PCA Derivation (Optional)

Eigendecomposition

All covariance matrices have an eigendecomposition

- $\mathbf{C}_{\mathbf{X}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\top}$ (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: $\mathbf{C}_{\mathbf{x}}\mathbf{u} = \lambda\mathbf{u}$

- 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

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

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 $||\mathbf{p}||_2 = 1$

$$\sigma_{\mathbf{z}}^2 = \frac{1}{n}||\mathbf{z}||_2^2$$

Relationship between Euclidean distance and dot product $\quad = \frac{1}{n}\mathbf{z}^\top \mathbf{z}$

Definition: $\mathbf{z} = \mathbf{X}\mathbf{p}$ $\quad = \frac{1}{n}(\mathbf{X}\mathbf{p})^\top(\mathbf{X}\mathbf{p})$

Transpose property: $(\mathbf{X}\mathbf{p})^\top = \mathbf{p}^\top\mathbf{X}^\top$; associativity of multiply $\quad = \frac{1}{n}\mathbf{p}^\top\mathbf{X}^\top\mathbf{X}\mathbf{p}$

Definition: $\mathbf{C_X} = \frac{1}{n}\mathbf{X}^\top\mathbf{X}$ $\quad = \mathbf{p}^\top\mathbf{C_X}\mathbf{p}$

Restated Goal: $\max_{\mathbf{p}} \mathbf{p}^\top \mathbf{C_x} \mathbf{p}$ 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(d) local storage and computation on workers, O(dk) communication
- Iterative algorithm

$$\left[\ \mathbf{Z}\ \right] = \left[\quad \mathbf{X} \quad\right]\left[\ \mathbf{P}\ \right]$$

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(d) local storage and computation on workers, O(dk) communication
- Iterative algorithm

$$\left[\mathbf{Z}\right] = \left[\mathbf{X}\right]\left[\mathbf{P}\right]$$

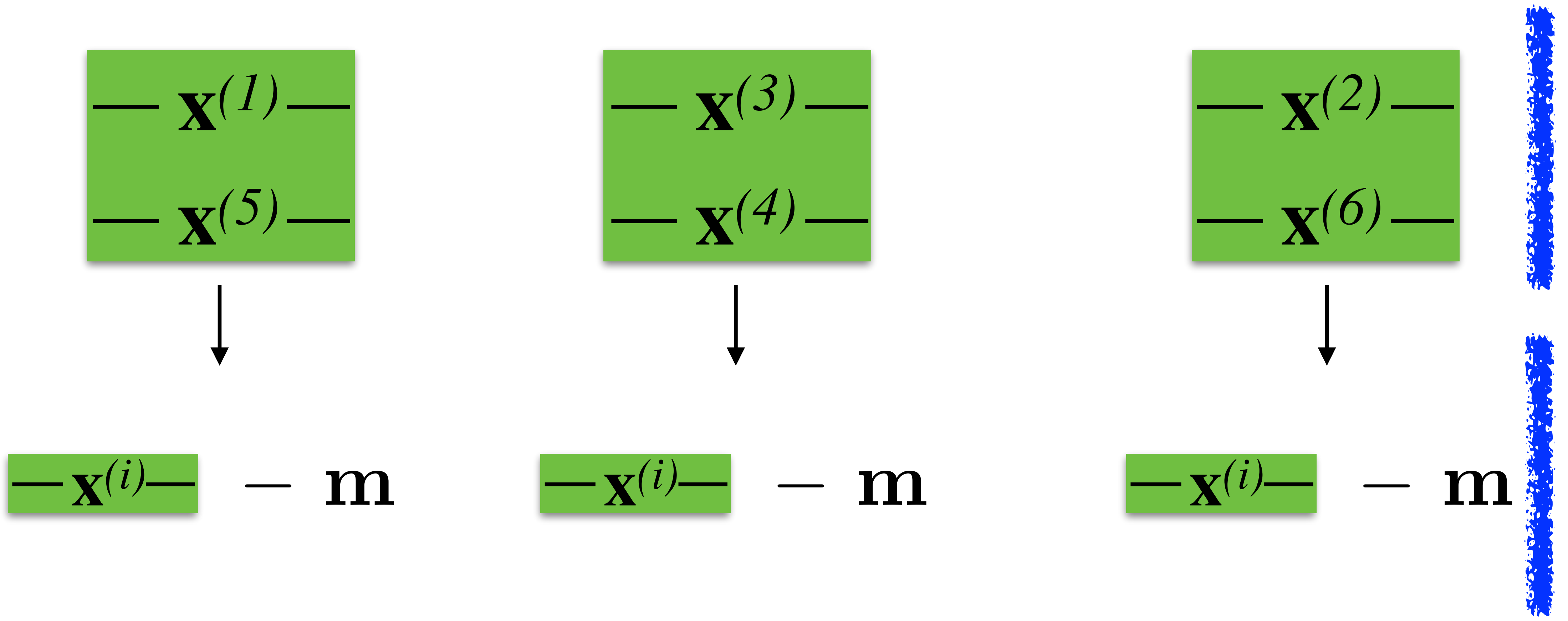
Step 1: Center Data

- 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!)

