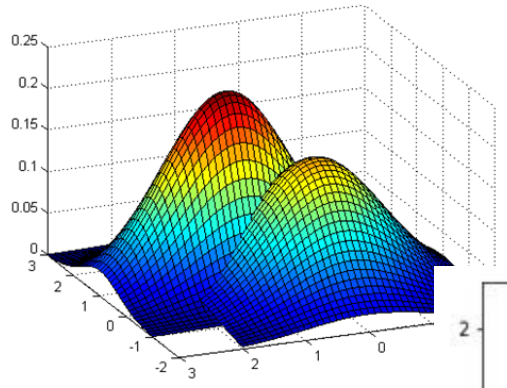
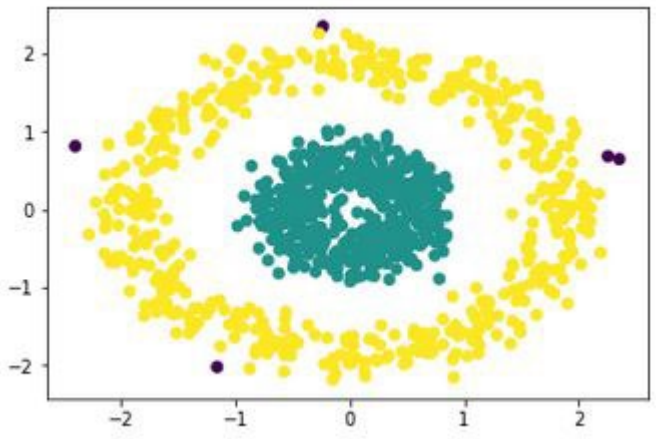
Unsupervised ML part II : Gaussian Mixture Model & DBScan





อ.ดร.ปัญญนัท อ้นพงษ์

ภาควิชาคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยศิลปากร

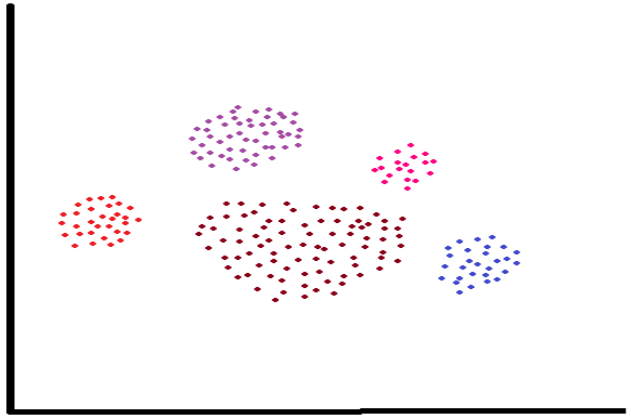
aonpong_p@su.ac.th

Outline

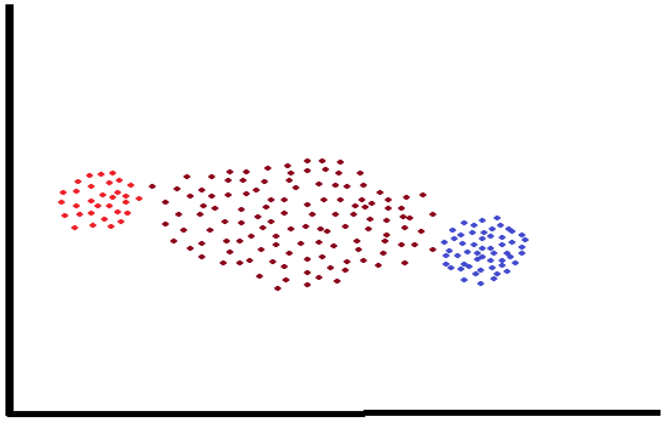
- **ข้อจำกัดของ k-Mean**
- Gaussian Mixture
  - Gaussian Distribution
  - Maximum Likelihood Estimation อย่างง่าย
  - Gaussian Mixture Model
    - แนวคิดของ Gaussian Mixture Model
    - การนำไปใช้
- DBScan

ข้อจำกัดของ k-Means

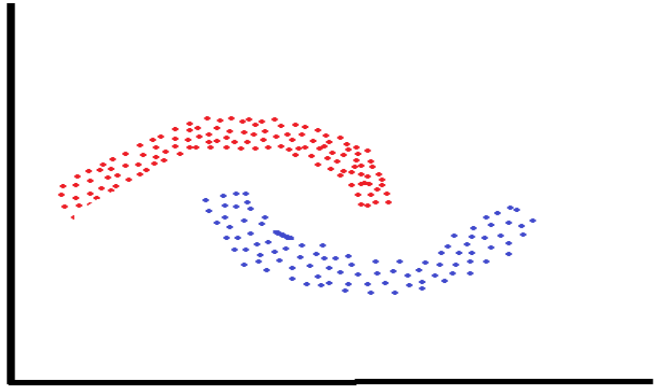
1. ปัญหาค่า k ที่เหมาะสม
2. ปัญหาการเลือกจุด centroid เริ่มต้นที่เหมาะสม
3. ข้อมูลที่มีไม่เหมาะสมกับกระบวนการวิธี k-Means



Cluster with different sizes

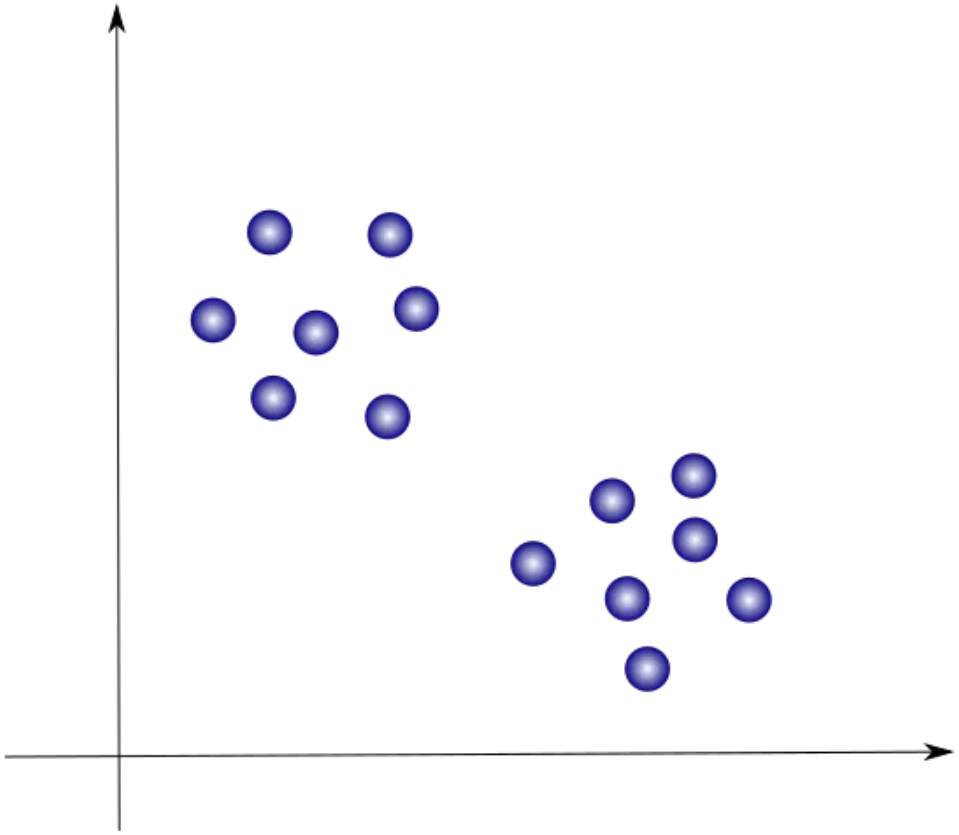


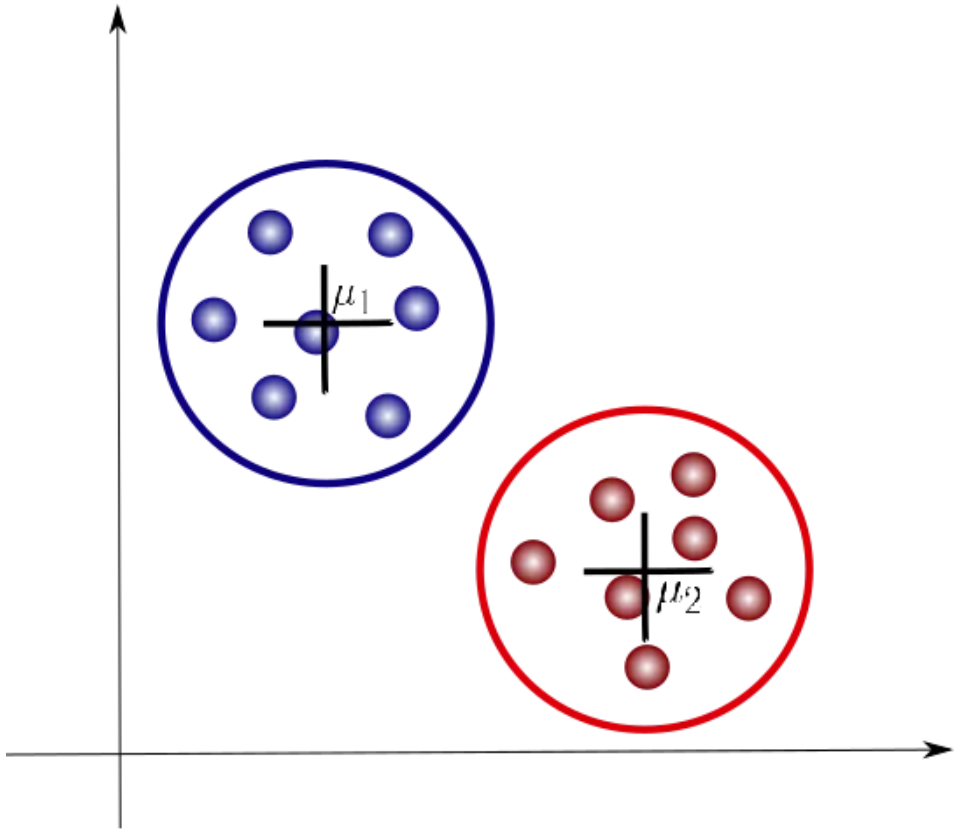
Cluster with different densities



Non-convex shaped clusters

ข้อจำกัดของ k-Means



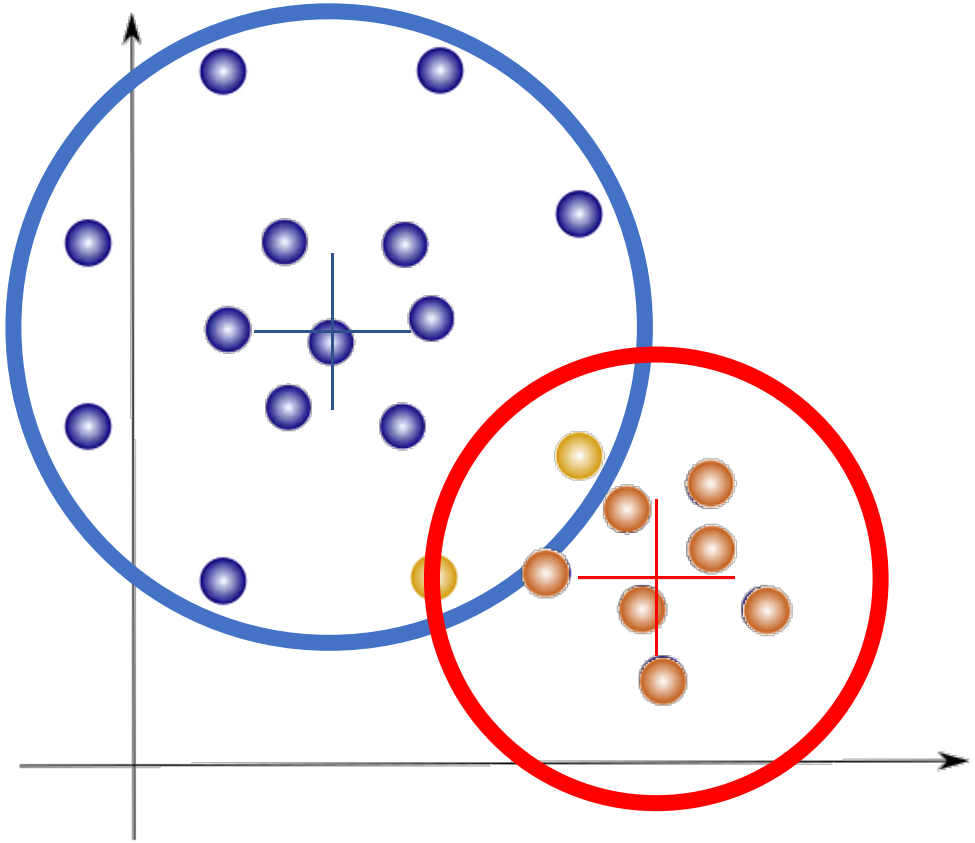


ข้อจำกัดของ k-Means

- ขั้นตอนวิธีการจัดกลุ่มที่เรารู้จักแล้วคือ K-mean ซึ่งจะทำตามขั้นตอนแบบวนซ้ำเพื่ออัปเดตพารามิเตอร์ต่างๆ รวมถึง Centroid ของแต่ละกลุ่ม
- ในความคุ้นเคยของเราตอนนี้ μ คือ Centriod ของคลัสเตอร์แต่ละจุด

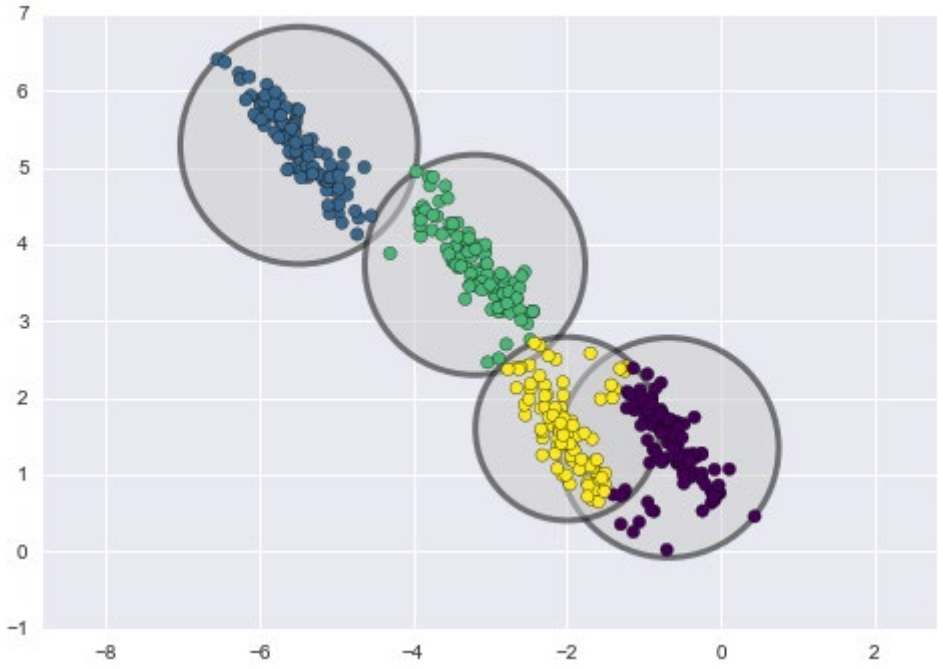
- ขั้นตอนวิธีการจัดกลุ่มที่เรารู้จักแล้วคือ K-mean ซึ่งจะทำตามขั้นตอนแบบวนซ้ำเพื่ออัปเดตพารามิเตอร์ต่างๆ รวมถึง Centroid ของแต่ละกลุ่ม
- ในความคุ้นเคยของเราตอนนี้ μ คือ Centriod ของคลัสเตอร์แต่ละจุด

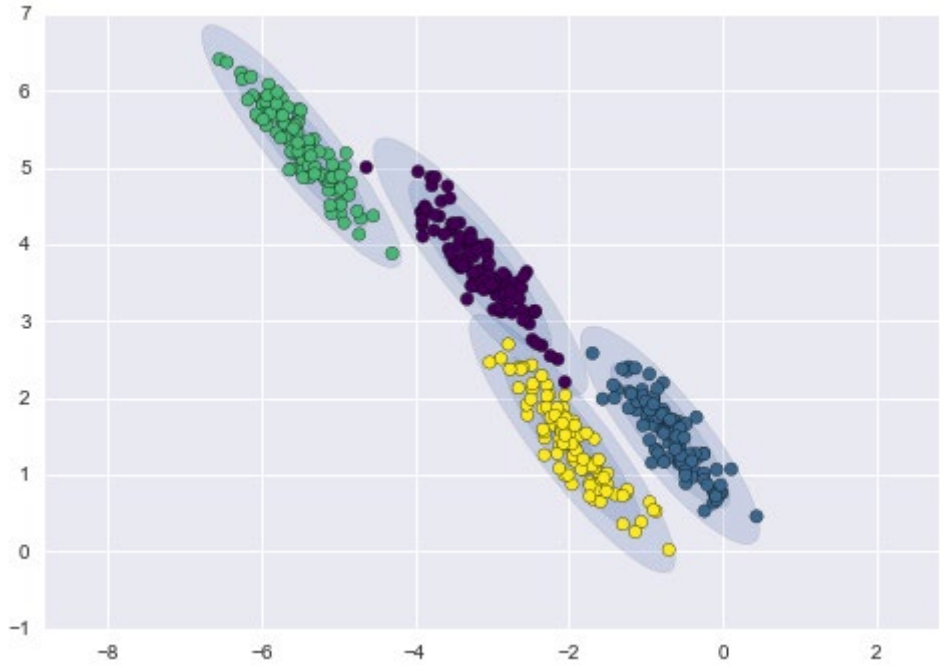
ข้อจำกัดของ k-Means



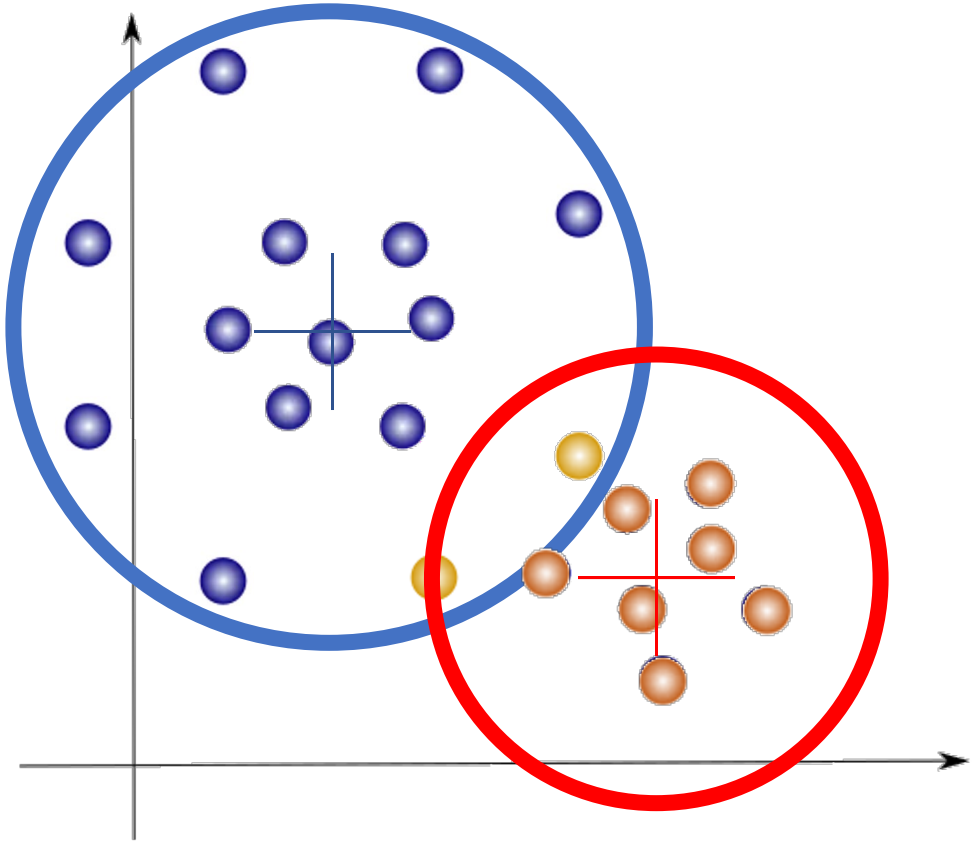
- ลักษณะสำคัญของ k-Mean คือการจัดกลุ่มแบบ Hard clustering โดยจะยึดค่า Centroid (ค่าเฉลี่ย) เพียง 1 ค่าต่อ 1 คลัสเตอร์เท่านั้น
- ข้อจำกัดของแนวทางนี้คือไม่มีการวัดความไม่แน่นอนหรือความน่าจะเป็นที่จะบอกเราว่าจุดข้อมูลเชื่อมโยงกับคลัสเตอร์มีความเฉพาะเจาะจงมากน้อยเพียงใด (วัดเพียง distance ระหว่างข้อมูลแต่ละตัวกับ mean เท่านั้น ถ้าข้อมูลกลุ่มหนึ่งกระจายมาก แต่ไปอยู่ใกล้ mean ตัวอื่น ข้อมูลนั้นจะถูกตีความเป็นอีกคลัสเตอร์หนึ่ง)

