Gaussian Mixtures

Abdullah Zafar
DEEP LEARNING 1
Hochschule Hamm-Lippstadt
Lippstadt, Germany
abdullah.zafar@stud.hshl.de

***Abstract*—This paper initials discusses the mathematical Basics of Gaussian Mixtures and how Gaussian Mixtures are proven mathematically. The equation used to describe Gaussian Mixtures and a brief summary describing all the variables that are mentioned in the equation. Furthermore, Expectation-Maximization in Gaussian mixture models are mentioned and how are they useful and their implementations in the given scenario. Drawbacks of K means Clustering furthermore emphasis on the important Gaussian Mixture Models and how they fulfill these gaps created by K means Clustering. Bayesian Gaussian Mixture Models and Anomaly detection using gaussian mixture models are mentioned in good detail aswell, as these are some important topics to be covered along Gaussian Mixtures.**

Mathematical Equations and usage is also discussed along the paper. Graphical images are used repeatedly to better explain the topic as Gaussian Mixtures rely heavily on graphical representation.
Implementing an example with python framework and applications are also discussed .Lastly, The paper includes Gaussian Mixture Model implementations and how they are implemented in real world. Summary and Conclusion is a brief look at the topic that are discussed in the paper.

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Machine Learning includes two methods of learning, supervised and unsupervised. The main distinguishing factor between the both includes the method it uses to deal with data. Clustering is a unsupervised method where clusters area made around the data set that share the same characteristics. Futhermore,Clustering could be regarded as grouping of similar data points together, based on the features and the attributes that they share. K-means is a popular method of clustering but a major drawbacks associated towards it is that it does not specify that how much a data point is associated with a specific cluster. This is commonly known as hard Clustering. Gaussian Mixture Models empowers us to implement soft clustering and get a better representation of the data available.

Gaussian Mixture Models presume that a specific number of Gaussian distributions exist and each one them represents a cluster. As a result, the data points belonging to a single distribution tend to be grouped together in a Gaussian Mixture Model.

A Gaussian mixture model is a probabilistic model that assumes all the given data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters and tends to group the given points to a single distribution that it belongs to.All the instances that are produced from the same Gaussian Distribution form a cluster that usually resembles an ellipsoid. Clusters vary in many different aspects from one another, few of those variations include shape, density, orientation, and size.

The Gaussian mixtures implement the expectation-maximization algorithm for fitting mixture-of-Gaussian models. Gaussian Mixtures further compute Bayesian Information Criterion to asses the number of clusters in a given data. Gaussian Mixture fit method enables the learning of a Gaussian Mixture Model from a given train data. Each sample is then assigned to a Gaussian using Gaussian Mixture predict method. [2]

II. Gaussian Mixtures mathematical basics

Inability of Gaussian Distribution to model real data sets overshadow its important analytical properties. Super position of several Gaussian always gives a better representation of the data available. Gaussian mixture models enables us to form super positions and can be formulated as probabilistic models known as mixture distributions.

By utilizing an adequate number of Gaussians, and by changing their methods and covariances just as the coefficients in the linear combination, practically any continuous density can be approximated to arbitrary precision. We in this manner consider a superposition of K Gaussian densities of the form, which are called Gaussian Mixtures. [1]

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Fig. 1. (1)
[1]

- μ that defines the center.
- covariance that is represented using $\sum$.
- A mixing probability π that defines how big or small the Gaussian function will be.

Each Gaussian density $N(x|\mu_k, \sum_k)$ is called component of the mixture and it possesses its own mean μ_k and covariance $\sum_k$ [1].

The parameters π_k in (1) are referred as mixing coefficients.integrating both sides of (1) with respect to x, and note that both p(x) and the individual Gaussian components are normalized, we obtain:

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

Fig. 2. (2)
[1]

Also, the requirement that p(x) $\geq$ 0, together with $N(x|\mu_k, \sum_k) \geq 0$, implies $\mu_k \geq 0$ for all k. Combing this with the condition in (2) we obtain: $0 \leq \mu_k \leq 1$.

