# Unsupervised Learning

George Konidaris
gdk@cs.brown.edu

**Fall 2021**

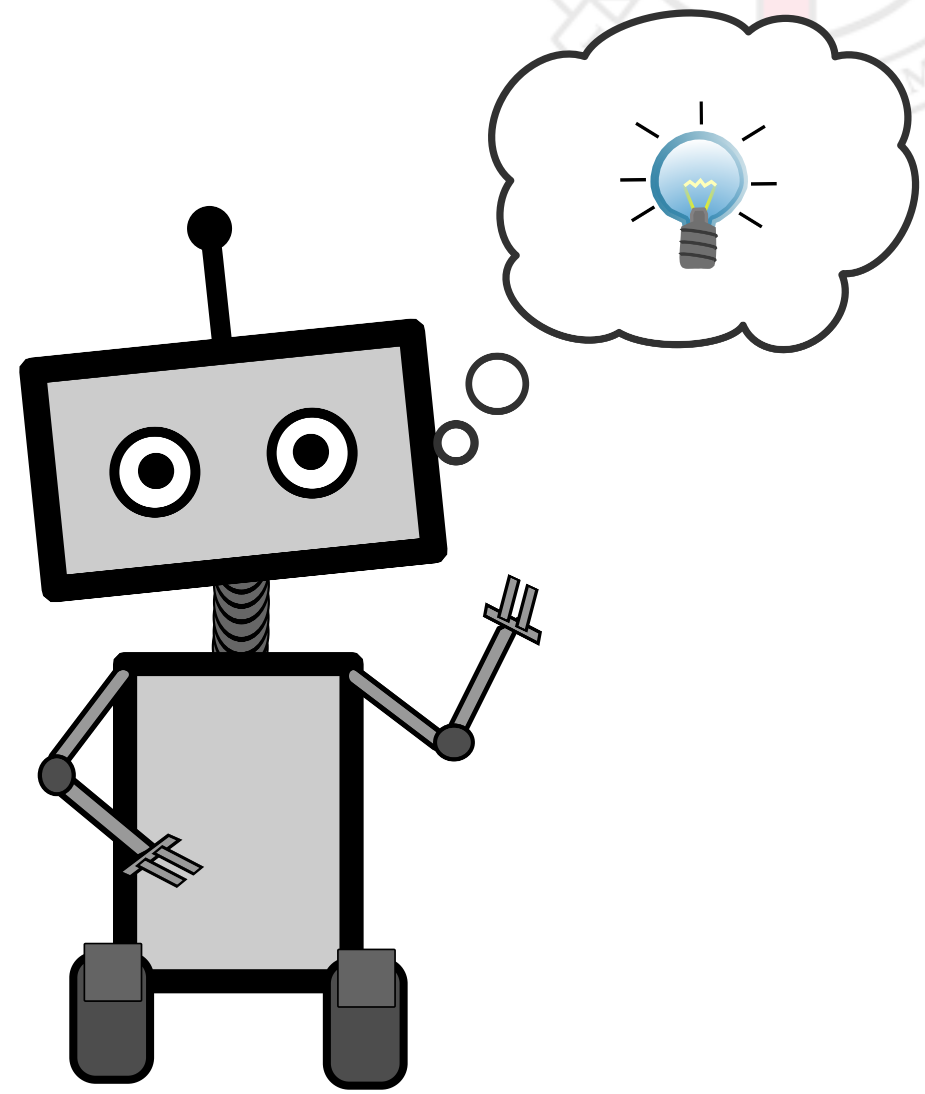
# Machine Learning

Subfield of AI concerned with *learning from data*.

Broadly, using:

- ***Experience***
- To Improve ***Performance***
- On Some ***Task***

*(Tom Mitchell, 1997)*



# Unsupervised Learning

Input:

$X = \{x_1, \ldots, x_n\}$ inputs

Try to understand the *structure of the data.*

*E.g., how many types of cars? How can they vary?*

# Clustering

One particular type of unsupervised learning:

- Split the data into discrete clusters.
- Assign new data points to each cluster.
- Clusters can be thought of as *types*.

**Formal definition**

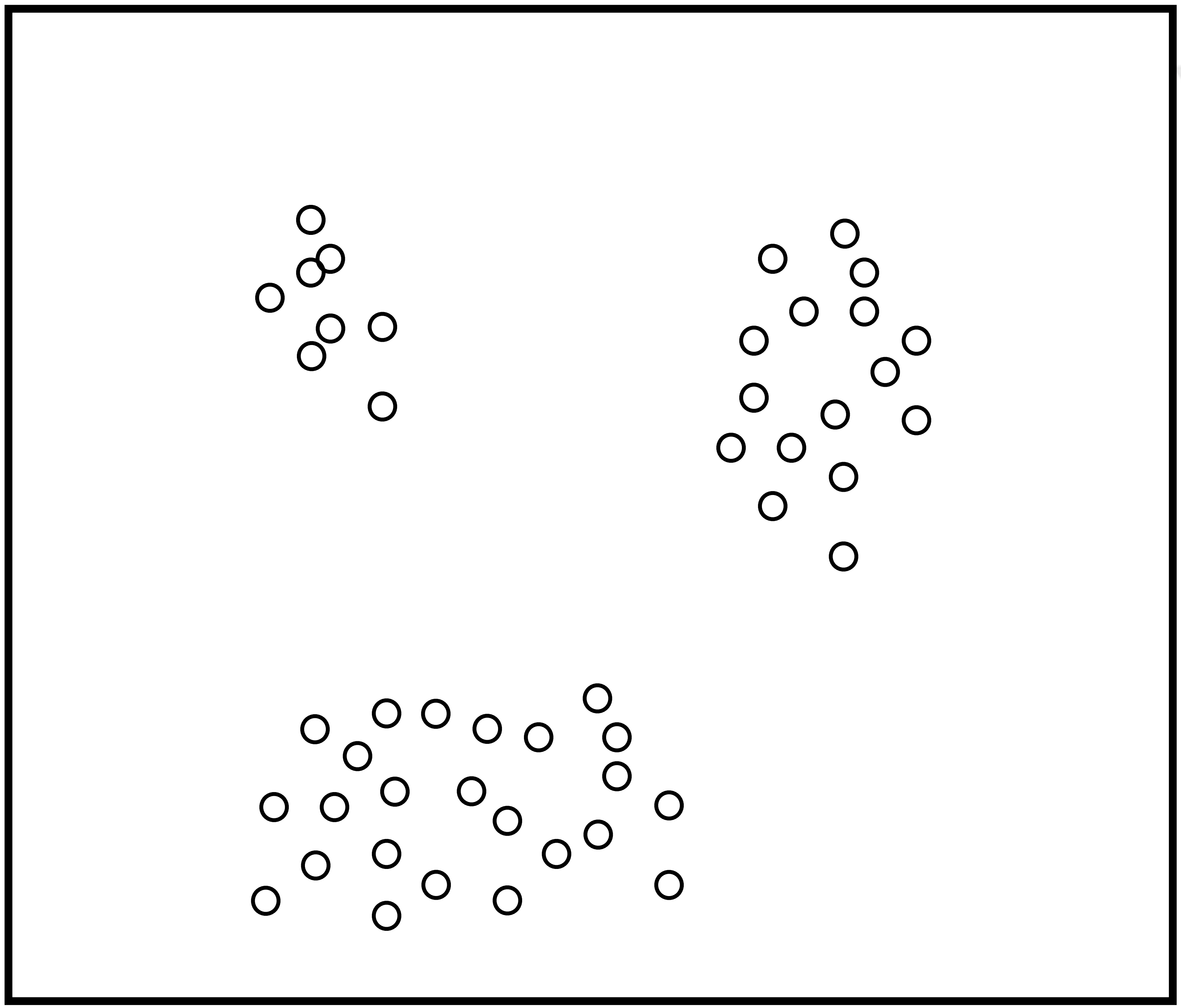
Given:

- Data points $X = \{x_1, \ldots, x_n\}$.

Find:

- Number of clusters $k$
- Assignment function $f(x) = \{1, \ldots, k\}$

# Clustering

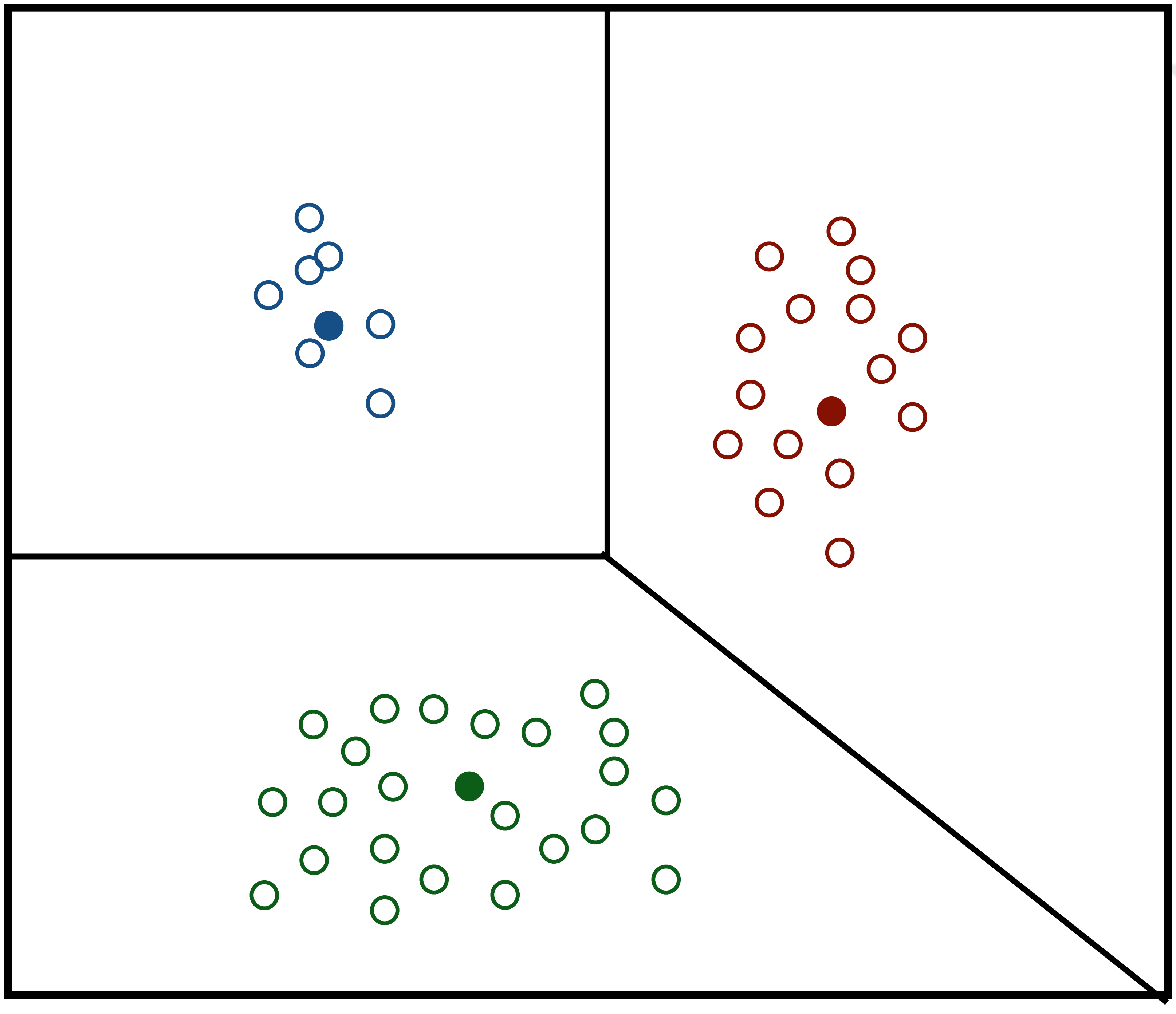


# k-Means

One approach:

- Pick *k*
- Place *k* points ("means") in the data
- Assign new point to *i*th cluster if nearest to *i*th "mean".

