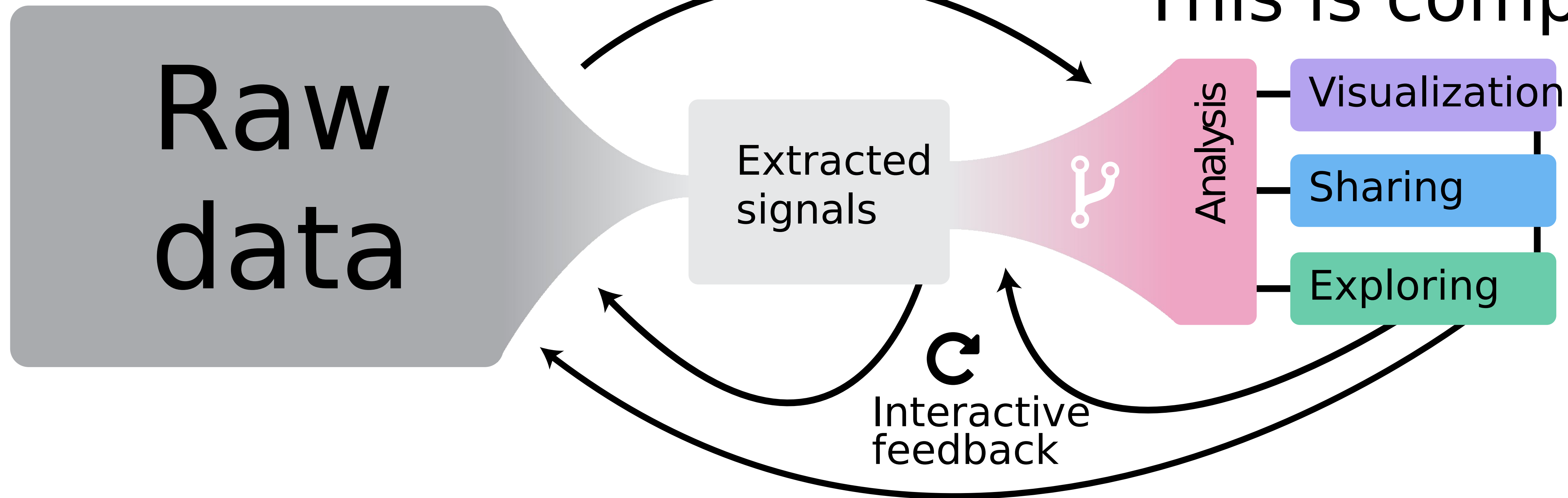


Exploratory Data Analysis

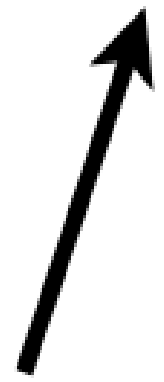
This is really big

This is complex



Supervised methods

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



predict
our data



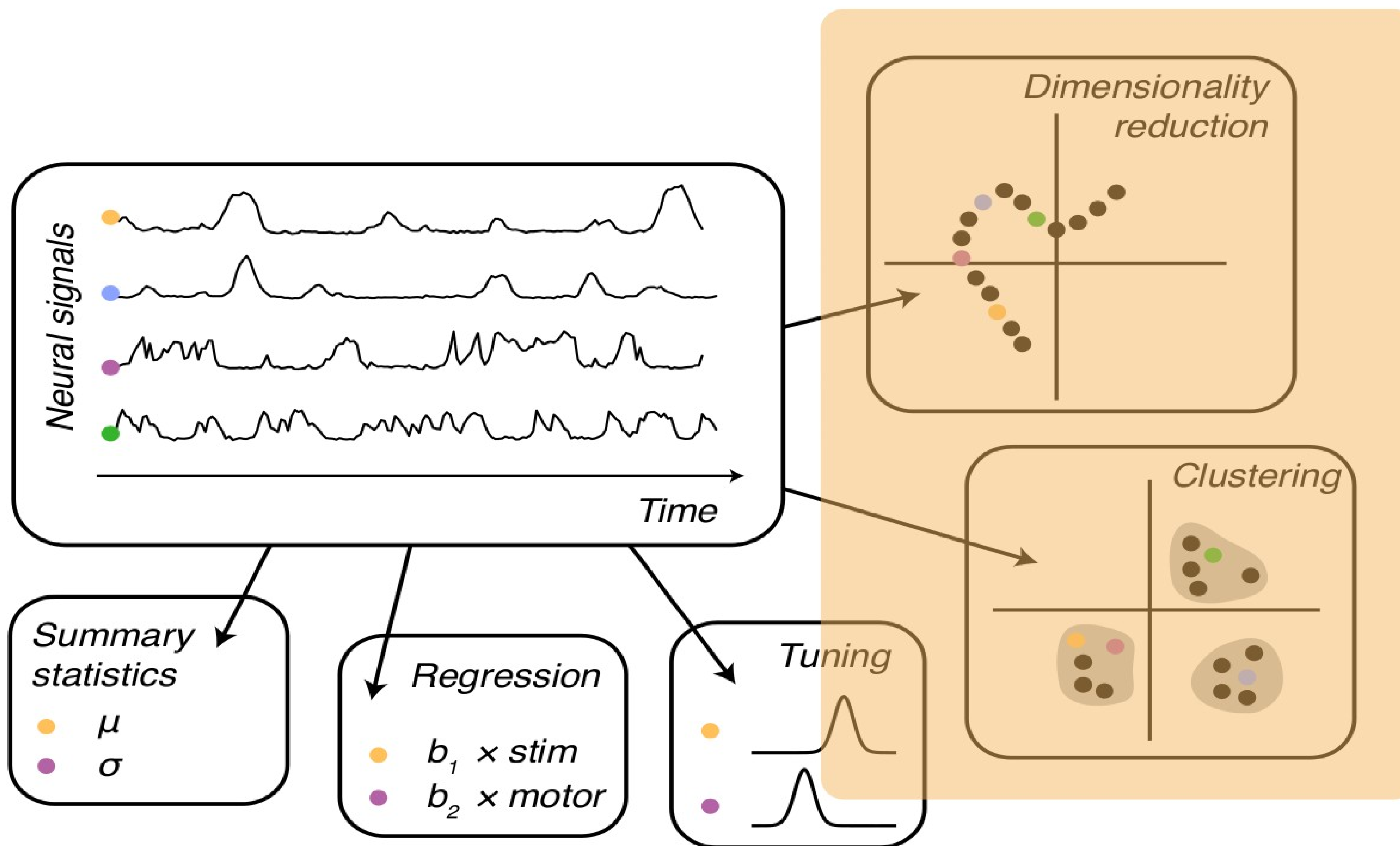
as a function
of other data

Unsupervised methods

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



find structure
in the data
on its own



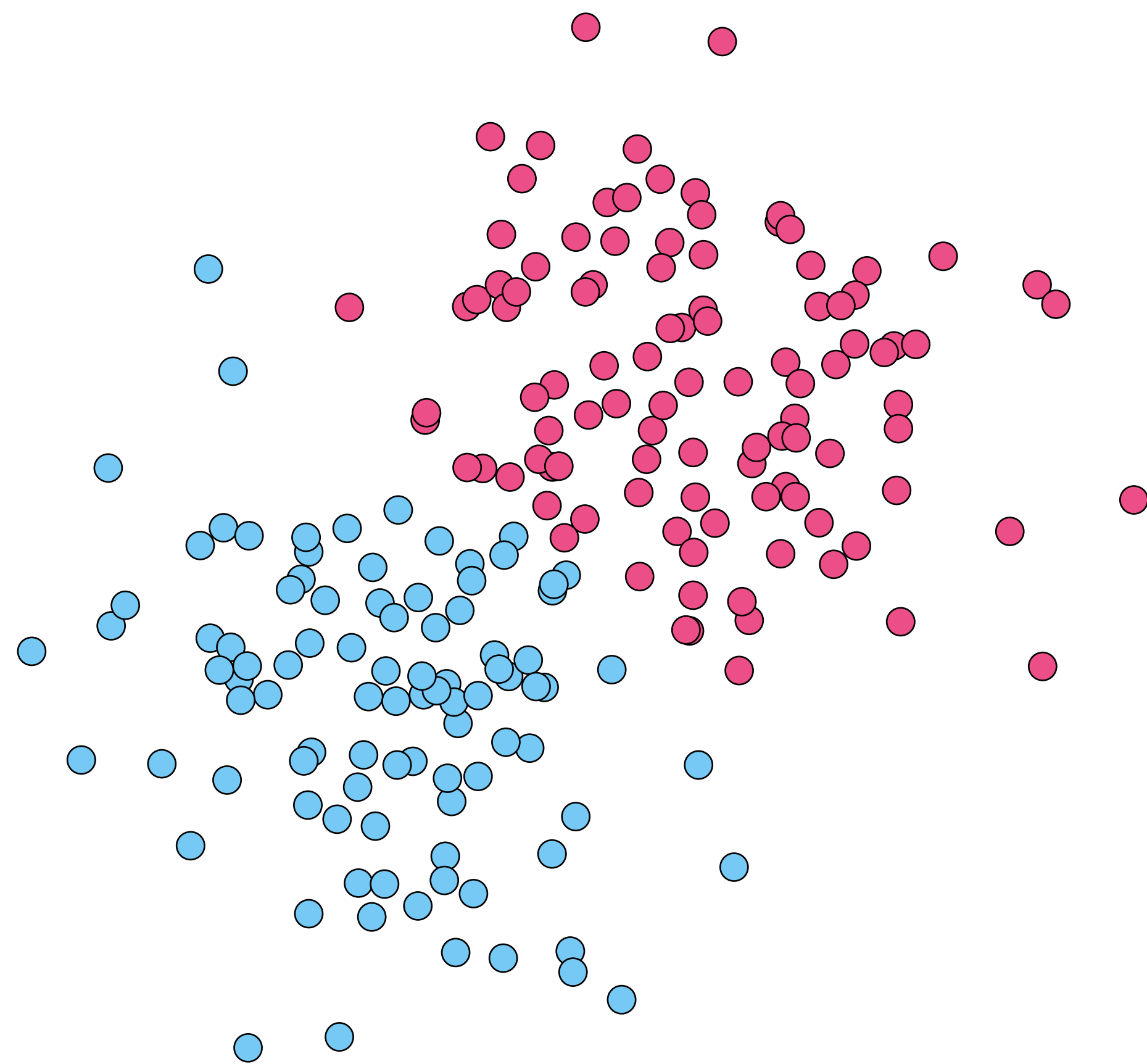
