Computer Science

CSC380: Principles of Data Science

Clustering : Mixture Models

Kyoungseok Jang

Announcements

- HW6 will be posted tomorrow (April 19th)

- Students Course Survey
  - https://scsonline.oia.arizona.edu/index.php
  - Your opinions are anonymous, valued, and lead to course improvements.

Clustering

Data are assigned to clusters based on features like color and shape

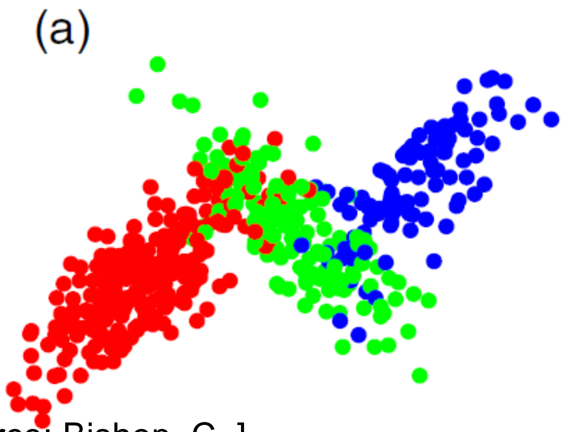
Hard Cluster Assignments

K-Means Assigns data to the cluster whose center is closest (in Euclidean distance)

$$z_n = \arg\min_j \left\| x_n - \mu_j \right\|^2$$

Cluster Center

Assignment **nth Data Point**



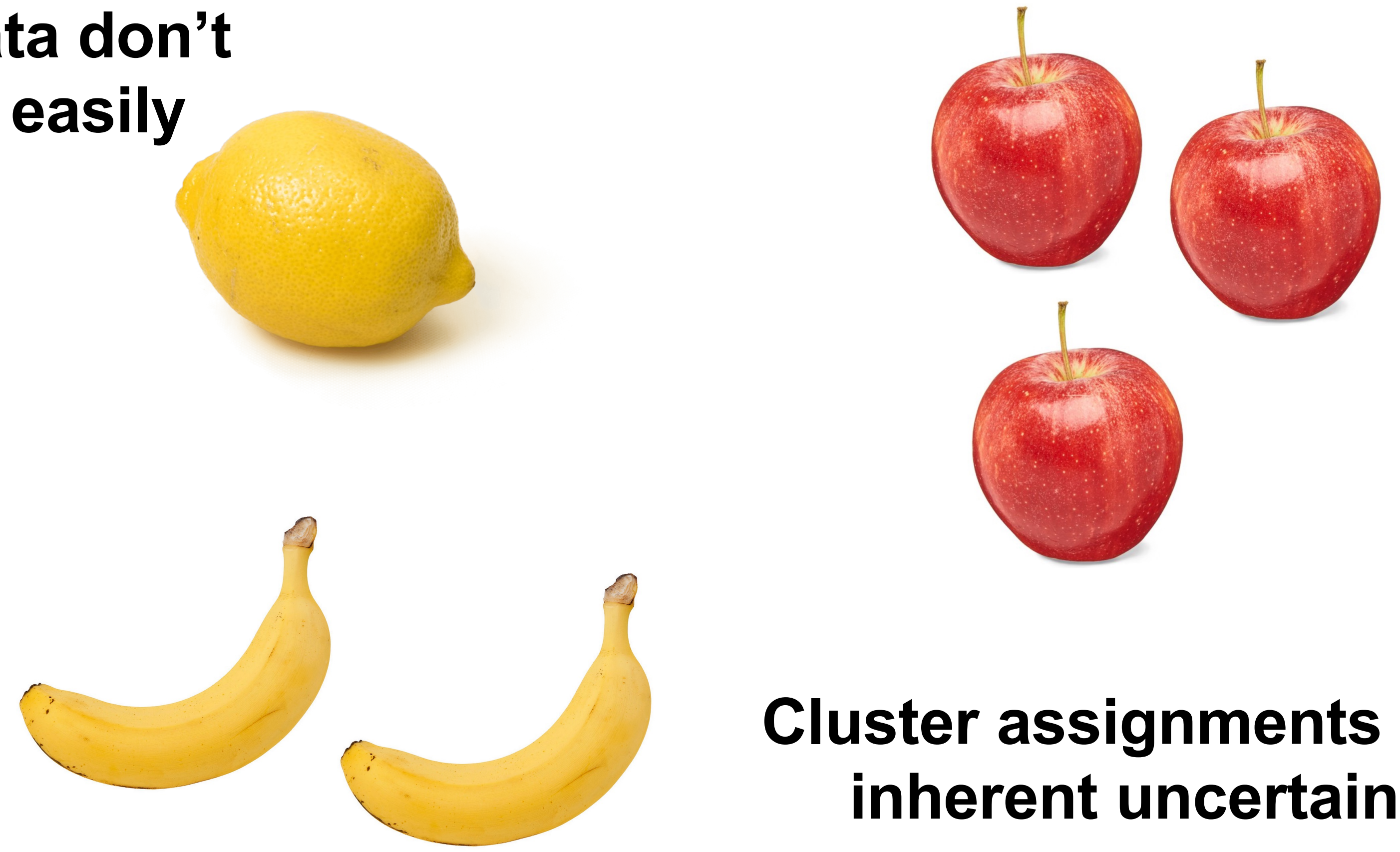
This is a *hard assignment* model

Each data is assigned to *exactly one* cluster

[Source: Bishop, C.]

Clustering

Some data don’t cluster easily



Cluster assignments have inherent uncertainty

Soft Cluster Assignments

Mixture Model Assignment is a *random variable* and learns a *posterior probability* over assignment

$$p(z_n \mid x_n)$$

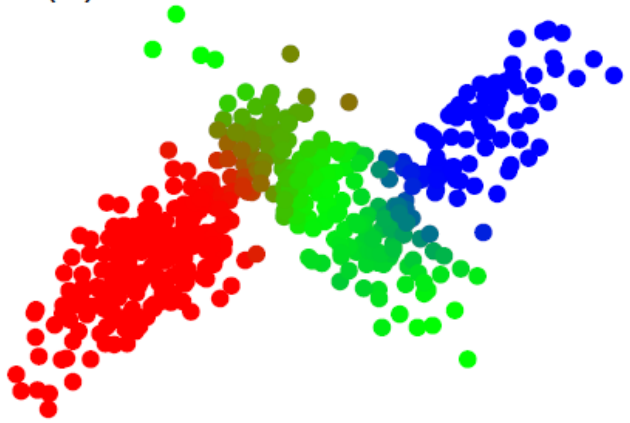
This is a *soft assignment* model

Data are assigned to *every cluster* with some probability

Predicted assignment generally,

$$\arg\max_k p(z_n = k \mid x_n)$$

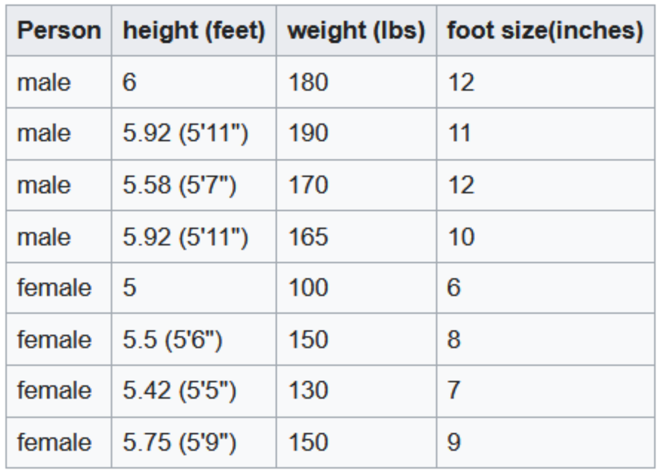
(c)



[Source: Bishop, C.]

