Machine Learning in Healthcare

#C13 K-means and GMM (unsupervised learning)

Technion-IIT, Haifa, Israel

Assist. Prof. Joachim Behar
Biomedical Engineering Faculty
Technion-IIT

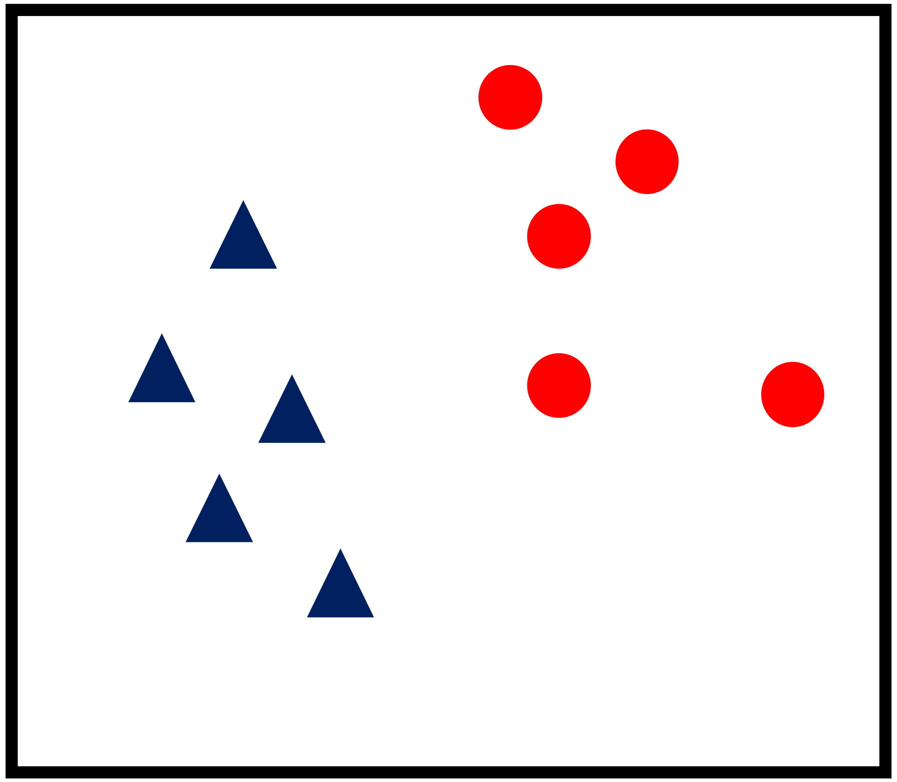
AIMLab.

Supervised vs. Unsupervised

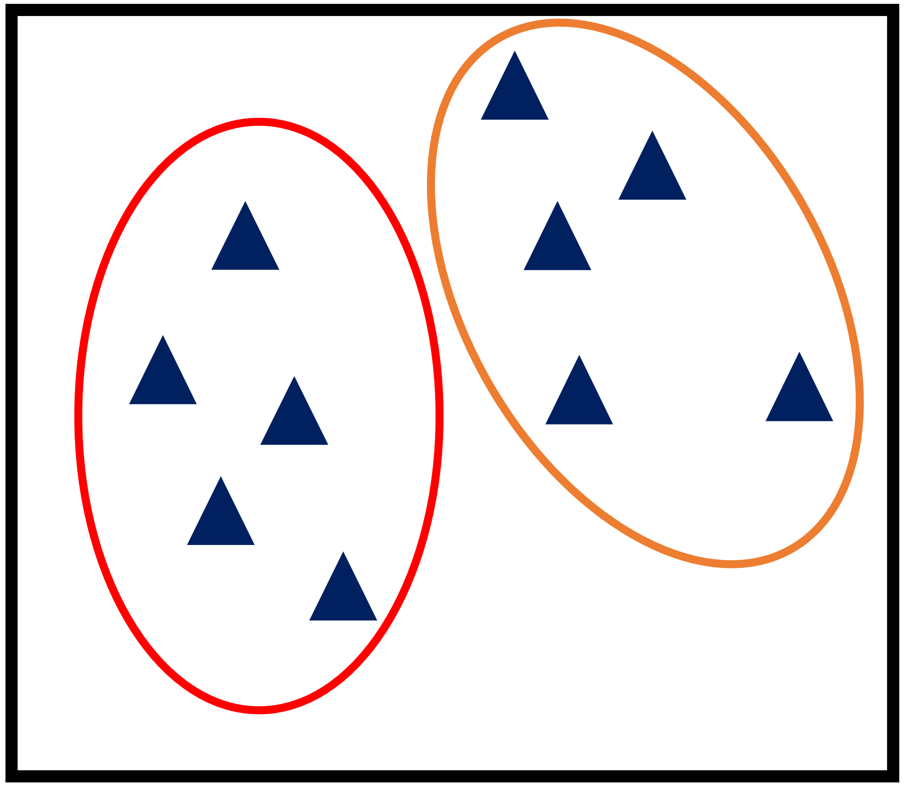
AIMLab.

Supervised versus Unsupervised Learning

Learning a mapping from data to outcome



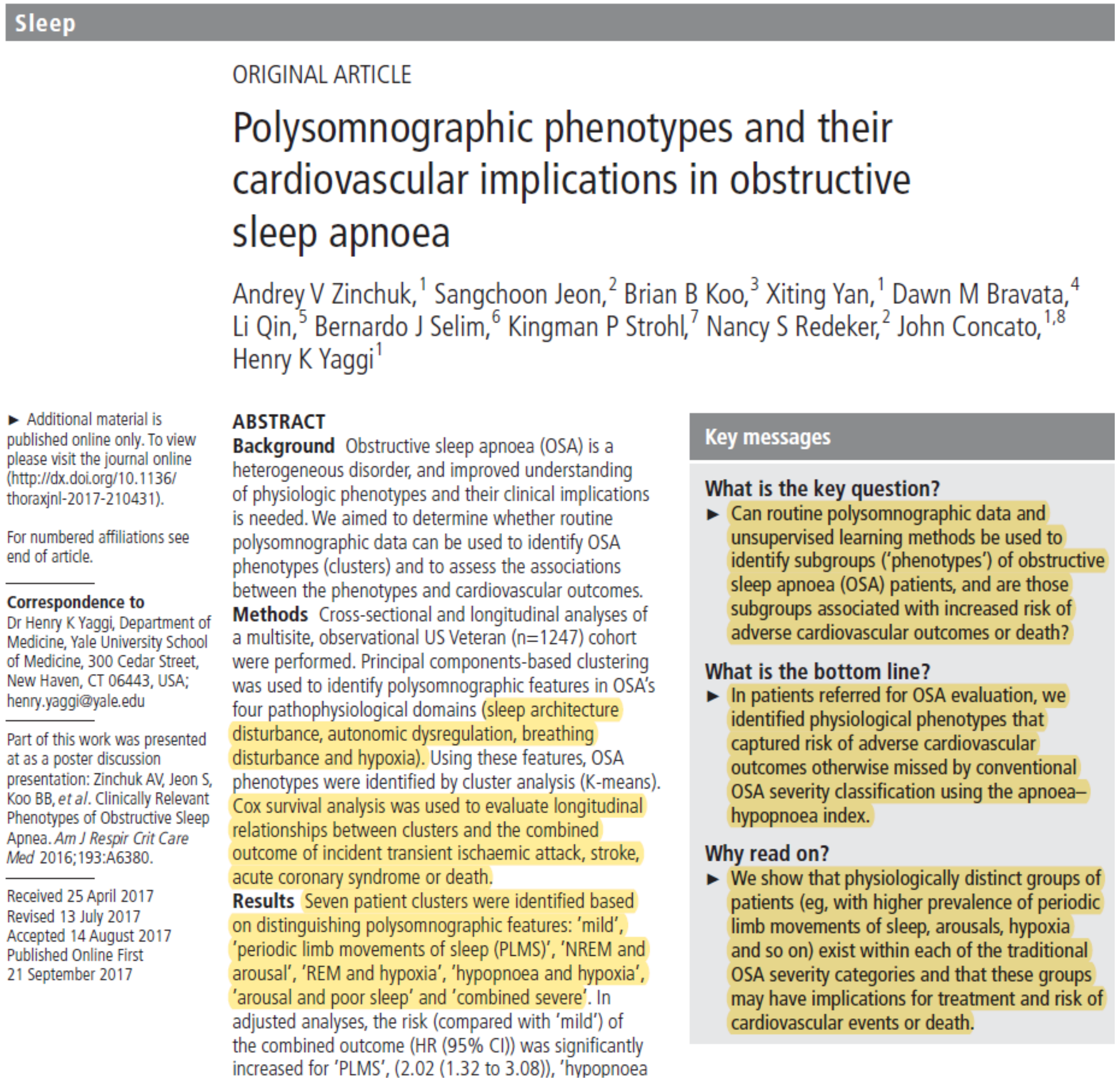
Study of patterns in data



AIMLab.

Why unsupervised?

- Find unknown patterns in data.
- Interpretability.
- No need to label.



Sleep

ORIGINAL ARTICLE

Polysomnographic phenotypes and their cardiovascular implications in obstructive sleep apnoea

Andrey V Zinchuk,[1] Sangchoon Jeon,[2] Brian B Koo,[3] Xiting Yan,[1] Dawn M Bravata,[4] Li Qin,[5] Bernardo J Selim,[6] Kingman P Strohl,[7] Nancy S Redeker,[2] John Concato,[1,8] Henry K Yaggi[1]

► Additional material is published online only. To view please visit the journal online (http://dx.doi.org/10.1136/thoraxjnl-2017-210431).

