Computer Science

CSC380: Principles of Data Science

Clustering : Mixture Models

Prof. Jason Pacheco
TA: Enfa Rose George
TA: Saiful Islam Salim

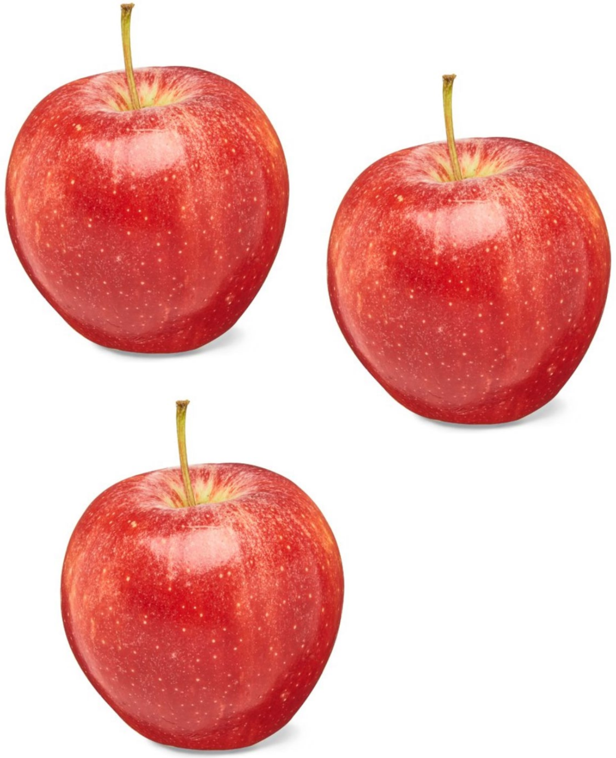
Administrative Items

- HW9 out (Due: Tue 12/7)
    - Released early by request

- Take-home Final Exam
    - Out next week (TBD)

Clustering

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



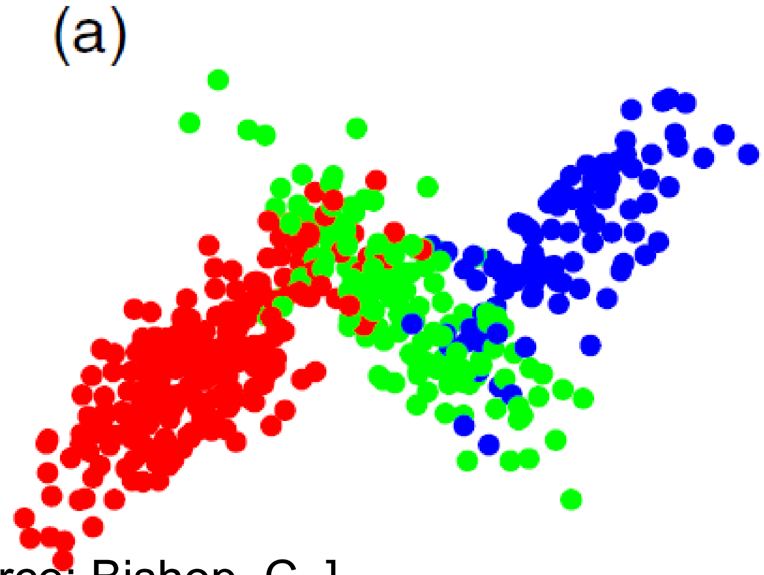
Hard Cluster Assignments

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

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

Cluster Center

Assignment **n^{th} Data Point**



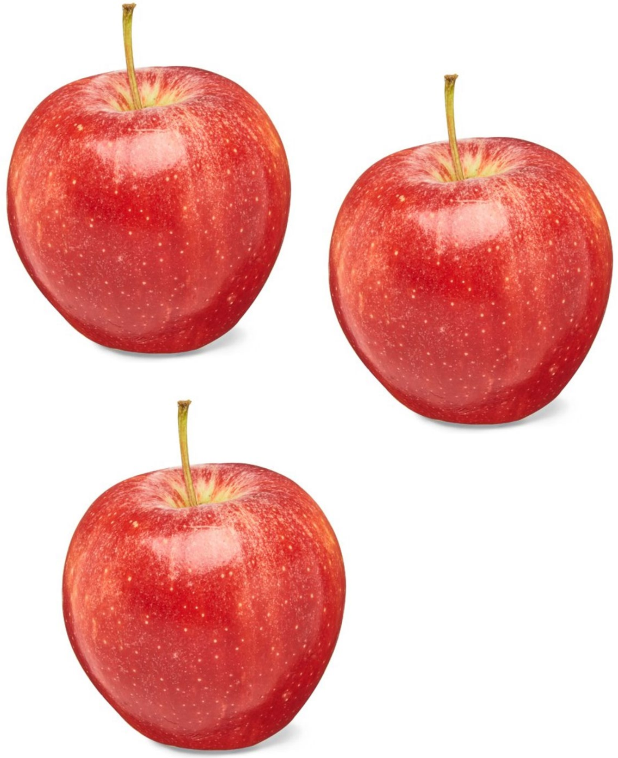
This is a *hard assignment* model

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

$$p(z_n \mid x_n)$$

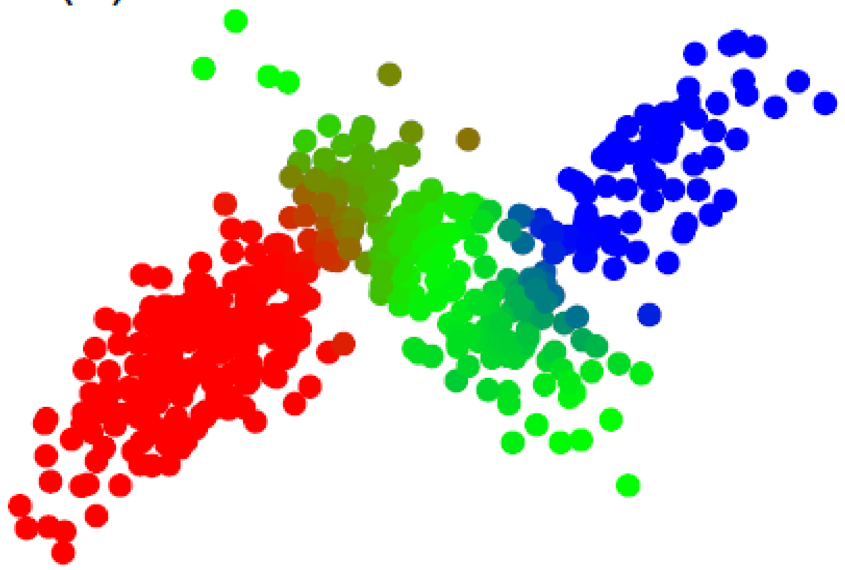
This is a *soft assignment* model

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.]