ข้อจำกัดของ k-Means



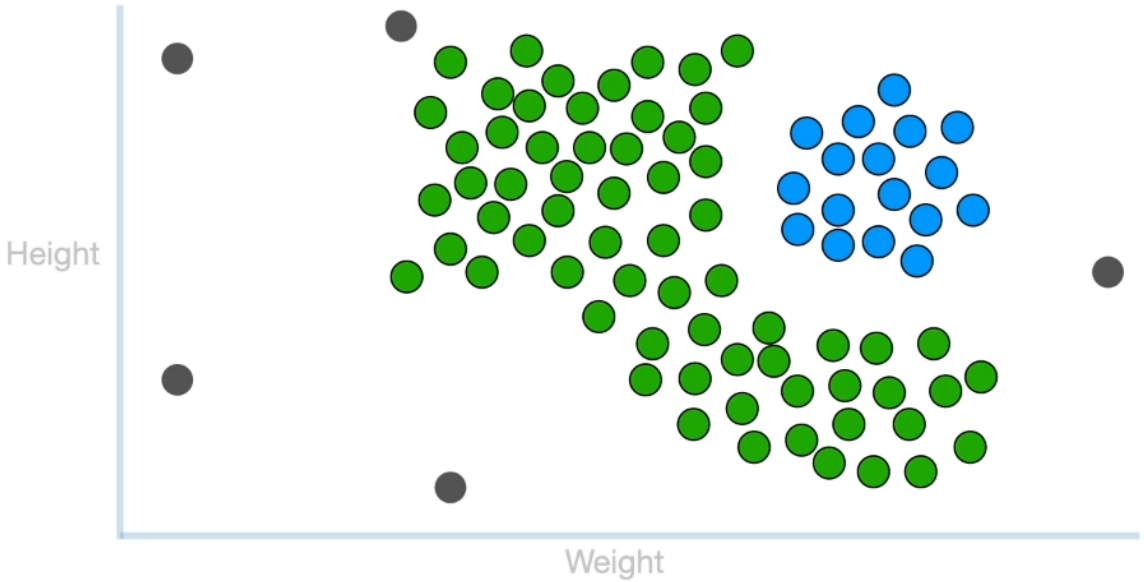


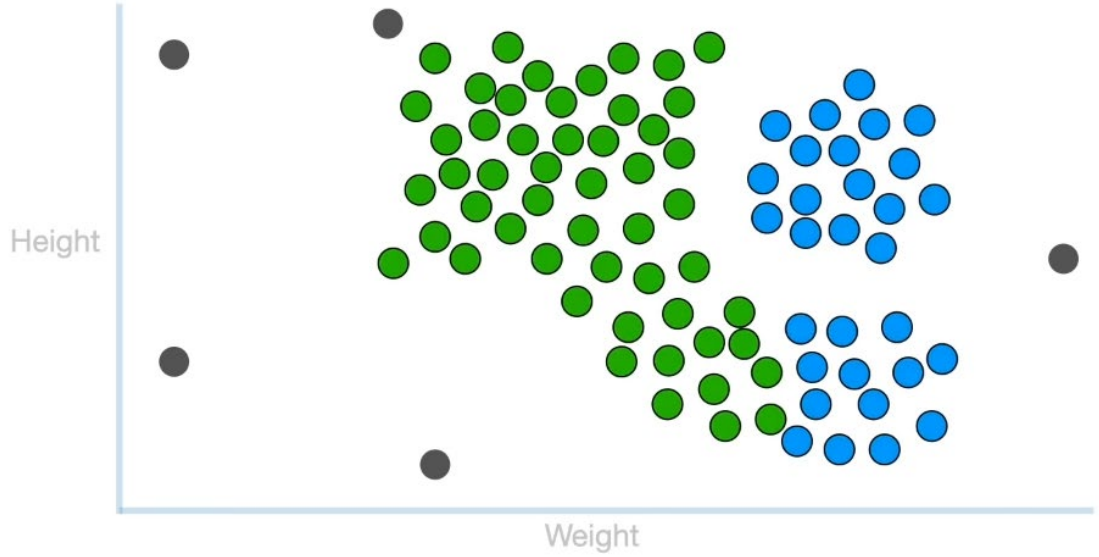
ข้อจำกัดของ k-Means



- ดังนั้นข้อเสียเปรียบของ k-Mean ในกรณีนี้คือ ไม่มีการวัดความไม่แน่นอนหรือความน่าจะเป็นของการเป็นข้อมูลแต่ละกลุ่มรวมอยู่ด้วย กล่าวคือ ไม่มีการระบุว่าข้อมูลที่เราสนใจอยู่มีความเชื่อมโยงกับคลัสเตอร์อื่นมากน้อยเพียงใด
- เราจะนำ Gaussian Mixture Model มาแก้ไขปัญหานี้

ข้อจำกัดของ k-Means





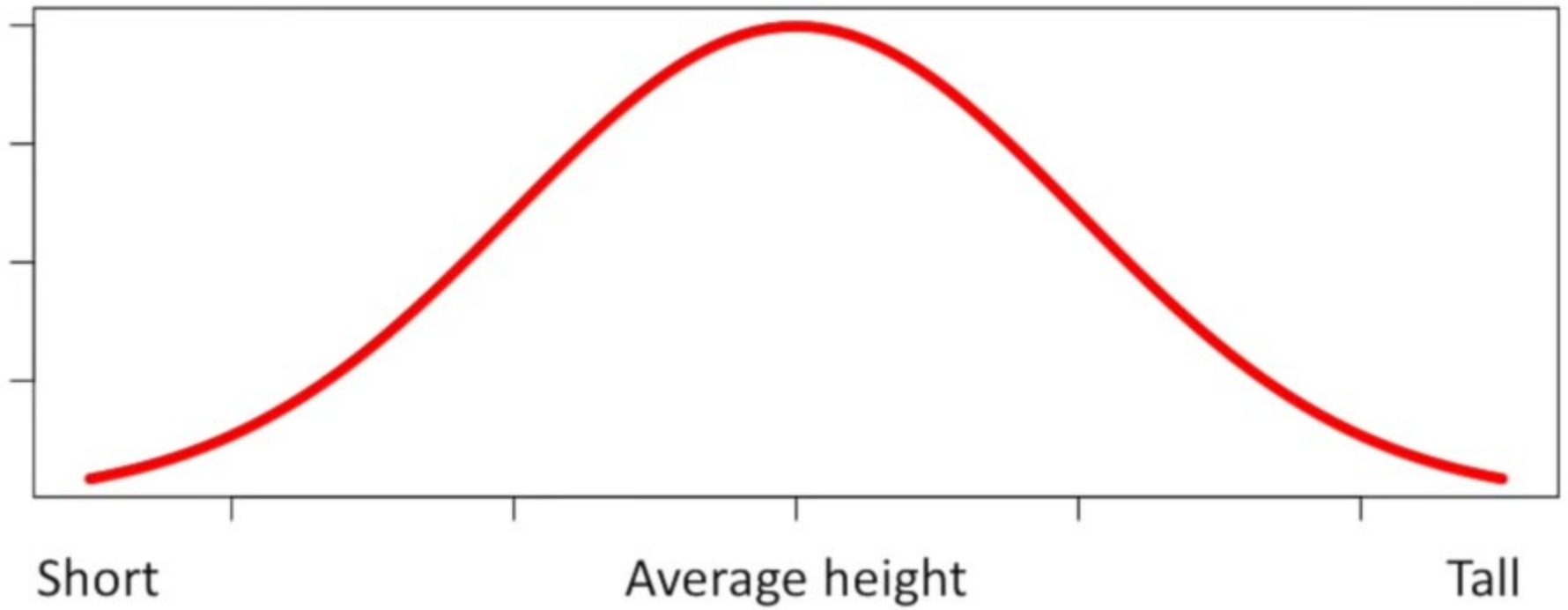
Outline

- ข้อจำกัดของ k-Mean
- **Gaussian Mixture**
  - **Gaussian Distribution**
  - Maximum Likelihood Estimation อย่างง่าย
  - Gaussian Mixture Model
    - แนวคิดของ Gaussian Mixture Model
    - การนำไปใช้
- DBScan

Gaussian Distribution

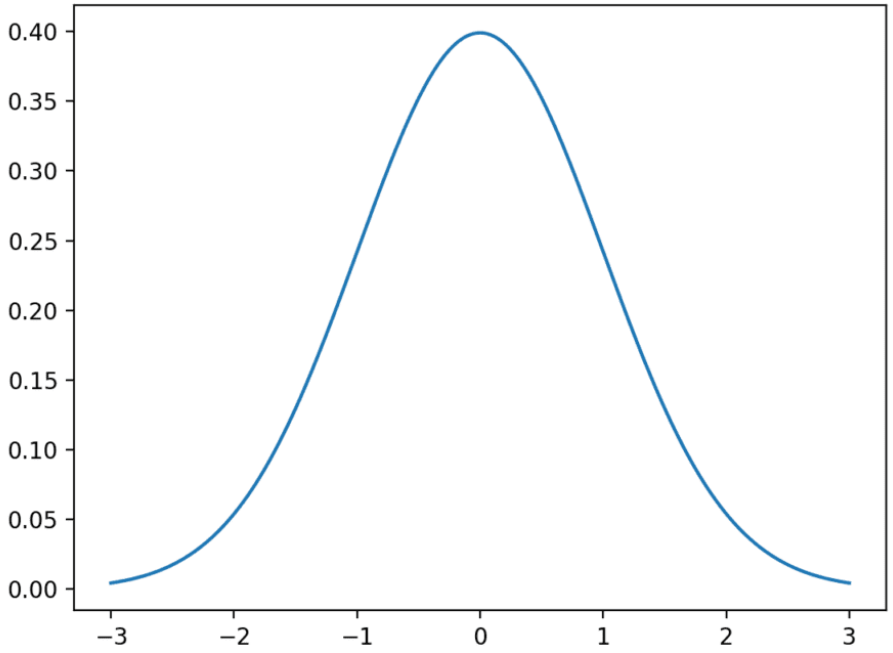
พิจารณากราฟต่อไปนี้

กราฟจำลองความถี่ของความสูงของมนุษย์ ที่มีอายุ 25 ปี



Gaussian Distribution

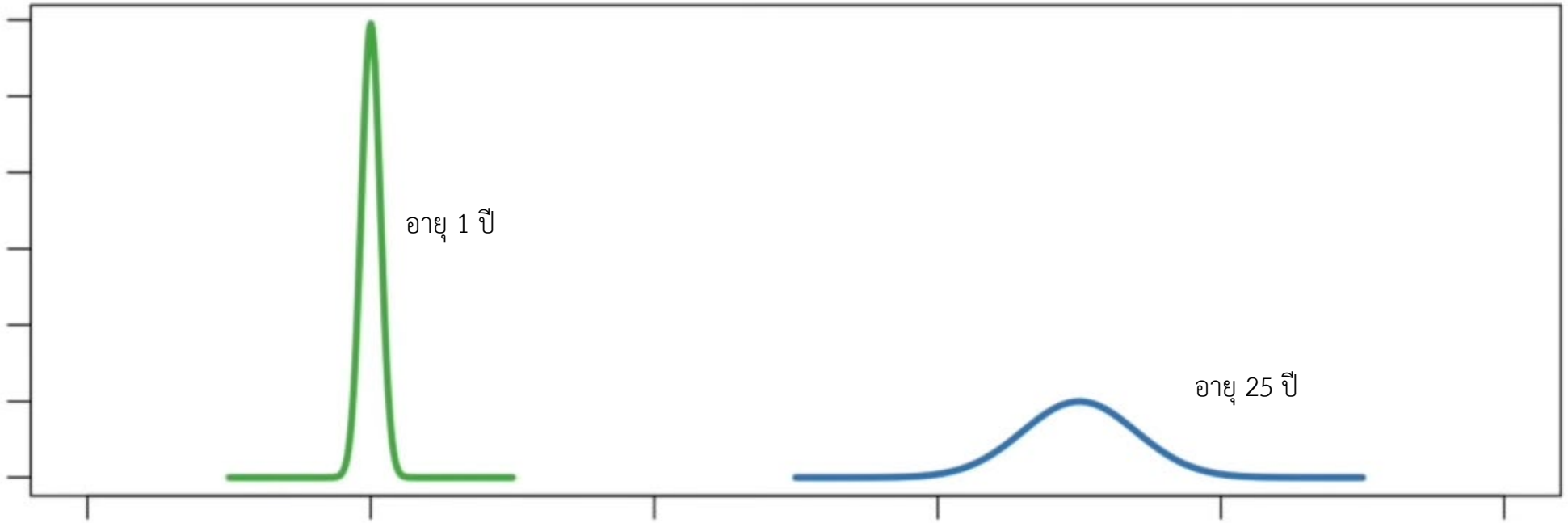
- Gaussian Distribution คือการแจกแจงแบบปกติ (อีกชื่อหนึ่งคือ Normal Distribution)
- ตามทฤษฎีความน่าจะเป็น Gaussian Distribution เป็นการแจกแจงความน่าจะเป็นของค่าตัวแปรสุ่มแบบต่อเนื่อง โดยค่าตัวแปรสุ่มจะมีค่าอยู่ใกล้ ๆ ค่าใดค่าหนึ่ง (ค่าเฉลี่ย)
- มักพบการกระจายตัวแบบนี้ได้ในธรรมชาติ
- กราฟแสดงความถี่ของประชากรที่เป็นไปตามนิยามของ Gaussian Distribution จะเป็นรูประฆังคว่ำ



Gaussian Distribution

กราฟของข้อมูลที่มีการกระจายตัวแบบ Gaussian Distribution บอกอะไรบ้าง?

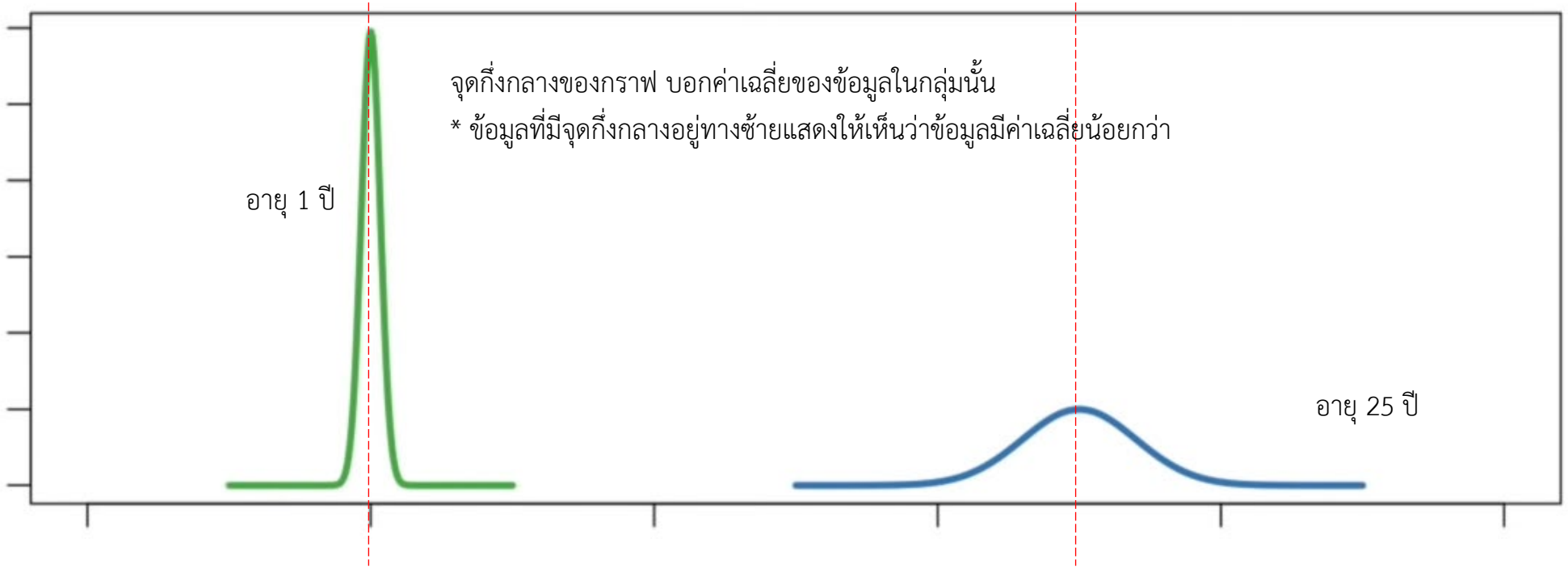
กราฟจำลองความถี่ของความสูงของมนุษย์ ที่มีอายุ 1 ปี และ 25 ปี



Gaussian Distribution

กราฟของข้อมูลที่มีการกระจายตัวแบบ Gaussian Distribution บอกอะไรบ้าง?

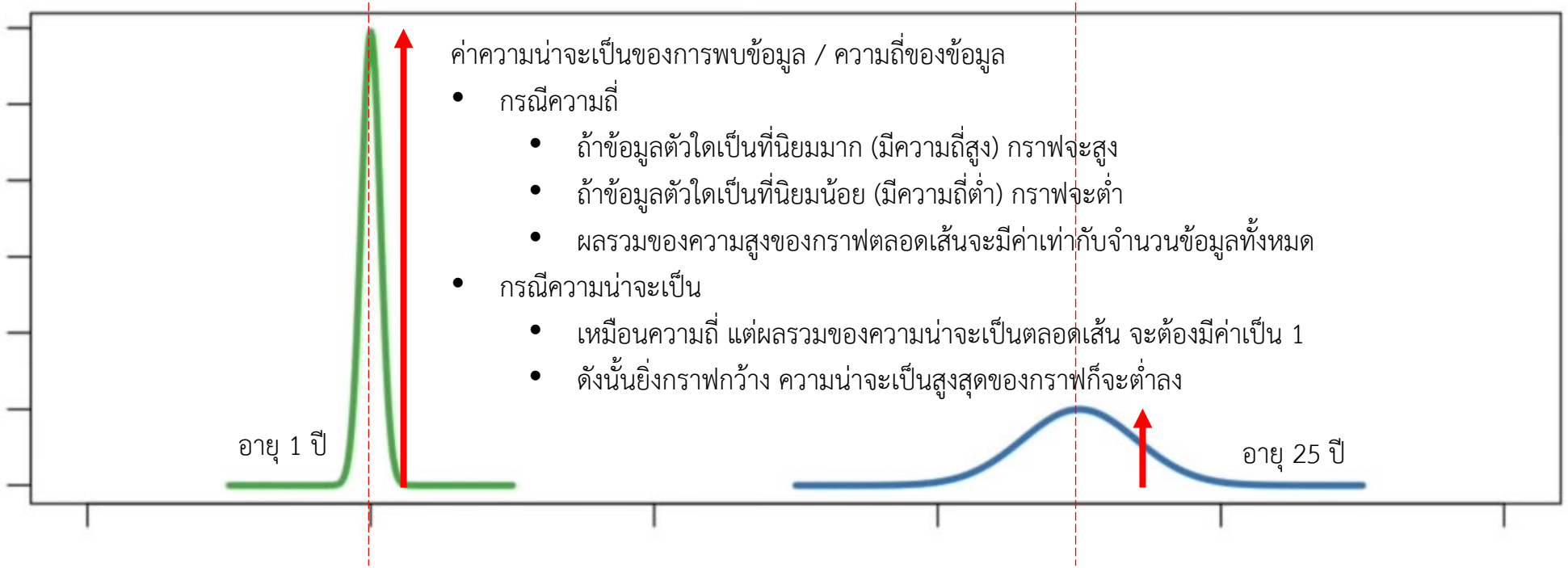
กราฟจำลองความถี่ของความสูงของมนุษย์ ที่มีอายุ 1 ปี และ 25 ปี



Gaussian Distribution

กราฟของข้อมูลที่มีการกระจายตัวแบบ Gaussian Distribution บอกอะไรบ้าง?

