Gaussian Mixture Models

- **A**mel **A**bdelraheem
- **E**lysee **G**asana
- **I**feoma **V**eronica **N**wabufo
- **J**oseph **N**wanna

22 April 2022

Project 6-Group 8

Table of contents

Task Formulation & Setup

- **Sentiment Analysis:** Detects the underlying sentiment in a piece of text (*Positive, Negative*).
- This can be useful when collecting reviews of a specific product or service.
- In this project we looked at movie reviews collected from the **IMBD** website.

Clean Text

- We begin by removing punctuations from our text.
- We also change all the letters to lowercase, otherwise later on pairs like ‘Clean’ & ‘clean’ will be given separate representations
- Lastly we split the text into an array of words for further processing.

Stop Words

- Stop words are Common words that appear frequently in text.
- removing these will increase computation and space efficiency.
- These words may not be indicative of the contextual content behind the text
  e.g.
  ‘this’, ’you’, ‘are’, ‘have’,‘has’, ‘had’...etc.

- We define a list containing all the stop words and then search and remove them from our text.

Stemming/ Normalizing Text

Porter Stemmer:

- Any word can be broken down to units of consonant & vowel letters [C]VC…[V]. (the brackets denote that the first & last units are optional).
- Porter then defines a number m such that [C]VC{m}[V], this will be a guide to how and when we should stop trimming the suffixes.
- This algorithm is not concerned with the readability of the resulting base.
- The rulings used only work on the English Language as it’s derived from common patterns.

Porter Stemmer Algorithm:

1. Check if the word is in plurals or past participles format:
   i.e :
   SSES -> SS
   IES -> I
   ING -> None

2. Check if m>0 and replace some common terminations:
   i.e:
   ATIONAL -> ATE
   TIONAL -> TION

3. Check if m>0 and replace these terminations:
   i.e:
   ATIVE -> None

4. Check if m > 1 and remove suffixes:
   i.e :
   AL -> None
   ANCE -> None

5. Final check if m>1 and if the last letter is E*:
   i.e:
   E -> None

Text Embeddings

Frequency count, TF-IDF

Frequency Count

- Simply count the occurrences of a word within all the documents and use that as your representation

	Word_1	Word_2	Word_3
Doc 1	1	0	2
Doc2	1	4	0
Embed.	2	4	2

Term Frequency - Inverse Document Frequency

- It is a statistical measure that evaluates how relevant a word is to a document in a collection of documents ⇒ less common words have higher weights suggesting that they are more indicative of the context
- **Term Frequency**: how many times a word appears in a document divided by the total number of words in the document
- **Inverse Document Frequency**: the log of dividing the total number of documents by the number of documents in which the word appears in

Tf = word_freq /tot_words_doc

Tdf = log [tot_docs/docs_with_word]

Tf-Idf = Tf * Idf

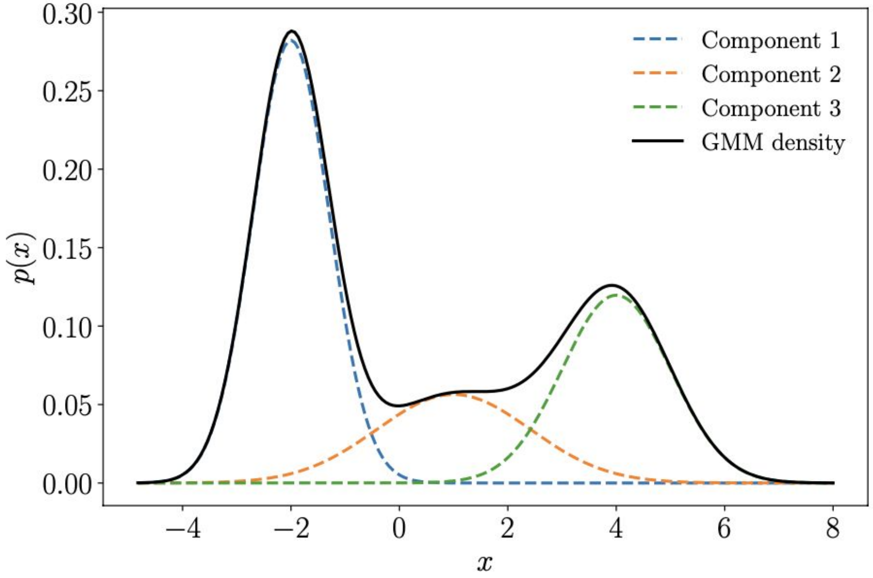
Gaussian Mixture Model

- Popular alternative to K-Means clustering .
- A probabilistic model for clustered data with real-valued components.
- Model the data as a weighted sum of gaussian distributions.
- It is a soft clustering algorithm; meaning all the points have probabilities describing their belonging to the data.

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

$$0 \leqslant \pi_k \leqslant 1 \,, \quad \sum_{k=1}^{K} \pi_k = 1 \,,$$

$$\boldsymbol{\theta} := \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k : \; k = 1, \ldots, K\}$$



* Deisenroth, M.P., Faisal, A.A. and Ong, C.S., 2020. *Mathematics for machine learning*. Cambridge University Press.