Mixture Model

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

- This is a **Bayesian model** – we have mostly seen frequentist models as of late

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

- Example of a *latent variable model*

Mixture Model

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

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,

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

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

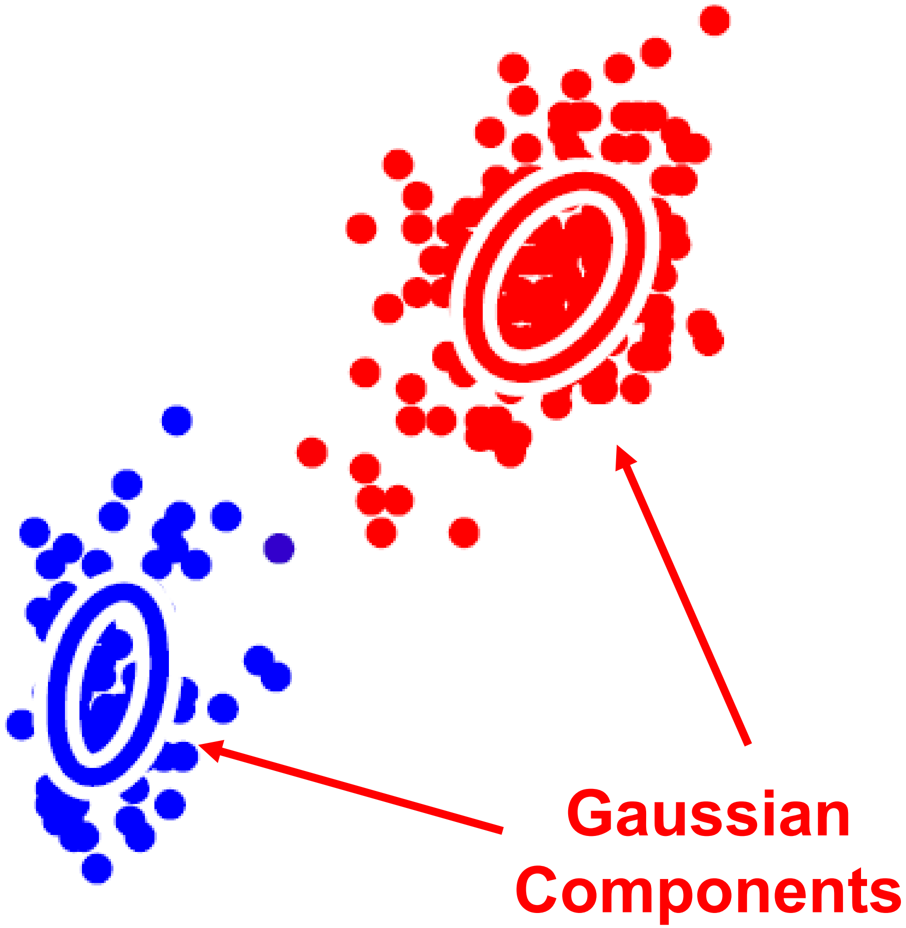
Gaussian Mixture Model

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

Assignment probabilities aren't just Euclidean distance



[Source: Bishop, C.]

Soft GMM Assignments (Responsibilities)

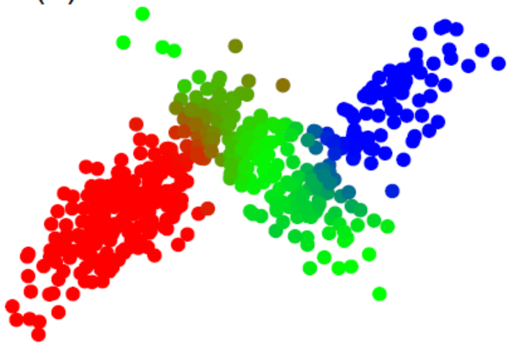
Recall that by Bayes' rule we have the posterior,

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

For Gaussian mixtures this is,

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

(c)



[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)

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

- 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…

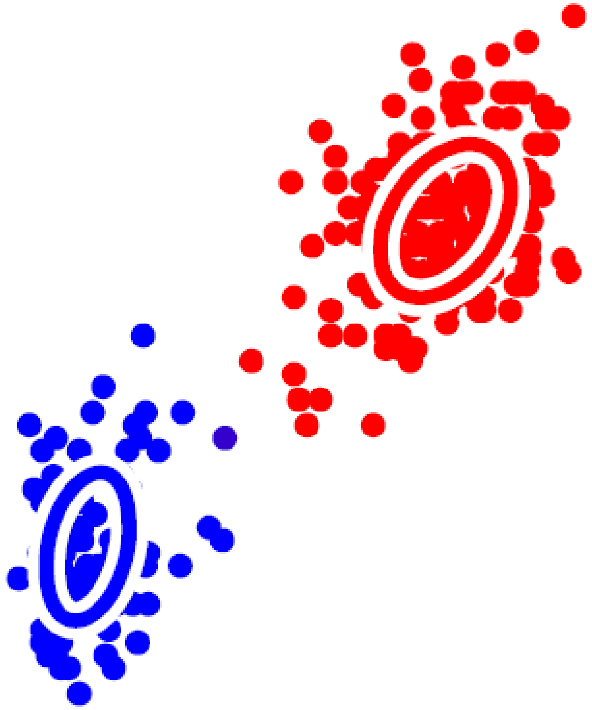
Learning Gaussian Mixture Models (GMMs)

For D-dimensional X need to learn…

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

D-dimensional vector of mean parameters

DxD Matrix of covariance parameters



…for K components this requires learning

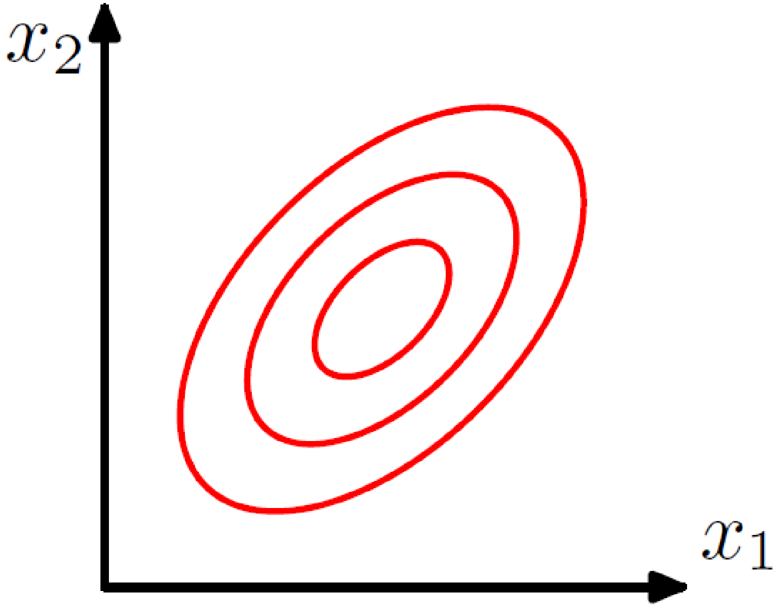
$$\mathcal{O}(KD + KD^2)$$

parameters

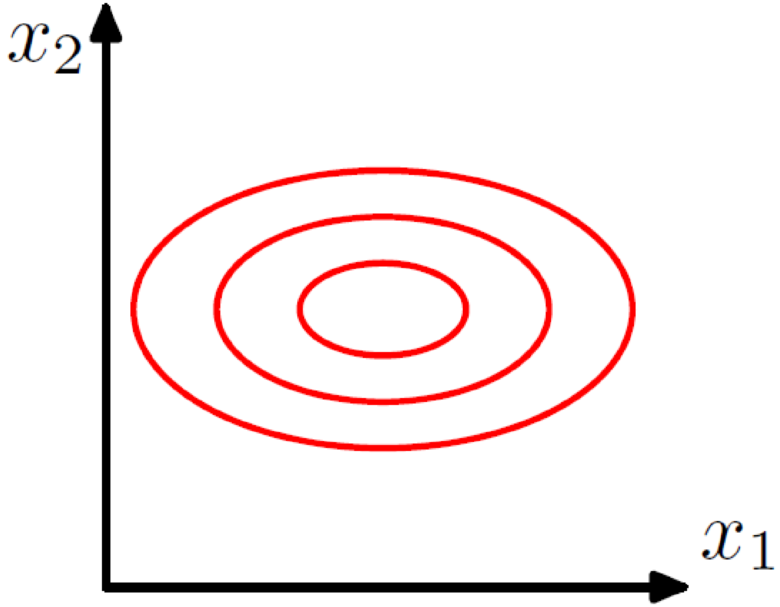
[Source: Bishop, C.]