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

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

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!)

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

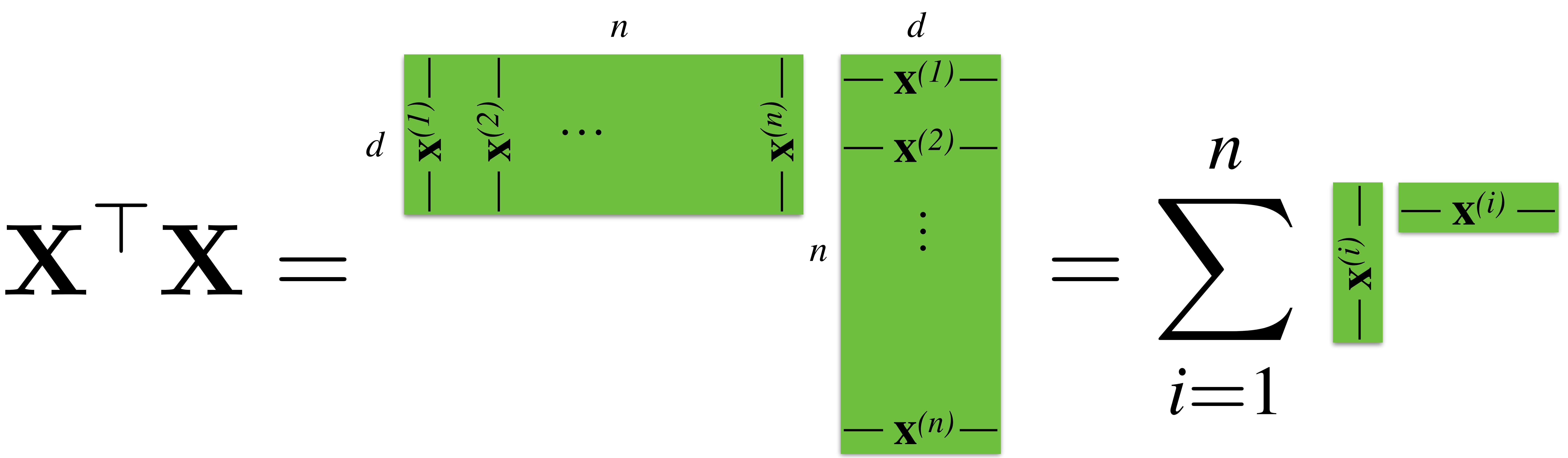
$$\begin{bmatrix} 9 & 18 \\ 4 & 8 \end{bmatrix} + \begin{bmatrix} 9 & -15 \\ 3 & -5 \end{bmatrix}$$