# k-Means



# k-Means

Major question:

- *Where to put the "means"?*

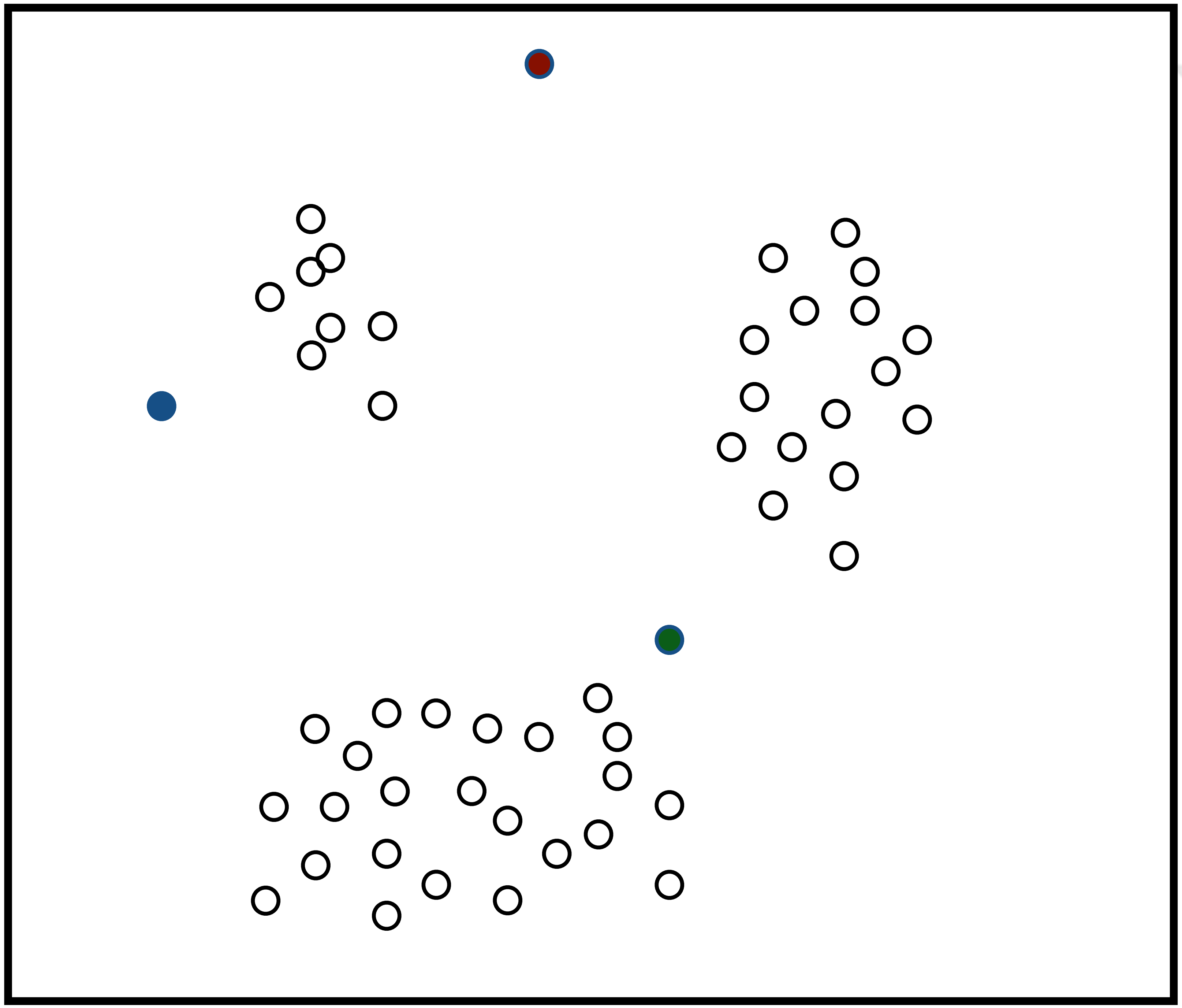
Very simple algorithm:

- Place k "means" $\{\mu_1, ..., \mu_k\}$ at random.
- Assign all points in the data to each "mean"
  $f(x_j) = i \text{ such that } d(x_j, \mu_i) \leq d(x_j, \mu_l) \forall l \neq i$

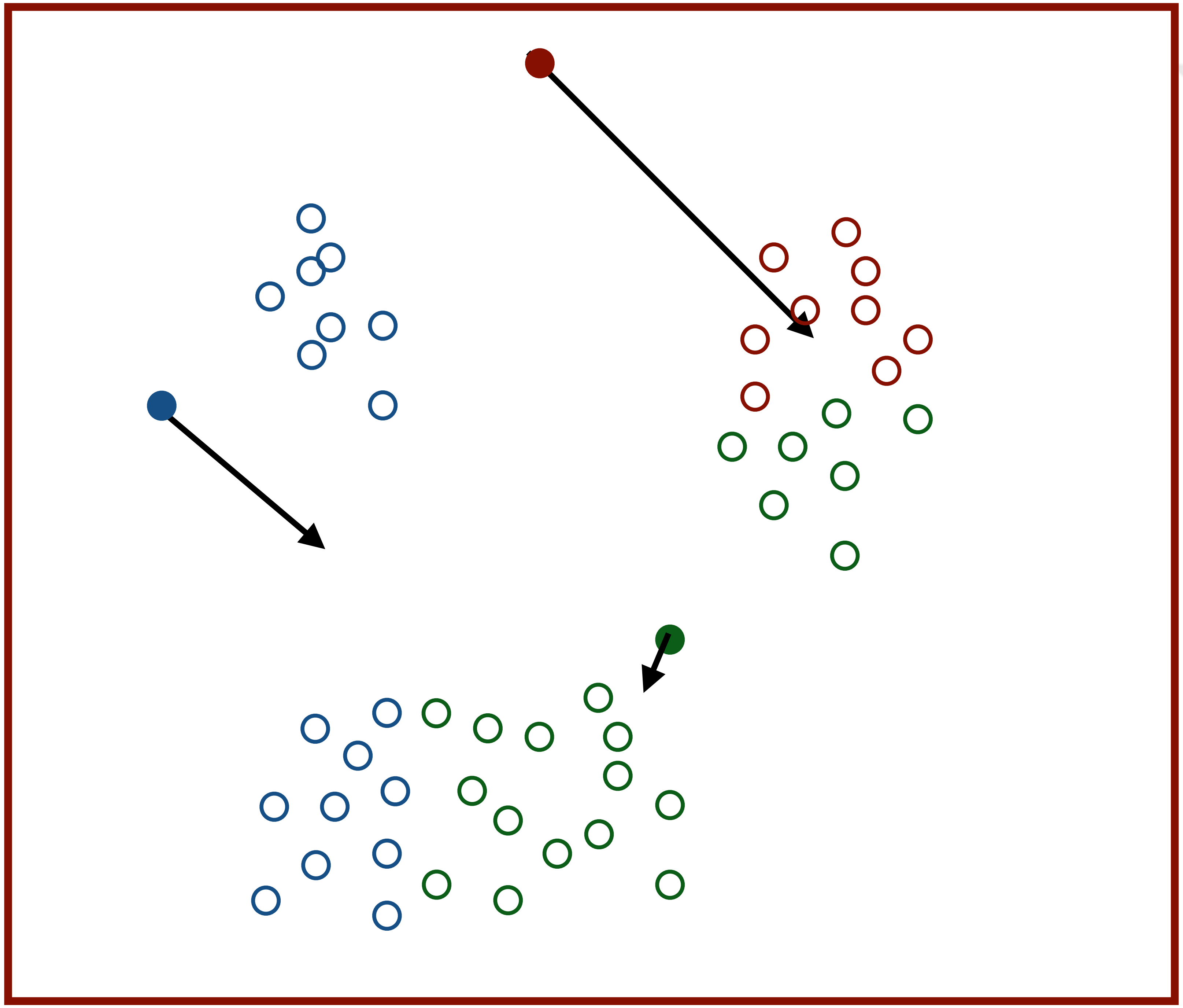
- Move each "mean" to mean of assigned data.