We therefore see that the mixing coefficients satisfy the requirements to be probabilities. From the sum and product rules, the marginal density is given by:

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(k) p(\mathbf{x}|k)$$

Fig. 3. (3)
[1]

which is equivalent to (1) in which we can view $\mu_k = p(k)$ as the prior probability of picking the k^{th} component, and the density $N(x|\mu_k, \sum_k) = p(x|k)$ as the probability of x conditioned on k.

III. GMM Graphical model interpretation

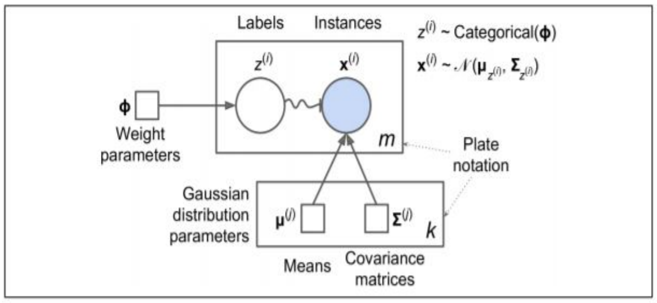

Labels
Instances
$z^{(i)}$ ~ Categorical($\boldsymbol{\phi}$)
$\mathbf{x}^{(i)} \sim \mathcal{N}(\boldsymbol{\mu}_{z^{(i)}}, \boldsymbol{\Sigma}_{z^{(i)}})$
$z^{(i)}$
$\mathbf{x}^{(i)}$
$\boldsymbol{\phi}$
Weight parameters
m
Plate notation
Gaussian distribution parameters
$\boldsymbol{\mu}^{(j)}$
$\boldsymbol{\Sigma}^{(j)}$
k
Means
Covariance matrices

Fig. 4.
[2]

Figure 4.0 represents the structure of the conditional dependencies between random variables. Illustration 4.0 can be interpreted in the following manner [2]:

- Random variables are represented by circles.
- The squares address fixed qualities (i.e., boundaries of the model).
- The big square rectangles are called plates: they show that their substance is repeated a few times.
- Each plate's bottom right hand side has a number that denotes how many times its content is repeated.
- The categorical distribution with weights ϕ is used to generate each variable z^i. Each variable x^i is selected from a normal distribution, with its cluster z^i defining the mean and covariance matrix.
- Conditional dependencies are represented by arrows.
- The wavy arrow that points from z^i to x^i represents a switch. This means that depending on the value of z^i, the instance x^i will get sampled from a different Gaussian Distribution.

IV. Anomaly detection using Gaussian mixture models

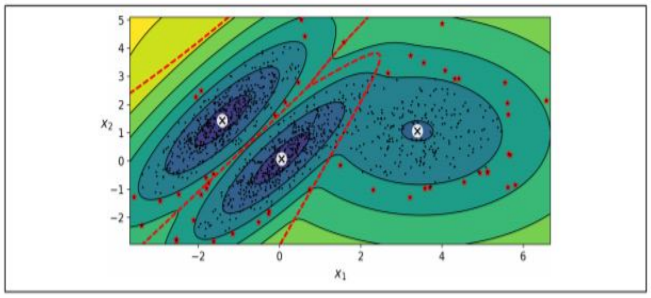


Fig. 5. Anomalies are represented as stars
[2]

Anomaly Detections is the process of locating the instances that oppose the convention in a very strong manner. These instances are commonly referred as anomalies or outliers, and the remaining ones are referred as inliers. Anomaly detection gets very handy when they are applied to real life scenarios. Few of the examples include, detecting fraud, detecting defective products in manufacturing or to obsolete dataset another feeding another model, which can improve the performance of the resulting models by many steps. Detecting anomalies in Gaussian mixture model is not a hefty task because any instance being located in a low density region can be referred as an anomaly.
Novelty detection differs from anomaly detection in such a way that in novelty detection, the algorithm is assumed to be trained on a ''clean" dataset, free of outliers, whereas anomaly detection does not make that assumption.
Gaussian mixture models tries to accommodate all the available data, including outliers, if outliers are more in number, the 'normality' gets effected and some outliers may be now considered as normal. If this happens, you can try to fit the model

once, use it to detect and remove the most extreme outliers, then fit the model again on the cleaned up dataset. Another approach is to use robust covariance estimation methods (see the EllipticEnvelope class).