Recall: Naïve Bayes Classifier

Training Data:



Person	height (feet)	weight (lbs)	foot size(inches)
male	6	180	12
male	5.92 (5'11")	190	11
male	5.58 (5'7")	170	12
male	5.92 (5'11")	165	10
female	5	100	6
female	5.5 (5'6")	150	8
female	5.42 (5'5")	130	7
female	5.75 (5'9")	150	9

Features

Task: Observe features $x_1, \dots, x_D$ and predict class label $y \in \{1, \dots, C\}$

Model: Assume that the feature x and its label y follows certain type of distribution $\mathcal{D}$ with parameter θ.

$$(x, y) \overset{\text{i.i.d.}}{\sim} \mathcal{D}_\theta$$

Training Algorithm: Estimate θ e.g., MLE $\hat{\theta}$

To classify: Compute

$$\hat{y} = \arg \max_{c \in \{1,\dots,C\}} p(y = c \mid x; \hat{\theta})$$

$$= \arg\max_{c \in \{1,\dots,C\}} p(y = c, x; \hat{\theta})$$

what comes after semicolon is the parameter of the distribution

Recall: Naïve Bayes Classifier

- Ex) Classifier that predicts cancer using two features
  - Feature $x^{(i)}$: (Smoke?, Drug?) (D=2)
  - Label $y^{(i)}$: (Cancer?)
- Suppose you trained Bernoulli Naïve Bayes Classifier (all features are binary) with the binary label (C=2).
- You will get 5 parameters
  - C-1 + CD= 5.
  - Class prior parameter: C-1 = 1
  - Likelihood parameter: $C * D = 4$

Recall: Naïve Bayes Classifier

- After long math, suppose that we got the parameters like this
  - Class-prior parameter: $\phi = p(y=1) = 0.25$,
  - Likelihood parameters: $\theta = (\theta_{01}, \theta_{02}, \theta_{11}, \theta_{12})$
    - $\theta_{12} = p(x_2 = 1 | y = 1) = 0.9$, $\theta_{11} = p(x_1 = 1 | y = 1) = 0.8$
    - $\theta_{02} = p(x_2 = 1 | y = 0) = 0.2$, $\theta_{01} = p(x_1 = 1 | y = 0) = 0.3$
- Prediction: When input $\mathrm{x} = (x_1, x_2) = (1,1)$, your prediction is
- $\hat{y} = \arg\max_{c \in \{0,1\}} p(y = c, x; \phi, \theta) = \max(p(y=0, x; \phi, \theta), p(y=1, x; \phi, \theta))$
- $p(y=1, x; \phi, \theta) = p(y=1) p(x_1 = 1, x_2 = 1 | y = 1; \phi, \theta) = p(y = 1) p(x_1 = 1 \, | y = 1; \phi, \theta) p(x_2 = 1 \, | y = 1; \phi, \theta) = 0.25 * 0.9 * 0.8 = 0.18$
- $p(y=0, x; \phi, \theta) = (1 - 0.25) * 0.2 * 0.3 = 0.045$
- 0.18>0.045, so $\hat{y} = 1$

Generative Model vs Discriminative Model

- Consider the supervised learning: $\{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$
- Generative model (e.g., Naïve Bayes)
  - p(x,y) = p(x|y) p(y)
  - Model p(x|y) and p(y) separately.

Recall) Probablistic approach:
When x is the input feature vector,

$$\hat{y} = \arg \max_{c \in \{1,\ldots,C\}} p(y = c|x)$$
$$= \arg \max_{c \in \{1,\ldots,C\}} p(x, y = c)$$

- Discriminative model (e.g., linear regression, logistic regression, etc.)
  - p(x,y) = p(y|x) p(x)
  - You could model p(x), but it will not change anything!!
  - Thus, just model p(y|x)

Q: what did this look like for linear regression and logistic regression?
A: $y = w^\top x + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma^2) \Rightarrow y|x \sim \mathcal{N}(w^\top x, \sigma^2)$
$y = \sigma(w^\top x) + \epsilon,\ y|x \sim Ber(w^\top x)$

Note: SVM is neither generative or discriminative; it's not even a probabilistic model to begin with!

Mixture Model

- Generative model: Models joint distribution over *data* and unknown *assignment*

- *Wait, we do not observe labels (=assignments)!!*
    - *Let's create an (artificial) hidden assignment, then!*

- The unknown assignment called a *latent variable* ("latent" means it is not observable and must be inferred)

- Example of a *latent variable model*

Mixture Model

We take one step further and assume that p(z) is not known!

$$p(z, x) = p(z)p(x \mid z)$$

~~Prior~~ probability of assignment

Likelihood

~~Prior encodes our belief about the latent variable (assignment) *before* observing any data,~~

$$p(z = k) = w_k \qquad 0 \leq w_k \leq 1 \qquad \sum_{k=1}^{K} w_k = 1$$

Likelihood captures the probability of the data *given a cluster assignment*

Mixture Model

Recall that the *law of total probability* allows us to calculate the *marginal probability* of the data,

$$p(x) = \sum_{k=1}^{K} p(z=k) p(x \mid z=k)$$

$$= \sum_{k=1}^{K} w_k p(x \mid z=k)$$

w_k is also unknown!!

(recall we will maximize $\prod_{i=1}^{m} p(x^{(i)};\theta)$)

Component distributions $p(x|z)$ can be any distribution on the data that you like

Gaussian Mixture Model

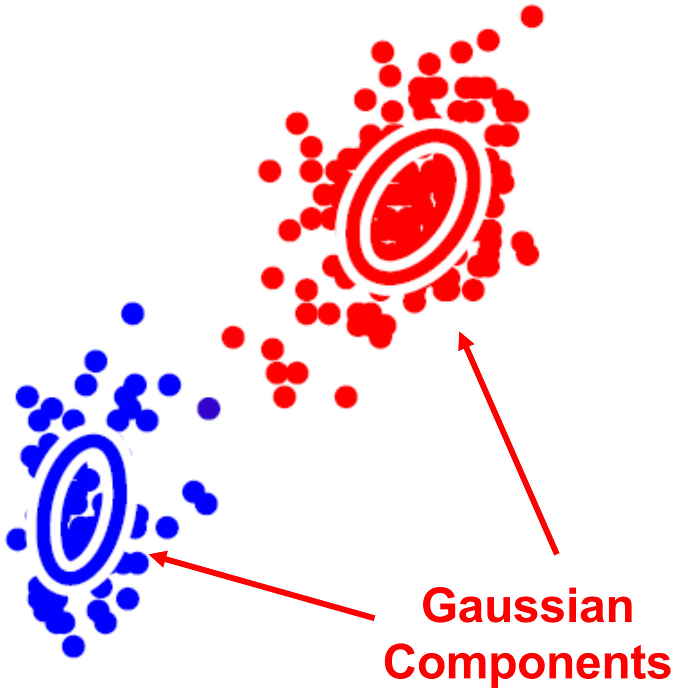
One of the most common mixtures are over Gaussians

$$p(x) = \sum_{k=1}^{K} w_k \mathcal{N}(x \mid m_k, \Sigma_k)$$

Unlike K-Means models correlation in clusters

New notations m_k and Σ_k for the mean and the covariance matrix of the Gaussian distribution – these are the key parameters!

Covariance: allows the distributions to take an ellipsoidal form!



[Source: Bishop, C.]

Soft GMM Assignments (Responsibilities)

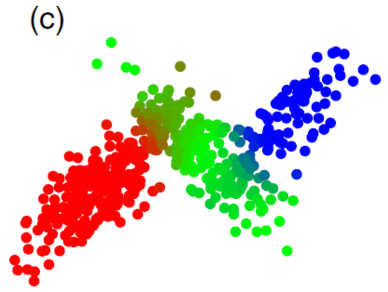
new notation:
z_n is n-th data point's cluster assignment
x_n is n-th data point's feature vector

Recall that by Bayes' rule we have the posterior,

$$p(z_n = k \mid x_n) = \frac{p(z_n = k)p(x_n \mid z_n = k)}{p(x_n)}$$

For Gaussian mixtures this is,

$$p(z_n = k \mid x_n) = \frac{w_k \mathcal{N}(x_n \mid m_k, \Sigma_k)}{\sum_{i=1}^{K} w_i \mathcal{N}(x_n \mid m_i, \Sigma_i)}$$



[Source: Bishop, C.]

