**PROJECT REPORT**

**ON**

**STUDENT MODELLING IN L.A. USING GMM Clustering**



**GUIDED BY:**

**PROF. OM PRAKASH VYAS**

**IIIT ALLAHABAD**

**(VISITING PROF. – UNIVERSITY OF PADERBORN, GERMANY)**

**SUBMITTED BY:**

**ADITYA VERMA**

**MTECH IT 1ST YEAR**, **IIIT ALLAHABAD**

**ABSTRACT**

Our objective includes, building of a robust and linear time working student model that can help us cluster various students based upon their test results, behaviors, participation in discussion groups, etc. Based upon the way they are clustered, instructor can send them the lecture material. This report also includes, the implementation for clustering of various students based upon their scores, attendances, number of times hands raised, how many times a student visited discussion forums, etc.

**KEYWORDS:** Learning Analytics, Student Models, Clustering Techniques

**INTRODUCTION**

Student models in intelligent tutoring systems represent student competencies and learning achievements. Modeling may involve techniques to represent content skills (e.g., mathematics, art history), knowledge about learning (e.g., metacognitive knowledge), and affective characteristics (e.g., emotional state). Although students’ general knowledge might be determined quickly from quiz results, their learning style, attitudes, and emotions are less easily determined and need to be inferred from long-term observations. Models may be used for assessment by measuring changes in the student in any or all three of these areas. Student models generally represent inferences about users (e.g. their level of knowledge, misconceptions, goals, plans, preferences, beliefs), relevant characteristics of users (stereotypes) and users' records, particularly past interactions with the system.

A student model in an intelligent tutor observes student behavior and creates a

qualitative representation of her cognitive and affective knowledge. This model

partially accounts for student performance (time on task, observed errors) and reasons about adjusting feedback to the student. By itself, the student model achieves very little; its purpose is to provide knowledge that is used to determine the conditions for adjusting feedback and it supplies data to other tutor modules, particularly the teaching module. One long-term goal of the field of AI and Education is to support learning for students with a range of abilities, disabilities, interests, backgrounds, and other characteristics.

In general terms, student modeling involves the construction of a qualitative representation that accounts for student behavior in terms of existing background knowledge about a domain and about students learning the domain. Such a representation, called a student model, can assist an intelligent tutoring system, an intelligent learning environment, or an intelligent collaborative

learner in adapting to specific aspects of student behavior.

While we can be thankful that some student modeling systems, notably those of the ACT\* tutors, have attained significant success, we have to admit that the construction of the background knowledge of many modelers, including those of the ACT\* tutors, involves the manual, tedious, time consuming and error-prone analysis of student protocols, the result of which are then hand-coded into knowledge bases, which, once coupled with a student modeler, remain fossilized until extended with human help. Machine learning techniques have been used to automatically extend the background knowledge, as well as to automatically induce the student model, but limitations as well as potentials remain.

This report examines the various ways in which machine learning or machine learning-like techniques have been used in the induction of student models and in the extension or construction of the background knowledge needed for student modeling.

However, we shall also see the promise of using machine learning techniques, particularly those for unsupervised learning, for constructing or extending the background knowledge needed for student modeling. Moreover, we will see that student modeling techniques might also be useful in machine learning.

This report discusses about the various attributes present in the data set that is used to build an unsupervised clustering based student model, that helps us in identifying and dividing the students among various categories/clusters. This report also compares various clustering techniques that can be used for clustering purpose and also compares each of them.

**LITERATURE REVIEW**

We are going to build a robust yet linear time working student model that can divide students into clusters, so that we can send them the lecture material if we find that certain group of students are not able to get or understand the lecture content.

Before choosing the appropriate clustering technique for this purpose we are going to look into the various attributes of the datasets based on which we are going to cluster them.

**DATASET SOURCE :**

Students' Academic Performance Dataset (xAPI-Edu-Data)

Data Set Characteristics: Multivariate

Number of Instances: 480

Area: E-learning, Education, Predictive models, Educational Data Mining

Attribute Characteristics: Integer/Categorical

Number of Attributes: 16

Date: 2016-11-8

Associated Tasks: Classification

Missing Values? No

File formats: xAPI-Edu-Data.csv

Source:

Elaf Abu Amrieh, Thair Hamtini, and Ibrahim Aljarah, The University of Jordan, Amman, Jordan, http://www.Ibrahimaljarah.com www.ju.edu.jo

Dataset Information:

This is an educational data set which is collected from learning management system (LMS) called Kalboard 360. Kalboard 360 is a multi-agent LMS, which has been designed to facilitate learning through the use of leading-edge technology. Such system provides users with a synchronous access to educational resources from any device with Internet connection.