Clustering for preprocessing

- 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



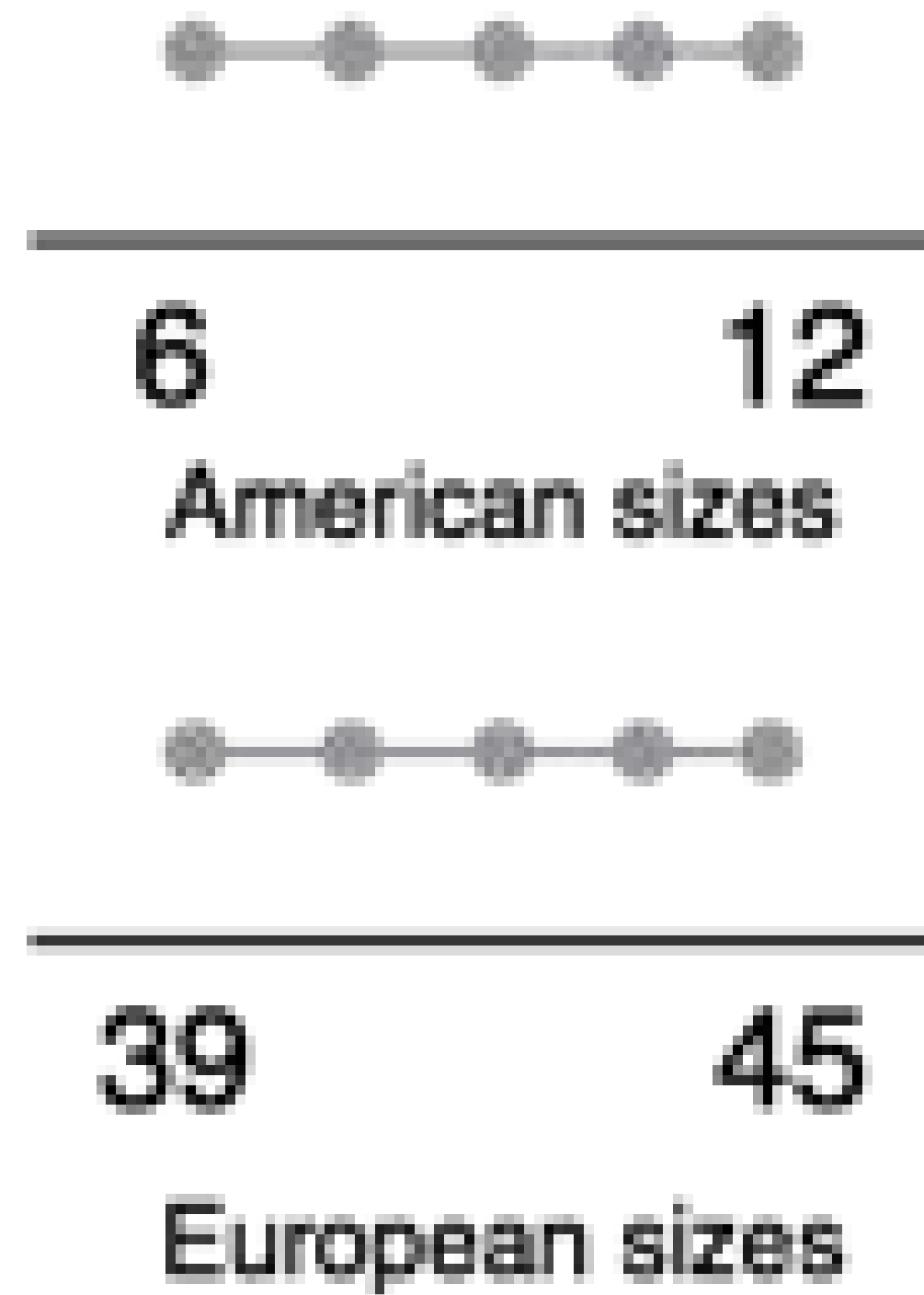
Clustered data



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

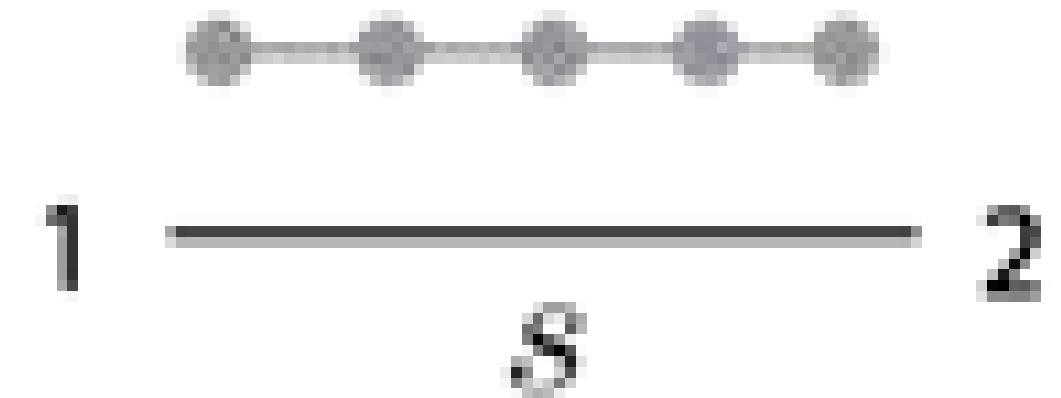
"Shoe store" space



weights

$$= \begin{bmatrix} 6s \\ 6s + 33 \end{bmatrix} * [s] \longleftrightarrow$$

"Size" space



**Dimensionality
reduction**

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’?