In mixture modeling we call this the *responsibility*, since it is how *responsible* cluster k is for data point n

Concept Recap

- Mixture model is a weighted combination of component distributions,

$$p(x) = \sum_{k=1}^{K} w_k p(x \mid z = k)$$

- Bayes' rule gives the posterior probability of assignment (responsibility)

$$p(z_n = k \mid x_n) = \frac{p(z_n = k) p(x_n \mid z_n = k)}{p(x_n)}$$

- A GMM uses Gaussian component distributions with responsibilities:

$$p(z_n = k \mid x_n) = \frac{w_k \mathcal{N}(x_n \mid m_k, \Sigma_k)}{\sum_{i=1}^{K} w_i \mathcal{N}(x_n \mid m_i, \Sigma_i)}$$

All that is left is how to learn the model…

Learning Gaussian Mixture Models (GMMs)

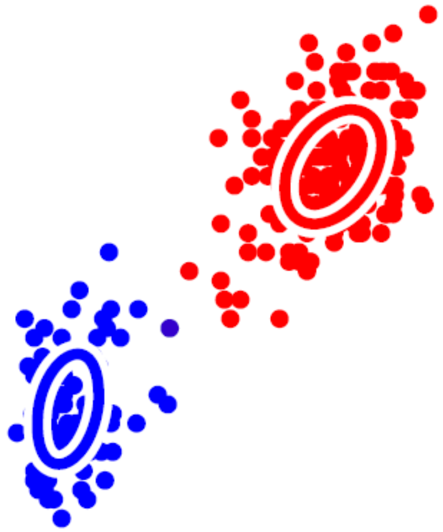
For D-dimensional X need to learn…

↓ the mixture weight

$$p(x) = \sum_{k=1}^{K} w_k \mathcal{N}(x \mid m_k, \Sigma_k)$$

D-dimensional vector of mean parameters

DxD Matrix of covariance parameters



[Source: Bishop, C.]

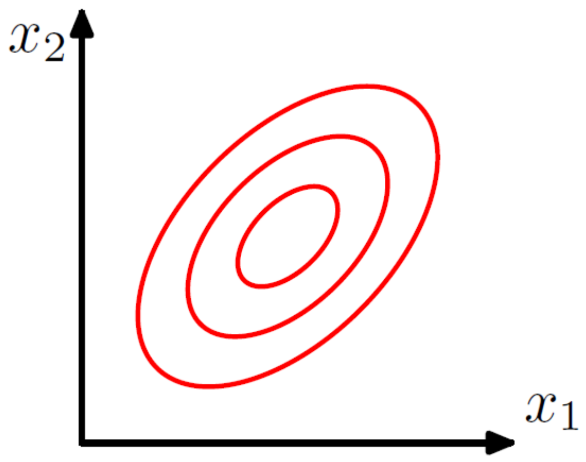
…for K components this requires learning

$$K + KD + KD^2 = O(KD^2)$$

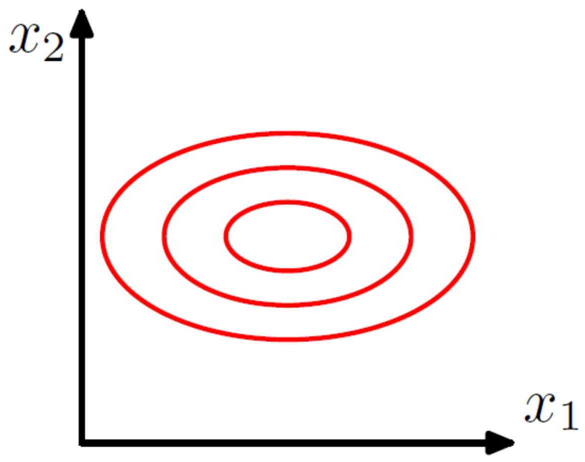
parameters

Covariance

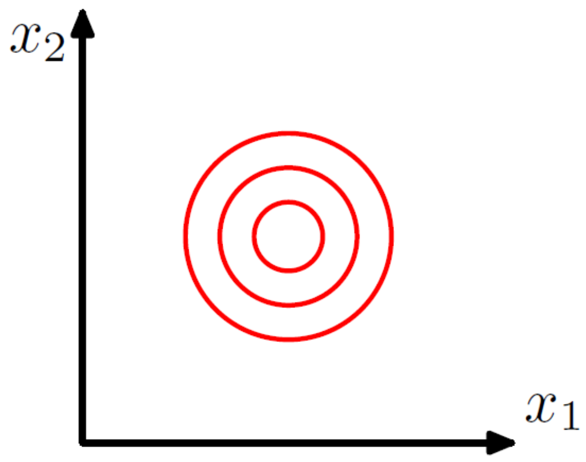
Captures correlation between random variables…can be viewed as set of ellipses…



Positive Correlation



Uncorrelated



Uncorrelated and same variance (isotropic / spherical)

Covariance Matrix

$$\Sigma = \mathrm{Cov}(X) = \begin{pmatrix} \sigma_{X_1}^2 & \rho\sigma_{X_1}\sigma_{X_2} \\ \rho\sigma_{X_1}\sigma_{X_2} & \sigma_{X_2}^2 \end{pmatrix}$$

4 elements, but represented with 3 variables.

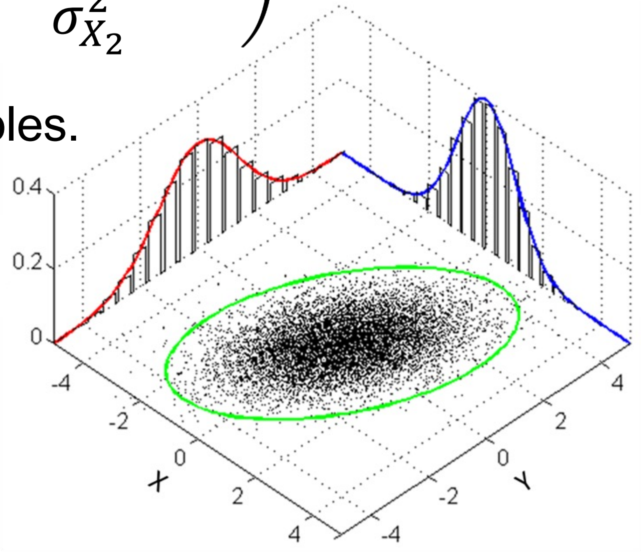
Covariance Matrix

Marginal variance of just the RV X_1

$$\Sigma = \mathrm{Cov}(X) = \begin{pmatrix} \sigma_{X_1}^2 & \rho\sigma_{X_1}\sigma_{X_2} \\ \rho\sigma_{X_1}\sigma_{X_2} & \sigma_{X_2}^2 \end{pmatrix}$$

4 elements, but represented with 3 variables.

i.e. How "spread out" is the distribution in the X_1 dimension…



Covariance Matrix

ρ: Correlation between X_1 and X_2

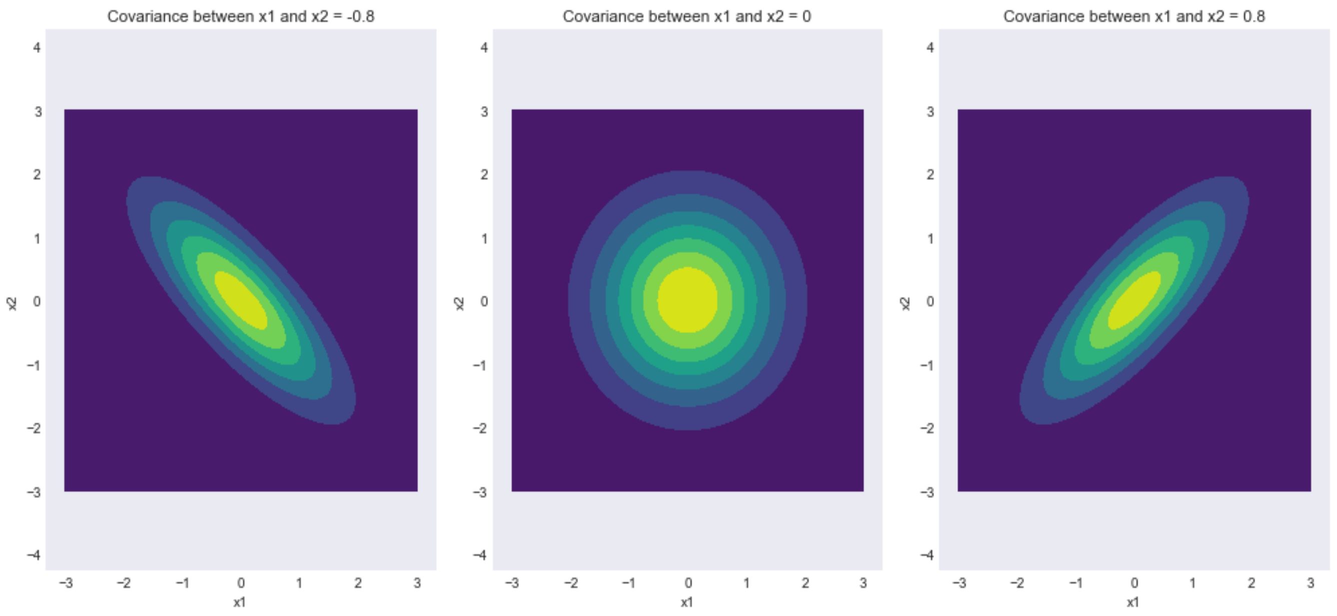
$$\Sigma = \mathrm{Cov}(X) = \begin{pmatrix} \sigma_{X_1}^2 & \boxed{\rho}\sigma_{X_1}\sigma_{X_2} \\ \boxed{\rho}\sigma_{X_1}\sigma_{X_2} & \sigma_{X_2}^2 \end{pmatrix}$$