Covariance

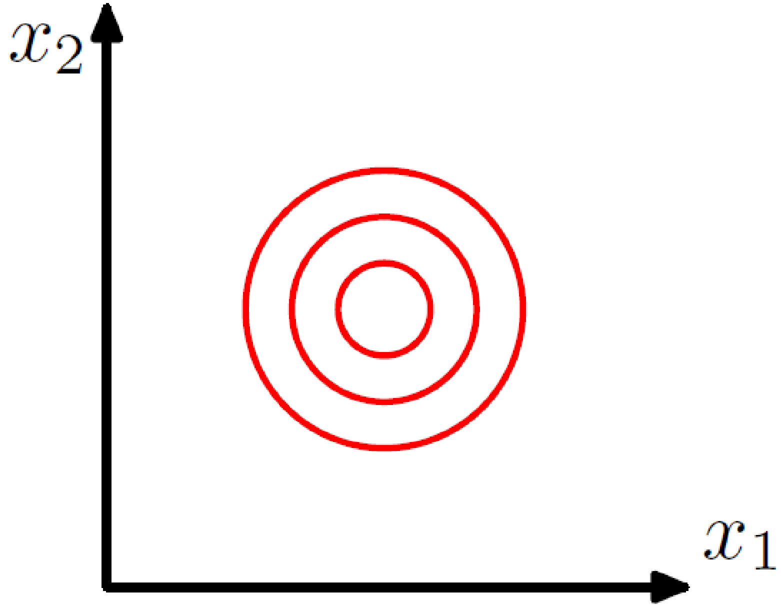
Captures correlation between random variables…can be viewed as set of ellipses…



Positive Correlation



Uncorrelated



Uncorrelated and same variance (isotropic / spherical)

Covariance Matrix

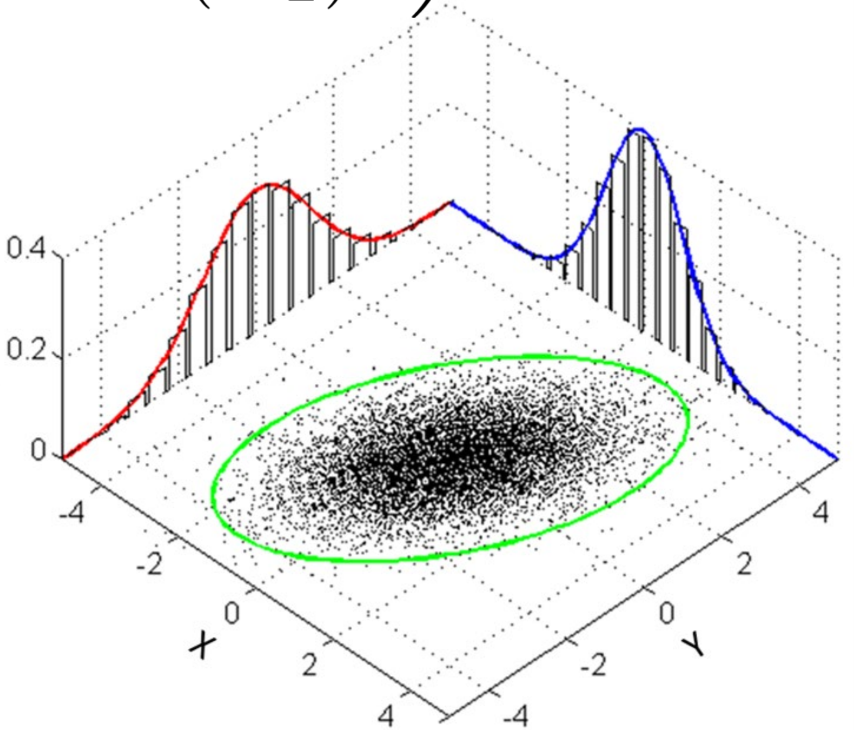
$$\Sigma = \mathrm{Cov}(X) = \begin{pmatrix} \mathrm{Var}(X_1) & \rho\sigma_{X_1}\sigma_{X_2} \\ \rho\sigma_{X_1}\sigma_{X_2} & \mathrm{Var}(X_2) \end{pmatrix}$$

Covariance Matrix

Marginal variance of just the RV X_1

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

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



Covariance Matrix

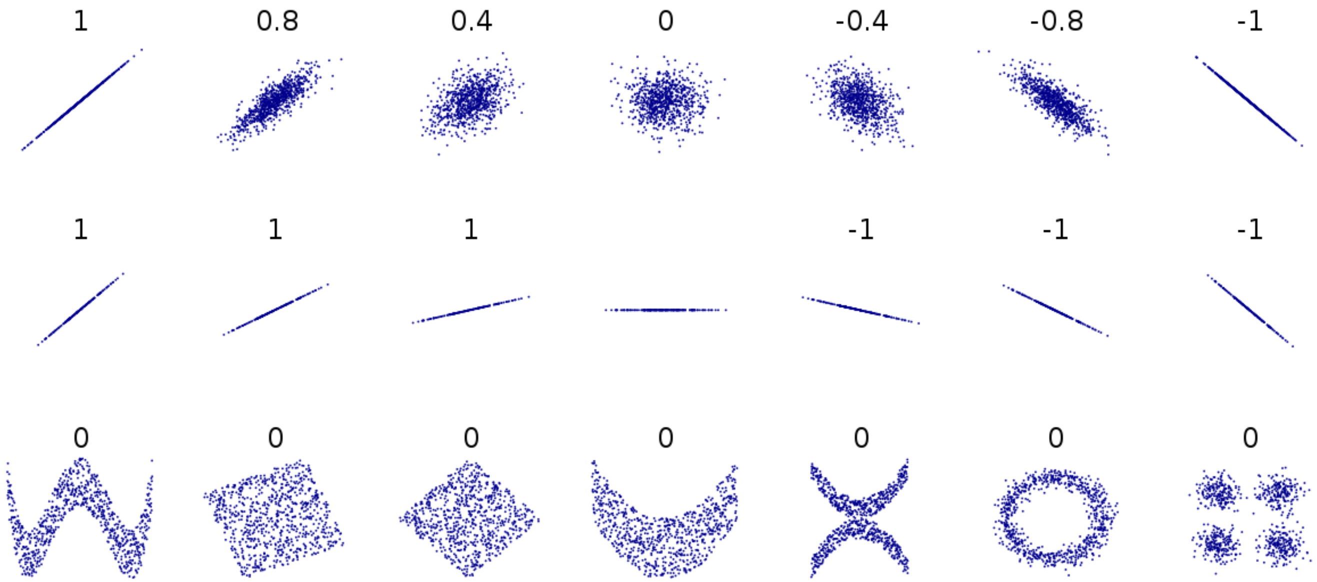
Correlation between X_1 and X_2

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

Recall, correlation is given by:

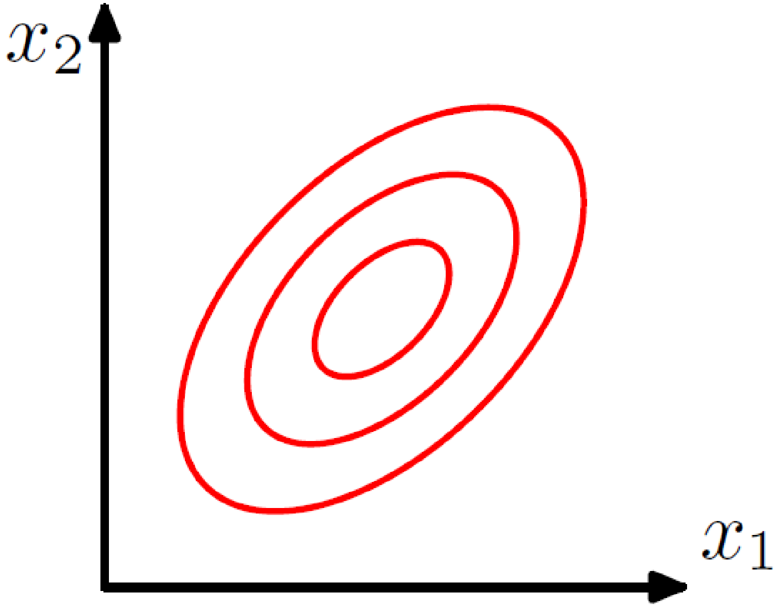
$$\rho = \frac{\mathbf{Cov}(X_1, X_2)}{\sigma_{X_1}\sigma_{X_2}}$$

Captures *linear dependence of RVs*



Covariance

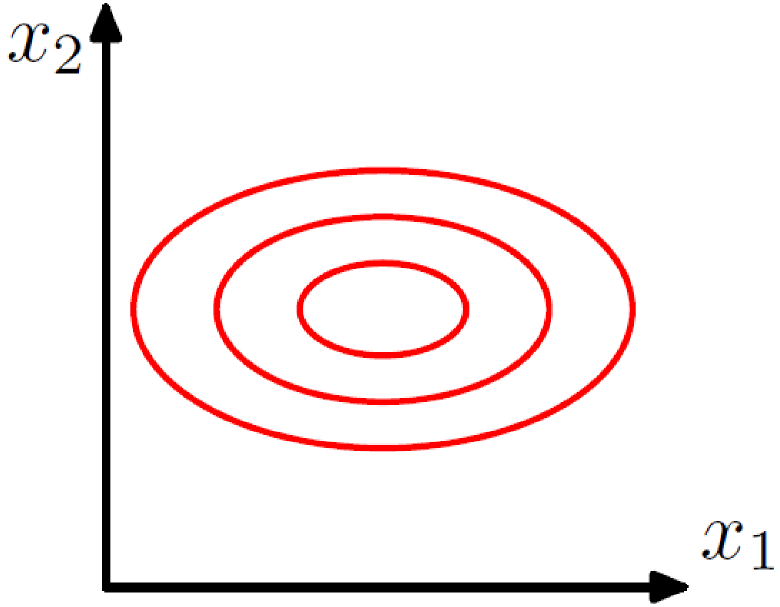
Captures correlation between random variables…can be viewed as set of ellipses…



Positive Correlation

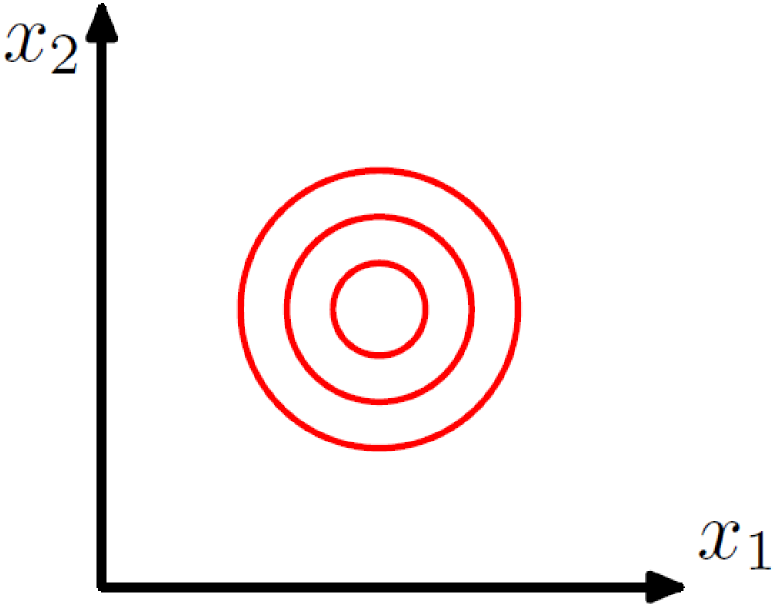
$\rho > 0$

Full matrix Σ



Uncorrelated

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



Isotropic / Spherical

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

Learning the GMM

Need to learn the mean / variance parameters…

m_k and Σ_k for all k=1,…,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,

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

This is a Gaussian mixture with K^N modes! 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,

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

True log-likelihood
Non-concave
Hard to maximize

Lower bound
Concave
Easy to maximize

We approximate maximum likelihood by optimizing the lower bound,

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

Expectation Maximization (EM)

Given a “guess” of parameters m^{old} and Σ^{old} forms the lower bound,

$$\widetilde{\mathcal{L}}(m,\Sigma) = \sum_{Z} \underbrace{p(Z \mid X, m^{\text{old}}, \Sigma^{\text{old}})}_{\text{Responsibility using our “guess” of component parameters}} \log p(X, Z \mid m, \Sigma)$$

Mixture model joint PDF as a function of parameters

- Lower bound $\mathcal{L} \geq \widetilde{\mathcal{L}}$ 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 bound to find new parameters

Expectation Maximization

Initialize Cluster parameters m_k^{old} and Σ_k^{old} randomly for all K

Expectation Step Compute responsibilities for all data points,

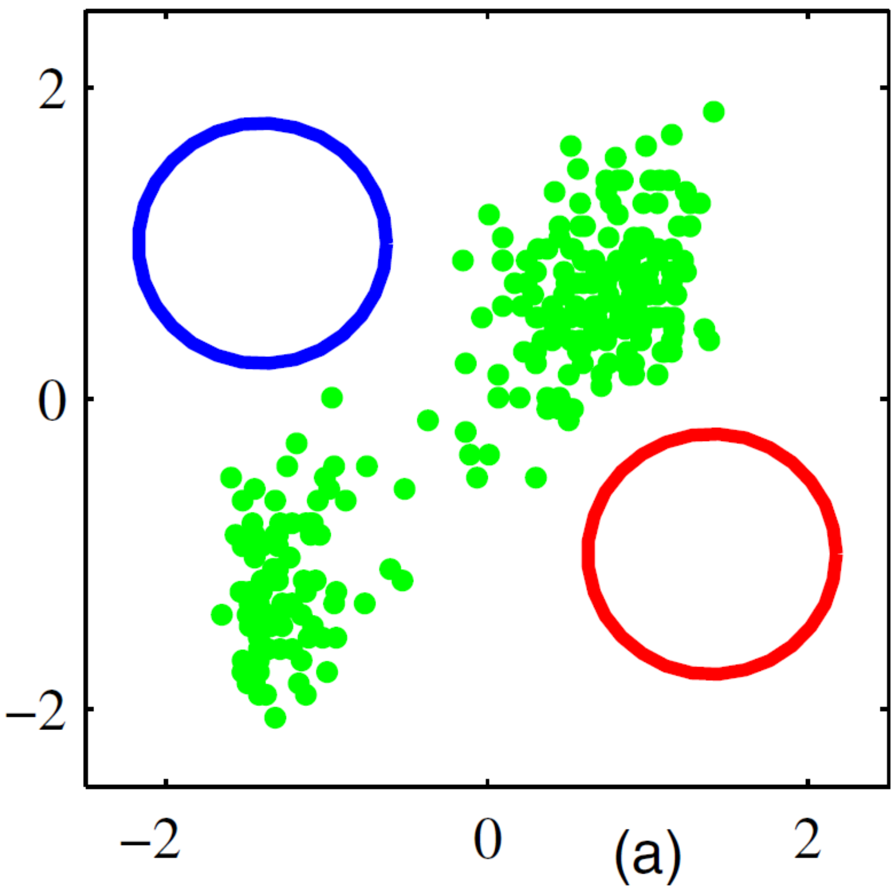
$$p(z_n = k \mid x_n) = \frac{w_k \mathcal{N}(x_n \mid m_k^{\text{old}}, \Sigma_k^{\text{old}})}{\sum_{i=1}^{K} w_i \mathcal{N}(x_n \mid m_i^{\text{old}}, \Sigma_i^{\text{old}})}$$