$$\mu_i = \sum_{v \in C_i} \frac{x_v}{|C_i|}$$

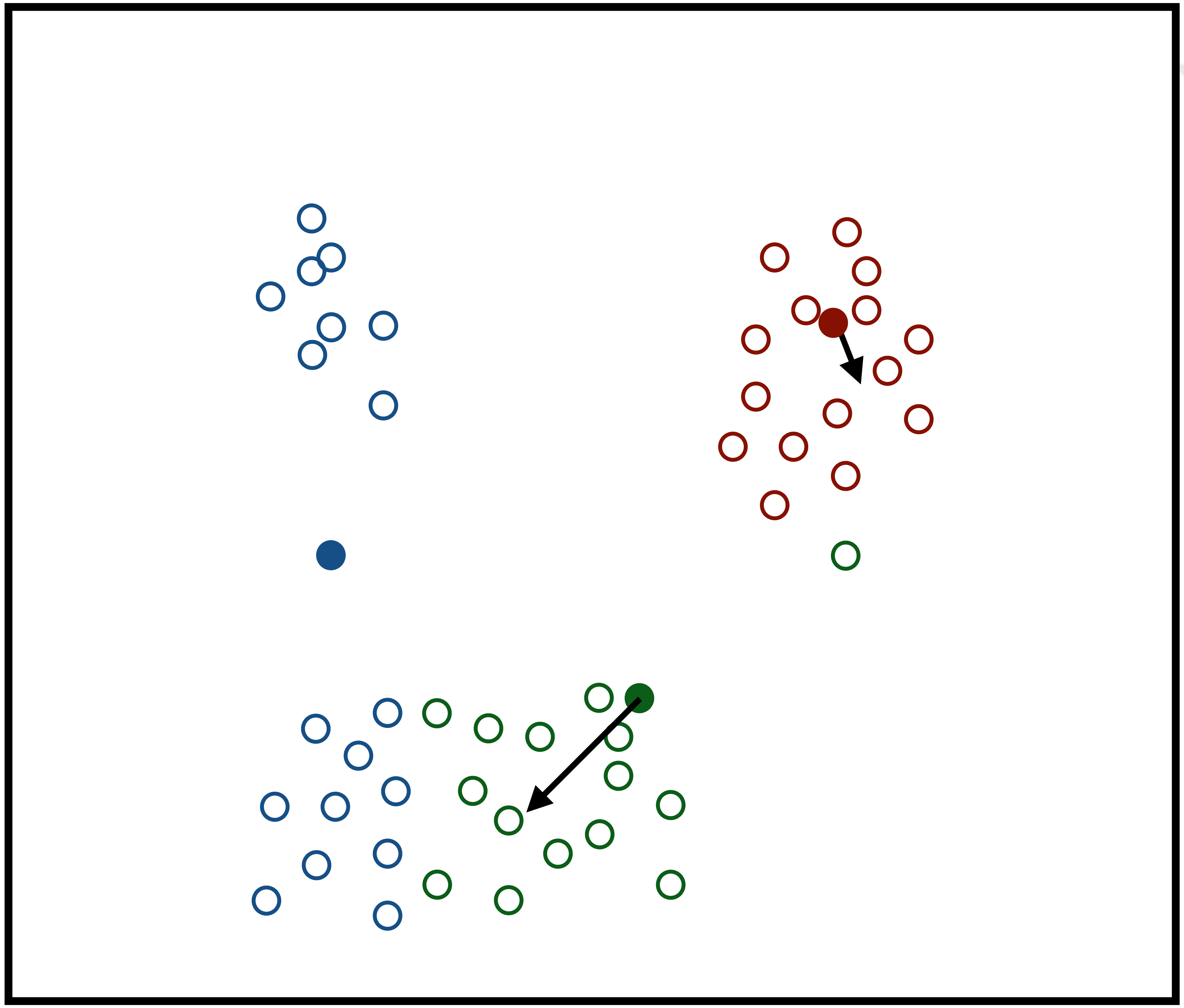
# k-Means



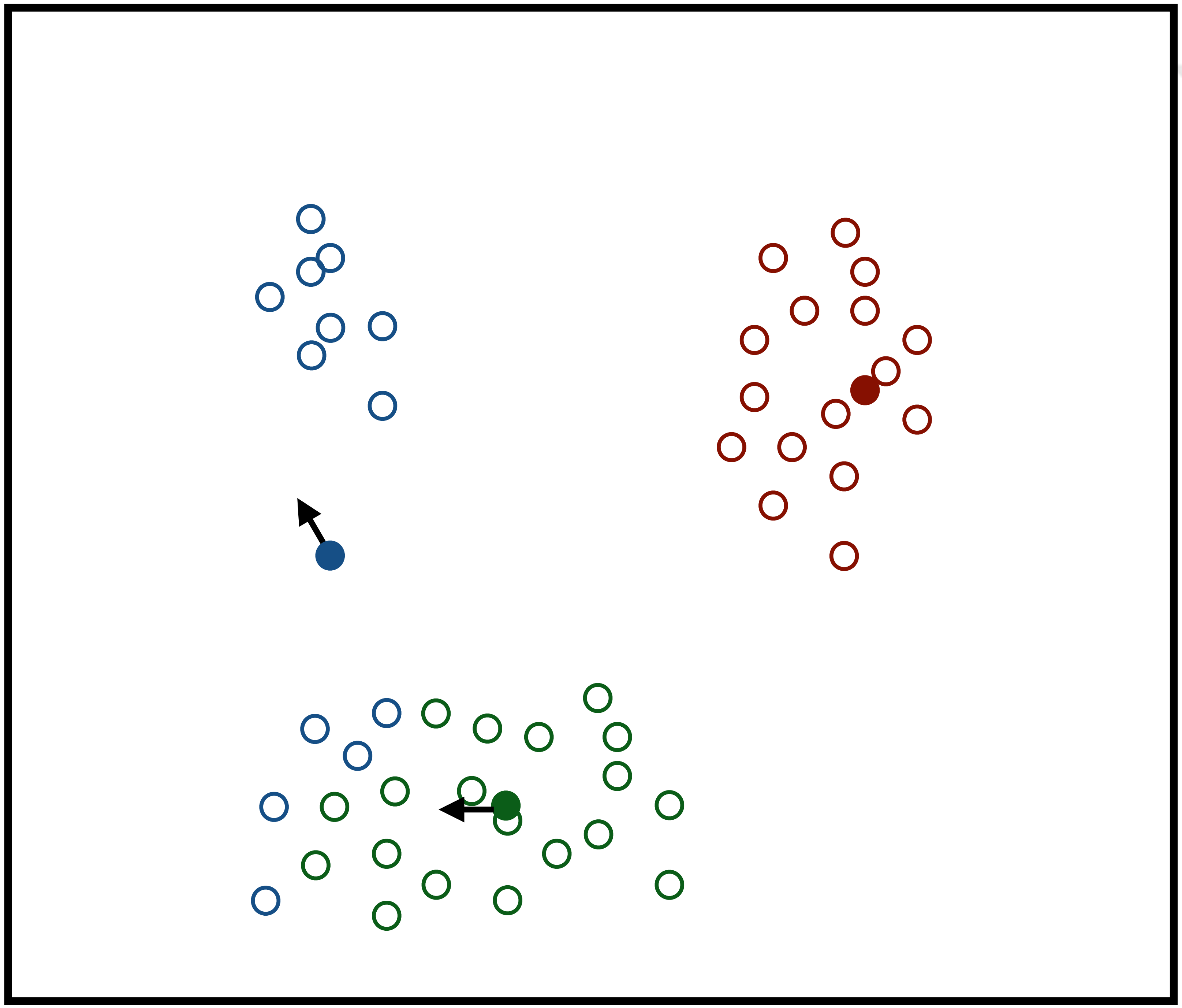
# k-Means



# k-Means



# k-Means



# k-Means

Remaining questions …

How to choose *k*?

What about bad initializations?

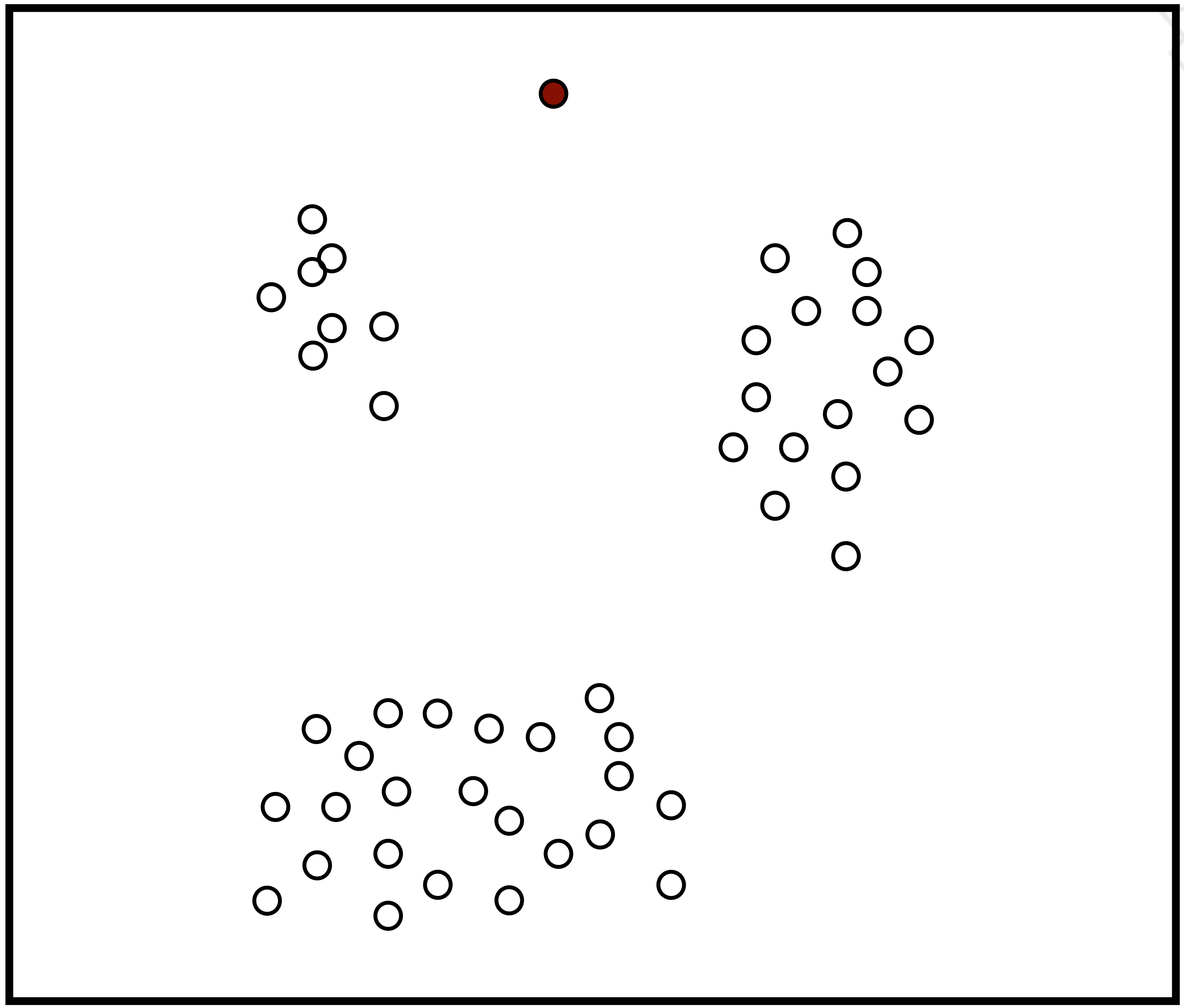
How to measure distance?

Broadly:

- Use a quality metric.
- Loop through *k*.
- Random restart initial position.
- Use distance metric *D*.

# Density Estimation

Clustering: can answer *which cluster,* but not *does this belong*?



# Density Estimation

Estimate the *distribution the data is drawn from*.

This allows us to evaluate the probability that a new point is drawn from the same distribution as the old data.

**Formal definition**

Given:

- Data points $X = \{x_1, \ldots, x_n\}$,

Find:

- PDF $P(X)$

# GMM

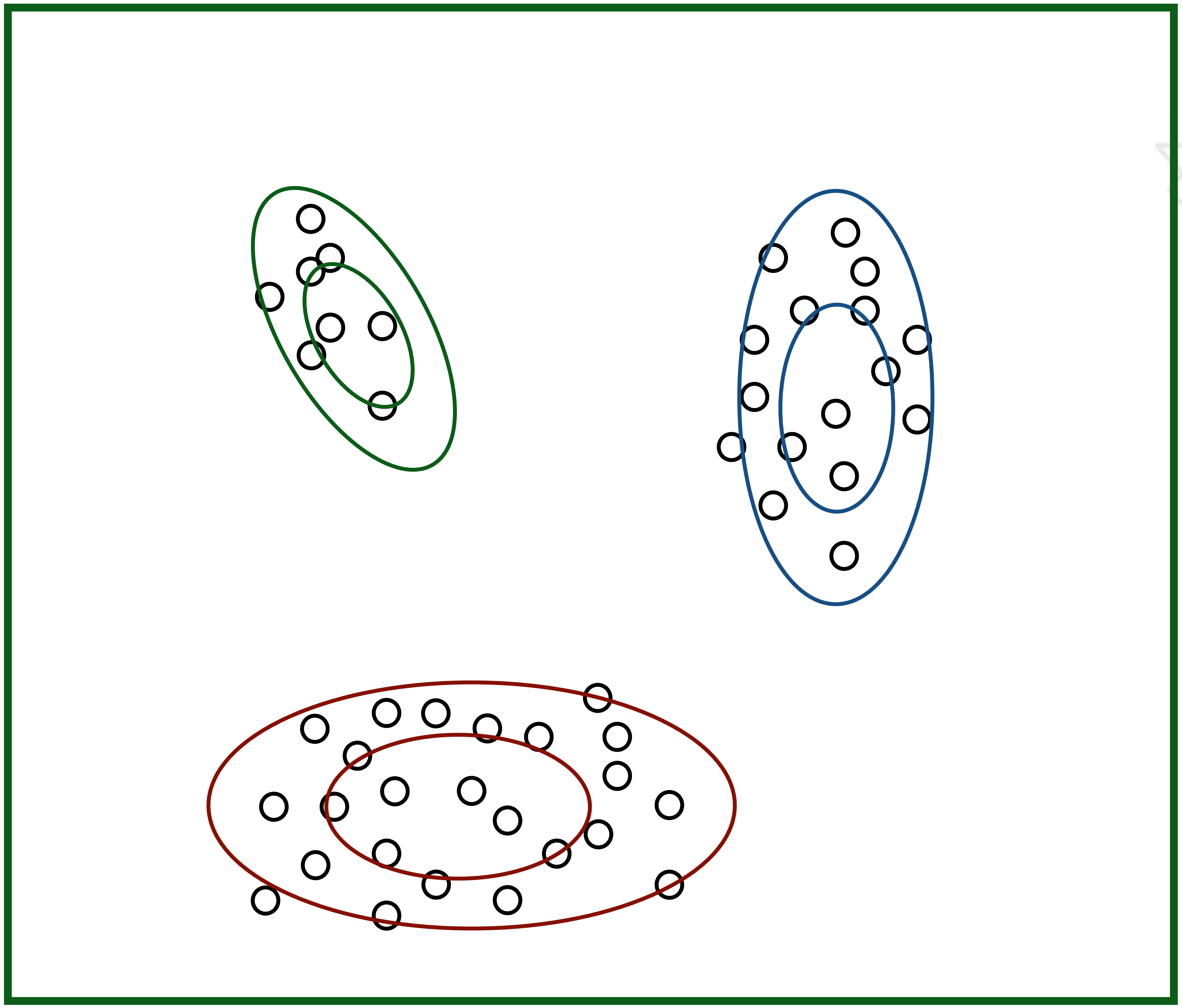
Simple approach:

- Model the data as a mixture of Gaussians.