Recall, correlation is given by:

$$\rho = \frac{\mathbf{Cov}(X_1, X_2)}{\sigma_{X_1}\sigma_{X_2}}$$

It captures *linear dependence of RVs*

$\rho = -0.8$ $\qquad$ $\rho = 0.0$ $\qquad$ $\rho = 0.8$



https://www.geeksforgeeks.org/visualizing-the-bivariate-gaussian-distribution-in-python/

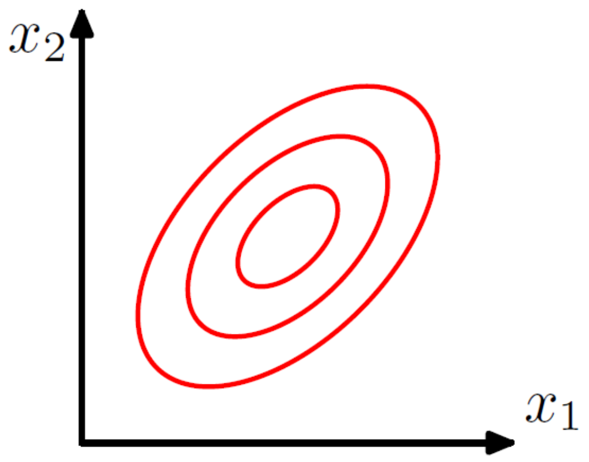
Example

- $Var(X_1) = Var(X_2) = 1$

Changing $Var(X_1)$ or $Var(X_2)$ will only stretch it horizontally or vertically (respectively)!

Covariance

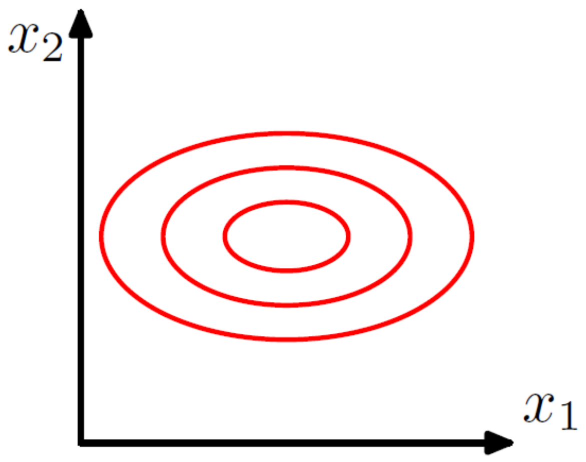
Captures correlation between random variables…can be viewed as set of ellipses…



Positive Correlation

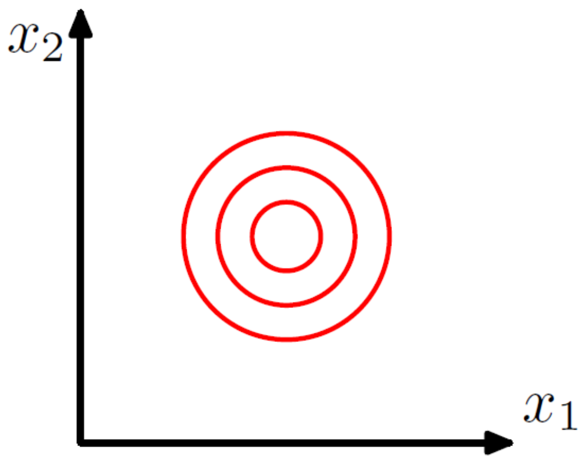
$\rho > 0$

Full matrix Σ



Uncorrelated

$$\Sigma = \begin{pmatrix} \sigma^2_{X_1} & 0 \\ 0 & \sigma^2_{X_2} \end{pmatrix}$$



Isotropic / Spherical

$$\Sigma = \begin{pmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{pmatrix} = \sigma^2 I$$

Learning the GMM

Need to learn …

mixture weight $w_1, \ldots, w_K \in \mathbb{R}$ such that $\sum_i w_i = 1$
the mean / variance parameters m_k and Σ_k for all $k = 1, \ldots, K$

Q: What method should we use to learn these?

A: Maximum likelihood estimation!

- Form log-likelihood over all data
- Find parameters that maximize log-likelihood

MLE for GMM

Recall that the likelihood of a *single data point* is given by,

$$p(x) = \sum_{k=1}^{K} w_k \mathcal{N}(x \mid m_k, \Sigma_k)$$

For N i.i.d. data points, the log-likelihood function is,

$$\mathcal{L}_N(w, m, \Sigma) = \sum_{n=1}^{N} \log \left\{ \sum_{k=1}^{K} w_k \mathcal{N}(x_n \mid m_k, \Sigma_k) \right\}$$

Turns out, it is highly non-convex and difficult to optimize…

Likelihood Lower Bound

Idea Form a lower bound of the non-convex log-likelihood with something that is easy to maximize,

$$\mathcal{L}_N(w, m, \Sigma) \geq \tilde{\mathcal{L}}_N(w, m, \Sigma)$$

True log-likelihood
Non-concave
Hard to maximize

Lower bound
Concave
Easy to maximize

We approximate maximum likelihood by optimizing the lower bound,

$$\max_{w,m,\Sigma} \mathcal{L}_N(w, m, \Sigma) \geq \max_{w,m,\Sigma} \tilde{\mathcal{L}}_N(w, m, \Sigma)$$

Expectation Maximization (EM)

Given a "guess" of the parameters $w^{old}, m^{old}, \Sigma^{old}$, we can compute the responsibilities