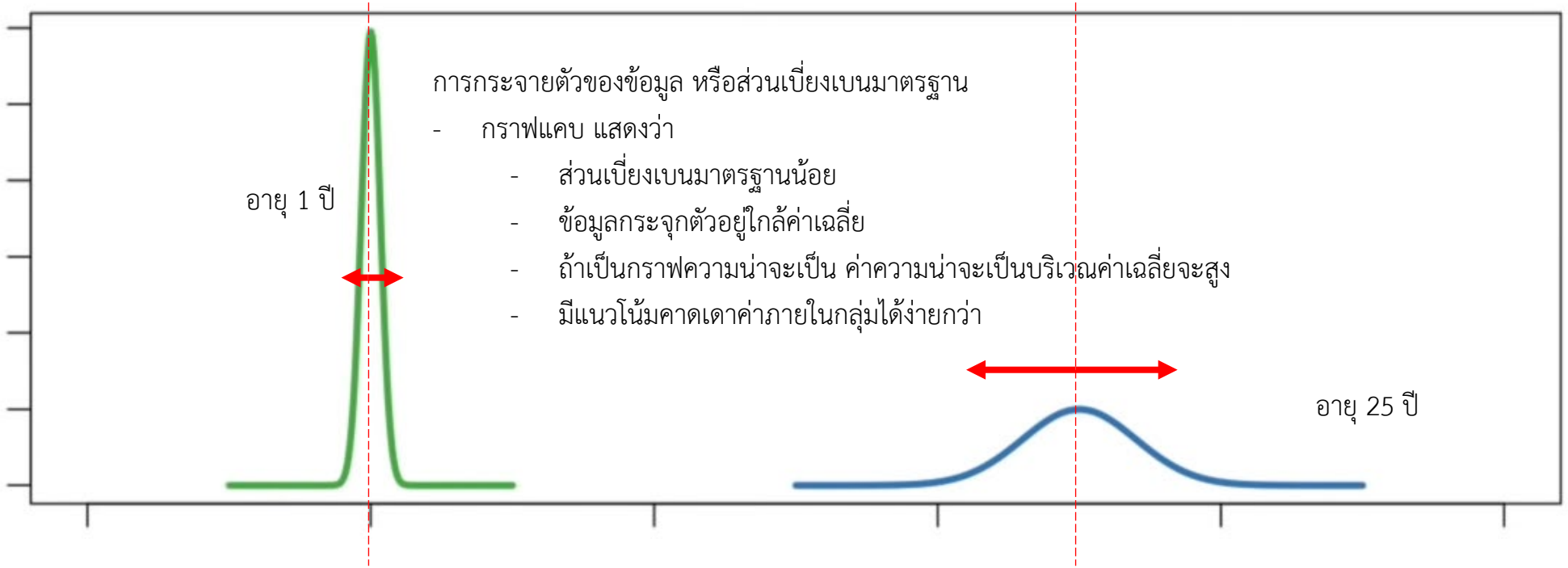
กราฟจำลองความถี่ของความสูงของมนุษย์ ที่มีอายุ 1 ปี และ 25 ปี



Gaussian Distribution

กราฟของข้อมูลที่มีการกระจายตัวแบบ Gaussian Distribution บอกอะไรบ้าง?

กราฟจำลองความถี่ของความสูงของมนุษย์ ที่มีอายุ 1 ปี และ 25 ปี



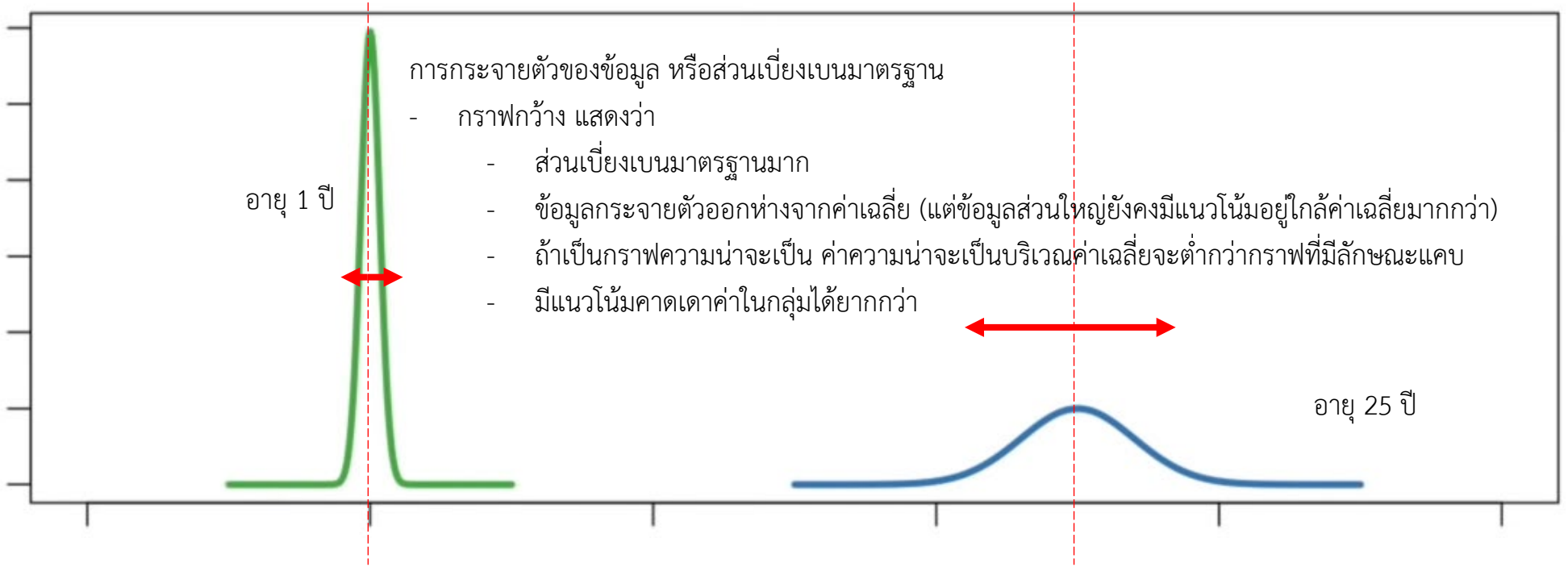
การกระจายตัวของข้อมูล หรือส่วนเบี่ยงเบนมาตรฐาน

- กราฟแคบ แสดงว่า
  - ส่วนเบี่ยงเบนมาตรฐานน้อย
  - ข้อมูลกระจุกตัวอยู่ใกล้ค่าเฉลี่ย
  - ถ้าเป็นกราฟความน่าจะเป็น ค่าความน่าจะเป็นบริเวณค่าเฉลี่ยจะสูง
  - มีแนวโน้มคาดเดาค่าภายในกลุ่มได้ง่ายกว่า

Gaussian Distribution

กราฟของข้อมูลที่มีการกระจายตัวแบบ Gaussian Distribution บอกอะไรบ้าง?

กราฟจำลองความถี่ของความสูงของมนุษย์ ที่มีอายุ 1 ปี และ 25 ปี



Gaussian Distribution

กราฟของข้อมูลที่มีการกระจายตัวแบบ Gaussian Distribution บอกอะไรบ้าง?

1. ค่าเฉลี่ยของข้อมูล
   - ค่ามาก กราฟจะเลื่อนไปทางขวา
   - ค่าน้อย กราฟจะเลื่อนไปทางซ้าย
2. ค่าความน่าจะเป็น และ/หรือ ความถี่
   - ค่ามาก กราฟสูง
   - ค่าน้อย กราฟเตี้ย
3. ค่าความเบี่ยงเบนมาตรฐาน
   - ค่ามาก กราฟกว้าง
   - ค่าน้อย กราฟแคบ

Gaussian density function

$$\mathcal{N}(\mathbf{x}|\mu,\Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right)$$

Outline

- ข้อจำกัดของ k-Mean
- Gaussian Mixture
  - Gaussian Distribution
  - **Maximum Likelihood Estimation อย่างง่าย**
  - Gaussian Mixture Model
    - แนวคิดของ Gaussian Mixture Model
    - การนำไปใช้
- DBScan

Maximum Likelihood Estimation; MLE

จริงๆ แล้ว MLE มีความซับซ้อนมากกว่าที่เราจะได้เรียนรู้กันมาก แต่ในที่นี้จะกล่าวถึงเฉพาะคอนเซปต์ที่สามารถนำไปต่อยอดกับ Gaussian Mixture ได้เท่านั้น

Step 1) $\ln\left(\frac{1}{\sqrt{2\pi\sigma^2}} \times e^{-(x_1-\mu)^2/2\sigma^2}\right)$

Step 2) $\ln\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) + \ln\left(e^{-(x_1-\mu)^2/2\sigma^2}\right)$

Step 3) $\ln\left[(2\pi\sigma^2)^{-1/2}\right] - \frac{(x_1-\mu)^2}{2\sigma^2}\ln(e)$

Step 4) $-\frac{1}{2}\ln(2\pi\sigma^2) - \frac{(x_1-\mu)^2}{2\sigma^2}$

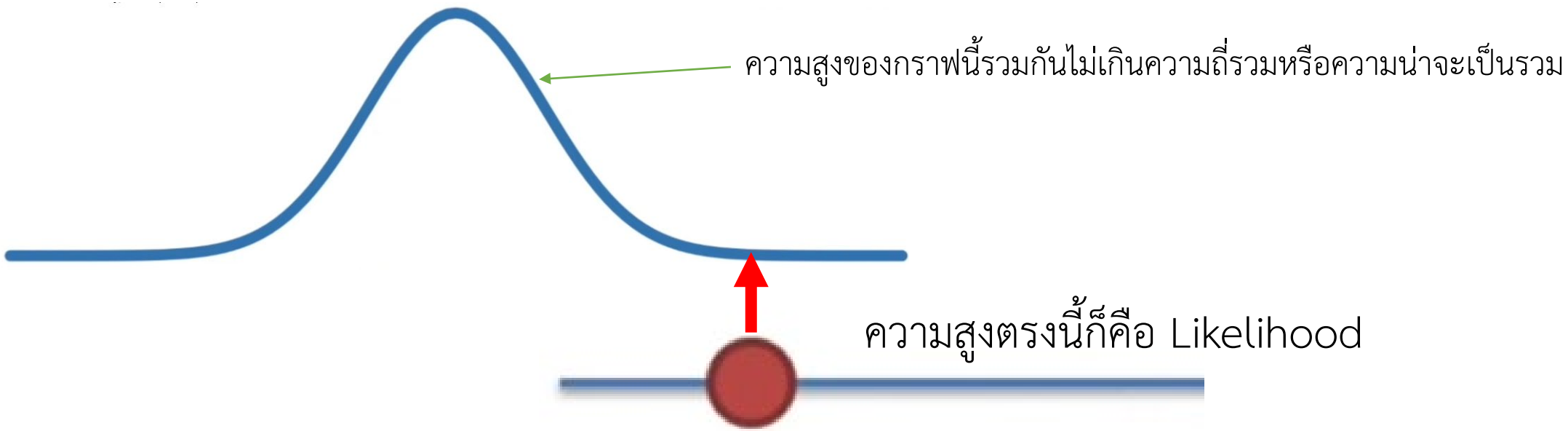
Step 5) $-\frac{1}{2}\ln(2\pi) - \frac{1}{2}\ln(\sigma^2) - \frac{(x_1-\mu)^2}{2\sigma^2}$

Step 6) $-\frac{1}{2}\ln(2\pi) - \frac{2}{2}\ln(\sigma) - \frac{(x_1-\mu)^2}{2\sigma^2}$

Step 7) $-\frac{1}{2}\ln(2\pi) - \ln(\sigma) - \frac{(x_1-\mu)^2}{2\sigma^2}$

Maximum Likelihood Estimation; MLE

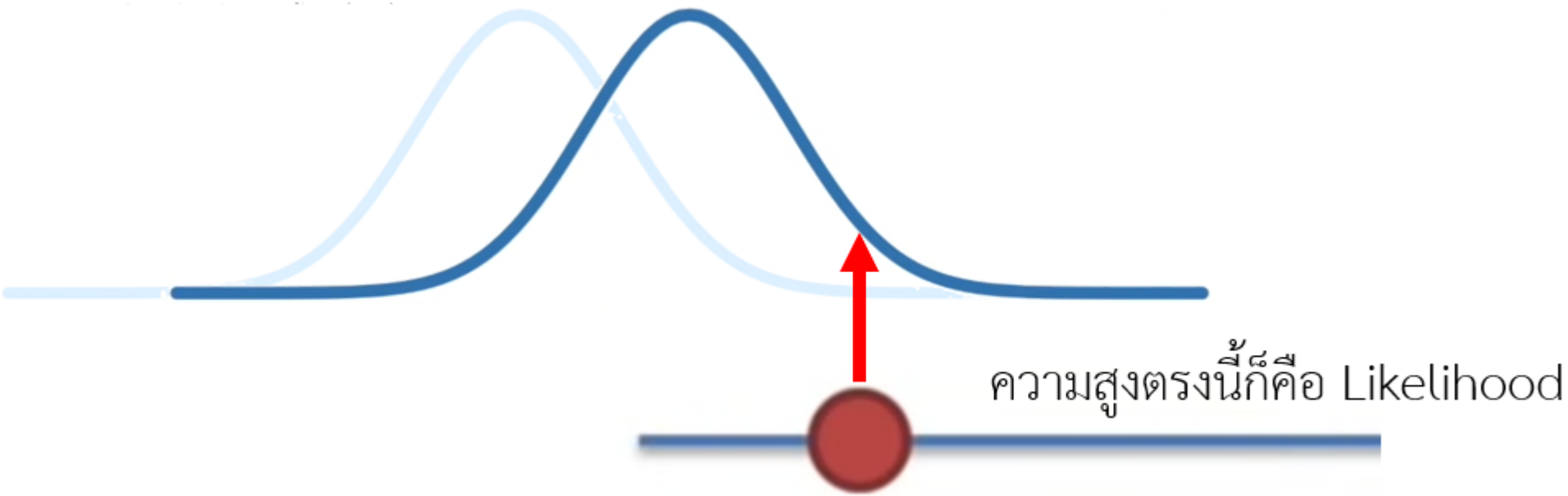
เป้าหมายของ MLE คือการพยายามหาค่าพารามิเตอร์ของฟังก์ชันที่ทำให้ฟิตกับฟังก์ชันที่เราคาดหวัง (ในที่นี้คือ Gaussian) สมมติว่าข้อมูลมีจุดเดียวก่อน ถ้านำกราฟ Gaussian มาทาบ



Maximum Likelihood Estimation; MLE

สมมติว่าข้อมูลมีจุดเดียวก่อน ถ้านำกราฟ Gaussian มาทาบ

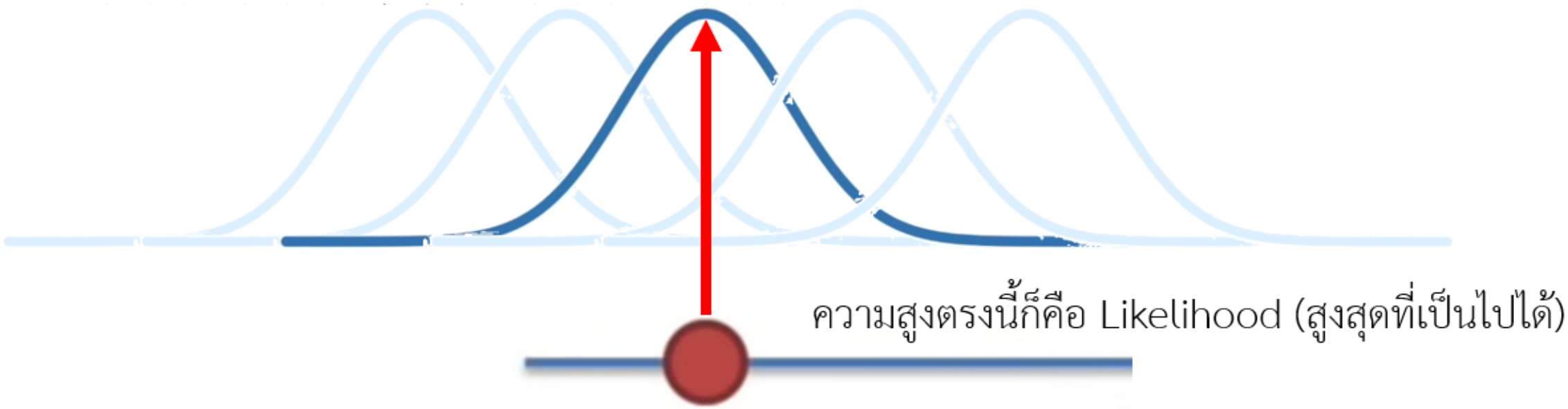
จากนั้นเลื่อนกราฟไปทางขวา จะพบว่า Likelihood เพิ่มขึ้น



Maximum Likelihood Estimation; MLE

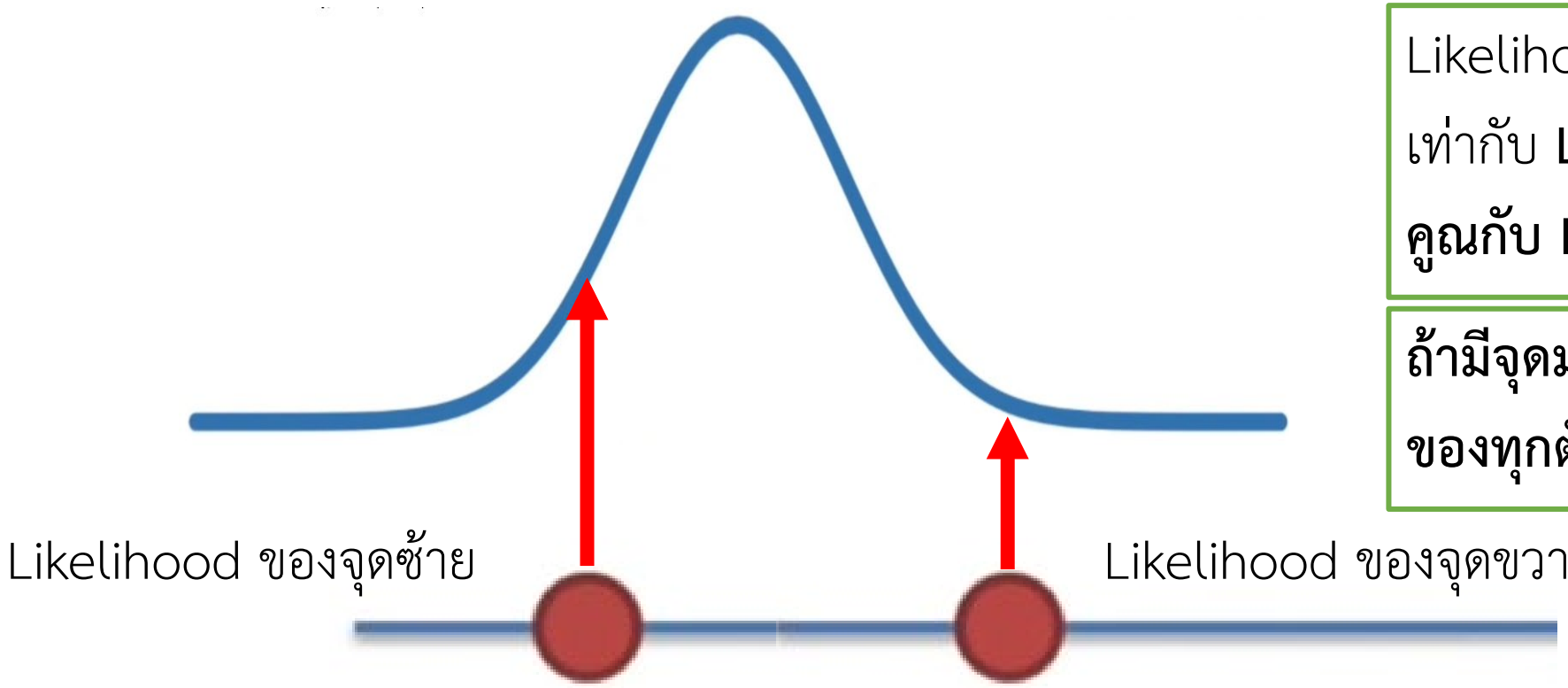
สมมติว่าข้อมูลมีจุดเดียวก่อน ถ้านำกราฟ Gaussian มาทาบ

เมื่อเลื่อนไปเรื่อย ๆ จะมีจุดที่ Likelihood เพิ่มขึ้นจนถึงจุดสูงสุด และกลับลดต่ำลงมา



Maximum Likelihood Estimation; MLE

ที่นี้ถ้ามีข้อมูลมากกว่า 1 จุด แล้วนำกราฟ Gaussian มาทาบ



Likelihood รวมของกราฟจะมีค่าเท่ากับ **Likelihood ของจุดซ้ายคูณกับ Likelihood ของจุดขวา**