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

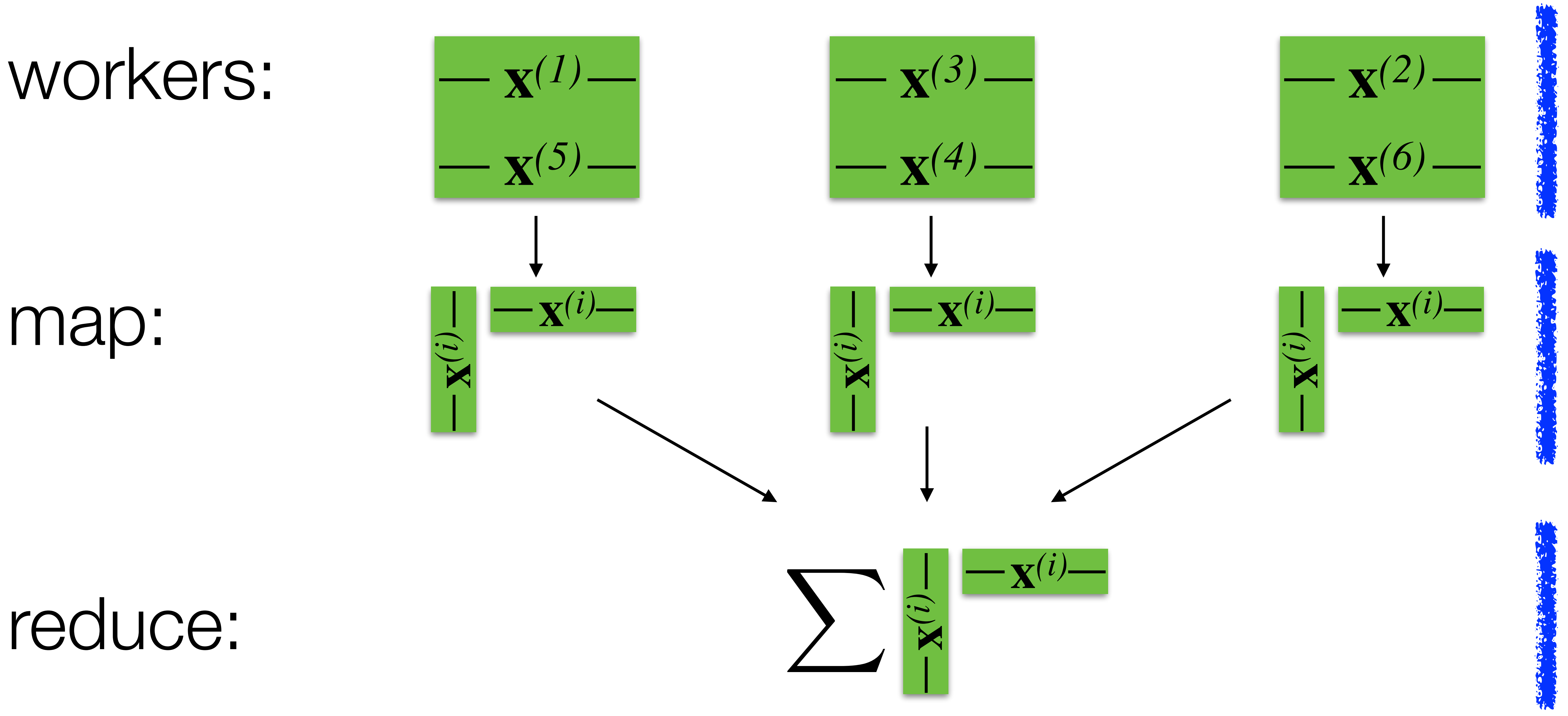
- Compute matrix product via outer products (just like we did for closed-form linear regression!)

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

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



Example: $n = 6$; 3 workers



$O(nd)$ Distributed Storage

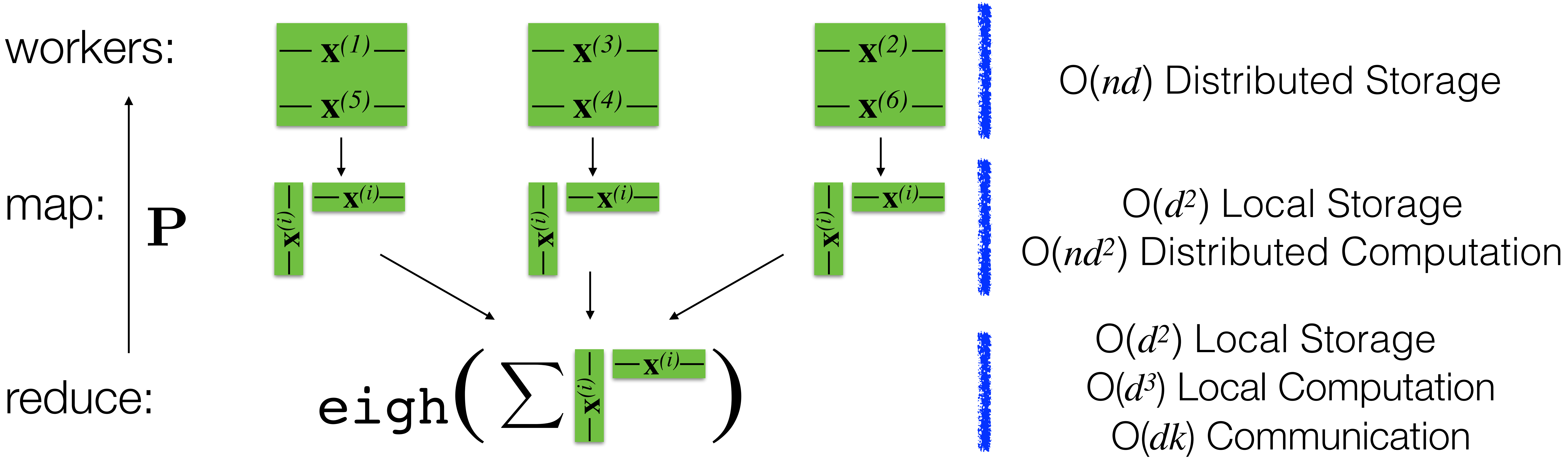
$O(d^2)$ Local Storage
$O(nd^2)$ Distributed Computation