Maximization Step Update parameter estimates by maximum likelihood,

$$m^{\text{new}}, \Sigma^{\text{new}} = \arg\max_{m,\Sigma} \widetilde{\mathcal{L}}_N(m, \Sigma)$$

EM for GMM

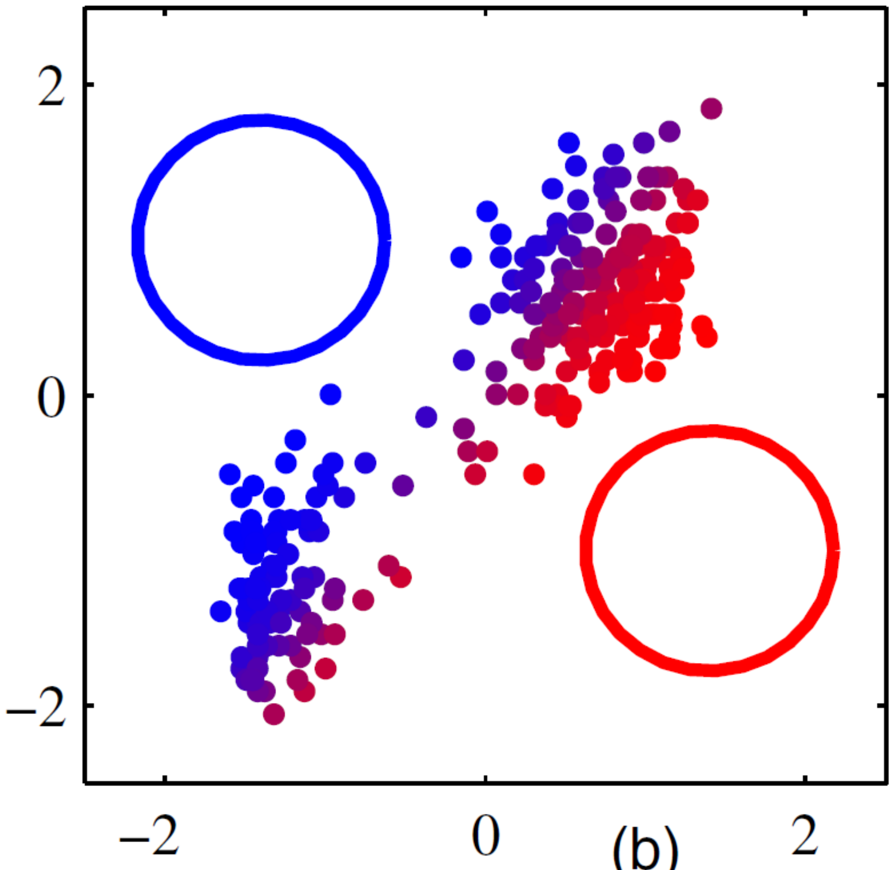
Initialize Cluster parameters



[Source: Bishop, C.]

EM for GMM

E-Step Compute responsibilities

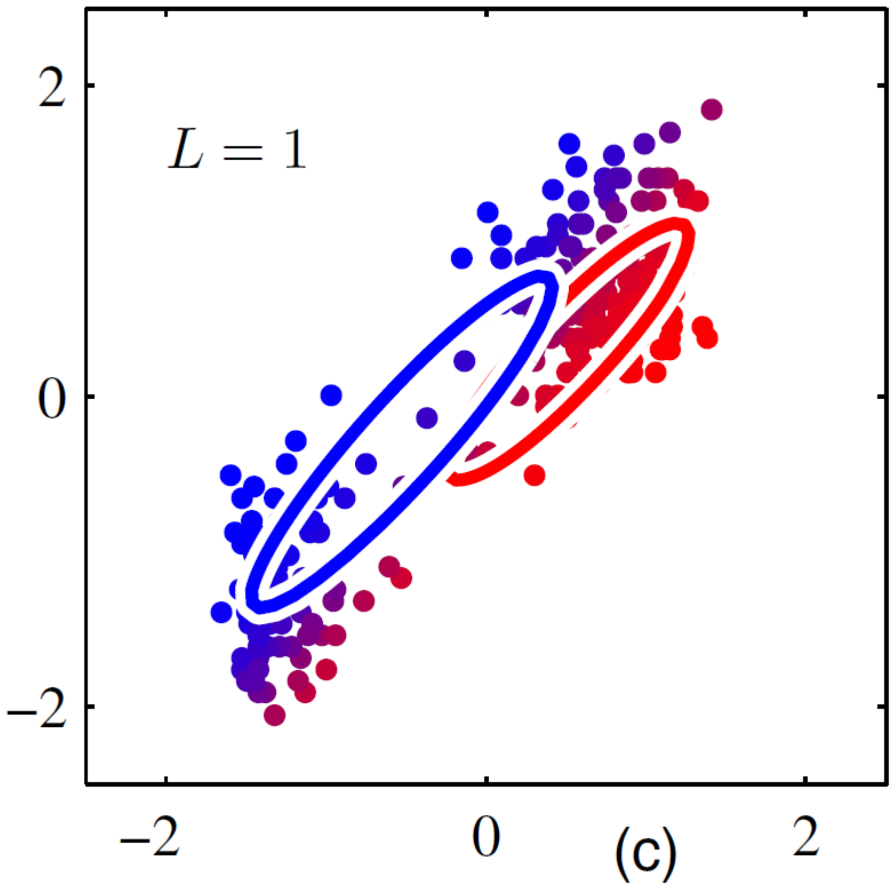


(b)

[Source: Bishop, C.]