For numbered affiliations see end of article.

Correspondence to
Dr Henry K Yaggi, Department of Medicine, Yale University School of Medicine, 300 Cedar Street, New Haven, CT 06443, USA; henry.yaggi@yale.edu

Part of this work was presented at as a poster discussion presentation: Zinchuk AV, Jeon S, Koo BB, *et al*. Clinically Relevant Phenotypes of Obstructive Sleep Apnea. *Am J Respir Crit Care Med* 2016;193:A6380.

Received 25 April 2017
Revised 13 July 2017
Accepted 14 August 2017
Published Online First 21 September 2017

ABSTRACT

Background Obstructive sleep apnoea (OSA) is a heterogeneous disorder, and improved understanding of physiologic phenotypes and their clinical implications is needed. We aimed to determine whether routine polysomnographic data can be used to identify OSA phenotypes (clusters) and to assess the associations between the phenotypes and cardiovascular outcomes.
Methods Cross-sectional and longitudinal analyses of a multisite, observational US Veteran (n=1247) cohort were performed. Principal components-based clustering was used to identify polysomnographic features in OSA's four pathophysiological domains (sleep architecture disturbance, autonomic dysregulation, breathing disturbance and hypoxia). Using these features, OSA phenotypes were identified by cluster analysis (K-means). Cox survival analysis was used to evaluate longitudinal relationships between clusters and the combined outcome of incident transient ischaemic attack, stroke, acute coronary syndrome or death.
Results Seven patient clusters were identified based on distinguishing polysomnographic features: 'mild', 'periodic limb movements of sleep (PLMS)', 'NREM and arousal', 'REM and hypoxia', 'hypopnoea and hypoxia', 'arousal and poor sleep' and 'combined severe'. In adjusted analyses, the risk (compared with 'mild') of the combined outcome (HR (95% CI)) was significantly increased for 'PLMS', (2.02 (1.32 to 3.08)), 'hypopnoea

Key messages

What is the key question?
► Can routine polysomnographic data and unsupervised learning methods be used to identify subgroups ('phenotypes') of obstructive sleep apnoea (OSA) patients, and are those subgroups associated with increased risk of adverse cardiovascular outcomes or death?

What is the bottom line?
► In patients referred for OSA evaluation, we identified physiological phenotypes that captured risk of adverse cardiovascular outcomes otherwise missed by conventional OSA severity classification using the apnoea–hypopnoea index.

Why read on?
► We show that physiologically distinct groups of patients (eg, with higher prevalence of periodic limb movements of sleep, arousals, hypoxia and so on) exist within each of the traditional OSA severity categories and that these groups may have implications for treatment and risk of cardiovascular events or death.

K-Means Intuition

K-means Intuition

- Looks like two clusters
- How do we figure out these two clusters?
- Standardize the data (always!)
- Choose some initial guess of the cluster centers.

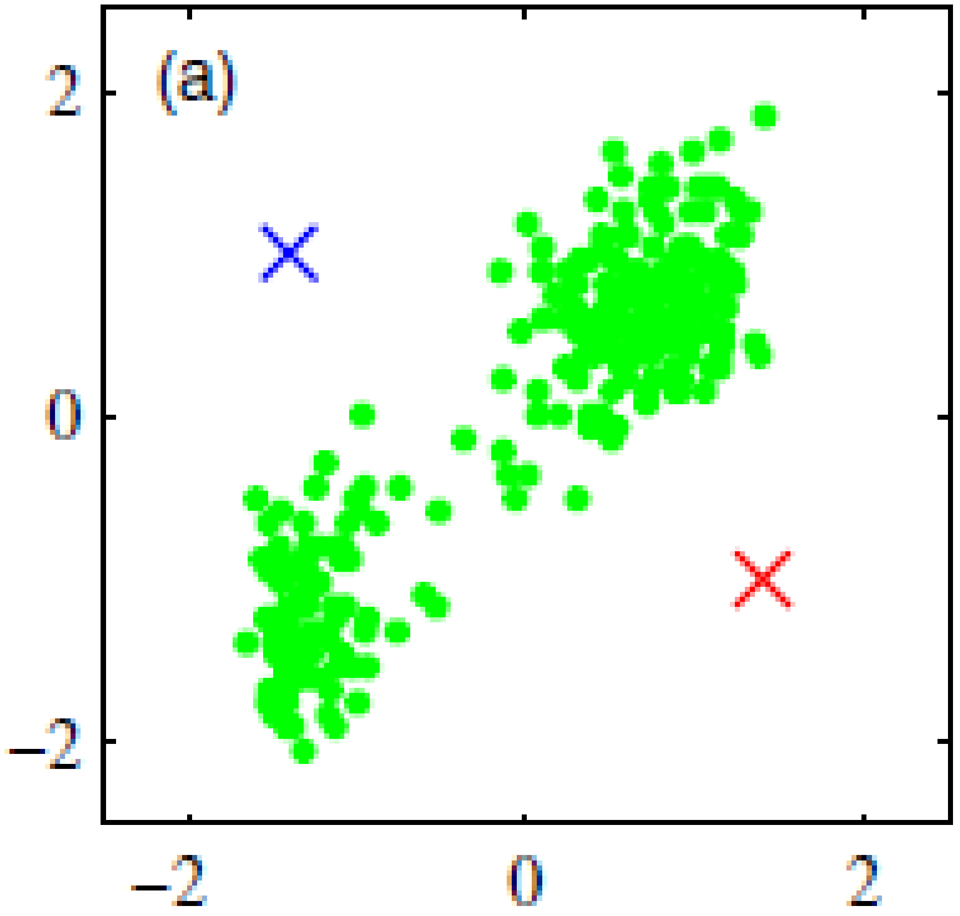


Image: From Bishop's Pattern recognition and machine learning, Figure 9.1

K-means Intuition

- Assign each point to the closest center (b)
- Compute new class centers (c)
- Assign point to closest center (d)
- Computer cluster centers (e)
- Iterate.