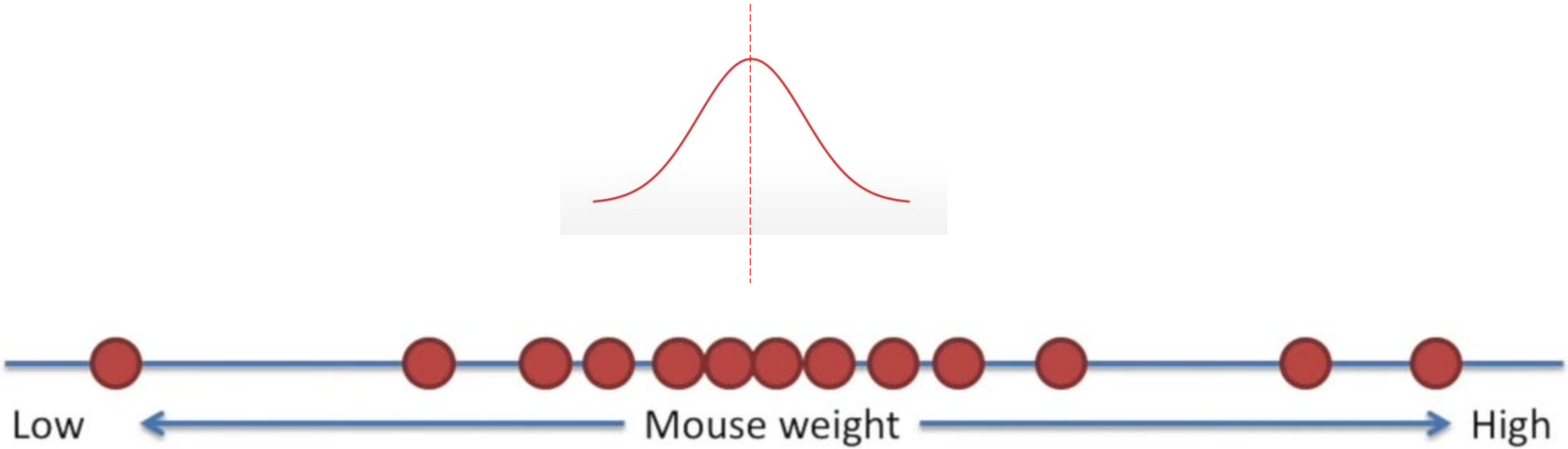
ถ้ามีจุดมากกว่านี้ก็นำ Likelihood ของทุกตัวมาคูณกันทั้งหมด

Maximum Likelihood Estimation; MLE

สมมติเรามีข้อมูลน้ำหนักของหนูทดลองดังนี้

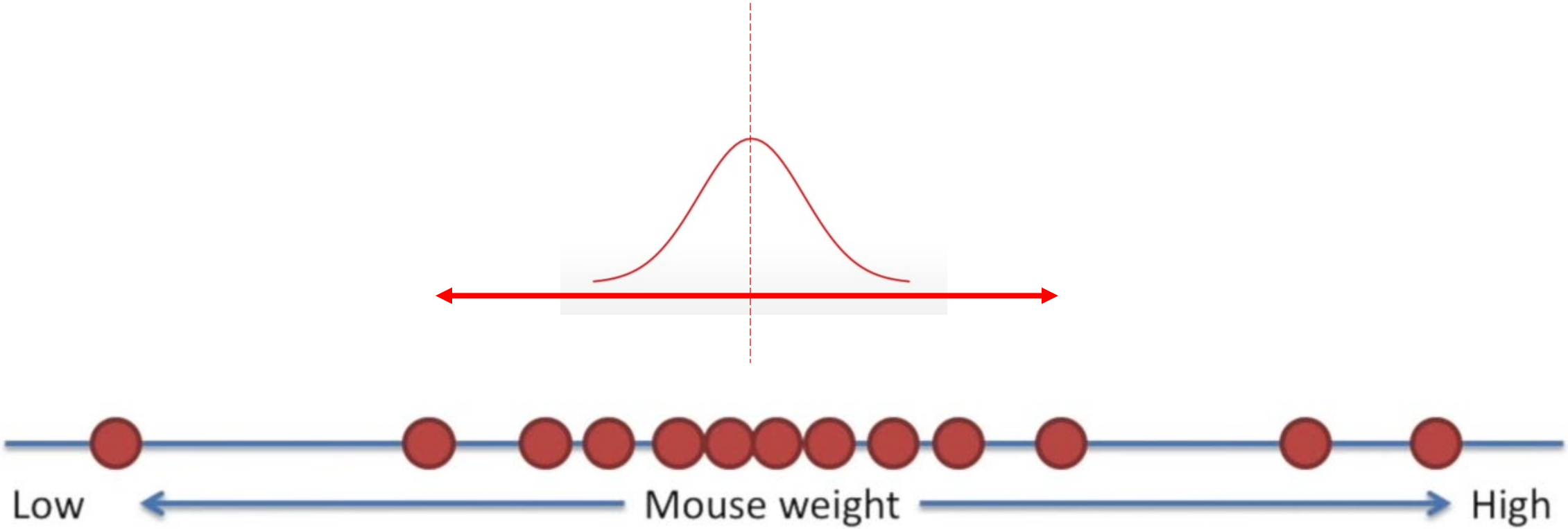
Maximum Likelihood Estimation; MLE

คำตอบก็คือ เอา mean ไปตั้งไว้บนกราฟ ณ จุดที่มีหนูหนาแน่นที่สุด



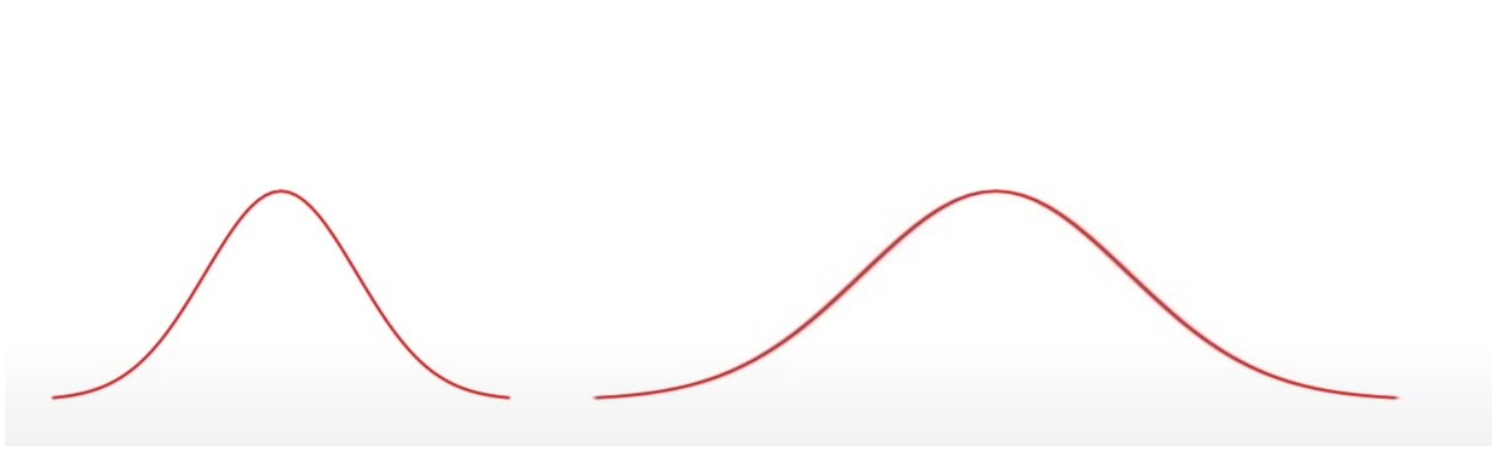
Maximum Likelihood Estimation; MLE

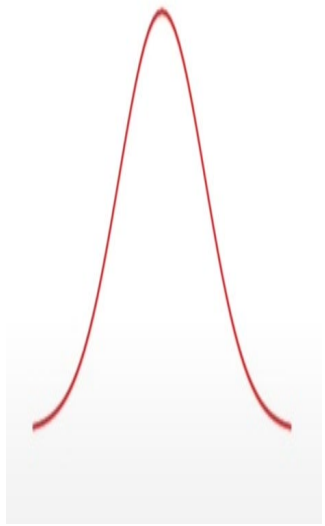
Tip: เรายังคาดหวังด้วยว่ากราฟจะไม่เบ้ไปทางใดทางหนึ่งมากมายนัก (แม้จะไม่สมมาตรโดยสมบูรณ์ก็ไม่เป็นไร)



Maximum Likelihood Estimation; MLE

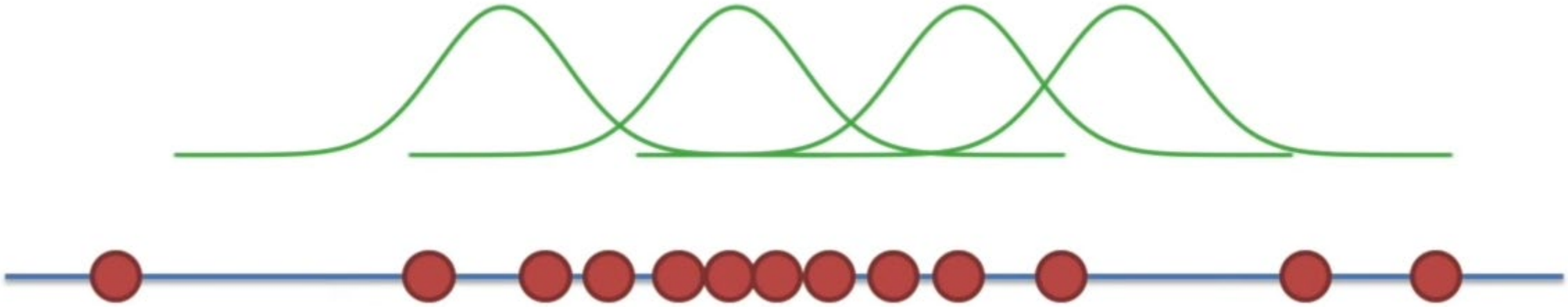
แต่เราจะยังไม่กังวลเกี่ยวกับเรื่องรูปร่างของกราฟ ณ ตอนนี้ และจะไม่พูดถึงสมการมากนัก (เน้นคอนเซปต์)





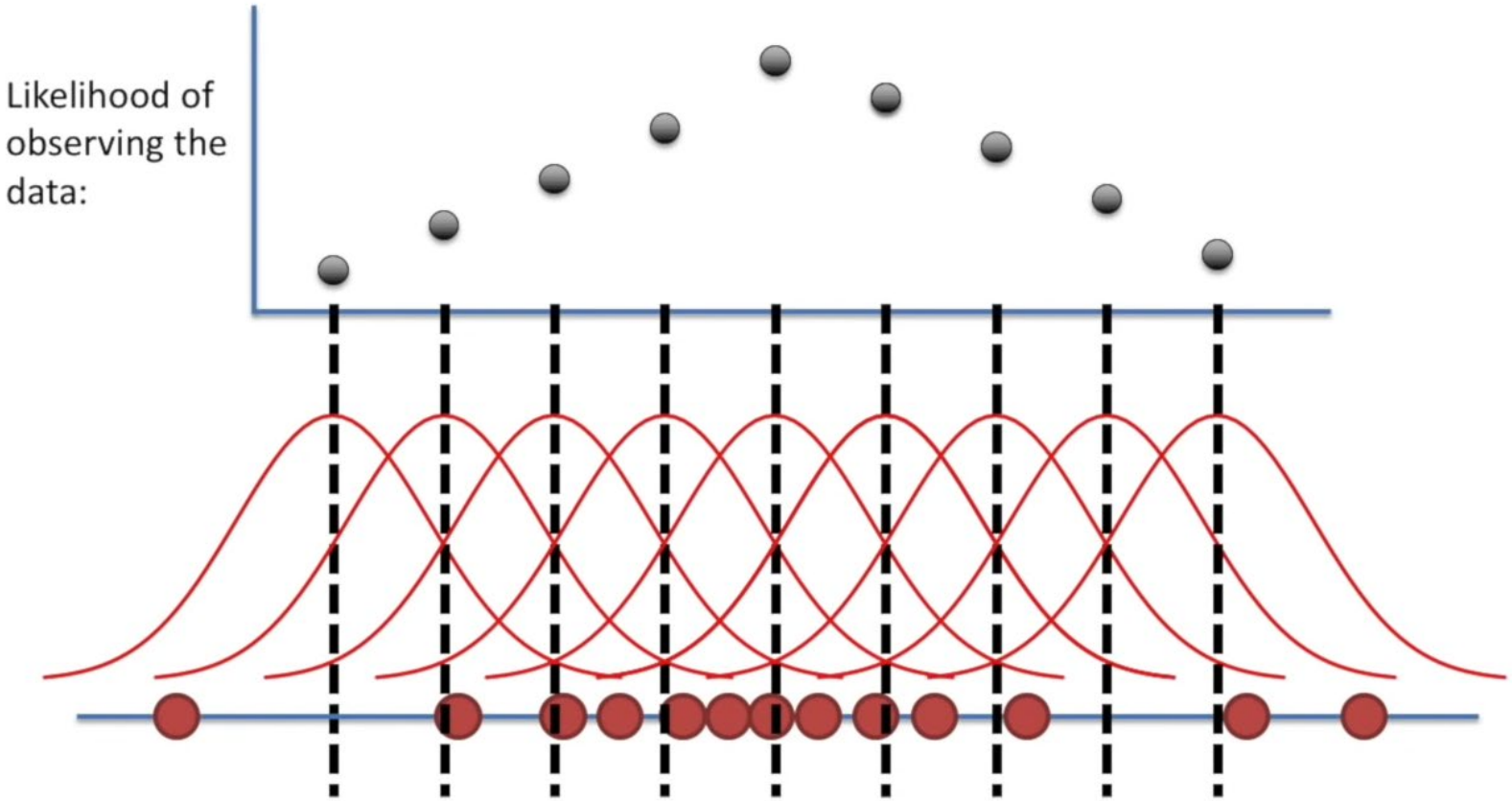
Maximum Likelihood Estimation; MLE

หาจุดกึ่งกลาง หรือ mean (μ) ที่เหมาะกับข้อมูลของเราโดยค่อยๆ เลื่อนจุดกึ่งกลางไปเรื่อย ๆ แล้วแทนค่าในสมการ