The data is collected using a learner activity tracker tool, which called experience API (xAPI). The xAPI is a component of the training and learning architecture (TLA) that enables to monitor learning progress and learner’s actions like reading an article or watching a training video. The experience API helps the learning activity providers to determine the learner, activity and objects that describe a learning experience. The dataset consists of 480 student records and 16 features. The features are classified into three major categories: (1) Demographic features such as gender and nationality. (2) Academic background features such as educational stage, grade Level and section. (3) Behavioral features such as raised hand on class, opening resources, answering survey by parents, and school satisfaction.

The dataset consists of 305 males and 175 females. The students come from different origins such as 179 students are from Kuwait, 172 students are from Jordan, 28 students from Palestine, 22 students are from Iraq, 17 students from Lebanon, 12 students from Tunis, 11 students from Saudi Arabia, 9 students from Egypt, 7 students from Syria, 6 students from USA, Iran and Libya, 4 students from Morocco and one student from Venezuela.

The dataset is collected through two educational semesters: 245 student records are collected during the first semester and 235 student records are collected during the second semester.

The data set includes also the school attendance feature such as the students are classified into two categories based on their absence days: 191 students exceed 7 absence days and 289 students their absence days under 7.

This dataset includes also a new category of features; this feature is parent parturition in the educational process. Parent participation feature have two sub features: Parent Answering Survey and Parent School Satisfaction. There are 270 of the parents answered survey and 210 are not, 292 of the parents are satisfied from the school and 188 are not.

(See the related papers for more details).

Attributes

1. Gender - student's gender (nominal: 'Male' or 'Female’)

2 .Nationality- student's nationality (nominal:’ Kuwait’,’ Lebanon’,’ Egypt’,’ SaudiArabia’,’ USA’,’ Jordan’,’ Venezuela’,’ Iran’,’ Tunis’,’ Morocco’,’ Syria’,’ Palestine’,’ Iraq’,’ Lybia’)

3 . Place of birth- student's Place of birth (nominal:’ Kuwait’,’ Lebanon’,’ Egypt’,’ SaudiArabia’,’ USA’,’ Jordan’,’ Venezuela’,’ Iran’,’ Tunis’,’ Morocco’,’ Syria’,’ Palestine’,’ Iraq’,’ Lybia’)

4. Educational Stages- educational level student belongs (nominal: ‘lowerlevel’,’MiddleSchool’,’HighSchool’)

5. Grade Levels- grade student belongs (nominal: ‘G-01’, ‘G-02’, ‘G-03’, ‘G-04’, ‘G-05’, ‘G-06’, ‘G-07’, ‘G-08’, ‘G-09’, ‘G-10’, ‘G-11’, ‘G-12 ‘)

6. Section ID- classroom student belongs (nominal:’A’,’B’,’C’)

7. Topic- course topic (nominal:’ English’,’ Spanish’, ‘French’,’ Arabic’,’ IT’,’ Math’,’ Chemistry’, ‘Biology’, ‘Science’,’ History’,’ Quran’,’ Geology’)

8. Semester- school year semester (nominal:’ First’,’ Second’)

9. Parent responsible for student (nominal:’mom’,’father’)

10. Raised hand- how many times the student raises his/her hand on classroom (numeric:0-100)

11. Visited resources- how many times the student visits a course content(numeric:0-100)

12. Viewing announcements-how many times the student checks the new announcements(numeric:0-100)

13. Discussion groups- how many times the student participate on discussion groups (numeric:0-100)

14. Parent Answering Survey- parent answered the surveys which are provided from school or not (nominal:’Yes’,’No’)

15. Parent School Satisfaction- the Degree of parent satisfaction from school(nominal:’Yes’,’No’)

16. Student Absence Days-the number of absence days for each student (nominal: above-7, under-7)

**VAROIUS CLUSTERING TECHNIQUES :**

**1 . K-Means Clustering –**

* Centroid based clustering model.
* Works in Linear Time O(n).
* Fast working algorithm
* We need to know the number of clusters that will be formed beforehand.
* Random choice of clusters yields to different results on different runs of the algorithm

**2. K- Medians Clustering –**

* same as k-means algorithm but uses median vector of the group instead of the mean for re-computing the center points.
* Method is less sensitive to outliers.
* Much slower for larger databases

**3. Mean Shift Clustering –**

* a sliding-window-based algorithm that is used to find dense clusters of data points
* there is no need to select the number of clusters
* The con is that the selection of the window size/radius “r” can be non-trivial.

