THE UNIVERSITY OF CHICAGO

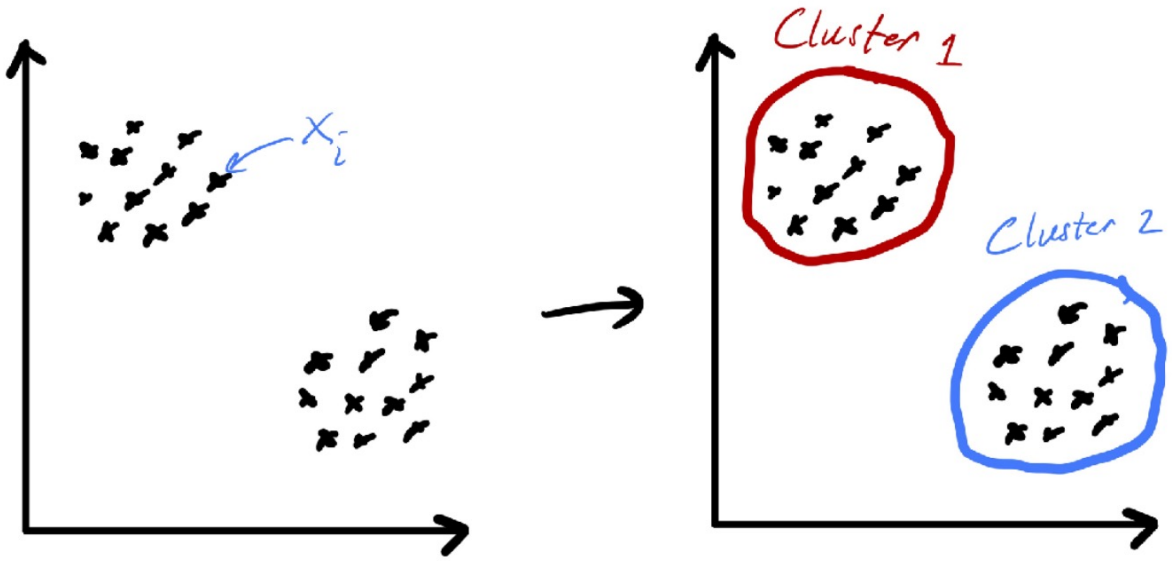
STAT 37710 / CMSC 35400 / CAAM 37710 Machine Learning

Clustering: K-means and Gaussian Mixture Model

Cong Ma

Clustering

It is an unsupervised learning procedure (i.e., applies to data without ground truth labels)



K-means algorithm

- Input {x1, x2, xn} in Rd
- Input K: number of clusters
- Expected output: K centers {c_i}, and K clusters {C_i}

- Question: given centers, how would you assign clusters?
- Question: given clusters, how would you determine centers?

K-means

1. Arbitrarily choose an initial k centers $\mathcal{C} = \{c_1, c_2, \cdots, c_k\}$.
2. For each $i \in \{1, \ldots, k\}$, set the cluster C_i to be the set of points in $\mathcal{X}$ that are closer to c_i than they are to c_j for all $j \neq i$.
3. For each $i \in \{1, \ldots, k\}$, set c_i to be the center of mass of all points in C_i: $c_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$.
4. Repeat Steps 2 and 3 until $\mathcal{C}$ no longer changes.

It is standard practice to choose the initial centers uniformly at random from $\mathcal{X}$. For Step 2, ties may be broken arbitrarily, as long as the method is consistent.

What is K-means doing?

For the **k-means** problem, we are given an integer k and a set of n data points $\mathcal{X} \subset \mathbb{R}^d$. We wish to choose k centers $\mathcal{C}$ so as to minimize the potential function,

$$\phi = \sum_{x \in \mathcal{X}} \min_{c \in \mathcal{C}} \|x - c\|^2.$$

- K-means can be viewed as an alternating minimization algorithm to solve this loss function

Drawback of original k-means

- Finding best clusters is NP-hard
- No theoretical guarantee for k-means
- Fix: k-means++

K-means++

We propose a specific way of choosing centers for the `k-means` algorithm. In particular, let $D(x)$ denote the shortest distance from a data point to the closest center we have already chosen. Then, we define the following algorithm, which we call `k-means++`.