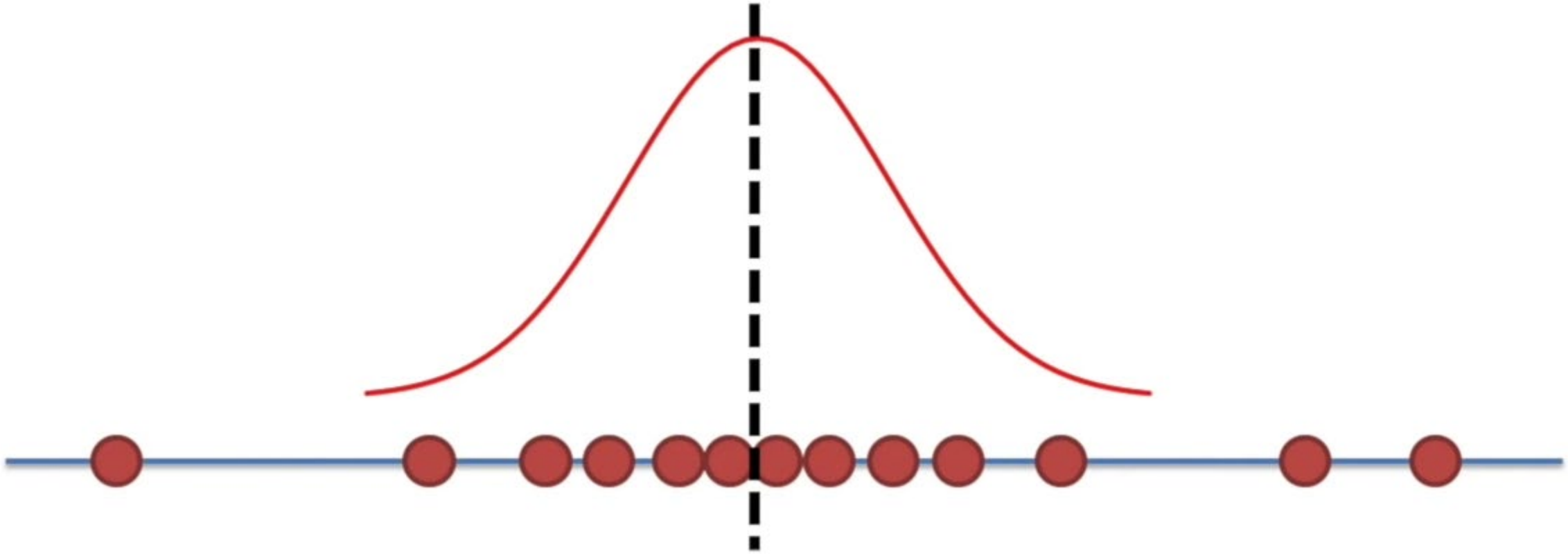
Maximum Likelihood Estimation; MLE

มาหาจุดกึ่งกลาง หรือ mean (μ) ที่เหมาะกับข้อมูลของเรา



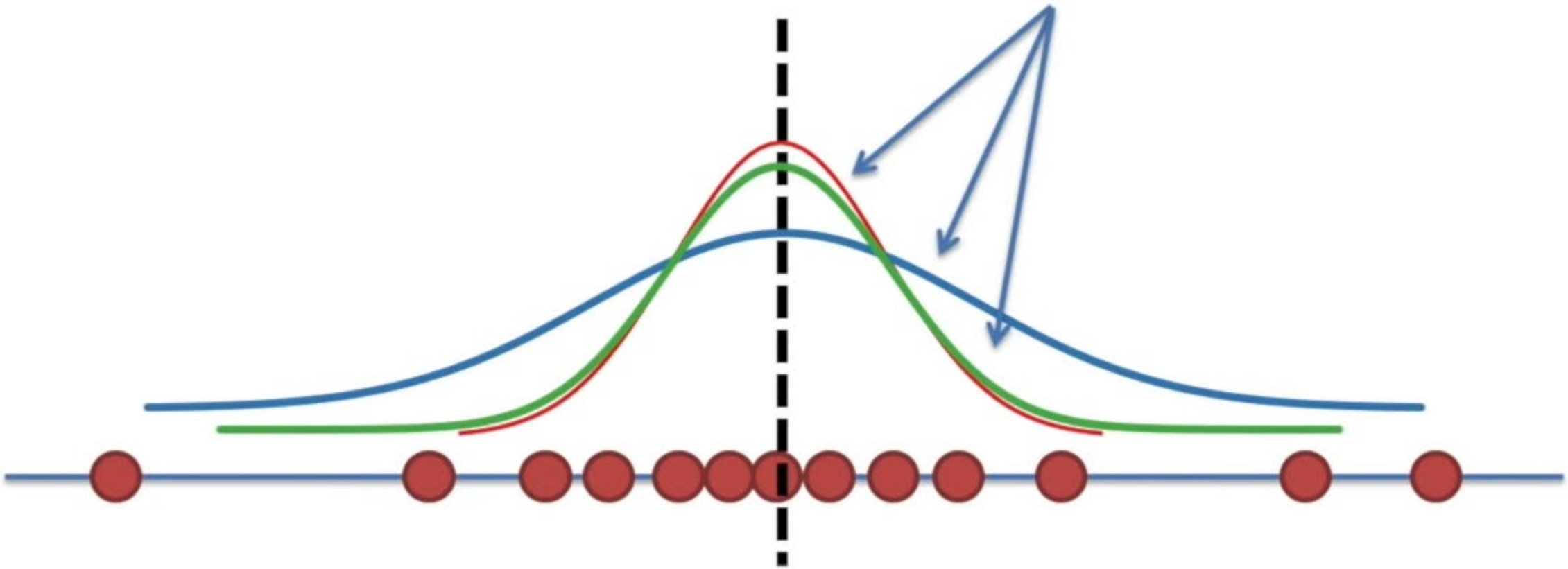
Maximum Likelihood Estimation; MLE

ค่าที่ให้ผลลัพธ์ Likelihood สูงที่สุด คือค่าเฉลี่ย (μ) เหมาะสม



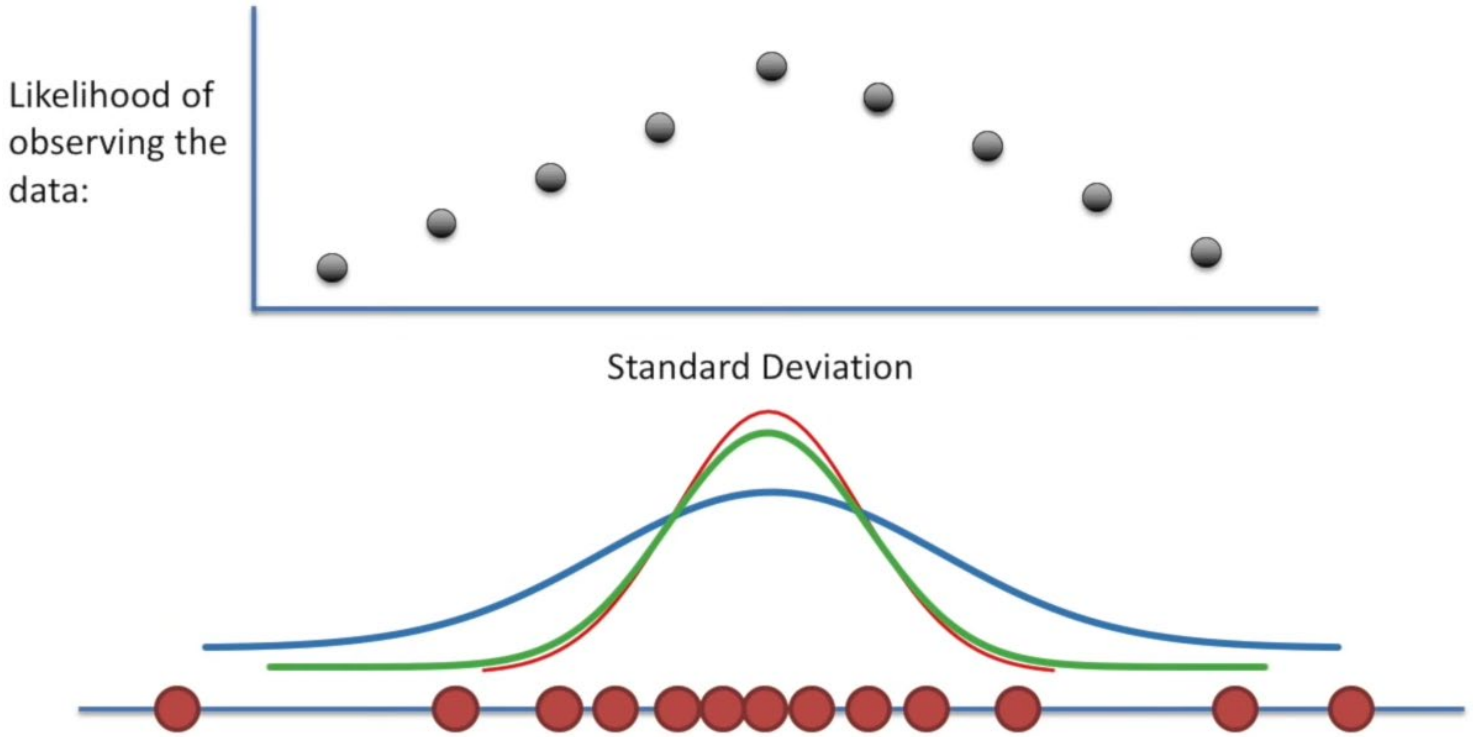
Maximum Likelihood Estimation; MLE

ทีนี้มาดูรูปทรงของกราฟที่เหมาะสมบ้าง เมื่อค่าเบี่ยงเบนมาตรฐานต่างกัน ก็จะทำให้ฐานของกราฟกว้างไม่เท่ากัน



Maximum Likelihood Estimation; MLE

ในการหาค่าที่ fit กับข้อมูลของเรามากที่สุด เราจะหาค่า Likelihood ของค่าเบี่ยงเบนมาตรฐานค่าต่าง ๆ สร้างกราฟ และเลือกค่าเบี่ยงเบนมาตรฐานที่ทำให้ Likelihood ที่สูงที่สุด



Outline

- ข้อจำกัดของ k-Mean
- Gaussian Mixture
  - Gaussian Distribution
  - Maximum Likelihood Estimation อย่างง่าย
  - **Gaussian Mixture Model**
    - **แนวคิดของ Gaussian Mixture Model**
    - การนำไปใช้
- DBScan

แนวคิดของ Gaussian Mixture Model

- Gaussian Mixture Model (GMM) เป็นฟังก์ชันที่ประกอบด้วยข้อมูลที่มีการกระจายแบบ Gaussian จำนวนหลายๆกลุ่ม โดยกำหนดให้แต่ละกลุ่ม เป็น

$$k \in \{1, \ldots, K\}$$

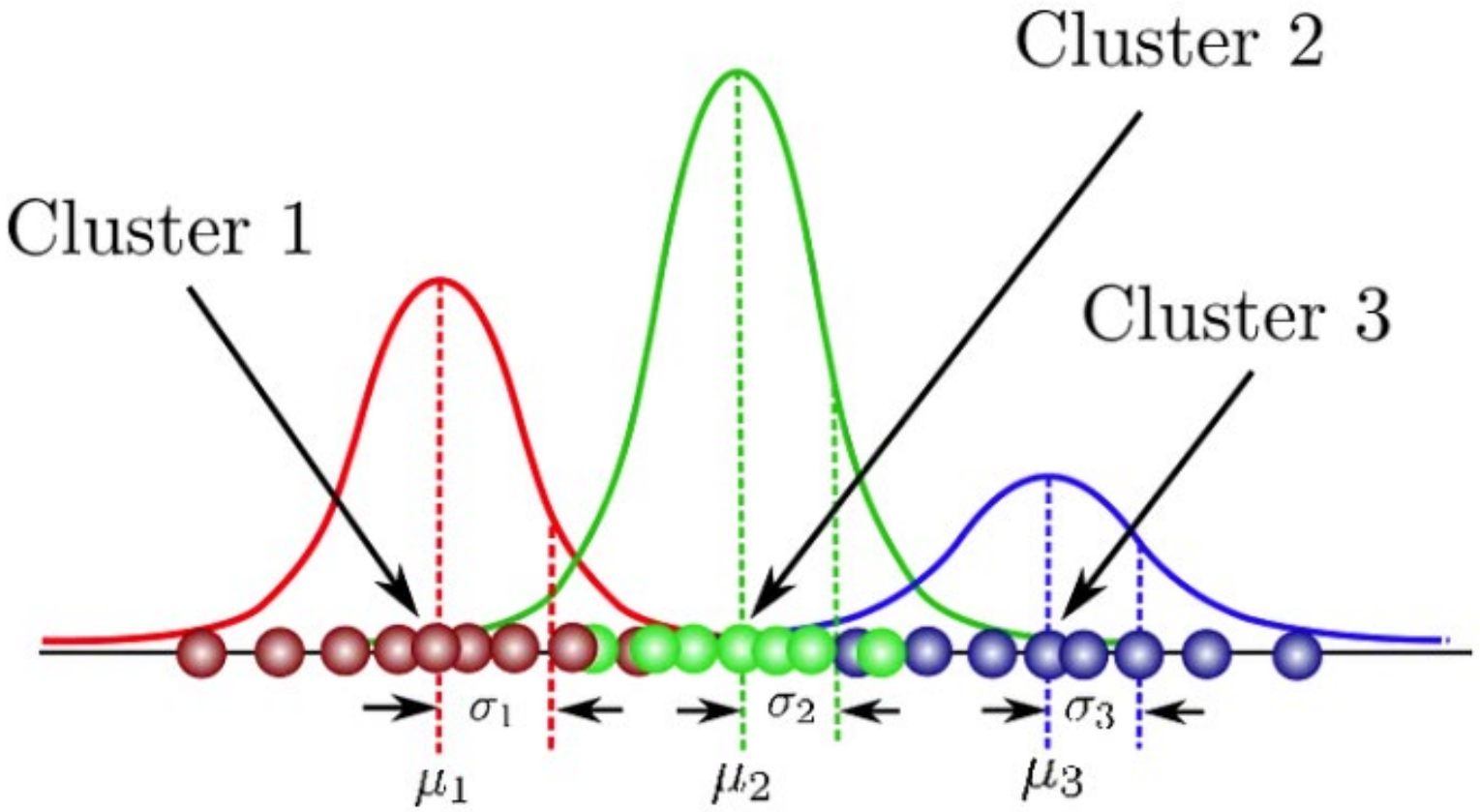
โดยที่ K แทนจำนวนกลุ่มหรือ Cluster ภายในชุดข้อมูล

k แทนฟังก์ชันของคลัสเตอร์แต่ละกลุ่ม โดยแต่ละกลุ่มจะกระจายตัวแบบ Gaussian

สิ่งที่อยู่ในฟังก์ชัน k ประกอบไปด้วยค่าพารามิเตอร์ต่างๆ ดังนี้

- Mean (μ) แทนจุดกึ่งกลางของคลัสเตอร์
- Covariance (Σ) แทนความกว้างของคลัสเตอร์ (ค่านี้อาจมีมากกว่า 1 ค่า; เป็นเมทริกซ์; ในกรณีข้อมูลมีหลาย Attribute แต่ละตัวจะแทนความกว้างแต่ละด้านของทรงรี)
- Mixing Probability; Weight (π) แทนความสูงของฟังก์ชัน Gaussian

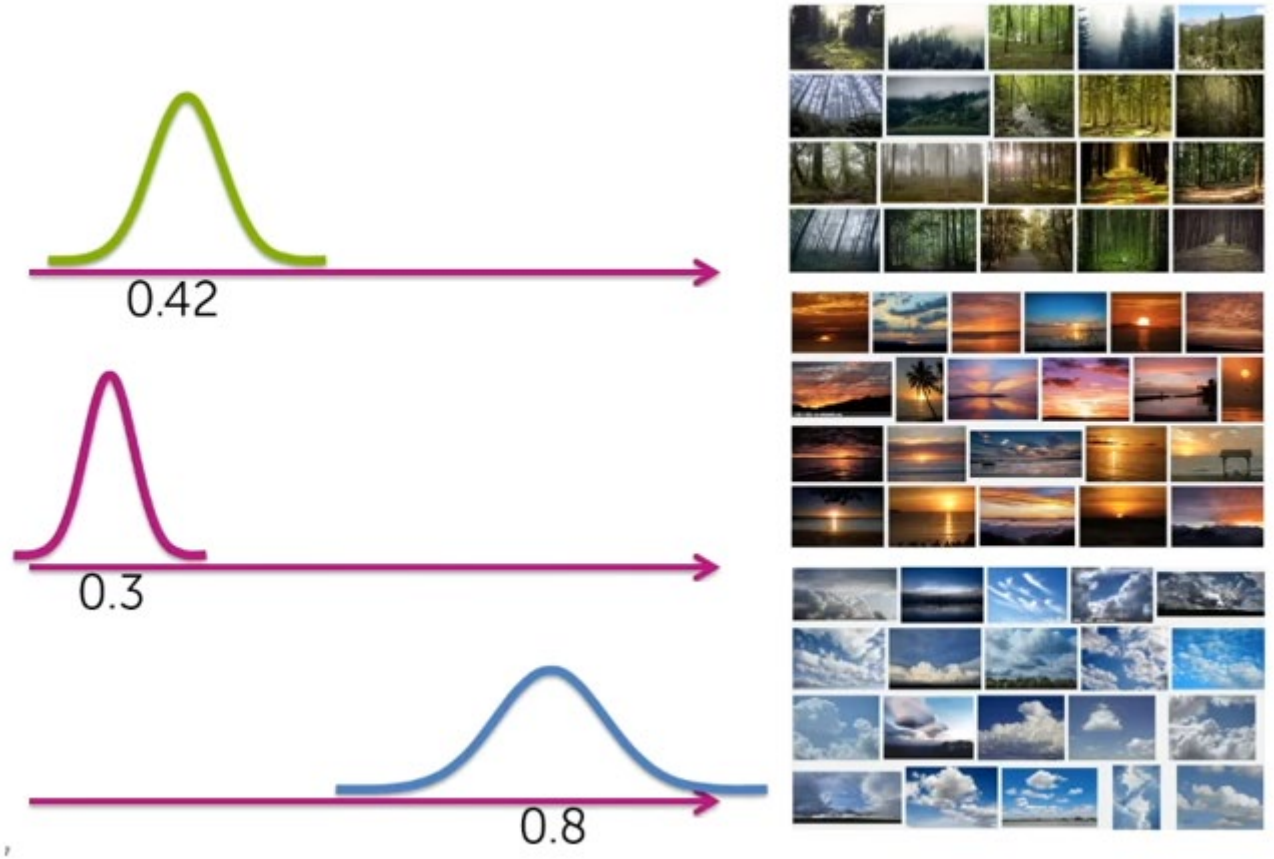
แนวคิดของ Gaussian Mixture Model



$$\sum_{k=1}^{K} \pi_k = 1$$

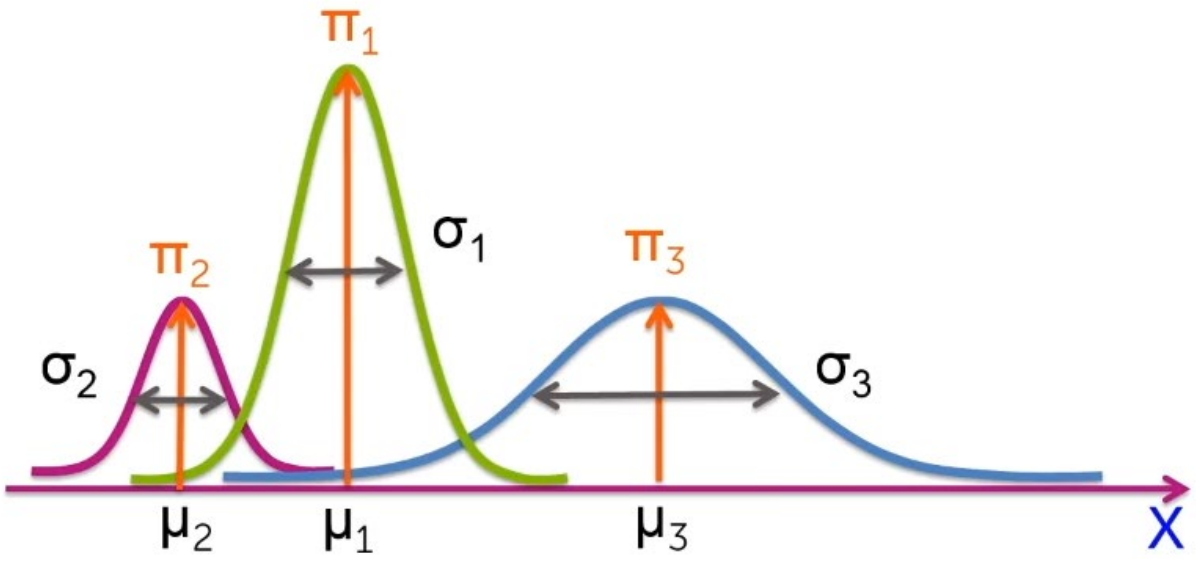
แนวคิดของ Gaussian Mixture Model

- โดยรูปแบบของกราฟดังกล่าว เราต้องการแยกองค์ประกอบให้ได้ประมาณนี้
- แต่อย่าลืมว่าข้อมูลของเราไม่มี Label ดังนั้นการแยกข้อมูลแบบนี้จึงไม่สามารถคำนวณออกมาด้วยสถิติพื้นฐานตรงๆ เช่นค่าเฉลี่ยและส่วนเบี่ยงเบนมาตรฐานได้ (เพราะยังไม่ได้แยกกลุ่มกราฟที่เห็นจะยังเป็นเส้นเดียวผสมกัน)
- ต้องอาศัยเทคนิคพิเศษบางอย่าง



แนวคิดของ Gaussian Mixture Model

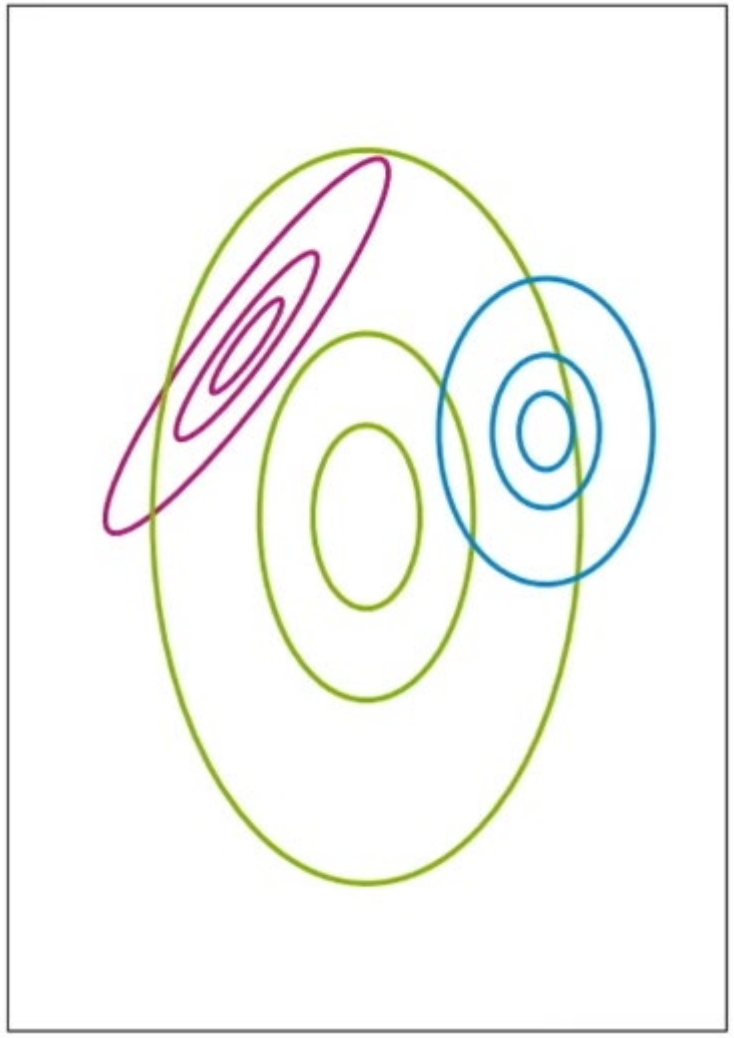
- สิ่งที่ต้องทำก็คือ พยายามหาค่าของตัวแปรเหล่านี้ไปใส่ในฟังก์ชัน ให้ฟิตกับข้อมูลทุกตัวนั่นเอง
- จำนวนคลัสเตอร์ขึ้นอยู่กับการกำหนดของเรา (จำนวน k คลัสเตอร์)
- แต่ละคลัสเตอร์ จะมีตัวแปร 3 ตัว
- เมื่อกำหนดจำนวนคลัสเตอร์แล้ว เรามักใช้ MLE เพื่อประมาณค่าตัวแปรทั้ง 3 ตัวของแต่ละคลาส โดยที่จะพยายามให้ Likelihood ของทุกคลาสมีค่าสูงที่สุด (ไม่ได้ปรับมั่วๆ แต่มีขั้นตอนและกฎเกณฑ์ในการปรับค่าชัดเจน เรียกว่า EM algorithm)



แนวคิดของ Gaussian Mixture Model

Tip:

- บางครั้งเราอาจเห็นภาพกราฟแบบนี้
- กราฟแบบนี้เป็นกราฟของข้อมูลแบบ 3 มิติที่วาดลงบนกราฟ 2 มิตินั่นเอง
- วงรีแต่ละสีแสดง Curve ของแต่ละกราฟ
- วงรีที่ซ้อนกันหลายระดับแสดงระดับความสูง