$O(d^2)$ 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

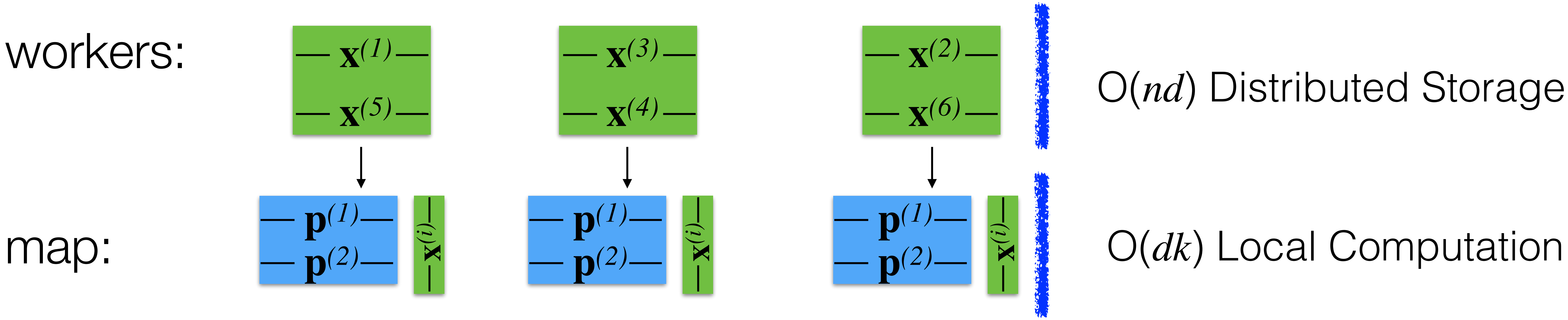
Example: n = 6; 3 workers



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

- $O(d)$ local storage and computation on workers, $O(dk)$ communication
- Iterative algorithm

$$\left[\,\mathbf{Z}\,\right] = \left[\quad\mathbf{X}\quad\right]\left[\,\mathbf{P}\,\right]$$

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 data, O(d) local storage on workers
- We don't need to compute the covariance matrix!

Repeat for O(k) iterations:

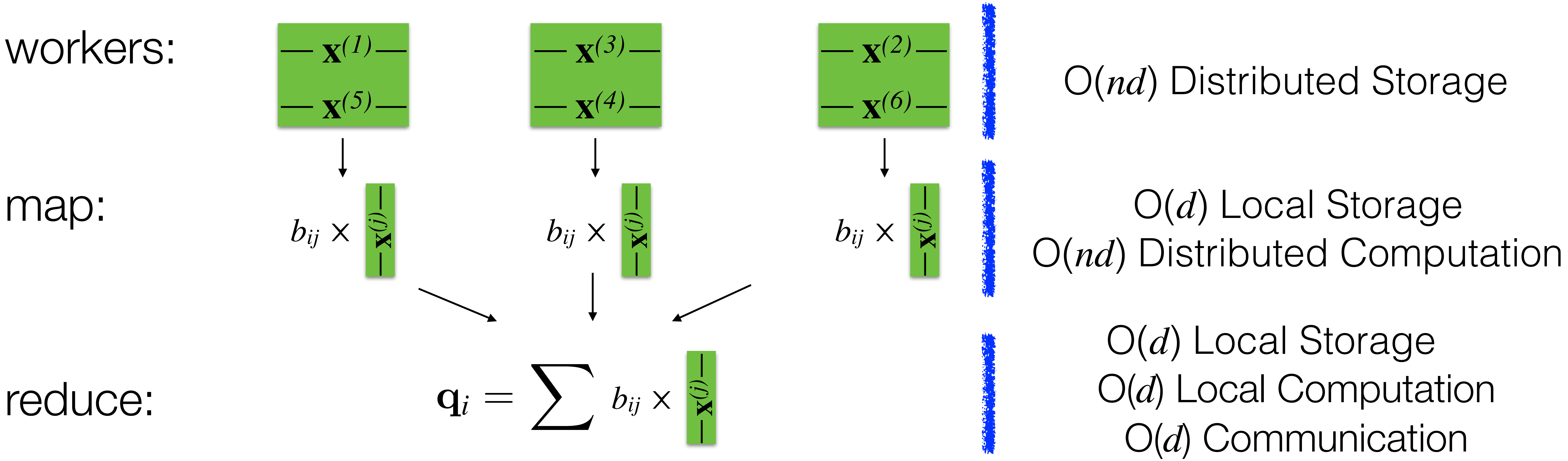
1. Communicate $\mathbf{v}_i \in \mathbb{R}^d$ to all workers
2. Compute $\mathbf{q}_i = \mathbf{X}^\top \mathbf{X} \mathbf{v}_i$ in a distributed fashion
   - Step 1: $\mathbf{b}_i = \mathbf{X} \mathbf{v}_i$
   - Step 2: $\mathbf{q}_i = \mathbf{X}^\top \mathbf{b}_i$
   - Perform in single map-reduce!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

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

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

- $b_{ij} = \mathbf{v}_i^\top \mathbf{x}^{(j)}$ and $\mathbf{q}_i = \sum_{j=1}^{n} b_{ij} \mathbf{x}^{(j)}$
- Locally compute each dot product and rescale each point before summing all rescaled points in reduce step!