V. Bayesian Gaussian Mixture Models

Selecting number of clusters can be a long task but it is possible to use Bayesian Gaussian mixture class which is in fact capable of giving weights equal or close to zero to the clusters those are unnecessary. Simply set the number of clusters n components to a value that you have good cause to believe is more than the optimal number of clusters, and the algorithm will automatically remove the extraneous clusters.

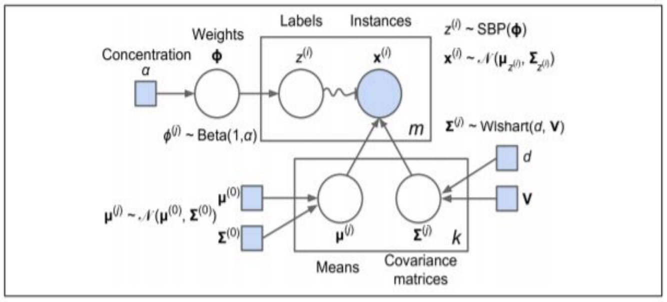

Labels
Instances
Weights
Concentration
α
ϕ
$z^{(i)}$
$x^{(i)}$
$z^{(i)} \sim \text{SBP}(\phi)$
$x^{(i)} \sim \mathcal{N}(\mu_{z^{(i)}}, \Sigma_{z^{(i)}})$
$\phi^{(i)} \sim \text{Beta}(1,\alpha)$
m
$\Sigma^{(j)} \sim \text{Wishart}(d, V)$
d
V
$\mu^{(0)}$
$\Sigma^{(0)}$
$\mu^{(j)} \sim \mathcal{N}(\mu^{(0)}, \Sigma^{(0)})$
$\mu^{(j)}$
$\Sigma^{(j)}$
k
Means
Covariance matrices

Fig. 6. Bayesin

In figure bayesin , the cluster parameters (including the weights, means and covariance matrices) are not treated as fixed model parameters anymore, but as latent random variables.

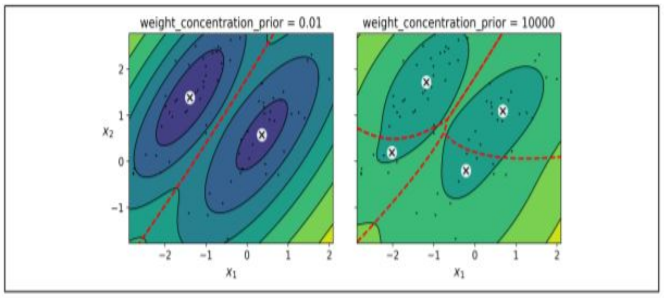


Fig. 7. Using different values of $p(z)$

Prior information of the latent variables z can be represented in the prior, which is a probability distribution p(z). For instance, We might be having an educated guess that the clusters are likely to be many(high concentration), or few(low concentration). This can be easily adjusted with the help of weightconcentrationprior hyperparameter. Figure 7 shows the results of setting the values to 0.01 and 1000 respectively, resulting into different clustering. [2]

VI. Bayes' theorem

Bayes's theorem can be used along estimated model paraments to estimate the posteriori component assignment probability. Learning the fact that an instance belongs to a specific distribution provides an important tool for learning clusters.Moreover, It assists in updating the probability distribution over the latent variables after data x has been observed. Bayes theorem computes the posterior distribution $P(z|x)$, which is the conditional probability of z given X. [2]

$$p(\mathbf{z}|\mathbf{X}) = \text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Evidence}} = \frac{p(\mathbf{X}|\mathbf{z})\, p(\mathbf{z})}{p(\mathbf{X})}$$

Fig. 8. Bayes Theorem
[2]

VII. Estimation Algorithm : Expectation-Maximization

Expectation maximization is a method that is used to estimate the parameters of the mixture model. Models are typically learned by maximum likelihood estimation, which tend to seek the maximum probability of the given data model parameters. It is usually analytically impossible to find maximum likelihood solution for mixture models by differential of log likelihood and solving for 00.
Expectation maximization is an iterative algorithmic method that tends to increase with each subsequent iteration. Generally two steps are involved in calculating Expectation maximization for Gaussian Mixture Models. [3]

The first step involves the calculation of the expectation of the component assignments for each data point given the model parameter such as ϕ μ and σ.The first step is referred as E step.