EM for GMM

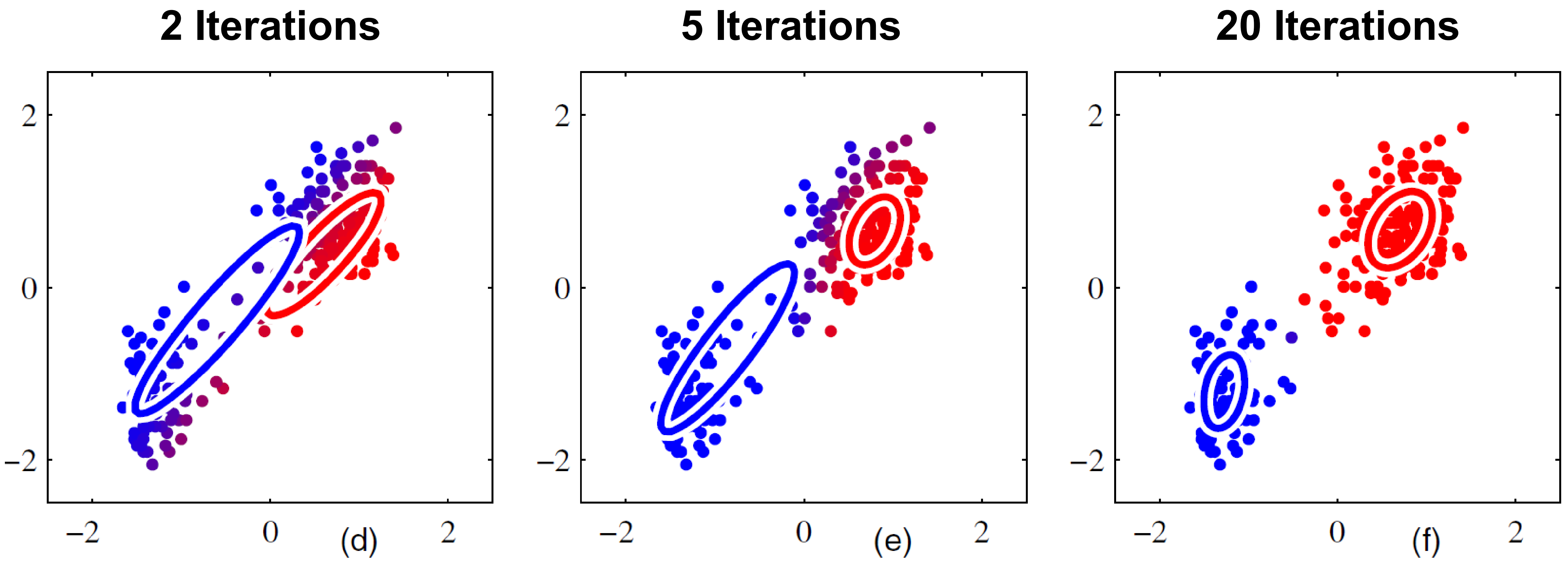
M-Step Maximize lower bound to find new component parameters



(c)

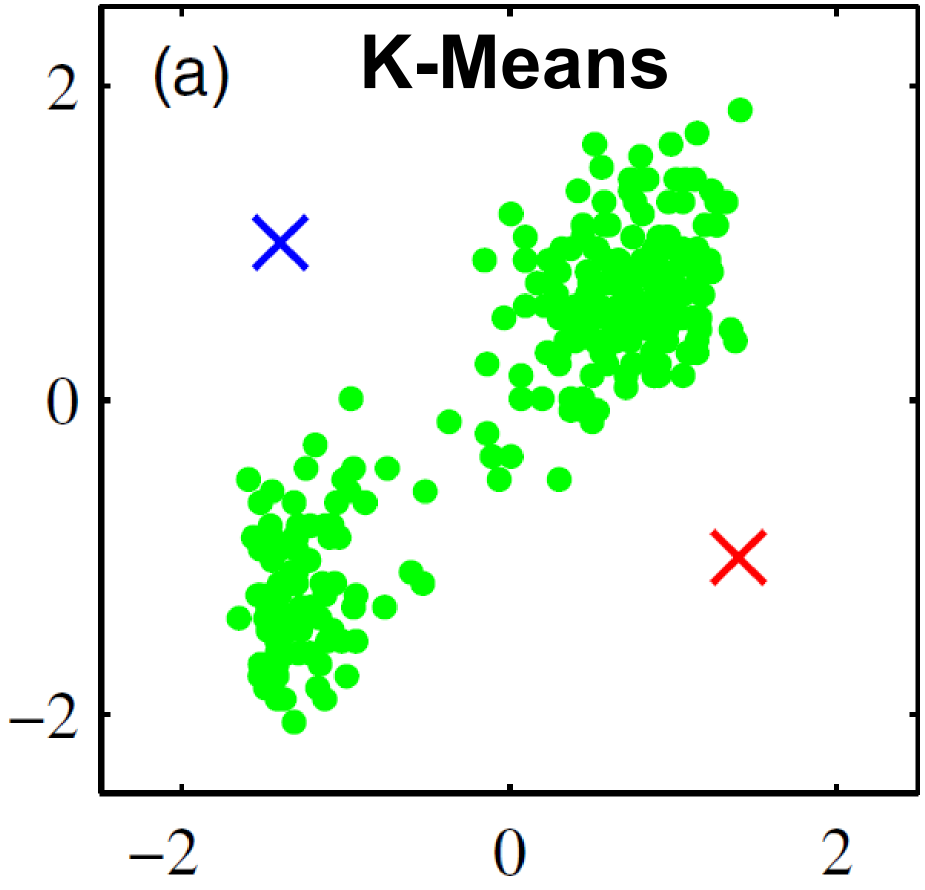
[Source: Bishop, C.]

EM for GMM

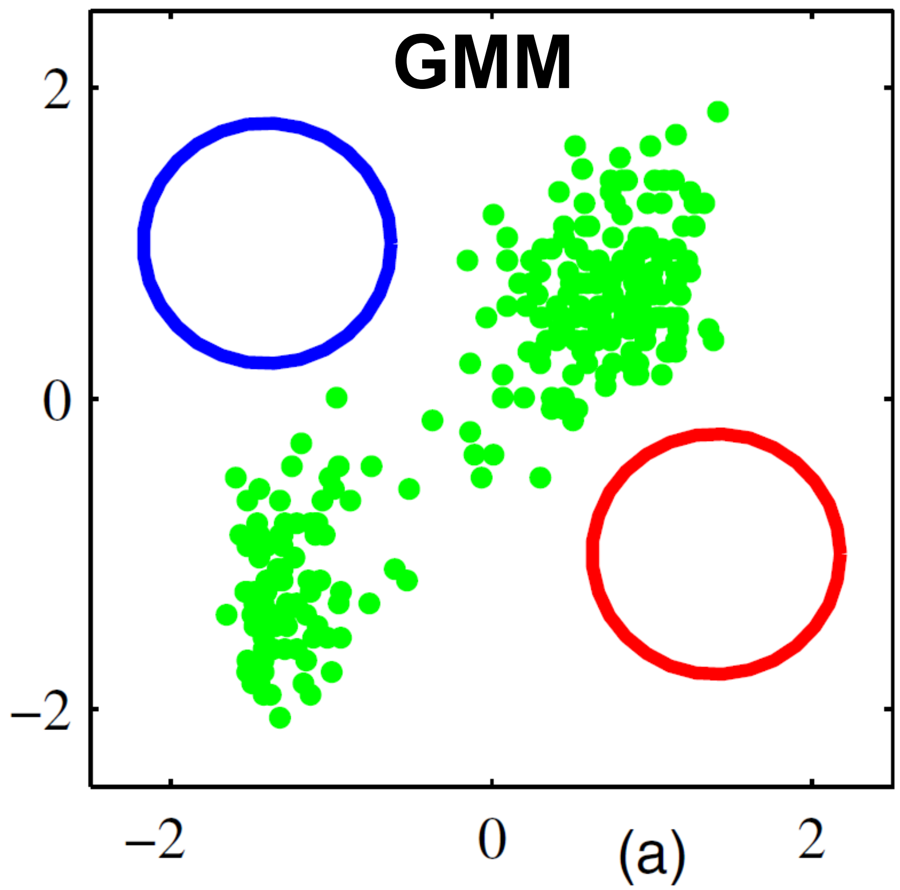


[Source: Bishop, C.]