Each Gaussian has its own mean and variance.
Each has its own *weight* (sum to 1).

**Weighted sum of Gaussians still a PDF.**

# GMM



# GMM

Algorithm - broadly as before:

- Place *k* “means” $\{\mu_1, ..., \mu_k\}$ at random.
- Set variances to be high.

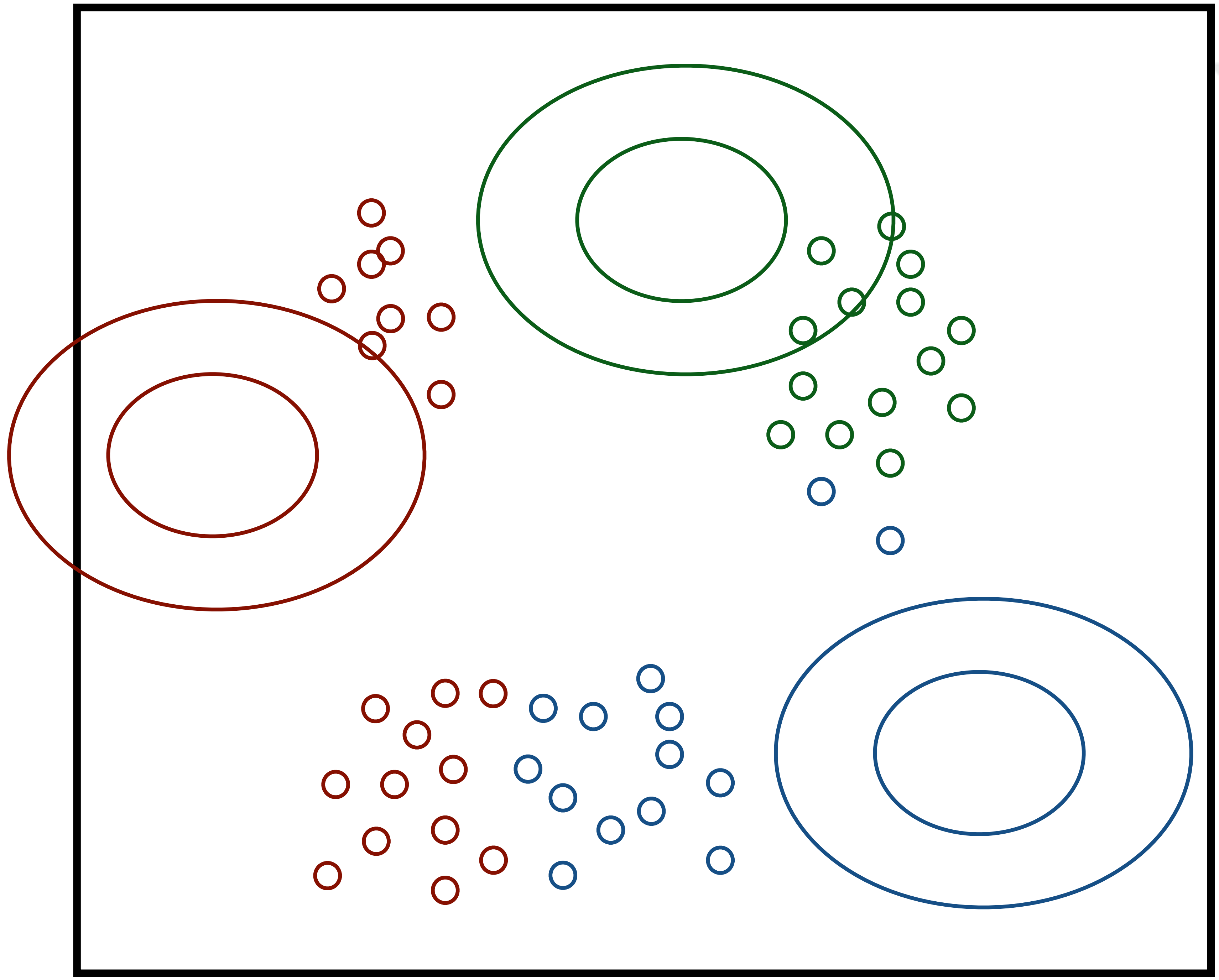
- Assign all points to highest probability distribution.

$$C_i = \{x_v | N(x_v|\mu_i, \sigma_i^2) > N(x_v|\mu_j, \sigma_j^2), \forall j\}$$

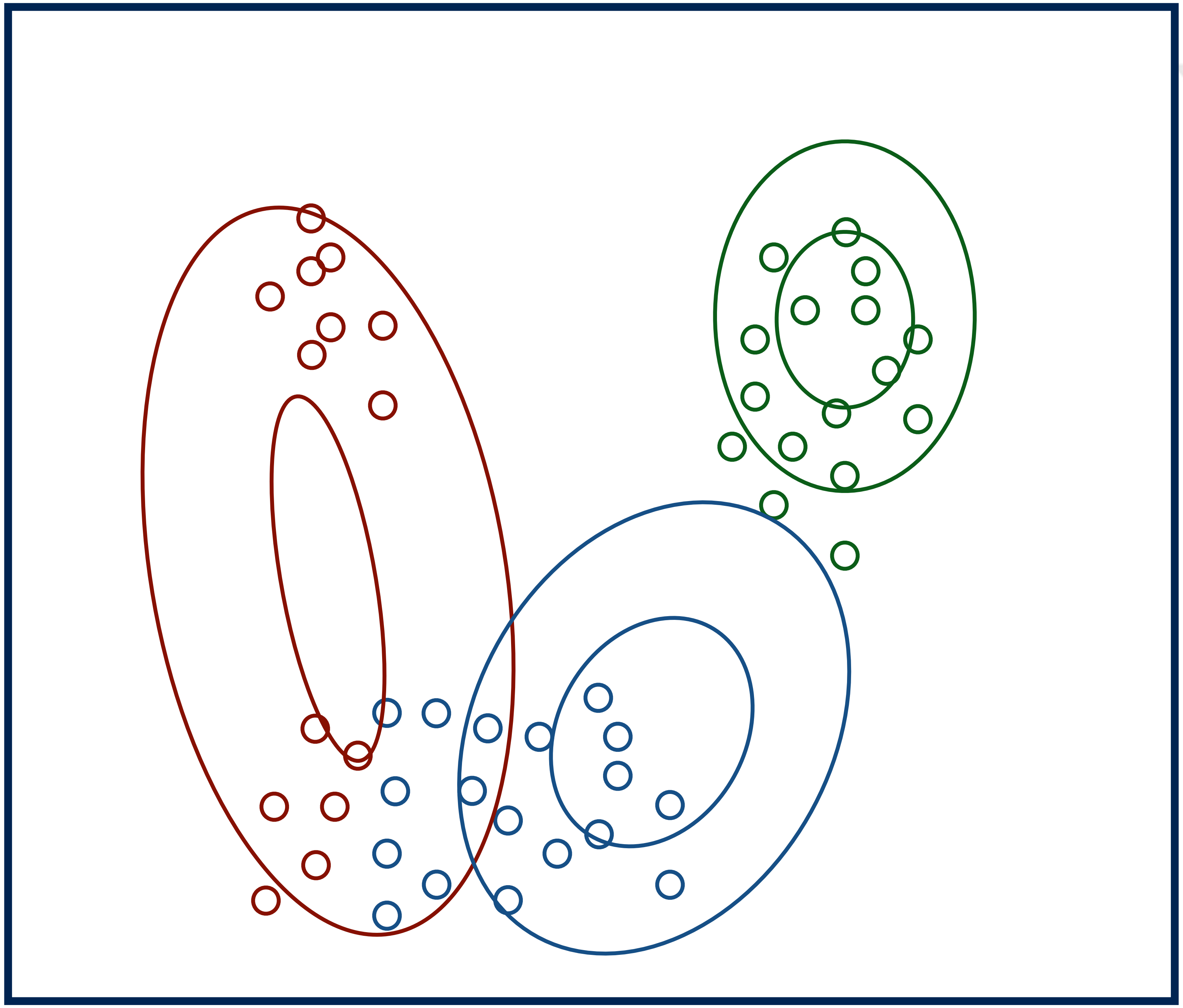
- Set mean, variance, weights to match assigned data.

$$\mu_i = \sum_{v \in C_i} \frac{x_v}{|C_i|} \qquad \sigma_i^2 = \text{variance}(C_i) \qquad w_i = \frac{|C_i|}{\sum_j |C_j|}$$

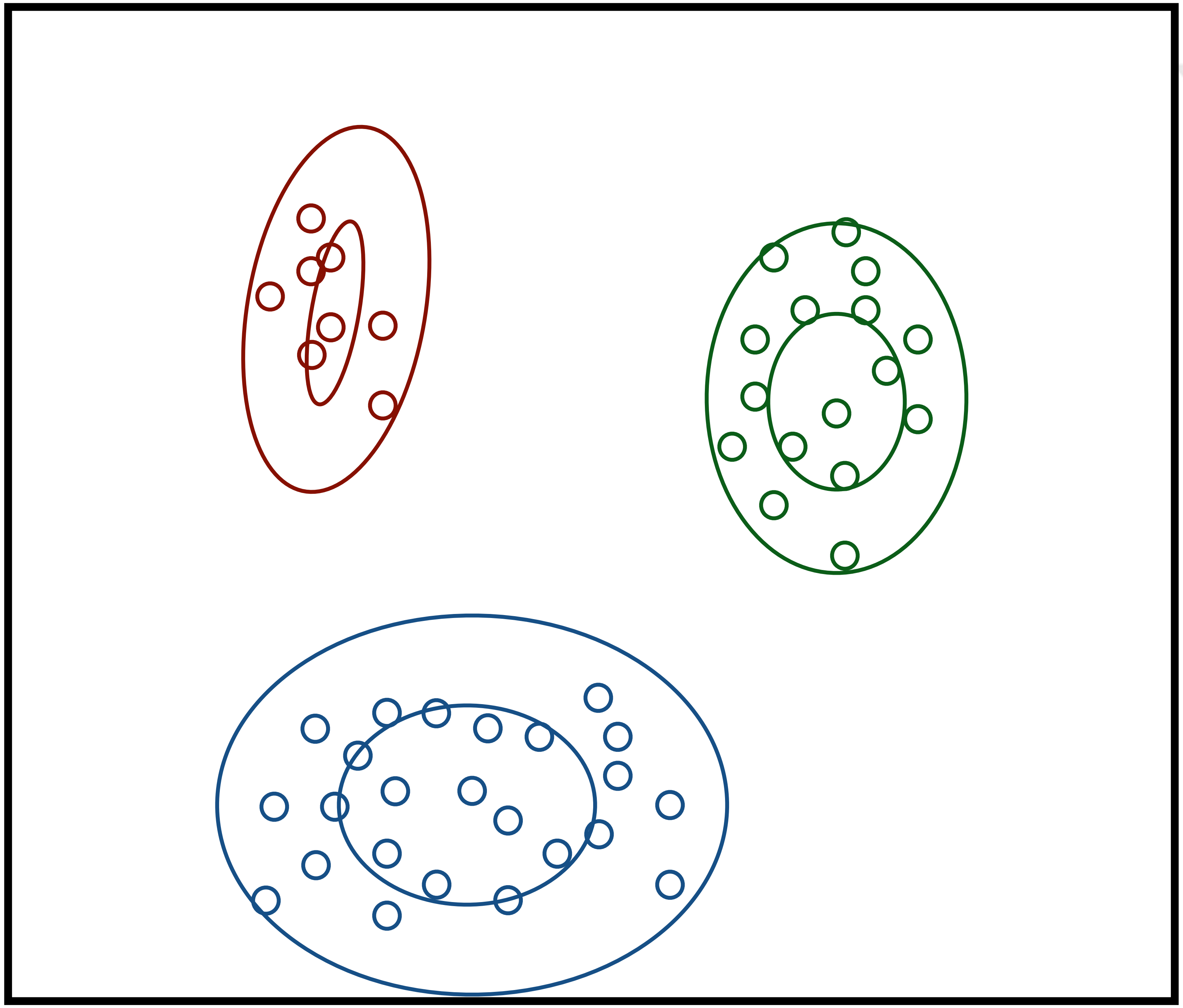
# GMM



# GMM



# GMM



# GMM

Major issue:

- How to decide between two GMMs?
- How to choose *k*?