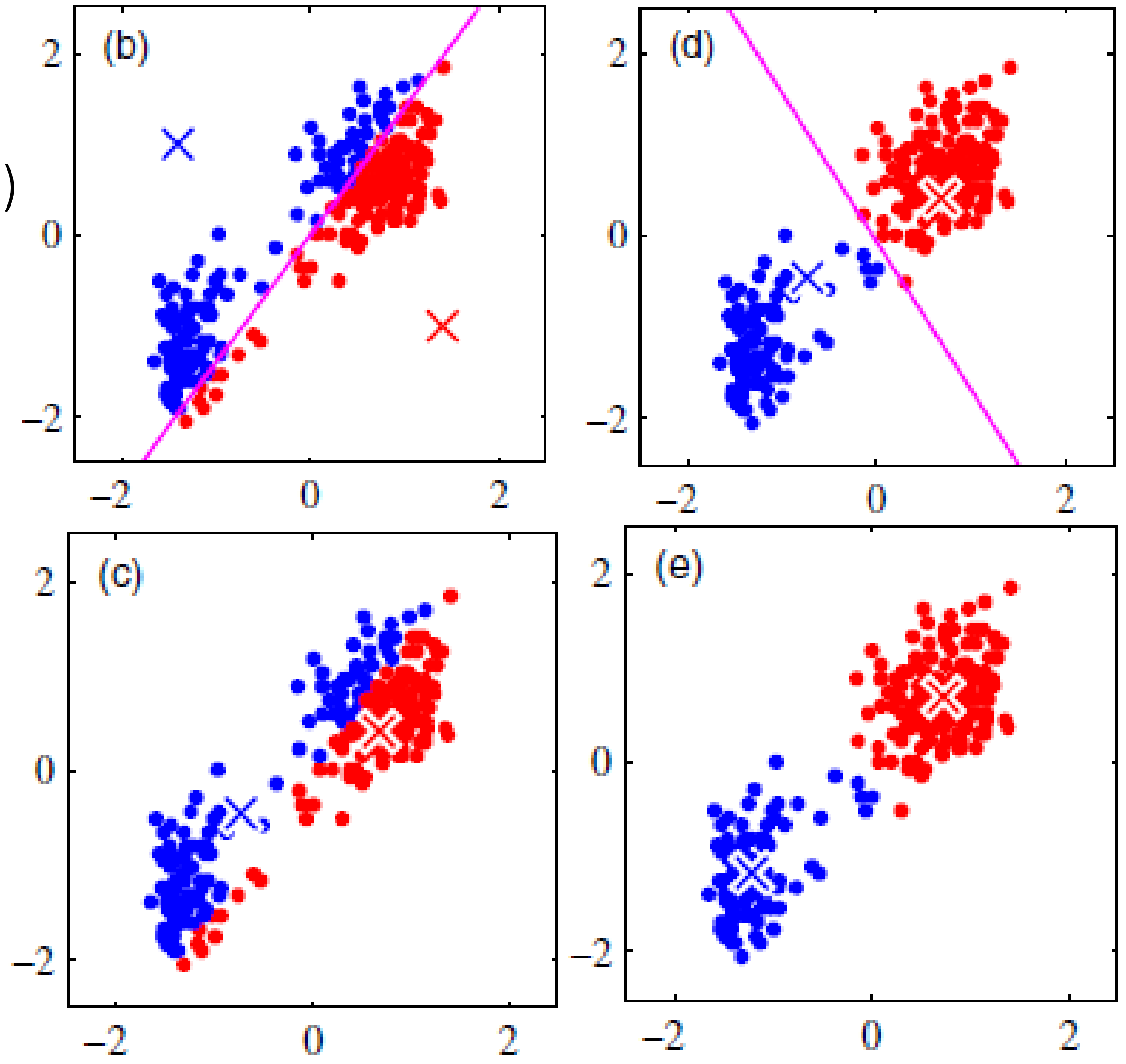


Image: From Bishop's Pattern recognition and machine learning, Figure 9.1

K-means Intuition

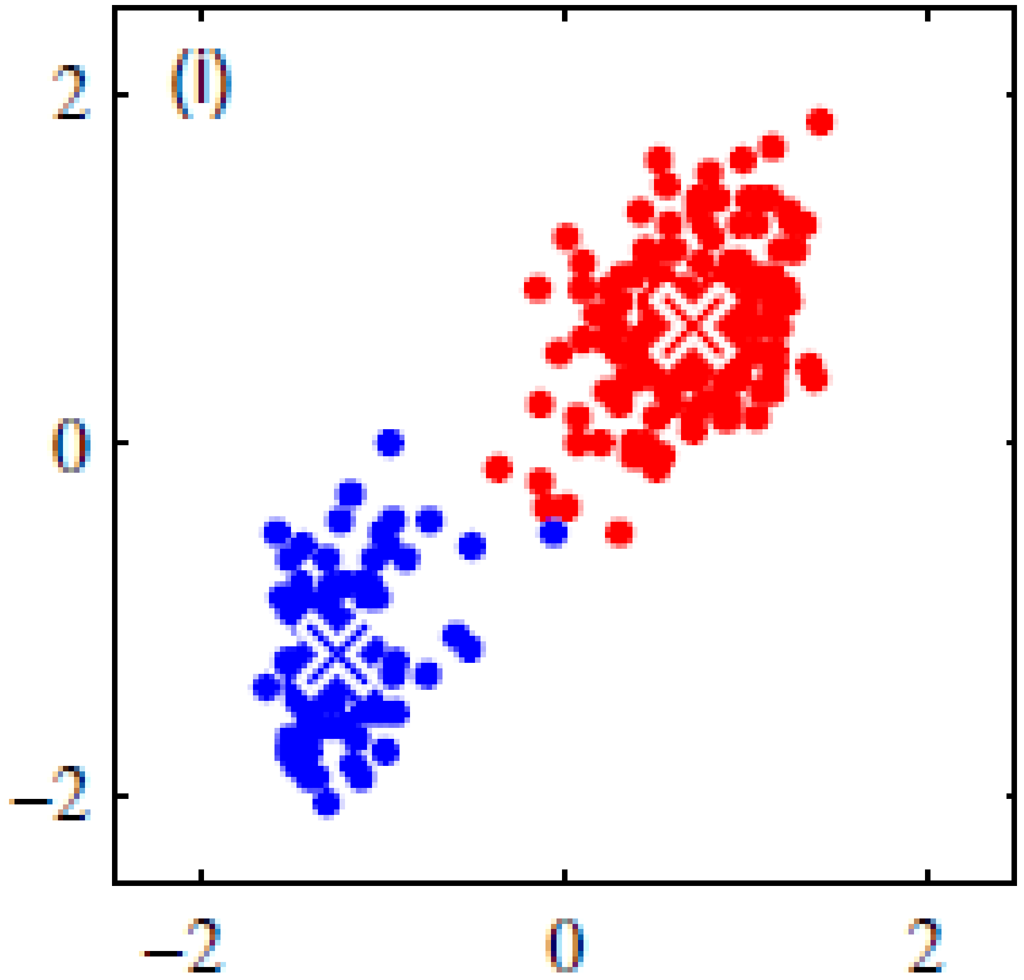
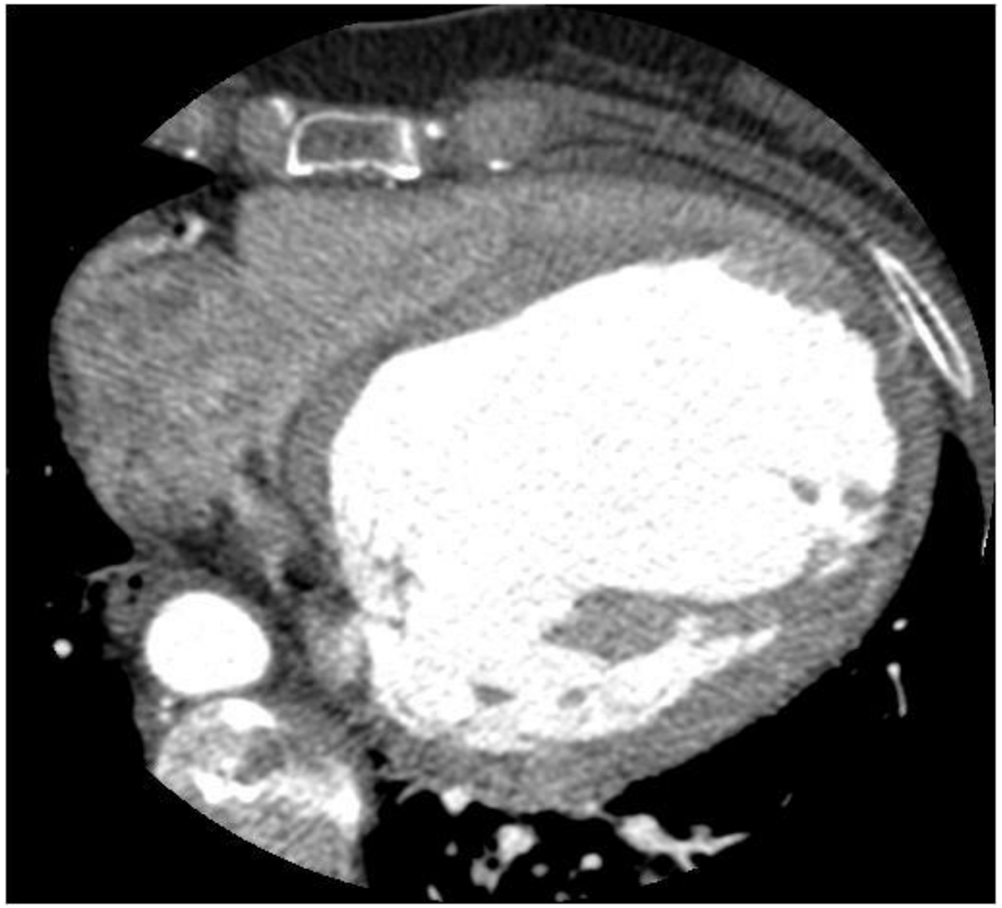
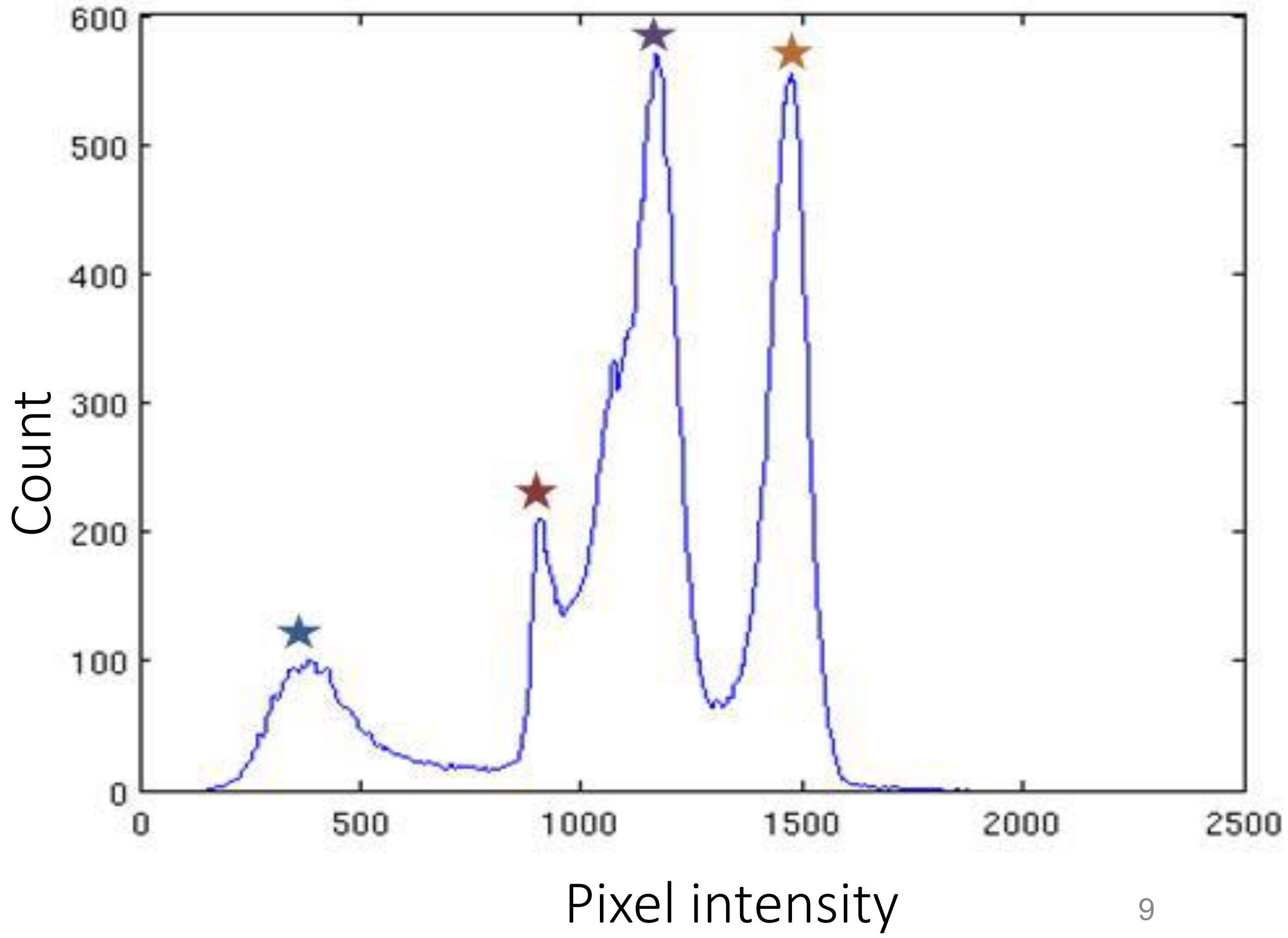