**4. DBSCAN**

* All the above algorithms can be used for spherical shaped clusters but DBSCAN algorithm works well both spherical shaped clusters as well as no spherical based clusters
* Divides the points into three categories : noice points, border points and core points , depending upon
* it does not require a pe-set number of clusters at all
* identifies outliers as noises unlike mean-shift which simply throws them into a cluster even if the data point is very different
* it is able to find arbitrarily sized and arbitrarily shaped clusters quite well.
* main drawback of DBSCAN is that it doesn’t perform as well as others when the clusters are of varying density

**5. Expectation–Maximization (EM) Clustering using Gaussian Mixture Models (GMM)**

* With GMMs we assume that the data points are Gaussian distributed
* we have two parameters to describe the shape of the clusters: the mean and the standard deviation
* this means that the clusters can take any kind of elliptical shape
* In order to find the parameters of the Gaussian for each cluster (e.g the mean and standard deviation) we will use an optimization algorithm called Expectation–Maximization (EM)
* GMMs are a lot more flexible in terms of cluster covariance than K-Means; due to the standard deviation parameter, the clusters can take on any ellipse shape, rather than being restricted to circles

**PREVIOUS WORK:**

‘ The Application of Gaussian Mixture Models for the Identification of At-Risk Learners in Massive Open Online Courses ’ has an application similar to what we are trying to achieve but we are using it in clustering students based upon certain dataset attributes , whereas they used it for the identication of At-Risk Learners in a Massive Open Online Courses.

**PROPOSED METHODOLOGY**

Our implementation includes the use of one of the most optimal techniques for clustering that is GMM, Gaussian Mixture Model.

This algorithm works in linear time and also does not depend only upon mean but also considers median for its working.

We are going to take input from our dataset and using some of the dataset attributes we are going to cluster students into various groups and categories and deliver the content accordingly.

If someone lies in a cluster that has low score then we are going to send the course material to all such students lying in the same clusters.

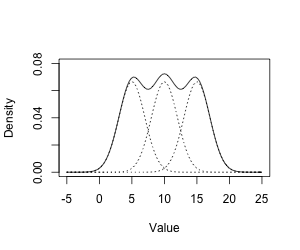
For students lying in the cluster of excellent score, we are not going to deliver them the same content but we give them further content from other topics of study.

**MORE ABOUT GMM**

Gaussian mixture models are a probabilistic model for representing normally distributed subpopulations within an overall population. Mixture models in general don't require knowing which subpopulation a data point belongs to, allowing the model to learn the subpopulations automatically. Since subpopulation assignment is not known, this constitutes a form of unsupervised learning.

For example, in modeling human height data, height is typically modeled as a normal distribution for each gender with a mean of approximately 5'10" for males and 5'5" for females. Given only the height data and not the gender assignments for each data point, the distribution of all heights would follow the sum of two scaled (different variance) and shifted (different mean) normal distributions. A model making this assumption is an example of a Gaussian mixture model (GMM), though in general a GMM may have more than two components. Estimating the parameters of the individual normal distribution components is a canonical problem in modeling data with GMMs.

GMMs have been used for feature extraction from speech data, and have also been used extensively in object tracking of multiple objects, where the number of mixture components and their means predict object locations at each frame in a video sequence.



## Expectation Maximization

With multiple Gaussian curves to learn, we now have to turn to the EM algorithm. Let's take a look at the math behind it to get started. Let's start with a single Gaussian function:

beginalignNxmusigmafrac1sigmasqrt2piefracxmu2sigma2endalign

To have a *mixture* or a **linear combination** of k Gaussians, the equation would look something like this:

beginalignpxboldsymbolmuboldsymbolsigmasumiin0kpiiNxmuisigmaiendalign

Before you get bogged down in all those symbols, let me explain what they mean. The probability now comtains a summation - that's the linear combination. We have a weighting factor pii per Gaussian curve. Also note that we now use mui and sigmai when calculating the probability. We pass in the corresponding mean and standard deviation.

On the left hand side, we now have bolded boldsymbolmu and boldsymbolsigma. These represent a vector of means and standard deviations.

beginalignboldsymbolmuleftbeginmatrixmu0mu1dotsmuk1endmatrixrightendalign

This is just for convenience. Imagine passing 50 variables to a function - that would be inconvenient. However, passing an array is convenient. This is the mathematical equivalent.

### Hidden variables

The mean and standard deviations can be inferred from the datapoints directly. We did that in the previous part - you simply plug in the formulae and calculate the mean and standard deviation. However, in our new model, we have a so called **Hidden variable** which is pii (the weighting factor for each Gaussian).

