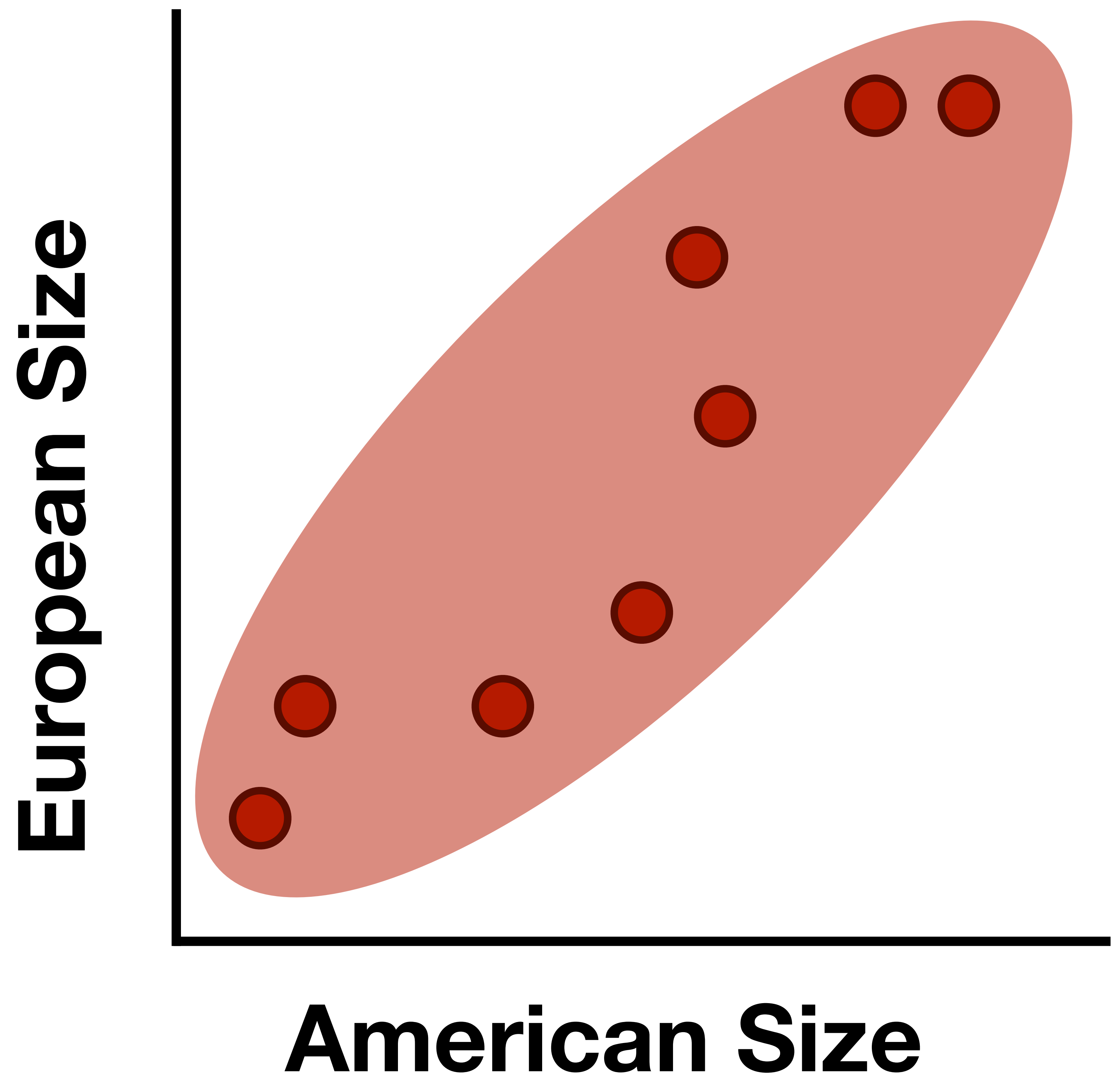


Another Goal: Maximize Variance

To identify patterns we want to study variation across observations

Can we do ‘better’, i.e., find a compact representation that captures variation?

PCA solution finds directions of maximal variance!



PCA Assumptions and Solution

PCA Formulation

PCA: find lower-dimensional representation of raw data

- $\mathbf{X}$ is $n \times d$ (raw data)
- $\mathbf{Z} = \mathbf{X}\mathbf{P}$ is $n \times k$ (reduced representation, PCA 'scores')
- $\mathbf{P}$ is $d \times k$ (columns are k principal components)
- Variance constraints

Linearity assumption ($\mathbf{Z} = \mathbf{X}\mathbf{P}$) simplifies problem

$$\begin{bmatrix} \mathbf{Z} \end{bmatrix} = \begin{bmatrix} \mathbf{X} \end{bmatrix} \begin{bmatrix} \mathbf{P} \end{bmatrix}$$

Given n training points with d features:

- $\mathbf{X} \in \mathbb{R}^{n \times d}$: matrix storing points
- $x_j^{(i)}$: jth feature for ith point
- μ_j : mean of jth feature

Variance of 1st feature $\sigma_1^2 = \frac{1}{n} \sum_{i=1}^{n} \left(x_1^{(i)} - \mu_1 \right)^2$

Variance of 1st feature (assuming zero mean) $\sigma_1^2 = \frac{1}{n} \sum_{i=1}^{n} \left(x_1^{(i)} \right)^2$

Given n training points with d features:

- $\mathbf{X} \in \mathbb{R}^{n \times d}$: matrix storing points
- $x_j^{(i)}$: jth feature for ith point
- μ_j : mean of jth feature

Covariance of 1st and 2nd features (assuming zero mean) $\sigma_{12} = \frac{1}{n} \sum_{i=1}^{n} x_1^{(i)} x_2^{(i)}$

- Symmetric: $\sigma_{12} = \sigma_{21}$
- Zero → uncorrelated
- Large magnitude → (anti) correlated / redundant
- $\sigma_{12} = \sigma_1^2 = \sigma_2^2$ → features are the same

Covariance Matrix

Covariance matrix generalizes this idea for many features

$d \times d$ covariance matrix with zero mean features

$$\mathbf{C}_{\mathbf{X}} = \frac{1}{n}\mathbf{X}^\top \mathbf{X}$$

- ith diagonal entry equals variance of ith feature
- ijth entry is covariance between ith and jth features
- Symmetric (makes sense given definition of covariance)

$$\text{Variance: } \sigma_1^2 = \frac{1}{n}\sum_{i=1}^{n}\left(x_1^{(i)}\right)^2$$

$$\underbrace{\begin{bmatrix} \boxed{2 & -1 & -1} \\ 3 & 2 & -5 \end{bmatrix}}_{\mathbf{X}^\top} \underbrace{\begin{bmatrix} 2 & 3 \\ -1 & 2 \\ -1 & -5 \end{bmatrix}}_{\mathbf{X}} = \underbrace{\begin{bmatrix} 6 & \\ & \end{bmatrix}}_{\mathbf{X}^\top\mathbf{X}}$$

Variance: $\sigma_1^2 = \frac{1}{n}\sum_{i=1}^{n}\left(x_1^{(i)}\right)^2$

Covariance: $\sigma_{12} = \frac{1}{n}\sum_{i=1}^{n} x_1^{(i)} x_2^{(i)}$

$$\underbrace{\begin{bmatrix} \boxed{2 \quad -1 \quad -1} \\ 3 \quad 2 \quad -5 \end{bmatrix}}_{\mathbf{X}^\top} \underbrace{\begin{bmatrix} 2 & \boxed{\begin{matrix} 3 \\ 2 \\ -5 \end{matrix}} \\ -1 & \\ -1 & \end{bmatrix}}_{\mathbf{X}} = \underbrace{\begin{bmatrix} 6 & 9 \\ 9 & 38 \end{bmatrix}}_{\mathbf{X}^\top\mathbf{X}}$$

Dividing by n yields covariance matrix

PCA Formulation

PCA: find lower-dimensional representation of raw data

- $\mathbf{X}$ is $n \times d$ (raw data)
- $\mathbf{Z} = \mathbf{XP}$ is $n \times k$ (reduced representation, PCA 'scores')
- $\mathbf{P}$ is $d \times k$ (columns are k principal components)
- Variance / Covariance constraints

What constraints make sense in reduced representation?