Outline

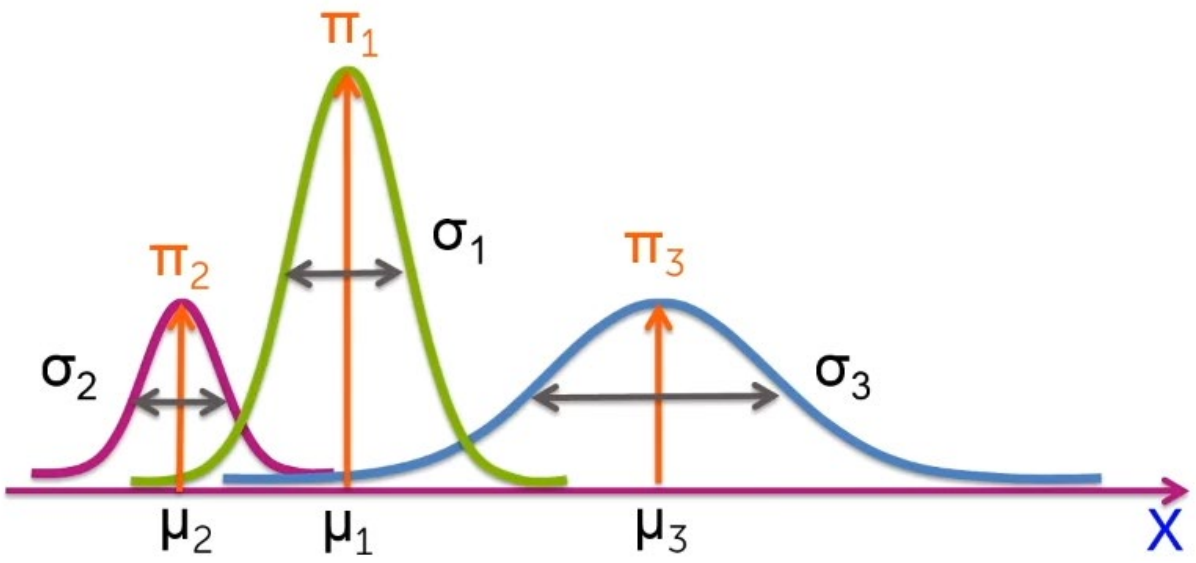
- ข้อจำกัดของ k-Mean
- Gaussian Mixture
  - Gaussian Distribution
  - Maximum Likelihood Estimation อย่างง่าย
  - Gaussian Mixture Model
    - แนวคิดของ Gaussian Mixture Model
    - **การนำไปใช้**
- DBScan

การนำ GMM ไปใช้

- ถ้าเราพิจารณาข้อมูลตัวหนึ่งโดยไม่มอง ความน่าจะเป็นของแต่ละคลาสจะเป็นไปตาม $\boldsymbol{\pi}$ ของแต่ละคลัสเตอร์

$$p(z_i = k) = \pi_k$$

- แต่ถ้าเราเห็นข้อมูลที่กำลังพิจารณา เราสามารถคำนวณความน่าจะเป็นให้แม่นยำยิ่งขึ้นได้ โดยใช้สมการความน่าจะเป็นที่อ้างถึงการกระจายตัวแบบ Gaussian ได้ดังนี้



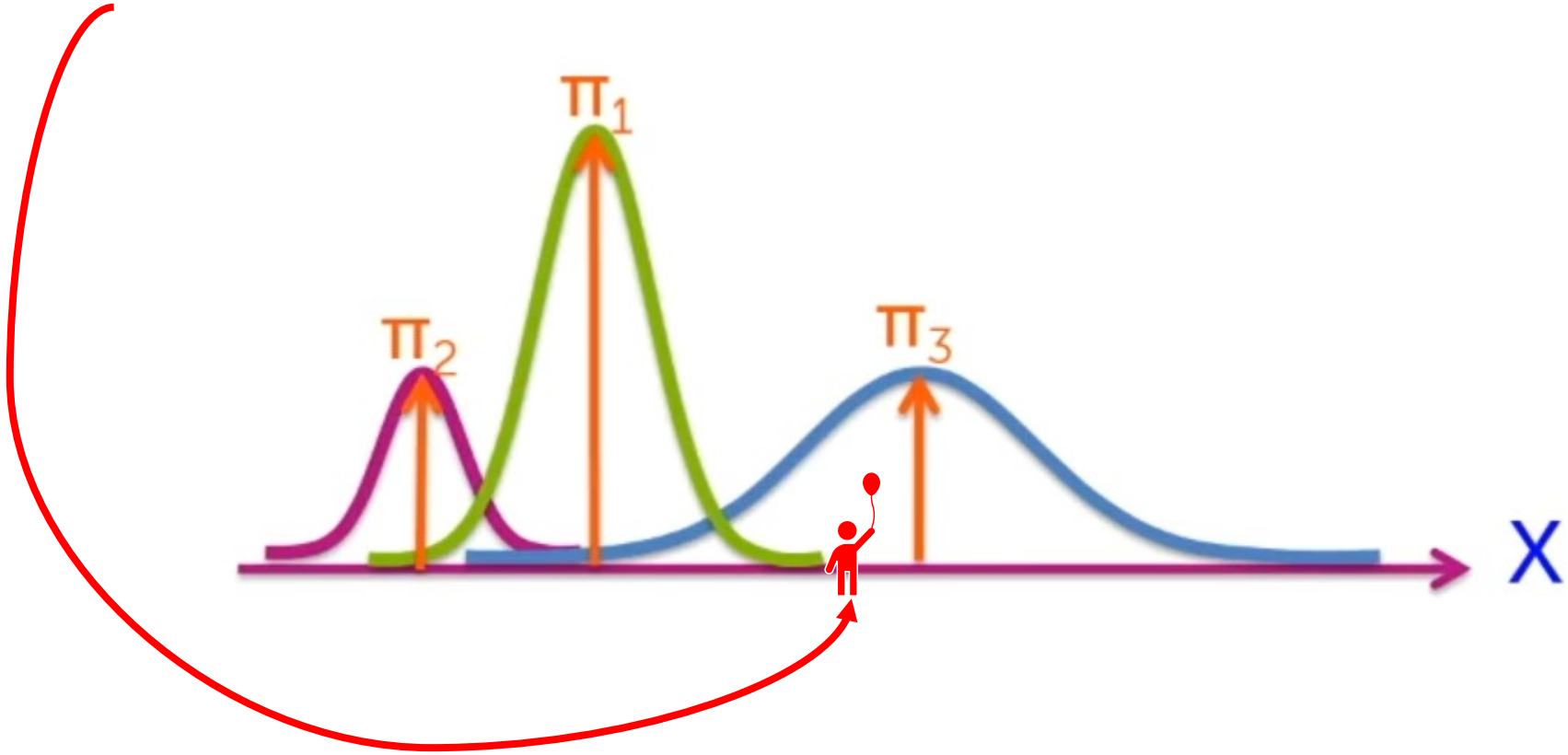
$$p(x_i \mid z_i = k, \mu_k, \Sigma_k) = N(x_i|\mu_k, \Sigma_k)$$

การนำ GMM ไปใช้

- จากสมการดังกล่าว

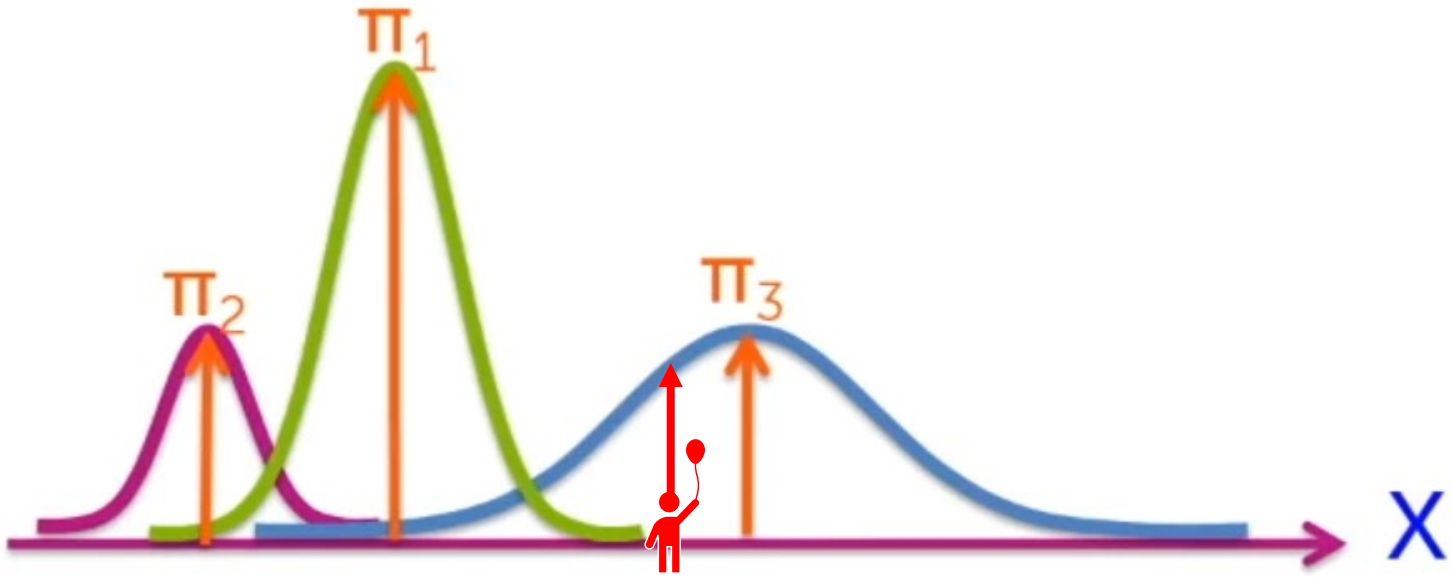
$$p(x_i \mid z_i = k, \mu_k, \Sigma_k) = N(x_i|\mu_k, \Sigma_k)$$

- ถ้าเรามีข้อมูลตัวหนึ่ง



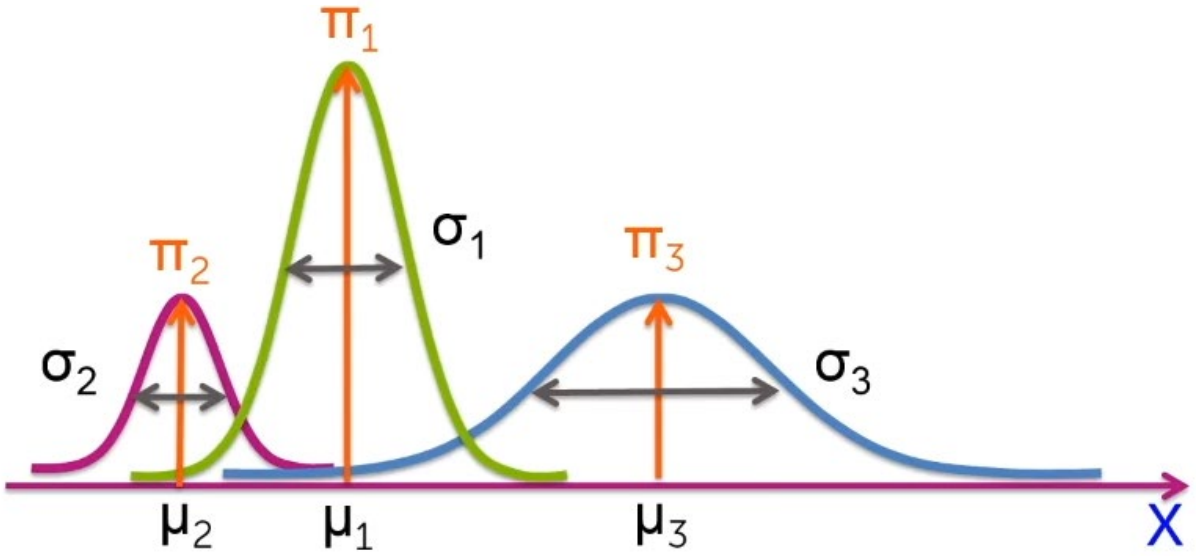
การนำ GMM ไปใช้

- จากสมการดังกล่าว $$p(x_i \mid z_i = k, \mu_k, \Sigma_k) = N(x_i|\mu_k, \Sigma_k)$$
- ถ้าเรามีข้อมูลตัวหนึ่ง เราจะสามารถหา probability ของการเป็นสมาชิกของแต่ละคลัสเตอร์ได้
- ข้อมูลนั้นจะถูกจัดเป็นกลุ่มที่มี probability สูงที่สุด



แนวคิดของ Gaussian Mixture Model

- ต่อจากนี้จะเริ่มอ้างถึงสมการ $p(z_i = k) = \pi_k$
- สมการนี้คือสมการความน่าจะเป็นของการเป็นคลัสเตอร์ k ของข้อมูลตัวที่ i (หรือ $\boldsymbol{x_i}$)
- แต่เรายังไม่เห็นข้อมูล กล่าวคือไม่รู้คลัสเตอร์ที่ถูกต้อง จึงทำให้ไม่รู้ค่า $\boldsymbol{\pi_k}$ได้ ดังนั้นต้องใช้กฎของ Bayes มาช่วยในการคำนวณ จะได้ว่า



$$p(x_i \mid z_i = k, \mu_k, \Sigma_k) = N(x_i|\mu_k, \Sigma_k)$$

Outline

- ข้อจำกัดของ k-Mean
- Gaussian Distribution
- Maximum Likelihood Estimation อย่างง่าย
- Gaussian Mixture Model
  - แนวคิดของ Gaussian Mixture Model
  - การนำไปใช้
  - **คู่มือการใช้โค้ดและตัวอย่างการประยุกต์ใช้**

คู่มือการใช้โค้ดและตัวอย่างการประยุกต์ใช้

class sklearn.mixture.GaussianMixture(n_components=1, *, covariance_type='full', tol=0.001, reg_covar=1e-06, max_iter=100, n_init=1, init_params='kmeans', weights_init=None, means_init=None, precisions_init=None, random_state=None, warm_start=False, verbose=0, verbose_interval=10)[source]

Gaussian Mixture.

Representation of a Gaussian mixture model probability distribution. This class allows to estimate the parameters of a Gaussian mixture distribution.

Read more in the User Guide.

New in version 0.18.

Parameters:

n_components : *int, default=1*
The number of mixture components.

covariance_type : *{'full', 'tied', 'diag', 'spherical'}, default='full'*
String describing the type of covariance parameters to use. Must be one of:

- 'full': each component has its own general covariance matrix.
- 'tied': all components share the same general covariance matrix.
- 'diag': each component has its own diagonal covariance matrix.
- 'spherical': each component has its own single variance.

tol : *float, default=1e-3*
The convergence threshold. EM iterations will stop when the lower bound average gain is below this threshold.

reg_covar : *float, default=1e-6*
Non-negative regularization added to the diagonal of covariance. Allows to assure that the covariance matrices are all positive.

คู่มือการใช้โค้ดและตัวอย่างการประยุกต์ใช้

class sklearn.mixture.GaussianMixture(n_components=1, *, covariance_type='full', tol=0.001, reg_covar=1e-06, max_iter=100, n_init=1, init_params='kmeans', weights_init=None, means_init=None, precisions_init=None, random_state=None, warm_start=False, verbose=0, verbose_interval=10)[source]

max_iter : *int, default=100*
The number of EM iterations to perform.

n_init : *int, default=1*
The number of initializations to perform. The best results are kept.

init_params : *{'kmeans', 'k-means++', 'random', 'random_from_data'}, default='kmeans'*
The method used to initialize the weights, the means and the precisions. String must be one of:

- 'kmeans' : responsibilities are initialized using kmeans.
- 'k-means++' : use the k-means++ method to initialize.
- 'random' : responsibilities are initialized randomly.
- 'random_from_data' : initial means are randomly selected data points.

Changed in version v1.1: `init_params` now accepts 'random_from_data' and 'k-means++' as initialization methods.

weights_init : *array-like of shape (n_components,), default=None*
The user-provided initial weights. If it is None, weights are initialized using the `init_params` method.

คู่มือการใช้โค้ดและตัวอย่างการประยุกต์ใช้

class sklearn.mixture.GaussianMixture(n_components=1, *, covariance_type='full', tol=0.001, reg_covar=1e-06, max_iter=100, n_init=1, init_params='kmeans', weights_init=None, means_init=None, precisions_init=None, random_state=None, warm_start=False, verbose=0, verbose_interval=10)[source]

means_init : *array-like of shape (n_components, n_features), default=None*

The user-provided initial means, If it is None, means are initialized using the `init_params` method.

precisions_init : *array-like, default=None*

The user-provided initial precisions (inverse of the covariance matrices). If it is None, precisions are initialized using the 'init_params' method. The shape depends on 'covariance_type':

```
(n_components,)                        if 'spherical',
(n_features, n_features)               if 'tied',
(n_components, n_features)             if 'diag',
(n_components, n_features, n_features) if 'full'
```

random_state : *int, RandomState instance or None, default=None*

Controls the random seed given to the method chosen to initialize the parameters (see `init_params`). In addition, it controls the generation of random samples from the fitted distribution (see the method `sample`). Pass an int for reproducible output across multiple function calls. See Glossary.

คู่มือการใช้โค้ดและตัวอย่างการประยุกต์ใช้

class sklearn.mixture.GaussianMixture(n_components=1, *, covariance_type='full', tol=0.001, reg_covar=1e-06, max_iter=100, n_init=1, init_params='kmeans', weights_init=None, means_init=None, precisions_init=None, random_state=None, warm_start=False, verbose=0, verbose_interval=10)[source]

Attributes:

weights_ : *array-like of shape (n_components,)*

The weights of each mixture components.

means_ : *array-like of shape (n_components, n_features)*

The mean of each mixture component.

covariances_ : *array-like*

The covariance of each mixture component. The shape depends on `covariance_type`:

```
(n_components,)                        if 'spherical',
(n_features, n_features)               if 'tied',
(n_components, n_features)             if 'diag',
(n_components, n_features, n_features) if 'full'
```

precisions_ : *array-like*

The precision matrices for each component in the mixture. A precision matrix is the inverse of a covariance matrix. A covariance matrix is symmetric positive definite so the mixture of Gaussian can be equivalently parameterized by the precision matrices. Storing the precision matrices instead of the covariance matrices makes it more efficient to compute the log-likelihood of new samples at test time. The shape depends on `covariance_type`:

```
(n_components,)                        if 'spherical',
(n_features, n_features)               if 'tied',
(n_components, n_features)             if 'diag',
(n_components, n_features, n_features) if 'full'
```

คู่มือการใช้โค้ดและตัวอย่างการประยุกต์ใช้

class sklearn.mixture.GaussianMixture(n_components=1, *, covariance_type='full', tol=0.001, reg_covar=1e-06, max_iter=100, n_init=1, init_params='kmeans', weights_init=None, means_init=None, precisions_init=None, random_state=None, warm_start=False, verbose=0, verbose_interval=10)[source]

Examples

```
>>> import numpy as np
>>> from sklearn.mixture import GaussianMixture
>>> X = np.array([[1, 2], [1, 4], [1, 0], [10, 2], [10, 4], [10, 0]])
>>> gm = GaussianMixture(n_components=2, random_state=0).fit(X)
>>> gm.means_
array([[10.,  2.],
       [ 1.,  2.]])
>>> gm.predict([[0, 0], [12, 3]])
array([1, 0])
```

คู่มือการใช้โค้ดและตัวอย่างการประยุกต์ใช้

class sklearn.mixture.GaussianMixture(n_components=1, *, covariance_type='full', tol=0.001, reg_covar=1e-06, max_iter=100, n_init=1, init_params='kmeans', weights_init=None, means_init=None, precisions_init=None, random_state=None, warm_start=False, verbose=0, verbose_interval=10)[source]

Methods

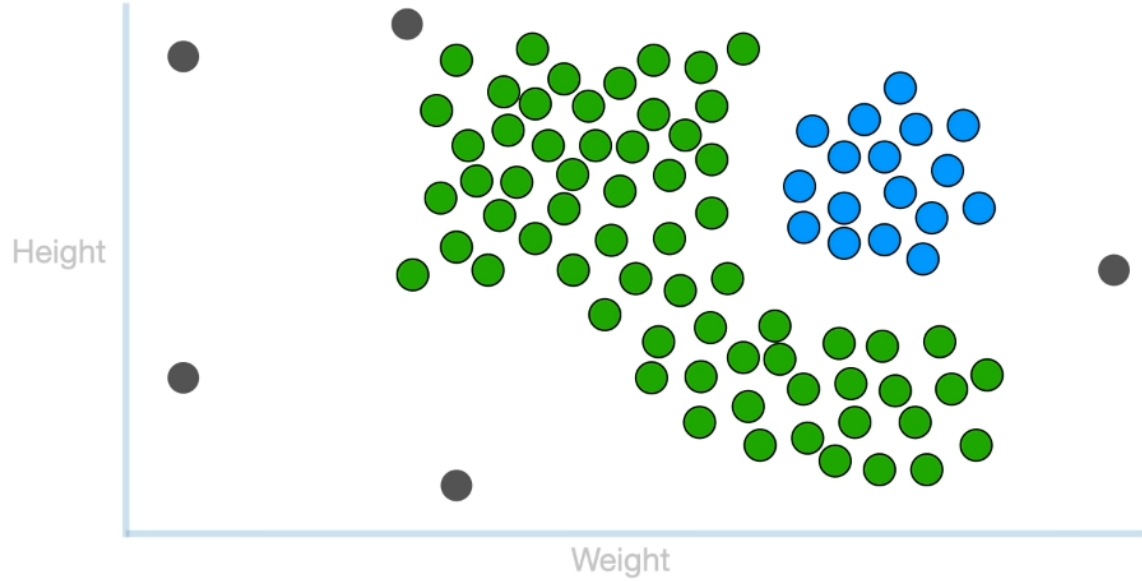
`aic`(X)	Akaike information criterion for the current model on the input X.
`bic`(X)	Bayesian information criterion for the current model on the input X.
`fit`(X[, y])	Estimate model parameters with the EM algorithm.
`fit_predict`(X[, y])	Estimate model parameters using X and predict the labels for X.
`get_params`([deep])	Get parameters for this estimator.
`predict`(X)	Predict the labels for the data samples in X using trained model.
`predict_proba`(X)	Evaluate the components' density for each sample.
`sample`([n_samples])	Generate random samples from the fitted Gaussian distribution.
`score`(X[, y])	Compute the per-sample average log-likelihood of the given data X.
`score_samples`(X)	Compute the log-likelihood of each sample.
`set_params`(**params)	Set the parameters of this estimator.