This is a new variable *we* have introduced and isn't observable from the data. How do we learn it if we can't even observe it? That's where Expectation Maximization comes into picture.

The goal of this technique is to assume some initial mean, standard deviation and pi values and iteratively improve the estimate. Every single iteration is made up of two steps - the so *E* step and the *M* step.

**The E step**: This is the expectation part. Using the current mean and standard deviation guess, we calculate probabilities. We calculate these values for each Gaussian. This helps us predict which Gaussian is responsble for which datapoint (called the *responsibility*).

**The M step**: This is the maximization part. Using the responsibilities in the previous step, we update the mean and standard deviations. These calculations are somewhat analogous to their single Gaussian equivalents.

Keep doing these two steps one after the other. Eventually, the values of the mean and standard deviation will stabilize - that will be the learned model.

### The Expectation Step

The expectation step is for calculating responsibilities.

beginalignricfracpicNximucsigmacsumjin0kpijNximujsigmajendalign

Let's look at the numerator and denominator of this equation separately. The *responsibility* is calculated for each data point and for every cluster. So if you have 100 datapoints and have a mixture of five Gaussians, you would have to calculate 500 numbers. In practice, it is useful to think of this as a 100times5 array or matrix.

So ric is the responsibility of datapoint i with respect to the Gaussian curve c (out of k possible curves). This is represented by the numerator - which is just the probability of xi under the c Gaussian curve. We multiply it by the weight as well - since this is a linear combination and we need to account for the constant factor.

Now let's look at the denominator. It is simply the sum of probabilties of the data point xi under all the Gaussian curves. This means, it acts as a normalizing factor - ensuring that the value of ric is between zero and one for all curves. This ensures that the value we get is a valid probability.

And that's it - the E step just involves computing this quantity called responsibility!

### The Maximization Step

Now that we have the responsibilities for every datapoint with respect to each Gaussian curve, we can use this to improve our guess of each curve's mean, standard deviation and the weighting factor pic.

To calculate the new mean for every curve c, we have:

beginalignmucnewfrac1Ncsumiricxiendalign

This is very similar to learning the mean of a single Gaussian curve. The idea is, you sum up all the points and divide them by the number of points. The only difference here is that we're using the "responsibility" to account for membership in a Gaussian curve.

In a single Gaussian curve, every datapoint is a member of the curve - thus the responsibility is 1. However for multiple Gaussians, a single datapoint might contribute more to curve A rather than curve B. Ths responsibility accounts for this.

The denominator Nc takes into account this as well.

beginalignNcsumiricendalign

For a single Gaussian curve, this would simply be n - the number of points in the dataset. However, for multiple curves, each datapoint has a different responsibility - and thus adds only a fraction to the total point count for curve c.

We have a very similar equation for calculating the standard deviation as well:

beginalignsigmacnewfrac1Ncsumiricximucnew2endalign

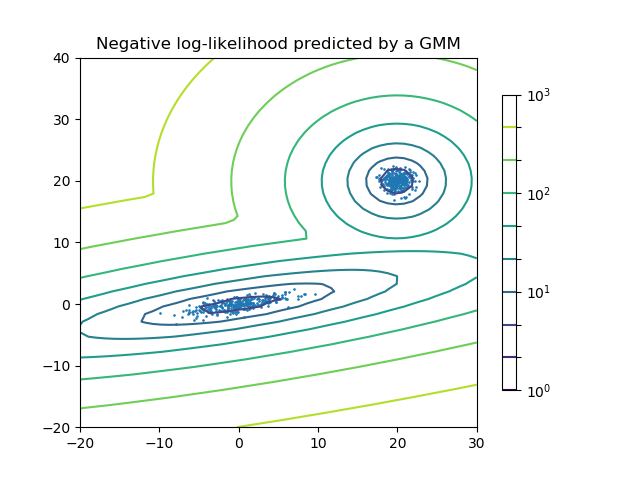
Again, the idea is that each point only partially contributes to each curve. Similarly, we can calculate the new values of pic as well:

beginalignpicfracNcnendalign

Here, n is the total number of datapoints in the dataset (irrespective of which curve they belong to).

**IMPLEMENTATION**

sklearn.mixture is a package which enables one to learn Gaussian Mixture Models (diagonal, spherical, tied and full covariance matrices supported), sample them, and estimate them from data. Facilities to help determine the appropriate number of components are also provided.