The second step is generally referred as maximization and it considers calculating the maximum expectation that was previously calculated in the first step with relation to the model parameters and this part consists of updating the values of ϕ μ and σ. As mentioned before, this method is an iterative method, so iterations take place until and unless the algorithm converges, hence giving the maximum likelihood estimate. Altering between the values that are known to be fixed or assumed, maximum likelihood successfully estimates the values those are non-fixed and they can hence be calculated in an efficient manner.

A. *Steps involved in applying EM for Gaussian Mixtures*

For the given Gaussian mixture model, the aim is maximizing the likelihood function with respect to its parameters . [1]

1) The initial step consists of initializing the parameters($\sum_k$,μ_k,π_k) and evaluating the initial value of the loglikelihood.
2) E Step. Utilizing current parameters for responsibilities evaluation.

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.$$

3) M step. Finding Re-estimated values of the parameters using the current responsibilities.

$$\begin{aligned}\boldsymbol{\mu}_k^{\text{new}} &= \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})\mathbf{x}_n \\ \boldsymbol{\Sigma}_k^{\text{new}} &= \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})(\mathbf{x}_n-\boldsymbol{\mu}_k^{\text{new}})(\mathbf{x}_n-\boldsymbol{\mu}_k^{\text{new}})^{\mathrm{T}} \\ \pi_k^{\text{new}} &= \frac{N_k}{N}\end{aligned}$$

:

$$N_k = \sum_{n=1}^{N}\gamma(z_{nk}).$$

4) The convergence of parameter or the log likelihood are to be checked, in case the convergence criterion is not fulfilled, step 2 needs to be returned to.

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^{N}\ln\left\{\sum_{k=1}^{K}\pi_k\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right\}$$

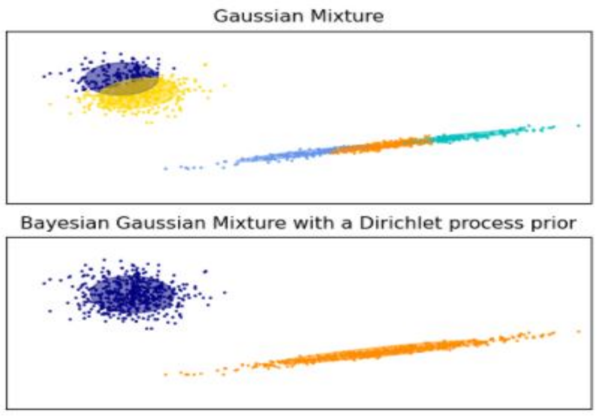
VIII. Variational Bayesian Gaussian Mixture

A. *Variational Inference*

[4]

Variational inference is a continuation of expectation-maximization that tends to maximize a lower bound on model evidence rather than data likelihood. The theory behind expectation-maximization and variational inference is very similar, but variational methods tend to regularization by integrating information from prior distributions. This helps to avoid the singularities that are found in expectation-maximization solutions. Inference is however quite slow as compared to expectation maximization.

Variational algorithm's Bayesian nature requires it to have more hyper-parameters as compared to expectation-maximization. weight_concentration_prior could be regarded as one of the most important concentration parameters. Setting a high concentration prior encourages a big number of components in the mixture to be active. On the other hand, setting a low concentration causes the model to place the majority of its weight on a small number of components, effectively reducing the weights of the remaining components to zero.



[4]

The illustration above compares GMM with a specific number of components to the variational Gaussian mixture models with a Dirichlet process prior. It can be concluded from the given illustration that variational Gaussian mixture with a Dirichlet process prior is able to stay within only two components, whereas the Gaussian mixture fits the data involving a fixed number of components that must be set a priori by the user. The number of components in the given set was selected as 5, however it does not match the true generative distribution of the given dataset. It can easily be concluded that Gassuian mixture models with a Dirichlet process prior tends to fit only one component.