- No feature correlation, i.e., all off-diagonals in $\mathbf{C_Z}$ are zero
- Rank-ordered features by variance, i.e., sorted diagonals of $\mathbf{C_Z}$

PCA Formulation

PCA: find lower-dimensional representation of raw data

- $\mathbf{X}$ is $n \times d$ (raw data)
- $\mathbf{Z} = \mathbf{XP}$ is $n \times k$ (reduced representation, PCA 'scores')
- $\mathbf{P}$ is $d \times k$ (columns are k principal components)
- Variance / Covariance constraints

$$\begin{bmatrix} \mathbf{Z} \end{bmatrix} = \begin{bmatrix} \mathbf{X} \end{bmatrix} \begin{bmatrix} \mathbf{P} \end{bmatrix}$$

$\mathbf{P}$ equals the top k eigenvectors of $\mathbf{C_X}$

PCA Solution

All covariance matrices have an eigendecomposition

- $\mathbf{C_X} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ (eigendecomposition)
- $\mathbf{U}$ is $d \times d$ (column are eigenvectors, sorted by their eigenvalues)
- $\mathbf{\Lambda}$ is $d \times d$ (diagonals are eigenvalues, off-diagonals are zero)

The d eigenvectors are orthonormal directions of max variance

- Associated eigenvalues equal variance in these directions
- 1st eigenvector is direction of max variance (variance is λ_1)

In lab, we'll use the **`eigh`** function from **`numpy.linalg`**

Choosing k

How should we pick the dimension of the new representation?

Visualization: Pick top 2 or 3 dimensions for plotting purposes

Other analyses: Capture 'most' of the variance in the data

- Recall that eigenvalues are variances in the directions specified by eigenvectors, and that eigenvalues are sorted
- Fraction of retained variance: $\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$

Can choose k such that we retain some fraction of the variance, e.g., 95%

Other Practical Tips

PCA assumptions (linearity, orthogonality) not always appropriate

- Various extensions to PCA with different underlying assumptions, e.g., manifold learning, Kernel PCA, ICA

Centering is crucial, i.e., we must preprocess data so that all features have zero mean before applying PCA

PCA results dependent on scaling of data

- Data is sometimes rescaled in practice before applying PCA

PCA Algorithm

Orthogonal and Orthonormal Vectors

Orthogonal vectors are **perpendicular** to each other

- Equivalently, their dot product equals zero
- $\mathbf{a}^\top \mathbf{b} = 0$ and $\mathbf{d}^\top \mathbf{b} = 0$, but $\mathbf{c}$ isn't orthogonal to others

$\mathbf{a} = [1 \quad 0]^\top$ $\mathbf{b} = [0 \quad 1]^\top$ $\mathbf{c} = [1 \quad 1]^\top$ $\mathbf{d} = [2 \quad 0]^\top$

Orthonormal vectors are orthogonal and have unit norm

- $\mathbf{a}$ are $\mathbf{b}$ are orthonormal, but $\mathbf{b}$ are $\mathbf{d}$ are not orthonormal

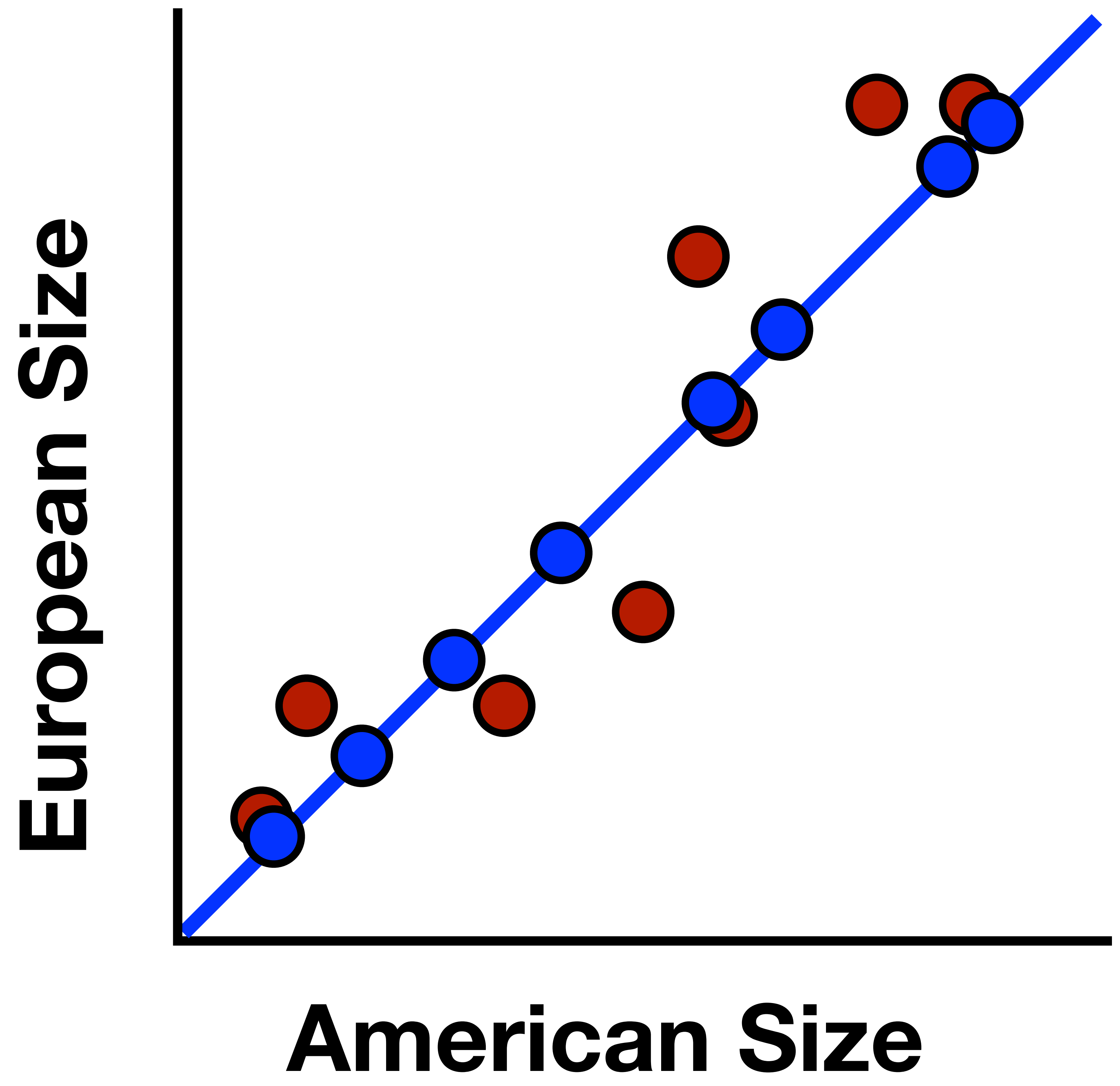
PCA Iterative Algorithm

$k = 1$: Find direction of max variance, project onto this direction

- Locations along this direction are the new 1D representation

More generally, for i in $\{1, \ldots, k\}$:

- Find direction of max variance that is *orthonormal* to previously selected directions, project onto this direction
- Locations along this direction are the ith feature in new representation



PCA Derivation (Optional)

Eigendecomposition

All covariance matrices have an eigendecomposition

- $\mathbf{C_X} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ (eigendecomposition)
- $\mathbf{U}$ is $d \times d$ (column are eigenvectors, sorted by their eigenvalues)
- $\mathbf{\Lambda}$ is $d \times d$ (diagonals are eigenvalues, off-diagonals are zero)

Eigenvector / Eigenvalue equation: $\mathbf{C_x u} = \lambda \mathbf{u}$

- By definition $\mathbf{u}^\top \mathbf{u} = 1$ (unit norm)
- Example: $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \Rightarrow$ eigenvector: $\mathbf{u} = \begin{bmatrix} 1 & 0 \end{bmatrix}^\top$ eigenvalue: $\lambda = 1$

PCA Formulation

PCA: find lower-dimensional representation of raw data

- $\mathbf{X}$ is $n \times d$ (raw data)
- $\mathbf{Z} = \mathbf{XP}$ is $n \times k$ (reduced representation, PCA ‘scores’)
- $\mathbf{P}$ is $d \times k$ (columns are k principal components)
- Variance / Covariance constraints

$$\begin{bmatrix} \mathbf{Z} \end{bmatrix} = \begin{bmatrix} \mathbf{X} \end{bmatrix} \begin{bmatrix} \mathbf{P} \end{bmatrix}$$

PCA Formulation, $k = 1$

PCA: find one-dimensional representation of raw data

- $\mathbf{X}$ is $n \times d$ (raw data)
- $\mathbf{z} = \mathbf{X}\mathbf{p}$ is $n \times 1$ (reduced representation, PCA 'scores')
- $\mathbf{p}$ is $d \times 1$ (columns are k principal components)
- Variance constraint

$$\sigma_{\mathbf{z}}^2 = \frac{1}{n} \sum_{i=1}^{n} \left(z^{(i)} \right)^2 = ||\mathbf{z}||_2^2$$

Goal: Maximizes variance, i.e., $\max_{\mathbf{p}} \sigma_{\mathbf{z}}^2$ where $||\mathbf{p}||_2 = 1$

Goal: Maximizes variance, i.e., $\max_{\mathbf{p}} \sigma_{\mathbf{z}}^2$ where $||\mathbf{p}||_2 = 1$