**Two-component Gaussian mixture model:** *data points, and equi-probability surfaces of the model.*

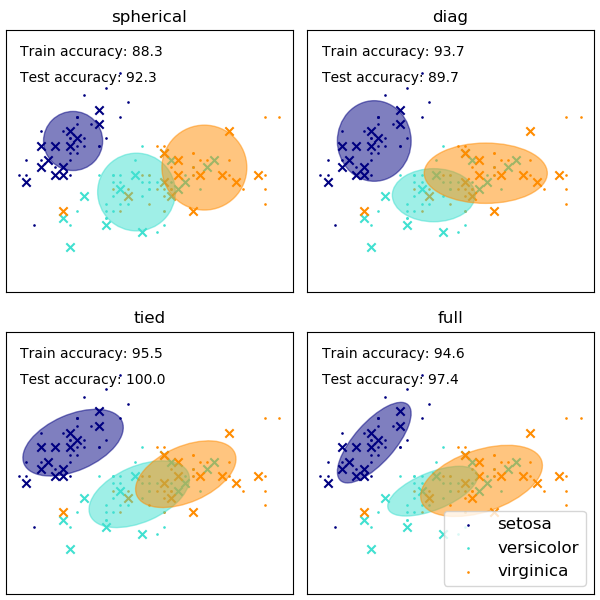
A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. One can think of mixture models as generalizing k-means clustering to incorporate information about the covariance structure of the data as well as the centers of the latent Gaussians.

Scikit-learn implements different classes to estimate Gaussian mixture models, that correspond to different estimation strategies, detailed below.

GAUSSIAN MIXTURE -

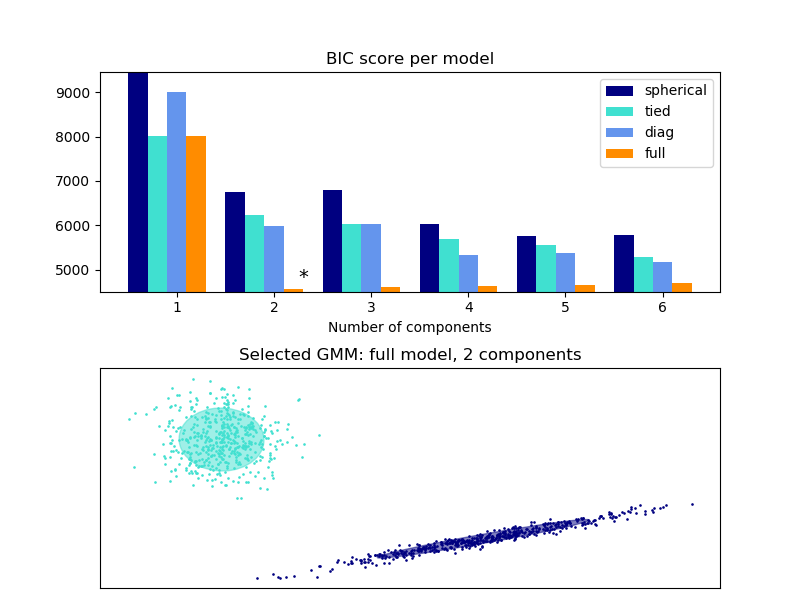
The **GaussianMixture** object implements the expectation-maximization (EM) algorithm for fitting mixture-of-Gaussian models. It can also draw confidence ellipsoids for multivariate models, and compute the Bayesian Information Criterion to assess the number of clusters in the data. A **GaussianMixture.fit** method is provided that learns a Gaussian Mixture Model from train data. Given test data, it can assign to each sample the Gaussian it mostly probably belong to using the **GaussianMixture.predict** method.

The **GaussianMixture** comes with different options to constrain the covariance of the difference classes estimated: spherical, diagonal, tied or full covariance.



### Selecting the number of components in a classical Gaussian Mixture Model

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 Variational Bayesian Gaussian mixture avoids the specification of the number of components for a Gaussian mixture model.



### Estimation algorithm Expectation-maximization

The main difficulty in learning Gaussian mixture models from unlabeled data is that it is one usually doesn’t know which points came from which latent component (if one has access to this information it gets very easy to fit a separate Gaussian distribution to each set of points). Expectation-maximization is a well-founded statistical algorithm to get around this problem by an iterative process. First one assumes random components (randomly centered on data points, learned from k-means, or even just normally distributed around the origin) and computes for each point a probability of being generated by each component of the model. Then, one tweaks the parameters to maximize the likelihood of the data given those assignments. Repeating this process is guaranteed to always converge to a local optimum.