Image: From Bishop's Pattern recognition and machine learning, Figure 9.1

K-means Intuition

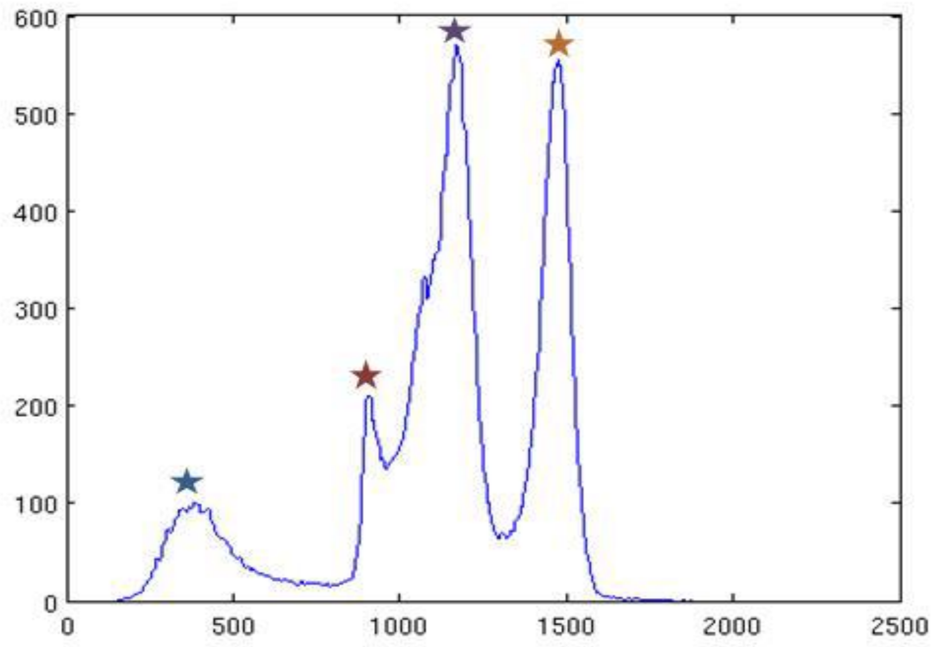
- Example: segmentation of left ventricle