IX. How Gaussian Mixture Models differs from K-means

K-Means Clustering is a method that divides sample data into K clusters depending on features. Clustering is accomplished by shortening the distance between the sample and the cluster's center.

Hard clustering is a more popular term for this. Lesser processing time, better for high-dimensional data, and it's a lot easier to install are just a few of the benefits connected with K means. Hard clustering has a number of drawbacks, including the possibility of erroneous data and the fact that, unlike Gaussian mixtures, it does not function well with complex geometrically formed data.

K-Means clustering and Gaussian Mixtures have no logical relationship. The data will cluster into roughly spherical clumps centered on the means of each mixture component if it fits a spherical Gaussian mixture model effectively. K-means clustering excels at the following types of data: Clusters that individually match to a mixture component will be found, with cluster centers near the mixture.
However, you can use k-means clustering without making any assumptions about the data's origins. It can be used to split up data into useful and reasonably homogeneous bits in the same way that other clustering algorithms may, with no presumption that those bits are genuine objects (eg, for market segmentation). You can prove what k-means estimates are without assuming mixed models.

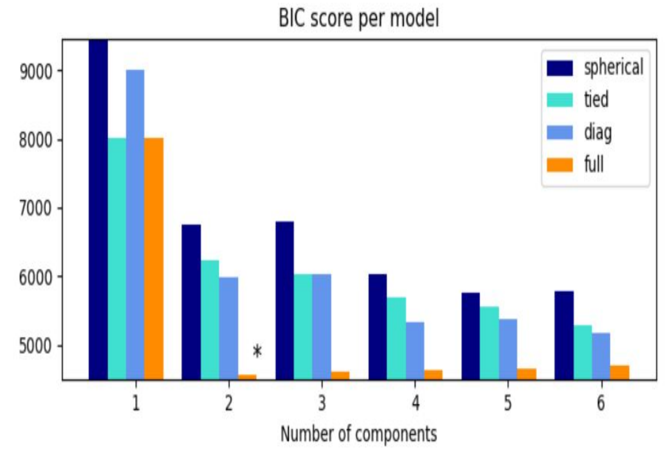
Gaussian mixture models can be fitted using maximum likelihood, which is a different estimator and methodology

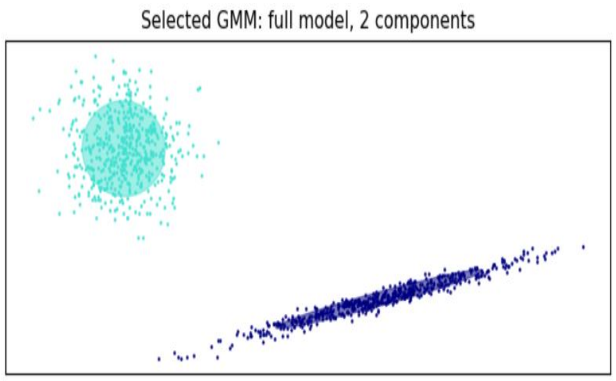
than k-means. Alternatively, Bayesian estimators and their accompanying methodologies can be used.
Spherical Gaussian mixture models and k-means clustering are very comparable in several ways. They're not only distinct, but they're also different kinds of things.

X. Python Implementation(Gaussian Mixture Model Selection)

The BIC criterion can be used to select the number of components in a Gaussian Mixture in an efficient way. In theory, it recovers the true number of components only in the asymptotic regime (i.e. if much data is available and assuming that the data was actually generated i.i.d. from a mixture of Gaussian distribution). Note that using a Variations Bayesian Gaussian mixture avoids the specification of the number of components for a Gaussian mixture model. This example shows that model selection can be performed with Gaussian Mixture Models using information-theoretic criteria (BIC). Model selection concerns both the covariance type and the number of components in the model. In that case, AIC also provides the right result (not shown to save time), but BIC is better suited if the problem is to identify the right model. Unlike Bayesian procedures, such inferences are prior-free.

In that case, the model with 2 components and full covariance (which corresponds to the true generative model) is selected.