1a. Take one center c_1, chosen uniformly at random from $\mathcal{X}$.
1b. Take a new center c_i, choosing $x \in \mathcal{X}$ with probability $\frac{D(x)^2}{\sum_{x \in \mathcal{X}} D(x)^2}$.
1c. Repeat Step 1b. until we have taken k centers altogether.
2-4. Proceed as with the standard `k-means` algorithm.

We call the weighting used in Step 1b simply "D^2 weighting".

Log(K) competitive ratio

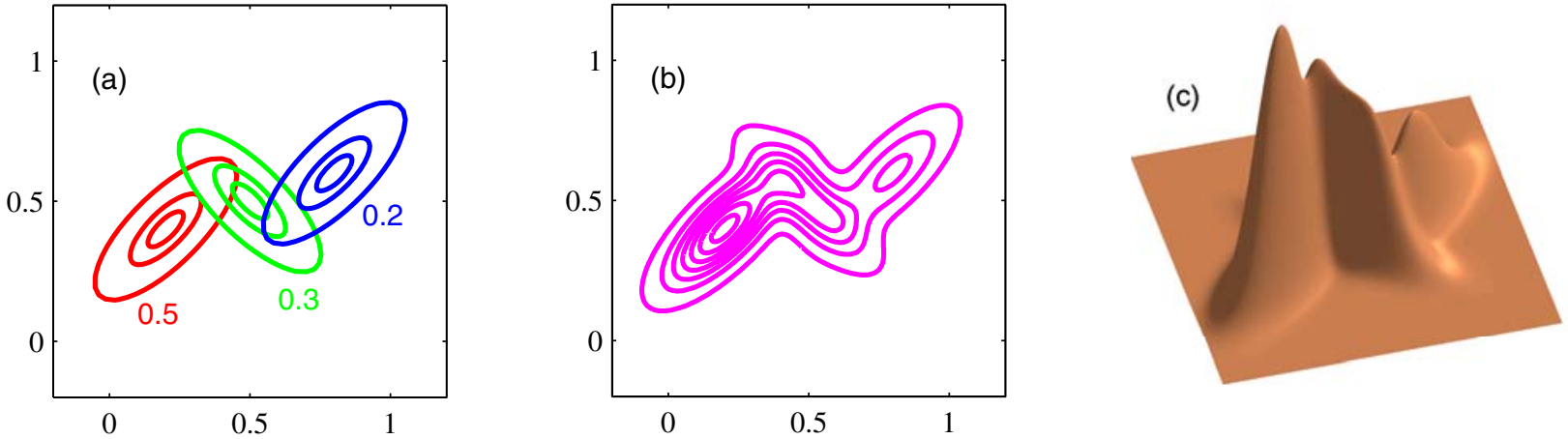
Theorem 3.1. *If $\mathcal{C}$ is constructed with* **k-means++**, *then the corresponding potential function ϕ satisfies, $E[\phi] \leq 8(\ln k + 2)\phi_{\mathrm{OPT}}$.*

Problems with k-means

Does not allow overlapped clusters

We need a probabilistic view on this

Gaussian mixture model



[Bishop]

- Definition: Gaussian mixture
  - Convex-combination of Gaussian distributions

$$P(\mathbf{x} \mid \theta) = P(\mathbf{x} \mid \mu, \Sigma, \mathbf{w}) = \sum_{i=1}^{k} w_i \mathcal{N}(\mathbf{x}; \mu_i, \Sigma_i)$$

where $w_i > 0$ and $\sum_i w_i = 1$.