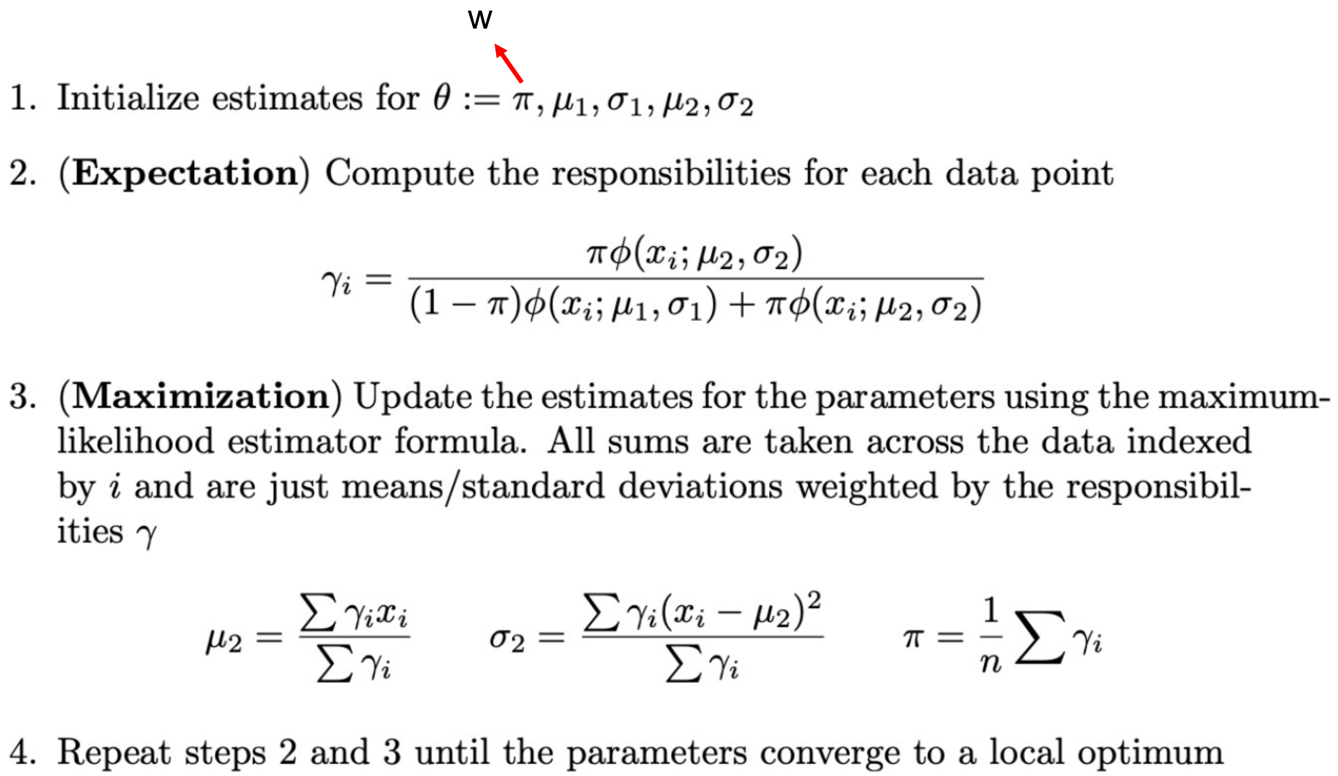
$$\gamma_{nk}^{old} \propto w_k^{old}\, p(x_n\,; m_k^{old}, \Sigma_k^{old}) \quad \textit{(normalize it over } k = 1, \dots, K)$$

Then, $$\tilde{\mathcal{L}}_N(w, m, \Sigma) = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk}^{old} \log \underbrace{p(x_n, z_n; w, m, \Sigma)}_{\text{Mixture model joint PDF as a function of parameters}}$$

- Lower bound $\mathcal{L} \geq \tilde{\mathcal{L}}$ is a result of Jensen's inequality (beyond scope)
- EM iteratively updates bound and finds new parameters with 2 steps
  - Expectation (**E-Step**) : Update responsibilities
  - Maximization (**M-Step**) : Maximize $\tilde{\mathcal{L}}$ to find new parameters

Expectation Maximization (EM)

- Let K=2, ϕ represents normal distribution.



1. Initialize estimates for $\theta := \overset{w}{\pi}, \mu_1, \sigma_1, \mu_2, \sigma_2$
2. (**Expectation**) Compute the responsibilities for each data point

$$\gamma_i = \frac{\pi\phi(x_i; \mu_2, \sigma_2)}{(1-\pi)\phi(x_i; \mu_1, \sigma_1) + \pi\phi(x_i; \mu_2, \sigma_2)}$$

3. (**Maximization**) Update the estimates for the parameters using the maximum-likelihood estimator formula. All sums are taken across the data indexed by i and are just means/standard deviations weighted by the responsibilities γ

$$\mu_2 = \frac{\sum \gamma_i x_i}{\sum \gamma_i} \qquad \sigma_2 = \frac{\sum \gamma_i (x_i - \mu_2)^2}{\sum \gamma_i} \qquad \pi = \frac{1}{n} \sum \gamma_i$$

4. Repeat steps 2 and 3 until the parameters converge to a local optimum

EM algorithm for GMM

↓ initialize just like k-means

- Initialize: $w \in \Delta^K$, $\{\mu_k \in \mathbb{R}^d, \Sigma_k \in \mathbb{R}^{d\times d}\}_{k=1}^K$

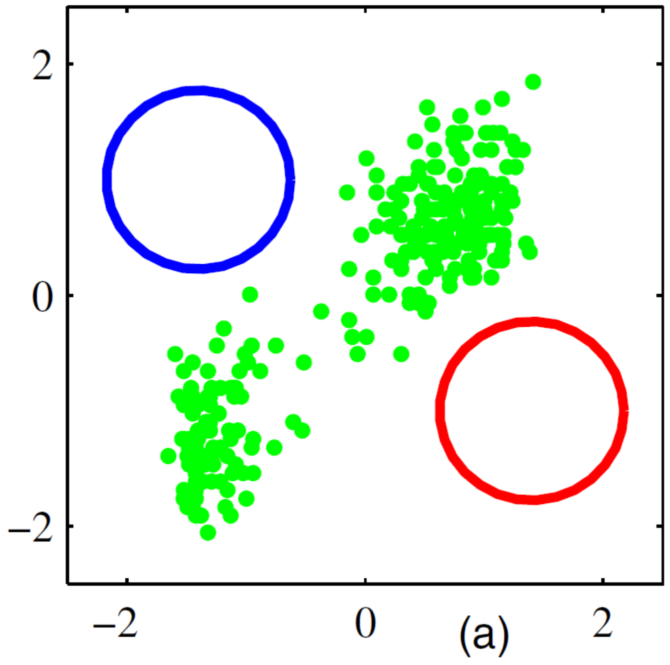
↑ usually, uniform weight.

↑ can set it to be a diagonal matrix that has c in the diagonal (e.g., $c = 1$)

- (E)xpectation step:
  - $\gamma_{nk} = \frac{w_k\, p(x_n \mid z_n = k)}{\sum_{k'=1}^K w_{k'}\, p(x_n \mid z_n = k')}$ responsibility
  - Let $N_k = \sum_{n=1}^N \gamma_{nk}$ soft counts
- (M)aximization step:
  - $\mu_k' = \frac{1}{N_k}\sum_{n=1}^N \gamma_{nk} x_n$
  - $\Sigma_k' = \frac{1}{N_k}\sum_{n=1}^N \gamma_{nk}(x_n - \mu_k')(x_n - \mu_k')^\top$ note we use μ_k' rather than μ_k
  - $w_k' = \frac{n_k}{n}$
  - Set $\mu_k \leftarrow \mu_k'$, $\Sigma_k \leftarrow \Sigma_k'$, $w_k \leftarrow w_k'$
- Stop when: the log likelihood does not increase much or the parameters do not change much.

EM for GMM

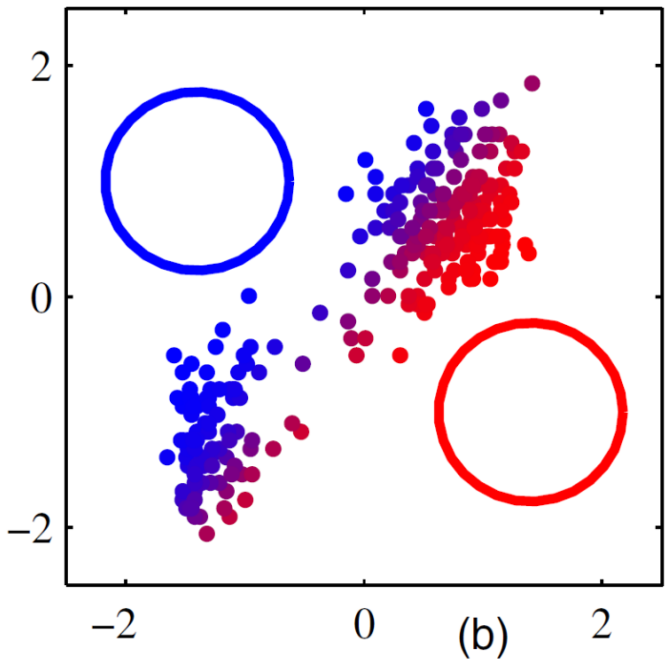
Initialize Cluster parameters



[Source: Bishop, C.]