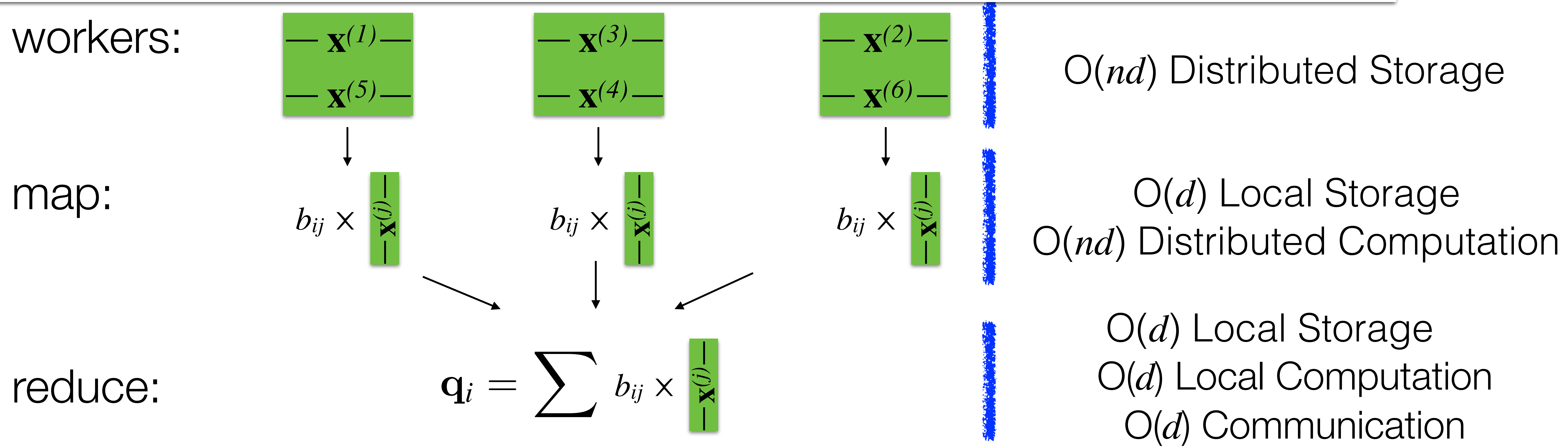
Example: $n = 6$; 3 workers



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

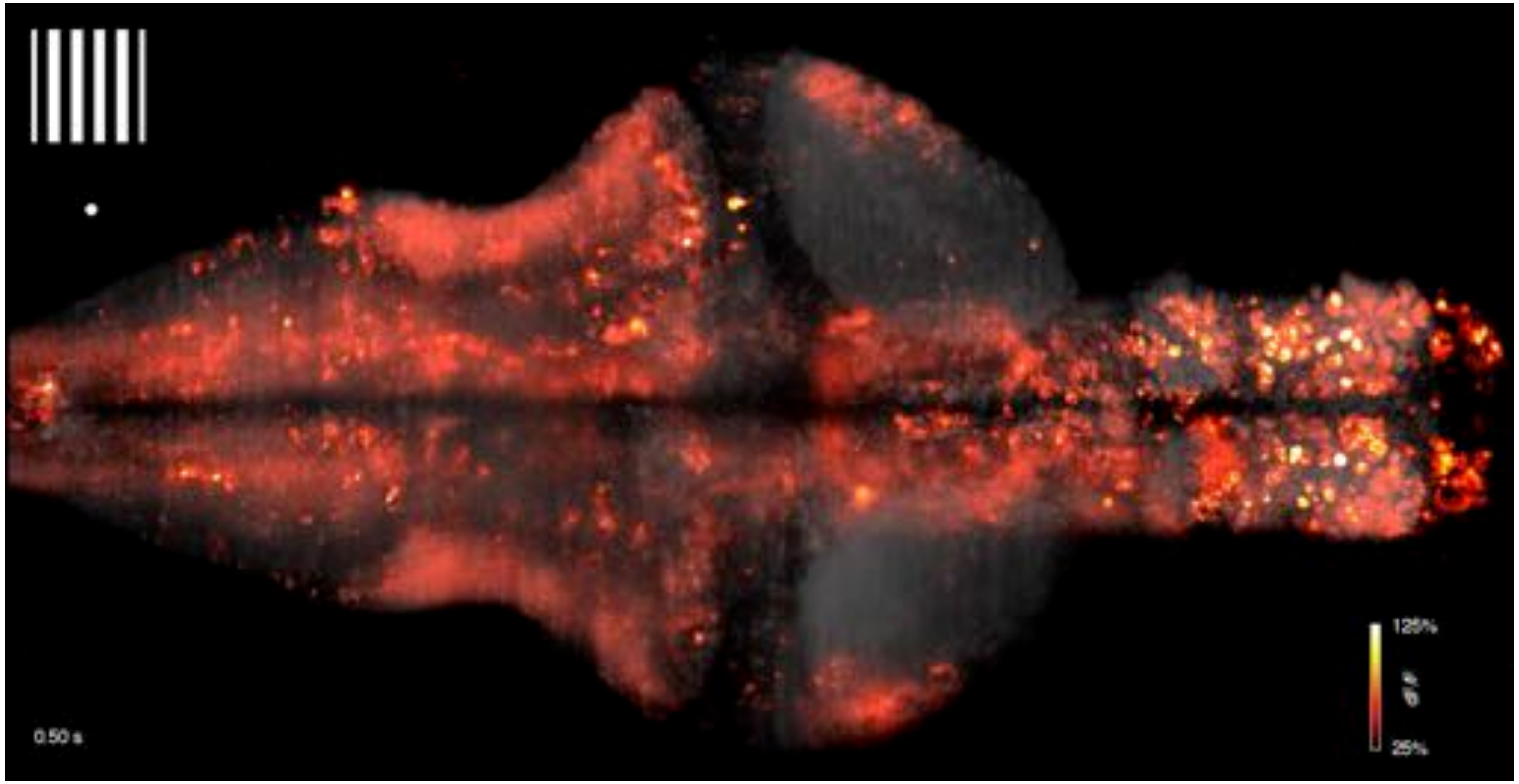
- $b_{ij} = \mathbf{v}_i^\top \mathbf{x}^{(j)}$ and $\mathbf{q}_i = \sum_{j=1}^{n} b_{ij} \mathbf{x}^{(j)}$