Mixture modeling

- Model each cluster $j \in \{1, \ldots, k\}$ as a probability distribution

$$P(\mathbf{x} \mid \theta_j)$$

- Assume data $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ is sampled *i.i.d.* with likelihood

$$P(D \mid \theta) = \prod_{i=1}^{n} \sum_{j=1}^{k} w_j P(\mathbf{x}_i \mid \theta_j)$$

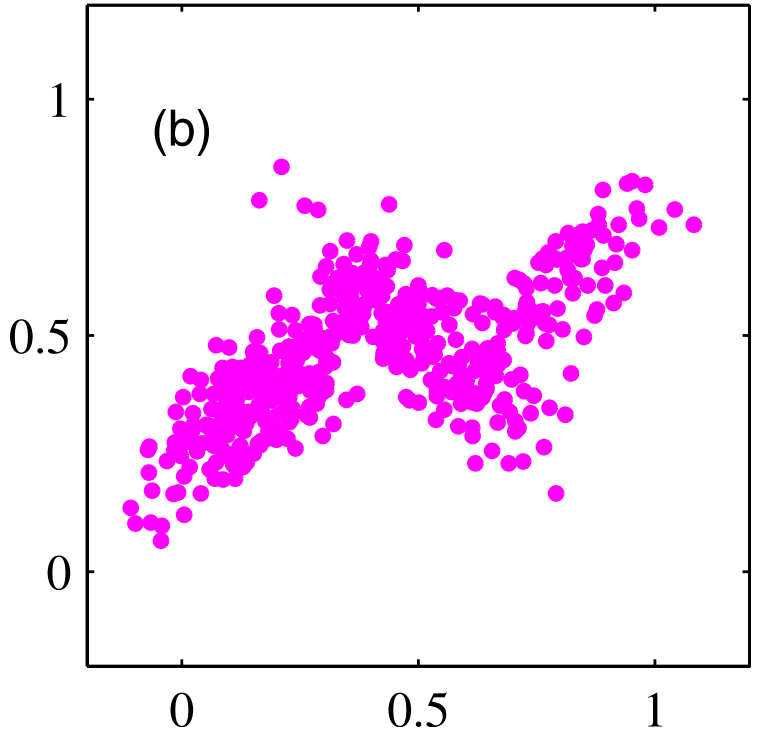
- Choose parameters to minimize negative log likelihood

$$L(D; \theta) = -\sum_{i=1}^{n} \log \sum_{j=1}^{k} w_j P(\mathbf{x}_i \mid \theta_j)$$

Fitting a mixture model

$$(\hat{\mu}, \hat{\Sigma}, \hat{\mathbf{w}}) = \arg\min - \sum_{i=1}^{n} \log P(\mathbf{x}_i \mid \mu, \Sigma, \mathbf{w})$$

$$= \arg\min - \sum_{i=1}^{n} \log \sum_{j=1}^{k} w_j \mathcal{N}(\mathbf{x}_i; \mu_j, \Sigma_j)$$



MLE for Gaussian mixture model (GMM)

$$L(\mu_{1:k}, \Sigma_{1:k}, w_{1:k}) = -\sum_{i=1}^{n} \log \sum_{j=1}^{k} w_j \mathcal{N}(\mathbf{x}_i; \mu_j, \Sigma_j)$$

- Non-convex
  - Gradient descent?

- Constrained optimization on weights and covariance matrices
  - weights must sum to 1
  - covariance matrices must remain symmetric positive definite

- Gradient-based approach not well suited for this problem

GMMs vs Gaussian Bayes classifiers

- Let z be cluster index, GMM: $P(z, \mathbf{x}) = w_z \mathcal{N}(\mathbf{x}; \mu_z, \Sigma_z)$
- In contrast to GBCs, in GMMs the (label) variable z is unobserved

Fitting a GMM = Training a GBC without labels

Clustering = latent variable modeling

- If we knew the labels z=y (i.e. GBC) → compute MLE in closed form!