Comparison to K-Means

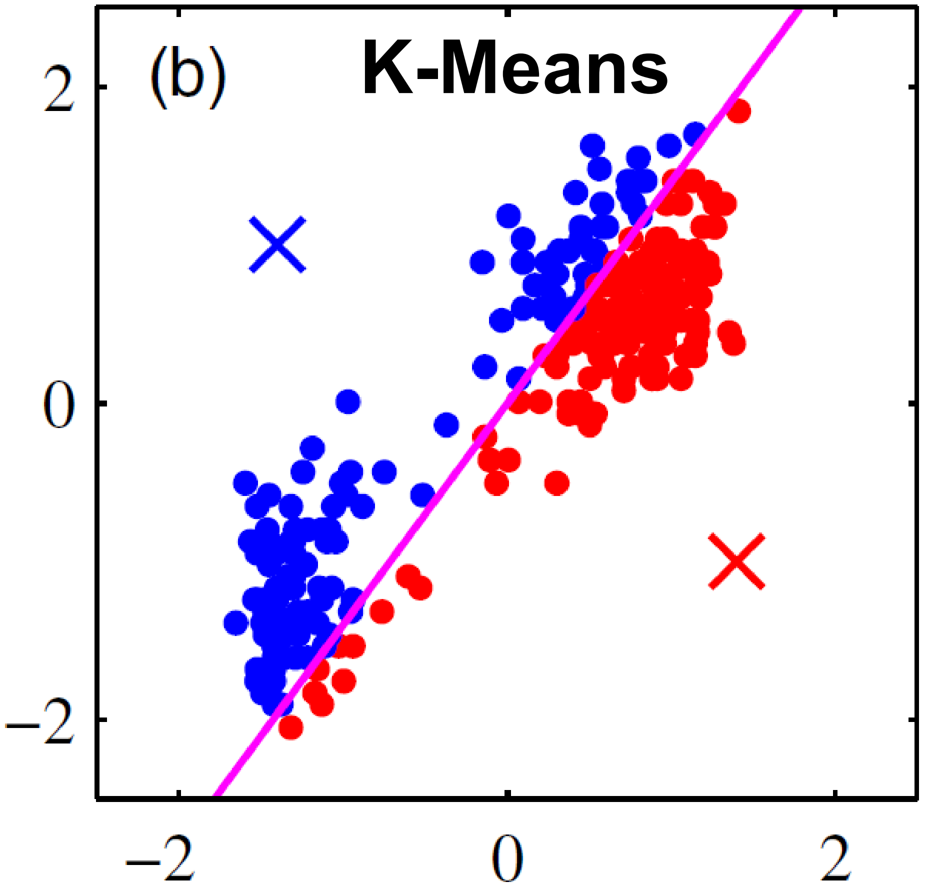


Initialize cluster centers

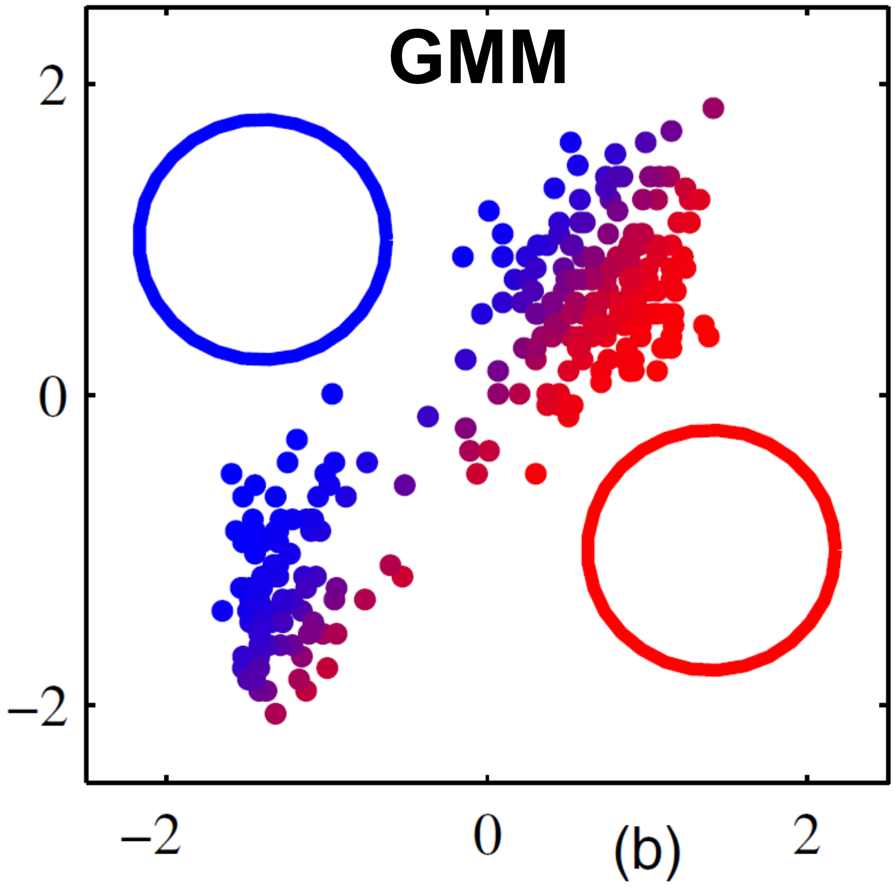


Initialize cluster mean / covariance

Comparison to K-Means

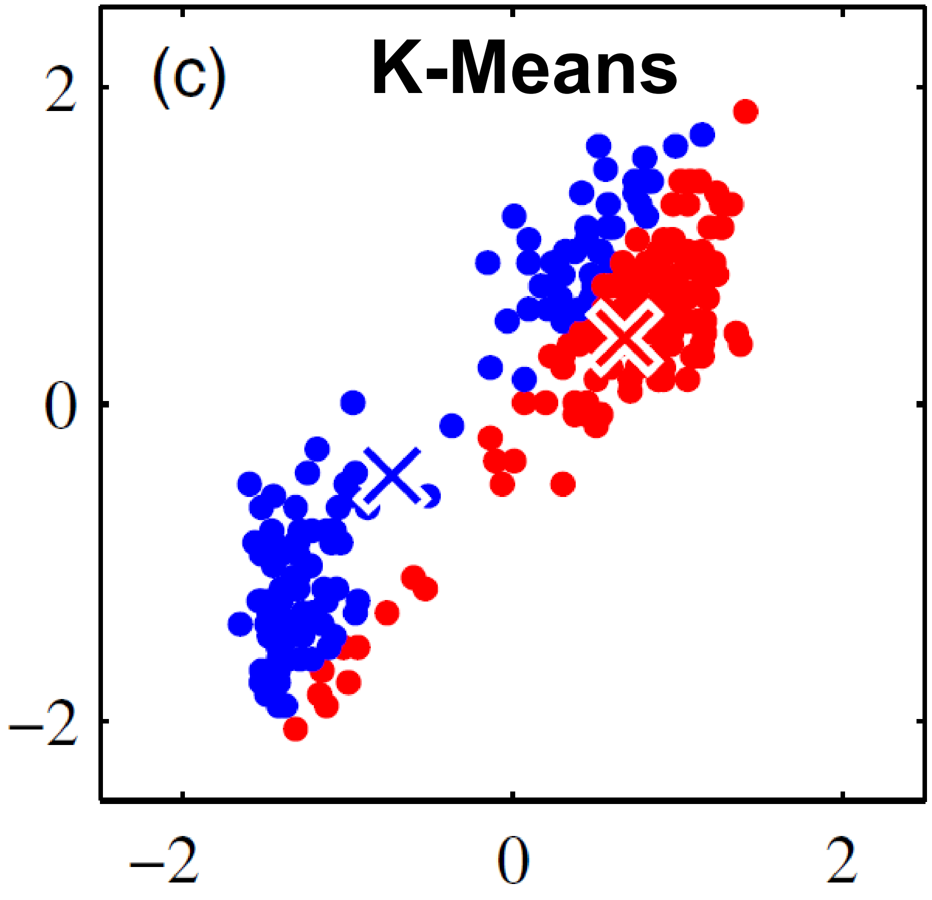


Assign data to cluster with closest center

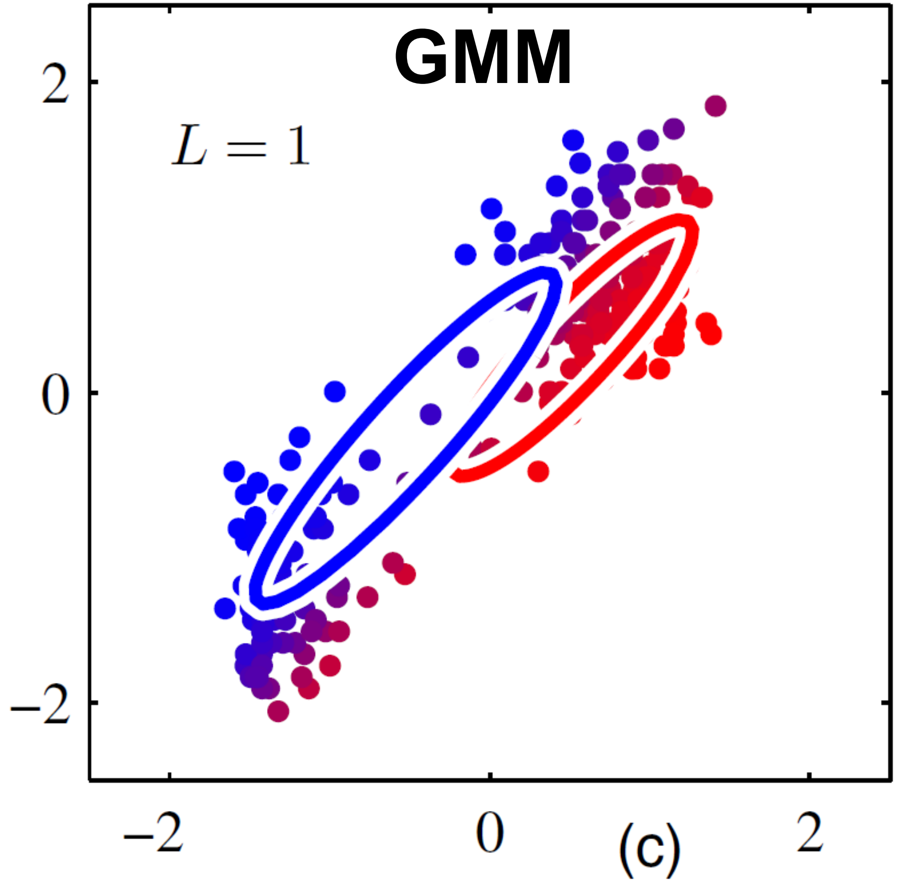


E-Step: Compute responsibilities

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 is an advantage over K-Means, but is not generative

`sklearn.mixture`.GaussianMixture

Input 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:

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,

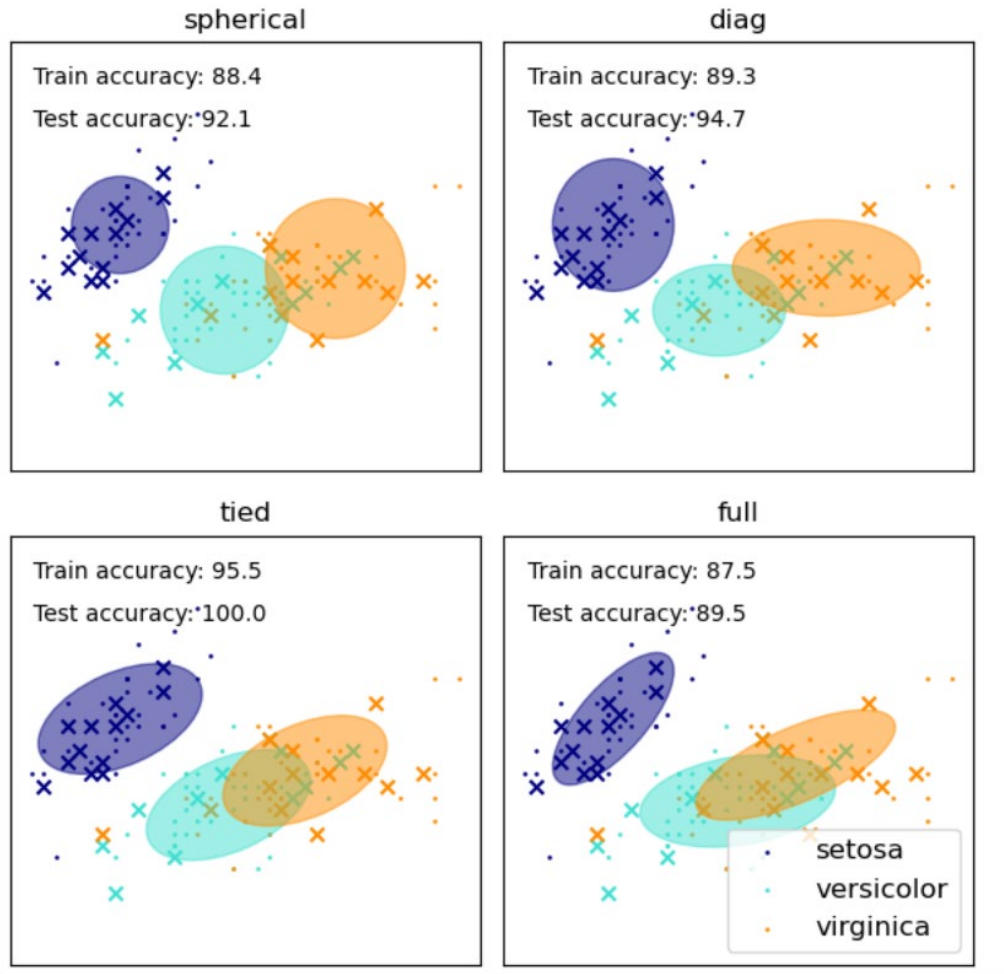
```
X_train = iris.data[train_index]
y_train = iris.target[train_index]
X_test = iris.data[test_index]
y_test = iris.target[test_index]
```

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)
```



Parting notes on GMM

- In some ways, more sensitive to initialization than K-Means
  - Needs to learn more "stuff" (DxD covariance matrices)
  - K-Means exactly maximizes objective, whereas EM maximizes lower bound on non-concave function

- 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`