## Variational Bayesian Gaussian Mixture

The **BayesianGaussianMixture** object implements a variant of the Gaussian mixture model with variational inference algorithms. The API is similar as the one defined by **GaussianMixture**.

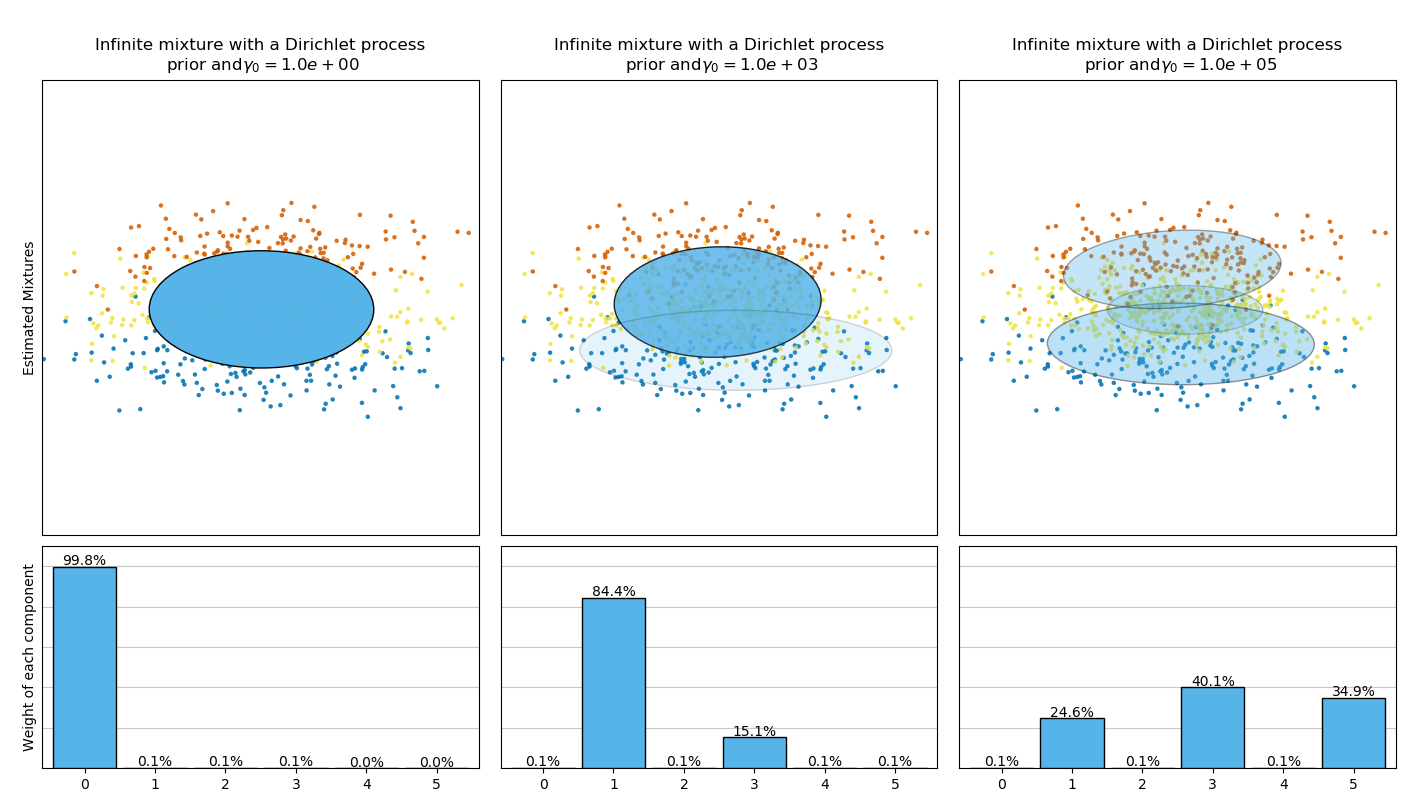
### Estimation algorithm: variational inference

Variational inference is an extension of expectation-maximization that maximizes a lower bound on model evidence (including priors) instead of data likelihood. The principle behind variational methods is the same as expectation-maximization (that is both are iterative algorithms that alternate between finding the probabilities for each point to be generated by each mixture and fitting the mixture to these assigned points), but variational methods add regularization by integrating information from prior distributions. This avoids the singularities often found in expectation-maximization solutions but introduces some subtle biases to the model. Inference is often notably slower, but not usually as much so as to render usage unpractical.