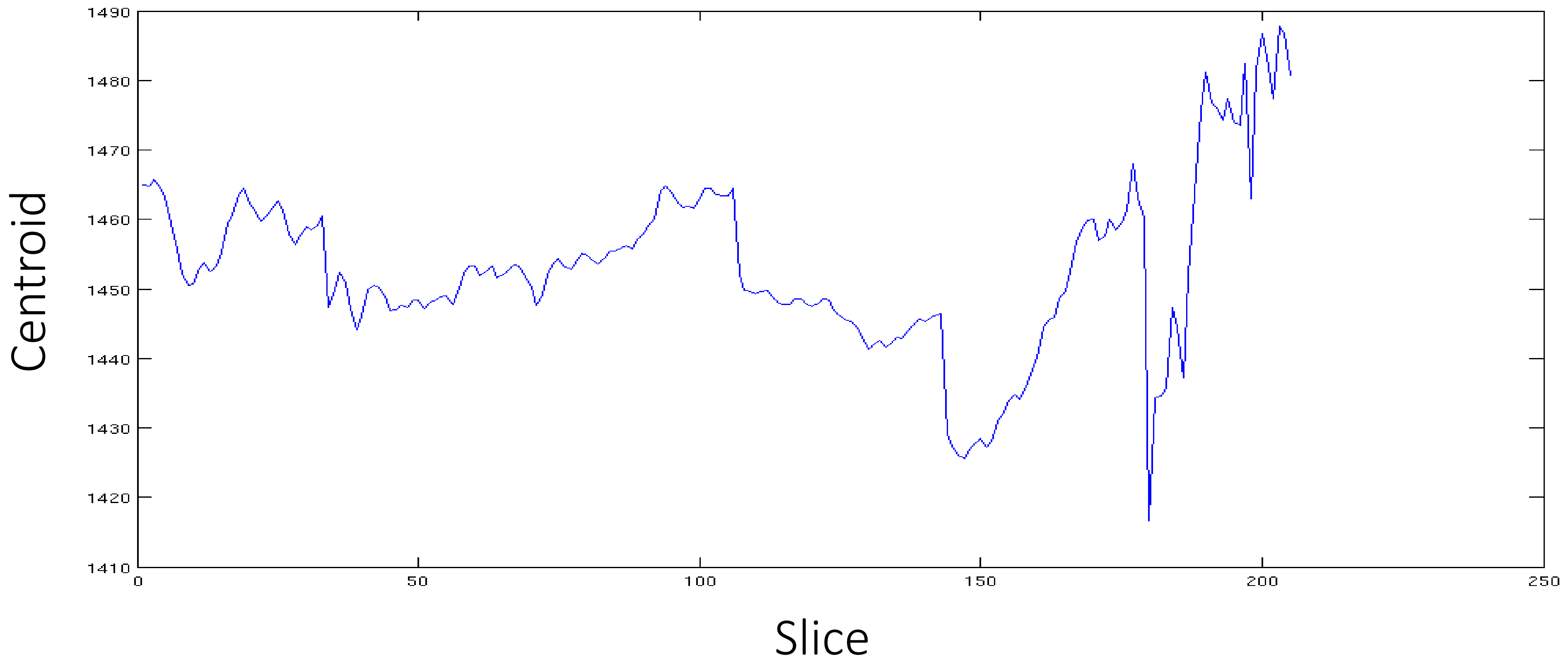
K-means Intuition

- Example: segmentation of left ventricle



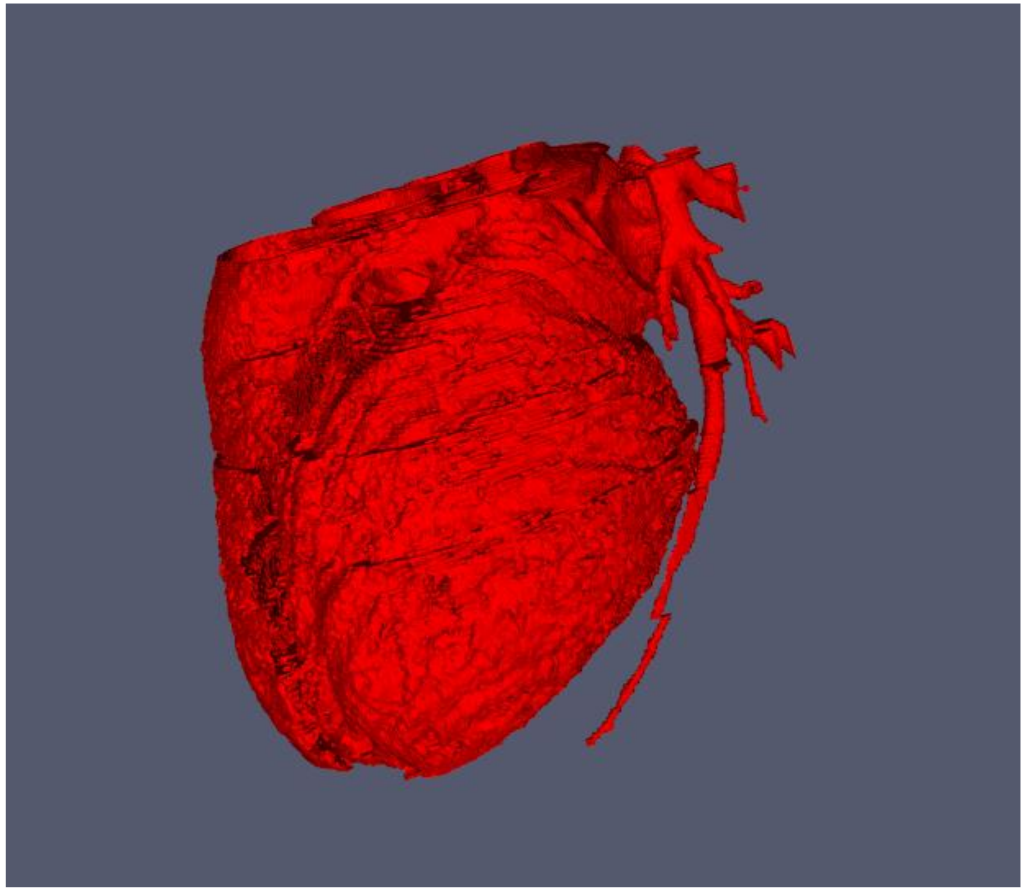
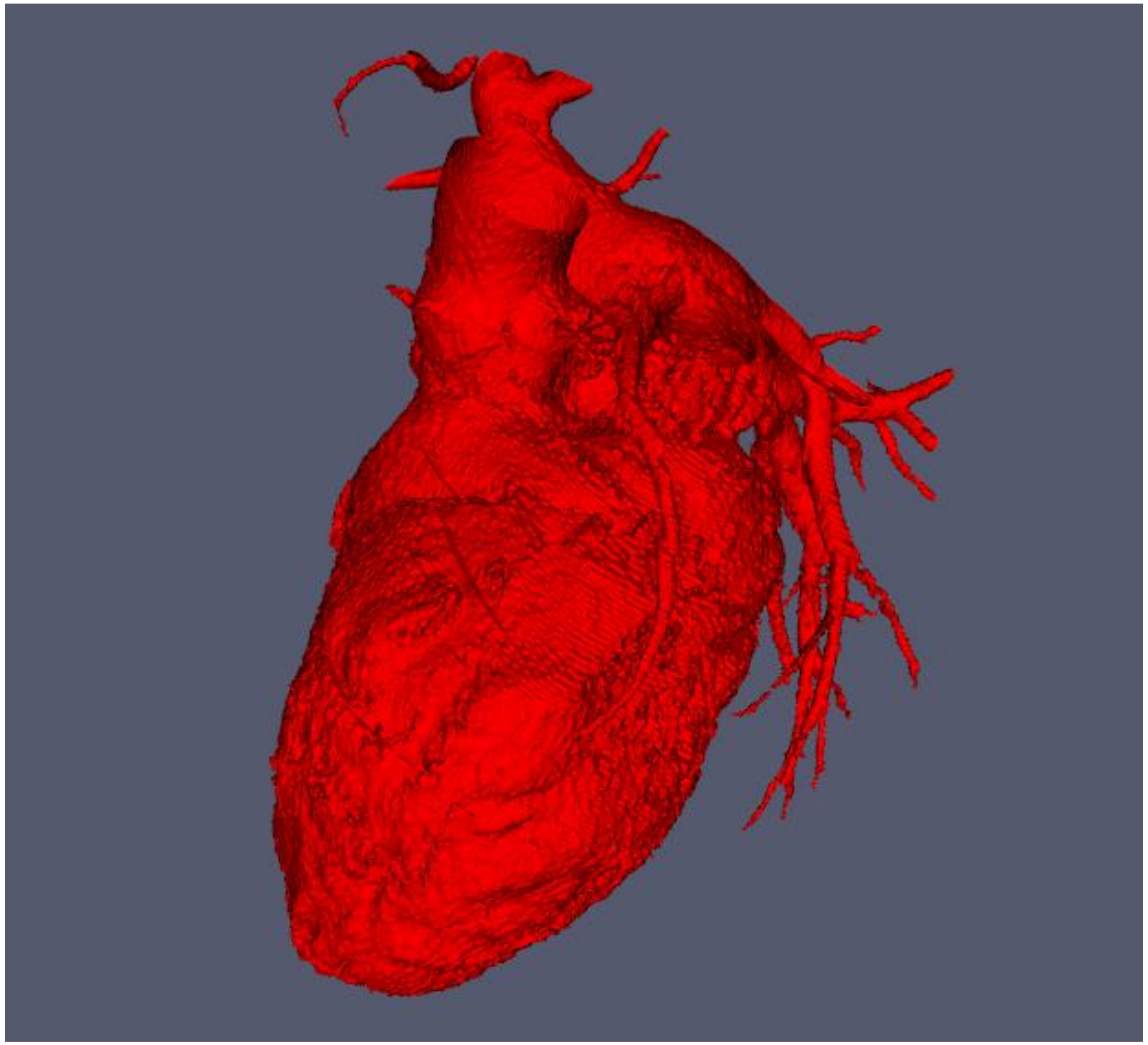
K-means Intuition