Outline

- ข้อจำกัดของ k-Mean
- Gaussian Mixture
  - Gaussian Distribution
  - Maximum Likelihood Estimation อย่างง่าย
  - Gaussian Mixture Model
    - แนวคิดของ Gaussian Mixture Model
    - การนำไปใช้
- DBScan

DBSCAN

- DBSCAN ย่อมาจาก Density-Based นั่นเอง
- ซึ่งก็คือการใช้ความหนาแน่นเป็นพื้นฐานในการแบ่งแยกคลัสเตอร์
- ไม่ต้องรู้จำนวนคลัสเตอร์ล่วงหน้า

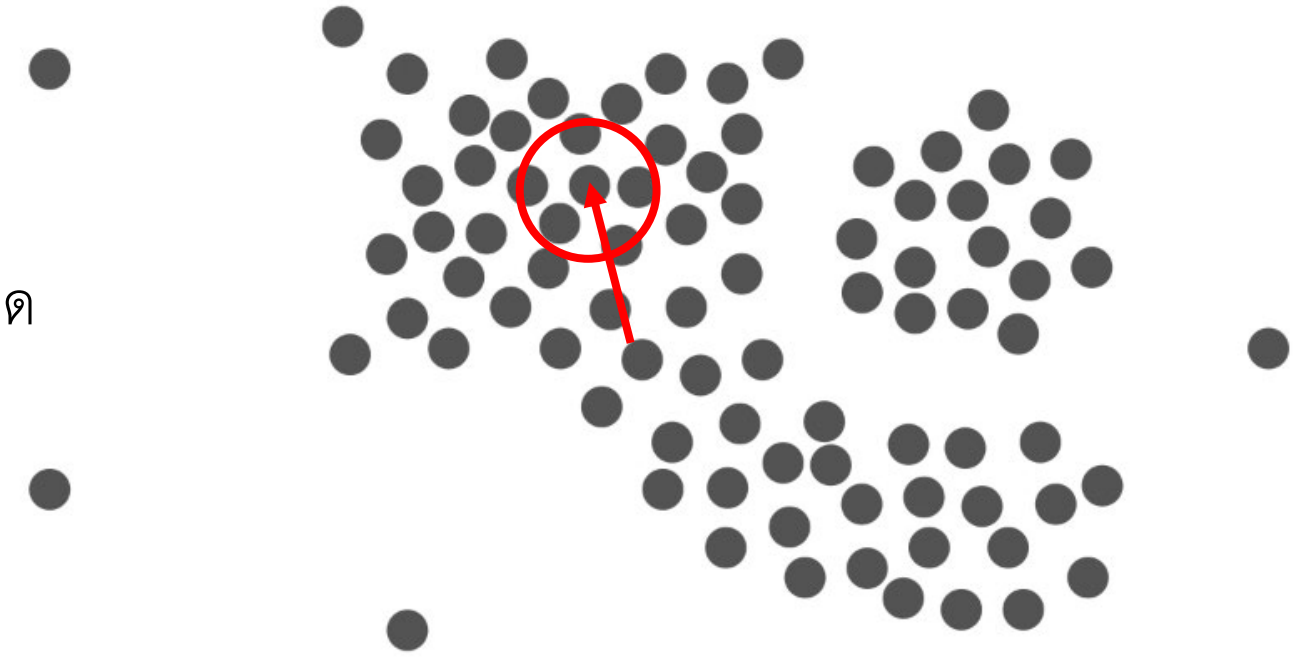


DBSCAN

- ขั้นตอนการทำ DBSCAN

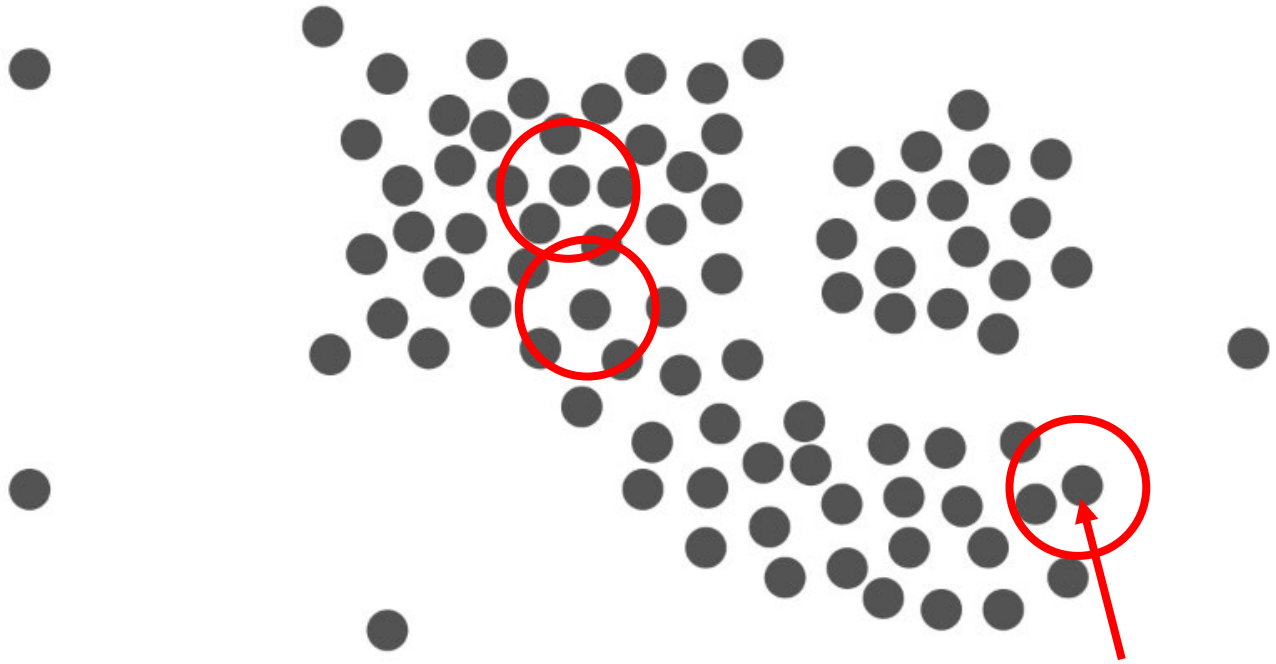
1. นับจุดข้างเคียงของแต่ละจุด

เช่นถ้าเลือกจุดที่ลูกศรชี้ จุดอื่นที่มีระยะห่างไม่เกินที่กำหนดไว้จะถูกนับ ดังนั้นจุดนี้จะนับได้ 8



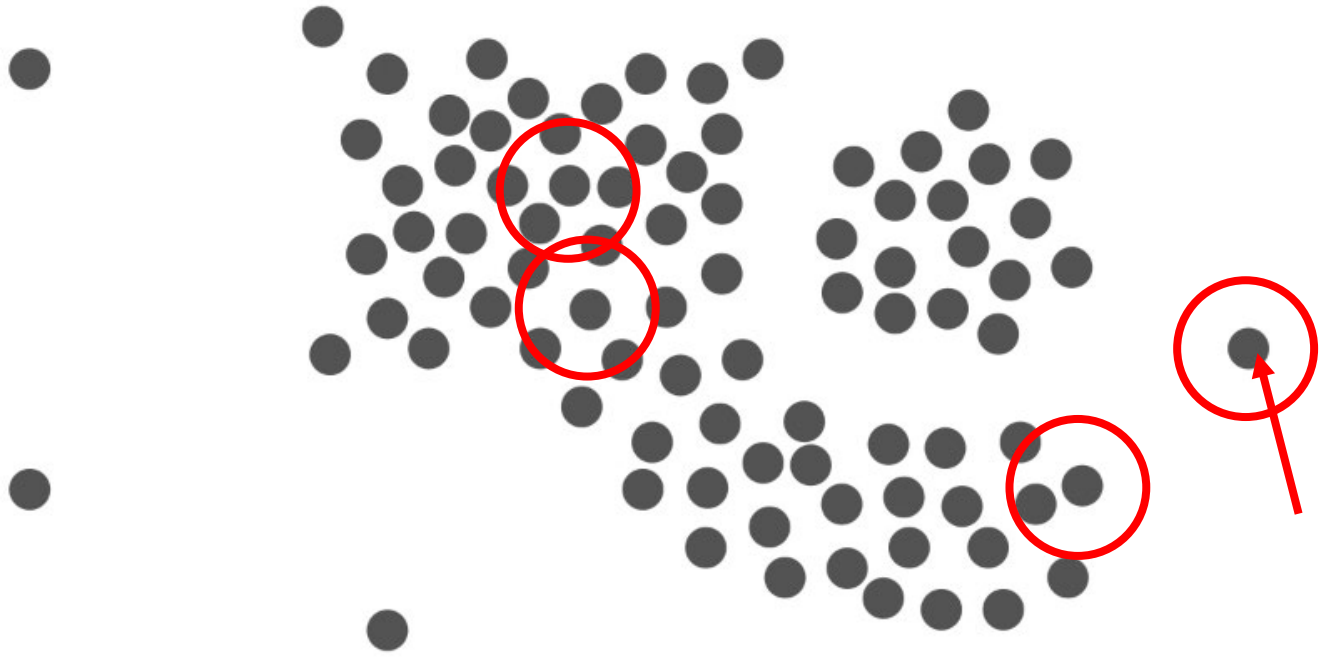
DBSCAN

- ขั้นตอนการทำ DBSCAN
  1. นับจุดข้างเคียงของแต่ละจุด
     แต่จุดนี้นับได้ 2 เท่านั้น



DBSCAN

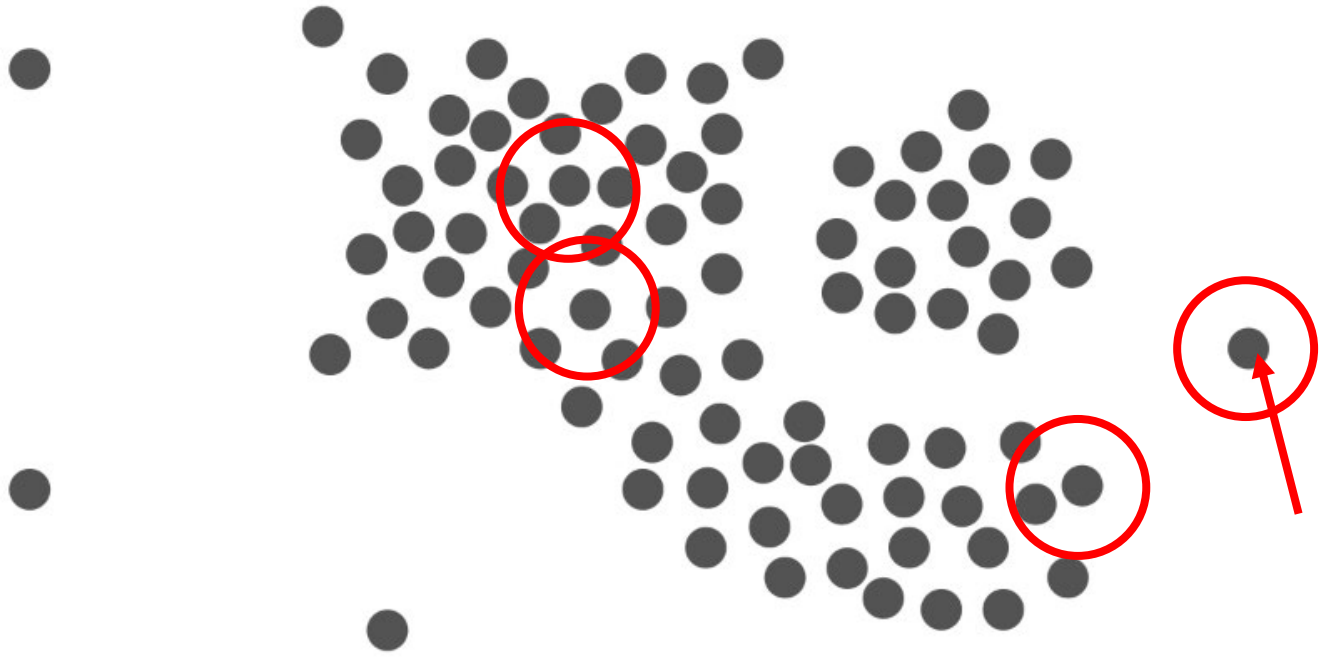
- ขั้นตอนการทำ DBSCAN
  1. นับจุดข้างเคียงของแต่ละจุด และจุดนี้ไม่มีเลย



DBSCAN

- ขั้นตอนการทำ DBSCAN
  1. นับจุดข้างเคียงของแต่ละจุด
  2. ระบุ Core point (จุดที่มีจุดข้างเคียงมากกว่าจำนวนที่กำหนด)

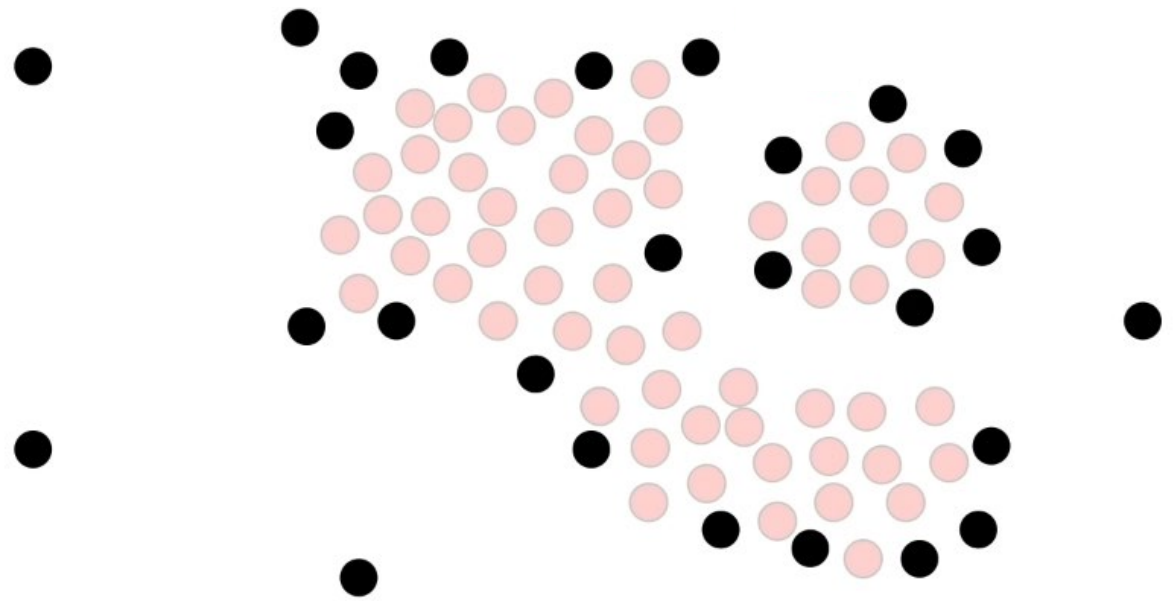
     เช่นในที่นี้จะยกตัวอย่างโดยกำหนดให้ Core point มีจำนวนจุดข้างเคียงมากกว่า 4 จุด ดังนั้น จุดใดที่มีจุดข้างเคียงน้อยกว่า 4 จุด จะถูกตัดออกจากการเป็น Core Point



DBSCAN

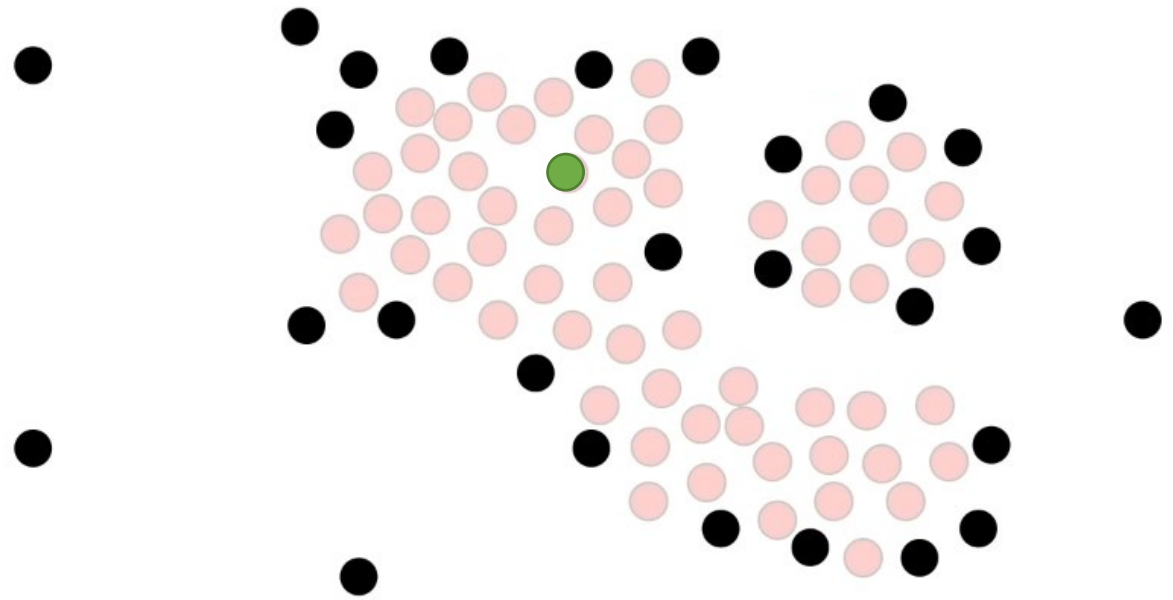
- ขั้นตอนการทำ DBSCAN
  1. นับจุดข้างเคียงของแต่ละจุด
  2. ระบุ Core point (จุดที่มีจุดข้างเคียงมากกว่าจำนวนที่กำหนด)

  จุดสีแดงคือ Core point



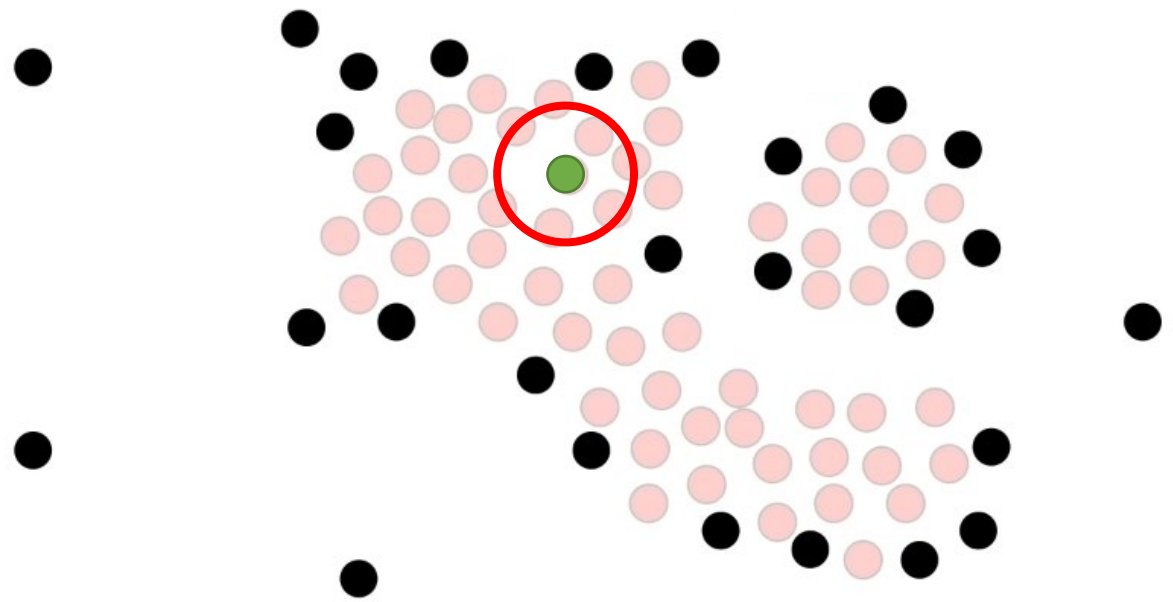
DBSCAN

- ขั้นตอนการทำ DBSCAN
  1. นับจุดข้างเคียงของแต่ละจุด
  2. ระบุ Core point (จุดที่มีจุดข้างเคียงมากกว่าจำนวนที่กำหนด)
  3. สุ่มเลือกจุด Core point มา 1 จุดและระบุให้เป็นคลัสเตอร์ที่ 1



