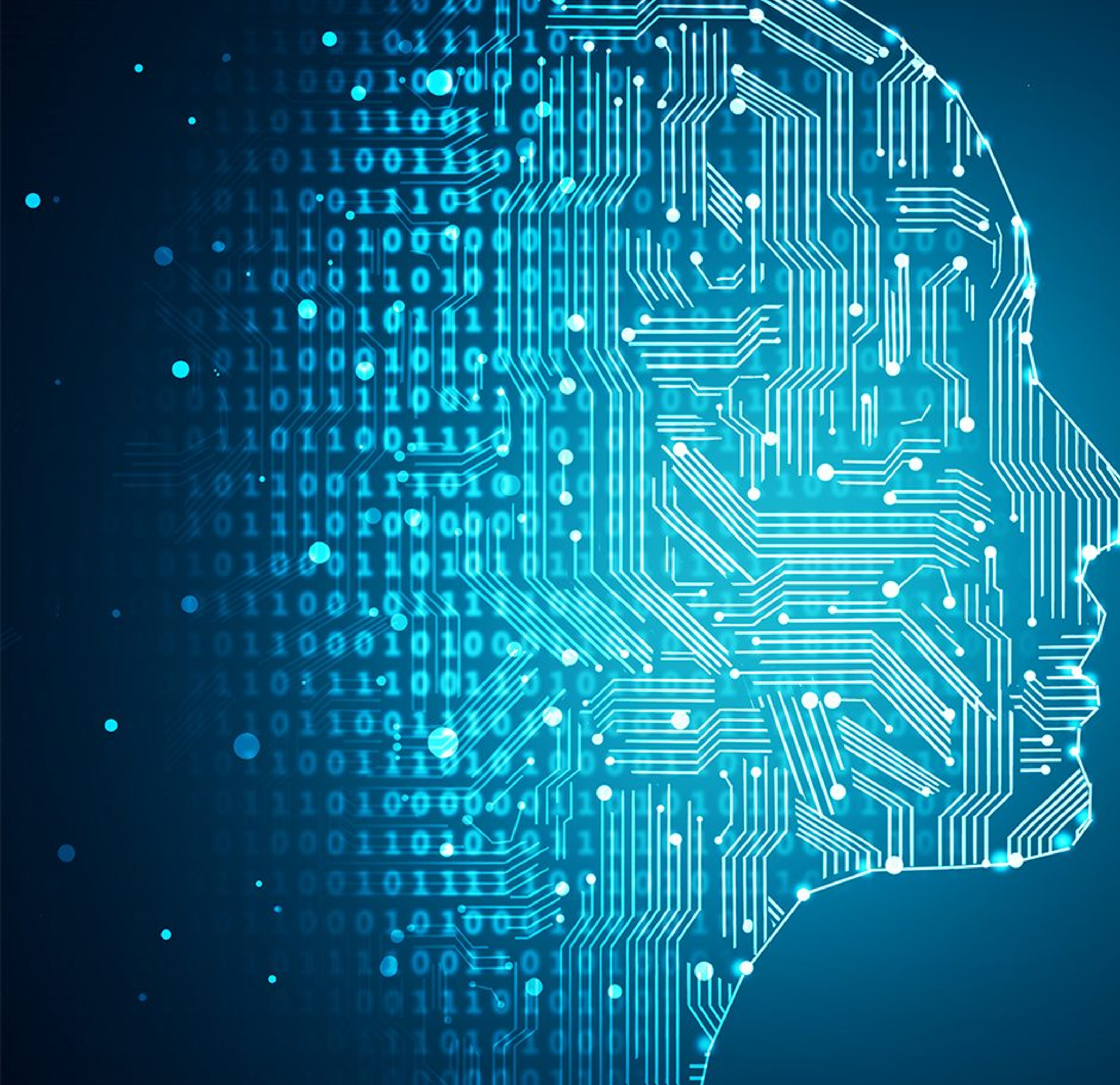
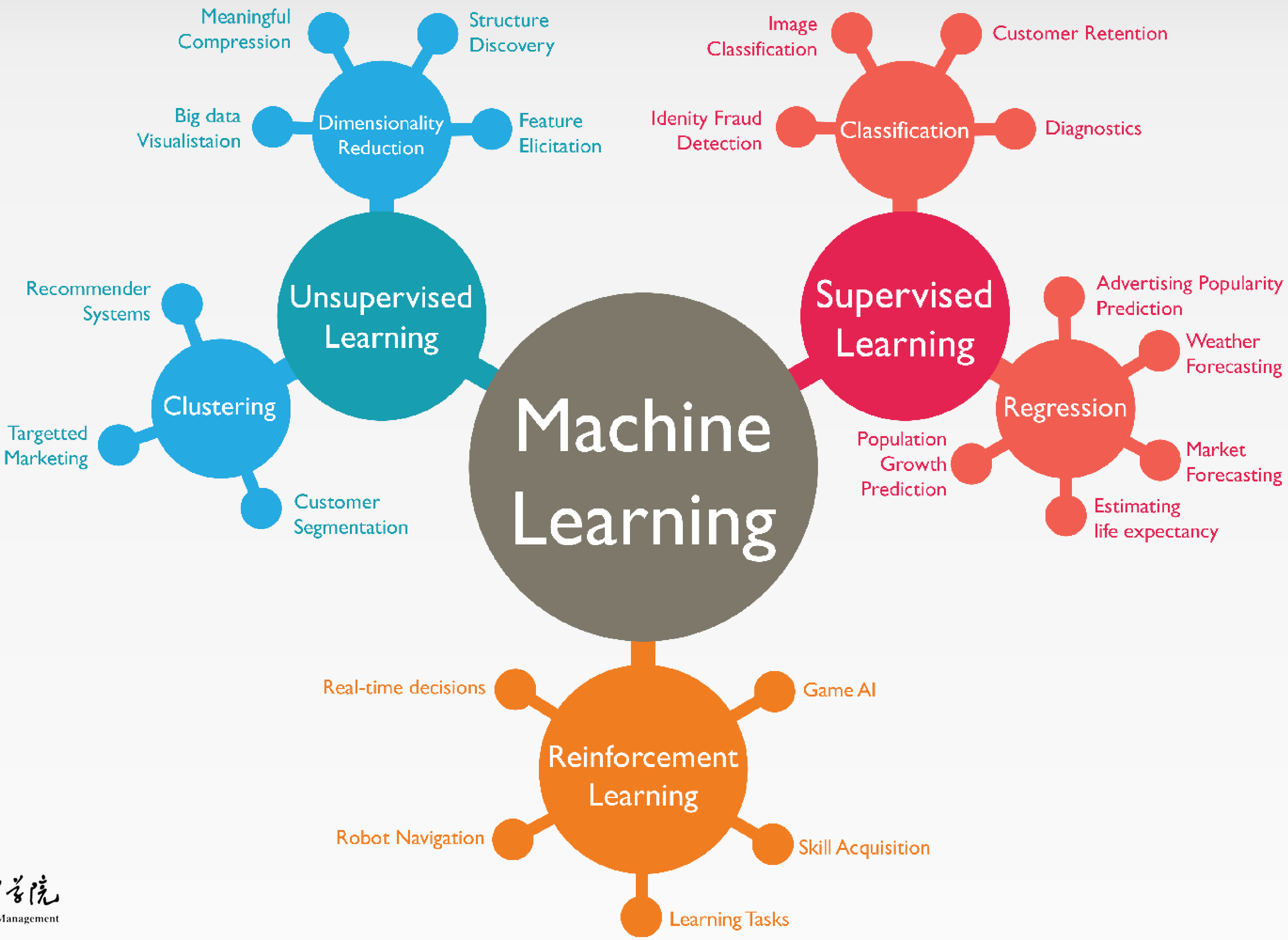


机器学习与人工智能
Machine Learning and Artificial Intelligence

Lecture 6 Ensemble Models, HMM, Clustering

Yingjie Zhang (张颖婕)
Peking University
yingjiezhang@gsm.pku.edu.cn

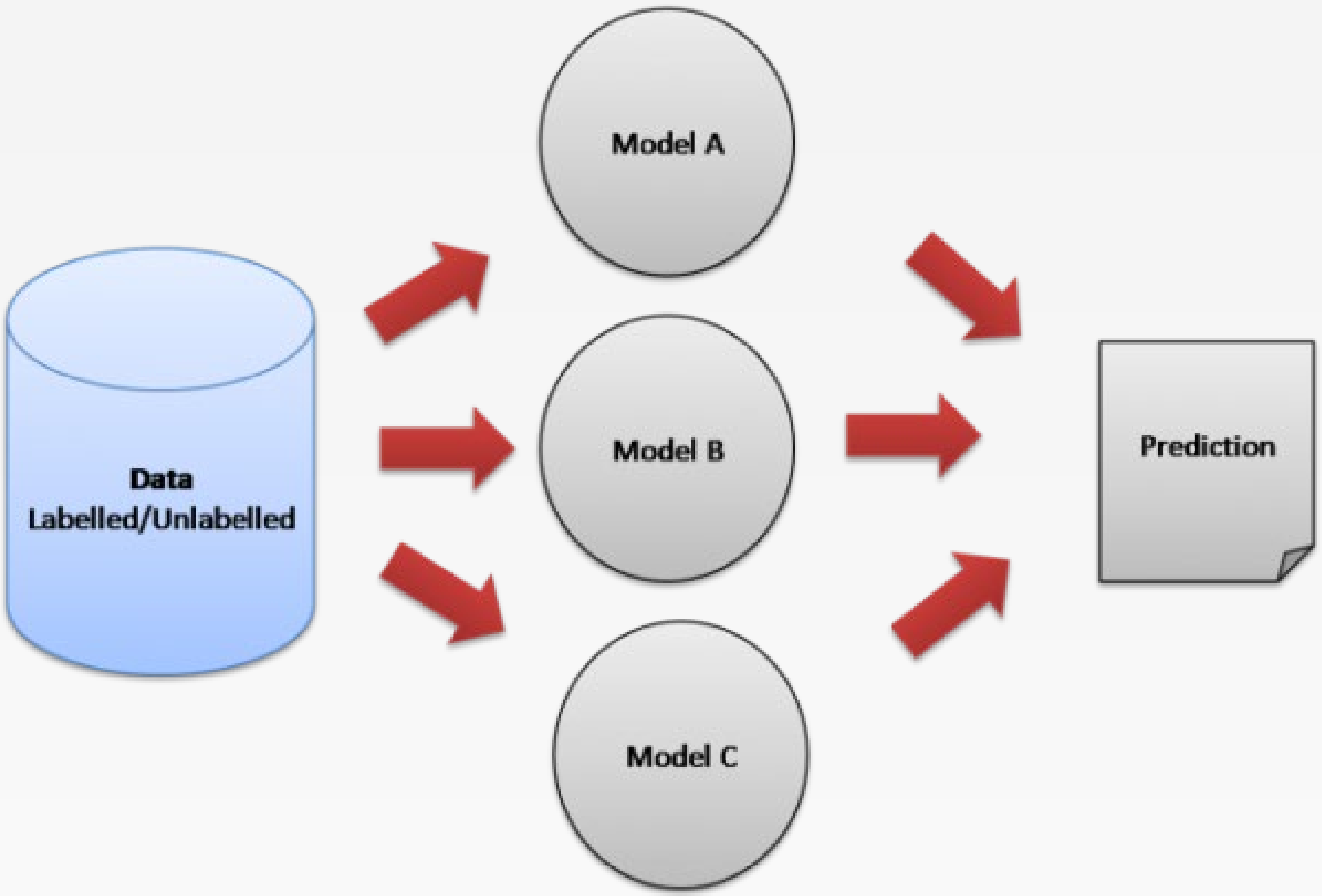
2021 Fall



光华管理学院
Guanghua School of Management

Ensemble Models

Wisdom of the Crowd



Ensemble Methods

- Why it works:
  - Diversity!
  - Image that we have 5 completely independent classifiers; each of them individually is correct 70% of the time
    - Prob(correctly classify a record by a majority vote)

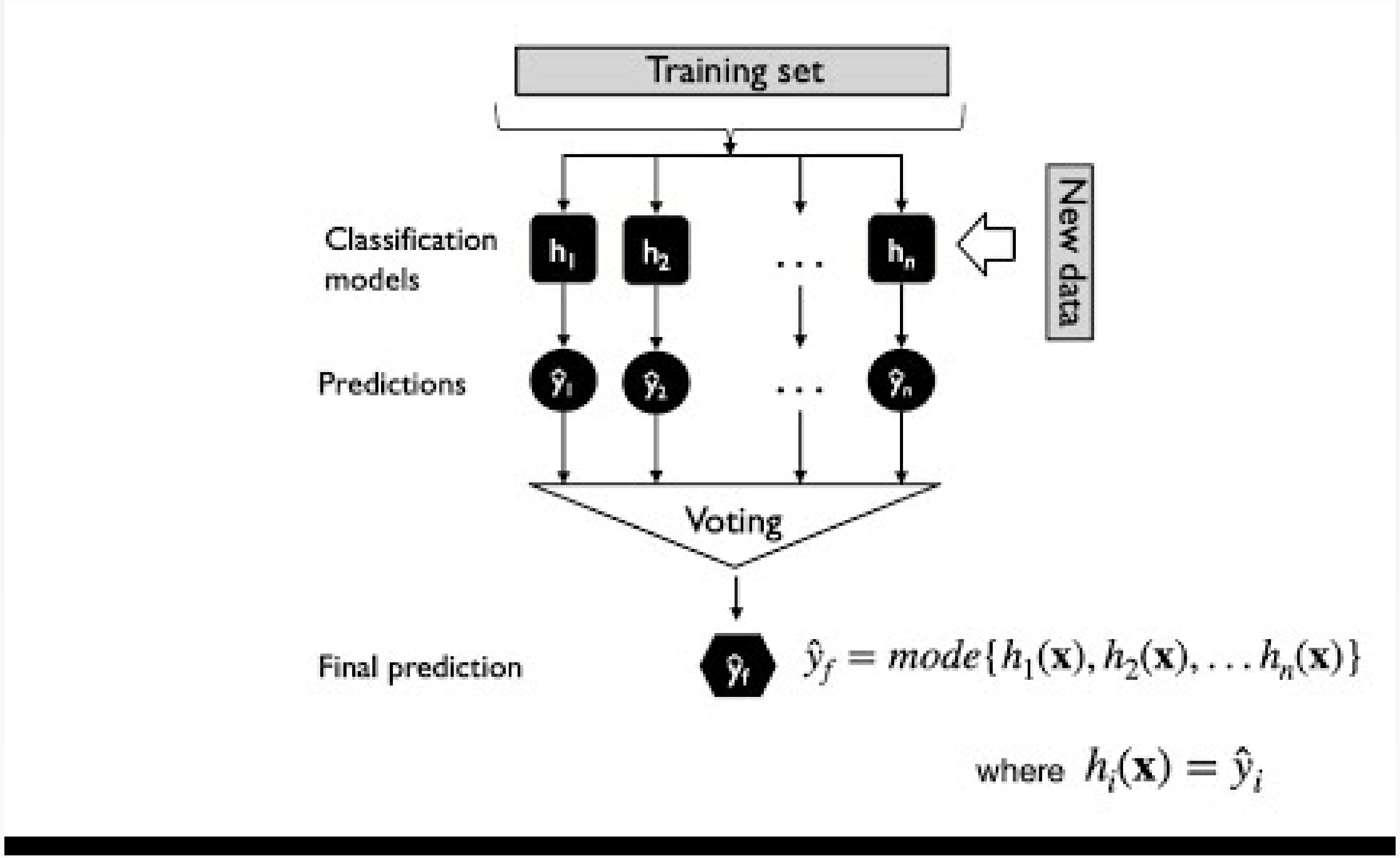
$$= C_{(5,3)}(0.7)^3(0.3)^2 + C_{(5,4)}(0.7)^4(0.3)^1 + C_{(5,5)}(0.7)^5 = 0.837$$

- Downside:
  - Increased complexity, more difficult to interpret
  - Does not always guarantee performance improvements

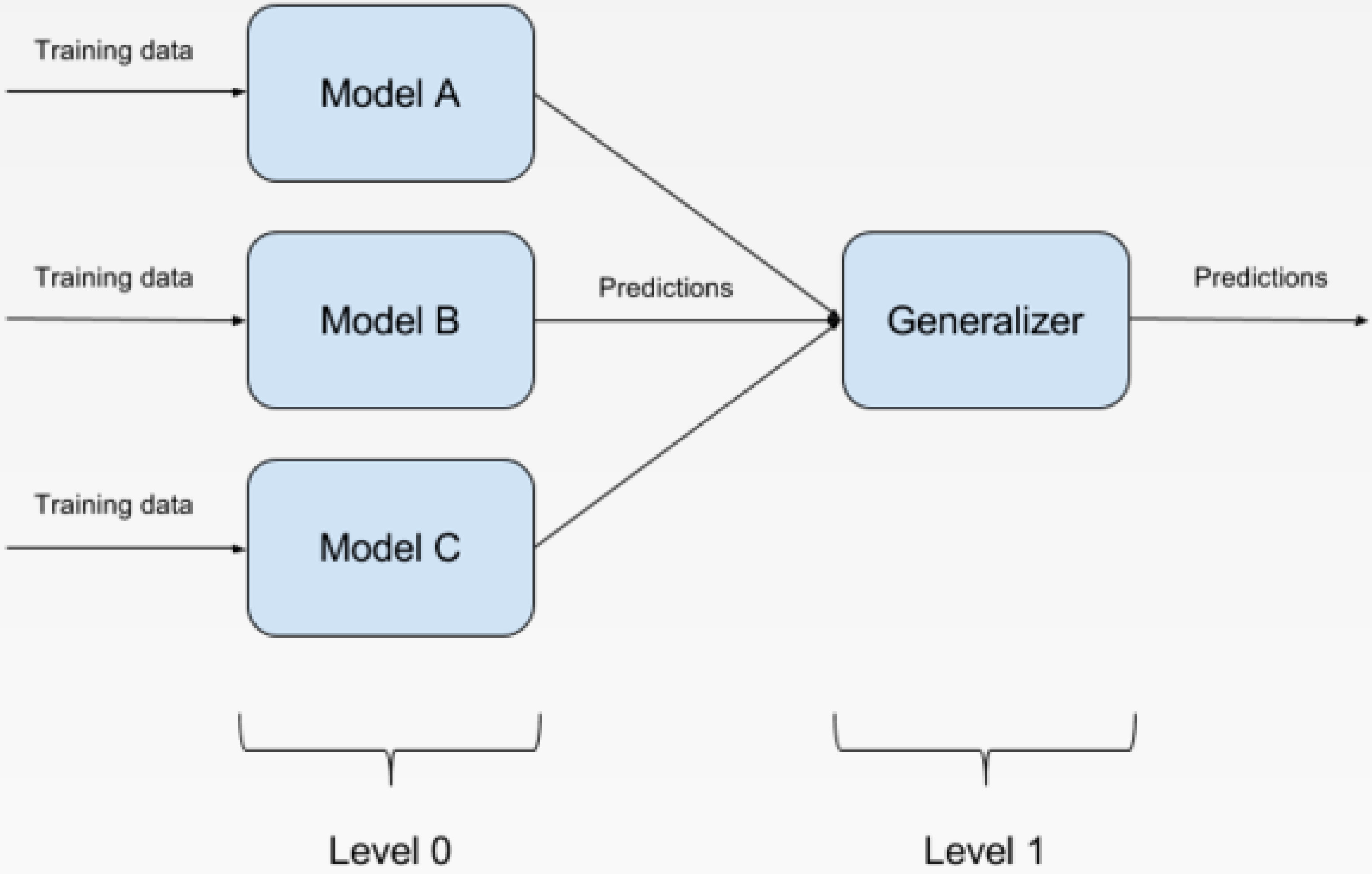
Ensemble Methods

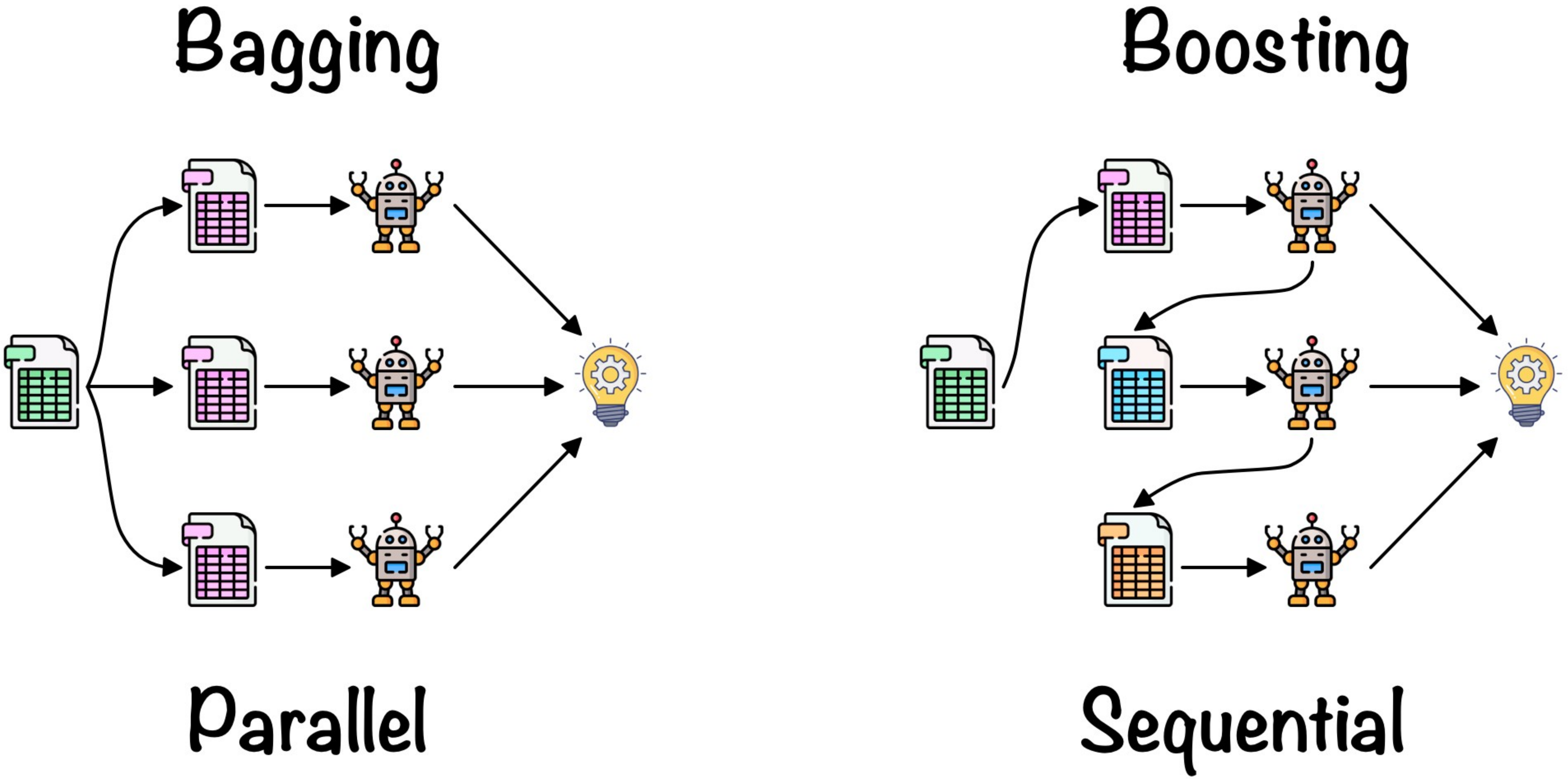
- Voting Classifiers
- Stacking
- Bagging
- Boosting