Good old Maximum Likelihood Estimation (MLE)?

$$p(\mathcal{X} \mid \boldsymbol{\theta}) = \prod_{n=1}^{N} p(\boldsymbol{x}_n \mid \boldsymbol{\theta}), \quad p(\boldsymbol{x}_n \mid \boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \mathcal{N}\big(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\big)$$

$$\log p(\mathcal{X} \mid \boldsymbol{\theta}) = \sum_{n=1}^{N} \log p(\boldsymbol{x}_n \mid \boldsymbol{\theta}) = \underbrace{\sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}\big(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\big)}_{=:\mathcal{L}}$$

$$p(\mathcal{X}\,|\,\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\boldsymbol{x}_n\,|\,\boldsymbol{\theta})\,, \quad \ldots \boldsymbol{\mu}_k,\, \boldsymbol{\Sigma}_k)$$

$$\log p(\mathcal{X}\,| \ldots \underbrace{\sum_{n=1} \log \sum_{k=1}^{K} \pi_k \mathcal{N}\big(\boldsymbol{x}_n\,|\,\boldsymbol{\mu}_k,\, \boldsymbol{\Sigma}_k\big)}_{=:\mathcal{L}}$$

No Closed Form Solution!

Latent Variable Perspective

- A GMM has an equivalent representation as a generative model for our data:

$$
\begin{aligned}
z_i &\overset{\text{iid}}{\sim} \text{Mult}(\pi, 1) \\
x_i | z_i &\overset{\text{ind}}{\sim} \mathcal{N}(\mu_{z_i}, \Sigma_{z_i}).
\end{aligned}
$$

  where z_i represents the latent component indicator or latent class / cluster for datapoint x_i
- If we assume we have an initial GMM, our clustering task amounts to inferring the latent component z_i responsible for each x_i

$$
\begin{aligned}
p(z_i = j | x_i) &= \frac{p(z_i = j) p(x_i | z_i = j)}{p(x_i)} \\
&= \frac{\pi_j \mathcal{N}(x_i; \mu_j, \Sigma_j)}{\sum_{l=1}^{k} \pi_l \mathcal{N}(x_i; \mu_l, \Sigma_l)}.
\end{aligned}
$$

*STATS 306B: Unsupervised Learning, Lester Mackey

Latent Variable Perspective

- Before we observe x_i , we have the prior belief that it belongs to cluster j with probability πj ; after observing x_i , we can update this belief in accordance with the likelihood of xi under each Gaussian component
- If we knew the zi 's, then the problem of the sum inside of each log (representing an expectation over an unknown cluster assignment zi) would be solved, since we could instead maximize the complete log likelihood

$$
\begin{aligned}
\sum_{i=1}^{n} \log(p(x_i, z_i)) &= \sum_{i=1}^{n} (\log(p(x_i|z_i)) + \log(p(z_i))) \\
&= \sum_{i=1}^{n} (\log(\mathcal{N}(x_i; \mu_{z_i}, \Sigma_{z_i})) + \log(\pi_{z_i})) \\
&= \sum_{i=1}^{n} \sum_{j=1}^{k} (\mathbb{I}[z_i = j] \log \mathcal{N}(x_i; \mu_j, \Sigma_j) + \mathbb{I}[z_i = j] \log \pi_j).
\end{aligned}
$$

*STATS 306B: Unsupervised Learning, Lester Mackey

Latent Variable Perspective

- The complete log likelihood, viewed as a function of the parameters of the GMM(π, μ, Σ), has closed form maxima:

- If we knew the zi ’s, then the problem of the sum inside of each log (representing an expectation over an unknown cluster assignment zi) would be solved, since we could instead maximize the complete log likelihood

$$\pi_j^* = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}[z_i = j],$$

$$\mu_j^* = \frac{\sum_{i=1}^{n} \mathbb{I}[z_i = j] x_i}{\sum_{i=1}^{n} \mathbb{I}[z_i = j]},$$

$$\Sigma_j^* = \frac{\sum_{i=1}^{n} \mathbb{I}[z_i = j](x_i - \mu_j^*)(x_i - \mu_j^*)^T}{\sum_{i=1}^{n} \mathbb{I}[z_i = j]}.$$

*STATS 306B: Unsupervised Learning, Lester Mackey

Expectation Maximization

1. The Expectation-Maximization (EM) algorithm, leverages the latent-variable problem structure to form parameter estimates.

$$\log p(\mathcal{X} \mid \boldsymbol{\theta}) = \sum_{n=1}^{N} \log p(\boldsymbol{x}_n \mid \boldsymbol{\theta}) = \underbrace{\sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}\big(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\big)}_{=:\mathcal{L}}$$

2. If we could change the summation term inside the log, we could easily carry out the MLE $\Rightarrow$ **Initialize values for the parameters**
3. We will then estimate the distribution given the data and the current value of the parameters
4. Next, we can estimate cluster assignments using probabilistic inference

*STATS 306B: Unsupervised Learning, Lester Mackey

1. Expectation-Maximization for GMMs:
    1.1. Initialize cluster weight(π), and for each gaussian assign mean and covariance values(μ, Σ)
    1.2. Alternate until convergence:
        1.2.1. (**E-step**) [Expectation step]: Compute soft class memberships (probabilities), given the current parameters:

$$\tau_{ij} = P(z_i = j | x_{ij}, \pi, (\mu_\ell, \Sigma_\ell)).$$

        1.2.2. (**M-step**) [Maximization step]: Update parameters by plugging in τij (our guess) for the unknown I[zi = j], which gives us:

$$\pi_j = \frac{1}{n} \sum_{i=1}^{n} \tau_{ij}, \quad \mu_j = \frac{\sum_{i=1}^{n} \tau_{ij} x_i}{\sum_{i=1}^{n} \tau_{ij}},$$

$$\Sigma_j = \frac{\sum_{i=1}^{n} \tau_{ij}(x_i - \mu_j)(x_i - \mu_j)^T}{\sum_{i=1}^{n} \tau_{ij}}.$$

This is similar to the case in which all zi 's are known, but now each xi is partially assigned to each cluster j through the conditional probability that zi = j.

*STATS 306B: Unsupervised Learning, Lester Mackey

Thanks for Listening!