- Clustering of Objects in Images



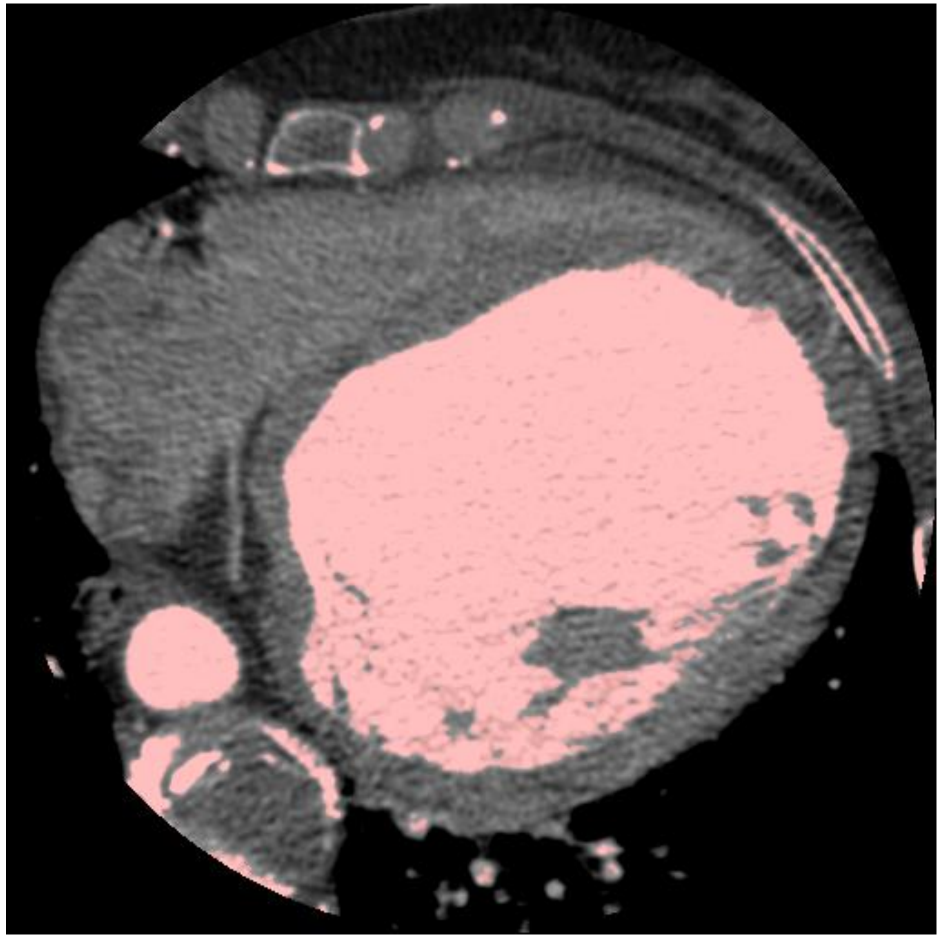
K-means Intuition

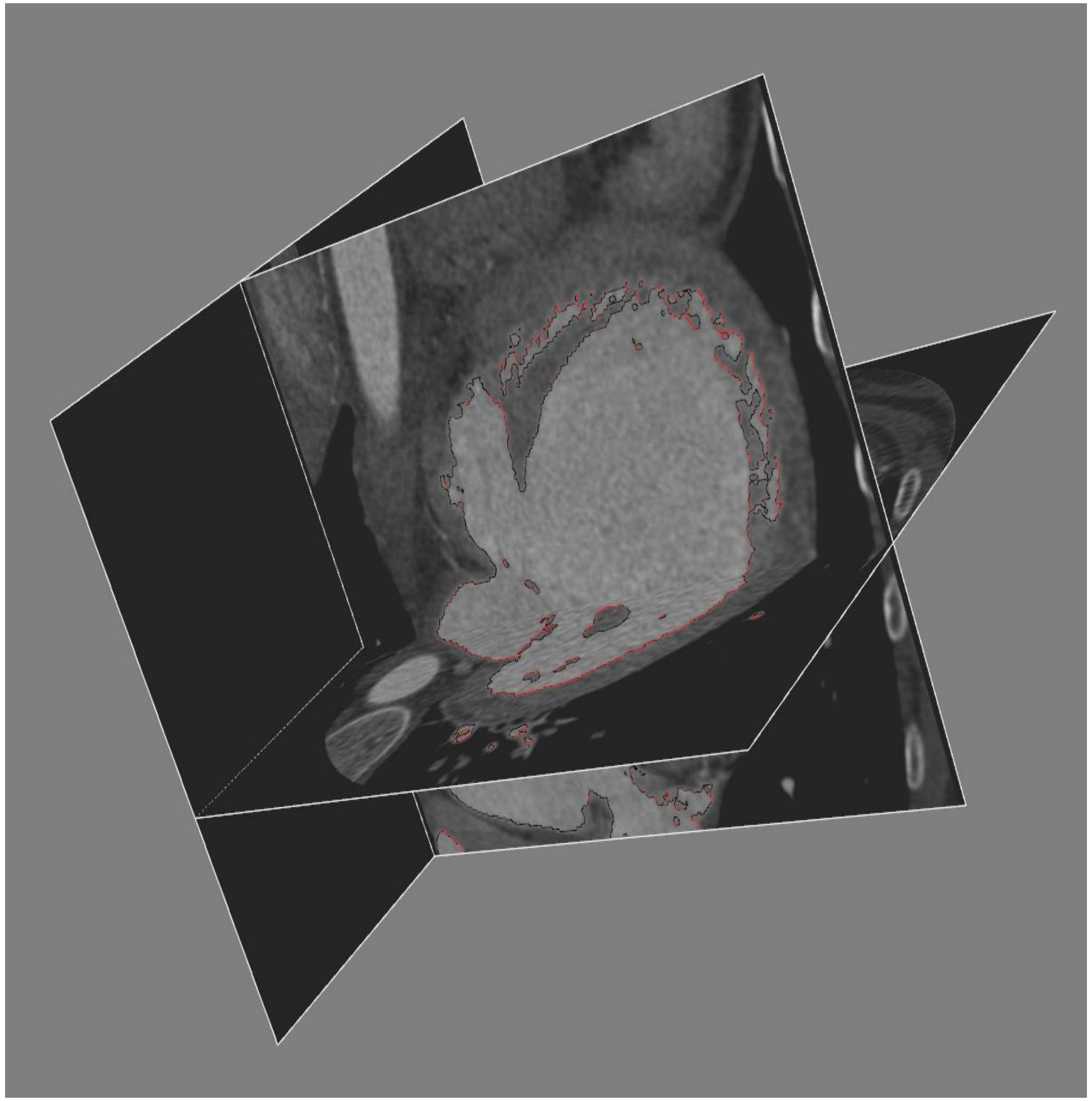
- Clustering of Objects in Images

K-means Intuition

- Clustering of Objects in Images





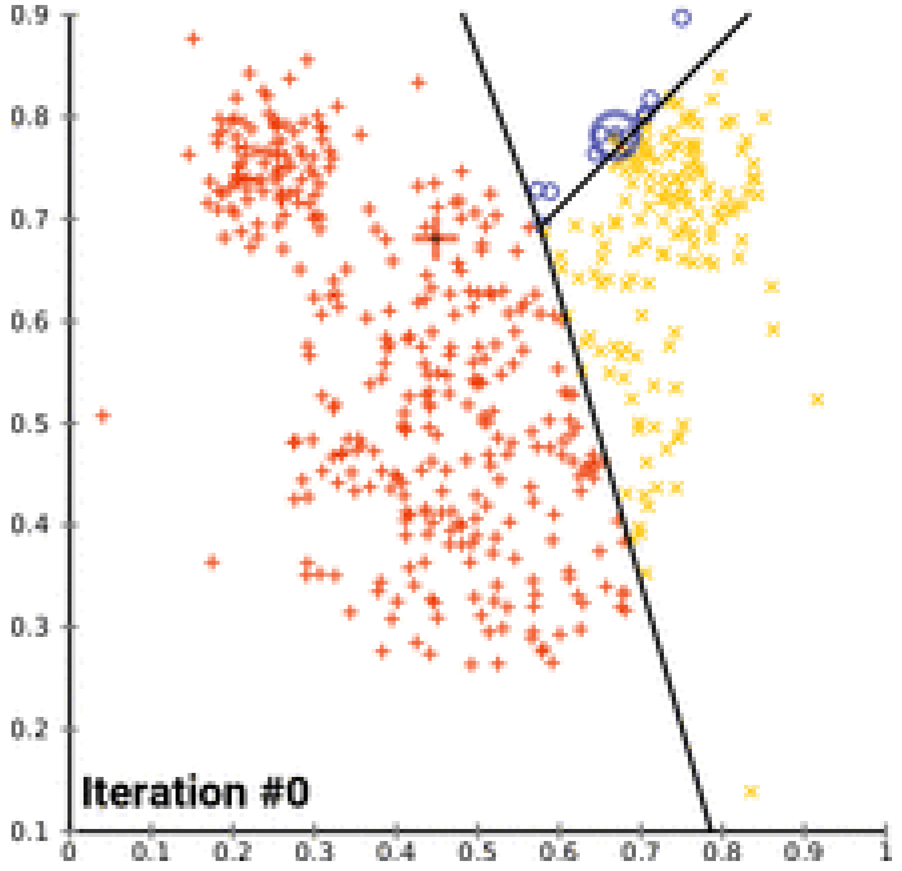
K-Means Formalized

K-means Formalized

- Let $\aleph$ be a space with some distance measure d
- We will consider $\aleph = \mathbb{R}^n$, $d(x, x') = \|x - x'\|$
- We consider a dataset of examples $D = \{x^{(1)}, \dots, x^{(m)}\}$
- We want to partition/cluster data D into k sets $C_1, \dots, C_k$
- We define the **centroid** of C_i to be: $\mu_i = \mu(C_i) = argmin_{\mu \in \mathbb{R}^n}\{\sum_{x \in c_i} d(x, \mu)^2\}$
- Of note, for the Euclidean distance on $\mathbb{R}^n$, $\mu(C_i)$ is the mean of C_i (Prove it!)
- The K-means objective functions:
    - $J_{Kmean}(C_1, \dots, C_k) = \sum_{i=1}^{k} \sum_{x \in C_i} d\big(x, \mu(C_i)\big)^2$

K-means Formalized

- Input $D = \{x^{(1)}, \dots, x^{(m)}\} \in \mathbb{R}^n$
- Initialize: (randomly) choose the initial centroids:
  - $\mu_1, \dots, \mu_k \in D$
- Repeat until convergence the following steps:
  - $\forall i, C_i = \{x \in D, i = argmin_j d(x, \mu_j)\}$
  - $\forall i, \mu_i = argmin_{\mu \in \mathbb{R}^n} \sum_{x \in C_i} d(x, \mu)^2$
- For the Euclidian distance i.e. $d(x, x') = \|x - x'\|$
  - $\forall i, C_i = \left\{x \in D, i = argmin_j \|x - \mu_j\|^2\right\}$
  - $\forall i, \mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$ (i.e. mean of C_i).

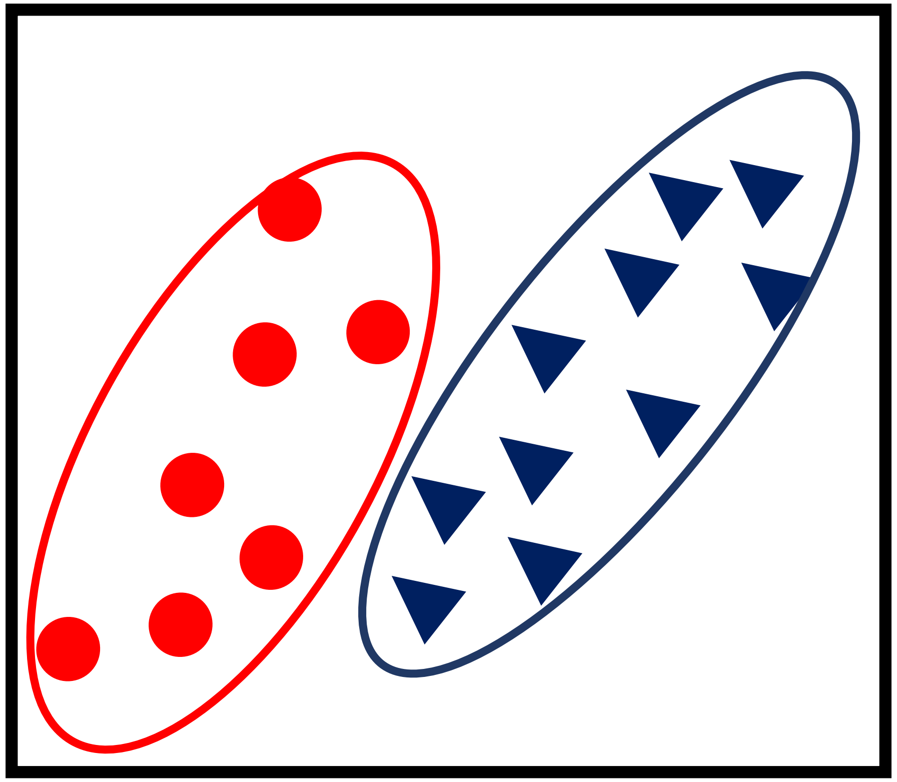


K-means Formalized

- Convergence
  - Repeat steps until convergence. Because each phase reduces the value of the objective function J, **convergence of the algorithm is assured**.
  - **However, it may converge to a local rather than global minimum** of J because J is not convex. [see Mac Queen 1967 for study of K-mean convergence].
  - General recommendation: re-run K-means with several random starting initial centroids.
- Can we use gradient descent to solve K-mean?
  - Yes but not the most popular method.
  - See the following paper for a theoretical and experimental analysis of the convergence of K-means using different optimization algorithms:
    - Bottou, Leon, and Yoshua Bengio. "Convergence properties of the k-means algorithms." Advances in neural information processing systems. 1995.

K-means Limitations

- Is it always the case that the assignment to the closest cluster in the L2 norm sense be the right solution (or even a solution)?
- How can we make the difference between a point that is borderline and one clearly within its cluster?
- We will look into GMM to provide a probabilistic treatment to our clustering problematic.
- A probabilistic formulation might bring with it numerous benefits.