Voting Classifiers



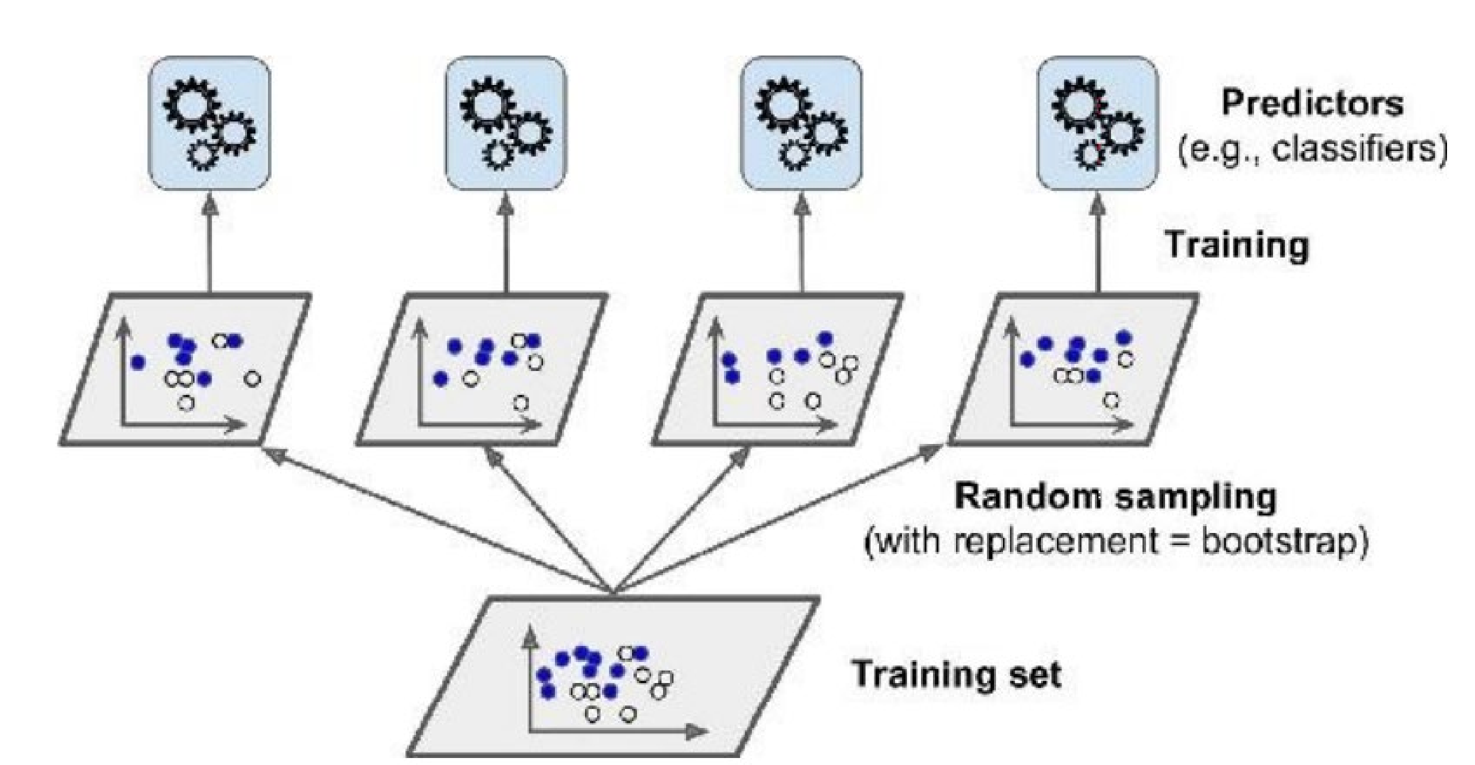
Stacking


Training data
Training data
Training data
Model A
Model B
Model C
Predictions
Generalizer
Predictions
Level 0
Level 1


Bagging
Boosting
Parallel
Sequential

Bagging: Bootstrap Aggregation

- Ideas:
  - Use the same training algorithm for every predictor, but to train them on different random subsets of the training set



Bagging

- Given
  - Labelled dataset
  - Specific predictive modeling techniques

- Train k models on different training data samples
  - Bootstrap samples: sampled with replacement, typically of the same size as the original training data

- Final prediction is done by combining (i.e., majority vote, averaging) the predictions of k individual models

Overview

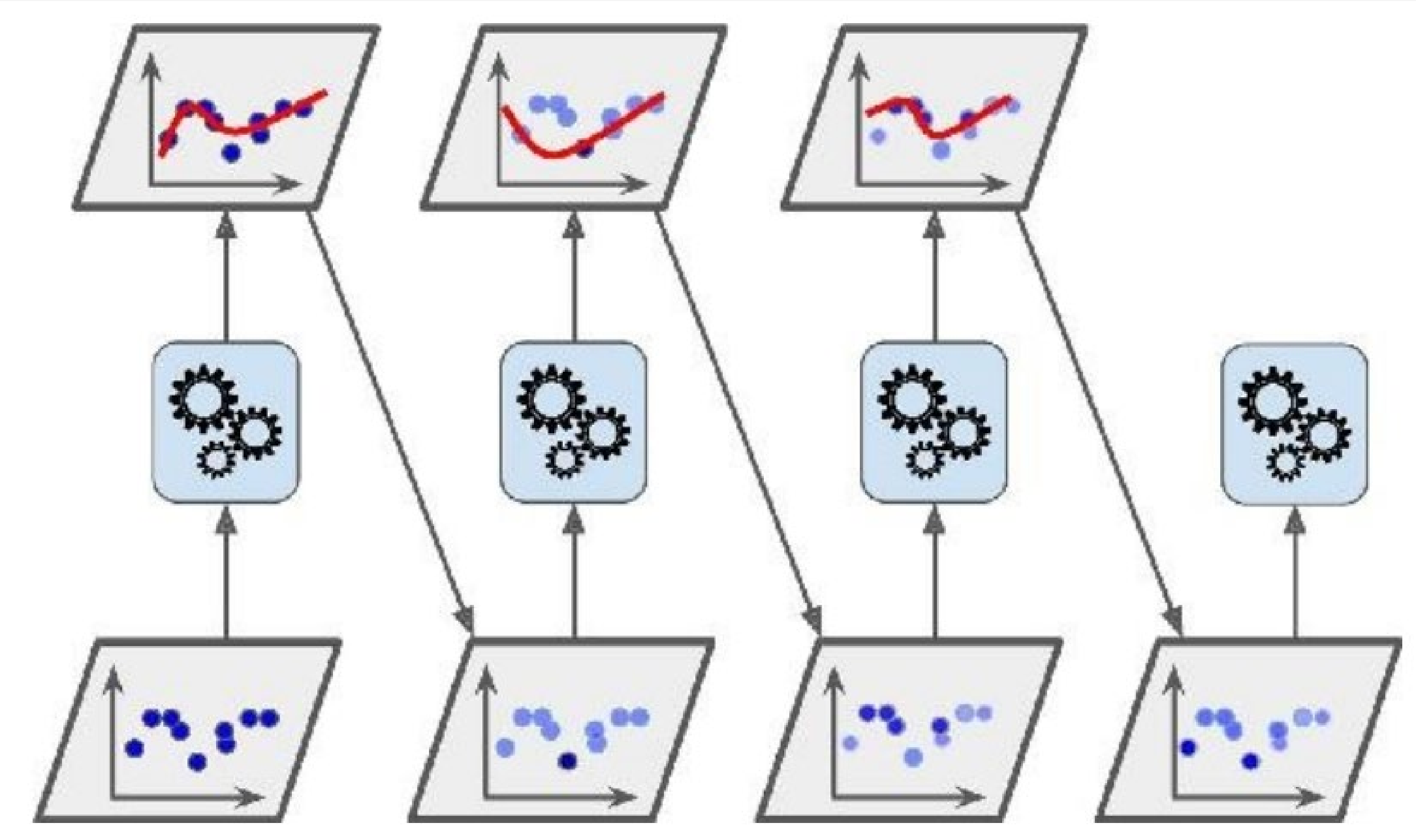
- Definition
  - Collection of unpruned trees
  - Rule to combine individual tree decisions
- Purpose
  - Improve prediction accuracy
  - Improve efficiency
- Principle
  - Encouraging diversity among the tree
- Solution: randomness
  - Bagging
  - Random decision trees

Details

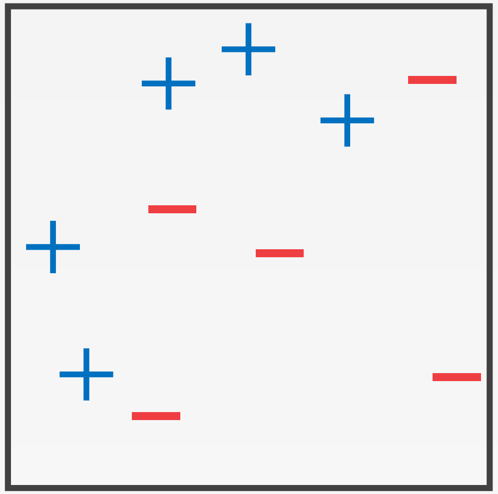
- Build many “random” trees
- Randomness: using only a random sample of m attributes to calculate each split
- For each tree:
  - Choose a different training sample
  - For each node, choose m random attributes and find the best split
  - Trees are often fully grown (not pruned)
- Predication: majority vote among all the trees

Boosting

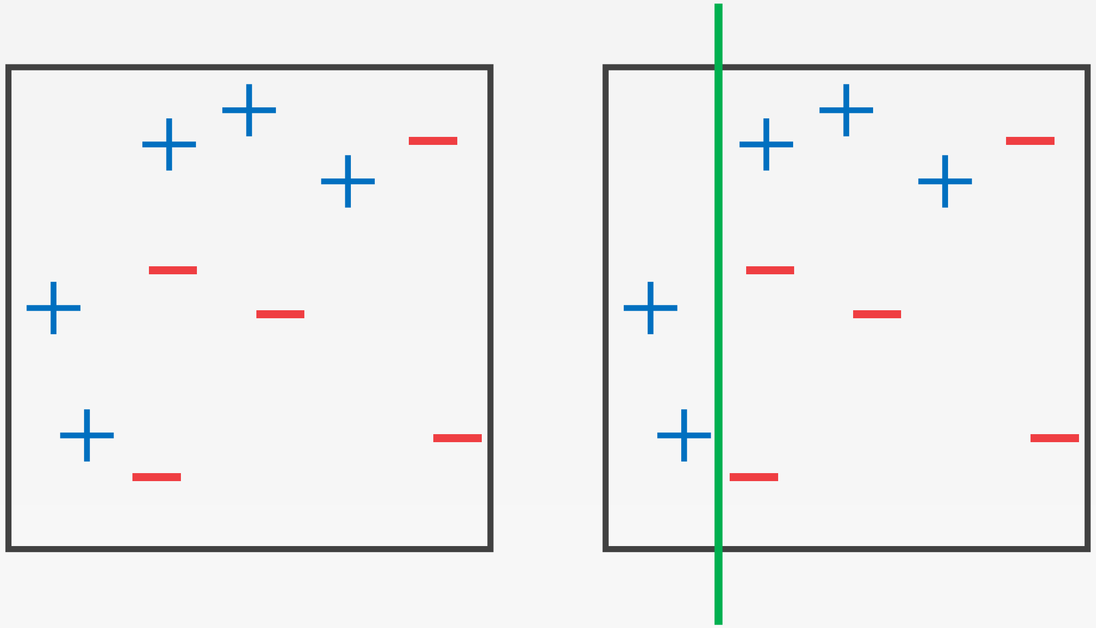
- AdaBoost



AdaBoost: Toy Example

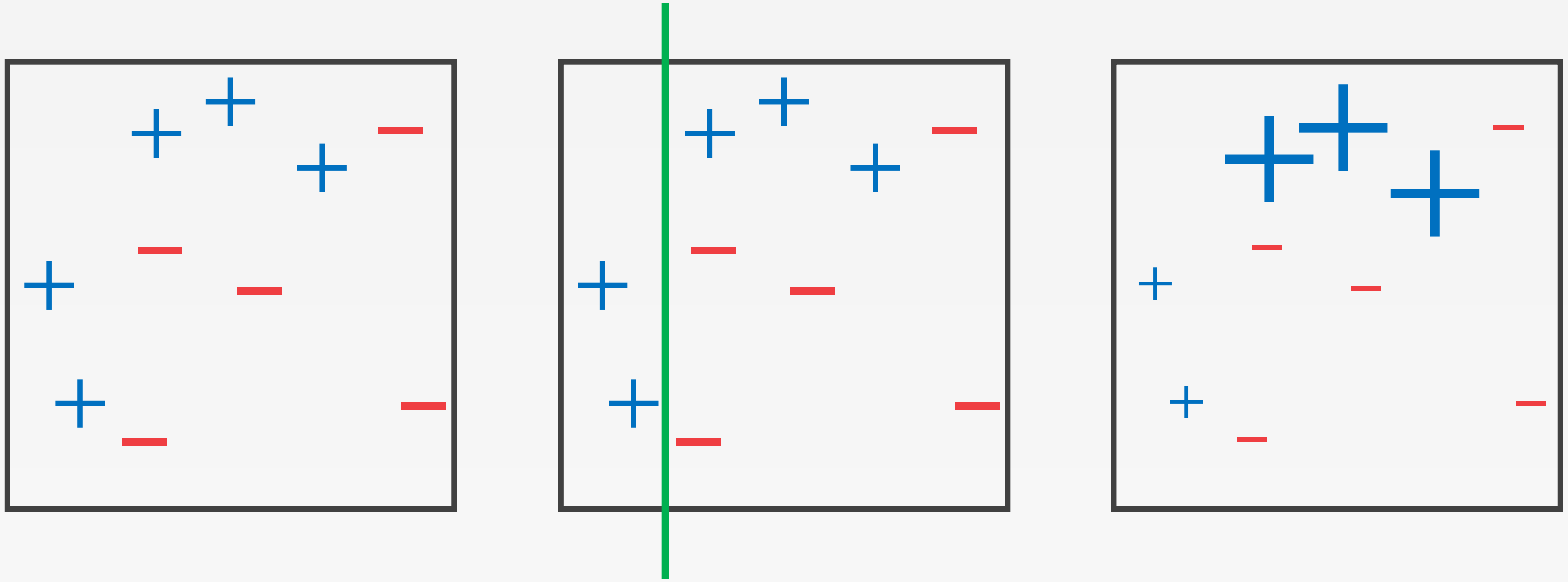


AdaBoost: Toy Example



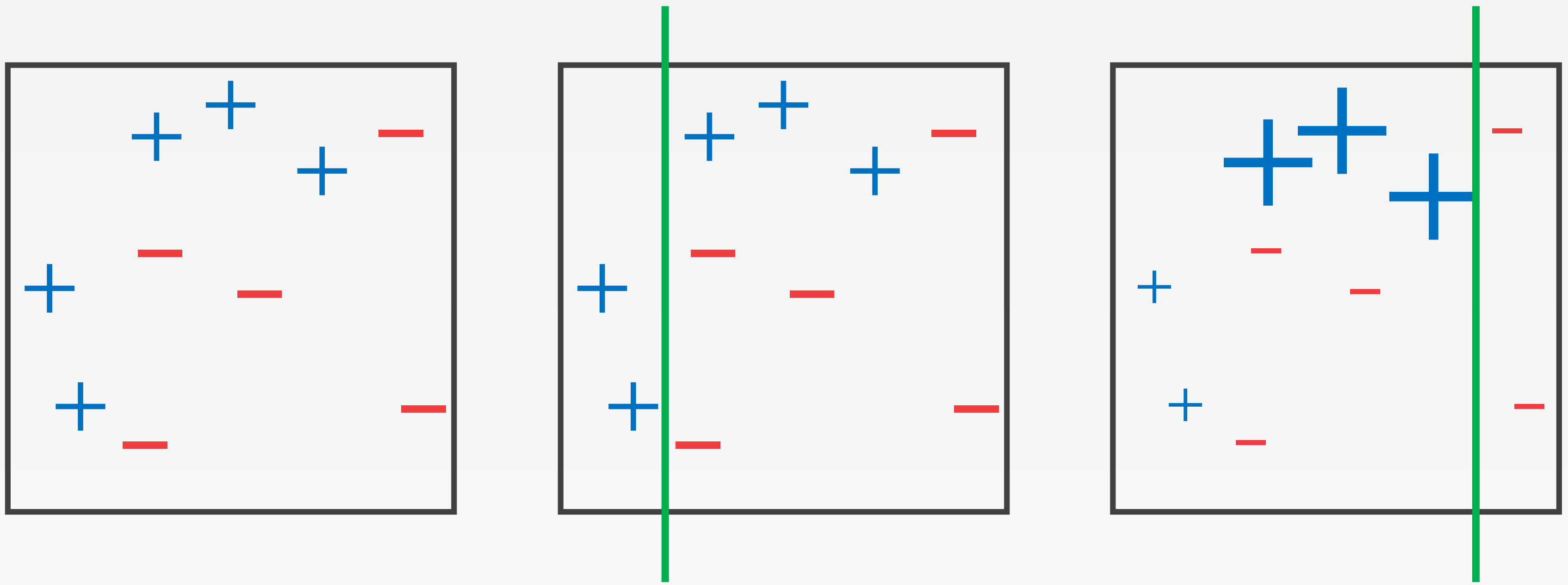
$\varepsilon_1 = 0.3$
$\alpha_1 = 0.42$

AdaBoost: Toy Example



$$\varepsilon_1 = 0.3$$
$$\alpha_1 = 0.42$$

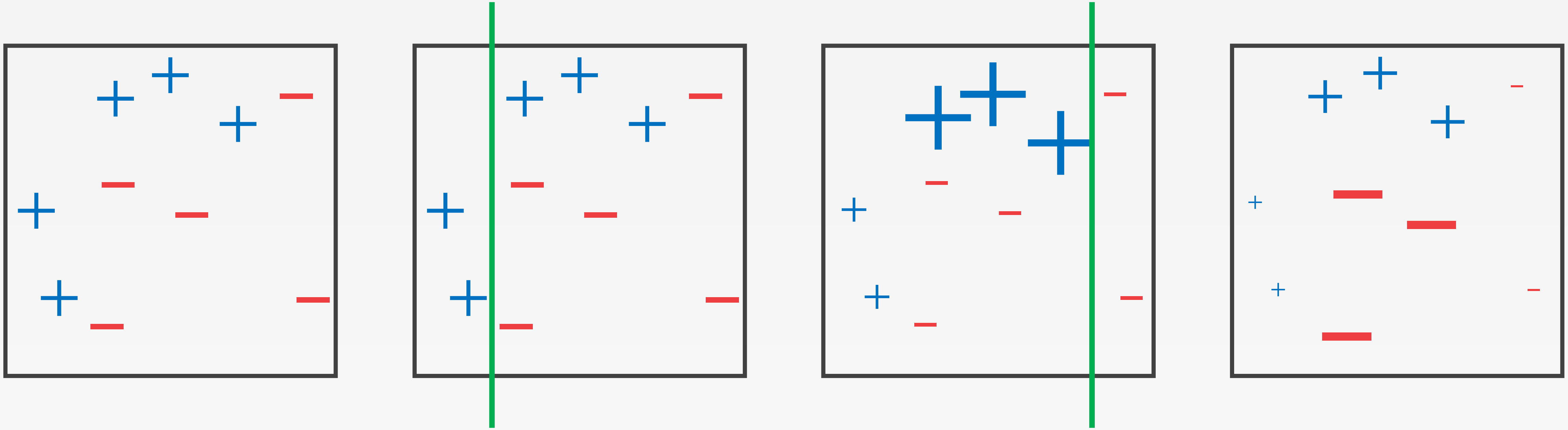
AdaBoost: Toy Example



$$\varepsilon_1 = 0.3$$
$$\alpha_1 = 0.42$$

$$\varepsilon_2 = 0.21$$
$$\alpha_2 = 0.65$$

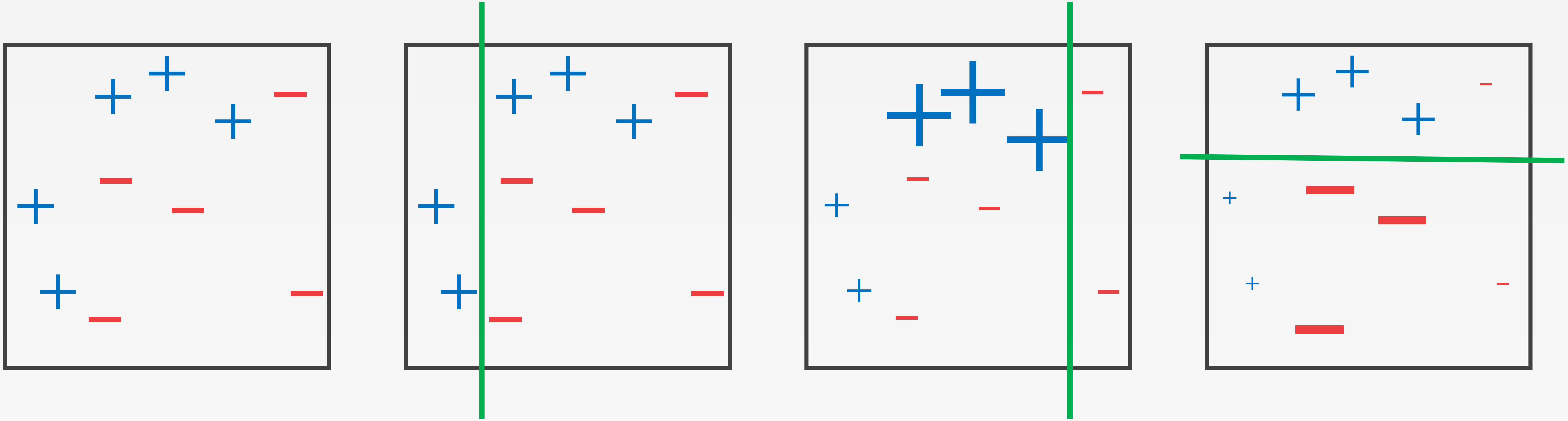
AdaBoost: Toy Example



$\varepsilon_1 = 0.3$
$\alpha_1 = 0.42$

$\varepsilon_2 = 0.21$
$\alpha_2 = 0.65$

AdaBoost: Toy Example

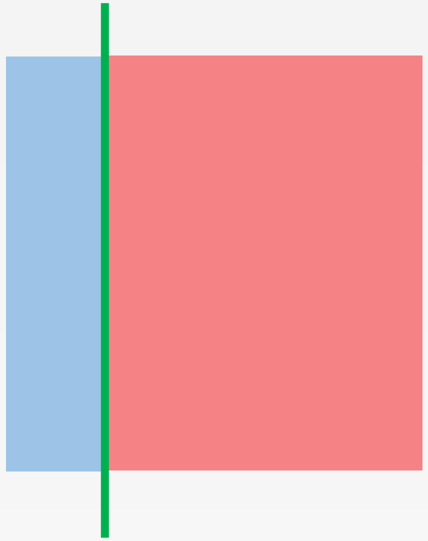
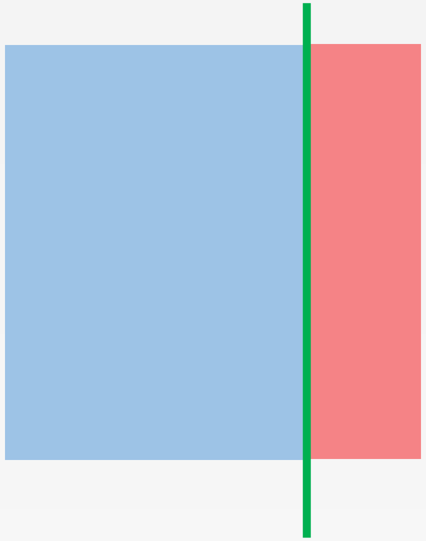