	$\sigma_{\mathbf{z}}^2 = ||\mathbf{z}||_2^2$
Relationship between Euclidean distance and dot product	$= \mathbf{z}^\top \mathbf{z}$
Definition: $\mathbf{z} = \mathbf{X}\mathbf{p}$	$= (\mathbf{X}\mathbf{p})^\top (\mathbf{X}\mathbf{p})$
Transpose property: $(\mathbf{X}\mathbf{p})^\top = \mathbf{p}^\top \mathbf{X}^\top$; associativity of multiply	$= \mathbf{p}^\top \mathbf{X}^\top \mathbf{X}\mathbf{p}$
Definition: $\mathbf{C}_{\mathbf{X}} = \frac{1}{n}\mathbf{X}^\top \mathbf{X}$	$= \boxed{n}\mathbf{p}^\top \mathbf{C}_{\mathbf{x}} \mathbf{p}$

Restated Goal: $\max_{\mathbf{p}} \mathbf{p}^\top \mathbf{C}_{\mathbf{x}} \mathbf{p}$ where $||\mathbf{p}||_2 = 1$

Connection to Eigenvectors

Recall eigenvector / eigenvalue equation: $\mathbf{C_x u} = \lambda \mathbf{u}$

- By definition $\mathbf{u}^\top \mathbf{u} = 1$, and thus $\mathbf{u}^\top \mathbf{C_x u} = \lambda$
- But this is the expression we're optimizing, and thus maximal variance achieved when $\mathbf{p}$ is top eigenvector of $\mathbf{C_X}$

Similar arguments can be used for $k > 1$

Restated Goal: $\max_{\mathbf{p}} \mathbf{p}^\top \mathbf{C_x p}$ where $||\mathbf{p}||_2 = 1$

Distributed PCA

Computing PCA Solution

Given: $n \times d$ matrix of uncentered raw data

Goal: Compute $k \ll d$ dimensional representation

Step 1: Center Data

Step 2: Compute Covariance or Scatter Matrix

- $\frac{1}{n}\mathbf{X}^\top\mathbf{X}$ versus $\mathbf{X}^\top\mathbf{X}$

Step 3: Eigendecomposition

Step 4: Compute PCA Scores

$$\begin{bmatrix} \mathbf{Z} \end{bmatrix} = \begin{bmatrix} \mathbf{X} \end{bmatrix} \begin{bmatrix} \mathbf{P} \end{bmatrix}$$

PCA at Scale

Case 1: Big n and Small d

- O(d^2) local storage, O(d^3) local computation, O(dk) communication
- Similar strategy as closed-form linear regression

Case 2: Big n and Big d

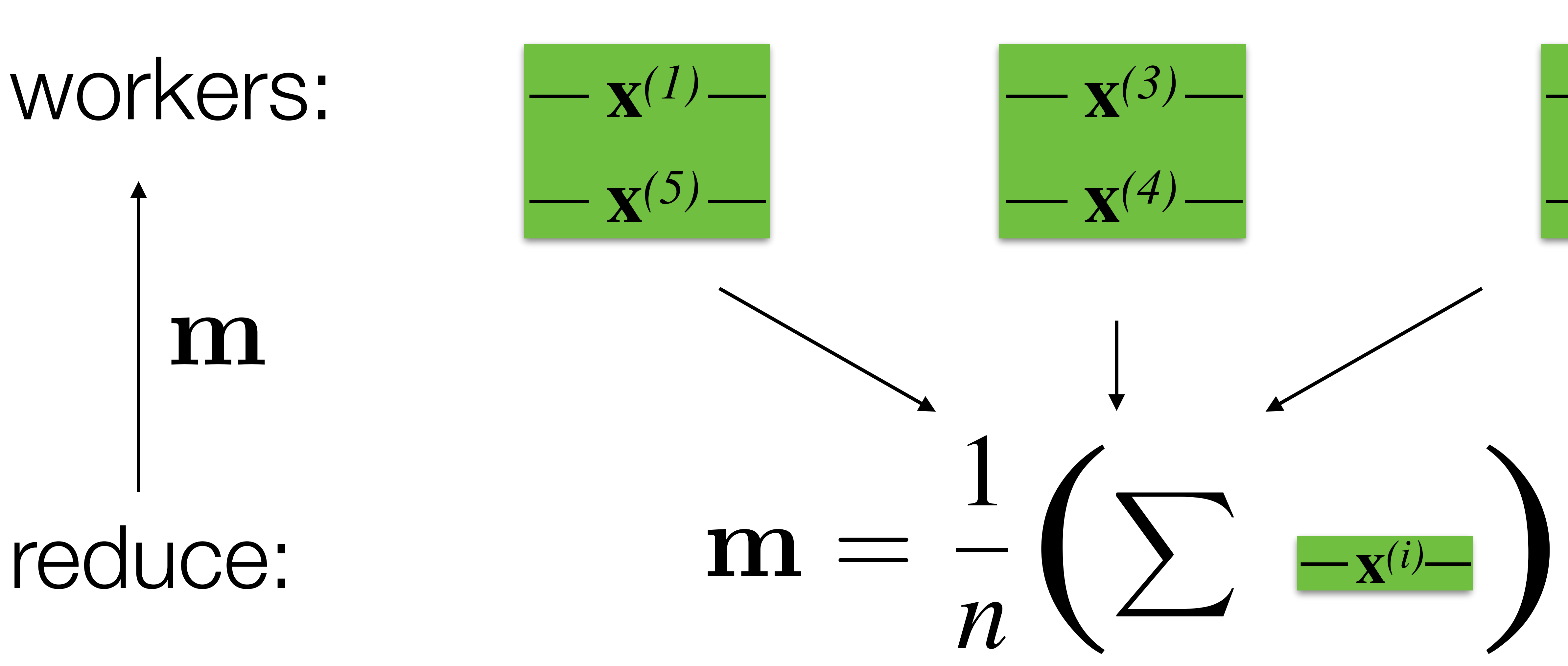
- O($dk + n$) local storage, computation; O($dk + n$) communication
- Iterative algorithm

$$\mathbf{Z} = \mathbf{X}\mathbf{P}$$

Step 1: Center Data

- Compute d feature means, $\mathbf{m} \in \mathbb{R}^d$
- Communicate $\mathbf{m}$ to all workers

Example: $n = 6$; 3 workers



O(nd) Distributed Storage

O(d) Local Storage
O(d) Local Computation
O(d) Communication

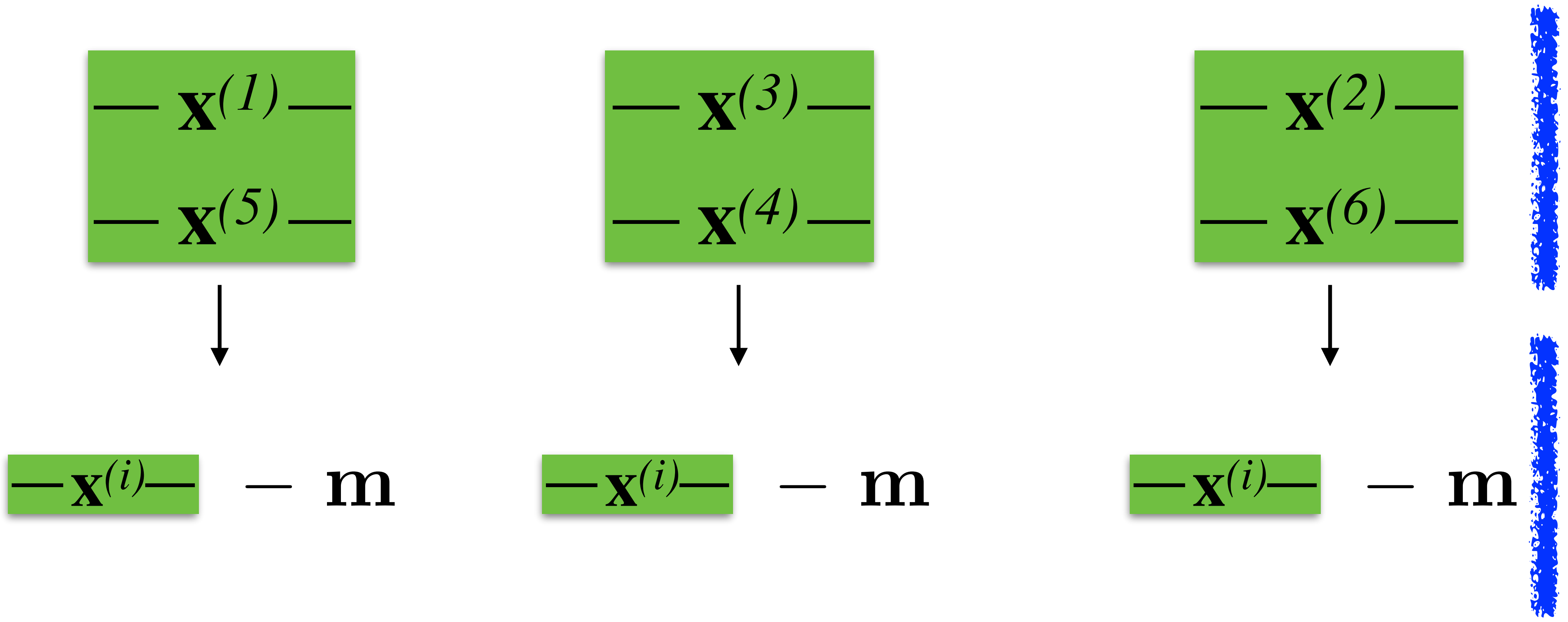
Step 1: Center Data

- Compute d feature means, $\mathbf{m} \in \mathbb{R}^d$
- Communicate $\mathbf{m}$ to all workers
- Subtract $\mathbf{m}$ from each data point