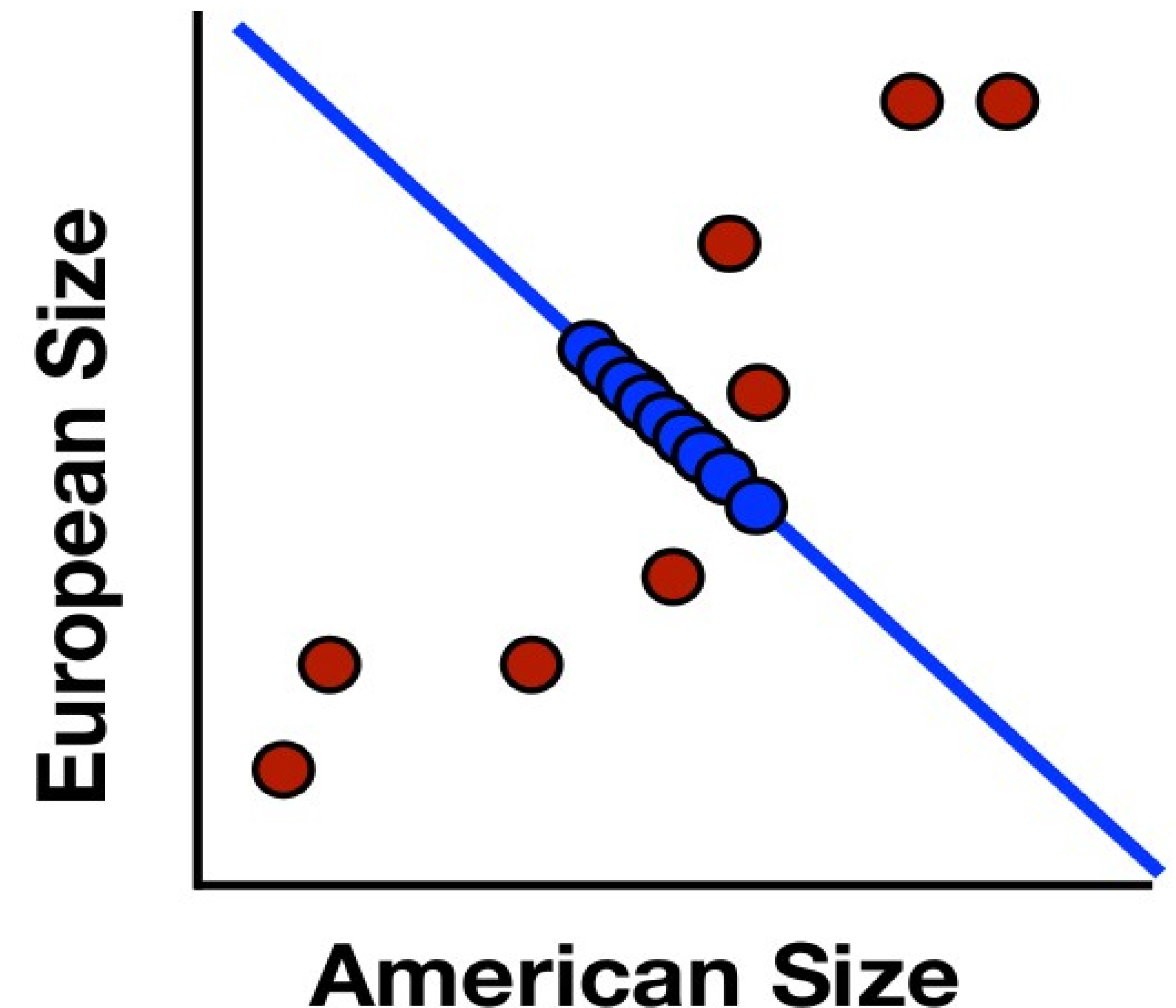
E.g., Shoe Size

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



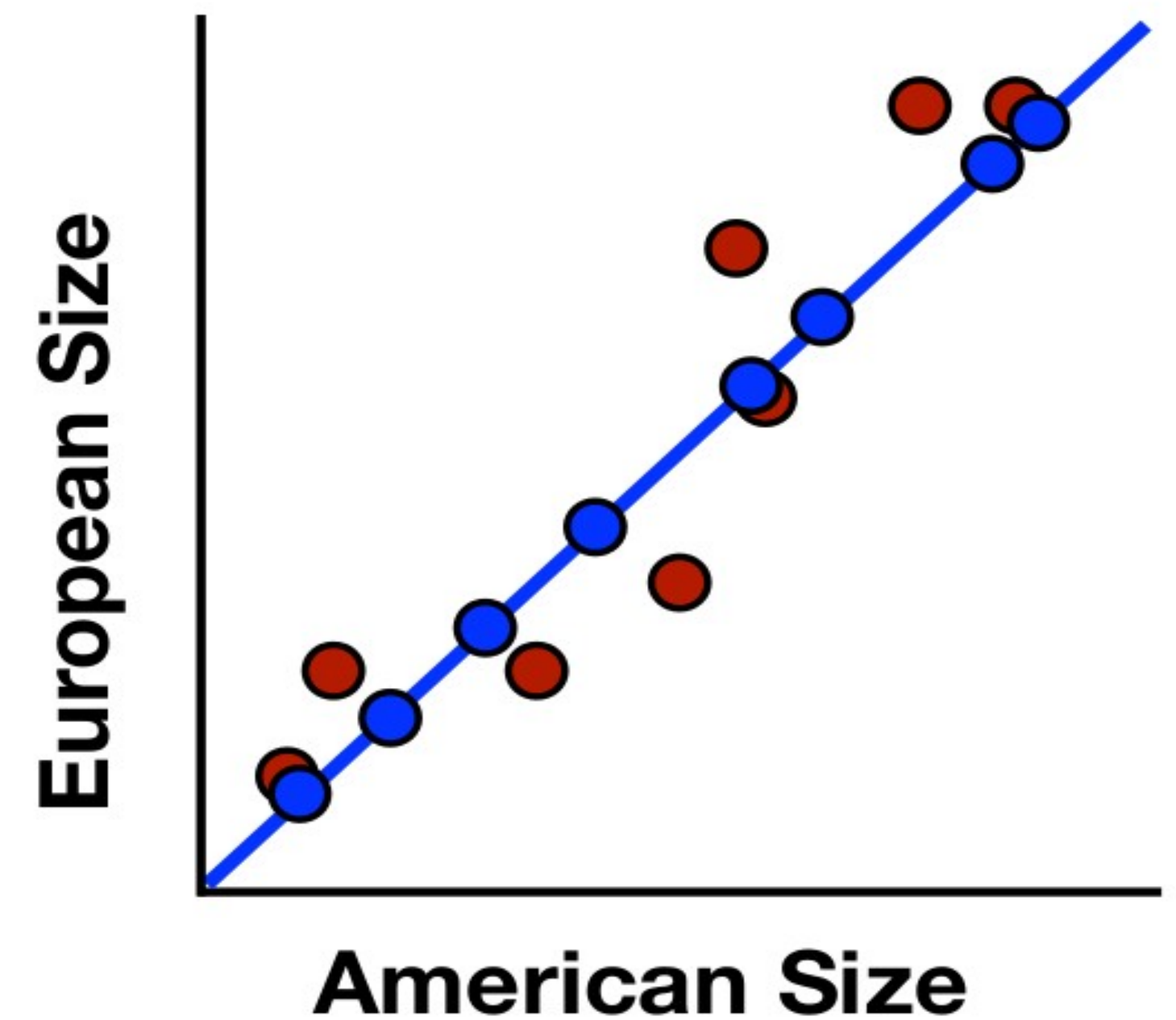
E.g., Shoe Size

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?