EM for GMM

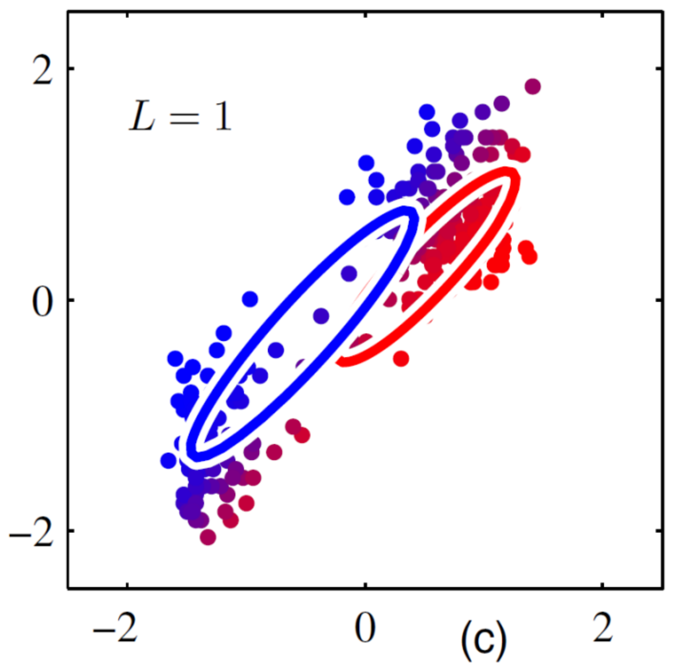
E-Step Compute responsibilities



[Source: Bishop, C.]

EM for GMM

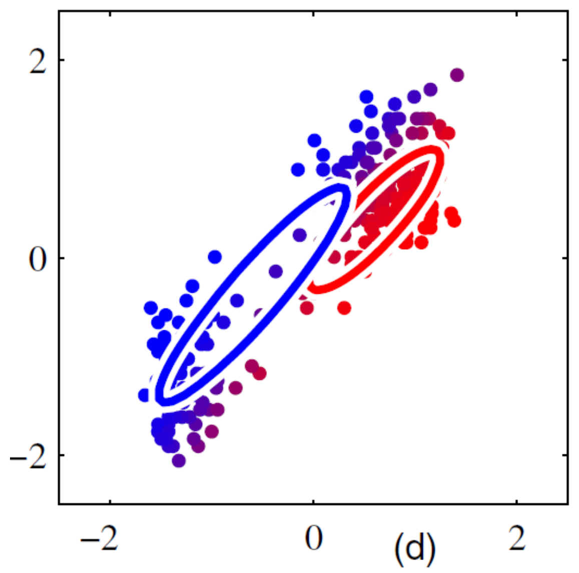
M-Step Maximize lower bound to find new component parameters



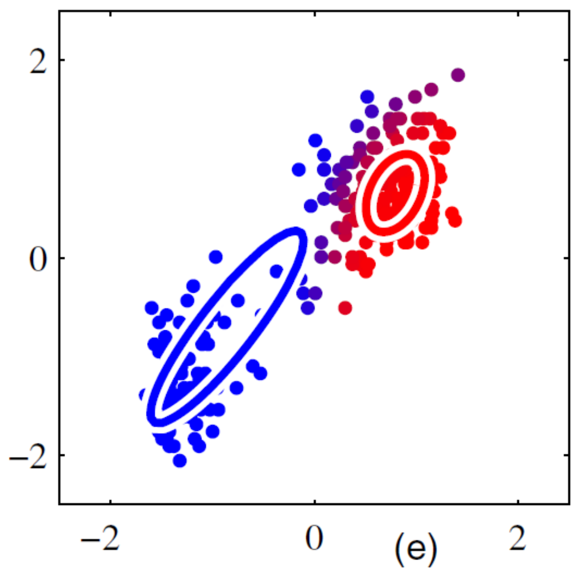
[Source: Bishop, C.]

EM for GMM

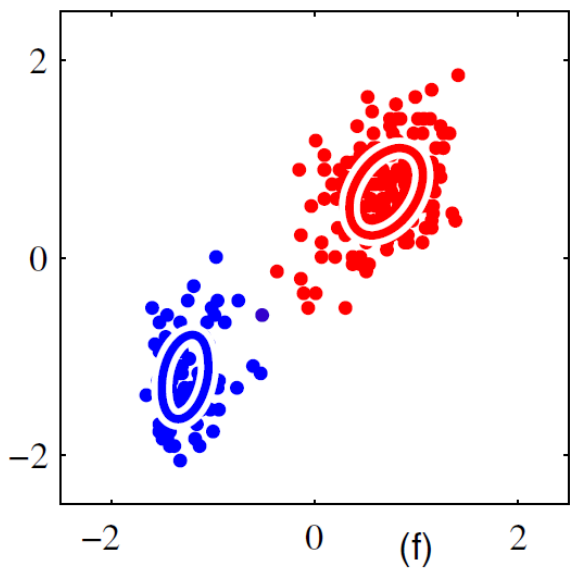
2 Iterations



5 Iterations

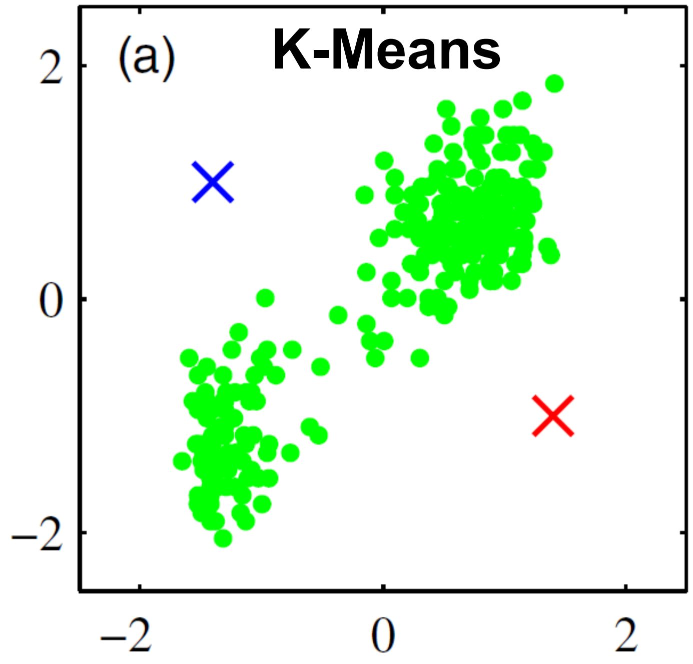


20 Iterations

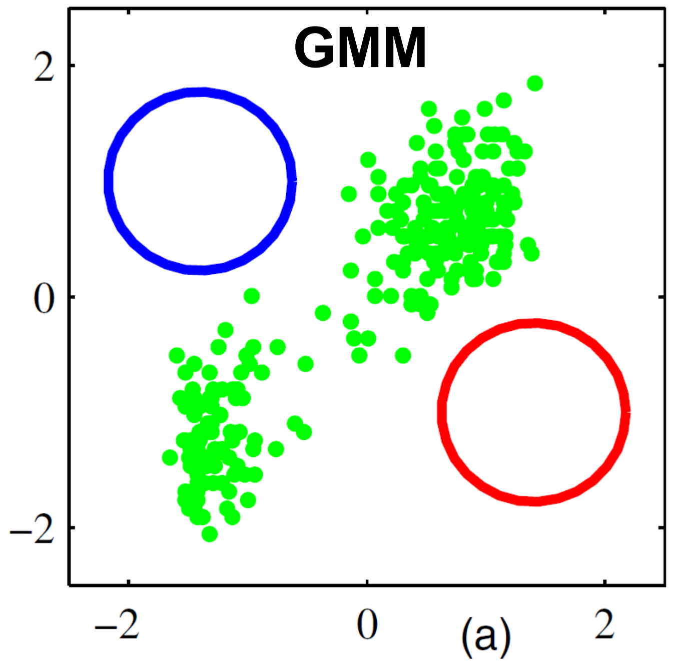


[Source: Bishop, C.]