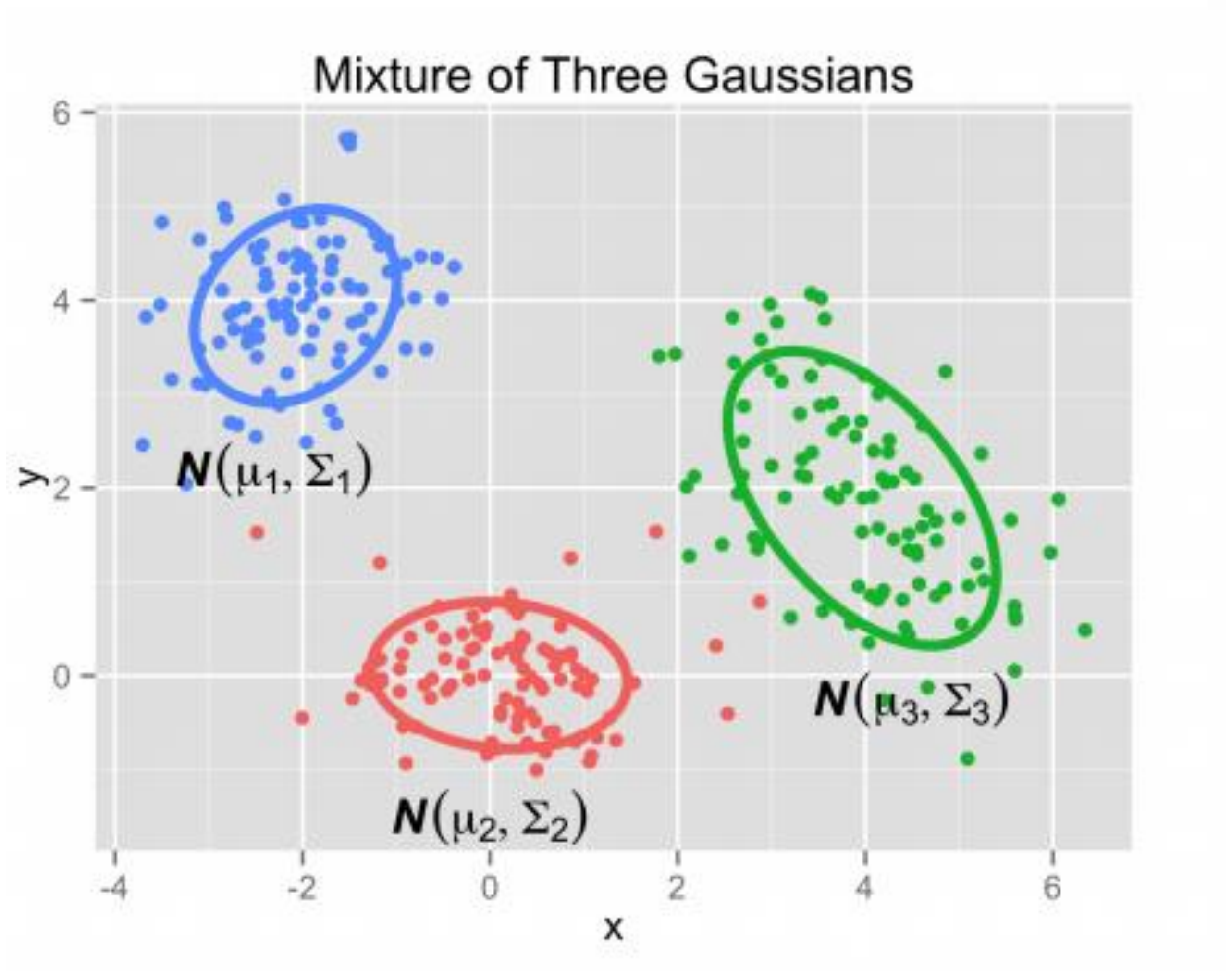
Gaussian Mixture Models (GMM)

Gaussian Mixture Model

- Probabilistic Model for Clustering
- Suppose:
  - There are k clusters
  - We have a probability density for each cluster
- The clustering algorithm:
  - Use the training data to fit the parameters of the model.
  - For each point, choose the cluster with the highest likelihood based on the model.

Gaussian Mixture Model

- GMM with $k = 3$

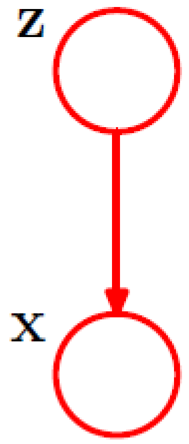


Adapted from: David Rosenberg, Brett Bernstein lecture notes. 2017

Gaussian Mixture Model

- Gaussian Mixture Model Parameters
- We assume k components
- We need to estimate the following parameters to create our model:
  - Cluster probabilities: $\pi = (\pi_1, \dots, \pi_k)$
  - Cluster means: $\mu = (\mu_1, \dots, \mu_k)$
  - Cluster covariance matrices: $\Sigma = (\Sigma_1, \dots, \Sigma_k)$
- $\pi_i$are also called the **mixing coefficients**. They can be seen as the prior probability of an observation belonging to a certain class.

Gaussian Mixture Model



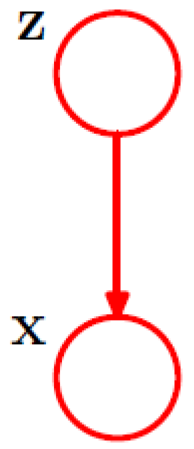
- We observe x and we do not know the true cluster assignment z.
- z is called a **hidden variable** or **latent variable**.
- Definition: A **latent variable model** is a probability model for which certain variables are never observed.
- The **GMM is a latent variable model**.
- We write the joint distribution:
  - $p(x,z) = p(z)p(x|z) = \pi_z \mathcal{N}(x|\mu_z, \Sigma_z)$
- In practice, we observe x. We want to know z.
- The conditional distribution of the cluster z given x is:
  - $p(z|x) = p(x,z)/p(x)$
- The conditional distribution is a **soft assignment.** How do we get a hard assignment?
  - $z^* = argmax_{z\epsilon\{1,\dots,k\}}\, p(z|x)$
  - So if we have the model, clustering is easy.

Adapted from: David Rosenberg, Brett Bernstein lecture notes. 2017

Gaussian Mixture Model

$p(x,z) = p(z)p(x|z)$
$= \pi_z \mathcal{N}(x|\mu_z, \Sigma_z)$

- The **marginal distribution** for a single observation x is:
  - $p(x) = \sum_{z=1}^{k} p(x,z) = \sum_{z=1}^{k} \pi_z \, \mathcal{N}(x|\mu_z, \Sigma_z)$
- This is a mixture of distribution or **mixture model** i.e. we can write the probability density as a convex combination of probability densities.
- Given a set of examples$x^{(1)}, \dots, x^{(n)}$how do we estimate the GMM.
- In other words, how to we estimate the parameters
  - Cluster probabilities: $\pi = (\pi_1, \dots, \pi_k)$
  - Cluster means: $\mu = (\mu_1, \dots, \mu_k)$
  - Cluster covariance matrices: $\Sigma = (\Sigma_1, \dots, \Sigma_k)$
- We will use **maximum likelihood**. In other words, we look for finding the set of parameters value that give the observed data the highest likelihood. Formally, we want to maximize: $P(X|parameters)$.

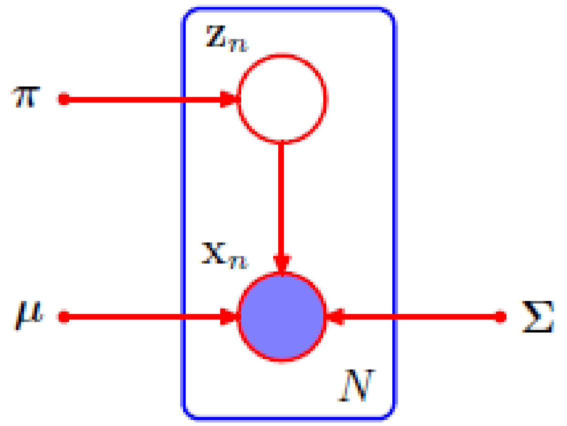


Adapted from: David Rosenberg, Brett Bernstein lecture notes. 2017

Learning in GMM

- Given $D = \{x^{(1)}, \dots, x^{(m)}\}$ we want to maximize:
  - $L(\pi, \mu, \Sigma) = \prod_{i=1}^{m} p\left(x^{(i)}\right) = \prod_{i=1}^{m} \sum_{z=1}^{k} \pi_z \mathcal{N}(x^{(i)} | \mu_z, \Sigma_z)$
- We take the objective function as being the log likelihood:
  - $J(\pi, \mu, \Sigma) = \sum_{i=1}^{m} \log\{\sum_{z=1}^{k} \pi_z \mathcal{N}(x^{(i)} | \mu_z, \Sigma_z)\}$
- For a single Gaussian it is easy because the log will cancel the exponential in the Gaussian density.
- However, for GMM with more than a single Gaussian, the sum inside the log prevents this cancellation. The result will be that there is no closed form solution to the MLE problem.
- So how do we solve it?

$$
\begin{aligned}
p(x) &= \sum_{z=1}^{k} p(x, z) \\
&= \sum_{z=1}^{k} \pi_z \mathcal{N}(x \mid \mu_z, \Sigma_z)
\end{aligned}
$$



Learning in GMM

$$J(\pi,\mu,\Sigma) = \sum_{i=1}^{m} \log\left\{\sum_{z=1}^{k} \pi_z \mathcal{N}(x^{(i)}|\mu_z,\Sigma_z)\right\}$$

- Using gradient descent or SGD?
- It can be done but necessitate the sigma matrices to be positive semidefinite.
- Gradient descent is usually not the way to go in GMM.
- Instead we prefer the **expectation maximization (EM) framework**.
- Let's make derivatives with respect to the mean, covariance matrices and the mixing coefficients:
  - $\mu_c^{new} = \frac{1}{n_c}\sum_{i=1}^{m} \gamma_i^c x^{(i)}$
  - $\Sigma_c^{new} = \frac{1}{n_c}\sum_{i=1}^{m} \gamma_i^c (x^{(i)} - \mu_c^{new})(x^{(i)} - \mu_c^{new})^T$
  - $\pi_c^{new} = \frac{n_c}{n}$
  - Where we write the **responsibilities**:
    - $\gamma_i^j = \frac{\pi_j \mathcal{N}(x^{(i)}|\mu_j,\Sigma_j)}{\sum_{c=1}^{k} \pi_c \mathcal{N}(x^{(i)}|\mu_c,\Sigma_c)}$

Learning in GMM

$$J(\pi, \mu, \Sigma) = \sum_{i=1}^{m} \log \left\{ \sum_{z=1}^{k} \pi_z \, \mathcal{N}(x^{(i)} | \mu_z, \Sigma_z) \right\}$$

- The responsibility that cluster j takes for observation x_i
  - $\gamma_i^j = P\left(Z = j | X = x^{(i)}\right) = p(Z = j, X = x^{(i)})/p(x)$

$$= \frac{\pi_j \, \mathcal{N}(x^{(i)} | \mu_j, \Sigma_j)}{\sum_{c=1}^{k} \pi_c \mathcal{N}(x^{(i)} | \mu_c, \Sigma_c)}$$

- So it is the probability of an observation to be associated with cluster j
- The vector $(\gamma_i^1, \dots, \gamma_i^k)$ is the soft assignment of $x^{(i)}$.