- Locally compute each dot product and rescale each point before summing all rescaled points in reduce step!

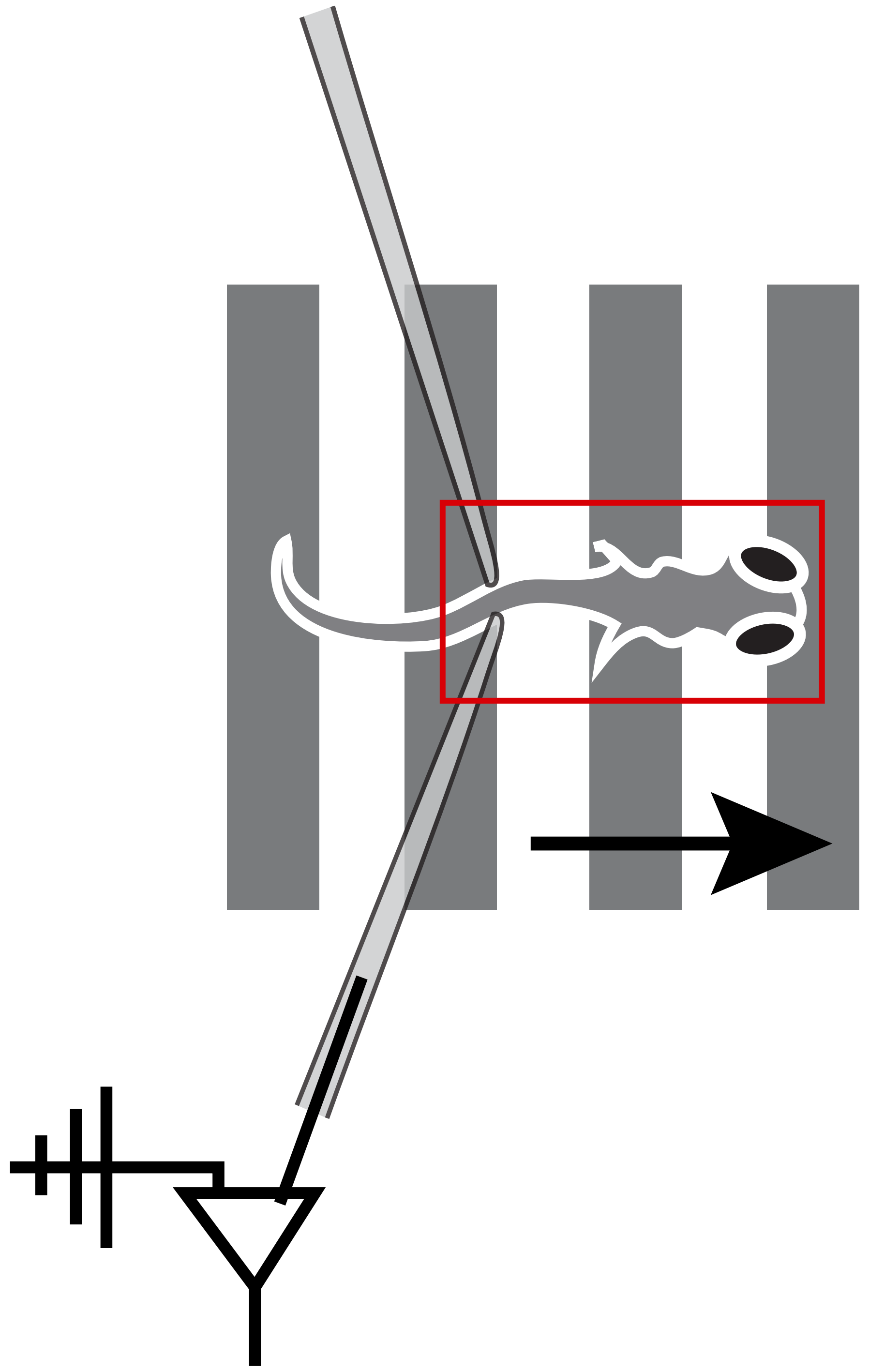
```
> q = trainData.map(rescaleByBi)
                .reduce(sumVectors)
```