Comparison to K-Means

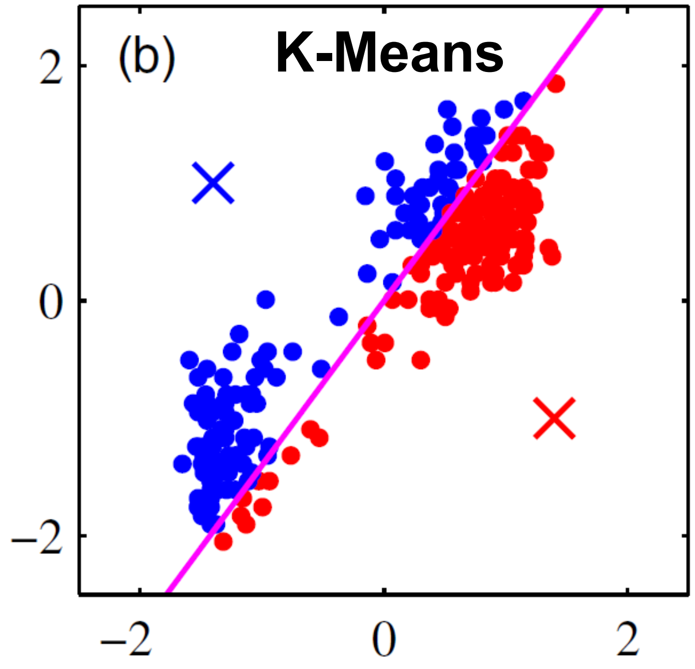


Initialize cluster centers

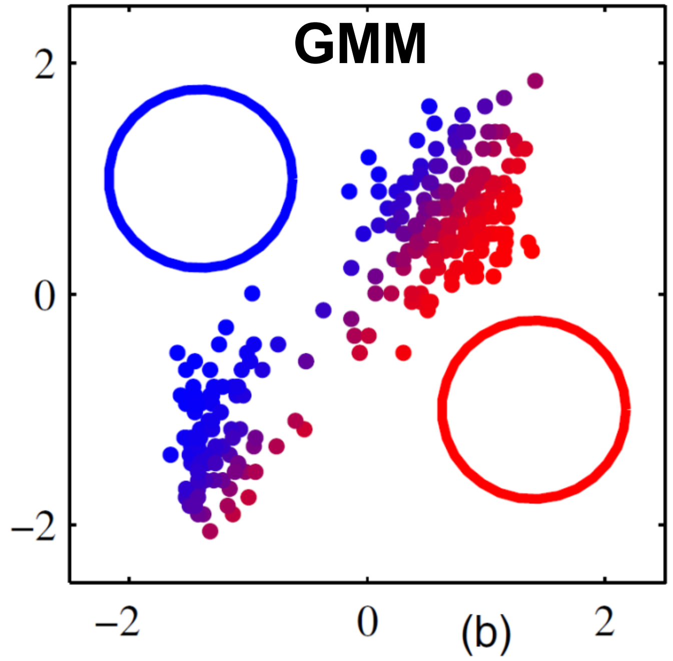


Initialize cluster mean / covariance

Comparison to K-Means

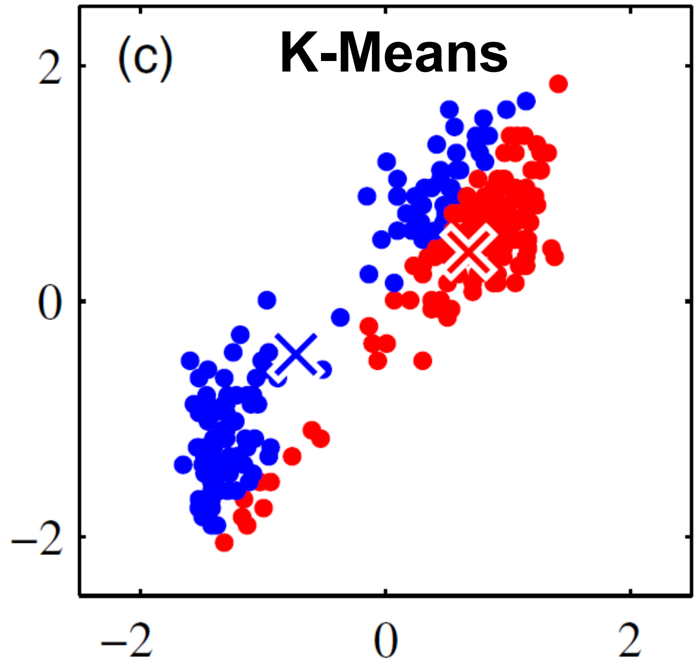


Assign data to cluster with closest center

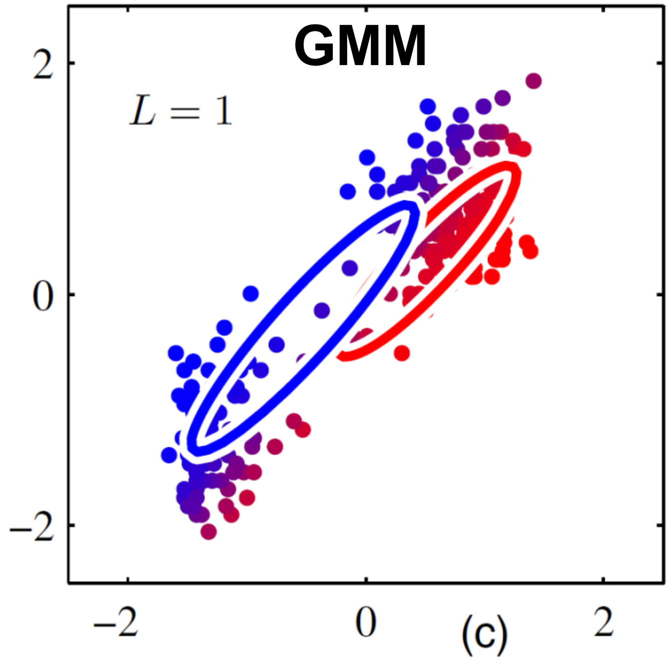


E-Step: Compute responsibilities

Comparison to K-Means



Recompute cluster centers as average of all data in cluster



M-Step: Maximize lower bound to compute new mean / covariances

Generating Data

Mixture Models are generative, and define a joint distribution over the assignment and data,

$$p(z, x) = p(z)p(x \mid z)$$

Can use this to generate new *synthetic data*:

Step 1: Sample cluster assignment from prior,

$$z_n \sim p(z)$$

Step 2: Sample data from component distribution,

$$x_n \sim p(x \mid z_n)$$

This may not seem useful, but with an advanced modeling, you could generate fake faces!

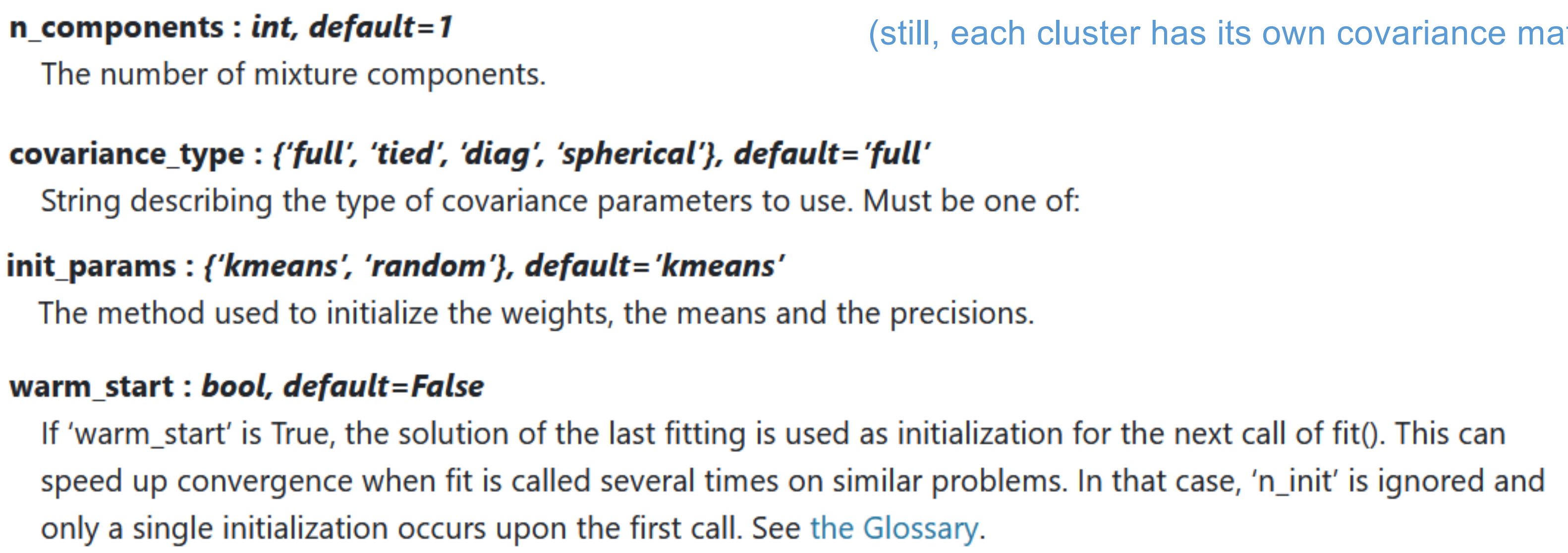
sklearn.mixture.GaussianMixture

Input parameters:

diag: no correlation

spherical: same variance for all coordinates

(still, each cluster has its own covariance matrix)