General statistical question: model selection.
Several good answers for this.

Simple example: **Bayesian information criterion (BIC).**
Trades off model complexity (k) with fit (likelihood).

$$-2 \log L + k \log n$$

likelihood

# parameters in model

# data points

# Nonparametric Density Estimation

Parametric:

- Define a parametrized model (e.g., a Gaussian)
- Fit parameters
- Done!

**Key assumptions**:

- Data is distributed according to the parametrized form.
- We know *which* parametrized form in advance.

**What is the shape of the distribution over images representing flowers?**

# Nonparametric Density Estimation

Nonparametric alternative:

- Avoid fixed parametrized form.
- Compute density estimate directly from the data.

Kernel density estimator:

$$PDF(x) = \frac{1}{nb} \sum_{i=1}^{n} D\left(\frac{x_i - x}{b}\right)$$

where:

- *D* is a special kind of distance metric called a kernel.
  - Falls away from zero, integrates to one.
- *b* is bandwidth: controls how fast kernel falls away.

# Nonparametric Density Estimation

$$PDF(x) = \frac{1}{nb} \sum_{i=1}^{n} D\left(\frac{x_i - x}{b}\right)$$

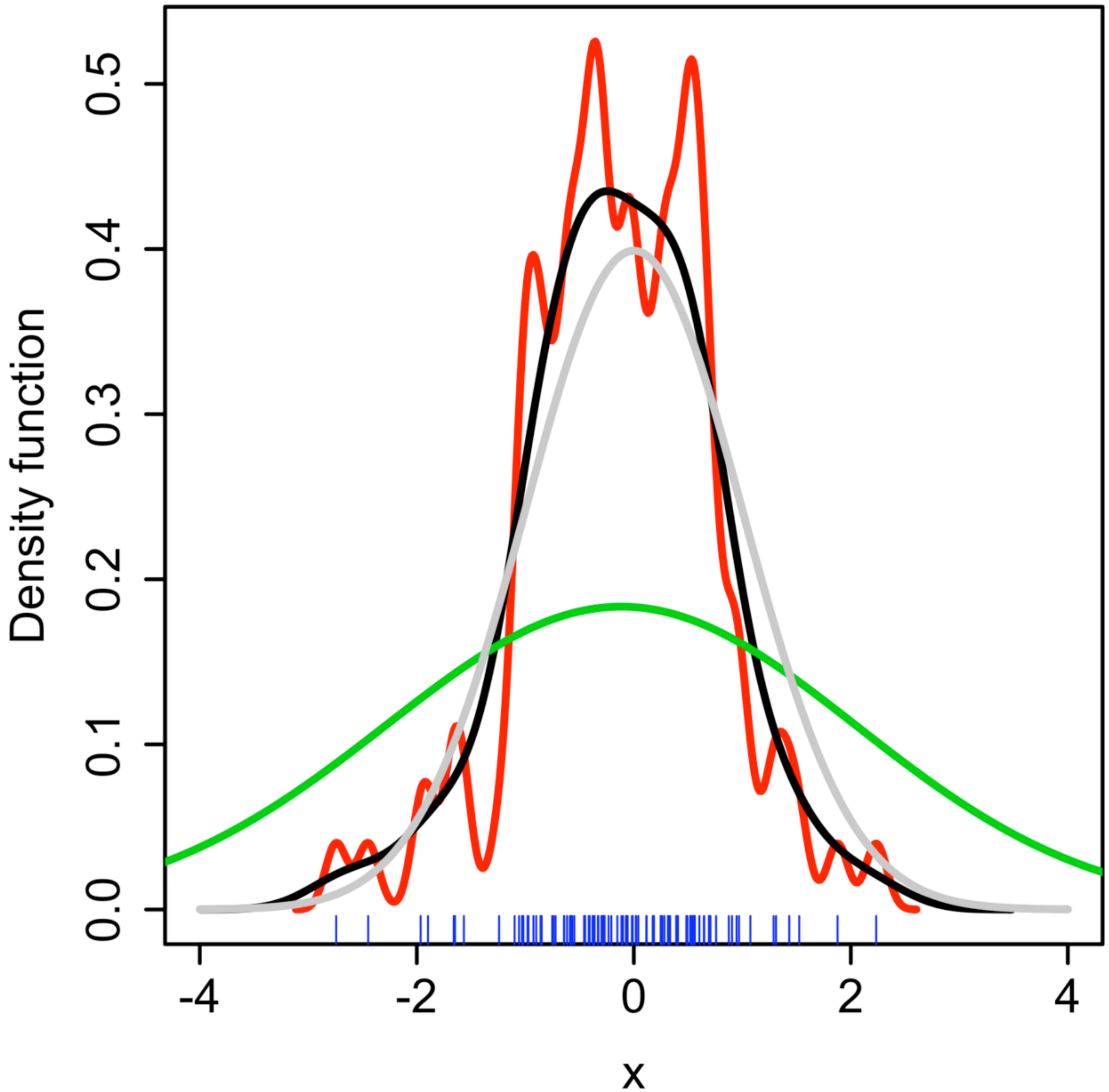
Kernel:

- Lots of choices, Gaussian often works in practice.

Bandwidth:

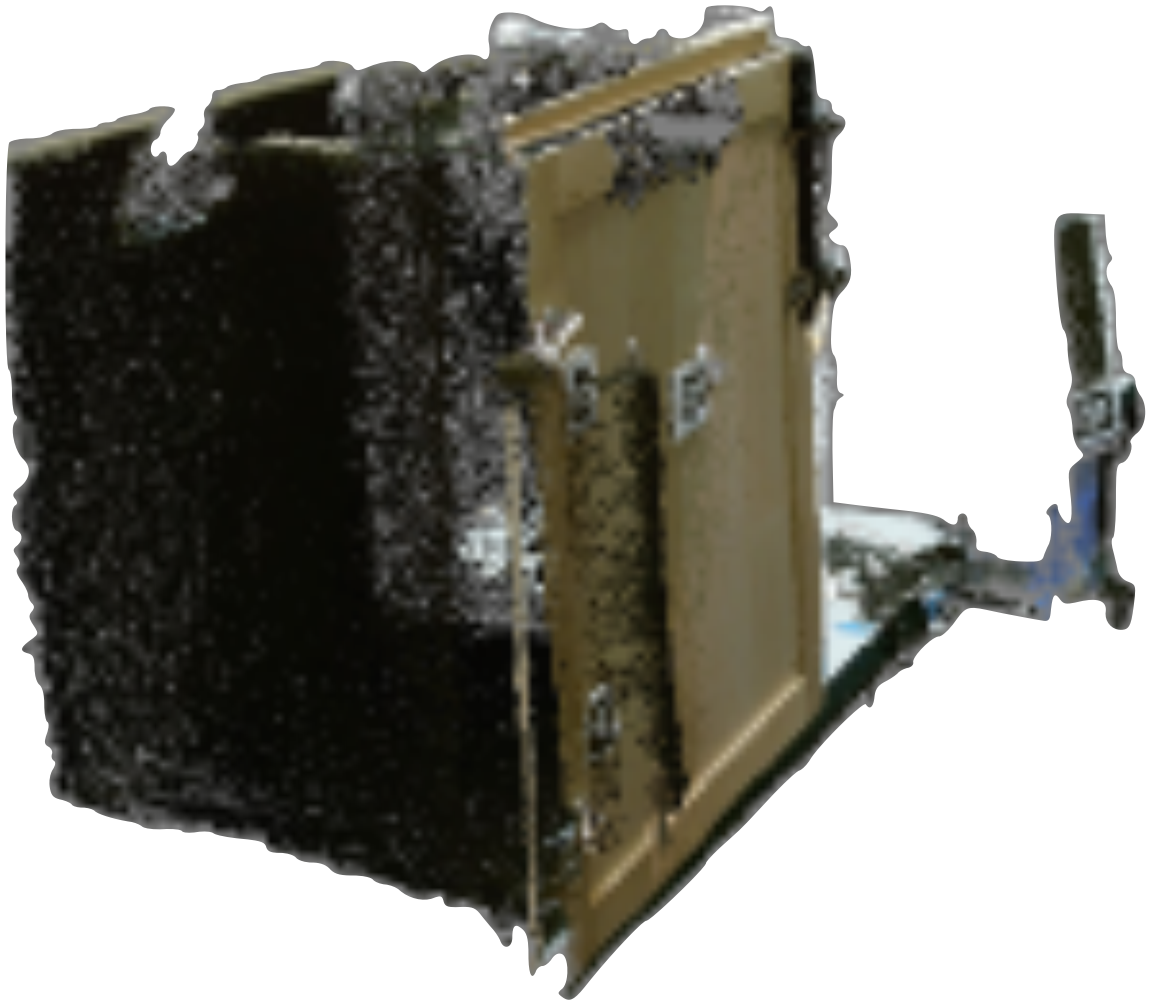
- High: distant points have higher "contribution" to sum.
- Low: distant points have lower.

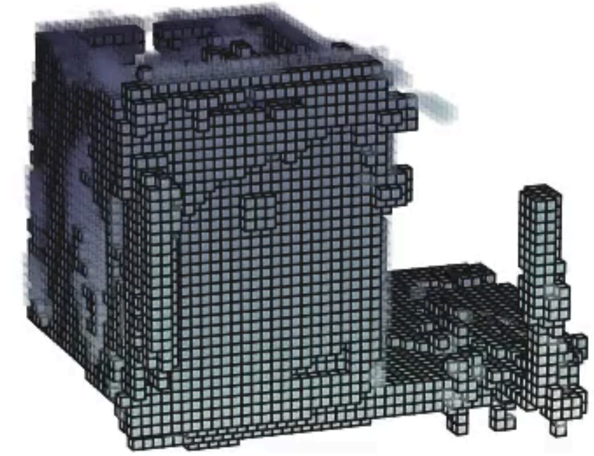
# Nonparametric Density Estimation

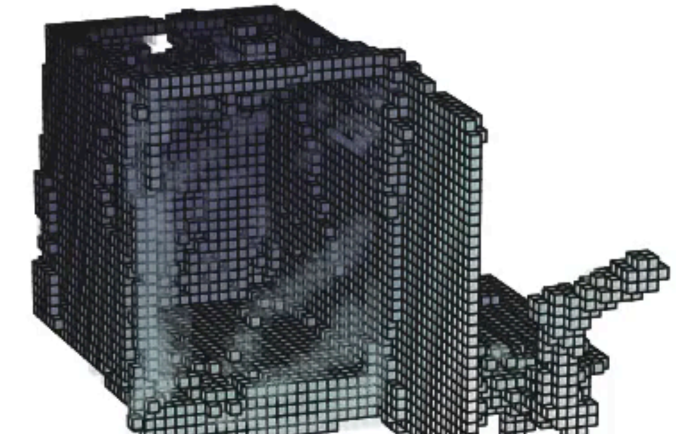



(wikipedia)

# Nonparametric Density Estimator







# Dimensionality Reduction

$X = \{x^1, \ldots, x^n\}$, each $x^i$ has $m$ dimensions: $x^i = [x_1, \ldots, x_m]$.