Lab Preview

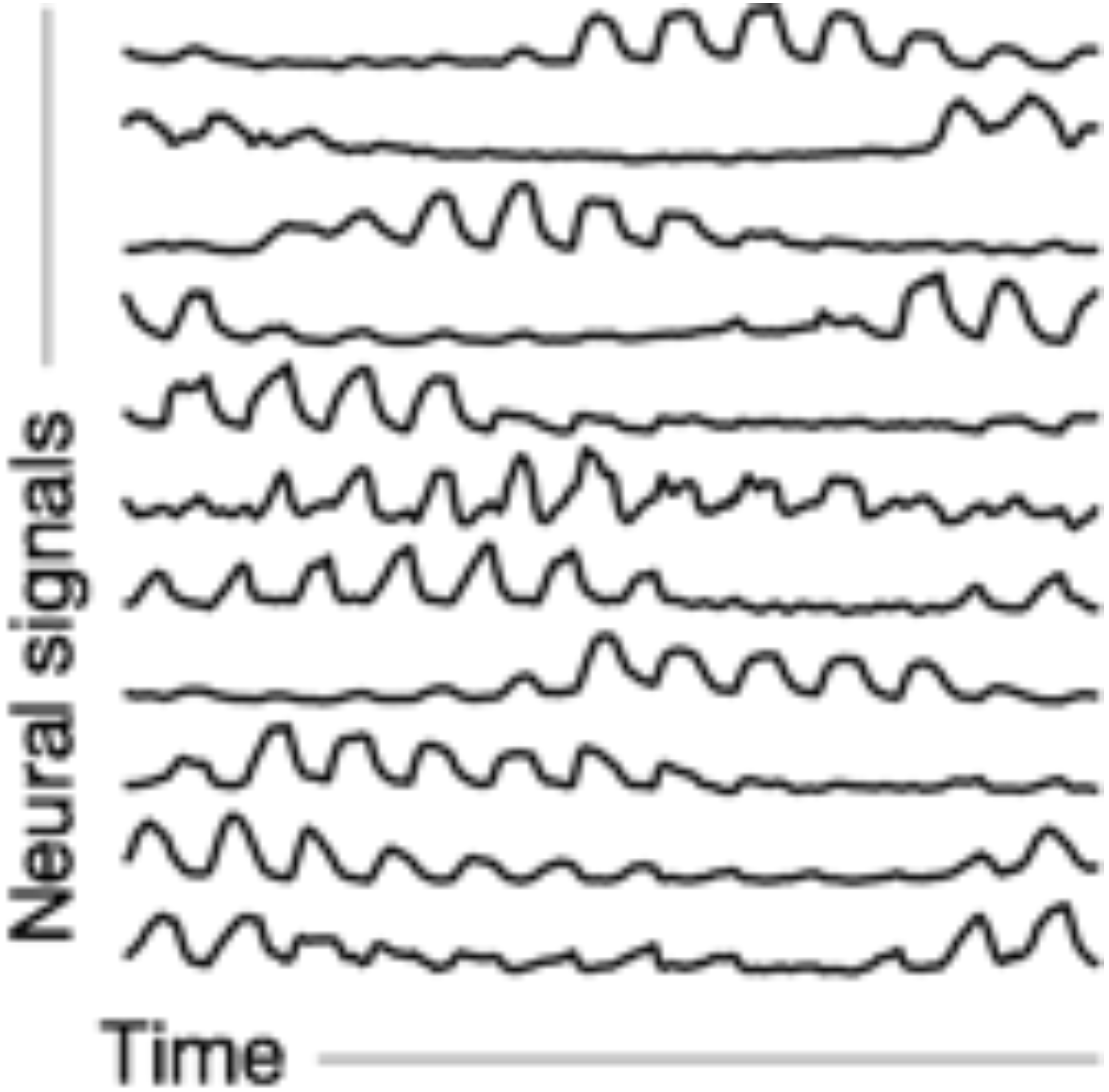
Vladimirov et al., 2014





Which areas are active at which times?