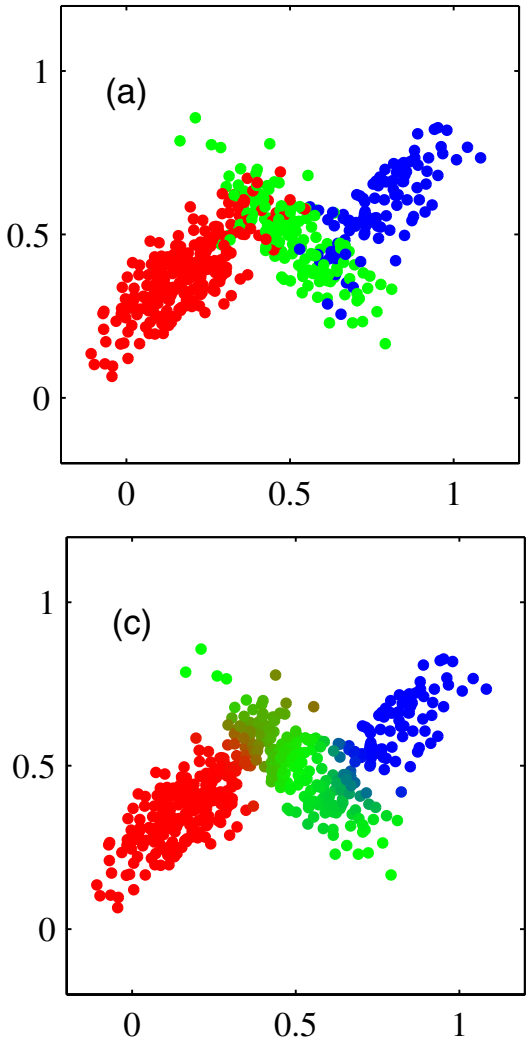
Responsibility of cluster z for x

- Given model

$$P(z \mid \theta), P(\mathbf{x} \mid z, \theta)$$

- For each **x**, compute posterior distribution over cluster membership:

$$\gamma_j(\mathbf{x}) := \underbrace{P(Z = j \mid \mathbf{x}, \mu, \Sigma, \mathbf{w}) = \frac{w_j P(\mathbf{x} \mid \mu_j, \Sigma_j)}{\sum_\ell w_\ell P(\mathbf{x} \mid \mu_\ell, \Sigma_\ell)}}_{P(z|\mathbf{x},\theta) = \frac{P(z|\theta)P(\mathbf{x}|z,\theta)}{\sum_z P(z|\theta)P(\mathbf{x}|z,\theta)}}$$



MLE for Gaussian mixture

- Can show that, at MLE: $(\hat{\mu}, \hat{\Sigma}, \hat{\mathbf{w}}) = \arg\min - \sum_i \log \sum_{j=1}^{k} w_j \mathcal{N}(\mathbf{x}_i; \mu_j, \Sigma_j)$

- It holds that

$$\hat{w}_j = \frac{1}{n} \sum_{i=1}^{n} \gamma_j(\mathbf{x}_i)$$

$$\hat{\mu}_j = \frac{\sum_{i=1}^{n} \gamma_j(\mathbf{x}_i)\mathbf{x}_i}{\sum_{i=1}^{n} \gamma_j(\mathbf{x}_i)}$$

$$\hat{\Sigma}_j = \frac{\sum_{i=1}^{n} \gamma_j(\mathbf{x}_i)(\mathbf{x}_i - \hat{\mu}_i)(\mathbf{x}_i - \hat{\mu}_j)^\top}{\sum_{i=1}^{n} \gamma_j(\mathbf{x}_i)}$$

[Cf. Bishop Eq 9.16]