If $m$ is high, data can be hard to deal with.

- High-dimensional decision boundary.
- Need more data.
- But data is often not really high-dimensional.

**Dimensionality reduction:**

- Reduce or compress the data
- Try not to lose too much!
- Find intrinsic dimensionality

# Dimensionality Reduction

For example, imagine if $x_1$ and $x_2$ are meaningful features, and $x_3$ … $x_m$ are random noise.

What happens to k-nearest neighbors?

What happens to a decision tree?

What happens to the perceptron algorithm?

What happens if you want to do clustering?

# Dimensionality Reduction

Often can be phrased as a projection:

$$f : X \to X'$$

where:

- $|X'| << |X|$
- our goal: retain as much *sample variance* as possible.

Variance captures what *varies within the data*.

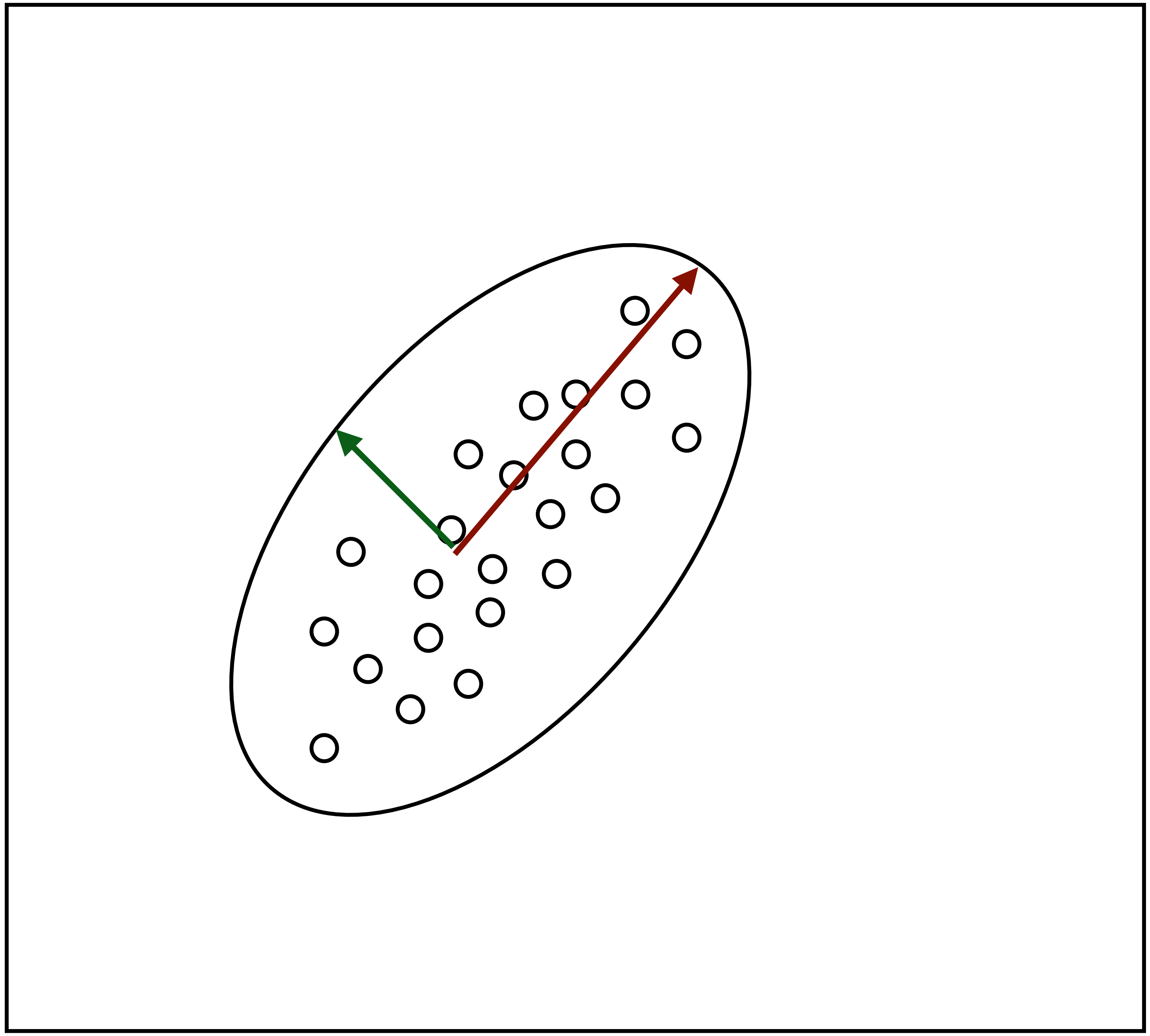
# PCA

**Principle Components Analysis.**

Project data into a new space:

- Dimensions are linearly uncorrelated.
- We have a measure of importance for each dimension.

# PCA



# PCA

- Gather data $x^1, \ldots, x^n$.
- Adjust data to be zero-mean:

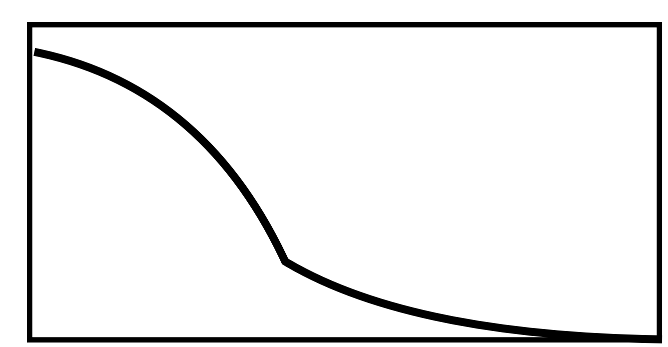
$$x^i = x^i - \sum_j \frac{x^j}{n}$$

- Compute covariance matrix C (*m* x *m*).
- Compute unit eigenvectors $V_i$ and eigenvalues $v_i$ of C.

Each $V_i$ is a direction, and each $v_i$ is its importance - the amount of the data's variance it accounts for.

New data points:

$$\hat{x}^i = [V_1, \ldots, V_p]x^i$$



# PCA

Let's focus on this equation:

$$\hat{x}^i = \boxed{[V_1, ..., V_p]} x^i$$

compressed data point

$p$ x 1

compression matrix

$p$ x $m$

original data point

$m$ x 1

# PCA

If you want to recover the original data point:

$V = [V_1, ..., V_p]$

$$\bar{x}^i = V^{-1}\hat{x}^i$$

$$\Big\downarrow \quad \textbf{\textit{V} is orthonormal}$$

$$\bar{x}^i = V^T\hat{x}^i$$

so:

$$\bar{x}^i = V_1\hat{x}^i_1 + V_2\hat{x}^i_2 + ... + V_p\hat{x}^i_p$$

# PCA

Reconstruction:

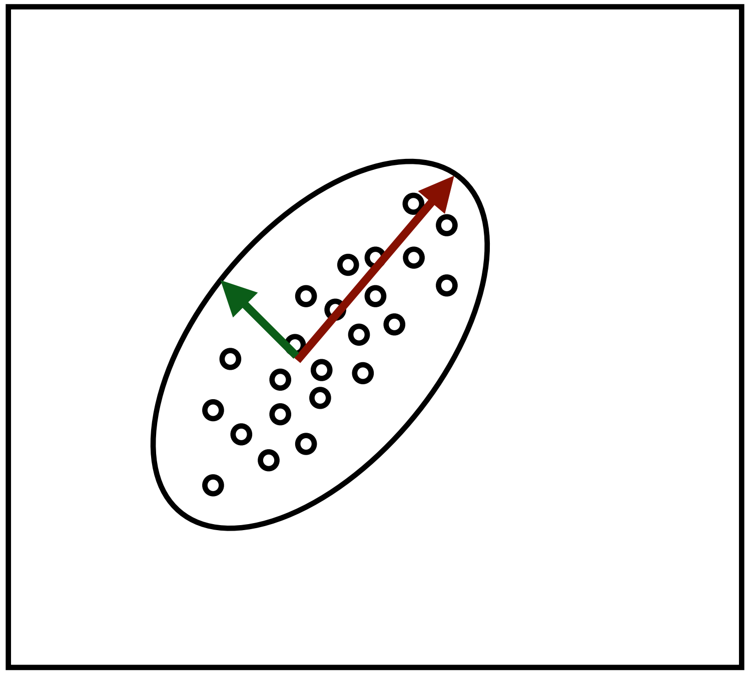
$$\bar{x}^i = V_1 \hat{x}^i_1 + V_2 \hat{x}^i_2 + \ldots + V_p \hat{x}^i_p$$