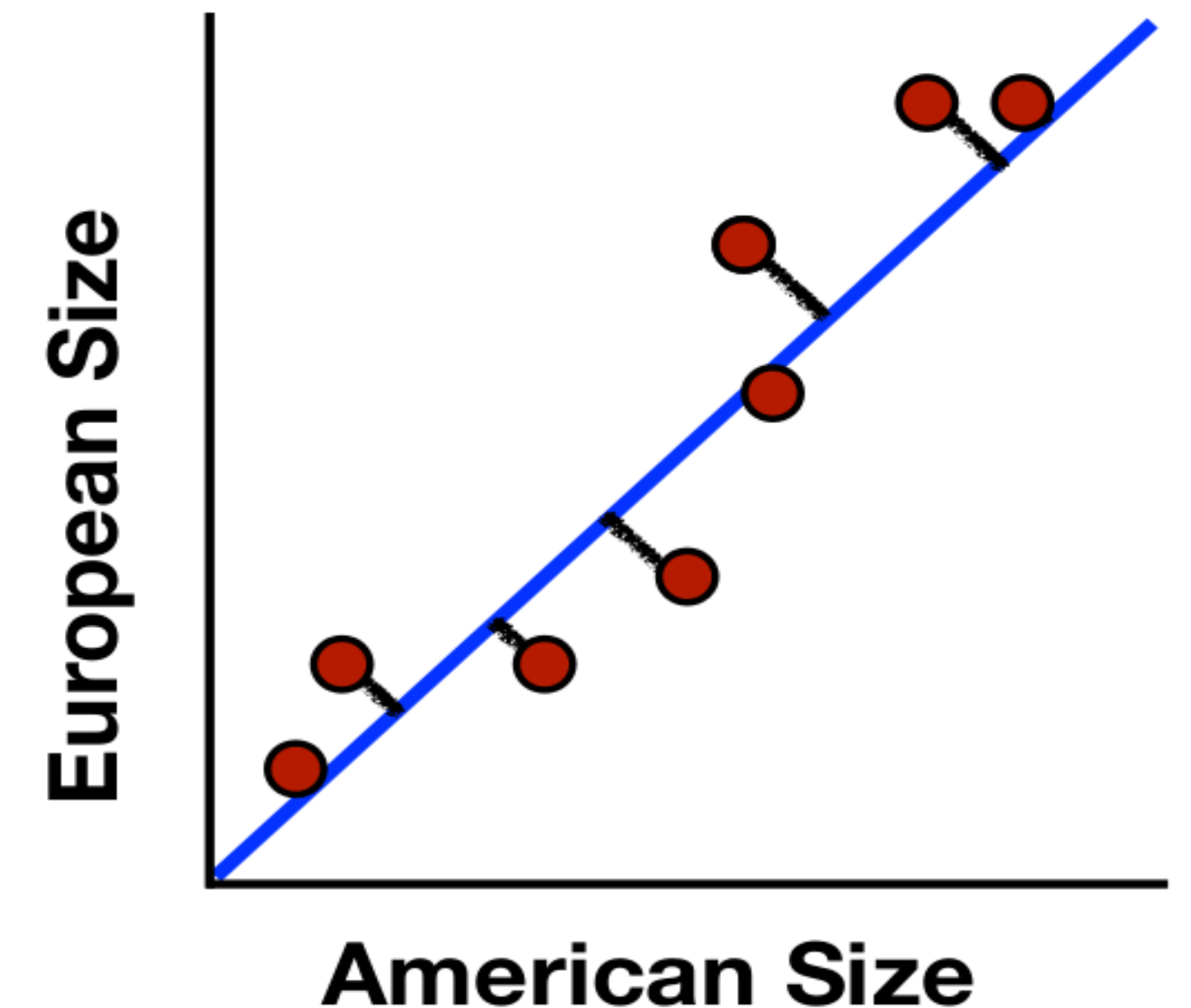
- Pick a direction and project onto this direction