Example: $n = 6$; 3 workers

workers:

map:



O(nd) Distributed Storage

O(d) Local Computation

Step 2: Compute Scatter Matrix ($\mathbf{X}^\top \mathbf{X}$)

- Compute matrix product via outer products (just like we did for closed-form linear regression!)

$$\begin{bmatrix} \boxed{\begin{matrix} 9 \\ 4 \end{matrix}} & \begin{matrix} 3 \\ 1 \end{matrix} & \begin{matrix} 5 \\ 2 \end{matrix} \end{bmatrix} \begin{bmatrix} \boxed{1 \quad 2} \\ 3 \quad -5 \\ 2 \quad 3 \end{bmatrix} = \begin{bmatrix} \quad & \quad \end{bmatrix}$$

$$\begin{bmatrix} 9 & 18 \\ 4 & 8 \end{bmatrix}$$

Step 2: Compute Scatter Matrix ($\mathbf{X}^\top \mathbf{X}$)

- Compute matrix product via outer products (just like we did for closed-form linear regression!)

$$\begin{bmatrix} 9 & 3 & 5 \\ 4 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & -5 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} \quad & \quad \end{bmatrix}$$

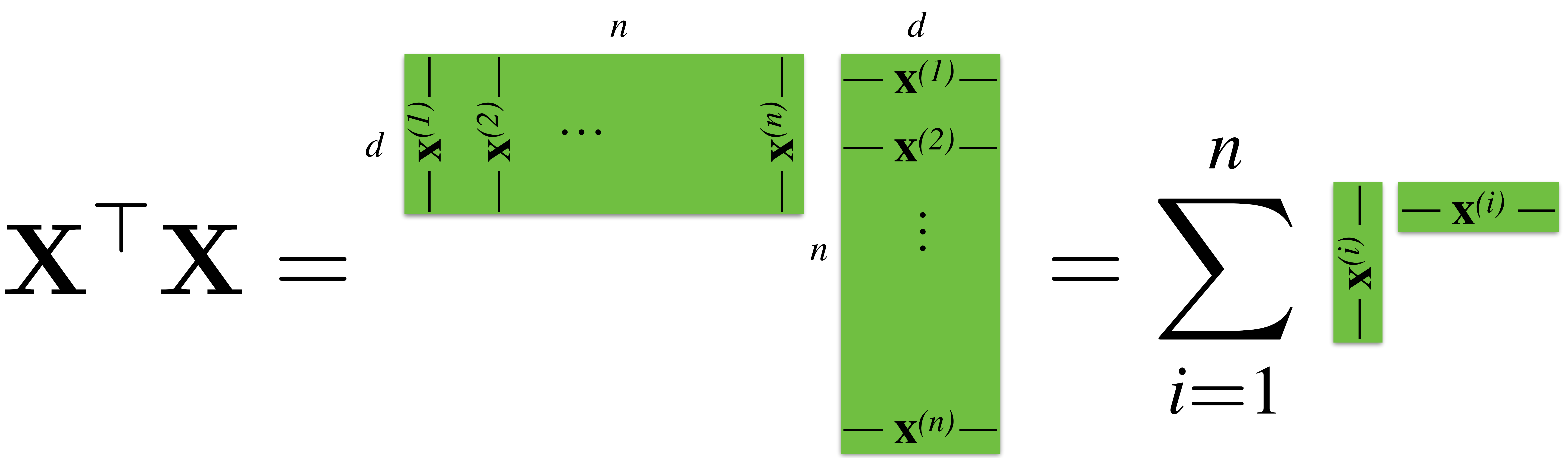
$$\begin{bmatrix} 9 & 18 \\ 4 & 8 \end{bmatrix} + \begin{bmatrix} 9 & -15 \\ 3 & -5 \end{bmatrix}$$

Step 2: Compute Scatter Matrix ($\mathbf{X}^\top \mathbf{X}$)

- Compute matrix product via outer products (just like we did for closed-form linear regression!)

$$\begin{bmatrix} 9 & 3 & 5 \\ 4 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & -5 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} 28 & 18 \\ 11 & 9 \end{bmatrix}$$

$$\begin{bmatrix} 9 & 18 \\ 4 & 8 \end{bmatrix} + \begin{bmatrix} 9 & -15 \\ 3 & -5 \end{bmatrix} + \begin{bmatrix} 10 & 15 \\ 4 & 6 \end{bmatrix}$$

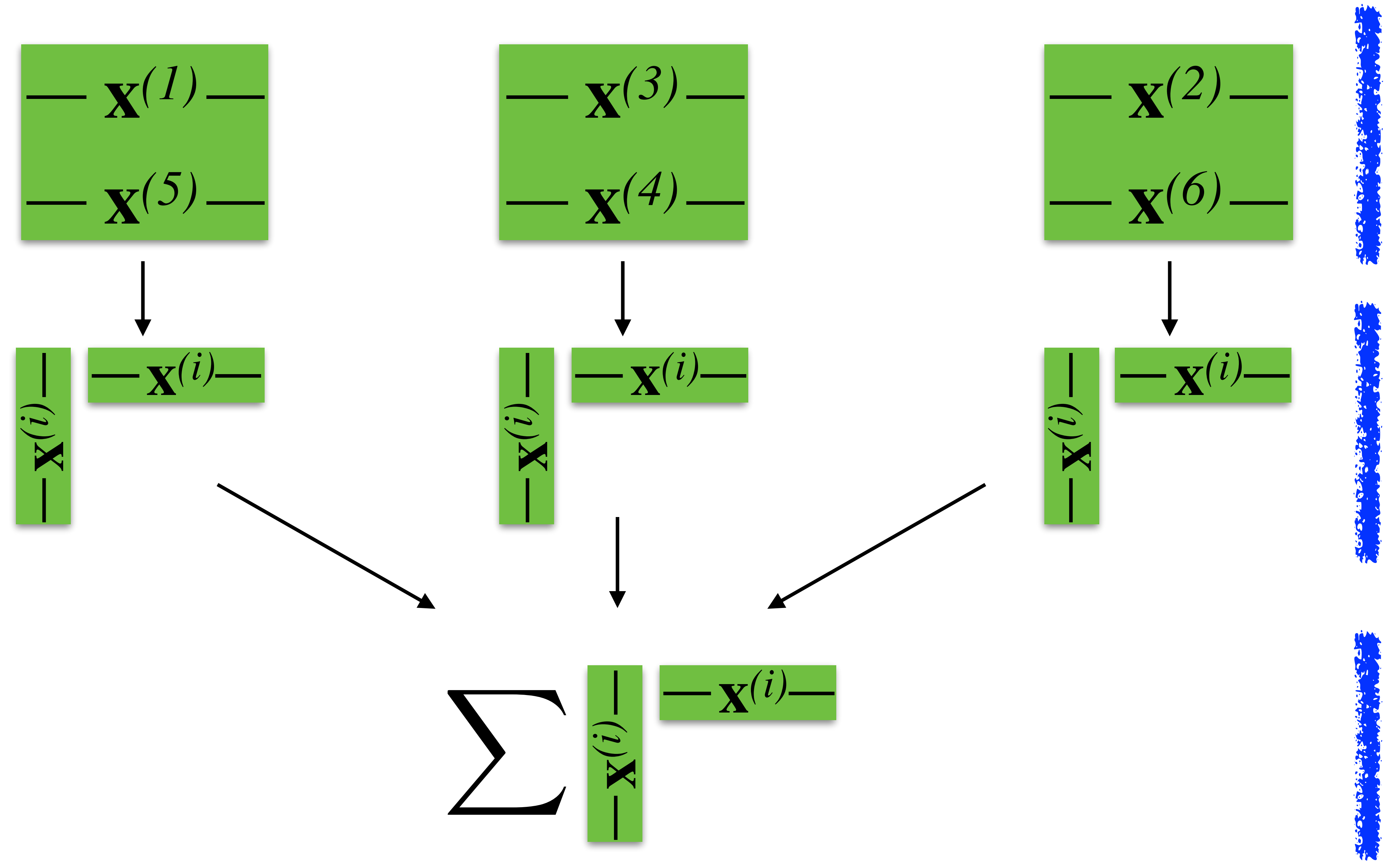


Example: n = 6; 3 workers

workers:

map:

reduce:



O(nd) Distributed Storage

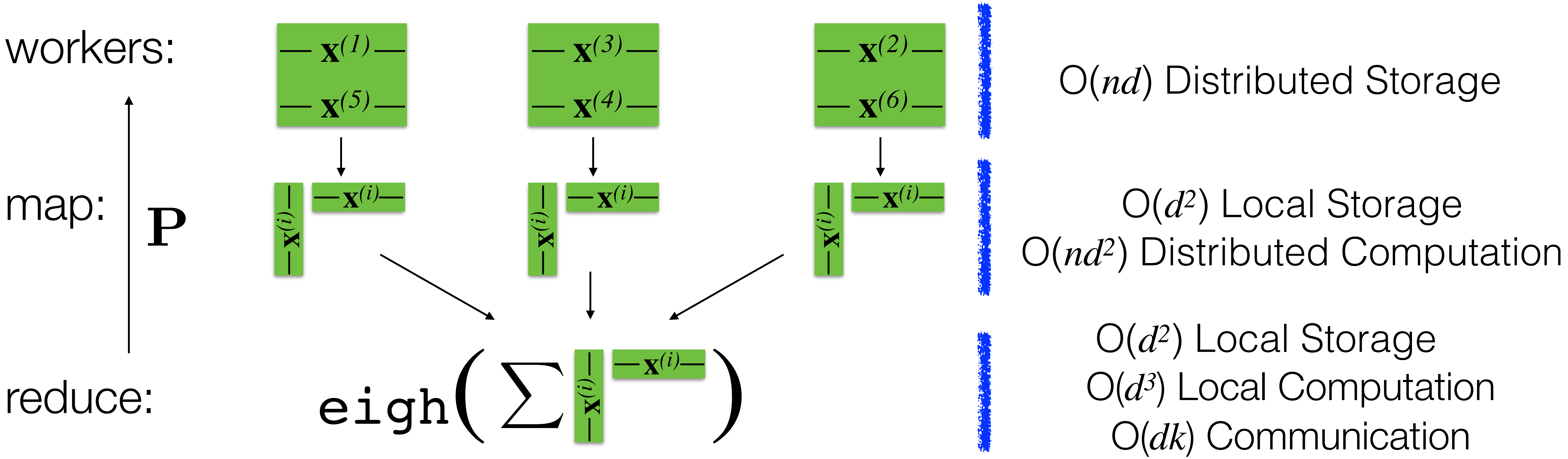
O(d^2) Local Storage
O(nd^2) Distributed Computation

O(d^2) Local Storage
O(d^2) Local Computation

Step 3: Eigendecomposition

- Perform locally since d is small
- Communicate k principal components ($\mathbf{P} \in \mathbb{R}^{d \times k}$) to workers

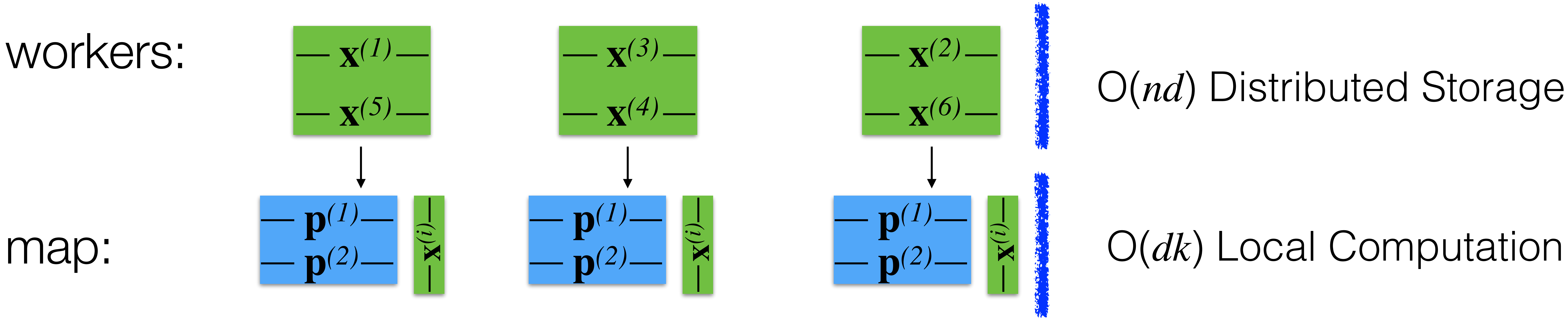
Example: n = 6; 3 workers



Step 4: Compute PCA Scores

- Multiply each point by principal components, $\mathbf{P}$

Example: n = 6; 3 workers



Distributed PCA, Part II (Optional)

PCA at Scale

Case 1: Big n and Small d

- O(d^2) local storage, O(d^3) local computation, O(dk) communication
- Similar strategy as closed-form linear regression

Case 2: Big n and Big d

- O($dk + n$) local storage, computation; O($dk + n$) communication
- Iterative algorithm

$$\begin{bmatrix}\mathbf{Z}\end{bmatrix} = \begin{bmatrix}\mathbf{X}\end{bmatrix}\begin{bmatrix}\mathbf{P}\end{bmatrix}$$

An Iterative Approach

We can use algorithms that rely on a **sequence of matrix-vector products** to compute top k eigenvectors ($\mathbf{P}$)

- E.g., Krylov subspace or random projection methods

Krylov subspace methods (used in MLlib) iteratively compute $\mathbf{X}^\top\mathbf{X}\mathbf{v}$ for some $\mathbf{v} \in \mathbb{R}^d$ provided by the method

- Requires O(k) passes over the data and O(dk) local storage
- We don't need to compute the covariance matrix!

Repeat for O(k) iterations:

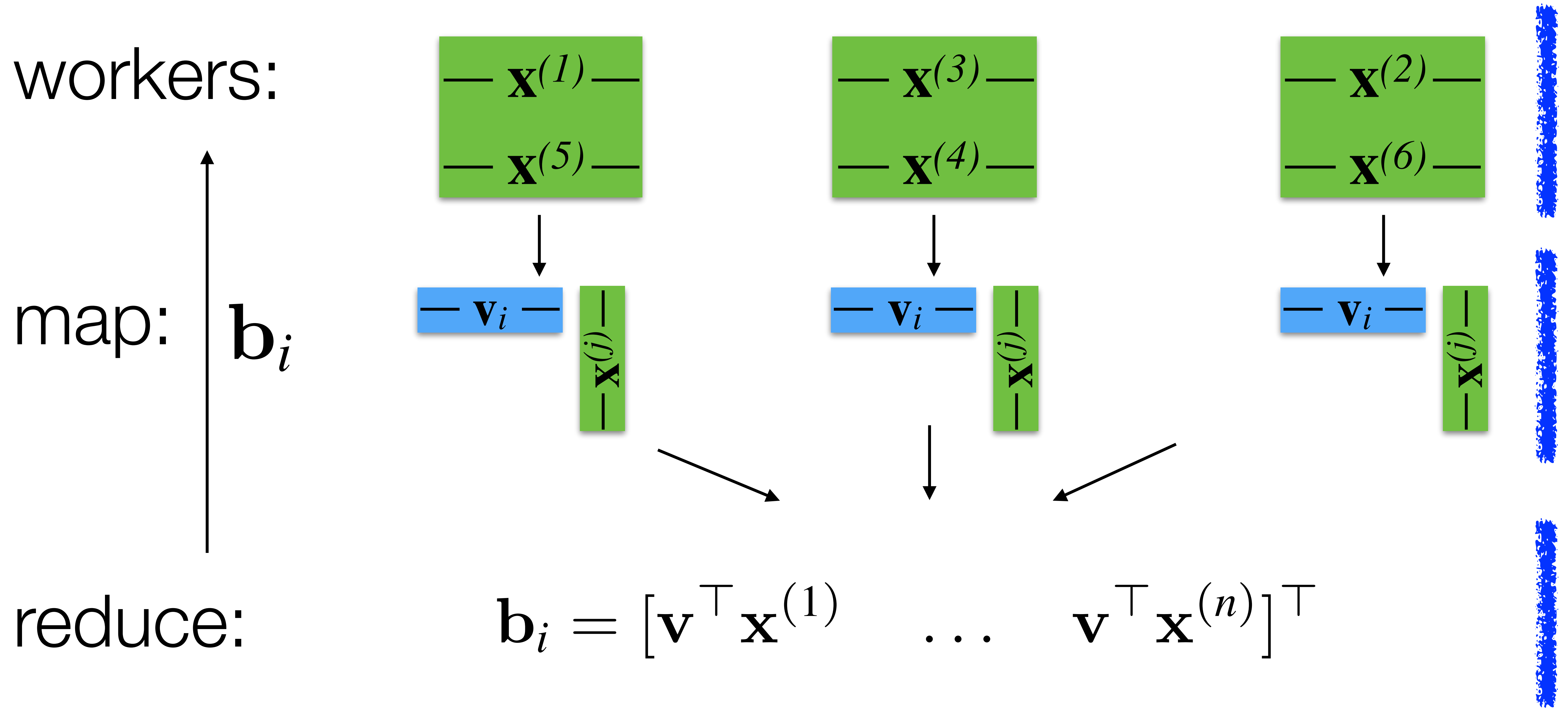
1. Communicate $\mathbf{v}_i \in \mathbb{R}^d$ to all workers
2. Compute $\mathbf{q}_i = \mathbf{X}^\top \mathbf{X} \mathbf{v}_i$ in a distributed fashion
   - Step 1: $\mathbf{b}_i = \mathbf{X} \mathbf{v}_i$
   - Step 2: $\mathbf{q}_i = \mathbf{X}^\top \mathbf{b}_i$
3. Driver uses $\mathbf{q}_i$ to update estimate of $\mathbf{P}$