Learning in GMM

$$J(\pi,\mu,\Sigma) = \sum_{i=1}^{m} log\left\{\sum_{z=1}^{k} \pi_z \mathcal{N}(x^{(i)}|\mu_z,\Sigma_z)\right\}$$

- This is not a closed-form solution for the parameters we want to estimate. The covariance matrix depends on the other parameters in a complex non-linear manner.
- But it suggests an iterative scheme for solving it. This is done by estimating the responsibility while assuming that the parameters of the GMM are known (corresponding to their current estimate) and in a second step making a new estimate of the GMM parameters while assuming that the responsibility is known. These two steps are repeated until convergence. In other words:
  - If we know π_j, μ_j and σ_j for all j then we can compute the responsibility γ_i^j
  - If we know the soft assignments, then we can compute π_j, μ_j, σ_j for all j
  - We can repeat these two steps until convergence.

Learning in GMM

$$J(\pi, \mu, \Sigma) = \sum_{i=1}^{m} \log \left\{ \sum_{z=1}^{k} \pi_z \, \mathcal{N}(x^{(i)} | \mu_z, \Sigma_z) \right\}$$

- Expectation Maximization (EM) algorithm:
  - Initialize model parameters: μ_j, Σ_j, π_j
  - E-step: given the model parameters evaluate the responsibilities
    - $\gamma_i^j = \frac{\pi_j \, \mathcal{N}(x^{(i)} | \mu_j, \Sigma_j)}{\sum_{c=1}^{k} \pi_c \mathcal{N}(x^{(i)} | \mu_c, \Sigma_c)}$
  - M-step: update the model parameters using the responsibilities
    - $\mu_c^{new} = \frac{1}{n_c} \sum_{i=1}^{n} \gamma_i^c x^{(i)}$
    - $\Sigma_c^{new} = \frac{1}{n_c} \sum_{i=1}^{n} \gamma_i^c (x^{(i)} - \mu_c^{new})(x^{(i)} - \mu_c^{new})^T$
    - $\pi_c^{new} = \frac{n_c}{n}$
  - Repeat E-step and M-step until convergence.

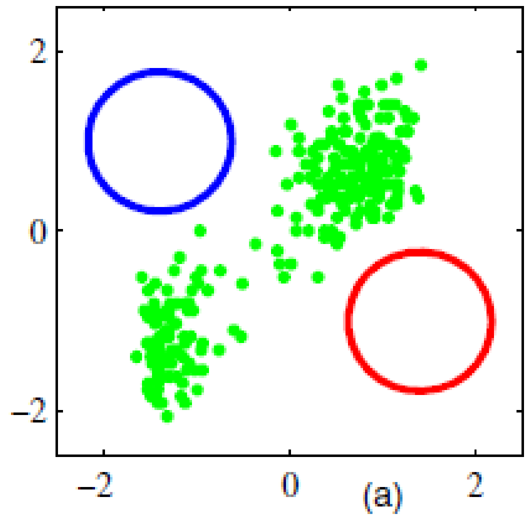


Figure 9.8 Illustration of the EM algorithm using the Old Faithful set as used for the illustration of the K-means algorithm in Figure 9.1. See the text for details.

Image: From Bishop's Pattern recognition and machine learning, Figure 9.8

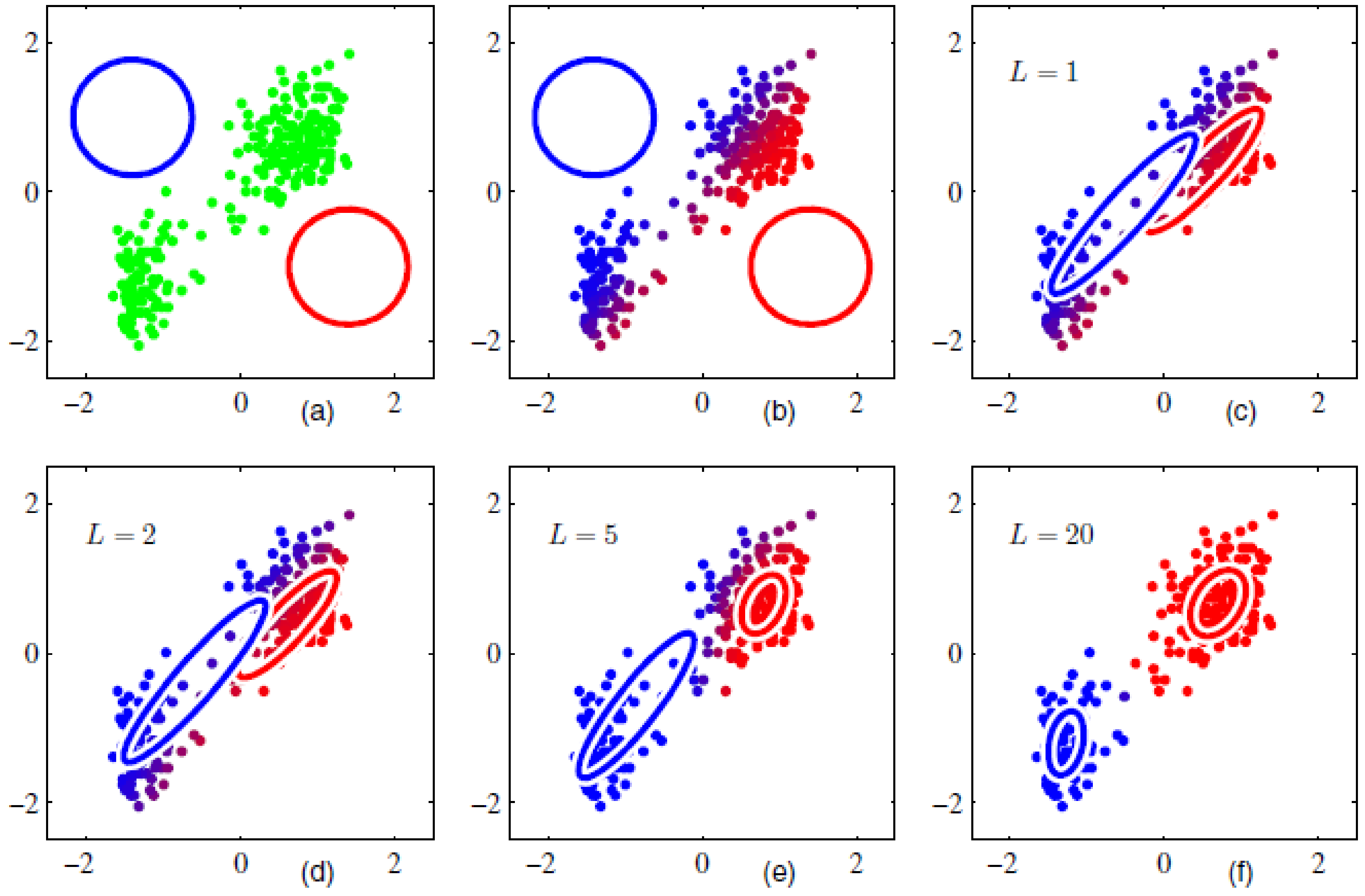


Figure 9.8 Illustration of the EM algorithm using the Old Faithful set as used for the illustration of the K-means algorithm in Figure 9.1. See the text for details.

Image: From Bishop's Pattern recognition and machine learning, Figure 9.8

Relation to K-Means

- EM for GMM seems similar to the way we solved for K-means
- GMM is equivalent to K-mean in the specific case where:
  - The cluster covariance matrices are $\sigma \cdot I$
  - With this the density for each Gaussian only depends on distance to the mean
  - If $\sigma \to 0$ the update equations converge to k-means. (Prove it!)
    - Soft assignment converge to hard assignments.

Adapted from: David Rosenberg, Brett Bernstein lecture notes. 2017

Take home

- Unsupervised Learning
  - Study patterns in the data
  - We do not have labels
- Data modelling
- Hard threshold (K-Means) versus soft threshold (GMM) algorithms
- K-means algorithm:
  - Non parametric
  - Lack of flexibility in cluster shape
  - Hard threshold: lack of probabilistic cluster assignment
- GMM:
  - Parametric
  - Soft threshold

References

[1] Pattern Recognition and Machine Learning. Springer 2006. Christopher M. Bishop. (Chapter 9)

[2] Behar Joachim, stage 2A, Ecole des Mines 2011.

[3] ML in Biomedical Engineering course note 336017

[4] Gaussian Mixture Models, David Rosenberg, Brett Bernstein. Lecture notes. New York University, 2017 https://davidrosenberg.github.io/mlcourse/Archive/2017/Lectures/13.lab.mixture-models.pdf