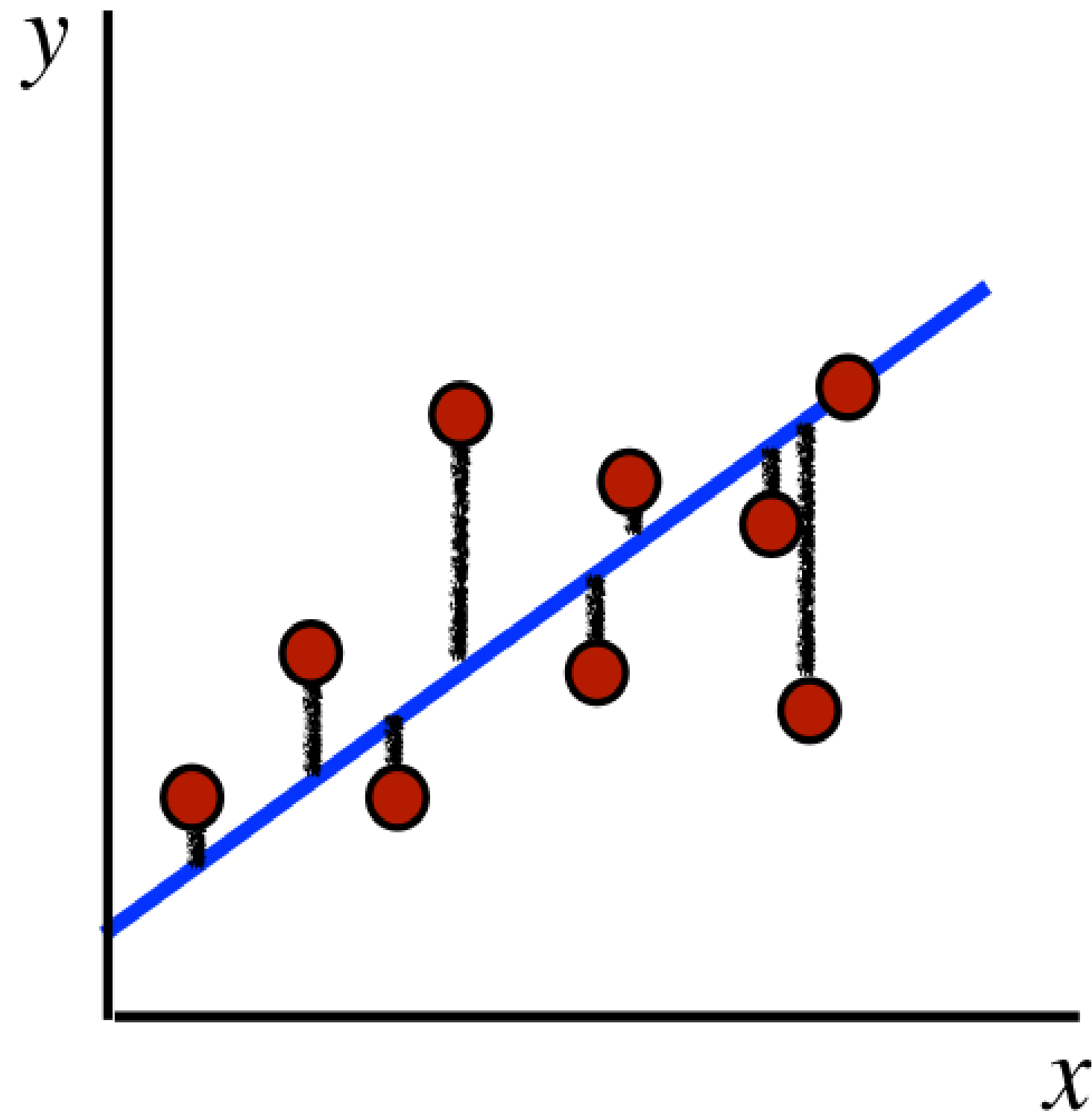
Goal: Minimize Reconstruction Error

Minimize Euclidean distances between original points and their projections

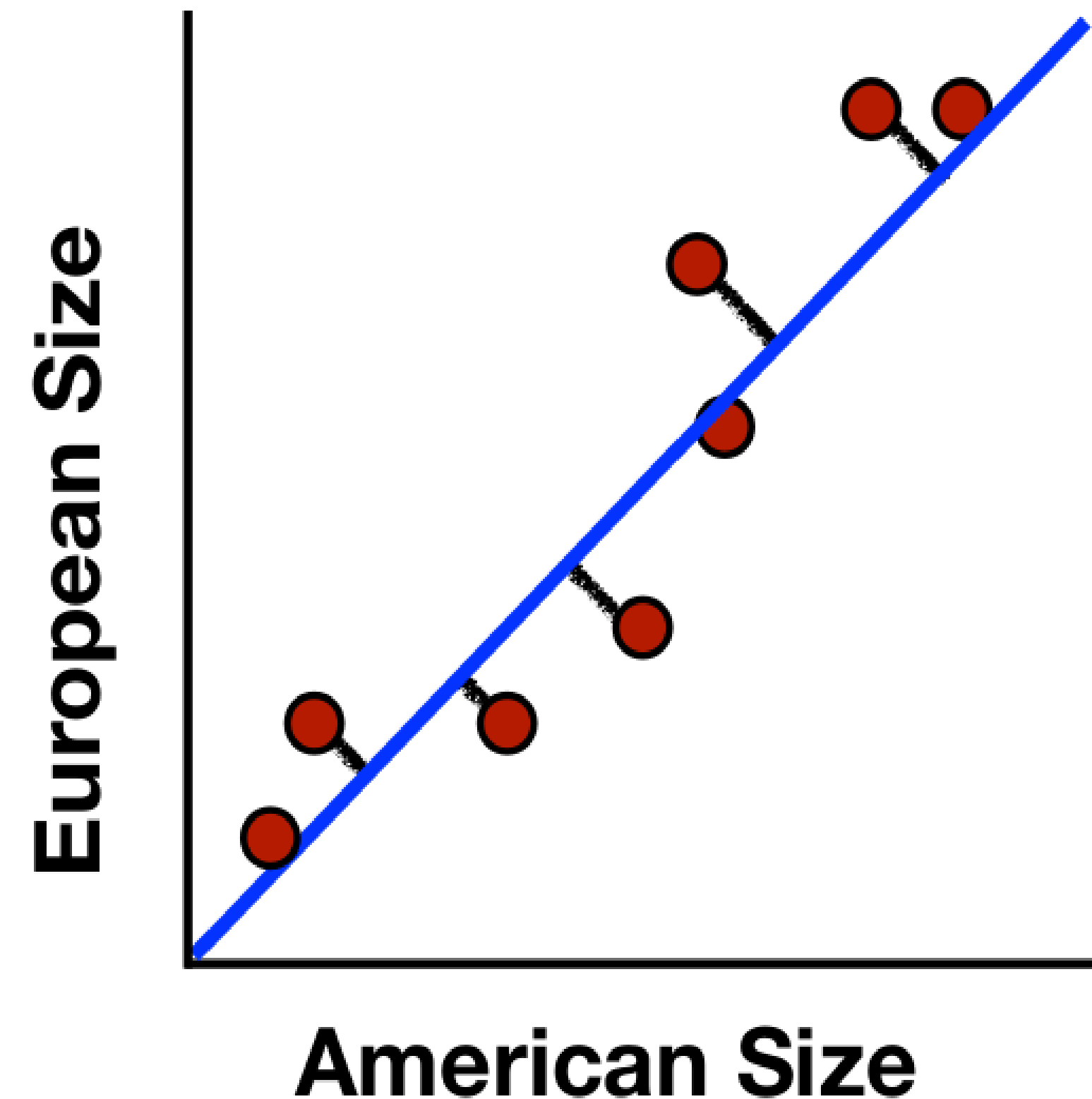
PCA solution solves this problem!



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



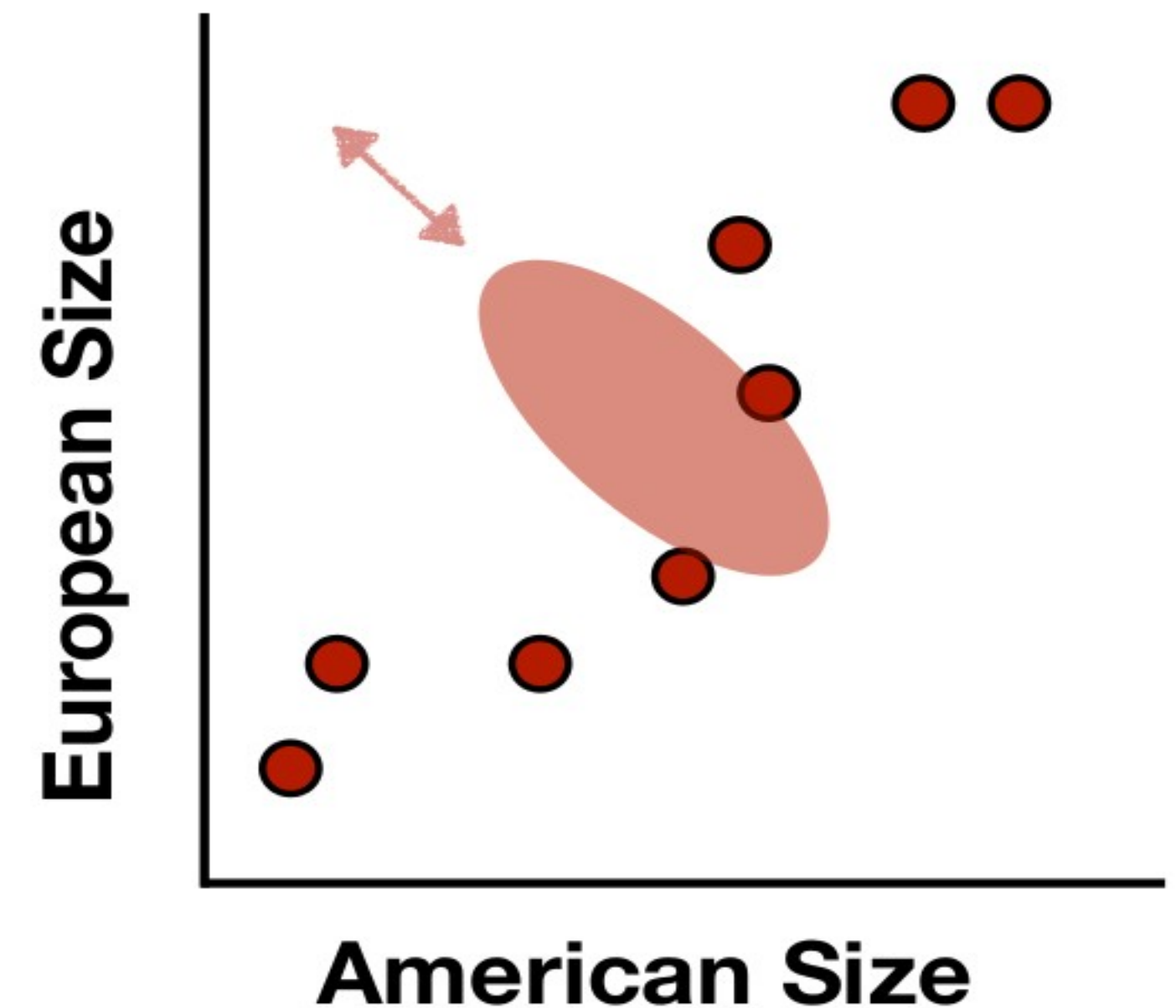
PCA — reconstruct 2D data via 2D
data with single degree of freedom.
Evaluate reconstructions (represented
by blue line) by **Euclidean** distances



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?