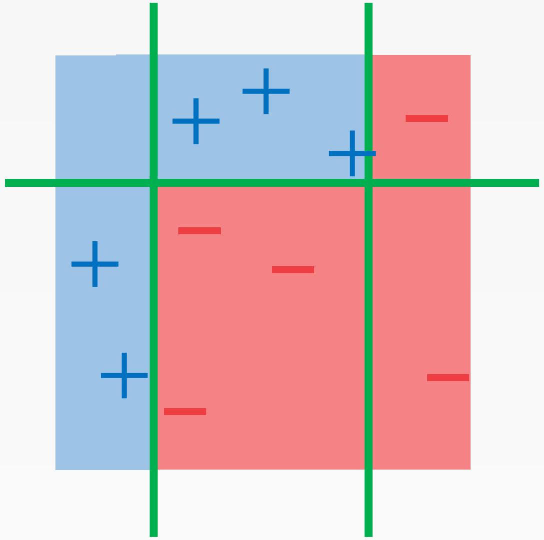
$\varepsilon_1 = 0.3$
$\alpha_1 = 0.42$

$\varepsilon_2 = 0.21$
$\alpha_2 = 0.65$

$\varepsilon_3 = 0.14$
$\alpha_3 = 0.92$

AdaBoost: Toy Example

$H_{final} = sign\Big($ 0.42  +0.65  +0.92  $\Big)$



AdaBoost Algorithm

Given: $(x_1, y_1), \dots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = 1/m$

For t=1,…,T:

Train weak learner using distribution D_t

Get week hypothesis $h_t: X \to \{-1, +1\}$ with error $\varepsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$

Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$

Update:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} \ if \ h_t(x_i) = y_i \\ e^{\alpha_t} \ \ if \ h_t(x_i) \neq y_i \end{cases} = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution)

Output the final hypothesis: $H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$

光华管理学院 Guanghua School of Management

Hidden Markov Models


Genie S-65
Genie

HMM

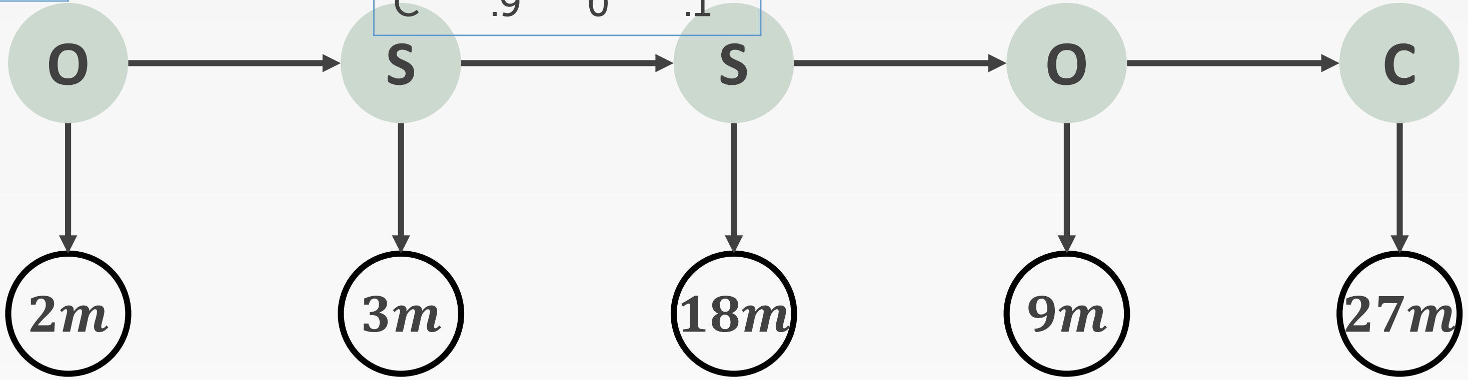
- An HMM provides a joint distribution with an assumption of dependence between adjacent states

$$p(O,S,S,O,C,2m,3m,18m,9m,27m) = (.8 * .2 * .08 * .03 * .7 * \cdots)$$

O	.8
S	.1
C	.1

	O	S	C
O	.9	.08	.02
S	.2	.7	.1
C	.9	0	.1

	1m	2m	3m	…
O	.1	.2	.3	
S	.01	.02	.03	
C	0	0	0	



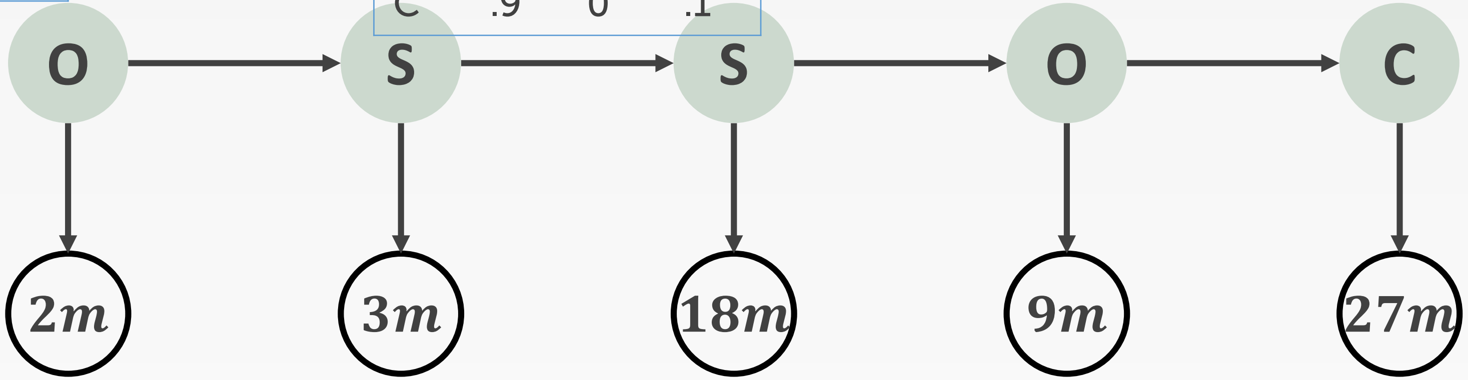
HMM

- An HMM provides a joint distribution with an assumption of dependence between adjacent states

$$p(O, S, S, O, C, 2m, 3m, 18m, 9m, 27m) = (.8 * .2 * .08 * .03 * .7 * \cdots)$$

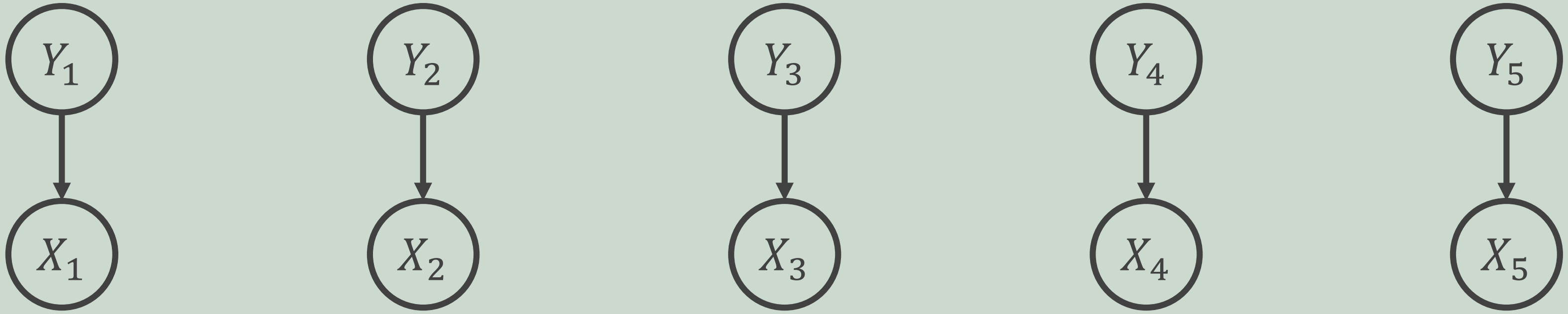
O	.8
S	.1
C	.1

	O	S	C
O	.9	.08	.02
S	.2	.7	.1
C	.9	0	.1



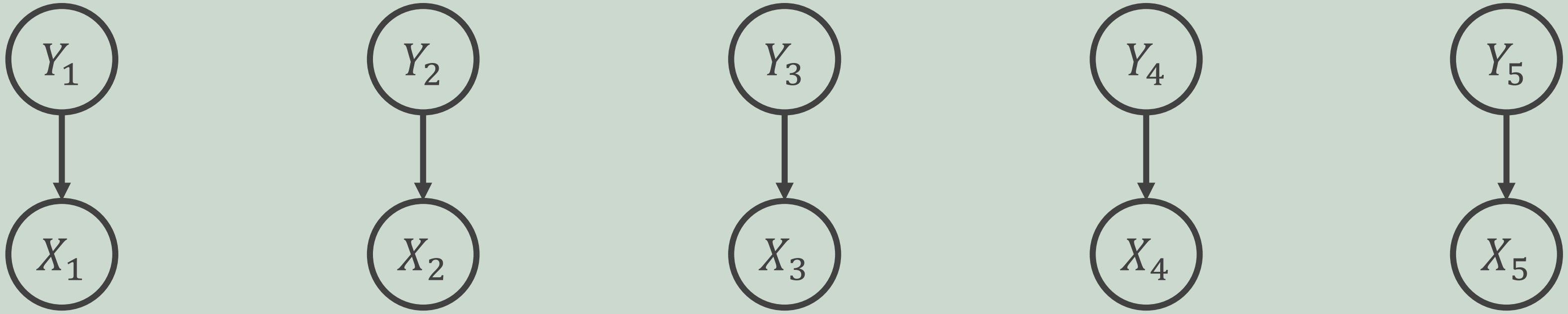
	1m	2m	3m	…
O	.1	.2	.3	
S	.01	.02	.03	
C	0	0	0	

HMM

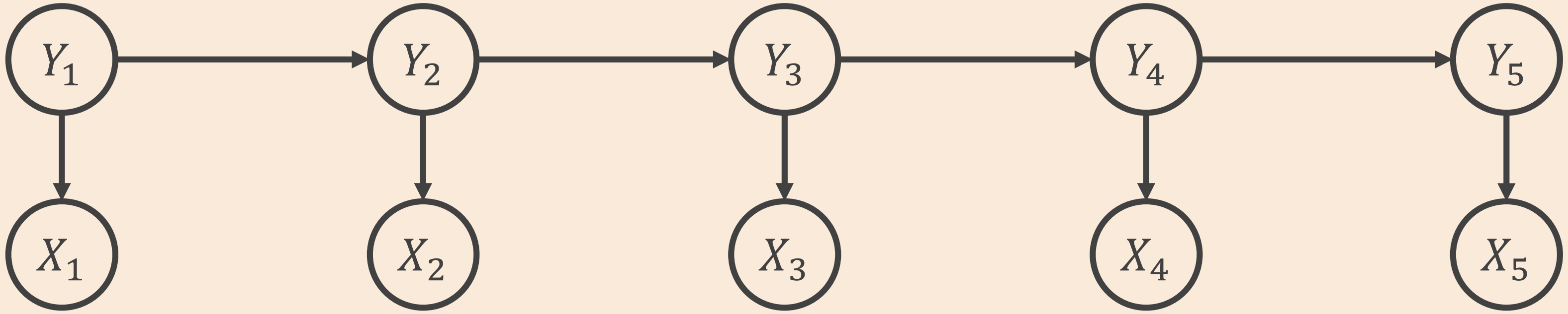


Naïve Bayes: $P(\boldsymbol{X}, \boldsymbol{Y}) = \prod_{t=1}^{T} P(X_t | Y_t) p(Y_t)$

HMM

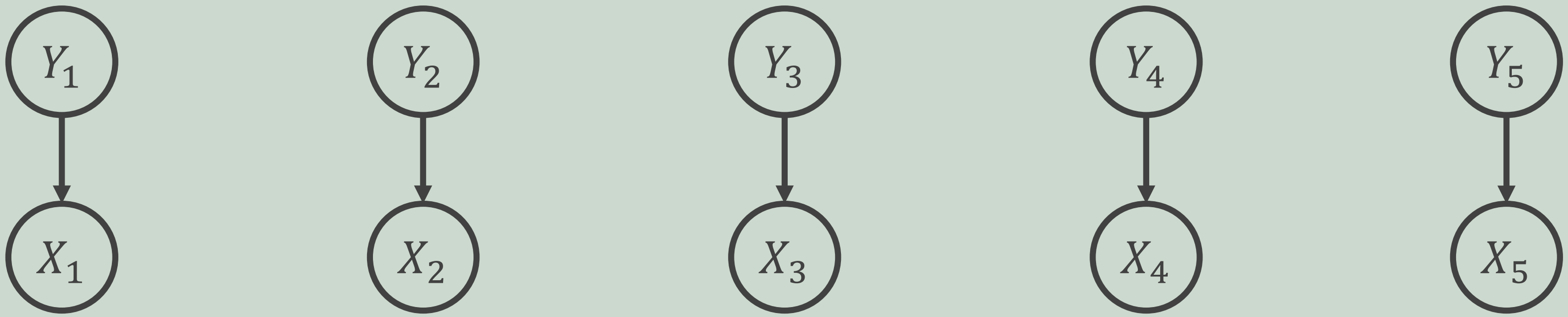


Naïve Bayes: $P(\boldsymbol{X}, \boldsymbol{Y}) = \prod_{t=1}^{T} P(X_t|Y_t)p(Y_t)$

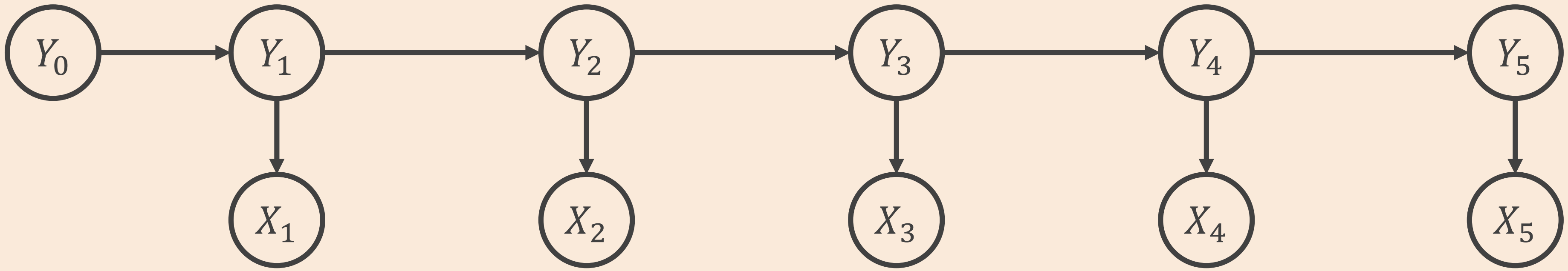


HMM: $P(\boldsymbol{X}, \boldsymbol{Y}|Y_0) = \prod_{t=1}^{T} P(X_t|Y_t)p(Y_t|Y_{t-1})$

HMM



Naïve Bayes: $P(\boldsymbol{X}, \boldsymbol{Y}) = \prod_{t=1}^{T} P(X_t|Y_t)p(Y_t)$



HMM: $P(\boldsymbol{X}, \boldsymbol{Y}|Y_0) = \prod_{t=1}^{T} P(X_t|Y_t)p(Y_t|Y_{t-1})$

Supervised Learning for HMM

- HMM Parameters:
  - Emission matrix, A, where $P(X_t = k|Y_t = j) = A_{jk}, \forall t, k$
  - Transition matrix, B, where $P(Y_t = k|Y_{t-1} = j) = B_{jk}, \forall t, k$
- Assumption: $y_0 = START$
- Generative Story:
  - $Y_t \sim Multinomial\left(B_{Y_{t-1}}\right), \forall t$
  - $X_t \sim Multinomial\left(A_{Y_t}\right), \forall t$
- Joint Distribution:
  - $p(\boldsymbol{X}, \boldsymbol{Y}|y_0) = \prod_{t=1}^{T} p(x_t|y_t)p(y_t|y_{t-1}) = \prod_{t=1}^{T} A_{y_t, x_t} B_{y_{t-1}, y_t}$

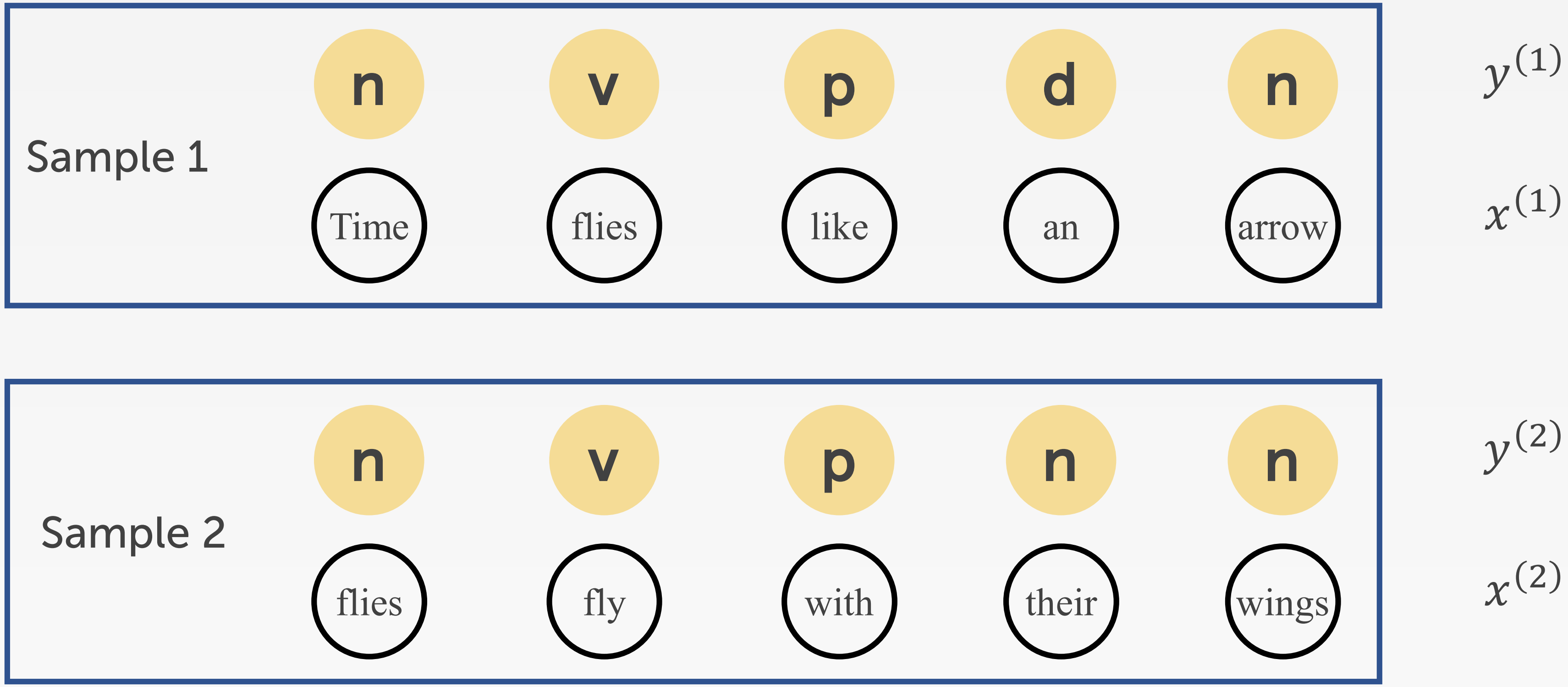
Unsupervised Learning for HMMs

- We don't observe any y's
- This unsupervised learning setting can be achieved by finding parameters that maximize the marginal likelihood
- We optimize using the Expectation-Maximization (EM) algorithm
  - Marginal probability: $p_\theta(x) = \sum_{y \in \mathbb{Y}} p_\theta(x, y)$
  - $l(\theta) = \log \prod_{i=1}^{N} p_\theta\left(x^{(i)}\right) = \sum_{i=1}^{N} \log \sum_{y \in \mathbb{Y}} p_\theta\left(x^{(i)}, y\right)$