- Equations are coupled -- Difficult to solve jointly

Expectation-Maximization (EM) for GMMs

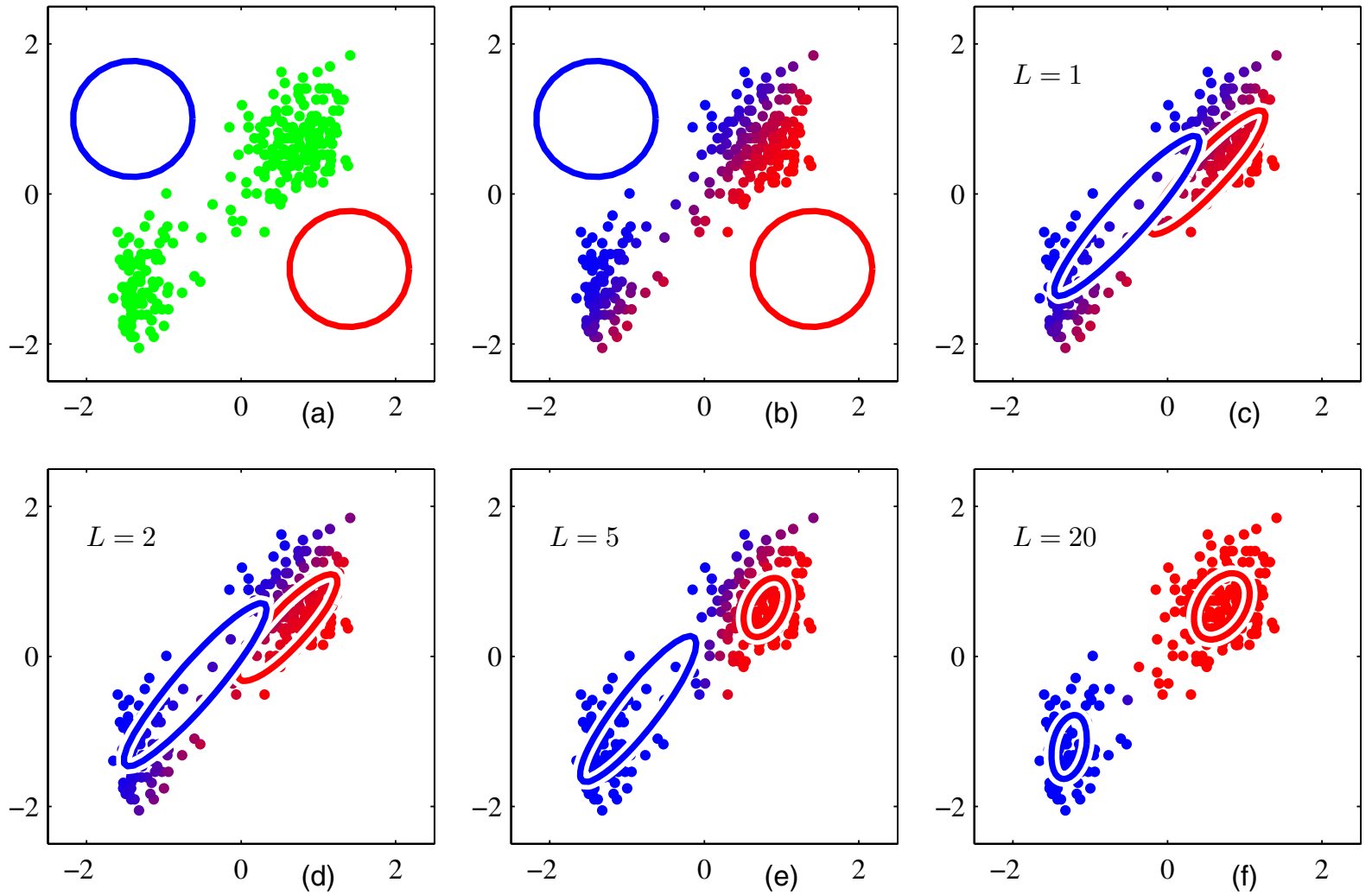
- Initialize parameters $w_{1:k}^{(0)}, \mu_{1:k}^{(0)}, \Sigma_{1:k}^{(0)}$
- While not converged:
  - E-step: Calculate cluster membership weights for each point

$$\gamma_j^{(t)}(\mathbf{x}_i) \leftarrow \frac{w_j P(\mathbf{x}_i \mid \mu_j^{(t-1)}, \Sigma_j^{(t-1)})}{\sum_\ell w_\ell P(\mathbf{x}_i \mid \mu_\ell^{(t-1)}, \Sigma_\ell^{(t-1)})}$$