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

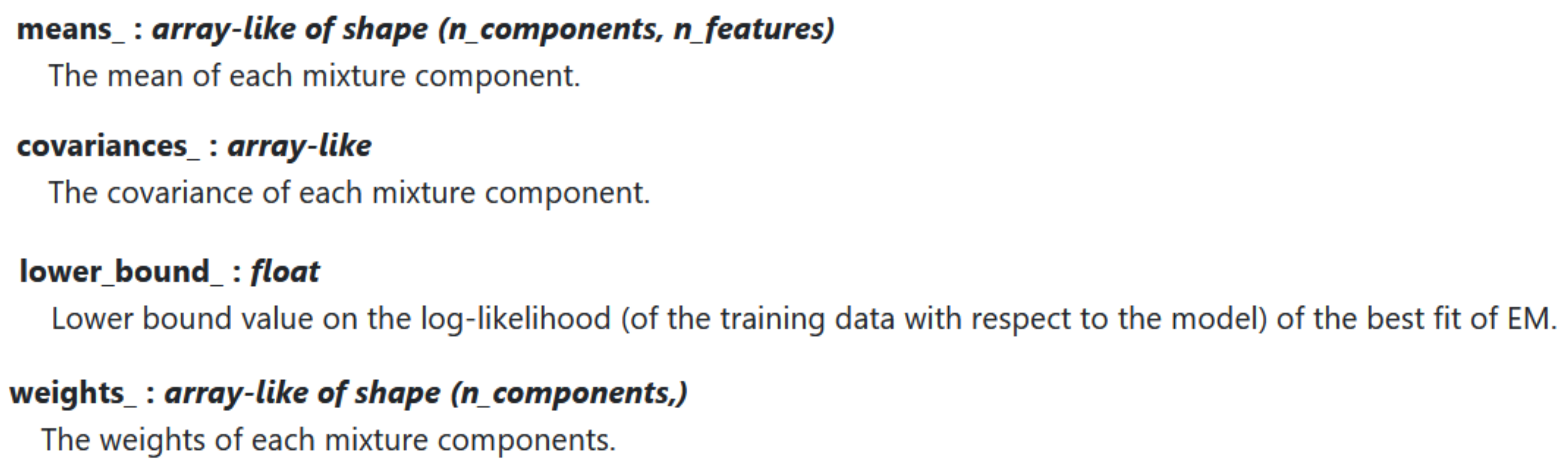
covariance_type : *{‘full’, ‘tied’, ‘diag’, ‘spherical’}, default=’full’*
String describing the type of covariance parameters to use. Must be one of:

init_params : *{‘kmeans’, ‘random’}, default=’kmeans’*
The method used to initialize the weights, the means and the precisions.

warm_start : *bool, default=False*
If ‘warm_start’ is True, the solution of the last fitting is used as initialization for the next call of fit(). This can speed up convergence when fit is called several times on similar problems. In that case, ‘n_init’ is ignored and only a single initialization occurs upon the first call. See the Glossary.

sklearn.mixture.GaussianMixture

Attributes -- most available after calling fit(X):



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

The mean of each mixture component.

covariances_ : *array-like*

The covariance of each mixture component.

lower_bound_ : *float*

Lower bound value on the log-likelihood (of the training data with respect to the model) of the best fit of EM.

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

The weights of each mixture components.

Scikit-Learn : GMM Example

Load Iris dataset,

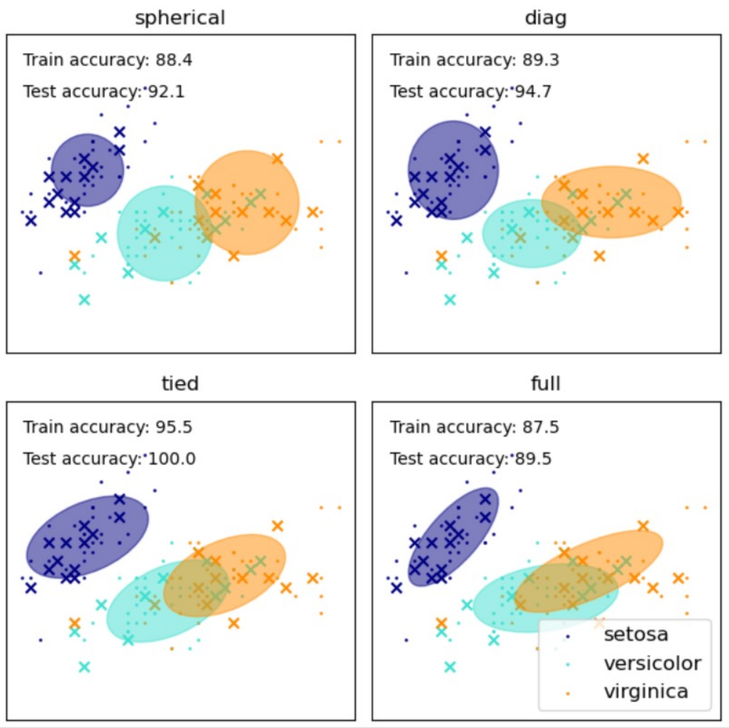
```
X_train = iris.data[train_index]
y_train = iris.target[train_index]
X_test = iris.data[test_index]
y_test = iris.target[test_index]
```

Define several 3-component GMMs with different covariances,

```
estimators = {
    cov_type: GaussianMixture(
        n_components=n_classes, covariance_type=cov_type, max_iter=20
    )
    for cov_type in ["spherical", "diag", "tied", "full"]
}
```

Fit each of them…

```
for index, (name, estimator) in enumerate(estimators.items()):
    estimator.fit(X_train)
```



accuracy is meaningful for illustration purpose; it's not available in practice!

Comparison with k-means

↓ initialize just like k-means

- Initialize: $w \in \Delta^K$, $\{\mu_k \in \mathbb{R}^d, \Sigma_k \in \mathbb{R}^{d \times d}\}_{k=1}^K$

  ↑ usually, uniform weight.

  ↑ can set it to be a diagonal matrix that has c in the diagonal (e.g., $c = 1$)

- (E)xpectation step:
  - $\gamma_{nk} = \frac{w_k\, p(x_n \mid z_n = k)}{\sum_{k'=1}^K w_{k'}\, p(x_n \mid z_n = k')}$ responsibility
  - Let $N_k = \sum_{n=1}^N \gamma_{nk}$ soft counts
- (M)aximization step:
  - $\mu_k' = \frac{1}{N_k} \sum_{n=1}^N \gamma_{nk} x_n$
  - ~~$\Sigma_k' = \frac{1}{N_k} \sum_{n=1}^N \gamma_{nk} (x_n - \mu_k')(x_n - \mu_k')^\top$~~ note we use μ_k' rather than μ_k
  - ~~$w_k' = \frac{N_k}{N}$~~
  - ~~Set $\mu_k \leftarrow \mu_k'$, $\Sigma_k \leftarrow \Sigma_k'$, $w_k \leftarrow w_k'$~~
- Stop when: the log likelihood does not increase much or the parameters do not change much.

MODIFY:
1. assume $\Sigma_k = I$
2. for each n,
   force the largest γ_{nk} to be 1 and set 0 otherwise
   (thus, $(\gamma_{n1}, \ldots, \gamma_{nK})$ is a one-hot vector)

Parting notes on GMM

- In some ways, more sensitive to initialization than K-Means
  - Needs to learn more “stuff” (DxD covariance matrices)
- Generally good practice to regularize covariance matrix
  - Covariance can shrink to zero in some extreme cases
  - Scikit-Learn allows addition of small constant value to diagonal
- Fully Bayesian model adds prior probabilities to mean / covariance parameter
  - Estimates mean / covariance using maximum a posteriori MAP
  - Scikit-Learn supports this in: `sklearn.mixture.BayesianGaussianMixture`