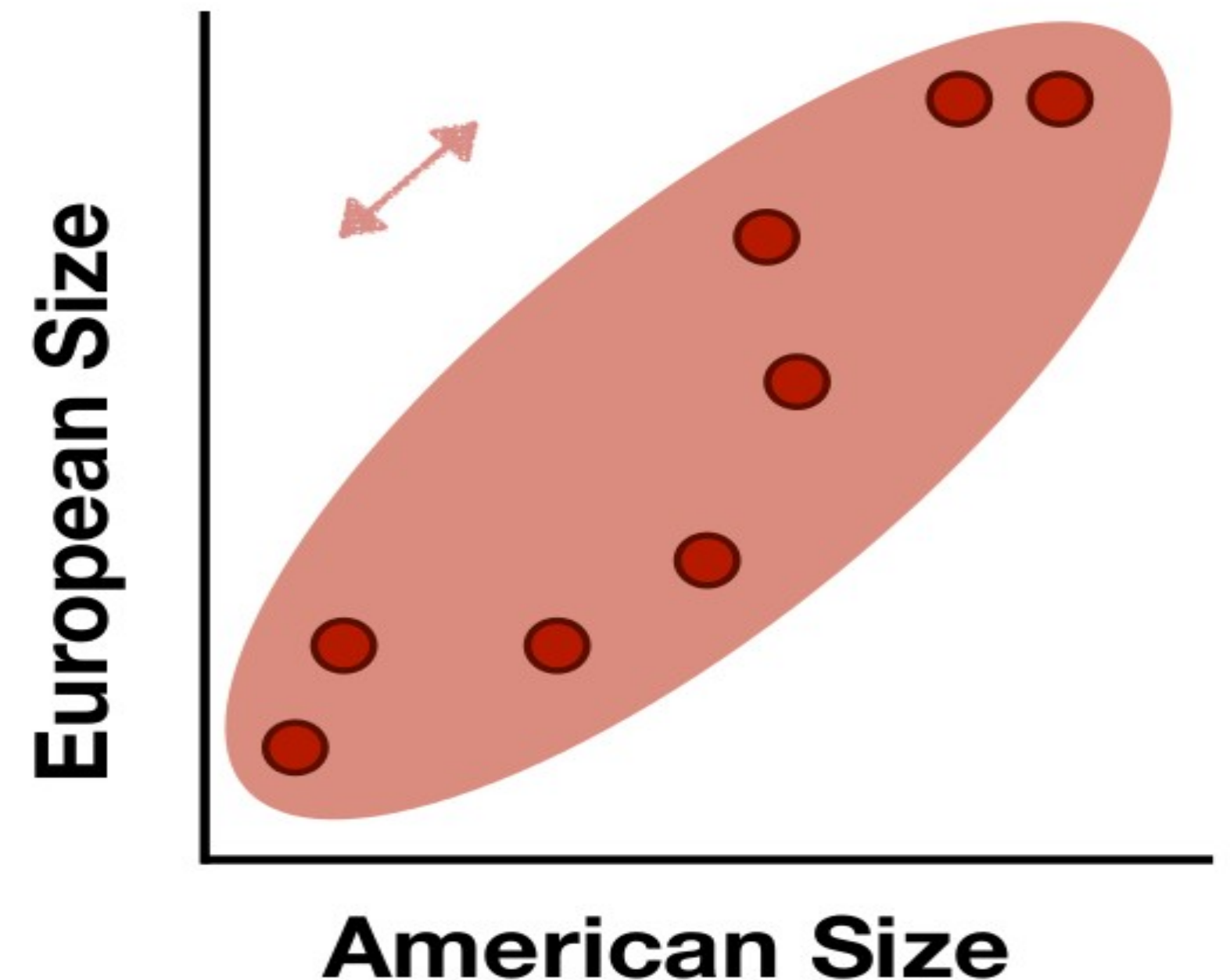


En **teoría de probabilidad**, la varianza de una **variable aleatoria** es una **medida de dispersión** definida como la **esperanza** del cuadrado de la desviación de dicha variable respecto a su media.

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?



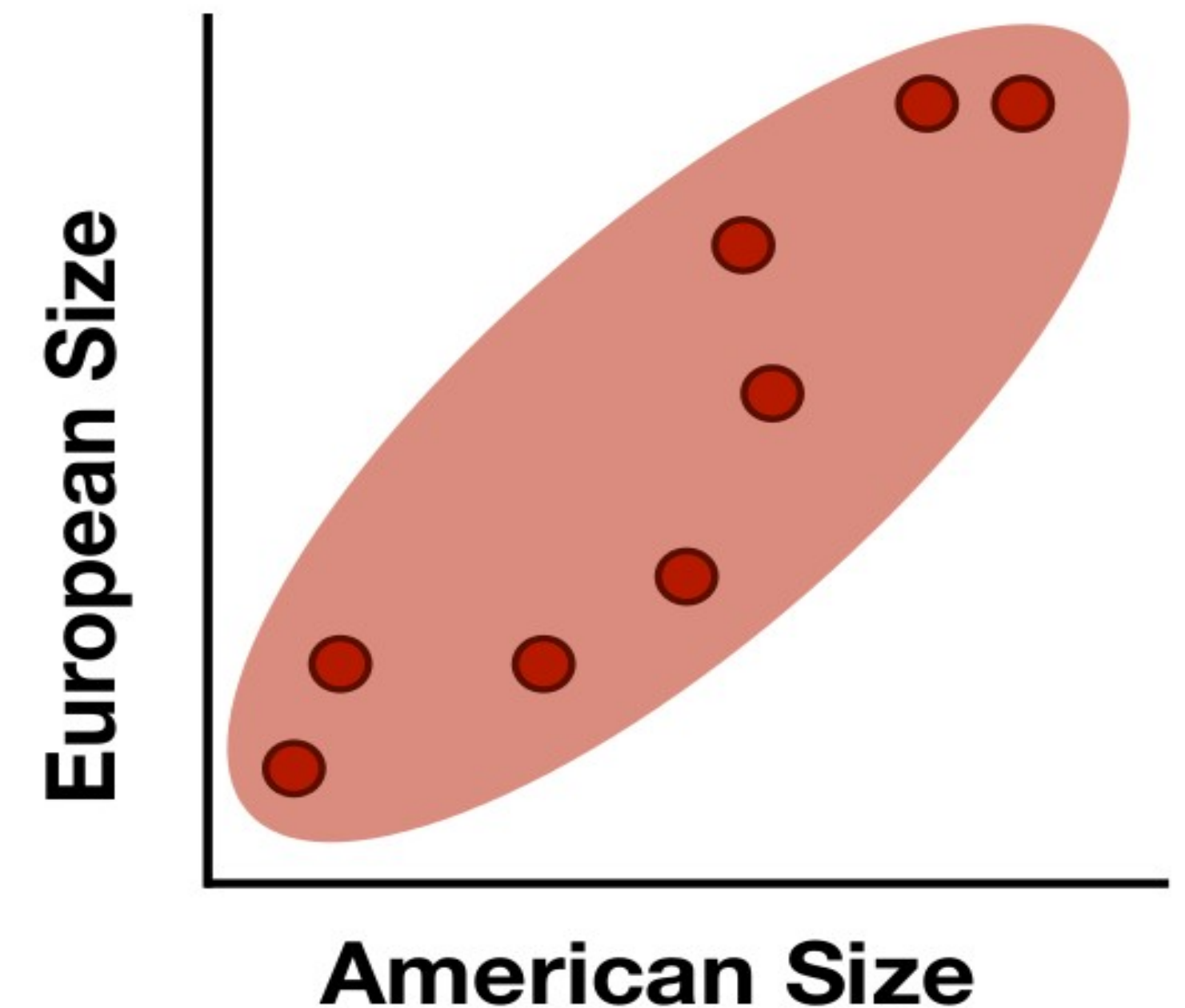
Another Goal: Maximize Variance

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

Linearity assumption ($\mathbf{Z} = \mathbf{XP}$) 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
- $x_j^{(i)}$: j th feature for i th point
- μ_j : mean of j th 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
- $x_j^{(i)}$: j th feature for i th point
- μ_j : mean of j th 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 \rightarrow uncorrelated
- Large magnitude \rightarrow (anti) correlated / redundant
- $\sigma_{12} = \sigma_1^2 = \sigma_2^2 \rightarrow$ features are the same

Covariance Matrix

Covariance matrix generalizes this idea for many features

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

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

- i th diagonal entry equals variance of i th feature
- ij th entry is covariance between i th and j th features
- Symmetric (makes sense given definition of covariance)

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

$$\begin{bmatrix} 2 & -1 & -1 \\ 3 & 2 & -5 \end{bmatrix} \begin{bmatrix} 2 & 3 \\ -1 & 2 \\ -1 & -5 \end{bmatrix} = \begin{bmatrix} 6 & \end{bmatrix}$$

$\mathbf{X}^\top \qquad \qquad \mathbf{X} \qquad \qquad \mathbf{X}^\top \mathbf{X}$