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

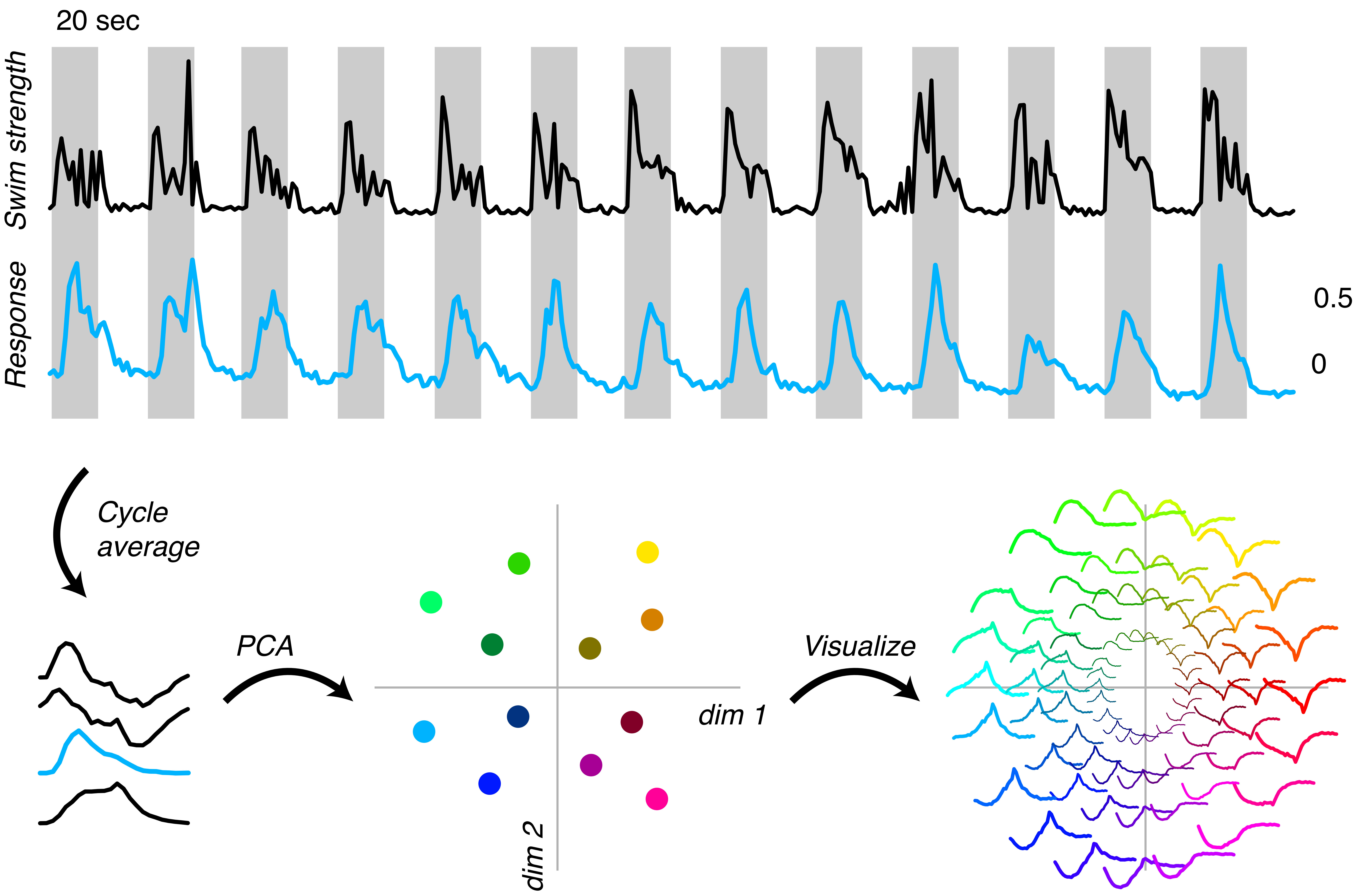


Given

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