Due to its Bayesian nature, the variational algorithm needs more hyper- parameters than expectation-maximization, the most important of these being the concentration parameter weight\_concentration\_prior. Specifying a low value for the concentration prior will make the model put most of the weight on few components set the remaining components weights very close to zero. High values of the concentration prior will allow a larger number of components to be active in the mixture.

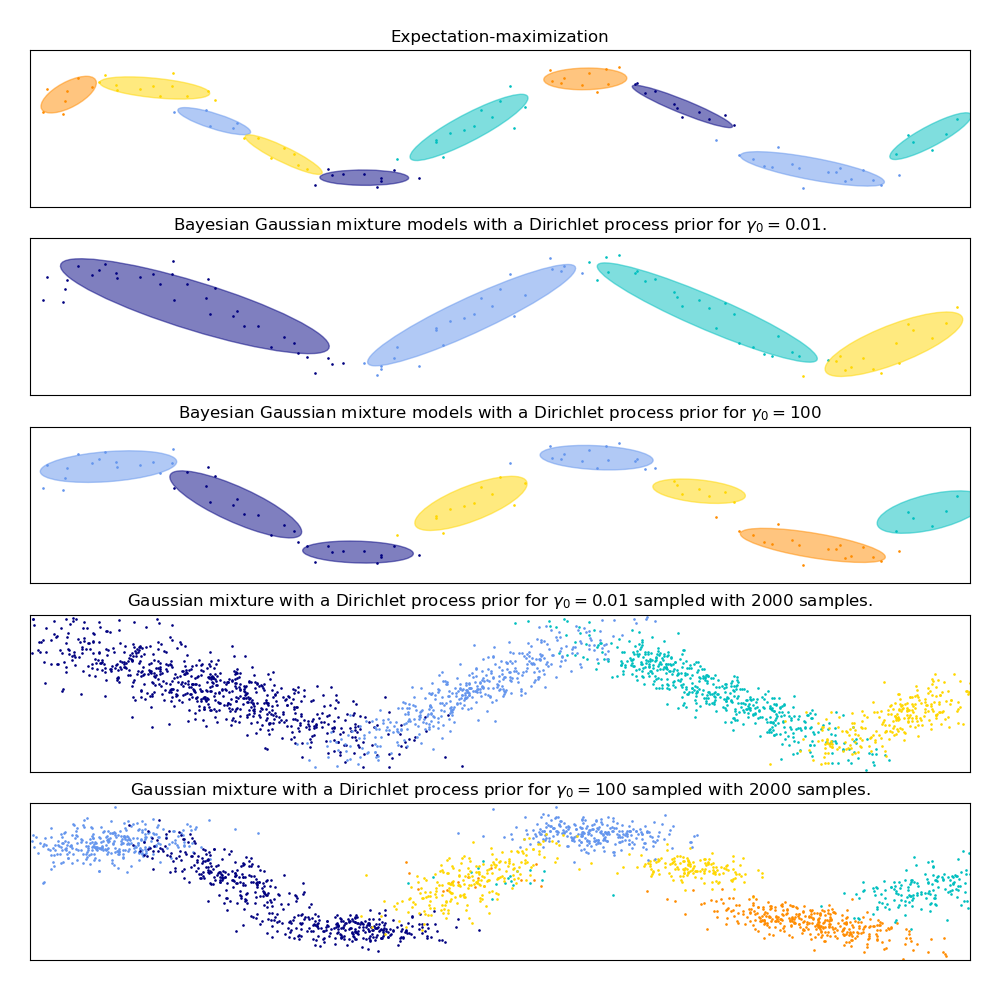
The parameters implementation of the **BayesianGaussianMixture** class proposes two types of prior for the weights distribution: a finite mixture model with Dirichlet distribution and an infinite mixture model with the Dirichlet Process. In practice Dirichlet Process inference algorithm is approximated and uses a truncated distribution with a fixed maximum number of components (called the Stick-breaking representation). The number of components actually used almost always depends on the data.

The next figure compares the results obtained for the different type of the weight concentration prior (parameter weight\_concentration\_prior\_type) for different values of weight\_concentration\_prior. Here, we can see the value of the weight\_concentration\_prior parameter has a strong impact on the effective number of active components obtained. We can also notice that large values for the concentration weight prior lead to more uniform weights when the type of prior is ‘dirichlet\_distribution’ while this is not necessarily the case for the ‘dirichlet\_process’ type (used by default).