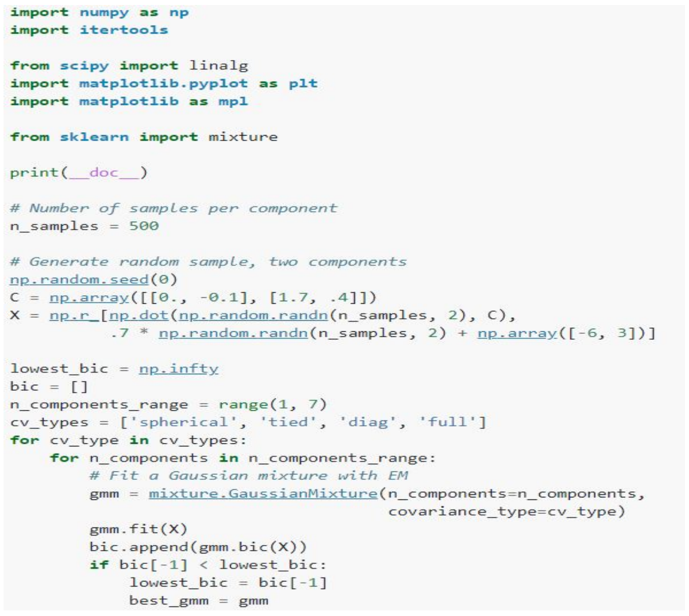




The following is a brief description of the code that was used to generate the illustration.

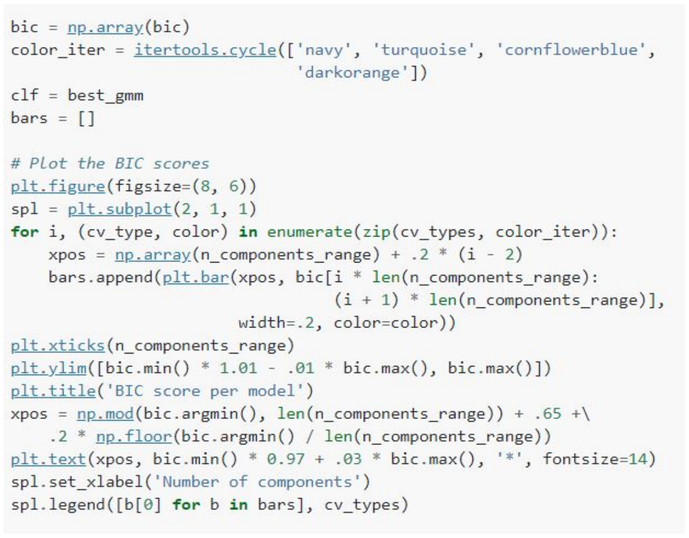
The following is a brief description of the code that was used to generate the illustration.

The initial part of the code contains importing the required libraries. nsamples basically represents the number of samples per components, which in this example is declared as 500. np.random.seed(0) function is used to generate random samples. fit[] is a method that helps in estimating the model parameters with the EM algorithm. bic[] is used as Bayesian information criterion for the current model on the input X. Bayesian information criterion is basically used to select a model among a finite number of models and the model with the lowest value of Bic is selected.