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

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

$$\begin{bmatrix} 2 & -1 & -1 \\ 3 & 2 & -5 \end{bmatrix} \begin{bmatrix} 2 & 3 \\ -1 & 2 \\ -1 & -5 \end{bmatrix} = \begin{bmatrix} 6 & 9 \\ 9 & 38 \end{bmatrix}$$

$\mathbf{X}^\top \qquad \qquad \mathbf{X} \qquad \qquad \mathbf{X}^\top \mathbf{X}$

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

\mathbf{P} equals the top k eigenvectors of $\mathbf{C}_\mathbf{X}$

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

PCA Solution

All covariance matrices have an eigendecomposition

- $\mathbf{C}_X = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ (eigendecomposition)
- \mathbf{U} is $d \times d$ (columns 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`

En **álgebra lineal**, los vectores propios, autovectores o eigenvectores de un **operador lineal** son los vectores no nulos que, cuando son transformados por el operador, dan lugar a un múltiplo escalar de sí mismos, con lo que no cambian su dirección. Este escalar recibe el nombre valor propio

Un conjunto de vectores es ortonormal si es a la vez un conjunto ortogonal y la norma de cada uno de sus vectores es igual a 1

dos vectores $x \in V$ e $y \in V$ son ortogonales si el producto escalar de $\langle x, y \rangle$ es cero

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



$\mathbf{a} = \begin{bmatrix} 1 & 0 \end{bmatrix}^\top$ $\mathbf{b} = \begin{bmatrix} 0 & 1 \end{bmatrix}^\top$ $\mathbf{c} = \begin{bmatrix} 1 & 1 \end{bmatrix}^\top$ $\mathbf{d} = \begin{bmatrix} 2 & 0 \end{bmatrix}^\top$

Orthonormal vectors are orthogonal and have unit norm

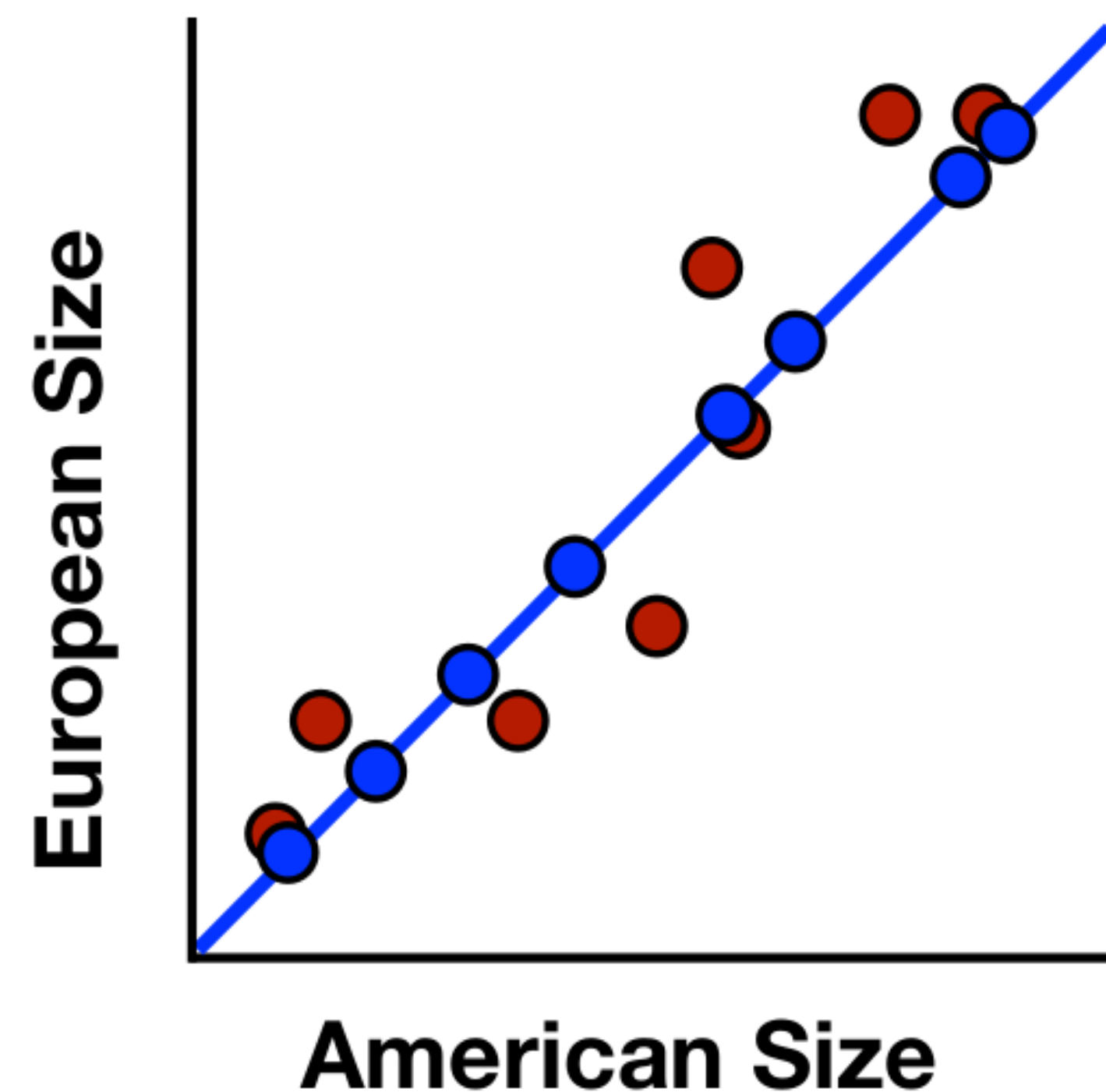
- \mathbf{a} and \mathbf{b} are orthonormal, but \mathbf{b} and \mathbf{d} are not orthonormal

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 $\{1, \dots, k\}$:

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



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 Para obtener de la matriz de covarianza los k autovectores que van en P

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

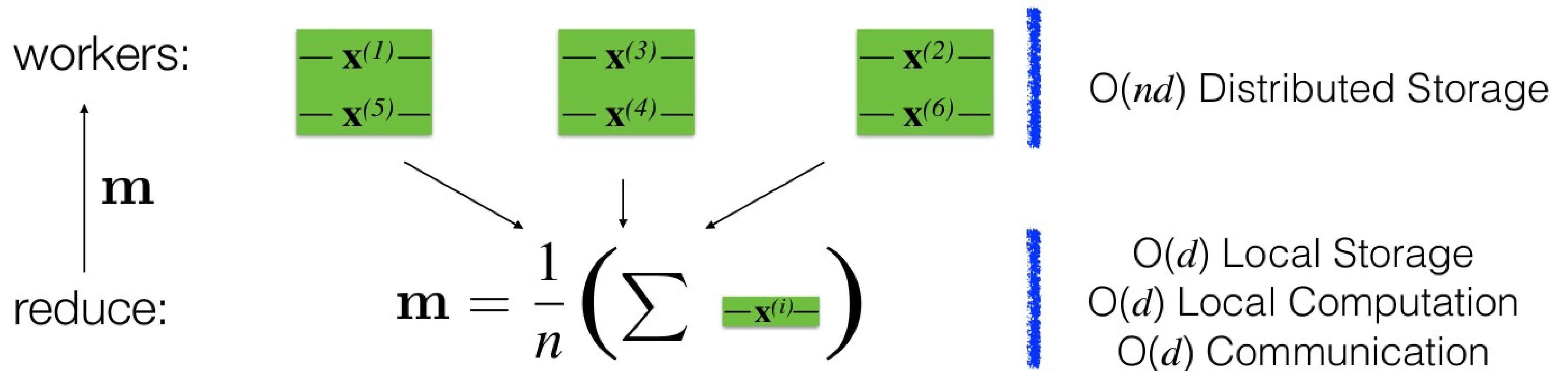
- $O(dk + n)$ local storage, computation;
 $O(dk + n)$ communication
- Iterative algorithm