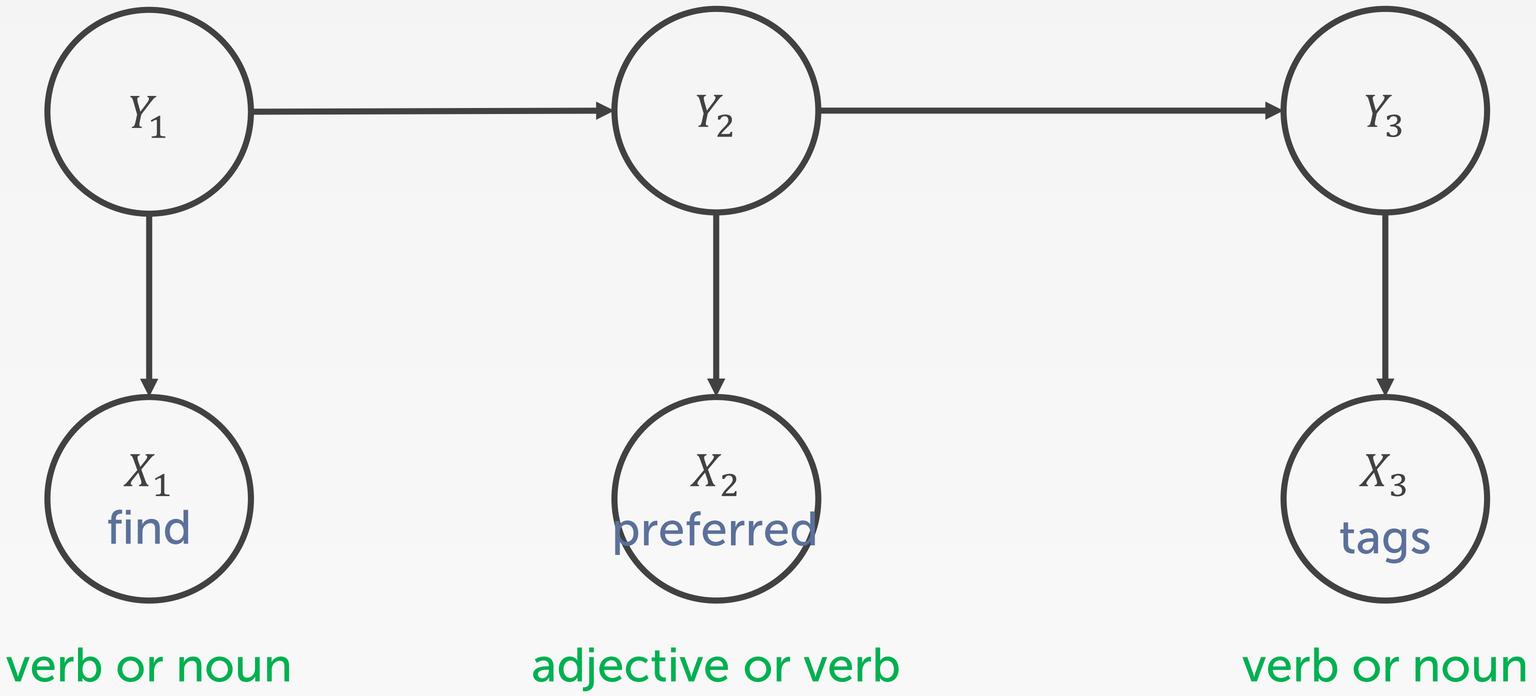
Inference for HMMs

- **Evaluation**: Compute the probability of a given sequence of observations
- **Viterbi Decoding**: Find the most-likely sequence of hidden states, given a sequence of observations
- **Learning**: find the optimal parameters to maximize the probability of the sequence of observations

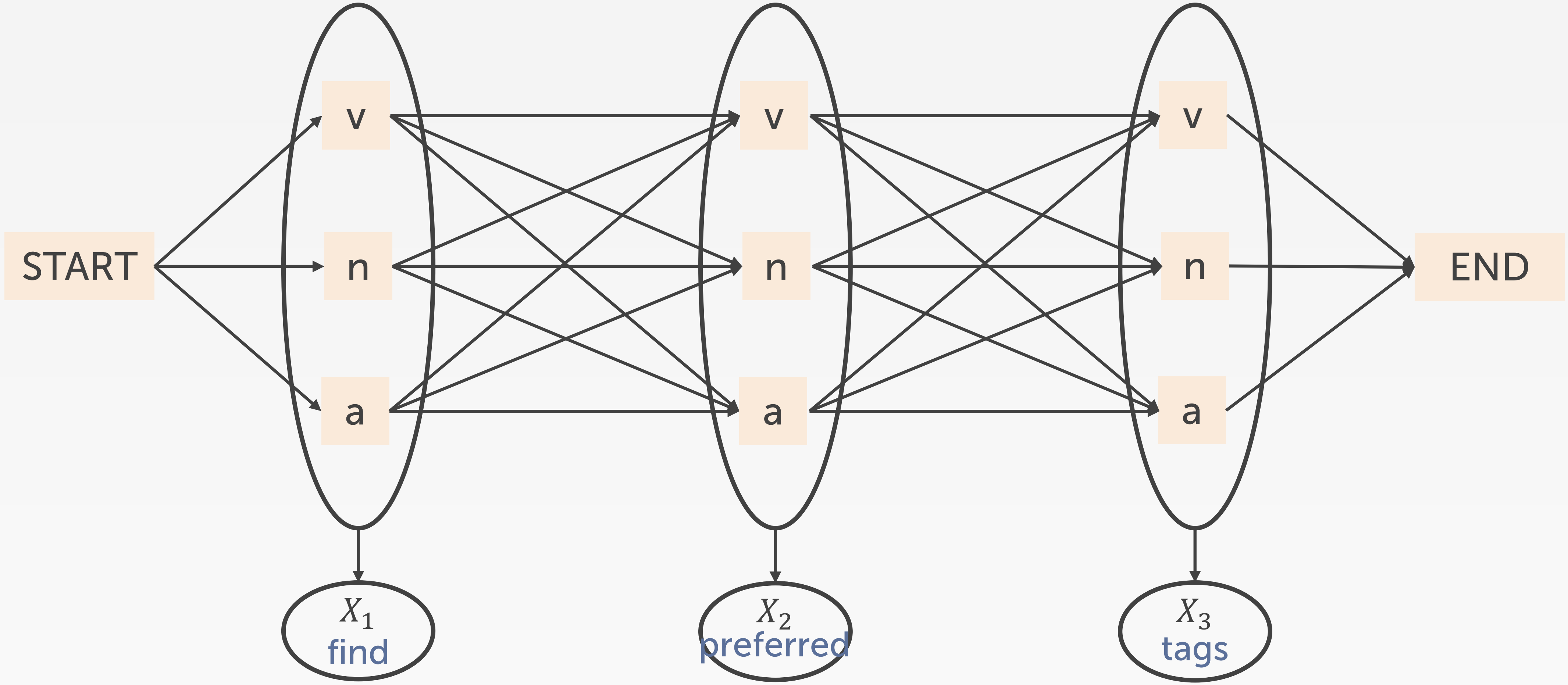
Part-of-Speech (POS) Tagging



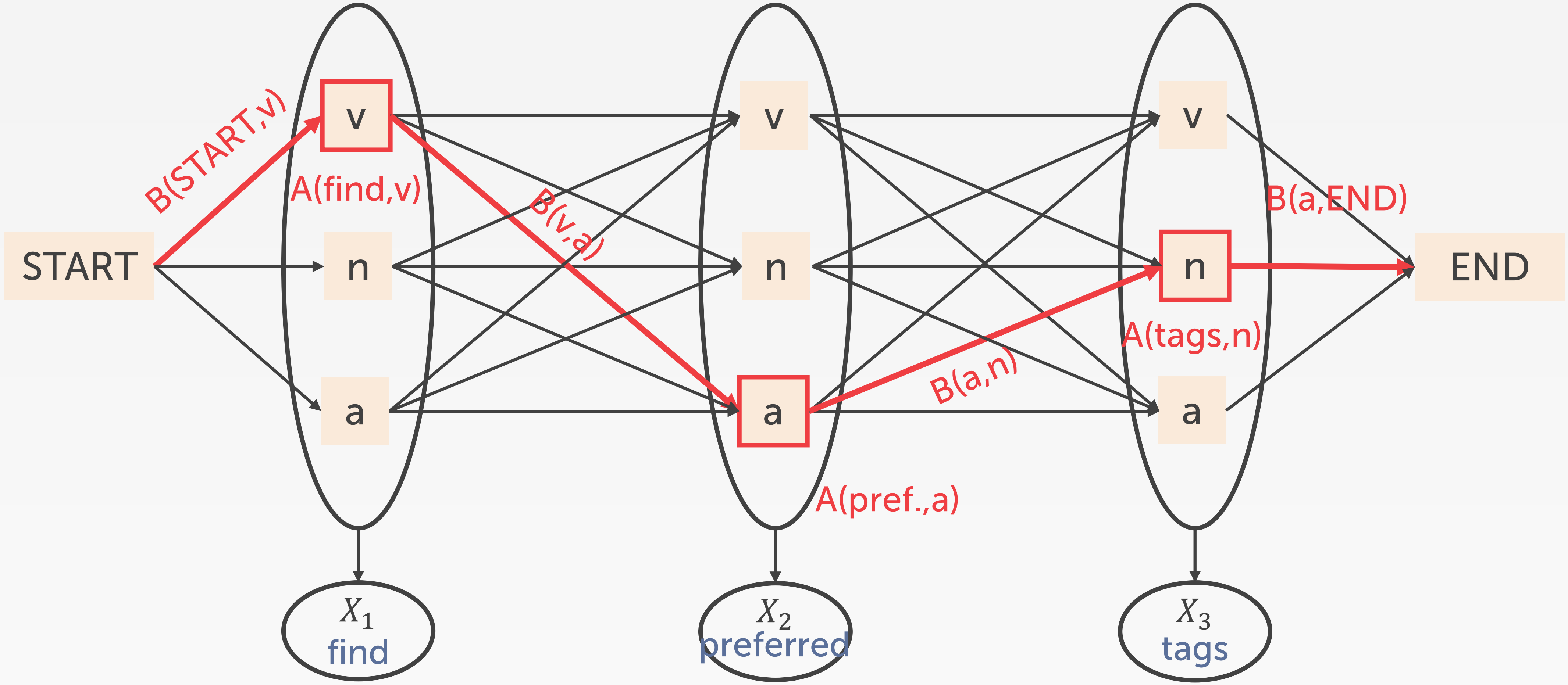
Forward-Backward Algorithm



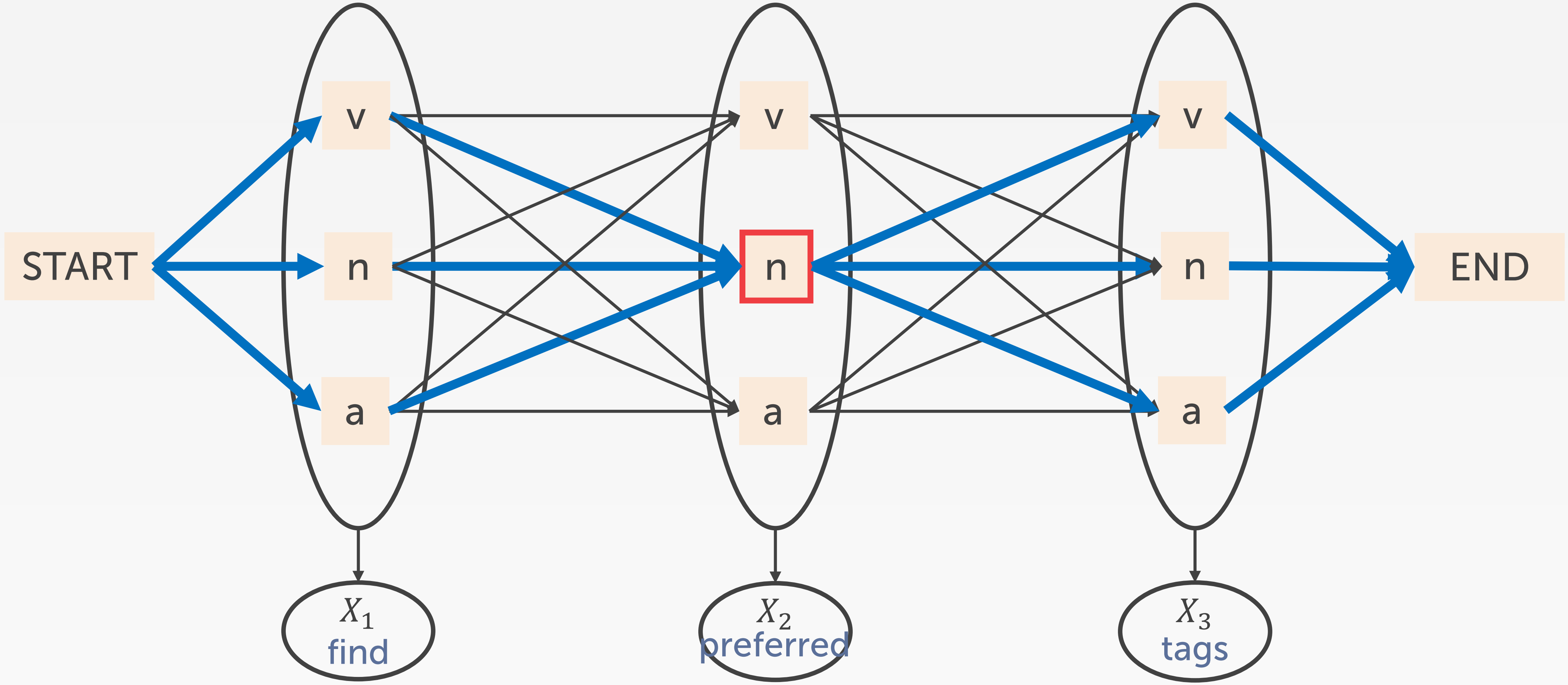
Forward-Backward Algorithm



Forward-Backward Algorithm



Forward-Backward Algorithm: Finds Marginals



Forward-Backward Algorithm

- Define $\alpha_t(k) \triangleq p(x_1, \dots, x_t, y_t = k)$, $\beta_t(k) \triangleq p(x_{t+1}, \dots, x_T \,|y_t = k)$
- Assume $y_0 = START$, $y_{T+1} = END$

1. Initialize $\alpha_0(START) = \beta_{T+1}(END) = 1$, $\alpha_0(k) = 0, \forall k \neq START$, $\beta_{T+1}(k) = 0, \forall k \neq END$
2. Forward algorithm:
   for t = 1,…, T:
       for k = 1, …, K:
           $\alpha_t(k) = p(x_t|y_t = k) \sum_{j=1}^{K} \alpha_{t-1}(j) p(y_t = k|y_{t-1} = j)$
3. Backward algorithm:
   for t = T,…,1:
       for k = 1, …, K:
           $\beta_t(k) = \sum_{j=1}^{K} p(x_{t+1}|y_{t+1} = j)\beta_{t+1}(j)p(y_{t+1} = j|y_t = k)$
4. Evaluation: $p(\vec{x}) = \alpha_{T+1}(END)$
5. Marginal: $p(y_t = k|\vec{x}) = \frac{\alpha_t(k)\beta_t(k)}{p(\vec{x})}$

Viterbi Algorithm (Decoding)

- Define $\omega_t(k) \triangleq \max_{y_1,\dots,y_{t-1}} p(x_1, \dots, x_t, y_1, \dots, y_t = k)$,

  $b_t(k) \triangleq \operatorname{argmax}_{y_1,\dots,y_{t-1}} p(x_1, \dots, x_t, y_1, \dots, y_t = k)$
- Assume $y_0 = START$

1. Initialize $\omega_0(START) = 1$, $\omega_0(k) = 0, \forall k \neq START$
2. For $t = 1, \dots T$:

   for $k = 1, \dots, K$:

   $$\omega_t(k) = \max_{j \in \{1,\dots,K\}} p(x_t|y_t = k)\omega_{t-1}(j)p(y_t = k|y_{t-1} = j)$$

   $$b_t(k) = \operatorname{argmax}_{j \in \{1,\dots,K\}} p(x_t|y_t = k)\omega_{t-1}(j)p(y_t = k|y_{t-1} = j)$$

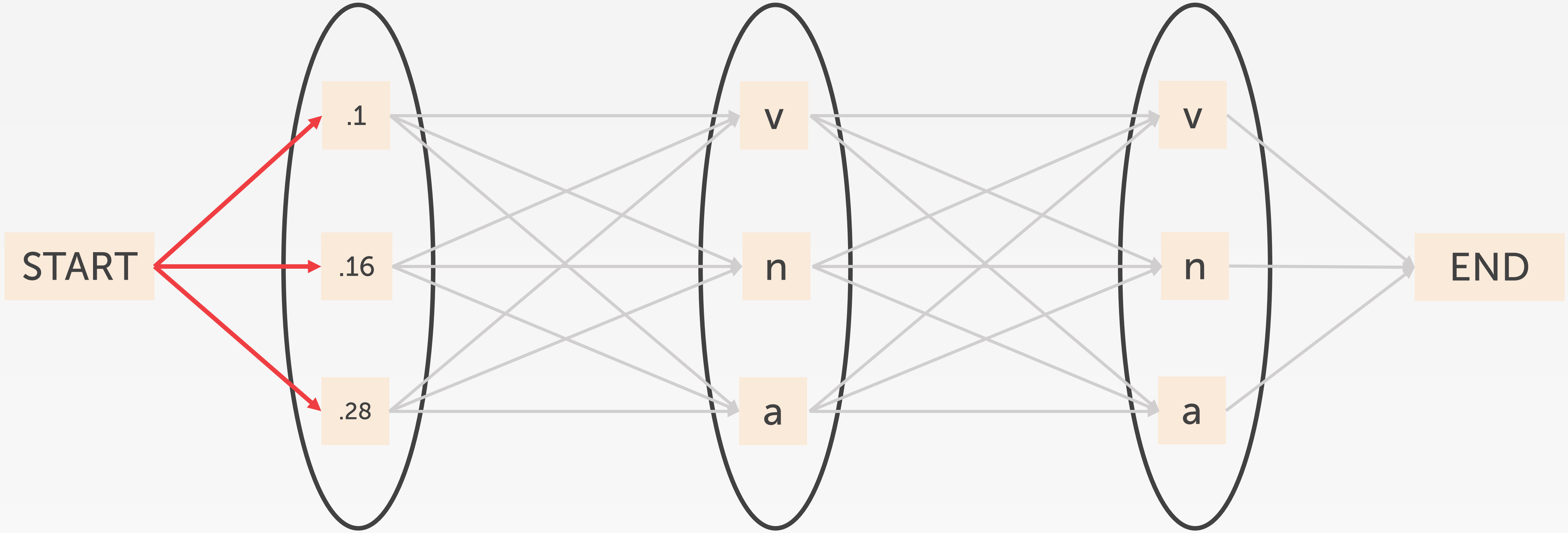
3. Compute most probable assignment

   $\widehat{y_T} = b_{T+1}(END)$

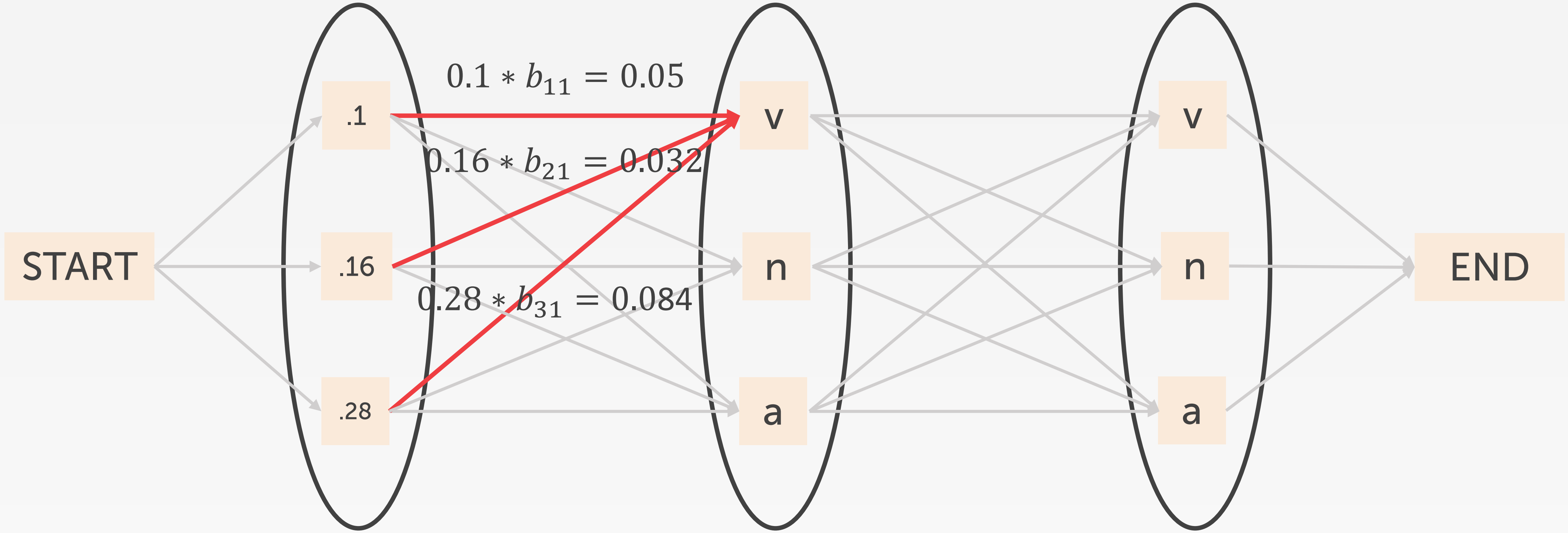
   for $t = T-1, \dots, 1$: $\widehat{y_t} = b_{t+1}\left(\widehat{y_{t+1}}\right)$