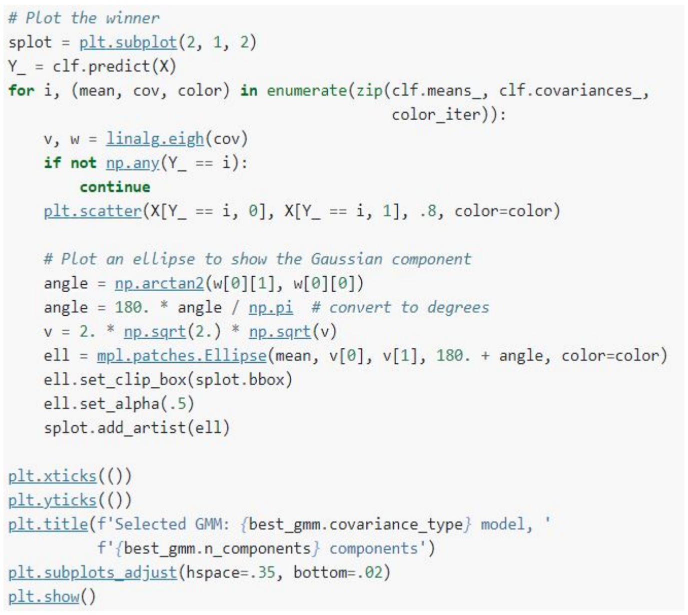
```python
import numpy as np
import itertools

from scipy import linalg
import matplotlib.pyplot as plt
import matplotlib as mpl

from sklearn import mixture

print(__doc__)

# Number of samples per component
n_samples = 500

# Generate random sample, two components
np.random.seed(0)
C = np.array([[0., -0.1], [1.7, .4]])
X = np.r_[np.dot(np.random.randn(n_samples, 2), C),
          .7 * np.random.randn(n_samples, 2) + np.array([-6, 3])]

lowest_bic = np.infty
bic = []
n_components_range = range(1, 7)
cv_types = ['spherical', 'tied', 'diag', 'full']
for cv_type in cv_types:
    for n_components in n_components_range:
        # Fit a Gaussian mixture with EM
        gmm = mixture.GaussianMixture(n_components=n_components,
                                      covariance_type=cv_type)
        gmm.fit(X)
        bic.append(gmm.bic(X))
        if bic[-1] < lowest_bic:
            lowest_bic = bic[-1]
            best_gmm = gmm
```

The next section of the code helps to plot the bic scores. There are the types of covariance used in this example, namely, full, tied, diag and spherical.



```python
bic = np.array(bic)
color_iter = itertools.cycle(['navy', 'turquoise', 'cornflowerblue',
                              'darkorange'])
clf = best_gmm
bars = []

# Plot the BIC scores
plt.figure(figsize=(8, 6))
spl = plt.subplot(2, 1, 1)
for i, (cv_type, color) in enumerate(zip(cv_types, color_iter)):
    xpos = np.array(n_components_range) + .2 * (i - 2)
    bars.append(plt.bar(xpos, bic[i * len(n_components_range):
                                  (i + 1) * len(n_components_range)],
                        width=.2, color=color))
plt.xticks(n_components_range)
plt.ylim([bic.min() * 1.01 - .01 * bic.max(), bic.max()])
plt.title('BIC score per model')
xpos = np.mod(bic.argmin(), len(n_components_range)) + .65 +\
    .2 * np.floor(bic.argmin() / len(n_components_range))
plt.text(xpos, bic.min() * 0.97 + .03 * bic.max(), '*', fontsize=14)
spl.set_xlabel('Number of components')
spl.legend([b[0] for b in bars], cv_types)
```

The last part of the code plots an ellipse to show the Gaussian component and plots the winner aswell.



```
# Plot the winner
splot = plt.subplot(2, 1, 2)
Y_ = clf.predict(X)
for i, (mean, cov, color) in enumerate(zip(clf.means_, clf.covariances_,
                                           color_iter)):
    v, w = linalg.eigh(cov)
    if not np.any(Y_ == i):
        continue
    plt.scatter(X[Y_ == i, 0], X[Y_ == i, 1], .8, color=color)

    # Plot an ellipse to show the Gaussian component
    angle = np.arctan2(w[0][1], w[0][0])
    angle = 180. * angle / np.pi  # convert to degrees
    v = 2. * np.sqrt(2.) * np.sqrt(v)
    ell = mpl.patches.Ellipse(mean, v[0], v[1], 180. + angle, color=color)
    ell.set_clip_box(splot.bbox)
    ell.set_alpha(.5)
    splot.add_artist(ell)

plt.xticks(())
plt.yticks(())
plt.title(f'Selected GMM: {best_gmm.covariance_type} model, '
          f'{best_gmm.n_components} components')
plt.subplots_adjust(hspace=.35, bottom=.02)
plt.show()
```

XI. Applications of GMM

Gaussian Mixture Models is an effective clustering technique and it is widely used in daily life applications. Listed below are a few examples:

- Fuzzy image segmentations
- Hand writing recognition
- Data stream clustering
- defect detection in products
- Modeling weather observations in geo-science
- Speech recognition

XII. Summary and conclusion

References

[1] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[2] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[3] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.