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

Step 1: Center Data

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

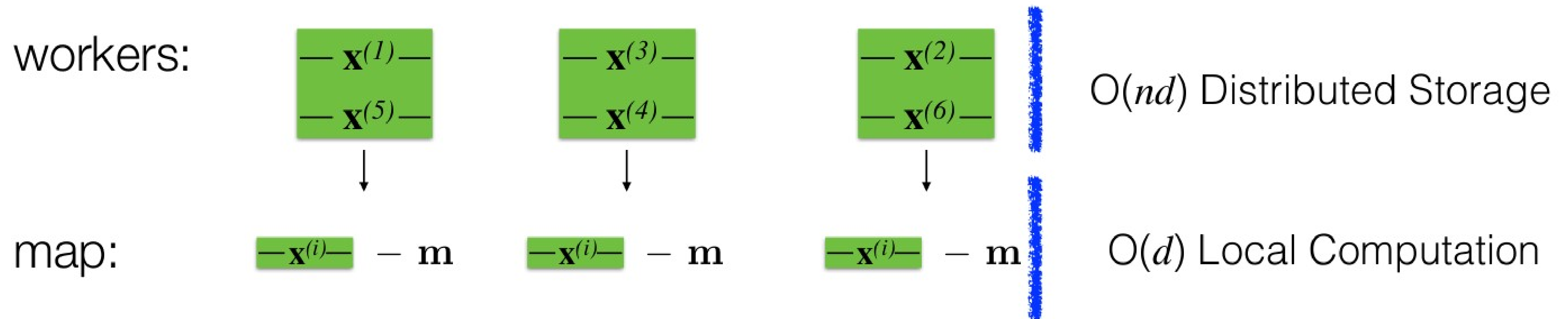
Example: $n = 6$; 3 workers



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



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!)

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

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

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!)

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

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

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!)

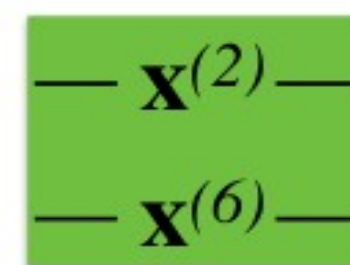
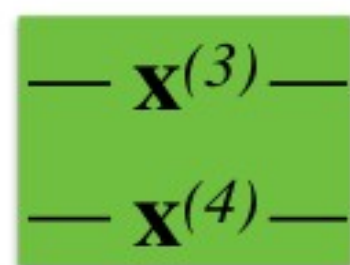
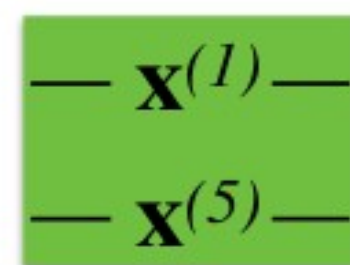
$$\begin{bmatrix} 9 & 3 & 5 \\ 4 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & -5 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} 28 & 18 \\ 11 & 9 \end{bmatrix}$$

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

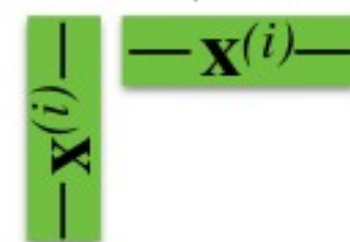
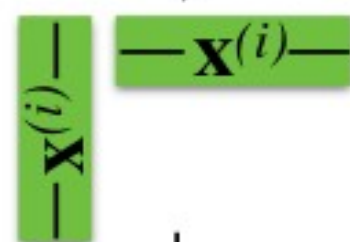
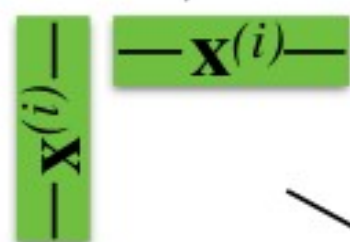
$$\mathbf{X}^\top \mathbf{X} = \begin{matrix} & \begin{matrix} \text{---} \mathbf{x}^{(1)} \text{---} \\ \text{---} \mathbf{x}^{(2)} \text{---} \\ \vdots \\ \text{---} \mathbf{x}^{(n)} \text{---} \end{matrix} \\ \begin{matrix} d \\ n \end{matrix} & \end{matrix} = \sum_{i=1}^n \begin{matrix} \text{---} \mathbf{x}^{(i)} \text{---} \\ \text{---} \mathbf{x}^{(i)} \text{---} \end{matrix}$$

Example: $n = 6$; 3 workers

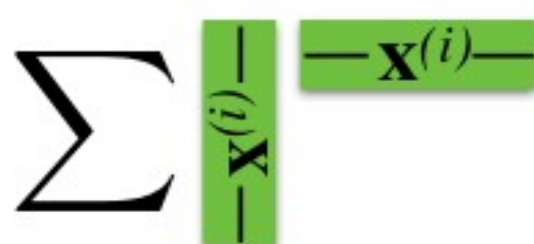
workers:



map:



reduce:



$O(nd)$ Distributed Storage

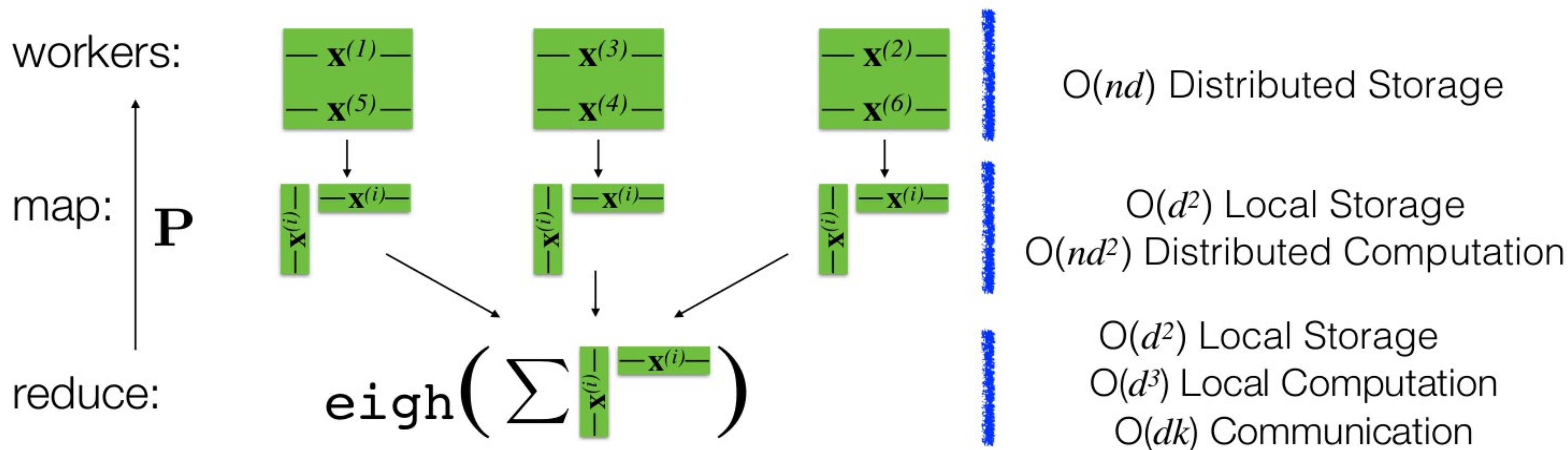
$O(d^2)$ Local Storage
 $O(nd^2)$ Distributed Computation

$O(d^2)$ 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

workers:

— $\mathbf{x}^{(1)}$ —
— $\mathbf{x}^{(5)}$ —

— $\mathbf{x}^{(3)}$ —
— $\mathbf{x}^{(4)}$ —

— $\mathbf{x}^{(2)}$ —
— $\mathbf{x}^{(6)}$ —

$O(nd)$ Distributed Storage

map:

— $\mathbf{p}^{(1)}$ —
— $\mathbf{p}^{(2)}$ —

— $\mathbf{x}^{(i)}$ —

— $\mathbf{p}^{(1)}$ —
— $\mathbf{p}^{(2)}$ —

— $\mathbf{x}^{(i)}$ —

— $\mathbf{p}^{(1)}$ —
— $\mathbf{p}^{(2)}$ —

— $\mathbf{x}^{(i)}$ —

$O(dk)$ Local Computation