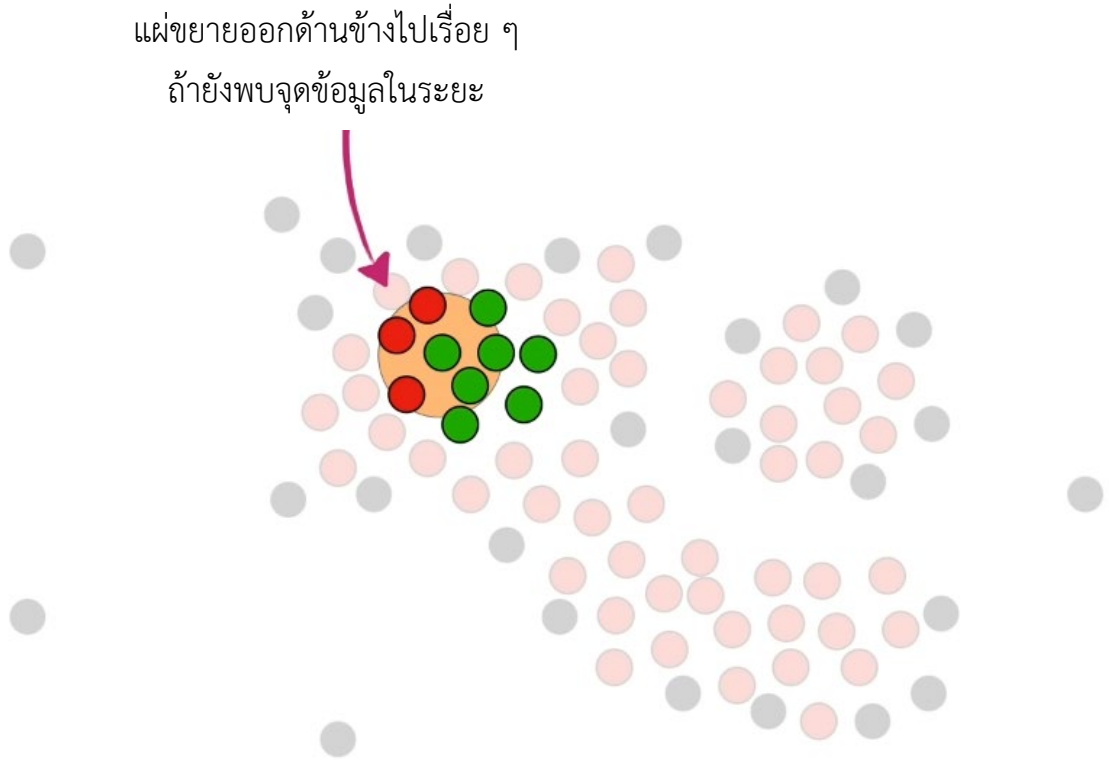
DBSCAN

- ขั้นตอนการทำ DBSCAN
  1. นับจุดข้างเคียงของแต่ละจุด
  2. ระบุ Core point (จุดที่มีจุดข้างเคียงมากกว่าจำนวนที่กำหนด)
  3. สุ่มเลือกจุด Core point มา 1 จุดและระบุให้เป็นคลัสเตอร์ที่ 1
  4. จุด Core point ที่ซ้อนทับในระยะวงกลมสีแดง (Distance น้อยกว่าที่กำหนด) จะถูกตีความว่าเป็นคลัสเตอร์เดียวกัน



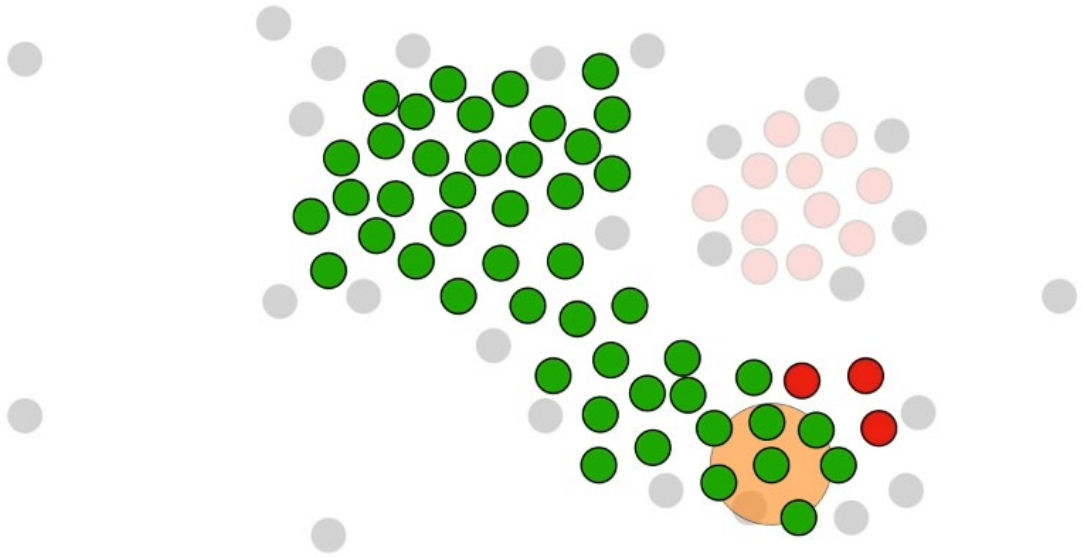
DBSCAN

- ขั้นตอนการทำ DBSCAN
  1. นับจุดข้างเคียงของแต่ละจุด
  2. ระบุ Core point (จุดที่มีจุดข้างเคียงมากกว่าจำนวนที่กำหนด)
  3. สุ่มเลือกจุด Core point มา 1 จุดและระบุให้เป็นคลัสเตอร์ที่ 1
  4. จุด Core point ที่ซ้อนทับในระยะวงกลมสีแดง (Distance น้อยกว่าที่กำหนด) จะถูกตีความว่าเป็นคลัสเตอร์เดียวกัน



DBSCAN

- ขั้นตอนการทำ DBSCAN
  1. นับจุดข้างเคียงของแต่ละจุด
  2. ระบุ Core point (จุดที่มีจุดข้างเคียงมากกว่าจำนวนที่กำหนด)
  3. สุ่มเลือกจุด Core point มา 1 จุดและระบุให้เป็นคลัสเตอร์ที่ 1
  4. จุด Core point ที่ซ้อนทับในระยะวงกลมสีแดง (Distance น้อยกว่าที่กำหนด) จะถูกตีความว่าเป็นคลัสเตอร์เดียวกัน

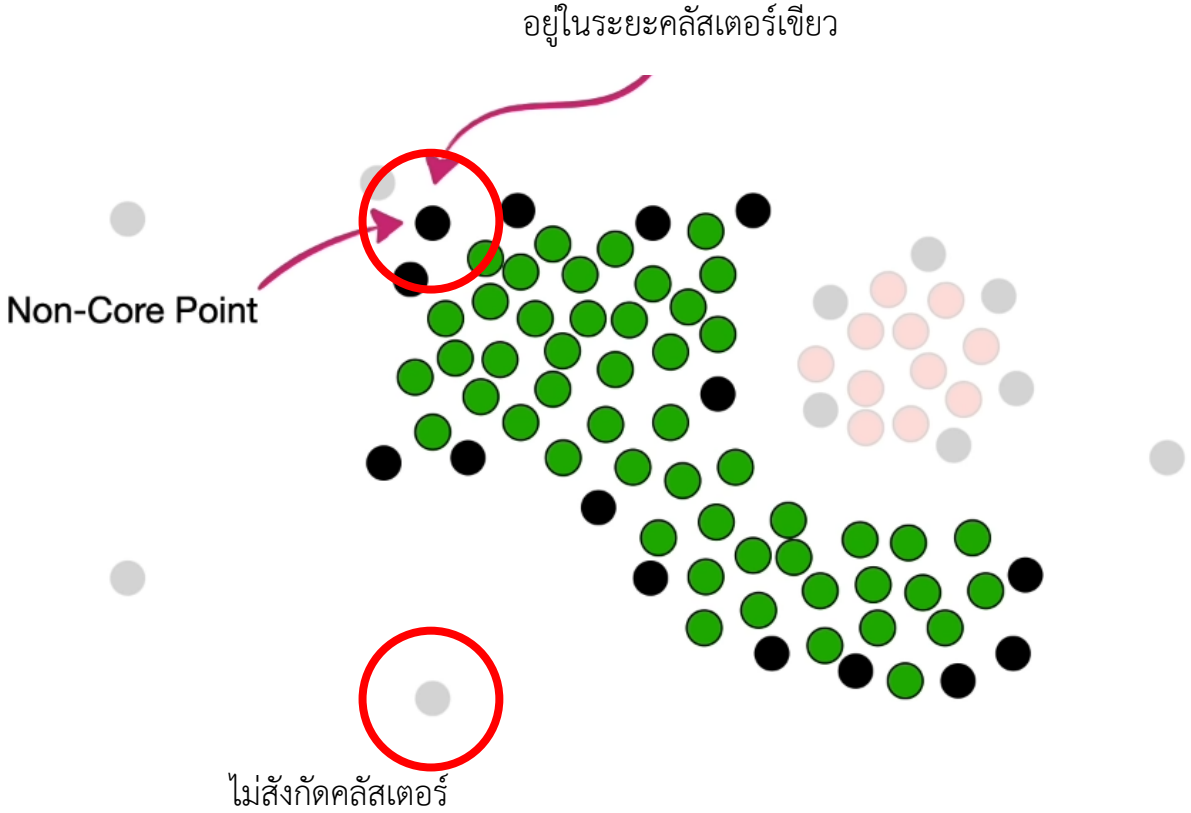


DBSCAN

- ขั้นตอนการทำ DBSCAN

  5. เมื่อ Core Point ครบทั้งก้อนแล้ว จึง พิจารณา Non-core point ถ้า non-core point อยู่ในระยะของ Core point ที่มีสังกัด Cluster แล้ว จะถือว่าอยู่ใน cluster นั้น

  * Non-core point ไม่สามารถขยายขอบเขตของคลัสเตอร์ได้

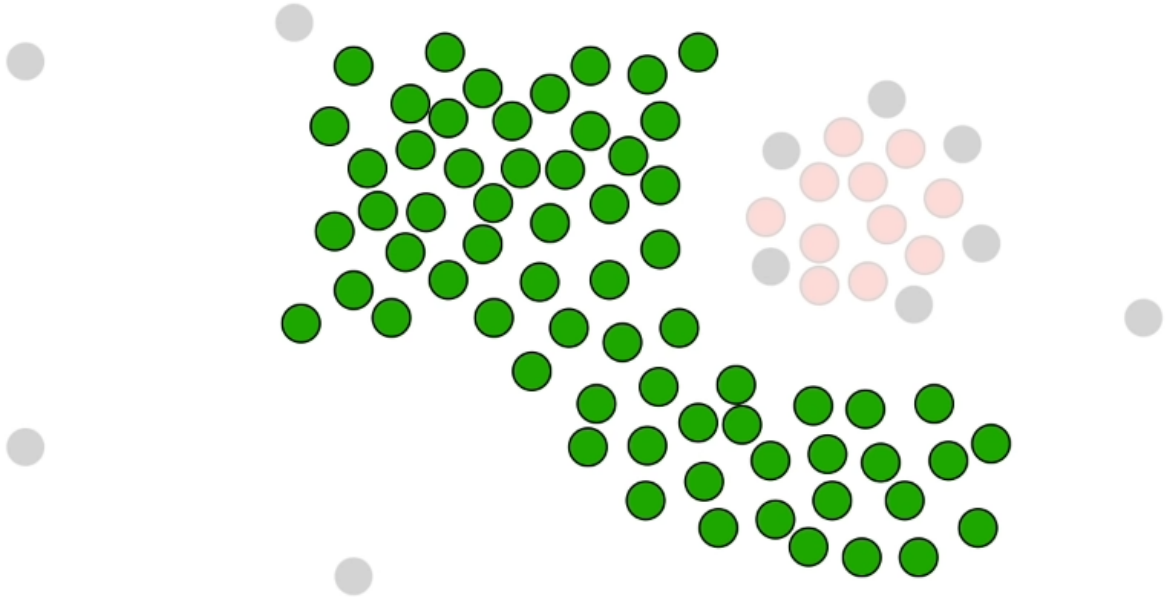


DBSCAN

- ขั้นตอนการทำ DBSCAN

  5. เมื่อ Core Point ครบทั้งก้อนแล้ว จึงพิจารณา Non-core point ถ้า non-core point อยู่ในระยะของ Core point ที่มีสังกัด Cluster แล้ว จะถือว่าอยู่ใน cluster นั้น

  6. พิจารณา Core Point ที่ยังไม่มีสังกัด กำหนดสังกัดเป็นคลัสเตอร์ใหม่ และเริ่มทำตามลำดับ 1-5 อีกครั้ง

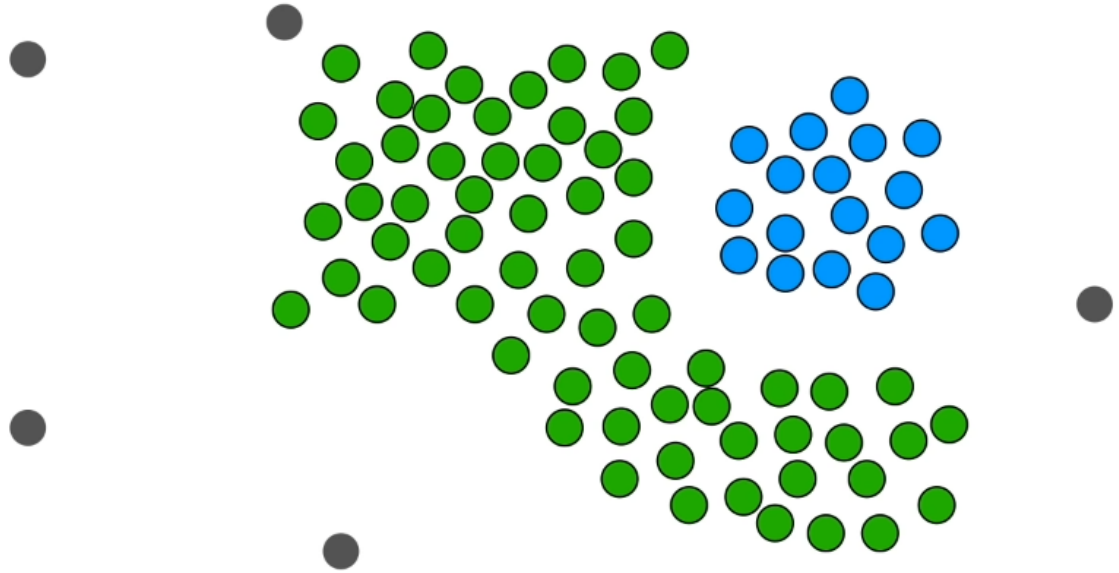


DBSCAN

- ขั้นตอนการทำ DBSCAN

  5. เมื่อ Core Point ครบทั้งก้อนแล้ว จึงพิจารณา Non-core point ถ้า non-core point อยู่ในระยะของ Core point ที่มีสังกัด Cluster แล้ว จะถือว่าอยู่ใน cluster นั้น

  6. พิจารณา Core Point ที่ยังไม่มีสังกัด กำหนดสังกัดเป็นคลัสเตอร์ใหม่ และเริ่มทำตามลำดับ 1-5 อีกครั้ง



DBSCAN

class `sklearn.cluster.`**DBSCAN**(*eps=0.5, *, min_samples=5, metric='euclidean', metric_params=None, algorithm='auto', leaf_size=30, p=None, n_jobs=None*)

Parameters:

eps : *float, default=0.5*

The maximum distance between two samples for one to be considered as in the neighborhood of the other. This is not a maximum bound on the distances of points within a cluster. This is the most important DBSCAN parameter to choose appropriately for your data set and distance function.

min_samples : *int, default=5*

The number of samples (or total weight) in a neighborhood for a point to be considered as a core point. This includes the point itself.

metric : *str, or callable, default='euclidean'*

The metric to use when calculating distance between instances in a feature array. If metric is a string or callable, it must be one of the options allowed by `sklearn.metrics.pairwise_distances` for its metric parameter. If metric is "precomputed", X is assumed to be a distance matrix and must be square. X may be a sparse graph, in which case only "nonzero" elements may be considered neighbors for DBSCAN.

New in version 0.17: metric *precomputed* to accept precomputed sparse matrix.

DBSCAN

class `sklearn.cluster.`**DBSCAN**(*eps=0.5, *, min_samples=5, metric='euclidean', metric_params=None, algorithm='auto', leaf_size=30, p=None, n_jobs=None*)

Parameters:

eps : *float, default=0.5*

The maximum distance between two samples for one to be considered as in the neighborhood of the other. This is not a maximum bound on the distances of points within a cluster. This is the most important DBSCAN parameter to choose appropriately for your data set and distance function.

min_samples : *int, default=5*

The number of samples (or total weight) in a neighborhood for a point to be considered as a core point. This includes the point itself.

metric : *str, or callable, default='euclidean'*

The metric to use when calculating distance between instances in a feature array. If metric is a string or callable, it must be one of the options allowed by `sklearn.metrics.pairwise_distances` for its metric parameter. If metric is "precomputed", X is assumed to be a distance matrix and must be square. X may be a sparse graph, in which case only "nonzero" elements may be considered neighbors for DBSCAN.

New in version 0.17: metric *precomputed* to accept precomputed sparse matrix.

DBSCAN

class `sklearn.cluster.`**DBSCAN**(*eps=0.5, *, min_samples=5, metric='euclidean', metric_params=None, algorithm='auto', leaf_size=30, p=None, n_jobs=None*)

fit(*X, y=None, sample_weight=None*) [source]

Perform DBSCAN clustering from features, or distance matrix.

Parameters:

X : *{array-like, sparse matrix} of shape (n_samples, n_features), or (n_samples, n_samples)*
Training instances to cluster, or distances between instances if `metric='precomputed'`. If a sparse matrix is provided, it will be converted into a sparse `csr_matrix`.

y : *Ignored*
Not used, present here for API consistency by convention.

sample_weight : *array-like of shape (n_samples,), default=None*
Weight of each sample, such that a sample with a weight of at least `min_samples` is by itself a core sample; a sample with a negative weight may inhibit its eps-neighbor from being core. Note that weights are absolute, and default to 1.

Returns:

self : *object*
Returns a fitted instance of self.

DBSCAN

class `sklearn.cluster.`**DBSCAN**(*eps=0.5, *, min_samples=5, metric='euclidean', metric_params=None, algorithm='auto', leaf_size=30, p=None, n_jobs=None*)

get_params(*deep=True*) [source]

Get parameters for this estimator.

Parameters:	**deep : *bool, default=True*** If True, will return the parameters for this estimator and contained subobjects that are estimators.
Returns:	**params : *dict*** Parameter names mapped to their values.

set_params(***params*) [source]

Set the parameters of this estimator.

The method works on simple estimators as well as on nested objects (such as `Pipeline`). The latter have parameters of the form `<component>__<parameter>` so that it's possible to update each component of a nested object.

Parameters:	****params : *dict*** Estimator parameters.
Returns:	**self : *estimator instance*** Estimator instance.

DBSCAN

class `sklearn.cluster.`**DBSCAN**(*eps=0.5, *, min_samples=5, metric='euclidean', metric_params=None, algorithm='auto', leaf_size=30, p=None, n_jobs=None*)

```
>>> from sklearn.cluster import DBSCAN
>>> import numpy as np
>>> X = np.array([[1, 2], [2, 2], [2, 3],
...               [8, 7], [8, 8], [25, 80]])
>>> clustering = DBSCAN(eps=3, min_samples=2).fit(X)
>>> clustering.labels_
array([ 0,  0,  0,  1,  1, -1])
>>> clustering
DBSCAN(eps=3, min_samples=2)
```

Conclusion

- ข้อจำกัดของ k-Mean
- Gaussian Mixture
  - Gaussian Distribution
  - Maximum Likelihood Estimation อย่างง่าย
  - Gaussian Mixture Model
    - แนวคิดของ Gaussian Mixture Model
    - การนำไปใช้
- DBScan