光华管理学院 Guanghua School of Management

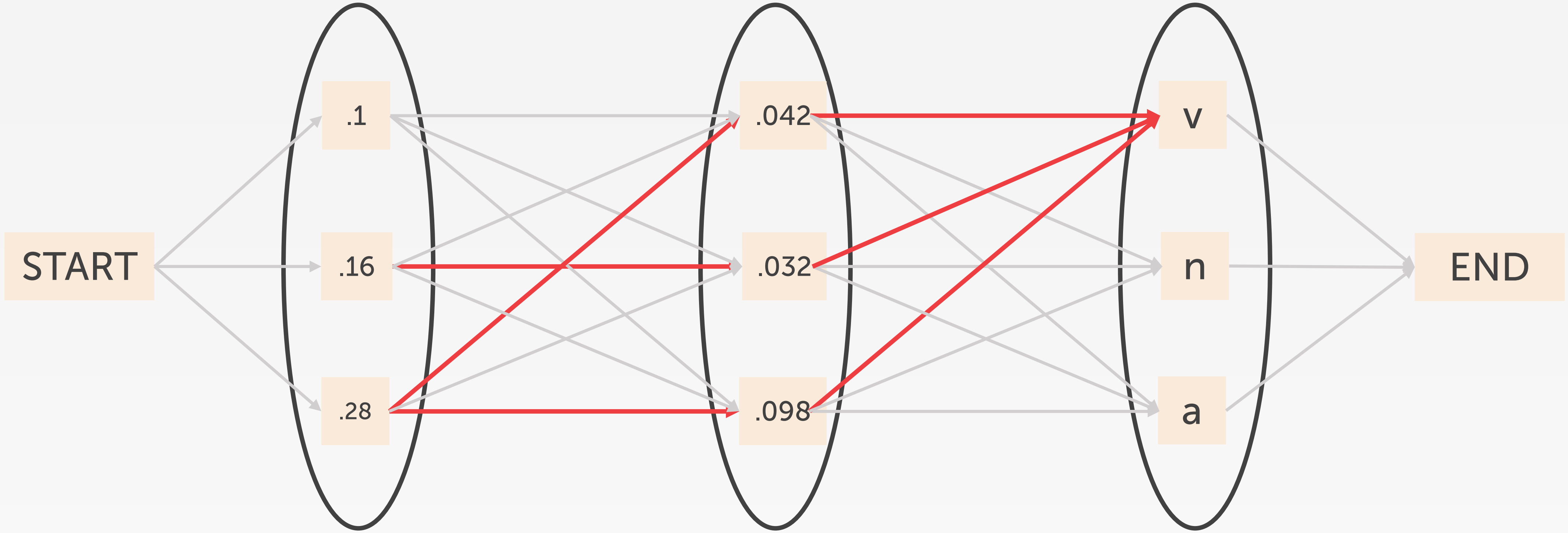
Example: Viterbi Algorithm



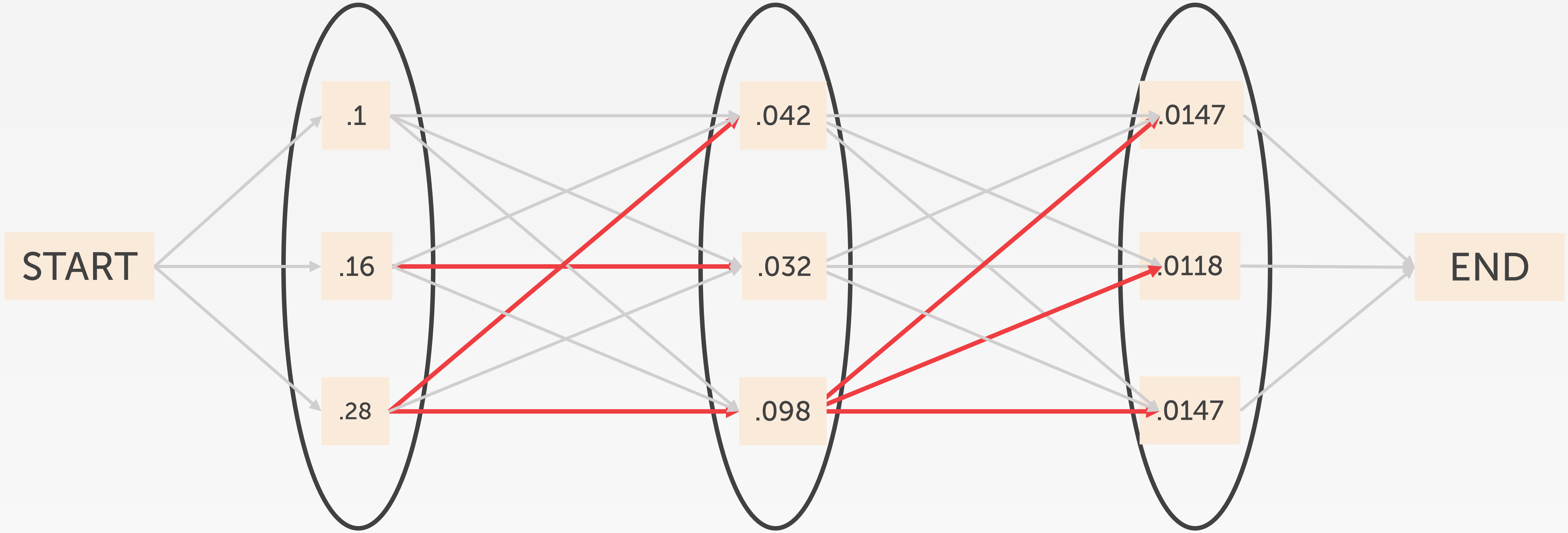
Example: Viterbi Algorithm



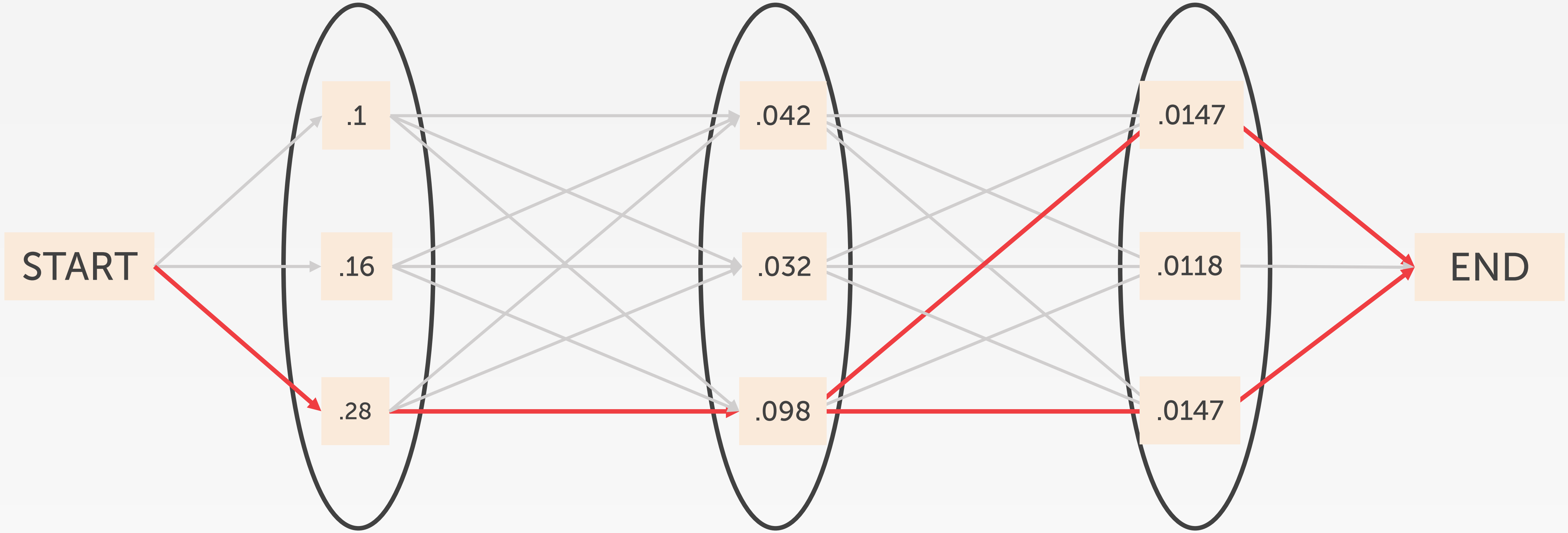
Example: Viterbi Algorithm



Example: Viterbi Algorithm



Example: Viterbi Algorithm



Unsupervised Learning

Learning Paradigms

Paradigm	Data	
Supervised	$\mathcal{D} = \{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$	$\boldsymbol{x} \sim p^*(\cdot)$ and $y = c^*(\cdot)$
↪ Regression	$y^{(i)} \in \mathbb{R}$	
↪ Classification	$y^{(i)} \in \{1, \ldots, K\}$	
↪ Binary classification	$y^{(i)} \in \{+1, -1\}$	
Unsupervised	$\mathcal{D} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$	$\boldsymbol{x} \sim p^*(\cdot)$
↪ Clustering	Predict $\{z^{(i)}\}_{i=1}^{N}$ where $z^{(i)} \in \{1, \ldots, K\}$	
↪ Dimensionality Reduction	Convert each $\boldsymbol{x}^{(i)} \in \mathbb{R}^M$ to $\boldsymbol{u}^{(i)} \in \mathbb{R}^K$ with $K \ll M$	
Semi-supervised	$\mathcal{D} = \{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=1}^{N_1} \cup \{\boldsymbol{x}^{(j)}\}_{j=1}^{N_2}$	
Reinforcement Learning	$\mathcal{D} = \{(s^{(1)}, a^{(1)}, r^{(1)}), (s^{(2)}, a^{(2)}, r^{(2)}), \ldots\}$	

Goals

- To discover interesting things from the data:
  - Is there an informative way to visualize the data?
  - Can we discover subgroups among the variables?

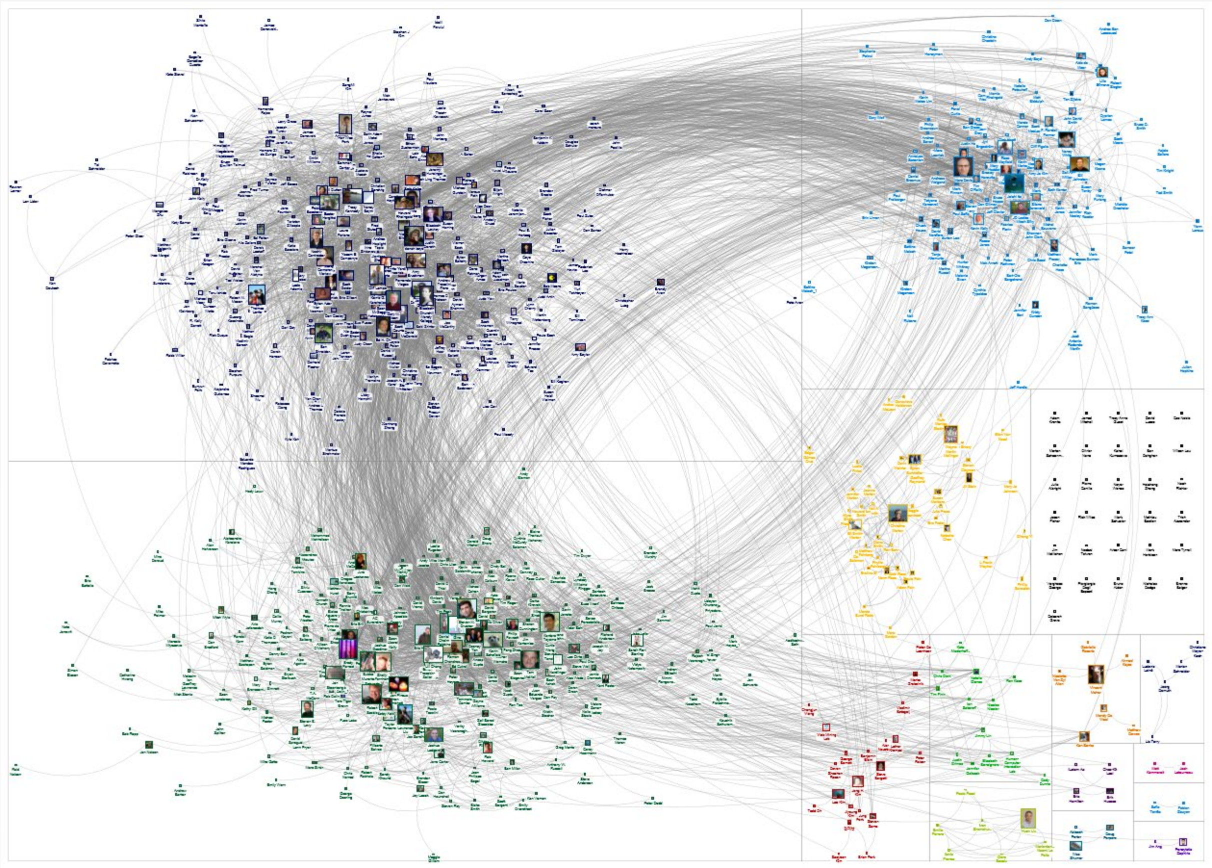
- Models:
  - Clustering
    - K-means
    - DBSCAN
    - Hierarchical Clustering

Clustering

Clustering

- Partition **unlabeled** data into groups (clusters)
- Points within a cluster should be "similar"
- Points in different clusters should be "different"

Applications



K-Means

Overview

- K-means (MacQueen, 1967)
- Each cluster has a cluster center, called centroid
- K is specified by the user

K-means Algorithm

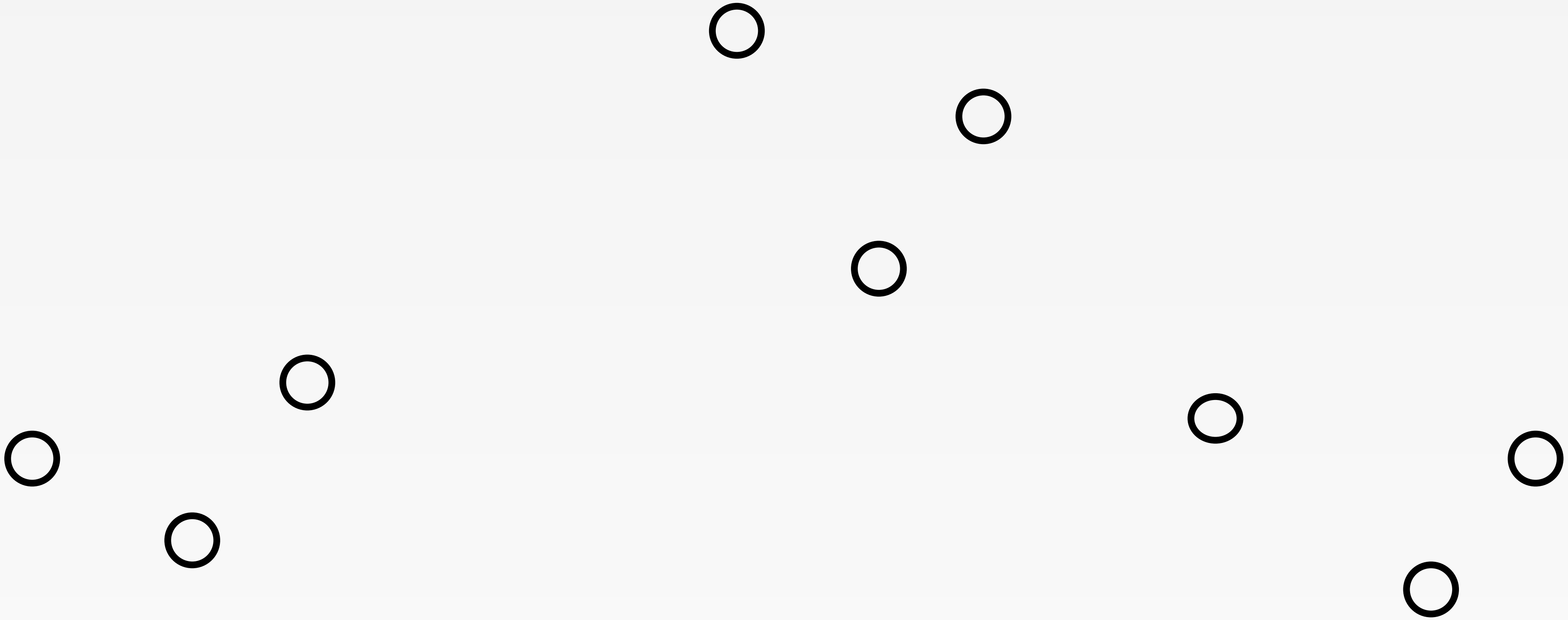
- Given K and unlabeled feature vectors $D = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(N)}\}$
- Initialize cluster center $c = \left\{\boldsymbol{c}^{(1)}, \ldots, \boldsymbol{c}^{(K)}\right\}$ and cluster assignments $z = \left\{z^{(1)}, z^{(2)}, \ldots, z^{(N)}\right\}$
- Repeat until convergence:
  - For j in {1,…,K}

    $\boldsymbol{c}^{(j)}$ is the mean of all points assigned to cluster j
  - for i in {1,…,N}

    $z^{(i)}$ is the index j of cluster center nearest to $\boldsymbol{x}^{(i)}$

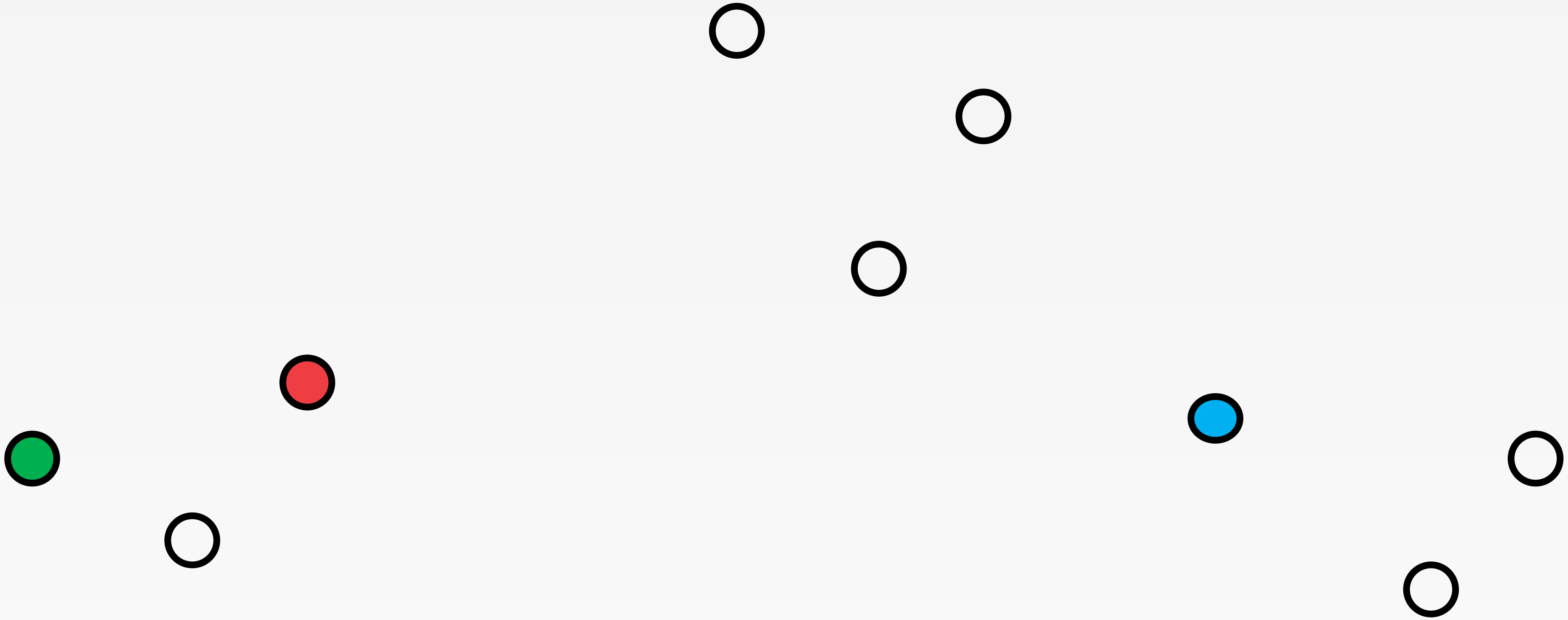
Illustrative Example

Given a set of data points



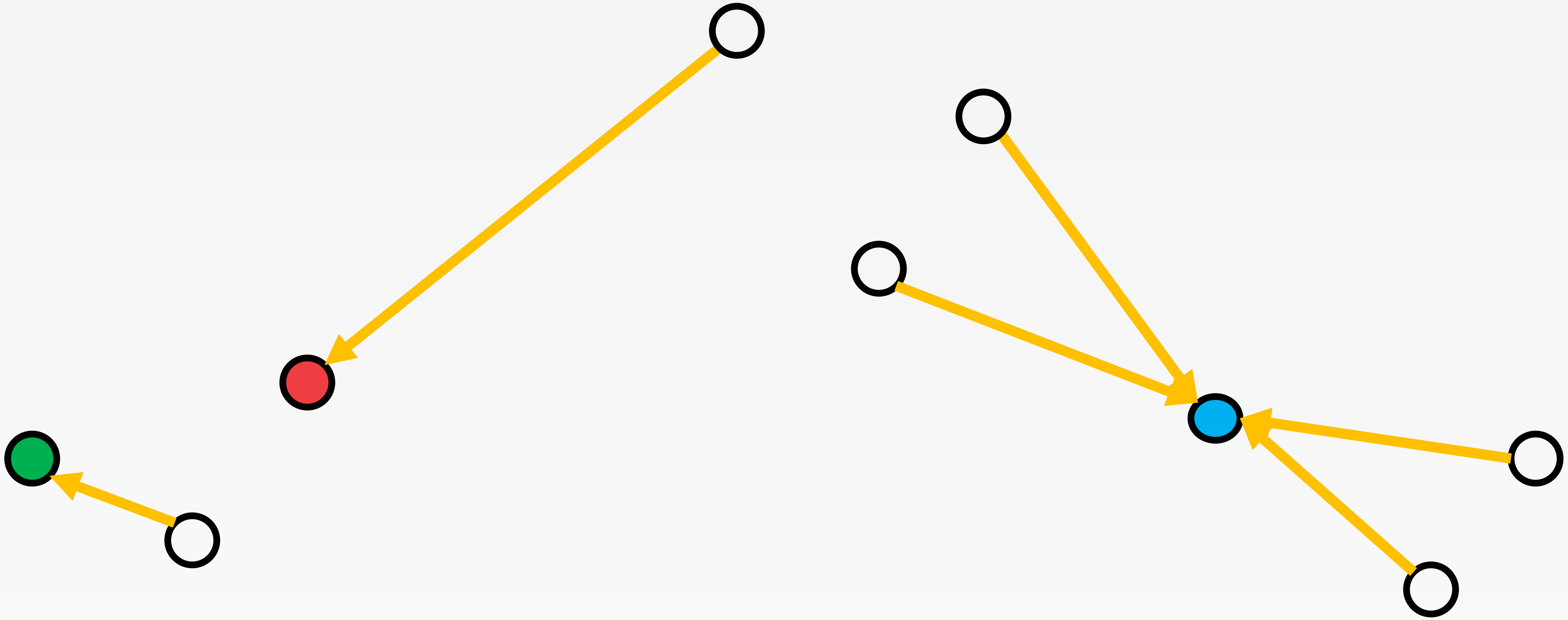
Illustrative Example

Select initial centers at random (k=3)