****

The examples below compare Gaussian mixture models with a fixed number of components, to the variational Gaussian mixture models with a Dirichlet process prior. Here, a classical Gaussian mixture is fitted with 5 components on a dataset composed of 2 clusters. We can see that the variational Gaussian mixture with a Dirichlet process prior is able to limit itself to only 2 components whereas the Gaussian mixture fits the data with a fixed number of components that has to be set a priori by the user. In this case the user has selected n\_components=5 which does not match the true generative distribution of this toy dataset. Note that with very little observations, the variational Gaussian mixture models with a Dirichlet process prior can take a conservative stand, and fit only one component.

On the following figure we are fitting a dataset not well-depicted by a Gaussian mixture. Adjusting the weight\_concentration\_prior, parameter of the **BayesianGaussianMixture** controls the number of components used to fit this data. We also present on the last two plots a random sampling generated from the two resulting mixtures.



### Pros and cons of variational inference with BayesianGaussianMixture

#### Pros

|  |  |
| --- | --- |
| **Automatic selection:** | |
|  | when weight\_concentration\_prior is small enough and n\_components is larger than what is found necessary by the model, the Variational Bayesian mixture model has a natural tendency to set some mixture weights values close to zero. This makes it possible to let the model choose a suitable number of effective components automatically. Only an upper bound of this number needs to be provided. Note however that the “ideal” number of active components is very application specific and is typically ill-defined in a data exploration setting. |
| **Less sensitivity to the number of parameters:** | |
|  | unlike finite models, which will almost always use all components as much as they can, and hence will produce wildly different solutions for different numbers of components, the variational inference with a Dirichlet process prior (weight\_concentration\_prior\_type='dirichlet\_process') won’t change much with changes to the parameters, leading to more stability and less tuning. |
| **Regularization:** | due to the incorporation of prior information, variational solutions have less pathological special cases than expectation-maximization solutions. |

#### Cons

|  |  |
| --- | --- |
| **Speed:** | the extra parametrization necessary for variational inference make inference slower, although not by much. |
| **Hyperparameters:** | |
|  | this algorithm needs an extra hyperparameter that might need experimental tuning via cross-validation. |
| **Bias:** | there are many implicit biases in the inference algorithms (and also in the Dirichlet process if used), and whenever there is a mismatch between these biases and the data it might be possible to fit better models using a finite mixture. |

### The Dirichlet Process

Here we describe variational inference algorithms on Dirichlet process mixture. The Dirichlet process is a prior probability distribution on *clusterings with an infinite, unbounded, number of partitions*. Variational techniques let us incorporate this prior structure on Gaussian mixture models at almost no penalty in inference time, comparing with a finite Gaussian mixture model.

An important question is how can the Dirichlet process use an infinite, unbounded number of clusters and still be consistent. While a full explanation doesn’t fit this manual, one can think of its stick breaking process analogy to help understanding it. The stick breaking process is a generative story for the Dirichlet process. We start with a unit-length stick and in each step we break off a portion of the remaining stick. Each time, we associate the length of the piece of the stick to the proportion of points that falls into a group of the mixture. At the end, to represent the infinite mixture, we associate the last remaining piece of the stick to the proportion of points that don’t fall into all the other groups. The length of each piece is a random variable with probability proportional to the concentration parameter. Smaller value of the concentration will divide the unit-length into larger pieces of the stick (defining more concentrated distribution). Larger concentration values will create smaller pieces of the stick (increasing the number of components with non zero weights).

Variational inference techniques for the Dirichlet process still work with a finite approximation to this infinite mixture model, but instead of having to specify a priori how many components one wants to use, one just specifies the concentration parameter and an upper bound on the number of mixture components (this upper bound, assuming it is higher than the “true” number of components, affects only algorithmic complexity, not the actual number of components used).

**RESULTS**

Successfully implemented student model using gmm clustering for grouping various students in clusters based upon their performance on various parameters.

**REFERENCES**

* MEDIUM POST - <https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68>
* KAGGLE DATASET : <https://www.kaggle.com/c/student-academic-performance>
* A Reference Model for Learning Analytics , M.A. Chatti, A.L. Dyckhoff, U. Schroeder, and H. Thüs - <https://www.thues.com/upload/pdf/2012/CDST12_IJTEL.pdf>
* Learning Analytics in Higher Education-A review of UK and international practice-Full report-April 2016-Authors-Niall Sclater, Alice Peasgood, Joel Mullan - <https://www.jisc.ac.uk/sites/default/files/learning-analytics-in-he-v2_0.pdf>
* Student Modeling and Machine Learning-Raymund Sison, Masamichi Shimura - <http://www.info2.uqam.ca/~nkambou_r/DIC9340/seances/seance2-3-4-6/Student-Modelling/paper.pdf>