  - M-step: Fit clusters to weighted data points (closed-form)

$$w_j^{(t)} \leftarrow \frac{1}{n}\sum_{i=1}^{n} \gamma_j^{(t)}(\mathbf{x}_i); \quad \mu_j^{(t)} \leftarrow \frac{\sum_{i=1}^{n} \gamma_j^{(t)}(\mathbf{x}_i)\mathbf{x}_i}{\sum_{i=1}^{n} \gamma_j^{(t)}(\mathbf{x}_i)}; \quad \Sigma_j^{(t)} \leftarrow \frac{\sum_{i=1}^{n} \gamma_j^{(t)}(\mathbf{x}_i)(\mathbf{x}_i - \mu_i^{(t)})(\mathbf{x}_i - \mu_j^{(t)})^\top}{\sum_{i=1}^{n} \gamma_j^{(t)}(\mathbf{x}_i)}$$

EM example



[Bishop]

The general EM algorithm

- EM algorithm is equivalent to the following procedure:

  - E-step: Calculate the Expected complete-data log likelihood (= function of θ)

$$Q(\theta; \theta^{(t-1)}) = \mathbb{E}_{Z_{1:n}} \left[\log P(\mathbf{x}_{1:n}, Z_{1:n} \mid \theta) \;\middle|\; \mathbf{x}_{1:n}, \theta^{(t-1)} \right]$$

  - M-step: Maximize

$$\theta^t = \arg\max_{\theta} Q(\theta; \theta^{(t-1)})$$

EM objective function (for GMMs)

$$
\begin{aligned}
Q(\theta; \theta^{(t-1)}) &= \mathbb{E}_{Z_{1:n}} \left[\log P(\mathbf{x}_{1:n}, Z_{1:n} \mid \theta) \mid \mathbf{x}_{1:n}, \theta^{(t-1)} \right] \\
&\overset{\text{i.i.d.}}{=} \mathbb{E}_{Z_{1:n}} \left[\sum_{i=1}^{n} \log P(\mathbf{x}_i, Z_i \mid \theta) \mid \mathbf{x}_{1:n}, \theta^{(t-1)} \right] \\
&= \sum_{i=1}^{n} \mathbb{E}_{Z_i} \left[\log P(\mathbf{x}_i, Z_i \mid \theta) \mid \mathbf{x}_i, \theta^{(t-1)} \right] \\
&= \sum_{i=1}^{n} \sum_{z=1}^{k} \underbrace{P(Z_i = z \mid \mathbf{x}_i, \theta^{(t-1)})}_{\gamma_z(\mathbf{x}_i)} \log \underbrace{P(\mathbf{x}_i, Z_i = z \mid \theta)}_{w_z \mathcal{N}(\mathbf{x}_i; \mu_z, \Sigma_z)} \\
&= \sum_{i=1}^{n} \sum_{z_i=1}^{k} \gamma_{z_i}(\mathbf{x}_i) \log P(\mathbf{x}_i, z_i \mid \theta)
\end{aligned}
$$

EM algorithm: E-step and M-step

- Objective $Q(\theta; \theta^{(t-1)}) = \sum_{i=1}^{n} \sum_{z_i=1}^{k} \gamma_{z_i}(\mathbf{x}_i) \log P(\mathbf{x}_i, z_i \mid \theta)$ $\gamma_z(\mathbf{x}_i) = P(z \mid \mathbf{x}_i, \theta^{(t-1)})$
  - E-step: compute $\gamma_z(\mathbf{x}_i)$ (expected sufficient statistics)
  - M-step: compute $\theta^{(t)} = \arg\max_{\theta} Q(\theta; \theta^{(t-1)})$ (MLE)

- Recall MLE in Gaussian Bayes classifiers:

$$\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{n} \log P(\mathbf{x}_i, z_i \mid \theta)$$

- M-step is equivalent to training a GBC with weighted data

Convergence of EM algorithm?