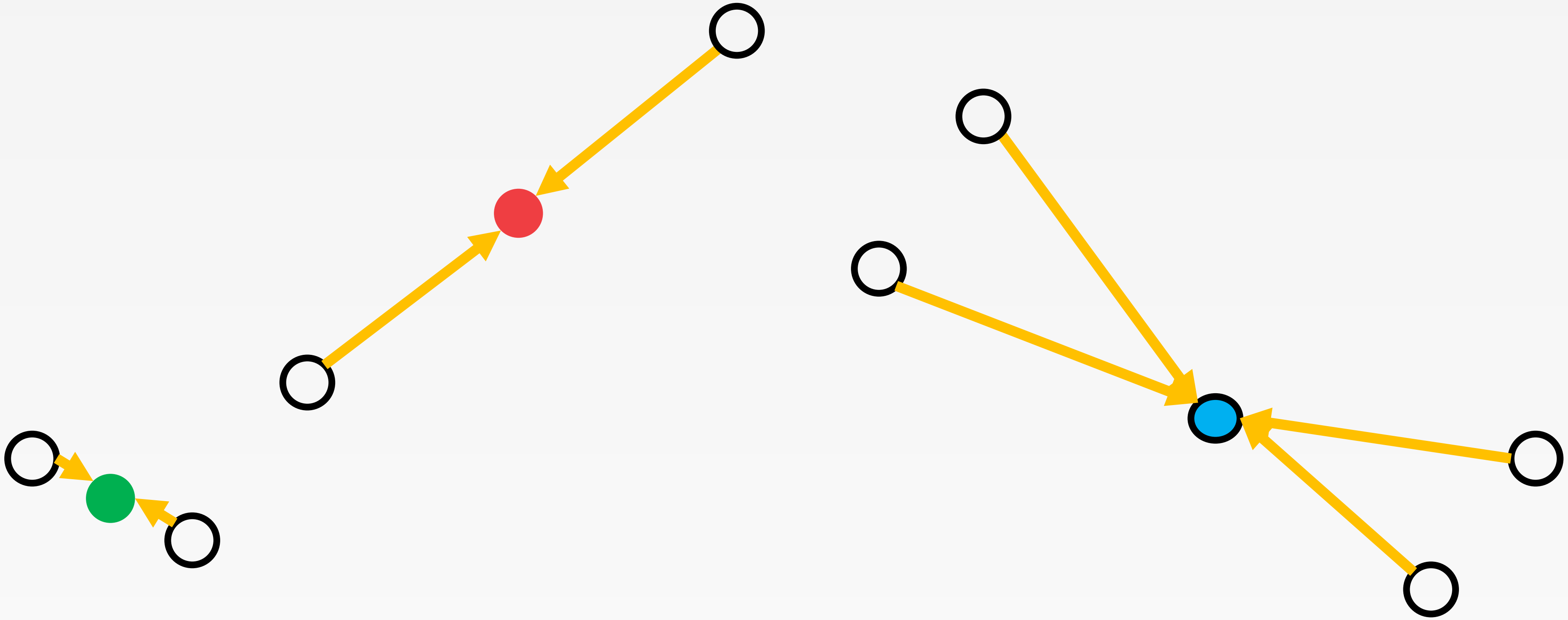
Illustrative Example

Assign each point to its nearest center



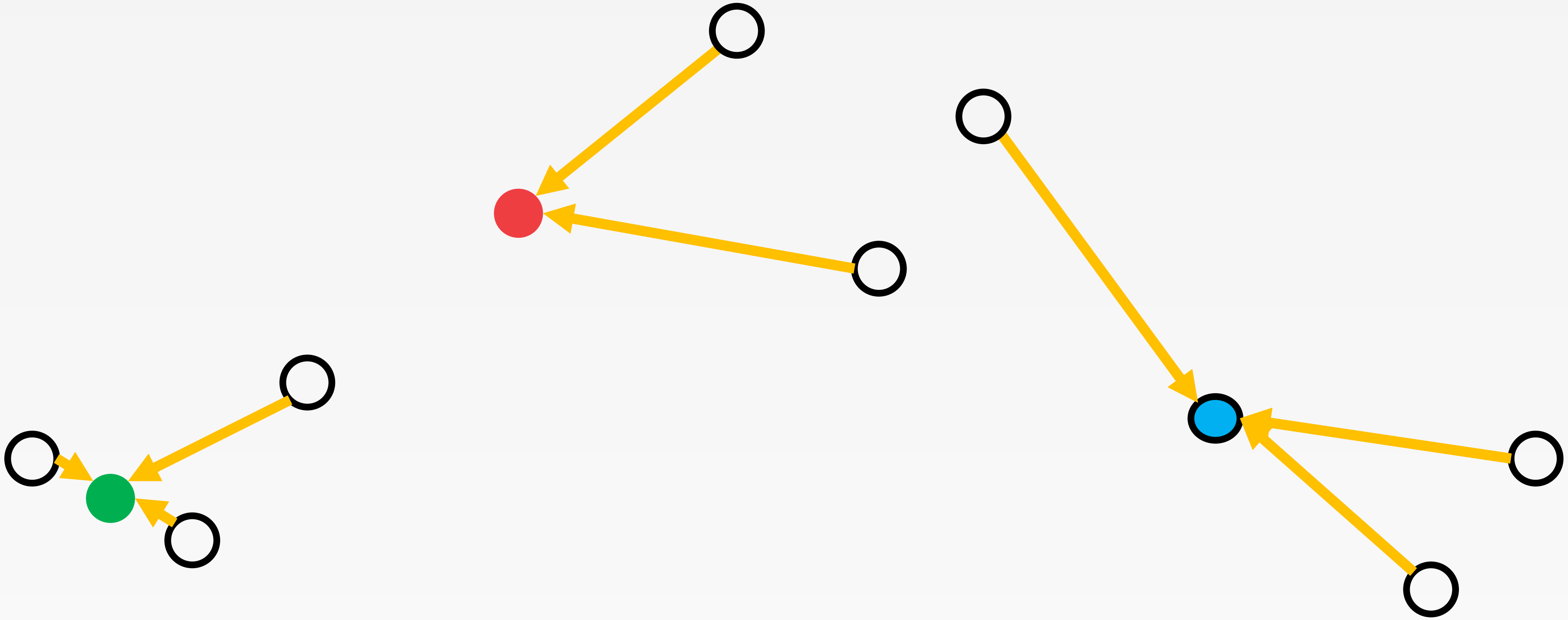
Illustrative Example

Recompute optimal centers given a fixed clustering



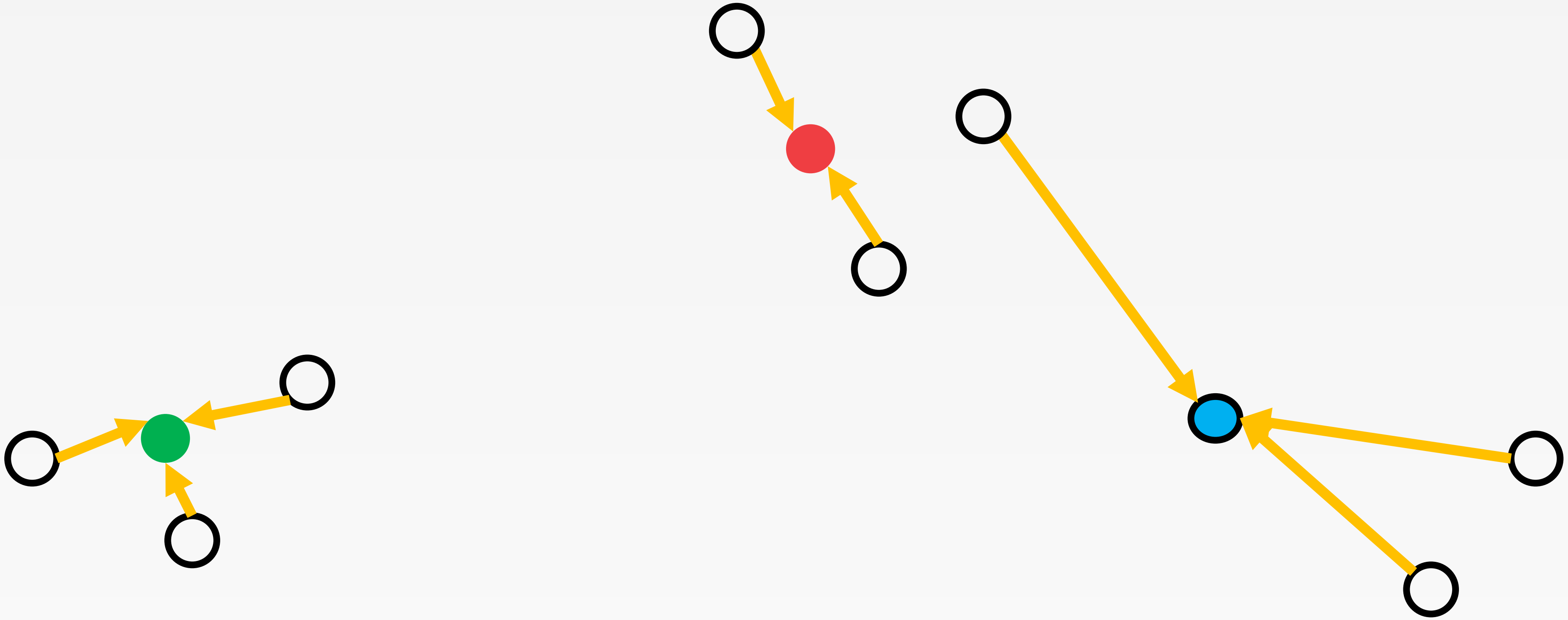
Illustrative Example

Assign each point to its nearest center



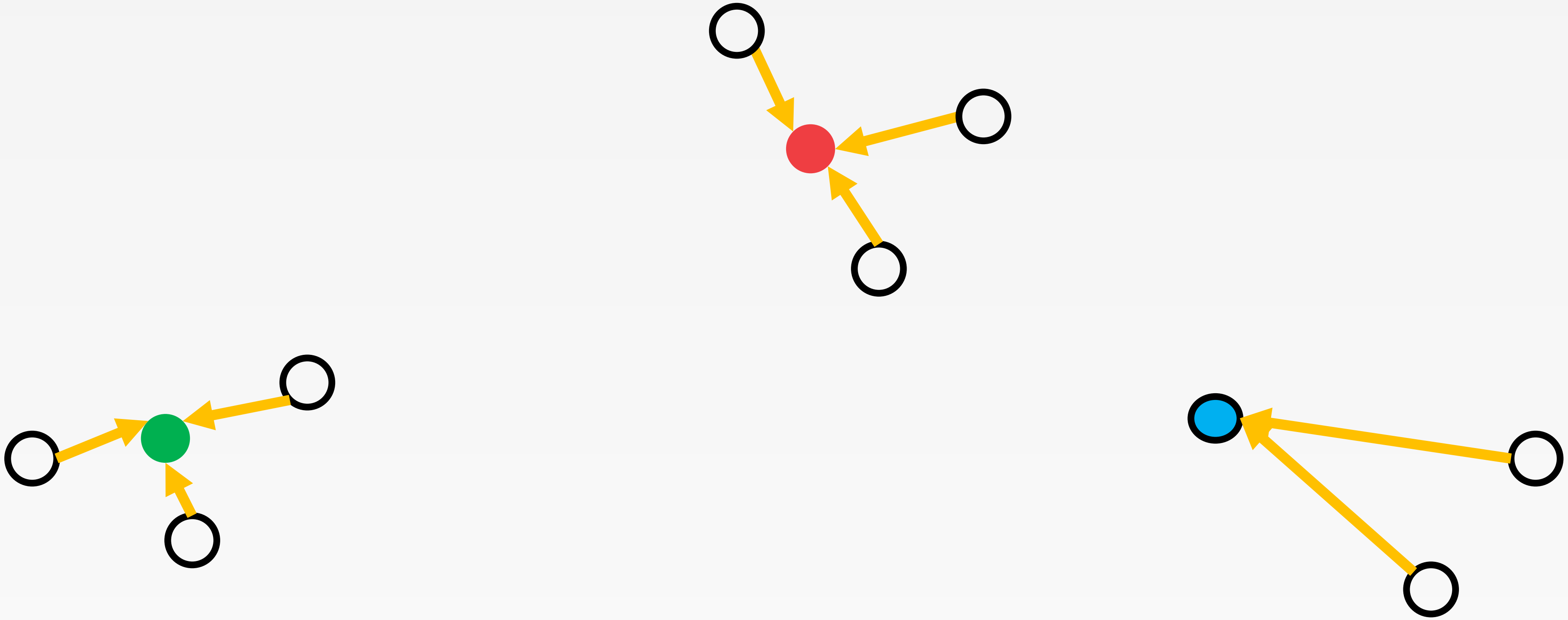
Illustrative Example

Recompute optimal centers given a fixed clustering



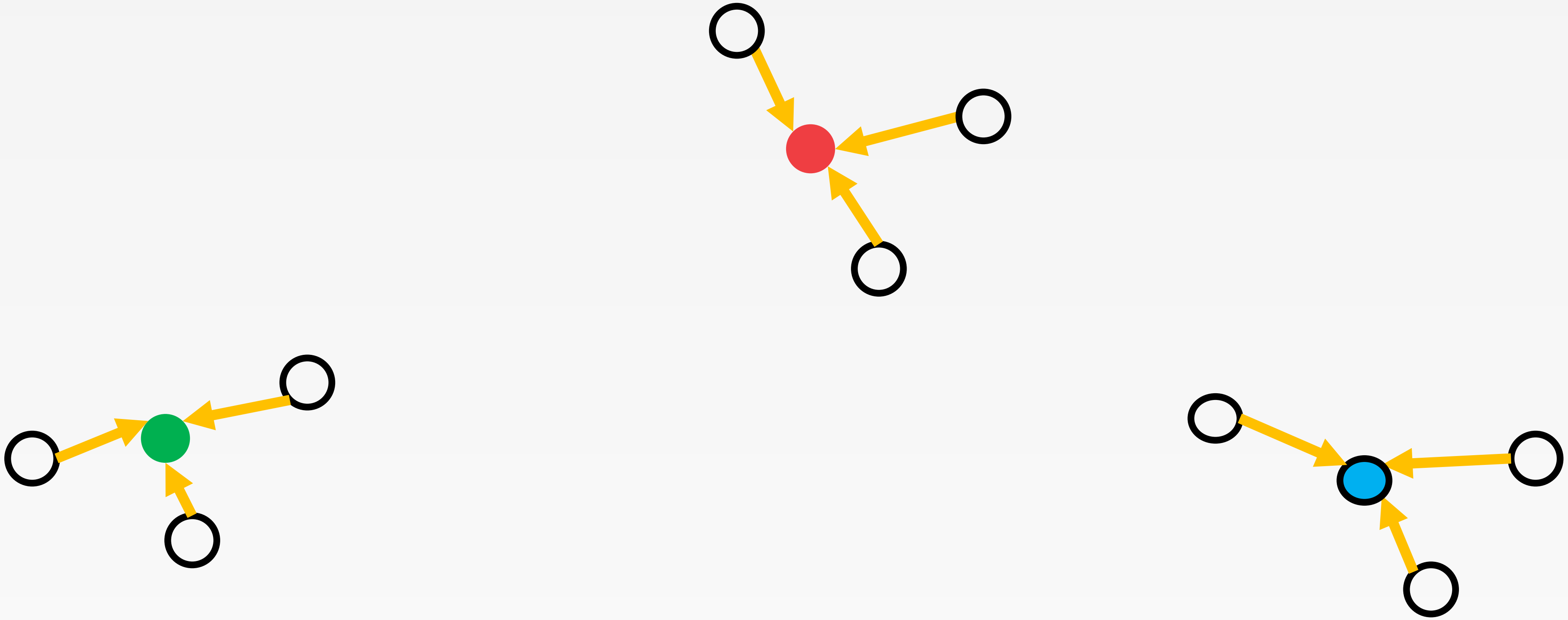
Illustrative Example

Assign each point to its nearest center



Illustrative Example

Recompute optimal centers given a fixed clustering



Measure the Distance

- Similarity measure (distance measure)
  - Euclidean distance $d(x,y) = \sqrt{(x-y)^2} = \sqrt{\sum_{i=1}^{d}(x_i - y_i)^2}$
  - Manhattan distance $d(x,y) = |x-y| = \sum_{i=1}^{d}|x_i - y_i|$

Stopping Criterion

- no (or minimum) re-assignments of data points to different clusters, *or*
- no (or minimum) change of centroids, or
- minimum decrease in the **sum of squared error**(SSE),

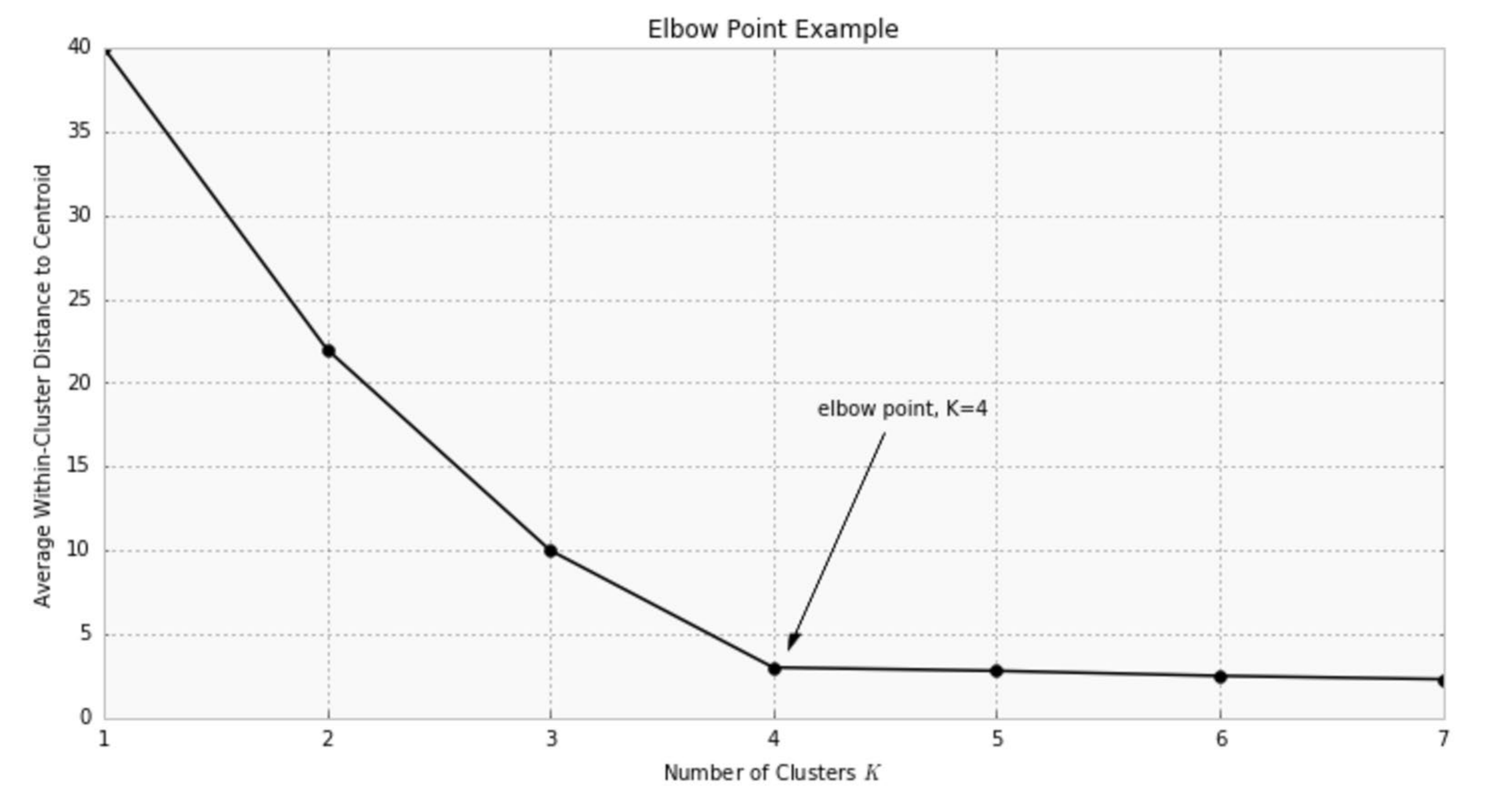
How to choose k?