Compute $\mathbf{q}_i = \mathbf{X}^\top \mathbf{X} \mathbf{v}_i$ in a distributed fashion

- $\mathbf{b}_i = \mathbf{X}\mathbf{v}_i$: each component is dot product, then concatenate

```
> b = np.array(trainData.map(dotProduct).collect())
```

Example: $n = 6$; 3 workers



O(nd) Distributed Storage

O(d) Local Storage
O(nd) Distributed Computation

O(n) Local Storage
O(n) Local Computation
O(n) Communication

Compute $\mathbf{q}_i = \mathbf{X}^\top \mathbf{X} \mathbf{v}_i$ in a distributed fashion

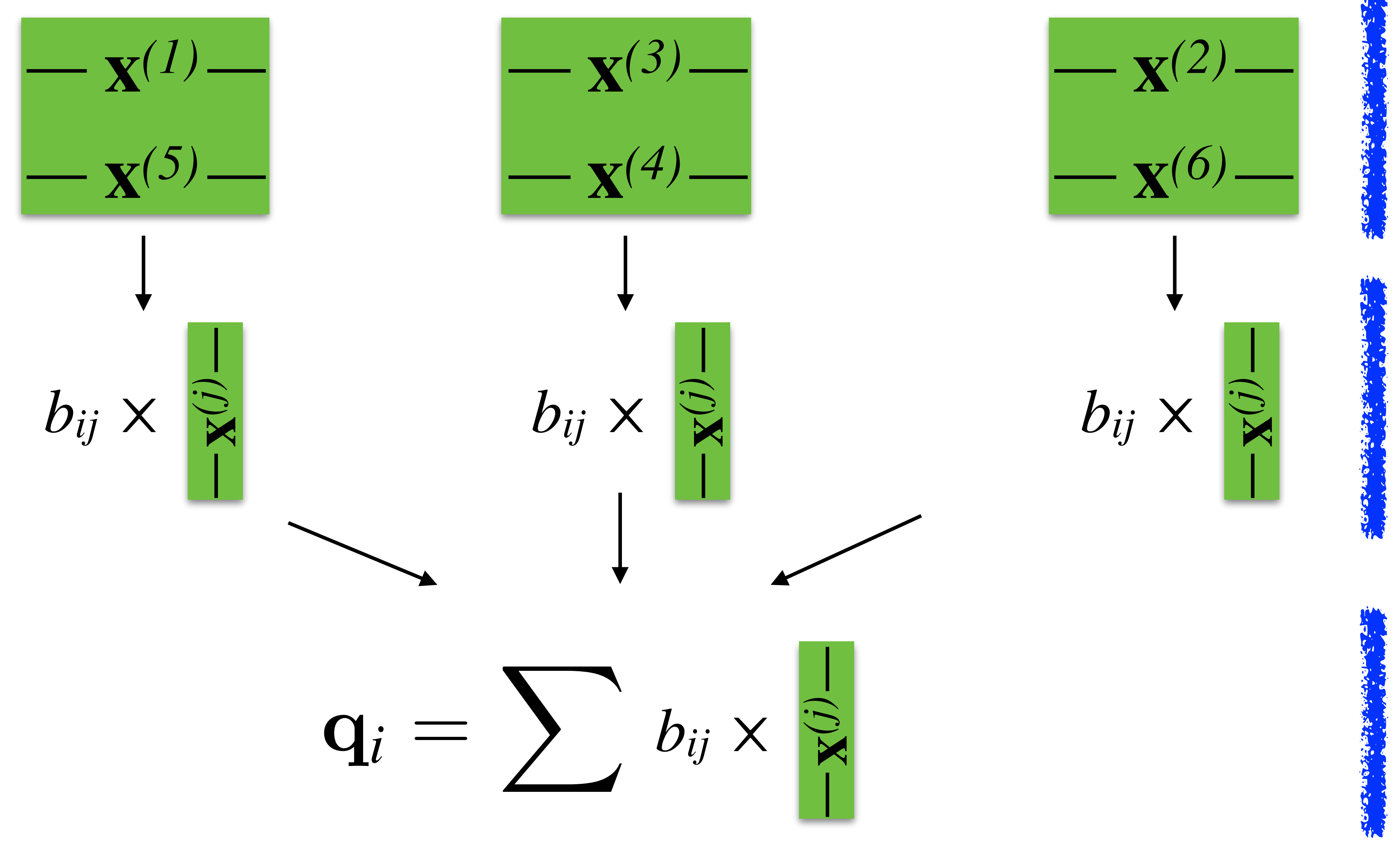
- $\mathbf{b}_i = \mathbf{X} \mathbf{v}_i$: each component is dot product, then concatenate
- $\mathbf{q}_i = \mathbf{X}^\top \mathbf{b}_i$: sum of rescaled data points $\mathbf{q}_i = \sum_{j=1}^{n} b_{ij} \mathbf{x}^{(j)}$

Example: n = 6; 3 workers

workers:

map:

reduce:



O(nd) Distributed Storage

O(n) Local Storage
O(nd) Distributed Computation

O(d) Local Storage
O(d) Local Computation
O(d) Communication

Compute $\mathbf{q}_i = \mathbf{X}^\top \mathbf{X} \mathbf{v}_i$ in a distributed fashion

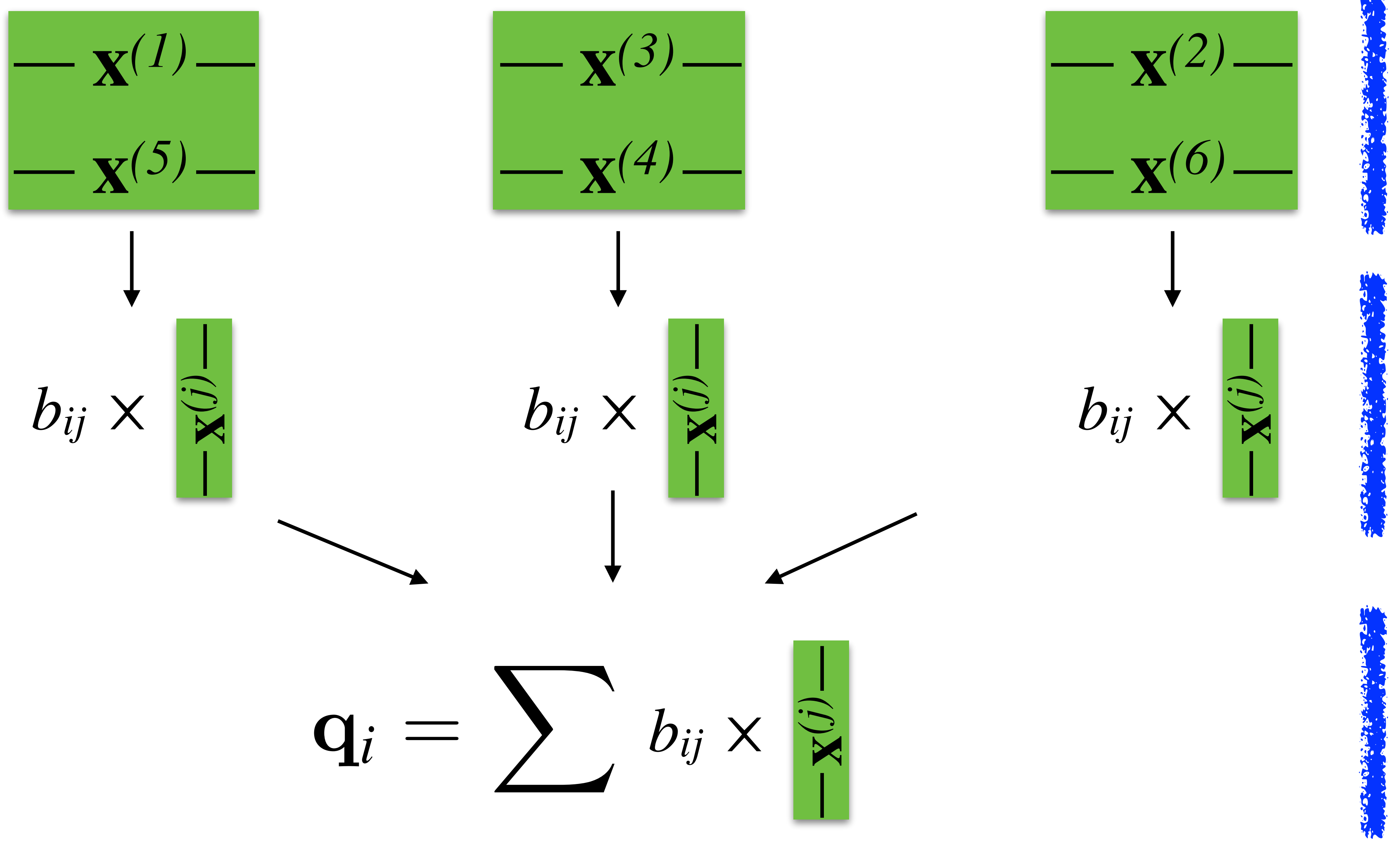
- $\mathbf{b}_i = \mathbf{X}\mathbf{v}_i$: each component is dot product, then concatenate
- $\mathbf{q}_i = \mathbf{X}^\top \mathbf{b}_i$: sum of rescaled data points $\mathbf{q}_i = \sum_{j=1}^{n} b_{ij}\mathbf{x}^{(j)}$

```
> q = trainData.map(rescaleByBi)
               .reduce(sumVectors)
```