- EM Algorithm monotonically increases the (log) likelihood

$$\log P(\mathbf{x}_{1:n} \mid \theta^{(t)}) \geq \log P(\mathbf{x}_{1:n} \mid \theta^{(t-1)})$$

(proof not discussed here)

- Can we claim that EM will converge?
  - Not in the degenerate case when log likelihood is unbounded!

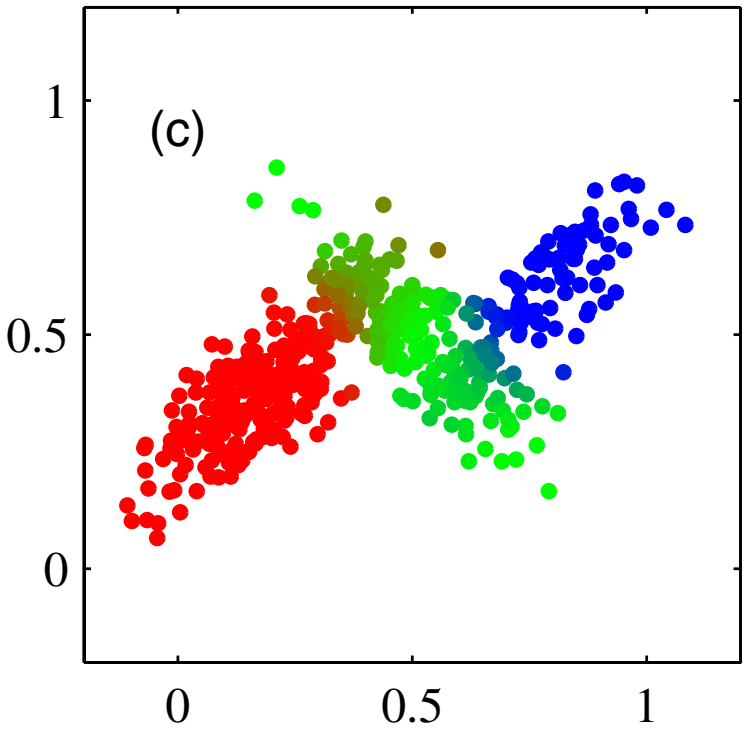
- Quality of solution highly depends on initialization
  - Common strategy: Rerun algorithm multiple times, and use the solution with largest likelihood

Use cases of mixture models

- Clustering/ unsupervised learning

- Can be used as part of more complex statistical models, e.g., for classification
  - or more general probabilistic models

- Can output likelihood P(**x**) of a point **x**, which is useful for anomaly/outlier detection



GMMs for density estimation

- We may be interested in fitting a Gaussian Mixture Model not for clustering but for density estimation

- Generative modeling of P(Y,X)
  - Model P(X) as Gaussian mixture
  - Model P(Y | X) using logistic regression, neural network, etc.
  - Combines the advantage of accurate predictions/robustness from discriminative model, with the ability to detect outliers!

References & acknowledgement

- C. Bishop (2006). “Pattern Recognition and Machine Learning”
  - Ch 2.3.9, “Mixtures of Gaussians”
  - Ch 9.2, “Mixture Models and EM: Mixtures of Gaussians”

- CMSC 35300. “Mathematical Foundations of Machine Learning” (UChicago, 2019)
  - Lecture 17 (clustering and k-means)

- A. Krause, “Introduction to Machine Learning” (ETH Zurich, 2019)
  - More demos and examples: Lecture on probabilistic modeling

- A. Moore, “Clustering with Gaussian Mixtures”
  - GMM use cases: https://www.cs.cmu.edu/~./awm/tutorials/gmm14.pdf