Elbow method:

run k-means clustering on the dataset for a range of values of *k*

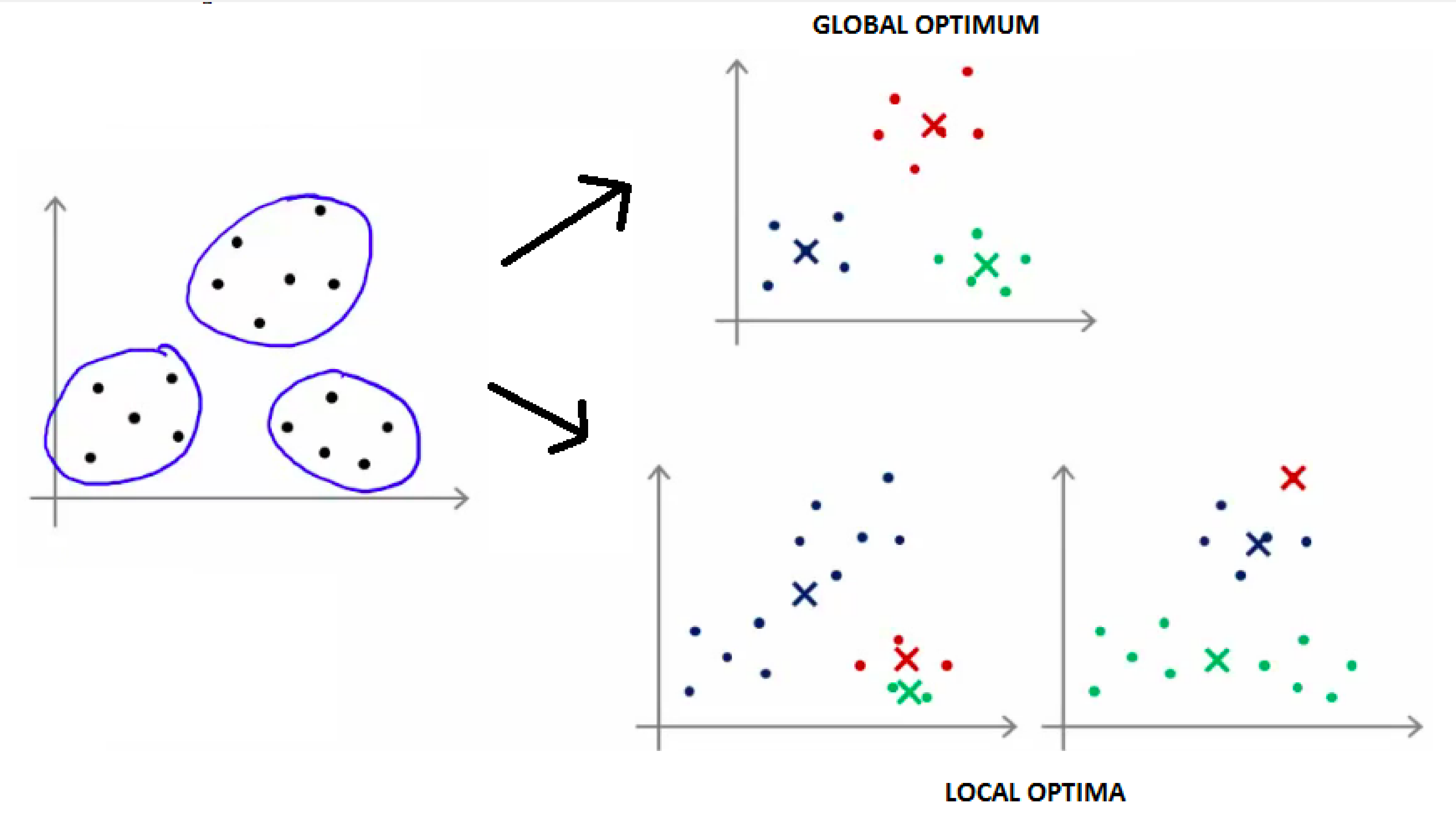
for each value of *k* calculate the sum of squared errors (SSE)

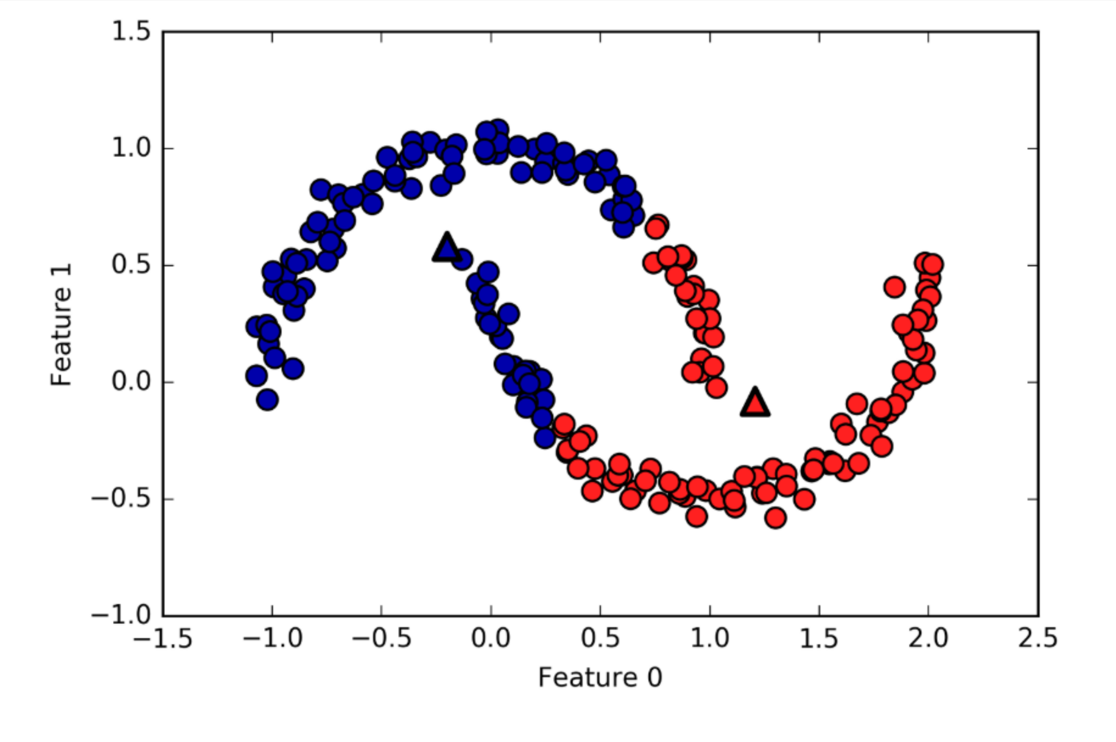
If the line chart looks like an arm, then the "elbow" on the arm is the value of *k* that is the best

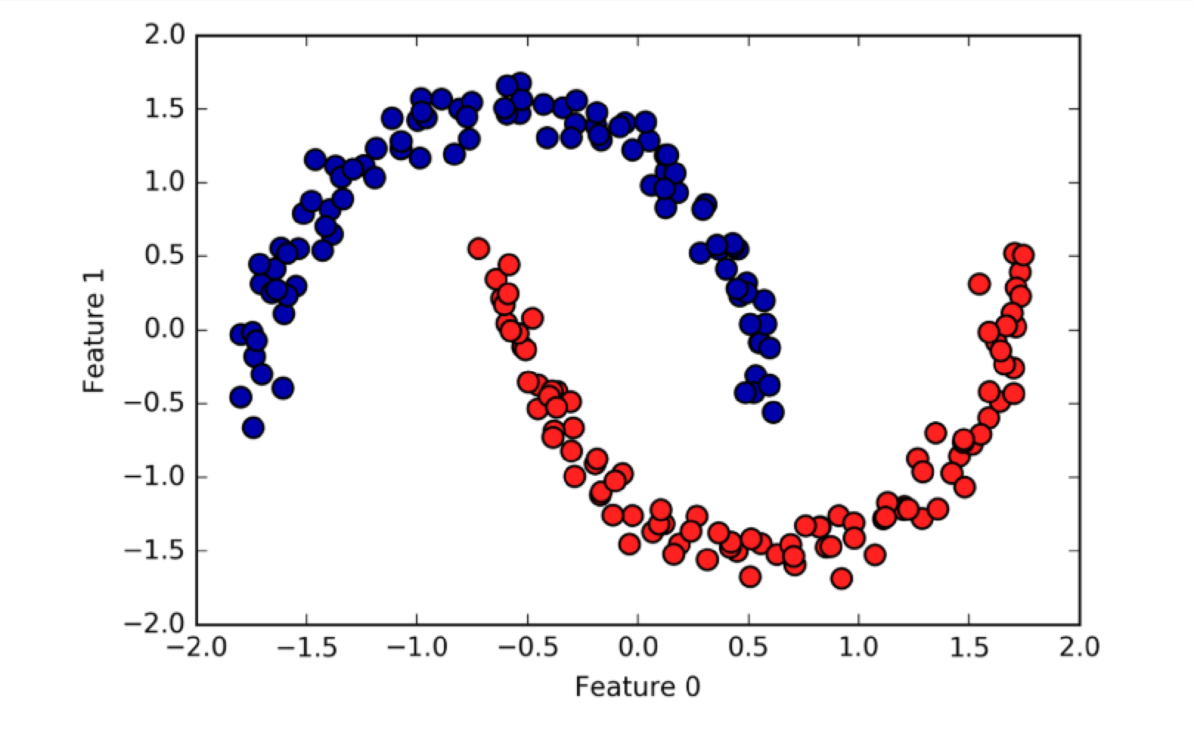

Elbow Point Example
Average Within-Cluster Distance to Centroid
0
5
10
15
20
25
30
35
40
1
2
3
4
5
6
7
Number of Clusters K
elbow point, K=4

Pros and Cons

- Strengths:
  - Simple: each to understand and to implement
  - Efficient
- Weakness:
  - The algorithm is sensitive to outliers
  - it terminates at a local optimum if SSE is used. The global optimum is hard to find due to complexity
    - Might be sensitive to initial seeds
  - Only simple cluster shapes


GLOBAL OPTIMUM
LOCAL OPTIMA


Feature 1
Feature 0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5


Feature 1
Feature 0
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

DBSCAN

Density-Based Spatial Clustering of Applications with Noise

Source: https://cse.buffalo.edu/~jing/cse601/fa12/materials/clustering_density.pdf

Density-based Clustering

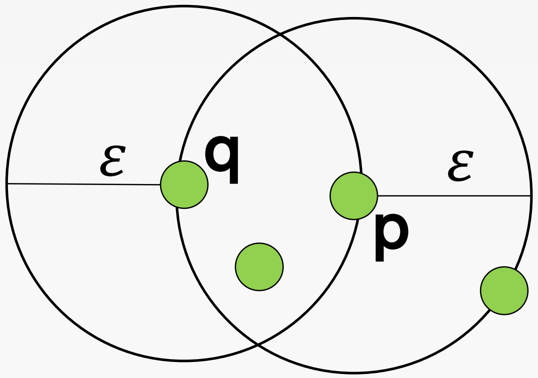
- Basic Idea:
  - Clusters are dense regions in the data space, separated by regions of lower object density
  - A cluster is defined as a maximal set of density-connected points

Density Definition

- ε-Neighborhood – Objects within a radius of $\boldsymbol{\varepsilon}$ from an object

$$N_\varepsilon(p): \{q | d(p, d) \leq \varepsilon\}$$

- “High density” -- ε-Neighborhood of an object contains at least $\boldsymbol{MinPts}$ of objects

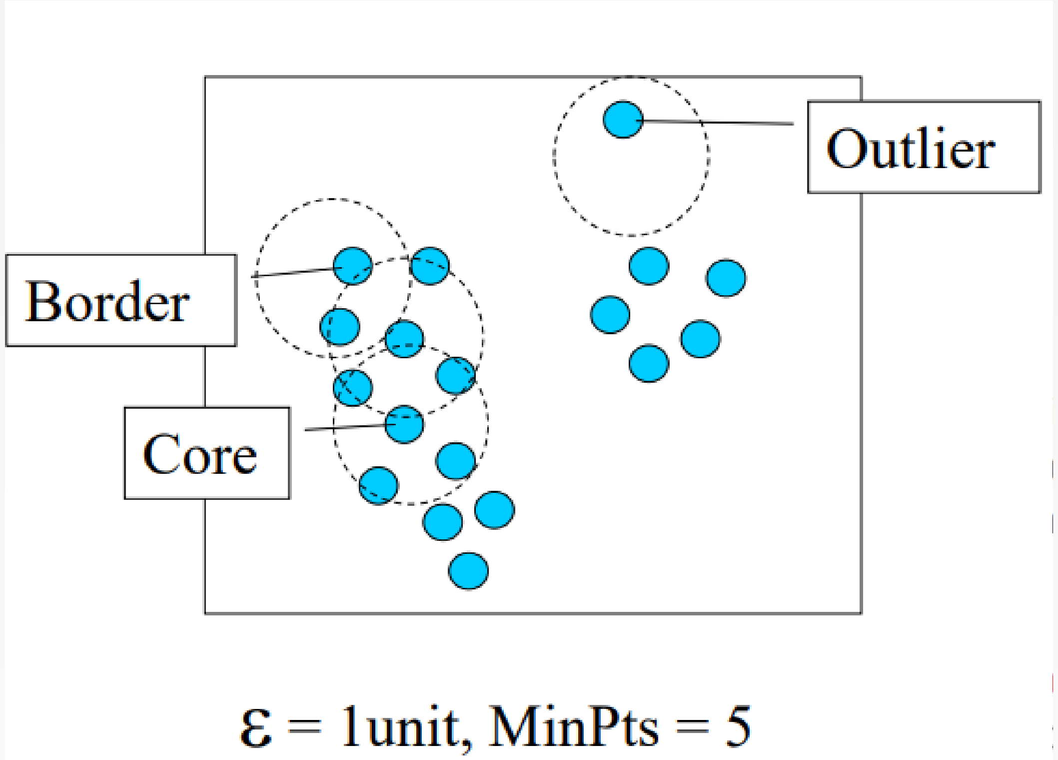


Density of p is “high” ($MinPts$ = 4)

Density of q is “low” ($MinPts$ = 3)

Core, Border, Outlier

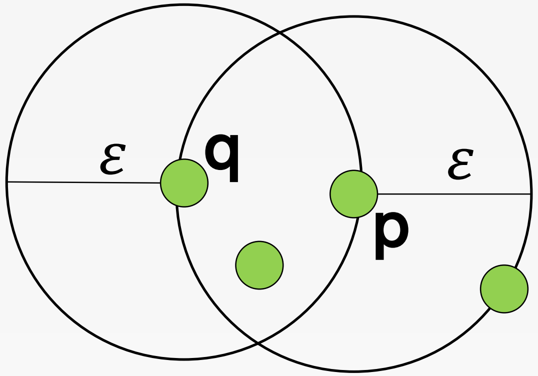
- Given ε and $MinPts$, categorize the objects into three exclusive groups:
  - <u>Core point</u>: has more than $MinPts$ points within ε (these are points that are at the interior of a cluster)
  - <u>Border point</u>: has fewer than $MinPts$ within ε, but is the neighborhood of a core point
  - <u>Noise point</u>: any point that is neither a core nor a border point



光华管理学院
Guanghua School of Management

Density-reachability

- An object q is directly density-reachable from object p if p is a core object and q is in p's ε-neighborhood.



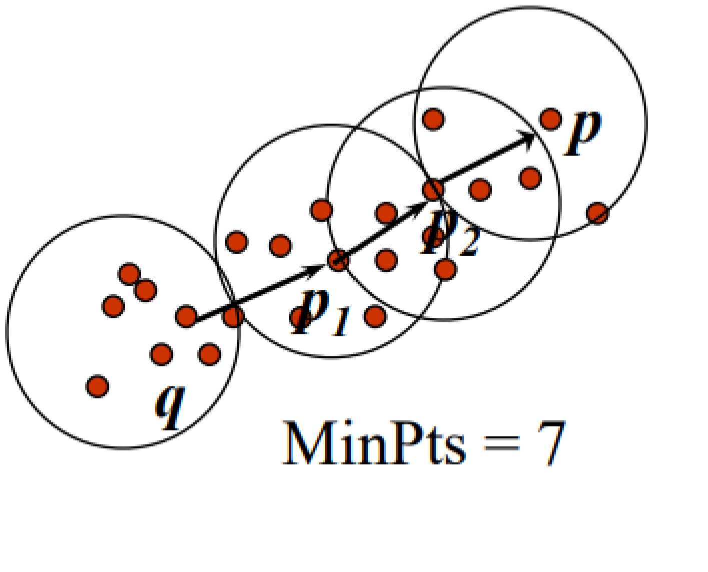
$MinPts$=4

q is directly density-reachable from p

p is not directly density-reachable from q

Density-reachability is asymmetric

Density-reachability



A point p is directly density-reachable from p_2

p_2 is directly density-reachable from p_1

p_1 is directly density-reachable from q

$p \leftarrow p_2 \leftarrow p_1 \leftarrow q$ form a chain

DBSCAN Algorithm

for each $o \in D$ **do**

  if o is not yet classified **then**

    if $|o$**'s** ε-neighborhood$|$ < $MinPts$

      assign o to *NOISE*

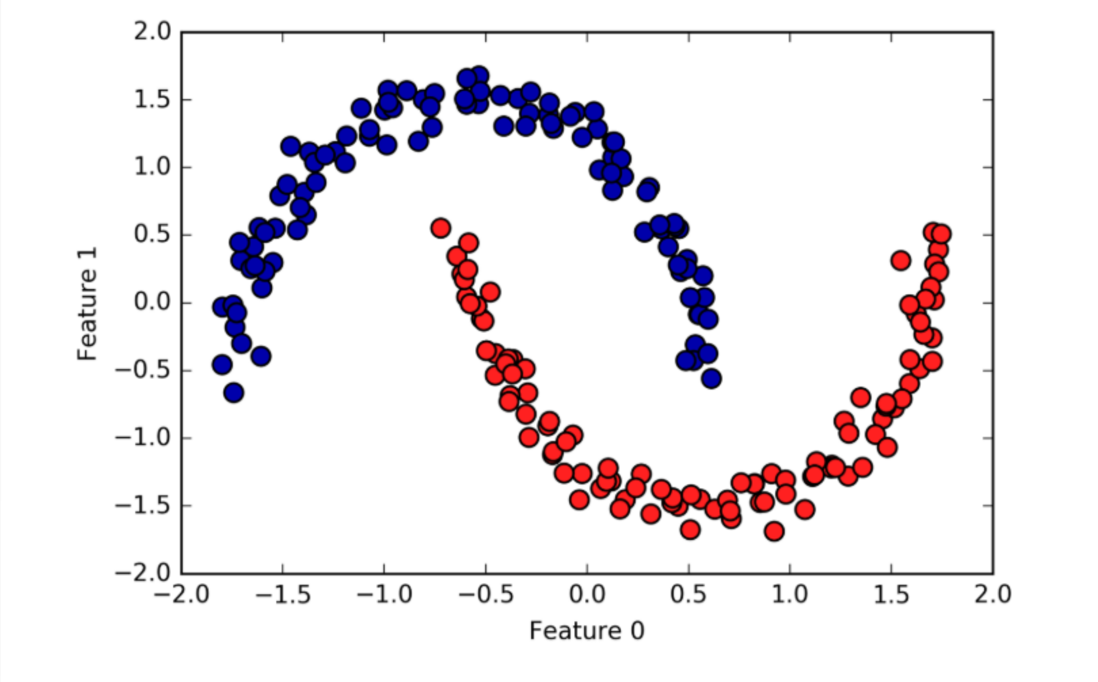
    else

      collect all objects density-reachable from o

      and assign them to a new cluster

Pros and Cons

- Can learn arbitrary cluster shapes (resistant to noise)
- Can detect outliers
- Needs two parameters to adjust