orthogonal axes

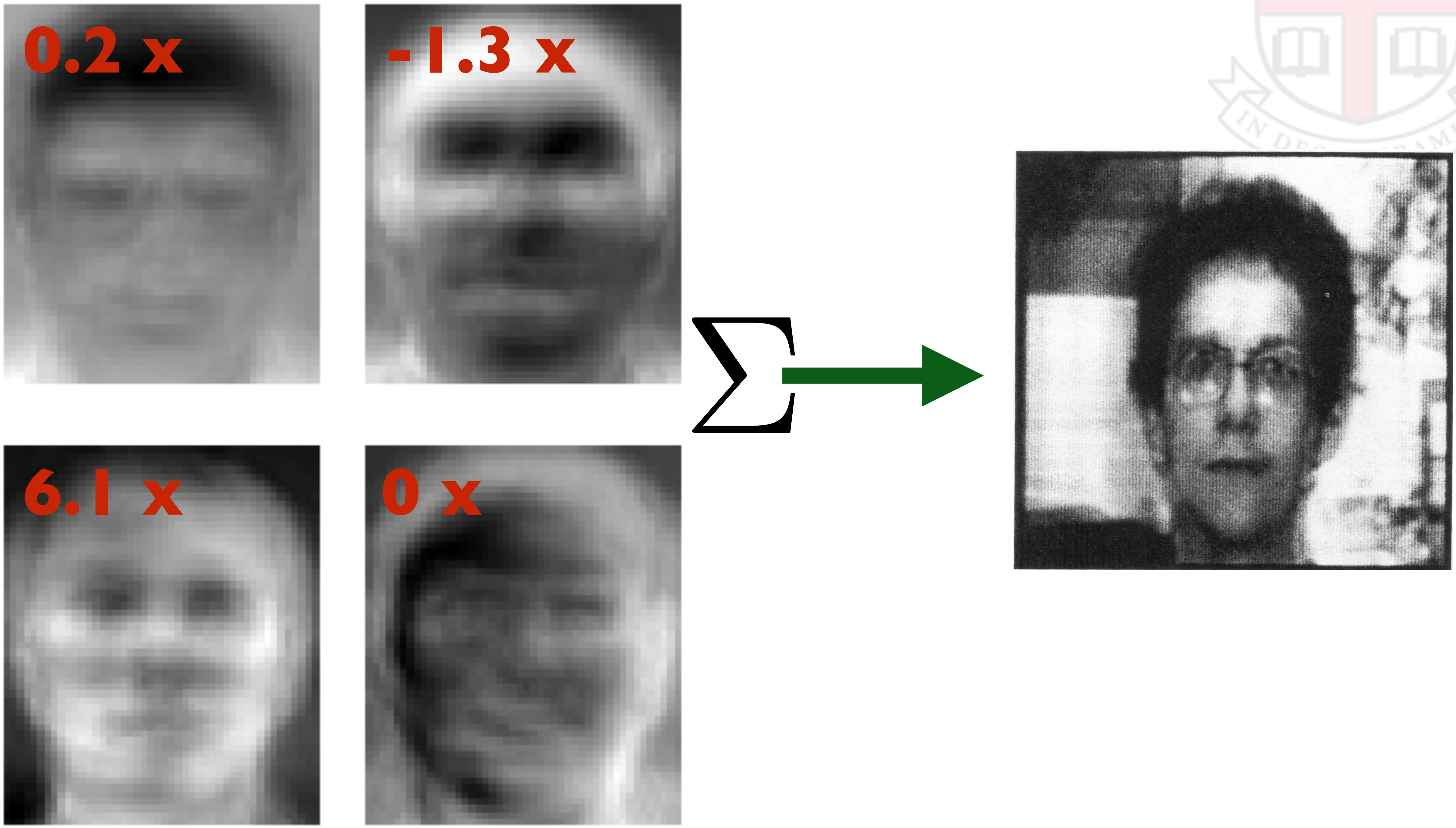
real valued numbers

**Every data point is expressed as a point in a new coordinate frame.**

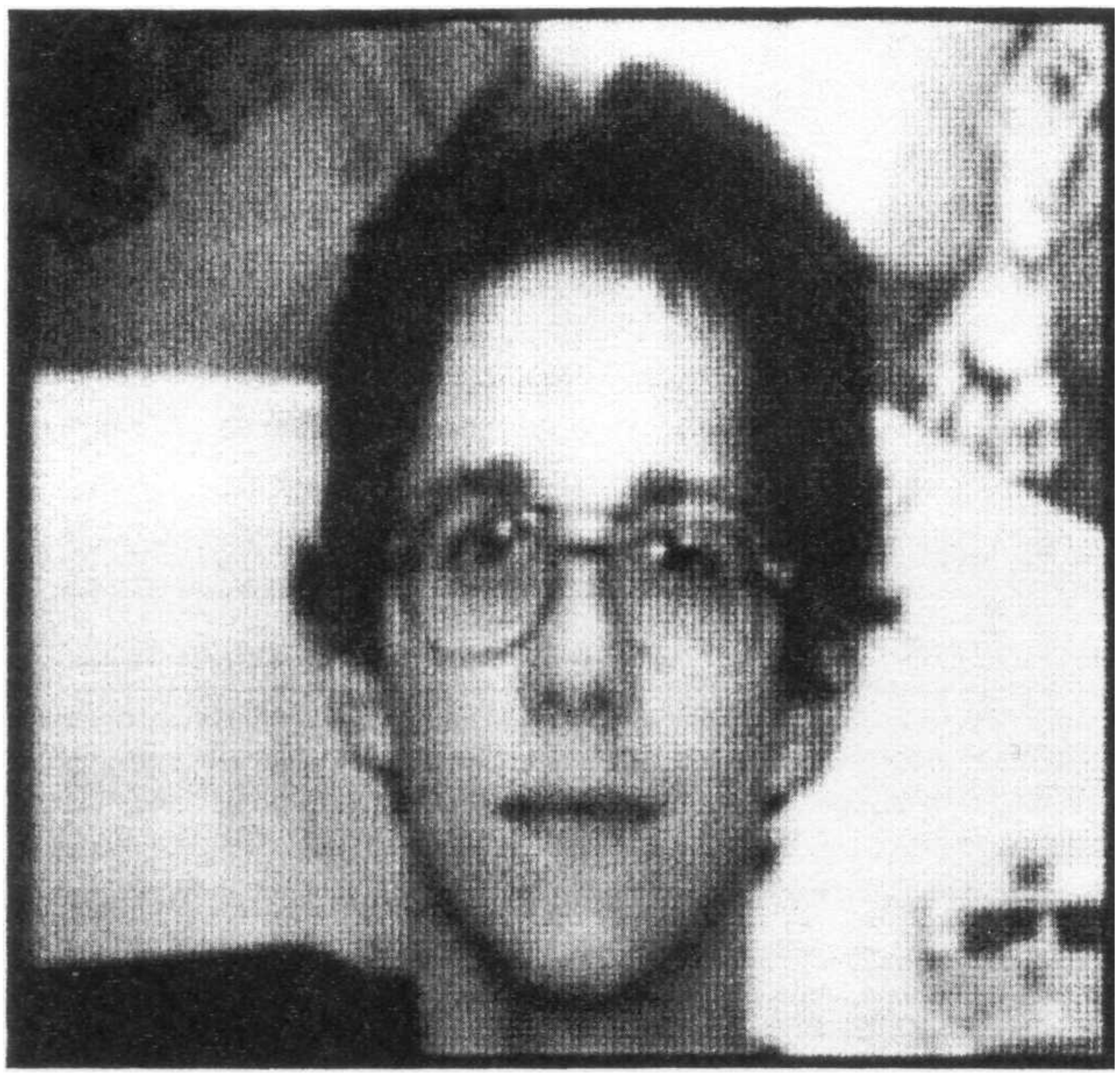
**Equivalently: weighted sum of basis (eigenvector) functions.**



# Eigenfaces



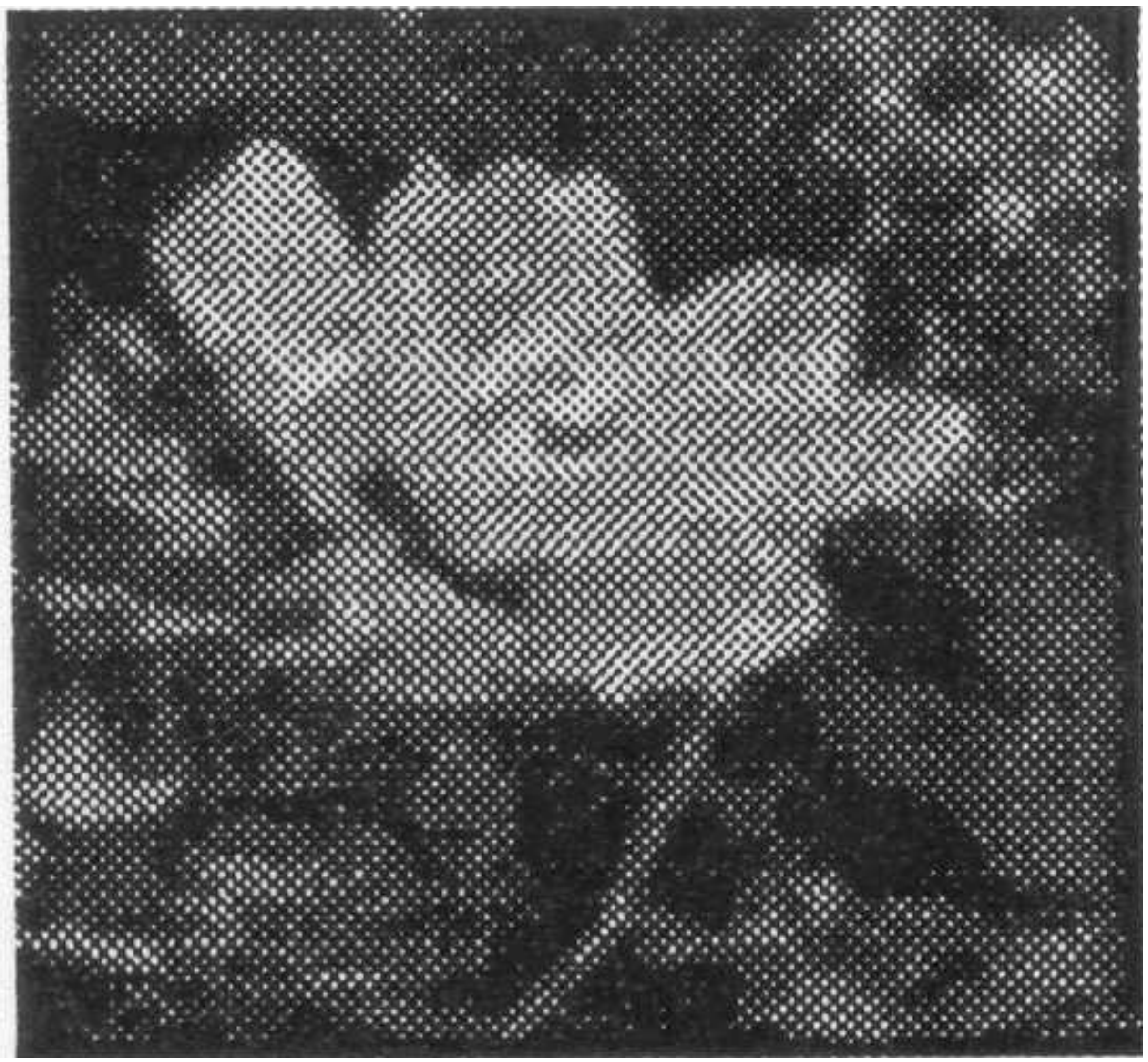
# Eigenfaces



(40 basis functions)

(Turk and Pentland, 1991)

# Eigenfaces



(40 basis functions)

(Turk and Pentland, 1991)

# PCA for Supervised Learning

Given data $x^1, \ldots, x^n$, labels $y^1, \ldots, y^n$:

- Compute compressor matrix *V*.
- Compute compressed data $\hat{x}^1, \ldots, \hat{x}^n$.
- Use compressed data to learn classifier:

$$f : \hat{X} \rightarrow Y$$

- Given a new data point *x*, run *f* on *Vx*.

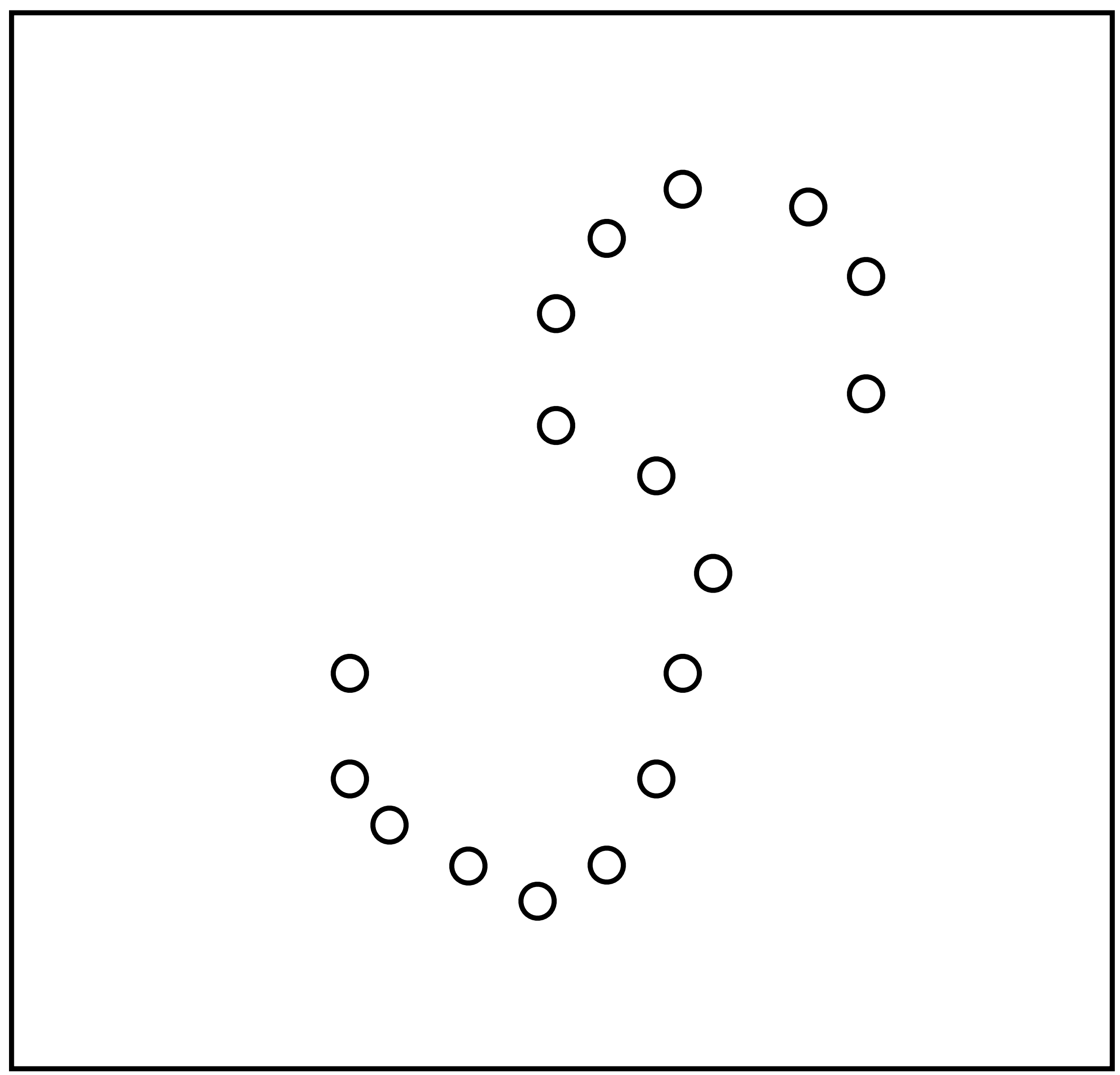
Why?