workers:

map:

reduce:



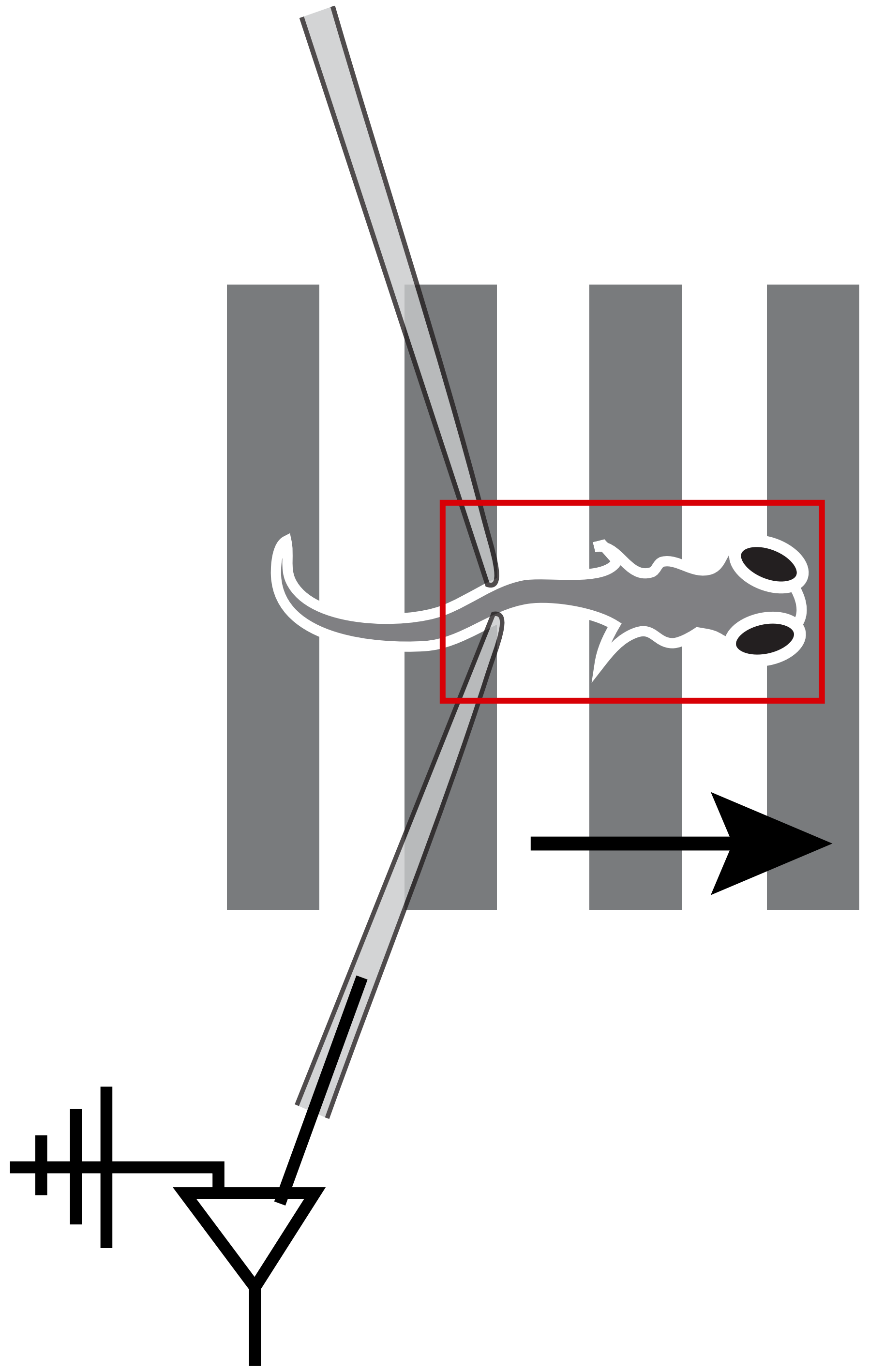
$O(nd)$ Distributed Storage

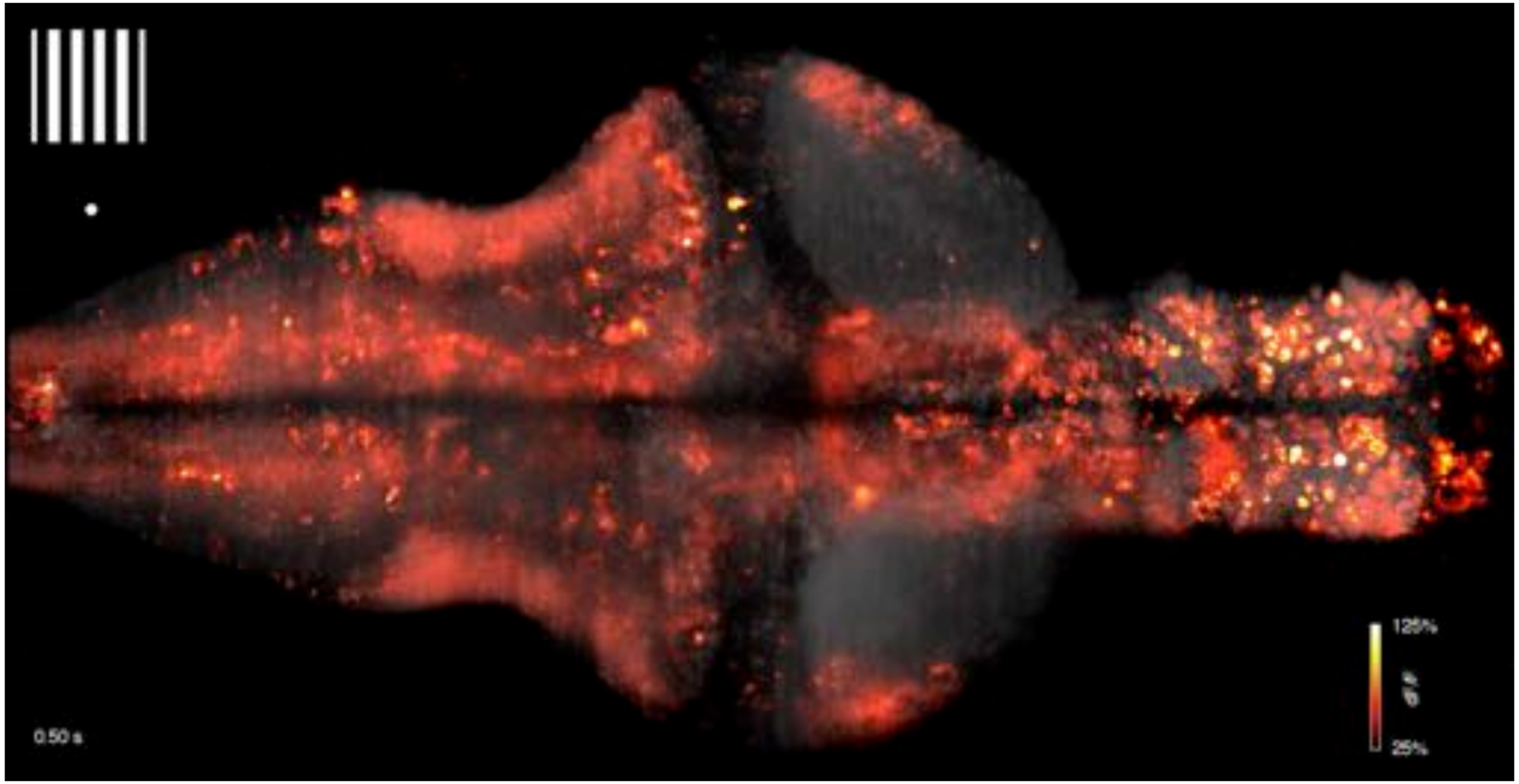
$O(n)$ Local Storage
$O(nd)$ Distributed Computation