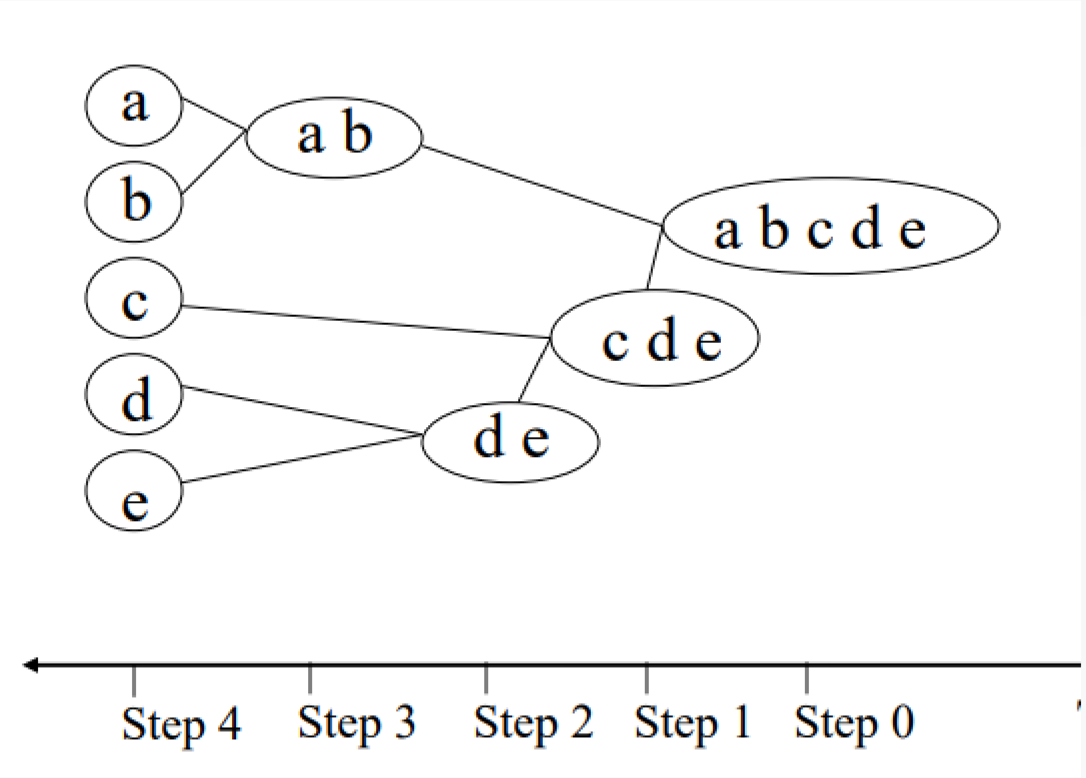


Hierarchical Clustering

https://cse.buffalo.edu/~jing/cse601/fa12/materials/clustering_hierarchical.pdf

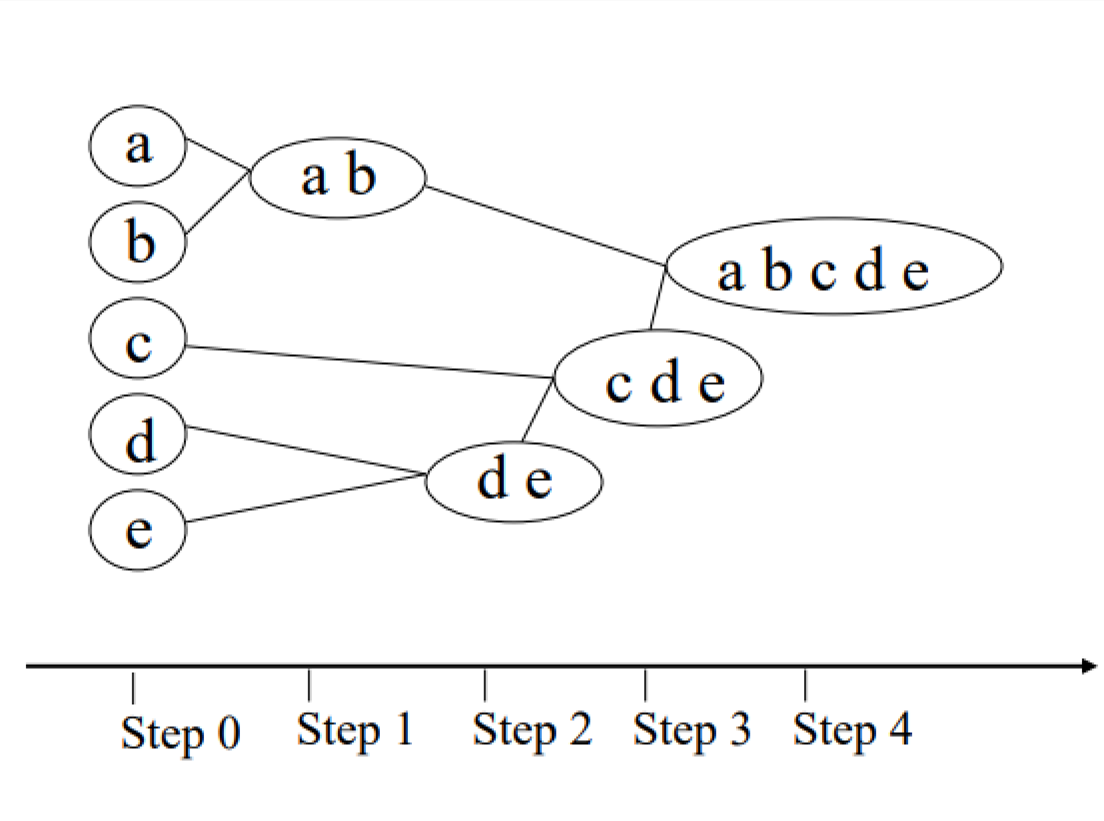
Types

- Divisive (top-down) clustering
  - All objects in one cluster
  - Select a cluster and split it into two sub clusters
  - Until each leaf cluster contains only one object



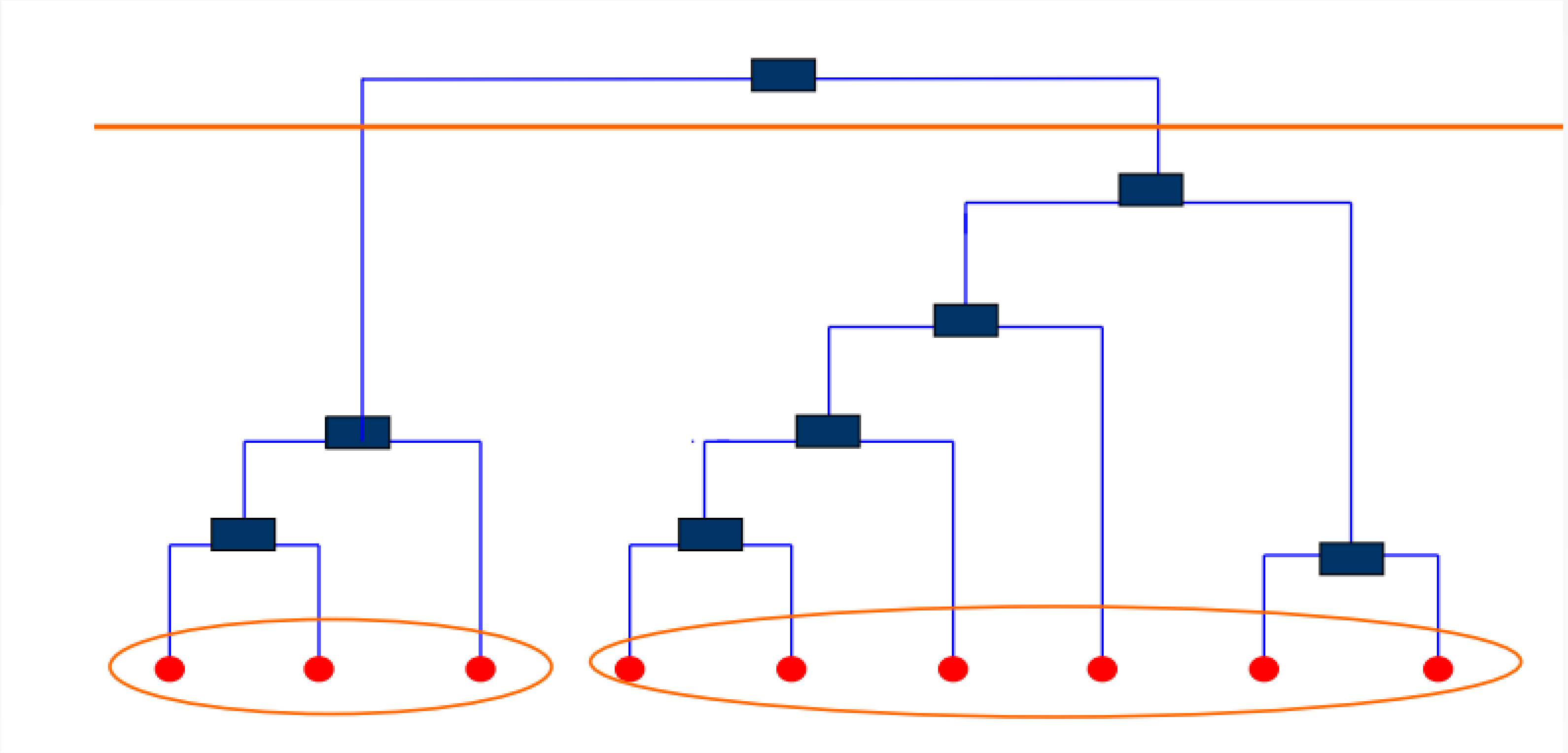
Types

- Agglomerative (bottom-up) clustering
  - Each object is a cluster
  - Merge two clusters which are most similar to each other
  - Until all objects are merged into a single cluster

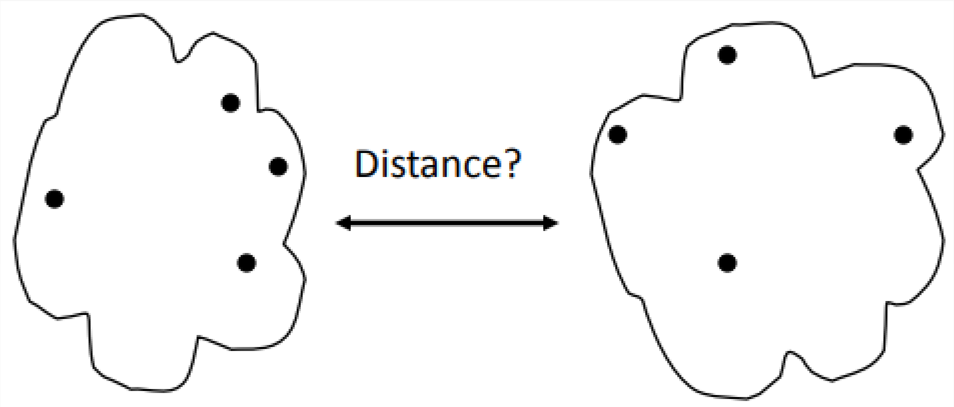


Dendrogram

- A tree that shows how clusters are merged/split hierarchically
- Each node on the tree is a cluster; each leaf node is a singleton cluster
- A clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster

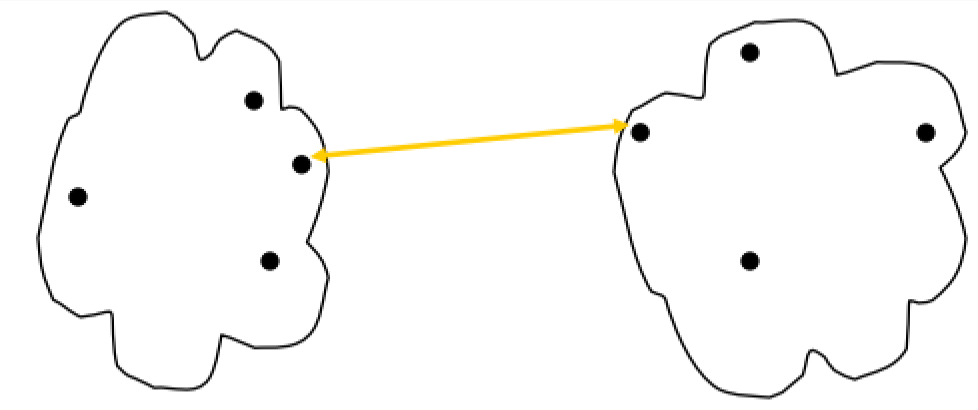


Inter-Cluster Distance

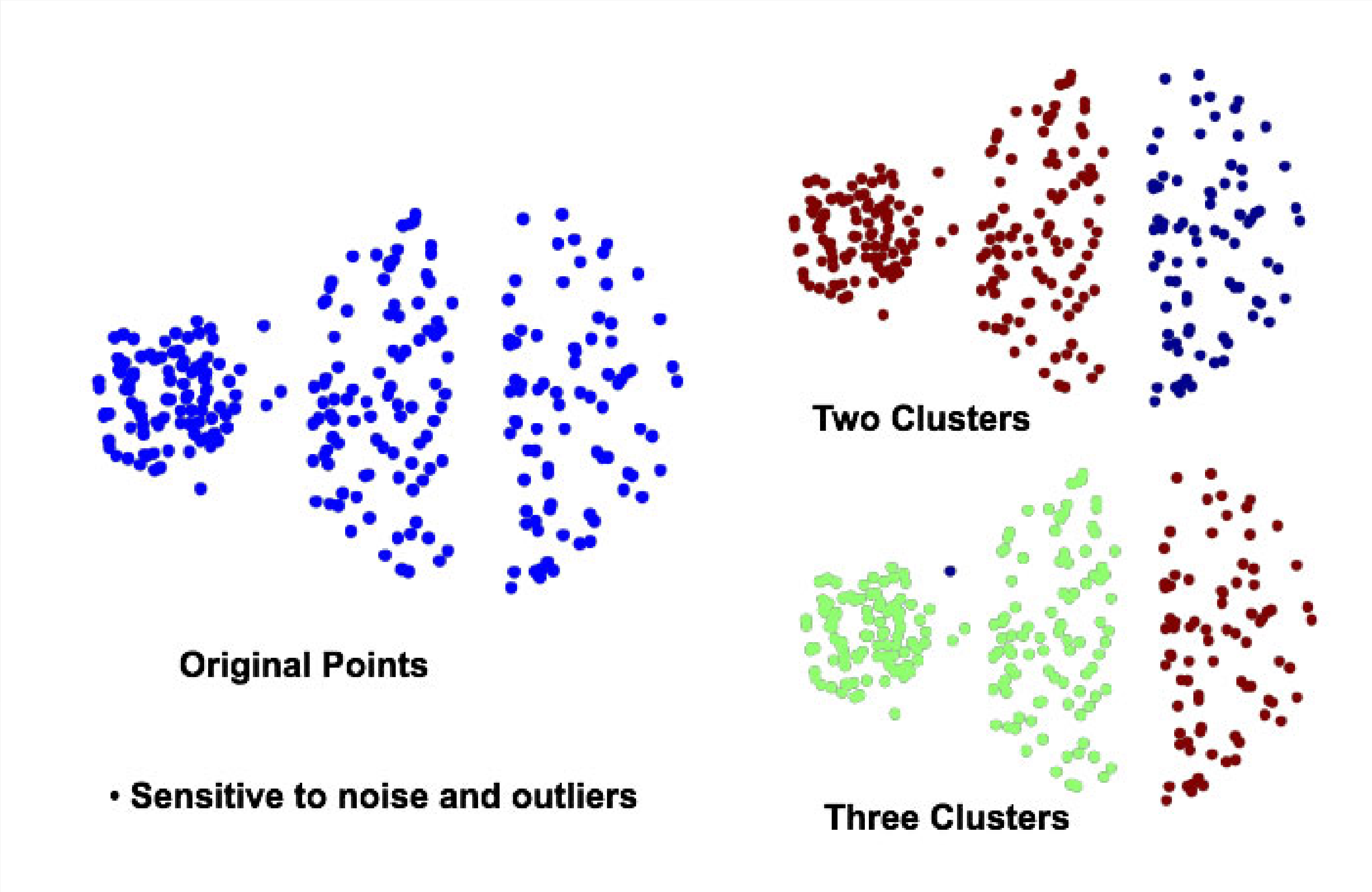


MIN (Single Link)

- The distance between two clusters is represented by the distance of the closest pair of data objects belonging to different clusters.
- Determined by one pair of points, i.e., by one link in the proximity graph

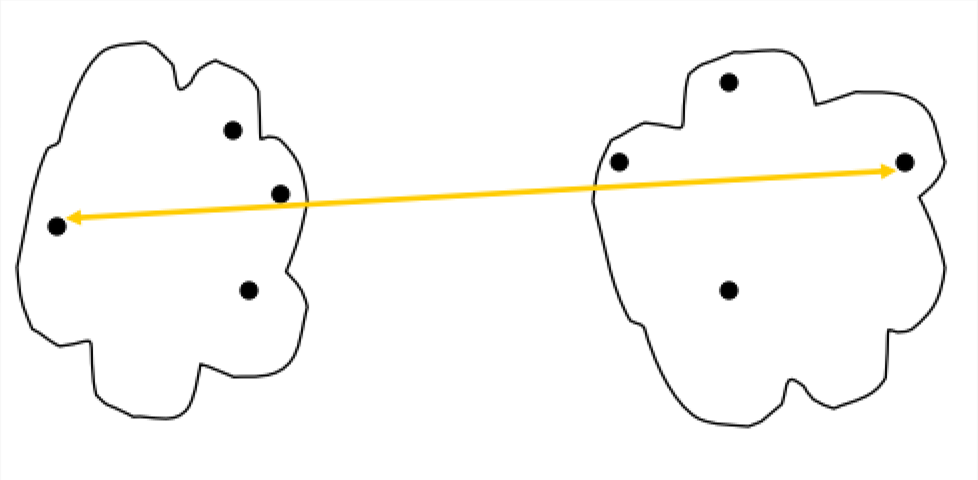


- Limitation: sensitive to noise/outliers


Original Points
Two Clusters
Three Clusters
• Sensitive to noise and outliers

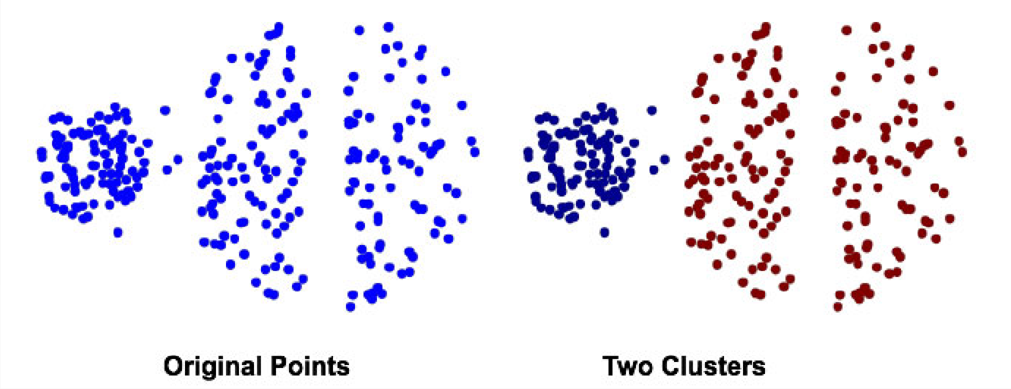
MAX (Complete link)

- The distance between two clusters is represented by the distance of the farthest pair of data objects belonging to different clusters

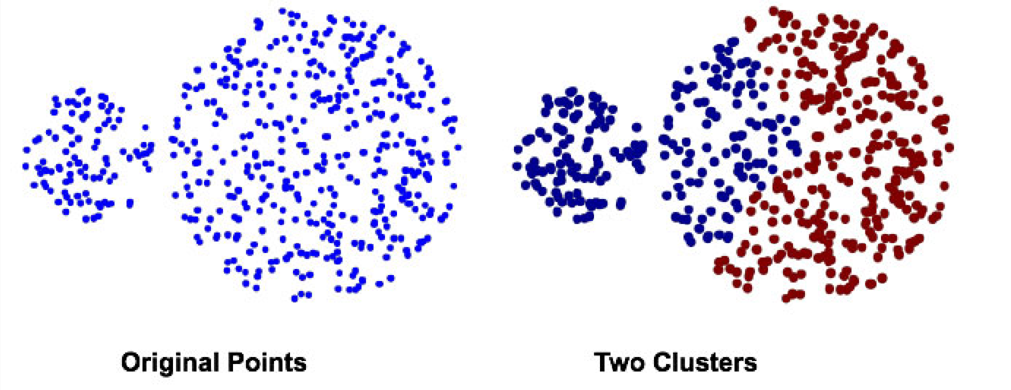


MAX (Complete link)

- Strength: less sensitive to noise/outliers

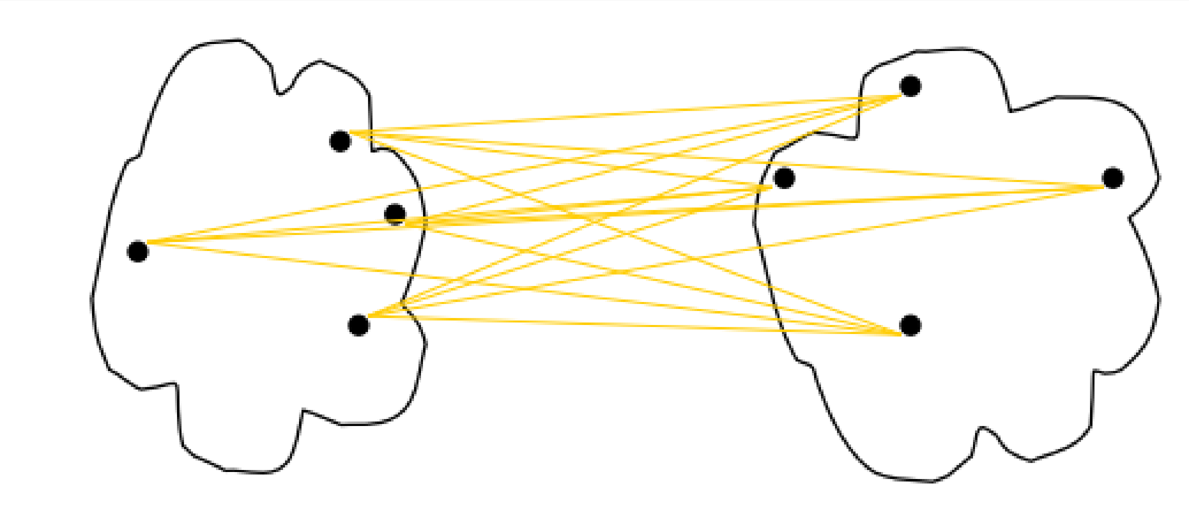


- Limitations: tends to break large clusters



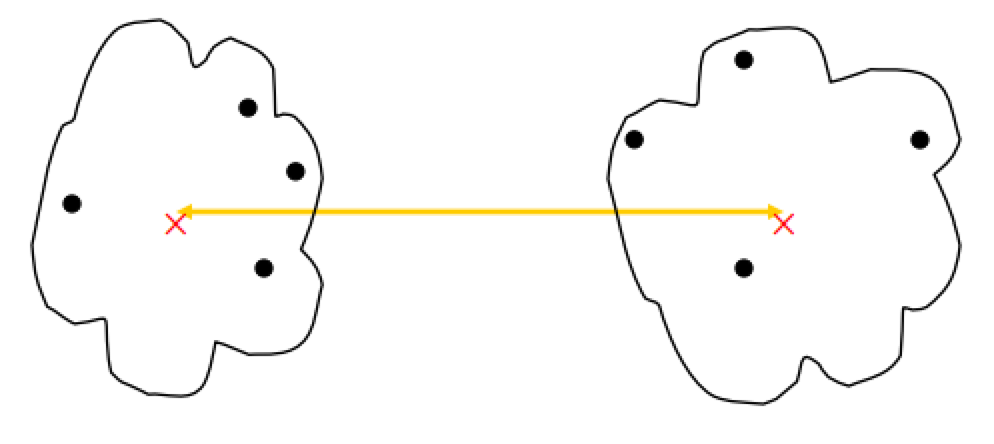
Group average

- The distance between two clusters is represented by the average distance of all pairs of data objects belonging to different clusters
- Determined by all pairs of points in the two clusters



Centroid Distance

- The distance between two clusters is represented by the distance between the centers of the clusters
- – Determined by cluster centroids



Ward's Method

- Similarity of two clusters is based on the increase in squared error when two clusters are merged
- Similar to group average if distance between points is distance squared
- Less susceptible to noise and outliers