- Low amount of data relative to dimensionality.
- Dimensions may be highly correlated.
- Dimensions may be mostly noise/irrelevant/constant.
- *Not all data need be labelled.*

# ISOMAP

Another approach:

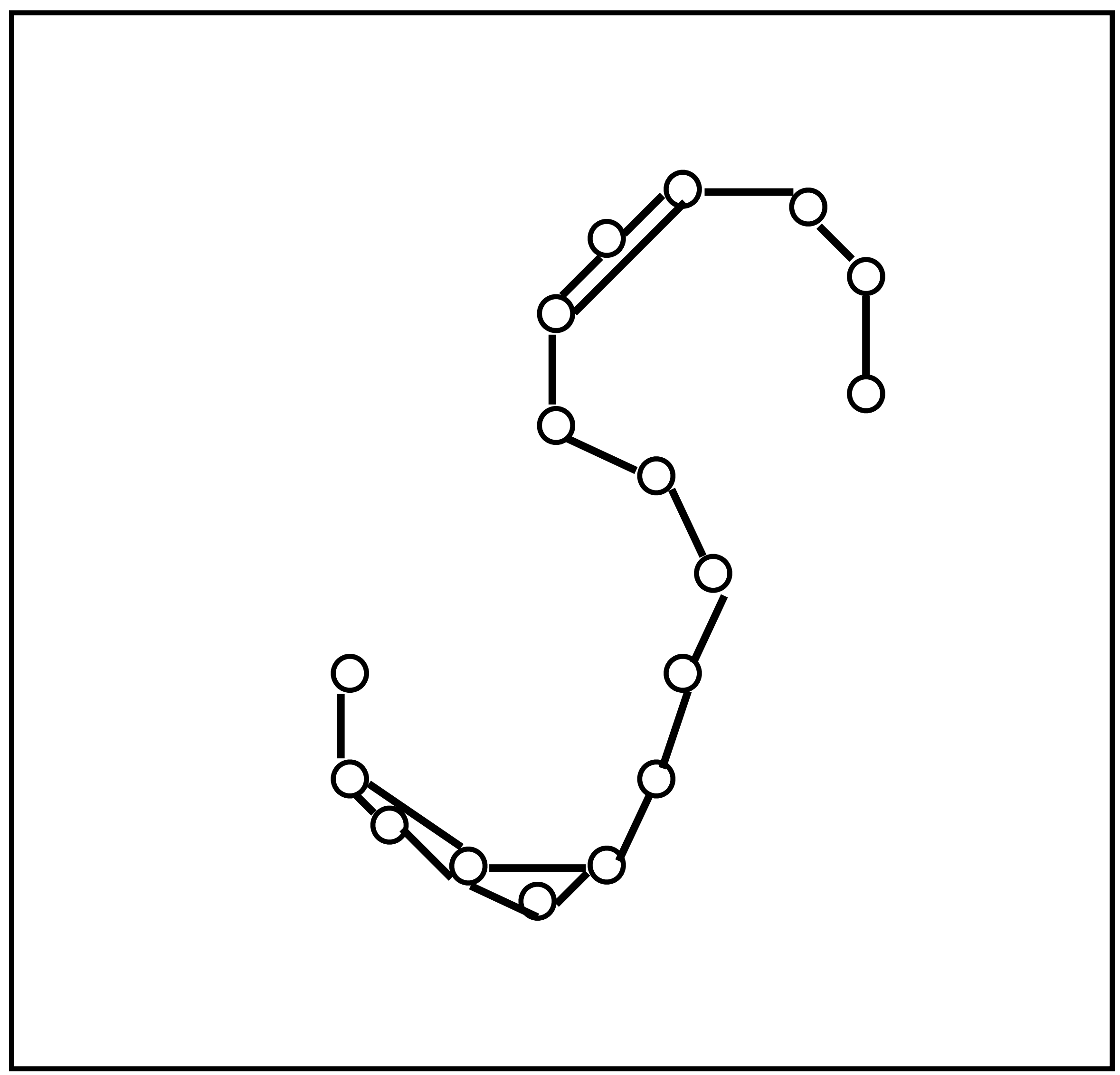
- Estimate intrinsic geometric dimensionality of data.
- Recover natural distance metric



# ISOMAP

Core idea: distance metric *locally Euclidean*

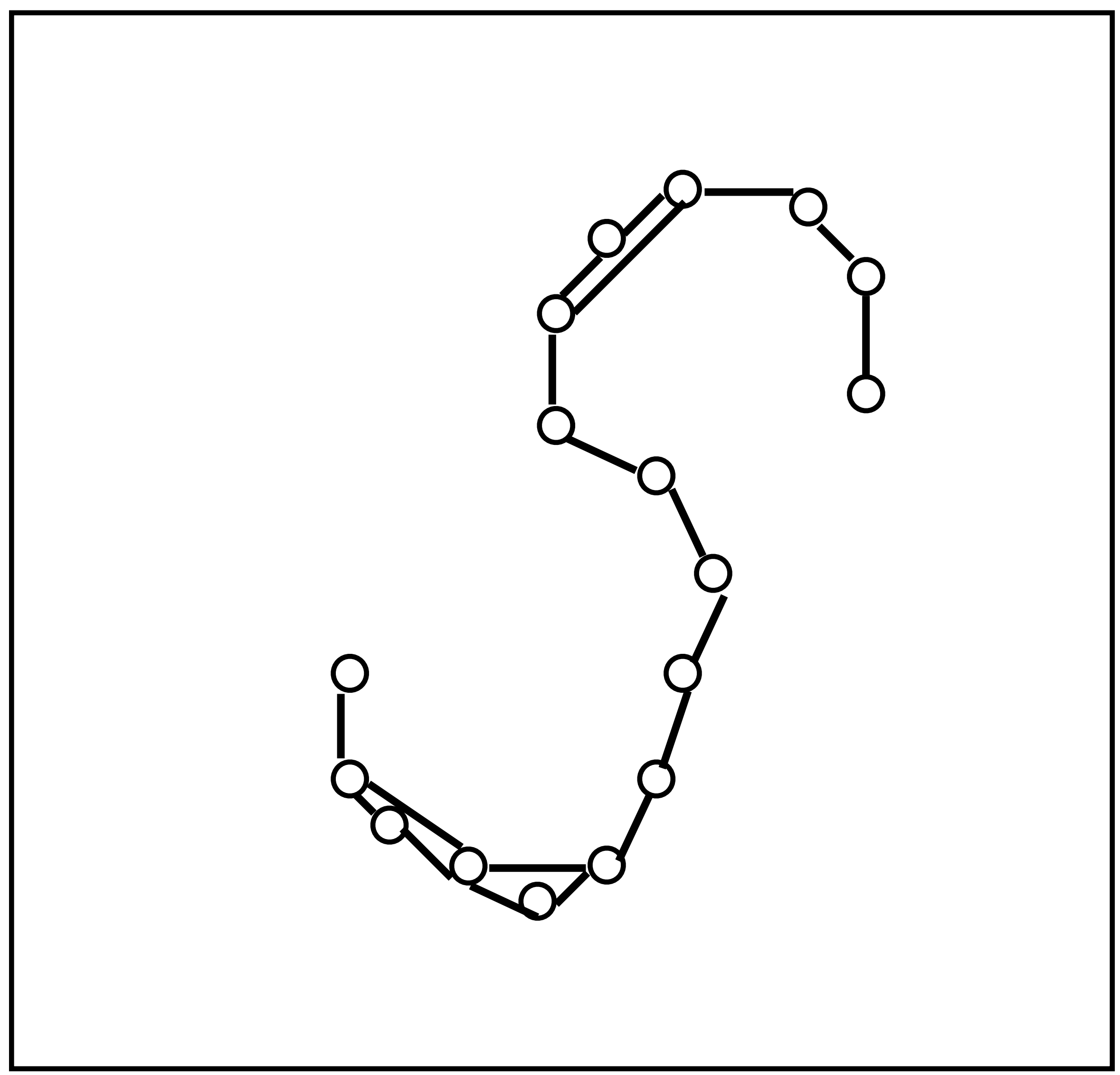
- Small radius *r*, connect each point to neighbors
- Weight based on Euclidean distance



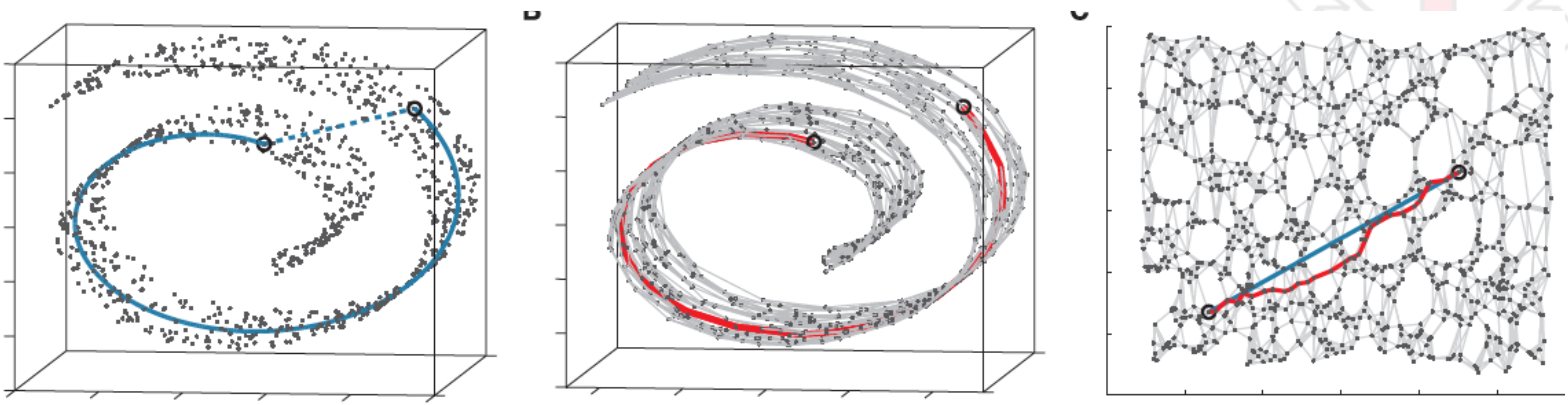
# ISOMAP

Solve all-points shortest pairs:

- Transforms local distance to global distance.
- Compute embedding.



# ISOMAP



From Tenenbaum, de Silva, and Langford, *Science* 290:2319-2323, December 2000.

# Application: Novelty Detection

Intrusion detection - when is a user behaving *unusually*?

First proposed by Prof. Dorothy Denning in 1986.
(1995 ACM Fellow)