$O(d)$ Local Storage
$O(d)$ Local Computation
$O(d)$ Communication

Lab Preview

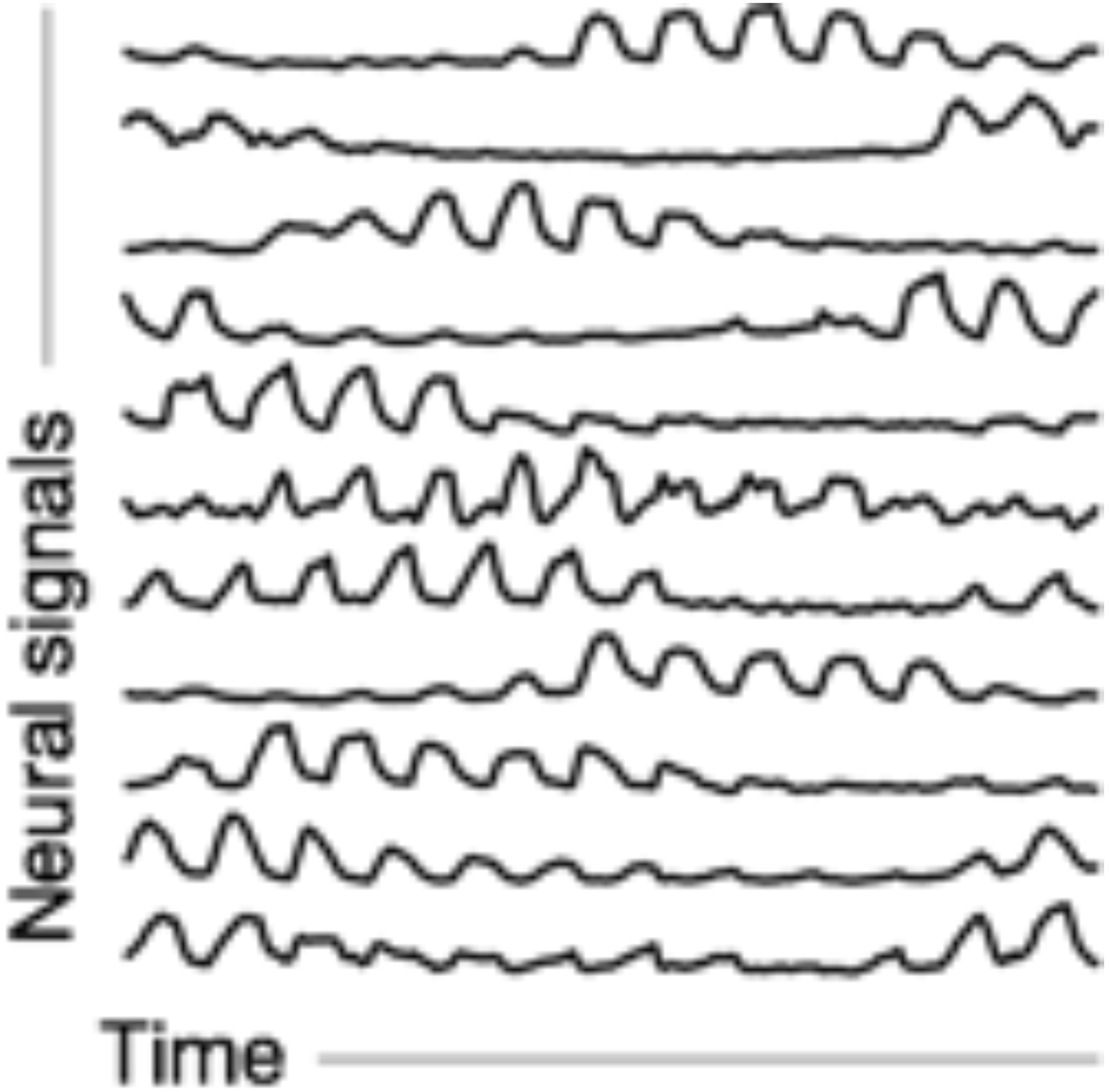
Vladimirov et al., 2014





Which areas are active at which times?

Which neuronal populations are activated by different directions of the stimulus?

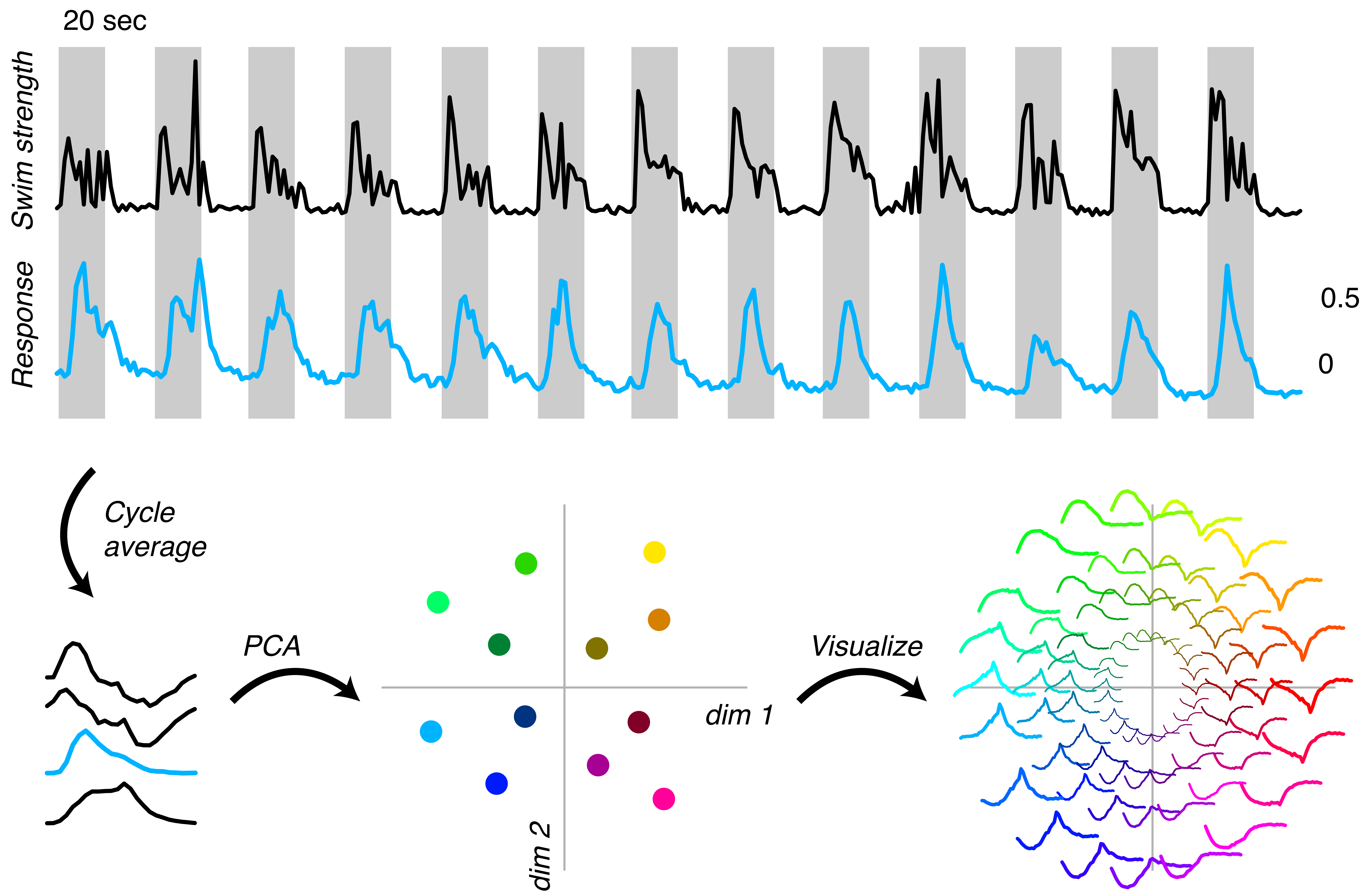


Given

Collection of neural time series

Goal

Find representations of data that reveal how responses are organized across space and time



20 sec
Swim strength
Response
0.5
0
Cycle average
PCA
dim 1
dim 2
Visualize